Overview of Deployment Options on AWS

AWS Whitepaper

Overview of Deployment Options on AWS: AWS Whitepaper

Table of Contents

Overview of Deployment Options on AWS

Publication date: **June 3, 2020** (*Document Revisions* (p. 20))

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 those 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 (compute, storage, database, etc.) 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 (Amazon EC2, Amazon Virtual Private Cloud [Amazon VPC], subnets, etc.) or managed service infrastructure (Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service [Amazon RDS], Amazon CloudFront, etc.) 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 component(s) 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 resources usage, deployment success/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 the issue of 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 provides a number of services that provide management capabilities for one or more aspects of your application lifecycle. Depending on your desired balance of control (i.e., manual management of resources) versus convenience (i.e., 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. A 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.

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. CloudFormation is commonly used with other AWS deployment services or third-party tools; combining 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) (AWS CDK) is an open source software development kit (SDK) to programmatically model AWS infrastructure with TypeScript, Python, Java, or .NET.
- AWS Serverless Application Model (SAM) is an open source framework to simplify building serverless applications on AWS.

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 CloudFormation templates.
Configure	CloudFormation templates offer extensive flexibility to customize and update all infrastructure components.

Capability	Description
	Refer to CloudFormation Template Anatomy for more details on customizing templates.
Deploy	Update your CloudFormation templates to alter the resources in a stack. Depending on your application architecture, you may need to use an additional deployment service to update the application version running on your infrastructure. Refer to Deploying Applications on EC2 with AWS CloudFormation for more details on how CloudFormation can be used as a deployment solution.
Scale	CloudFormation will not automatically handle infrastructure scaling on your behalf; however, you can configure auto scaling policies for your resources in a CloudFormation template.
Monitor	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 specifications. 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 CloudFormation monitors infrastructure updates.

The following diagram shows a common use case for CloudFormation. Here, 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 CloudFormation to deploy the latest version of our application onto EC2 instances; however, it is also a common practice to combine additional deployment services with CloudFormation (using CloudFormation only for its infrastructure management and provisioning capabilities). Note that more than one CloudFormation template is used to create the infrastructure.

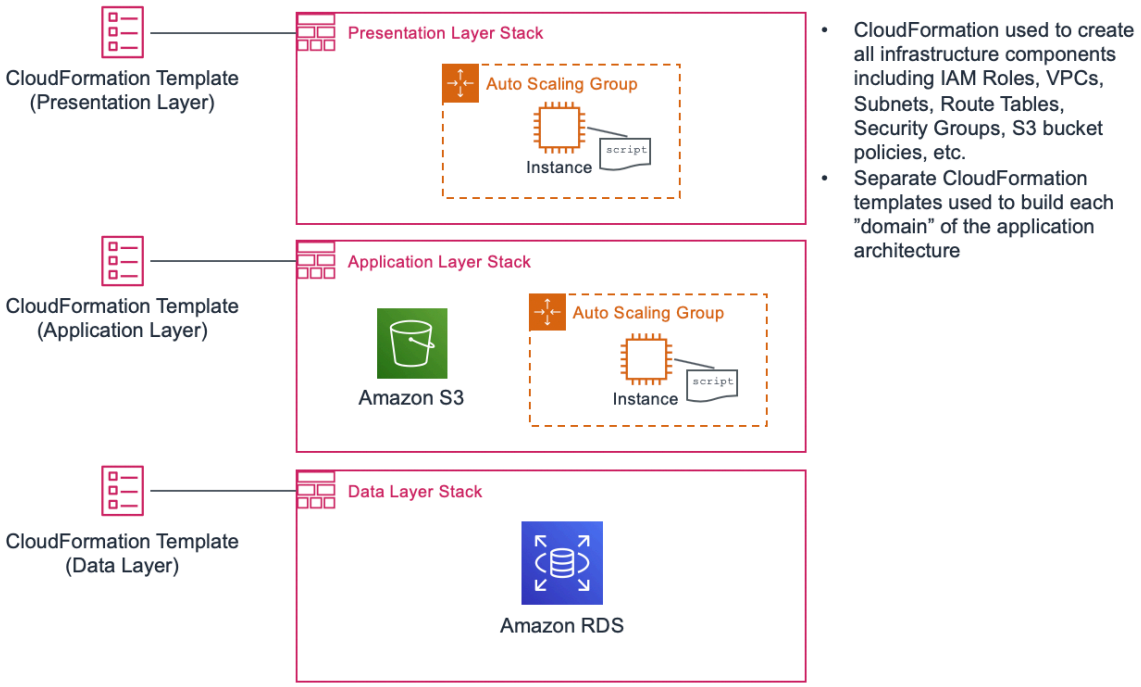

Figure 1: AWS CloudFormation use case

AWS Elastic Beanstalk

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, 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.

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 additional infrastructure, this will have to be created outside of Elastic Beanstalk.

Capability	Description
	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 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 will automatically handle scaling of your infrastructure with managed auto scaling groups for your application instances. 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 environment 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.

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.

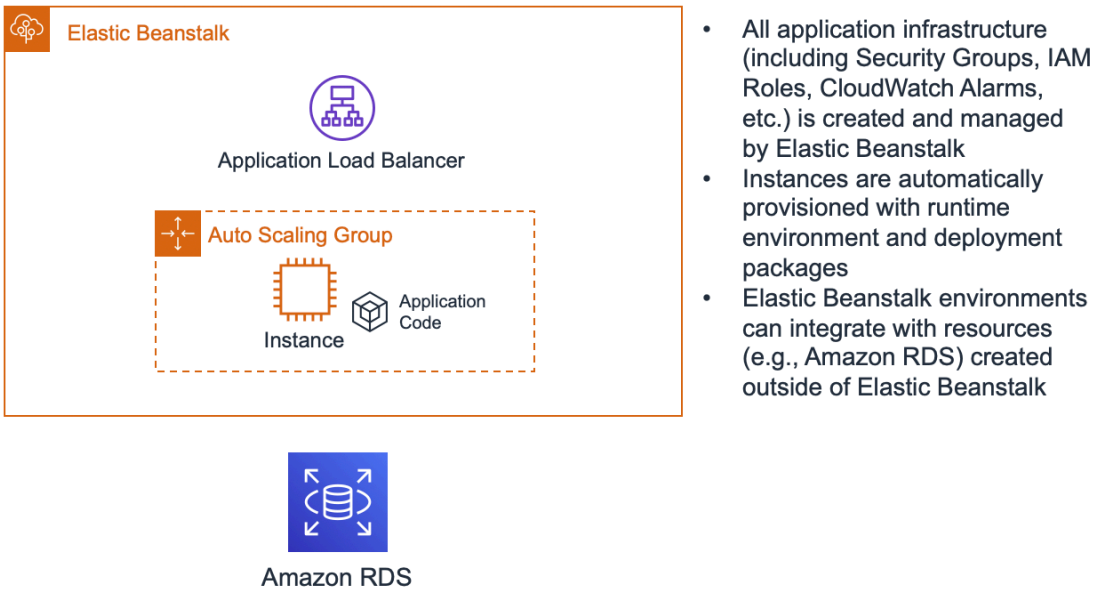

Figure 2: AWS Elastic Beanstalk use case

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.

Capability	Description
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, depending on your deployment configurations, it may create additional resources to support blue/green deployments
Monitor	CodeDeploy offers monitoring of the success or failure of deployments, as well as 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 EC2 instances.

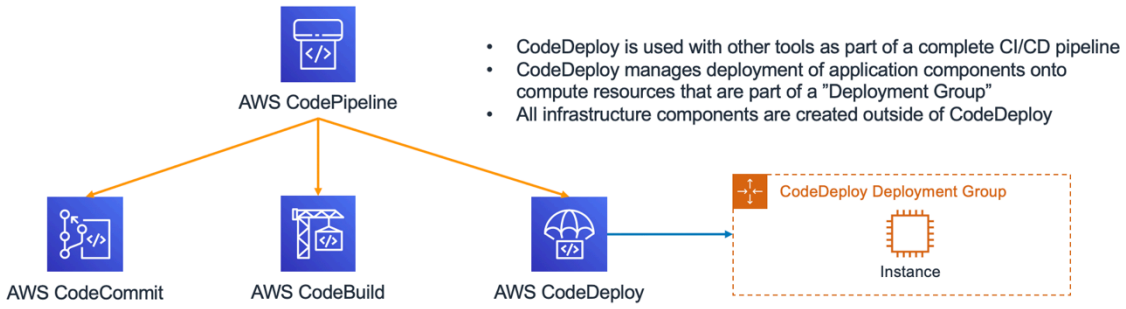

Figure 3: AWS CodeDeploy use case

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 containerized application, as well as the runtime conditions of the application containers (e.g., environment variables, exposed ports, reserved memory/CPU). Customization 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.
Scale	Amazon ECS can be used with auto-scaling policies to automatically adjust the number of containers running in your Amazon ECS cluster.

Capability	Description
	Refer to Service Auto Scaling for more details on configuring auto scaling for your containerized 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

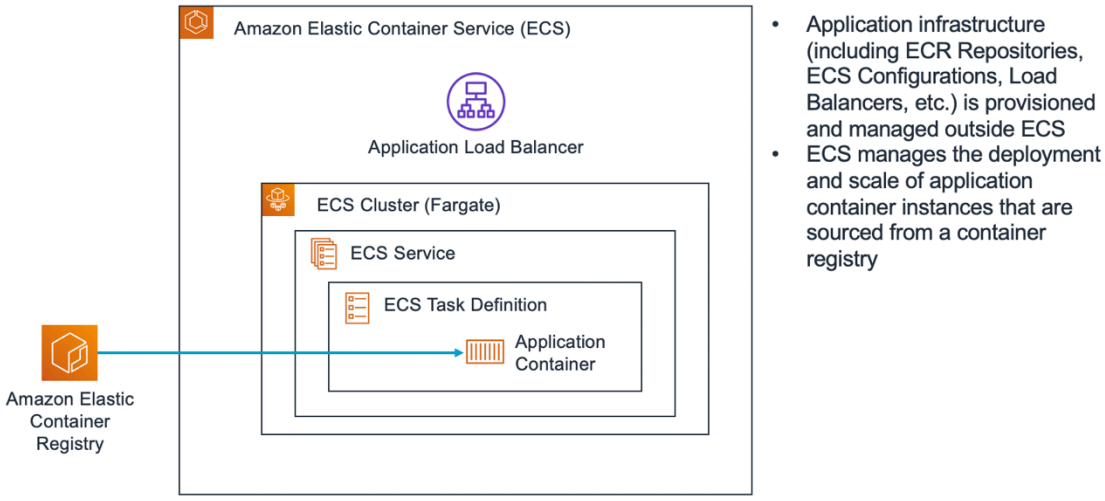

Figure 4: Amazon ECS use case

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 also integrates with AWS App Mesh and provides a Kubernetes-native experience to consume service mesh features and bring rich observability, traffic controls and security features to applications. Amazon EKS provides a scalable, highly-available control plane for Kubernetes workloads. When running applications on Amazon EKS, as with Amazon ECS, you can choose to provide the underlying compute power for your containers with EC2 instances or with AWS Fargate.

Table 5: Amazon EKS deployment features

Capability	Description
Provision	Amazon EKS provisions certain resources to support containerized applications: • Load Balancers, if needed. • Compute Resources ("workers"). Amazon EKS supports Windows and Linux. • Application Container Instances ("pods"). Refer to Getting Started with Amazon EKS for more details on Amazon EKS cluster provisioning.
Configure	Amazon EKS supports customization of the compute resources ("workers") if using EC2 instances to supply compute power. EKS also supports customization of the runtime conditions of the application containers ("pods"). 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.
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.

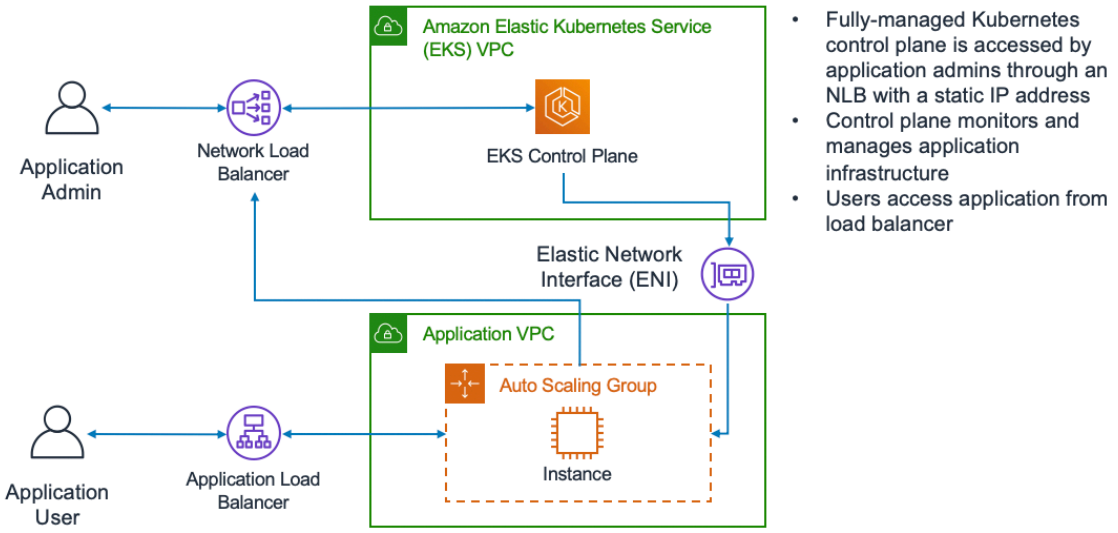

Figure 5: Amazon EKS use case

AWS OpsWorks

AWS OpsWorks is a configuration management service that enables customers to construct, manage, and operate a wide variety of application architectures, from simple web applications to highly complex custom applications. Organizations deploying applications with OpsWorks use the automation platforms Chef or Puppet to manage key operational activities like server provisioning, software configurations, package installations, database setups, scaling, and code deployments. There are three ways to use OpsWorks:

- **AWS OpsWorks for Chef Automate**: fully managed configuration management service that hosts Chef Automate.
- **AWS OpsWorks for Puppet Enterprise**: fully managed configuration management service that hosts Puppet Enterprise.
- **AWS OpsWorks Stacks**: application and server management service that supports modeling applications using the abstractions of "stacks" and "layers" that depend on Chef recipes for configuration management.

With OpsWorks for Chef Automate and OpsWorks for Puppet Enterprise, AWS creates a fully managed instance of Chef or Puppet running on Amazon EC2. This instance manages configuration, deployment, and monitoring of nodes in your environment that are registered to the instance. When using OpsWorks with Chef Automate or Puppet Enterprise, additional services (e.g., CloudFormation) may need to be used to create and manage infrastructure components that are not supported by OpsWorks.

OpsWorks Stacks provides a simple and flexible way to create and manage application infrastructure. When working with OpsWorks Stacks, you model your application as a "stack" containing different "layers." A layer contains infrastructure components necessary to support a particular application function, such as load balancers, databases, or application servers. OpsWorks Stacks does not require the creation of a Chef server, but uses Chef recipes for each layer to handle tasks such as installing packages on instances, deploying applications, and managing other resource configurations. OpsWorks Stacks will create and provision infrastructure on your behalf, but does not support all AWS services.

Provided that a node is network reachable from an OpsWorks Puppet or Chef instance, any node can be registered with the OpsWorks, making this solution a good choice for organizations already using Chef or Puppet and working in a hybrid environment. With OpsWorks Stacks, an on-premises node must be able to communicate with public AWS endpoints.

Table 6: AWS OpsWorks deployment features

Capability	Description
Provision	OpsWorks Stacks can create and manage certain AWS services as part of your application using Chef recipes. With OpsWorks for Chef Automate or Puppet Enterprise, infrastructure must be created elsewhere and registered to the Chef or Puppet instance. Refer to Create a New Stack for more details on creating resources with OpsWorks Stacks.
Configure	All OpsWorks operating models support configuration management of registered nodes. OpsWorks Stacks supports customization of other infrastructure in your environment through layer customization. Refer to OpsWorks Layer Basics for more details on customizing resources with OpsWorks Layers.
Deploy	All OpsWorks operating models support deployment and update of applications running on registered nodes. Refer to Deploying Apps for more details on how to deploy applications with OpsWorks Stacks.
Scale	OpsWorks Stacks can handle automatically scaling instances in your environment based on changes in incoming traffic. Refer to Using Automatic Load-based Scaling for more details on auto scaling with OpsWorks Stacks.
Monitor	OpsWorks provides several features to monitor your application infrastructure and deployment success. In addition to Chef/Puppet logs, OpsWorks provides a set of configurable Amazon CloudWatch and AWS CloudTrail metrics for full-stack monitoring Refer to Monitoring Stacks using Amazon CloudWatch for more details on resource monitoring in OpsWorks.

OpsWorks provides a complete, flexible, and automated solution that works with existing and popular tools while allowing application owners to maintain full-stack control of an application. The following example shows a typical use case for AWS OpsWorks Stacks as it is used to create and manage a three-tier web application.

Figure 6: AWS OpsWorks Stacks use case

This next example shows a typical use case for AWS OpsWorks for Chef Automate or Puppet Enterprise as it is used to manage the compute instances of a web application.

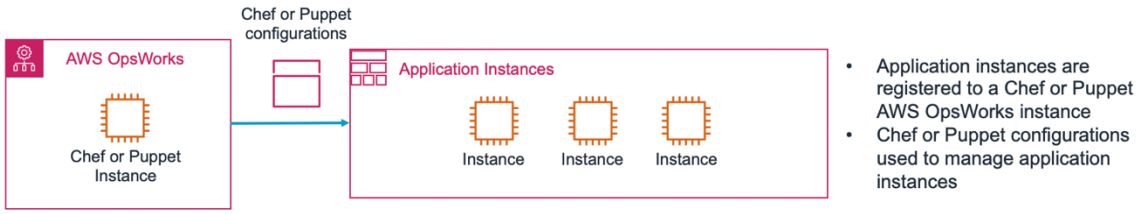

Figure 7: AWS OpsWorks with Chef Automate or Puppet Enterprise use case

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.

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.

Refer to Best practices for building AMIs for more details on creating optimized AMIs for your application.

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.

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:

- Bryant Bost, AWS ProServe Consultant

Further Reading

For additional information, see:

- AWS Whitepapers page

Document Revisions

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

Change	Description	Date
Minor update (p. 20)	*Blue/Green Deployments* section revised for clarity.	April 8, 2021
Whitepaper updated (p. 20)	Updated with latest services and features.	June 3, 2020
Initial publication (p. 20)	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.