AWS Whitepaper

Overview of Deployment Options on AWS

Overview of Deployment Options on AWS: AWS Whitepaper

Table of Contents

Overview of Deployment Options on AWS

Publication date: **May 31, 2024** (*Document Revisions*)

Abstract

Amazon Web Services (AWS) offers multiple options for provisioning infrastructure and deploying your applications. Whether your application architecture is a simple three-tier web application or a complex set of workloads, AWS offers deployment services to meet the requirements of your application and your organization.

This whitepaper is intended for individuals looking for an overview of the different deployment services offered by AWS. It lays out common features available in these deployment services, and articulates basic strategies for deploying and updating application stacks.

Introduction

Designing a deployment solution for your application is a critical part of building a well-architected application on AWS. Based on the nature of your application and the underlying services that it requires, you can use AWS services to create a flexible deployment solution that can be tailored to fit the needs of both your application and your organization.

The constantly growing catalog of AWS services not only complicates the process of deciding which services will compose your application architecture, but also the process of deciding how you will create, manage, and update your application. When designing a deployment solution on AWS, you should consider how your solution will address the following capabilities:

- **Provision** - Create the raw infrastructure or managed service infrastructure required for your application.
- **Configure** - Customize your infrastructure based on environment, runtime, security, availability, performance, network or other application requirements.
- **Deploy** - Install or update your application components onto infrastructure resources and manage the transition from a previous application version to a new application version.
- **Scale** - Proactively or reactively adjust the amount of resources available to your application based on a set of user-defined criteria.
- **Monitor** - Provide visibility into the resources that are launched as part of your application architecture. Track resource usage, deployment success or failure, application health, application logs, configuration drift, and more.

This whitepaper highlights the deployment services offered by AWS and outlines strategies for designing a successful deployment architecture for any type of application.

AWS Deployment Services

The task of designing a scalable, efficient, and cost-effective deployment solution should not be limited to how you will update your application version, but should also consider how you will manage supporting infrastructure throughout the complete application lifecycle. Resource provisioning, configuration management, application deployment, software updates, monitoring, access control, and other concerns are all important factors to consider when designing a deployment solution.

AWS services can provide management capabilities for one or more aspects of your application lifecycle. Depending on your desired balance of control (manual management of resources) versus convenience (AWS management of resources) and the type of application, these services can be used on their own or combined to create a feature-rich deployment solution. This section will provide an overview of the AWS services that can be used to enable organizations to more rapidly and reliably build and deliver applications.

AWS CloudFormation

AWS CloudFormation is a service that enables customers to provision and manage almost any AWS resource using a custom template language expressed in YAML or JSON. An AWS CloudFormation template creates infrastructure resources in a group called a *stack*, and allows you to define and customize all components needed to operate your application while retaining full control of these resources. Using templates introduces the ability to implement version control on your infrastructure, and the ability to quickly and reliably replicate your infrastructure.

AWS CloudFormation offers granular control over the provisioning and management of all application infrastructure components, from low-level components such as route tables or subnet configurations, to high-level components such as CloudFront distributions. AWS CloudFormation is commonly used with other AWS deployment services or third-party tools, combining AWS CloudFormation with more specialized deployment services to manage deployments of application code onto infrastructure components.

AWS offers extensions to the CloudFormation service in addition to its base features:

- AWS Cloud Development Kit (AWS CDK) is an open source software development kit (SDK) to programmatically model AWS infrastructure with TypeScript, JavaScript, Python, Java, or C#/.NET.

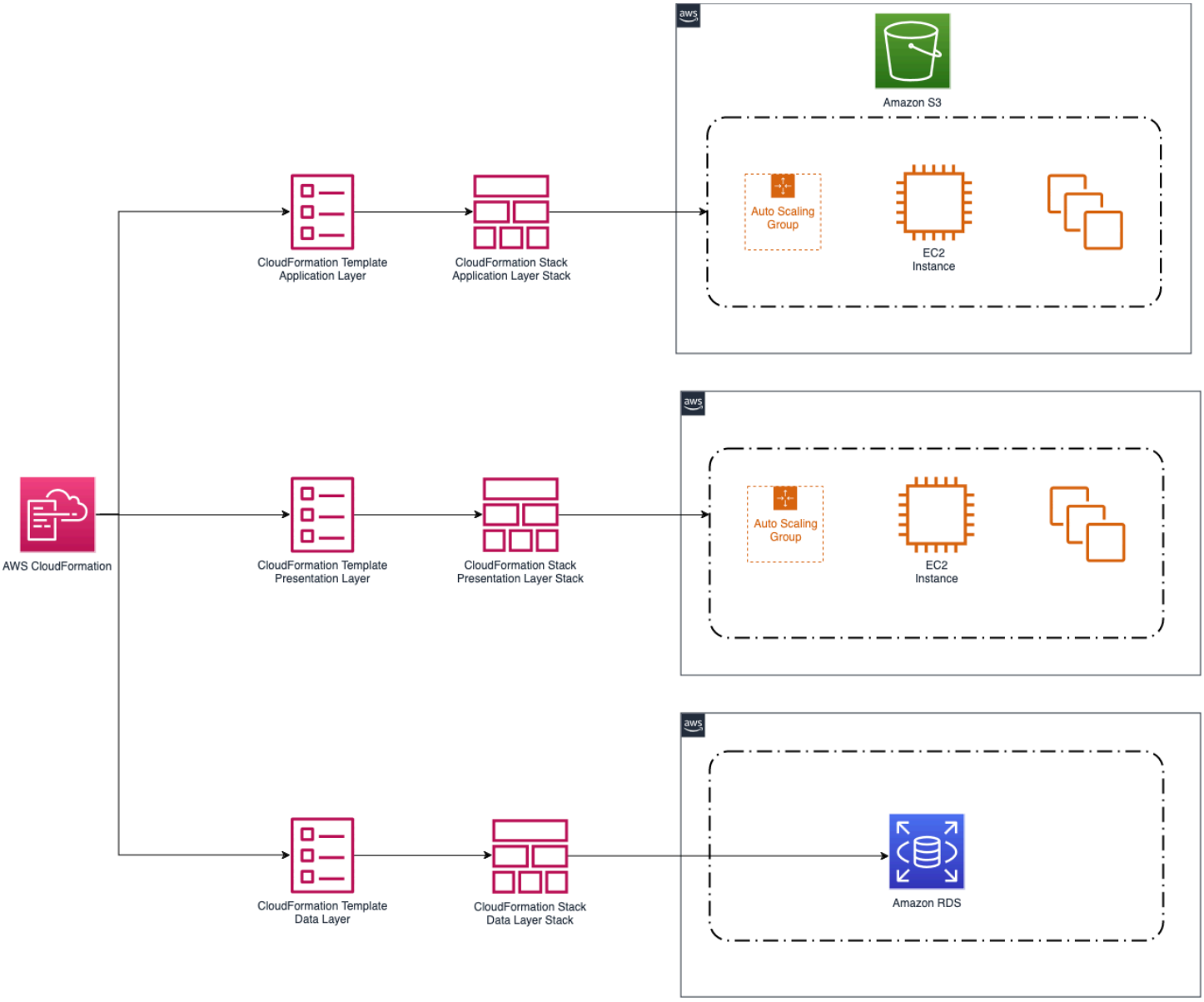
- AWS Serverless Application Model (AWS SAM) is an open source framework to simplify building serverless applications on AWS. It provides shorthand syntax to express functions, APIs, databases, and event source mappings.

Table 1: AWS CloudFormation deployment features

Capability	Description
Provision	CloudFormation will automatically create and update infrastructure components that are defined in a template. Refer to AWS CloudFormation Best Practices for more details on creating infrastructure using AWS CloudFormation templates.
Configure	AWS CloudFormation templates offer extensive flexibility to customize and update all infrastructure components. Refer to AWS CloudFormation Template Anatomy for more details on customizing templates.
Deploy	Update your AWS CloudFormation templates to alter the resources in a stack. Depending on your application architecture, you might need an additional deployment service to update the application version running on your infrastructure. Refer to Deploying Applications on Amazon EC2 with AWS CloudFormation for more details on how AWS CloudFormation can be used as a deployment solution.
Scale	AWS CloudFormation will not automatic ally handle infrastructure scaling on your

Capability	Description
	behalf; however, you can configure auto scaling policies for your resources in a AWS CloudFormation template.
Monitor	AWS CloudFormation provides native monitoring of the success or failure of updates to infrastructure defined in a template, as well as *drift detection* to monitor when resources defined in a template do not meet specifica tions. Additional monitoring solutions will need to be in place for application-level monitoring and metrics. Refer to Monitoring the Progress of a Stack Update for more details on how AWS CloudFormation monitors infrastructure updates.

The following diagram shows a common use case for AWS CloudFormation. Here, AWS CloudFormation templates are created to define all infrastructure components necessary to create a simple three-tier web application. In this example, we are using bootstrap scripts defined in AWS CloudFormation to deploy the latest version of our application onto Amazon EC2 instances; however, it is also a common practice to combine additional deployment services with AWS CloudFormation (using AWS CloudFormation only for its infrastructure management and provisioning capabilities). Note that more than one AWS CloudFormation template is used to create the infrastructure. In the diagram, AWS CloudFormation is used to create all infrastructure components including IAM roles, VPCs, subnets, route tables, security groups, and Amazon S3 bucket policies. Separate AWS CloudFormation templates are used to build each domain of the application architecture.



AWS CloudFormation use case

AWS Elastic Beanstalk

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, .NET Core, PHP, Node.js, Python, Ruby, Go, or Docker on familiar servers such as Apache, Nginx, Passenger, and IIS. Elastic Beanstalk is a complete application management solution, and manages all infrastructure and platform tasks on your behalf.

With Elastic Beanstalk, you can quickly deploy, manage, and scale applications without the operational burden of managing infrastructure. Elastic Beanstalk reduces management complexity for web applications, making it a good choice for organizations that are new to AWS or wish to deploy a web application as quickly as possible.

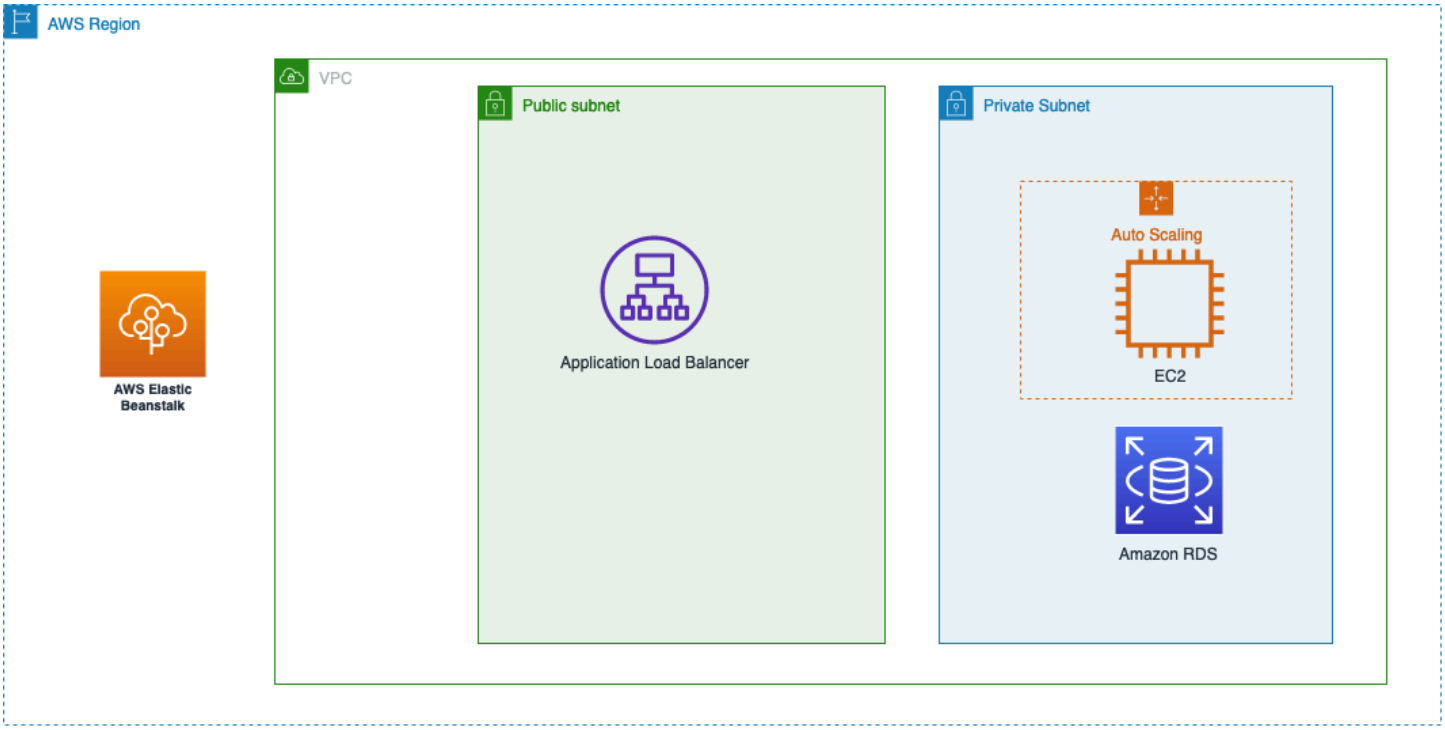
When using Elastic Beanstalk as your deployment solution, simply upload your source code and Elastic Beanstalk will provision and operate all necessary infrastructure, including servers, databases, load balancers, networks, and auto scaling groups. Although these resources are created on your behalf, you retain full control of these resources, allowing developers to customize as needed. Elastic Beanstalk meets the criteria for ISO, PCI, SOC 1, SOC 2, and SOC 3 compliance along with the criteria for HIPAA eligibility. This means applications running on Elastic Beanstalk can process regulated financial data or protected health information (PHI).

Table 2: AWS Elastic Beanstalk Deployment Features

Capability	Description
Provision	Elastic Beanstalk will create all infrastructure components necessary to operate a web application or service that runs on one of its supported platforms. If you need additiona l infrastructure, this will have to be created outside of Elastic Beanstalk. Refer to Elastic Beanstalk Platforms for more details on the web application platforms supported by Elastic Beanstalk.
Configure	Elastic Beanstalk provides a wide range of options for customizing the resources in your environment. Refer to Configuring Elastic Beanstalk environments for more information about customizing the resources that are created by Elastic Beanstalk.
Deploy	Elastic Beanstalk automatically handles application deployments, and creates an

Capability	Description
	environment that runs a new version of your application without impacting existing users. Refer to Deploying Applications to AWS Elastic Beanstalk for more details on application deployments with Elastic Beanstalk.
Scale	Elastic Beanstalk uses Elastic Load Balancing and Auto Scaling to automatically scale your application in and out based on its specific needs. Multiple availability zones give you an option to improve application reliability and availability. Refer to Auto Scaling Group for your Elastic Beanstalk Environment for more details about auto scaling with Elastic Beanstalk.
Monitor	Elastic Beanstalk offers built-in environme nt monitoring for applications including deployment success/failures, environment health, resource performance, and application logs. Refer to Monitoring an Environment for more details on full-stack monitoring with Elastic Beanstalk.
Graviton support	AWS Graviton arm64-based processors deliver the best price performance for your cloud workloads running in Amazon EC2. With AWS Graviton on Elastic Beanstalk, you can select Amazon EC2 instance types to meet optimizat ion needs of your workloads and benefit from improved price performance over a comparabl e x86-based processor.

Elastic Beanstalk makes it easy for web applications to be quickly deployed and managed in AWS. The following example shows a general use case for Elastic Beanstalk as it is used to deploy a simple web application. All application infrastructure (including security groups, IAM roles, and CloudWatch alarms) is created and managed by Elastic Beanstalk. The Amazon EC2 instances are automatically provisioned with runtime environment and deployment packages. Elastic Beanstalk environments can integrate with resources like Amazon Relational Database Service (Amazon RDS) that are created outside of Elastic Beanstalk.



AWS Elastic Beanstalk use case

AWS CodeDeploy

AWS CodeDeploy is a fully managed deployment service that automates application deployments to compute services such as Amazon EC2, Amazon Elastic Container Service (Amazon ECS), AWS Lambda, or on-premises servers. Organizations can use CodeDeploy to automate deployments of an application and remove error prone manual operations from the deployment process. CodeDeploy can be used with a wide variety of application content including code, serverless functions, configuration files, and more.

CodeDeploy is intended to be used as a *building block* service that is focused on helping application developers deploy and update software that is running on existing infrastructure. It is not an end-to-end application management solution, and is intended to be used in conjunction with other

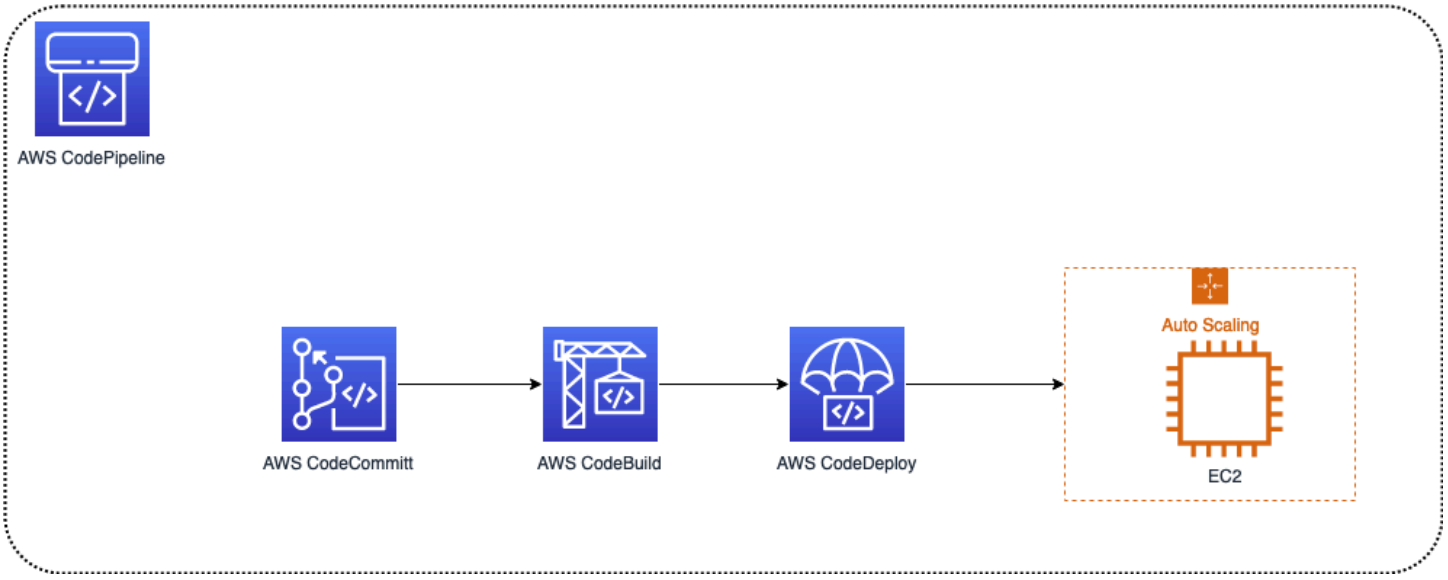
AWS deployment services such as AWS CodeStar, AWS CodePipeline, other AWS Developer Tools, and third-party services (see AWS CodeDeploy Product Integrations for a complete list of product integrations) as part of a complete CI/CD pipeline. Additionally, CodeDeploy does not manage the creation of resources on behalf of the user.

Table 3: AWS CodeDeploy deployment features

Capability	Description
Provision	CodeDeploy is intended for use with existing compute resources and does not create resources on your behalf. CodeDeploy requires compute resources to be organized into a construct called a *deployment group* in order to deploy application content. Refer to Working with Deployment Groups in CodeDeploy for more details on linking CodeDeploy to compute resources.
Configure	CodeDeploy uses an application specification file to define customizations for compute resources. Refer to CodeDeploy AppSpec File Reference for more details on the resource customizations with CodeDeploy.
Deploy	Depending on the type of compute resource that CodeDeploy is used with, CodeDeploy offers different strategies for deploying your application. Refer to Working with Deployments in CodeDeploy for more details on the types of deployment processes that are supported.
Scale	CodeDeploy does not support scaling of your underlying application infrastructure; however,

Capability	Description
	depending on your deployment configurations, it might create additional resources to support blue/green deployments.
Monitor	CodeDeploy can monitor the success or failure of deployments and offers a history of all deployments, but does not provide performance or application-level metrics. Refer to Monitoring Deployments in CodeDeploy for more details on the types of monitoring capabilities offered by CodeDeploy

The following diagram illustrates a general use case for CodeDeploy as part of a complete CI/CD solution. In this example, CodeDeploy is used in conjunction with additional AWS Developer Tools, namely AWS CodePipeline (automate CI/CD pipelines), AWS CodeBuild (build and test application components), and AWS CodeCommit (source code repository) to deploy an application onto a group of Amazon EC2 instances. CodeDeploy is used with other tools as part of a complete CI/CD pipeline. CodeDeploy manages deployment of application components onto compute resources that are part of a deployment group. All infrastructure components are created outside of CodeDeploy.



AWS CodeDeploy use case

AWS CodeDeploy for AWS Lambda

AWS CodeDeploy for AWS Lambda enables you to automate your serverless deployments, giving you greater control and visibility over your application releases. You can use CodeDeploy to deploy a new version of your serverless function to a small percentage of users or traffic and gradually increase traffic as you gain confidence in the new version. With CodeDeploy, you can define deployment groups, which represent a set of Lambda functions that receive traffic from the same event source. For example, you can create a deployment group for a set of Lambda functions that are initiated by API Gateway or an Amazon EventBridge rule. You can then create a deployment using CodeDeploy, which deploys the new version of your erverless function to a specified deployment group.

CodeDeploy also enables you to define a deployment configuration, which specifies the settings for a deployment, such as the deployment type, deployment strategy, and traffic shifting rules. You can use the Canary deployment strategy to deploy the new version of your serverless function to a small percentage of traffic and monitor the health and performance of the new version before increasing traffic to it.

By using CodeDeploy for serverless, you can automate your deployment process, reduce the time and effort required to release new versions of your application, and increase the stability and reliability of your serverless functions.

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service that supports Docker containers and allows you to easily run applications on a managed cluster. Amazon ECS eliminates the need to install, operate, and scale container management infrastructure, and simplifies the creation of environments with familiar AWS core features like Security Groups, Elastic Load Balancing, and AWS Identity and Access Management (IAM).

When running applications on Amazon ECS, you can choose to provide the underlying compute power for your containers with Amazon EC2 instances or with AWS Fargate, a serverless compute engine for containers. In either case, Amazon ECS automatically places and scales your containers onto your cluster according to configurations defined by the user. Although Amazon ECS does not create infrastructure components such as Load Balancers or IAM roles on your behalf, the Amazon ECS service provides a number of APIs to simplify the creation and use of these resources in an Amazon ECS cluster.

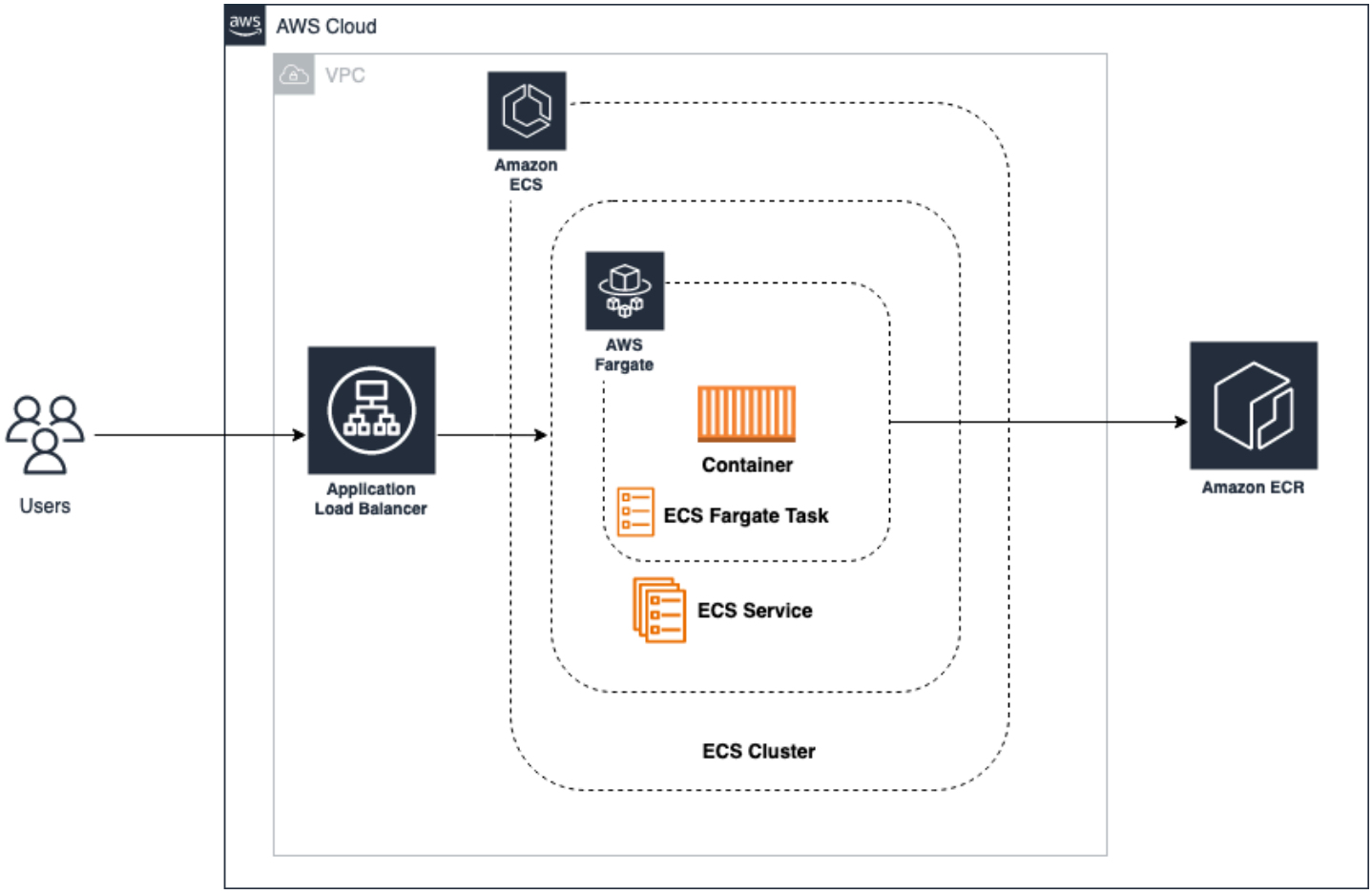
Amazon ECS allows developers to have direct, fine-grained control over all infrastructure components, allowing for the creation of custom application architectures. Additionally, Amazon ECS supports different deployment strategies to update your application container images.

Table 4: Amazon ECS deployment features

Capability	Description
Provision	Amazon ECS will provision new application container instances and compute resources based on scaling policies and Amazon ECS configurations. Infrastructure resources such as Load Balancers will need to be created outside of Amazon ECS. Refer to Getting Started with Amazon ECS for more details on the types of resources that can be created with Amazon ECS.
Configure	Amazon ECS supports customization of the compute resources created to run a container ized application, as well as the runtime conditions of the application containers (for example, environment variables, exposed ports, reserved memory/CPU). Customiza tion of underlying compute resources is only available if using Amazon EC2 instances. Refer to Creating a Cluster for more details on how to customize an Amazon ECS cluster to run containerized applications.
Deploy	Amazon ECS supports several deployment strategies for you containerized applications. Refer to Amazon ECS Deployment Types for more details on the types of deployment processes that are supported.

Capability	Description
Scale	Amazon ECS can be used with auto scaling policies to automatically adjust the number of containers running in your Amazon ECS cluster. Refer to Service Auto Scaling for more details on configuring auto scaling for your container ized applications on Amazon ECS.
Monitor	Amazon ECS supports monitoring compute resources and application containers with CloudWatch. Refer to Monitoring Amazon ECS for more details on the types of monitoring capabilities offered by Amazon ECS.

The following diagram illustrates Amazon ECS being used to manage a simple containerized application. In this example, infrastructure components are created outside of Amazon ECS, and Amazon ECS is used to manage the deployment and operation of application containers on the cluster



Amazon ECS use case

Note

- Application infrastructure (including Amazon Elastic Container Registry (Amazon ECR) repositories, Amazon ECS configurations, and Load Balancers) is provisioned and managed outside of your Amazon ECS deployment.
- Amazon ECS manages the deployment of application containers running inside the Amazon ECS service as *tasks* that are sourced from a container registry like Amazon ECR.

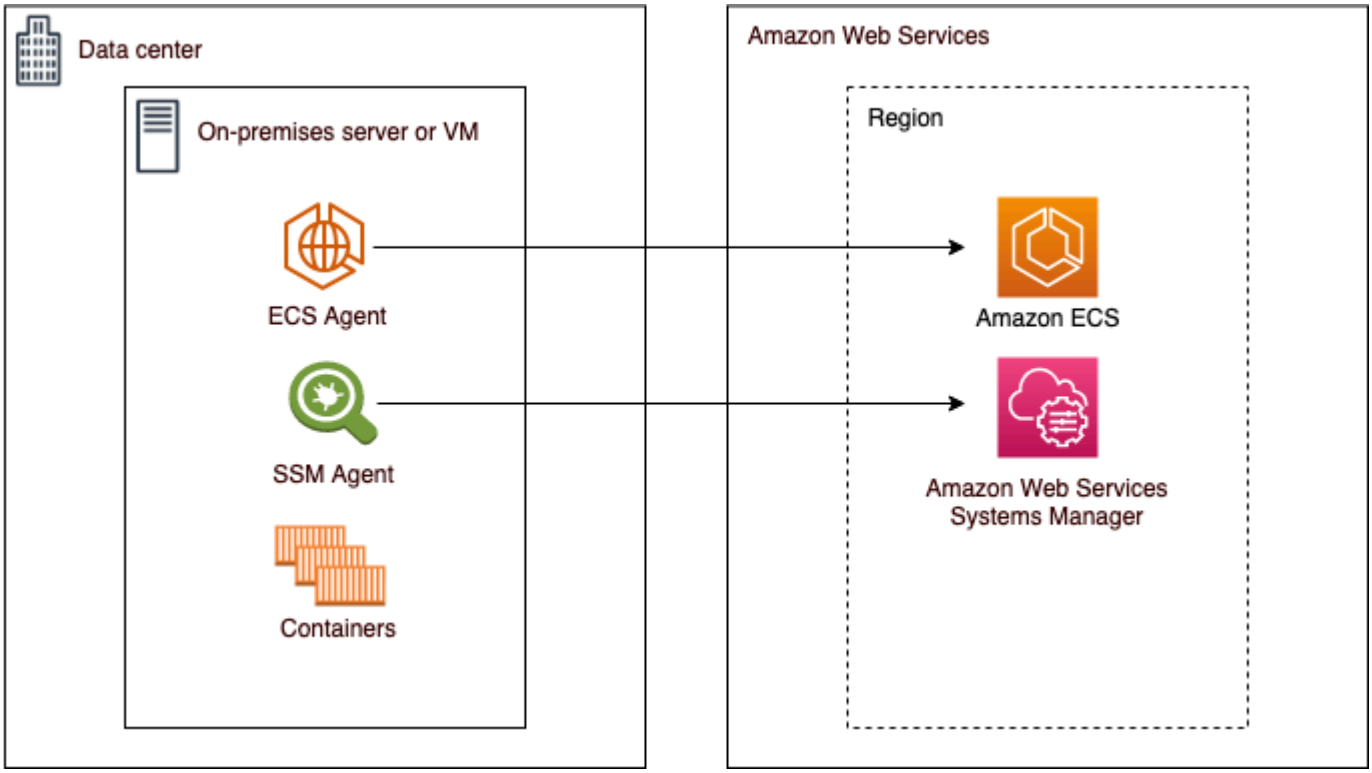
Amazon ECS supports multiple container instance types such as Linux and Windows, as well as external instance types such as an on-premises virtual machine (VM) with Amazon ECS Anywhere.

Amazon ECS Anywhere

Amazon ECS Anywhere allows you to run Amazon ECS tasks anywhere, whether it's on-premises or in other cloud environments. With Amazon ECS Anywhere, you can easily deploy and manage containerized applications across your hybrid infrastructure, while maintaining a consistent operational experience. The service works by extending the Amazon ECS platform to any environment, including on-premises data centers, remote offices, and other cloud environments. It enables you to use the same familiar Amazon ECS APIs and tooling to deploy and manage containers across all of your environments, without having to worry about the underlying infrastructure.

Amazon ECS Anywhere uses the Amazon ECS agent to manage the deployment and lifecycle of containers, enabling you to use the same Amazon ECS task definitions and configuration files that you use in the AWS Cloud. This can help to simplify the process of deploying and managing containers across your hybrid infrastructure, and reduce the time and effort r equired for manual configuration and management.

With Amazon ECS Anywhere, you can also leverage other AWS services, such as IAM, AWS CloudFormation, and Amazon ECR, to manage your containerized applications. This can help to ensure that your applications are secure, compliant, and integrated with other AWS services.



Amazon ECS Anywhere architecture

Amazon Elastic Container Service on AWS Outposts

Amazon ECS on AWS Outposts is a fully managed AWS service that enables you to run Amazon ECS tasks on-premises, using the same APIs and tooling that you use in the AWS Cloud. With Amazon ECS on AWS Outposts, you can deploy and manage containerized applications in a consistent and familiar way, whether you're running them on-premises or in the cloud. AWS Outposts is a fully managed service that extends AWS infrastructure, services, APIs, and tools to your on-premises environments. With Amazon ECS on AWS Outposts, you can run Amazon ECS tasks on hardware that is dedicated to your organization, without having to worry about the underlying infrastructure. This can help to ensure that your applications are deployed in a secure and compliant manner, while also enabling you to take advantage of the flexibility and scalability of the cloud.

Amazon ECS on AWS Outposts works by deploying a set of AWS services and APIs to your on-premises environment, which enables you to run Amazon ECS tasks on dedicated hardware. This includes the Amazon ECS agent, which manages the deployment and lifecycle of containers, and the AWS Outposts infrastructure, which provides a secure and compliant environment for running containerized applications. With Amazon ECS on AWS Outposts, you can use the same Amazon ECS APIs and tooling that you use in the AWS Cloud, making it easy to deploy and manage containerized applications in a consistent and familiar way. This can help to reduce the time and effort required for manual configuration and management, and improve consistency and reliability across your hybrid infrastructure. Amazon ECS on AWS Outposts also integrates with other AWS services, such as IAM, AWS CloudFormation, and Amazon ECR, to manage your containerized applications. This can help to ensure that your applications are secure, compliant, and integrated with other AWS services.

Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service (Amazon EKS) is a fully-managed, certified Kubernetes conformant service that simplifies the process of building, securing, operating, and maintaining Kubernetes clusters on AWS. Amazon EKS integrates with core AWS services such as CloudWatch, Auto Scaling Groups, and IAM to provide a seamless experience for monitoring, scaling, and load balancing your containerized applications.

Amazon EKS provides a scalable, highly-available control plane for Kubernetes workloads. When you run applications on Amazon EKS, as with Amazon ECS, you can choose to provide the underlying compute power for your containers with Amazon EC2 instances or with AWS Fargate.

Amazon VPC Lattice is a fully managed application networking service built directly into the AWS networking infrastructure that you can use to connect, secure, and monitor your services across multiple accounts and virtual private clouds (VPCs). With Amazon EKS, you can leverage VPC Lattice through the use of the AWS Gateway API Controller, an implementation of the Kubernetes Gateway API. Using VPC Lattice, you can set up cross-cluster connectivity with standard Kubernetes semantics in a simple and consistent manner.

You can use Amazon EKS with any of the following deployment options:

- **Amazon EKS Distro** – Amazon EKS Distro is a distribution of the same open-source Kubernetes software and dependencies deployed by Amazon EKS in the cloud. Amazon EKS Distro follows the same Kubernetes version release cycle as Amazon EKS and is provided as an open-source project. To learn more, see Amazon EKS Distro.
- **Amazon EKS on AWS Outposts** – AWS Outposts enables native AWS services, infrastructure, and operating models in your on-premises facilities. Amazon EKS on AWS Outposts, you can choose to run extended or local clusters. With extended clusters, the Kubernetes control plane runs in an AWS Region and the nodes run on AWS Outposts. With local clusters, the entire Kubernetes cluster runs locally on AWS Outposts, including both the Kubernetes control plane and nodes.
- **Amazon EKS Anywhere** – Amazon EKS Anywhere is a deployment option for Amazon EKS that enables you to easily create and operate Kubernetes clusters on-premises. Both Amazon EKS and Amazon EKS Anywhere are built on the Amazon EKS Distro. To learn more about Amazon EKS Anywhere, see Running Hybrid Container workloads with Amazon EKS Anywhere, Amazon EKS Anywhere Overview, and Comparing Amazon EKS Anywhere to Amazon EKS.

When choosing which deployment options to use for your Kubernetes cluster, consider the following:

Table 5: Kubernetes deployment features

Feature	Amazon EKS	Amazon EKS on AWS Outposts	Amazon EKS Anywhere	Amazon EKS Distro
Hardware	AWS-supplied	AWS-supplied	Supplied by you	Supplied by you
Deployment location	AWS Cloud	Your data center	Your data center	Your data center

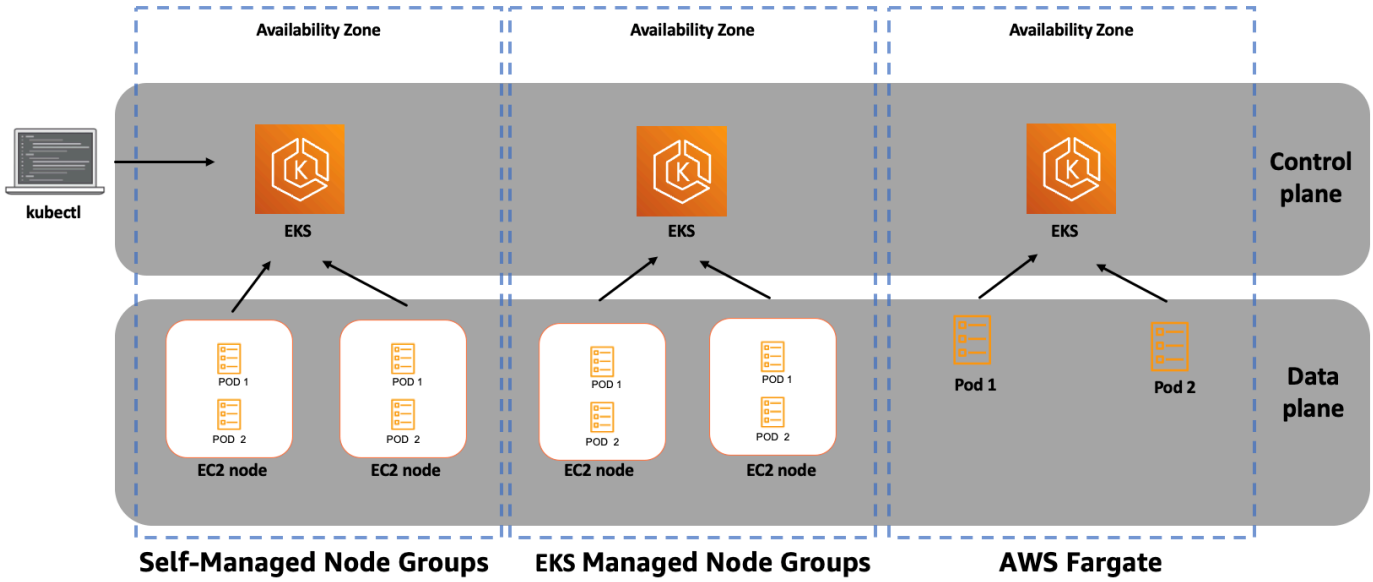
Feature	Amazon EKS	Amazon EKS on AWS Outposts	Amazon EKS Anywhere	Amazon EKS Distro
Kubernetes control plane location	AWS Cloud	AWS Cloud or your data center	Your data center	Your data center
Kubernetes data plane location	AWS Cloud	Your data center	Your data center	Your data center
Support	AWS support	AWS support	AWS support	OSS community support

Table 6: Amazon EKS deployment features

Capability	Description
Provision	Amazon EKS provisions certain resources to support containerized applications: • Load Balancers, if needed • Compute resources, or *workers* (Amazon EKS supports Windows and Linux) • Application Container Instances, or *pods* Refer to Getting Started with Amazon EKS for more details on Amazon EKS cluster provision ing.
Configure	Amazon EKS supports customization of the compute resources (workers) if you use Amazon EC2 instances to supply compute power. Amazon EKS also supports customiza tion of the runtime conditions of the applicati on containers (pods).

Capability	Description
	Refer to Worker Nodes and Fargate Pod Configuration documentation for more details.
Deploy	Amazon EKS supports the same deployment strategies as Kubernetes. See Writing a Kubernetes Deployment Spec -> Strategy for more details.
Scale	Amazon EKS scales workers with Kubernetes Cluster Autoscaler, and pods with Kubernetes Horizontal Pod Autoscaler and Kubernetes Vertical Pod Autoscaler. Amazon EKS also supports Karpenter, an open source, flexible, high-performance Kubernetes cluster autoscaler to help improve your application availability and cluster efficiency by rapidly launching right-sized compute resources in response to changing application load.
Monitor	The Amazon EKS control plane logs provide audit and diagnostic information directly to CloudWatch Logs. The Amazon EKS control plane also integrates with AWS CloudTrail to record actions taken in Amazon EKS. Refer to Logging and Monitoring Amazon EKS for more details.

Amazon EKS allows organizations to leverage open source Kubernetes tools and plugins, and can be a good choice for organizations migrating to AWS with existing Kubernetes environments. The following diagram illustrates Amazon EKS being used to manage a general containerized application.



Amazon EKS use case

Amazon EKS Anywhere

Amazon EKS Anywhere lets you create and operate Kubernetes clusters on your own infrastructure. Amazon EKS Anywhere builds on the strengths of Amazon EKS Distro and provides open-source software that's up to date and patched so you can have an on-premises Kubernetes environment that's more reliable than a self-managed Kubernetes offering.

Amazon EKS Anywhere creates a Kubernetes cluster on-premises to a chosen provider. Supported providers include Bare Metal (via Tinkerbell), CloudStack, and vSphere. To manage that cluster, you can run cluster create and delete commands from an Ubuntu or Mac Administrative machine.

AWS App Runner

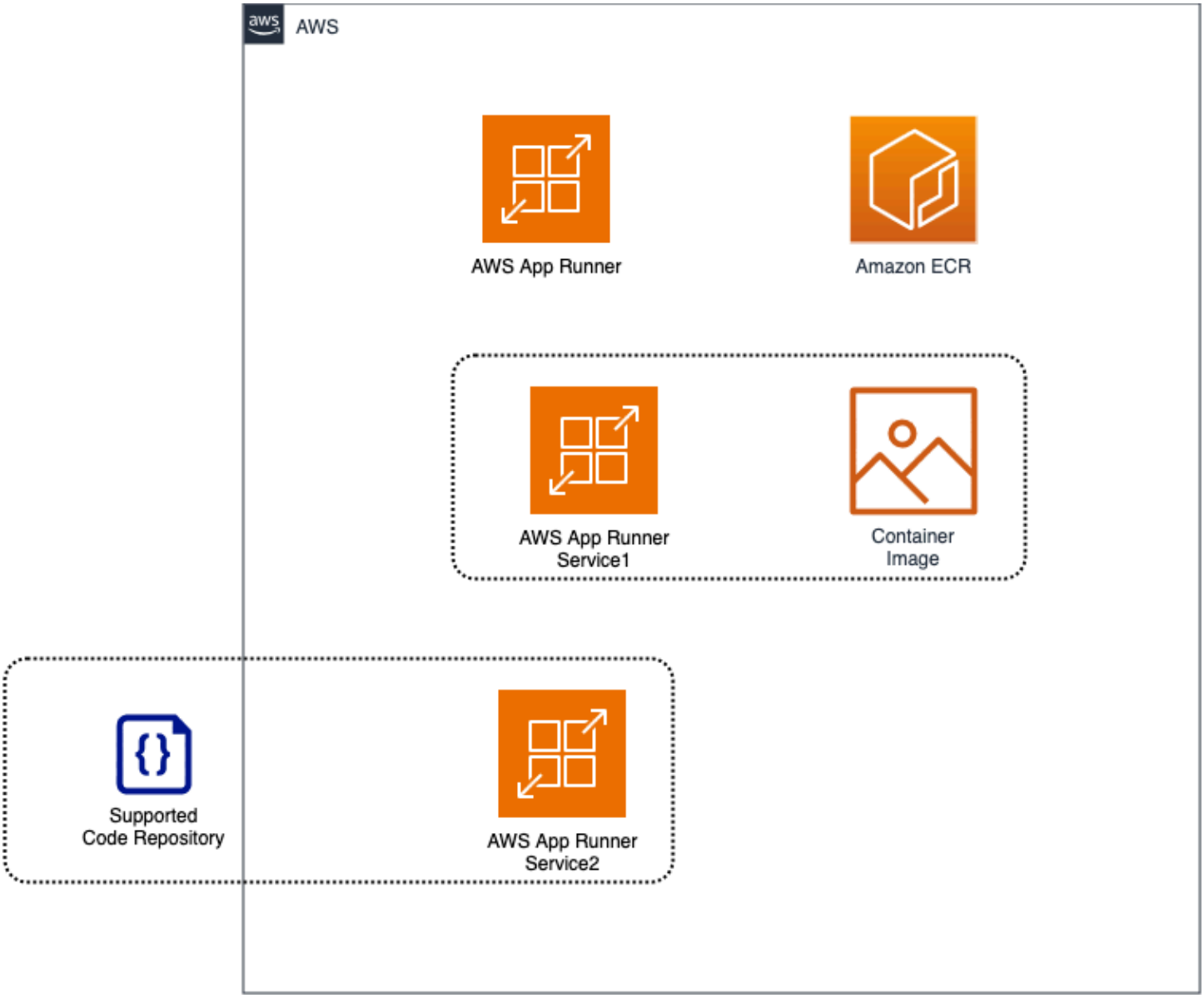
AWS App Runner is a fully managed container application service that lets you build, deploy, and run containerized web applications and API services without prior infrastructure or container experience. App Runner connects directly to your code or image repository. It provides an automatic integration and delivery pipeline with fully managed operations, high performance, scalability, and security.

App Runner takes your source code or source image from a repository and then creates and maintains a running web service for you in the AWS Cloud. Typically, you need to call just one

App Runner action, `CreateService`, to create your service. With a source image repository, you provide a ready-to-use container image that App Runner can deploy to run your web service. With a source code repository, you provide your code and instructions for building and running a web service and you target a specific runtime environment. App Runner supports several programming platforms, each with one or more managed runtimes for platform major versions. App Runner supports container images as well as runtimes and web frameworks including Node.js and Python. App Runner monitors the number of concurrent requests sent to your application and automatically adds additional instances based on request volume. If your application receives no incoming requests, App Runner will scale the containers down to a provisioned instance, a CPU-throttled instance ready to serve incoming requests within milliseconds.

At this time, App Runner can retrieve your source code from a GitHub repository, or retrieve your source image from Amazon ECR in your AWS account.

The following diagram shows an overview of the App Runner service architecture. In the diagram, there are two example services: one deploys source code from GitHub, and the other deploys a source image from Amazon ECR.



App Runner use case

App Runner supports full stack development, including both frontend and backend web applications that use HTTP and HTTPS protocols. These applications include API services, backend web services, and websites. App Runner supports container images as well as runtimes and web frameworks including Node.js and Python.

Amazon Lightsail

Amazon Lightsail is a simple and cost-effective cloud service that makes it easy for small businesses, startups, and individuals to deploy and manage their applications in the cloud. It provides a user-friendly interface that abstracts away much of the underlying infrastructure

management and makes it easy to launch and run applications in the cloud. With Lightsail, you can quickly deploy and manage virtual private servers (VPS), databases, and storage instances. The service provides pre-configured instances that are optimized for various workloads, such as WordPress, Drupal, and Joomla, among others. This can help to reduce the time and effort required to set up and configure your environment. Lightsail also provides an integrated load balancer and automatic scaling, enabling you to handle changes in traffic demand without manual intervention. The service also provides monitoring and alerting, so you can stay on top of the health and performance of your applications.

One of the key benefits of Lightsail is its simplicity and ease of use. The service is designed to be accessible to users with minimal cloud computing experience, making it a good option for small businesses or individuals who want to get started quickly in the cloud. Additionally, Lightsail is cost-effective, with predictable pricing that includes compute, storage, and data transfer.

Amazon Lightsail Containers

Amazon Lightsail Containers is a fully managed container service AWS that makes it easy to deploy and manage containerized applications in the cloud. It provides a simple and cost-effective way to launch and run containers using popular container management tools, such as Docker and Kubernetes.

Lightsail Containers provides an integrated environment for building, testing, and deploying containerized applications. It simplifies the process of deploying and managing containers by providing a user-friendly interface that abstracts away much of the underlying infrastructure management.

With Lightsail Containers, you can deploy your containerized applications to a VPC in just a few clicks. The service provides pre-configured container images for popular programming languages, such as Node.js, Python, Ruby, and Java. This can help to reduce the time and effort required to set up and configure your container environment.

Lightsail Containers also provides an integrated load balancer that can automatically distribute traffic across your container instances, improving application availability and scalability. Additionally, the service provides automatic scaling of container instances, enabling you to handle changes in traffic demand without manual intervention.

With Lightsail Containers, you can monitor the performance of your containerized applications using built-in metrics and logs. You can also integrate with other AWS services, such as Amazon S3,

Amazon RDS, and AWS CodePipeline, to create a fully automated and integrated CI/CD pipeline for your containerized applications.

Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS (ROSA) is a managed service that's available through the AWS Management Console. With ROSA, as a Red Hat OpenShift user, you can build, scale, and manage containerized applications on AWS. You can use ROSA to create Kubernetes clusters using the Red Hat OpenShift APIs and tools, and have access to the full breadth and depth of AWS services. ROSA streamlines moving on-premises Red Hat OpenShift workloads to AWS, and offers tight integration with other AWS services. You can also access Red Hat OpenShift licensing, billing, and support all directly through AWS.

Each ROSA cluster comes with a fully managed control plane and compute nodes. Installation, management, maintenance, and upgrades are performed by Red Hat SRE with joint Red Hat and Amazon support. Cluster services (such as logging, metrics, and monitoring) are available as well. Only Red Hat Enterprise Linux CoreOS (RHCOS) workers are supported by ROSA.

ROSA will integrate with a range of AWS compute, storage, database, analytics, machine learning, networking, mobile, and various application services, which will enable customers to benefit from the robust portfolio of AWS services that scale on-demand across the globe. These AWS native services will be directly accessible to quickly deploy and scale services through the same management interface.

AWS Local Zones

An AWS Local Zone is an extension of an AWS Region in close geographic proximity to your users. Local Zones have their own connections to the internet and support AWS Direct Connect. Resources that are created in a Local Zone can serve local users with low-latency communications. A Local Zone is represented by a Region code followed by an identifier that indicates the location (for example, us-west-2-lax-1a).

Amazon ECS supports workloads that use Local Zones when low latency or local data processing is a requirement. The Amazon ECS control plane will always run in the AWS Region.

Amazon EKS supports certain resources in Local Zones. This includes self-managed Amazon EC2 nodes, Amazon EBS volumes, and Application Load Balancers. The Amazon EKS managed

Kubernetes control plane always runs in the AWS Region. The Amazon EKS managed Kubernetes control plane can't run in the Local Zone. Because Local Zones appear as a subnet within your VPC, Kubernetes sees your Local Zone resources as part of that subnet.

AWS Wavelength

AWS Wavelength is an AWS infrastructure that allows you to deploy workloads closer to 5G-connected users and devices. You can use Wavelength to deploy Amazon EC2 instances, Amazon EKS clusters, and a suite of supported partner solutions available on the AWS Marketplace. Wavelength Zones are logically isolated data centers within telecommunication providers' networks that are connected back to the AWS Region through redundant, low latency, and high-throughput connectivity.

Some of the key features of Wavelength include the ability to create Amazon EC2 instances, Amazon EBS volumes, and Amazon VPC subnets and carrier gateways in Wavelength Zones. You can also use services that orchestrate or work with Amazon EC2, Amazon EBS, and Amazon VPC such as Amazon EC2 Auto Scaling, Amazon EKS clusters, Amazon ECS clusters, Amazon EC2 Systems Manager, Amazon CloudWatch, AWS CloudTrail, AWS CloudFormation, and Application Load Balancer. Wavelength services are part of a VPC connected over a reliable, high-bandwidth connection to an AWS Region for easy access to services including Amazon DynamoDB and Amazon Relational Database Service (Amazon RDS).

Additional Deployment Services

Amazon Simple Storage Service (Amazon S3) can be used as a web server for static content and single-page applications (SPA). Combined with Amazon CloudFront to increase performance in static content delivery, using Amazon S3 can be a simple and powerful way to deploy and update static content. More details on this approach can be found in Hosting Static Websites on AWS whitepaper.

AWS Proton

AWS Proton is a fully managed service that simplifies and automates the process of deploying and managing microservices and container-based applications. It provides a unified and consistent deployment experience that integrates with popular DevOps tools and services, making it easier to manage and streamline application development. Proton enables developers to define and

create application components, such as infrastructure, code, and pipelines, as reusable templates. These templates can be used to create multiple environments, such as development, testing, and production, and can be shared across teams or organizations. This approach helps to reduce the complexity of deploying and managing microservices and container-based applications, which can be time-consuming and error-prone.

AWS Proton provides pre-built templates for common types of microservices, such as web applications, APIs, and databases, that can be customized to meet specific needs. It also integrates with popular DevOps tools such as AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild, to enable continuous integration and deployment (CI/CD) workflows.

By using AWS Proton, developers can reduce the time and effort required to deploy and manage microservices and container-based applications. This approach enables teams to focus on developing and improving their applications, rather than spending time on the deployment and management process.

AWS App2Container

AWS App2Container is a command line tool for migrating and modernizing Java and .NET web applications into container format. App2Container analyzes and builds an inventory of applications running in bare metal, virtual machines, Amazon EC2 instances, or in the cloud. You simply select the application you want to containerize, and App2Container packages the application artifact and identified dependencies into container images, configures the network ports, and generates the ECS task and Kubernetes pod definitions. App2Container identifies the supported ASP.NET and Java applications running in a virtual machine to build a comprehensive inventory all applications in your environment. App2Container can containerize ASP.NET web applications running in IIS on Windows or Java Applications running on Linux, standalone or on application servers such as JBoss, Apache Tomcat, Springboot, IBM Websphere, and Oracle Weblogic.

AWS Copilot

AWS Copilot is a command line interface (CLI) that you can use to quickly launch and manage containerized applications on AWS. It simplifies running applications on Amazon ECS, Fargate, and App Runner. AWS Copilot currently supports Linux, macOS, and Windows systems. Copilot enables you to use service patterns like a load balanced web service to provision infrastructure, deploy to multiple environments like testing or production, and even use an AWS CodePipeline release pipeline for automated deployments.

AWS Serverless Application Model

The AWS Serverless Application Model (AWS SAM) is an open source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML. During deployment, SAM transforms and expands the SAM syntax into AWS CloudFormation syntax, enabling you to build serverless applications faster.

The AWS SAM CLI is an open source command-line tool that makes it easy to develop, test, and deploy serverless applications on AWS. It is a command-line interface for building serverless applications using the AWS SAM specification, which is an extension of AWS CloudFormation.

The AWS SAM CLI enables developers to define and test their serverless applications locally before deploying them to AWS. It provides a local testing environment that simulates AWS Lambda and API Gateway, enabling developers to test their code and configurations before deploying them to the cloud.

The AWS SAM CLI also includes a variety of helpful features, such as automatic code deployment, logging, and debugging capabilities. It enables developers to build, package, and deploy their applications with a single command, reducing the time and effort required to deploy and manage serverless applications.

Additionally, the AWS SAM CLI provides support for various programming languages, including Node.js, Python, Java, and .NET Core, among others. This allows developers to use their preferred programming language and tools to build and deploy their serverless applications.

AWS SAM CLI integrates with other AWS services, such as AWS CodePipeline and AWS CodeBuild, to provide a fully automated and integrated CI/CD pipeline for serverless applications. It also enables developers to use other AWS services, such as Amazon S3, Amazon DynamoDB, and Amazon SNS, as part of their serverless applications.

AWS Cloud Development Kit (AWS CDK)

The AWS Cloud Development Kit (AWS CDK) (AWS CDK) is an open source software development framework for defining cloud infrastructure as code with modern programming languages and deploying it through AWS CloudFormation. AWS Cloud Development Kit (AWS CDK) accelerates cloud development using common programming languages to model your applications. The AWS CDK lets you build reliable, scalable, cost-effective applications in the cloud with the considerable expressive power of a programming language.

Think of the AWS CDK as a developer-centric toolkit leveraging the full power of modern programming languages to define your AWS infrastructure as code. When AWS CDK applications are run, they compile down to fully formed CloudFormation JSON/YAML templates that are then submitted to the CloudFormation service for provisioning. Because the AWS CDK leverages CloudFormation, you still enjoy all the benefits CloudFormation provides such as safe deployment, automatic rollback, and drift detection.

This approach yields many benefits, including:

- Build with high-level constructs that automatically provide sensible, secure defaults for your AWS resources, defining more infrastructure with less code.
- Use programming idioms like parameters, conditionals, loops, composition, and inheritance to model your system design from building blocks provided by AWS and others.
- Put your infrastructure, application code, and configuration all in one place, ensuring that at every milestone you have a complete, cloud-deployable system.
- Employ software engineering practices such as code reviews, unit tests, and source control to make your infrastructure more robust.
- AWS Solutions Constructs is an open-source library extension of AWS CDK. AWS Solutions Constructs provides you with a collection of vetted, multi-service architecture patterns built using the best practices established by the AWS Well-Architected Framework.

AWS Serverless Application Model and AWS CDK both abstract AWS infrastructure as code making it easier for you to define your cloud infrastructure. AWS SAM is specifically focused on serverless use cases and architectures and allows you to define your infrastructure in compact, declarative JSON/YAML templates. AWS CDK offers broad coverage across all of AWS services and allows you to define cloud infrastructure in modern programming languages

Amazon EC2 Image Builder

EC2 Image Builder simplifies the building, testing, and deployment of VM and container images for use on AWS or on-premises. Keeping VM and container images up-to-date can be time consuming, resource intensive, and error-prone. Currently, customers either manually update and snapshot VMs or have teams that build automation scripts to maintain images. Image Builder significantly reduces the effort of keeping images up-to-date and secure by providing a simple graphical interface, built-in automation, and AWS-provided security settings. With Image Builder, there are no manual steps for updating an image nor do you have to build your own automation pipeline.

Image Builder is offered at no cost, other than the cost of the underlying AWS resources used to create, store, and share the images.

EC2 Image Builder can help make deployments easier on AWS by simplifying the process of creating and managing custom images for use with Amazon EC2, containers, and on-premises servers. The service provides a simplified and flexible way to create and manage custom images, with automated build pipelines that enable you to streamline the image creation and management process.

EC2 Image Builder provides a user-friendly interface that abstracts away much of the underlying infrastructure management, making it easier for developers to create and manage custom images. With EC2 Image Builder, developers can specify the operating system, applications, and packages they want to include in the image, and the service automates the process of building and testing the image, including updates, patches, and security fixes. Automated build pipelines enable developers to streamline the image creation and management process, reducing the time and effort required for manual image creation and testing. This can help to improve consistency, reduce errors, and ensure that images are up-to-date, secure, and compliant.

The following are some of the benefits of EC2 Image Builder:

- Simplified image creation: EC2 Image Builder provides a simplified and flexible way to create custom images for use with Amazon EC2, containers, and on-premises servers. This can help to reduce the time and effort required to create and maintain custom images, and enable you to focus on other aspects of deployment, such as application development and testing.
- Automated image build pipelines: EC2 Image Builder provides automated pipelines for building, testing, and deploying custom images, which can help to streamline the image creation and management process. This can help to ensure that your images are up-to-date, secure, and compliant, and reduce the time and effort required for manual image creation and testing.
- Integration with AWS services: EC2 Image Builder integrates with other AWS services, such as Amazon Elastic Container Registry (ECR) and Amazon Elastic Kubernetes Service (EKS), to enable you to build custom images for use with containers. This can help to streamline the container build and deployment process, enabling you to build custom images that include your applications, libraries, and configurations.
- Flexible image creation: EC2 Image Builder provides a flexible way to create custom images, enabling you to specify the operating system, applications, and packages you want to include in the image. This can help to ensure that your images are tailored to your specific use case and requirements, and reduce the risk of errors or incompatibilities during deployment.

- Improved image security and compliance: EC2 Image Builder enables you to automate image testing, including vulnerability and compliance scans, to ensure that your images are secure and compliant. This can help to reduce the risk of security breaches and improve compliance, and enable you to deploy your applications with confidence.

Deployment strategies

In addition to selecting the right tools to update your application code and supporting infrastructure, implementing the right deployment processes is a critical part of a complete, well-functioning deployment solution. The deployment processes that you choose to update your application can depend on your desired balance of control, speed, cost, risk tolerance, and other factors.

Each AWS deployment service supports a number of deployment strategies. This section will provide an overview of general-purpose deployment strategies that can be used with your deployment solution.

Prebaking vs. bootstrapping AMIs

If your application relies heavily on customizing or deploying applications onto Amazon EC2 instances, then you can optimize your deployments through *bootstrapping* and *prebaking* practices.

Installing your application, dependencies, or customizations whenever an Amazon EC2 instance is launched is called *bootstrapping* an instance. If you have a complex application or large downloads required, this can slow down deployments and scaling events.

An Amazon Machine Image (AMI) provides the information required to launch an instance (operating systems, storage volumes, permissions, software packages, etc.). You can launch multiple, identical instances from a single AMI. Whenever an EC2 instance is launched, you select the AMI that is to be used as a template. *Prebaking* is the process of embedding a significant portion of your application artifacts within an AMI.

Prebaking application components into an AMI can speed up the time to launch and operationalize an Amazon EC2 instance. Prebaking and bootstrapping practices can be combined during the deployment process to quickly create new instances that are customized to the current environment.

Blue/green deployments

A blue/green deployment is a deployment strategy in which you create two separate, but identical environments. One environment (blue) is running the current application version and one environment (green) is running the new application version. Using a blue/green deployment strategy increases application availability and reduces deployment risk by simplifying the rollback

process if a deployment fails. Once testing has been completed on the green environment, live application traffic is directed to the green environment and the blue environment is deprecated.

A number of AWS deployment services support blue/green deployment strategies including Elastic Beanstalk, OpsWorks, CloudFormation, CodeDeploy, and Amazon ECS. Refer to Blue/Green Deployments on AWS for more details and strategies for implementing blue/green deployment processes for your application.

Rolling deployments

A rolling deployment is a deployment strategy that slowly replaces previous versions of an application with new versions of an application by completely replacing the infrastructure on which the application is running. For example, in a rolling deployment in Amazon ECS, containers running previous versions of the application will be replaced one-by-one with containers running new versions of the application.

A rolling deployment is generally faster than a blue/green deployment; however, unlike a blue/green deployment, in a rolling deployment there is no environment isolation between the old and new application versions. This allows rolling deployments to complete more quickly, but also increases risks and complicates the process of rollback if a deployment fails.

Rolling deployment strategies can be used with most deployment solutions. Refer to CloudFormation Update Policies for more information on rolling deployments with CloudFormation; Rolling Updates with Amazon ECS for more details on rolling deployments with Amazon ECS; Elastic Beanstalk Rolling Environment Configuration Updates for more details on rolling deployments with Elastic Beanstalk; and Using a Rolling Deployment in AWS OpsWorks for more details on rolling deployments with OpsWorks.

Canary deployments

Canary deployments are a type of blue/green deployment strategy that is more risk-averse. This strategy involves a phased approach in which traffic is shifted to a new version of the application in two increments. The first increment is a small percentage of the traffic, which is referred to as the canary group. This group is used to test the new version, and if it is successful, the traffic is shifted to the new version in the second increment.

Canary deployments can be implemented in two steps or linearly. In the two-step approach, the new application code is deployed and exposed for trial. Upon acceptance, it is rolled out either

to the rest of the environment or in a linear fashion. The linear approach involves incrementally increasing traffic to the new version of the application until all traffic flows to the new release.

In-place deployments

An in-place deployment is a deployment strategy that updates the application version without replacing any infrastructure components. In an in-place deployment, the previous version of the application on each compute resource is stopped, the latest application is installed, and the new version of the application is started and validated. This allows application deployments to proceed with minimal disturbance to underlying infrastructure.

An in-place deployment allows you to deploy your application without creating new infrastructure; however, the availability of your application can be affected during these deployments. This approach also minimizes infrastructure costs and management overhead associated with creating new resources.

Refer to Overview of an in-place deployment for more details on using in-place deployment strategies with CodeDeploy.

Combining Deployment Services

There is not a “one size fits all” deployment solution on AWS. In the context of designing a deployment solution, it is important to consider the type of application as this can dictate which AWS services are most appropriate. To deliver complete functionality to provision, configure, deploy, scale, and monitor your application, it is often necessary to combine multiple deployment services

A common pattern for applications on AWS is to use CloudFormation (and its extensions) to manage general-purpose infrastructure, and use a more specialized deployment solution for managing application updates. In the case of a containerized application, CloudFormation could be used to create the application infrastructure, and Amazon ECS and Amazon EKS could be used to provision, deploy, and monitor containers.

AWS deployment services can also be combined with third-party deployment services. This allows organizations to easily integrate AWS deployment services into their existing CI/CD pipelines or infrastructure management solutions. For example, OpsWorks can be used to synchronize configurations between on-premises and AWS nodes, and CodeDeploy can be used with a number of third-party CI/CD services as part of a complete pipeline.

Conclusion

AWS provides number of tools to simplify and automate the provisioning of infrastructure and deployment of applications; each deployment service offers different capabilities for managing applications. To build a successful deployment architecture, evaluate the available features of each service against the needs your application and your organization.

Contributors

Contributors to this document include:

- Manikandan Chandrasekaran, Principal Technologist
- Anil Nadiminti, Senior Solutions Architect
- Bryant Bost, AWS ProServe Consultant

Further Reading

For additional information, see:

- AWS Whitepapers page
- Introduction to DevOps on AWS - Deployment Strategies

Document Revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Updated throughout for latest deployment services and strategies	May 31, 2024
Minor update	*Blue/Green Deployments* section revised for clarity.	April 8, 2021
Whitepaper updated	Updated with latest services and features.	June 3, 2020
Initial publication	Whitepaper first published	March 1, 2015

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.