aws

Developer Guide

Amazon Application Recovery Controller (ARC)

Amazon Application Recovery Controller (ARC): Developer Guide

Table of Contents

What is Amazon Application Recovery Controller (ARC)?

Amazon Application Recovery Controller (ARC) (ARC) helps you prepare for and complete faster recovery for applications running on AWS. ARC provides two sets of capabilities: *Multi-Availability Zone (AZ) recovery*, which includes zonal shift and zonal autoshift, and *multi-Region recovery*, which includes routing control and readiness check. With ARC, you can leverage highly-available recovery tools to quickly mitigate impairments that are impacting your multi-Region or multi-AZ applications. You can also use readiness check to gain insights into whether your applications and resources are prepared for recovery.

The AWS Global Cloud Infrastructure provides fault tolerance and resilience, with each AWS Region made up of multiple, fully-isolated Availability Zones. ARC works within this AWS structure to help your applications be resilient.

Multi-AZ recovery

If you have applications that are built to take advantage of Availability Zones in AWS, you can quickly isolate and recover from AZ impairments using zonal shift. *Zonal shift* enables you to recover from Availability Zone (AZ) impairments, by temporarily moving traffic for a supported resource away from an AZ, to healthy AZs in the AWS Region. Starting a zonal shift helps your application recover quickly, for example, from a developer's bad code deployment or from an AWS impairment in a single Availability Zone. By moving traffic away, you reduce the impact for clients who are using your application when there's an issue in one AZ.

You can start a zonal shift for any supported resource in your account in a Region. AWS services automatically register supported AWS resources with zonal shift in ARC, so that you can start a zonal shift at any time.

Zonal autoshift is a capability in ARC that you can enable to authorize AWS to shift traffic away from an AZ for supported resources, on your behalf, to healthy AZs in the AWS Region. AWS starts an autoshift when internal telemetry indicates that there is an impairment in one AZ in a Region that could potentially impact customers. The internal telemetry incorporates metrics from multiple sources, including the AWS network, and the Amazon EC2 and Elastic Load Balancing services.

Zonal shifts and autoshifts are temporary. When you start a manual zonal shift, you must specify an (extendable) expiration, of up to three days initially. If you want to continue to keep traffic away from an AZ, you can update the zonal shift and set a new expiration. With zonal autoshift, AWS ends an autoshift when indicators show that there is no longer an issue or potential issue.

To learn more about these capabilities, see the following chapters:

- Zonal shift in Amazon Application Recovery Controller (ARC)
- Zonal autoshift in Amazon Application Recovery Controller (ARC)

Multi-Region recovery

If you have an application that you've designed to operate out of another AWS Region to continue operations you can use routing control for failover. *Routing control* enables you to fail over traffic from one AWS Region to another when there's an issue, so that you can ensure that your application stays available. Routing control includes safety rules, which help protect you from unintended outcomes, by imposing guardrails that you define. Using these rules, you can make sure, for example that only one of your application replicas, active or standby, is enabled and in use at a time.

For multi-Region recovery, ARC can help you fail over DNS traffic across AWS Regions. The extremely reliable routing controls in ARC enable you to recover your application by rerouting traffic away from a Region with an impairment to a healthy Region.

With *readiness check*, ARC continually monitors AWS resource quotas, capacity, and network routing policies, and can notify you about changes that would affect your ability to fail over to a replica and recover. Continual readiness checks help make sure, on an ongoing basis, that you can maintain your multi-Region applications in a state that is scaled and configured to handle failover traffic. Readiness check is useful when you first configure ARC, and during normal application operation. Readiness check is not intended to be used in the critical path for failover during an event.

To learn more about these capabilities, see the following chapters:

- Routing control in Amazon Application Recovery Controller (ARC)
- Readiness check in Amazon Application Recovery Controller (ARC)

Compare multi-AZ and multi-Region recovery capabilities in Amazon Application Recovery Controller (ARC)

Zonal shift, zonal autoshift, and routing control in Amazon Application Recovery Controller (ARC) can all achieve rapid recovery and help you to ensure resilience for your AWS applications. These options are highly available and help support recovery in scenarios when your application is experiencing increased latency or reduced availability. These options help recover applications quickly by shifting traffic away from isolated impairments, which limits the impact and time lost from impairments.

Routing control is primarily focused on AWS applications that are in multiple AWS Regions (multi-Region), while zonal shift and zonal autoshift only support shifting traffic for load balancers with multi-AZ applications. There are other differences as well, as described in this section.

The information in the following table includes some of the key features of zonal shift, zonal autoshift, and routing control, and how the options compare to each other. These descriptions can help you better understand how a specific option might be the best choice for your organization's disaster recovery needs.

Routing control	Zonal shift	Zonal autoshift
Regional	**Zonal**	**Zonal**
Reroutes traffic from one AWS Region to another (primarily) Can also be used to reroute across Availability Zones	Moves traffic away from an Availability Zone Traffic goes to other Availability Zones in the Region, not to a specific target	Moves traffic away from an Availability Zone Traffic goes to other Availability Zones in the Region, not to a specific target
Requires setup	**Available without setup**	**Requires practice run setup**
Requires configuration and setup	Enabled automatically by supported services	Available for supported services

Routing control	Zonal shift	Zonal autoshift
	(currently Network Load Balancer and Application Load Balancer)	(currently Network Load Balancer and Application Load Balancer)
Customer-initiated Customer determines when to re-route traffic	**Customer-initiated** Customer determines when to start a zonal shift	**AWS-initiated** AWS shifts application traffic away from an AZ on your behalf
Fee-based Requires separate charges for routing control	**Included with services** Creating zonal shifts to move traffic away from AZs is included for supported load balancers	**Included with services** Starting autoshifts to move traffic away from AZs on your behalf is included for supported load balancers
Does not expire Traffic can be rerouted to a replica indefinitely	**Temporary** All zonal shifts must be set to expire	**Temporary** AWS starts and ends autoshift s

To learn more about each of these features, see the following chapters:

- Zonal shift in Amazon Application Recovery Controller (ARC)
- Zonal autoshift in Amazon Application Recovery Controller (ARC)
- Routing control in Amazon Application Recovery Controller (ARC)

Use zonal shift and zonal autoshift to recover applications in Amazon Application Recovery Controller (ARC)

This section explains how to use capabilities in Amazon Application Recovery Controller (ARC) to reliably recover your AWS application from an issue in an Availability Zone (AZ). These capabilities, zonal shift and zonal autoshift, temporarily move traffic away from an AZ for an Elastic Load Balancing resource, to reduce time to recovery for your applications.

The primary difference between zonal shift and zonal autoshift is that one is a manual traffic shift that you control, and the other shifts traffic away from an impairment automatically on your behalf.

- With zonal shift, you manually move traffic for a managed Elastic Load Balancing resource in an AWS Region away from an Availability Zone.
- With zonal autoshift, Elastic Load Balancing traffic is automatically shifted away from an impaired AZ to healthy AZs in a Region during events, on your behalf.

The following topics describe the zonal shift and zonal autoshift capabilities, and how to use them.

Topics

Zonal shift in Amazon Application Recovery Controller (ARC)

With zonal shift in Amazon Application Recovery Controller (ARC), you can move traffic for an Elastic Load Balancing resource away from an Availability Zone in an AWS Region, to quickly mitigate an issue and quickly recover your application. Note that the Elastic Load Balancing resources must have cross-zone load balancing turned off to use this capability.

When you deploy and run AWS applications on load balancers in multiple (typically three) AZs in a Region, you can quickly recover an application in an impaired AZ by starting a zonal shift. Shifting your application traffic to healthy AZs reduces the duration and severity of impact caused by power outages, or hardware or software issues in an AZ.

You might choose to shift traffic, for example, because a bad deployment is causing latency issues, or because the Availability Zone is impaired. A zonal shift requires no advance configuration steps, but your AWS configuration must support handling your client load without the Availability Zone that you shift away from. Supported load balancer resources are automatically registered with Amazon Application Recovery Controller (ARC) for you, so that you can simply start a zonal shift for the load balancer when needed.

Starting a zonal shift requires no setup or configuration. After you ensure that you have sufficient capacity to shift traffic away from an Availability Zone, choose the Availability Zone to shift away from and the resource to shift traffic away for, and then start the zonal shift. You can cancel the shift at any time, to have traffic begin returning to the Availability Zone.

All zonal shifts are temporary mitigations. You set an initial expiration when you start a zonal shift, from one hour up to three days (72 hours), which you can extend, if you need to continue the traffic shift.

Be aware that, in specific scenarios, zonal shift does not shift traffic away from the AZ. For more information about zonal shift support, see Resources and scenarios supported for zonal shift and zonal autoshift.

How a zonal shift works

When you start a zonal shift for a load balancer resource, traffic for the resource is moved away from the Availability Zone that you've specified. To start the shift, Amazon Application Recovery Controller (ARC) requests the load balancer health check for the Availability Zone to be set to unhealthy, so that it fails its health check. An unhealthy health check, in turn, results in Amazon Route 53 automatically withdrawing the corresponding IP addresses for the resource from DNS, so that traffic is redirected from the Availability Zone. New connections are now routed to other Availability Zones in the AWS Region instead.

It's important to note that zonal shift does not use health checks in the typical way, where a health check monitors the underlying health of load balancers or applications. Instead, ARC uses health checks as a mechanism to move traffic away from an Availability Zone. The mechanism requests a health check to be explicitly set to unhealthy, and then to healthy again, to change how traffic flows.

Traffic begins to shift - When you start a zonal shift in ARC, because of the steps involved with traffic flow, you might not see traffic move out of the Availability Zone immediately. It also can take a short time for existing, in-progress connections in the Availability Zone to complete,

depending on client behavior and connection reuse. Depending on your DNS settings and other factors, existing connections can complete in just a few minutes, or might take longer. For more information, see Ensuring that traffic shifts finish quickly.

Traffic shift ends - When a zonal shift expires or you cancel it, ARC takes steps to stop shifting traffic. It reverses the process for starting a traffic shift, and requests the Route 53 health checks to be set to healthy again. Healthy health checks result in the original zonal IP addresses being restored. Now, the recovered Availability Zone is included in the load balancer's routing again and traffic begins to resume flowing to the AZ.

You must set all zonal shifts to expire when you start the shifts. You can initially set a zonal shift to expire in a maximum of three days (72 hours). However, you can update a zonal shift to set a new expiration at any time. You can also cancel a zonal shift before it expires, if you're ready to restore traffic to the Availability Zone.

When traffic does not shift away

In specific scenarios, a zonal shift does not shift traffic from the Availability Zone. For example, say you start a zonal shift for a load balancer when the load balancer target groups in the AZs don't have any instances, or if all of the instances are unhealthy. In this scenario, the load balancer is in a fail open state and starting a zonal shift does not shift away traffic.

Before you start a zonal shift for a resource, make sure that all the conditions for a successful zonal shift are met. For more information about zonal shift support, see Resources and scenarios supported for zonal shift and zonal autoshift.

AWS Region availability for zonal shift

For detailed information about Regional support and service endpoints for Amazon Application Recovery Controller (ARC), see Amazon Application Recovery Controller (ARC) endpoints and quotas in the *Amazon Web Services General Reference*.

Zonal shift is currently available in the AWS Regions listed here. Zonal shift is also available in the China Regions, that is, China (Beijing) Region and China (Ningxia) Region.

Region Name	Region	Endpoint	Protocol	
US East (Ohio)	us-east-2	arc-zonal-shift.us-east-2.amazonaws.com	HTTPS	

Region Name	Region	Endpoint	Protocol	
US East (N. Virginia)	us-east-1	arc-zonal-shift.us-east-1.amazonaws.com	HTTPS	
US West (N. California)	us-west-1	arc-zonal-shift.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	arc-zonal-shift.us-west-2.amazonaws.com	HTTPS	
Africa (Cape Town)	af-south-1	arc-zonal-shift.af-south-1.amazonaws.com	HTTPS	
Asia Pacific (Hong Kong)	ap-east-1	arc-zonal-shift.ap-east-1.amazonaws.com	HTTPS	
Asia Pacific (Hyderabad)	ap-south-2	arc-zonal-shift.ap-south-2.amazonaws.com	HTTPS	
Asia Pacific (Jakarta)	ap-southeast-3	arc-zonal-shift.ap-southeast-3.amazonaws.com	HTTPS	
Asia Pacific (Malaysia)	ap-southeast-5	arc-zonal-shift.ap-southeast-5.amazonaws.com	HTTPS	

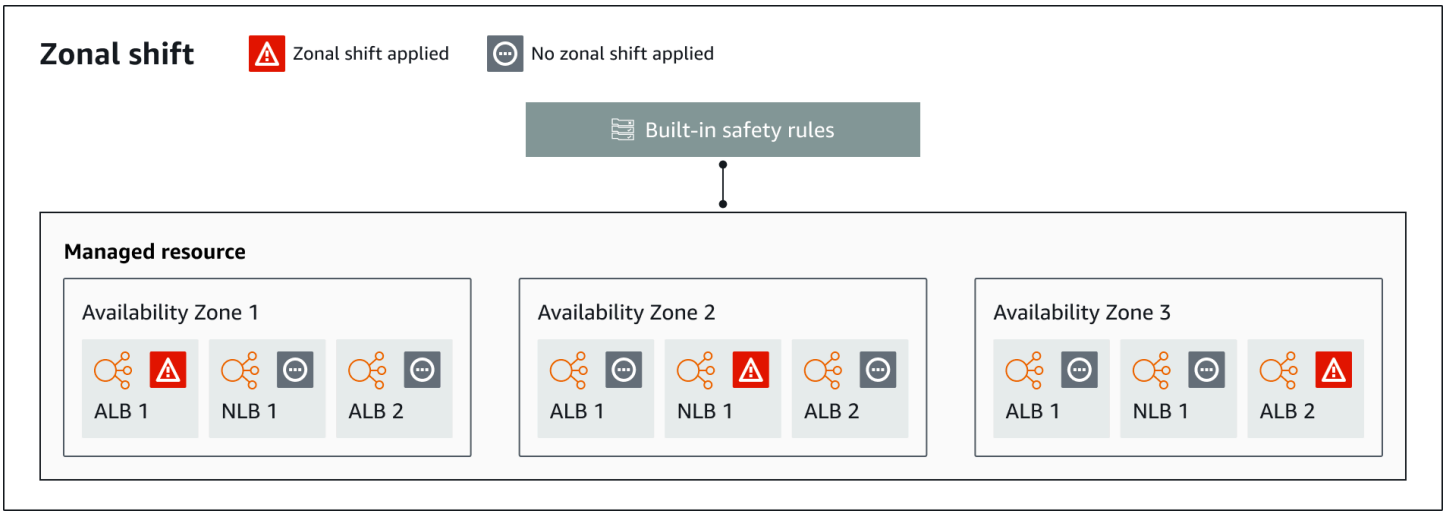
Region Name	Region	Endpoint	Protocol	
Asia Pacific (Melbourne)	ap-southeast-4	arc-zonal-shift.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacific (Mumbai)	ap-south-1	arc-zonal-shift.ap-south-1.amazonaws.com	HTTPS	
Asia Pacific (Osaka)	ap-northeast-3	arc-zonal-shift.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	arc-zonal-shift.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacific (Singapore)	ap-southeast-1	arc-zonal-shift.ap-southeast-1.amazonaws.com	HTTPS	
Asia Pacific (Sydney)	ap-southeast-2	arc-zonal-shift.ap-southeast-2.amazonaws.com	HTTPS	
Asia Pacific (Tokyo)	ap-northeast-1	arc-zonal-shift.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Central)	ca-central-1	arc-zonal-shift.ca-central-1.amazonaws.com	HTTPS	
Canada West (Calgary)	ca-west-1	arc-zonal-shift.ca-west-1.amazonaws.com	HTTPS	

Region Name	Region	Endpoint	Protocol	
Europe (Frankfurt)	eu-central-1	arc-zonal-shift.eu-central-1.amazonaws.com	HTTPS	
Europe (Ireland)	eu-west-1	arc-zonal-shift.eu-west-1.amazonaws.com	HTTPS	
Europe (London)	eu-west-2	arc-zonal-shift.eu-west-2.amazonaws.com	HTTPS	
Europe (Milan)	eu-south-1	arc-zonal-shift.eu-south-1.amazonaws.com	HTTPS	
Europe (Paris)	eu-west-3	arc-zonal-shift.eu-west-3.amazonaws.com	HTTPS	
Europe (Spain)	eu-south-2	arc-zonal-shift.eu-south-2.amazonaws.com	HTTPS	
Europe (Stockholm)	eu-north-1	arc-zonal-shift.eu-north-1.amazonaws.com	HTTPS	
Europe (Zurich)	eu-central-2	arc-zonal-shift.eu-central-2.amazonaws.com	HTTPS	
Israel (Tel Aviv)	il-central-1	arc-zonal-shift.il-central-1.amazonaws.com	HTTPS	
Middle East (Bahrain)	me-south-1	arc-zonal-shift.me-south-1.amazonaws.com	HTTPS	
Middle East (UAE)	me-central-1	arc-zonal-shift.me-central-1.amazonaws.com	HTTPS	

Region Name	Region	Endpoint	Protocol	
South America (São Paulo)	sa-east-1	arc-zonal-shift.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (US-East)	us-gov-east-1	arc-zonal-shift.us-gov-east-1.amazonaws.com	HTTPS	
AWS GovCloud (US-West)	us-gov-west-1	arc-zonal-shift.us-gov-west-1.amazonaws.com	HTTPS	

Zonal shift components

The following diagram illustrates an example of a zonal shift shifting traffic away from an Availability Zone in an AWS Region. Checks that are built into zonal shift prevent you from starting another zonal shift for a resource when it already has an active shift.



The following are components of the zonal shift capability in ARC.

Zonal shift

You start a zonal shift for a managed resource in your AWS account to temporarily move traffic away from an Availability Zone in an AWS Region, to healthy AZs in the Region, to quickly recover from an issue in one AZ. Currently you can start a zonal shift only for Network Load Balancers and Application Load Balancers that do not have cross-zone load balancing configured. Supported load balancers are automatically registered for you in ARC.

Built-in safety checks

Checks that are built into ARC prevent more than one traffic shift for a resource from being in effect at a time. That is, only one customer-initiated zonal shift, practice run zonal shift, or autoshift for the resource can be actively shifting traffic away from an Availability Zone. For example, if you start a zonal shift for a resource when it is currently shifted away with autoshift, your zonal shift takes precedence. For more information, see Zonal autoshift in Amazon Application Recovery Controller (ARC) and Outcomes for practice runs.

Resource identifier

The identifier for a resource to include in a zonal shift. The identifier is the Amazon Resource Name (ARN) for the resource.

For a zonal shift, you can only choose resources in your account for an AWS service that is supported by ARC. Supported resources in those AWS services are automatically registered with ARC by the AWS service.

> **Note**
>
> Currently, you can only start a zonal shift for Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

Managed resource

AWS services register resources automatically with ARC for zonal shift. A resource that has been registered is a managed resource in ARC.

Resource name

The name of a resource in ARC that you can specify for a zonal shift.

Status (zonal shift status)

A status for a zonal shift. The `Status` for a zonal shift can have one of the following values:

- **ACTIVE**: The zonal shift is started and active.
- **EXPIRED**: The zonal shift has expired (the expiry time was exceeded).
- **CANCELED**: The zonal shift was canceled.

Applied status

An applied status indicates whether a shift is in effect for a resource. The shift that has the status `APPLIED` determines the Availability Zone where application traffic has been shifted away for a resource, and when that shift ends.

Expiry time (expiration time)

The expiry time (expiration time) for a zonal shift. Zonal shifts are temporary. For a customer-initiated zonal shift, you can initially set a zonal shift to be active for up to three days (72 hours).

When you start a zonal shift, you specify how long you want it to be active, which ARC converts to an expiry time (expiration time). You can cancel a customer-initiated zonal shift, for example, if you're ready to restore traffic to the Availability Zone. Or you can extend a customer-initiated zonal shift by updating it to specify another length of time to expire in.

You can cancel both customer-initiated zonal shifts and zonal shifts that AWS starts for a practice run with zonal autoshift.

Data and control planes for zonal shift

As you plan for failover and disaster recovery, consider how resilient your failover mechanisms are. We recommend that you make sure that the mechanisms that you depend on during failover are highly available, so that you can use them when you need them in a disaster scenario. Typically, you should use data plane functions for your mechanisms whenever you can, for the greatest reliability and fault tolerance. With that in mind, it's important to understand how the functionality of a service is divided between control planes and data planes, and when you can rely on an expectation of extreme reliability with a service's data plane.

As with most AWS services, the functionality for the zonal shift capability is supported by control planes and data planes. While both of these are built to be reliable, a control plane is optimized for data consistency, while a data plane is optimized for availability. A data plane is designed for resilience so that it can maintain availability even during disruptive events, when a control plane might become unavailable.

In general, a *control plane* enables you to do basic management functions, such as create, update, and delete resources in the service. A *data plane* provides a service's core functionality.

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see the Static stability using Availability Zones paper in the Amazon Builders' Library.

Pricing for zonal shift in Amazon Application Recovery Controller (ARC)

For zonal shift, you can start a zonal shift for supported resources, to recover your application from an issue in an Availability Zone. There is no additional charge for using zonal shift.

You only pay for what you use in Amazon Application Recovery Controller (ARC). For detailed pricing information for ARC and pricing examples, see Amazon Route 53 Pricing and scroll down to Amazon Application Recovery Controller (ARC).

Best practices for zonal shifts in ARC

We recommend the following best practices for using zonal shifts for multi-AZ recovery in ARC. Zonal shifts typically remove capacity from a live application, so it's important to be careful when you use them in production.

Topics

- Capacity planning and pre-scaling
- Limit the time that clients stay connected to your endpoints
- Test starting zonal shifts, in advance
- Ensure that all Availability Zones are healthy and taking traffic
- Use data plane API operations for disaster recovery
- Move traffic with a zonal shift only temporarily

Capacity planning and pre-scaling

Ensure that you have planned for, and either pre-scaled or can auto-scale, sufficient capacity to accommodate the extra load imposed on Availability Zones when you start a zonal shift. With a recovery-oriented architecture, a typical recommendation is to pre-scale compute capacity to include enough headroom to serve your peak traffic when one of your (typically) three replicas is offline.

When you start a zonal shift for a single load balancer resource, for example, the capacity of one Availability Zone is temporarily removed from behind the load balancer. Depending on the zonal shifts that you start and how your load balancers are configured, you must make sure that you've carefully planned for managing the increased load on the remaining Availability Zones.

Limit the time that clients stay connected to your endpoints

When Amazon Application Recovery Controller (ARC) shifts traffic away from an impairment, for example, by using zonal shift or zonal autoshift, the mechanism that ARC uses to move your application traffic is a DNS update. A DNS update causes all new connections to be directed away from the impaired location.

However, clients with pre-existing open connections might continue to make requests against the impaired location until the clients reconnect. To ensure a quick recovery, we recommend that you limit the amount of time clients stay connected to your endpoints.

If you use an Application Load Balancer, you can use the `keepalive` option to configure how long connections continue. For more information, see HTTP client keepalive duration in the Application Load Balancer User Guide.

By default, Application Load Balancers set the HTTP client keepalive duration value to 3600 seconds, or 1 hour. We suggest that you lower the value to be inline with your recovery time goal for your application, for example, 300 seconds. When you choose an HTTP client keepalive duration time, consider that this value is a trade off between reconnecting more frequently in general, which can affect latency, and more quickly moving all clients away from an impaired AZ or Region.

Test starting zonal shifts, in advance

Regularly test moving traffic away from Availability Zones for your application by starting zonal shifts. Plan for and execute starting zonal shifts, preferably in both test and production environments, as part of regular failover testing for recovering your applications in the event of a disaster. Regular testing is a critical part of ensuring that you're ready for and have the confidence to mitigate issues when an operational event occurs.

Ensure that all Availability Zones are healthy and taking traffic

Zonal shifts work by marking a resource, that is, an application replica, as unhealthy in an Availability Zone. This means that it's critical to ensure that the targets in the load balancers for your applications are generally healthy and actively taking traffic in the Availability Zones in a Region. We recommend that you have dashboards to track this, including, for example, Elastic Load Balancing metrics for unhealthy targets and bytesProcessed per Availability Zone.

Consider monitoring the health of your resources from a second, adjacent Region. Advantages of this approach are that it can be more representative of your end users' experience, and it also reduces the risk of both your application and your monitoring being impacted by the same disaster at the same time ("shared fate").

Use data plane API operations for disaster recovery

For starting a zonal shift when you need to recover an application quickly, with few dependencies, we recommend using the AWS Command Line Interface or API with zonal shift actions, with pre-stored credentials, if possible. You can also start zonal shifts in the AWS Management Console, for ease of use. But when fast, reliable recovery is critical, data plane operations are a better choice. For more information, see Zonal Shift API Reference Guide.

Move traffic with a zonal shift only temporarily

A zonal shift moves traffic away from an Availability Zone on a temporary basis, to mitigate an impairment. You should restore the resource for the application to service as soon as you've taken action to correct a problem. This ensures that your overall application is restored to its original fully redundant, resilient state.

Zonal shift API operations

The following table lists ARC API operations that you can use using zonal shift, which moves traffic away from an Availability Zone for multi-AZ applications. The table also includes links to relevant documentation.

For examples of how to use common zonal shift API operations with the AWS Command Line Interface, see Examples of using the AWS CLI with zonal shift.

Action	Using the ARC console	Using the ARC API
Start a zonal shift	See Starting a zonal shift	See StartZonalShift
Update a zonal shift	See Updating or canceling a zonal shift	See UpdateZonalShift
List zonal shifts	See Zonal shift in Amazon Application Recovery Controller (ARC)	See ListZonalShifts

Action	Using the ARC console	Using the ARC API
List managed resources	See Resources and scenarios supported for zonal shift and zonal autoshift	See ListManagedResources
Get managed resource	See Resources and scenarios supported for zonal shift and zonal autoshift	See GetManagedResource
Cancel a zonal shift	See Updating or canceling a zonal shift	See CancelZonalShift

Examples of using the AWS CLI with zonal shift

This section walks through simple application examples of using zonal shift, using the AWS Command Line Interface to work with the zonal shift capability in Amazon Application Recovery Controller (ARC) using API operations. The examples are intended to help you develop a basic understanding of how to work with zonal shift using the CLI.

Zonal shift in ARC enables you to temporarily move traffic for supported resources away from an Availability Zone so that your application can continue to operate normally with other Availabilty Zones in an AWS Region. Zonal shift currently supports Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

Let's look at an example of starting a zonal shift using the AWS Command Line Interface. You can also use the AWS CLI to update a zonal shift, for example, to set a new expiration. All zonal shifts are temporary and must be set initially to expire within three days. However, you can update a zonal shift later to set a new expiration.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For a list of zonal shift API actions and links to more information, see Zonal shift API operations.

Start zonal shift

You can start a zonal shift with the CLI by using the `start-zonal-shift` command.

```
aws arc-zonal-shift start-zonal-shift \
```

```
      --resource-identifier="arn:aws:testservice::111122223333:ExampleALB123456890" \
      --away-from="usw2-az1" \
      --expires-in="5m" \
      --comment="Shifting traffic away from USW2-AZ1"
```

```
{
   "zonalShiftId": "2222222-3333-444-1111",
   "resourceIdentifier": "arn:aws:testservice::111122223333:ExampleALB123456890",
   "awayFrom": "usw2-az1",
   "expiryTime": 2022-11-14T01:40:42+00:00,
   "startTime": 2022-11-14T01:35:42+00:00,
   "status": "ACTIVE",
   "comment": "Shifting traffic away from USW2-AZ1"
}
```

Get managed resource

You can get information about a managed resource with the CLI by using the `get-managed-resource` command.

```
aws arc-zonal-shift get-managed-resource \
   --resource-identifier="arn:aws:testservice::111122223333:ExampleALB123456890
```

```
{
    "arn": "arn:aws:testservice::111122223333:ExampleALB123456890",
    "name": "TestResource",
    "appliedWeights": {
        "usw2-az1": 1.0,
        "usw2-az2": 1.0,
        "usw2-az3": 1.0
    },
    "zonalShifts": []
}
```

List managed resources

You can list the managed resources in your account with the CLI by using the `list-managed-resources` command.

```
aws arc-zonal-shift list-managed-resources
```

```
{
    "items": [
        {
            "arn": "arn:aws:testservice::111122223333:ExampleALB123456890",
            "name": "TestResource",
            "availabilityZones": [
                "usw2-az1",
                "usw2-az2",
                "usw2-az3"
            ]
        }
    ]
}
```

List zonal shifts

You can list the zonal shifts in your account with the CLI by using the `list-zonal-shifts` command.

```
aws arc-zonal-shift list-zonal-shifts
```

```
{
    "items": [
        {
            "zonalShiftId": "2222222-3333-444-1111",
            "resourceIdentifier":
 "arn:aws:testservice::111122223333:ExampleALB123456890",
            "awayFrom": "usw2-az1",
            "expiryTime": 2022-11-15T09:10:42+00:00,
            "startTime": 2022-11-13T01:35:42+00:00,
            "status": "ACTIVE",
            "comment": "Shifting traffic away from USW2-AZ1"
        }
    ]
}
```

Update zonal shift

You can update a zonal shift with the CLI by using the `update-zonal-shift` command.

```
aws arc-zonal-shift update-zonal-shift \
```

```
   --zonal-shift-id="="arn:aws:testservice::111122223333:ExampleALB123456890" \
   --expires-in="1h" \
   --comment="Still shifting traffic away from USW2-AZ1"
```

```
{
    "zonalShiftId": "2222222-3333-444-1111",
    "resourceIdentifier": "arn:aws:testservice::111122223333:ExampleALB123456890",
    "awayFrom": "usw2-az1",
    "expiryTime": 2022-11-15T10:35:42+00:00,
    "startTime": 2022-11-15T09:35:42+00:00,
    "status": "ACTIVE",
    "comment": "Still shifting traffic away from USW2-AZ1"
}
```

Cancel zonal shift

You can cancel a zonal shift with the CLI by using the `cancel-zonal-shift` command.

```
aws arc-zonal-shift cancel-zonal-shift \
   --zonal-shift-id="="arn:aws:testservice::111122223333:ExampleALB123456890"
```

```
{
    "zonalShiftId": "2222222-3333-444-1111",
    "resourceIdentifier": "arn:aws:testservice::111122223333:ExampleALB123456890",
    "awayFrom": "usw2-az1",
    "expiryTime": 2022-11-15T10:35:42+00:00,
    "startTime": 2022-11-15T09:35:42+00:00,
    "status": "CANCELED",
    "comment": "Shifting traffic away from USW2-AZ1"
}
```

Resources and scenarios supported for zonal shift and zonal autoshift

Amazon Application Recovery Controller (ARC) currently supports the following resources for zonal shift and zonal autoshift:

- Network Load Balancers with cross-zone load balancing disabled
- Application Load Balancers with cross-zone load balancing disabled

Load balancing resources that are supported are automatically registered with ARC so you can use them with zonal shift (and zonal autoshift). You can start a zonal shift for a load balancer in the Elastic Load Balancing console (in most AWS Regions) or in ARC.

Review the following conditions for working with zonal shifts, zonal autoshift, and resources in ARC:

- Zonal shift and zonal autoshift aren't supported with cross-zone load balancing. For a load balancer to be registered with ARC, make sure that you've disabled or turned off cross-zone load balancing for the load balancer in Elastic Load Balancing.
- In specific scenarios, zonal shift does not shift traffic away from the Availability Zone. If the load balancer target groups in the AZs in a Region don't have any instances, or if all of the instances are unhealthy, then the load balancer is in a fail open state. If you start a zonal shift for a load balancer in this scenario, the zonal shift does not change which AZs the load balancer uses because the load balancer is already in a fail open state. This is expected behavior. Zonal shift cannot force one AZ to be unhealthy and shift traffic to the other AZs in a Region if all AZs are failing open (unhealthy).
- Both public and internal (private) Network Load Balancers and Application Load Balancers are supported.
- A resource must be active and fully provisioned to shift traffic for it. Before you start a zonal shift for a resource, check to make sure that it's a managed resource in ARC. For example, view the list of managed resources in the AWS Management Console, or use the `get-managed-resource` operation with the resource's identifier.
- When an Application Load Balancer is the target of a Network Load Balancer, start the zonal shift from the Network Load Balancer. If you start the zonal shift from the Application Load Balancer, the Network Load Balancer doesn't stop sending traffic to the Application Load Balancer and its targets.
- The resource for a zonal shift must be a managed resource that has been registered with ARC by an AWS service. Elastic Load Balancing automatically registers with ARC all Network Load Balancers and Application Load Balancers that have cross-zone load balancing turned off.
- To start a zonal shift with a resource, it must be deployed in the Availability Zone and AWS Region where you start the shift. Make sure that you start a zonal shift in the same Region that the AZ you want to shift away from is in, and that the resource that you're shifting traffic for is in the same AZ and Region as well.
- Make sure that you have the correct IAM permissions to use zonal shift with a resource. For more information, see IAM and permissions for zonal shift.

Starting, updating, or canceling a zonal shift

This section provides procedures for working with zonal shifts, including starting a zonal shift and canceling a zonal shift.

Starting a zonal shift

The steps in this section explain how to start a customer-initiated zonal shift on the Amazon Application Recovery Controller (ARC) console. To work with zonal shift programmatically, see the Zonal Shift API Reference Guide.

In addition to starting a zonal shift in ARC, you can also start a zonal shift for a load balancer in the Elastic Load Balancing console (in supported Regions). For more information, see Zonal shift in the Elastic Load Balancing User Guide.

To start a zonal shift

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal shift**.
3. On the **Zonal shift** page, choose **Start zonal shift**.
4. Select the Availability Zone that you want to move traffic away from.
5. Select a load balancer from the **Resources** table to move traffic away for.
6. For **Set zonal shift expiration**, choose or enter an expiration for the zonal shift. A zonal shift can set to be active initially for 1 minute or up to three days (72 hours).

   All zonal shifts are temporary. You must set an expiration, but you can update active shifts later to set a new expiration period of up to three days.
7. Enter a comment. You can update the zonal shift later to edit the comment, if you like.
8. Select the check box to acknowledge that starting a zonal shift will reduce available capacity for your application by shifting traffic away from the Availability Zone.
9. Choose **Start**.

Updating or canceling a zonal shift

The steps in this section explain how to update a zonal shift that you initiate, or cancel a zonal shift, on the Amazon Application Recovery Controller (ARC) console. To work with zonal shift programmatically, see the Zonal Shift API Reference Guide.

You can update a zonal shift to set a new expiration, or edit or replace the comment for the zonal shift. You can cancel a zonal shift any time before it expires.

You can cancel zonal shifts that you initiate, or zonal shifts that AWS starts for a resource for a practice run for zonal autoshift. To learn more about practice shifts in zonal autoshift, see How zonal autoshift and practice runs work.

To update a zonal shift

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal shift**.
3. Select a zonal shift that you want to update, and then choose **Update zonal shift**.
4. For **Set zonal shift expiration**, optionally select or enter an expiration.
5. For **Comment**, optionally edit the existing comment or enter a new comment.
6. Choose **Update**.

To cancel a zonal shift

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal shift**.
3. Select a zonal shift that you want to cancel, and then choose **Cancel zonal shift**.
4. On the confirmation modal dialog, choose **Confirm**.

Logging and monitoring for zonal shift in Amazon Application Recovery Controller (ARC)

You can use AWS CloudTrail and Amazon EventBridge for monitoring zonal shift in Amazon Application Recovery Controller (ARC), to analyze patterns and help troubleshoot issues.

Topics

Logging zonal shift API calls using AWS CloudTrail

Zonal shift for Amazon Route 53 Application Recovery Controller is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Route 53 ARC. CloudTrail captures all API calls for zonal shift as events. The calls captured include calls from the Route 53 ARC console and code calls to the Route 53 ARC API operations for zonal shift.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for zonal shift. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to Route 53 ARC for zonal shift, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Zonal shift information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Route 53 ARC for zonal shift, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Working with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for zonal shift in Route 53 ARC, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services, to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations

- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All Route 53 ARC actions are logged by CloudTrail and are documented in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. For example, calls to the `StartZonalShift` and `ListManagedResources` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Viewing Route 53 ARC events in event history

CloudTrail lets you view recent events in **Event history**. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*.

Understanding zonal shift log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `ListManagedResources` action for zonal shift.

```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "AssumedRole",
```

```
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AROA33L3W36EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/admin",
            "accountId": "111122223333",
            "userName": "EXAMPLENAME"
          },
          "webIdFederationData": {},
          "attributes": {
            "creationDate": "2022-11-14T16:01:51Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2022-11-14T16:14:41Z",
      "eventSource": "arc-zonal-shift.amazonaws.com",
      "eventName": "ListManagedResources",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.50",
      "userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64
exec-env/AWS_Lambda_python3.8 Botocore/1.20.102",
      "requestParameters": null,
      "responseElements": null,
      "requestID": "VGXG4ZUE7UZTVCMTJGIAF_EXAMPLE",
      "eventID": "4b5c42df-1174-46c8-be99-d67_EXAMPLE",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "managementEvent": true,
      "recipientAccountId": "111122223333"
      "eventCategory": "Management"
      }
    }
```


The following example shows a CloudTrail log entry that demonstrates the `StartZonalShift` action with a conflict exception for zonal shift.


```
{
      "eventVersion": "1.08",
```

```
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AROA33L3W36EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/admin",
            "accountId": "111122223333",
            "userName": "EXAMPLENAME"
          },
          "webIdFederationData": {},
          "attributes": {
            "creationDate": "2022-11-14T16:01:51Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2022-11-14T16:10:38Z",
      "eventSource": "arc-zonal-shift.amazonaws.com",
      "eventName": "StartZonalShift",
     "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.50",
      "userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64
 exec-env/AWS_Lambda_python3.8 Botocore/1.20.102",
      "errorCode": "ConflictException",
      "errorMessage": "There's already an active zonal shift for that resource
 identifier: 'arn:aws:testservice:us-west-2:077059137270:testResource/456apples'.
 Active zonal shift: 'bac23b74-176e-c073-de8f-484ca508910f'",
      "requestParameters": {
        "resourceIdentifier": "arn:aws:testservice:us-
west-2:077059137270:testResource/456apples",
        "awayFrom": "usw2-az1",
        "expiresIn": "2m",
        "comment": "HIDDEN_FOR_SECURITY_REASONS"
      },
      "responseElements": null,
      "requestID": "OP4OYXZ54HUPMIPGWH_EXAMPLE",
      "eventID": "0bca6660-e999-43a5-9008-EXAMPLE",
      "readOnly": false,
      "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management"
    }
  }
```

Using zonal shift with Amazon EventBridge

Using Amazon EventBridge, you can set up event-driven rules that monitor your zonal shift resources and initiate target actions that use other AWS services. For example, you can set a rule for sending out email notifications by signaling an Amazon SNS topic when a zonal shift starts.

You can create rules in Amazon EventBridge to act on zonal shift. An event for zonal shift specifies status information about zonal shifts. For example, an event is created when you start a zonal shift.

To capture specific zonal shift events that you're interested in, define event-specific patterns that EventBridge can use to detect the events. Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

Events are emitted on a best effort basis. They're delivered from ARC to EventBridge in near real-time, under normal operational circumstances. However, situations can arise that might delay or prevent delivery of an event.

For information about how EventBridge rules work with event patterns, see Events and Event Patterns in EventBridge.

Monitor a zonal shift resource with EventBridge

With EventBridge, you can create rules that define actions to take when ARC emits events for its resources. For example, you can create a rule that sends an email message when you start a zonal shift.

To type or copy and paste an event pattern into the EventBridge console, select to the option to use **Enter my own** option in the console. To help you determine event patterns that might be useful for you, this topic includes examples of zonal shift event-matching patterns.

To create a rule for a resource event

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.

2. Choose the AWS Region that you want to create the rule in, that is the Region that you're interested in watching events for.
3. Choose **Create rule**.
4. Enter a **Name** for the rule, and, optionally, a description.
5. For **Event bus**, leave the default value, **default**.
6. Choose **Next**.
7. For the **Build event pattern** step, for **Event source**, leave the default value, **AWS events**.
8. Under **Sample event**, choose **Enter my own**.
9. For **Sample events**, type or copy and paste an event pattern.

Example ARC event patterns

Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

- *Select all events from ARC zonal shift.*

```
{
    "source": [
        "aws.arc-zonal-shift"
    ]
}
```

Specify a CloudWatch log group to use as a target

When you create an EventBridge rule, you must specify the target where events that are matched to the rule are sent. For a list of available targets for EventBridge, see Targets available in the EventBridge console. One of the targets that you can add to an EventBridge rule is an Amazon CloudWatch log group. This section describes the requirements for adding CloudWatch log groups as targets, and provides a procedure for adding a log group when you create a rule.

To add a CloudWatch log group as a target, you can do one of the following:

- Create a new log group
- Choose an existing log group

If you specify a new log group using the console when you create a rule, EventBridge automatically creates the log group for you. Make sure that the log group that you use as a target for the EventBridge rule starts with `/aws/events`. If you want to choose an existing log group, be aware that only log groups that start with `/aws/events` appear as options in the drop-down menu. For more information, see Create a new log group in the *Amazon CloudWatch User Guide*.

If you create or use a CloudWatch log group to use as a target using CloudWatch operations outside of the console, make sure that you set permissions correctly. If you use the console to add a log group to an EventBridge rule, then the resource-based policy for the log group is updated automatically. But, if you use the AWS Command Line Interface or an AWS SDK to specify a log group, then you must update resource-based policy for the log group. The following example policy illustrates the permissions that you must define in a resource-based policy for the log group:

```
{
  "Statement": [
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "events.amazonaws.com",
          "delivery.logs.amazonaws.com"
        ]
      },
      "Resource": "arn:aws:logs:region:account:log-group:/aws/events/*:*",
      "Sid": "TrustEventsToStoreLogEvent"
    }
  ],
  "Version": "2012-10-17"
}
```

You can't configure a resource-based policy for a log group by using the console. To add the required permissions to a resource-based policy, use the CloudWatch PutResourcePolicy API operation. Then, you can use the describe-resource-policies CLI command to check that your policy was applied correctly.

To create a rule for a resource event and specify a CloudWatch log group target

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Choose the AWS Region that you want to create the rule in.
3. Choose **Create rule** and then enter any information about that rule, such as the event pattern or schedule details.

   For more information about creating EventBridge rules for ARC, see the sections earlier in this topic.
4. On the **Select target** page, choose **CloudWatch** as your target.
5. Choose a CloudWatch log group from the drop-down menu.

Identity and Access Management for zonal shift in Amazon Application Recovery Controller (ARC)

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Route 53 ARC resources. IAM is an AWS service that you can use with no additional charge.

Contents

How zonal shift works with IAM

Before you use IAM to manage access to zonal shift in Amazon Application Recovery Controller (ARC), learn what IAM features are available to use with zonal shift.

IAM features you can use with zonal shift

IAM feature	Zonal shift support
Identity-based policies	Yes

IAM feature	Zonal shift support
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level, overall view of how AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Route 53 ARC

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

To view examples of Route 53 ARC identity-based policies, see Identity-based policy examples in Amazon Route 53 Application Recovery Controller.

Resource-based policies within Route 53 ARC

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource.

Policy actions for zonal shift

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Route 53 ARC actions for zonal shift, see Actions defined by Amazon Route 53 Zonal Shift in the *Service Authorization Reference*.

Policy actions in Route 53 ARC for zonal shift use the following prefixes before the action:

```
arc-zonal-shift
```

To specify multiple actions in a single statement, separate them with commas. For example, the following:

```
"Action": [
      "arc-zonal-shift:action1",
      "arc-zonal-shift:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "arc-zonal-shift:Describe*"
```

To view examples of ARC identity-based policies for zonal shift, see Identity-based policy examples for zonal shift in Amazon Route 53 Application Recovery Controller.

Policy resources for zonal shift

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of resource types and their ARNs, and the actions that you can specify with the ARN of each resource, see the following topic in the *Service Authorization Reference*:

- Actions defined by Amazon Route 53 - Zonal Shift

To see the actions and resources that you can use with a condition key, see the following topic in the *Service Authorization Reference*:

- Condition keys defined by Amazon Route 53 - Zonal Shift

To view examples of ARC identity-based policies for zonal shift, see Identity-based policy examples for zonal shift in Amazon Route 53 Application Recovery Controller.

Policy condition keys for zonal shift

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of zonal shift condition keys, see the following topic in the *Service Authorization Reference*:

- Condition keys defined by Amazon Route 53 - Zonal Shift

To see the actions and resources that you can use with a condition key, see the following topics in the *Service Authorization Reference*:

- Actions defined by Amazon Route 53 - Zonal Shift
- Resource types defined by Amazon Route 53 - Zonal Shift

To view examples of ARC identity-based policies for zonal shift, see Identity-based policy examples for zonal shift in Amazon Route 53 Application Recovery Controller.

Access control lists (ACLs) in Route 53 ARC

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Route 53 ARC

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

ARC includes the following partial support for ABAC:

- Zonal shift supports ABAC for managed resources that are registered in ARC for zonal shift. For more information about ABAC for Network Load Balancer and Application Load Balancer managed resources, see ABAC with Elastic Load Balancing in the Elastic Load Balancing User Guide.

Using temporary credentials with Route 53 ARC

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Route 53 ARC

Supports forward access sessions (FAS): Yes

When you use an IAM entity (user or role) to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions.

To see whether an action requires additional dependent actions in a policy, see the following topic in the *Service Authorization Reference*:

- Amazon Route 53 Zonal Shift

Service roles for Route 53 ARC

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for Route 53 ARC

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Zonal shift does not use service-linked roles.

IAM and permissions for zonal shift

This section provides additional information about how permissions work for the zonal shift feature in Amazon Application Recovery Controller (ARC), especially if you work with the feature from another AWS service, such as Elastic Load Balancing. To learn about how ARC features works with IAM and permissions in general, review the information in the overview topic, Identity and Access Management for zonal shift in Amazon Application Recovery Controller (ARC).

In addition to the permissions outlined in the IAM overview topic, the following applies to zonal shift for IAM and permissions:

- Make sure that you have the required permissions for working with zonal shift in ARC. For more information, see zonal shift console access and zonal shift operations access.
- You do not need to add additional Elastic Load Balancing permissions with IAM to work with zonal shifts for managed load balancer resources in your account in ARC.
- An AWS managed policy that provides full access for Elastic Load Balancing includes permissions for working with zonal shifts. If you use AWS managed policies for Elastic Load Balancing access, you do not need additional permissions in IAM for zonal shift to start zonal shifts for load balancers or work with in the Elastic Load Balancing console. For more information, see AWS managed policies for Elastic Load Balancing.

Identity-based policy examples for zonal shift in Amazon Route 53 Application Recovery Controller

By default, users and roles don't have permission to create or modify Route 53 ARC resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Route 53 ARC, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Route 53 Application Recovery Controller in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Route 53 ARC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies

adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Example: Zonal shift console access

To access the Amazon Route 53 Application Recovery Controller console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Route 53 ARC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To give users full access to use zonal shift in the AWS Management Console, attach a policy like the following to the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                   "arc-zonal-shift:ListManagedResources",
                   "arc-zonal-shift:GetManagedResource",
                   "arc-zonal-shift:ListZonalShifts",
                   "arc-zonal-shift:StartZonalShift",
                   "arc-zonal-shift:UpdateZonalShift",
                   "arc-zonal-shift:CancelZonalShift"
             ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeAvailabilityZones",
            "Resource": "*"
        }
    ]
}
```

Example: Zonal shift API actions

The zonal shift API temporarily moves traffic away from an Availability Zone to recover an application.

To ensure that a user can use zonal shift API actions, attach a policy that corresponds to the API operations that the user needs to work with, such as the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                   "arc-zonal-shift:ListManagedResources",
                   "arc-zonal-shift:GetManagedResource",
                   "arc-zonal-shift:ListZonalShifts",
                   "arc-zonal-shift:StartZonalShift",
                   "arc-zonal-shift:UpdateZonalShift",
                   "arc-zonal-shift:CancelZonalShift"
             ],
            "Resource": "*"
        }
    ]
}
```

Zonal autoshift in Amazon Application Recovery Controller (ARC)

With zonal autoshift, you authorize AWS to shift away resource traffic for an application from an Availability Zone during events, on your behalf, to help reduce time to recovery. AWS starts an

autoshift when internal telemetry indicates that there is an Availability Zone impairment that could potentially impact customers. When AWS starts an autoshift, application traffic to resources that you've configured for zonal autoshift starts shifting away from the Availability Zone.

Be aware that ARC does not inspect the health of individual resources. AWS starts an autoshift when AWS telemetry detects that there is an Availability Zone impairment that could potentially impact customers. In some cases, traffic might be shifted away for resources that are not experiencing impact.

With zonal autoshift, you also authorize AWS to shift away resource traffic for an application from an Availability Zone, on your behalf, for regular practice runs. Practice runs are required for zonal autoshift. The zonal shifts that ARC starts for practice runs help you to ensure that shifting away traffic from an Availability Zone during an autoshift is safe for your application. Practice runs regularly test that your application can operate normally without one Availability Zone by starting zonal shifts that shift traffic for a resource away from an Availability Zone. Practice runs take place weekly, and provide an outcome—such as `SUCCEEDED` or `FAILED`—to help you understand if the application operates as expected.

> **Important**
>
> Before you configure practice runs or enable zonal autoshift, we strongly recommend that you pre-scale your application resource capacity in all Availability Zones in the Region where your application resources are deployed. You should not rely on scaling on demand when an autoshift or practice run starts. Zonal autoshift, including practice runs, works independently, and does not wait for auto scaling actions to complete. Relying on auto scaling, instead of pre-scaling, can result in it taking longer for your application to recover. If you use auto scaling to handle regular cycles of traffic, we strongly recommend that you configure the minimum capacity of your auto scaling to continue operating normally with the loss of an Availability Zone.

If you plan to enable zonal autoshift or configure practice runs, after you pre-scale your application resource capacity, test that your application can operate normally without one Availability Zone. To test this, start a zonal shift to move traffic for a resource away from an Availability Zone.

To ensure your tests with zonal shift are effective, it's important to validate that traffic drains as expected from the AZ you shift away from. Both Application Load Balancers and Network Load Balancers provide per AZ metrics in Amazon CloudWatch that you can use to monitor this.

Depending on how long a service and clients reuse connections, traffic might continue to the AZ that you have shifted away from for longer than you expect. To learn more, see Limit the time that clients stay connected to your endpoints.

After you verify, by starting and evaluating a zonal shift, that your application can continue operating normally with traffic shifted away from an Availability Zone, the regular practice runs that ARC performs help you to confirm, on an ongoing basis, that you have enough capacity for an autoshift.

In addition to enabling zonal autoshift for a load balancer resource in the ARC console, you have the option to instead enable zonal autoshift for a specific load balancer in the Amazon EC2 console. To learn more about enabling zonal autoshift with Elastic Load Balancing, see Zonal shift in the Elastic Load Balancing User Guide.

Autoshifts and practice run zonal shifts are temporary. With autoshifts, when the affected Availability Zone recovers, AWS stops shifting traffic for resources away from the Availability Zone. Application traffic for customers returns to all Availability Zones in the Region. With a practice run, traffic is shifted away from an Availability Zone for a single resource for about 30 minutes, and then shifted back to all Availability Zones in the Region.

You can configure Amazon EventBridge notifications to alert you about autoshifts and practice runs. For more information, see Using zonal autoshift with Amazon EventBridge.

How zonal autoshift and practice runs work

The zonal autoshift capability in Amazon Application Recovery Controller (ARC) allows AWS to shift traffic for a resource away from an Availability Zone, on your behalf, when AWS determines that there's an impairment that could potentially affect customers in the Availability Zone. Zonal autoshift is designed for a resource that is pre-scaled in all Availability Zones in an AWS Region, so that an application can operate normally with the loss of one Availability Zone.

With zonal autoshift, you are required to configure practice runs, where ARC regularly shifts traffic for the resource away from one Availability Zone. ARC schedules practice runs about weekly for each resource that has a practice run configuration associated with it. Practice runs for each resource are scheduled independently.

For each practice run, ARC records an outcome. If a practice run is interrupted by a blocking condition, the practice run outcome is not marked as successful. For more information about practice run outcomes, see Outcomes for practice runs.

You can configure Amazon EventBridge notifications to send you information about autoshifts and practice runs. For more information, see Using zonal autoshift with Amazon EventBridge.

Topics

When AWS starts and stop autoshifts

When you enable zonal autoshift for a resource, you authorize AWS to shift away resource traffic for an application from an Availability Zone during events, on your behalf, to help reduce time to recovery.

To achieve this, zonal autoshift uses AWS telemetry to detect, as early as possible, that there is an Availability Zone impairment that could potentially impact customers. When AWS starts an autoshift, traffic to configured resources immediately starts shifting away from the impaired Availability Zone that could potentially impact customers.

Zonal autoshift is a capability designed for customers who have pre-scaled their application resources for all Availability Zones in an AWS Region. You should not rely on scaling on demand when an autoshift or practice run starts.

AWS ends an autoshift when it determines that the Availability Zone has recovered.

When ARC schedules, starts, and ends practice runs

ARC schedules a practice run for a resource weekly, for about 30 minutes. ARC schedules, starts, and manages practice runs for each resource independently. ARC does not batch together practice runs for resources in the same account.

When a practice run continues for the expected duration, without interruption, it is marked with an outcome of SUCCESSFUL. There are several other possible outcomes: FAILED,

INTERRUPTED, and PENDING. Outcome values and descriptions are included in the Outcomes for practice runs section.

There are some scenarios when ARC interrupts a practice run and ends it. For example, if an autoshift starts during a practice run, ARC interrupts the practice run and ends it. As another example, say that the resource has an adverse response to a practice run and causes an alarm that you've specified to monitor the practice run to go into an ALARM state. In this scenario, ARC also interrupts the practice run and ends it.

In addition, there are several scenarios when ARC does not start a schedule practice run for a resource.

In response to interrupted and blocked practice runs for a resource, ARC does the following:

- If a practice run for a resource is interrupted while it's in progress, ARC considers the weekly practice run to be over, and schedules a new practice run for the resource for the next week. The weekly practice outcome is INTERRUPTED in this scenario, not FAILED. The practice run outcome set to FAILED only when the outcome alarm that monitors the practice run goes into an ALARM state during the practice run.
- If there is a blocking constraint when a practice run for a resource is scheduled to be started, ARC does not start the practice run. ARC continues regular monitoring, to determine if there are still one or more blocking constraints. When there aren't any blocking constraints, ARC starts the practice run for the resource.

The following are examples of blocking constraints that stop ARC from starting, or continuing, a practice run for a resource:

- ARC does not start or continue practice runs when there is an AWS Fault Injection Service experiment in progress. If an AWS FIS event is active when ARC has scheduled a practice run to start, ARC does not start the practice run. ARC monitors throughout practice runs for blocking constraints, including an AWS FIS event. If an AWS FIS event starts while a practice run is active, ARC ends the practice run and doesn't attempt to start another one until the next regularly scheduled practice run for the resource.
- If there is a current AWS event in a Region, ARC does not start practice runs for resources, and ends active practice runs, in the Region.

When the practice run finishes without being interrupted, ARC schedules the next practice run in a week, as usual. If a practice run isn't started because of a blocking constraint, such as a AWS FIS experiment or a blocked time window that you've specified, ARC continues to attempt to start a practice run until the practice run can be started.

Notifications for practice runs and autoshifts

You can choose to be notified about practice runs and autoshifts for your resource by setting up Amazon EventBridge notifications. You can also set up EventBridge notifications when you haven't enabled zonal autoshift for any resources, known as *autoshift observer notification*. With autoshift observer notification, you are notified about all autoshifts that ARC starts when an Availability Zone is potentially impaired. Note that you must configure this option in each AWS Region that you want to receive notifications about.

To see the steps for enabling autoshift observer notification, see Enabling and working with zonal autoshift. To learn more about notification options and how to configure them in EventBridge, see Using zonal autoshift with Amazon EventBridge.

Precedence for zonal shifts, practice runs, and autoshifts

There can be no more than one traffic shift for a resource that is in effect at once—that is, only one practice run zonal shift, customer-initiated zonal shift, or autoshift for the resource. When there is more than one traffic shift in progress, ARC follows a precedence to determine which traffic shift is in effect for a resource.

The overall principle for precedence is that zonal shifts that you start as a customer take precedence over autoshifts, which take precedence over practice runs. That is, customer-initiated zonal shifts > autoshifts > practice run zonal shifts.

To illustrate this, the following is how precedence works for a few example scenarios:

- If there is an active autoshift and you start a zonal shift for a resource that has autoshift enabled, the zonal shift that you start is `APPLIED`. The resource is now shifted away from the Availability Zone that the zonal shift applies to. If the zonal shift ends before AWS ends the autoshift, then the autoshift becomes the `APPLIED` shift. So, the resource is shifted away from the Availability Zone where AWS has the autoshift in progress.
- If there's an active zonal shift that you're started for a resource that has autoshift enabled, and AWS starts an autoshift, the autoshift exists for the resource. However, the zonal shift is set to `APPLIED` and the autoshift is set to `NOT APPLIED` until the zonal shift ends. Then, the status for the autoshift is updated to `APPLIED` and the autoshift shifts traffic away for the resource until AWS ends the autoshift.
- If there's an active practice run for a resource and you start a zonal shift for the resource that shifts traffic away for the same Availability Zone, the practice run is interrupted. If you start a zonal shift that shifts traffic away from a different Availability Zone, the practice run continues as usual.

- If there's an active zonal shift for a resource and ARC is scheduled to start a practice run, the practice run is deferred for an hour. Then ARC attempts again to start the practice run. ARC continues to check hourly until a practice run can be started.

The traffic shift that is currently in effect for the resource has an applied zonal shift status set to `APPLIED`. Only one shift is set to `APPLIED` at any time. Other shifts that are in progress are set to `ACTIVE`.

Stopping an active autoshift or practice run for a resource

To stop an in-progress autoshift for a resource, disable zonal autoshift for the resource.

When you disable zonal autoshift, the practice run configuration for the resource is not affected. Regular practice runs still take place for the resource, on the same schedule. If you want to stop practice runs in addition to disabling autoshifts, you must delete the practice run configuration associated with the resource.

When you delete a practice run configuration, AWS stops performing practice runs that shift traffic for the resource away from an Availability Zone each week. In addition, because zonal autoshift requires practice runs, when you delete a practice run configuration using the ARC console, this action also disables zonal autoshift for the resource. However, note that if you use the zonal autoshift API to delete a practice run, you must first disable zonal autoshift for the resource.

To stop a active practice run, cancel the practice run zonal shift. For more information, see Canceling a practice run zonal shift.

How traffic is shifted away

For autoshifts and for practice run zonal shifts, traffic is shifted away from an Availability Zone using the same mechanism that ARC uses for customer-initiated zonal shifts. To shift traffic away from an Availability Zone for load balancers that have cross-zone load balancing turned off, ARC sets the load balancer health check for the Availability Zone to unhealthy, so that it fails its health check. An unhealthy health check, in turn, results in Amazon Route 53 withdrawing the corresponding IP addresses for the resource from DNS, so that traffic is redirected from the Availability Zone. New connections are now routed to other Availability Zones in the AWS Region instead.

With an autoshift, when an Availability Zone recovers and AWS decides to end the autoshift, ARC reverses the health check process, requesting the Route 53 health checks to be reverted.

Then, the original zonal IP addresses are restored and, if the health checks continue to be healthy, the Availability Zone is included in the load balancer's routing again.

It's important to be aware that autoshifts are not based on health checks that monitor the underlying health of load balancers or applications. ARC uses health checks to move traffic away from Availability Zones, by requesting health checks to be set to unhealthy, and then restores health checks to normal again when it ends an autoshift or zonal shift.

Alarms for practice runs

You can specify two CloudWatch alarms for practice runs in zonal autoshift. The first alarm, the *outcome alarm*, is required. You should configure the outcome alarm to monitor the health of your application when traffic is shifted away from an Availability Zone during each 30-minute practice run.

For a practice run to be effective, specify as an outcome alarm a CloudWatch alarm that monitors metrics for the resource, or your application, that respond with an `ALARM` state when your application is adversely affected by the loss of one Availability Zone. For more information, see the **Alarms that you specify for practice runs** section in Best practices when you configure zonal autoshift.

The outcome alarm also provides information for the practice run result that ARC reports for each practice run. If the alarm enters an `ALARM` state, the practice run is ended and the practice run outcome is returned as `FAILED`. If the practice run completes the 30 minute scheduled test period and the outcome alarm does not enter an `ALARM` state, the outcome is returned as `SUCCEEDED`. A list of all outcome values, with descriptions, is provided in the Outcomes for practice runs section.

Optionally, you can specify a second alarm, the *blocking alarm*. The blocking alarm blocks practice runs from starting, or continuing, when it's in an `ALARM` state. This alarm blocks practice run traffic shifts from being started—and stops any practice runs in progress—when the alarm is in an `ALARM` state.

For example, in a large architecture with multiple microservices, when one microservice is experiencing a problem, you typically want to stop all other changes in the application environment, which would including blocking practice runs.

Blocked dates and blocked windows (UTC)

You have the option to block practice runs for specific calendar dates, or for specific time windows, that is, days and times, in UTC.

For example, if you have an application update scheduled to launch on May 1, 2024, and you don't want practice runs to shift traffic away at that time, you could set a blocked date for `2024-05-01`.

Or, say you run business report summaries three days a week. For this scenario, you might set the following recurring days and times as blocked windows, for example, in UTC: `MON-20:30-21:30 WED-20:30-21:30 FRI-20:30-21:30`.

About zonal autoshift

Zonal autoshift is a capability where AWS shifts application resource traffic away from an Availability Zone, on your behalf. AWS starts an autoshift when internal telemetry indicates that there is an Availability Zone impairment that could potentially impact customers. The internal telemetry incorporates metrics from several sources, including the AWS network, and the Amazon EC2 and Elastic Load Balancing services.

You can enable zonal autoshift for Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

When you deploy and run AWS applications on load balancers in multiple (typically three) AZs in a Region, and you pre-scale to support static stability, AWS can quickly recover customer applications in an AZ by shifting traffic away with an autoshift. By shifting away resource traffic to other AZs in the Region, AWS can reduce the duration and severity of potential impact caused by power outages, hardware or software issues in an AZ, or other impairments.

When AWS begins an autoshift for a load balancing resource, ARC sets Amazon Route 53 health checks to unhealthy for the corresponding IP addresses for the load balancer resource, so that traffic for the resource is no longer directed to the AZ. When AWS determines that the AZ is ready for application traffic to return, ARC restores the Route 53 health checks, and the original zonal IP addresses are restored.

When you enable zonal autoshift for a resource, you must also configure a practice run for the resource. AWS performs practice runs about weekly, for 30 minutes, to help you make sure that you have enough capacity to run your application without one of the Availability Zones in the Region.

As with zonal shift, there are a few specific scenarios where zonal autoshift does not shift traffic away from the AZ. For example, if the load balancer target groups in the AZs don't have any instances, or if all of the instances are unhealthy, then the load balancer is in a fail open state and you can't shift away one of the AZs.

To learn more about zonal autoshift, see Zonal autoshift in Amazon Application Recovery Controller (ARC).

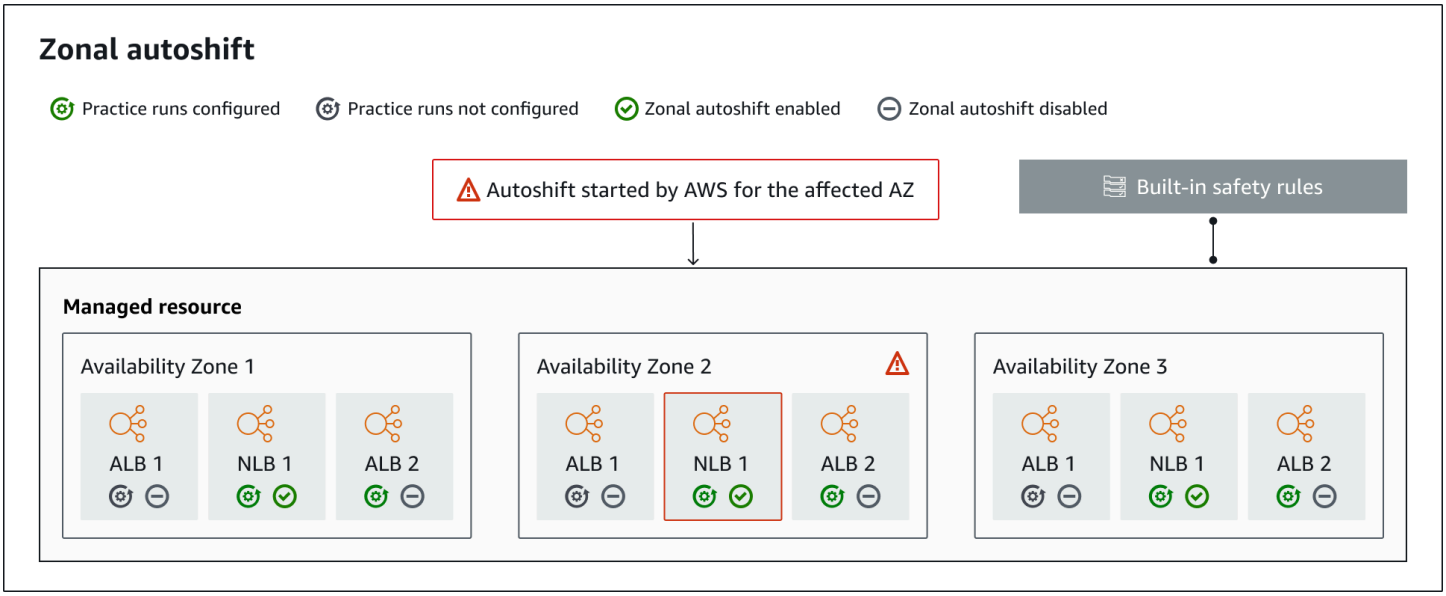
AWS Region availability for zonal autoshift

Zonal autoshift is currently available in the commercial AWS Regions.

For detailed information about Regional support and service endpoints for Amazon Application Recovery Controller (ARC), see Amazon Application Recovery Controller (ARC) endpoints and quotas in the *Amazon Web Services General Reference*.

Zonal autoshift components

The following diagram illustrates an example of an autoshift shifting traffic away from an Availability Zone. AWS starts an autoshift when internal telemetry indicates that there is an Availability Zone impairment that could potentially impact customers.



The following are components of the zonal autoshift capabilities in ARC.

Zonal autoshift

Zonal autoshift shifts traffic away for a resource, without requiring you to take any action. Zonal autoshift is a capability in ARC where AWS starts an autoshift when internal telemetry indicates that there is an Availability Zone impairment that could potentially impact customers. Be aware that, in some cases, resources might be shifted away that are not experiencing impact.

Practice runs

When you enable zonal autoshift for a resource, you must also configure zonal autoshift *practice runs* for the resource. AWS performs a zonal shift for practice runs about weekly, for about 30 minutes. Practice runs make sure that your application can run normally with the loss of one Availability Zone. In a practice run, AWS shifts traffic for a resource away from one Availability Zone with a zonal shift, and then shifts traffic back when the practice run ends.

Practice run configuration

A practice run configuration defines the blocked dates and windows, if any, and the CloudWatch alarms that you specify for the practice run for a resource in zonal autoshift. You can edit a practice run at any time, to add or change blocked dates or windows, or to update the alarms for the practice run.

To enable zonal autoshift, you must have a practice run configuration in place for a resourceyou can also delete a practice run. To delete a practice run configuration for a resource, zonal autoshift must be disabled.

Practice run alarm

When you configure practice runs, you specify CloudWatch alarms that you create in CloudWatch, based on your resource and application requirements. The alarms that you specify can block a practice run from starting, or can stop a practice run in progress, if your application is adversely affected by the practice run.

If an alarm that you specify goes into an `ALARM` state, ARC ends the zonal shift for the practice run, so that traffic for the resource is no longer shifted away from the Availabilty Zone.

There are two types of alarms that you specify for practice runs: an *outcome* alarm, to monitor the health of your resource and application during the practice run, and a *blocking* alarm, which you can configure to prevent practice runs from starting, or to stop an in-progress practice run. The outcome alarm is required; the blocking alarm is optional.

Practice run outcome

ARC reports an outcome for each practice run. The following are the possible practice run outcomes:

- **PENDING:** The zonal shift for the practice run is active (in progress). There's no outcome to return yet.

- **SUCCEEDED:** The outcome alarm did not enter an `ALARM` state during the practice run, and the practice run completed the full 30 minute test period.
- **INTERRUPTED:** The practice run ended for a reason that was not the outcome alarm entering an `ALARM` state. A practice run can be interrupted for a variety of reasons. For example, a practice run that ends because the blocking alarm specified for the practice run entered an `ALARM` state has an outcome of `INTERRUPTED`. For more information about reasons for an `INTERRUPTED` outcome, see Outcomes for practice runs.
- **FAILED:** The outcome alarm entered an `ALARM` state during the practice run.

Built-in safety rules

Safety rules built into ARC prevent more than one traffic shift for a resource from being in effect at a time. That is, only one customer-initiated zonal shift, practice run zonal shift, or autoshift for the resource can be actively shifting traffic away from an Availability Zone. For example, if you start a zonal shift for a resource when it is currently shifted away with autoshift, your zonal shift takes precedence. For more information, see Outcomes for practice runs.

Resource identifier

The identifier for a resource to enable zonal autoshift for, which is the Amazon Resource Name (ARN) for the resource.

You can only enable zonal autoshift for resources in your account that are in an AWS service that is supported by ARC. Supported resources in those AWS services are automatically registered with ARC by the AWS service.

> **Note**
>
> You can only configure zonal autoshift for Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

Managed resource

AWS services register resources automatically with ARC for zonal autoshift. A resource that has been registered is a managed resource in ARC.

Resource name

The name of a managed resource in ARC.

Applied status

An applied status indicates whether a traffic shift is in effect for a resource. When you configure zonal autoshift, a resource can have more than one active traffic shift—that is, a practice run zonal shift, customer-initiated zonal shift, or autoshift. However, only one is applied, that is, is in effect for the resource at a time. The shift that has the status `APPLIED` determines the Availability Zone where application traffic has been shifted away for a resource, and when that traffic shift ends.

Data and control planes for zonal autoshift

As you plan for failover and disaster recovery, consider how resilient your failover mechanisms are. We recommend that you make sure that the mechanisms that you depend on during failover are highly available, so that you can use them when you need them in a disaster scenario. Typically, you should use data plane functions for your mechanisms whenever you can, for the greatest reliability and fault tolerance. With that in mind, it's important to understand how the functionality of a service is divided between control planes and data planes, and when you can rely on an expectation of extreme reliability with a service's data plane.

In general, a *control plane* enables you to do basic management functions, such as create, update, and delete resources in the service. A *data plane* provides a service's core functionality.

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see the Static stability using Availability Zones paper in the Amazon Builders' Library.

Pricing for zonal autoshift in Amazon Application Recovery Controller (ARC)

For zonal autoshift, AWS shifts traffic away from an Availability Zone on your behalf for supported resources when AWS determines that there is a potential issue that can adversely affect customer applications. There is no additional charge for enabling zonal autoshift.

You only pay for what you use in Amazon Application Recovery Controller (ARC). For detailed pricing information for ARC and pricing examples, see Amazon Route 53 Pricing and scroll down to Amazon Application Recovery Controller (ARC).

Best practices when you configure zonal autoshift

Be aware of the following best practices and considerations when you enable zonal autoshift in Amazon Application Recovery Controller (ARC).

Zonal autoshift includes two types of traffic shifts: autoshifts and practice run zonal shifts.

- With an *autoshift*, AWS helps reduce your time to recovery by shifting away application resource traffic from an Availability Zone during events, on your behalf.
- With *practice runs*, ARC starts a zonal shift on your behalf. The zonal shift shifts traffic away from an Availability Zone for a resource, and back again, on a weekly cadence. Practice runs help you to make sure that you have scaled up sufficient capacity for Availability Zones in a Region for your application to tolerate the loss of one Availability Zone.

There are several best practices and considerations to keep in mind with autoshifts and practice runs. Review the following topics before you enable zonal autoshift or configure practice runs for a resource.

Topics

Limit the time that clients stay connected to your endpoints

When Amazon Application Recovery Controller (ARC) shifts traffic away from an impairment, for example, by using zonal shift or zonal autoshift, the mechanism that ARC uses to move your application traffic is a DNS update. A DNS update causes all new connections to be directed away from the impaired location. However, clients with pre-existing open connections might continue to make requests against the impaired location until the clients reconnect. To ensure a quick recovery, we recommend that you limit the amount of time clients stay connected to your endpoints.

If you use an Application Load Balancer, you can use the `keepalive` option to configure how long connections continue. We suggest that you lower the `keepalive` value to be inline with your recovery time goal for your application, for example, 300 seconds. When you choose a `keepalive` time, consider that this value is a trade off between reconnecting more frequently in general, which can affect latency, and more quickly moving all clients away from an impaired AZ or Region.

For more information about setting the `keepalive` option for Application Load Balancer, see the HTTP client keepalive duration in the Application Load Balancer User Guide.

Prescale your resource capacity and test shifting traffic

When AWS shifts traffic away from one Availability Zone for a zonal shift or an autoshift, it's important that the remaining Availability Zones can service the increased request rates for your resource. This pattern is known as *static stability*. For more information, see the Static stability using Availability Zones whitepaper in the Amazon Builder's Library.

For example, if your application requires 30 instances to serve its clients, you should provision 15 instances across three Availability Zones, for a total of 45 instances. By doing this, when AWS shifts traffic away from one Availability Zone—with an autoshift or during a practice run—AWS can still serve your application's clients with the remaining total of 30 instances, across two Availability Zones.

The zonal autoshift capability in ARC helps you to quickly recover from AWS events in an Availability Zone when you have an application with resources that are pre-scaled to work normally with the loss of one Availability Zone. Before you enable zonal autoshift for a resource, scale your resource capacity in all configured Availability Zones in an AWS Region. Then, start zonal shifts for the resource, to test that your application still runs normally when traffic is shifted away from an Availability Zone.

After you test with zonal shifts, then enable zonal autoshift and configure practice runs for application resources. Regular practice runs with zonal autoshift help you to make sure—on an ongoing basis—that your capacity is still scaled appropriately. With sufficient capacity across Availability Zones, your application can continue to serve clients, without interruption, during an autoshift.

For more information about starting a zonal shift for a resource, see Zonal shift in Amazon Application Recovery Controller (ARC).

Be aware of resource types and restrictions

Zonal autoshift supports shifting traffic out of an Availability Zone for all resources that are supported by zonal shift. In general, Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off are supported. In a few specific resource scenarios, zonal autoshift does not shift traffic from an Availability Zone for an autoshift.

For example, if the load balancer target groups in the Availability Zones don't have any instances, or if all of the instances are unhealthy, then the load balancer is in a fail open state. If AWS starts an autoshift for a load balancer in this scenario, an autoshift does not change which Availability Zones the load balancer uses because the load balancer is already in a fail open state. This is expected behavior. Autoshift cannot cause one Availability Zone to be unhealthy and shift traffic to the other Availability Zones in an AWS Region if all Availability Zones are failing open (unhealthy).

A second scenario is if AWS starts an autoshift for an Application Load Balancer that is an endpoint for an accelerator in AWS Global Accelerator. As with zonal shift, autoshift isn't supported for Application Load Balancers that are endpoints of accelerators in Global Accelerator.

To see details about supported resources, including all of the requirements and exceptions to be aware of, see Resources and scenarios supported for zonal shift and zonal autoshift.

Specify alarms for practice runs

You configure at least one alarm—the outcome alarm—for practice runs with zonal autoshift. Optionally, you can also configure a second alarm—the blocking alarm—.

When you consider the CloudWatch alarms that you configure for practice runs for your resource, keep in mind the following:

- For the outcome alarm, which is required, we recommend that you configure a CloudWatch alarm to go into an `ALARM` state when metrics for the resource, or your application, indicate that shifting traffic away from the Availability Zone adversely impacts performance. For example, you can determine a threshold for request rates for your resource, and then configure an alarm to go into an `ALARM` state when the threshold is exceeded. You are responsible for configuring an appropriate alarm that causes AWS to end the practice run and return a `FAILED` outcome.
- We recommend that you follow the AWS Well Architected Framework, which advises you to implement key performance indicators (KPIs) as CloudWatch alarms. If you do so, you can use these alarms to create a composite alarm to use as a safety trigger, to prevent practice runs

from starting if they might cause your application to miss a KPI. When the alarm is no longer in an ALARM state, ARC starts practice runs the next time a practice run is scheduled for the resource.

- For the practice run blocking alarm, if you choose to configure it, you might choose to track a specific metric that you use to indicate that you don't want a practice run to start.
- For practice run alarms, you specify the Amazon Resource Name (ARN) for each alarm, which you must first configure in Amazon CloudWatch. The CloudWatch alarms that you specify can be composite alarms, to enable you to include several metrics and checks for your application and resource that can trigger the alarm to go into an ALARM state. For more information, see Combining alarms in the Amazon CloudWatch User Guide.
- Make sure that the CloudWatch alarms that you specify for practice runs are in the same Region as the resource that you're configuring a practice run for.

Evaluate outcomes for practice runs

ARC reports an outcome for each practice run. After a practice run, evaluate the outcome, and determine if you need to take action. For example, you might need to scale capacity or adjust the configuration for an alarm.

The following are the possible practice run outcomes:

- **SUCCEEDED:** The outcome alarm did not enter an ALARM state during the practice run, and the practice run completed the full 30 minute test period.
- **FAILED:** The outcome alarm entered an ALARM state during the practice run.
- **INTERRUPTED:** The practice run ended for a reason that was not the outcome alarm entering an ALARM state. A practice run can be interrupted for a variety of reasons, including the following:
  - Practice run was ended because AWS started an autoshift in the AWS Region or there was an alarm condition in the Region.
  - Practice run was ended because the practice run configuration was deleted for the resource.
  - Practice run was ended because a customer-initiated zonal shift was started for the resource in the Availability Zone that the practice run zonal shift was shifting traffic away from.
  - Practice run was ended because a CloudWatch alarm that was specified for the practice run configuration can no longer be accessed.
  - Practice run was ended because the blocking alarm specified for the practice run entered an ALARM state.

  - Practice run was ended for an unknown reason.
- **PENDING:** The practice run is active (in progress). There's no outcome to return yet.

Zonal autoshift API operations

The following table lists ARC API operations that you can use with zonal autoshift. For examples of using zonal autoshift API operations with the AWS CLI, see .

For examples of how to use common zonal autoshift API operations with the AWS Command Line Interface, see Examples of using the AWS CLI with zonal autoshift.

Action	Using the ARC console	Using the ARC API
Create a practice run configuration	See Enabling or disabling zonal autoshift	See CreatePracticeRunConfiguration
Delete a practice run configuration	See Configuring, editing, or deleting a practice run configuration	See DeletePracticeRunConfiguration
List autoshifts	See Zonal autoshift in Amazon Application Recovery Controller (ARC)	See ListAutoshifts
List resources for zonal autoshift	See Resources and scenarios supported for zonal shift and zonal autoshift	See ListManagedResources
Get resources for zonal autoshift	See Resources and scenarios supported for zonal shift and zonal autoshift	See GetManagedResource
Edit a practice run configuration	See Configuring, editing, or deleting a practice run configuration	See UpdatePracticeRunConfiguration
Enable or disable zonal autoshift	See Enabling or disabling zonal autoshift	See UpdateZonalAutoshiftConfiguration

Action	Using the ARC console	Using the ARC API
Enable or disable autoshift observer notification	See Enabling and working with zonal autoshift	See UpdateAutoshiftObserverNotificationStatus

Examples of using the AWS CLI with zonal autoshift

This section walks through simple application examples of working with zonal autoshift, using the AWS Command Line Interface to work with the zonal autoshift capability in Amazon Application Recovery Controller (ARC) using API operations. The examples are intended to help you develop a basic understanding of how to work with zonal autoshift using the CLI.

Zonal autoshift is a capability in ARC. With zonal autoshift, you authorize AWS to shift away supported application resource traffic from an Availability Zone during events, on your behalf, to help reduce your time to recovery. Zonal autoshift includes practice runs, which also shift traffic away from Availability Zones, to help verify, on an ongoing basis, that autoshifts are safe for your application.

Zonal autoshift currently supports Network Load Balancers and Application Load Balancers with cross-zone load balancing turned off.

For more information, see Resources and scenarios supported for zonal shift and zonal autoshift.

This section provides the following examples to illustrate how to get started with and work with zonal autoshift:

- Create a practice run configuration for a resource.
- Enable and disable autoshifts for a resource.
- End an in-progress practice run by canceling the zonal shift started by the practice run.
- End an in-progress autoshift by disabling the zonal autoshift feature for a resource.
- Edit a practice run configuration for a resource to change the specified alarms or blocked dates or windows.
- Delete a practice run configuration for a resource.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For a list of zonal autoshift API actions and links to more information, see Zonal autoshift API operations.

Create practice run configuration

Before you can enable zonal autoshift for a resource, you must create a practice run configuration for the resource, to choose options for the required practice runs. You create a practice run configuration for a resource with the CLI by using the `create-practice-run-configuration` command.

Note the following when you create a practice run configuration for a resource:

- The only supported alarm type at this time is `CLOUDWATCH`.
- You must use alarms that are in the same AWS Region that your resource is deployed in.
- Specifying an outcome alarm is required. Specifying a blocking alarm is optional.
- Specifying blocked dates or blocked windows is optional.

You create a practice run configuration with the CLI by using the `create-practice-run-configuration` command.

For example, to create a practice run configuration for a resource, use a command like the following:

```
aws arc-zonal-shift create-practice-run-configuration \
      --resource-
identifier="arn:aws:elasticloadbalancing:Region:111122223333:ExampleALB123456890" \
      --outcome-alarms
 type=CLOUDWATCH,alarmIdentifier=arn:aws:cloudwatch:Region:111122223333:alarm:Region-
MyAppHealthAlarm \
      --blocking-alarms
 type=CLOUDWATCH,alarmIdentifier=arn:aws:cloudwatch:Region:111122223333:alarm:Region-
BlockWhenALARM \
      --blocked-dates 2023-12-01 --blocked-windows Mon:10:00-Mon:10:30
```

```
{
   "arn": "arn:aws:elasticloadbalancing:us-west-2:111122223333:ExampleALB123456890",
   "name": "zonal-shift-elb"
   "zonalAutoshiftStatus": "DISABLED",
   "practiceRunConfiguration": {
       "blockingAlarms": [
                  {
              "type": "CLOUDWATCH",
```

```
                "alarmIdentifier": "arn:aws:cloudwatch:us-west-2:111122223333:alarm:us-
west-2-BlockWhenALARM"
            }
        ]
        "outcomeAlarms": [
            {
                "type": "CLOUDWATCH",
                "alarmIdentifier": "arn:aws:cloudwatch:us-west-2:111122223333:alarm:us-
west-2-MyAppHealthAlarm"
            }
        ],
        "blockedWindows": [
            "Mon:10:00-Mon:10:30"
        ],
        "blockedDates": [
            "2023-12-01"
        ]
}
```

Enable or disable autoshifts

You enable or disable autoshifts for a resource by updating the zonal autoshift status with the CLI. To change the zonal autoshift status, use the `update-zonal-autoshift-configuration` command.

For example, to enable autoshifts for a resource, use a command like the following:

```
aws arc-zonal-shift update-zonal-autoshift-configuration \
      --resource-
identifier="arn:aws:elasticloadbalancing:Region:111122223333:ExampleALB123456890" \
      --zonal-autoshift-status="ENABLED"
```

```
{
   "resourceIdentifier": "arn:aws:elasticloadbalancing:us-
west-2:111122223333:ExampleALB123456890",
   "zonalAutoshiftStatus": "ENABLED"
}
```

Cancel an in-progress autoshift

To cancel an in-progress autoshift for a resource, you disable the zonal autoshift feature. This is the same command that you use to disable zonal autoshift in general, so when you disable zonal

autoshift to cancel an in-progress autoshift, the resource is also not affected by future autoshifts. You can update zonal autoshift to enable it again at any time.

Note that you can disable zonal autoshift for a resource without deleting the practice run configuration for the resource.

To cancel an autoshift with the CLI, disable zonal sutoshift by using the `update-zonal-autoshift-configuration` command. For example, to end an autoshift for a resource, use a command like the following:

```
aws arc-zonal-shift update-zonal-autoshift-configuration \
      --resource-
identifier="arn:aws:elasticloadbalancing:Region:111122223333:ExampleALB123456890" \
      --zonal-autoshift-status="DISABLED"
```

```
{
   "resourceIdentifier": "arn:aws:elasticloadbalancing:us-
west-2:111122223333:ExampleALB123456890",
   "zonalAutoshiftStatus": "DISABLED"
}
```

Cancel an in-progress practice run

You can cancel an in-progress practice run with the CLI by canceling the zonal shift that the practice run started for the resource. To cancel a practice run, use the `cancel-zonal-shift` command.

For example, to cancel a practice run for a resource, use a command like the following:

```
aws arc-zonal-shift cancel-zonal-shift \
   --zonal-shift-id="="arn:aws:testservice::111122223333:ExampleALB123456890"
```

```
{
    "zonalShiftId": "2222222-3333-444-1111",
    "resourceIdentifier": "arn:aws:testservice::111122223333:ExampleALB123456890",
    "awayFrom": "usw2-az1",
    "expiryTime": 2024-11-15T10:35:42+00:00,
    "startTime": 2024-11-15T09:35:42+00:00,
    "status": "CANCELED",
```

```
    "comment": "Practice Run Started"
}
```

Edit a practice run configuration

You can edit a practice run configuration for a resource with the CLI to update different configuration options, such as changing the alarms for practice runs or updating the blocked dates or blocked windows, when ARC won't start practice runs. To edit a practice run configuration, use the `update-practice-run-configuration` command.

Note the following when you edit a practice run configuration for a resource:

- The only supported alarm type at this time is `CLOUDWATCH`.
- You must use alarms that are in the same AWS Region that your resource is deployed in.
- Specifying an outcome alarm is required. Specifying a blocking alarm is optional.
- Specifying blocked dates or blocked windows is optional.
- The blocked dates or blocked windows that you specify replace any existing values.

For example, to edit a practice run configuration for a resource to specify a new blocked date, use a command like the following:

```
aws arc-zonal-shift update-practice-run-configuration \
      --resource-
identifier="arn:aws:elasticloadbalancing:Region:111122223333:ExampleALB123456890" \
      --blocked-dates 2024-03-01
```

```
{
   "arn": "arn:aws:elasticloadbalancing:us-west-2:111122223333:ExampleALB123456890",
   "name": "zonal-shift-elb"
   "zonalAutoshiftStatus": "DISABLED",
   "practiceRunConfiguration": {
       "blockingAlarms": [
                  {
              "type": "CLOUDWATCH",
              "alarmIdentifier": "arn:aws:cloudwatch:us-west-2:111122223333:alarm:us-
west-2-BlockWhenALARM"
           }
       ]
```

```
        "outcomeAlarms": [
            {
                "type": "CLOUDWATCH",
                "alarmIdentifier": "arn:aws:cloudwatch:us-west-2:111122223333:alarm:us-
west-2-MyAppHealthAlarm"
            }
        ],
        "blockedWindows": [
            "Mon:10:00-Mon:10:30"
        ],
        "blockedDates": [
            "2024-03-01"
        ]
}
```

Delete a practice run configuration

You can delete a practice run configuration for a resource, but you must first disable zonal autoshift for the resource. A resource is required to have a practice run configuration to have zonal autoshift enabled. Regular practice runs help you to make sure that your application can run normally without one Availability Zone.

To delete a practice run configuration by using the CLI, first, disable zonal autoshift, if needed by using the `update-zonal-autoshift` command. Then, to delete the practice run configuration, use the `delete-practice-run-configuration` command.

First, disable zonal autoshift for the resource, using a command like the following:

```
aws arc-zonal-shift update-zonal-autoshift-configuration \
      --resource-
identifier="arn:aws:elasticloadbalancing:Region:111122223333:ExampleALB123456890" \
      --zonal-autoshift-status="DISABLED"
```

```
{
   "resourceIdentifier": "arn:aws:elasticloadbalancing:us-
west-2:111122223333:ExampleALB123456890",
   "zonalAutoshiftStatus": "DISABLED"
}
```

Then, delete the practice run configuration, using a command like the following:

```
aws arc-zonal-shift delete-practice-run-configuration \
      --resource-
identifier="arn:aws:elasticloadbalancing:Region:111122223333:ExampleALB123456890"
```

```
{
   "arn": "arn:aws:elasticloadbalancing:us-west-2:111122223333:ExampleALB123456890",
   "name": "TestResource",
   "zonalAutoshiftStatus": "DISABLED"
}
```

Enabling and working with zonal autoshift

This section provides procedures for working with zonal autoshifts in Amazon Application Recovery Controller (ARC), including enabling and disabling zonal autoshift, configuring practice runs, canceling in-progress practice runs, and enabling autoshift observer notifications.

Enabling or disabling zonal autoshift

The steps in this section explain how to enable or disable zonal autoshift on the Amazon Application Recovery Controller (ARC) console. To work with zonal autoshift programmatically, see the Zonal Shift and Zonal Autoshift API Reference Guide.

When zonal autoshift is enabled, you authorize AWS to shift away application resource traffic from an Availability Zone during events, on your behalf, to help reduce your time to recovery.

To enable or disable zonal autoshift

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal autoshift**.
3. Under **Resource zonal autoshift configurations**, choose a resource.
4. In the **Actions** menu, choose **Enable zonal autoshift** or **Disable zonal autoshift**, then follow the steps to complete the update.

If the resource doesn't have a practice run configuration, **Enable zonal autoshift** is not available. To configure a practice run configuration and enable zonal autoshift, choose **Configure zonal autoshift**.

Configuring, editing, or deleting a practice run configuration

The steps in this section explain how to edit or delete a practice run configuration on the Amazon Application Recovery Controller (ARC) console. To work with zonal autoshift programmatically, including changes to practice run configurations, see the Zonal Shift and Zonal Autoshift API Reference Guide.

If you delete a practice run configuration in the console, zonal autoshift is disabled. Before you can delete a practice run configuration with an API operation, you must disable zonal autoshift. You can configure a practice run without enabling zonal autoshift. However, for zonal autoshift to be enabled for a resource, you are required to have a practice run configured for the resource.

To configure a practice run

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal autoshift**.
3. Choose **Configure zonal autoshift**.
4. Choose a resource to configure for zonal autoshift.
5. Choose to disable zonal autoshift if you don't want AWS to start an autoshift for a resource when there's an AWS event. You can continue with the wizard to configure a practice run configuration without enabling autoshifts, if you choose.
6. Choose options for practice runs for the resource. For alarms, you can do the following:

   - (Required) Specify an outcome alarm to monitor practice runs for this resource.
   - (Optional) Specify a blocking alarm for practice runs for this resource.

   For more information, see the **Alarms that you specify for practice runs** section in Best practices when you configure zonal autoshift.
7. Optionally, specify blocked dates and blocked windows. Choose dates or windows (days and times) to block ARC from starting practice runs for this resource. All dates and times are in UTC.
8. Select the check box to confirm that you have read the acknowledgement note.
9. Choose **Create**.

To edit a practice run configuration

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal autoshift**.
3. Under **Resource zonal autoshift configurations**, choose a resource.
4. In the **Actions** menu, choose **Edit practice run configuration**.
5. Make changes to the practice run configuration, to do one or more of the following:
   - For alarms, you can do the following:
     - For the blocking alarm, you can add an alarm, delete the alarm, or specify a different blocking alarm.
     - For the outcome alarm that monitors practice runs, you can specify a different CloudWatch alarm to use. Outcome alarms are required, so you can't delete the outcome alarm.
   - For blocked dates and blocked windows, you can add new dates or days and times, or you can remove or update existing dates or days and times. All dates and times are in UTC.
6. Choose **Save**.

To delete a practice run configuration

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal autoshift**.
3. Under **Resource zonal autoshift configurations**, choose a resource.
4. In the **Actions** menu, choose **Delete practice run configuration**.
5. On the confirmation modal dialog, type `Delete`, and then choose **Delete**.

   Note that deleting a practice run configuration in the console also disables zonal autoshift for the resource. Zonal autoshift requires a practice run to be configured for the resource.

Canceling a practice run zonal shift

The steps in this section explain how to cancel a zonal shift on the Amazon Application Recovery Controller (ARC) console. To work with zonal shift and zonal autoshift programmatically, see the Zonal Shift and Zonal Autoshift API Reference Guide.

You can cancel zonal shifts that you initiate yourself. You can also cancel zonal shifts that AWS starts for a resource for a practice run for zonal autoshift.

To cancel a practice run zonal shift

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Multi-AZ**, choose **Zonal shift**.
3. Select a zonal shift that you want to cancel, and then choose **Cancel zonal shift**.
4. On the confirmation modal dialog, choose **Confirm**.

Enabling or disabling autoshift observer notification

You can configure zonal autoshift to notify you, through Amazon EventBridge, whenever AWS starts an autoshift to shift traffic away from a potentially impaired Availability Zone. You must configure this option in each AWS Region that you want to receive notifications about. You do not have to configure any specific resources with zonal autoshift to enable these separate notifications. For more information, see Using zonal autoshift with Amazon EventBridge.

The steps in this section explain how to enable autoshift observer notification by using the Amazon Application Recovery Controller (ARC) console. To work with zonal autoshift programmatically, see the Zonal Shift and Zonal Autoshift API Reference Guide.

To enable or disable autoshift observer notification

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Under **Getting started**, choose **Enable autoshift observer notification**.
3. In the confirmation dialog box, choose **Enable observer notification**.

Logging and monitoring for zonal autoshift in Amazon Application Recovery Controller (ARC)

You can use AWS CloudTrail and Amazon EventBridge for monitoring zonal autoshift in Amazon Application Recovery Controller (ARC), to analyze patterns and help troubleshoot issues.

Topics

Logging zonal autoshift API calls using AWS CloudTrail

Zonal autoshift for Amazon Route 53 Application Recovery Controller is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Route 53 ARC. CloudTrail captures all API calls for zonal shift as events. The calls captured include calls from the Route 53 ARC console and code calls to the Route 53 ARC API operations for zonal shift.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for zonal shift. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to Route 53 ARC for zonal shift, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Zonal autoshift information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Route 53 ARC for zonal autoshift, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Working with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for zonal autoshift in Route 53 ARC, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs

events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services, to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All Route 53 ARC actions are logged by CloudTrail and are documented in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. For example, calls to the `StartZonalShift` and `ListManagedResources` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Viewing Route 53 ARC events in event history

CloudTrail lets you view recent events in **Event history**. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*.

Understanding zonal autoshift log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `ListManagedResources` action for zonal autoshift.


```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AROA33L3W36EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/admin",
            "accountId": "111122223333",
            "userName": "EXAMPLENAME"
          },
          "webIdFederationData": {},
          "attributes": {
            "creationDate": "2022-11-14T16:01:51Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2022-11-14T16:14:41Z",
      "eventSource": "arc-zonal-shift.amazonaws.com",
      "eventName": "ListManagedResources",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.50",
      "userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64
 exec-env/AWS_Lambda_python3.8 Botocore/1.20.102",
      "requestParameters": null,
      "responseElements": null,
      "requestID": "VGXG4ZUE7UZTVCMTJGIAF_EXAMPLE",
      "eventID": "4b5c42df-1174-46c8-be99-d67_EXAMPLE",
      "readOnly": true,
      "eventType": "AwsApiCall",
      "managementEvent": true,
      "recipientAccountId": "111122223333"
      "eventCategory": "Management"
      }
```

```
    }
```

Using zonal autoshift with Amazon EventBridge

Using Amazon EventBridge, you can set up event-driven rules that monitor your zonal autoshift resources and initiate target actions that use other AWS services. For example, you can set a rule for sending out email notifications by signaling an Amazon SNS topic when a practice run starts for zonal autoshift.

You can create rules in Amazon EventBridge to act on zonal autoshift. An event for zonal autoshift specifies status information about practice runs or autoshifts, for example, when a practice run is started. You can configure zonal autoshift to notify you about zonal autoshift events for resources that you enable for the service.

You can also choose, in addition to or instead of other notifications, to enable autoshift observer notification, which provides a notification event whenever AWS starts an autoshift for a potentially impaired Availability Zone. Autoshift observer notification is separate from notifications that you receive when the traffic for resources that you have enabled for zonal autoshift is shifted away from an Availability Zone. You don't need to configure any resources with zonal autoshift to enable autoshift observer notification. For more information, see Enabling and working with zonal autoshift.

To capture specific zonal autoshift events that you're interested in, define event-specific patterns that EventBridge can use to detect the events. Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

Events are emitted on a best effort basis. They're delivered from ARC to EventBridge in near real-time, under normal operational circumstances. However, situations can arise that might delay or prevent delivery of an event.

For information about how EventBridge rules work with event patterns, see Events and Event Patterns in EventBridge.

Monitor a zonal autoshift resource with EventBridge

With EventBridge, you can create rules that define actions to take when ARC emits events for its resources. For example, you can create a rule that sends an email message when a practice run starts for zonal autoshift.

To type or copy and paste an event pattern into the EventBridge console, select to the option to use **Enter my own** option in the console. To help you determine event patterns that might be useful for you, this topic includes examples of both zonal autoshift event-matching patterns and zonal autoshift events that you can use.

To create a rule for a resource event

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Choose the AWS Region that you want to create the rule in, that is the Region that you're interested in watching events for.
3. Choose **Create rule**.
4. Enter a **Name** for the rule, and, optionally, a description.
5. For **Event bus**, leave the default value, **default**.
6. Choose **Next**.
7. For the **Build event pattern** step, for **Event source**, leave the default value, **AWS events**.
8. Under **Sample event**, choose **Enter my own**.
9. For **Sample events**, type or copy and paste an event pattern.

Example zonal autoshift event patterns

Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

You can copy and paste event patterns from this section into EventBridge to create rules that you can use to monitor zonal autoshift actions and resources.

When you create event patterns for zonal autoshift events, you can specify any of the following for the `detail-type`:

- `Autoshift In Progress`
- `Autoshift Completed`
- `Practice Run Started`
- `Practice Run Succeeded`
- `Practice Run Interrupted`
- `Practice Run Failed`

When a practice run is interrupted, for more information about what caused the interruption, see the `additionalFailureInfo` field.

You can choose to monitor all AWS autoshifts by enabling *autoshift observer notifications*. After you enable autoshift observer notification, to receive the notifications, choose to be notified for the zonal autoshift detail type `Autoshift In Progress`. To see the steps for enabling autoshift observer notification, see Enabling and working with zonal autoshift.

For examples, see the Example zonal autoshift events section.

- *Select all events from zonal autoshift where an autoshift has started.*

  Note the following:

  - If you have autoshift observer notification enabled, ARC returns all autoshift events.
  - If you do not have autoshift observer notification enabled, ARC returns autoshift events only when a resource that you have configured for zonal autoshift is included in an autoshift.

```
{
    "source": [
        "aws.arc-zonal-shift"
    ],
    "detail-type": [
        "Autoshift In Progress"
    ]
}
```

- *Select all events from zonal autoshift where a practice run has started.*

```
{
    "source": [
        "aws.arc-zonal-shift"
    ],
    "detail-type": [
        "Practice Run Started"
    ]
}
```

- *Select all events from zonal autoshift where a practice run has failed.*

```
{
    "source": [
```

```
        "aws.arc-zonal-shift"
    ],
    "detail-type": [
        "Practice Run Failed"
    ]
}
```

Example zonal autoshift events

This section includes example events for *zonal autoshift* actions.

The following is an example event for the `Autoshift In Progress` action, when 1) autoshift observer notification is *enabled* and 2) you have not configured a resource with zonal autoshift that is included in an autoshift:

```
{
    "version": "0",
    "id": "05d4d2d5-9c76-bfea-72d2-d4614802adb4",
    "detail-type": "Autoshift In Progress",
    "source": "aws.arc-zonal-shift",
    "account": "111122223333",
    "time": "2023-11-16T23:38:14Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "version": "0.0.1",
        "data": "",
        "metadata": {
            "awayFrom": "use1-az2",
            "notes":"AWS has started an autoshift for an impaired Availability Zone.
 This notification
            is separate from autoshift notifications for resources, if any, that you
 have configured for
            zonal autoshift. For details, see the Developer Guide."
        }
    }
}
```

The following is an example event for the `Autoshift In Progress` action, when 1) autoshift observer notification is *disabled* and 2) you have configured a resource with zonal autoshift that is included in an autoshift:

```
{
    "version": "0",
    "id": "05d4d2d5-9c76-bfea-72d2-d4614802adb4",
    "detail-type": "Autoshift In Progress",
    "source": "aws.arc-zonal-shift",
    "account": "111122223333",
    "time": "2023-11-16T23:38:14Z",
    "region": "us-east-1",
    "resources": [
        "TEST-EXAMPLE-2023-11-16-23-28-11-5"
    ],
    "detail": {
        "version": "0.0.1",
        "data": "",
        "metadata": {
            "awayFrom": "use1-az2",
            "notes":""
        }
    }
}
```

The following is an example event for the `Practice Run Interrupted` action:

```
{
    "version": "0",
    "id": "05d4d2d5-9c76-bfea-72d2-d4614802adb4",
    "detail-type": "Practice Run Interrupted",
    "source": "aws.arc-zonal-shift",
    "account": "111122223333",
    "time": "2023-11-16T23:38:14Z",
    "region": "us-east-1",
    "resources": [
        "TEST-EXAMPLE-2023-11-16-23-28-11-5"
    ],
    "detail": {
        "version": "0.0.1",
        "data": {
            "additionalFailureInfo": "Practice run interrupted. The blocking alarm
 entered ALARM state."
        },
        "metadata": {
            "awayFrom": "use1-az2"
        }
```

```
    }
}
```

Specify a CloudWatch log group to use as a target

When you create an EventBridge rule, you must specify the target where events that are matched to the rule are sent. For a list of available targets for EventBridge, see Targets available in the EventBridge console. One of the targets that you can add to an EventBridge rule is an Amazon CloudWatch log group. This section describes the requirements for adding CloudWatch log groups as targets, and provides a procedure for adding a log group when you create a rule.

To add a CloudWatch log group as a target, you can do one of the following:

- Create a new log group
- Choose an existing log group

If you specify a new log group using the console when you create a rule, EventBridge automatically creates the log group for you. Make sure that the log group that you use as a target for the EventBridge rule starts with `/aws/events`. If you want to choose an existing log group, be aware that only log groups that start with `/aws/events` appear as options in the drop-down menu. For more information, see Create a new log group in the *Amazon CloudWatch User Guide*.

If you create or use a CloudWatch log group to use as a target using CloudWatch operations outside of the console, make sure that you set permissions correctly. If you use the console to add a log group to an EventBridge rule, then the resource-based policy for the log group is updated automatically. But, if you use the AWS Command Line Interface or an AWS SDK to specify a log group, then you must update resource-based policy for the log group. The following example policy illustrates the permissions that you must define in a resource-based policy for the log group:

```
{
  "Statement": [
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "events.amazonaws.com",
```

```
          "delivery.logs.amazonaws.com"
        ]
      },
      "Resource": "arn:aws:logs:region:account:log-group:/aws/events/*:*",
      "Sid": "TrustEventsToStoreLogEvent"
    }
  ],
  "Version": "2012-10-17"
}
```

You can't configure a resource-based policy for a log group by using the console. To add the required permissions to a resource-based policy, use the CloudWatch PutResourcePolicy API operation. Then, you can use the describe-resource-policies CLI command to check that your policy was applied correctly.

To create a rule for a resource event and specify a CloudWatch log group target

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Choose the AWS Region that you want to create the rule in.
3. Choose **Create rule** and then enter any information about that rule, such as the event pattern or schedule details.

   For more information about creating EventBridge rules for ARC, see the sections earlier in this topic.
4. On the **Select target** page, choose **CloudWatch** as your target.
5. Choose a CloudWatch log group from the drop-down menu.

Identity and Access Management for zonal autoshift

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Route 53 ARC resources. IAM is an AWS service that you can use with no additional charge.

Contents

- AWS managed policies for zonal autoshift in Amazon Application Recovery Controller (ARC)

How zonal autoshift in Amazon Route 53 Application Recovery Controller works with IAM

Before you use IAM to manage access to zonal autoshift in Amazon Application Recovery Controller (ARC), learn what IAM features are available to use with zonal autoshift.

IAM features that you can use with zonal autoshift in Amazon Route 53 Application Recovery Controller

IAM feature	Zonal autoshift support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level, overall view of how AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Route 53 ARC

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

To view examples of Route 53 ARC identity-based policies, see Identity-based policy examples in Amazon Route 53 Application Recovery Controller.

Resource-based policies within Route 53 ARC

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource.

Policy actions for Route 53 ARC

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Route 53 ARC actions for zonal autoshift, see Actions defined by Amazon Route 53 Zonal Shift in the *Service Authorization Reference*.

Policy actions in Route 53 ARC for zonal autoshift use the following prefixes before the action:

```
arc-zonal-shift
```

To specify multiple actions in a single statement, separate them with commas. For example, the following:

```
"Action": [
      "arc-zonal-shift:action1",
      "arc-zonal-shift:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "arc-zonal-shift:Describe*"
```

To view examples of ARC identity-based policies for zonal autoshift, see Identity-based policy examples for zonal autoshift.

Policy resources for zonal autoshift in Route 53 ARC

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of resource types and their ARNs, and the actions that you can specify with the ARN of each resource, see the following topic in the *Service Authorization Reference*:

- Actions defined by Amazon Route 53 - Zonal Shift

To see the actions and resources that you can use with a condition key, see the following topic in the *Service Authorization Reference*:

- Condition keys defined by Amazon Route 53 - Zonal Shift

To view examples of ARC identity-based policies for zonal autoshift, see Identity-based policy examples for zonal autoshift.

Policy condition keys for zonal autoshift in Route 53 ARC

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Route 53 ARC condition keys for zonal autoshift, see the following topics in the *Service Authorization Reference*:

- Condition keys for Amazon Route 53 Zonal Shift

To see the actions and resources that you can use with a condition key, see the following topics in the *Service Authorization Reference*:

- Actions defined by Amazon Route 53 Zonal Shift

To view examples of ARC identity-based policies for zonal autoshift, see Identity-based policy examples for zonal autoshift.

Access control lists (ACLs) in Route 53 ARC

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Route 53 ARC

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Zonal autoshift in ARC includes the following partial support for ABAC:

- Zonal autoshift supports ABAC for managed resources that are registered in ARC for zonal shift. For more information about ABAC for Network Load Balancer and Application Load Balancer

managed resources, see ABAC with Elastic Load Balancing in the Elastic Load Balancing User Guide.

Using temporary credentials with Route 53 ARC

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Route 53 ARC

Supports forward access sessions (FAS): Yes

When you use an IAM entity (user or role) to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions.

To see whether an action requires additional dependent actions in a policy, see the following topic in the *Service Authorization Reference*:

- Amazon Route 53 Zonal Shift

Service roles for Route 53 ARC

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for Route 53 ARC

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Route 53 ARC service-linked roles, see Using the service-linked role for zonal autoshift in Route 53 ARC.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for zonal autoshift

By default, users and roles don't have permission to create or modify Route 53 ARC resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Route 53 ARC, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Route 53 Application Recovery Controller in the *Service Authorization Reference*.

Topics

- Policy best practices
- Example: Zonal autoshift console access
- Examples: Route 53 ARC API actions

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Route 53 ARC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Example: Zonal autoshift console access

To access the Amazon Route 53 Application Recovery Controller console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Route 53 ARC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To perform some tasks, users must have permission to create the service-linked role that is associated with zonal autoshift in ARC. To learn more, see Using the service-linked role for zonal autoshift in Route 53 ARC.

To give users full access to use zonal autoshift in the AWS Management Console, attach a policy like the following to the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                  "arc-zonal-shift:ListManagedResources",
                  "arc-zonal-shift:GetManagedResource",
                  "arc-zonal-shift:ListZonalShifts",
                  "arc-zonal-shift:StartZonalShift",
                  "arc-zonal-shift:UpdateZonalShift",
                  "arc-zonal-shift:CancelZonalShift",
                  "arc-zonal-shift:CreatePracticeRunConfiguration",
                  "arc-zonal-shift:DeletePracticeRunConfiguration",
                  "arc-zonal-shift:ListAutoshifts",
                  "arc-zonal-shift:UpdatePracticeRunConfiguration",
                  "arc-zonal-shift:UpdateZonalAutoshiftConfiguration"
             ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeAvailabilityZones",
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:DescribeAlarms",
            "Resource": "*"
        }
    ]
}
```

Examples: Route 53 ARC API actions

You can use a policy to ensure that a user can use Route 53 ARC API actions for zonal autoshift to configure zonal autoshift so that AWS shifts away application resource traffic from an Availability Zone, on your behalf, to healthy AZs in the AWS Region, to help reduce your time to recovery during events. To provide these permissions, attach a policy that corresponds to the API operations that the user needs to work with, as described below.

To perform some tasks, users must have permissions for the service-linked role that is associated with ARC. Permissions needed to create the service-linked role are included in the following example policy. To learn more, see Using the service-linked role for zonal autoshift in Route 53 ARC.

To work with API operations for zonal autoshift, attach a policy such as the following to the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                   "arc-zonal-shift:ListManagedResources",
                   "arc-zonal-shift:GetManagedResource",
                   "arc-zonal-shift:ListZonalShifts",
                   "arc-zonal-shift:StartZonalShift",
                   "arc-zonal-shift:UpdateZonalShift",
                   "arc-zonal-shift:CancelZonalShift",
                   "arc-zonal-shift:CreatePracticeRunConfiguration",
                   "arc-zonal-shift:DeletePracticeRunConfiguration",
                   "arc-zonal-shift:ListAutoshifts",
                   "arc-zonal-shift:UpdatePracticeRunConfiguration",
                   "arc-zonal-shift:UpdateZonalAutoshiftConfiguration"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect" : "Allow",
            "Action" : [
                    "cloudwatch:DescribeAlarms",
                    "health:DescribeEvents"
            ],
            "Resource" : "*"
        },
        {
            "Effect" : "Allow",
            "Action" : [
                    "arc-zonal-shift:CancelZonalShift",
                    "arc-zonal-shift:GetManagedResource",
                    "arc-zonal-shift:StartZonalShift",
                    "arc-zonal-shift:UpdateZonalShift"
            ],
            "Resource" : "*"
        }
    ]
}
```

Using the service-linked role for zonal autoshift in Route 53 ARC

Zonal autoshift in Amazon Route 53 Application Recovery Controller uses a AWS Identity and Access Management (IAM) service-linked role. A service-linked role is a unique type of IAM role that is linked directly to a service— in this case, Route 53 ARC. The service-linked role is predefined by Route 53 ARC and includes all the permissions that the service requires to call other AWS services on your behalf for specific purposes.

A service-linked role makes setting up Route 53 ARC easier because you don't have to manually add the necessary permissions. Route 53 ARC defines the permissions for the service-linked role, and unless defined otherwise, only Route 53 ARC can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Route 53 ARC zonal autoshift resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services that work with IAM and look for the services that have **Yes** in the **Service-linked role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWSServiceRoleForZonalAutoshiftPracticeRun

Route 53 ARC uses the service-linked role named **AWSServiceRoleForZonalAutoshiftPracticeRun** to do the following:

- Monitor customer-provided Amazon CloudWatch alarms and customer AWS Health Dashboard events for practice runs
- Manage practice runs (practice zonal shifts)

This section describes the permissions for the service-linked role, and information about creating, editing, and deleting the role.

Service-linked role permissions for AWSServiceRoleForZonalAutoshiftPracticeRun

This service-linked role uses the managed policy `AWSZonalAutoshiftPracticeRunSLRPolicy`.

The **AWSServiceRoleForZonalAutoshiftPracticeRun** service-linked role trusts the following service to assume the role:

- `practice-run.arc-zonal-shift.amazonaws.com`

To view the permissions for this policy, see AWSZonalAutoshiftPracticeRunSLRPolicy in the *AWS Managed Policy Reference*.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating the AWSServiceRoleForZonalAutoshiftPracticeRun service-linked role for Route 53 ARC

You don't need to manually create the **AWSServiceRoleForZonalAutoshiftPracticeRun** service-linked role. When you create the first practice run configuration in the AWS Management Console, the AWS CLI, or an AWS SDK, Route 53 ARC creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create the first practice run configuration, Route 53 ARC creates the service-linked role for you again.

Editing the AWSServiceRoleForZonalAutoshiftPracticeRun service-linked role for Route 53 ARC

Route 53 ARC does not allow you to edit the **AWSServiceRoleForZonalAutoshiftPracticeRun** service-linked role. After you create the service-linked role, you cannot change the name of the role because other entities might reference it. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting the AWSServiceRoleForZonalAutoshiftPracticeRun service-linked role for Route 53 ARC

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for a service-linked role before you can manually delete it.

After you have disabled autoshift, then you can delete the **AWSServiceRoleForZonalAutoshiftPracticeRun** service-linked role. For more information about the autoshift capability, see Zonal shift in Amazon Application Recovery Controller (ARC).

> **Note**
>
> If the Route 53 ARC service is using the role when you try to delete the resources, then the service role deletion might fail. If that happens, wait for a few minutes and try the again to delete the role.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForZonalAutoshiftPracticeRun service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Updates to the ARC service-linked role for zonal autoshift

For updates to the AWS managed policies for the Route 53 ARC service-linked roles, see the AWS managed policies updates table for ARC. You can also subscribe to automatic RSS alerts on the Route 53 ARC Document history page.

AWS managed policies for zonal autoshift in Amazon Application Recovery Controller (ARC)

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSZonalAutoshiftPracticeRunSLRPolicy

You can't attach `AWSZonalAutoshiftPracticeRunSLRPolicy` to your IAM entities. This policy is attached to a service-linked role that allows Amazon Application Recovery Controller (ARC) to do the following for zonal autoshift:

- Monitor customer-provided Amazon CloudWatch alarms and customer AWS Health Dashboard events for practice runs
- Manage practice runs (practice zonal shifts)

For more information, see Using the service-linked role for zonal autoshift in Route 53 ARC.

Updates for AWS managed policies for zonal autoshift

For details about updates to AWS managed policies for zonal autoshift in ARC since this service began tracking these changes, see Updates to AWS managed policies for Amazon Application Recovery Controller (ARC). For automatic alerts about changes to this page, subscribe to the RSS feed on the ARC Document history page.

Use routing control to recover multi-Region applications in Amazon Application Recovery Controller (ARC)

This section explains how to use the routing control capability in Amazon Application Recovery Controller (ARC) to minimize disruption and help provide continuity for your users when you have an AWS application deployed in multiple AWS Regions.

You can also learn about readiness check, a capability in ARC that you can use to gain insights into whether your applications and resources are prepared for recovery.

The topics in this section describe the routing control and readiness check capabilities, how to set them up, and how to use them.

Topics

Routing control in Amazon Application Recovery Controller (ARC)

To fail over traffic to application replicas multiple AWS Regions, you can use routing controls in Amazon Application Recovery Controller (ARC) that are integrated with a specific kind of health check in Amazon Route 53. *Routing controls* are simple on-off switches that enable you to switch your client traffic from one Regional replica to another. The traffic rerouting is accomplished by *routing control health checks* that are set up with Amazon Route 53 DNS records. For example, DNS failover records, associated with domain names that front your application replicas in each Region.

This section explains how routing control works, how to set up routing control components, and how to use them to reroute traffic for failover.

The routing control components in ARC are: clusters, control panels, routing controls, and routing control health checks. All routing controls are grouped on control panels. You can group them on the default control panel that ARC creates for your cluster, or create your own custom control panels. You must create a cluster before you can create a control panel or a routing control. Each cluster in ARC is a data plane of endpoints in five AWS Regions.

After you create routing controls and routing control health checks, you can create safety rules for routing control to help prevent unintentional recovery automation side effects. You can update routing control states to reroute traffic, individually or in batches, by using the AWS CLI or API actions (recommended), or by using the AWS Management Console.

This section explains how routing controls work, and how to create and use them to reroute traffic for your application.

> **Important**
>
> To learn about preparing to use ARC to reroute traffic as part of a failover plan for your application in a disaster scenario, see Best practices for routing control in ARC.

About routing control

Routing control redirects traffic by using health checks in Amazon Route 53 that are configured with DNS records associated with the top-level resource of the cells in your recovery group, such as an Elastic Load Balancing load balancer. You can redirect traffic from one cell to another, for example, by updating a routing control state to `Off` (to stop traffic flow to one cell) and updating another routing control state to On (to start traffic flow to another). The process that changes the traffic flow is the Route 53 health check associated with the routing control, after ARC updates it to set it as healthy or unhealthy, based on the corresponding routing control state.

Routing controls support failover across any AWS service that has a DNS endpoint. You can update routing control states to fail over traffic for disaster recovery, or when you detect latency drops for your application, or other issues.

You can also configure safety rules for routing control, to make sure that rerouting traffic by using routing controls doesn't impair availability. For more information, see Creating safety rules for routing control .

It's important to note that routing controls are not themselves health checks that monitor the underlying health of endpoints. For example, unlike a Route 53 health check, a routing control doesn't monitor response times or TCP connection times. A routing control is a simple on-off switch that controls a health check. Typically, you change the state to redirect traffic, and that state change moves the traffic to go to a particular endpoint for an entire application stack, or prevents routing to the whole application stack. For example, in a simple scenario, when you change a

routing control state from `On` to `Off`, it updates a Route 53 health check, which you've associated with a DNS failover record to move the traffic off of an endpoint.

How to use routing control

To update a routing control state, so that you can reroute traffic, you must connect to one of your cluster endpoints in ARC. If the endpoint that you try to connect to is unavailable, try changing the state with another cluster endpoint. Your process for changing routing control states should be prepared to try each endpoint in rotation, since cluster endpoints are cycled through available and unavailable states for regular maintenance and updates.

When you create routing controls, you configure your DNS records to associate routing control health checks with Route 53 DNS names that front each application replica. For example, to control traffic failovers across two load balancers, one in each of two Regions, you create two routing control health checks and associate them with two DNS records, for example, Alias records with failover routing policies, with the domain names of the respective load balancers.

You can also set up more complex traffic failover scenarios by using ARC routing control together with Route 53 health checks and DNS record sets, using DNS records with weighted routing policies. To see a detailed example, see the section on failing over user traffic in the following AWS blog post: Building highly resilient applications using Amazon Application Recovery Controller (ARC), Part 2: Multi-Region stack

When you start a failover for an AWS Region using routing control, because of the steps involved with traffic flow, you might not see traffic move out of the Region immediately. It also can take a short time for existing, in-progress connections in the Region to complete, depending on client behavior and connection reuse. Depending on your DNS settings and other factors, existing connections can complete in just a few minutes, or might take longer. For more information, see Ensuring that traffic shifts finish quickly.

How to use routing control

A routing control in ARC has several benefits over rerouting traffic with traditional health checks. For example:

- A routing control gives you a way to fail over an entire application stack. This is in contrast to failing over individual components of a stack, as Amazon EC2 instances do, based on resource-level health checks.
- A routing control gives you a safe, simple manual override that you can use to shift traffic to do maintenance or to recover from failures when internal monitors don't detect an issue.

- You can use a routing control together with safety rules to prevent common side effects that can happen with fully automated health check-based automation, such as failing over to standby infrastructure that isn't prepared for failover.

Here's an example of incorporating routing controls into your failover strategy, to improve the resilience and availability of your applications in AWS.

You can support highly available AWS applications on AWS by running multiple (typically three) redundant replicas across Regions. Then you can use Amazon Route 53 routing control to route traffic to the appropriate replica.

For example, you can set up one application replica to be active and serve application traffic, while another is a standby replica. When your active replica has failures, you can reroute user traffic there to restore availability to your application. You should decide whether to fail away from or to a replica based on information from your monitoring and health check systems.

If you want to enable faster recoveries, another option that you can choose for your architecture is an active-active implementation. With this approach, your replicas are active at the same time. This means that you can recover from failures by moving users away from an impaired application replica by just rerouting traffic to another active replica.

AWS Region availability for routing control

For detailed information about Regional support and service endpoints for Amazon Application Recovery Controller (ARC), see Amazon Application Recovery Controller (ARC) endpoints and quotas in the *Amazon Web Services General Reference*.

> **Note**
>
> Routing control in Amazon Application Recovery Controller (ARC) is a global feature. However, you must specify the US West (Oregon) Region (specify the parameter `--region us-west-2`) in Regional ARC AWS CLI commands. That is, when you create resources such as clusters, control panels, or routing controls.

A ARC routing control is an on/off switch that changes the state of a ARC health check, which can then be associated with a DNS record that redirects traffic, for example, from a primary to a standby deployment replica.

If there's an application failure or latency issue, you can update routing control states to shift traffic from your primary replica to, for example, a standby replica. By using the highly reliable ARC data plane API operations to make routing control queries and routing control state updates, you can rely on ARC for failover during disaster recovery scenarios. For more information, see Getting and updating routing control states using the ARC API (recommended).

ARC maintains routing control states in a *cluster*, which is a set of five redundant Regional endpoints. ARC propagates routing control state changes across the cluster, which is located in an Amazon EC2 fleet, to get a quorum across five AWS Regions. After propagation, when you query ARC for a routing control state, using the API and the highly-reliable data plane, it returns the consensus view.

You can interact with any one of the five cluster endpoints to update the state of a routing control from, for example, `Off` to On. Then ARC propagates the update across the five Regions of the cluster.

Data consistency across all five cluster endpoints is achieved within 5 seconds on average, and after no more than 15 seconds maximum.

ARC offers extreme reliability with its data plane for you to manually fail over your application across cells. ARC ensures that at least three out of the five cluster endpoints are always accessible to you to perform routing control state changes. Note that each ARC cluster is single-tenant, to ensure that you're not affected by "noisy neighbors" that might slow down your access patterns.

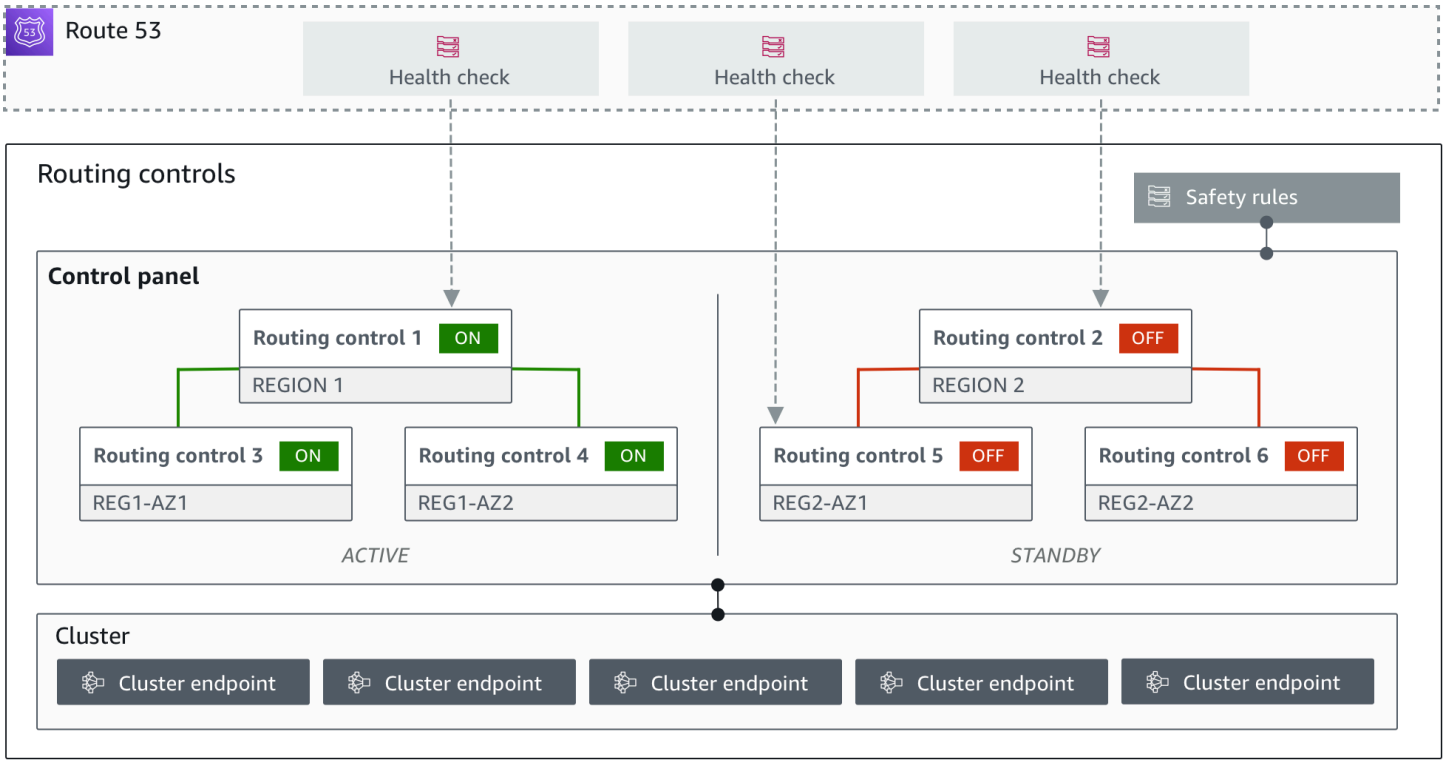
When you make changes to routing control states, you rely on the following three criteria, which are highly unlikely to fail:

- At least three of your five endpoints are available and take part in the quorum.
- You have working IAM credentials and can authenticate against a working Regional cluster endpoint.
- The Route 53 data plane is healthy (this data plane is designed to meet a 100% availability SLA).

Routing control components

The following diagram illustrates an example of components that support the routing control feature in ARC. The routing controls shown here (grouped into one control panel) let you manage traffic to two Availability Zones in each of two Regions. When you update routing control states, ARC changes health checks in Amazon Route 53, which redirect DNS traffic to different cells. Safety

rules that you configure for routing controls help avoid fail-open scenarios and other unintentional consequences.



The following are components of the routing control feature in ARC.

Cluster

A cluster is a set of five redundant Regional endpoints against which you initiate API calls to update or get routing control states. A cluster includes a default control panel, and you can host multiple control panels and routing controls on one cluster.

Routing controls

A routing control is a simple on/off switch, hosted on a cluster, that you use to control routing of client traffic in and out of cells. When you create a routing control, you add a ARC health check in Route 53. This enables you to reroute traffic (using the health checks, configured with DNS records for your applications) when you update the routing control state in ARC.

Routing control health check

Routing controls are integrated with health checks in Route 53. The health checks are associated with DNS records that front each application replica, for example, failover records. When you change routing control states, ARC updates the corresponding health checks, which redirect traffic—for example, to failover to your standby replica.

Control panel

A control panel groups together a set of related routing controls. You can associate multiple routing controls with one control panel, and then create safety rules for the control panel to ensure that the traffic redirection updates that you make are safe. For example, you can configure a routing control for each of your load balancers in each Availability Zone, and then group them in the same control panel. Then you can add a safety rule (an "assertion rule") that makes sure that at least one zone (represented by a routing control) is active at any one time, to avoid unintended "fail-open" scenarios.

Default control panel

When you create a cluster, ARC creates a default control panel. By default, all routing controls that you create on the cluster are added to the default control panel. Or, you can create your own control panels to group related routing controls.

Safety rule

Safety rules are rules that you add to routing control to ensure that recovery actions don't accidentally impair your application's availability. For example, you can create a safety rule that creates a routing control that acts as an overall "on/off" switch so that you can enable or disable a set of other routing controls.

Endpoint (cluster endpoint)

Each cluster in ARC has five Regional endpoints that you can use for setting and retrieving routing control states. Your process for accessing the endpoints should assume that ARC regularly brings the endpoints up and down for maintenance, so you should try each endpoint in succession until you connect to one. You access the endpoints to get the current state of routing controls (On or Off) and to trigger failovers for your applications by changing routing control states.

Data and control planes for routing control

As you plan for failover and disaster recovery, consider how resilient your failover mechanisms are. We recommend that you make sure that the mechanisms that you depend on during failover are highly available, so that you can use them when you need them in a disaster scenario. Typically, you should use data plane functions for your mechanisms whenever you can, for the greatest reliability and fault tolerance. With that in mind, it's important to understand how the functionality of a service is divided between control planes and data planes, and when you can rely on an expectation of extreme reliability with a service's data plane.

As with most AWS services, the functionality for the routing control capability is supported by control planes and data planes. While both of these are built to be reliable, a control plane is optimized for data consistency, while a data plane is optimized for availability. A data plane is designed for resilience so that it can maintain availability even during disruptive events, when a control plane might become unavailable.

In general, a *control plane* enables you to do basic management functions, such as create, update, and delete resources in the service. A *data plane* provides a service's core functionality. Because of this, we recommend that you use data plane operations when availability is important, for example, when you need to reroute traffic to a standby replica during an outage.

For routing control, the control planes and data planes are divided as follows:

- The control plane API for routing control is the Recovery Control Configuration API, supported in the US West (Oregon) Region (us-west-2). You use these API operations or the AWS Management Console to create or delete clusters, control panels, and routing controls, to help prepare for a disaster recovery event when you might need to reroute traffic for your application. *The routing control configuration control plane is not highly available.*
- The routing control data plane is a dedicated cluster across five geographically-isolated AWS Regions. Each customer creates one or more clusters using the routing control control plane. The cluster hosts control panels and routing controls. Then you use the Routing Control (Recovery Cluster) API to get, list, and update routing control states when you want to reroute traffic for your application. *The routing control data plane IS highly available.*

Because the routing control data plane is highly available, we recommend that you plan to use the AWS Command Line Interface to make API calls to work with routing control states when you want to fail over to recover from an event. For more information about key considerations when you prepare for and complete a recovery operation with routing control, see Best practices for routing control in ARC.

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see the Static stability using Availability Zones paper in the Amazon Builders' Library.

Tagging for routing control in Amazon Application Recovery Controller (ARC)

Tags are words or phrases (meta data) that you use to identify and organize your AWS resources. You can add multiple tags to each resource, and each tag includes a key and a value that you define. For example, the key might be environment and the value might be production. You can search and filter your resources based on the tags you add.

You can tag the following resources in routing control in ARC:

- Clusters
- Control panels
- Safety rules

Tagging in ARC is available only through the API, for example, by using the AWS CLI.

The following are examples of tagging in routing control by using the AWS CLI.

```
aws route53-recovery-control-config --region us-west-2 create-cluster --
cluster-name example1-cluster --tags Region=PDX,Stage=Prod
```

```
aws route53-recovery-control-config --region us-west-2 create-control-panel
--control-panel-name example1-control-panel --cluster-arn arn:aws:route53-
recovery-control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh
--tags Region=PDX,Stage=Prod
```

For more information, see TagResource in the *Recovery Control Configuration API Reference Guide* for Amazon Application Recovery Controller (ARC).

Pricing for routing control in ARC

With Amazon Application Recovery Controller (ARC), you only pay for what you configure to use in the service. For routing control in ARC, you pay an hourly cost per cluster that you create. Each cluster can host multiple routing controls, which you use to trigger application failovers.

To help manage costs and improve efficiency, you can set up cross-account sharing for a cluster, to share one cluster with multiple AWS accounts. For more information, see Support cross-account for clusters in ARC.

For detailed pricing information for ARC and pricing examples, see Amazon Application Recovery Controller (ARC) Pricing and scroll down to Amazon Application Recovery Controller (ARC).

Getting started with multi-Region recovery in Amazon Application Recovery Controller (ARC)

To fail over your applications by using routing control in Amazon Application Recovery Controller (ARC), you must have AWS applications that are in multiple AWS Regions. To get started, first, make sure that your applications are set up in siloed replicas in each Region, so that you can fail over from one to another during an event. Then, you can create routing controls to reroute the application traffic to fail over from a primary application to a secondary, maintaining continuity for your users.

> **Note**
>
> If you have an application that is siloed by Availability Zones, consider using zonal shift or zonal autoshift for failover recovery. No setup is required to use zonal shift or zonal autoshift to reliably recover applications from Availability Zone impairments. For more information, see Use zonal shift and zonal autoshift to recover applications in Amazon Application Recovery Controller (ARC).

So that you can use ARC routing control to recover applications during an event, we recommend that you set up at least two applications that are replicas of each other. Each replica, or *cell*, represents an AWS Region. After you've set up your application resources to align with Regions, make sure that your application set up for successful recovery by taking the following steps.

Tip: To help simplify setup, we provide AWS CloudFormation and HashiCorp Terraform templates that create an application with redundant replicas that fail independently of one another. To learn more and download the templates, see Setting up an example app.

To prepare to use routing control, make sure that your application is set up to be resilient by doing the following:

1. Build independent copies of your application stack (networking and compute layer) that are replicas of each other in each Region so that you can fail over traffic from one to the other when there's an event. Make sure that you don't have any cross-Region dependencies in your application code that would cause the failure of one replica to impact the other. To successfully fail over between AWS Regions, your stack boundaries should be within a Region.

2. Duplicate all the required stateful data for your application across the replicas. You can use AWS database services to help replicate your data.

Get started with routing control for traffic failover

Routing control in Amazon Application Recovery Controller (ARC) enables you to trigger failover for your traffic to fail over between redundant application copies, or replicas, that are running in separate AWS Regions. Failover is performed with DNS, using the Amazon Route 53 data plane.

After you set up your replicas in each Region, as described in the next section, you can associate each one with a routing control. First, you associate routing controls with the top-level domain names of your replicas in each Region. Then, you add a routing control health check to the routing control so that it can turn traffic flow on and off. This enables you to control traffic routing across replicas of your application.

You can update routing control states in the AWS Management Console to fail over traffic, but we recommend that instead you use ARC actions, using the API or AWS CLI, to change them. API actions aren't dependent on the console, so they're more resilient.

For example, to fail over between Regions, from us-west-1 to us-east-1, you can use the `update-routing-control-state` API action to set the state of `us-west-1` to `Off` and `us-east-1` to `On`.

Before you create routing control components to set up failover for your application, make sure that your application is siloed into Regional replicas, so that you can fail over from one to the other. To learn more and get started siloing a new application or creating a example stack, see the next sections.

Setting up an example app

To help you understand how routing control works, we provide an example application called `TicTacToe`. The example uses AWS CloudFormation templates to simplify the process, as well as downloadable AWS CloudFormation and HashiCorp Terraform templates with a sample app so that you can quickly explore setting up and using ARC yourself.

After you deploy the sample app, you can use the templates to create ARC components, and then explore using routing controls to manage traffic flow to the app. You can adapt the templates and process for your own scenario and applications.

- **AWS CloudFormation:** To get started with a sample application and AWS CloudFormation templates, see the README instructions here on this Amazon S3 bucket. You can learn more about using AWS CloudFormation templates by reading AWS CloudFormation concepts in the AWS CloudFormation User Guide.
- **HashiCorp Terraform:** To get started with a sample application and Terraform templates, see the README instructions here on this Amazon S3 bucket. You can learn more about using Terraform templates by reading the HashiCorp documentation.

Best practices for routing control in ARC

We recommend the following best practices for recovery and failover preparedness for routing control in Amazon Application Recovery Controller (ARC).

Topics

Keep purpose-built, long-lived AWS credentials secure and always accessible

In a disaster recovery (DR) scenario, keep system dependencies to a minimum by using a simple approach to accessing AWS and performing recovery tasks. Create IAM long-lived credentials specifically for DR tasks, and keep the credentials securely in an on-premises physical safe or a virtual vault, to access when needed. With IAM, you can centrally manage security credentials, such as access keys, and permissions for access to AWS resources. For non-DR tasks, we recommend that you continue to use federated access, using AWS services such as AWS Single Sign-On.

To perform failover tasks in ARC with the recovery cluster data plane API, you can attach a ARC IAM policy to your user. To learn more, see Identity-based policy examples in Amazon Route 53 Application Recovery Controller.

Choose lower TTL values for DNS records involved in failover

For DNS records that you might need to change as part of your failover mechanism, especially records that are health checked, using lower TTL values is appropriate. Setting a TTL of 60 or 120 seconds is a common choice for this scenario.

The DNS TTL (time to live) setting tells DNS resolvers how long to cache a record before requesting a new one. When you choose a TTL, you make a trade-off between latency and reliability, and responsiveness to change. With a shorter TTL on a record, DNS resolvers notice updates to the record more quickly because the TTL specifies that they must query more frequently.

For more information, see *Choosing TTL values for DNS records* in Best practices for Amazon Route 53 DNS.

Limit the time that clients stay connected to your endpoints

When you use routing controls to shift from one AWS Region to another, the mechanism that Amazon Application Recovery Controller (ARC) uses to move your application traffic is a DNS update. This update causes all new connections to be directed away from the impaired location.

However, clients with pre-existing open connections might continue to make requests against the impaired location until the clients reconnect. To ensure a quick recovery, we recommend that you limit the amount of time clients stay connected to your endpoints.

If you use an Application Load Balancer, you can use the `keepalive` option to configure how long connections continue. For more information, see HTTP client keepalive duration in the Application Load Balancer User Guide.

By default, Application Load Balancers set the HTTP client keepalive duration value to 3600 seconds, or 1 hour. We suggest that you lower the value to be inline with your recovery time goal for your application, for example, 300 seconds. When you choose an HTTP client keepalive duration time, consider that this value is a trade off between reconnecting more frequently in general, which can affect latency, and more quickly moving all clients away from an impaired AZ or Region.

Bookmark or hard code your five Regional cluster endpoints and routing control ARNs

We recommend that you keep a local copy of your ARC Regional cluster endpoints, in bookmarks or saved in automation code that you use to retry your endpoints. During a failure event, you might not be able to access some API operations, including ARC API operations that

are not hosted on the extremely reliable data plane cluster. You can list the endpoints for your ARC clusters by using the DescribeCluster API operation.

Choose one of your endpoints at random to update your routing control states

We recommend that when you need to fail over, you update (and retrieve) routing control states using a random endpoint from your five Regional cluster endpoints. If that endpoint fails, then retry each of your other Regional endpoints. For information about using code examples with the AWS SDK, including examples for trying cluster endpoints, see Code examples for Application Recovery Controller using AWS SDKs.

Use the extremely reliable data plane API to list and update routing control states, not the console

Using the ARC data plane API, view your routing controls and states with the ListRoutingControls operation and update routing control states to redirect traffic for failover with the UpdateRoutingControlState operation. You can use the AWS CLI (as in these examples) or code that you write using one of the AWS SDKs. ARC offers extreme reliability with the API in the data plane to fail over traffic. We recommend using the API instead of changing routing control states in the AWS Management Console.

Connect to one of your Regional cluster endpoints for ARC to use the data plane API. If the endpoint is unavailable, try connecting to another cluster endpoint.

If a safety rule blocks a routing control state update, you can bypass it to make the update and fail over traffic. For more information, see Overriding safety rules to reroute traffic.

Test failover with ARC

Test failover regularly with ARC routing control, to fail over from your primary application stack to a secondary application stack. It's important to make sure that the ARC structures that you've added are aligned with the correct resources in your stack, and that everything works as you expect it to. You should test this after you set up ARC for your environment, and continue to test periodically, so that your failover environment is prepared, before you experience a failure situation in which you need your secondary system to be up and running quickly to avoid downtime for your users.

Routing control API operations

This section includes tables with lists API operations that you can use for setting up and using routing control in Amazon Application Recovery Controller (ARC), with links to relevant documentation.

For examples of how to use common routing control configuration API operations with the AWS Command Line Interface, see Examples of using ARC routing control API operations with the AWS CLI.

The following table lists ARC API operations that you can use for routing control configuration, with links to relevant documentation.

Action	Using the ARC console	Using the ARC API
Create a cluster	See Creating routing control components in ARC	See CreateCluster
Describe a cluster	See Creating routing control components in ARC	See DescribeCluster
Delete a cluster	See Creating routing control components in ARC	See DeleteCluster
List clusters for an account	See Creating routing control components in ARC	See ListClusters
Create a routing control	See Creating routing control components in ARC	See CreateRoutingControl
Describe a routing control	See Creating routing control components in ARC	See DescribeRoutingControl
Update a routing control	See Creating routing control components in ARC	See UpdateRoutingControl
Delete a routing control	See Creating routing control components in ARC	See DeleteRoutingControl

Action	Using the ARC console	Using the ARC API
List routing controls	See Creating routing control components in ARC	See ListRoutingControls
Create a control panel	See Creating routing control components in ARC	See CreateControlPanel
Describe a control panel	See Creating routing control components in ARC	See DescribeControlPanel
Update a control panel	See Creating routing control components in ARC	See UpdateControlPanel
Delete a control panel	See Creating routing control components in ARC	See DeleteControlPanel
List control panels	See Creating routing control components in ARC	See ListControlPanels
Create a safety rule	See Creating safety rules for routing control	See CreateSafetyRule
Describe a safety rule	See Creating safety rules for routing control	See DescribeSafetyRule
Update a safety rule	See Creating safety rules for routing control	See UpdateSafetyRule
Delete a safety rule	See Creating safety rules for routing control	See DeleteSafetyRule
List safety rules	See Creating safety rules for routing control	See ListSafetyRules
List associated Route 53 health checks	See Creating a routing control health check in ARC	See ListAssociatedRoute53HealthChecks
List the AWS RAM resource policies for cluster sharing	See Support cross-account for clusters in ARC	See GetResourcePolicy

The following table lists common ARC API operations that you can use for managing traffic failover with the routing control data plane, with links to relevant documentation.

Action	Using the ARC console	Using the ARC API
Get a routing control state	See Getting and updating routing control states in the AWS Management Console	See GetRoutingControlState
List routing controls	N/A	See ListRoutingControls
Update a routing control state	See Getting and updating routing control states in the AWS Management Console	See UpdateRoutingControlState
Update multiple routing control states	See Getting and updating routing control states in the AWS Management Console	See UpdateRoutingControlStates

Using this service with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples

SDK documentation	Code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

For examples specific to this service, see Code examples for Application Recovery Controller using AWS SDKs.

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Examples of using ARC routing control API operations with the AWS CLI

This section walks through simple application examples of working with routing control, using the AWS Command Line Interface to work with the routing control capability in Amazon Application Recovery Controller (ARC) using API operations. The examples are intended to help you develop a basic understanding of how to work with routing control using the CLI.

With routing control in Amazon Application Recovery Controller (ARC), you can trigger traffic failovers between redundant application copies, or replicas, that are running in separate AWS Regions or Availability Zones.

You organize routing controls into groups called control panels that are provisioned on a cluster. A ARC cluster is a Regional set of endpoints that is globally deployed. Cluster endpoints provide

a highly available API that you can use to set and retrieve routing control states. For more information about the components of the routing control feature, see Routing control components .

> **Note**
>
> ARC is a global service that supports endpoints in multiple AWS Regions. However, you must specify the US West (Oregon) Region—that is, specify the parameter `--region us-west-2`— in most ARC CLI commands. For example, use the `region` parameter when you create recovery groups, control panels, and clusters.
>
> When you create a cluster, ARC provides you with a set of Regional endpoints. To get or update routing control states, you must specify the Regional endpoint (the AWS Region and the endpoint URL) in your CLI command.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For a list of routing control API actions, see Routing control API operations and Routing control API operations.

We'll start by creating the components you need to manage failover by using routing controls, beginning with creating a cluster.

Set up routing control components

Our first step is to create a cluster. A ARC cluster is a set of five endpoints, one in each of five different AWS Regions. The ARC infrastructure supports these endpoints to work in coordination so that they guarantee high availability and sequential consistency of failover operations.

1. Create a cluster

1a. Create a cluster.

```
aws route53-recovery-control-config --region us-west-2 create-cluster --cluster-name
 NewCluster
```

```
{
    "Cluster": {
        "ClusterArn": "arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh",
        "Name": "NewCluster",
```

```
        "Status": "PENDING"
    }
}
```

When you first create a ARC resource, it has a status of `PENDING` while the cluster is created. You can check in on its progress by calling `describe-cluster`.

1b. Describe a cluster.

```
aws route53-recovery-control-config --region us-west-2 \
    describe-cluster --cluster-arn arn:aws:route53-recovery-
control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh
```

```
{
    "Cluster":{
        "ClusterArn": "arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh",
        "ClusterEndpoints":[
            {"Endpoint": "https://host-aaaaaa.us-east-1.example.com", "Region":"us-
east-1"},
            {"Endpoint": "https://host-bbbbbb.ap-southeast-2.example.com",
 "Region":"ap-southeast-2"},
            {"Endpoint": "https://host-cccccc.eu-west-1.example.com", "Region":"eu-
west-1"},
            {"Endpoint": "https://host-dddddd.us-west-2.example.com", "Region":"us-
west-2"},
            {"Endpoint": "https://host-eeeeee.ap-northeast-1.example.com",
 "Region":"ap-northeast-1"}
        ]
        "Name": "NewCluster",
        "Status": "DEPLOYED"
    }
}
```

When the status is DEPLOYED, ARC has successfully created the cluster with the set of endpoints for you to interact with. You can list all of your clusters by calling `list-clusters`.

1c. List your clusters.

```
aws route53-recovery-control-config --region us-west-2 list-clusters
```

```
{
```

```
    "Clusters": [
        {
            "ClusterArn": "arn:aws:route53-recovery-
control::111122223333:cluster/1234abcd-abcd-1234-abcd-1234abcdefgh",
            "ClusterEndpoints":[
                {"Endpoint": "https://host-aaaaaa.us-east-1.example.com", "Region":"us-
east-1"},
                {"Endpoint": "https://host-bbbbbb.ap-southeast-2.example.com",
 "Region":"ap-southeast-2"},
                {"Endpoint": "https://host-cccccc.eu-west-1.example.com", "Region":"eu-
west-1"},
                {"Endpoint": "https://host-dddddd.us-west-2.example.com", "Region":"us-
west-2"},
                {"Endpoint": "https://host-eeeeee.ap-northeast-1.example.com",
 "Region":"ap-northeast-1"}
            ],
            "Name": "AnotherCluster",
            "Status": "DEPLOYED"
        },
        {
            "ClusterArn": "arn:aws:route53-recovery-
control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh",
            "ClusterEndpoints": [
                {"Endpoint": "https://host-ffffff.us-east-1.example.com", "Region":"us-
east-1"},
                {"Endpoint": "https://host-gggggg.ap-southeast-2.example.com",
 "Region":"ap-southeast-2"},
                {"Endpoint": "https://host-hhhhhh.eu-west-1.example.com", "Region":"eu-
west-1"},
                {"Endpoint": "https://host-iiiiii.us-west-2.example.com", "Region":"us-
west-2"},
                {"Endpoint": "https://host-jjjjjj.ap-northeast-1.example.com",
 "Region":"ap-northeast-1"}
            ],
            "Name": "NewCluster",
            "Status": "DEPLOYED"
        }
    ]
}
```

2. Create a control panel

A control panel is a logical grouping for organizing your ARC routing controls. When you create a cluster, ARC automatically provides a control panel for you called `DefaultControlPanel`. You can use this control panel right away.

A control panel can only exist in one cluster. If you want to move a control panel to another cluster, you must delete it and then create it in the second cluster. You can see all of the control panels in your account by calling `list-control-panels`. To see just the control panels in a specific cluster, add the `--cluster-arn` field.

2a. List control panels.

```
aws route53-recovery-control-config --region us-west-2 \
    list-control-panels --cluster-arn arn:aws:route53-recovery-
control::111122223333:cluster/eba23304-1a51-4674-ae32-b4cf06070bdd
```

```
{
    "ControlPanels": [
        {
            "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/1234567dddddd1234567dddddd1234567",
            "ClusterArn": "arn:aws:route53-recovery-
control::111122223333:cluster/5678abcd-abcd-5678-abcd-5678abcdefgh",
            "DefaultControlPanel": true,
            "Name": "DefaultControlPanel",
            "RoutingControlCount": 0,
            "Status": "DEPLOYED"
        }
    ]
}
```

Optionally, create your own control panel by calling `create-control-panel`.

2b. Create a control panel.

```
aws route53-recovery-control-config --region us-west-2 create-control-panel \
        --control-panel-name NewControlPanel2 \
        --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh
```

```
{
    "ControlPanel": {
        "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
        "ClusterArn": "arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh",
        "DefaultControlPanel": false,
        "Name": "NewControlPanel2",
        "RoutingControlCount": 0,
        "Status": "PENDING"
    }
}
```

When you first create a ARC resource, it has a status of `PENDING` while it's being created. You can check on progress by calling `describe-control-panel`.

2c. Describe a control panel.

```
aws route53-recovery-control-config --region us-west-2 describe-control-panel \
    --control-panel-arn arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456
```

```
{
    "ControlPanel": {
        "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
        "ClusterArn": "arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh",
        "DefaultControlPanel": true,
        "Name": "DefaultControlPanel",
        "RoutingControlCount": 0,
        "Status": "DEPLOYED"
    }
}
```

3. Create a routing control

Now that you've set up the cluster and looked at control panels, you can begin creating routing controls. When you create a routing control, you must at least specify the Amazon Resource Name (ARN) of the cluster that you want the routing control to be in. You can also specify the ARN of a

control panel for the routing control. You'll also need to specify the cluster where the control panel is located.

If you don't specify a control panel, your routing control is added to the automatically created control panel, `DefaultControlPanel`.

Create a routing control by calling `create-routing-control`.

3a. Create a routing control.

```
aws route53-recovery-control-config --region us-west-2 create-routing-control \
    --routing-control-name NewRc1 \
    --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh
```

```
{
    "RoutingControl": {
        "ControlPanelArn": " arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
        "Name": "NewRc1",
        "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567",
        "Status": "PENDING"
    }
}
```

Routing controls follow the same creation pattern as other ARC resources, so you can track their progress by calling a describe operation.

3b. Describe routing control.

```
aws route53-recovery-control-config --region us-west-2 describe-routing-control \
       --routing-control-arn arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567
```

```
{
    "RoutingControl": {
        "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
        "Name": "NewRc1",
```

```
        "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567",
        "Status": "DEPLOYED"
    }
}
```

You can list the routing controls in a control panel by calling `list-routing-controls`. The control panel ARN is required.

3c. List routing controls.

```
aws route53-recovery-control-config --region us-west-2 list-routing-controls \
        --control-panel-arn arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456
```

```
{
    "RoutingControls": [
        {
            "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
            "Name": "Rc1",
            "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567",
            "Status": "DEPLOYED"
        },
        {
            "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
            "Name": "Rc2",
            "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
hijklmnop987654321",
            "Status": "DEPLOYED"
        }
    ]
}
```

In the following example, where we work with routing control states, we assume that you have the two routing controls listed in this section (Rc1 and Rc2). In this example, each routing control represents an Availability Zone that your application is deployed in.

4. Create safety rules

When you work with several routing controls at the same time, you might decide that you want some safeguards in place when you enable and disable them, to avoid unintentional consequences, like turning both routing controls off and stopping all traffic flow. To create these safeguards, you create routing control safety rules.

There are two types of safety rules: assertion rules and gating rules. To learn more about safety rules, see Creating safety rules for routing control .

The following call provides an example of creating an assertion rule that makes sure that at least one of two routing controls is set to On at any given time. To create the rule, you run `create-safety-rule` with the `assertion-rule` parameter.

For detailed information about the assertion rule API operation, see AssertionRule in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

4a. Create an assertion rule.

```
aws route53-recovery-control-config --region us-west-2 create-safety-rule \
        --assertion-rule '{"Name": "TestAssertionRule",
        "ControlPanelArn": "arn:aws:route53-recovery-
control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx",
        "WaitPeriodMs": 5000,
        "AssertedControls":
        ["arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/def123def123def"
        "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/ghi456ghi456ghi"],
        "RuleConfig": {"Threshold": 1, "Type": "ATLEAST", "Inverted": false}}'
```

```
{
    "Rule": {
        "ASSERTION": {
            "Arn": "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/safetyrule/333333444444",
            "AssertedControls": [
                "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/def123def123def"
                "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/ghi456ghi456ghi"],
```

```
            "ControlPanelArn": "arn:aws:route53-recovery-
control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx",
            "Name": "TestAssertionRule",
            "RuleConfig": {
                "Inverted": false,
                "Threshold": 1,
                "Type": "ATLEAST"
            },
            "Status": "PENDING",
            "WaitPeriodMs": 5000
        }
    }
}
```

The following call provides an example of creating a gating rule that provides an overall "on/off" or "gating" switch for a set of target routing controls in a control panel. This lets you disallow updating the target routing controls so that, for example, automation can't make unauthorized updates. In this example, the gating switch is a routing control specified by the `GatingControls` parameter and the two routing controls that are controlled or "gated" are specified by the `TargetControls` parameter.

> **Note**
>
> Before you create the gating rule, you must create the gating routing control, which does not include DNS failover records, and the target routing controls, which you do configure with DNS failover records.

To create the rule, you run `create-safety-rule` with the `gating-rule` parameter.

For detailed information about the assertion rule API operation, see GatingRule in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

4b. Create a gating rule.

```
aws route53-recovery-control-config --region us-west-2 create-safety-rule \
        --gating-rule '{"Name": "TestGatingRule",
        "ControlPanelArn": "arn:aws:route53-recovery-
control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx",
        "WaitPeriodMs": 5000,
```

```
        "GatingControls": ["arn:aws:route53-recovery-
control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/
def123def123def"]
        "TargetControls": ["arn:aws:route53-recovery-
control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/
ghi456ghi456ghi",
        "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/lmn789lmn789lmn"],
        "RuleConfig": {"Threshold": 0, "Type": "OR", "Inverted": false}}'
```

```
{
    "Rule": {
        "GATING": {
            "Arn": "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/safetyrule/444444444444",
            "GatingControls": [
                "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/def123def123def"
            ],
            "TargetControls": [
                "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/ghi456ghi456ghi"
                "arn:aws:route53-recovery-control::888888888888:controlpanel/
zzz123yyy456xxx789zzz123yyy456xxx/routingcontrol/lmn789lmn789lmn"
            ],
            "ControlPanelArn": "arn:aws:route53-recovery-
control::888888888888:controlpanel/zzz123yyy456xxx789zzz123yyy456xxx",
            "Name": "TestGatingRule",
            "RuleConfig": {
                "Inverted": false,
                "Threshold": 0,
                "Type": "OR"
            },
            "Status": "PENDING",
            "WaitPeriodMs": 5000
        }
    }
}
```

As with other routing control resources, you can describe, list, or delete safety rules after they propagate to the data plane.

After you set up one or more safety rules, you can continue to interact with the cluster, to set, or retrieve state for routing controls. If a `set-routing-control-state` operation breaks a rule that you created, you'll receive an exception similar to the following:

```
Cannot modify control state for [0123456bbbbbbb0123456bbbbbb01234560123
abcdefg1234567] due to failed rule evaluation
0123456bbbbbbb0123456bbbbbb0123456333333444444
```

The first identifier is the control panel ARN concatenated with the routing control ARN. The second identifier is the control panel ARN concatenated with the safety rule ARN.

5. Create health checks

To use routing controls to fail over traffic, you create health checks in Amazon Route 53, and then associate the health checks with your DNS records. To fail over traffic, a ARC routing control sets the health check to fail, so that Route 53 reroutes the traffic. (The health check doesn't valid the health of your application; it's simply used as a method for rerouting traffic.)

As an example, let's say you have two cells (Regions or Availability Zones). You configure one as the primary cell for your application, and the other as the secondary, to fail over to.

To set up health checks for failover, you can do the following, for example:

1. Use the ARC CLI to create a routing control for each cell.
2. Use the Route 53 CLI to create a ARC health check in Route 53 for each routing control.
3. Use the Route 53 CLI to create two failover DNS records in Route 53, and associate a health check with each one.

5a. Create a routing control for each cell.

```
aws route53-recovery-control-config --region us-west-2 create-routing-control \
        --routing-control-name RoutingControlCell1 \
        --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh
```

```
aws route53-recovery-control-config --region us-west-2 create-routing-control \
        --routing-control-name RoutingControlCell2 \
        --cluster-arn arn:aws:route53-recovery-control::111122223333:cluster/5678abcd-
abcd-5678-abcd-5678abcdefgh
```

5b. Create a health check for each routing control.

> **Note**
>
> You create ARC health checks by using the Amazon Route 53 CLI.

```
aws route53 create-health-check --caller-reference RoutingControlCell1 \
        --health-check-config \
        Type=RECOVERY_CONTROL,RoutingControlArn=arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567
```

```
{
    "Location": "https://route53.amazonaws.com/2015-01-01/healthcheck/11111aaaa-bbbb-
cccc-dddd-ffffff22222",
    "HealthCheck": {
        "Id": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
        "CallerReference": "RoutingControlCell1",
        "HealthCheckConfig": {
            "Type": "RECOVERY_CONTROL",
            "Inverted": false,
            "Disabled": false,
            "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567"
        },
        "HealthCheckVersion": 1
    }
}
```

```
aws route53 create-health-check --caller-reference RoutingControlCell2 \
    --health-check-config \
    Type=RECOVERY_CONTROL,RoutingControlArn=arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567
```

```
{
    "Location": "https://route53.amazonaws.com/2015-01-01/healthcheck/11111aaaa-bbbb-
cccc-dddd-ffffff22222",
    "HealthCheck": {
```

```
        "Id": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
        "CallerReference": "RoutingControlCell2",
        "HealthCheckConfig": {
            "Type": "RECOVERY_CONTROL",
            "Inverted": false,
            "Disabled": false,
            "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567"
        },
        "HealthCheckVersion": 1
    }
}
```

5c. Create two failover DNS records, and associate a health check with each one.

You create failover DNS records in Route 53 using the Route 53 CLI. To create the records, follow the directions in the Amazon Route 53 AWS CLI Command Reference for the change-resource-record-sets command. In the records, specify the DNS value for each cell together with the corresponding `HealthCheckID` value that Route 53 created for the health check (see 6b).

For the primary cell:

```
{
    "Name": "myapp.yourdomain.com",
    "Type": "CNAME",
    "SetIdentifier": "primary",
    "Failover": "PRIMARY",
    "TTL": 0,
    "ResourceRecords": [
        {
            "Value": "cell1.yourdomain.com"
        }
    ],
    "HealthCheckId": "xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```

For the secondary cell:

```
{
    "Name": "myapp.yourdomain.com",
    "Type": "CNAME",
```

```
    "SetIdentifier": "secondary",
    "Failover": "SECONDARY",
    "TTL": 0,
    "ResourceRecords": [
        {
            "Value": "cell2.yourdomain.com"
        }
    ],
    "HealthCheckId": "yyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy"
}
```

Now, to fail over from your primary cell to your secondary cell, you can follow the CLI example in step 4b to update the state of `RoutingControlCell1` to `OFF` and `RoutingControlCell2` to `ON`.

List and update routing controls and states with the AWS CLI

After you create your Amazon Application Recovery Controller (ARC) resources, such as cluster, routing controls, and control panels, you can interact with the cluster to list and update routing control states for failover.

For each cluster that you create, ARC provides you with a set of cluster endpoints, one in each of five AWS Regions. You must specify one of these Regional endpoints (the AWS Region and the endpoint URL) when you make calls to the cluster to retrieve or set routing control states to `On` or `Off`. When you use the AWS CLI, to get or update routing control states, in addition to the Regional endpoint, you must also specify the `--region` of the Regional endpoint, as shown in the examples in this section.

You can use any of the Regional cluster endpoints. We recommend that your systems rotate through the regional endpoints, and be prepared to retry with each of the available endpoints. For code samples that illustrate trying cluster endpoints in sequence, see Actions for Application Recovery Controller using AWS SDKs .

For more information about using the AWS CLI, see the AWS CLI Command Reference. For a list of routing control API actions and links to more information, see Routing control API operations.

> **Important**
>
> Although you can update a routing control state on the Amazon Route 53 console, we recommend that you update routing control states by using the AWS CLI or an AWS SDK. ARC offers extreme reliability with the ARC routing control data plane for rerouting traffic

and failing over across cells. For more recommendations about using ARC for failover, see Best practices for routing control in ARC.

When you create a routing control, the state is set to `Off`. This means that traffic is not routed to the target cell for that routing control. You can verify the state of the routing control by running the command `get-routing-control-state`.

To determine the Region and the endpoint to specify, run the `describe-clusters` command to view the `ClusterEndpoints`. Each `ClusterEndpoint` includes a Region and corresponding endpoint that you can use to get or update routing control states. *DescribeCluster is a recovery control configuration API operation. We recommend that you keep a local copy of your ARC Regional cluster endpoints, in bookmarks or hardcoded in automation code that you use to retry your endpoints.*

1. List routing controls

You can view your routing controls and routing control states using the highly reliable ARC data plane endpoints.

1. List routing controls for a specific control panel. If you don't specify a control panel, `list-routing-controls` returns all the routing controls in the cluster.

```
aws route53-recovery-cluster list-routing-controls --control-panel-arn \
        arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456 \
        --region us-west-2 \
        --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

```
{
    "RoutingControls": [{
        "ControlPanelArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
        "ControlPanelName": "ExampleControlPanel",
        "RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567",
        "RoutingControlName": "RCOne",
        "RoutingControlState": "On"
    },
    {
```

```
        "ControlPanelArn": "arn:aws:route53-recovery-
control::023759465626:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
        "ControlPanelName": "ExampleControlPanel",
        "RoutingControlArn": "arn:aws:route53-recovery-
control::023759465626:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
zzzzxxxxyyyy123456",
        "RoutingControlName": "RCTwo",
        "RoutingControlState": "Off"
    }
]
```

2. Get routing controls

2. Get a routing control state.

```
aws route53-recovery-cluster get-routing-control-state --routing-control-arn \
        arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567 \
        --region us-west-2 \
        --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

```
{"RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567",
    "RoutingControlName": "RCOne",
    "RoutingControlState": "On"
}
```

2. Update routing controls

To route traffic to the target endpoint controlled by the routing control, you update the routing control state to On. Update the routing control state by running the command `update-routing-control-state`. (When the request is successful, the response is empty.)

2a. Update a routing control state.

```
aws route53-recovery-cluster update-routing-control-state \
        --routing-control-arn \
        arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567 \
```

```
        --routing-control-state On \
        --region us-west-2 \
        --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

```
{}
```

You can update several routing controls at the same time with one API call: `update-routing-control-states`. (When the request is successful, the response is empty.)

2b. Update several routing control states at once (batch updates).

```
aws route53-recovery-cluster update-routing-control-states \
        --update-routing-control-state-entries \
        '[{"RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567",
        "RoutingControlState": "Off"}, \
        {"RoutingControlArn": "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
hijklmnop987654321",
        "RoutingControlState": "On"}]' \
        --region us-west-2 \
        --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

```
{}
```

Working with routing control components in ARC

Topics

- Creating routing control components in ARC
- Viewing and updating routing control states in ARC
- Creating safety rules for routing control
- Support cross-account for clusters in ARC

Creating routing control components in ARC

This section explains how to create a cluster, routing controls, health checks, and control panels for working with routing control in Amazon Application Recovery Controller (ARC).

Start by creating a cluster, to host your routing controls and the control panels that you use to group them. Then create routing controls and health checks so you can reroute traffic to fail over from one cell to another, so that traffic goes to your backup replica, for example.

Note that you are charged by the hour for each cluster that you create. You typically only need one cluster to host the routing controls and control panels for recovery control management for an application. In addition, you can set up resource sharing by using AWS Resource Access Manager, so that one cluster can host routing controls and other ARC resources owned by multiple AWS accounts. To learn about resource sharing in ARC, Support cross-account for clusters in ARC. For pricing information, see Amazon Application Recovery Controller (ARC) Pricing and scroll down to Amazon Route 53.

To use routing controls to fail over traffic, you create routing control health checks that you associate with Amazon Route 53 DNS records for resources in your application. As an example, let's say you have two cells, one that you've configured as the primary cell for your application, and the other that you've configured as the secondary, to fail over to.

To set up health checks for failover, do the following:

1. Create a routing control for each cell.
2. Create a health check for each routing control.
3. Create two DNS records, for example, two DNS failover records, and associate a health check with each one.

Another scenario when you might create a routing control is when you create a safety rule that is a gating rule. In this case, you don't associate health checks and DNS records with the routing control because you will use it as a *gating routing control*. For more information, see Creating safety rules for routing control .

The steps to create the components for routing control on the ARC console are included in these sections. To learn about using recovery control configuration API operations with ARC, see the Routing control API operations.

Creating a cluster in ARC

You must create a cluster to host routing controls and control panels in ARC.

A *cluster* is a set of redundant Regional endpoints against which you can execute API calls to update or get the state of one or more routing controls. A single cluster can host a number of routing controls.

> **Important**
>
> Be aware that you are charged by the hour for each cluster that you create. One cluster can host a number of routing controls and control panels for recovery control management, typically enough for an application.

To create a cluster

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Clusters**.
3. Choose **Create**, and then enter a name for your cluster.
4. Choose **Create cluster**.

Creating a routing control in ARC

Create a routing control for each cell that you want to route traffic to. For example, when you have an application with resources that you have siloed for recoverability, you might have a cell for each AWS Region, and nested cells for each Availability Zone within each Region. In this scenario, you would create a routing control for each cell and each nested cell.

When you create routing controls, keep in mind that routing control names must be unique within each control panel.

After you create routing controls to use for rerouting traffic, you associate each one with a health check, which allows you to route traffic to cells, based on the DNS records that you've associated with each one. If you're setting up a gating rule as a safety rule and creating a gating routing control, you don't add a health check to the routing control.

To create a routing control

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.

2. Choose **Routing control**.
3. On the **Routing control** page, choose **Create**, and then choose a **Routing control**.
4. Enter a name for your routing control, choose the cluster to add the control to, and choose to add it to an existing control panel, including using the default control panel. Or, create a new control panel.
5. If you choose to create a new control panel, choose a cluster to create the control panel on, and then enter a name for the panel.
6. Choose **Create routing control**.
7. Follow the steps to name and create the routing control.

Creating a routing control health check in ARC

You associate a routing control health check with each routing control that you want to use for rerouting traffic. Then you configure each health check with a Amazon Route 53 DNS record, for example, a failover DNS record. Then you can reroute traffic in Amazon Application Recovery Controller (ARC) simply by updating the state of the associated routing control, to set it to On or `Off`.

> **Note**
>
> You can't edit an existing routing control health check to associate it with a different routing control.

To create a routing control health check

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Routing control**.
3. On the **Routing control** page, choose a routing control.
4. On the **Routing control** detail page, choose a **Create health check**.
5. Enter a name for the health check, and then choose **Create**.

Next, you create Route 53 DNS records, and associate your routing control health checks with each one. For example, let's assume that you want to use two DNS failover records to associate

your routing control health checks with. For ARC to correctly fail over traffic by using routing controls, start by creating the two failover records in Route 53: a primary and a secondary. For more information about configuring DNS failover records, see Health checking concepts.

When you create the primary failover record, the values should be something like the following:

```
Name: myapp.yourdomain.com
Type: CNAME
Set Identifier: Primary
Failover: Primary
TTL: 0
Resource Records:
Value: cell1.yourdomain.com
Health Check ID: xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

The secondary failover record values should be something like the following:

```
Name: myapp.yourdomain.com
Type: CNAME
Set Identifier: Secondary
Failover: Secondary
TTL: 0
Resource Records:
Value: cell2.yourdomain.com
Health Check ID: xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Now, say that you want to reroute traffic because there's a failure. To do this, you update the associated routing control states to change the primary routing control state to `OFF` and the secondary routing control state to `ON`. When you do this, the associated health checks stop traffic from going to the primary replica and route it instead to the secondary replica. For more information about failing over traffic with routing controls, see Getting and updating routing control states using the ARC API (recommended).

To see examples of the AWS CLI commands for creating routing controls and the associated health checks using ARC API operations, see Examples of using ARC routing control API operations with the AWS CLI.

Creating a control panel in ARC

A control panel in Amazon Application Recovery Controller (ARC) lets you group together related routing controls. A control panel can have routing controls that represent a microservice within an application, an entire application itself, or a group of applications, depending on the scope of your failover. A benefit of grouping routing controls into a control panel is that you can use safety rules with a control panel to help safeguard traffic routing changes.

When you create a cluster, ARC creates a default control panel. You can use the default control panel for your routing controls, or you can create one or more control panels to group your routing controls. Note that only ASCII characters are supported for control panel names.

The steps to create a control panel on the ARC console are included in this section. For information about using recovery control configuration API operations with ARC, see the Routing control API operations.

To create a control panel

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Routing control**.
3. On the **Routing control** page, choose **Create**, and then choose a **Control panel**.
4. Choose a cluster to create the control panel on, and then enter a name for the panel.
5. Choose **Create control panel**.

Viewing and updating routing control states in ARC

This section describes how to view and update routing control states in Amazon Application Recovery Controller (ARC). Routing controls are simple on-off switches that manage traffic flow to cells in your recovery group. Cells are typically AWS Regions, or sometimes Availability Zones, that includes your resources. When a routing control state is On, traffic flows to the cell that is controlled by that routing control.

You group routing controls into control panels, which are logical failover groupings. When you open a control panel on the console, for example, you can view all of the routing controls for a grouping at once, to see where traffic is flowing.

You can update a routing control state on the ARC console or by using the ARC API. We recommend that you update routing control states by using the API. First, ARC offers extreme reliability with

the API in the data plane to perform these actions. That's important when you're changing these states because routing state changes fail over across cells by rerouting application traffic. In addition, by using the API, you can try connecting to different cluster endpoints in rotation, as needed, if a cluster endpoint that you try connecting to is unavailable.

You can update one routing control state, or you can update several routing control states at once. For example, you might want to set one routing control state to `Off` to stop traffic from flowing to one cell, such as an Availability Zone where an application is experiencing increased latency. At the same time, you might want to set another routing control state to On to start traffic flowing to another cell or Availability Zone. In this scenario, you can update both routing control states at the same time, so traffic continues to flow.

Topics

- Getting and updating routing control states using the ARC API (recommended)
- Getting and updating routing control states in the AWS Management Console

Getting and updating routing control states using the ARC API (recommended)

We recommend that you use Amazon Application Recovery Controller (ARC) API operations to get or update routing control states, by using an AWS CLI command or by using code that you have developed to use ARC API operations with one of the AWS SDKs. We recommend using API operations, with the CLI or in code, for working with routing control states, rather than using the AWS Management Console.

ARC offers extreme reliability for failing over across cells (AWS Regions) by updating routing control states using the API because routing controls are stored in a highly available cluster. ARC ensures that at least three out of the five Regional cluster endpoints are always accessible to you to make routing control state changes. To get or change a routing control state using the API, you connect to one of your Regional cluster endpoints. If the endpoint is unavailable, you can try connecting to another one of your cluster endpoints.

You can view the list of Regional cluster endpoints for your cluster in the Route 53 console, or by using an API action, DescribeCluster. Your process for getting and changing routing control states should try each endpoint in rotation, as needed, since cluster endpoints are cycled through available and unavailable states for regular maintenance and updates.

We provide detailed information and code examples for using ARC API operations to get and update routing control states, and work with Regional cluster endpoints. For more information, see the following :

- For code examples that explain how to rotate through Regional cluster endpoints to get and set routing control states, see Actions for Application Recovery Controller using AWS SDKs .
- For information about using the AWS CLI to get and update routing control states, see List and update routing controls and states with the AWS CLI.

Getting and updating routing control states in the AWS Management Console

You can get and update routing control states in the AWS Management Console. Be aware, though, that you can't choose different Regional cluster endpoints in the console. That is, there isn't a process for choosing and rotating through cluster endpoints in the console as you can do by using the Amazon Application Recovery Controller (ARC) API. In addition, the console is not highly available while the ARC data plane offers extreme reliability. For these reasons, we recommend that you use the ARC API to get and update routing control states for production operations.

For more recommendations about using ARC for failover, see Best practices for routing control in ARC.

To view and update routing controls in the console, follow the steps in the following procedures.

To get routing control states

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Routing control**.
3. From the list, choose a control panel and view the routing controls.

To update one or multiple routing control states

1. Open the Amazon Route 53 console at https://console.aws.amazon.com/route53/home.
2. Under **Application Recovery Controller**, choose **Routing control**.
3. Choose **Action**, and then choose **Change traffic routing**.
4. Update the states of one or more routing controls to be `Off` or `On`, depending on where you want traffic to flow or stop flowing for your application.

5. Enter `confirm` in the text box.
6. Choose **Update traffic routing**.

Creating safety rules for routing control

When you work with several routing controls at the same time, you might decide that you want safeguards in place to avoid unintended consequences. For example, you might want to prevent inadvertently turning off all the routing controls for an application, which would result in a fail-open scenario. Or you might want to implement a master on-off switch to disable a set of routing controls, perhaps to prevent automation from rerouting traffic. To establish safeguards like these for routing control in ARC, you create *safety rules*.

You configure safety rules for routing control with a combination of routing controls, rules, and other options that you specify. Each safety rule is associated with a single control panel, but a control panel can have more than one safety rule. When you create safety rules, keep in mind that safety rule names must be unique within each control panel.

Topics

Types of safety rules

There are two types of safety rules, *assertion rules* and *gating rules*, which you can use to safeguard failover in different ways.

Assertion rule

With an assertion rule, when you change one or a set of routing control states, ARC enforces that the criteria that you set when you configured the rule is met, or else the routing control states aren't changed.

An example of when this is useful is to prevent a fail-open scenario, like a scenario where you stop traffic from going to one cell but do not start traffic flowing to another cell. To avoid this, an assertion rule makes sure that at least one routing control in a set of routing controls in a

control panel is On at any given time. This ensures that traffic flows to at least one Region or Availability Zone for an application.

To see an example AWS CLI command that creates an assertion rule to enforce this criteria, see *Create safety rules* in Examples of using ARC routing control API operations with the AWS CLI.

For detailed information about the assertion rule API operation properties, see AssertionRule in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

Gating rule

With a gating rule, you can enforce an overall on-off switch over a set of routing controls so that whether those routing control states can be changed is enforced based on a set of criteria that you specify in the rule. The simplest criteria is whether a single routing control that you specify as the switch is set to ON or OFF.

To implement this, you create a *gating routing control*, to use as the overall switch, and *target routing controls*, to control traffic flow to different Regions or Availability Zones. Then, to prevent manual or automated state updates to the target routing controls that you've configured for the gating rule, you set the gating routing control state to Off. To allow updates, you set it to On.

To see an example AWS CLI command that creates a gating rule that implements this kind of overall switch, see *Create safety rules* in Examples of using ARC routing control API operations with the AWS CLI.

For detailed information about the gating rule API operation properties, see GatingRule in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

Creating a safety rule on the console

The steps in this section explain how to create a safety rule on the ARC console. The steps are similar whether you create an assertion rule or a gating rule. The differences are noted in the procedure.

To learn about using recovery and routing control API operations with Amazon Application Recovery Controller (ARC), see Routing control API operations.

To create a safety rule

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Routing control**.
3. On the **Routing control** page, choose a control panel.
4. On the control panel details page, choose **Action**, and then choose **Add safety rule**.
5. Choose a type of rule to add: **Assertion rule** or **Gating rule**.
6. Choose a name and, optionally, change the wait period.
7. Specify the configuration options for the safety rule.

   - For an assertion rule, specify the asserted routing controls.
   - For a gating rule, specify the gating routing control and target routing controls.

   For both rules, specify the rule configuration by choosing the type and threshold, and whether the rule is inverted.

   > **Note**
   >
   > To learn more about specifying an assertion rule, see the information provided for AssertionRule operation in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. To learn more about specifying a gating rule, see the information provided for the GatingRule operation in the Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller.

8. Choose **Create**.

Editing or deleting a safety rule on the console

The steps in this section explain how to edit or delete a safety rule on the ARC console. You can make only limited edits to a safety rule, to change the name or update the wait period. To make other changes, delete and recreate the safety rule.

To learn about using API operations with Amazon Application Recovery Controller (ARC), see the Routing control API operations.

To delete a safety rule

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Routing control**.
3. On the **Routing control** page, choose a control panel.
4. On the control panel details page, choose a safety rule, and then choose **Delete** or **Edit**.

Overriding safety rules to reroute traffic

There are scenarios when you might want to bypass the routing control safeguards that are enforced with safety rules that you've configured. For example, you might want to fail over quickly for disaster recovery, and one or more safety rules might be unexpectedly preventing you from updating a routing control state to reroute traffic. In a "break glass" scenario like this, you can override one or more safety rules to change a routing control state and fail over your application.

You can bypass safety rules when you update a routing control state (or multiple routing control states) by using the `update-routing-control-state` or `update-routing-control-states` AWS CLI command with the `safety-rules-to-override` parameter. Specify the parameter with the Amazon Resource Name (ARN) of the safety rule that you want to override, or specify a comma-separated list of ARNs to override two or more safety rules.

When a safety rule blocks a routing control state update, the error message includes the ARN of the rule that blocked the update. So you can make a note of the ARN, and then specify it in a routing control state CLI command with the safety rule override parameter.

> **Note**
>
> Because more than one safety rule might be in place for the routing controls that you're updating, you could run the CLI command to update your routing control state with one safety rule override but get an error that another safety rule is blocking the update. Continue to add safety rule ARNs to the list of rules to override in the update command, separated by commas, until the update command completes successfully.

To learn more about using the `SafetyRulesToOverride` property with the API and SDKs, see UpdateRoutingControlState.

The following are two examples of CLI commands to override safety rules to update routing control states.

Override one safety rule

```
aws route53-recovery-cluster --region us-west-2 update-routing-control-state \
    --routing-control-arn \
    arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/
routingcontrol/abcdefg1234567 \
    --routing-control-state On \
    --safety-rules-to-override arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/safetyrule/
yyyyyyy8888888 \
    --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

Override two safety rules

```
aws route53-recovery-cluster --region us-west-2 update-routing-control-state \
    --routing-control-arn \
    arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/
routingcontrol/abcdefg1234567 \
    --routing-control-state On \
    --safety-rules-to-override "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/safetyrule/
yyyyyyy8888888" \
    "arn:aws:route53-recovery-
control::111122223333:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/safetyrule/
qqqqqqq7777777"
    --endpoint-url https://host-dddddd.us-west-2.example.com/v1
```

Support cross-account for clusters in ARC

Amazon Application Recovery Controller (ARC) integrates with AWS Resource Access Manager to enable resource sharing. AWS RAM is a service that enables you to share resources with other AWS accounts or through AWS Organizations. For ARC, you can share the cluster resource.

With AWS RAM, you share resources that you own by creating a *resource share*. A resource share specifies the resources to share, and the *participants* to share them with. Participants can include:

- Specific AWS accounts inside or outside of owner's organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- Its entire organization in AWS Organizations

For more information about AWS RAM, see the *AWS RAM User Guide*.

By using AWS Resource Access Manager to share cluster resources across accounts in ARC, you can use one cluster to host control panels and routing controls owned by several different AWS accounts. When you opt to share a cluster, other AWS accounts that you specify can use the cluster to host their own control panels and routing controls, allowing more control and flexibility over routing capabilities across different teams.

AWS RAM is a service that helps AWS customers to securely share resources across AWS accounts. With AWS RAM, you can share resources within an organization or organizational units (OUs) in AWS Organizations, by using IAM roles and users. AWS RAM is a centralized and controlled way to share a cluster.

When you share a cluster, you can reduce the number of total clusters that your organization requires. With a shared cluster, you can allocate the total cost of running the cluster across different teams, to maximize the benefits of ARC with lower cost. (Creating resources that are hosted in a cluster does not have additional costs, for the owner or for participants.) Sharing clusters across accounts can also ease the process of onboarding multiple applications to ARC, especially if you have a large number of applications distributed across several accounts and operations teams.

To get started with cross-account sharing in ARC, you create a *resource share* in AWS RAM. The resource share specifies *participants* who are authorized to share the cluster that your account owns. Then, participants can create resources, such as control panels and routing controls, in the cluster, by using the AWS Management Console or by running ARC API operations using the AWS Command Line Interface or AWS SDKs.

This topic explains how to share resources that you own, and how to use resources that are shared with you.

Contents

- Identifying a shared cluster
- Responsibilities and permissions for shared clusters
- Billing costs
- Quotas

Prerequisites for sharing clusters

- To share a cluster, you must own it in your AWS account. This means that the resource must be allocated or provisioned in your account. You cannot share a cluster that has been shared with you.
- To share a cluster with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable sharing with AWS Organizations in the *AWS RAM User Guide*.

Sharing a cluster

When you share a cluster that you own, the participants that you specify to share the cluster can create and host their own ARC resources in the cluster.

To share a cluster, you must add it to a resource share. A resource share is an AWS RAM resource that lets you share your resources across AWS accounts. A resource share specifies the resources to share, and the participants they're shared with. To share a cluster you can create a new resource share or add the resource to an existing resource share. To create a new resource share, you can use the AWS RAM console, or use AWS RAM API operations with the AWS Command Line Interface or AWS SDKs.

If you are part of an organization in AWS Organizations and sharing within your organization is enabled, participants in your organization are automatically granted access to the shared cluster. Otherwise, participants receive an invitation to join the resource share and are granted access to the shared cluster after accepting the invitation.

You can share a cluster that you own by using the AWS RAM console, or by using AWS RAM API operations with the AWS CLI or SDKs.

To share a cluster that you own by using the AWS RAM console

See Creating a resource share in the *AWS RAM User Guide*.

To share a cluster that you own by using the AWS CLI

Use the create-resource-share command.

Unsharing a shared cluster

When you unshare a cluster, the following applies to participants and owners:

- Current participant resources continue to exist in the unshared cluster.
- Participants can continue to update routing control states in the unshared cluster, to manage routing for application failover.
- Participants can no longer create new resources in the unshared cluster.
- If participants still have resources in an unshared cluster, the owner cannot delete the shared cluster.

To unshare a shared cluster that you own, remove it from the resource share. You can do this by using the AWS RAM console or by using AWS RAM API operations with the AWS CLI or SDKs.

To unshare a shared cluster that you own using the AWS RAM console

See Updating a resource share in the *AWS RAM User Guide*.

To unshare a shared cluster that you own using the AWS CLI

Use the disassociate-resource-share command.

Identifying a shared cluster

Owners and participants can identify shared clusters by viewing information in AWS RAM. They can also get information about shared resources by using the ARC console and AWS CLI.

In general, to learn more about the resources that you've shared or that have been shared with you, see the information in the AWS Resource Access Manager User Guide:

- As an owner, you can view all resources that you are sharing with others by using AWS RAM. For more information, see Viewing your shared resources in AWS RAM.
- As a participant, you can view all resources shared with you by using AWS RAM. For more information, see Viewing your shared resources in AWS RAM.

As an owner, you can determine if you're sharing a cluster by viewing information in the AWS Management Console or by using the AWS Command Line Interface with ARC API operations.

To identify if a cluster that you own is shared by using the console

In the AWS Management Console, on the details page for a cluster, see the **Cluster sharing status**.

To identify if a cluster that you own is shared by using the AWS CLI

Use the get-resource-policy command. If there is a resource policy for a cluster, the command returns information about the policy.

As a participant, when a cluster is shared with you, you typically must accept the share. In addition, the **Owner** field for the cluster contains the account of the cluster owner.

Responsibilities and permissions for shared clusters

Permissions for owners

When you share a cluster that you own with other AWS accounts, participants who are permitted to use the cluster can create control panels, routing controls, and other resources in the cluster.

As a cluster owner, you are responsible for creating, managing, and deleting clusters. You can't modify or delete resources created by participants, such as routing controls and safety rules. For example, you can't update a routing control created by a participant to change the routing control state.

However, you can view the details for routing controls that are created by participants in a cluster that you own. For example, you can view routing control states by calling a ARC routing control API operation, using the AWS Command Line Interface or AWS SDKs.

If you need to modify resources create by participants, they can set up a role in IAM with permission to access the resources, and add your account to the role.

Permissions for participants

In general, participants can create and use control panels, routing controls, safety rules, and health checks that they create in a cluster that is shared with them. They can only view, modify, or delete cluster resources in the shared cluster if they own the resources. For example, participants can create and delete safety rules for control panels that they have created.

The following restrictions apply for participants:

- Participants cannot view, modify, or delete control panels created by other accounts using a shared cluster.

- Participants cannot view, create, or modify routing controls, including routing control states, for resources created in a shared cluster by other accounts.
- Participants cannot create, modify, or view safety rules created by other accounts in a shared cluster.
- Participants cannot add resources in the default control panel in a shared cluster because it belongs to the cluster owner.

As noted, participants cannot create routing controls in the default control panel for a shared cluster, because the cluster owner owns the default control panel. However, the cluster owner can create a cross-account IAM role that provides permission to access the default control panel for the cluster. Then, the owner can grant a participant permissions to assume the role, so that the participant can access the default control panel to use it however the owner has specified through the role's permissions.

Billing costs

The owner of a cluster in ARC is billed for costs associated with the cluster. There are no additional costs, for cluster owners or for participants, for creating resources hosted in a cluster.

For detailed pricing information and examples, see Amazon Application Recovery Controller (ARC) Pricing and scroll down to Amazon Application Recovery Controller (ARC).

Quotas

All resources created in a shared cluster—including resources created by all participants with access to the shared cluster—count toward quotas in effect for the cluster and other resources, such as routing controls. If accounts that share the cluster resource have a higher quota than the cluster owner's quotas, the cluster owner's quotas takes precedence over the quotas for the accounts that are sharing.

To better understand how this works, see the following examples. To illustrate how quotas work with resource sharing, for these examples, let's say that the cluster owner is Owner and an account that the cluster has been shared with is Participant.

Control panels quota

Quotas are enforced for Owner's total control panels per cluster.

For example, say Owner has a quota of 50 for the number of control panels per cluster, and has 13 control panels in the cluster. Now, say that Participant has the quota set to 150. In

this scenario, Participant can only create up to 37 control panels (that is, 50-13) in the shared cluster.

In addition, if other accounts that share the cluster also create control panels, those also all count toward the cluster overall quota of 50 control panels.

Routing control quotas

Routing controls have multiple quotas: a quota per control panel, a quota per cluster, and a quota per safety rule. Owner's quotas take precedence for all of these quotas.

For example, say Owner has a quota of 300 for the number of routing controls per cluster, and already has 300 routing controls in the cluster. Now, say that Participant has this quota set to 500. In this scenario, Participant cannot create any new routing controls in the shared cluster.

Safety rules quotas

Quotas are enforced for Owner's safety rules per control panel quota.

For example, say Owner has a quota of 20 for the number of safety rules per control panel and Participant has this quota set to 80. In this scenario, because Owner's lower limit takes precedence, Participant can only create up to 20 safety rules in a control panel in the shared cluster.

For a list of routing control quotas, see Quotas for routing control.

Logging and monitoring for routing control in Amazon Application Recovery Controller (ARC)

You can use AWS CloudTrail for monitoring routing control in Amazon Application Recovery Controller (ARC), to analyze patterns and help troubleshoot issues.

Topics

Logging Route 53 ARC API calls using AWS CloudTrail

Amazon Route 53 Application Recovery Controller is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Route 53 ARC. CloudTrail

captures all API calls for Route 53 ARC as events. The calls captured include calls from the Route 53 ARC console and code calls to the Route 53 ARC API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Route 53 ARC. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to Route 53 ARC, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Route 53 ARC information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Route 53 ARC, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Working with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for Route 53 ARC, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All Route 53 ARC actions are logged by CloudTrail and are documented in the Recovery Readiness API Reference Guide for Amazon Route 53 Application Recovery Controller, Recovery Control Configuration API Reference Guide for Amazon Route 53 Application Recovery Controller, and Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. For example, calls to the `CreateCluster`, `UpdateRoutingControlState` and `CreateRecoveryGroup` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Viewing Route 53 ARC events in event history

CloudTrail lets you view recent events in **Event history**. To view events for Route 53 ARC API requests, you must choose **US West (Oregon)** in the Region selector at the top of the console. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*.

Understanding Route 53 ARC log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateCluster` action for configuring routing control.

```
{
  "eventVersion": "1.08",
   "userIdentity": {
     "type": "IAMUser",
     "principalId": "A1B2C3D4E5F6G7EXAMPLE",
     "arn": "arn:aws:iam::111122223333:user/smithj",
     "accountId": "111122223333",
     "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
     "sessionContext": {
          "sessionIssuer": {
              "type": "Role",
              "principalId": "A1B2C3D4E5F6G7EXAMPLE",
              "arn": "arn:aws:iam::111122223333:role/smithj",
```

```
            "accountId": "111122223333",
            "userName": "smithj"
        },
        "webIdFederationData": {},
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2021-06-30T04:44:41Z"
        }
    }
  },
  "eventTime": "2021-06-30T04:45:46Z",
  "eventSource": "route53-recovery-control-config.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.50",
  "userAgent": "aws-cli/2.0.0 Python/3.8.2 Darwin/19.6.0 botocore/2.0.0dev7",
  "requestParameters": {
      "ClientToken": "12345abcdef-1234-5678-abcd-12345abcdef",
      "ClusterName": "XYZCluster"
  },
  "responseElements": {
      "Cluster": {
          "Arn": "arn:aws:route53-recovery-control::012345678901:cluster/abc123456-
aa11-bb22-cc33-abc123456",
          "ClusterArn": "arn:aws:route53-recovery-control::012345678901:cluster/
abc123456-aa11-bb22-cc33-abc123456",
          "Name": "XYZCluster",
          "Status": "PENDING"
      }
  },
  "requestID": "6090509a-5a97-4be6-8e6a-7d73example",
  "eventID": "9cab44ef-0777-41e6-838f-f249example",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333"
}
```


The following example shows a CloudTrail log entry that demonstrates the `UpdateRoutingControlState` action for routing control.


```
{
```

```
  "eventVersion": "1.08",
   "userIdentity": {
     "type": "AssumedRole",
     "principalId": "A1B2C3D4E5F6G7EXAMPLE",
     "arn": "arn:aws:sts::111122223333:assumed-role/admin/smithj",
     "accountId": "111122223333",
     "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
     "sessionContext": {
          "sessionIssuer": {
              "type": "Role",
              "principalId": "A1B2C3D4E5F6G7EXAMPLE",
              "arn": "arn:aws:iam::111122223333:role/admin",
              "accountId": "111122223333",
              "userName": "admin"
          },
          "webIdFederationData": {},
          "attributes": {
              "mfaAuthenticated": "false",
              "creationDate": "2021-06-30T04:44:41Z"
          }
      }
  },
  "eventTime": "2021-06-30T04:45:46Z",
  "eventSource": "route53-recovery-control-config.amazonaws.com",
  "eventName": "UpdateRoutingControl",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.50",
  "userAgent": "aws-cli/2.0.0 Python/3.8.2 Darwin/19.6.0 botocore/2.0.0dev7",
  "requestParameters": {
      "RoutingControlName": "XYZRoutingControl3",
      "RoutingControlArn": "arn:aws:route53-recovery-
control::012345678:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567"
  },
  "responseElements": {
      "RoutingControl": {
          "ControlPanelArn": "arn:aws:route53-recovery-
control::012345678:controlpanel/0123456bbbbbbb0123456bbbbbb0123456",
          "Name": "XYZRoutingControl3",
          "Status": "DEPLOYED",
          "RoutingControlArn": "arn:aws:route53-recovery-
control::012345678:controlpanel/0123456bbbbbbb0123456bbbbbb0123456/routingcontrol/
abcdefg1234567"
      }
```

```
  },
  "requestID": "6090509a-5a97-4be6-8e6a-7d73example",
  "eventID": "9cab44ef-0777-41e6-838f-f249example",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333"
}
```

Identity and Access Management for routing control

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Route 53 ARC resources. IAM is an AWS service that you can use with no additional charge.

Contents

How routing control in Amazon Route 53 Application Recovery Controller works with IAM

Before you use IAM to manage access to routing control in Amazon Application Recovery Controller (ARC), learn what IAM features are available to use with routing control.

IAM features that you can use with routing control in Amazon Route 53 Application Recovery Controller

IAM feature	Routing control support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes

IAM feature	Routing control support
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	No

To get a high-level, overall view of how AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Route 53 ARC

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

To view examples of ARC identity-based policies for routing control, see Identity-based policy examples for routing control in Amazon Route 53 Application Recovery Controller.

Resource-based policies within routing control

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource.

Policy actions for routing control

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Route 53 ARC actions for routing control, see Actions defined by Amazon Route 53 Recovery Controls and Actions defined by Amazon Route 53 Recovery Cluster in the *Service Authorization Reference*.

Policy actions in Route 53 ARC for routing control use the following prefixes before the action, depending on the API that you're working with:

```
route53-recovery-control-config
route53-recovery-cluster
```

To specify multiple actions in a single statement, separate them with commas. For example, you could do the following:

```
"Action": [
      "route53-recovery-control-config:action1",
      "route53-recovery-control-config:action2"
```

```
        ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "route53-recovery-control-config:Describe*"
```

To view examples of ARC identity-based policies for routing control, see Identity-based policy examples for routing control in Amazon Route 53 Application Recovery Controller.

Policy resources for Route 53 ARC

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

In the *Service Authorization Reference*, you can see the following information related to Route 53 ARC:

To see a list of resource types and their ARNs, and the actions that you can specify with the ARN of each resource, see the following topics in the *Service Authorization Reference*:

- Actions defined by Amazon Route 53 Recovery Controls
- Actions defined by Amazon Route 53 Recovery Cluster.

To view examples of ARC identity-based policies for routing control, see Identity-based policy examples for routing control in Amazon Route 53 Application Recovery Controller.

Policy condition keys for Route 53 ARC

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Route 53 ARC condition keys for routing control, see the following topics in the *Service Authorization Reference*:

- Condition keys for Amazon Route 53 Recovery Controls
- Condition keys for Amazon Route 53 Recovery Cluster

To see the actions and resources that you can use with a condition key, see the following topics in the *Service Authorization Reference*:

- To see a list of resource types and their ARNs, see Actions defined by Amazon Route 53 Recovery Controls and Actions defined by Amazon Route 53 Recovery Cluster.
- To see a list of the actions that you can specify with the ARN of each resource, see Resources defined by Amazon Route 53 Recovery Controls and Resources defined by Amazon Route 53 Recovery Cluster.

To view examples of ARC identity-based policies for routing control, see Identity-based policy examples for routing control in Amazon Route 53 Application Recovery Controller

Access control lists (ACLs) in Route 53 ARC

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Route 53 ARC

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

ARC routing control includes the following support for ABAC:

- Recovery Control Config supports ABAC.
- Recovery Cluster does not support ABAC.

Using temporary credentials with Route 53 ARC

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Route 53 ARC

Supports forward access sessions (FAS): Yes

When you use an IAM entity (user or role) to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions.

To see whether an action requires additional dependent actions in a policy, see the following topics in the *Service Authorization Reference*:

- Amazon Route 53 Recovery Cluster
- Amazon Route 53 Recovery Controls

Service roles for Route 53 ARC

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for Route 53 ARC

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Routing control does not use service-linked roles.

Identity-based policy examples for routing control in Amazon Route 53 Application Recovery Controller

By default, users and roles don't have permission to create or modify Route 53 ARC resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Route 53 ARC, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Route 53 Application Recovery Controller in the *Service Authorization Reference*.

Topics

- Policy best practices
- Example: Route 53 ARC console access for routing control
- Examples: Route 53 ARC API actions for routing control configuration

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Route 53 ARC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Example: Route 53 ARC console access for routing control

To access the Amazon Route 53 Application Recovery Controller console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Route 53 ARC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the Route 53 ARC console when you allow access to only specific API operations, also attach a `ReadOnly` AWS managed policy for Route 53 ARC to the entities. For more information, see the Route 53 ARC Route 53 ARC managed policies page or Adding permissions to a user in the *IAM User Guide*.

To give users full access to use Route 53 ARC routing control features through the console, attach a policy like the following to the user, to give the user full permissions to configure Route 53 ARC routing control resources and operations:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                   "route53-recovery-cluster:GetRoutingControlState",
                   "route53-recovery-cluster:UpdateRoutingControlState",
                   "route53-recovery-cluster:UpdateRoutingControlStates",
                   "route53-recovery-control-config:CreateCluster",
                   "route53-recovery-control-config:CreateControlPanel",
                   "route53-recovery-control-config:CreateRoutingControl",
                   "route53-recovery-control-config:CreateSafetyRule",
                   "route53-recovery-control-config:DeleteCluster",
                   "route53-recovery-control-config:DeleteControlPanel",
                   "route53-recovery-control-config:DeleteRoutingControl",
                   "route53-recovery-control-config:DeleteSafetyRule",
                   "route53-recovery-control-config:DescribeCluster",
                   "route53-recovery-control-config:DescribeControlPanel",
                   "route53-recovery-control-config:DescribeSafetyRule",
                   "route53-recovery-control-config:DescribeRoutingControl",
```

```
                "route53-recovery-control-config:ListAssociatedRoute53HealthChecks",
                "route53-recovery-control-config:ListClusters",
                "route53-recovery-control-config:ListControlPanels",
                "route53-recovery-control-config:ListRoutingControls",
                "route53-recovery-control-config:ListSafetyRules",
                "route53-recovery-control-config:UpdateControlPanel",
                "route53-recovery-control-config:UpdateRoutingControl",
                "route53-recovery-control-config:UpdateSafetyRule"
            ],
            "Resource": "*"
        },
                {
            "Effect": "Allow",
            "Action": [
                "route53:GetHealthCheck",
                "route53:CreateHealthCheck",
                "route53:DeleteHealthCheck",
                "route53:ChangeTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Examples: Route 53 ARC API actions for routing control configuration

To ensure that a user can use Route 53 ARC API actions to work with Route 53 ARC routing control configuration, attach a policy that corresponds to the API operations that the user needs to work with, as described below.

To work with API operations for recovery control configuration, attach a policy like the following to the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "route53-recovery-control-config:CreateCluster",
                "route53-recovery-control-config:CreateControlPanel",
                "route53-recovery-control-config:CreateRoutingControl",
                "route53-recovery-control-config:CreateSafetyRule",
```

```
                "route53-recovery-control-config:DeleteCluster",
                "route53-recovery-control-config:DeleteControlPanel",
                "route53-recovery-control-config:DeleteRoutingControl",
                "route53-recovery-control-config:DeleteSafetyRule",
                "route53-recovery-control-config:DescribeCluster",
                "route53-recovery-control-config:DescribeControlPanel",
                "route53-recovery-control-config:DescribeSafetyRule",
                "route53-recovery-control-config:DescribeRoutingControl",
                "route53-recovery-control-config:GetResourcePolicy",
                "route53-recovery-control-config:ListAssociatedRoute53HealthChecks",
                "route53-recovery-control-config:ListClusters",
                "route53-recovery-control-config:ListControlPanels",
                "route53-recovery-control-config:ListRoutingControls",
                "route53-recovery-control-config:ListSafetyRules",
                "route53-recovery-control-config:ListTagsForResource",
                "route53-recovery-control-config:UpdateControlPanel",
                "route53-recovery-control-config:UpdateRoutingControl",
                "route53-recovery-control-config:UpdateSafetyRule",
                "route53-recovery-control-config:TagResource",
                "route53-recovery-control-config:UntagResource"
            ],
            "Resource": "*"
        }
    ]
}
```

To perform tasks in ARC routing control with the recovery cluster data plane API, for example, updating routing control states to fail over during a disaster event, you can attach a ARC IAM policy such as the following to your IAM user.

The `AllowSafetyRuleOverride` boolean gives permission to override safety rules that you've configured as safeguards for routing controls. This permission might be required in "break glass" scenarios to bypass the safeguards in disasters or other urgent failover scenarios. For example, an operator might need to fail over quickly for disaster recovery, and one or more safety rules might unexpectedly prevent a routing control state update required to reroute traffic. This permission allows the operator to specify safety rules to override when making API calls to update routing control states. For more information, see Overriding safety rules to reroute traffic.

If you want to allow an operator to use the recovery cluster data plane API but *prevent* overriding safety rules, you can attach a policy such as the following, with `AllowSafetyRuleOverrides` boolean to `false`. To allow the operator to override safety rules, set the `AllowSafetyRuleOverrides` boolean to `true`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "route53-recovery-cluster:GetRoutingControlState",
                "route53-recovery-cluster:ListRoutingControls"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "route53-recovery-cluster:UpdateRoutingControlStates",
                "route53-recovery-cluster:UpdateRoutingControlState"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "route53-recovery-cluster:AllowSafetyRulesOverrides": "false"
                }
            }
        }
    ]
}
```

AWS managed policies for routing control in Amazon Application Recovery Controller (ARC)

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users,

groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonRoute53RecoveryControlConfigFullAccess

You can attach `AmazonRoute53RecoveryControlConfigFullAccess` to your IAM entities. This policy grants full access to actions for working with recovery control configuration in ARC. Attach it to IAM users and other principals who need full access to recovery control configuration actions.

At your discretion, you can add access to additional Amazon Route 53 actions to enable users to create health checks for routing controls. For example, you might allow permission for one or more of the following actions: `route53:GetHealthCheck`, `route53:CreateHealthCheck`, `route53:DeleteHealthCheck`, and `route53:ChangeTagsForResource`.

To view the permissions for this policy, see AmazonRoute53RecoveryControlConfigFullAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AmazonRoute53RecoveryControlConfigReadOnlyAccess

You can attach `AmazonRoute53RecoveryControlConfigReadOnlyAccess` to your IAM entities. It's useful for users who need to view routing control and safety rule configurations. This policy grants read-only access to actions for working with recovery control configuration in ARC. These users can't create, update, or delete recovery control resources.

To view the permissions for this policy, see AmazonRoute53RecoveryControlConfigReadOnlyAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AmazonRoute53RecoveryClusterFullAccess

You can attach `AmazonRoute53RecoveryClusterFullAccess` to your IAM entities. This policy grants full access to actions for working with the cluster data plane in ARC. Attach it to IAM users and other principals who need full access to updating and retrieving routing control states.

To view the permissions for this policy, see AmazonRoute53RecoveryClusterFullAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AmazonRoute53RecoveryClusterReadOnlyAccess

You can attach `AmazonRoute53RecoveryClusterReadOnlyAccess` to your IAM entities. This policy grants read-only access to the cluster data plane in ARC. These users can retrieve routing control states but can't update them.

To view the permissions for this policy, see AmazonRoute53RecoveryClusterReadOnlyAccess in the *AWS Managed Policy Reference*.

Updates for AWS managed policies for routing control

For details about updates to AWS managed policies for routing control in ARC since this service began tracking these changes, see Updates to AWS managed policies for Amazon Application Recovery Controller (ARC). For automatic alerts about changes to this page, subscribe to the RSS feed on the ARC Document history page.

Quotas for routing control

Routing control in Amazon Application Recovery Controller (ARC) is subject to the following quotas (formerly referred to as limits).

Entity	Quota
Number of clusters per account	2
Number of control panels per cluster	50
Number of routing controls per control panel	100
Total number of routing controls (in all control panels) per cluster	300
Number of safety rules per control panel	20
Number of routing controls per UpdateRoutingControlStates operation call	10

Entity	Quota
Number of mutating API calls to a cluster endpoint, per second	3

Readiness check in Amazon Application Recovery Controller (ARC)

With readiness check in Amazon Application Recovery Controller (ARC), you can gain insights into whether your applications and resources are prepared for recovery. After you model your AWS application in ARC and create readiness checks, the checks continually monitor information about your application, such as AWS resource quotas, capacity, and network routing policies. Then, you can choose to be notified about changes that would affect your ability to fail over to a replica of your application, to recover from an event. Readiness checks help make sure, on an ongoing basis, that you can maintain your multi-Region applications in a state that is scaled and configured to handle failover traffic.

This chapter explains how to model your application in ARC to set up the structure that enables readiness checks to work, by creating a recovery group and cells that describe your application. Then, you can follow the steps to add readiness checks and readiness scopes so that ARC can audit readiness for your application.

After you create readiness checks, you can monitor the readiness status of your resources. Readiness checks help you to ensure that a standby application replica and its resources match your production replica on an ongoing basis, reflecting the capacity, routing policies, and other configuration details of your production application. If the replica doesn't match, you can add capacity or change a configuration so that your application replicas are aligned again.

> **Important**
>
> Readiness checks are most useful for verifying, on an ongoing basis, that application replica configurations and runtime states are aligned. Readiness checks shouldn't be used to indicate whether your production replica is healthy, nor should you rely on readiness checks as a primary trigger for failover during a disaster event.

What is readiness check in Amazon Application Recovery Controller (ARC)?

A readiness check in ARC continually (at one-minute intervals) audits for mismatches in AWS provisioned capacity, service quotas, throttle limits, and configuration and version discrepancies for the resources included in the check. Readiness checks can notify you of these differences so that you can make sure that each replica has the same configuration setup and the same runtime state. Although readiness checks ensure that your configured capacities across replicas are consistent, you should not expect them to decide on your behalf what the capacity of your replica should be. For example, you should understand your application requirements so that you size your Auto Scaling groups with enough buffer capacity in each replica to manage if another cell is unavailable.

For quotas, when ARC detects a mismatch with a readiness check, it can take steps to align the quotas for the replicas by increasing the lower quota to match the higher quota. When the quotas match, the readiness check status shows `READY`. (Note that this isn't an immediate update process, and the total time depends on the specific resource type and other factors.)

The first step is setting up readiness checks to create a recovery group that represents your application. Each recovery group includes *cells* for each individual failure-containment unit or *replica* of your application. Next, you create resource sets for each resource type in your application, and associate *readiness checks* with the resource sets. Finally, you associate the resources with *readiness scopes*, so you can get readiness status about the resources in a recovery group (your application) or individual cells (replicas, which are Regions or Availability Zones (AZs)).

Readiness (that is, `READY` or `NOT READY`) is based on the resources that are in the scope of the readiness check and the set of rules for a resource type. There are sets of readiness rules for each resource type, which ARC checks use to audit resources for readiness. Whether a resource is `READY` or not is based on how each readiness rule is defined. All readiness rules evaluate resources, but some compare resources to each other and some look at specific information about each resource in the resource set.

By adding readiness checks, you can monitor readiness status, in one of several ways: with EventBridge, in the AWS Management Console, or by using ARC API actions. You can also monitor readiness status of resources in different contexts, including the readiness of cells and the readiness of your application. Use the cross-account authorization feature in ARC to make it easier to set up and monitor distributed resources from a single AWS account.

Monitoring application replicas with readiness checks

ARC audits your application replicas by using *readiness checks* to ensure that each one has the same configuration setup and the same runtime state. A readiness check continually audits AWS resource capacity, configuration, AWS quotas, and routing policies for an application, information that you can use to help make sure that replicas are ready for failover. Readiness checks help you to ensure that your recovery environment is scaled and configured to fail over to when needed.

The following sections provide more details about how readiness check works.

Readiness checks and your application replicas

To be prepared for recovery, you must maintain sufficient spare capacity in replicas at all times, to absorb failover traffic from another Availability Zone or Region. ARC continually (once a minute) inspects your application to ensure that your provisioned capacity matches across all Availability Zones or Regions.

The capacity that ARC inspects includes, for example, Amazon EC2 instance counts, Aurora read and write capacity units, and Amazon EBS volume size. If you scale up the capacity in your primary replica for resource values but forget to also increase the corresponding values in your standby replica, ARC detects the mismatch so that you can increase the values in the standby.

> **Important**
>
> Readiness checks are most useful for verifying, on an ongoing basis, that application replica configurations and runtime states are aligned. Readiness checks shouldn't be used to indicate whether your production replica is healthy, nor should you rely on readiness checks as a primary trigger for failover during a disaster event.

In an active-standby configuration, you should make decisions about whether to fail away from or to a cell based on your monitoring and health check systems, and consider readiness checks as a complementary service to those systems. ARC readiness checks are not highly available, so you should not depend on the checks being accessible during an outage. In addition, the resources that are checked might also not be available during a disaster event.

You can monitor the readiness status for your application's resources in specific cells (AWS Regions or Availability Zones) or for your overall application. You can be notified when a readiness check status changes, for example, to `Not ready`, by creating rules in EventBridge. For more

information, see Using readiness check in ARC with Amazon EventBridge. You can also view readiness status in the AWS Management Console, or by using API operations, such as `get-recovery-readiness`. For more information, see Readiness check API operations.

How readiness check works

ARC audits your application replicas by using *readiness checks* to ensure that each one has the same configuration setup and the same runtime state.

To be prepared for recovery, for example, you must maintain sufficient spare capacity at all times to absorb failover traffic from another Availability Zone or Region. ARC continually (once a minute) inspects your application to ensure that your provisioned capacity matches across all Availability Zones or Regions. The capacity that ARC inspects includes, for example, Amazon EC2 instance counts, Aurora read and write capacity units, and Amazon EBS volume size. If you scale up the capacity in your primary replica for resource values but forget to also increase the corresponding values in your standby replica, ARC detects the mismatch so that you can increase the values in the standby.

> **Important**
>
> Readiness checks are most useful for verifying, on an ongoing basis, that application replica configurations and runtime states are aligned. Readiness checks shouldn't be used to indicate whether your production replica is healthy, nor should you rely on readiness checks as a primary trigger for failover during a disaster event.

In an active-standby configuration, you should make decisions about whether to fail away from or to a cell based on your monitoring and health check systems, and consider readiness checks as a complementary service to those systems. ARC readiness checks are not highly available, so you should not depend on the checks being accessible during an outage. In addition, the resources that are checked might also not be available during a disaster event.

You can monitor the readiness status for your application's resources in specific cells (AWS Regions or Availability Zones) or for your overall application. You can be notified when a readiness check status changes, for example, to `Not ready`, by creating rules in EventBridge. For more information, see Using readiness check in ARC with Amazon EventBridge. You can also view readiness status in the AWS Management Console, or by using API operations, such as `get-recovery-readiness`. For more information, see Readiness check API operations.

How readiness rules determine readiness status

ARC readiness checks determine readiness status based on the predefined rules for each resource type and the way those rules are defined. ARC includes one group of rules for each type of resource that it supports. For example, ARC has groups of readiness rules for Amazon Aurora clusters, Auto Scaling groups, and so on. Some readiness rules compare resources in a set to each other, and some look at specific information about each resource in the resource set.

You can't add, edit, or remove readiness rules, or groups of rules. However, you can create an Amazon CloudWatch alarm and create a readiness check to monitor the state of the alarm. For example, you can create a custom CloudWatch alarm to monitor Amazon EKS container services, and create a readiness check to audit the readiness status of the alarm.

You can view all the readiness rules for each resource type in the AWS Management Console when you create a resource set, or you can view the readiness rules later by navigating to the details page for a resource set. You can also view readiness rules in the following section: Readiness rules in ARC.

When a readiness check audits a set of resources with a set of rules, the way each rule is defined determines whether the result will be `READY` or `NOT READY` for all the resources or if the result will be different for different resources. In addition, you can view readiness status in multiple ways. For example, you can view the readiness status of a group of resources in a resource set or view a summary of readiness status for a recovery group or a cell (that is, an AWS Region or Availability Zone, depending on how you've set up your recovery group).

The wording in each rule description explains how it evaluates the resources to determine the readiness status when that rule is applied. A rule is defined to inspect *each resource* or to inspect *all resources* in a resource set to determine readiness. Specifically, the rules work as follows:

- The rule inspects *each resource* in the resource set to ensure a condition.
  - If all resources succeed, all resources are set as `READY`.
  - If one resource fails, that resource is set as `NOT READY`, and the other cells remain `READY`.

  For example: **MskClusterState:** Inspects each Amazon MSK cluster to ensure that it is in an `ACTIVE` state.
- The rule inspects *all resources* in the resource set to ensure a condition.
  - If the condition is ensured, all resources are set as `READY`.
  - If any fails to meet the condition, all resources are set as `NOT READY`.

For example: **VpcSubnetCount:** Inspects all VPC subnets to ensure that they have the same number of subnets.

- Non-critical rule: The rule inspects all resources in the resource set to ensure a condition.
  - If any fails, the readiness status is unchanged. A rule with this behavior has a note in its description.

  For example: **ElbV2CheckAzCount:** Inspects each Network Load Balancer to ensure that it is attached to only one Availability Zone. Note: This rule does not affect readiness status.

In addition, ARC takes an extra step for quotas. If a readiness check detects a mismatch across cells for service quotas (the maximum value for resource creation and operations) for any supported resource, ARC automatically raises the quota for the resource with the lower quota. This applies only to quotas (limits). For capacity, you should add additional capacity as required for your application needs.

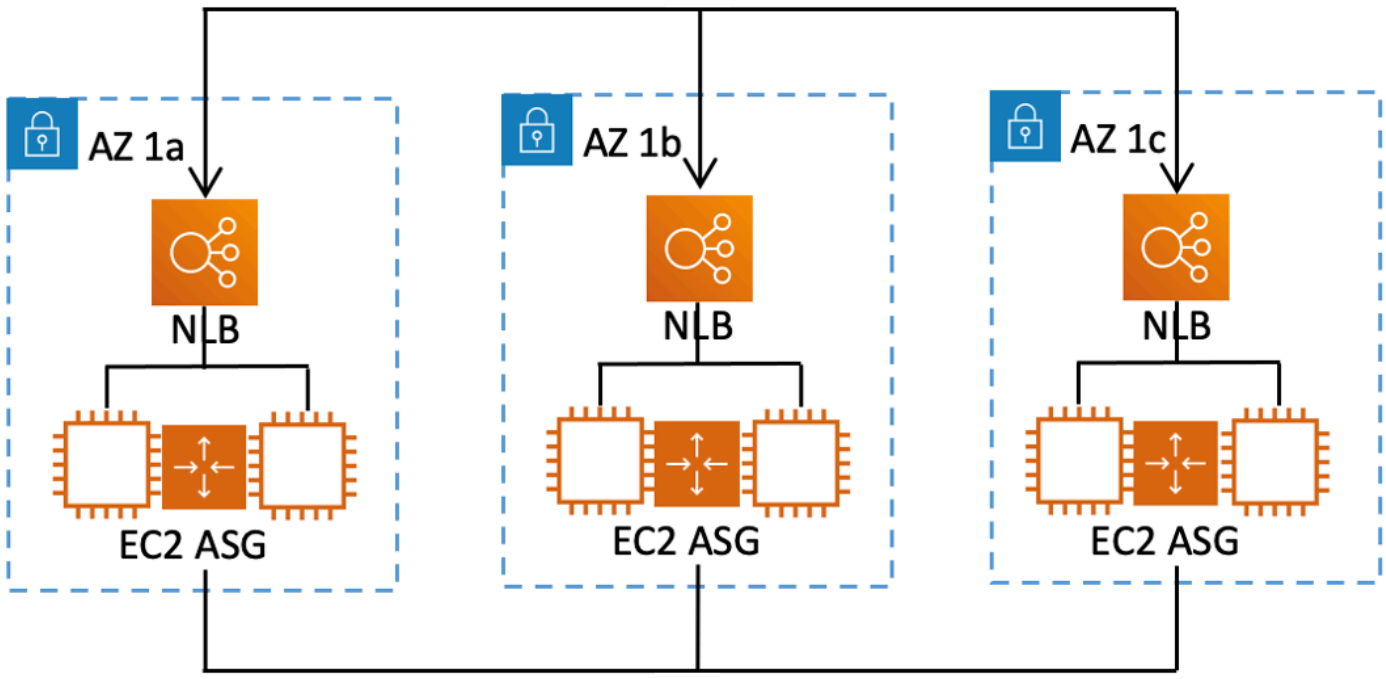
You can also set up an Amazon EventBridge notification for readiness checks, for example, when any readiness check status changes to `NOT READY`. Then when a configuration mismatch is detected, EventBridge sends you a notification and you can take corrective action to make sure that your application replicas are aligned and prepared for recovery. For more information, see Using readiness check in ARC with Amazon EventBridge.

How readiness checks, resource sets, and readiness scopes work together

Readiness checks always audit groups of resources in *resource sets*. You create resource sets (separately, or while you're creating a readiness check) to group the resources that are in the cells (Availability Zones or AWS Regions) in your ARC recovery group, so that you can define readiness checks. A resource set is typically a group of same type of resources (like Network Load Balancers) but can also be DNS target resources, for architectural readiness checks.

You typically create one resource set and readiness check for each type of resource in your application. For an architectural readiness check, you create a top level DNS target resource and a global (recovery group level) resource set for it, and then create cell level DNS target resources, for a separate resource set.

The following diagram shows an example of a recovery group with three cells (Availability Zones), each with a Network Load Balancer (NLB) and Auto Scaling group (ASG).



In this scenario, you would create a resource set and readiness check for the three Network Load Balancers, and a resource set and readiness check for the three Auto Scaling groups. Now you have a readiness check for each set of resources for your recovery group, by resource type.

By creating *readiness scopes* for resources, you can add readiness check summaries for cells or recovery groups. To specify a readiness scope for a resource, you associate the ARN of the cell or recovery group with each resource in a resource set. You can do this when you're creating a readiness check for a resource set.

For example, when you add a readiness check for a resource set for the Network Load Balancers for this recovery group, you can add readiness scopes to each NLB at the same time. In this case, you would associate the ARN of AZ 1a to the NLB in AZ 1a, the ARN of AZ 1b to the NLB AZ 1b, and the ARN of AZ 1c to the NLB in AZ 1c. When you create a readiness check for the Auto Scaling groups, you would do the same, assigning readiness scopes to each of them when you create the readiness check for the Auto Scaling group resource set.

It's optional to associate readiness scopes when you create a readiness check, however, we strongly recommend that you set them. Readiness scopes enable ARC to show the correct READY or NOT READY readiness status for recovery group summary readiness checks and cell level summary readiness checks. Unless you set readiness scopes, ARC can't provide these summaries.

Note that when you add an application-level or a global resource, such as a DNS routing policy, you don't choose a recovery group or cell for the readiness scope. Instead, you choose **global resource (no cell)**.

DNS target resource readiness checks: Auditing resiliency readiness

With DNS target resource readiness checks in ARC, you can audit the architectural and resiliency readiness of your application. This type of readiness check continually scans your application's architecture and Amazon Route 53 routing policies to audit for cross-zone and cross-Region dependencies.

A recovery-oriented application has multiple replicas that are siloed into Availability Zones or AWS Regions, so that the replicas can fail independently of one another. If your application needs adjusting to be siloed correctly, ARC will suggest changes that you can make, if needed, to update your architecture to help ensure that it's resilient and ready for failover.

ARC automatically detects the number and the scope of cells (representing replicas, or failure-containment units) in your application, and whether the cells are siloed by Availability Zone or by Region. Then, ARC identifies and provides information to you about the application resources in the cells, to determine if they are correctly siloed to zones or Regions. For example, if you have cells that are scoped to specific zones, readiness checks can monitor if your load balancers and the targets behind them are also siloed to those zones.

With this information, you can determine if there are changes that you need to make to align resources in your cells to the correct zones or Regions.

To get started, you create DNS target resources for your application, and resource sets and readiness checks for them. For more information, see Getting architecture recommendations in ARC.

Readiness checks and disaster recovery scenarios

ARC readiness checks give you insights into whether your applications and resources are ready for recovery by helping you make sure that your applications are scaled to handle failover traffic. Readiness check statuses should not be used as a signal to indicate that a production replica is healthy. You can, however, use readiness checks as a supplement to your application and infrastructure monitoring or health checker systems to determine whether to fail away from or to a replica.

In an urgent situation or an outage, use a combination of health checks and other information to determine that your standby is scaled up, healthy, and ready for you to fail over production traffic. For example, check to see if canaries that run against your standby cell are meeting your success criteria, in addition to verifying that readiness check statuses for the standby are READY.

Be aware that ARC readiness checks are hosted in a single AWS Region, US West (Oregon), and during an outage or disaster, readiness check information could become stale or the checks could become unavailable. For more information, see Data and control planes for routing control.

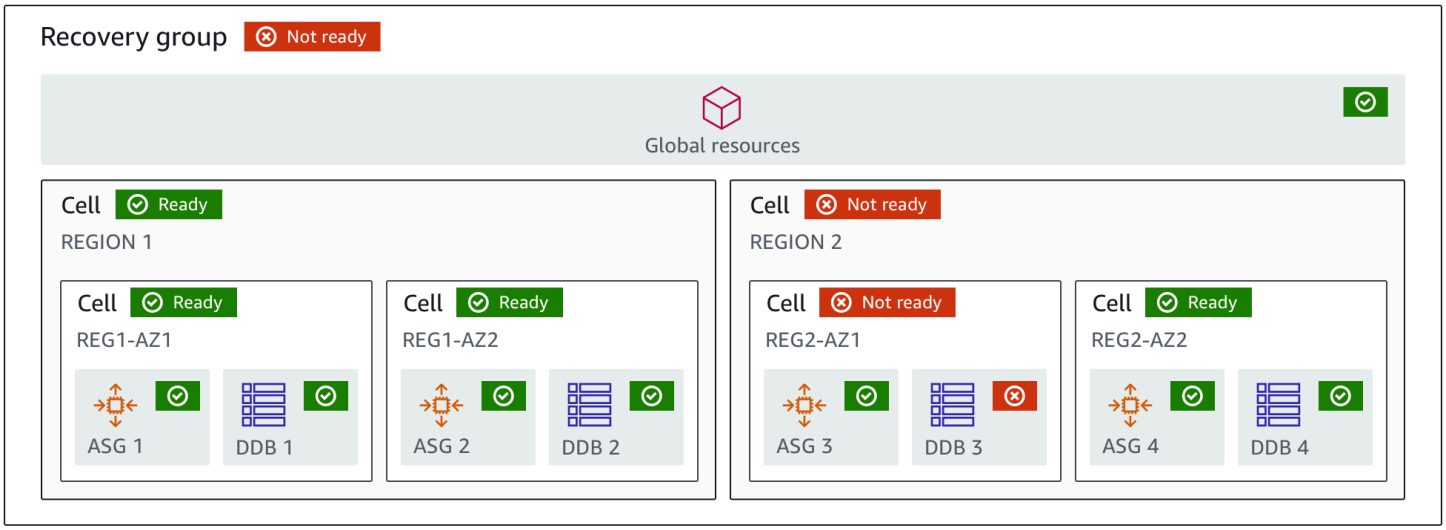
AWS Region availability for readiness check

For detailed information about Regional support and service endpoints for Amazon Application Recovery Controller (ARC), see Amazon Application Recovery Controller (ARC) endpoints and quotas in the *Amazon Web Services General Reference*.

> **Note**
>
> Readiness check in Amazon Application Recovery Controller (ARC) is a global feature. However, readiness check resources are in the US West (Oregon) Region, so you must specify the US West (Oregon) Region (specify the parameter `--region us-west-2`) in Regional ARC AWS CLI commands, for example, when you create resources such as resource sets and readiness checks.

Readiness check components

The following diagram illustrates a sample recovery group that is configured to support the readiness check feature. Resources in this example are grouped into cells (by AWS Region) and nested cells (by Availability Zones) in a recovery group. There is an overall readiness status for the recovery group (application), as well as individual readiness statuses for each cell (Region) and nested cell (Availability Zone).



The following are components of the readiness check feature in ARC.

Cell

A cell defines your application's replicas or independent units of failover. It groups all the AWS resources that are necessary for your application to run independently within the replica. For example, you might have one set of resources in a primary cell and another set in a standby cell. You determine the boundary of what a cell includes, but cells typically represent an Availability Zone or a Region. You can have multiple cells (nested cells) within a cell, such as AZs within a Region. Each nested cell represents an isolated unit of failover.

Recovery group

Cells are collected into a recovery group. A recovery group represents an application or group of applications that you want to check failover readiness for. It consists of two or more cells, or replicas, that match each other in terms of functionality. For example, if you have a web application that is replicated across us-east-1a and us-east-1b, where us-east-1b is your failover environment, you can represent this application in ARC as a recovery group with two cells: one in us-east-1a and one in us-east-1b. A recovery group can also include a global resource, such as a Route 53 health check.

Resources and resource identifiers

When you create components for readiness checks in ARC, you specify a resource, such as an Amazon DynamoDB table, a Network Load Balancer, or a DNS target resource, by using a resource identifier. A resource identifier is either the Amazon Resource Name (ARN) for the resource or, for a DNS target resource, the identifier that ARC generates when it creates the resource.

DNS target resource

A DNS target resource is the combination of your application's domain name and other DNS information, such as the AWS resource that the domain points to. Including an AWS resource is optional but if you provide it, it must be a Route 53 resource record or a Network Load Balancer. When you provide the AWS resource, you can get more detailed architectural recommendations that can help you improve your application's recovery resiliency. You can create resource sets in ARC for DNS target resources, and then create a readiness check for the resource set so that you can get architecture recommendations for your application. The readiness check also monitors the DNS routing policy for your application, based on the readiness rules for DNS target resources.

Resource set

A resource set is a set of resources, including AWS resources or DNS target resources, that span multiple cells. For example, you might have a load balancer in us-east-1a and another one in us-east-1b. To monitor the recovery readiness of the load balancers, you can create a resource set that includes both load balancers, and then create a readiness check for the resource set. ARC will continually check the readiness of the resources in the set. You can also add a readiness scope to associate resources in a resource set with the recovery group that you create for your application.

Readiness rule

Readiness rules are audits that ARC performs against a set of resources in a resource set. ARC has a set of readiness rules for each type of resource that it supports readiness checks for. Each rule includes an ID and a description that explains what ARC inspects the resources for.

Readiness check

A readiness check monitors a resource set in your application, such as a set of Amazon Aurora instances, that ARC is auditing recovery readiness for. Readiness checks can include auditing, for example, capacity configurations, AWS quotas, or routing policies. For example, if you want to audit readiness for your Amazon EC2 Auto Scaling groups across two Availability Zones, you can create a readiness check for a resource set with two resource ARNs, one for each Auto Scaling group. Then, to make sure that each group is scaled equally, ARC continually monitors the instance types and the counts in the two groups.

Readiness scope

A readiness scope identifies the grouping of resources that a specific readiness check encompasses. The scope of a readiness check can be a recovery group (that is, global to the

whole application) or a cell (that is, a Region or Availability Zone). For a resource that is a global resource for ARC, set the readiness scope at to recovery group or global resource level. For example, a Route 53 health check is a global resource in ARC because it isn't specific to a Region or Availability Zone.

Data and control planes for readiness check

As you plan for failover and disaster recovery, consider how resilient your failover mechanisms are. We recommend that you make sure that the mechanisms that you depend on during failover are highly available, so that you can use them when you need them in a disaster scenario. Typically, you should use data plane functions for your mechanisms whenever you can, for the greatest reliability and fault tolerance. With that in mind, it's important to understand how the functionality of a service is divided between control planes and data planes, and when you can rely on an expectation of extreme reliability with a service's data plane.

As with most AWS services, the functionality for the readiness check capability is supported by control planes and data planes. While both of these are built to be reliable, a control plane is optimized for data consistency, while a data plane is optimized for availability. A data plane is designed for resilience so that it can maintain availability even during disruptive events, when a control plane might become unavailable.

In general, a *control plane* enables you to do basic management functions, such as create, update, and delete resources in the service. A *data plane* provides a service's core functionality.

For readiness check, there is a single API, the Recovery Readiness API, for both the control plane and data plane. Readiness checks and readiness resources are only in the US West (Oregon) Region (us-west-2). *The readiness check control plane and data plane are reliable but not highly available.*

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see the Static stability using Availability Zones paper in the Amazon Builders' Library.

Tagging for readiness check in Amazon Application Recovery Controller (ARC)

Tags are words or phrases (meta data) that you use to identify and organize your AWS resources. You can add multiple tags to each resource, and each tag includes a key and a value that you

define. For example, the key might be environment and the value might be production. You can search and filter your resources based on the tags you add.

You can tag the following resources in readiness check in ARC:

- Resource sets
- Readiness checks

Tagging in ARC is available only through the API, for example, by using the AWS CLI.

The following are examples of tagging in readiness check by using the AWS CLI.

```
aws route53-recovery-readiness --region us-west-2 create-resource-
set --resource-set-name dynamodb_resource_set --resource-set-type
AWS::DynamoDB::Table --resources ReadinessScopes=arn:aws:aws-recovery-
readiness::111122223333:cell/PDXCell,ResourceArn=arn:aws:dynamodb:us-
west-2:111122223333:table/PDX_Table ReadinessScopes=arn:aws:aws-recovery-
readiness::111122223333:cell/IADCell,ResourceArn=arn:aws:dynamodb:us-
east-1:111122223333:table/IAD_Table --tags Stage=Prod
```

```
aws route53-recovery-readiness --region us-west-2 create-readiness-
check --readiness-check-name dynamodb_readiness_check --resource-set-name
dynamodb_resource_set --tags Stage=Prod
```

For more information, see TagResource in the *Recovery Readiness API Reference Guide* for Amazon Application Recovery Controller (ARC).

Pricing for readiness check in ARC

With Amazon Application Recovery Controller (ARC), you only pay for what you configure to use in the service. For readiness check, you pay an hourly cost per readiness check that you configure.

For detailed pricing information for ARC and pricing examples, see Amazon Application Recovery Controller (ARC) Pricing and scroll down to Amazon Application Recovery Controller (ARC).

Set up a resilient recovery process for your application

To use Amazon Application Recovery Controller (ARC) with AWS applications that are in multiple AWS Regions, there are guidelines to follow to set up your applications for resilience, so that you can support recovery readiness effectively. Then, you can create readiness checks for your

application and set up routing controls to reroute traffic for failover. You can also review the recommendations ARC provides to about your application's architecture that can improve resiliency.

> **Note**
>
> If you have an application that is siloed by Availability Zones, consider using zonal shift or zonal autoshift for failover recovery. No setup is required to use zonal shift or zonal autoshift to reliably recover applications from Availability Zone impairments.
> To move traffic away from an Availability Zone for load balancer resources, start a zonal shift in the ARC console or in the Elastic Load Balancing console. Or, you can use the AWS Command Line Interface or AWS SDK with zonal shift API actions. For more information, see Zonal shift in Amazon Application Recovery Controller (ARC).

To learn more about getting started with resilient failover configurations, see Getting started with multi-Region recovery in Amazon Application Recovery Controller (ARC).

Best practices for readiness check in ARC

We recommend the following best practice for readiness check in Amazon Application Recovery Controller (ARC).

Add notifications for readiness status changes

Set a rule in Amazon EventBridge to send a notification whenever a readiness check status changes, for example, from `READY` to `NOT READY`. When you receive a notification, you can investigate and address the issue, to make sure that your application and resources are ready for failover when you expect them to be.

You can set EventBridge rules to send notifications for several readiness check status changes, including for your recovery group (for your application), for a cell (such as an AWS Region), or for a readiness check for a resource set.

For more information, see Using readiness check in ARC with Amazon EventBridge.

Readiness check API operations

The following table lists ARC operations that you can use for recovery readiness (readiness check), with links to relevant documentation.

For examples of how to use common recovery readiness API operations with the AWS Command Line Interface, see Examples of using ARC readiness check API operations with the AWS CLI.

Action	Using the ARC console	Using the ARC API
Create a cell	See Creating, updating, and deleting recovery groups in ARC	See CreateCell
Get a cell	See Creating, updating, and deleting recovery groups in ARC	See GetCell
Delete a cell	See Creating, updating, and deleting recovery groups in ARC	See DeleteCell
Update a cell	N/A	See UpdateCell
List cells for an account	See Creating, updating, and deleting recovery groups in ARC	See ListCells
Create a recovery group	See Creating, updating, and deleting recovery groups in ARC	See CreateRecoveryGroup
Get a recovery group	See Creating, updating, and deleting recovery groups in ARC	See GetRecoveryGroup
Update a recovery group	See Creating, updating, and deleting recovery groups in ARC	See UpdateRecoveryGroup
Delete a recovery group	See Creating, updating, and deleting recovery groups in ARC	See DeleteRecoveryGroup

Action	Using the ARC console	Using the ARC API
List recovery groups	See Creating, updating, and deleting recovery groups in ARC	See ListRecoveryGroups
Create a resource set	See Creating and updating readiness checks in ARC	See CreateResourceSet
Get a resource set	See Creating and updating readiness checks in ARC	See GetResourceSet
Update a resource set	See Creating and updating readiness checks in ARC	See UpdateResourceSet
Delete a resource set	See Creating and updating readiness checks in ARC	See DeleteResourceSet
List resource sets	See Creating and updating readiness checks in ARC	See ListResourceSets
Create a readiness check	See Creating and updating readiness checks in ARC	See CreateReadinessCheck
Get a readiness check	See Creating and updating readiness checks in ARC	See GetReadinessCheck
Update a readiness check	See Creating and updating readiness checks in ARC	See UpdateReadinessCheck
Delete a readiness check	See Creating and updating readiness checks in ARC	See DeleteReadinessCheck
List readiness checks	See Creating and updating readiness checks in ARC	See ListReadinessChecks
List readiness rules	See Readiness rules descriptions in ARC	See ListRules

Action	Using the ARC console	Using the ARC API
Check status of an entire readiness check	See Monitoring readiness status in ARC	See GetReadinessCheckStatus
Check status of a resource	See Monitoring readiness status in ARC	See GetReadinessCheckResourceStatus
Check status of a cell	See Monitoring readiness status in ARC	See GetCellReadinessSummary
Check status of a recovery group	See Monitoring readiness status in ARC	See GetRecoveryGroupReadinessSummary

Examples of using ARC readiness check API operations with the AWS CLI

This section walks through simple application examples, using the AWS Command Line Interface to work with readiness check features in Amazon Application Recovery Controller (ARC) using API operations. The examples are intended to help you develop a basic understanding of how to work with readiness check capabilities using the CLI.

Readiness check in ARC audits for mismatches for the resources in your application replicas. To set up readiness checks for your application, you must set up—or model—your application resources in ARC *cells* that align with the replicas that you've created for your application. You then set up readiness checks that audit these replicas, to help you make sure that your standby application replica and its resources match your production replica, on an ongoing basis

Let's look at a simple case where you have an application named Simple-Service that currently runs in the US East (N. Virginia) Region (us-east-1). You also have a standby copy of the application in the US West (Oregon) Region (us-west-2). In this example, we'll configure readiness checks to compare these two versions of the application. This lets us ensure that the standby, US West (Oregon) Region, is ready to receive traffic, if it needs to in a failover scenario.

For more information about using the AWS CLI, see the AWS CLI Command Reference. For a list of readiness API actions and links to more information, see Readiness check API operations.

Cells in ARC represent fault boundaries (like Availability Zones or Regions) and are collected into *recovery groups*. A recovery group represents an application that you want to check failover readiness for. For more information about the components of readiness check, see Readiness check components .

> **Note**
>
> ARC is a global service that supports endpoints in multiple AWS Regions but you must specify the US West (Oregon) Region (that is, specify the parameter `--region us-west-2`) in most ARC CLI commands. For example, to create resources such as recovery groups or readiness checks.

For our application example, we'll start by creating one cell for each Region where we have resources. Then we'll create a recovery group, and then complete the setup for a readiness check.

1. Create cells

1a. Create a us-east-1 cell.

```
aws route53-recovery-readiness --region us-west-2 create-cell \
    --cell-name east-cell
```

```
{
    "CellArn": "arn:aws:route53-recovery-readiness::111122223333:cell/east-cell",
    "CellName": "east-cell",
    "Cells": [],
    "ParentReadinessScopes": [],
    "Tags": {}
}
```

1b. Create a us-west-1 cell.

```
aws route53-recovery-readiness --region us-west-2 create-cell \
    --cell-name west-cell
```

```
{
    "CellArn": "arn:aws:route53-recovery-readiness::111122223333:cell/west-cell",
    "CellName": "west-cell",
    "Cells": [],
```

```
    "ParentReadinessScopes": [],
    "Tags": {}
}
```

1c. Now we have two cells. You can verify that they exist by calling the `list-cells` API.

```
aws route53-recovery-readiness --region us-west-2 list-cells
```

```
{
    "Cells": [
        {
            "CellArn": "arn:aws:route53-recovery-readiness::111122223333:cell/east-
cell",
            "CellName": "east-cell",
            "Cells": [],
            "ParentReadinessScopes": [],
            "Tags": {}
        },
        {
            "CellArn": "arn:aws:route53-recovery-readiness::111122223333:cell/west-
cell",
            "CellName": "west-cell"
            "Cells": [],
            "ParentReadinessScopes": [],
            "Tags": {}
        }
    ]
}
```

2. Create a recovery group

Recovery groups are the top-level resource for recovery readiness in ARC. A recovery group represents an application as a whole. In this step, we'll create a recovery group to model an overall application, and then add the two cells that we created.

2a. Create a recovery group.

```
aws route53-recovery-readiness --region us-west-2 create-recovery-group \
    --recovery-group-name simple-service-recovery-group \
       --cells "arn:aws:route53-recovery-readiness::111122223333:cell/east-cell"\
       "arn:aws:route53-recovery-readiness::111122223333:cell/west-cell"
```

```
{
    "Cells": [],
    "RecoveryGroupArn": "arn:aws:route53-recovery-readiness::111122223333:recovery-
group/simple-service-recovery-group",
    "RecoveryGroupName": "simple-service-recovery-group",
    "Tags": {}
}
```

2b. (Optional) You can verify that your recovery group was created correctly by calling the `list-recovery-groups` API.

```
aws route53-recovery-readiness --region us-west-2 list-recovery-groups
```

```
{
    "RecoveryGroups": [
        {
            "Cells": [
                "arn:aws:route53-recovery-readiness::111122223333:cell/east-cell",
                "arn:aws:route53-recovery-readiness::111122223333:cell/west-cell"
            ],
            "RecoveryGroupArn": "arn:aws:route53-recovery-
readiness::111122223333:recovery-group/simple-service-recovery-group",
            "RecoveryGroupName": "simple-service-recovery-group",
            "Tags": {}
        }
    ]
}
```

Now that we have a model for our application, let's add the resources to be monitored. In ARC, a group of resources that you want to monitor is called a resource set. Resource sets contain resources that are all of the same type. We compare the resources in a resource set to each other to help determine a cell's readiness for failover.

3. Create a resource set

Let's assume our Simple-Service application is indeed very simple and only uses DynamoDB tables. It has a DynamoDB table in us-east-1 and another one in us-west-2. A resource set also contains a readiness scope, which identifies the cell that each resource is contained in.

3a. Create a resource set that reflects our Simple-Service application's resources.

```
aws route53-recovery-readiness --region us-west-2 create-resource-set \
    --resource-set-name ImportantInformationTables \
    --resource-set-type AWS::DynamoDB::Table \
    --resources
    ResourceArn="arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsWest2",ReadinessScopes="arn:aws:route53-recovery-readiness::111122223333:cell/
west-cell"
    ResourceArn="arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsEast1",ReadinessScopes="arn:aws:route53-recovery-readiness::111122223333:cell/
east-cell"
```

```
{
    "ResourceSetArn": "arn:aws:route53-recovery-readiness::111122223333:resource-set/
sample-resource-set",
    "ResourceSetName": "ImportantInformationTables",
    "Resources": [
        {
            "ReadinessScopes": [
                "arn:aws:route53-recovery-readiness::111122223333:cell/west-cell"
            ],
            "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsWest2"
        },
        {
            "ReadinessScopes": [
                "arn:aws:route53-recovery-readiness::111122223333:cell/east-cell"
            ],
            "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsEast1"
        }
    ],
    "Tags": {}
}
```

3b. (Optional) You can verify what's included in the resource set by calling the `list-resource-sets` API. This lists all the resource sets for an AWS account. Here you can see that we have just the one resource set that we created above.

```
aws route53-recovery-readiness --region us-west-2 list-resource-sets
```

```
{
    "ResourceSets": [
        {
            "ResourceSetArn": "arn:aws:route53-recovery-
readiness::111122223333:resource-set/ImportantInformationTables",
            "ResourceSetName": "ImportantInformationTables",
            "Resources": [
                {
                    "ReadinessScopes": [
                        "arn:aws:route53-recovery-readiness::111122223333:cell/west-
cell"
                    ],
                    "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsWest2"
                },
                {
                    "ReadinessScopes": [
                        "arn:aws:route53-recovery-readiness::111122223333:cell/east-
cell"
                    ],
                    "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsEast1"
                }
            ],
            "Tags": {}
        }
    ]
}{
    "ResourceSets": [
        {
            "ResourceSetArn": "arn:aws:route53-recovery-
readiness::111122223333:resource-set/ImportantInformationTables",
            "ResourceSetName": "ImportantInformationTables",
            "Resources": [
                {
                    "ReadinessScopes": [
                        "arn:aws:route53-recovery-readiness::111122223333:cell/west-
cell"
                    ],
                    "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsWest2"
                },
                {
```

```
                    "ReadinessScopes": [
                        "arn:aws:route53-recovery-
readiness::&ExampleAWSAccountNo1;:cell/east-cell"
                    ],
                    "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsEast1"
                }
            ],
            "Tags": {}
        }
    ]
}
```

Now we've created the cells, recovery group, and resource set to model the Simple-Service application in ARC. Next, we'll set up readiness checks to monitor the readiness of the resources for fail over.

4. Create a readiness check

A readiness check applies a set of rules to each resource in the resource set that is attached to the check. Rules are specific to each resource type. That is, there are different rules for `AWS::DynamoDB::Table`, `AWS::EC2::Instance`, and so on. Rules check a variety of dimensions for a resource, including configuration, capacity (where available and applicable), limits (where available and applicable), and routing configurations.

> **Note**
>
> To see the rules that are applied to a resource in a readiness check, you can use the `get-readiness-check-resource-status` API, as described in step 5. To see a list of all the readiness rules in ARC, use `list-rules` or see Readiness rules descriptions in ARC. ARC has a specific set of rules that it runs for each resource type; they're not customizable at this time.

4a. Create a readiness check for the resource set, ImportantInformationTables.

```
aws route53-recovery-readiness --region us-west-2 create-readiness-check \
    --readiness-check-name ImportantInformationTableCheck --resource-set-name
 ImportantInformationTables
```

```
{
    "ReadinessCheckArn": "arn:aws:route53-recovery-readiness::111122223333:readiness-
check/ImportantInformationTableCheck",
    "ReadinessCheckName": "ImportantInformationTableCheck",
    "ResourceSet": "ImportantInformationTables",
    "Tags": {}
}
```

4b. (Optional) To verify that the readiness check was created successfully, run the `list-readiness-checks` API. This API shows all the readiness checks in an account.

```
aws route53-recovery-readiness --region us-west-2 list-readiness-checks
```

```
{
    "ReadinessChecks": [
        {
            "ReadinessCheckArn": "arn:aws:route53-recovery-
readiness::111122223333:readiness-check/ImportantInformationTableCheck",
            "ReadinessCheckName": "ImportantInformationTableCheck",
            "ResourceSet": "ImportantInformationTables",
            "Tags": {}
        }
    ]
}
```

5. Monitor readiness checks

Now that we've modeled the application and added a readiness check, we're ready to monitor resources. You can model the readiness of your application at four levels: the readiness check level (a group of resources), the individual resource level, the cell level (all the resources in an Availability Zone or Region), and the recovery group level (the application as a whole). Commands for getting each of these types of readiness statuses are provided below.

5a. See the status of your readiness check.

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-status\
    --readiness-check-name ImportantInformationTableCheck
```

```
{
    "Readiness": "READY",
```

```
    "Resources": [
        {
            "LastCheckedTimestamp": "2021-01-07T00:53:39Z",
            "Readiness": "READY",
            "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsWest2"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:53:39Z",
            "Readiness": "READY",
            "ResourceArn": "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsEast2"
    ]
}
```

5b. See the detailed readiness status of a single resource in a readiness check, including the status of each rule that is checked.

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-resource-status \
    --readiness-check-name ImportantInformationTableCheck \
    --resource-identifier "arn:aws:dynamodb:us-west-2:111122223333:table/
TableInUsWest2"
```

```
{"Readiness": "READY",
    "Rules": [
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoTableStatus"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoCapacity"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoPeakRcuWcu"
        },
```

```
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoGSIsPeakRcuWcu"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoGSIsConfig"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoGSIsStatus"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoGSIsCapacity"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoReplicationLatency"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoAutoScalingConfiguration"
        },
        {
            "LastCheckedTimestamp": "2021-01-07T00:55:41Z",
            "Messages": [],
            "Readiness": "READY",
            "RuleId": "DynamoLimits"
        }
    ]
```

```
}
```

5c. See the overall readiness for a cell.

```
aws route53-recovery-readiness --region us-west-2 get-cell-readiness-summary \
    --cell-name west-cell
```

```
{
    "Readiness": "READY",
    "ReadinessChecks": [
        {
            "Readiness": "READY",
            "ReadinessCheckName": "ImportantTableCheck"
        }
    ]
}
```

5d. Finally, see the top-level readiness of your application, at the recovery group level.

```
aws route53-recovery-readiness --region us-west-2 get-recovery-group-readiness-summary
 \
    --recovery-group-name simple-service-recovery-group
```

```
{
    "Readiness": "READY",
    "ReadinessChecks": [
        {
            "Readiness": "READY",
            "ReadinessCheckName": "ImportantTableCheck"
        }
    ]
}
```

Working with recovery groups and readiness checks

This section describes and provides procedures for recovery groups and readiness checks, including creating, updating, and deleting these resources.

Creating, updating, and deleting recovery groups in ARC

A recovery group represents your application in Amazon Application Recovery Controller (ARC). It typically consists of two or more *cells* that are replicas of each other in terms of resources and functionality, so that you can fail over from one to the other. Each cell includes the Amazon Resource Names (ARNs) for the active resources for one AWS Region or Availability Zone. The resources might be an Elastic Load Balancing load balancer, an Auto Scaling group, or other resources. A corresponding cell representing another zone or Region has standby resources of the same type that are in your active cell – a load balancer, Auto Scaling group, and so on.

A cell represents replicas of your application. Readiness checks in ARC help you determine if your application is ready to fail over from one replica to another. However, you should make decisions about whether to fail away from or to a replica based on your monitoring and health check systems, and consider readiness checks as a complementary service to those systems.

Readiness checks audit resources to determine their readiness based on a set of pre-defined rules for that type of resource. After you create your recovery group with the replicas, you add ARC readiness checks for the resources in your application, so ARC can help make sure that the replicas have the same setup and configuration over time.

Topics

Creating recovery groups

The steps in this section explain how to create a recovery group on the ARC console. To learn about using recovery readiness API operations with Amazon Application Recovery Controller (ARC), see Readiness check API operations.

To create a recovery group

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. On the **Recovery readiness** page, choose **Create**, and then choose a **Recovery group**.
4. Enter a name for your recovery group, and then choose **Next**.

5. Choose **Create cells**, and then choose **Add cell**.
6. Enter a name for the cell. For example, if you have an application replica in US West (N. California), you could add a cell named `MyApp-us-west-1`.
7. Choose **Add cell**, and add a name for a second cell. For example, if you have a replica in US East (Ohio), you could add a cell named `MyApp-us-east-2`.
8. If you want to add nested cells (replicas in Availability Zones within Regions), choose **Action**, choose **Add nested cell**, and then enter a name.
9. When you've added all of the cells and nested cells for your application replicas, choose **Next**.
10. Review your recovery group, and then choose **Create recovery group**.

Updating and deleting recovery groups and cells

The steps in this section explain how to update and delete a recovery group, and delete a cell on the ARC console. To learn about using recovery readiness API operations with Amazon Application Recovery Controller (ARC), see Readiness check API operations.

To update or delete a recovery group, or delete a cell

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. On the **Recovery readiness** page, choose a recovery group.
4. To work with a recovery group, choose **Action**, and then choose **Edit recovery group** or **Delete recovery group**.
5. When you edit a recovery group, you can add or remove cells or nested cells.
   - To add a cell, choose **Add cell**.
   - To remove a cell, under the **Action** label next to the cell, choose **Delete cell**.

Creating and updating readiness checks in ARC

This section provides procedures for readiness checks and resource sets, including creating, updating, and deleting these resources.

Creating and updating a readiness check

The steps in this section explain how to create a readiness check on the ARC console. To learn about using recovery readiness API operations with Amazon Application Recovery Controller (ARC), see Readiness check API operations.

To update a readiness check, you can edit the resource set for the readiness check, to add or remove resources or to change the readiness scope for a resource.

To create a readiness check

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. On the **Readiness** page, choose **Create**, and then choose a **Readiness check**.
4. Enter a name for your readiness check, choose the resource type that you want to check, and then choose **Next**.
5. Add a resource set for your readiness check. A resource set is a group of resources of the same type in different replicas. Choose one of the following:

   - Create a readiness check with resources in a resource set that you've already created.
   - Create a new resource set.

   If you choose to create a new resource set, enter a name for it and choose **Add**.
6. Copy and paste Amazon Resource Names (ARNs) one by one for each resource that you want to include in the set, and then choose **Next**.

   > **Tip**
   >
   > For examples and more information about the ARN format that ARC expects for each resource type, see Resource types and ARN formats in ARC.

7. If you like, view the readiness rules that will be used when ARC checks the type of resource you included in this readiness check. Then choose **Next**.
8. (Optional) Under **Recovery group name**, choose a recovery group to associate the readiness check with and then, for each resource ARN, choose a cell (Region or Availability Zone) from

the drop-down menu that the resource is in. If it's an application-level resource, like a DNS routing policy, choose **global resource (no cell)**.

This specifies the readiness scopes for the resources in the readiness check.

> **Important**
>
> Although this step is optional, readiness scopes must be added to get summary readiness information for your recovery group and cells. If you skip this step and don't associate the readiness check with your recovery group's resources by choosing readiness scopes here, ARC cannot return summary readiness information for the recovery group or cells.

9. Choose **Next**.
10. Review the information on the confirmation page, and then choose **Create readiness check**.

To delete a readiness check

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. Choose a readiness check, and under **Actions**, choose **Delete**.

Creating and editing resource sets

Typically, you create a resource set as part of creating a readiness check, but you can create a resource set separately as well. You can also edit a resource set to add or remove resources. The steps in this section explain how to create or edit a resource set on the ARC console. To learn about using recovery readiness API operations with Amazon Application Recovery Controller (ARC), see Readiness check API operations.

To create a resource set

1. Open the Route 53 console at https://console.aws.amazon.com/route53/home.
2. Under **Application Recovery Controller**, choose **Resource sets**.
3. Choose **Create**.
4. Enter a name for the resource set, and then choose the type of resource to include in the set.

5. Choose **Add**, and then enter the Amazon Resource Name (ARN) for the resource to add to the set.
6. After you've finished adding resources, choose **Create resource set**.

To edit a resource set

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. Under **Resource sets**, choose **Action**, and then choose **Edit**.
4. Do one of the following:
   - To remove a resource from the set, choose **Remove**.
   - To add a resource to the set, choose **Add**, and then enter the Amazon Resource Name (ARN) for the resource.
5. You can also edit the readiness scope for the resource, to associate the resource with a different cell for the readiness check.
6. Choose **Save**.

Monitoring readiness status in ARC

You can see readiness for your application in Amazon Application Recovery Controller (ARC) at the following levels:

- The readiness check level for the resources in a resource set
- The individual resource level
- The cell (application replica) level for all the resources in an Availability Zone or AWS Region
- The recovery group level for the application as a whole

You can be notified about readiness status changes, or you can monitor readiness status changes in the Route 53 console or by using ARC CLI commands.

Readiness status notification

You can use Amazon EventBridge to set up event-driven rules to monitor ARC resources and notify you about changes in readiness status. For more information, see Using readiness check in ARC with Amazon EventBridge.

Monitoring readiness status in the ARC console

The following procedure describes how to monitor recovery readiness in the AWS Management Console.

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. On the **Readiness** page, under **Recovery group**, view the **Recovery group readiness status** for each recovery group (application).

   You can also view the readiness of specific cells or individual resources.

Monitoring readiness status by using CLI commands

This section provides examples of AWS CLI commands to use to see the readiness status for your application and resources at different levels.

Readiness for a resource set

The status of a readiness check you've created for a resource set (a group of resources).

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-status --readiness-check-name ReadinessCheckName
```

Readiness for a single resource

To get the status of a single resource in a readiness check, including the status of each readiness rule that is checked, specify the readiness check name and a resource ARN. For example:

```
aws route53-recovery-readiness --region us-west-2 get-readiness-check-status --readiness-check-name ReadinessCheckName --resource-arn "arn:aws:dynamodb:us-west-2:111122223333:table/TableName"
```

Readiness for a cell

The status of a single cell, that is, a Region or Availability Zone.

```
aws route53-recovery-readiness --region us-west-2 get-cell-readiness-
summary --cell-name CellName
```

Readiness for an application

The status of the overall application, at the recovery group level.

```
aws route53-recovery-readiness --region us-west-2 get-recovery-group-
readiness-summary --recovery-group-name RecoveryGroupName
```

Getting architecture recommendations in ARC

If you have an existing application, Amazon Application Recovery Controller (ARC) can evaluate the architecture of your application and routing policies to provide recommendations for modifying the design to improve your application's recovery resiliency. After you create a recovery group in ARC that represents your application, follow the steps in this section to get recommendations for your application's architecture.

We recommend that you specify a target resource for the DNS target resource for your recovery group, if you haven't specified one yet, so that we can provide more detailed recommendations. When you provide additional information, ARC can provide better recommendations for you. For example, if you enter an Amazon Route 53 resource record or a Network Load Balancer as a target resource, ARC can provide information about whether you've created the optimal number of cells for your recovery group.

Note the following for DNS target resources:

- Specify only a Route 53 resource record or Network Load Balancer for a target resource.
- Create only one DNS target resource for each recovery group.
- Recommended: Create one DNS target resource for each cell.
- Group the DNS target resources into one resource set with a readiness check.

The following procedure explains how to create DNS target resources and get architecture recommendations for your application.

To get recommendations for updating your architecture

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. Under **Recovery group name**, choose the recovery group that represents your application.
4. On the **Recovery group details** page, on the **Action** menu, choose **Get architecture recommendations for this recovery group**.
5. If you haven't created a DNS target resource readiness check yet, create one so that ARC can provide architecture recommendations. Choose **Create a DNS target resource**.

   For more information about DNS target resources, see Readiness check components .
6. To create a resource set for a DNS target resource, you create a readiness check. Enter a name for the readiness check, and then, for the type of readiness check, choose **DNS target resource**.
7. Enter a name for the resource set.
8. Enter the attributes for your application, including the DNS name, hosted zone ARN, and record set ID.

   > **Tip**
   >
   > To see the format for a hosted zone ARN, see **ARN format for hosted zone** in Resource types and ARN formats in ARC.

   Optionally, but strongly recommended, choose **Add optional attribute** and provide a Network Load Balancer ARN or your domain's Route 53 resource record.
9. (Optional) In **Recovery group configuration**, choose a cell for your DNS target resource, to set the readiness scope.
10. Choose **Create resource set**.
11. On the **Recovery group** details page, choose **Get architecture recommendations**. ARC displays a set of recommendations on the page.

Review the list of recommendations. Then you can decide whether and how to make changes to improve your app's recovery resilience.

Creating cross-account authorizations in ARC

You might have your resources distributed across multiple AWS accounts, which can make it challenging to get a comprehensive view of your application's health. It can also make it hard to get the information required to make quick decisions. To help streamline this for readiness check in Amazon Application Recovery Controller (ARC), you can use *cross-account authorization*.

Cross-account authorization in ARC works with the readiness check feature. With cross-account authorization, you can use one central AWS account to monitor your resources that are located in multiple AWS accounts. In each account that has resources that you want to monitor, you authorize the central account to have access to those resources. Then the central account can create readiness checks for the resources in all the accounts and from the central account, you can monitor readiness for failover.

> **Note**
>
> Cross-account authorization setup isn't available in the console. Instead, use ARC API operations to set up and work with cross-account authorization. To help you get started, this section provides AWS CLI command examples.

Let's say that an application has an account that has resources in the US West (Oregon) Region (us-west-2), and there's also an account that has resources that you'd like to monitor in the US East (N. Virginia) Region (us-east-1). ARC can allow access for you to monitor both sets of resources from one account, us-west-2, by using cross-account authorization.

For example, let's say that you have the following AWS accounts:

- US-West account: 999999999999
- US-East account: 111111111111

In the us-east-1 account (111111111111), we can enable cross-account authorization to allow access by the us-west-2 account (999999999999) by specifying the Amazon Resource Name (ARN) for the (root) user in the us-west-2 IAM account: `arn:aws:iam::999999999999:root`. After we create the authorization, the us-west-2 account can add resources owned by us-east-1 to resource sets and create readiness checks to run on the resource sets.

The following example illustrates setting up cross-account authorization for one account. You must enable cross-account authorization in each additional account that has AWS resources that you want to add and monitor in ARC.

> **Note**
>
> ARC is a global service that supports endpoints in multiple AWS Regions but you must specify the US West (Oregon) Region (that is, specify the parameter `--region us-west-2`) in most ARC CLI commands.

The following AWS CLI command shows how to set up cross-account authorization for this example:

```
aws route53-recovery-readiness --region us-west-2 --profile profile-in-us-east-1-account \
    create-cross-account-authorization --cross-account-authorization
 arn:aws:iam::999999999999:root
```

To disable this authorization, do the following:

```
aws route53-recovery-readiness --region us-west-2 --profile profile-in-us-east-1-account \
    delete-cross-account-authorization --cross-account-authorization
 arn:aws:iam::999999999999:root
```

To check in a specific account for all the accounts that you've provided cross-account authorization for, use the `list-cross-account-authorizations` command. Note that at this time, you can't check in the other direction. That is, there isn't an API operation that you can use with an account profile to list all of the accounts for which it has been granted cross-account authorization to add and monitor resources.

```
aws route53-recovery-readiness --region us-west-2 --profile profile-in-us-east-1-account \
    list-cross-account-authorizations
```

```
{
    "CrossAccountAuthorizations": [
```

```
        "arn:aws:iam::999999999999:root"
    ]
}
```

Readiness rules, resource types, and ARNS

This section includes reference information about the readiness rules descriptions, and supported resource types and the format for Amazon Resource Names (ARNs) that you use for resource sets.

Readiness rules descriptions in ARC

This section lists the readiness rules descriptions for all the types of resources supported by Amazon Application Recovery Controller (ARC). To see a list of the resource types supported by ARC, see Resource types and ARN formats in ARC.

You can also view the readiness rules descriptions on the ARC console or by using an API operation, by doing the following:

- To view readiness rules in the console, follow the steps in the following procedure: View readiness rules on the console.
- To view readiness rules by using the API, see the ListRules operation.

Topics

Readiness rules in ARC

This section lists the set of readiness rules for each resource type that is supported by ARC.

As you look through the rule descriptions, you can see that most of them include the terms **Inspects all** or **Inspects each**. To understand how these terms explain how a rule works in the context of a readiness check, and other details about how ARC sets readiness status, see How readiness rules determine readiness status.

Readiness rules

ARC audits resources by using the following readiness rules.

Amazon API Gateway Version 1 stages

- **ApiGwV1ApiKeyCount**: Inspects all API Gateway stages to ensure that they have the same number of API Keys linked to them.
- **ApiGwV1ApiKeySource**: Inspects all API Gateway stages to ensure that they have the same value for `API Key Source`.
- **ApiGwV1BasePath**: Inspects all API Gateway stages to ensure that they are linked to the same base path.
- **ApiGwV1BinaryMediaTypes**: Inspects all API Gateway stages to ensure that they support the same binary media types.
- **ApiGwV1CacheClusterEnabled**: Inspects all API Gateway stages to ensure that either all have `Cache Cluster` enabled, or none do.
- **ApiGwV1CacheClusterSize**: Inspects all API Gateway stages to ensure that they have the same `Cache Cluster Size`. If one has a greater value, the others are marked NOT READY.
- **ApiGwV1CacheClusterStatus**: Inspects all API Gateway stages to ensure that the `Cache Cluster` is in the AVAILABLE state.
- **ApiGwV1DisableExecuteApiEndpoint**: Inspects all API Gateway stages to ensure that either all have `Execute API Endpoint` disabled, or none do.
- **ApiGwV1DomainName**: Inspects all API Gateway stages to ensure that they are linked to the same domain name.
- **ApiGwV1EndpointConfiguration**: Inspects all API Gateway stages to ensure that they are linked to a domain with the same endpoint configuration.
- **ApiGwV1EndpointDomainNameStatus**: Inspects all API Gateway stages to ensure that the domain name that they are linked to is in the AVAILABLE state.
- **ApiGwV1MethodSettings**: Inspects all API Gateway stages to ensure that they have the same value for `Method Settings`.
- **ApiGwV1MutualTlsAuthentication**: Inspects all API Gateway stages to ensure that they have the same value for `Mutual TLS Authentication`.
- **ApiGwV1Policy**: Inspects all API Gateway stages to ensure that either all use API level policies, or none do.
- **ApiGwV1RegionalDomainName**: Inspects all API Gateway stages to ensure that they are linked to the same Regional domain name. Note: This rule does not affect readiness status.
- **ApiGwV1ResourceMethodConfigs**: Inspects all API Gateway stages to ensure that they have a similar resource hierarchy, including the related configurations.

- **ApiGwV1SecurityPolicy**: Inspects all API Gateway stages to ensure that they have the same value for `Security Policy`.
- **ApiGwV1Quotas**: Inspects all API Gateway groups to ensure that they conform to quotas (limits) that are managed by Service Quotas.
- **ApiGwV1UsagePlans**: Inspects all API Gateway stages to ensure that they are linked to `Usage Plans` with the same configuration.

Amazon API Gateway Version 2 stages

- **ApiGwV2ApiKeySelectionExpression**: Inspects all API Gateway stages ensure that they have the same value for `API Key Selection Expression`.
- **ApiGwV2ApiMappingSelectionExpression**: Inspects all API Gateway stages to ensure that they have the same value for `API Mapping Selection Expression`.
- **ApiGwV2CorsConfiguration**: Inspects all API Gateway stages to ensure that they have the same CORS related configuration.
- **ApiGwV2DomainName**: Inspects all API Gateway stages to ensure that they are linked to the same domain name.
- **ApiGwV2DomainNameStatus**: Inspects all API Gateway stages to ensure that the domain name is in the AVAILABLE state.
- **ApiGwV2EndpointType**: Inspects all API Gateway stages to ensure that they have the same value for `Endpoint Type`.
- **ApiGwV2Quotas**: Inspects all API Gateway groups to ensure that they conform to quotas (limits) that are managed by Service Quotas.
- **ApiGwV2MutualTlsAuthentication**: Inspects all API Gateway stages to ensure that they have the same value for `Mutual TLS Authentication`.
- **ApiGwV2ProtocolType**: Inspects all API Gateway stages to ensure that they have the same value for `Protocol Type`.
- **ApiGwV2RouteConfigs**: Inspects all API Gateway stages to ensure that they have the same hierarchy of routes with the same configuration.
- **ApiGwV2RouteSelectionExpression**: Inspects all API Gateway stages to ensure that they have the same value for `Route Selection Expression`.
- **ApiGwV2RouteSettings**: Inspects all API Gateway stages to ensure that they have the same value for `Default Route Settings`.
- **ApiGwV2SecurityPolicy**: Inspects all API Gateway stages to ensure that they have the same value for `Security Policy`.

- **ApiGwV2StageVariables**: Inspects all API Gateway stages to ensure that they all have the same `Stage Variables` as the other stages.
- **ApiGwV2ThrottlingBurstLimit**: Inspects all API Gateway stages to ensure that they have the same value for `Throttling Burst Limit`.
- **ApiGwV2ThrottlingRateLimit**: Inspects all API Gateway stages to ensure that they have the same value for `Throttling Rate Limit`.

Amazon Aurora clusters

- **RdsClusterStatus**: Inspects each Aurora cluster to ensure that it has a status of either `AVAILABLE` or `BACKING-UP`.
- **RdsEngineMode**: Inspects all Aurora clusters to ensure that they have the same value for `Engine Mode`.
- **RdsEngineVersion**: Inspects all Aurora clusters to ensure that they have the same value for `Major Version`.
- **RdsGlobalReplicaLag**: Inspects each Aurora cluster to ensure that it has a `Global Replica Lag` of less than 30 seconds.
- **RdsNormalizedCapacity**: Inspects all Aurora clusters to ensure that they have a normalized capacity within 15% of the maximum in the resource set.
- **RdsInstanceType**: Inspects all Aurora clusters to ensure that they have the same instance types.
- **RdsQuotas**: Inspects all Aurora clusters to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Auto Scaling groups

- **AsgMinSizeAndMaxSize**: Inspects all Auto Scaling groups to ensure that they have the same minimum and maximum group sizes.
- **AsgAZCount**: Inspects all Auto Scaling groups to ensure that they have the same number of Availability Zones.
- **AsgInstanceTypes**: Inspects all Auto Scaling groups to ensure that they have the same instance types. Note: This rule does not affect readiness status.
- **AsgInstanceSizes**: Inspects all Auto Scaling groups to ensure that they have the same instance sizes.
- **AsgNormalizedCapacity**: Inspects all Auto Scaling groups to ensure that they have a normalized capacity within 15% of the maximum in the resource set.

- **AsgQuotas**: Inspects all Auto Scaling groups to ensure that they conform to quotas (limits) that are managed by Service Quotas.

CloudWatch alarms

- **CloudWatchAlarmState**: Inspects CloudWatch alarms to ensure that each is not in the `ALARM` or `INSUFFICIENT_DATA` state.

Customer gateways

- **CustomerGatewayIpAddress**: Inspects all customer gateways to ensure that they have the same IP address.
- **CustomerGatewayState**: Inspects customer gateways to ensure that each is in the `AVAILABLE` state.
- **CustomerGatewayVPNType**: Inspects all customer gateways to ensure that they have the same VPN type.

DNS target resources

- **DnsTargetResourceHostedZoneConfigurationRule**: Inspects all DNS target resources to ensure that they have the same Amazon Route 53 hosted zone ID and that each hosted zone is not private. Note: This rule does not affect readiness status.
- **DnsTargetResourceRecordSetConfigurationRule**: Inspects all DNS target resources to ensure that they have the same resource record cache time to live (TTL) and that the TTLs are less than or equal to 300.
- **DnsTargetResourceRoutingRule**: Inspects each DNS target resource associated with an alias resource record set to ensure that it routes traffic to the DNS name configured on the target resource. Note: This rule does not affect readiness status.
- **DnsTargetResourceHealthCheckRule**: Inspects all DNS target resources to ensure that health checks are associated with their resource record sets when appropriate and not otherwise. Note: This rule does not affect readiness status.

Amazon DynamoDB tables

- **DynamoConfiguration**: Inspects all DynamoDB tables to ensure that they have the same keys, attributes, server-side encryption, and streams configurations.
- **DynamoTableStatus**: Inspects each DynamoDB table to ensure that it has a status of ACTIVE.
- **DynamoCapacity**: Inspects all DynamoDB tables to ensure that their provisioned read capacities and write capacities are within 20% of the maximum capacities in the resource set.
- **DynamoPeakRcuWcu**: Inspects each DynamoDB table to ensure that it has had similar peak traffic to the other tables, to assure provisioned capacity.

- **DynamoGsiPeakRcuWcu**: Inspects each DynamoDB table to ensure that it has had similar maximum read and write capacity to the other tables, to assure provisioned capacity.
- **DynamoGsiConfig**: Inspects all DynamoDB tables that have global secondary indexes to ensure that the tables use the same index, key schema, and projection.
- **DynamoGsiStatus**: Inspects all DynamoDB tables that have global secondary indexes to ensure that the global secondary indexes have an ACTIVE status.
- **DynamoGsiCapacity**: Inspects all DynamoDB tables that have global secondary indexes to ensure that the tables have provisioned GSI read capacities and GSI write capacities within 20% of the maximum capacities in the resource set.
- **DynamoReplicationLatency**: Inspects all DynamoDB tables that are global tables to ensure that they have the same replication latency.
- **DynamoAutoScalingConfiguration**: Inspects all DynamoDB tables that have Auto Scaling enabled to ensure that they have the same minimum, maximum, and target read and write capacities.
- **DynamoQuotas**: Inspects all DynamoDB tables to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Elastic Load Balancing (Classic Load Balancers)

- **ElbV1CheckAzCount**: Inspects each Classic Load Balancer to ensure that it is attached to only one Availability Zone. Note: This rule does not affect readiness status.
- **ElbV1AnyInstances**: Inspects all Classic Load Balancers to ensure that they have at least one EC2 instance.
- **ElbV1AnyInstancesHealthy**: Inspects all Classic Load Balancers to ensure that they have at least one healthy EC2 instance.
- **ElbV1Scheme**: Inspects all Classic Load Balancers to ensure that they have the same load balancer scheme.
- **ElbV1HealthCheckThreshold**: Inspects all Classic Load Balancers to ensure that they have the same health check threshold value.
- **ElbV1HealthCheckInterval**: Inspects all Classic Load Balancers to ensure that they have the same health check interval value.
- **ElbV1CrossZoneRoutingEnabled**: Inspects all Classic Load Balancers to ensure that they have the same value for cross-zone load balancing (ENABLED or DISABLED).
- **ElbV1AccessLogsEnabledAttribute**: Inspects all Classic Load Balancers to ensure that they have the same value for access logs (ENABLED or DISABLED).

- **ElbV1ConnectionDrainingEnabledAttribute**: Inspects all Classic Load Balancers to ensure that they have the same value for connection draining (ENABLED or DISABLED).
- **ElbV1ConnectionDrainingTimeoutAttribute**: Inspects all Classic Load Balancers to ensure that they have the same connection draining timeout value.
- **ElbV1IdleTimeoutAttribute**: Inspects all Classic Load Balancers to ensure that they have the same value for idle timeout.
- **ElbV1ProvisionedCapacityLcuCount**: Inspects all Classic Load Balancers with a provisioned LCU greater than 10 to ensure that they are within 20% of the highest provisioned LCU in the resource set.
- **ElbV1ProvisionedCapacityStatus**: Inspects the provisioned capacity status on each Classic Load Balancer to ensure that it does not have a value of DISABLED or PENDING.

Amazon EBS volumes

- **EbsVolumeEncryption**: Inspects all EBS volumes to ensure that they have the same value for encryption (ENABLED or DISABLED).
- **EbsVolumeEncryptionDefault**: Inspects all EBS volumes to ensure that they have the same value for encryption by default (ENABLED or DISABLED).
- **EbsVolumeIops**: Inspects all EBS volumes to ensure that they have the same input/output operations per second (IOPS).
- **EbsVolumeKmsKeyId**: Inspects all EBS volumes to ensure that they have the same default AWS KMS key ID.
- **EbsVolumeMultiAttach**: Inspects all EBS volumes to ensure that they have the same value for multi-attach (ENABLED or DISABLED).
- **EbsVolumeQuotas**: Inspects all EBS volumes to ensure that they conform to quotas (limits) that are set by Service Quotas.
- **EbsVolumeSize**: Inspects all EBS volumes to ensure that they have the same readable size.
- **EbsVolumeState**: Inspects all EBS volumes to ensure that they have the same volume state.
- **EbsVolumeType**: Inspects all EBS volumes to ensure that they have the same volume type.

AWS Lambda functions

- **LambdaMemorySize**: Inspects all Lambda functions to ensure that they have the same memory size. If one has more memory, the others are marked `NOT READY`.
- **LambdaFunctionTimeout**: Inspects all Lambda functions to ensure that they have the same timeout value. If one has a greater value, the others are marked `NOT READY`.

- **LambdaFunctionRuntime**: Inspects all Lambda functions to ensure that they all have the same runtime.
- **LambdaFunctionReservedConcurrentExecutions**: Inspects all Lambda functions to ensure that they all have the same value for `Reserved Concurrent Executions`. If one has a greater value, the others are marked `NOT READY`.
- **LambdaFunctionDeadLetterConfig**: Inspects all Lambda functions to ensure that they either all have a `Dead Letter Config` defined, or that none of them do.
- **LambdaFunctionProvisionedConcurrencyConfig**: Inspects all Lambda functions to ensure that they have the same value for `Provisioned Concurrency`.
- **LambdaFunctionSecurityGroupCount**: Inspects all Lambda functions to ensure that they have the same value for `Security Groups`.
- **LambdaFunctionSubnetIdCount**: Inspects all Lambda functions to ensure that they have the same value for `Subnet Ids`.
- **LambdaFunctionEventSourceMappingMatch**: Inspects all Lambda functions to ensure that all of the chosen `Event Source Mapping` properties match between them.
- **LambdaFunctionLimitsRule**: Inspects all Lambda functions to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Network Load Balancers and Application Load Balancers

- **ElbV2CheckAzCount**: Inspects each Network Load Balancer to ensure that it is attached to only one Availability Zone. Note: This rule does not affect readiness status.
- **ElbV2TargetGroupsCanServeTraffic**: Inspects each Network Load Balancer and Application Load Balancer to ensure that it has at least one healthy Amazon EC2 instance.
- **ElbV2State**: Inspects each Network Load Balancer and Application Load Balancer to ensure that it is in the `ACTIVE` state.
- **ElbV2IpAddressType**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same IP address types.
- **ElbV2Scheme**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same scheme.
- **ElbV2Type**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same type.
- **ElbV2S3LogsEnabled**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for Amazon S3 server access logs (ENABLED or DISABLED).

- **ElbV2DeletionProtection**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for deletion protection (ENABLED or DISABLED).
- **ElbV2IdleTimeoutSeconds**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for idle time seconds.
- **ElbV2HttpDropInvalidHeaders**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for HTTP drop invalid headers.
- **ElbV2Http2Enabled**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for HTTP2 (ENABLED or DISABLED).
- **ElbV2CrossZoneEnabled**: Inspects all Network Load Balancers and Application Load Balancers to ensure that they have the same value for cross-zone load balancing (ENABLED or DISABLED).
- **ElbV2ProvisionedCapacityLcuCount**: Inspects all Network Load Balancers and Application Load Balancers with a provisioned LCU greater than 10 to ensure that they are within 20% of the highest provisioned LCU in the resource set.
- **ElbV2ProvisionedCapacityEnabled**: Inspects all Network Load Balancers and Application Load Balancers provisioned capacity status to ensure that it does not have a value of DISABLED or PENDING.

Amazon MSK clusters

- **MskClusterClientSubnet**: Inspects each MSK cluster to ensure that it has only two or only three client subnets.
- **MskClusterInstanceType**: Inspects all MSK clusters to ensure that they have the same Amazon EC2 instance type.
- **MskClusterSecurityGroups**: Inspects all MSK clusters to ensure that they have the same security groups.
- **MskClusterStorageInfo**: Inspects all MSK clusters to ensure that they have the same EBS storage volume size. If one has a greater value, the others are marked NOT READY.
- **MskClusterACMCertificate**: Inspects all MSK clusters to ensure that they have the same list of client authorization certificate ARNs.
- **MskClusterServerProperties**: Inspects all MSK clusters to ensure that they have the same value for `Current Broker Software Info`.
- **MskClusterKafkaVersion**: Inspects all MSK clusters to ensure that they have the same Kafka version.

- **MskClusterEncryptionInTransitInCluster**: Inspects all MSK clusters to ensure that they have the same value for `Encryption In Transit In Cluster`.
- **MskClusterEncryptionInClientBroker**: Inspects all MSK clusters to ensure that they have the same value for `Encryption In Transit Client Broker`.
- **MskClusterEnhancedMonitoring**: Inspects all MSK clusters to ensure that they have the same value for `Enhanced Monitoring`.
- **MskClusterOpenMonitoringInJmx**: Inspects all MSK clusters to ensure that they have the same value for `Open Monitoring JMX Exporter`.
- **MskClusterOpenMonitoringInNode**: Inspects all MSK clusters to ensure that they have the same value for `Open Monitoring Not Exporter.`
- **MskClusterLoggingInS3**: Inspects all MSK clusters to ensure that they have the same value for `Is Logging in S3`.
- **MskClusterLoggingInFirehose**: Inspects all MSK clusters to ensure that they have the same value for `Is Logging In Firehose`.
- **MskClusterLoggingInCloudWatch**: Inspects all MSK clusters to ensure that they have the same value for `Is Logging Available In CloudWatch Logs`.
- **MskClusterNumberOfBrokerNodes**: Inspects all MSK clusters to ensure they have the same value for `Number of Broker Nodes`. If one has a greater value, the others are marked NOT READY.
- **MskClusterState**: Inspects each MSK cluster to ensure that it is in an ACTIVE state.
- **MskClusterLimitsRule**: Inspects all Lambda functions to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon Route 53 health checks

- **R53HealthCheckType**: Inspects each Route 53 health check to ensure that it is not of type CALCULATED and that all checks are of the same type.
- **R53HealthCheckDisabled**: Inspects each Route 53 health check to ensure that it does not have a DISABLED state.
- **R53HealthCheckStatus**: Inspects each Route 53 health check to ensure that it has a SUCCESS status.
- **R53HealthCheckRequestInterval**: Inspects all Route 53 health checks to ensure that they all have the same value for `Request Interval`.
- **R53HealthCheckFailureThreshold**: Inspects all Route 53 health checks to ensure that they all have the same value for `Failure Threshold.`

- **R53HealthCheckEnableSNI**: Inspects all Route 53 health checks to ensure that they all have the same value for `Enable SNI.`
- **R53HealthCheckSearchString**: Inspects all Route 53 health checks to ensure that they all have the same value for `Search String.`
- **R53HealthCheckRegions**: Inspects all Route 53 health checks to ensure that they all have the same list of AWS Regions.
- **R53HealthCheckMeasureLatency**: Inspects all Route 53 health checks to ensure that they all have the same value for `Measure Latency`.
- **R53HealthCheckInsufficientDataHealthStatus**: Inspects all Route 53 health checks to ensure that they all have the same value for `Insufficient Data Health Status`.
- **R53HealthCheckInverted**: Inspects all Route 53 health checks to ensure that they are all Inverted, or are all not Inverted.
- **R53HealthCheckResourcePath**: Inspects all Route 53 health checks to ensure that they all have the same value for `Resource Path`.
- **R53HealthCheckCloudWatchAlarm**: Inspects all Route 53 health checks to ensure that the CloudWatch alarms associated with them have the same settings and configurations.

Amazon SNS subscriptions

- **SnsSubscriptionProtocol**: Inspects all SNS subscriptions to ensure that they have the same protocol.
- **SnsSubscriptionSqsLambdaEndpoint**: Inspects all SNS subscriptions that have Lambda or SQS endpoints to ensure that they have different endpoints.
- **SnsSubscriptionNonAwsEndpoint**: Inspects all SNS subscriptions that have a non-AWS service endpoint type, for example, email, to ensure that the subscriptions have the same endpoint.
- **SnsSubscriptionPendingConfirmation**: Inspects all SNS subscriptions to ensure that they have the same value for 'Pending Confirmations'.
- **SnsSubscriptionDeliveryPolicy**: Inspects all SNS subscriptions that use HTTP/S to ensure that they have the same value for 'Effective Delivery Period'.
- **SnsSubscriptionRawMessageDelivery**: Inspects all SNS subscriptions to ensure that they have the same value for 'Raw Message Delivery'.
- **SnsSubscriptionFilter**: Inspects all SNS subscriptions to ensure that they have the same value for 'Filter Policy'.

- **SnsSubscriptionRedrivePolicy**: Inspects all SNS subscriptions to ensure that they have the same value for 'Redrive Policy'.
- **SnsSubscriptionEndpointEnabled**: Inspects all SNS subscriptions to ensure that they have the same value for 'Endpoint Enabled'.
- **SnsSubscriptionLambdaEndpointValid**: Inspects all SNS subscriptions that have Lambda endpoints to ensure that they have valid Lambda endpoints.
- **SnsSubscriptionSqsEndpointValidRule**: Inspects all SNS subscriptions that use SQS endpoints to ensure that they have valid SQS endpoints.
- **SnsSubscriptionQuotas**: Inspects all SNS subscriptions to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon SNS topics

- **SnsTopicDisplayName**: Inspects all SNS topics to ensure that they have the same value for `Display Name`.
- **SnsTopicDeliveryPolicy**: Inspects all SNS topics that have HTTPS subscribers to ensure that they have the same `EffectiveDeliveryPolicy`.
- **SnsTopicSubscription**: Inspects all SNS topics to ensure that they have the same number of subscribers for each of their protocols.
- **SnsTopicAwsKmsKey**: Inspects all SNS topics to ensure that all of the topics or none of the topics have an AWS KMS key.
- **SnsTopicQuotas**: Inspects all SNS topics to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon SQS queues

- **SqsQueueType**: Inspects all SQS queues to ensure that they are all the same value for `Type`.
- **SqsQueueDelaySeconds**: Inspects all SQS queues to ensure that they all have the same value for `Delay Seconds`.
- **SqsQueueMaximumMessageSize**: Inspects all SQS queues to ensure that they all have the same value for `Maximum Message Size`.
- **SqsQueueMessageRetentionPeriod**: Inspects all SQS queues to ensure that they all have the same value for `Message Retention Period`.
- **SqsQueueReceiveMessageWaitTimeSeconds**: Inspects all SQS queues to ensure that they all have the same value for `Receive Message Wait Time Seconds`.
- **SqsQueueRedrivePolicyMaxReceiveCount**: Inspects all SQS queues to ensure that they all have the same value for `Redrive Policy Max Receive Count`.

- **SqsQueueVisibilityTimeout**: Inspects all SQS queues to ensure that they all have the same value for `Visibility Timeout`.
- **SqsQueueContentBasedDeduplication**: Inspects all SQS queues to ensure that they all have the same value for `Content-Based Deduplication`.
- **SqsQueueQuotas**: Inspects all SQS queues to ensure that they conform to quotas (limits) that are managed by Service Quotas.

Amazon VPCs

- **VpcCidrBlock**: Inspects all VPCs to ensure that they all have the same value for CIDR block network size.
- **VpcCidrBlocksSameProtocolVersion**: Inspects all VPCs that have the same CIDR blocks to ensure that they have the same value for Internet Stream Protocol version number.
- **VpcCidrBlocksStateInAssociationSets**: Inspects all CIDR block association sets for all VPCs to ensure that they all have CIDR blocks that are in an `ASSOCIATED` state.
- **VpcIpv6CidrBlocksStateInAssociationSets**: Inspects all CIDR block association sets for all VPCs to ensure that they all have CIDR blocks with the same number of addresses.
- **VpcCidrBlocksInAssociationSets**: Inspects all CIDR block association sets for all VPCs to ensure that they all have the same size.
- **VpcIpv6CidrBlocksInAssociationSets**: Inspects all IPv6 CIDR block association sets for all VPCs to ensure that they have the same size.
- **VpcState**: Inspects each VPC to ensure that it is in an `AVAILABLE` state.
- **VpcInstanceTenancy**: Inspects all VPCs to ensure that they all have the same value for `Instance Tenancy`.
- **VpcIsDefault**: Inspects all VPCs to ensure that they have the same value for `Is Default.`
- **VpcSubnetState**: Inspects each VPC subnet to ensure that it is in an AVAILABLE state.
- **VpcSubnetAvailableIpAddressCount**: Inspects each VPC subnet to ensure that it has an available IP address count greater than zero.
- **VpcSubnetCount**: Inspects all VPC subnets to ensure that they have the same number of subnets.
- **VpcQuotas**: Inspects all VPC subnets to ensure that they conform to quotas (limits) that are managed by Service Quotas.

AWS VPN connections

- **VpnConnectionsRouteCount**: Inspects all VPN connections to ensure that they have at least one route, and also the same number of routes.

- **VpnConnectionsEnableAcceleration**: Inspects all VPN connections to ensure that they have the same value for `Enable Accelerations`.
- **VpnConnectionsStaticRoutesOnly**: Inspects all VPN connections to ensure that they have the same value for `Static Routes Only.`
- **VpnConnectionsCategory**: Inspects all VPN connections to ensure that they have a category of VPN.
- **VpnConnectionsCustomerConfiguration**: Inspects all VPN connections to ensure that they have the same value for `Customer Gateway Configuration`.
- **VpnConnectionsCustomerGatewayId**: Inspects each VPN connection to ensure that it has a customer gateway attached.
- **VpnConnectionsRoutesState**: Inspects all VPN connections to ensure that they are in an `AVAILABLE` state.
- **VpnConnectionsVgwTelemetryStatus**: Inspects each VPN connection to ensure that it has a VGW status of UP.
- **VpnConnectionsVgwTelemetryIpAddress**: Inspects each VPN connection to ensure that it has a different outside IP address for each VGW telemetry.
- **VpnConnectionsTunnelOptions**: Inspects all VPN connections to ensure that they have the same tunnel options.
- **VpnConnectionsRoutesCidr**: Inspects all VPN connections to ensure that they have the same destination CIDR blocks.
- **VpnConnectionsInstanceType**: Inspects all VPN connections to ensure that they have the same `Instance Type`.

AWS VPN gateways

- **VpnGatewayState**: Inspects all VPN gateways to ensure that they are in an AVAILABLE state.
- **VpnGatewayAsn**: Inspects all VPN gateways to ensure that they have the same ASN.
- **VpnGatewayType**: Inspects all VPN gateways to ensure that they have the same type.
- **VpnGatewayAttachment**: Inspects all VPN gateways to ensure that they have the same attachment configurations.

View readiness rules on the console

You can view readiness rules on the AWS Management Console, listed by each resource type.

To view readiness rules on the console

1. Open the ARC console at https://console.aws.amazon.com/route53recovery/home#/dashboard.
2. Choose **Readiness check**.
3. Under **Resource type**, choose the resource type that you want to view the rules for.

Resource types and ARN formats in ARC

When you create a resource set in Amazon Application Recovery Controller (ARC), you specify the type of resource to include in the set and Amazon Resource Names (ARNs) for each of the resources to include. ARC expects a specific ARN format for each resource type. This section lists the resource types supported by ARC and the associated ARN formats for each one.

The specific format depends on the resource. When you provide an ARN, replace the *italicized* text with your resource-specific information.

> **Note**
>
> Be aware that the ARN format that ARC requires for resources might differ from the ARN format that a service itself requires for its resources. For example, the ARN formats that are described in the **Resource type** sections for each service in the Service Authorization Reference might not include the AWS account ID or other information that ARC needs to support features in the ARC service.

AWS::ApiGateway::Stage

An Amazon API Gateway Version 1 stage.

- **ARN format:** `arn:`*`partition`*`:apigateway:`*`region`*`:`*`account`*`:/restapis/`*`api-id`*`/stages/`*`stage-name`*

  Example: `arn:aws:apigateway:us-east-1:111122223333:/restapis/123456789/stages/ExampleStage`

  For more information, see API Gateway Amazon Resource Name (ARN) reference.

AWS::ApiGatewayV2::Stage

An Amazon API Gateway Version 2 stage.

- **ARN format:** `arn:partition:apigateway:region:account:/apis/api-id/stages/stage-name`

  Example: `arn:aws:apigateway:us-east-1:111122223333:/apis/123456789/stages/ExampleStage`

  For more information, see API Gateway Amazon Resource Name (ARN) reference.

AWS::CloudWatch::Alarm

An Amazon CloudWatch alarm.

- **ARN format:** `arn:partition:cloudwatch:region:account:alarm:alarm-name`

  Example: `arn:aws:cloudwatch:us-west-2:111122223333:alarm:test-alarm-1`

  For more information, see Resource types defined by Amazon CloudWatch.

AWS::DynamoDB::Table

An Amazon DynamoDB table.

- **ARN format:** `arn:partition:dynamodb:region:account:table/table-name`

  Example: `arn:aws:dynamodb:us-west-2:111122223333:table/BigTable`

  For more information, see DynamoDB resources and operations.

AWS::EC2::CustomerGateway

A customer gateway device.

- **ARN format:** `arn:partition:ec2:region:account:customer-gateway/CustomerGatewayId`

  Example: `arn:aws:ec2:us-west-2:111122223333:customer-gateway/vcg-123456789`

  For more information, see Resource types defined by Amazon EC2.

AWS::EC2::Volume

An Amazon EBS volume.

- **ARN format:** `arn:partition:ec2:region:account:volume/VolumeId`

  Example: `arn:aws:ec2:us-west-2:111122223333:volume/volume-of-cylinder-is-pi`

For more information, see API Gateway Amazon Resource Name (ARN) reference.

AWS::ElasticLoadBalancing::LoadBalancer

A Classic Load Balancer.

- **ARN format:** `arn:partition:elasticloadbalancing:region:account:loadbalancer/LoadBalancerN`

  Example: `arn:aws:elasticloadbalancing:us-west-2:111122223333:loadbalancer/123456789abcbdeCLB`

  For more information, see Elastic Load Balancing resources.

AWS::ElasticLoadBalancingV2::LoadBalancer

A Network Load Balancer or an Application Load Balancer.

- **ARN format for Network Load Balancer:** `arn:partition:elasticloadbalancing:region:account:loadbalancer/net/LoadBalancerName`

  Example for Network Load Balancer: `arn:aws:elasticloadbalancing:us-west-2:111122223333:loadbalancer/net/sandbox-net/123456789acbdeNLB`

- **ARN format for Application Load Balancer:** `arn:partition:elasticloadbalancing:region:account:loadbalancer/app/LoadBalancerName`

  Example for Application Load Balancer: `arn:aws:elasticloadbalancing:us-west-2:111122223333:loadbalancer/app/sandbox-alb/123456789acbdeALB`

  For more information, see Elastic Load Balancing resources.

AWS::Lambda::Function

An AWS Lambda function.

- **ARN format:** `arn:partition:lambda:region:account:function:FunctionName`

  Example: `arn:aws:lambda:us-west-2:111122223333:function:my-function`

  For more information, see Resources and conditions for Lambda actions.

AWS::MSK::Cluster

An Amazon MSK cluster.

- **ARN format:** `arn:partition:kafka:region:account:cluster/ClusterName/UUID`

  Example: `arn:aws:kafka:us-east-1:111122223333:cluster/demo-cluster-1/123456-1111-2222-3333`

  For more information, see Resource types defined by Amazon Managed Streaming for Apache Kafka.

AWS::RDS::DBCluster

An Aurora DB cluster.

- **ARN format:** `arn:partition:rds:region:account:cluster:DbClusterInstanceName`

  Example: `arn:aws:rds:us-west-2:111122223333:cluster:database-1`

  For more information, see Working with Amazon Resource Names (ARNs) in Amazon RDS.

AWS::Route53::HealthCheck

An Amazon Route 53 health check.

- **ARN format:** `arn:partition:route53:::healthcheck/Id`

  Example: `arn:aws:route53:::healthcheck/123456-1111-2222-3333`

AWS::SQS::Queue

An Amazon SQS queue.

- **ARN format:** `arn:partition:sqs:region:account:QueueName`

  Example: `arn:aws:sqs:us-west-2:111122223333:StandardQueue`

  For more information, see Amazon Simple Queue Service resource and operations.

AWS::SNS::Topic

An Amazon SNS topic.

- **ARN format:** `arn:partition:sns:region:account:TopicName`

  Example: `arn:aws:sns:us-west-2:111122223333:TopicName`

For more information, see Amazon SNS resource ARN format.

AWS::SNS::Subscription

An Amazon SNS subscription.

- **ARN format:** `arn:partition:sns:region:account:TopicName:SubscriptionId`

  Example: `arn:aws:sns:us-west-2:111122223333:TopicName:12345678901234567890`

AWS::EC2::VPC

A virtual private cloud (VPC).

- **ARN format:** `arn:partition:ec2:region:account:vpc/VpcId`

  Example: `arn:aws:ec2:us-west-2:111122223333:vpc/vpc-123456789`

  For more information, see VPC Resources.

AWS::EC2::VPNConnection

A virtual private network (VPN) connection.

- **ARN format:** `arn:partition:ec2:region:account:vpn-connection/VpnConnectionId`

  Example: `arn:aws:ec2:us-west-2:111122223333:vpn-connection/vpn-123456789`

  For more information, see Resource types defined by Amazon EC2.

AWS::EC2::VPNGateway

A virtual private network (VPN) gateway.

- **ARN format:** `arn:partition:ec2:region:account:vpn-gateway/VpnGatewayId`

  Example: `arn:aws:ec2:us-west-2:111122223333:vpn-gateway/vgw-123456789acbdefgh`

  For more information, see Resource types defined by Amazon EC2.

AWS::Route53RecoveryReadiness::DNSTargetResource

A DNS target resource for readiness checks includes the DNS record type, domain name, Route 53 hosted zone ARN, and Network Load Balancer ARN or Route 53 record set ID.

- **ARN format for hosted zone:** `arn:partition:route53::account:hostedzone/Id`

  Example for a hosted zone: `arn:aws:route53::111122223333:hostedzone/abcHostedZone`

  NOTE: You must include the account ID in hosted zone ARNs, as specified here. The account ID is required so that ARC can poll the resource. The format is intentionally different from the ARN format that Amazon Route 53 requires, described in the Route 53 service Resource types in the *Service Authorization Reference*.

- **ARN format for Network Load Balancer:** `arn:partition:elasticloadbalancing:region:account:loadbalancer/net/LoadBalancerName`

  Example for Network Load Balancer: `arn:aws:elasticloadbalancing:us-west-2:111122223333:loadbalancer/net/sandbox-net/123456789acbdefgh`

  For more information, see Elastic Load Balancing resources.

Logging and monitoring for readiness check in Amazon Application Recovery Controller (ARC)

You can use Amazon CloudWatch, AWS CloudTrail, and Amazon EventBridge for monitoring readiness check in Amazon Application Recovery Controller (ARC), to analyze patterns and help troubleshoot issues.

> **Note**
>
> You must view CloudWatch metrics and logs for ARC in the US West (Oregon) Region, both in the console and when using the AWS CLI. When you use the AWS CLI, specify the US West (Oregon) Region for your command by including the following parameter: `--region us-west-2`.

Topics

- Using Amazon CloudWatch with readiness check in ARC
- Logging readiness check API calls using AWS CloudTrail
- Using readiness check in ARC with Amazon EventBridge

Using Amazon CloudWatch with readiness check in ARC

Amazon Application Recovery Controller (ARC) publishes data points to Amazon CloudWatch for your readiness checks. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as *metrics*. Think of a metric as a variable to monitor, and the data points as the values of that variable over time. For example, you can monitor traffic through an AWS Region over a specified time period. Each data point has an associated time stamp and an optional unit of measurement.

You can use metrics to verify that your system is performing as expected. For example, you can create a CloudWatch alarm to monitor a specified metric and initiate an action (such as sending a notification to an email address) if the metric goes outside what you consider an acceptable range.

For more information, see the Amazon CloudWatch User Guide.

Topics

ARC metrics

The `AWS/Route53RecoveryReadiness` namespace includes the following metrics.

Metric	Description
`ReadinessChecks`	Represents the number of readiness checks processed by ARC. The metric can be dimensioned by its states, listed below. **Unit**: `Count`. **Reporting criteria**: There is a nonzero value. **Statistics**: The only useful statistic is `Sum`. **Dimensions** • `READY`

Metric	Description
	• NOT_READY • NOT_AUTHORIZED • UNKNOWN
Resources	Represents the number of resources processed by ARC, which can be dimensioned by their resource identifier, as defined by the API. **Unit**: Count. **Reporting criteria**: There is a nonzero value. **Statistics**: The only useful statistic is Sum. **Dimensions** • ResourceSetType : These are the resource types, filtered by the number of resources per given type evaluated by ARC For example: AWS::CloudWatch::Alarm

Statistics for ARC metrics

CloudWatch provides statistics based on the metric data points published by ARC. Statistics are aggregations of metric data over a specified period of time. When you request statistics, the returned data stream is identified by the metric name and dimension. A dimension is a name/value pair that uniquely identifies a metric.

The following are examples of metric/dimension combinations that you might find useful:

- View the number of readiness checks evaluated for readiness by ARC.
- View the total number of resources for a given resource set type evaluated by ARC.

View CloudWatch metrics in ARC

You can view the CloudWatch metrics for ARC using the CloudWatch console or the AWS CLI. In the console, metrics are displayed as monitoring graphs.

You must view CloudWatch metrics for ARC in the US West (Oregon) Region, both in the console or when using the AWS CLI. When you use the AWS CLI, specify the US West (Oregon) Region for your command by including the following parameter: `--region us-west-2`.

To view metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**.
3. Select the **Route53RecoveryReadiness** namespace.
4. (Optional) To view a metric across all dimensions, type its name in the search field.

To view metrics using the AWS CLI

Use the following list-metrics command to list the available metrics:

```
aws cloudwatch list-metrics --namespace AWS/Route53RecoveryReadiness --region us-west-2
```

To get the statistics for a metric using the AWS CLI

Use the following get-metric-statistics command to get statistics for a specified metric and dimension. Note that CloudWatch treats each unique combination of dimensions as a separate metric. You can't retrieve statistics using combinations of dimensions that were not specifically published. You must specify the same dimensions that were used when the metrics were created.

The following example lists the total readiness checks evaluated, per minute, for an account in ARC.

```
aws cloudwatch get-metric-statistics --namespace AWS/Route53RecoveryReadiness \
--metric-name ReadinessChecks \
--region us-west-2 \
--statistics Sum --period 60 \
--dimensions Name=State,Value=READY \
--start-time 2021-07-03T01:00:00Z --end-time 2021-07-03T01:20:00Z
```

The following is example output from the command:

```
{
    "Label": "ReadinessChecks",
    "Datapoints": [
        {
```

```
            "Timestamp": "2021-07-08T18:00:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:04:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:01:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:02:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2021-07-08T18:03:00Z",
            "Sum": 1.0,
            "Unit": "Count"
        }
    ]
}
```


Logging readiness check API calls using AWS CloudTrail

Amazon Route 53 Application Recovery Controller is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Route 53 ARC. CloudTrail captures all API calls for Route 53 ARC as events. The calls captured include calls from the Route 53 ARC console and code calls to the Route 53 ARC API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Route 53 ARC. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to Route 53 ARC, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Route 53 ARC information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Route 53 ARC, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Working with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for Route 53 ARC, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All Route 53 ARC actions are logged by CloudTrail and are documented in the Recovery Readiness API Reference Guide for Amazon Route 53 Application Recovery Controller, Recovery Control Configuration API Reference Guide for Amazon Route 53 Application Recovery Controller, and Routing Control API Reference Guide for Amazon Route 53 Application Recovery Controller. For example, calls to the `CreateCluster`, `UpdateRoutingControlState` and `CreateRecoveryGroup` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Viewing Route 53 ARC events in event history

CloudTrail lets you view recent events in **Event history**. To view events for Route 53 ARC API requests, you must choose **US West (Oregon)** in the Region selector at the top of the console. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*.

Understanding Route 53 ARC log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateRecoveryGroup` action for readiness check.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "A1B2C3D4E5F6G7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA33L3W36EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/admin",
                "accountId": "111122223333",
                "userName": "EXAMPLENAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-07-06T17:38:05Z"
            }
        }
    },
    "eventTime": "2021-07-06T18:08:03Z",
    "eventSource": "route53-recovery-readiness.amazonaws.com",
    "eventName": "CreateRecoveryGroup",
```

```
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.50",
    "userAgent": "Boto3/1.17.101 Python/3.8.10 Linux/4.14.231-180.360.amzn2.x86_64
 exec-env/AWS_Lambda_python3.8 Botocore/1.20.102",
    "requestParameters": {
        "recoveryGroupName": "MyRecoveryGroup"
    },
    "responseElements": {
        "Access-Control-Expose-Headers": "x-amzn-errortype,x-amzn-requestid,x-amzn-
errormessage,x-amzn-trace-id,x-amzn-requestid,x-amz-apigw-id,date",
        "cells": [],
        "recoveryGroupName": "MyRecoveryGroup",
        "recoveryGroupArn": "arn:aws:route53-recovery-readiness::111122223333:recovery-
group/MyRecoveryGroup",
        "tags": "***"
    },
    "requestID": "fd42dcf7-6446-41e9-b408-d096example",
    "eventID": "4b5c42df-1174-46c8-be99-d67aexample",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```


Using readiness check in ARC with Amazon EventBridge

Using Amazon EventBridge, you can set up event-driven rules that monitor your readiness check resources in Amazon Application Recovery Controller (ARC), and then initiate target actions that use other AWS services. For example, you can set a rule for sending out email notifications by signaling an Amazon SNS topic when a readiness check status changes from **READY** to **NOT READY**.

> **Note**
>
> ARC only publishes EventBridge events for readiness check in the US West (Oregon) (us-west-2) AWS Region. To receive EventBridge events for readiness check, create EventBridge rules in the US West (Oregon) Region.

You can create rules in Amazon EventBridge to act on the following ARC readiness check event:

- *Readiness check readiness.* The event specifies if readiness check status changes, for example, from **READY** to **NOT READY**.

To capture specific ARC events that you're interested in, define event-specific patterns that EventBridge can use to detect the events. Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

Events are emitted on a best effort basis. They're delivered from ARC to EventBridge in near real-time under normal operational circumstances. However, situations can arise that might delay or prevent delivery of an event.

For information about how EventBridge rules work with event patterns, see Events and Event Patterns in EventBridge.

Monitor a readiness check resource with EventBridge

With EventBridge, you can create rules that define actions to take when ARC emits events for readiness check resources.

To type or copy and paste an event pattern into the EventBridge console, in the console, select to the option **Enter my own** option. To help you determine event patterns that might be useful for you, this topic includes example readiness event patterns.

To create a rule for a resource event

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. For the AWS Region to create the rule in, choose US West (Oregon). This is the required Region for readiness events.
3. Choose **Create rule**.
4. Enter a **Name** for the rule, and, optionally, a description.
5. For **Event bus**, leave the default value, **default**.
6. Choose **Next**.
7. For the **Build event pattern** step, for **Event source**, leave the default value, **AWS events**.
8. Under **Sample event**, choose **Enter my own**.
9. For **Sample events**, type or copy and paste an event pattern. For examples, see the next section.

Example readiness event patterns

Event patterns have the same structure as the events that they match. The pattern quotes the fields that you want to match and provides the values that you're looking for.

You can copy and paste event patterns from this section into EventBridge to create rules that you can use to monitor ARC actions and resources.

The following event patterns provide examples that you might use in EventBridge for the readiness check capability in ARC.

- *Select all events from ARC readiness check.*

```
{
    "source": [
        "aws.route53-recovery-readiness"
    ]
}
```

- *Select only events related to cells.*

```
{
    "source": [
        "aws.route53-recovery-readiness"
    ],
    "detail-type": [
        "Route 53 Application Recovery Controller cell readiness status change"
    ]
}
```

- *Select only events related to a specific cell called `MyExampleCell`.*

```
{
    "source": [
        "aws.route53-recovery-readiness"
    ],
    "detail-type": [
        "Route 53 Application Recovery Controller cell readiness status change"
    ],
    "resources": [
        "arn:aws:route53-recovery-readiness::111122223333:cell/MyExampleCell"
    ]
```

```
}
```

- *Select only events when any recovery group, cell, or readiness check status becomes NOT READY.*

```
{
   "source":[
      "aws.route53-recovery-readiness"
   ],
   "detail-type":{
      "new-state":{
         "readiness-status":[
            "NOT_READY"
         ]
      }
   }
}
```

- *Select only events when any recovery group, cell, or readiness check becomes anything except READY*

```
{
   "source":[
      "aws.route53-recovery-readiness"
   ],
   "detail":{
      "new-state":{
         "readiness-status":[
            {
               "anything-but":"READY"
            }
         ]
      }
   }
}
```

The following is an example ARC event for a *recovery group readiness status change*:

```
{
    "version": "0",
    "account":"111122223333",
    "detail-type":"Route 53 Application Recovery Controller recovery group readiness
 status change",
```

```
    "source":"route53-recovery-readiness.amazonaws.com",
    "time":"2020-11-03T00:31:54Z",
    "id": "1234a678-1b23-c123-12fd3f456e78",
    "region": "us-west-2",
    "resources":[
        "arn:aws:route53-recovery-readiness::111122223333:recovery-group/BillingApp"
    ],
    "detail": {
        "recovery-group-name": "BillingApp",
        "previous-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        },
        "new-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        }
    }
}
```

The following is an example ARC event for a *cell readiness status change*:

```
{
    "version": "0",
    "account":"111122223333",
    "detail-type":"Route 53 Application Recovery Controller cell readiness status
 change",
    "source":"route53-recovery-readiness.amazonaws.com",
    "time":"2020-11-03T00:31:54Z",
    "id": "1234a678-1b23-c123-12fd3f456e78",
    "region": "us-west-2",
    "resources":[
        "arn:aws:route53-recovery-readiness::111122223333:cell/PDXCell"
    ],
    "detail": {
        "cell-name": "PDXCell",
        "previous-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        },
        "new-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        }
    }
}
```

The following is an example ARC event for a *readiness check status change*:

```
{
    "version": "0",
    "account":"111122223333",
    "detail-type":"Route 53 Application Recovery Controller readiness check status
 change",
    "source":"route53-recovery-readiness.amazonaws.com",
    "time":"2020-11-03T00:31:54Z",
    "id": "1234a678-1b23-c123-12fd3f456e78",
    "region": "us-west-2",
    "resources":[
        "arn:aws:route53-recovery-readiness::111122223333:readiness-check/
UserTableReadinessCheck"
    ],
    "detail": {
    "readiness-check-name": "UserTableReadinessCheck",
        "previous-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        },
        "new-state": {
            "readiness-status": "READY|NOT_READY|UNKNOWN|NOT_AUTHORIZED"
        }
    }
}
```

Specify a CloudWatch log group to use as a target

When you create an EventBridge rule, you must specify the target where events that are matched to the rule are sent. For a list of available targets for EventBridge, see Targets available in the EventBridge console. One of the targets that you can add to an EventBridge rule is an Amazon CloudWatch log group. This section describes the requirements for adding CloudWatch log groups as targets, and provides a procedure for adding a log group when you create a rule.

To add a CloudWatch log group as a target, you can do one of the following:

- Create a new log group
- Choose an existing log group

If you specify a new log group using the console when you create a rule, EventBridge automatically creates the log group for you. Make sure that the log group that you use as a target for the

EventBridge rule starts with `/aws/events`. If you want to choose an existing log group, be aware that only log groups that start with `/aws/events` appear as options in the drop-down menu. For more information, see Create a new log group in the *Amazon CloudWatch User Guide*.

If you create or use a CloudWatch log group to use as a target using CloudWatch operations outside of the console, make sure that you set permissions correctly. If you use the console to add a log group to an EventBridge rule, then the resource-based policy for the log group is updated automatically. But, if you use the AWS Command Line Interface or an AWS SDK to specify a log group, then you must update resource-based policy for the log group. The following example policy illustrates the permissions that you must define in a resource-based policy for the log group:

```
{
  "Statement": [
    {
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "events.amazonaws.com",
          "delivery.logs.amazonaws.com"
        ]
      },
      "Resource": "arn:aws:logs:region:account:log-group:/aws/events/*:*",
      "Sid": "TrustEventsToStoreLogEvent"
    }
  ],
  "Version": "2012-10-17"
}
```

You can't configure a resource-based policy for a log group by using the console. To add the required permissions to a resource-based policy, use the CloudWatch PutResourcePolicy API operation. Then, you can use the describe-resource-policies CLI command to check that your policy was applied correctly.

To create a rule for a resource event and specify a CloudWatch log group target

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
2. Choose the AWS Region that you want to create the rule in.

3. Choose **Create rule** and then enter any information about that rule, such as the event pattern or schedule details.

   For more information about creating EventBridge rules for readiness, see Monitor a readiness check resource with EventBridge.
4. On the **Select target** page, choose **CloudWatch** as your target.
5. Choose a CloudWatch log group from the drop-down menu.

Identity and Access Management for readiness check

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Route 53 ARC resources. IAM is an AWS service that you can use with no additional charge.

Contents

How readiness check in SERVICElong; works with IAM

Before you use IAM to manage access to Route 53 ARC, learn what IAM features are available to use with Route 53 ARC.

Before you use IAM to manage access to readiness check in Amazon Application Recovery Controller (ARC), learn what IAM features are available to use with readiness check.

IAM features you can use with readiness check in Amazon Route 53 Application Recovery Controller

IAM feature	Readiness check support
Identity-based policies	Yes

IAM feature	Readiness check support
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level, overall view of how AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for readiness check

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

To view examples of Route 53 ARC identity-based policies, see Identity-based policy examples in Amazon Route 53 Application Recovery Controller.

Resource-based policies within readiness check

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-based policies, service administrators can use them to control access to a specific resource.

Policy actions for readiness check

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Route 53 ARC actions for readiness check, see Actions defined by Amazon Route 53 Recovery Readiness in the *Service Authorization Reference*.

Policy actions in Route 53 ARC for readiness check use the following prefixes before the action:

```
route53-recovery-readiness
```

To specify multiple actions in a single statement, separate them with commas. For example, the following:

```
"Action": [
      "route53-recovery-readiness:action1",
      "route53-recovery-readiness:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "route53-recovery-readiness:Describe*"
```

To view examples of ARC identity-based policies for readiness check, see Identity-based policy examples for readiness check in Amazon Route 53 Application Recovery Controller.

Policy resources for readiness check

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Route 53 ARC actions for zonal shift, see Actions defined by Amazon Route 53 Recovery Readiness.

To view examples of ARC identity-based policies for readiness check, see Identity-based policy examples for readiness check in Amazon Route 53 Application Recovery Controller.

Policy condition keys for readiness check

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Route 53 ARC actions for readiness check, see Condition keys for Amazon Route 53 Recovery Readiness

To see the actions and resources that you can use with a condition key with readiness check, see Actions defined by Amazon Route 53 Recovery Readiness

To view examples of ARC identity-based policies for readiness check, see Identity-based policy examples for readiness check in Amazon Route 53 Application Recovery Controller.

Access control lists (ACLs) in readiness check

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with readiness check

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Recovery Readiness (readiness check) supports ABAC.

Using temporary credentials with readiness check

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for readiness check

Supports forward access sessions (FAS): Yes

When you use an IAM entity (user or role) to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions.

To see whether an action in readiness check requires additional dependent actions in a policy, see Amazon Route 53 Recovery Readiness

Service roles for readiness check

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

Service-linked roles for readiness check

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Route 53 ARC service-linked roles, see Using service-linked role for readiness check in Route 53 ARC.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for readiness check in Amazon Route 53 Application Recovery Controller

By default, users and roles don't have permission to create or modify Route 53 ARC resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Route 53 ARC, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Route 53 Application Recovery Controller in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Route 53 ARC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when

API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Example: Readiness check console access

To access the Amazon Route 53 Application Recovery Controller console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Route 53 ARC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the readiness check console when you allow access to only specific API operations, also attach a `ReadOnly` AWS managed policy for readiness check to the entities. For more information, see the readiness check Readiness check managed policies page or Adding permissions to a user in the *IAM User Guide*.

To perform some tasks, users must have permission to create the service-linked role that is associated with readiness check in ARC. To learn more, see Using service-linked role for readiness check in Route 53 ARC.

To give users full access to use readiness check features through the console, attach a policy like the following to the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                   "route53-recovery-readiness:CreateCell",
                   "route53-recovery-readiness:CreateCrossAccountAuthorization",
                   "route53-recovery-readiness:CreateReadinessCheck",
                   "route53-recovery-readiness:CreateRecoveryGroup",
                   "route53-recovery-readiness:CreateResourceSet",
```

```
                    "route53-recovery-readiness:DeleteCell",
                    "route53-recovery-readiness:DeleteCrossAccountAuthorization",
                    "route53-recovery-readiness:DeleteReadinessCheck",
                    "route53-recovery-readiness:DeleteRecoveryGroup",
                    "route53-recovery-readiness:DeleteResourceSet",
                    "route53-recovery-readiness:GetArchitectureRecommendations",
                    "route53-recovery-readiness:GetCell",
                    "route53-recovery-readiness:GetCellReadinessSummary",
                    "route53-recovery-readiness:GetReadinessCheck",
                    "route53-recovery-readiness:GetReadinessCheckResourceStatus",
                    "route53-recovery-readiness:GetReadinessCheckStatus",
                    "route53-recovery-readiness:GetRecoveryGroup",
                    "route53-recovery-readiness:GetRecoveryGroupReadinessSummary",
                    "route53-recovery-readiness:GetResourceSet",
                    "route53-recovery-readiness:ListCells",
                    "route53-recovery-readiness:ListCrossAccountAuthorizations",
                    "route53-recovery-readiness:ListReadinessChecks",
                    "route53-recovery-readiness:ListRecoveryGroups",
                    "route53-recovery-readiness:ListResourceSets",
                    "route53-recovery-readiness:ListRules",
                    "route53-recovery-readiness:UpdateCell",
                    "route53-recovery-readiness:UpdateReadinessCheck",
                    "route53-recovery-readiness:UpdateRecoveryGroup",
                    "route53-recovery-readiness:UpdateResourceSet"
                ],
                "Resource": "*"
            }
        ]
}
```

Examples: Readiness check API actions for readiness check

To ensure that a user can use Route 53 ARC API actions to work with the Route 53 ARC readiness check control plane – for example, to create recovery groups, resource sets, and readiness checks – attach a policy that corresponds to the API operations that the user needs to work with, as described below.

To perform some tasks, users must have permission to create the service-linked role that is associated with readiness check in ARC. To learn more, see Using service-linked role for readiness check in Route 53 ARC.

To work with API operations for readiness check, attach a policy like the following to the user:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                   "route53-recovery-readiness:CreateCell",
                   "route53-recovery-readiness:CreateCrossAccountAuthorization",
                   "route53-recovery-readiness:CreateReadinessCheck",
                   "route53-recovery-readiness:CreateRecoveryGroup",
                   "route53-recovery-readiness:CreateResourceSet",
                   "route53-recovery-readiness:DeleteCell",
                   "route53-recovery-readiness:DeleteCrossAccountAuthorization",
                   "route53-recovery-readiness:DeleteReadinessCheck",
                   "route53-recovery-readiness:DeleteRecoveryGroup",
                   "route53-recovery-readiness:DeleteResourceSet",
                   "route53-recovery-readiness:GetArchitectureRecommendations",
                   "route53-recovery-readiness:GetCell",
                   "route53-recovery-readiness:GetCellReadinessSummary",
                   "route53-recovery-readiness:GetReadinessCheck",
                   "route53-recovery-readiness:GetReadinessCheckResourceStatus",
                   "route53-recovery-readiness:GetReadinessCheckStatus",
                   "route53-recovery-readiness:GetRecoveryGroup",
                   "route53-recovery-readiness:GetRecoveryGroupReadinessSummary",
                   "route53-recovery-readiness:GetResourceSet",
                   "route53-recovery-readiness:ListCells",
                   "route53-recovery-readiness:ListCrossAccountAuthorizations",
                   "route53-recovery-readiness:ListReadinessChecks",
                   "route53-recovery-readiness:ListRecoveryGroups",
                   "route53-recovery-readiness:ListResourceSets",
                   "route53-recovery-readiness:ListRules",
                   "route53-recovery-readiness:ListTagsForResources",
                   "route53-recovery-readiness:UpdateCell",
                   "route53-recovery-readiness:UpdateReadinessCheck",
                   "route53-recovery-readiness:UpdateRecoveryGroup",
                   "route53-recovery-readiness:UpdateResourceSet",
                   "route53-recovery-readiness:TagResource",
                   "route53-recovery-readiness:UntagResource"
             ],
            "Resource": "*"
        }
    ]
}
```

Using service-linked role for readiness check in Route 53 ARC

Amazon Route 53 Application Recovery Controller uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to a service— in this case, Route 53 ARC. Service-linked roles are predefined by Route 53 ARC and include all the permissions that the service requires to call other AWS services on your behalf for specific purposes.

Service-linked roles make setting up Route 53 ARC easier because you don't have to manually add the necessary permissions. Route 53 ARC defines the permissions of its service-linked roles, and unless defined otherwise, only Route 53 ARC can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting its related resources. This protects your Route 53 ARC resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services that work with IAM and look for the services that have **Yes** in the **Service-linked role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Route 53 ARC has the following service-linked roles, which are described in this chapter:

- Route 53 ARC uses the service-linked role named **Route53RecoveryReadinessServiceRolePolicy** to access resources and configurations to check readiness.
- Route 53 ARC uses the service-linked role named for autoshift practice runs, to monitor customer-provided Amazon CloudWatch alarms and customer AWS Health Dashboard events, and to start practice runs.

Service-linked role permissions for Route53RecoveryReadinessServiceRolePolicy

Route 53 ARC uses a service-linked role named **Route53RecoveryReadinessServiceRolePolicy** to access resources and configurations to check readiness. This section describes the permissions for the service-linked role, and information about creating, editing, and deleting the role.

Service-linked role permissions for Route53RecoveryReadinessServiceRolePolicy

This service-linked role uses the managed policy `Route53RecoveryReadinessServiceRolePolicy`.

The **Route53RecoveryReadinessServiceRolePolicy** service-linked role trusts the following service to assume the role:

- `route53-recovery-readiness.amazonaws.com`

To view the permissions for this policy, see Route53RecoveryReadinessServiceRolePolicy in the *AWS Managed Policy Reference*.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating the Route53RecoveryReadinessServiceRolePolicy service-linked role for Route 53 ARC

You don't need to manually create the **Route53RecoveryReadinessServiceRolePolicy** service-linked role. When you create the first readiness check or cross account authorization in the AWS Management Console, the AWS CLI, or the AWS API, Route 53 ARC creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create the first readiness check or cross account authorization, Route 53 ARC creates the service-linked role for you again.

Editing the Route53RecoveryReadinessServiceRolePolicy service-linked role for Route 53 ARC

Route 53 ARC does not allow you to edit the **Route53RecoveryReadinessServiceRolePolicy** service-linked role. After you create the service-linked role, you cannot change the name of the role because other entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting the Route53RecoveryReadinessServiceRolePolicy service-linked role for Route 53 ARC

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don’t have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

After you have removed your readiness checks and your cross-account authorizations, then you can delete the **Route53RecoveryReadinessServiceRolePolicy** service-linked role. For more information about readiness checks, see Readiness check in Amazon Application Recovery Controller (ARC). For

more information about cross-account authorizations, see Creating cross-account authorizations in ARC.

> **Note**
>
> If the Route 53 ARC service is using the role when you try to delete the resources, then the service role deletion might fail. If that happens, wait for a few minutes and try the again to delete the role.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the Route53RecoveryReadinessServiceRolePolicy service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Updates to the ARC service-linked role for readiness check

For updates to the AWS managed policies for the Route 53 ARC service-linked roles, see the AWS managed policies updates table for ARC. You can also subscribe to automatic RSS alerts on the Route 53 ARC Document history page.

AWS managed policies for readiness check in Amazon Application Recovery Controller (ARC)

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: Route53RecoveryReadinessServiceRolePolicy

You can't attach `Route53RecoveryReadinessServiceRolePolicy` to your IAM entities. This policy is attached to a service-linked role that allows Amazon Application Recovery Controller (ARC) to access AWS services and resources that are used or managed by ARC. For more information, see Using service-linked role for readiness check in Route 53 ARC.

AWS managed policy: AmazonRoute53RecoveryReadinessFullAccess

You can attach `AmazonRoute53RecoveryReadinessFullAccess` to your IAM entities. This policy grants full access to actions for working with recovery readiness (readiness check) in ARC. Attach it to IAM users and other principals who need full access to recovery readiness actions.

To view the permissions for this policy, see AmazonRoute53RecoveryReadinessFullAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AmazonRoute53RecoveryReadinessReadOnlyAccess

You can attach `AmazonRoute53RecoveryReadinessReadOnlyAccess` to your IAM entities. This policy grants read-only access to actions for working with recovery readiness in ARC. It's useful for users who need to view readiness statuses and recovery group configurations. These users can't create, update, or delete recovery readiness resources.

To view the permissions for this policy, see AmazonRoute53RecoveryReadinessReadOnlyAccess in the *AWS Managed Policy Reference*.

Updates for AWS managed policies for readiness

For details about updates to AWS managed policies for readiness check in ARC since this service began tracking these changes, see Updates to AWS managed policies for Amazon Application Recovery Controller (ARC). For automatic alerts about changes to this page, subscribe to the RSS feed on the ARC Document history page.

Quotas for readiness check

Readiness check in Amazon Application Recovery Controller (ARC) is subject to the following quotas (formerly referred to as limits).

Entity	Quota
Number of recovery groups per account	5
Number of cells per account	15
Number of nested cells per cell	3
Number of cells per recovery group	3
Number of resources per cell	10
Number of resources per recovery group	10
Number of resources per resource set	6
Number of resource sets per account	200
Number of readiness checks per account	200
Number of cross-account authorizations	100

Code examples for Application Recovery Controller using AWS SDKs

The following code examples show how to use Application Recovery Controller with an AWS software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Basic examples for Application Recovery Controller using AWS SDKs
  - Actions for Application Recovery Controller using AWS SDKs
    - Use GetRoutingControlState with an AWS SDK or CLI
    - Use UpdateRoutingControlState with an AWS SDK or CLI

Basic examples for Application Recovery Controller using AWS SDKs

The following code examples show how to use the basics of Amazon Route 53 Application Recovery Controller with AWS SDKs.

Examples

- Actions for Application Recovery Controller using AWS SDKs
  - Use GetRoutingControlState with an AWS SDK or CLI
  - Use UpdateRoutingControlState with an AWS SDK or CLI

Actions for Application Recovery Controller using AWS SDKs

The following code examples demonstrate how to perform individual Application Recovery Controller actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the Amazon Route 53 Application Recovery Controller API Reference.

Examples

- Use GetRoutingControlState with an AWS SDK or CLI
- Use UpdateRoutingControlState with an AWS SDK or CLI

Use `GetRoutingControlState` with an AWS SDK or CLI

The following code examples show how to use `GetRoutingControlState`.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static GetRoutingControlStateResponse
 getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
```

```
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region())).build();
                return client.getRoutingControlState(
                        GetRoutingControlStateRequest.builder()
                                .routingControlArn(routingControlArn).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
    }
```

- For API details, see GetRoutingControlState in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
```

```
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn
            )
            return response
        except Exception as error:
            print(error)
            raise error
```

- For API details, see GetRoutingControlState in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateRoutingControlState` with an AWS SDK or CLI

The following code examples show how to use `UpdateRoutingControlState`.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    public static UpdateRoutingControlStateResponse
 updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
            String routingControlArn,
            String routingControlState) {
        // As a best practice, we recommend choosing a random cluster endpoint to
 get or
        // set routing control states.
        // For more information, see
        // https://docs.aws.amazon.com/r53recovery/latest/dg/route53-arc-best-
practices.html#route53-arc-best-practices.regional
        Collections.shuffle(clusterEndpoints);
        for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
            try {
                System.out.println(clusterEndpoint);
                Route53RecoveryClusterClient client =
 Route53RecoveryClusterClient.builder()
                        .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                        .region(Region.of(clusterEndpoint.region()))
                        .build();
                return client.updateRoutingControlState(
                        UpdateRoutingControlStateRequest.builder()

 .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
            } catch (Exception exception) {
                System.out.println(exception);
            }
        }
        return null;
```

```
    }
```

- For API details, see UpdateRoutingControlState in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
import boto3


def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
 specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        "route53-recovery-cluster",
        endpoint_url=cluster_endpoint["Endpoint"],
        region_name=cluster_endpoint["Region"],
    )



def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state
):
    """
    Updates the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.
```

```
    :param routing_control_arn: The ARN of the routing control to update the
 state for.
    :param cluster_endpoints: The list of cluster endpoints to try.
    :param routing_control_state: The new routing control state.
    :return: The routing control update response.
    """

    # As a best practice, we recommend choosing a random cluster endpoint to get
 or set routing control states.
    # For more information, see https://docs.aws.amazon.com/r53recovery/latest/
dg/route53-arc-best-practices.html#route53-arc-best-practices.regional
    random.shuffle(cluster_endpoints)
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.update_routing_control_state(
                RoutingControlArn=routing_control_arn,
                RoutingControlState=routing_control_state,
            )
            return response
        except Exception as error:
            print(error)
```

- For API details, see UpdateRoutingControlState in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using this service with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Security in Amazon Route 53 Application Recovery Controller

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Route 53 Application Recovery Controller, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Route 53 ARC. The following topics show you how to configure Route 53 ARC to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Route 53 ARC resources.

Topics

Data protection in Amazon Route 53 Application Recovery Controller

The AWS shared responsibility model applies to data protection in Amazon Route 53 Application Recovery Controller. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Route 53 ARC or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

Customer configuration information is stored in service-owned Amazon DynamoDB global tables, and is encrypted at rest.

Datasets that contain the status of cells in a Route 53 ARC cluster are written to an Amazon EBS volume for backup. Route 53 ARC uses the default Amazon EBS encryption while the data is at rest.

Encryption in transit

Customer requests and responses—for Route 53 ARC configuration, readiness status queries, cell state updates, and so on—are encrypted during transport throughout the service by using TLS.

Identity and Access Management for Amazon Route 53 Application Recovery Controller

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Route 53 ARC resources. IAM is an AWS service that you can use with no additional charge.

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Route 53 ARC.

Service user – If you use the Route 53 ARC service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Route 53 ARC features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Route 53 ARC, see Troubleshooting Amazon Route 53 Application Recovery Controller identity and access.

Service administrator – If you're in charge of Route 53 ARC resources at your company, you probably have full access to Route 53 ARC. It's your job to determine which Route 53 ARC features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with

Route 53 ARC, see How Amazon Route 53 Application Recovery Controller capabilities work with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Route 53 ARC. To view example Route 53 ARC identity-based policies that you can use in IAM, see Identity-based policy examples in Amazon Route 53 Application Recovery Controller.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and

is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A

user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon Route 53 Application Recovery Controller capabilities work with IAM

For information about how each Amazon Route 53 Application Recovery Controller capability works with IAM, see the following topics:

- IAM for zonal shift
- IAM for zonal autoshift
- IAM for routing control
- IAM for readiness check

Identity-based policy examples in Amazon Route 53 Application Recovery Controller

To see identity-based policy examples for each capability in Amazon Application Recovery Controller (ARC), see the following topics in the AWS Identity and Access Management chapters for each capability:

- Identity-based policy examples for zonal autoshift
- Identity-based policy examples for zonal shift in Amazon Route 53 Application Recovery Controller
- Identity-based policy examples for routing control in Amazon Route 53 Application Recovery Controller
- Identity-based policy examples for readiness check in Amazon Route 53 Application Recovery Controller

AWS managed policies for Amazon Application Recovery Controller (ARC)

For information about the AWS managed policies for the Amazon Route 53 Application Recovery Controller capabilities with managed policies, including a managed policy for a service-linked role, see the following topics:

- Managed polices for zonal autoshift
- Managed polices for routing control
- Managed polices for readiness check

Updates to AWS managed policies for Amazon Application Recovery Controller (ARC)

View details about updates to AWS managed policies for capabilities in ARC since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the ARC Document history page.

Change	Description	Date
AWSServiceRoleForPercPracticePolicy – New policy	Route 53 ARC added a new service-linked role for autoshift and practice runs. Route 53 ARC uses the permissions enabled by the service-linked role to monitor customer-provided Amazon CloudWatch alarms and customer AWS Health Dashboard events for practice runs, and to start practice runs. To learn more about the new service-linked role, see Service-linked role permissions for AWSServiceRoleForZonalAutoshiftPracticeRun.	November 30, 2023
AmazonRoute53RecoveryControlConfigReadOnlyAccess – Updated policy	Adds permissions for `GetResourcePolicy` , to support returning details about AWS Resource Access Manager resource policies for shared resources.	October 18, 2023

Change	Description	Date

Route53RecoveryReadinessServiceRolePolicy – Updated policy	Route 53 ARC added new permissions to query information about Amazon EC2 instances.
	
	Route 53 ARC uses the following permissions to support polling Amazon EC2 instances, to run readiness checks and determine the readiness status for the instances.
	
	`ec2:DescribeVpnGateways`
	
	`ec2:DescribeCustomerGateways`
		February 17, 2023

Route53RecoveryReadinessServiceRolePolicy – Updated policy	Route 53 ARC added a new permission to query information about Lambda functions. Route 53 ARC uses the following permission to query information about Lambda functions to run readiness checks and determine the readiness status for the functions. `lambda:ListProvisionedConcurrencyConfigs`	August 31, 2022

Change	Description	Date
AmazonRoute53RecoveryControlConfigFullAccess – Updated policy	Removed Amazon Route 53 permissions from the policy and added note listing the optional permissions.	May 26, 2022
AmazonRoute53RecoveryControlConfigFullAccess – Updated policy	Added missing required Amazon Route 53 permissions to the policy.	April 15, 2022
AmazonRoute53RecoveryClusterReadOnlyAccess – Updated policy	ARC added a new permission, `route53-recovery-cluster:ListRoutingControls`, to allow listing routing control ARNs with high availability.	March 15, 2022
AmazonRoute53RecoveryControlConfigReadOnlyAccess – Updated policy	ARC added a new permission, `route53-recovery-control-config:ListTagsForResources`, to allow listing tags for a resource.	December 20, 2021
Route53RecoveryReadinessServiceRolePolicy – Updated policy	Route 53 ARC added a new permission to query information about Amazon API Gateway. Route 53 ARC uses the permission, `apigateway:GET`, to query information about API Gateway to run readiness checks and determine the readiness status.	October 28, 2021

Change	Description	Date
AmazonRoute53RecoveryReadinessReadOnlyAccess – Added new permissions	ARC added two new permissions to AmazonRoute53RecoveryReadinessReadOnlyAccess: ARC uses `route53-recovery-readiness:GetArchitectureRecommendations` and `route53-recovery-readiness:GetCellReadinessSummary` to allow read-only access to these actions for working with recovery readiness.	October 15, 2021

Change	Description	Date

Route53RecoveryReadinessServiceRolePolicy – Updated policy	Route 53 ARC added new permissions to query information about Lambda functions.
	
	Route 53 ARC uses the following permissions to query information about Lambda functions to run readiness checks and determine the readiness status for those functions.
	
	`lambda:GetFunctionConcurrency`
	
	`lambda:GetFunctionConfiguration`
	
	`lambda:GetProvisionedConcurrencyConfig`
	
	`lambda:ListAliases`
	
	`lambda:ListVersionsByFunction`
	
	`lambda:ListEventSourceMappings`
	
	`lambda:ListFunctions`
		October 8, 2021

Change	Description	Date
Route53RecoveryReadinessServiceRolePolicy – Added new managed policies	ARC added the following new managed policies: AmazonRoute53RecoveryReadinessFullAccess AmazonRoute53RecoveryReadinessReadOnlyAccess AmazonRoute53RecoveryClusterFullAccess AmazonRoute53RecoveryClusterReadOnlyAccess AmazonRoute53RecoveryControlConfigFullAccess AmazonRoute53RecoveryControlConfigReadOnlyAccess	August 18, 2021
ARC started tracking changes	ARC started tracking changes for its AWS managed policies.	July 27, 2021

Troubleshooting Amazon Route 53 Application Recovery Controller identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Application Recovery Controller (ARC) and IAM.

Topics

- I am not authorized to perform an action in Route 53 ARC
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Route 53 ARC resources

I am not authorized to perform an action in Route 53 ARC

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your credentials.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `route53-recovery-readiness:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 route53-recovery-readiness:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `route53-recovery-readiness:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Route 53 ARC.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Route 53 ARC. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Route 53 ARC resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Route 53 ARC supports these features, see How Amazon Route 53 Application Recovery Controller capabilities work with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Logging and monitoring in Amazon Route 53 Application Recovery Controller

Monitoring is an important part of maintaining the availability and performance of Amazon Route 53 Application Recovery Controller and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Route 53 ARC resources and activity, and responding to potential incidents, for example, AWS CloudTrail and Amazon CloudWatch.

For information about monitoring for each capability in ARC, see the following topics:

- Logging and monitoring for zonal shift
- Logging and monitoring for zonal autoshift
- Logging and monitoring for routing control

- Logging and monitoring for readiness check

Compliance validation for Amazon Route 53 Application Recovery Controller

Third-party auditors assess the security and compliance of Amazon Route 53 Application Recovery Controller as part of multiple AWS compliance programs. These include SOC, PCI, HIPAA, and others.

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

  > **Note**
  >
  > Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of

Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).

- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon Route 53 Application Recovery Controller

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Route 53 ARC offers several features to help support your data resiliency and backup needs.

Infrastructure security in Amazon Route 53 Application Recovery Controller

As a managed service, Amazon Route 53 Application Recovery Controller is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices

for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Route 53 ARC through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Document history for the Amazon Application Recovery Controller (ARC) Developer Guide

The following entries describe important changes made to the Amazon Application Recovery Controller (ARC) documentation.

- **Version:** latest
- **Latest documentation update:** July 12, 2024

Change	Description	Date
Autoshift observer notificat ions	With autoshift observer notifications, you can configure zonal autoshift to notify you, through Amazon EventBridge, whenever AWS starts an autoshift to shift traffic away from a potential ly impaired Availability Zone. You do not have to configure any specific resources with zonal autoshift to enable these separate notifications. For more information, see Using zonal autoshift with Amazon EventBridge.	July 12, 2024
Doc reorganization by each capability	Reorganizes the developer guide content to be siloed into sub-dev guides. That is, there are now separate sections that contain comprehensive informati on for each capability in	April 30, 2024

Change	Description	Date
	ARC: zonal shift and zonal autoshift for multi-AZ rec overy, and routing control and readiness check for multi-Region recovery. For more information, see What is Amazon Application Recovery Controller (ARC).	
Adds zonal autoshift capabilit y	Adds a new capability in ARC where you authorize AWS to shift away resource traffic for an application from an Availability Zone, on your behalf, to help reduce time to recovery during events. For more information, see Zonal autoshift in Amazon Application Recovery Controller (ARC).	November 30, 2023
Adds new service-linked role	Adds a new service-linked role, **AWSServiceRoleForZ onalAutoshiftPracticeRun**, for zonal autoshift practice runs. For more information, see Service-linked role permissio ns for AWSServiceRoleForZ onalAutoshiftPracticeRun.	November 30, 2023

Change	Description	Date
Adds cross-account support for clusters	Adds cross-account support for clusters in ARC with AWS Resource Access Manager, so that you can easily and securely use one cluster to host control panels and routing controls owned by several different AWS accounts. For more information, see Support cross-account for clusters in ARC.	October 18, 2023
Updates a managed policy	Updates the `AmazonRoute53RecoveryControlConfigReadOnly` managed policy to add permissions for `GetResourcePolicy` , to support returning details about AWS Resource Access Manager resource policies for shared resources. For more information, see AWS managed policies.	September 19, 2023

Change	Description	Date
Updated service-linked role	Added new permissions, `ec2:DescribeVpnGateways` and `ec2:DescribeCustomerGateways`, to the service-linked role for ARC, to support polling Amazon EC2 instances. For more information, see Using service-linked roles for ARC.	February 17, 2023
GA release for zonal shift	Supports the GA release of zonal shift for ARC, which includes attribute-based access control (ABAC) for managed resources that are registered in ARC for zonal shift. For more information, see Attribute-based access control (ABAC) with ARC.	January 10, 2023
Added new multi-AZ zonal shift	Added content describing a new service in ARC, zonal shift, for multi-AZ applications. You can start a zonal shift to temporarily move traffic for a load balancer resource away from an Availability Zone. For more information, see Zonal shift in ARC.	November 28, 2022

Change	Description	Date
Updated service-linked role	Added a new permission, `lambda:ListProvisionedConcurrencyConfigs`, to the service-linked role for ARC to query information about Lambda functions. For more information, see Using service-linked roles for ARC.	August 31, 2022
Updated managed policy	Updated the `AmazonRoute53RecoveryControlConfigFullAccess` managed policy to remove Amazon Route 53 permissions and list them as optional. For more information, see AWS managed policies for Amazon Application Recovery Controller (ARC).	May 26, 2022
Updated managed policy	Updated the `AmazonRoute53RecoveryControlConfigFullAccess` managed policy to include required Amazon Route 53 permissions. For more information, see AWS managed policies for Amazon Application Recovery Controller (ARC).	April 15, 2022

Change	Description	Date
Added CLI example for the new list routing controls API	Added example CLI command and best practices recommendations for the new list routing controls API operation included in the extremely reliable ARC data plane API. For more information, see List and update routing controls and states.	March 31, 2022
Added support for overriding safety rules	Added support for overridin g safety rules, which allows you to bypass routing control safeguards that are enforced with safety rules that you've configured. Safety rule overrides could be required, for example, in a "break glass" scenario during failover for di saster recovery. For more information, see Override safety rules to reroute traffic.	March 2, 2022
Added additional tagging support	Added support for tagging additional resources in ARC, including clusters, control panels, routing controls, and safety rules. For more information, see Tagging in Amazon Applicati on Recovery Controller (ARC).	December 20, 2021

Change	Description	Date
Updated managed policy	Updated the `AmazonRoute53RecoveryControlConfigReadOnly` managed policy to add permission to list tags for a resource. For more information, see AWS managed policies for Amazon Application Recovery Controller (ARC)	December 20, 2021
Added support for real-time alerts with EventBridge	Added support for EventBridge, which means that now you can add rules to get alerts and act on ARC readiness check status changes, for example, when a status changes from **READY** to **NOT READY**. For more information, see Using ARC with Amazon EventBridge.	December 20, 2021
Added routing control state code samples	Added code samples to illustrate trying cluster endpoints in sequence when you use API operations to get or update routing control states. For more information, see API examples for Amazon Application Recovery Controller (ARC).	November 16, 2021

Change	Description	Date
Added new permissions to a read-only policy	Added two new permissions to the policy `AmazonRoute53RecoveryReadinessReadOnlyAccess` : `route53-recovery-readiness:GetArchitectureRecommendations` and `route53-recovery-readiness:GetCellReadinessSummary` . For more information, see AWS managed policies for Amazon Application Recovery Controller (ARC).	November 9, 2021
Added support for Amazon API Gateway resource type	Added a new resource type, Amazon API Gateway, and updated the ARC service-linked role permissions so that ARC can audit API Gateway with readiness checks. For more information, see Readiness rules and supported resource types and Using service-linked roles for ARC.	October 28, 2021

Change	Description	Date
Added support for Lambda functions resource type	Added a new resource type, Lambda functions, and updated the ARC service-linked role permissions so that ARC can audit Lambda functions with readiness checks. For more information, see Readiness rules and supported resource types and Using service-linked roles for ARC.	October 8, 2021
Added links to CloudFormation and Terraform templates	Added links to downloadable AWS CloudFormation and Hashicorp Terraform templates to help you quickly get started with using ARC.For more information, see Recovery readiness with a new application.	September 13, 2021

Change	Description	Date
Added new managed policies	Added the following AWS managed policies for ARC: `AmazonRoute53RecoveryReadinessFullAccess`, `AmazonRoute53RecoveryReadinessReadOnlyAccess`, `AmazonRoute53RecoveryClusterFullAccess`, `AmazonRoute53RecoveryClusterReadOnlyAccess`, `AmazonRoute53RecoveryControlConfigFullAccess`, and `AmazonRoute53RecoveryControlConfigReadOnlyAccess`. For more information, see AWS managed policies for Amazon Application Recovery Controller (ARC).	August 18, 2021
Started tracking AWS managed policies for Amazon Application Recovery Controller (ARC)	Updates for managed policies will be tracked from the initial release date forward. For more information, see AWS managed policies for Amazon Application Recovery Controller (ARC).	July 27, 2021

Change	Description	Date
Initial release of Amazon Application Recovery Controller (ARC)	ARC improves application availability by centrally coordinating failovers within an AWS Region or across multiple Regions. ARC provides readiness checks to ensure that your applications are scaled to handle failover traffic and configured to route around failures. It also provides extremely reliable routing control so that you can recover applications by rerouting traffic, for example, across Availability Zones or Regions. For more information, see What is ARC?.	July 27, 2021