# 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

# Service Agents

**Systems Manager Agent, Inspector Agent, CloudWatch Agent, ECS Agent, Kinesis**

# ElasticBeanstalk

Creates an .ebextensions folder where you can configure Beanstalk environments. Under the .ebextensions folder you will find configs aws:autoscaling:updatepolicy:rollingupdate: under timebase.config or even healthbased.config

## .Ebextensions

This folder will house yaml files ending in .config that will configure environment specifics via Beanstalk. Here you can use a variety of option settings. Packages, sources, files, users, groups, commands, container\_commands, and services

## Rolling

Will not create extra instances, instead it will replace instances in a batch. Say you have 5 instances, if you do a Rolling update, you will take 1 or 2 down, and then only 3 would be available. Then it would be deployed, back up online, and then the next 3 instances will be deployed. RollingBatch, would bring up instances to keep a Min Capacity will the others are taken down.

## Rolling Batch

There are a few values to pay attention to. **Batch Size**, which is the number of instances that will be replaces. Defaults to 1/3 min size of the ASG. **Minimum Capacity,** the min # of instances that should be running during batch replacement. **Pause Time,** Amount of time to wait before starting the next batch after the batch is complete.

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

## Blue/Green

**If you have an RDS** instance in the environment. It will delete the RDS instance so you can’t do blue/green. Do not forget about RDS and ELB Blue/Green Deploys. They are dangerous because you lose the db.

## Fixing an RDS Environment

1. Create a snapshot of the RDS db
2. Disable db Termination
3. Ensure you apply immediately and validate it is not able to terminate
4. Create a second environment with EB without an RDS instance.
5. Connect to the existing RDS instance from the new environment
6. Swap the URLs, wait amount of time to ensure the dns propagates
7. Remove the Security group access from original environment
8. Delete old environment
9. Remove the stack
10. Delete the snapshot

# 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.

# AWS Organizations

**AWSServiceRoleForOrganizations** – service-linked role is meant to only allow Organizations to create service-linked roles for AWS Services.

# AWS Config

Take snapshots of the state of resources.

Continuous Monitoring, Continuous 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 preconfigured. Specifically, CFN Templates, 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

## Remediation with SSM

You can use Systems Manager Automation Documents (some are premade) to remediate findings. These are called AWS Config rules.

# Systems Manager

## Patch Manager

You should create tags on Environment and OS if you need to tag a bunch of instances in different environments.

## Automation

Self-Service Automation runbooks for infrastructure, create an AMI, use SSM Docs, or author your own workflow

Creating AMI, you can use prebuilt automation scripts like AWS-UpdateLinuxAMI, or AWS-UpdateWindowsAmi

## Inventory

When paired with AWS Config to audit applications configurations

## Maintenance Window

## Patch Manager

Roll out patches at scale, integrate with AWS Security Hub to receive alerts when a node goes out of compliance and monitor patch status of fleet

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.

You will want to create multiple Patch Groups to ensure that all servers aren’t taken offline at the same time.

## Run Command

Manage instances at scale without SSH access. Using CloudTrail you can audit any run commands that were called. When using a password, don’t use it without using System Parameters SecureString as it won’t be shown in the logs.

## Session Manager

How to manage a user’s ssh sessions through connect

## State Manager

Automatically remediate findings generated by Amazon Inspector. Use tags to create application groups for your nodes and then target nodes using the Targets parameter instead of specifying individual nodes. Centralized configuration repository for SSM Documents

# ACM/SSL

When you need to ensure end to end encryption if you are dealing with Viewers to CloudFront you need to set the Viewer Protocol Policy to HTTPS only, and you need a certificate issued by a trusted CA (Comodo, DigiCert, Symantec, or other 3rd party). Then if the custom origin is an ELB, you need the same thing and these can use ACM. If it is an EC2 it MUST BE 3rd Party or other trusted CA.

# CloudFormation

With an AMI you can utilize dynamic parameters. By creating a parameter in the parameter section of CFN Template like

1. Parameters :
2. AmiId :
3. Type : ‘AWS::SSM::Parameters::Value<AWS::EC2:ImageId>
4. Value : ‘/aws/service/ami/your-key’

The parameters used in the CFN Template will have access directly to the key value. The Ami can be referenced dynamically.

# CodeCommit

Does not have an EVENTS tab, you can setup notifications, but Events are a different beast entirely.

Also, must be within the same SNS region as repo if you want to trigger notifications directly.

**Trigger** Lambda Function of Create Branch, Delete Branch, and/or Push. You must create the Lambda Function, and the Lambda Function could have all of the destinations at its disposal.

## Communication

**Notification** – These can fire off SNS or Slack messages. From SNS, you can pass the notification along to Kinesis, Lambda, Email, Https, etc.… You can also Create a Lambda Function and set the Trigger as an SNS Topic.

**Triggers** – Here you can set the source to be any number of branches and various events. They will send information to either SNS or Lambda which can be handled similarly to Notification

You should **trigger** an action off the Trigger and leave the Notifications to be just that sending notifications. Meaning if you want to do something after some action occurred in CodeCommit, you would use trigger. If you need to send emails/SNS/Slack you would use **notifications**.

# CodeBuild

Uses Docker under the hood, alternative to Jenkins.

**Integrates** with KMS for artifacts, IAM for build permissions.

**Buildspec.yml** specifies the build instructions.

CloudWatch/S3 for Logging. Metrics for statistics. CW Alarms to notify you if you meet “thresholds” for failures.

Logs to S3 and CloudWatch. However, you can get the API calls made to CB via CloudTrail. CloudEvents to.

**Triggers** via SNS

Advantage to CodeBuild is it is serverless and everything will be destroyed after.

You can utilize SSM Parameter store in Environment variables during the build to ensure your db creds, or other things are not exposed in code.

## Communication

CodeBuild can trigger a SNS or a Slackbot from within the service. You can trigger off various events Build States – Failed/Succeeded/In-Progress/Stopped or Build Phase Fail/Success

CodeBuild -> SNS Topic -> SNS Subscription (Kinesis Firehose, SQS, Lambda, Https)/(Email) -> Down the rabbit hole

CodeBuild can also write logs to CW Logs during the build

## Triggering Build

You can setup a schedule in the Console.

# CodeDeploy

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

## Environment Variables

There is an appsec.yml file that contains various things needed for deployments

**Deployment Id, Deployment Group Name, Deployment Group Id, and Lifecycle Event**

## On-Prem

Create IAM Role with CD permissions, obtain and store a set of STS credentials allowing AssumeRole, set up a cron job to refresh, Install CodeDeploy Agent, Register Server with CodeDeploy, create tags for On-prem servers, Set up deployment group based on tags, Deploy app using deployment group

## Notifications/Events Triggering

AWS Lambda Functions, Kinesis Streams, SQS, CloudWatch alarm actions, SNS topics.

## Deployment Issues

There can be a timeout if your script runs too long. If you configure the timeout yourself and the scripts wind up taking too long.

## 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 Code Deployments**

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**

**Ec2 Instance Profile** needs access to S3 since it will get the app from there, or GitHub.

Code Deploy requires and EC2 Agent installed on the EC2 instance that will poll CodeDeploy until a job comes up. The CodeDeploy is where you will have appspec.yml

# CodePipeline

Think of CodePipeline as the way to integrate CodeCommit and CodeDeploy by creating artifacts in S3. The optimal usage is to have CodeCommit trigger a CW Event that will kick off your CodeDeploy. You could have CodePipeline check for changes in CodeCommit every few seconds.

**MultiRegions**

You could use a CodePipeline to trigger CodeDeploy in different regions. That is a possibility.

## 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. 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**

## Parallel Runs

Setting the **runOrder** to be the same, you can make actions run in parallel.

# 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.

## Events

**Data Events** – Disabled by default, can be used to track changes to data in S3.

**Management Events** – Provide insight into management operations that are performed on AWS Resources. Can include API calls, and other things like ConsoleLogin

**Insight Events** – Are logged if you have Insight Events enabled, CloudTrail can log unusual activity

## S3 Integration

If you want near real-time change tracking to data, and configuration. You can utilize S3 Data Events in CloudTrail.

## Defaults

In the Console, the CloudTrails default to all regions. However, in the API or CLI, it will default to one region. Max of **5** trails per region, and any cross-region trails will count toward the 5.

**Organization Trail** will log events of All Accounts if the Trail was created by the management account

# 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

## Targets

Api Destination, Api Gateway, Batch Job Queue, CW Log Group, CodeBuild Project, CodePipeline, EC2 CreateSnapshot API, EC2 ImageBuilder, EC2 RebootInstance, EC2 StopInstance, EC2 TerminateInstance, ECS Task, Other Event Bus in current/different account, Firehose Delivery, Glue workflow, Incident Manager, Inspector, Kinesis, Lambda, Redshift, Sagemaker, SNS, SQS, Step Functions, Systems Manager (Automation, OpsItem, Run Command, Logs),

## Service Source

Athena, Autoscaling, CodeStar, Glue, Xray, Backup, Batch, CloudFormation, Cloud Front, Cloud (Trial, Watch , Shell , Watch App Insights, all Code Services, Cognito, DynamoDB, Beanstalk, ECS, Kinesis, Organizations, Lambda, KMS, Redshift, RDS, Route 53, Secrets Manager, STS, SQS, SNS, S3, Step Functions, Systems Manager, WAF, Trusted Advisor, Inspector, Health, IAM, GuardDuty

# 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

# CloudWatch Alarms

Can’t get results directly from Lambda

The Alarm can trigger a Notification to either an email, or SNS

The Alarm can also Trigger **AutoScalingGroup** for EC2 or EC2, **EC2 Per-Instance**, or **System Manager** with an OpsItem of various categories. There is also an Incident that can be fired off which will initiate a **Response Plan**

You get CW Metrics feed to CloudWatcj, from the Metrics you can create an Alarm. The Alarm has 3 states that can Trigger actions. Ok, In Alarm, Insufficient Data. Each can kick off SNS, EC2 Action, SSM CreateOpsItem/Create Incident that can fire off some SSM Automation, AutoScalingGroup can also happen (EC2 or ECS)

# Placement Group

The placement group is more of strategy. There are 3 different strategies.

**Cluster** – Which ensures low latency, high network traffic. But the availability of the application can take a hit because the core of this idea is to ensure instances are all in the same AZ on the same hardware. There is a small issue that could occur when using this strategy. If you add a new instance, and there isn’t enough space on the shared hardware, you can get a Capacity Error. You should create the instances together; ensure they are all the same type. You don’t want to start and stop an instance because you run the risk of not being able to start the instance.

**Partition** ­– The happy medium where instances share the same hardware but are on different partitions across different hardware. There are 3 partitions, and they don’t share a power source between the racks. If a rack completely goes down. You only lose a 1/3 of computing power for the time.

**Spread** – Is the opposite of Cluster. The instances are not allowed to share the same hardware. This is to protect from any failures that could cause multiple instances to go down because of the same underlying failures.

# ASG (Auto Scaling Groups)

There is some pretty difficult things to remember with ASG. One major thing is the Lifecycle for an ASG on an EC2 instance is when you get to the **Terminating** phase, 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 SystemManager 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 rolls-back to the old ASG. CFN won’t detach 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.

## Ensure High Availability

Use AutoScalingReplacingUpdate

## Troubleshoot/Diagnose issues with AutoScaling

Configure **WaitOnResourceSignals** to false

Update **MinSuccessfulInstancesPercent** to be something more than 1 or the default of 1/3 min number of instances in ASG

Suspend HealthCheck, ReplaceUnhealthy, AZRebalance, AlarmNotification, ScheduleActions

# AWS Nested Stacks

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

# Kinesis

Currently the only streaming service that supports cross-accounts

# 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

## AppSec Hooks for Deployment

**Start** -> BeforeAllowTraffic -> **AllowTraffic** -> AfterAllowTraffic -> **End**

## As Event Handler

DynamoDB, Kinesis, and S3

# ECS

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

## AppSec Hooks for Deployment

**Start** -> BeforeInstall -> **Install** -> AfterInstall -> **AllowTestTraffic** -> AfterAllowTestTraffic -> BeforeAllowTraffic -> **AllowTraffic** -> AfterAllowTraffic -> **End**

Container -> 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)

## Redeploy

You can Force a new Deployment, but “Redeploy” is not an option. No such thing as “automatic platform version”

## 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.

## Secrets Manager

Amazon Metadata Service talks to ECS Agent, which talks to the ECS ACS (Amazon Common Service), which then the ECS Agent Talks to the ECS Task, that talks to Parameter Store, who then talks to KMS

## Fargate

Using the CLI to update a service will only take platform version if it is a Fargate launch type

## Xray

Uses the Xray Daemon, on UDP Port 2000. The configuration of network mode, and port settings needs to happen in the Task definition. Xray-daemon.config is for Beanstalk,

# EC2

## ASG

Has a different lifecycle than a typical EC2 instance.

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

------------------------------------>Terminating->(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 must 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.

**Important note** There is an EC2 in place, EC2 with ELB in place, and EC2 with ELB Blue/Green. If you have an ELB, you will see the BlockTraffic, and AllowTraffic hooks. However, If it is Blue/Green, the BlockTraffic won’t happen until after ValidateService. Where the in-place will start with block traffic.

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 lifecycle 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

## SQS

Doesn’t enable real time data, for ClickStreams. You would want Kinesis Data Streams, if you want to go expensive you could use EMR

# S3

How to monitor changes. CloudTrail that sends logs to CloudWatch Log Group, create a CloudWatch Event rule for S3 Bucket Policy events on the log group. Create an Alarm based on the event that will send a notification

Setup CloudTrail, then use a Lambda Function to be triggered with a target of CW Log Groups

# AWS Managed Services

Is basically 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 a really useful disaster recovery setup for a db as it can be promoted to master very quickly.

## RTO and RPO

**RTO** – Recovery Time Objective, is the amount of time you can be offline after being notified of the outage. This drives your speed of recovery

**RPO** – Recovery Point Objective, is the length of time between backups. Use Recovery point as a figure for your backup time.

Use RTO as the amount of time it takes you to recover and RPO as how much data you can lose.

# Caching

## ElastiCache w/Redis

Allows for multi AZ replication. This is important if you are talking about session management while working with multiple AZs.

## ElastiCache w/MemcacheD

Does not support replication

## Redis

A single instance of the cache, which is great for Key Value pair caching, or session management.

## MemcacheD

A single instance, which is not better for session logging

## Sticky Session

Good for a load balancer, but not during immutable deployments as the Sticky Session will ensure you are taken back to the same EC2 instance, not going to fix the issue of ALB losing sessions when deploying a new fleet, or batch.

# Oracle RAC

This is a tricky question. They throw Aurora and RDS at you but the real answer is to migrate to large EBS-Backed EC2 instances. It is the fastest way, then you can utilize SSM to do the Patch Management.

# DynamoDB

## Secondary Indexes

**Global Secondary Index** – Is an index with a partition key and sort key that is DIFFERENT from the original base table. So, if you have Team and Number as partition and sort for base. Then you created a GSI of Position and Name that would be a GSI. Where you could use a

**Local Secondary Index** – Is an index with the same partition key and a different sort key. So, you would use Team and Number, then Team and Position. You can specify what properties are projected into the LSI table. You can have up to 5 LSI per table.

**Note** You can’t add a LSI to an existing table, you have to actually create a second table and migrate the data…fml

## Dax

In system caching for DynamoDB.

## Kinesis

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

## Cost Effective Replication

You can create a Global DynamoDb with the DynamoDb Streams option, then set up replica tables in other regions where you want your data replicated. Then you can store the individual transactions in that regions DynamoDb, and replay events across the other dbs.

# 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 OpsWorks 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%

## BuildPipeline

Integrating CodePipeline with OpsWorks could actually be counter intuitive. If you need to deploy, utilize OpsWorks Stacks, and create a new stack, and deploy a layer using Blue/Green

# 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. It will help with security auditing

# Trusted Advisor

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

Use it for calculating Utilization…and you can kick off a Lambda Function

**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. Keep in mind that Trusted Advisor would create this Event, you can’t create the Event from Event Bridge

# 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 health checks.

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

# VPC

**Nat Gateway** - When you have instances on a private subnet, and you want them to be able to reach the internet, there needs to be a NAT Gateway (or Nat Instance which is much more work intensive, like the pro version of configuration. There are limitations and things you have to do to ensure it works). Unless we are talking IPV6 then you use an IG for network egress (going out, ingress is coming in).

1. Nat Gateway needs a public ip
2. Nat Gateway needs to live on a public subnet
3. Uses ports 1024-65535

**Internet Gateway** – You have one of these when you want your EC2 instances with a public IP to be able to reach the internet. You also need to have the subnet routing table routing traffic to the internet through your IG

**Security Groups** – Are stateful, meaning they only **allow** traffic, and the traffic is specifically outbound. They are called stateful because the outbound traffic on whatever port if allowed, the response traffic will not be blocked. Unless of course, the NACL is blocking that. These are set at instance level.

**NACL (Network Access Control List)** – Are stateless, meaning they **allow** and **deny** traffic. You could make a request to port 2000, and the response could come back on port 4000. If you do not have both ports allowed, that request will not be successful. The NACL is set on a subnet level.

# 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.

## Amazon Guard Duty

Is more about Threat Detection, Resource security, and malicious activity in your AWS Account

You should push findings to Kinesis Data Firehouse, not Kinesis Data Streams with a custom shell script in a Lambda Function

## Amazon Macie

More about compliance. Uses machine learning to discover, classify, and protect sensitive data. The protection bit, includes S3 for data leaks. Alerted if there are risks of unauthorized access or leaks

## Aws Shield

Is more about network security, specifically a DDOS protection service. Use AWS Shield and AWS WAF together to monitor and mitigate issues.

## EC2 Rescue

Use this when you have a windows instance that is having boot issues or ssh/rdp issues

## Amazon Cognito

Using Providers you will need to use the temporary auth token with STS AssumeRoleWithWebIdentity API. NOT AssumeRoleWithSAML as this is used for Active Directory

## Amazon Rekognition

Some sort of facial recognition service

## AWS Storage Gateway

Be careful with a tape gateway, it can’t handle real time connections since it is tied to Glacier, and archiving.

# Practice Exam Question Notes

**SAM Resources** – Lambda, DynamoDb, ApiGatewayV2, RestApi, StateMachine,

When you are dealing with a sam template you will see serverless in the template for resources, not Lambda, ApiGateway, and DynamoDb. Another thing to point out is for Lambda to run a CloudFormation update something has to change, either the BucketKey, BucketName, or the Object Version in S3

**AWS Config Rules** – This one is shitty because they try to get less specific with the usage of AWS Config. They say configuration item changes, and not compliance changes. It is sort of shitty because in this instance they allow you to assume the Event is being spawned of the AWS Config rule.

**CloudWatch Metrics** – When you are dealing with utilization metrics from the EC2 instance. Just create the Alarm off the metric to send the SNS. You don’t need to create a metric filter. You create a CW Log Filter when you need to selective take things from the Log stream. However, a metric is an expected value, unless the metric doesn’t already exist.

**ElasticBeanstalk CLI (eb create)** – It will override .ebextensions.config files. The CLI is outdated so the question is a little off, but the instance type can be overwritten by the CLI if you don’t specify.

--Running a bash script after the deployment will require you to get into the post steps <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/custom-platform-hooks.html>

**Canary Deploy** – When Using a ApiGateway backed by custom integration Lambada functions to call business logic. Apparently you can add a Canary release to the existing API production stage, which you can configure the settings to direct 10% and 90% traffic for a week. <https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html>

**SAM Deploy** – AmazonAPI Gateway that is integrated with AWS Lambda Functions where they store data on DynamoDb, and the Lambda Functions store session information in ElastiCache for Redis cluster can be done in a SAM template

**Service Quotas** – This is a weird one, you need Organizations and Service Quota together to be able to check Service usage on a per account basis.