
aws
certified
Solutions
Architect
Professional
DOLFINed

AWS SERVICES AND STRATEGIES FOR DEPLOYMENT AND OPERATIONS MONITORING/MANAGEMENT

AWS CLOUDFORMATION

Deployment Management in AWS

Cloud Formation – Part 1

AWS CloudFormation

What is AWS CloudFormation?

- AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources
  - This will allow you to spend less time managing those resources and more time focusing on your applications that run in AWS.
  - You can model your Cloudformation templates in JSON or YAML

- You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), then
  - AWS CloudFormation takes care of provisioning and configuring those resources for you.
    - The result is what is called a **"stack"**

- You don't need to individually create and configure AWS resources and figure out dependencies; AWS CloudFormation handles all of that.

AWS CloudFormation - Benefits

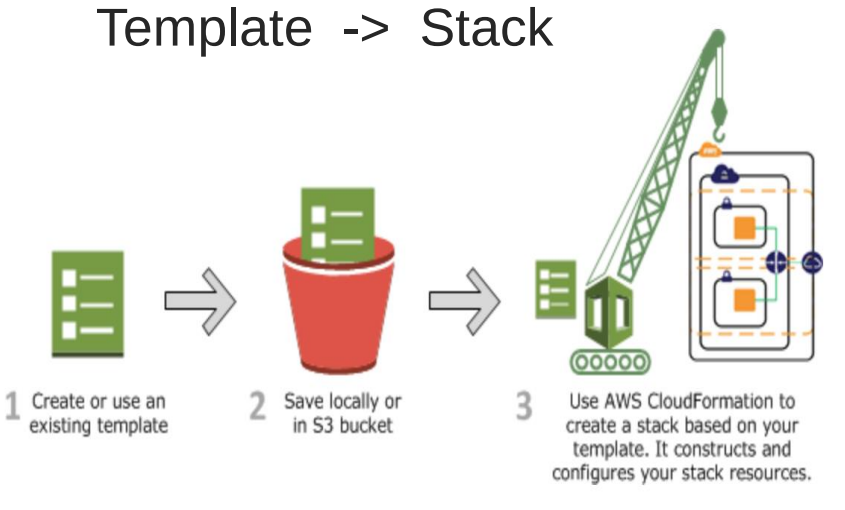
- Simplify Infrastructure Management
  - You can create, update, version control and delete your stacks and Cloud formation takes care of it.

- Quickly Replicate Your Infrastructure
  - When you use AWS CloudFormation, you can reuse your template to set up your resources consistently and repeatedly.
    - Just describe your resources once and then provision the same resources over and over in multiple regions.

- Easily Control and Track Changes to Your Infrastructure
  - Change your resources, upgrade, update, roll back changes, basically**, manage your infrastructure as code**

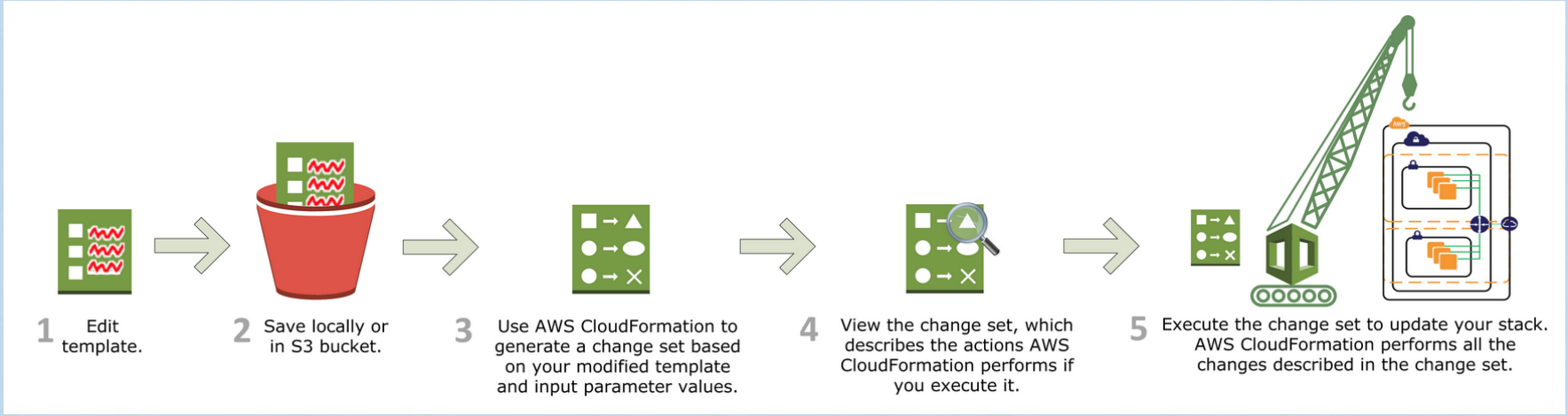
AWS CloudFormation – How it works

- AWS CloudFormation can perform only actions that you (who created the template) have permission to do.
- If you specify a template file stored locally, AWS CloudFormation uploads it to an S3 bucket in your AWS account.
- AWS CloudFormation creates a bucket for each region in which you upload a template file.
- The buckets are accessible to anyone with Amazon S3 permissions in your AWS account.
- If a bucket created by AWS CloudFormation is already present, the template is added to that bucket.
- You can use your own bucket and manage its permissions by manually uploading templates to Amazon S3.



AWS CloudFormation – Updating the Stack

- To update a stack, create a **change set** by submitting a **modified version of the original stack template**, different input parameter values, or both.
- AWS CloudFormation compares the modified template with the original template and generates a change set.



- Updates can cause interruptions.
- Depending on the resource that you are updating, an update might interrupt or even replace an existing resource.

AWS CloudFormation – Deleting a Stack

- If AWS CloudFormation cannot delete a resource,
  - The stack will not be deleted.
  - Any resources that haven't been deleted will remain until you can successfully delete the stack.

- If you want to delete a stack but want to retain some resources in that stack, **you can use a deletion policy to retain those resources.**

Deployment Management in AWS

Cloud Formation – Part 2

AWS CloudFormation – Template Components

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

- Templates include several major sections.
- **The Resources section is the only required section.**
- Some sections in a template can be in any order.
  - However, as you build your template, it can be helpful to use the logical order shown because values in one section might refer to values from a previous section.

AWS CloudFormation – Template Components

- **Format Version (optional)**
  - The AWS CloudFormation template version that the template conforms to.
- **Description (optional)**
  - A text string that describes the template.
- **Metadata (optional)**
  - Objects that provide additional information about the template.
- **Parameters (optional)**
  - Values to pass to your template at runtime (when you create or update a stack).
  - You can refer to parameters from **the Resources and Outputs** sections of the template.

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

AWS CloudFormation – Template Components

- **Mappings (optional)**
  - A mapping of keys and associated values that you can use to specify conditional parameter values.
    - Similar to a lookup table.
  - You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function in the **Resources and Outputs** section.
- **Conditions (optional)**
  - Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update.

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

AWS CloudFormation – Template Components

- **Transform (optional)** For serverless applications (also referred to as Lambda-based applications),
  - Specifies the AWS SAM version to use.
- **Resources (Is a must in a template –is required)**
  - The only template part that must be present in the template
  - Specifies the stack resources and their properties, such as an EC2 instances or S3 buckets.
  - You can refer to resources in **the Resources and Outputs sections** of the template.
- **Outputs (optional)**
  - Describes the values that are returned whenever you view your stack's properties.
    - For example, you can declare an output for an S3 bucket name and then call the aws cloudformation describe-stacks AWS CLI command to view the name.

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : "JSON string",

  "Metadata" : {
    template metadata
  },

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Transform" : {
    set of transforms
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

Deployment Management in AWS

Cloud Formation – Part 3

AWS CloudFormation - Intrinsic Functions

- Are AWS CloudFormation (several) built-in functions that help you manage your stacks.
- Use intrinsic functions in your templates to assign values to properties that are not available until runtime.
- You can use intrinsic functions in:
  - Resource properties, Outputs, Metadata attributes, and Update policy attributes.
  - You can also use intrinsic functions to conditionally create stack resources.

AWS CloudFormation - Intrinsic Functions

- Some of the Intrinsic functions are:
  - GetAtt
  - FindInMap
  - Ref
  - GetAZs
  - ImportValue

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html

AWS CloudFormation – Resource Attributes

They are used to further control additional resource relationships and behaviors of your resources specified. Below are the resource attributes you can add in your templates:

- CreationPolicy
  - When you want to delay a resource creation until another resource is created
- DeletionPolicy
  - If you want to retain or create snapshots for resources before deleting the template
- DependsOn
  - Specifies the resource's creation is dependent on the creation of another resource
- Metadata
  - If you want to add more metadata to your resources
- UpdatePolicy
  - Defines how the resource update should be carried out

AWS CloudFormation – Creation Policy

- Using the AWS::CloudFormation::WaitCondition resource and CreationPolicy attribute, you can do the following:
  - Coordinate stack resource creation with other configuration actions that are external to the stack creation
  - Track the status of a configuration process

Examples:

- You can start the creation of another resource after an application configuration is partially complete, or
- You can send signals during an installation and configuration process to track its progress.
- You can use it to instruct cloudformation to delay the creation of resource (ELB, ASG..etc) until an application has launched successfully on an EC2 instance

Deployment Management in AWS

Cloud Formation – Part 4

AWS CloudFormation – Deletion Policy

DeletionPolicy Attribute

- With the **DeletionPolicy attribute** you can preserve or (in some cases) backup a resource when its stack is deleted.
- You specify a DeletionPolicy attribute for each resource that you want to control.
- Note that this capability also applies to stack update operations that lead to resources being deleted from stacks.
  - For example, if you remove the resource from the stack template, and then update the stack with the template.

AWS CloudFormation – Deletion Policy Options

DeletionPolicy Attribute

- **Delete**
  - AWS CloudFormation deletes the resource and all its content if applicable during stack deletion. You can add this deletion policy to any resource type.
  - Be aware of the following considerations:
    - For AWS::RDS::DBCluster resources, the default policy is Snapshot.
    - For Amazon S3 buckets, you must delete all objects in the bucket for deletion to succeed.

- **Retain**
  - AWS CloudFormation keeps the resource without deleting the resource or its contents when its stack is deleted.
  - You can add this deletion policy to any resource type.

- **Snapshot**
  - Use it for resources that support snapshots (EBS volumes, Elasticache, RDS, Redshift)
    - AWS CloudFormation creates a snapshot for the resource before deleting it.

Deployment Management in AWS

Cloud Formation – Part 5

AWS CloudFormation – Nested Stacks

- Nested stacks are stacks that create other stacks.
- To create nested stacks, use the AWS::CloudFormation::Stack resource in your template to call/reference other templates within your template.
- Use Nested Stacks to Reuse Common Template Patterns
- As your infrastructure grows, common patterns can emerge in which you declare the same components in each of your templates.
- You can separate out these common components and create dedicated templates for them.
  - That way, you can mix and match different templates but use nested stacks to create a single, unified stack.

Grouping Resources by Lifecycle and Ownership

- As your stack grows in scale and broadens in scope, managing a single stack can be cumbersome and time consuming.
- By grouping resources with common lifecycles and ownership, owners can make changes to their set of resources by using their own process and schedule without affecting other resources.

CloudFormation – Cross Stack References

Use Cross-Stack References to Export Shared Resources

- When you organize your AWS resources based on lifecycle and ownership,
  - Instead of including all resources in a single stack,
  - You create related AWS resources in separate stacks;
    - Then you can refer to required resource outputs from other stacks.
- Use cross-stack references to export resources from a stack so that other stacks can use them.
- Stacks can use the exported resources by calling them using the Fn::ImportValue function.

CloudFormation Security Best Practices

- Only allow specific templates and stack policies (you can control the source of the stack template and stack policy documents)
  - Stack policy: It is a JSON document that describes what can you do during an update operation

- Restrict what resource types can an IAM principal create

CloudFormation and CloudTrail and SNS

- CloudFormation integrates with CloudTrail

- If logging is enabled, CloudTrail will log all AWS services API calls that were done by the CloudFormation template(s).

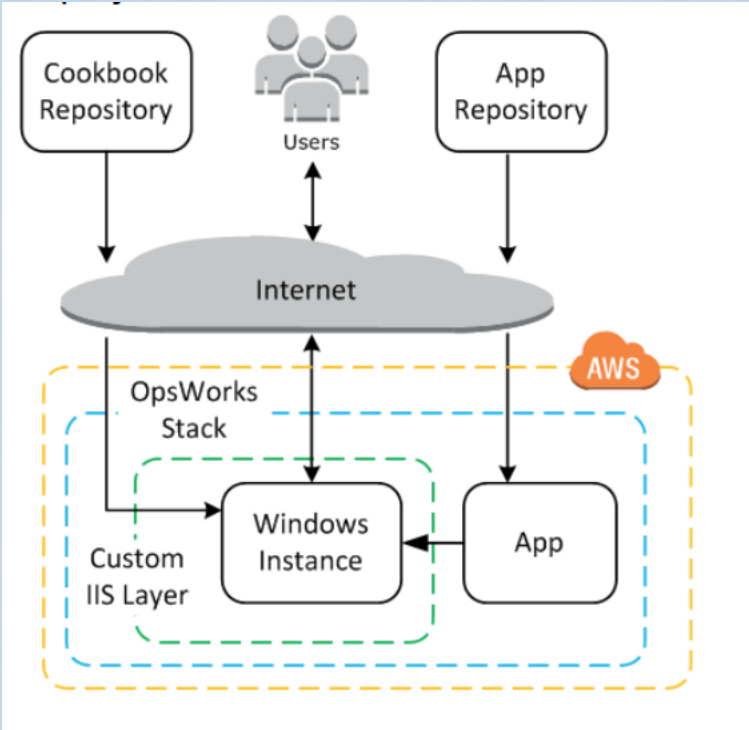
AWS OPSWORKS

Amazon OpsWorks

Introduction

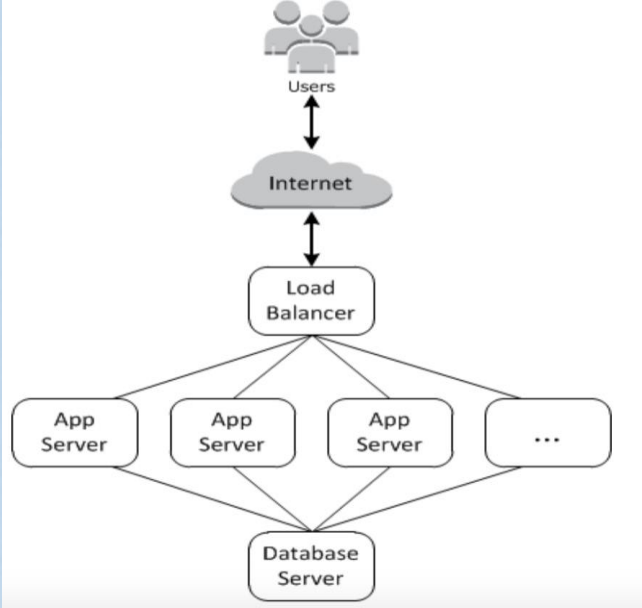
OpsWorks Stacks

- Cloud-based computing usually involves groups of AWS resources, such as Amazon EC2 instances and Amazon Relational Database Service (RDS) instances, which must be created and managed collectively.
  - In addition to creating the instances and installing the necessary packages, you typically need a way to distribute applications to the application servers, monitor the stack's performance, manage security and permissions, and so on.
- AWS OpsWorks Stacks provides a simple and flexible way to create and manage stacks and applications
- The stack is the core AWS OpsWorks component.



OpsWorks Stacks

- For example, a web application typically requires application servers, database servers, load balancers, and so on.
  - This group of instances is typically called a stack.
- OpsWorks allows SSH and RDP access to Stack instances
- Resources can be managed only in the region in which they are created.
- Resources that are created in one regional endpoint are not available, nor can they be cloned to, another regional endpoint.



OpsWorks – Chef Cookbooks and Recipes

- **A *cookbook***
  - It is a package file that contains configuration information, including instructions called recipes.
- **Recipes:**
  - A *recipe* is a set of one or more instructions, written with Ruby language syntax, that specifies the resources to use and the order in which those resources are applied.
    - AWS OpsWorks then automatically runs them at the appropriate time.
    - Recipes can also be run manually, at any time.
- ***Resource:***
  - *In Chef terms, A resource*, as used in Chef, is a statement of configuration policy.
    - (This is different from what a resource is to OpsWorks)

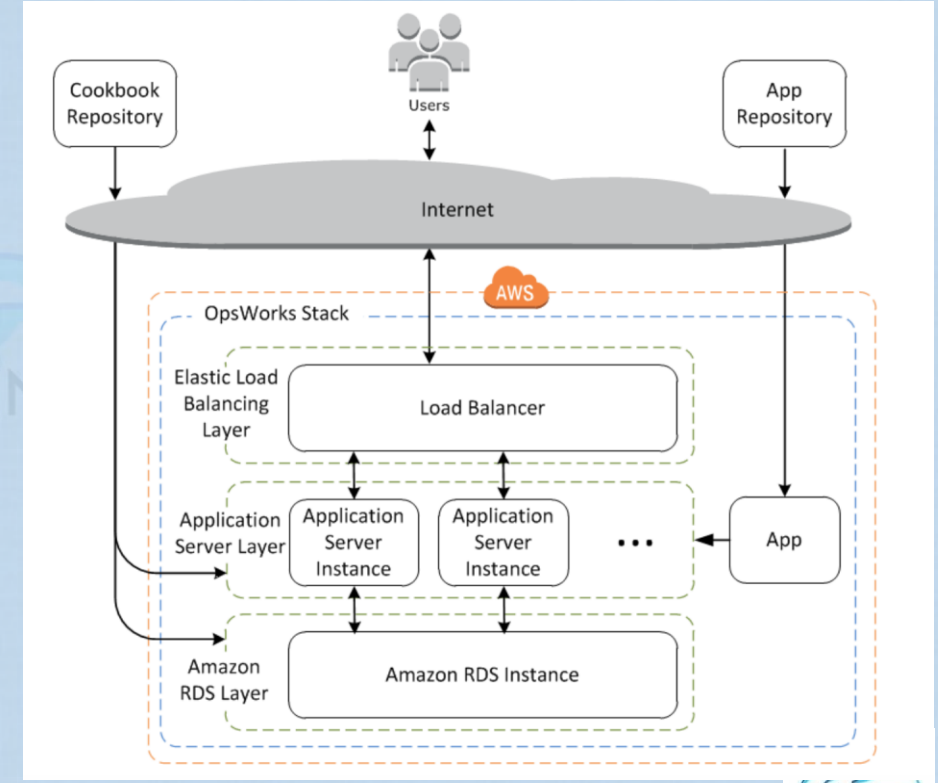
Amazon OpsWorks

Stacks and Layers

OpsWorks – Simple Application Stack

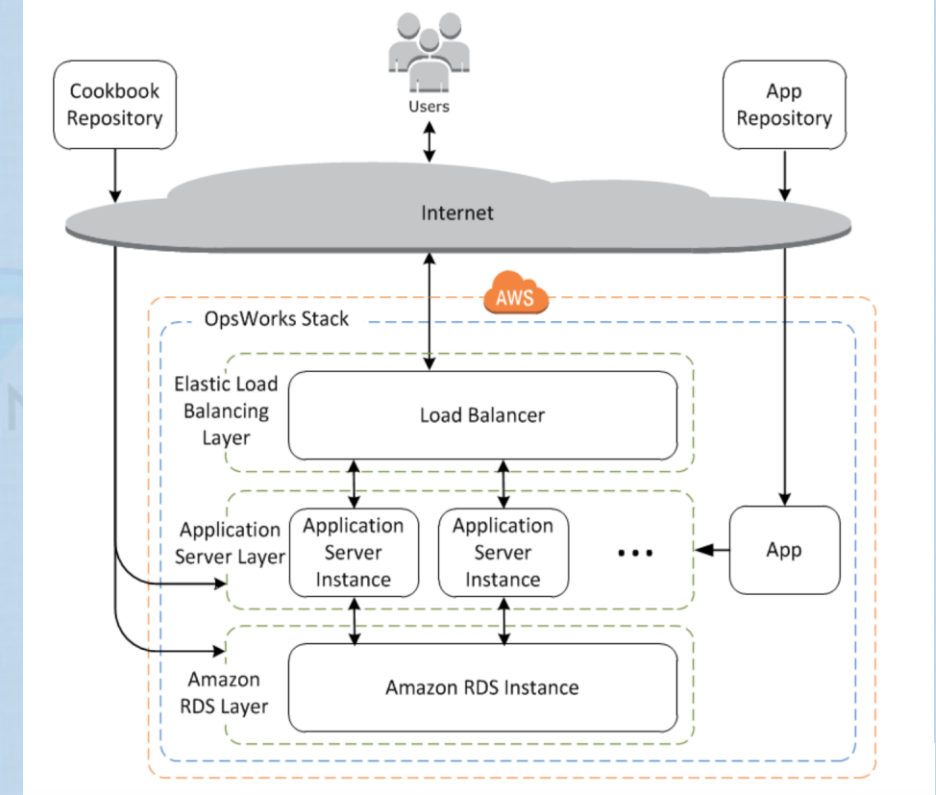
The stack is basically a container for AWS resources— Amazon EC2 instances, Amazon RDS and DynamoDB database instances, Elastic Load balancers.

- These resource have a common purpose and should be logically managed together.
- The stack helps to manage these resources as a group
- The stack also defines some default configuration settings, such as the instances' operating system and AWS region.



OpsWorks Stacks - Layers

- A stack can have one or more layers.
- A layer represents a set of Amazon EC2 instances that serve a particular purpose, such as serving applications or hosting a database server.
- Layers give complete control over which packages are installed, how they are configured, how applications are deployed, and more.
- For all stacks, AWS OpsWorks includes service layers, which represent the following AWS services:
  - AWS RDS
  - Elastic Load Balancing
  - Amazon Elastic Container Service



OpsWorks Stack Layers

- All stacks can include one or more layers, which start with only a minimal set of recipes.

- Layers depend on Chef recipes to handle tasks such as:
  - Installing packages on instances,
  - Deploying apps,
  - Running scripts.

- After you create a layer, some **properties (such as AWS region) are immutable**, but you can change most of the layer configuration at any time.

- You package your custom recipes and related files in one or more cookbooks and store the cookbooks in a repository such Amazon S3 or Git.

OpsWorks – Recipes and LifeCycle Events

- You can run recipes manually, but AWS OpsWorks Stacks also lets you automate the process by supporting a set of five lifecycle events
  - **Setup** occurs on a new instance after it successfully boots.
  - **Configure** occurs on all of the stack's instances when an instance enters or leaves the online state.
  - **Deploy** occurs when you deploy an app.
  - **Undeploy** occurs when you delete an app.
  - **Shutdown** occurs when you stop an instance.

- Each layer can have any number of recipes assigned to each event.
  - These recipes handle a variety of tasks for that event and layer

- When a lifecycle event occurs on a layer's instance, AWS OpsWorks runs the associated recipes.

Amazon OpsWorks

Components - Instances

OpsWorks - Instances

- When you start the OpsWorks instance,
  - AWS OpsWorks launches an Amazon EC2 instance using the configuration settings specified by the instance and its layer.
  - After the Amazon EC2 instance has finished booting,
    - **AWS OpsWorks installs an agent** that handles communication between the instance and the service and runs the appropriate recipes in response to lifecycle events.
- An instance can belong to multiple layers,
  - If that is the case, then AWS OpsWorks runs the recipes for each layer the instance belongs to

OpsWorks Auto Healing

- **AWS OpsWorks Instance Autohealing**
  - AWS OpsWorks support instance autohealing in the following manner
    - If an OpsWorks agent on an Instance stops communicating with the service,
    - AWS OpsWorks automatically stops and restarts the instance.
    - Note An instance can be a member of multiple layers.
      - If any of those layers has auto healing disabled, AWS OpsWorks Stacks does not heal the instance if it fails.
    - If a layer has auto healing enabled—the default setting—AWS OpsWorks Stacks automatically replaces the layer's failed instances

Amazon OpsWorks

- Applications
- Resource Management
- Integration with IAM

OpsWorks Components - Apps

- An AWS OpsWorks Stacks *app* represents code that you want to run on an application server.
  - The code itself resides in a repository such as an Amazon S3 archive; the app contains the information required to deploy the code to the appropriate application server instances.

For OpsWorks to manage your applications,

- Applications and related files need to be stored in a repository, such as an Amazon S3 bucket.

- Apps can be deployed,
  - Automatically
    - When you start OpsWorks instances, AWS OpsWorks automatically runs the instance's Deploy recipes.
  - Manually
    - If you have a new app or want to update an existing one, you can manually run the online instances' Deploy recipes.

OpsWorks – Integration with IAM

AWS OpsWorks integrates with AWS IAM to control access to AWS OpsWorks Stacks, including:

- How individual users (developers/admin..etc) can interact with each stack, this controls access rights such as
  - Whether they can create stack resources such as layers and instances, or
  - Whether they can use SSH or RDP to connect to a stack's Amazon EC2 instances.
- How AWS OpsWorks Stacks can act on your behalf to interact with AWS resources such as Amazon EC2 instances.
- How apps that run on AWS OpsWorks Stacks instances can access AWS resources such as Amazon S3 buckets.
- How to manage users' public SSH keys and RDP passwords and connect to an instance.

Amazon OpsWorks

- Elastic Load Balancing
- Best Practices
- Managing Permissions
- Monitoring

OpsWorks and Elastic Load Balancing (ELB)

- After you attach a load balancer to a layer, AWS OpsWorks Stacks does the following:
  - Deregisters any currently registered instances.
  - Automatically registers the layer's current instances and adds new instances as they come online.
  - Automatically deregisters instances when they leave the online state, including load-based and time-based instances.
  - Automatically activates and deactivates the instances' Availability Zones.

- Notes:
  - You can attach only one load balancer to a layer and each load balancer can handle only one layer.
    - This means that you must create a separate Elastic Load Balancing load balancer for each layer in each stack that you want to balance and use it only for that purpose
  - AWS OpsWorks Stacks does not support Application Load Balancer. You can only use Classic Load Balancer with AWS OpsWorks Stacks.

OpsWorks Best Practices – Managing Permissions

- Do not use your account's root credentials to access AWS resources.
- Create IAM users for your employees and attach policies that provide appropriate access.
  - Each employee can then use their IAM user credentials to access resources.
- Employees should have permissions to access only those resources that they need to perform their jobs.
- Employees should have permissions to use only those actions that they need to perform their jobs.

OpsWorks – Monitoring and Logging

- AWS OpsWorks provides several features to help you monitor your stack and troubleshoot issues with your stack and any recipes.
- For all stacks:
  - AWS OpsWorks provides a set of custom CloudWatch metrics for Linux stacks, which are summarized for your convenience on the Monitoring page.
  - AWS OpsWorks supports the standard CloudWatch metrics for Windows stacks.
    - You can monitor them with the CloudWatch console.
  - CloudTrail logs, which record API calls made by or on behalf of AWS OpsWorks in your AWS account.
  - An event log, which lists all events in your stack.
  - Chef logs that detail what transpired for each lifecycle event on each instance, such as which recipes were run and which errors occurred.
  - Linux-based stacks can also include a Ganglia master layer, which you can use to collect and display detailed monitoring data for the instances in your stack.

AWS ELASTIC BEANSTALK

AWS Elastic BeanStalk

Introduction

Elastic BeanStalk

- AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with supported programming languages, on familiar servers such as **Apache, Nginx, Passenger, Docker, Tomcat, and IIS**.

- With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications.

- Elastic Beanstalk provides developers and systems administrators an easy, fast way to deploy and manage their applications without having to worry about AWS infrastructure.

- AWS Elastic Beanstalk reduces management complexity without restricting choice or control.

Elastic BeanStalk – How?

- You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

- AWS Elastic Beanstalk provides platforms for:
  - Programming languages (Java, .NET, PHP, Node.js, Python, Ruby, Go),
  - Elastic Beanstalk supports different platform configurations for each language.
  - Web containers (Tomcat, Passenger, Puma) and Docker containers, with multiple configurations of each.
  - Elastic Beanstalk also supports custom built platforms that you can create and use

- Elastic Beanstalk provisions the resources needed to run your application, including one or more Amazon EC2 instances.

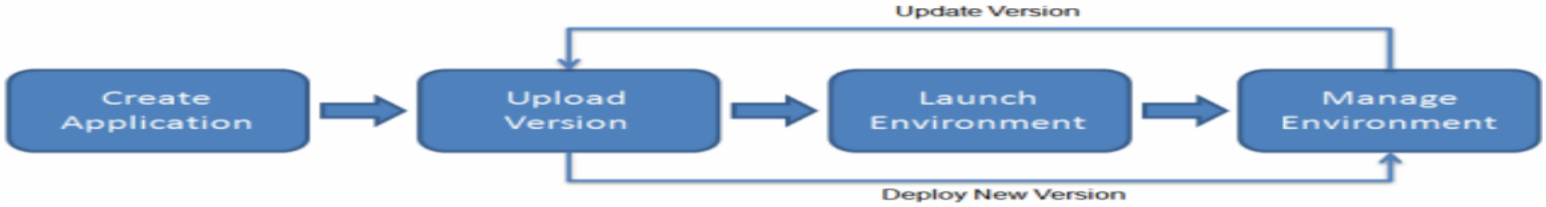
Elastic BeanStalk

- To use Elastic Beanstalk, you need to:
  - Create an application,
  - Upload an application version in the form of an application source bundle (for example, a Java .war file – Web application ARchive) to Elastic Beanstalk, and then,
  - Provide some information about the application.
  - Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code.
  - After your environment is launched, you can then manage your environment and deploy new application versions.

- After you create and deploy your application, information about the application—including metrics, events, and environment status—is available through the AWS Management Console, APIs, or Command Line Interfaces

Elastic BeanStalk – Data Persistence

- Elastic Beanstalk applications run on Amazon EC2 instances that have no persistent local storage.
- When the Amazon EC2 instances terminate, the local file system is not saved, and new Amazon EC2 instances start with a default file system.
- You should design your application to store data in a persistent data source.
- Amazon Web Services offers a number of persistent storage services that you can leverage for your application, such as S3, EFS, DynamoDB and RDS (not EBS)
- Elastic Beanstalk does not restrict your choice of persistent storage and database service options.



AWS Elastic BeanStalk

Concepts / Components

AWS Deployment – Elastic Beanstalk

Concepts and Components

- **Application:**
  - An application serves as a container for the environments that run your web app, and versions of your web app's source code, saved configurations, logs and other artifacts that you create while using Elastic Beanstalk.
- **Application version:**
  - In Elastic Beanstalk, an *application version* refers to a specific, labeled iteration of deployable code for a web application.
  - An application version points to an Amazon Simple Storage Service (Amazon S3) object that contains the deployable code such as a Java WAR (Web application ARchive) file for Java applications and .ZIP file for all other applications.
- **Environment**
  - Is a version that is deployed onto AWS resources.
  - Each environment runs only a single application version at a time, however you can run the same version or different versions in many environments at the same time.
  - When you create an environment, Elastic Beanstalk provisions the resources needed to run the application version you specified.

AWS Deployment – Elastic Beanstalk

Concepts and Components

- **Environment Tier:**
  - When you launch an Elastic Beanstalk environment, you first choose an environment tier.
  - The environment tier that you choose determines whether Elastic Beanstalk provisions resources to support an application that handles HTTP requests or an application that pulls tasks from a queue.
  - An application that serves HTTP requests runs in a web server environment.
  - An environment that pulls tasks from an Amazon Simple Queue Service queue runs in a worker environment.
- **Environment Configuration:**
  - An *environment configuration* identifies a collection of parameters and settings that define how an environment and its associated resources behave.
  - When you update an environment's configuration settings, Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources (depending on the type of change).

Concepts and Components

- **Configuration Template:**
  - A *configuration template* is a starting point for creating unique environment configurations.
  - Configuration templates can be created or modified by using the Elastic Beanstalk command line utilities or API.

AWS Elastic BeanStalk

EB Custom platform and Custom AMI

Elastic BeanStalk – Custom Platforms

- Elastic Beanstalk supports custom platforms.
- A custom platform lets you develop an entire new platform from scratch, customizing the operating system, additional software, and scripts that Elastic Beanstalk runs on platform instances.
- This flexibility allows you to build a platform for an application that uses a language or other infrastructure software, for which Elastic Beanstalk doesn't provide a platform out of the box.
- With custom platforms you use an automated, scripted way to create and maintain your customization, whereas with custom images you make the changes manually over a running instance.
- To create a custom platform, you build an Amazon Machine Image (AMI) from one of the supported operating systems—Ubuntu, RHEL, or Amazon Linux and add further customizations.
- You create your own Elastic Beanstalk platform using a Packer, which is an open-source tool for creating machine images for many platforms, including AMIs for use with Amazon EC2.

AWS Elastic BeanStalk

EB and Docker

Elastic BeanStalk - Docker

- A docker container is a standardized unit of software development, containing everything it needs to run including library binary files, system tools, code and runtime.

- Containers are created from a read—only template called a Docker Image
  - Docker images are in turn created from a Docker file,

- A **Dockerfile** is a script that contains collections of docker and OS commands and instructions that will be automatically executed in sequence in the docker environment for building a new docker image.
  - The docker file specifies the components that will be in the docket image, hence, in the containers run from that image.
- A Dockerfile is a text file that Docker reads in from top to bottom. It contains a bunch of instructions which informs Docker HOW the Docker image should get built.

Elastic BeanStalk and Docker Containers

- Elastic Beanstalk supports the deployment of web applications from Docker containers.
- All environment variables defined in the Elastic Beanstalk console are passed to the containers.
- By using Docker with Elastic Beanstalk, you have an infrastructure that automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.
- If a Docker container running in an Elastic Beanstalk environment crashes or is killed for any reason, Elastic Beanstalk restarts it automatically.

Elastic BeanStalk – Docker Platform Configurations

- **Single container configuration:**
  - It can be used to deploy a Docker image (described in a Dockerfile or Dockerrun.aws.json definition) and source code to EC2 instances running in an Elastic Beanstalk environment.
  - Use the single container configuration when you only need to run one container per instance.

- **Multi container configuration**
  - Multicontainer Docker, uses the Amazon Elastic Container Service (ECS) to coordinate a deployment of multiple Docker containers to an Amazon ECS cluster in an Elastic Beanstalk environment.
  - The instances in the environment each run the same set of containers, which are defined in a Dockerrun.aws.json file.
  - Use the multicontainer configuration when you need to deploy multiple Docker containers to each instance.

AWS Elastic BeanStalk

Integration with other AWS Services

Elastic BeanStalk with S3 and EFS

S3

- Elastic Beanstalk creates an Amazon S3 bucket named elasticbeanstalk-*region-account-id* for each region in which you create environments.
- Elastic Beanstalk uses this bucket to store objects required for the proper operation of your application.
- Elastic Beanstalk doesn't turn on default encryption for the Amazon S3 bucket that it creates. This means that by default, objects are stored unencrypted in the bucket
- You can request Elastic Beanstalk to retrieve instance log files (tail or bundle logs) and store them in Amazon S3.
  - You can also enable log rotation and configure your environment to publish logs automatically to Amazon S3 after they are rotated.
- Application versions, Custom Platform related data, Source bundles, are also saved in this S3 bucket

EFS

- In an EB environment, you can use Amazon EFS to create a shared directory that stores files uploaded or modified by users of your application.
- Your application can treat a mounted Amazon EFS volume like local storage, so you don't have to change your application code to scale up to multiple instances.

Elastic BeanStalk and RDS

- AWS Elastic Beanstalk provides support to run RDS instances in your EB environment
- Including the RDS instances within the EB environment is not a great idea for production, since it couples the RDS instance lifecycle to the EB environment lifecycle.
  - However this would be ok if in test or dev environments.
- To decouple your database instance from your environment, you can run a database instance in Amazon RDS and configure your application to connect to it on launch.
  - This would also enable you to:
    - Connect multiple environments to a database,
    - Terminate an environment without affecting the database, and
    - Perform seamless updates with blue-green deployments.

Elastic BeanStalk and DynamoDB

- You can use configuration files to create a DynamoDB table for your application.
- When you create a DynamoDB table using configuration files, the table isn't tied to your environment's lifecycle, and isn't deleted when you terminate your environment.
- To ensure that personal information isn't unnecessarily retained, delete any records that you don't need anymore, or delete the table.

AWS Elastic BeanStalk

Monitoring and Logging

Elastic BeanStalk and CloudWatch

- Elastic Beanstalk automatically uses Amazon CloudWatch to help you monitor your application and environment status.
- You can define custom metrics for your own use, and Elastic Beanstalk will push those metrics to Amazon CloudWatch.
- Amazon CloudWatch alarms help you implement decisions more easily by enabling you to send notifications or automatically make changes to the resources you are monitoring, based on rules that you define.

Elastic BeanStalk and CloudWatch Logs

- The **CloudWatch Logs agent** installed on each Amazon EC2 instance in your environment publishes metric data points [custom metrics] to the CloudWatch service for each log group you configure.
- Each log group applies its own filter patterns to determine what log stream events to send to CloudWatch as data points.
  - Log streams that belong to the same log group share the same retention, monitoring, and access control settings.

EB and CloudWatch Logs – Exporting Logs to S3

- You can export log data from your log groups to an Amazon S3 bucket and use this data in custom processing and analysis, or to load onto other systems.
- You can store the exported files in your Amazon S3 bucket and define Amazon S3 lifecycle rules to archive or delete exported files automatically.
- You can export logs from multiple log groups or multiple time ranges to the same S3 bucket.
  - To separate log data for each export task, you can specify a prefix that will be used as the Amazon S3 key prefix for all exported objects.

Elastic BeanStalk – Docker Platform Configurations

Logging with Docker

- Specify the directory inside the container to which your application writes logs.
- If you rotate logs to a folder named rotated within this directory, you can also configure Elastic Beanstalk to upload rotated logs to Amazon S3 for permanent storage.
- Elastic Beanstalk uploads any logs in this directory to Amazon S3 when you request tail or bundle logs.

Elastic BeanStalk and CloudTrail

- Elastic Beanstalk is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Elastic Beanstalk.

- Using the information collected by CloudTrail, you can determine the request that was made to Elastic Beanstalk, the IP address from which the request was made, who made the request, when it was made, and additional details.

Elastic BeanStalk – Patterns

- Quickly deploy an application prototype for testing
- No or minimal knowledge about AWS infrastructure when deploying an application to AWS
- Migrating web application that are running on custom application servers on premise to AWS
  - You need to create a dockerfile for the application, include all its components/dependencies
  - Create a docker image off of the dockerfile
  - Upload the image to a public or private Docker image Repository
  - Deploy in AWS with Elastic Beanstalk
- A software development project where app developers could be working on different compute, with different operating systems
  - Abstract the application code from the underlaying compute/OS by creating an identical app environment by creating docker image of the application and send it to all developers

Elastic BeanStalk Anti- Patterns

- If you already know the AWS resources you want to use and how they work, you might prefer defining a template for creating AWS resources with AWS CloudFormation.
- If complete control over the resource configurations is a requirement.
  - Apps with longer lifecycle and intensive configurations might be better served with Cloudformation or OpsWorks

AMAZON CLOUDWATCH

AWS Monitoring Services

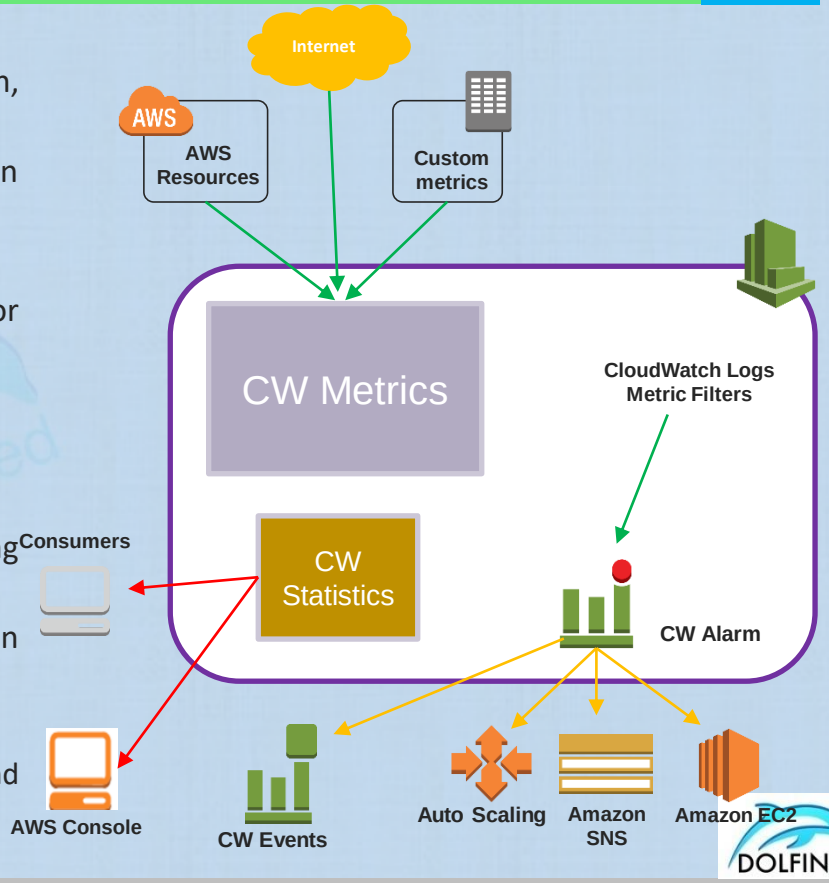
CloudWatch

CloudWatch

- Amazon CloudWatch:
  - Provides system-wide visibility into resource utilization, application performance, and operational health.
  - Monitors AWS resources and any applications that is run on AWS in real time
  - Can be used to collect and track metrics
  - Can monitor the built-in metrics that come with AWS or your custom metrics.

- Amazon CloudWatch is a metrics repository
  - Input to this repository can be:
    - Out of the box metrics from the AWS services using CW
    - Custom metrics based on custom data that you can decide

- VPC resources can connect to CloudWatch by creating and CloudWatch VPC Interface Endpoint



CloudWatch Concepts - Metrics

- Metrics are the fundamental concept in CloudWatch.
  - A metric represents a **time-ordered set of data points** that are published to CloudWatch.
    - Think of a metric as a variable to monitor,
    - The data points represent the values of that variable over time.
      - CPU usage of an EC2 instance is one metric provided by Amazon EC2.

- Data points themselves can come from any application or business activity from which you collect data.
- You can retrieve statistics about those data points as an ordered set of time-series data.

- AWS services send metrics to CloudWatch
  - You can send your own custom metrics to CloudWatch.

- Metrics exist only in the region in which they are created, in other words,
  - **Metrics are completely separate between regions.**

- Metrics are uniquely defined by a name, a namespace, and zero or more dimensions.

CloudWatch Concepts – Metric Retention

- Metrics cannot be deleted, but they automatically expire after 15 months if no new data is published to them.

- CloudWatch retains metric data as follows:
  - Data points with a period of less than 60 seconds are available for 3 hours. These data points are high-resolution custom metrics.
  - Data points with a period of 60 seconds (1 minute) are available for 15 days
  - Data points with a period of 300 seconds (5 minute) are available for 63 days
  - Data points with a period of 3600 seconds (1 hour) are available for 455 days (15 months)
    - Data points older than 15 months expire on a rolling basis; as new data points come in, data older than 15 months is dropped.
  - Data points that are initially published with a shorter period are aggregated together for long-term storage.

AWS CloudWatch

CloudWatch Alarms

CloudWatch Alarms

- A CloudWatch alarm can be created to watch a **single CloudWatch metric or the result of a math expression** based on CloudWatch metrics.

- Alarms can be created or can be based on CloudWatch Logs metric filters

- The alarm can perform one or more actions based on the value of the metric or expression relative to a threshold over a number of time periods (the duration of time over which the alarm is evaluated).

- The possible alarm states:
- **OK –** The metric or expression is within the defined threshold.
- **ALARM –** The metric or expression is outside of the defined threshold (Alarm is triggered).
- **INSUFFICIENT_DATA –** The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

- Alarms can be added to CloudWatch dashboards and monitor them visually (They turn RED when in ALARM state).

CloudWatch Alarms – Actions and Targets

- CloudWatch Alarms can do Auto Scaling, EC2, or SNS actions only.
- CloudWatch Alarms can NOT invoke Lambda functions directly
- The action can be an Amazon
  - EC2 action (Recover, Start, Reboot, or Terminate an EC2 instance),
  - Trigger an Amazon EC2 Auto Scaling action, or
  - Send a notification to an Amazon SNS topic.
- CloudWatch Alarms invoke actions for sustained state changes only, they do not invoke actions because they are in a particular state
  - The state must have changed and been maintained for a specified number of periods.
- CloudWatch doesn't test or validate the actions that you specify, nor does it detect any Amazon EC2 Auto Scaling or Amazon SNS errors resulting from an attempt to invoke nonexistent actions.

AWS CloudWatch

Custom Metrics

CloudWatch – Custom Metrics

- You can publish your own metrics to CloudWatch using **the AWS CLI or an API.**
- You can view statistical graphs of your published metrics with the AWS Management Console.
- CloudWatch stores data about a metric as a series of data points.
  - Each data point has an associated time stamp.
  - You can publish an aggregated set of data points called a statistic set

- Each metric is one of the following:
  - Standard resolution, with data having a one-minute granularity
  - High resolution, with data at a granularity of one second
- Metrics produced by AWS services are standard resolution by default.
- When you publish a custom metric, you can define it as either standard resolution or high resolution.
- When you publish a high-resolution metric, CloudWatch stores it with a resolution of 1 second, and you can read and retrieve it with a period of 1 second, 5 seconds, 10 seconds, 30 seconds, or any multiple of 60 seconds.

AWS Logging Service

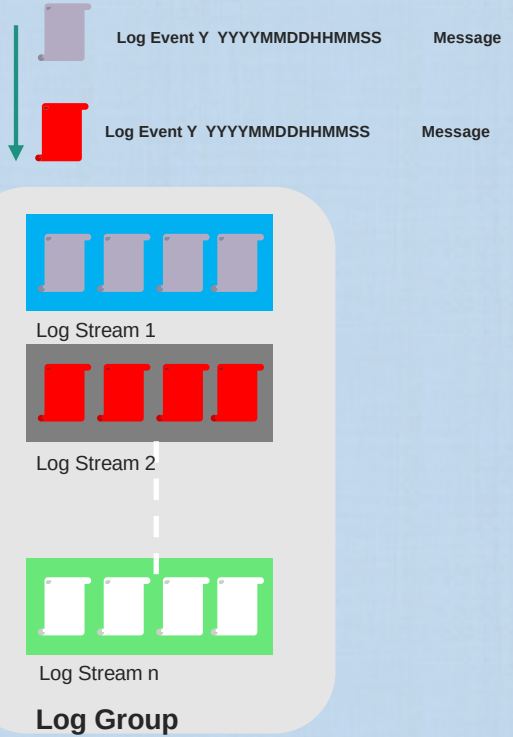
CloudWatch Logs

CloudWatch Logs

- Amazon CloudWatch Logs service can be used to monitor, store, and access log files from many **AWS and Non-AWS sources.**
  - CloudWatch Logs **uses your existing log data for monitoring**; so, no code changes are required.
- CloudWatch Logs enables you to centralize the logs from all systems, applications, and AWS services that are in use, in a single, highly scalable logging service.
- This makes it easy to:
  - View the logs, regardless of the source as a single and consistent flow of time ordered events.
  - Sort and query/search the logs for specific error codes or patterns,
  - Filter them based on specific fields, or
  - Archive them securely for future analysis.
- You can also create custom/powerful queries (Using CloudWatch Insights) and visualize the log data in dashboards.
- Log data Sources:
  - Amazon EC2/ECS instances and (Requires a CloudWatch Agent)
  - Logs from **On Premise Servers** (Requires CloudWatch agent)
  - AWS CloudTrail, Route 53 DNS queries logs, RDS Aurora, MySQL and MariaDB, Amazon Neptune
  - VPC Flow Logs, Elastic Bean Stalk for EC2 instances in the EB environment, API Gateway Execution logging
  - Lambda functions logs and other sources.

CloudWatch Logs - Concepts

- **Log Events**
  - A log event is a record of some activity recorded **by the application or resource being monitored.**
- **Log Streams**
  - A log stream is a sequence of log events that **share the same source** (e.g. Apache access log on a host).
  - AWS deletes empty log streams that are more than 2 months old, or you can delete them manually
- **Log Groups**
  - Log groups define groups of log streams **that share the same retention, monitoring, and access control settings.**
  - Each *log stream has to belong to one log group*.
  - There is no limit on the number of log streams that can belong to one log group.



CloudWatch Logs - Concepts

- **Log Retention**
  - By default, logs are kept indefinitely and never expire (Logs data incurs charges)
  - The retention policy's retention settings can be adjusted for each log group **(at the Log group level),** by choosing retention periods between 10 years and one day.
  - Expired log events get deleted automatically.
  - The retention settings assigned to a log group gets applied to its log streams.

- **Metric Filters**
  - You can use filters to configure **custom** CloudWatch metrics from CloudWatch logs
    - Metric filters are used to extract metric observations from ingested Log events and transform them to data points in a CloudWatch metric.
  - Metric filters are **assigned to log groups**, and all of the filters assigned to a log group are applied to their log streams.
    - When the term the filter is configured to search for is found, **CloudWatch Logs** reports the data to the specified **CloudWatch metric**.
  - CloudWatch Logs sends metrics to CloudWatch every minute.

Encrypting Log data in CloudWatch Logs using KMS

Encryption in Transit

- CloudWatch Logs uses end-to-end encryption of data in transit.
  - CloudWatch Logs service manages the server-side encryption keys.

Encryption at Rest

- CloudWatch Logs protects data at rest using encryption.
  - CloudWatch Logs service manages the server-side encryption keys.
  - You can encrypt the log data in CloudWatch Logs using an AWS KMS customer master key (CMK).
  - Encryption is enabled **at the log group level**, by associating a CMK with a log group,
    - This can be done either when you create the log group or after it exists.

- After you associate a CMK with a log group
  - All **newly** ingested data for the log group is encrypted using the CMK.
  - This data is stored in encrypted format throughout its retention period.
  - CloudWatch Logs decrypts this data whenever it is requested.
  - CloudWatch Logs must have permissions for the CMK whenever encrypted data is requested.

AWS CloudWatch Logs

- CloudWatch Logs Log Insights
- Real time monitoring for EC2
- CloudWatch and CloudTrail
- Exporting CloudWatch logs data to S3/ElasticSearch

CloudWatch Logs Insights

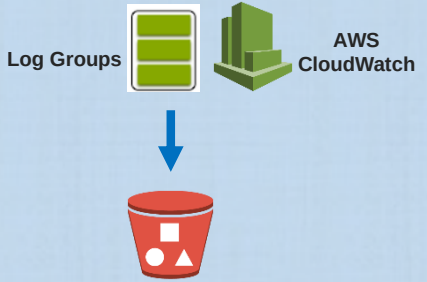
- CloudWatch Logs Insights enables the interactive search and analysis of the log data in Amazon CloudWatch Logs.
  - Queries can be used to help more efficiently and effectively respond to operational issues.
- CloudWatch Logs Insights includes a purpose-built query language with a few simple but powerful commands.
- CloudWatch Logs Insights automatically discovers fields in logs from AWS services such as Amazon Route 53, AWS Lambda, AWS CloudTrail, and Amazon VPC, and any application or custom log that emits log events as JSON.
- CloudWatch Logs Insights can be used to search older log data that was sent to CloudWatch on or after Nov 5th 2018
- A single request can query up to 20 log groups.

CloudWatch Logs - Monitoring AWS CloudTrail Logged Events

- You can create alarms in CloudWatch and receive notifications of particular API activity as captured by CloudTrail.
  - This notification can be used to perform troubleshooting.
  - The way it works is as follows
    - CloudTrail will send the Logs to CloudWatch logs,
    - CloudWatch Logs will have metric filters which will analyze based on criteria the Client sets,
    - Then CloudWatch Logs can send notifications, or a CloudWatch alarm action is triggered based on that.

CloudWatch Logs Storage and Archival – Exporting data to S3/ES

- Log data from your log groups can be exported to an Amazon S3 bucket
  - This data can then be used in custom processing and analysis, or to load onto other systems.
  - The data can also be archived if not required or deleted using Amazon S3 Life cycle rules
  - Log data can take up to 12 hours to become available for export.



- For near real-time analysis of log data,
  - Consider using Amazon CloudWatch Logs insights, or
  - Real-time processing of Log data using subscriptions instead of exporting to S3.

- A CloudWatch Logs log group can be configured **to stream data it receives to an Amazon Elasticsearch Service (Amazon ES) cluster** in **near real-time** through a CloudWatch Logs subscription.

AWS CW Logs

- CloudWatch Agent
- CloudWatch Logs Subscriptions
- Sharing CloudWatch Logs log data

Unified CloudWatch Agent

- To collect logs from your Amazon EC2 instances and on-premises servers into CloudWatch Logs,
  - AWS offers both a new unified CloudWatch agent, and an older CloudWatch Logs agent.
    - AWS recommends the use of the unified CloudWatch agent.
    - The CloudWatch agent needs to be installed on the instance/server to collect Metrics and Logs from it

- The **CloudWatch Logs agent** makes it easy to quickly send both rotated and non-rotated log data off of a host and into the log service.
  - You can then access the raw log data when you need it.

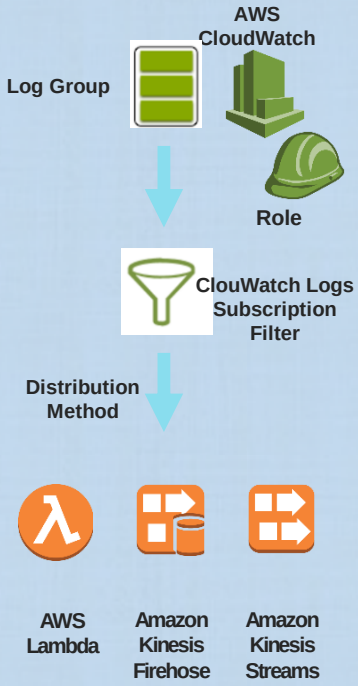
- The new unified agent has the following advantages over the older agent.
  - You can collect both logs and advanced metrics with the installation and configuration of just one agent.
  - The unified agent enables the collection of logs from servers running Windows Server.
  - If you are using the agent to collect CloudWatch metrics, the unified agent also enables the collection of additional system metrics, for in-guest visibility.
  - The unified agent provides better performance.

CloudWatch Logs - Real-time Processing of Log Data with Subscriptions

- Use subscriptions to get access to **a real-time feed of log events** from CloudWatch Logs and have it delivered to other services , for custom processing, analysis, or loading to other systems, such as:
  - **An Amazon Kinesis stream,**
  - **Amazon Kinesis Data Firehose stream, or**
  - **AWS Lambda**

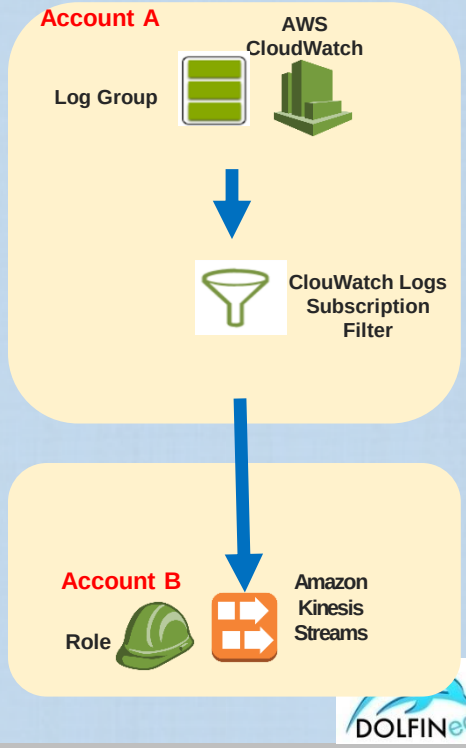
- The subscription filter defines the filter pattern to use for filtering which log events get delivered to your AWS resource, as well as information about where to send matching log events to.

- CloudWatch Logs produces CloudWatch metrics about the forwarding of log events to subscriptions.



CloudWatch Logs - Cross-Account Log Data Sharing with Subscriptions

- You can collaborate with an owner of a different AWS account and receive their log events on your AWS resources,
  - Such as an Amazon Kinesis stream (this is known as cross-account data sharing).
- To achieve this, you need to define a log data sender and receiver accounts
  - Log sender then create subscription filters against their log streams with the destination being the one defined by the receiver.
  - Kinesis streams are currently the only resource supported as a destination for cross-account subscriptions.
- The log group and the destination must be in the same AWS region. However, the AWS resource that the destination points to can be located in a different region.



AWS CloudWatch

CloudWatch Events

CloudWatch Events

- Amazon CloudWatch Events delivers **a near real-time** stream of system events that describe changes in Amazon Web Services (AWS) resources.

- Using a simple rule, Events can be matched and routed to one or more target functions or streams.

- CloudWatch Events becomes aware of operational changes as they occur.
  - CloudWatch Events responds to these operational changes and takes corrective action as necessary,
    - Sending messages to respond to the environment,
    - Activating functions,
    - Making changes, and
    - Capturing state information.

- You can also use CloudWatch Events to schedule automated actions that self-trigger at certain times using cron or rate expressions.

CloudWatch Events Concepts – Events, Targets and Rules

Events

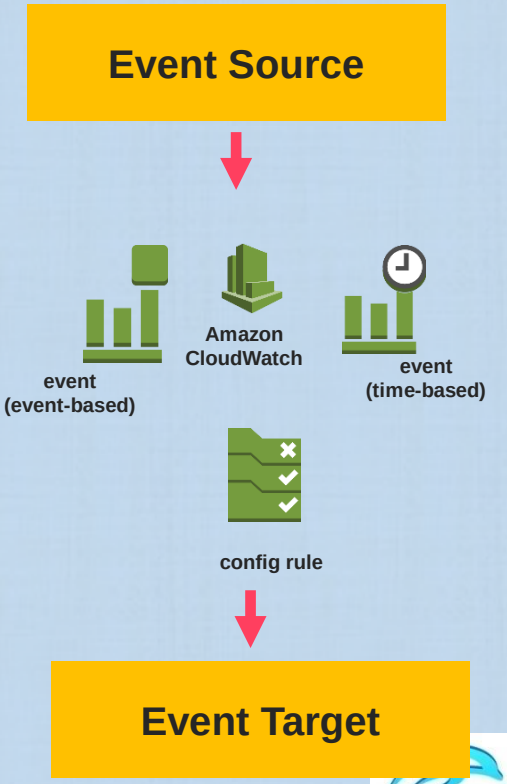
- An event indicates a change in your AWS environment (state change).
- Custom application-level events can also be created and published to CloudWatch Events.
- Scheduled events can be set up that are generated on a periodic basis (Example is creating EBS snapshots).

Targets

- A target receives events in JSON format and processes events.
- Targets include a long list of AWS services and **built-in targets**.

Rules

- A rule matches incoming events and routes them to targets for processing.
- A **single rule can route to multiple targets**, all of which are **processed in parallel**.
- Rules are not processed in a particular order.
  - This enables different parts of an organization to look for and process the events that are of interest to them.
- A rule can customize the JSON sent to the target, by passing only certain parts or by overwriting it with a constant.



CloudWatch Events & Related AWS Services

The following services are used in conjunction with CloudWatch Events:

- **AWS CloudTrail**
  - When CloudTrail logging is turned on, CloudWatch Events writes log files to an S3 bucket.
  - Each log file contains one or more records, depending on how many actions are performed to satisfy a request.
- **AWS CloudFormation**
  - CloudWatch Events rules can be used in your AWS CloudFormation templates.
- **AWS Config**
  - AWS Config rules can be created to check whether your resources are compliant or noncompliant with your organization's policies.
- **AWS Identity and Access Management (IAM)**
  - Use it for authentication and access control to the CloudWatch events
- **Amazon Kinesis Data Streams**
- **AWS Lambda**
  - Use it to build applications that respond quickly to new information.

CloudWatch Events

Sharing CloudWatch Events between Accounts

- An AWS account can be set up to send events to another AWS account, or to receive events from another account.
  - This can be useful if the two accounts belong to the same organization, or belong to organizations that are partners or have a similar relationship.
- You can specify which AWS accounts it sends events to or receives events from.
- Both sender and receiver accounts must be in the same AWS region
- Events sent from one account to another are charged to the sending account as custom events.
  - The receiving account is not charged.

AWS CloudWatch

CloudWatch Synthetics and ServiceLens

CloudWatch Synthetics

- Is a fully-managed service that enables canaries creation to monitor endpoints and APIs in your environment from the outside-in.

- Canaries are configurable scripts that follow the same routes and perform the same actions as a customer.
  - This enables the outside-in view of the customers' experiences, and also the service's availability from their point of view.
  - Canaries check the availability and latency of the endpoints, and can store load time data and screenshots of the UI.

- The canary can run once, or on a regular schedule.
  - Scheduled canaries can run 24 hours a day, as often as once per minute.

- You can have CloudWatch create alarms based on created canaries.
  - If an Alarm for a Canary is created, the alarm goes to ALARM state if the canary test runs fail a certain number of times, depending on how often the canary runs.

CloudWatch ServiceLens

Using ServiceLens to Monitor the Health of the Applications

- CloudWatch ServiceLens enhances the observability of services and applications by integrating traces, metrics, logs, and alarms into one place.
  - This enables investigating problems and their effect on the application.
- ServiceLens **integrates CloudWatch with AWS X-Ray** to provide an end-to-end view of the application to help more efficiently pinpoint performance bottlenecks and identify impacted users.
- A service map displays the service endpoints and resources as “nodes” and highlights the traffic, latency, and errors for each node and its connections.
- A node can be checked to see detailed insights about the correlated metrics, logs, and traces associated with that part of the service.
- ServiceLens supports logs correlation for Lambda functions, API Gateway, Java-based applications running on Amazon EC2, and Java-based applications running on Amazon EKS or Kubernetes with Container Insights deployed.
- ServiceLens integrates with Amazon CloudWatch Synthetics,
  - Created canaries appear on the ServiceLens service map.

AWS CONFIG

AWS Config

Introduction and Benefits

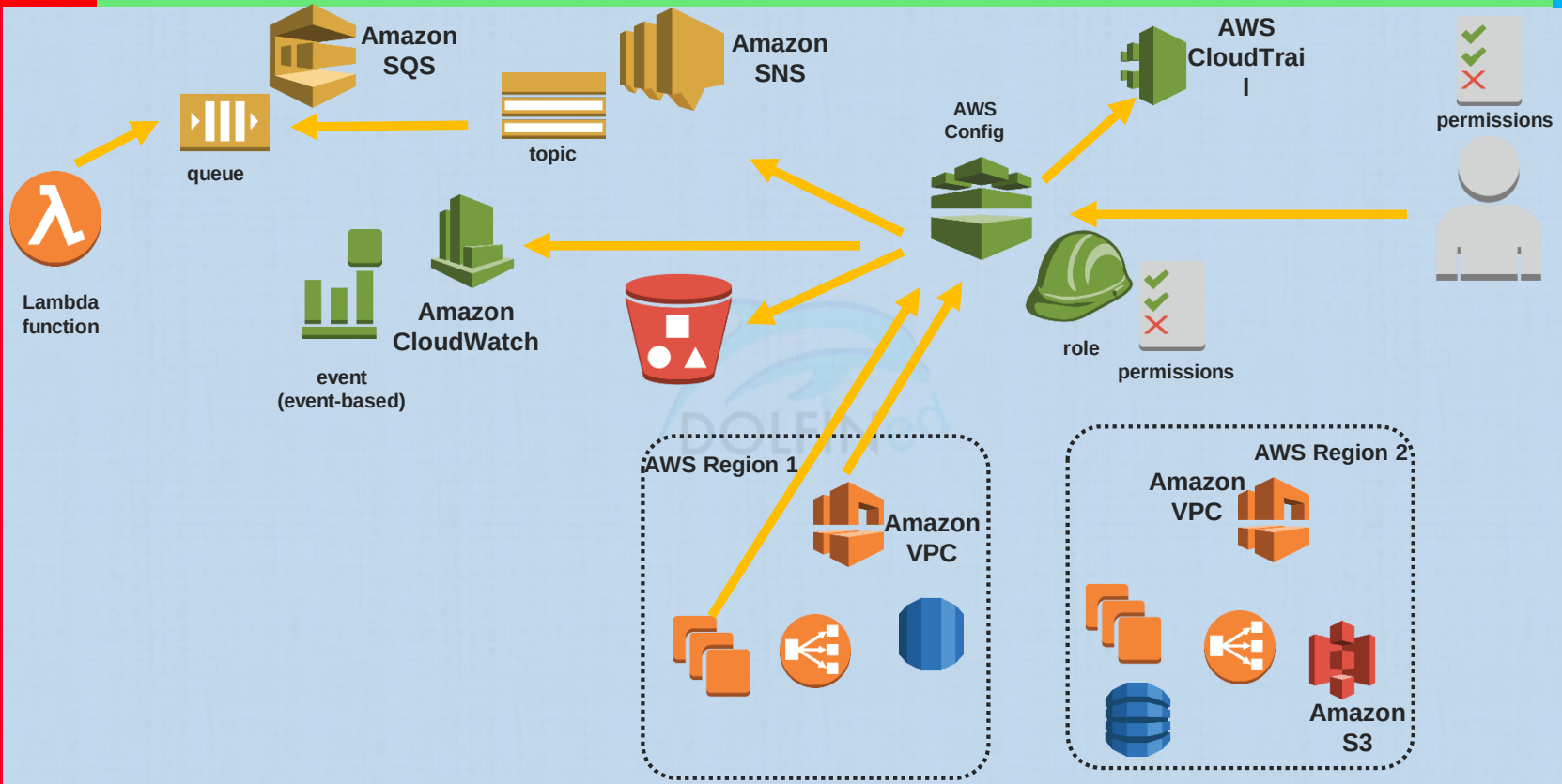
AWS Config – What is it?

- AWS Config provides a detailed view of the configuration of AWS resources in an AWS account.
  - This includes:
    - How the resources are related to one another
    - How they were configured in the past (Configuration and Changes history)

- An AWS resource is an entity you can work with in AWS, such as an Amazon EC2 instance, EBS volumes, Security groups, or a VPC

- A private connection can be established between a VPC and AWS Config. Such that communication between the VPC resources and AWS Config happens over the AWS infrastructure, not the internet.

AWS Config – What is it?



AWS Config – Benefits

AWS Config can help achieve the following:

- Evaluate your AWS resource configurations for desired settings.
- Get a snapshot of the current configurations of the supported resources that are associated with an AWS account.
- Retrieve configurations of one or more resources that exist in an account.
- Retrieve historical configurations of one or more resources.
- Receive a notification whenever a resource is created, modified, or deleted.
- View relationships between resources.

AWS Config Use Cases

As applications and associated infrastructure grows, a mechanism to track the configuration and changes of involved AWS resources becomes a must. AWS Config can help achieve the following:

- **Resource Administration, Auditing, and Compliance**
  - Achieve governance over the configuration and compliance status of your involved resources
  - Get notified whenever a change/deletion/creation of resources happen
  - Evaluate the configuration compliance of your AWS Resources.
  - Perform auditing using the historical configuration data from AWS Config

- **Managing and Troubleshooting Configuration Changes**
  - Using AWS Config, the relation among resources can be found, which minimizes the risk that a change in one resource might impact others
  - Use the historical configurations of the resources provided by AWS Config to troubleshoot issues and to access the last known good configuration of a problem resource.

- **Security Analysis**
  - To expose potential security exposures, historical configuration information can be used (IAM, Sec Groups..etc)
  - Track the history of IAM permissions granted to users or Groups.
  - Evaluate the configuration of security groups when troubleshooting certain connectivity or packet drop issues

AWS Config – Concepts

- **Configuration Items**
  - A configuration item is a point-in-time view of the various attributes of a supported AWS resource that exists in an account.
  - AWS Config creates a configuration item whenever it detects a change to a resource type that it is recording.

- **Configuration History**
  - A configuration history is a collection of the configuration items for a given resource over any time period.
  - AWS Config automatically delivers a configuration history file for each resource type that is being recorded to a specified Amazon S3 bucket
  - AWS Config retains your Configuration Items for the retention period. Minimum is 30 days and a maximum of 7 years.

- **Configuration Recorder**
  - When created, It stores the configurations of the supported resources in an account as configuration items.

AWS Config – Concepts

- **Configuration Snapshot**
  - A configuration snapshot is a collection of the configuration items for the supported resources that exist in an account.
  - The configuration snapshot can be a useful tool for validating your configuration.
  - The configuration snapshot can be delivered to an Amazon S3 bucket that you specify.

- **Configuration Stream**
  - A configuration stream is an **automatically updated list** of all configuration items for the resources that AWS Config is recording.
    - Every time a resource is created, modified, or deleted, AWS Config creates a configuration item and adds to the configuration stream.
    - The configuration stream works by using an Amazon SNS Topic .

- **Resource Relationship**
  - AWS Config discovers AWS resources in your account and then creates a map of relationships between AWS resources.

AWS Config

- AWS Config rules
- SNS Notifications
- Permissions

AWS Config – Delivering Configuration Items

AWS Config can deliver configuration items through one of the following channels:

- **Amazon S3 Bucket**
  - AWS Config tracks changes in the configuration of your AWS resources, and it regularly sends updated configuration details to an Amazon S3 bucket that you specify.
    - For each resource type that AWS Config records, it sends a configuration history file every six hours.
    - If no configuration changes occur, AWS Config does not send a file.
  - AWS Config doesn't modify the lifecycle policies for objects in the S3 bucket.

- **Amazon SNS Topic**
  - AWS Config uses the Amazon SNS topic that you specify to send the notifications.
  - For best results, use Amazon SQS as the notification endpoint for the SNS topic and then process the information in the notification programmatically.

AWS Config – Evaluating Resources with Rules

- Rules represent the ideal or desired configuration settings.
- Create AWS Config rules to evaluate the configuration settings of an account's AWS resources.
- AWS Config has predefined managed rules, also, custom rules can be created.
- AWS Config continuously tracks the configuration changes that occur among your resources, to ensure compliance to the configured rules
  - It checks whether these changes violate any of the conditions in your rules.
  - If a resource violates a rule, AWS Config flags the resource and the rule as noncompliant.
- Each rule is associated with an AWS Lambda function, which contains the evaluation logic for the rule.

AWS Config

- Multi Region, Multi Account Data aggregation
- Querying AWS Config configuration state
- Monitoring

AWS Config – Multi-Account Multi-Region Data Aggregation

- It allows for the aggregation of AWS Config configuration and compliance data from multiple accounts and regions into a single account.
  - It is useful for central IT administrators to monitor compliance for multiple AWS accounts in the enterprise.
- **The source account** is the AWS account from which the AWS Config data will be collected.
  - A source account can be an individual account or an organization in AWS Organizations.
- **The source region** is the AWS region from which AWS Config configuration and compliance data need to be aggregated
- **The aggregator** is a new resource type in AWS Config that collects AWS Config configuration and compliance data from multiple source accounts and regions.
  - It needs to be created in the region where the aggregated data needs to be collected.
- **The aggregator account** is an account where you create an aggregator.
- The Authorization is the permissions the source account owner grants to an aggregator account.
  - Not required if you are aggregating source accounts that are part of AWS Organizations.

Querying the Current Configuration State of AWS Resources

- AWS Config allows for the querying of the current configuration state of AWS resources based on configuration properties.
- Ad hoc, property-based queries against current AWS resource state metadata across all resources that AWS Config supports.
- This advanced query feature provides a single query endpoint and a powerful query language for obtaining current resource state metadata.
- You can use advanced query for:
  - **Inventory management;** for example, to retrieve a list of Amazon EC2 instances of a particular size.
  - **Security and operational intelligence;** for example, to retrieve a list of resources that have a specific configuration property enabled or disabled.
  - **Cost optimization;** for example, to identify a list of Amazon EBS volumes that are not attached to any EC2 instance.

Monitoring AWS Config

Using AWS SQS

- Send AWS Config SNS notifications from one or more topics to an SQS Queue then use code to go through messages for actions in certain events
  - SQS queue needs to be an endpoint (subscribed) to the SNS topic(s)

Using AWS CloudTrail

- AWS Config is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Config.

Using CloudWatch Events:

- Use Amazon CloudWatch Events to detect and react to changes in the status of AWS Config events.
- A rule can be created that runs whenever there is a state transition, or when there is a transition to one or more states that are of interest.
- Amazon CloudWatch Events will then invoke one or more target actions when an event matches the values specified in the rule.
  - An action can then be taken such as send notifications, capture event information, take corrective action, initiate events, or take other actions.

AWS SERVICE CATALOG

AWS Service Catalog

Introuduction

AWS Service Catalog – What is it?

- AWS Service Catalog enables organizations to create and manage catalogs of IT services that are approved for use on AWS.
- AWS Service Catalog allows organizations to centrally manage commonly deployed IT services, and helps organizations achieve consistent governance and meet compliance requirements.
  - These IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application architectures.
- End users can quickly deploy only the approved IT services they need, following the constraints set by the organization.

How it works:

- IT administrators create, manage, and distribute catalogs of approved products to end users, who can then access the products they need in a personalized portal.
  - Administrators can control which users have access to each product to enforce compliance with organizational business policies.
  - Administrators can also setup adopted roles so that End users only require IAM access to AWS Service Catalog in order to deploy approved resources.

AWS Service Catalog - Benefits

- **Standardization**
  - Manage and administer approved assets
  - The result is a standardized landscape for product provisioning for your entire organization.
- **Self-service discovery and launch**
  - Users browse listings of products (services or applications) that they have access to, locate the product that they want to use, and launch it all on their own as a provisioned product.
- **Fine-grain access control**
  - Administrators assemble portfolios of products from their catalog, add constraints and resource tags to be used at provisioning, and then grant access to the portfolio through AWS IAM users and groups.
- **Extensibility and version control**
  - Administrators can add a product to any number of portfolios and restrict it without creating another copy.

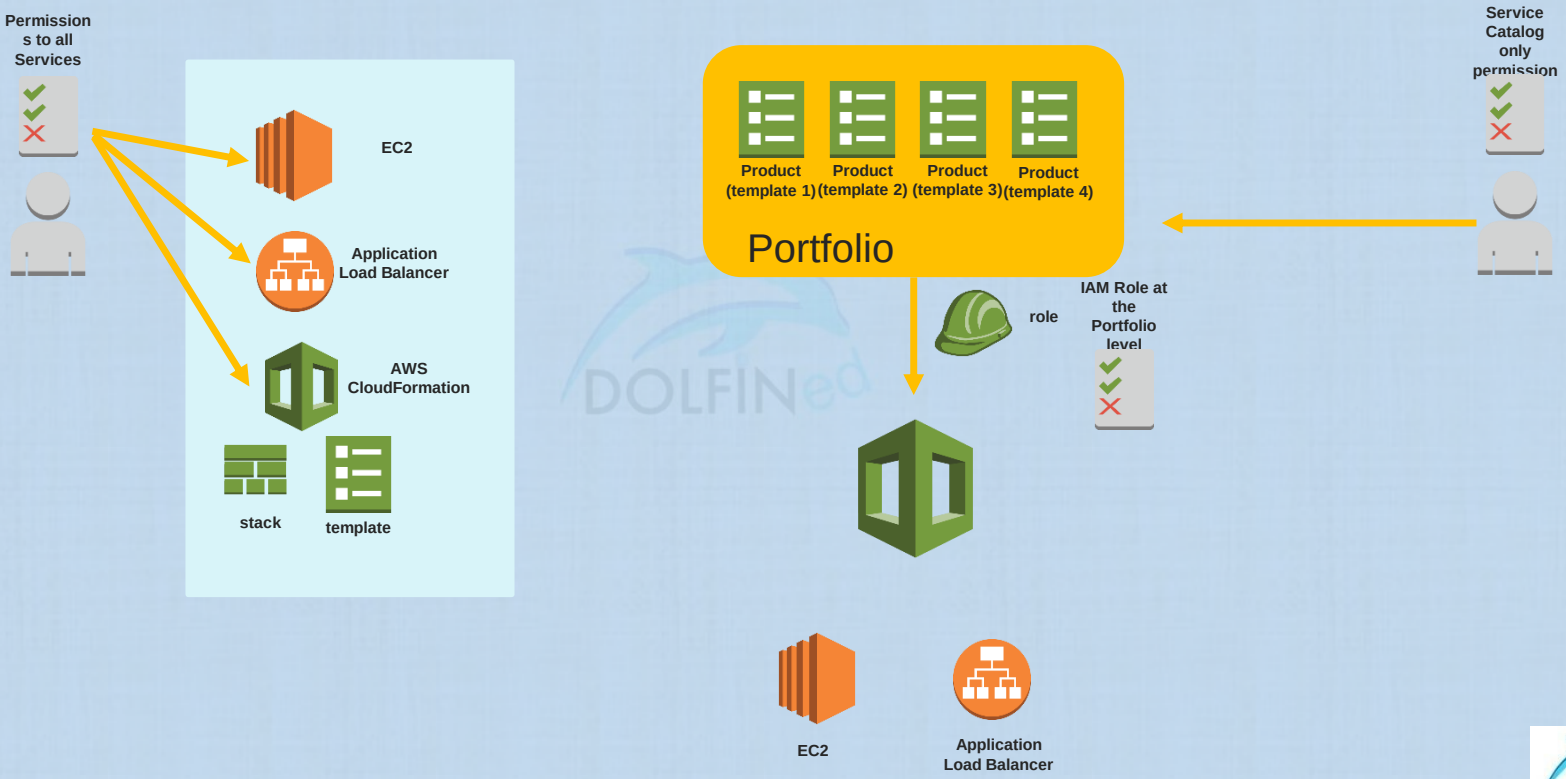
AWS Service Catalog – Who should use it?

- AWS Service Catalog was developed for organizations, IT teams, and managed service providers (MSPs) that need to centralize policies.
- It allows IT administrators to vend and manage AWS resource and services.
- For large organizations, it provides a standard method of provisioning cloud resources for thousands of users.
- It is also suitable for small teams, where front-line development managers can provide and maintain a standard dev/test environment.

AWS Service Catalog

- Products
- Portfolio
- Portfolio Publishing
- Provisioned Stacks
- Versions

AWS Service Catalog



AWS Service Catalog – Products and Portfolios

Products:

- A product is a service or application for end users.
- A catalog is a collection of products that the administrator creates, adds to portfolios, and provides updates for using AWS Service Catalog.
- A product can comprise one or more AWS resources, such as Amazon EC2 instances, Storage Volumes, DBs, Monitoring configurations, or packaged AWS Marketplace products, and networking components.
- Administrators distribute products to end users in portfolios.
- Administrators create catalogs of products by importing AWS CloudFormation templates.

Portfolio:

- A portfolio is a collection of products, with configuration information that determines who can use those products and how they can use them.
- Administrators can create a customized portfolio for each type of user in an organization and selectively grant access to the appropriate portfolio.
- By using portfolios, permissions, sharing, and constraints, administrators can ensure that users are launching products that are configured properly for the organization's needs. policies.

AWS Service Catalog – Publishing a portfolio & Provisioned Stacks

- You publish portfolios that you've created or that have been shared with you to make them available to IAM users in the AWS account.
- To publish a portfolio, you add IAM users, groups, or roles to the portfolio from the AWS Service Catalog console by navigating to the portfolio details page.
- When you add users to a portfolio, they can browse and launch any of the products in the portfolio.
- The same product can be included in multiple portfolios.
  - A single product can be published to multiple portfolios with different access permissions and provisioning policies.

Provisioned Stacks:

- A *provisioned product* is a stack. When an end user launches a product, the instance of the product that is provisioned by AWS Service Catalog is a stack with the resources necessary to run the product.

AWS Service Catalog - Versions

- AWS Service Catalog allows you to manage multiple versions of the products in your catalog.
- When a new product version gets created, the update is automatically distributed to all users who have access to the product, allowing the user to select which version of the product to use.
- When a new version of a product is published to a portfolio, end users can choose to launch the new version.
  - They can also choose to update their running stacks to this new version.
- AWS Service Catalog does not automatically update products that are in use when an update becomes available.

AWS Service Catalog

- Scope and VPC Access
- Permissions
- Constraints
- Sharing a stack with other accounts

AWS Service Catalog – Scope & VPC Access

- AWS Service Catalog is a region specific service
  - You can control the regions in which data is stored.
  - Portfolios and products are regional constructs which will need to be created per region and are only visible/usable on the regions in which they were created.

- AWS Service Catalog APIs can be privately accessed from a VPC by creating VPC Endpoints.

AWS Service Catalog – Permissions and Constraints

Permissions:

- IAM permissions can allow the control of who can view and modify a catalog.
- Granting a user access to a portfolio enables that user to browse the portfolio and launch the products in it.
- When a user launches a product that has an IAM role assigned to it, AWS Service Catalog uses the role to launch the product's cloud resources using AWS CloudFormation.
- By assigning an IAM role to each product, you can avoid giving users permissions to perform unapproved operations and enable them to provision resources using the catalog.

AWS Service Catalog - Constraints

- Rules can be defined to limit the parameter values that a user enters when launching a product.
  - These rules **are called template constraints** because they constrain how the AWS CloudFormation template for the product is deployed.
  - Constraints are applied at the individual products or the portfolio levels.
- AWS Service Catalog applies constraints when provisioning a new product or updating a product that is already in use.
  - It always applies **the most restrictive constraint** among all constraints applied to the portfolio and the product.
- **Launch constraints**, a role for a product in a portfolio can be specified.
  - This role is used to provision the resources at launch, thus user permissions can be restricted without impacting users' ability to provision products from the catalog.
- **Notification constraints** enable you to get notifications about stack events using an Amazon SNS topic.
- **Template constraints** restrict the configuration parameters that are available for the user when launching the product (for example, EC2 instance types or IP address ranges).

AWS Service Catalog – Sharing a portfolio with other accounts

- Portfolios can be shared with users in one or more other AWS accounts.

- When a Portfolio from Account A is shared with other accounts, Account A retains the control and ownership of the portfolio.
  - Only account A Admins can make changes, such as adding new products or updating products.
  - Account A can also remove this sharing at any time.
  - Any products, or stacks, currently in use will continue to run until the stack owner decides to terminate them.

- Administrators also can share portfolios with other AWS accounts and allow the administrators of those accounts to extend the portfolios by applying additional constraints (such as limiting EC2 instance types users can launch).

AWS SYSTEMS MANAGER (SSM)

AWS Systems Manager (SSM)

- What is it?
- SSM Agent
- How it works
- SSM Capabilities
- Managed Instances to SSM Connectivity
- Required Permissions

What is it?

- It is a collection of capabilities to configure and manage Amazon EC2 instances, on-premises servers and virtual machines, and other AWS resources at scale.
- Systems Manager includes a unified interface that allows to easily centralize operational data and automate tasks across a client's AWS resources.
- Systems Manager shortens the time to detect and resolve operational problems in a client's AWS infrastructure.
- Systems Manager gives a complete view of the client's infrastructure performance and configuration, simplifies resource and application management.
- It can be used with Windows, Linux, or RaspBian instance or virtual machines
- AWS Systems Manager allows for remotely and securely manage on-premises servers and virtual machines (VMs) both in Cloud and On premises.
  - You can also manage virtual machines in other cloud environments.

SSM and Hybrid Environments

- Benefits of using AWS Systems Manager with hybrid environments:
  - Creating a consistent and secure way to remotely manage the hybrid workloads from one location using the same tools or scripts.
  - Using IAM for centralized access control for actions that can be performed on the servers and VMs.
  - Using AWS CloudTrail for centralized auditing and view into the actions performed on your servers and VMs.
  - Configuring CloudWatch events and SNS for centralized monitoring to send notifications about service execution success.

How it works? A typical workflow

Configuring Systems Manager:

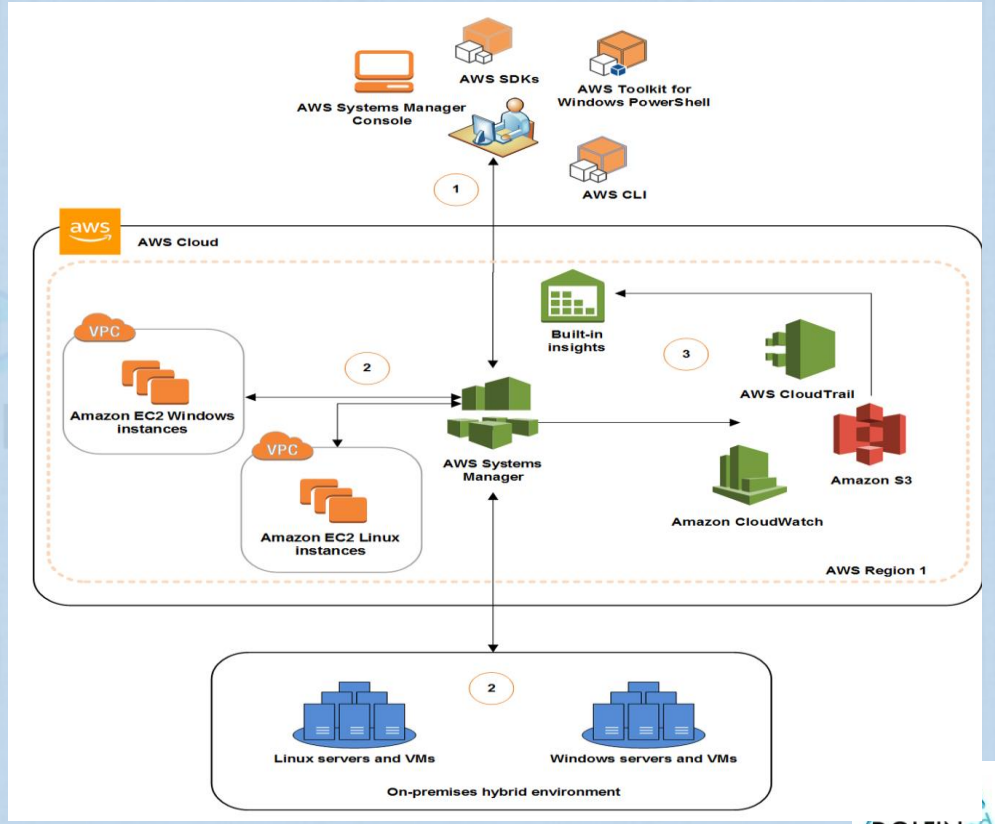
- Systems Manager console, SDK, AWS CLI, or AWS Tools for Windows PowerShell to configure, schedule, automate, and execute Systems Manager actions to perform on your AWS resources.
- Install SSM agent and configure permissions

Verification and processing:

- Systems Manager verifies the configurations, including permissions, and sends requests to the SSM Agent running on the instances or servers in a hybrid environment.
  - SSM Agent performs the specified configuration changes.

Reporting:

- SSM Agent reports the status of the configuration changes and actions to Systems Manager in the AWS cloud.
- Systems Manager then sends the status to the user and various AWS services, if configured.



How it Works? SSM Agent and Managed Instances

- A SSM managed instance is any Amazon EC2 instance or on-premises machine (Server or VM) in a hybrid environment that is configured for Systems Manager.
  - On-premises machines also require an activation code through a managed-instance activation process.
    - This is to provide secure access to the Systems Manager service from the on-premise managed instances.
  - AWS recommends automating the process of keeping SSM Agent up-to-date on the managed instances
- The SSM service operates as a Public Zone service and is accessible through the internet.
- For the managed instances and the Systems Manager service to communicate with each other, one of the following can be used:
  - Configure Systems Manager to use an interface Virtual Private Cloud (VPC) endpoint (Recommended),

  OR
  - Enable outbound internet access on your managed instances
    - Enabling inbound internet access is not required

- SSM to managed instances communications uses TLS

AWS Systems Manager (SSM)

- SSM Capabilities
- SSM Capabilities - Resource Groups
- SSM Capabilities - Insights

SSM Capabilities

- **Resource Groups**
- **Insights**
  - Built-in Insights
  - AWS CloudWatch dashboards
  - Inventory Management
  - Configuration Compliance
- **Actions**
  - Automation
  - Run Command
  - Session Manager
  - Distributor
  - Patch Manager
  - Maintenance Window
  - State Manager
- **Shared Resources**
  - Managed Instances
  - Activations
  - Systems Manager Documents
  - Parameter Store

SSM Capabilities - Resource Groups

- Resource groups make it easier to manage and automate tasks on large numbers of resources at one time.
  - An AWS resource can be an Amazon EC2 instance, an Amazon EBS volume, a security group, or an Amazon VPC.

- A resource group is a collection of AWS resources that **are all in the same AWS Region**, and that match criteria provided in a query.

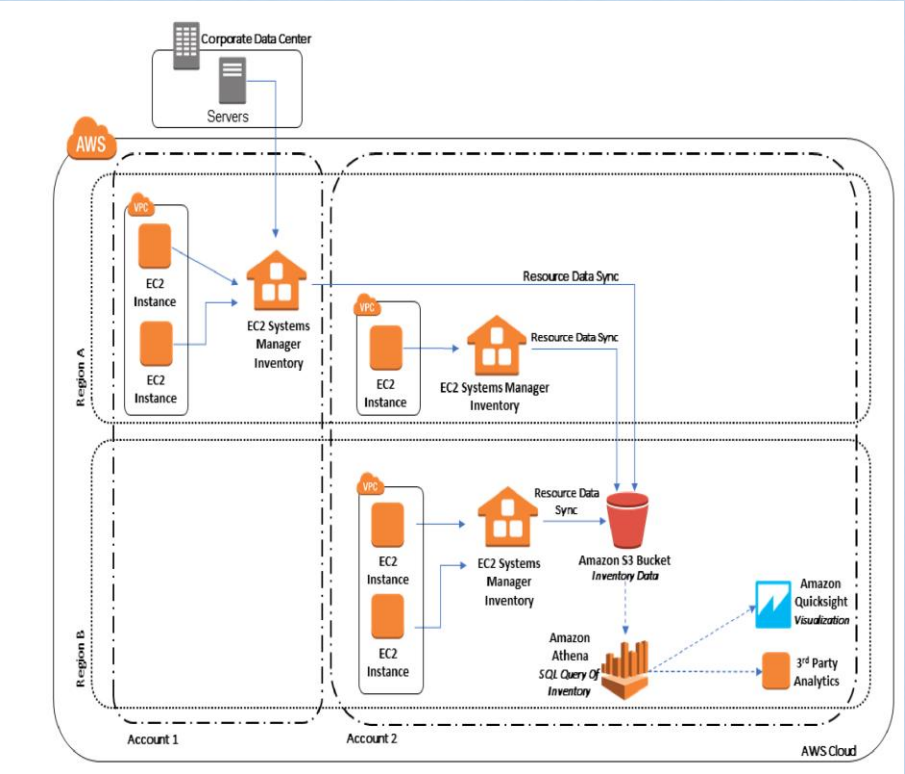
- Resource groups can also be the basis for viewing monitoring and configuration insights in AWS Systems Manager.

SSM Capabilities - Insights

- Built-in Insights
  - Insights show detailed information about the resources in the respective AWS Resource Groups, such as:
    - AWS CloudTrail logs,
    - Results of evaluations against AWS Config rules, and
    - AWS Trusted Advisor reports.
- Insights show information about a single, selected resource group at a time.
- Other means to view data include:
  - CloudWatch Dashboards
  - **Inventory Manager**

AWS Systems Manager Inventory

- AWS Systems Manager Inventory provides visibility into your Amazon EC2 and on-premises computing environment.
- Inventory can be used to collect metadata from your managed instances.
- This metadata can be stored in a central Amazon S3 bucket, and then use built-in tools to query the data and quickly determine which instances are running the software and configurations required by your software policy, and which instances need to be updated.
- Using Systems Manager Resource Data Sync, a one-time operation can be performed that synchronizes all Inventory data from all of your managed instances.
- Inventory data from multiple AWS Regions and accounts can be configured and viewed (Using Resource Data Sync which can be done from console or CloudFormation)
- Systems Manager Inventory stores Inventory data for 30 days.



Source:aws.amazon.com

Configuration Compliance

- You can use AWS Systems Manager Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies.

- You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren't compliant.

- Systems Manager Compliance offers the following additional benefits and features:
  - View compliance history and change tracking for Patch Manager patching data and State Manager associations by using **AWS Config**.
  - Remediate issues by using Systems Manager Run Command, State Manager, or Amazon CloudWatch Events.
  - Port data to **Amazon Athena and Amazon QuickSight** to generate fleet-wide reports.

- The service is customizable, and you can create custom compliance types based on your IT or business requirements.

AWS Systems Manager (SSM)

- SSM Capabilities – Actions - Automation
- SSM Capabilities – Actions – RUN command

SSM Capabilities – Actions - Automation

- Automation Actions refer to what SSM can do to/on the managed instances at scale

- Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon EC2 instances and other AWS resources. Automation enables you to do the following.
  - Build Automation workflows to configure and manage instances and AWS resources.
  - Create custom workflows or use pre-defined workflows maintained by AWS.
  - Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events.
  - Monitor Automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.

- Example common maintenance and deployment tasks that can be automated:
  - Create and update Amazon Machine Images,
  - Apply driver and agent updates,
  - Reset passwords on Windows instance,
  - Reset SSH keys on Linux instances, and
  - Apply OS patches or application updates.

SSM Capabilities – Actions (cont.) – RUN Command

- Is used to remotely and securely manage the configuration of the managed instances at scale
  - Run Command is used to automate common administrative tasks and perform ad hoc configuration changes

- Run Command can be used to perform on-demand changes like
  - Updating applications or
  - Running **Linux** shell scripts and **Windows** PowerShell (Run Command and the **AWS-RunShellScript** document) or Using Run Command and the **AWS-RunPowerShell** document for windows instances
  - Install or bootstrap applications,
  - Build a deployment pipeline,
  - Capture log files when an instance is terminated from an Auto Scaling group, and
  - Join instances to a Windows domain.

- **If CloudWatch logs is specified**
  - Run Command periodically sends all command output and error logs to CloudWatch Logs.
- **CloudWatch Events**
  - Amazon CloudWatch Events can be used to log command execution status changes.
  - A Run Command can be specified as a target action when a CloudWatch event occurs.

AWS Systems Manager (SSM)

- SSM - State Manager
- SSM - Session Manager
- SSM - Distributor
- SSM - Patch Manager
- SSM – Maintenance Window

SSM Capabilities – Actions – State Manager

- AWS Systems Manager State Manager is a secure and scalable configuration management service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define.
  - Can also be used to ensure that your managed instance are configured with specific applications, such as anti-virus or malware applications
  - Can be used to automate the process of updating the SSM Agent or other AWS packages or ensure that specific ports are closed or open.

- It can be used to ensure that:
  - Instances are patched with specific software updates.
  - Bootstrap instances with specific software at start-up
  - Download and update agents on a defined schedule, including SSM Agent
  - Configure network settings
  - Join instances to a Windows domain (Windows instances only)
  - Patch instances with software updates throughout their lifecycle
  - **Run scripts on Linux and Windows managed instances throughout their lifecycle**
    - Can set a a schedule for when these scripts are applied (every Tuesday at 3AM, or every Monday at 5pm for example).

State Manager – Use Cases

- State Manager integrates with AWS CloudTrail to provide a record of all executions that you can audit, and Amazon CloudWatch Events to track state changes.
  - You can also choose to store and view detailed command output in Amazon S3.

- A State Manager association is a configuration that is assigned to your managed instances.
  - The association specifies **a schedule for when the configuration is reapplied (every Tuesday at 3AM, or every Monday at 5pm for example)**.
  - The association also specifies actions to take when applying the configuration.

  - Example: a state manager association for anti-virus software is run once a day.
    - If the software is not installed, then State Manager installs it.
    - If the software is installed, but the service is not running, then the association might instruct State Manager to start the service.

SSM Capabilities – Actions (cont.) – Session Manager

- Session Manager is a fully managed AWS Systems Manager capability that lets you manage your Amazon EC2 instances through an **interactive one-click browser-based shell or through the AWS CLI**.
  - Session Manager provides secure and auditable instance management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys.

- Can be used to establish secure connections to both Windows and Linux Instances

Session Manager integrates with the following AWS services to provide logging and auditing capabilities:

- AWS CloudTrail
- Amazon S3
- Amazon CloudWatch Logs
- Amazon CloudWatch Events and Amazon Simple Notification Service

- Use case:
  - An on-call engineer in your IT department receives notification, at 2AM, of an issue that requires him to remotely connect to an instance to troubleshoot/fix.
  - Using the AWS Systems Manager console or the AWS CLI, he can start a session connecting him to the instance, runs commands on the instance needed to complete the task, and then terminates the session.

SSM Capabilities – Actions (cont.) – Patch Manager

- Used to automate the process of scanning managed instances for missing patching, and then updated their **OS security related patches.** patching the SSM managed instances.

- It can be used to apply any missing patches to instances **individually, or to a large groups** of instances by using Amazon EC2 instance tags.
  - You can patch fleets of Amazon EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type.

- Security patching can be automated on a regular basis by scheduling patching to run as a Systems Manager Maintenance Window task.

SSM Capabilities – Actions (cont.) - Distributor

- Used to create and deploy/publish software packages to SSM managed instances.
- Clients can package own software packages or find AWS-provided agent software packages.
  - Ex. Use it to install AmazonCloudWatchAgent on managed instances
- Automate deployment
  - To keep your environment current, use **State Manager to schedule packages for automatic deployment** on target instances when those instances are first launched.

AWS Systems Manager (SSM)

- SSM Shared Resources
- SSM Use Cases
- Pricing

SSM Capabilities – Shared Resources

The following shared resources are used by Systems Manager for managing and configuring your AWS resources.

- **Managed Instances**
  - AWS Systems Manager allows for Standard and Advanced instances
  - Advanced instances enable you to connect to your hybrid machines by using AWS Systems Manager Session Manager.
    - Session Manager provides interactive shell access to your instances.

- **Systems Manager (SSM) document**
  - It defines the actions that Systems Manager performs on your managed instances
  - SSM Documents include steps and parameters that you specify.
  - SSM documents are JSON or YAML documents
  - SSM document types include:
    - Command documents, which are used by SSM and Run Command, and
    - Automation documents, which are used by SSM Automation.
  - Systems Manager includes dozens of pre-configured documents that can be used by specifying parameters at runtime.

- **Parameter Store**

AWS Inventory and Other AWS Services

- Systems Manager Inventory provides a snapshot of your current inventory to help you manage software policy and improve the security posture of your entire fleet.

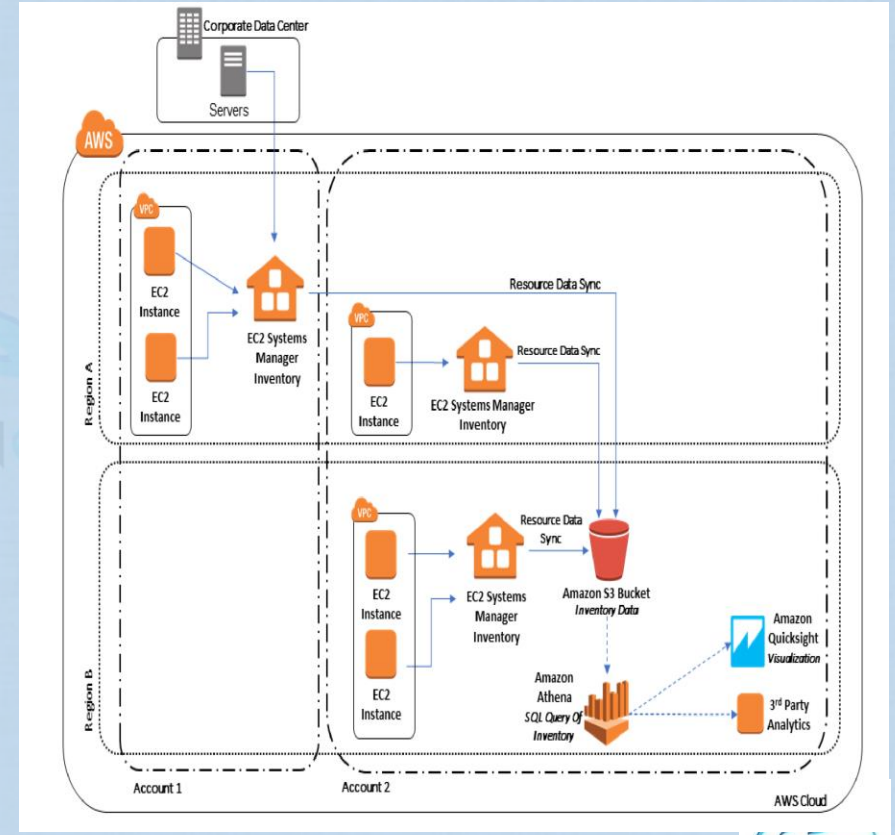
- **Systems Manager's Inventory management and migration capabilities can be extended using the following AWS services.**
  - **AWS Config** provides a historical record of changes to the inventory, along with the ability to create rules to generate notifications when a configuration item is changed.
  - **AWS Application Discovery Service** is designed to collect inventory on OS type, application inventory, processes, connections, and server performance metrics from your on-premises VMs to support a successful migration to AWS.
  - You can use **Systems Manager Resource Data Sync** to send Inventory data collected from all of your managed instances to a single **Amazon S3 bucket**.
    - **AWS KMS** can be used to encrypt Inventory data in the Amazon S3 bucket.
    - Resource Data Sync then automatically updates the centralized data when new Inventory data is collected.
    - With all Inventory data stored in a target Amazon S3 bucket, you can use services like **Amazon Athena** and **Amazon QuickSight** to query and analyze the aggregated data.

Use Case

- You client has 150 instances distributed among multiple AWS Regions and on premises. It is required to configure SSM Inventory to collect data about the configurations, operating system (OS) and applications running on them. The gathered data needs to be in one location to analyze it, find configuration inconsistencies/compliance, and generate visual reports for the management. Also it is required to ensure the inventory data is stored encrypted at rest.

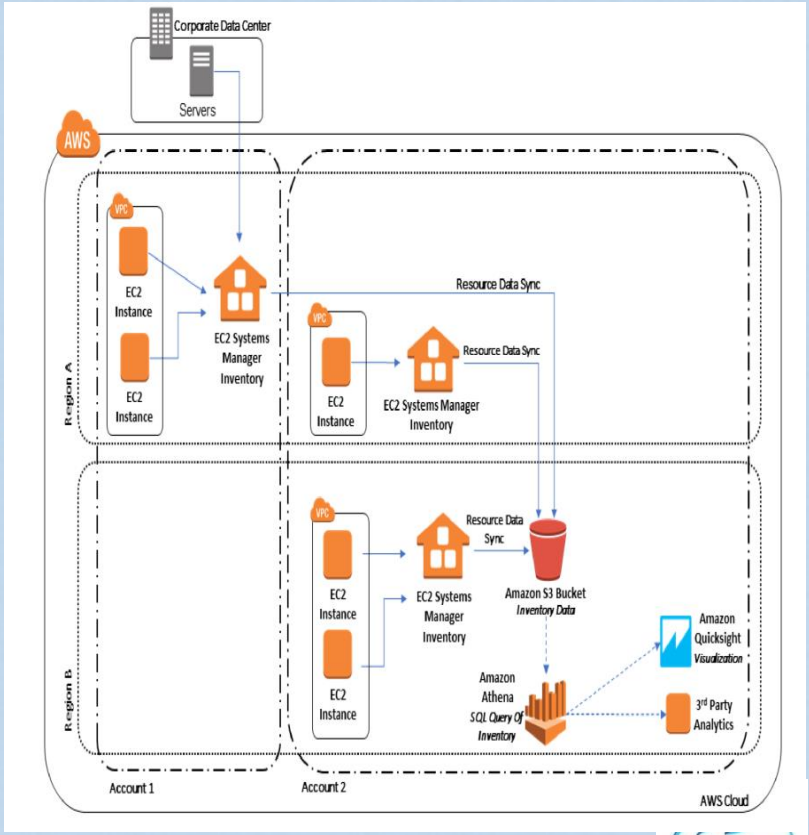
How can you achieve this?

- This can be done in multiple ways:
  - Manually gather the collected inventory data for each instance,
  - Create scripts to gather this information,
  - Then port the data into an application so that you can run queries and analyze it.
- OR, you can configured **Resource Data Sync**,
  - You need to perform a one-time operation that synchronizes all Inventory data from all of your managed instances.



Use Case

- After the sync is successfully created, Systems Manager creates a baseline of all Inventory data and saves it in the target Amazon S3 bucket.
- When new inventory data is collected, Systems Manager automatically updates the data in the Amazon S3 bucket.
- To synchronize inventory data from multiple AWS Accounts and Regions,
  - You must create a Resource Data Sync in each Region.
  - Repeat this procedure in each AWS Region where you want to collect inventory data and send it to the central Amazon S3 bucket.
- You can then quickly and cost-effectively port the data to Amazon Athena and Amazon QuickSight for analysis and visualization.
- AWS Systems Manager Inventory integrates with Amazon Athena to help you query inventory data from multiple AWS Regions and accounts. Athena integration uses Resource Data Sync
- This feature uses AWS Glue to crawl (discover and do ETL processing) the data in your Amazon S3 bucket, and Amazon Athena to query the data.



AMAZON PARAMETER STORE

AWS Parameter Store

Parameter Store

- AWS Systems Manager Parameter Store provides a serverless secure, hierarchical storage for configuration data management and secrets management.
- You can store data such as passwords, database strings, and license codes as parameter values.
- It can be used with On-Premise servers too.
- Highly scalable, available, and durable, Parameter Store is backed by the AWS Cloud.
- You can store values as plain text or encrypted data (using KMS, permissions to the KMS keys will need to be granted to the Instance using Parameter store in this case).

Parameter Store – Benefits and Features

- A secure, scalable, hosted secrets management service (No servers to manage).
- Improve the security posture by separating data from your code.
- Store configuration data and secure strings in hierarchies and track versions.
- Control and audit access at granular levels.
- Configure change notifications and trigger automated actions.
- Tag parameters individually, and then secure access from different levels, including operational, parameter, Amazon EC2 tag, or path levels.
- Reference AWS Secrets Manager secrets by using Parameter Store parameters.
- Use Parameter Store parameters with other Systems Manager capabilities and AWS services to retrieve secrets and configuration data from a central store.
- The following AWS services support Parameter Store parameters: Amazon EC2, Amazon ECS, AWS Lambda, AWS CloudFormation, AWS CodeBuild, and AWS CodeDeploy.
- Configure integration with AWS KMS, Amazon SNS, Amazon CloudWatch, and AWS CloudTrail for encryption, notification, monitoring, and audit capabilities.

Parameter Store

- You can reference Systems Manager parameters in your scripts, commands, SSM documents, and configuration and automation workflows.
- Parameters work with Systems Manager capabilities such as Run Command, State Manager, and Automation.
- You can also reference parameters in other AWS services such as Amazon Elastic Container Service and AWS Lambda.

Parameter Store and CloudWatch Agent config file

- The CloudWatch Agent enables the gathering of more metrics on Amazon EC2 instances than are available using SSM Agent.
  - Metrics from on-premises servers using the CloudWatch Agent can also be gathered.
  - Agent configuration settings can be stored in the Systems Manager Parameter Store for use with the CloudWatch Agent.

- You can store the contents of an Amazon CloudWatch Agent configuration file in Parameter Store.
  - By maintaining this configuration data in a parameter, multiple instances can derive their configuration settings from it,
    - This avoids having to create or manually update configuration files on the instances.
    - For example, Run Command can be used to write the contents of the parameter to configuration files on multiple instances, or use State Manager to help avoid configuration drift in the CloudWatch Agent configuration settings across a fleet of instances.

Parameter Store and AWS Secrets Managers

- Referencing AWS Secrets Manager Secrets from Parameter Store Parameters
- Secrets Manager helps organize and manage important configuration data such as credentials, passwords, and license keys.

- Parameter Store is integrated with Secrets Manager so that Secrets Manager secrets can be retrieved when using other AWS services that already support references to Parameter Store parameters.
  - These services include Amazon EC2, Amazon ECS, AWS Lambda, AWS CloudFormation, AWS CodeBuild, AWS CodeDeploy, and other Systems Manager capabilities.

- You can use AWS Systems Manager with partner and product technologies such as GitHub, Amazon S3, and the Volume Shadow Copy Service (VSS) to automate the deployment, configuration, and maintenance of your managed instances.