# Stack Policy

Ways to dictate how a stack can be updated.

The absence of a stack policy defaults to allow updated

Once a stack policy is applied it can't be deleted

Once a sp is applied by default ALL objects are Update:\* deny

# ElasticBeanstalk

Creates an .ebextensions folder where you can configure Beanstalk environments

Has deployment strategy of All At Once, Immutable, and rolling. However Immutable creates the least downtime because it creates an entire environment before deployment, where All At Once will replace the current instances

# AWS Config

Take snapshots of the state of resources.

Continous Monitoring, Continous assessment, Troubleshooting, Compliance Monitoring, Change Management

AWS Config you will need an S3 Bucket, you can stream config changes and notifications to sns.

You will need an AWS Config service-linked role

There are various AWS COnfig rules that are preconfigrued. Specifically CFN Tempaltes, can check if an SNS topic is triggered.

S3 Bucket, logging-enabled/replication-enabled/versioning-enabled/ss-encryption-enabled/ssl-requests-only

Across all accounts

configservice, NOT configure. That is more for sso

# AWS Health

Automatically monitors git repositories for any exposed keys, and generates a AWS\_RISK\_Credentials\_EXPOSED CloudWatch Event where aws.health is the Event Source. Using AWS Step Functions you can build a serverless workflow to delete the key, summarize API Activity in CloudTrail, and send a notification via SNS.

# Systems Manager

## Patch Manager

Specifically used for applying patches. You will need the AssumeRole to execute STS. Then any hybrid instances will have a prefix of mi- in the SSM Console.

# CodeDeploy

Monitor with CloudWatch Events, CloudWatch Alarms, and CloudWatch Logs. Can do **in-place** or **Blue/Green** deploys

## Deployment Groups

**ECS** – Is the Service, ALB, optional test listener and two target groups. It will specify when to route traffic to the replacement **Task** from the original **Task** – Functions Validate

**Lambda** – Defines a set of CodeDeploy configurations for future deploys of Lambda – AppSpec contains functions to validate

**EC2/On-Prem** – Set of individual Instances. AppSpec is always in yaml

## Revision

**ECS** – Is a json/yaml file that contains task definitions for the deployment, container name, and port mapping used to route traffic. And optional Lambda Functions to run on Lifecycle events

**Lambda** – AppSpec file that contains info about Lambda Function to deploy. Stored in S3

**EC2/On-Prem** – Archive file that contains binaries, and various files required for new application

EC2/On-Prem are the only setups that can do in-place deployments. Can’t do it with Lambda and ECS

## Rollback

Redeploy by deploying a previous version for EC2 and Lambda. Where ECS will redirect back to working Task

**CodeDeploy Agent** – Required for EC2, not Lambda and ECS

## Health Status

**Revision Health** – Current, Old, Unknown

**Instance Health** – Health or Unhealthy

## Deploy Configurations

### Lambda/ECS

Canary – Shift traffic in two increments \*REMEMBER it is 2 increments, not multiple

Linear – Equal Increments with Equal number of minutes between \*These allow you to set every X minute

All At Once

**You are charged for On-Prem CodeDeployments**

You Blue/Green with ECS and EC2. You In-Place with On-Prem, and Lambda doesn’t need a Deployment Type

**Rolling Deployments not supported by CodeDeploy**

# CodePipeline

## Custom Actions

When using a custom function in a CodePipeline, you must setup a worker that will poll CodePipeline for new jobs/requests, and return results. So if you have a long running job like some UI Automation, or maybe some sort of code scanning tool that runs on-prem, and it need to be part of the CodePipeline to deployment, you can create the worker and have it poll CodePipeline.

CodePipeline can’t poll S3 and it is insecure

## Stages

**Source -> Build -> Deploy -> Production**

# CloudTrail

CloudTrail has an integrity check enablement directly inside the service. Which basically creates a register for every event sent to allow for auditing of the trail

If you want to protect your CloudTrails, you want to trigger off the StopLogging event, not delete trial.

# AWS CloudWatch Events

Can react to changes in the state of a resource, and can link to Lambda, Kinesis, SQS, CloudWatch Alarm Actions, SNS Topics

# CloudWatch Agent

The agent runs on an EC2 instance and can do some real good. You can use StatsD/collectd on Linux or collectd on windows it will gather useful information on the instance

**Pending:wait** refers to scale out

# ASG (Auto Scaling Groups)

There is some pretty difficult things to remember with ASG. One major things is the Lifecycle for an ASG on an EC2 instance is when you get to the Terminating, and Pending states. What you need to keep in mind is the state changes of pending and terminating have life cycles you can hook into. However you need to configure a Systems Manager automation document, and a ASG Lifecycle Hook. These will all tie together fairly neatly either via CLI or CloudFormation. In Systems Manager you create a LifeCycleHookDoc, and in the ASG you put a LifeCycleHook. These should get tied together with a CloudWatch event. However you can utilize the CloudWatch Agent, since part of the ssm automation document is a script

ssm:sendcommand for both the script and the ec2. Arn:autoscalingGroup:\*:AutoScallingGroupName/\*. You would need a principal:{service:ssm.amazon.com},action:sts:AssumeRole, and potentially the same for events.amazon.com for CloudWatch Events

## Scaling Policy

**Target tracking scaling** – Increase/Decrease capacity of the group based on target value of specific metric

**Step Scaling** – Increase/Decrease capacity of the group based on a set of scaling adjustments, known as step adjustments, that vary based on the size of alarm breach

**Simple Scaling** ­– Simple metric to ensure scaling

## AutoScalingReplacingUpdate

During replacement the CloudFormation retains the old ASG until the new ASG finishes creating, if there is a failure with the new ones it rollsback to the old ASG. CFN won’t detattch any of the instances until the ASG is healthy. There is no AWS::AutoScaling::DeploymentUpdates resource.

AutoscalingReplacingUpdate will need WillReplace set to true to take precedence.

# AWS Nested Stacks

Require a root stack, and you would add AWS::CloudFormation::Stack

# ECS

deploy, manage, and scale Docker containers running applications, services, and batch processes integrates with ELB, EC2 Security Groups, EBS Volumes, and IAM Roles

Contianer -> Task -> Service -> Cluster

A container has the AMI

A Task has a group of containers

The service manages the Tasks in a cluster (ALB or not can have CFN Stack)

A cluster managers services (VPC ID and Subnets)

## Agent

There is an ECS agent running for an instance, and if you are periodically not getting the latest instances deployed it is possible you need to restart the agent.

# EC2

## ASG

Has a different lifecycle than typical EC2

Pending->(Wait,Proceed [EC2\_INSTANCE\_LAUNCHING])->In Service->Detaching->Detached

------------------------------------>Teminating->(Wait,Proceed[EC2\_INSTANCE\_TERMINATING])->Terminated

------------------------------------>EnteringStandBy->StandBy->Pending

## Deployment On Prem

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LifeCycle Event | In-Place | B/G Original Instance | B/G Replacement | B/G Rollback Original | B/G Rollback Replacement |
| ApplicationStop | X |  | X |  |  |
| DownloadBundle | X |  | X |  |  |
| BeforeInstall | X |  | X |  |  |
| Install | X |  | X |  |  |
| AfterInstall | X |  | X |  |  |
| ApplicationStart | X |  | X |  |  |
| ValidateService | X |  | X |  |  |
| BeforeBlockTraffic | X | X |  |  | X |
| BlockTraffic | X | X |  |  | X |
| AfterBlockTraffic | X | X |  |  | X |
| BeforeAllowTraffic | X |  | X | X |  |
| AllowTraffic | X |  | X | X |  |
| AfterAllowTraffic | X |  | X | X |  |
|  |  |  |  |  |  |

What this means is when doing in place deployments, everything in the EC2 lifecycle will occur. When doing Blue/Green you need to think about BlockTraffic, and AllowTraffic as the shifting from Original instances to Replacement instances. If you were going to AllowTraffic, it has to be on a new instance, and if you are going to BlockTraffic it should be on an old running instance. Those roles become reversed when rolling back.

There are a few places where only CodeDeploy can run scripts, Install, DownloadBundle, AllowTraffic, and BlockTraffic. CodeDeploy owns those steps so running scripts of your own seems to not be an option. Below is what can/should be done during each lifecyle event

**ApplicationStop** – This happens before the revision is even installed. This is where you can determine the scripts to gracefully stop the application and remove currently installed packages for the deployment. The AppSpec file and scripts used for this deployment are from the last successful deployment. Maybe stop services for deploy

**DownloadBundle –** CodeDeploy agent downloads application revision files to a temporary location /opt/codedeploy-agent/deployment-root/deployment-group-id/deployment-id/deployment-archive or ProgramData\Amazon\CodeDeploy\deployment-group-id\ it is reserved for CodeDeploy

**BeforeInstall** – Preinstall tasks, decrypt files, and create a backup of the current version

**Install** – CodeDeploy moves revision files from the temporary location to the final destination

**AfterInstall** – Configuration Application or changing file permissions

**ApplicationStart –** Restart services that were stopped

**ValidateServices** – Last deployment cycle event. Verify the deployment was successful

--------------------The next step is where traffic flow occurs, and shouldn’t be in the deployment stage--

**BeforeBlockTraffic** – Run scripts on instance before they are deregistered

**BlockTraffic** – CodeDeploy Deregisters the instance, or stops allowing traffic via ALB

**AfterBlockTraffic** – Run scripts on the instance after it is deregistered

**BeforeAllowTraffic** – Run Scripts on the instance before allowing traffic, possibly validate things are working

# AWS Managed Services

Is bascially an entire service platform that will allow a company to pay for an architect, and various pre managed services. Logging, infrastructure, compliance,change protection,

provisioning, network config, patch management, backup/recovery

Large corporations would use this

# Disaster Recovery

The read replica db is actually a really useful diaster recovery setup for a db as it can be promoted to master very quickly.

# DynamoDb w/Kinesis

Amazon Kinesis Adapter is the recommended way to handle DynamoDb streams, not Lambda Functions. That is how you get a stream throttling error

# AWS Lambda

Triggered by S3 Buckets, DynamoDb Table, Kinesis Streams, and SNS Notifications, Also Cloudwatch Logs, Cloudwatch events (Event Bridge), SQS, MQ

Create from scratch, blueprint, container image, and serverless app repository

You can create an execution role, and add some Policy Templates on create

You can attach it to a network, to allow VPC access

--Lifecycle Hooks on ECS (these are similar to EC2/On-Prem minus a few)

--BeforeInstall, Install, AfterInstall, AllowTestTraffic, AfterAllowTestTraffic, BeforeAllowTraffic, AllowTraffic, AfterAllowTraffic

# Step Functions

Allow you to orchestrate Lambda Functions for a purpose. It also gives you a graphical overview of the functions.

# OpsWorks

A fully managed configuration management system for Chef Automate or Puppet Enterprise, and Stacks for Chef Solo

OpsWorks Stacks/Chef is a declarative state engine. You state what you want and OpWorks Stacks/Chef makes it happen

Recipes tell OpsWorks Stacks/Chef WHAT you want the end results to be

## OpsWorks Stacks and Auto Scaling

**Time-based instances** follows a predictable time table

**Load-based instances** allows a stack to create and remove instances depending on the traffic. Start when above 80% utilization and stop when below 60%

# Amazon Inspector

Specifically for EC2 instances, it will need the Inspector Agent installed. You should ensure that it is installed, and using Step Functions with CloudWatch Event Rules will allow you to do it on a schedule.

# Trusted Advisor

It is a service that can scan your EC2 usage for Savings, Security, Service Limits, Performance, and Fault Tolerance

It does have a built in notification email, but that is done weekly. If you want daily stuff, you need to use Lambda Functions to trigger the scan, and either send the results to CloudWatch Event, or Publish straight to SNS. Then you could Have CloudWatch Events monitor Trusted Advisor checks and send SNS when they change

# NLB vs ALB vs API Gateway vs Route 53

Network Load Balancer does not support weighted routing

Application Load Balancer does not support Canary routing

Failover Routing only works when the route is considered Healthy or Unhealthy. It doesn’t care about 500s

API Gateway does support Canary Routing, however a private integration only works with a Network Load Balancer.

When thinking Canary think API Gateway, Lambda, and ECS. You can also utilize Route 53 weighted routing with another entire network behind the Alias. So a Classic LB with ASG with the new Application version for B/G Deployments.

# Route 53

Failover for Disaster Recovery. Obvious one is to have a Failover routing policy, route to healthy targets

Second is to Setup Health Checks for non-alias records to each service endpoint. Configure the NACL and Route Table to allow Route 53 to hit the healthchecks.

**Latency Routing Policy** – This is only really good for application performance. By routing users to the lowest latency region, it doesn’t actually help you for Failover scenarios.

# Centralizing VPC Logs and CloudWatch Logs

Using Kinesis and Lambda as an event handler to send logs to ES Cluster, you can create a CloudWatch Logs subscription filter and Kinesis data stream to stream them from sub-accounts.

# Other Services

## EMR

Big Data Cluster. Think of Hadoop. Stands for Elastic MapReduce

## AWS Migration Hub/Application Discovery Service

There is a dashboard for hybrid environments, but you need to deploy the Agentless Discovery Connector in an OVA file format to VMWare vCenter and install the Discovery Agent on EC2 Instances.