**Amazon Redshift**

**Availability and Durability**

RedShift uses replication and continuous backups to enhance availability and improve durability and can automatically recover from component and node failures.

Only available in one AZ but you can restore snapshots into another AZ.

Alternatively, you can run data warehouse clusters in multiple AZ’s by loading data into two Amazon Redshift data warehouse clusters in separate AZs from the same set of Amazon S3 input files.

Redshift replicates your data within your data warehouse cluster and continuously backs up your data to Amazon S3.

RedShift always keeps three copies of your data:

* The original.
* A replica on compute nodes (within the cluster).
* A backup copy on S3.

RedShift provides continuous/incremental backups:

* Multiple copies within a cluster.
* Continuous and incremental backups to S3.
* Continuous and incremental backups across regions.
* Streaming restore.

RedShift provides fault tolerance for the following failures:

* Disk failures.
* Nodes failures.
* Network failures.
* AZ/region level disasters.

For nodes failures the data warehouse cluster will be unavailable for queries and updates until a replacement node is provisioned and added to the DB.

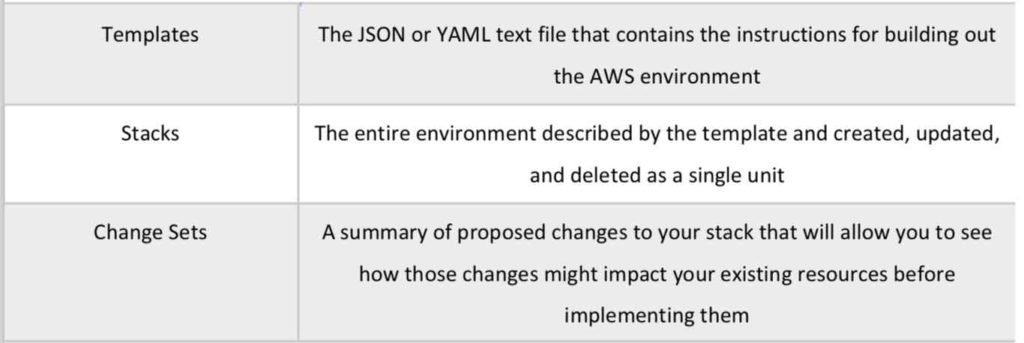
**High availability for RedShift:**

* Currently, RedShift does not support Multi-AZ deployments.
* The best HA option is to use multi-node cluster which supports data replication and node recovery.
* A single node RedShift cluster does not support data replication and you’ll have to restore from a snapshot on S3 if a drive fails.
* RedShift can asynchronously replicate your snapshots to S3 in another region for DR.
* **Single-node clusters do not support data replication (in a failure scenario you would need to restore from a snapshot).**
* Scaling requires a period of unavailability of a few minutes (typically during the maintenance window).
* During scaling operations RedShift moves data in parallel from the compute nodes in your existing data warehouse cluster to the compute nodes in your new cluster.
* By default, Amazon Redshift retains backups for 1 day. You can configure this to be as long as 35 days.
* If you delete the cluster you can choose to have a final snapshot taken and retained.
* **Manual backups are not automatically deleted when you delete a cluster.**

**Amazon Cloud Formation**

* AWS CloudFormation is a service that gives developers and businesses an easy way to create a collection of related AWS resources and provision them in an orderly and predictable fashion.
* AWS CloudFormation provides a common language for you to describe and provision all the infrastructure resources in your cloud environment.
* CloudFormation can be used to provision a broad range of AWS resources.
* Think of CloudFormation as deploying infrastructure as code.
* Elastic Beanstalk is more focused on deploying applications on EC2 (PaaS).
* CloudFormation can deploy Elastic Beanstalk-hosted applications however the reverse is not possible.
* Logical IDs are used to reference resources within the template.
* Physical IDs identify resources outside of AWS CloudFormation templates, but only after the resources have been created.

Concept of templates, stacks and change sets:



Templates:

* Architectural designs.
* Create, update and delete templates.
* Written in JSON or YAML.
* CloudFormation determines the order of provisioning.
* Don’t need to worry about dependencies.
* Modifies and updates templates in a controlled way (version control).
* Designer allows you to visualize using a drag and drop interface.

Stacks:

* Deployed resources based on templates.
* Create, update and delete stacks using templates.
* Deployed through the Management Console, CLI or APIs.

Template elements:

* Mandatory:
  + File format and version.
  + List of resources and associated configuration values.
* Not mandatory:
  + Template parameters (limited to 60).
  + Output values (limited to 60).
  + List of data tables.

**Amazon API Gateway**

* An Amazon API Gateway is a collection of resources and methods that are integrated with back-end HTTP endpoints, Lambda functions or other AWS services.
* API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale.
* API Gateway provides developers with a simple, flexible, fully managed, pay-as-you-go service that handles all aspects of creating and operating robust APIs for application back ends.
* API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.
* API calls include traffic management, authorization and access control, monitoring, and API version management.
* Together with Lambda, API Gateway forms the app-facing part of the AWS serverless infrastructure.
* Back-end services include Amazon EC2, AWS Lambda or any web application (public or private endpoints).
* CloudFront is used as the public endpoint for API Gateway.
* Supports API keys and Usage Plans for user identification, throttling or quota management.
* Using CloudFront behind the scenes, custom domains, and SNI are supported.
* Can be published as products and monetized on AWS Marketplace.
* Collections can be deployed in stages.
* Permissions to invoke a method are granted using IAM roles and policies or API Gateway custom authorizers.
* An API can present a certificate to be authenticated by the back-end.
* All of the APIs created with Amazon API Gateway expose HTTPS endpoints only (does not support unencrypted endpoints).
* By default API Gateway assigns an internal domain that automatically uses the API Gateway certificates.
* When configuring your APIs to run under a custom domain name you can provide your own certificate.
* Supported data formats include JSON, XML, query string parameters, and request headers.

Can enable Cross Origin Resource Sharing (CORS) for multiple domain use with Javascript/AJAX:

* Can be used to enable requests from domains other than the APIs domain.
* Allows the sharing of resources between different domains.
* The method (GET, PUT, POST etc) for which you will enable CORS must be available in the API Gateway API before you enable CORS.
* If CORS is not enabled and an API resource received requests from another domain the request will be blocked.
* Enable CORS on the APIs resources using the selected methods under the API Gateway.

Data types used with API Gateway:

* Any payload sent over HTTP (always encrypted over HTTPS).
* Data formats include JSON, XML, query string parameters and request headers.
* You can declare any content type for your APIs responses, and then use the transform templates to change the back-end response into your desired format.

You can add caching to API calls by provisioning an Amazon API Gateway cache and specifying its size in gigabytes.

**ENDPOINTS**

An [*API endpoint*](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-basic-concept.html#apigateway-definition-api-endpoints) type refers to the hostname of the API.

The API endpoint type can be *edge-optimized*, *regional*, or *private*, depending on where the majority of your API traffic originates from.

**Edge-Optimized Endpoint:**

* An edge-optimized API endpoint is best for geographically distributed clients. API requests are routed to the nearest CloudFront Point of Presence (POP). This is the default endpoint type for API Gateway REST APIs.
* Edge-optimized APIs capitalize the names of HTTP headers (for example, Cookie).
* CloudFront sorts HTTP cookies in natural order by cookie name before forwarding the request to your origin. For more information about the way CloudFront processes cookies, see Caching Content Based on Cookies.
* Any custom domain name that you use for an edge-optimized API applies across all regions.

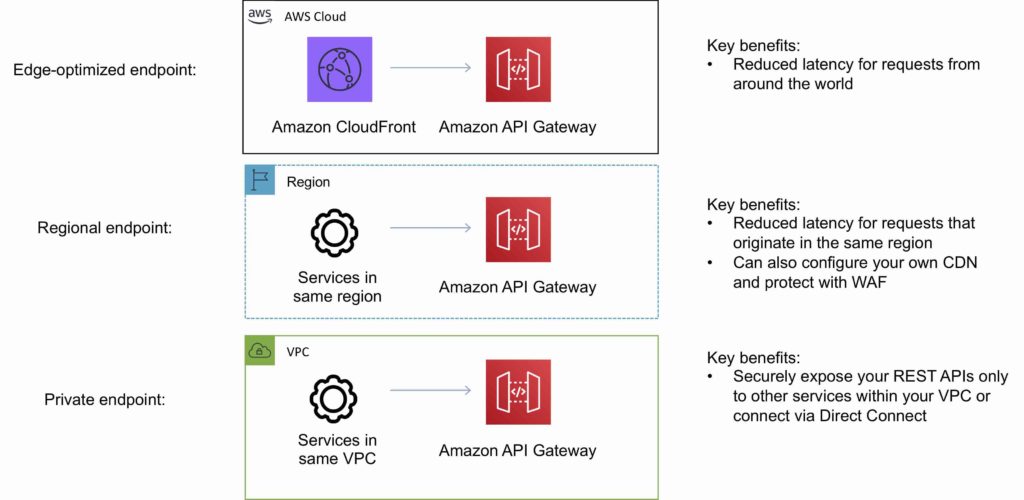
**Regional Endpoint:**

* A regional API endpoint is intended for clients in the same region.
* When a client running on an EC2 instance calls an API in the same region, or when an API is intended to serve a small number of clients with high demands, a regional API reduces connection overhead.
* For a regional API, any custom domain name that you use is specific to the region where the API is deployed.
* If you deploy a regional API in multiple regions, it can have the same custom domain name in all regions.
* You can use custom domains together with Amazon Route 53 to perform tasks such as latency-based routing.
* Regional API endpoints pass all header names through as-is.

**Private Endpoint:**

* A private API endpoint is an API endpoint that can only be accessed from your Amazon Virtual Private Cloud (VPC) using an interface VPC endpoint, which is an endpoint network interface (ENI) that you create in your VPC.
* Private API endpoints pass all header names through as-is.

The following diagram depicts the three different Amazon API Gateway endpoint types:



**ADDITIONAL FEATURES AND BENEFITS**

API Gateway provides several features that assist with creating and managing APIs:

* **Metering** – Define plans that meter and restrict third-party developer access to APIs.
* **Security** – API Gateway provides multiple tools to authorize access to APIs and control service operation access.
* **Resiliency** – Manage traffic with throttling so that backend operations can withstand traffic spikes.
* **Operations Monitoring** – API Gateway provides a metrics dashboard to monitor calls to services.
* **Lifecycle Management** – Operate multiple API versions and multiple stages for each version simultaneously so that existing applications can continue to call previous versions after new API versions are published.

API Gateway provides robust, secure, and scalable access to backend APIs and hosts multiple versions and release stages for your APIs.

* You can create and distribute API Keys to developers.
* Option to use AWS Sig-v4 to authorize access to APIs.
* You can throttle and monitor requests to protect your backend.
* API Gateway allows you to maintain a cache to store API responses.
* SDK Generation for iOS, Android and JavaScript.
* Reduced latency and distributed denial of service protection through the use of CloudFront.
* Request/response data transformation and API mocking.
* Provides Swagger support.
* Resiliency through throttling rules based on the number of requests per second for each HTTP method (GET, PUT).
* Throttling can be configured at multiple levels including Global and Service Call.
* A cache can be created and specified in gigabytes (not enabled by default).
* Caches are provisioned for a specific stage of your APIs.
* Caching features include customizable keys and time-to-live (TTL) in seconds for your API data which enhances response times and reduces load on back-end services.
* API Gateway can scale to any level of traffic received by an API.

**Logging and Monitoring**

* The Amazon API Gateway logs (near real time) back-end performance metrics such as API calls, latency, and error rates to CloudWatch.
* You can monitor through the API Gateway dashboard (REST API) allowing you to visually monitor calls to the services.
* API Gateway also meters utilization by third-party developers and the data is available in the API Gateway console and through APIs.
* Amazon API Gateway is integrated with AWS CloudTrail to give a full auditable history of the changes to your REST APIs.
* All API calls made to the Amazon API Gateway APIS to create, modify, delete, or deploy REST APIs are logged to CloudTrail.
* Charges
* With Amazon API Gateway, you only pay when your APIs are in use.
* There are no minimum fees or upfront commitments.
* You pay only for the API calls you receive and the amount of data transferred out.
* There are no data transfer out charges for Private APIs (however, AWS PrivateLink charges apply when using Private APIs in Amazon API Gateway).
* Amazon API Gateway also provides optional data caching charged at an hourly rate that varies based on the cache size you select.
* The API Gateway free tier includes one million API calls per month for up to 12 months.

**Session Management**

<https://aws.amazon.com/caching/session-management/>