AWS Decision Guide

Choosing a modern application strategy

Choosing a modern application strategy: AWS Decision Guide

Table of Contents

Choosing a modern application strategy

Taking the first step

Purpose	Help determine which modern application development approach is the best fit for your organization.
Last updated	November 23, 2023
Covered services	• Amazon ECS • Amazon EKS • AWS App Runner • AWS Fargate • AWS Lambda • Red Hat OpenShift Service on AWS

Introduction

Amazon Web Services (AWS) provides you with the flexibility to choose different compute options to build and run modern applications that map to your business needs. We provide you with access to the right operational model for your compute choice.

Developers and data engineers prefer a level of autonomy when choosing which compute models match which workloads. When initially developing modern applications, development teams need to manage and operate their applications directly.

As more workloads are developed, you might decide to create central platform or operations teams. The function of these central teams varies - some provide standard architectures and templated patterns for development teams to use, and others operate and manage workloads on behalf of multiple development teams.

These central teams strive to create standards for controlling costs, achieving the right performance and security, simplifying operations, and providing common architecture patterns. Achieving the right balance between autonomy and standardization is a challenge that enterprises and other large organizations often deal with.

It is common to choose one of two operational models to meet this challenge: serverless compute or Kubernetes.

Decision guide co-author Josh Kahn reflects on modern applications development strategy choices.

Understand

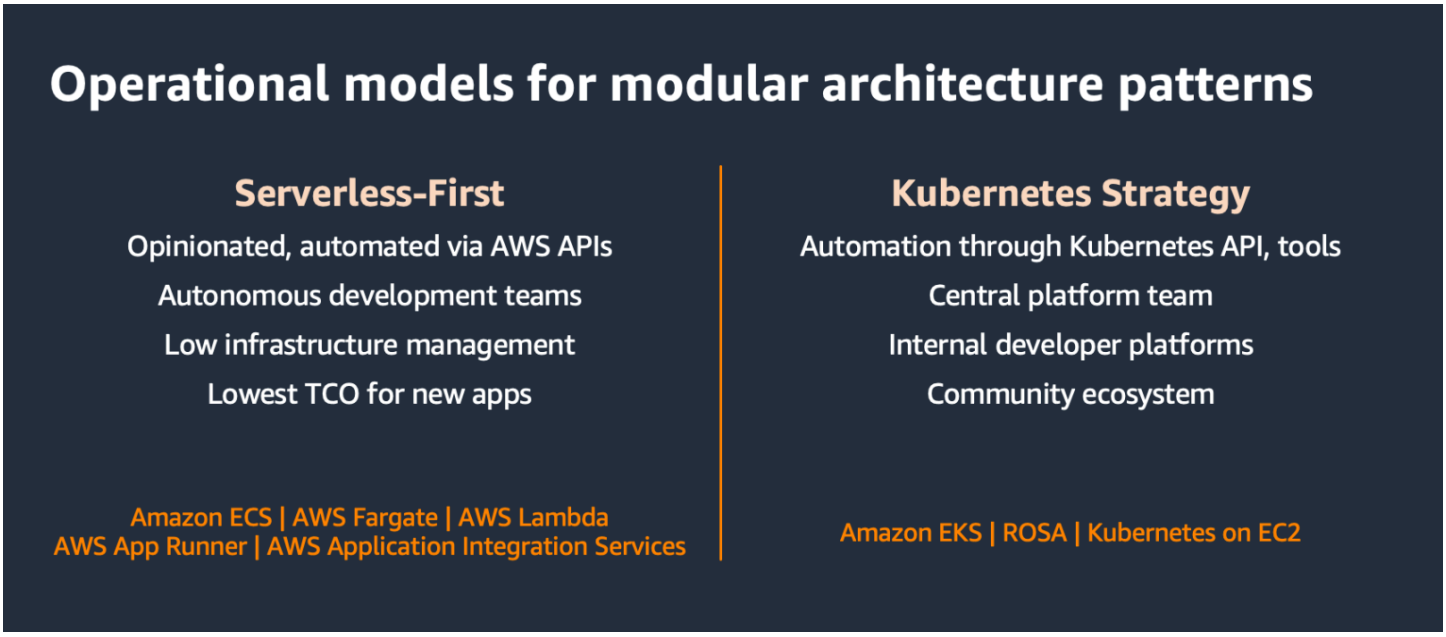
Developers and data engineers may have different compute requirements. For example, a developer might choose AWS Lambda because it is optimized for event-driven patterns and gives access to hundreds of managed integrations.

Alternatively, a data engineer might choose an open-source framework like KubeFlow or Ray on Amazon Elastic Kubernetes Service (Amazon EKS) because it simplifies deployment of machine-learning models but allows access to the right high-powered instances.

Guide co-author Roland Barcia discusses the key criteria to consider in choosing your strategy.

Each role tends to develop skills in technology stacks over time, and has preferences about the tools they use, both look at their compute choice on a workload-by-workload basis, which means operational roles need to support a variety of workloads since they often work across teams. These roles include platform engineers, cloud administrators, or site reliability engineers (SREs).

Those in operational roles are challenged to provide autonomy for developers and data engineers while making sure they can deploy, operate, and monitor all workloads consistently to meet security, performance, resiliency, and cost requirements. Over time, as you develop more modern applications, these roles need to standardize on the tools to automate the deployment and monitoring of their workloads.



The choice between serverless compute and Kubernetes as an operational model is often driven by the need to have the right balance between autonomy and standardization with the number of resources one dedicates to running and operating workloads. Many workloads can be built successfully using either of these options. But, for some workloads, there are inherent advantages of one over the other.

A good team can make the compute choice entirely transparent to developers. Poor choices can limit developers options, and lead to sub-standard outcomes. Operators are always affected by the selected operating model and it will determine needs such as common libraries and networking configuration - as well as how the organization will interact with underlying services it needs to configure.

The operational model will often be determined by your organizational structure and skill around automation and operations. These roles might be distributed across development teams, different parts of your organization, or be centralized. The structure and skills of these teams are often among the most important considerations when choosing between a serverless or Kubernetes operational model.

The serverless operational model prioritizes shifting much of the work involved in provisioning and managing compute resources to AWS. This model includes services such as AWS Lambda and the Amazon Elastic Container Service (Amazon ECS).

The second model, Kubernetes on AWS, meets an important need for organizations that prioritize the Cloud Native Computing Foundation (CNCF) ecosystem with a managed experience using Amazon EKS.

Consider

For developers and data engineers, we recommend that you evaluate the most appropriate compute option on a workload-by-workload basis within your operational strategy. Here are some of the key criteria to consider when determining your strategy.

Organizational structure and skill

Enterprises tend to organize their development teams around one of the following models:

- **Distributed:** Each team owns their development, deployment, and operational functions. This model provides a great level of autonomy and allow teams to innovate at their own pace.
- **Centralized:** A centralized team maintains standards, creates automations, and facilitates the sharing of knowledge and best practices.

These are separate ends of a spectrum, and organizations may start with a distributed model and later migrate towards a centralized model as the number of workloads increases and the need for consistency across cost, performance and security of workloads becomes an important factor. However, in both models, serverless can reduce the infrastructure management overhead.

Another model organizations adopt, especially in the Kubernetes space, are the formation of central teams that specialize in platform engineering. These skilled engineering teams have developer skills and build and maintain platforms made up of common automation, deployment, and observability tools. Amazon EKS is often the choice for these organizations.

Identifying the structure and skill of your development teams can be critical to choosing the appropriate platform.

Operational model

Organizations standardize on automation technologies to take full benefit of the cloud. The strategy and tools used by infrastructure, platform, and DevOps teams often drives decisions.

For example, organizations have choices for tooling that automates the creation, configuration, and maintenance of infrastructure, resources, and workloads. These tools include infrastructure-as-code (IaC) tools such as Terraform, CDK, AWS CloudFormation, or other community tools.

Organizations that use Kubernetes might use Kubernetes-based tools for automation using GitOps tools such as ArgoCD and Flux and Kubernetes API-based cloud provisioning tools like AWS Controllers for Kubernetes (ACK) or the CNCF Crossplane project.

These tooling choices extend to tools that extend to security, testing, networking, observability, performance, and more. For each of these categories, there is a stack of available tools.

These automation tools often have built in integrations and accelerators for compute choices. For example, AWS Serverless Application Model (AWS SAM) is optimized for serverless developers to build and quickly deploy Lambda functions. Other tools, like ArgoCD, automate the deployment and configuration of Kubernetes workloads.

As customers increase workloads, it can be a burden to support a large number of tooling choices. We recommend standardizing a set that support the most workload patterns.

Workload characteristics

It is important to evaluate the most appropriate compute option on a workload-by-workload basis within the default strategy. You should always strive to achieve the desired performance, security, and cost benefit for each workload.

A good standardization strategy will allow for different use cases such as microservices, modernized monoliths, event driven architectures, tools built by operation teams, and data processing workloads, such as machine learning, batch processing, and stream processing.

These workloads have different architectural characteristics. The strategy adopted should allow flexibility to support all the stages of a developer or data scientist workflow.

- **Application developer:** Needs to run multiple environments such as development, prototyping, test, staging, and production.
- **Data engineer:** These workflows involve streaming and acting on large data models, cleaning, training, running inference with models, and building applications and data pipelines that experiment with the data, including Jupyter Notebooks.
- **AI/ML scientist:** Generative Artificial Intelligence (AI) Large Language Models (LLM) may require specialized compute instances such as AWS Trainium or other GPU-based architecture.

Integrations

Applications do not exist in isolation. They are supported by technologies such as databases, messaging, streaming, orchestration, and other services. An effective modern app development strategy requires integration with these services. Managed integrations simplify operational overhead as much as management of the underlying infrastructure.

AWS serverless compute options, such as AWS Lambda, are integrated into the AWS ecosystem. Lambda can subscribe to events from more than 250 other services.

AWS-managed offerings for Kubernetes also provide integrations with many AWS managed offerings. For example, you can use AWS Controllers for Kubernetes to provision native AWS resources using a Kubernetes API. In addition, Kubernetes has a rich ecosystem, offering integration with numerous open-source projects. Amazon EKS works with many other AWS services to provide additional solutions for business challenges.

Prototyping

Many organizations need to create experiments to validate ideas. The ability to provide an environment where you can quickly write, deploy, and validate ideas is essential for a healthy environment.

This environment is often overlooked when developing a modern app development strategy, but the ability to innovate may depend on it. Enabling teams to use services that allow builders to rapidly build, test, and iterate help in discovering new business opportunities and receive feedback faster.

Serverless compute options like AWS Lambda and AWS App Runner are optimized to enable organizations write code quickly, deploy it, and change it. These capabilities provide useful options for doing fast prototyping work that doesn't require making a lot of choices upfront. Developers can quickly turn an idea into a modern, working application. The AWS serverless compute option provides choices to prototype with minimal cost or operational overhead.

Other options can be used for prototyping: for Amazon ECS, you can create dedicated clusters and use AWS Fargate or dedicated nodes for prototyping, for Kubernetes, platform teams can create dedicated clusters or have namespaces within a cluster dedicated to prototyping teams. Open source projects such as CNCF BuildPacks or Knative can be used to simplify the configuration experience.

Choose

AWS offers different container options, such as Amazon ECS, serverless containers with AWS Fargate, and AWS App Runner, and different Kubernetes options, such as Amazon EKS, ROSA, and self-managed Kubernetes on Amazon EC2.

The following comparison table can help you determine your approach based on your workload requirements. You might choose pieces of both approaches, or have different teams that use different approaches. It is not uncommon to see a very large organizations have departments with different strategies.

Modern application approach	When would you use it?	What workload is it optimized for?	Serverless services
Serverless	Use when AWS managed services and tools are your first choice, such as AWS Lambda, AWS App Runner, and Amazon ECS.	Optimized for enabling developers to focus solely on writing code without the need to manage or provision servers, minimizing operational overhead.	Amazon ECS AWS App Runner AWS Fargate AWS Lambda
Kubernetes	Use when Kubernetes is your primary compute platform interface.	Optimized for teams with central platforms teams that invest in platform engineering skills. Platform engineers are skilled at keeping clusters up to date with the fast-moving CNCF Kubernetes versioning strategy.	Amazon EKS Red Hat OpenShift Service on AWS (ROSA)

Use

Now that you have determined which approach best fits your workload for your environment, we recommend that you review the following service-specific resources to help you begin implementing your approach. This include links to in-depth documentation, hands-on tutorials, and other key assets to help get you started.

Amazon ECS

Getting started with Amazon ECS

We provide an introduction to the tools available to access Amazon ECS and introductory step-by-step procedures to run containers.

Explore the guide

Tutorials for Amazon ECS

Explore more than a dozen tutorials on how to perform common tasks—including the creation of clusters and VPCs.

Get started with the tutorials

What's new and what's next with Amazon ECS

Learn what's new since the launch of Amazon ECS Anywhere, new features of AWS Fargate, and a look ahead at the exciting enhancements to Amazon ECS

Watch the video

Amazon ECS deployment

This guide offers an overview of Amazon ECS deployment options on AWS and shows how it can be used to manage a simple containerized application.

Explore the guide

Amazon ECS workshop

This workshop is designed to educate those that might not be familiar with AWS Fargate, Amazon ECS, and possibly even Docker container workflow.

Explore the workshop

Deploy Docker containers on Amazon ECS

Learn how to run a Docker-enabled sample application on an Amazon ECS cluster behind a load balancer, test the application, and delete your resources to avoid charges.

Get started with the tutorial

AWS App Runner

Getting started with AWS App Runner

Learn how to configure the source code and deployment, service build, and service runtime to deploy your application to App Runner.

Get started with the tutorial

AWS App Runner: From code to a scalable, secure web application in minutes

Explore how AWS App Runner was designed to make it easier for you to deploy web apps and APIs to the cloud, regardless of the language they are written in, even for teams that lack prior experience deploying and managing containers or infrastructure.

Read the blog

Deploy a web app using AWS App Runner

Learn how to deploy a containerized web app using AWS App Runner. Start with your source code or a container image. App Runner automatically builds and deploys the

web application and load balances traffic with encryption.

Get started with the tutorial

AWS Fargate

Getting started with AWS Fargate

Understand the basics of AWS Fargate, a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of EC2 instances.

Explore the guide

Getting started with the console using Linux containers on AWS Fargate

Get started with Amazon ECS on AWS Fargate by using the Fargate launch type for your tasks in the Regions where Amazon ECS supports AWS Fargate.

Explore the guide

Creating a cluster with a Fargate Linux task using the AWS CLI

Learn how to set up a cluster, register a task definition, run a Linux task, and perform other common scenarios in Amazon ECS with the AWS CLI.

Get started with the tutorial

AWS Lambda

What is AWS Lambda

Learn more about AWS Lambda, a compute service that lets you run code without provisioning or managing servers.

Explore the guide

Guide to AWS Lambda Pricing

Explore and understand AWS Lambda pricing. You are charged based on the number of requests for your functions and the duration it takes for your code to start.

Explore the guide

Using AWS Lambda with other services

Explore common use cases, learn how invocation works and includes a table that covers the services that work with Lambda and how it can be invoked from that service.

Explore the guide

Amazon EKS

Getting started with Amazon EKS

Learn more about Amazon EKS, a managed service that you can use to run Kubernetes

Amazon EKS deployment

on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

Explore the guide

Explore Amazon EKS deployment options on AWS and learn how it can be used to manage a general containerized application.

Explore the guide

Amazon EKS Quick Start Reference Deployment

Using a Quick Start reference deployment guide, we provide step-by-step instructions for deploying Amazon Elastic Kubernetes Service (Amazon EKS) clusters.

Explore the guide

Deploy a Kubernetes application

Learn how to deploy a containerized application onto a Kubernetes cluster managed by Amazon EKS.

Get started with the tutorial

Amazon EKS workshop

Explore practical exercises to learn about Amazon EKS.

Visit the workshop

ROSA

Why would you use ROSA?

Getting started with Red Hat OpenShift Service on AWS

Learn how to get started using Red Hat OpenShift Service on AWS

Explore the guide

Learn when you might use Red Hat OpenShift over standard Kubernetes and explores ROSA on AWS in depth.

Watch the video

Explore

Architecture diagrams	Whitepapers	AWS Solutions
Explore reference architecture diagrams to help you implement your modern app development approach.	Explore whitepapers to learn best practices in implementing your modern app development approach.	Explore vetted solutions and architectural guidance for common modern app development use cases.
Explore architecture diagrams	Explore whitepapers	Explore solutions

Document history

The following table describes the important changes to this decision guide. For notifications about updates to this guide, you can subscribe to an RSS feed.

Change	Description	Date
Initial publication	Guide first published.	April 23, 2024