

Deploying and Managing Infrastructure at Scale Section

What is CloudFormation



- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
 - I want a security group
 - I want two EC2 instances using this security group
 - I want an S3 bucket
 - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

Benefits of AWS CloudFormation (1/2)

- **Infrastructure as code**

- No resources are manually created, which is excellent for control
- Changes to the infrastructure are reviewed through code

- **Cost**

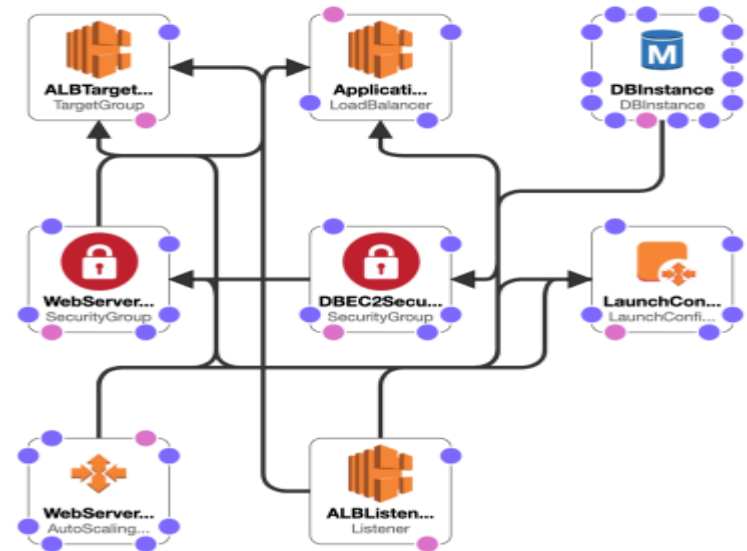
- Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
- You can estimate the costs of your resources using the CloudFormation template
- Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

Benefits of AWS CloudFormation (2/2)

- Productivity
 - Ability to destroy and re-create an infrastructure on the cloud on the fly
 - Automated generation of Diagram for your templates!
 - Declarative programming (no need to figure out ordering and orchestration)
- Don't re-invent the wheel
 - Leverage existing templates on the web!
 - Leverage the documentation
- **Supports (almost) all AWS resources:**
 - Everything we'll see in this course is supported
 - You can use "custom resources" for resources that are not supported

CloudFormation Stack Designer

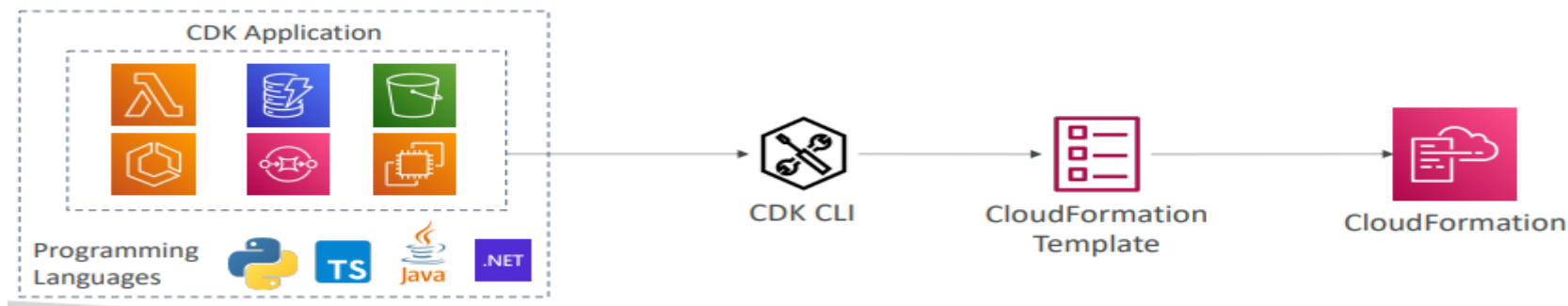
- Example: WordPress CloudFormation Stack
- We can see all the **resources**
- We can see the **relations** between the components



AWS Cloud Development Kit (CDK)



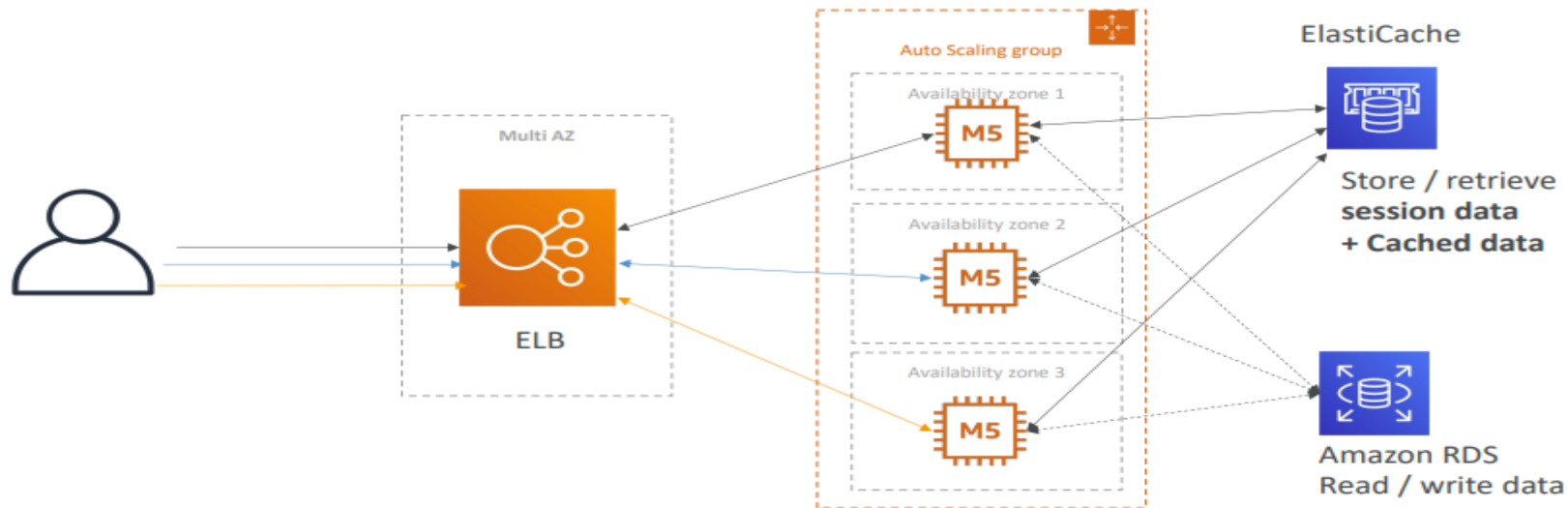
- Define your cloud infrastructure using a familiar language:
 - JavaScript/TypeScript, Python, Java, and .NET
- The code is "compiled" into a CloudFormation template (JSON/YAML)
- **You can therefore deploy infrastructure and application runtime code together**
 - Great for Lambda functions
 - Great for Docker containers in ECS / EKS



CDK Example

```
export class MyEcsConstructStack extends core.Stack {  
  constructor(scope: core.App, id: string, props?: core.StackProps)  
    super(scope, id, props);  
  
  const vpc = new ec2.Vpc(this, "MyVpc", {  
    maxAzs: 3 // Default is all AZs in region  
  });  
  
  const cluster = new ecs.Cluster(this, "MyCluster", {  
    vpc: vpc  
  });  
  
  // Create a load-balanced Fargate service and make it public  
  new ecs_patterns.ApplicationLoadBalancedFargateService(this, "My  
    cluster: cluster, // Required  
    cpu: 512, // Default is 256  
    desiredCount: 6, // Default is 1  
    taskImageOptions: { image: ecs.ContainerImage.fromRegistry("an  
    memoryLimitMiB: 2048, // Default is 512  
    publicLoadBalancer: true // Default is false  
  });  
}
```

Typical architecture: Web App 3-tier



Developer problems on AWS

- Managing infrastructure
 - Deploying Code
 - Configuring all the databases, load balancers, etc
 - Scaling concerns
-
- Most web apps have the same architecture (ALB + ASG)
 - All the developers want is for their code to run!
 - Possibly, consistently across different applications and environments

AWS Elastic Beanstalk Overview



- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG, ELB, RDS, etc...
- But it's all in one view that's easy to make sense of!
- We still have full control over the configuration

- **Beanstalk = Platform as a Service (PaaS)**
- Beanstalk is free but you pay for the underlying instances

Elastic Beanstalk

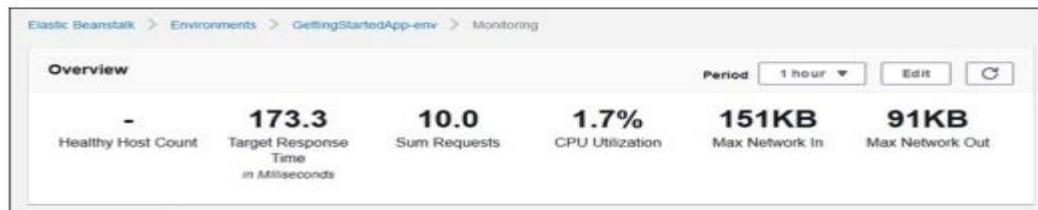
- Managed service
 - Instance configuration / OS is handled by Beanstalk
 - Deployment strategy is configurable but performed by Elastic Beanstalk
 - Capacity provisioning
 - Load balancing & auto-scaling
 - Application health-monitoring & responsiveness
- **Just the application code is the responsibility of the developer**
- Three architecture models:
 - Single Instance deployment: good for dev
 - LB + ASG: great for production or pre-production web applications
 - ASG only: great for non-web apps in production (workers, etc..)

Elastic Beanstalk

- Support for many platforms:
 - Go
 - Java SE
 - Java with Tomcat
 - .NET on Windows Server with IIS
 - Node.js
 - PHP
 - Python
 - Ruby
 - Packer Builder
- Single Container Docker
- Multi-Container Docker
- Preconfigured Docker
- If not supported, you can write your custom platform (advanced)

Elastic Beanstalk – Health Monitoring

- Health agent pushes metrics to CloudWatch
- Checks for app health, publishes health events



Recent events

Show all

Time	Type	Details
2020-11-20 16:35:04 UTC-0800	INFO	Environment health has transitioned from Severe to OK.
2020-11-20 16:35:04 UTC-0800	INFO	Added instance (i-03290130ba7b41717) to your environment.
2020-11-20 16:35:04 UTC-0800	WARN	Removed instance (i-0a4d7b0d5044a4c) from your environment due to a EC2 health check failure.
2020-11-20 16:35:04 UTC-0800	WARN	Environment health has transitioned from OK to Severe. ELB probes are not healthy on all instances. None of the instances are sending data. ELB health is failing or not available for all instances.
2020-11-20 15:19:06 UTC-0800	INFO	Environment health has transitioned from Info to Ok. Application updates completed 75 seconds ago and took 22 seconds.

Environment health overview																	
Overview: 1 Page 1/1																	
Last update: 2023-09-20 10:00:00																	
Environment ID	Status	Health	Instances	Instances in Service	Instances in Stopped	Instances in Pending	Instances in Terminating	Instances in Unknown	Instances in Unavailable	Instances in Out of Service	Instances in Incompatible	Instances in Invalid	Instances in Other	Instances in Error	Instances in Failed	Instances in Deleted	Instances in Archived
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-01234567	OK	Healthy	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
env-0123456																	



AWS CodeDeploy



- We want to deploy our application automatically
- Works with EC2 Instances
- Works with On-Premises Servers
- Hybrid service
- Servers / Instances must be provisioned and configured ahead of time with the CodeDeploy Agent

EC2 Instances being upgraded



On-premises Servers being upgraded



AWS CodeBuild



- Code building service in the cloud (name is obvious)
- **Compiles source code, run tests, and produces packages that are ready to be deployed (by CodeDeploy for example)**

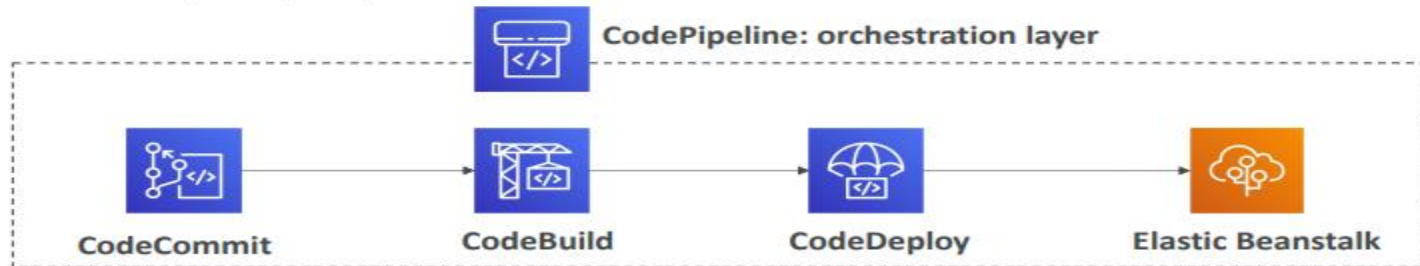


- Benefits:
 - Fully managed, serverless
 - Continuously scalable & highly available
 - Secure
 - Pay-as-you-go pricing – only pay for the build time

AWS CodePipeline



- **Orchestrate the different steps to have the code automatically pushed to production**
 - Code => Build => Test => Provision => Deploy
 - Basis for CI/CD (Continuous Integration & Continuous Delivery)
- **Benefits:**
 - Fully managed, compatible with CodeCommit, CodeBuild, CodeDeploy, Elastic Beanstalk, CloudFormation, GitHub, 3rd-party services (GitHub...) & custom plugins...
 - Fast delivery & rapid updates



AWS CodeArtifact



- Software packages depend on each other to be built (also called code dependencies), and new ones are created
- Storing and retrieving these dependencies is called **artifact management**
- Traditionally you need to setup your own artifact management system
- **CodeArtifact** is a secure, scalable, and cost-effective **artifact management** for software development
- Works with common dependency management tools such as Maven, Gradle, npm, yarn, twine, pip, and NuGet
- **Developers and CodeBuild** can then retrieve dependencies straight from **CodeArtifact**

AWS CodeStar



- **Unified UI** to easily manage software development activities **in one place**
- "Quick way" to get started to correctly set-up CodeCommit, CodePipeline, CodeBuild, CodeDeploy, Elastic Beanstalk, EC2, etc...
- Can edit the code "in-the-cloud" using **AWS Cloud9**



AWS Cloud9

- AWS Cloud9 is a cloud IDE (Integrated Development Environment) for writing, running and debugging code
- "Classic" IDE (like IntelliJ, Visual Studio Code...) are downloaded on a computer before being used
- A cloud IDE can be used within a web browser, meaning you can work on your projects from your office, home, or anywhere with internet with no setup necessary
- AWS Cloud9 also allows for code collaboration in real-time (pair programming)



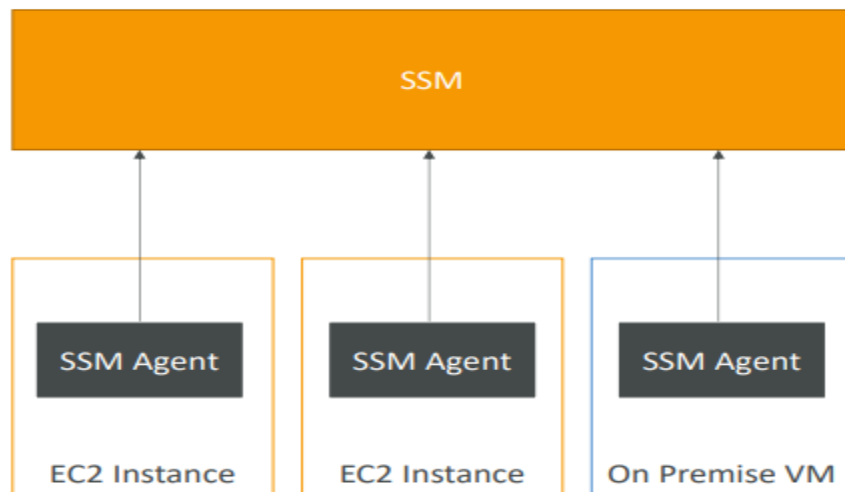
AWS Systems Manager (SSM)



- Helps you manage your **EC2 and On-Premises** systems at scale
- Another **Hybrid** AWS service
- Get operational insights about the state of your infrastructure
- Suite of 10+ products
- Most important features are:
 - **Patching automation** for enhanced compliance
 - **Run commands across an entire fleet of servers**
 - Store parameter configuration with the SSM Parameter Store
- Works for Linux, Windows, MacOS, and Raspberry Pi OS (Raspbian)

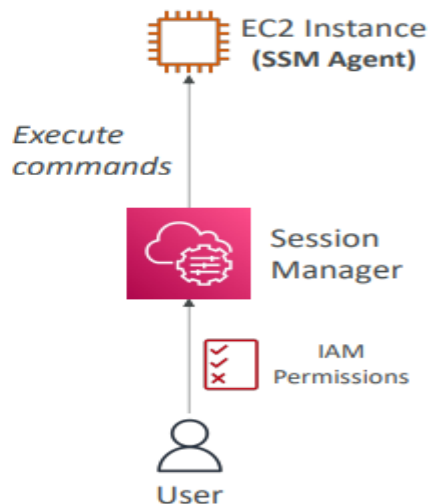
How Systems Manager works

- We need to install the SSM agent onto the systems we control
- Installed by default on Amazon Linux AMI & some Ubuntu AMI
- If an instance can't be controlled with SSM, it's probably an issue with the SSM agent!
- Thanks to the SSM agent, we can **run commands, patch & configure** our servers



Systems Manager – SSM Session Manager

- Allows you to start a secure shell on your EC2 and on-premises servers
- No SSH access, bastion hosts, or SSH keys needed
- No port 22 needed (better security)
- Supports Linux, macOS, and Windows
- Send session log data to S3 or CloudWatch Logs



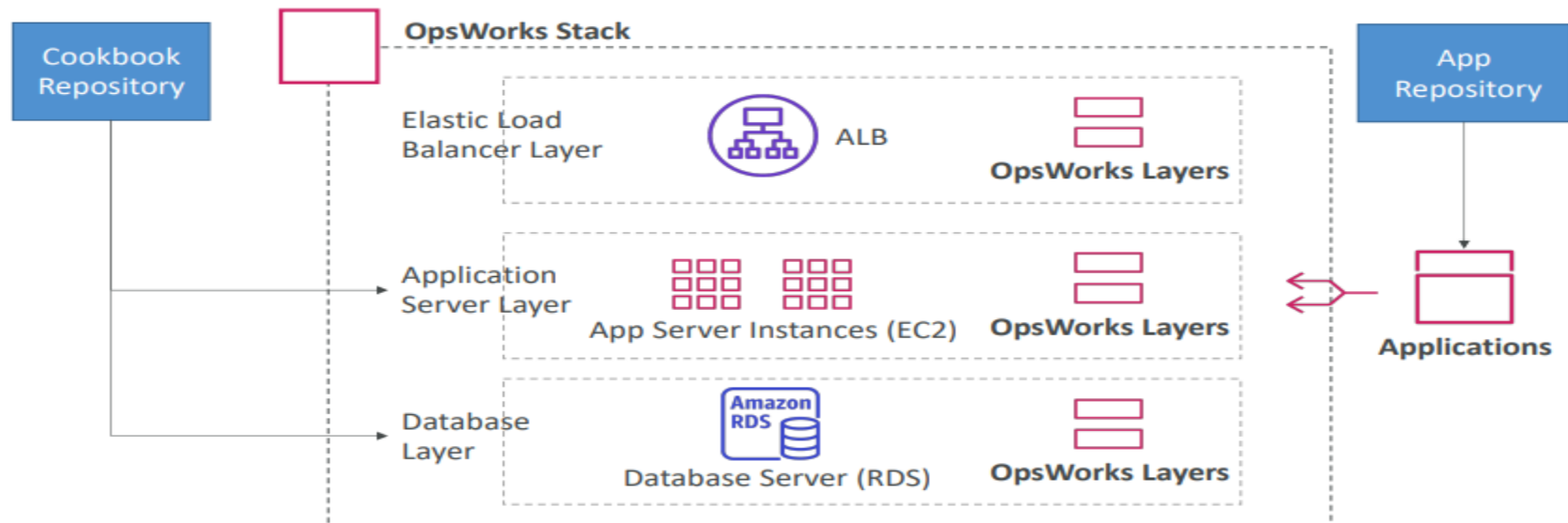
AWS OpsWorks



- Chef & Puppet help you perform server configuration automatically, or repetitive actions
- They work great with EC2 & On-Premises VM
- AWS OpsWorks = Managed Chef & Puppet
- It's an alternative to AWS SSM
- Only provision **standard AWS resources**:
 - EC2 Instances, Databases, Load Balancers, EBS volumes...
- In the exam: Chef or Puppet needed => AWS OpsWorks



OpsWorks Architecture



Deployment - Summary

- **CloudFormation:** (AWS only)
 - Infrastructure as Code, works with almost all of AWS resources
 - Repeat across Regions & Accounts
- **Beanstalk:** (AWS only)
 - Platform as a Service (PaaS), limited to certain programming languages or Docker
 - Deploy code consistently with a known architecture: ex, ALB + EC2 + RDS
- **CodeDeploy** (hybrid): deploy & upgrade any application onto servers
- **Systems Manager** (hybrid): patch, configure and run commands at scale
- **OpsWorks** (hybrid): managed Chef and Puppet in AWS

Developer Services - Summary

- **CodeCommit:** Store code in private git repository (version controlled)
- **CodeBuild:** Build & test code in AWS
- **CodeDeploy:** Deploy code onto servers
- **CodePipeline:** Orchestration of pipeline (from code to build to deploy)
- **CodeArtifact:** Store software packages / dependencies on AWS
- **CodeStar:** Unified view for allowing developers to do CICD and code
- **Cloud9:** Cloud IDE (Integrated Development Environment) with collab
- **AWS CDK:** Define your cloud infrastructure using a programming language