Management & Governance

CW, CloudTrail, Organizations, Config, Trusted Advisor, CloudFormation, OpsWorks

AWS Certified Solutions Architect– Professional (SAP-C01)

Study notes - Sep'2019

CW

Are we using CW agent? Which custom alarms are configured? Which CW Events?

CW (1/3) - Overall

https://docs.aws.amazon.com/CW/index.html

CW Monitoring 1
Use CW to monitor your AWS resources and your applications. The CW home page displays metrics about every AWS service you use. You can also create custom dashboards and create alarms which watch metrics and send notifications

CW Events 2
Use CW Events to send system events from AWS resources to AWS Lambda functions, Amazon SNS topics, streams in Amazon Kinesis, and other target types.

CW Logs 3
Use CW Logs to monitor, store, and access your log files from Amazon EC2 instances, AWS CloudTrail, or other sources.

Collecting Metrics and Logs from Amazon EC2 Instances and On-Premises Servers with the CW Agent

- Collect more system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances.
- Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
- Retrieve custom metrics from your applications or services using the StatsD and collectd protocols. StatsD is supported on both Linux servers and servers running Windows Server. collectd is supported only on Linux servers.
- Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.

You can store and view the metrics that you collect with the CW agent in CW just as you can with any other CW metrics.

The logs collected by the unified CW agent are processed and stored in Amazon CW Logs, just like logs collected by the older CW Logs agent. Metrics collected by the CW agent are billed as custom metrics.

CW (2/3) - Logs

https://docs.aws.amazon.com/AmazonCW/latest/logs/WhatIsCWLogs.html

CW Logs enables you to centralize the logs from all of your systems, applications, and AWS services that you use, in a single, highly scalable service. You can then easily **view** them, **search them for specific error codes or patterns**, **filter** them based on specific fields, or **archive** them securely for future analysis. CW Logs enables you to create custom computations with a powerful query language, and visualize log data in dashboards.

- **Monitor Logs from Amazon EC2 Instances**
- **Monitor AWS CloudTrail Logged Events**
- **Log Retention**
- **Archive Log Data**
- **Log Route 53 DNS Queries**

Analyzing Log Data with CW Logs Insights

CW Logs Insights enables you to interactively search and analyze your log data in Amazon CW Logs. You can perform queries to help you quickly and effectively respond to operational issues. If an issue occurs, you can use CW Logs Insights to identify potential causes and validate deployed fixes.

Sending Logs Directly to Amazon S3 - enabled for VPC Flow logs and AWS Global Accelerator flow logs

Some AWS services can publish logs directly to Amazon S3. This way, if your main requirement for logs is storage in Amazon S3, you can easily have the service producing the logs send them directly to Amazon S3 without setting up additional infrastructure.

Exporting Log Data to Amazon S3

You can export log data from your log groups to an Amazon S3 bucket and use this data in custom processing and analysis

Streaming CW Logs Data to Amazon Elasticsearch Service

You can configure a CW Logs log group to stream data to your ES cluster in near real-time through a CW Logs subscription.

Using CW Logs with Interface VPC Endpoints

Establish a private connection between your VPC and Cw Logs to send logs without sending them through the internet.

CW (3/3) - Events

https://docs.aws.amazon.com/AmazonCW/latest/events/WhatIsCWEvents.html

CW Events delivers a near real-time stream of system events that describe changes in resources. Using simple rule setup, you can match events and route them to one or more target functions or streams. CW Events becomes aware of operational changes as they occur. CW Events responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.

Use CW Events to schedule automated actions that self-trigger at certain times using cron or rate expressions.

Concepts

Events – Resources can generate events when their state changes. eg. EC2 -> pending to running. You can generate custom application-level events and publish them to CW Events. You can also set up scheduled events that are generated on a periodic basis.

Targets – A target processes events. Targets can include EC2 instances, Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, and built-in targets.

Rules – A rule matches incoming events and routes them to targets for processing.

Schedule Expressions for Rules

You can create rules that self-trigger on an automated schedule in CW Events using cron or rate expressions.
Rate expressions are simpler to define but don't offer the fine-grained schedule control that cron expressions support.

Tutorial: Use CW Events to Relay Events to AWS Systems Manager Run Command
Tutorial: Log the State of an Amazon EC2 Instance Using CW Events
Tutorial: Log Amazon S3 Object-Level Operations Using CW Events
Tutorial: Schedule AWS Lambda Functions Using CW Events
Tutorial: Set AWS Systems Manager Automation as a CW Events Target
Tutorial: Run an Amazon ECS Task When a File is Uploaded to an Amazon S3 Bucket

Real-time Processing of Log Data with Subscriptions

https://docs.aws.amazon.com/AmazonCW/latest/logs/Subscriptions.html

You can use subscriptions to get access to a real-time feed of log events from CW Logs and have it delivered to other services such as a

- Amazon Kinesis stream,
- Amazon Kinesis Data Firehose stream, or
- AWS Lambda for custom processing, analysis, or loading to other systems.

To begin subscribing to log events, create the receiving source, such as a Kinesis stream, where the events will be delivered. A 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.

CW Logs also produces CW metrics about the forwarding of log events to subscriptions.

Searching and Filtering Log Data

https://docs.aws.amazon.com/AmazonCW/latest/logs/MonitoringLogData.html

After the CW Logs agent begins publishing log data to CW, you can begin searching and filtering the log data by creating one or more **metric filters**. Metric filters define the terms and patterns to look for in log data as it is sent to CW Logs. CW Logs uses these metric filters to turn log data into numerical CW metrics that you can graph or set an alarm on. You can use any type of CW statistic, including percentile statistics, when viewing these metrics or setting alarms.

- Example: Count Log Events
- Example: Count Occurrences of a Term
- Example: Count HTTP 404 Codes
- Example: Count HTTP 4xx Codes
- Example: Extract Fields from an Apache Log

https://console.aws.amazon.com/cloudwatch/ → Logs → Create Metric Filter → Define Logs Metric Filter → Filter Pattern **[IP, UserInfo, User, Timestamp, RequestInfo, StatusCode=404, Bytes]** → Select Log Data to Test → Test Pattern

AWS Organizations

AWS Organizations - Features

https://docs.aws.amazon.com/organizations/latest/userguide/orgs_getting-started_concepts.html

AWS Organizations is an **account management service** that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

Features:

Centralized management of all of your AWS accounts

Consolidated billing for all member accounts
- Master account pays for all member accounts

Hierarchical grouping of your accounts to meet your budgetary, security, or compliance needs
- Group into OUs
- Attach policies to OUs that restrict access to services
- You can nest OUs upto 5 levels

Control over the AWS services and API actions that each account can access
- Use Service Control Policies (SCP) to set permissions
- Possible to restrict which AWS services, resources, and individual API actions the users and roles in each member account can access.
- These restrictions override the IAM policies set by administrators of the member accounts

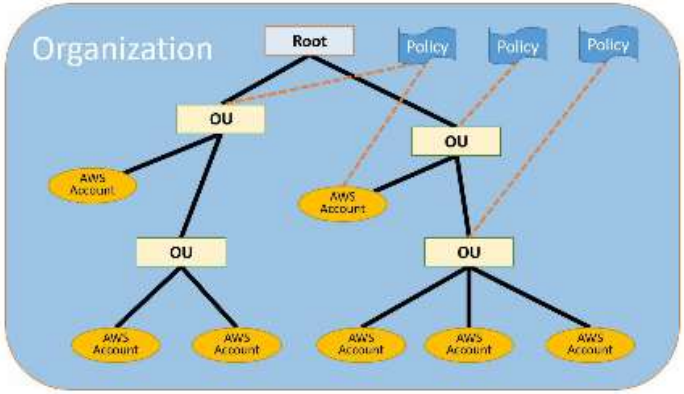
Integration and support for AWS IAM
- IAM provides granular control over users and roles in individual accounts.
- AWS Organizations expands that control to the account level by giving you control over what users and roles in an account or a group of accounts can do.
- Resulting permissions are a logical intersection of what is allowed by AWS Org. at the account level, and what permissions are explicitly granted by IAM at the user or role level within that account.
- In other words, the user can access only what is allowed by both the AWS Organizations policies and IAM policies. If either blocks an operation, the user can't access that operation.

Integration with other AWS services
- https://docs.aws.amazon.com/organizations/latest/userguide/orgs_integrated-services-list.html

Data replication that is eventually consistent

AWS Organizations - Concepts

https://docs.aws.amazon.com/organizations/latest/userguide/orgs_getting-started_concepts.html



- Master account creates the org. The rest of the accounts that belong to an org. are called member accounts.
- From the org.'s master account, you can create accounts in the org., invite other existing accounts to the org., remove accounts from the org., manage invitations, and apply policies to entities (roots, OUs, or accounts) within the org.
- An account can be a member of only one org. at a time.

Invitation:

1. Process of asking another account to join your org.
2. Invitations can also be sent to all current member accounts when the org. needs all members to approve the change from supporting only consolidated billing features to supporting all features in the org.

Handshake:

- One of its primary uses in AWS Org. is to serve as the underlying implementation for invitations.

Available feature sets

- All features OR
- Consolidated Billing

Service Control Policy (SCP)

- A policy that specifies the services and actions that users and roles can use in the accounts that the SCP affects.
- SCPs are similar to IAM permission policies except that they don't grant any permissions. Instead, SCPs specify the maximum permissions for an organization, OU or account.
- You can attach an SCP to 1) A root, which affects all accounts in the org. 2) An OU, which affects all accounts in that OU and all accounts in any OUs in that OU subtree 3) An individual account
- The master account is not affected by any SCPs that are attached either to it or to any root or OU the master account might be in.
- AllowList / whitelist (**default-FullAWSAccess-replace**)
- DenyList / blacklist (**default-FullAWSAccess-Add denylist**)

AWS Organizations - Moving from "Consolidated billing" to "All features"

Before changing from an organization that supports **only consolidated billing** features to an organization supporting **"All features"**, note the following:

- AWS Org. sends a request to every member account that you invited to join your org. **Every invited account must approve** enabling all features by accepting the request.
- Org. verifies that every account has a service-linked role named **AWSServiceRoleForOrganizations**. This role is mandatory in all accounts to enable all features.
- While enabling all features is in progress, you can continue to create accounts within the org. but you can't invite existing accounts to join the org.
- Because enabling all features makes it possible to use SCPs, be sure that your account administrators understand the effects of attaching SCPs to the org., OUs or accounts.
- **The master account isn't affected by any SCP**. You can't limit what users and roles in the master account can do by applying SCPs. **SCPs affect only member accounts.**
- The **migration from consolidated** billing features to all features is **one-way**.
- If your organization has only consolidated billing features enabled, member account administrators can choose to delete the service-linked role named AWSServiceRoleForOrganizations. However, when you enable all features in an organization, this role is required and is recreated in all accounts as part of accepting the invitation to enable all features.

AWS Organizations - Impact on member accounts

https://docs.aws.amazon.com/organizations/latest/userguide/orgs_getting-started_concepts.html

Impact on an AWS Account That You Invite to Join an Organization:

- When an invited account joins your organization, you do not automatically have full administrator control over the account, unlike created accounts. If you want the master account to have full administrative control over an invited member account, you must create the **OrganizationAccountAccessRole** IAM role in the member account and grant permission to the master account to assume the role.
- AWS Organizations does automatically create a service-linked role called **AWSServiceRoleForOrganizations** in invited member accounts to support integration between AWS Organizations and other AWS services.
- If you have any service control policies (SCPs) attached to the root of the OU tree, **those SCPs immediately apply to** all users and roles in the invited account.
- If you have enabled service trust for another AWS service for your organization, **that trusted service can create service-linked roles** or perform actions in any member account in the organization, including an invited account.

Impact on an AWS Account That You Create in an Organization

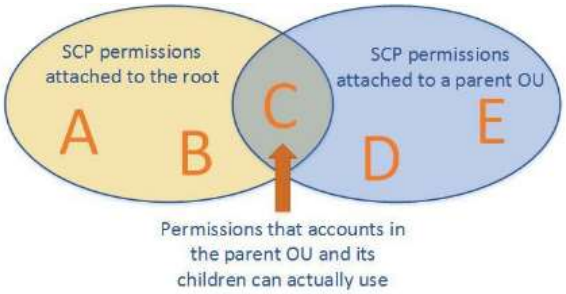
- AWS Organizations creates the IAM role **OrganizationAccountAccessRole**.
- AWS Organizations creates a service-linked role called **AWSServiceRoleForOrganizations**. The account must have this role if your organization supports all features.
- If you have any SCPs attached to the root of the OU tree, those SCPs immediately apply to all users and roles in the created account. New accounts are added to the root OU by default.
- If you have enabled service trust for another AWS service for your organization, **that trusted service can create service-linked roles** or perform actions in any member account in the organization, including an invited account.

AWS Organizations - Service Control Policy (SCP)

- SCPs offer central control over the maximum available permissions for all accounts in your organization, allowing you to ensure your accounts stay within your organization's access control guidelines.
- SCPs are available only in an organization that has all features enabled (not just consolidated billing).
- SCPs are necessary but not sufficient for granting access in the accounts in your org. You still need to attach IAM policies to users and roles in your org's accounts to actually grant permissions to them.

- Any account has only those permissions permitted by every parent above it. If a permission is blocked at any level above the account, either implicitly (by not being included in an Allow policy statement) or explicitly (by being included in a Deny policy statement), a user or role in the affected account can't use that permission, even if the account administrator attaches the AdministratorAccess IAM policy with */* permissions to the user.
- An SCP never grants permissions. Instead, SCPs are JSON policies that specify the maximum permissions for an organization or OU.
- A user without any IAM permission policies has no access at all, even if the applicable SCPs allow all services and all actions.
- If a user or role has an IAM permission policy that grants access to an action that is also allowed by the applicable SCPs, the user or role can perform that action.
- If a user or role has an IAM permission policy that grants access to an action that is either not allowed or explicitly denied by the applicable SCPs, the user or role can't perform that action.
- SCPs affect all users and roles in attached accounts, including the root user.
- SCPs do not affect any service-linked role.
- SCPs affect only principals that are managed by accounts that are part of the organization. They don't affect users or roles from accounts outside the organization.
- When you disable the SCP policy type in a root, all SCPs are automatically detached from all entities in that root.

AWS Organizations - How SCPs work



1. Because the root's OU doesn't allow D or E, nothing in the root or any of its children can use them, including the parent OU.
2. Also, because the OU's SCP doesn't allow A or B, those permissions are blocked for the parent OU and any of its children. However, other OUs under the root that are peers to the parent OU could allow A and B.

1. When you attach SCPs to the root, OUs, or directly to accounts, all policies that affect a given account are evaluated together using the same rules that govern IAM permission policies
2. Any action that has an explicit Deny in an SCP can't be delegated to users or roles in the affected accounts.
3. An explicit Deny statement overrides any Allow that other SCPs might grant.
4. Any action that has an explicit Allow in an SCP (such as the default "*" SCP or by any other SCP that calls out a specific service or action) can be delegated to users and roles in the affected accounts.
5. Any action that isn't explicitly allowed by an SCP is implicitly denied and can't be delegated to users or roles in the affected accounts.

- By default, an SCP named FullAWSAccess is attached to every root, OU, and account. This default SCP allows all actions and all services. So in a new organization, until you start creating or manipulating the SCPs, all of your existing IAM permissions continue to operate as they did.
- As soon as you apply a new or modified SCP to a root or OU that contains an account, the permissions that your users have in that account become filtered by the SCP. Permissions that used to work might now be denied if they're not allowed by the SCP at every level of the hierarchy down to the specified account.

AWS Organizations - Allow vs Denylist

- **A deny list** (default) – actions are allowed by default, and you specify what services and actions are prohibited
- **An allow list** – actions are prohibited by default, and you specify what services and actions are allowed

AWS Organizations attaches an AWS managed SCP named **FullAWSAccess** to every root and OU when it's created. This policy allows all services and actions.
This policy enables account administrators to delegate permissions for any service or action until you create and attach an SCP that denies some access.

An explicit Deny overrides an explicit Allow, which overrides the default implicit Deny

You could also configure this by leaving the FullAWSAccess policy in place and then attaching a second policy that has only the Deny statement in it, as shown here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "dynamodb:*",
            "Resource": "*"
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowsAllActions",
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Sid": "DenyDynamoDB",
            "Effect": "Deny",
            "Action": "dynamodb:*",
            "Resource": "*"
        }
    ]
}
```

The organization administrator can detach the FullAWSAccess policy and attach this one instead. This SCP still allows all other services and their actions.

Using SCPs as an Allow List:
To use SCPs as an allow list, you must **replace the AWS managed FullAWSAccess SCP** with an SCP that explicitly permits only those services and actions that you want to allow.

Service Control Policies (SCP)

- SCPs are mainly used along with AWS Organizations organizational units (OUs).
- SCPs do not replace IAM Policies such that they do not provide actual permissions. To perform an action, you would still need to grant appropriate IAM Policy permissions.
- Even if a Principal is allowed to perform a certain action (granted through IAM Policies), an attached SCP will override that capability if it enforces a Deny on that action.
- SCP takes precedence over IAM Policies.
- SCPs can be applied to the root of an organization or to individual accounts in an OU.
- When you apply an SCP to an OU or an individual AWS account, you choose to either enable (whitelist), or disable (blacklist) the specified AWS service. Access to any service that isn't explicitly allowed by the SCPs associated with an account, its parent OUs, or the master account is denied to the AWS accounts or OUs associated with the SCP.
- Any account has only those permissions permitted by every parent above it. If a permission is blocked at any level above the account, either implicitly (by not being included in an Allow policy statement) or explicitly (by being included in a Deny policy statement), a user or role in the affected account can't use that permission, even if there is an attached IAM policy granting Administrator permissions to the user.
- SCPs affect only principals that are managed by accounts that are part of the organization.

IAM Policies

- IAM Policies operate at the Principal level. There are two types of IAM policies
  - Identity-based policies - attached to an IAM user, group, or role.
  - Resource-based policies - attached to an AWS resource such as an S3 bucket.
- IAM Policies can grant/deny a Principal permissions to perform certain actions to certain resources. This can be used together with SCP to ensure stricter controls in AWS Organizations.
  An IAM policy can be applied only to IAM users, groups, or roles, and it can never restrict the root identity of the AWS account.
- IAM Policies cannot be attached to OUs.
- An IAM Policy can allow or deny actions. An explicit allow overrides an implicit deny. An explicit deny overrides an explicit allow.

CloudFormation

AWS CloudFormation Best practices

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html

Planning and organizing

- Organize Your Stacks By Lifecycle and Ownership
- Use Cross-Stack References to Export Shared Resources
- Use IAM to Control Access
- Reuse Templates to Replicate Stacks in Multiple Environments
- Verify Quotas for All Resource Types
- Use Nested Stacks to Reuse Common Template Patterns

Creating templates

- Do Not Embed Credentials in Your Templates
- Use AWS-Specific Parameter Types
- Use Parameter Constraints
- Use AWS::CloudFormation::Init to Deploy Software Applications on Amazon EC2 Instances
- Use the Latest Helper Scripts
- Validate Templates Before Using Them

Managing stacks

- Manage All Stack Resources Through AWS CloudFormation
- Create Change Sets Before Updating Your Stacks
- Use Stack Policies
- Use AWS CloudTrail to Log AWS CloudFormation Calls
- Use Code Reviews and Revision Controls to Manage Your Templates
- Update Your Amazon EC2 Linux Instances Regularly

AWS CloudFormation

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.

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

Currently, the only AWS CloudFormation resources that support creation policies are AWS::AutoScaling::AutoScalingGroup, AWS::EC2::Instance, and AWS::CloudFormation::WaitCondition.

Refer to Resource Outputs in Another AWS CloudFormation Stack

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html

To export resources from one AWS CloudFormation stack to another, create a **cross-stack reference**. Cross-stack references let you use a layered or service-oriented architecture. 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. By restricting cross-stack references to outputs, you control the parts of a stack that are referenced by other stacks.

For example, you might have a network stack with a VPC, a security group, and a subnet for public web applications, and a separate public web application stack. To ensure that the web applications use the security group and subnet from the network stack, you create a cross-stack reference that allows the web application stack to reference resource outputs from the network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.

To create a cross-stack reference, use the Export output field to flag the value of a resource output for export. Then, use the Fn::ImportValue intrinsic function to import the value.

DeletionPolicy Attribute

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html

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. If a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource by default.

DeletionPolicy Options

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. By default, if you don't specify a DeletionPolicy, AWS CloudFormation deletes your resources. However, be aware of the following considerations:

- For AWS::RDS::DBCluster resources, the default policy is Snapshot.
- For AWS::RDS::DBInstance resources that don't specify the DBClusterIdentifier property, the default policy is Snapshot.
- For Amazon S3 buckets, you must delete all objects in the bucket for deletion to succeed.

Retain

Keeps the resource without deleting the resource or its contents when its stack is deleted. When AWS CloudFormation completes the stack deletion, the stack will be in Delete_Complete state; however, resources that are retained continue to exist and continue to incur applicable charges until you delete those resources.

- If a resource is deleted, the DeletionPolicy retains the physical resource but ensures that it's deleted from CF scope.
- If a resource is updated such that a new physical resource is created to replace the old resource, then the old resource is completely deleted, including from AWS CloudFormation's scope.

Snapshot

For resources that support snapshots, AWS CloudFormation creates a snapshot for the resource before deleting it. Note that when AWS CloudFormation completes the stack deletion, the stack will be in the Delete_Complete state; however, the snapshots that are created with this policy continue to exist and continue to incur applicable charges until you delete those snapshots. Resources that support snapshots include:

- AWS::EC2::Volume, AWS::ElastiCache::CacheCluster, AWS::ElastiCache::ReplicationGroup, AWS::Neptune::DBCluster
- AWS::RDS::DBCluster, AWS::RDS::DBInstance, AWS::Redshift::Cluster

Elastic BeanStalk

AWS Elastic Beanstalk

- 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.

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.

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.
- An Elastic Beanstalk *application* is a logical collection of Elastic Beanstalk components, including *environments*, *versions*, and *environment configurations*.
- In Elastic Beanstalk an application is conceptually similar to a folder.
- Deleting an application terminates all associated environments and deletes all application versions and saved configurations that belong to the application.

AWS 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.
- An application version is part of an application.
- Applications can have many versions and each application version is unique.
- In a running environment, you can deploy any application version you already uploaded to the application or you can upload and immediately deploy a new application version.

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.

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).

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.

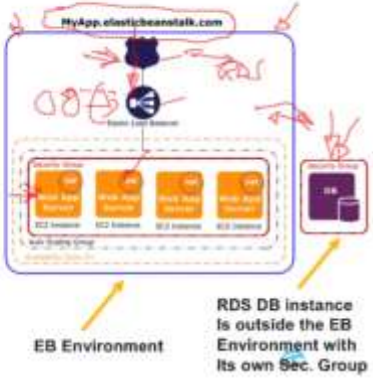
AWS Elastic Beanstalk

AWS resources created for an environment include one elastic load balancer, an Auto Scaling group, and one or more Amazon EC2 instances.

Every environment has a CNAME (URL) that points to a load balancer (Route53 Alias to the ELB)

The software stack running on the Amazon EC2 instances is dependent on the *container type*.

- A container type defines the infrastructure topology and software stack to be used for that environment.
- For example, an Elastic Beanstalk environment with an Apache Tomcat container uses the Amazon Linux operating system, Apache web server, and Apache Tomcat software.



Each Amazon EC2 server instance that runs your application uses one of these container types.

A software component called the *host manager (HM), an agent,* runs on each Amazon EC2 server instance. It is responsible for:

- Deploying the application
- Aggregating events and metrics for retrieval via the console, the API, or the command line
- Generating instance-level events
- Monitoring the application log files for critical errors
- Monitoring the application server
- Patching instance components
- **Rotating your application's log files and publishing them to Amazon S3**
- The host manager reports metrics, errors and events, and server instance status, which are available via Console, CLI, APIs

Key difference from OpsWorks

Elastic BeanStalk – Custom Platforms

- An Elastic Beanstalk platform comprises an AMI configured to run a set of software that supports an application, and metadata that can include custom configuration options and default configuration option settings.
- 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.

Elastic BeanStalk – Using a Custom AMI

- Applying configurations, however, can start to take a long time during environment creation and updates. If you do a lot of server configuration in configuration files, you can reduce this time by making a custom AMI that already has the software and configuration that you need.
- When you create an AWS Elastic Beanstalk environment, you can specify an Amazon Machine Image (AMI) to use instead of the standard Elastic Beanstalk AMI included in your platform configuration's solution stack.
- A custom AMI can improve provisioning times when instances are launched in your environment if you need to install a lot of software that isn't included in the standard AMIs.
- Using configuration files is great for configuring and customizing your environment quickly and consistently.

AWS Elastic Beanstalk

Migrating custom apps from on-prem to AWS, a possible option is to use custom built containers or use pre-configured docker platforms if it matches your tech stack

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.

Pre-configured Docker platform configurations:

- There are several *preconfigured* Docker platform configurations that you can use to run your application in a popular software stack such as *Java with Glassfish*.
- Use a preconfigured container **if it matches the software used by your application**.
- With preconfigured Docker platforms you cannot use a configuration file to customize and configure the software that your application depends on.
  - Instead, if **you want to customize** the preconfigured Docker platform to install additional software packages that your application needs, **you can add a Dockerfile to your application's root folder.**

Elastic BeanStalk and 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

AWS Elastic Beanstalk

Elastic BeanStalk and EFS

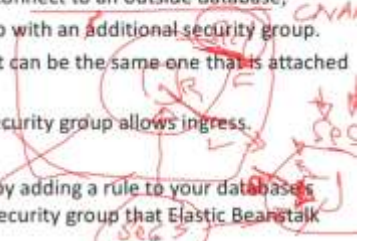
- With Amazon Elastic File System (Amazon EFS), you can create network file systems that can be mounted by instances across multiple Availability Zones.
- An Amazon EFS file system is an AWS resource that uses security groups to control access over the network in your default or custom VPC.
- In an Elastic Beanstalk 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 RDS

- To allow the Amazon EC2 instances in your environment to connect to an outside database,
  - You can configure the environment's Auto Scaling group with an additional security group.
  - The security group that you attach to your environment can be the same one that is attached to your database instance, or
  - A separate security group from which the database's security group allows ingress.
- Although you can connect your environment to a database by adding a rule to your database's security group that allows ingress from the autogenerated security group that Elastic Beanstalk attaches to your environment's Auto Scaling group.
  - The drawback will be that, doing so creates a dependency between the two security groups.
    - Subsequently, when you attempt to terminate the environment, Elastic Beanstalk will be unable to delete the environment's security group because the database's security group is dependent on it.



AWS Elastic Beanstalk - monitoring & 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.
  - Once Amazon CloudWatch contains your custom metrics, you can view those in the Amazon CloudWatch console.
- 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

With CloudWatch Logs, you can monitor and archive your **Elastic Beanstalk application, system, and custom log files from Amazon EC2** instances of your environments.

You can also configure alarms that make it easier for you to react to specific log stream events that your metric filters extract.

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.

Elastic BeanStalk and CloudWatch Logs

You can configure Elastic Beanstalk to **automatically stream logs** to the CloudWatch service

In addition to instance logs, if you enable enhanced health for your environment, you can configure the environment to stream health information to CloudWatch Logs.

To provide detailed health information about the EC2 instances running in your environment, Elastic Beanstalk includes a health agent in the Amazon Machine Image (AMI) for each platform configuration that supports enhanced health.

The health agent monitors web server logs and system metrics and relays them to the Elastic Beanstalk service.

- Elastic Beanstalk analyzes these metrics along with data from Elastic Load Balancing and Amazon EC2 Auto Scaling to provide an overall picture of an environment's health.

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.

To begin the export process, you must create an S3 bucket to store the exported log data.

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.

CloudTrail

AWS CloudTrail - Intro

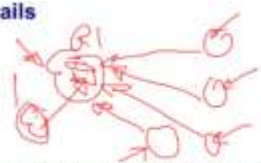
- AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account.
- Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
  - Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.
- CloudTrail is enabled on your AWS account when you create it (but not CloudTrail Logging).
- When activity occurs in your AWS account, that activity is recorded in a CloudTrail event.
  - You can easily view recent events in the CloudTrail console by going to Event history.

CloudTrail Event History and Trails

- Event history allows you to view, search, and download the past 90 days of activity in your account.
- You can create a **CloudTrail trail** to archive, analyze, and respond to changes in your AWS resources.
- CloudTrail logging, which is basically, sending the CloudTrail events to a bucket **is not enabled by default.**
  - You need to create a Trail and define a bucket, then CloudTrail will send events to this bucket, i.e will start logging the identified/selected events.
- A trail is a configuration that enables delivery of events to an Amazon S3 bucket that you specify.
- You can deliver and analyze events in a trail with Amazon CloudWatch (CW) Logs and CW Events.
- You can create a trail with the CloudTrail console, the AWS CLI, or the CloudTrail API.

CloudTrail Event History and Trails

You can create two types of trails:



- **A trail that applies to all regions (Recommended by AWS)**
  - When you create a trail that applies to all regions,
    - CloudTrail records events in each region and delivers the CloudTrail event log files to an S3 bucket that you specify.
      - This is effectively like creating the trail in each of these regions
  - If a region is added after you create a trail that applies to all regions,
    - That new region is automatically included, and events in that region are logged.

Advantages of All Regions Trails

- A trail that applies to all regions has the following advantages:
  - The configuration settings for the trail apply consistently across all regions.
  - Receiving CloudTrail events from all regions in a single S3 bucket and, optionally, in a CloudWatch Logs log group.
  - Managing trail configuration for all regions from one location.
  - Immediately receiving CloudTrail events from a new region when launched.
  - Ability to create trails in regions that you don't use often to monitor for unusual activity.
  - If CloudTrail is configured to use an Amazon SNS topic for the trail, SNS notifications about log file deliveries in all regions are sent to that single SNS topic.

AWS CloudTrail - Intro

The second type of Trail you can create,

- **A trail that applies to one region**
  - When you create a trail that applies to one region,
    - CloudTrail records the events in that region only.
    - It then delivers the CloudTrail event log files to an Amazon S3 bucket that you specify.
  - If you create additional single trails, you can have those trails deliver CloudTrail event log files to the same Amazon S3 bucket or to separate buckets.
  - This is the default option when you create a trail using the **AWS CLI or the CloudTrail API.**

Note

- For both types of trails, you can specify an Amazon S3 bucket from any region.

CloudTrail Events Logging

- You can store your log files in your bucket for as long as you want.
  - You can also define Amazon S3 lifecycle rules to archive or delete log files automatically.
- CloudTrail typically delivers log files within 15 minutes of account activity.
  - In addition, CloudTrail publishes log files multiple times an hour, about every five minutes.
  - These log files contain API calls from services in the account that support CloudTrail

Encryption of CloudTrail Events' Logging

- By default, the log files delivered by CloudTrail to your bucket are encrypted by Amazon server-side encryption with Amazon S3-managed encryption keys (SSE-S3).
- To provide a security layer that is directly manageable, you can instead use server-side encryption with AWS KMS–managed keys (SSE-KMS) for your CloudTrail log files.

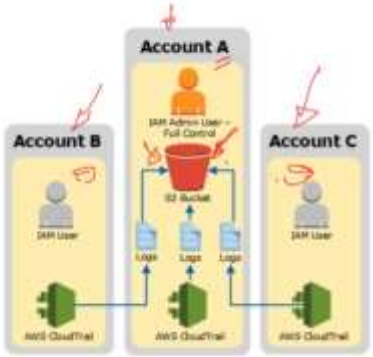
Note

- Enabling server-side encryption encrypts the log files but not the digest files with SSE-KMS. Digest files are encrypted with Amazon S3-managed encryption keys (SSE-S3).

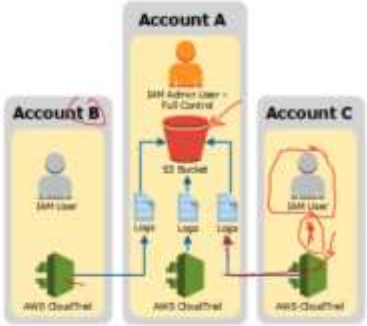
Cross-Account Scenarios

Receiving CloudTrail Log files from multiple AWS Accounts

- CloudTrail can deliver log files from multiple AWS accounts into a single Amazon S3 bucket.
- For example, you have three AWS accounts with account IDs A, B, C
  - You want to configure CloudTrail to deliver log files from all of these accounts to a bucket belonging to account A.
  - To accomplish this:
    - Turn on CloudTrail in the account where the destination bucket will belong, Account A
    - Do not turn on CloudTrail in any other accounts yet.



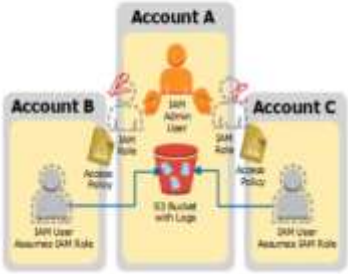
- Update the bucket policy on your destination bucket to grant cross-account permissions to CloudTrail
  - For a bucket to receive log files from multiple accounts, its bucket policy must grant CloudTrail permission to write log files from all the accounts you specify.
  - This means that you must modify the bucket policy on your destination bucket to grant CloudTrail permission to write log files from each specified account.
- Turn on CloudTrail in the other accounts B and C
- Configure CloudTrail in these accounts to use the same bucket belonging to account A



Sharing CloudTrail Log files between Accounts

Let's say in the previous example, now accounts B and C need access to their logs they uploaded in Account A's bucket.

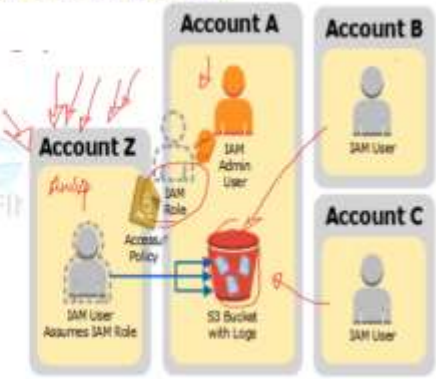
- To share log files between multiple AWS accounts, you must perform the following general steps.
  - In Account A, Create an IAM role for each account that you want to share log files with (B, C).
  - For each of these IAM roles, create an access policy that grants **read-only** access to the respective account (B or C).
  - An IAM user in each account (B and C) programmatically assume the appropriate role and retrieve the log files.



Sharing CloudTrail Log files between Accounts

If you want to grant/share logs with a third party (Auditor, Log analysis 3rd party provider, ..etc)

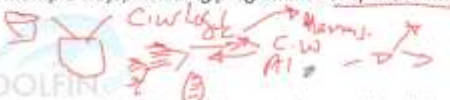
- You deal with it exactly the same
  - Create a Trust policy to the third party AWS account, in which you allow that account to assume an IAM role
    - The IAM role has an attached permission policy to define what is/isn't allowed to access in the CloudTrail log bucket
    - The third party account needs to Assume the IAM role and then will have access to the logs
    - It has to be confined to READ only access (GET and LIST operations)



CloudTrail Monitoring

SNS notifications:

- If you want notifications about log file delivery and validation,
  - Create and subscribe to an Amazon SNS topic to receive notifications about log file delivery to your bucket.
    - Amazon SNS can notify you in multiple ways, including programmatically with Amazon Simple Queue Service.



Monitor events with CloudWatch Logs

- You can configure your trail to send events to CloudWatch Logs. You can then use CloudWatch Logs to monitor your account for specific API calls and events

CloudWatch Logs and CloudTrail

- CloudWatch logs is a feature of CloudWatch that you can use specifically to monitor log data.
- Integration with CloudWatch Logs enables CloudTrail to send events containing API activity in your AWS account to a CloudWatch Logs log group.
- CloudTrail events that are sent to CloudWatch Logs can trigger alarms according to the metric filters you define.
  - You can optionally configure CloudWatch alarms to send notifications or make changes to the resources that you are monitoring based on log stream events that your metric filters extract.
- Using CloudWatch Logs, you can also track CloudTrail events alongside events from the operating system, applications, or other AWS services that are sent to CloudWatch Logs

CloudTrail VPC Endpoint

Using AWS CloudTrail with Interface VPC Endpoints

- If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and AWS CloudTrail.
  - You can use this connection to enable CloudTrail to communicate with your resources on your VPC without going through the public internet.
  - With VPC endpoints, the routing between the VPC and AWS Services is handled by the AWS network, and you can use IAM policies to control access to service resources.
  - To connect your VPC to CloudTrail, you define an interface VPC endpoint for CloudTrail.
    - An interface endpoint is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported AWS service.
    - The endpoint provides reliable, scalable connectivity to CloudTrail without requiring an internet gateway, network address translation (NAT) instance, or VPN connection.

OpsWorks

elastic beanstalk is the high level offering. it is the simplest way to deploy an application on aws. if you're looking for a no-frills, automagic, as-fully-managed-as-you-can-get-in-aws experience, this is it.

opsworks is the middle tier. operating as a full featured orchestration tool (thus the tight relationship with chef), opsworks combines straightforward deployment, configuration and management with the flexibility to handle complex implementations.

cloudformation is the nuts and bolts, low level utility. when you want granular control over everything in your environment, cfn is the choice. cfn can handle pretty much anything - from tiny footprint, one instance web server deployments to netflix - with a templatized, code driven approach. if you're doing serious work with aws, you're probably using cloudformation.

AWS OpsWorks

AWS OpsWorks is a configuration management service that helps you configure and operate applications in a cloud enterprise by using Puppet or Chef.

AWS OpsWorks Stacks - the original service, provides a simple and flexible way to create and manage stacks and applications. AWS OpsWorks Stacks lets you deploy and monitor applications in your stacks. You can create stacks that help you manage cloud resources in specialized groups called *layers*. A layer represents a set of EC2 instances that serve a particular purpose, such as serving applications or hosting a database server. Layers depend on Chef recipes to handle tasks such as installing packages on instances, deploying apps, and running scripts.

AWS OpsWorks for Chef Automate - lets you use Chef cookbooks and solutions for configuration management. lets you create AWS-managed Chef servers that include Chef Automate premium features, and use the Chef DK and other Chef tooling to manage them. A Chef server manages nodes in your environment, stores information about those nodes, and serves as a central repository for your Chef cookbooks.

OpsWorks for Puppet Enterprise lets you configure a Puppet Enterprise master server in AWS.

AWS OpsWorks

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.

AWS OpsWorks Stacks provides a rich set of customizable components that you can mix and match to create a stack that satisfies the application requirements.

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)

Chef

As the environments grow, manual configuration and deployment practices can result in operational expenses growing at an alarming rate.

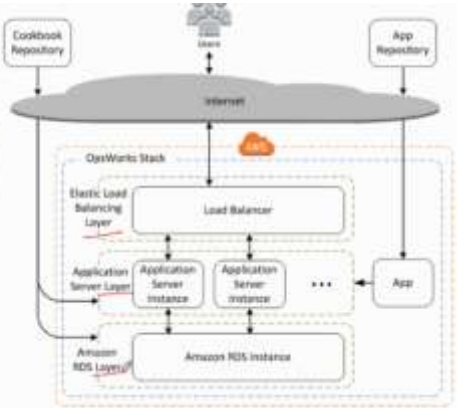
Chef provides automated configuration management that enables consistent configurations at scale

Chef helps in ensuring that configuration policy is flexible, versionable, testable and human readable.

Servers managed by Chef are continuously evaluated against their desired state, ensuring that configuration drift is automatically corrected, and configuration changes are universally applied.

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

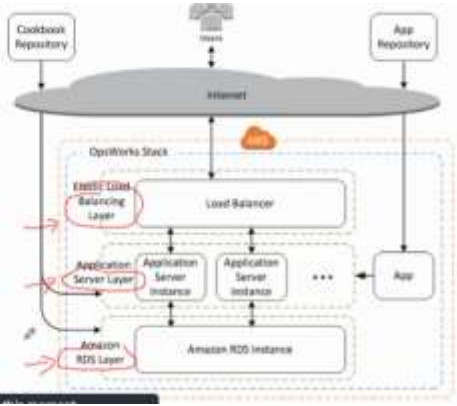
- These resource have a common purpose and should be logically managed together.
- The stack helps to manage these resources as a group
- The stack also defines some default configuration settings, such as the instances' operating system and AWS region.



AWS OpsWorks

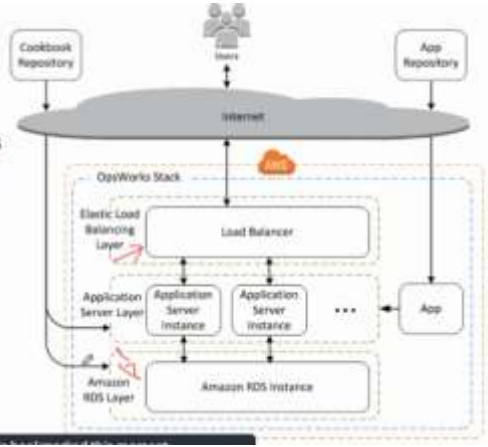
Layers:

- 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.



Layers:

- You can customize or extend layers by:
  - Modifying packages' default configurations,
  - Adding Chef recipes to perform tasks such as installing additional packages, 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 – 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.

OpsWorks – LifeCycle Events Example

An instance that belongs to a web server layer, once it finishes booting, AWS OpsWorks does the following.

- Runs the layer's **Setup recipes**, this will install the web server
- Runs the layer's **Deploy recipes**, this will deploy the apps from a repository to the instance
- Runs the **Configure recipes** on every instance in the stack so each instance can adjust its configuration as needed to accommodate the new instance.

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, → online
  - **AWS OpsWorks installs an agent** that handles communication between the instance and the service and runs the appropriate recipes in response to lifecycle events.

AWS OpsWorks

OpsWorks – Insance types

Classified according to how they are started and stopped,

- AWS OpsWorks Stacks supports the following instance types,.
  - **24/7 instances:**
    - Are started manually and run until you stop them.
  - **Time-based instances:**
    - Are run by AWS OpsWorks on a specified daily and weekly schedule.
    - They allow the stack to automatically adjust the number of instances to accommodate predictable usage patterns.
  - **Load-based instances:**
    - Are automatically started and stopped by AWS OpsWorks, based on specified load metrics, such as CPU utilization.
    - They allow the stack to automatically adjust the number of instances to accommodate variations in incoming traffic.
    - Currently, Load-based instances are available only for Linux-based stacks.

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

OpsWorks Components - Apps

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.

Typically AWS OpsWorks Stacks run the Deploy recipes on the entire stack,

- This would allow the other layers' instances to modify their configuration appropriately.

You can also limit deployment to a subset of instances if/when needed

OpsWorks Public and Elastic IPs (Layer's Network settings)

Elastic IP addresses	Public IP addresses: Yes	Public IP addresses: No
Yes	Instances receive an Elastic IP address when they are started for the first time, or a public IP address if an Elastic IP cannot be assigned.	Instances receive an Elastic IP address when they are started for the first time.
No	Instances receive a public IP address each time they are started.	Instances receive only a private IP address, which is not accessible from outside the VPC.

- Note
  - OpsWorks Instances must have a way to communicate with the AWS OpsWorks service, Linux package repositories, and cookbook repositories.
    - If you specify no public or Elastic IP address, your VPC must include a component such as a NAT that allows the layer's instances to communicate with external sites.

AWS OpsWorks

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 – 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.

OpsWorks – A common error

Unable to Manage Instances Problem:

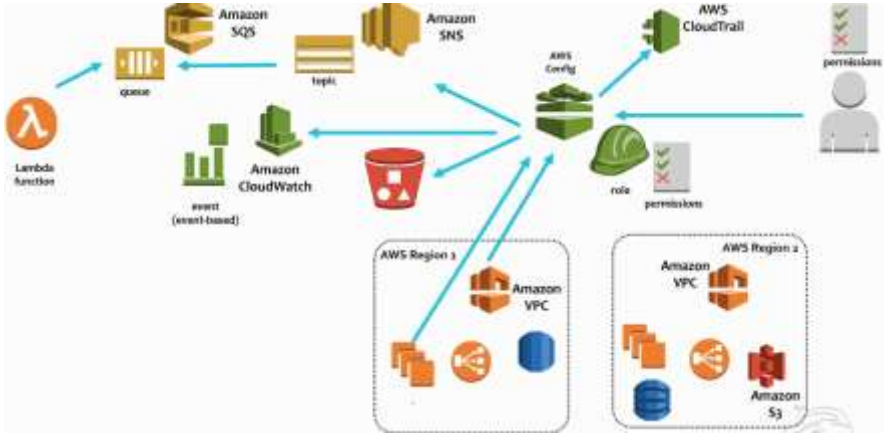
- If you are no longer able to manage an instance that has been manageable in the past. In some cases, logs can show an error similar to the following.
  Aws::CharlieInstanceService::Errors::UnrecognizedClientException - The security token included in the request is invalid.

- **Cause:**
  - This can occur if a resource outside AWS OpsWorks on which the instance depends was edited or deleted, the following can be reasons,
    - The IAM user or role associated with the instance has been deleted accidentally, outside of AWS OpsWorks Stacks.
    - Editing volume or storage configurations while an instance is offline can make an instance unmanageable.
    - Adding EC2 instances to an EIP manually. AWS OpsWorks reconfigures an assigned Elastic Load Balancing load balancer each time an instance enters or leaves the online state.

Stacks → Layers → Instances → Apps

AWS Config

AWS Config



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.

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

- **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.
    - The components of a configuration item include metadata, attributes, relationships, current configuration, and related events.
  - 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.
  - A configuration history can help find answers such as when the resource was created, when it was changed, the status at a specific point in time.
  - 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.
  - By default, the configuration recorder records all supported resources in the region where AWS Config is running.
    - This can be customized.

AWS Config

- **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
    - The configuration stream is helpful for observing configuration changes as they occur so that you can spot potential problems, generating notifications if certain resources are changed, or updating external systems that need to reflect the configuration of your AWS resources.

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

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.
      - Each configuration history file contains details about the resources that changed in that six-hour period.
      - Each file includes resources of one type, such as Amazon EC2 instances or Amazon EBS volumes.
    - 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.

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.

The rule has a trigger, it specifies when should the run be run, Periodically (ex ever day), or only when there are configuration changes

Each rule is associated with an AWS Lambda function, which contains the evaluation logic for the rule.
- AWS Config invokes the Lambda function to return the compliance status of the resource being evaluated.

AWS Config sends notifications for the following events:
- Configuration item change for a resource.
- Configuration history for a resource was delivered for your account.
- Configuration snapshot for recorded resources was started and delivered for your account.
- Compliance state of your resources and whether they are compliant with your rules.
- Evaluation started for a rule against your resources.
- AWS Config failed to deliver the notification to your account.

When using AWS Config rules, and a violation is spotted AWS Config flags a resource and the rule as noncompliant, and AWS Config notifies you through Amazon SNS.

AWS Config

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.

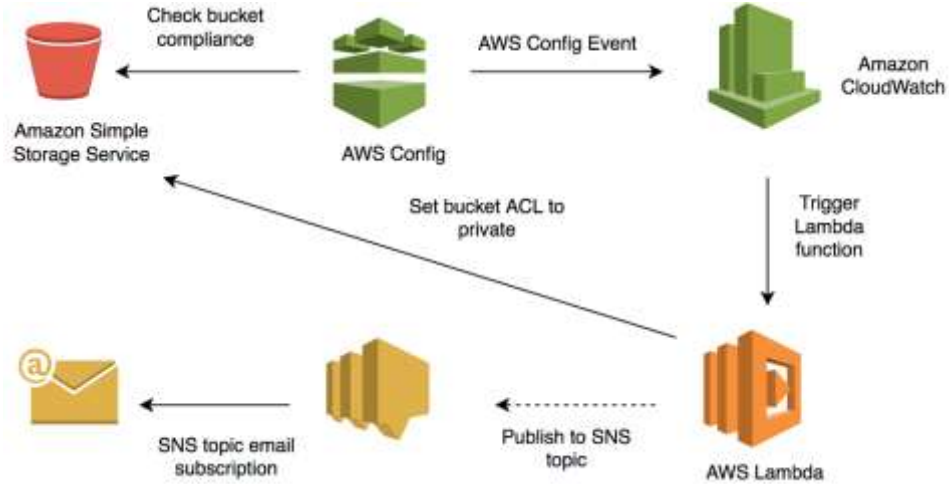
Monitoring - Using Amazon CloudWatch Events

Use Amazon CloudWatch Events to detect and react to changes in the status of AWS Config events.

How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access

https://aws.amazon.com/blogs/security/how-to-use-aws-config-to-monitor-for-and-respond-to-amazon-s3-buckets-allowing-public-access/

AWS Config enables continuous monitoring of your AWS resources, making it simple to assess, audit, and record resource configurations and changes. AWS Config does this through the use of rules that define the desired configuration state of your AWS resources. AWS Config provides a number of AWS managed rules that address a wide range of security concerns such as checking if you encrypted your EBS volumes, tagged your resources appropriately, and enabled MFA for root accounts. You can also create custom rules to codify your compliance requirements through the use of AWS Lambda functions.



1. Enable Config to monitor S3 bucket ACLs and policies for compliance violations.
2. Create an IAM Role and Policy that grants a Lambda function permissions to read S3 bucket policies and send alerts through SNS.
3. Create and configure a CloudWatch Events rule that triggers the Lambda function when AWS Config detects an S3 bucket ACL or policy violation.
4. Create a Lambda function that uses the IAM role to review S3 bucket ACLs and policies, correct the ACLs, and notify your team of out-of-compliance policies.
5. Verify the monitoring solution.