AWS CloudFormation Basics

To use AWS CloudFormation, we must know a few basic concepts involved:

1. Template: This forms **the blueprint to build the infrastructure in CloudFormation**. It can either be a **JSON or a YAML file with extensions as .json, .yaml, .txt, .extension**. The template will contain all the data of the infrastructure that is required to host an application.

The following is a sample template used to create an EC2 instance and attach an EBS volume to it:

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "A sample template",

"Resources" : {

"MyEC2Instance" : {

"Type" : "AWS::EC2::Instance",

"Properties" : {

"ImageId" : "ami-0ff8a91507f77f867",

"InstanceType" : "t2.micro",

"KeyName" : "testkey",

"BlockDeviceMappings" : [

{

"DeviceName" : "/dev/sdm",

"Ebs" : {

"VolumeType" : "io1",

"Iops" : "200",

"DeleteOnTermination" : "false",

"VolumeSize" : "20"

}

}

]

}

}

}

}

1. Stacks: AWS CloudFormation stack is a group of resources that can be controlled and created by using the template. You can **create, update, and delete a collection of resources** by creating, updating, and deleting stacks. Stacks can be created by using console, CLI, and API.
2. Change Sets: To make changes to the running resources in the stack, you have to update the stack. However, before making changes to the resources through updated stack, you can create a **Change Set** which consists of the required changes. These help to examine the impact of the changes made on the resources.

**CloudFormation Designer** is a tool used to create and modify templates easily in AWS through drag and drop options. You draw a diagram of resources, an appropriate JSON is created, and details such as names, regions, and so on, can be edited manually by using the integrated editor.

* Provides a graphical representation of resources and their relationships, thereby helping in parsing the complex JSONs.
* Helps in error free infrastructure coding, and validation of the code after manual changes.
* You spend less time in coding, and get more time to design the infrastructure.

##### Intrinsic Functions

**Intrinsic Functions** are useful when you want to assign values to properties in a template that are not available until runtime.

CloudFormation provides a set of intrinsic functions that help in managing stacks:

* **Fn : : Base64** : Returns the Base64 representation of the input string.
* **Fn : : Cidr** : Returns an array of CIDR address blocks, and the number of blocks is dependent on the parameter used.
* **Fn : : FindInMap** : Returns values corresponding to keys in a two-level map that are declared in the Mappings section.
* **Fn : : GetAtt** : Returns the value of an attribute from a resource.
* **Fn : : GetAZs** : Returns the array list of AZs in a specified region.
* **Fn : : ImportValue** : Returns a value that is exported, or the output of another stack.
* **Fn : : Join** : Returns the appended value separated by a specified delimiter.
* **Fn : : Select** : Returns the value from a list of objects by index.
* **Fn : : Split** : Splits a string into a list of string values, so that you can select an element from the resulting string list. The location of the split is defined by a delimiter that is used in conjunction with **Select**.
* **Fn : : Sub** : Substitutes variables in an input string with values you specify.
* **Fn : : Transform** : Specifies a macro to perform a specific process on a specific part of a stack.
* **Condition Functions** : All the intrinsic functions can be used to create a stack with conditions, or apply conditions for certain resources. There are five conditional functions; **Fn : : If , And , Equals , Not , Or**. All conditions are used in the Conditions section, except for **Fn : : If**. Details about conditions, and examples are available in [Condition Documentation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-conditions.html).

For detailed explanation with examples of intrinsic functions, refer to [Intrinsic Functions Documentation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html).

##### CloudFormation Helper Scripts

**Helper Scripts** are python scripts that help in installing softwares and services in EC2 instances as a part of a stack.

They are of four types :

1. **Cfn-init :** Reads the template metadata from AWS : : CloudFormation : : Init, parses the metadata, installs packages, writes files, and starts/stops services.
2. **Cfn-Signal :** Makes CloudFormation indicate the successful creation of EC2 instances with all softwares installed.
3. **Cfn-get-metadata :** Used to print metadata block from CloudFormation and print it.
4. **Cfn-hup :** Used to detect changes in instance metadata, and performs specified actions.

For more details about Helper Scripts, refer to [helper scripts](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-helper-scripts-reference.html).

**Nested Stacks** are stacks created within another stack by using AWS : : CloudFormation : : Stack.\*

* Are very useful when you use resources with the same configuration multiple times. These can be consolidated into a stack, and further used.
* Can have more nested stacks within themselves.
* Each nested stack has a parent stack and only one root stack.
* Using nested stacks is considered a good practice.

##### CloudWatch - Introduction

**CloudWatch** is a real-time management and monitoring service provided by AWS, which:

* Provides you with complete insight of an application at one place.
* Is extensively used for application resources such as EC2 instances, ELBs, Database Servers, EBS, and so on.
* Helps you collect metrics, log files, set alarms, and also automatically react to changes in resources.
* Automatically provides metrics for CPU utilization, latency, and request counts.
* Is very useful when taking automated actions, troubleshooting issues, requiring a total insight of the application, and optimizing the application.
* Is used to create custom dashboards to display metrics.
* Can be accessed through the Console, AWS CLI, CloudWatch API, and various SDKs

##### CloudWatch Components

CloudWatch has three key capabilities:

* **CloudWatch Monitoring** - To collect metrics data, and set alarms.
* **CloudWatch Logs** - To collect and analyze log files from AWS services, and applications running on EC2 machines.
* **CloudWatch Events** - To trigger events based on changes in AWS resources, time, or cron expressions.
* **Metrics:** This is a fundamental component of CloudWatch Monitoring. Metrics are ordered sets of data points collected over a period of time, that are published to CloudWatch. This metric is the variable to monitor, and provides information on resource performance, utilization, and health.
  + You can send custom metrics to CloudWatch, and add data sets in any order and rate.
  + Metrics can be got from on-premise and hybrid servers which are not under AWS control, by using **CloudWatch Agent**.
  + Metrics exist only in the region where they are created.
  + Metrics cannot be deleted, but they expire after 15 months, if there is no data flow.
  + Metrics are identified peculiarly by using names, namespaces, zero, or more dimensions.
* **Namespaces** : Also known as the **container for metrics**. Namespaces are used to isolate the metrics of different services without getting them mixed up with each other.
* **Dimension** : A Dimension is a name or value pair that is used to identify a metric. You can assign upto 10 dimensions to a metric.

Dimensions: Server=Prod, Domain=Frankfurt, Unit: Count, Timestamp: 2016-10-31T12:30:00Z, Value: 105

Dimensions: Server=Beta, Domain=Frankfurt, Unit: Count, Timestamp: 2016-10-31T12:31:00Z, Value: 115

* **Statistics** : As the name suggests, **it is the accumulation of metric data over a specific period** of time. This aggregation is made by using the namespace, metric name, dimensions, and the data point unit of measure within the time period specified. The available statistics are Maximum, Minimum, Sum, Average, SampleCount, and pNN.NN.

***Note:*** pNN.NN is the value of the specified percentile up to two decimal points.

* **Alarm** : Alarms monitor metrics on your behalf, and perform specified action depending on the threshold value set. A popular action is sending a message to an SNS topic. Apart from specifying the threshold value, you can also set sophisticated rules.

##### Monitoring Types

There are two types of Metrics monitoring in CloudWatch:

* **Basic Monitoring :** This is free, and collects data from AWS resources at an interval of five minutes. All resources, except ELB and RDS provide metrics in basic monitoring. For other services, you can enable detailed monitoring.
* **Detailed Monitoring :** This is charged, and collects metrics at an interval of one minute. A new **high-resolution metric** is available which collects metrics at 1-second resolution. ELB and RDS is facilitated only in detailed monitoring.

There will be at least five metrics for a single EC2 instance. Therefore, how do you manage hundreds of them? Moreover, many of them might not be useful to you. To overcome this, we have **Dashboards** which are customizable home pages.

* Dashboards help in monitoring all required metrics and alarms from different regions in a single page.
* It creates a common view of resource data which can be shared by the team, and enables faster resolutions to issues.
* With Dashboards, you can display the latest value of a metric, a simple line graph of one or more metrics, or a stacked area graph of multiple metrics.

CloudWatch Logs

**CloudWatch Logs** is a place to monitor, access, and store logs from various AWS services such as EC2 instances, Lambda, S3, VPC, and also the application running on EC2.

* It enables centralized logging, and easier monitoring and troubleshooting of issues.
* Helps you monitor logs in near real time.
* Logs can be sent to other services of AWS like S3, Lambda, and Elastic Search for storing, processing, and reporting.
* Logs are stored indefinitely and never expire. Also, you can set a retention policy and store logs with a timespan of 1 day to 10 years.
* Logs can be archived in highly durable storage by **logs agent**.
* Helps in logging DNS queries from Route 53.

There are few terms that are exclusive to CloudWatch, and helps us understand Logging better:

* **Log Events :** A log event is a record of activity of the application or resource being monitored by CloudWatch. It contains two properties: the timestamp of the event, and the raw event message.
* **Log Streams :** A log stream is the stream of events from an instance or application that is being monitored.
* **Log Groups :** A log group is the collection of log streams that have the same retention, monitoring and access control. Each log stream belongs to a log group, and there is no limit on how many log streams can be there in a log group.
* **Metric Filters :** Metric filters extract metric observations from events, and transform them into data points in CloudWatch metrics.

**CloudWatch Logs Insight** is a highly available, and interactive log analytics service for CloudWatch logs.

* The service was introduced in November, 2018.
* It enables you to explore, visualize, and analyze logs quickly, and troubleshoot operational issues.
* It has a query language with few commands, to perform operations.
* It can find the fields present in the logs that are emitted, in the form of a JSON. - It automatically generates three fields to logs for use in queries are@message , @timestamp, @logstream.

For more information about log insights, refer to [AWS official documentation](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html).

CloudWatch Events

**Events** is also an importance feature of CloudWatch. It provides a near real-time event stream, as changes take place in the AWS resources present. The change of events can trigger predefined actions on targets, such as send notifications and emails, invoke lambda functions, change instance state,and so on.

* By using simple rules, you can match events, and route them to one or more targets, functions, or streams.
* Helps to identify operational changes when they take place.
* Can be used to schedule automated actions that self-trigger at certain times, by using cron or rate expressions.
* It supports many crucial services as its targets in AWS.

##### AWS CloudTrail

**CloudTrail** is a security and management service which enables governance, compliance, and risk auditing of the AWS account. It continuously logs and monitors activities related to actions in your account.

* Every action taken through console, AWS CLI, AWS SDK's, and other AWS services is logged. - The information logged contains the identity of the API caller, time of the call, the source IP, request parameters, and response of the call.
* It helps in security analysis, resource change tracking, and compliance auditing.
* It is enabled by default when the AWS account is created.
* All the events are saved in event history in CloudTrail and saved for 90 days.
* You can create a trail to archive, analyze, and respond to changes, and log all of them to the S3 bucket you specify.

##### AWS CloudTrail

**CloudTrail** is a security and management service which enables governance, compliance, and risk auditing of the AWS account. It continuously logs and monitors activities related to actions in your account.

* Every action taken through console, AWS CLI, AWS SDK's, and other AWS services is logged. - The information logged contains the identity of the API caller, time of the call, the source IP, request parameters, and response of the call.
* It helps in security analysis, resource change tracking, and compliance auditing.
* It is enabled by default when the AWS account is created.
* All the events are saved in event history in CloudTrail and saved for 90 days.
* You can create a trail to archive, analyze, and respond to changes, and log all of them to the S3 bucket you specify.

##### Validating CloudTrail Logs

In an organization, many people may have access to CloudTrail logs, and also privileges to alter the log files. Therefore, to find out if a log has been edited, deleted, or left unchanged after CloudTrail delivery, you can use **CloudTrail log file integrity validation**.

* This is built by using industry standard algorithms such as SHA-256 for hashing, and SHA-256 with RSA for digital signing, making it impossible to alter log files without detection.
* When CloudTrail log validation is enabled every hour, it also creates and delivers a file named **digest file** that references the log files for the last hour, and contains a hash of each file.
* **AWS CLI** is used to validate CloudTrail log integrity.

##### CloudFront

**CloudFront** is a content delivery network (CDN) service that offers low-latent, fast, secure, and high-speed static and dynamic content delivery to end-users, globally.

* Is integrated with AWS global infrastructure, and works with many services.
* Uses edge locations of AWS which are a distributed network of data centers.
* Works seamlessly with services like S3, Elastic Load Balancer, EC2 instances, and AWS Shield for DDoS migration and AWS Lambda.

**CloudFront** is one of the most used services of AWS. It has a huge customer base, and organizations like King, Prime Video, Rovio, Discovery, Supercell, Bandai Namco use it for its following iconic features:

* The CloudFront network has 166 POP locations across the globe, aiding in superior performance and high-speed delivery to users.
* CloudFront is a highly secure CDN, providing both application and network security at no extra cost. Also, users have the ability to configure AWS Certificate manager to manage SSL certificates.
* It works with many AWS services, and with Lambda@Edge functions triggered by CloudFront events, you can take your code closer to the user to improve responsiveness and serve private content.
* It is very useful in video streaming, both live and pre-recorded video.
* It is a comparatively less-expensive service, and you are only charged for content delivered and requested.

**Edge locations** also known as the **POP locations**, are situated at almost all strategic locations, globally.

Therefore, when a user requests content from the origin server which might be an S3 Bucket or HTTP server, EC2 instance,

1. The request goes through the Edge location, and if the content is already cached there, it is provided to the requested user immediately.
2. If not cached in the Edge location (first request by first user), the Edge location's (POP location's) request is routed to the origin. The content is then sent to the Edge location, and from there it is delivered to the user.
3. The content is cached in the Edge location, and stays there as per the defined TTL.
4. Another user who requests the same object is serviced from the Edge location, with low latency.

##### Types of Origins

The previous example showed an S3 Bucket serving content to the CloudFront. However, it also supports various other services to act as an origin:

* **Using MediaStore Container/Package Channel :** This is used to stream video content to CloudFront, both on-demand and live streaming. On-demand video is streamed by using services like **AWS Elemental Media Convert** to convert media files into streaming package and is stored in S3.

For live streaming, either a live-event or 24/7 live channel, encoders such as **Media live or AWS Elemental Media Package** are used. Also, there are many third-party tools and servers too that does the encoding, converts the video to deliverable format, and uses CloudFront for delivery.

For the detailed steps and tasks involved in video streaming for both on-demand and live video streaming, refer to [AWS Documentation](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/on-demand-streaming-video.html).

* **EC2 Server or other HTTP server :** In this, the origin is a custom HTTP server, or an EC2 instance that you manage. An S3 bucket that is configured for website hosting is also a custom origin. When you configure this HTTP custom server as origin, you specify the DNS, HTTP, or HTTPS ports and protocol that CloudFront should use to fetch content.
* In this custom HTTP origin server, CloudFront does not support Real-Time Messaging Protocol Secure(RTMPS) and private content delivery.
* **CloudFront origin groups :** These origin groups are used for failover scenarios when you need high availability. Use origin failover to designate a primary origin for CloudFront, and a secondary origin, so that CloudFront automatically switches to the secondary origin when the primary origin returns specific HTTP status code.

##### Security and Limiting Access to Content

When data is transferred over the internet, there could be many unwanted threats. We must ensure that the data is not corrupted, and also restrict the content delivered to certain users, or users in a particular area.

CloudFront provides a set of solutions for this:

* **Using HTTPS :** CloudFront users can utilize HTTPS protocols to request objects. It can also be used to make HTTPS requests to the origin. This makes the connections encrypted, and highly secure.

This can be enabled in the viewer protocol policy and origin protocol policy sections of the dashboards respectively. You can also add your own certificates for other domains.

**Using WAF :** By using Web Application Firewall (WAF), you can restrict access to content, based on the request location, IP address, values of query strings, protocol used, and so on. After you create a web ACL with WAF, update the web distribution of CloudFront.

WAF is a very useful security service provided by AWS, and is associated with CloudFront for high-level security.

* **Using Origin Access Identity (OAI) :** CloudFront does not expose the S3 URLs. However, users may know these URLs from previous S3 URLs. If these S3 links are used, the CloudFront-signed URLs become obsolete. Therefore, to ensure usage of CloudFront URLs only, the OAI feature is used.

You change the bucket policy such that only the OAI has permission to access the objects in the bucket.

* **Using Field-Level Encryption :** Although HTTPS allows you to enable end-to-end encryption, field-level encryption enables an additional level of security, which protects certain data throughout system processing. This is used for securing user-sensitive, and highly confidential information.

Organizations that distribute content, may want to restrict a group of users from accessing some content, and provide access to users that have paid a fee. This can be achieved by using **Cloudfront-signed URLs** or **Signed Cookies**.

This can be achieved in two ways:

* Restricting access to objects in S3 directly (unless it is configured as a website).
* Restricting access to CloudFront edge cache content.

When you develop your application, allow them to create and distribute signed URLs, or send Set-Cookie headers which set signed cookies on the viewers for authenticated users. Using this, you can specify restrictions such as:

* An ending timestamp after which the URL is no longer valid.
* The date and time the URL becomes valid.
* IP address or the range of IPs of devices that can access.

A part of a signed URL or a signed cookie is hashed and signed, using the private key from a public/private key pair. When a signed URL or signed cookie is used to access a file, CloudFront compares the signed and unsigned portions of the URL or cookie. If they do not match, CloudFront will not serve the file.

##### AWS Simple Notification Service (SNS)

**AWS SNS** is a highly available, secure, fully managed messaging service that helps you decouple microservices, distributed systems, and serverless architecture.\*

* SNS provides topics for high-throughput, many-to-many, and push-based messaging.
* By using SNS topics, publisher systems can send messages to a large number of subscriber endpoints for parallel processing, including Amazon SQS queues, AWS Lambda functions, and HTTP/HTTPS webhooks.
* SNS can be used to send notifications to end-users through mobile push, SMS, and email.
* SNS helps in filtering notifications, by dropping a filtering logic in the subscriber system, and message-routing policies in the publisher side. Thus, you receive notifications of your interest only.
* With encryption, you can secure message content, and restrict the publisher and subscriber.

SNS Pricing

SNS uses a pay-as-you-go model. There are no upfront commitments.

* With SNS free tier, your first million push notifications are free every month.
* You pay based on the number of notifications you publish.
* SNS currently allows a maximum limit of 256 KB for published messages.
* Each 64 KB chunk of published data is billed as 1 request.
* Delivery prices vary depending on the end point, and SNS offers SMS services to more than 200 countries.

Amazon SQS

Amazon Simple Queue Service (SQS) is a web service that enables web service applications quickly and reliably queue messages, that one component or application generates, to be used by another component or application.

* It is used to decouple microservices, serverless systems, and distributed systems.
* It acts as a buffer between the component producing and saving the data, and the component receiving the data for processing.
* Messages can be 256 KB of text in any format, but are billed at 64 KB chunk size.
* SQS ensures delivery of each message at least once.
* A single queue can be used by multiple applications (on either side) simultaneously.
* The retention period for queue messages is 1 minute to 14 days, and the default is 4 days.
* SQS is a pull message service.
* There are two types of queues, **Standard Queue** and **FIFO Queue**.

##### Standard Queue

**Standard Queue** is the general type of queue, and is used widely for many applications. It is categorized by the following:

* They provide a loose First In First Out (FIFO) capability that does not guarantee preserving the order of messages.
* These queues are designed to be highly scalable, and have unlimited throughput. This is a reason for it not being linear.
* Messages are delivered at least once, and sometimes more than once.
* Is used when throughput between applications is important.

##### FIFO Queue

**First In First Out (FIFO) Queue** is a new type of queue service introduced recently. In FIFO,

* The queue preserves the exact order in which messages are received.
* The queue provide **exactly-once processing**, which means that each message is delivered once, and is available until a consumer processes it and deletes it.
* The throughput in this queue is limited to 3000 messages per second with batching, and 300 messages per second without batching.
* Duplicates introduced into the queue are deleted within a 5-minute de-duplication interval.

FIFO queue is used for applications where the events order is important.