|  |
| --- |
|  |

Premium support: Basic, Developer, Business, Enterprise

I have basic

Max response time for Business level Premium support case: 1 hour

https://console.aws.amazon.com/support/plans/home#/

FAQ:

<https://aws.amazon.com/faqs/>

Blogs, services, tips:

https://tutorialsdojo.com/aws-cheat-sheets/

<https://digitalcloud.training/>

# General Concepts

*Stateless Applications*: app that does not save client data generated in one [session](https://searchmicroservices.techtarget.com/definition/session) for use in the next session. Each session is carried out as if it was the first time and responses are not dependent upon data from a previous session. The server does not store any state about the client session. The session data is stored on the client and passed to the server as needed.

In contrast, a stateful application saves data about each client session and uses that data the next time the client makes a request

E.g. if a web server stores user data in a backend manner and uses it to identify you as a constantly connected client, the service is stateful.

*Systems Manager*

Resource Groups: Collection of resources that share one or more tags. Insights: dashboards, inventory, compliance. Actions: Automation, session manager, patch manager, and all kinds of shit

SSM parameter store: storage for configuration and secrets (KMS, passwords, and lots)

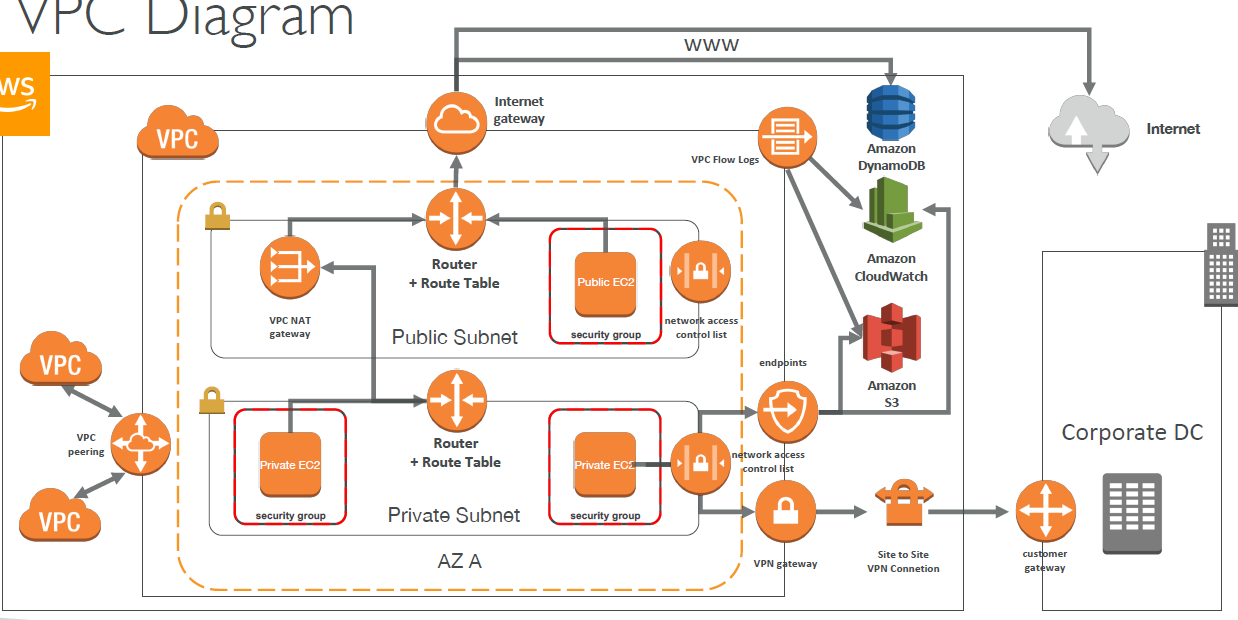
*AWS Organizations*

You can create groups of accounts, automate account creation, apply and manage policies for those groups. It allows you to create Service Control Policies (SCPs) that centrally control AWS service use across multiple AWS accounts

*SLA:*

Uptime Percentage of at least 99.99% for Amazon EC2 and Amazon EBS within a Region.

# VPC



Per VPC:

5 EIPs

5 VPC per region (request more)

200 subnets per VPC

500 SG per VPC

200 NACLs per VPC

Max CIDR per VPC is 5. For each CIDR:

• Min size is **/28** = 16 IP Addresses

• Max size is **/16** = 65536 IP Addresses

/28 = 16 2^(32-28)

/27 = 32 2^(32-27)

/26 = 64 2^(32-26)

CIDR: AWS reserve both the first 4 and last 1 IP addresses in each subnet’s CIDR block. Once a VPC is set to dedicated hosting, it is not possible to change the VPC or the instances to Default hosting. You must re-create the VPC

Default VPC includes an Internet gateway and each default subnet is a public subnet in each AZ. Each instance that you launch into a default subnet has a private IP address and a public IP address

**if a subnet in the a non-default VPC is not explicitly associated with a routing rule it uses the main rule set which does not have a route to the internet.**

Every **non default VPC** comes with a default SG to which newly launched instances is associated in case no other security group had been specified for them. Allows no inbound traffic (except to itself) and all outgoing traffic. It can’t be deleted

Also created a default NACL which allows all

Also creates a default route table with only a local route setup

No subnets, IGW, NAT Gateway are created

If **enableDnsHostname** is set to true, then it will assign public DNS name to EC2 instances like in a default VPC. Defaults to false. Should set this to true if you need **private Zone RT53** names to work

*ACL*

Numbered list of rules that evaluate in order, starting with the lowest numbered to determine whether traffic is allowed in or out of any subnet associated with the network ACL.

Each network ACL includes a rule \*. This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied. You can't modify or remove this rule.

The VPC default ACL allows ALL inbound and ALL outbound unless you change it

Have to associate subnets with custom ACLS!

When you create your own ACL ALL is DENIED!

ACLs can be across multiple subnets but subnets can only be bound to one ACL

Stateless: Return traffic must be explicitly allowed by rules. Need to allow both inbound and outbound traffic in order for EC2 instances in a network to be able to communicate over a particular protocol

*Subnet/Route Table*

ONE route table per subnet (uses main route table by default). Route table can be associated with MANY subnets

Each subnet is mapped to one AZ and CANNOT span across multiple AZ

Each subnet can communicate by default with the others in the VPC

Build the IGW and Attach to VPC

Need to create route table to connect to the IGW

Then add a route in the table with target being the IGW THEN associate subnets to the route.

*VPC Flow Logs*

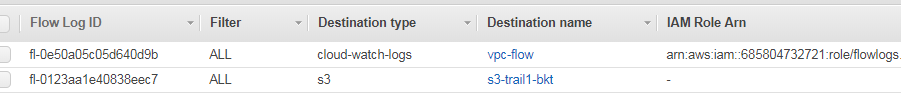
Can be created at the VPC, subnet, and network interface levels. You can set up a Flow Log for the group of instances and forward them to CloudWatch or S3

Flow Log Syntax

<version> <account-id> <interface-id> <srcaddr> <dstaddr> <srcport><dstport> <protocol> <packets> <bytes> <start> <end> <action> <logstatus>

Query VPC flow logs using Athena on S3 or CloudWatch Logs Insights

<https://docs.aws.amazon.com/athena/latest/ug/vpc-flow-logs.html>



*ENI*

An ENI in one subnet can be attached to an instance in another subnet in the same AZ

ENI can be attached to an instance when it’s running (hot attach), when it’s stopped (warm attach), or when the instance is being launched (cold attach)

## VPC Endpoints

Virtual Devices allowing a private connection between your VPC and another AWS service without requiring access over the Internet, or VPN connection

*Interface Endpoints*

Uses AWS PrivateLink and is an ENI with a private IP address that serves as an entry point for traffic destined to a supported service


     Using an interface endpoint to access an AWS service
    

*Gateway VPC Endpoints*

https://docs.aws.amazon.com/vpc/latest/userguide/vpce-gateway.html

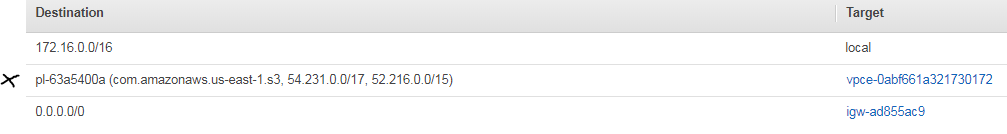
Private connection between your VPC and another AWS service using its private IP address.

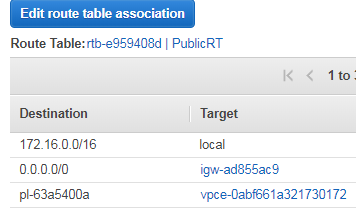
Single Region

Supports only S3 and DynamoDB at this time

*Scenario for public instance:*

Create the endpoint to s3: A rule with destination pl-63a5400a (com.amazonaws.us-east-1.s3) and a target with this endpoints' ID will be added to the route table associated with the subnet of our control instance (subnet-public)



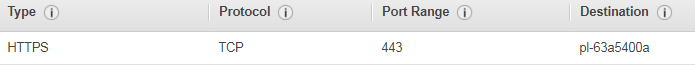


It will also generate a policy by default: Full Access - Allow access by any user or service within the VPC using credentials from any AWS accounts to any resources in this AWS service.

Can optionally create a custom policy

Change the SG of control instance to sshSG that only allows inbound ssh and outbound nothing. This will REMOVE access to S3

Add an outbound rule to this SG to allow HTTPS (s3 uses that) to the s3 service prefix name. Can also get that from the [describe-prefix-lists](https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-prefix-lists.html) command

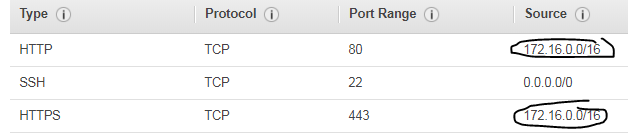


## NAT

*Instance*

Need an SG that allows public to private inbound and outbound rules for the NAT instance: must allow HTTP/HTTPS inbound from the VPC ONLY and outbound to 0.0.0.0/0

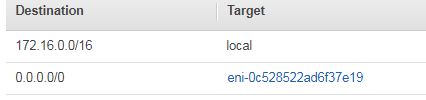
NatSG (sg-08f8d035d3d6b2e67)



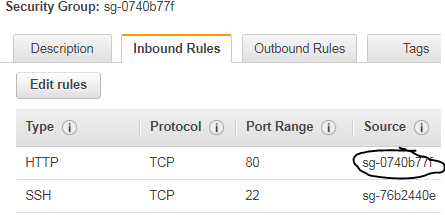
Configure a route table in private subnet: add route for all traffic to target the NAT instance with destination 0.0.0.0/0

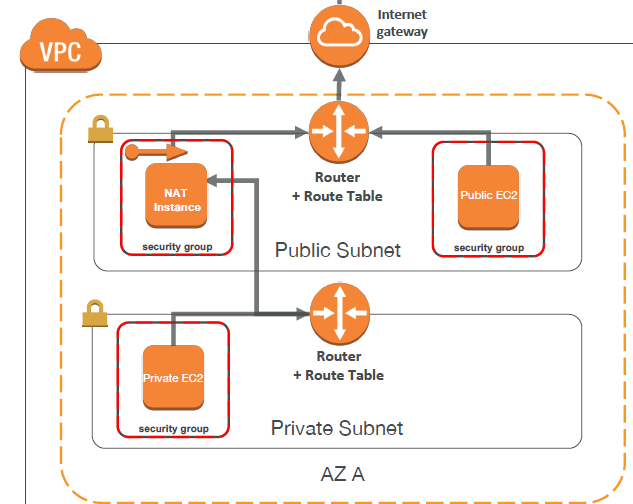
This interface on the NAT server should have a static EIP, although DHCP will still work

rtb-08bb7628314131286:



Private instance SG must have a rule to allow inbound port 80 etc. with the source being the private subnet CIDR (anywhere will work too). SSH allowed from Bastion SG

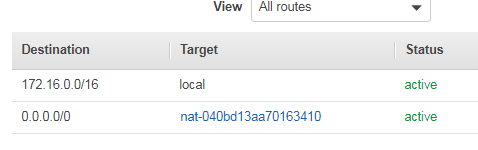




*NAT gateways*

Create the NAT gateway on the public subnet (creates an EIP)

Add a route on the private RT and associate to the private sub with rule (0/0) to the Nat gateway



They are not associated with SGs like NAT instances and you want them in multi AZs for redundancy. They are more secure than NAT instance

Can scale up to 45Gpbs automatically

*Bastion*

Deploy a Bastion host in a public subnet within each Availability Zone for HA

Allows you to login to instances in the Private subnet securely without having to store the private keys on the Bastion host (using ssh-agent forwarding or RDP gateways)

All the Instances in the private subnet should be hardened to accept SSH/RDP connections ONLY from the Bastion host

*Egress-Only Internet Gateway*

Egress only Internet Gateway is for **IPv6** only. Similar to a NAT

IPv6 are all public addresses, all our instances with IPv6 are publicly accessible

Gives our IPv6 instances access to the internet, but they won’t be directly reachable by the internet

After creating an Egress Only Internet Gateway, edit the route tables

## VPN

https://docs.aws.amazon.com/vpc/latest/userguide/vpn-connections.html

*Site to Site*


      Using a virtual private gateway
     

Virtual Private Gateway

You can enable access to your network from your VPC with a VPN by attaching a VPG (each connection is secured by a pre-shared key in conjunction with the IP address of the customer gateway device) to the VPC, creating a custom route table, and updating your security group rules.

Customer gateway

A physical device or software application on your side of the VPN connection. Assign a public, static, and internet routable IP address on the customer gateway external interface for the on-premise network:

If behind NAT, use the public IP address of the NAT

When you create a VPN connection, the VPN tunnel comes up when traffic is generated from your side of the VPN connection. When you use a BGP device, you don't need to specify static routes to the VPN connection because the device uses BGP to advertise its routes to the virtual private gateway. Best practice to setup redundant VPN connections: Each VPN connection has 2 tunnels, with each tunnel using a unique virtual private gateway public IP address. When one tunnel becomes unavailable (down for maintenance), network traffic is automatically routed to the available tunnel for that specific VPN connection.

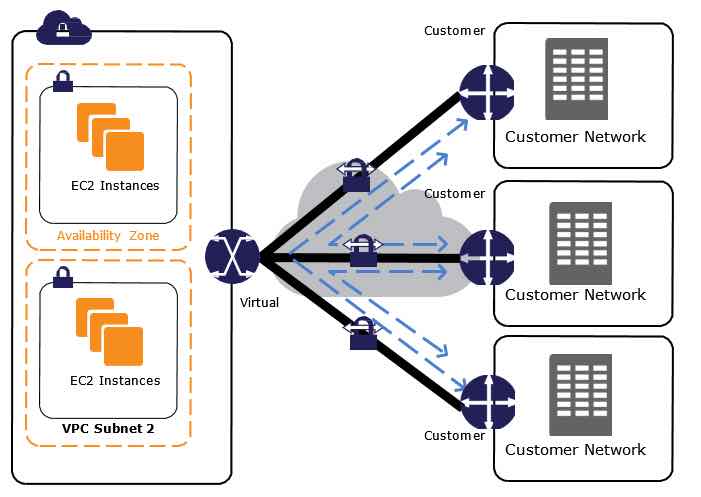
*Software VPN*

You fully manage both sides of your Amazon VPC connectivity by creating a VPN connection between your remote network and a software VPN appliance running in your VPC

*VPN CloudHub*

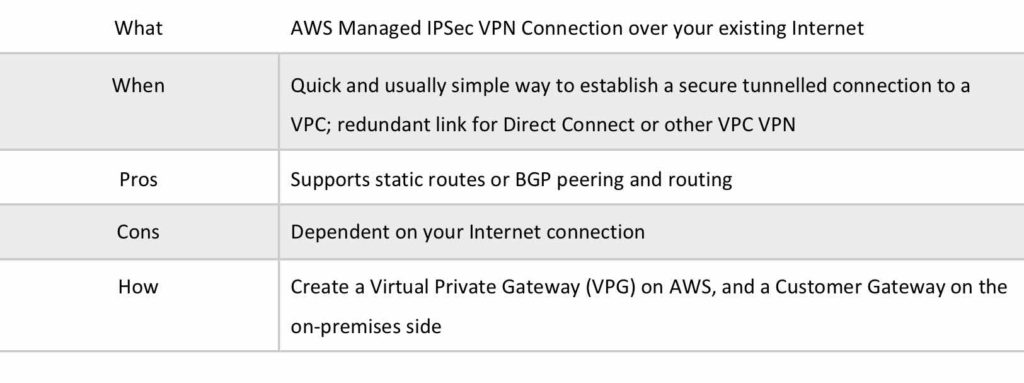
Operates on a simple hub-and-spoke model that you can use with or without a VPC

Use this design if you have multiple branch offices and existing internet connections and would like to implement a convenient, potentially low cost hub-and-spoke model for primary or backup connectivity between these remote offices



*AWS Managed*

Provides the option of creating an IPsec VPN connection between remote customer networks and their VPC over the internet Consider taking this approach when you want to take advantage of an AWS managed VPN endpoint that includes automated multi–data center redundancy and failover built into the AWS side of the VPN connection



## VPC peering

Can create connection cross account and cross region:

Data sent between VPCs in different regions is encrypted (traffic charges apply). **You must update route tables in each VPC’s subnets to ensure instances can communicate**. You may need to update the inbound and outbound rules for the security group to reference security groups in the peered VPC ie

When creating a VPC peering connection with another account you need to enter the account ID and VPC ID from the other account

Not transitive, always 1 to 1. Max of 50 VPCs. No single point of failure because it uses the existing AWS network infrastructure.

Instances behave as they were on one VPC star configuration. Can do cross account peering as well. Can’t overlap CIDR blocks.

Cannot do **edge to edge** routing:

If either VPC in a peering relationship has one of the following connections, you cannot extend the peering relationship to that connection:

A VPN connection or an AWS Direct Connect connection to a corporate network

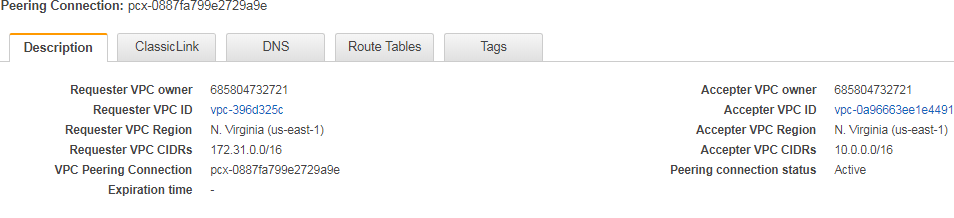
An internet connection through an internet gateway

An internet connection in a private subnet through a NAT device

A VPC endpoint to an AWS service; for example, an endpoint to Amazon S3.

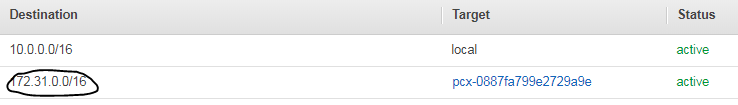
*My working Setup*:

Peer Default PVC to a test VPC:



Edit the main route tables of both VPCs:





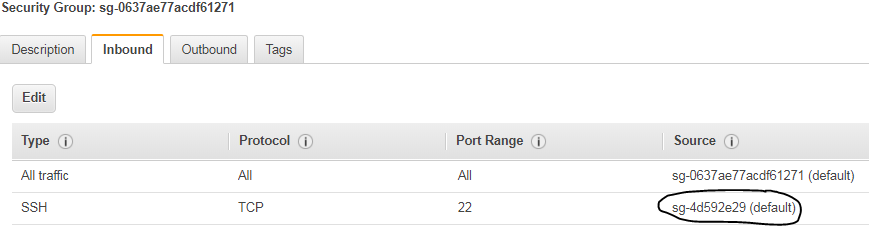
The default VPC has an instance on a public subnet and the test VPC has an instance on a nonpublic subnet (the test VPC has no IGW associated), so we need to adjust the SG of the test VPC, nonpublic subnet

To allow the SG of the Default VPC public instance so we can then ssh from the public instance on the default to the nonpublic on the test



Default VPC SG: sg-4d592e29

Test VPC SG: sg-0637ae77acdf61271



## Direct Connect

1 or 10 GB speed using industry standard 802.1q VLANs

Links your internal network to a Direct Connect location over fiber-optic cable using **a Virtual Private Gateway.** With this connection, you can create *virtual interfaces* directly to public AWS services (e.g. Amazon S3) or to another VPC, bypassing internet service providers in your network path while maintaining network separation between the public and private environments

Ensure you have a route connecting your VPC back to the on premise data center

Can take months to get it set up

*Direct Connect Gateway:* If you want to setup a Direct Connect to one or more VPC in many different regions (same account). The VPCs are not peered or connected to each other. It’s one to many from your DCG

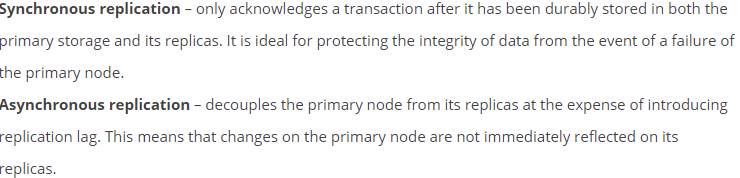

    AWS Direct Connect
  ­

## Direct Connect Plus VPN

Can combine one or more AWS Direct Connect dedicated network connections with the VPN

This combination provides an IPsec-encrypted private connection that also reduces network costs, increases bandwidth throughput, and provides a more consistent network experience than internet-based VPN connections

# Storage



## S3

*General*

Can have 100 buckets per account

Read after write for consistency for puts of new objects

Eventual consistency for overwrite puts and deletes (will be eventually consistent across facilities)

S3 Standard, S3 Standard-IA, and S3 Glacier replicate data across a minimum of three AZs to protect against the loss of one entire AZ.

5TB limit per object, 0 byte min, unlimited storage. A single PUT limited to 5GB – use Multi Part if bigger

Charges are storage, requests, data xfer

Data transferred to amazon EC2 from Amazon S3 in the same region is free

Object: https://s3.amazonaws.com/name/object

Bucket URLS:

Vhost style: <http://name.s3-region.amazonaws.com>

If don’t specify region, it routes to US-east-1 by default: http://name.s3.amazonaws.com

Path style: https://s3-region.amazonaws.com/name

For us-east-1: https://s3.amazonaws.com/name

*Performance:*

Now provides increased performance to support at least 3,500 requests per second to put data and 5,500 requests per second to retrieve data.

S3 support for parallel requests means you can scale your S3 performance by the factor of your compute cluster, without making any customizations to your application. Performance scales per prefix, so you can use as many prefixes as you need in parallel to achieve the required throughput. There are no limits to the number of prefixes.

This S3 request rate performance increase removes any previous guidance to randomize object prefixes to achieve faster performance. That means you can now use logical or sequential naming patterns in S3 object naming without any performance implications. This improvement is now available in all AWS Regions.

*Hosted Web Site*

Hosted web site: http://name.s3-website-region.amazonaws.com or you can use RT53 alias record. If you use custom domain name, bucket must use that same name

E.g. http://hello.thebarncat.net.s3-website-us-east-1.amazonaws.com/

*CORS*:

On s3 hosted web site a browser would normally block JavaScript from allowing those requests, but with CORS, you can configure your bucket to explicitly enable cross-origin requests

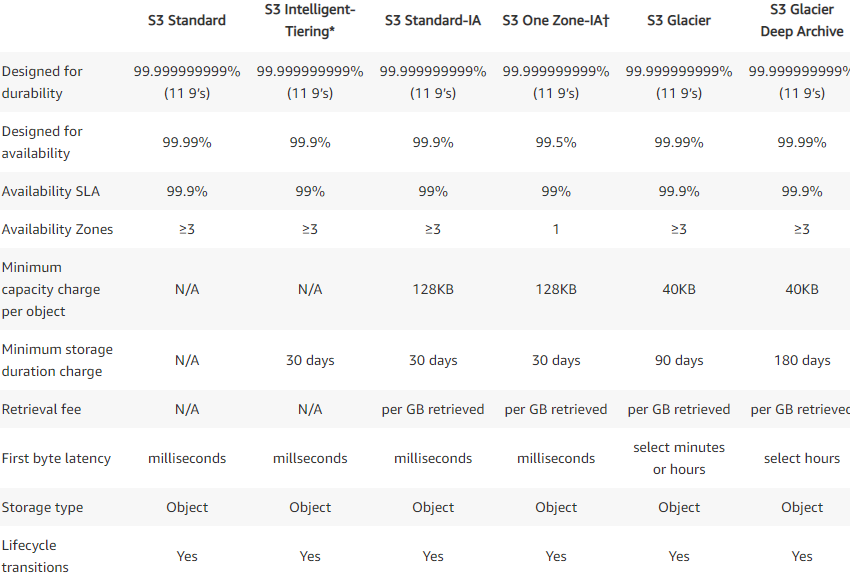
If you request data from another S3 bucket, CORS needs to be enabled. It’s configured in the Permission section of the bucket

*Cross Region*  
new objects and new VERSIONs will replicate from source to target bucket. Use an IAM role for this with a policy attached. Copying is **asynchronou**s.

*Storage Tiers*

S3-IA (infrequent access) tier, cheaper, charges per retrieval and can retrieve right away.

One Zone IA. Costs 20% less than S3 Standard-IA with slight less availability than standard IA. Lower-cost option for infrequently accessed data but does not require the availability and redundancy of S3 Standard or S3-IA. Good choice for storing secondary backup copies of on-premises data or easily re-creatable data. 99.95% Availability.



Note RRS only offer 99.99 Durability

*Glacier*

To retrieve data from Glacier, you initiate a retrieval job via S3 REST API or can use the S3 console.

E.g.: aws s3api restore-object

Cost per storage per month ($0.004 / GB) + retrieval cost

Once it completes, your data will be available to download for **24 hours**

* Expedited is 1-5 minutes retrieval (most expensive)
* Standard is 3.5 hours retrieval (cheaper, 10GB data retrieval free per month)
* Bulk retrieval is 5-12 hours (cheapest, use for large quantities of data)

Can retrieve up to 5% of your average monthly storage, pro-rated daily, for free each month. Retrieve data using http get request. It automatically encrypts the data using AES256 and stores it durably in an immutable form.

**Glacier stores data encrypted at rest in S3 or Glacier using respective server side encryption by default**

*Encryption Data at Rest options*

Server-side encryption: S3 encrypts the object data before saving it on disks in its data centers and decrypts it when the user downloads the objects

Client-side encryption: refers to encrypting data before sending it to Amazon S3.

Server Side Encryption at rest:

- S3 managed keys (**SSE-S3**). Must set header: “x-amz-server-side-encryption": "AES256". HTTP/S + header

- Keys stored in S3’s key management service (**SSE-KMS**) which is a bit more secure and has logging/audit features. Uses CMK as the key. You have control of the rotation policy.

Must set header: “x-amz-server-side-encryption": "aws:kms”

- Customer-provided and managed keys (**SSE-C**). HTTPS must be used when sending to S3. Data key stored in header. S3 manages both the encryption, as it writes to disks, and decryption, when you access your objects

Client side

-Use AWS KMS–Managed Customer Master Key (CMK)

-Use Client-Side Master Key

-Can use client library such as Amazon S3 Encryption Client or your own before uploading to S3.

-Customer fully manages the keys and encryption cycle

You can Import your own keys, disable and re-enable keys and define key management roles in IAM

You can view the default and custom keys in the KMS console

Can now use **DEFAULT encryption** option on the bucket – newer feature and you can now encrypt existing shit.

*Versioning*

Version ID: all objects have an ID, so all versions of a file take up space

Delete the "delete marker" file to restore after versioning turned on

Restore later versions to go back to previous versions

Can now use MFA for delete for: Change the versioning state of your bucket and permanently delete an object version. Done via CLI

Only the bucket owner (root account) can enable/disable MFA-Delete

*Life Cycle Management*

Have to wait 30 days before sending to IA and 30 days after that if then send to glacier. Files of at least 128k

Transition actions: It defines when objects are transitioned to another storage class.

E.g.: We can choose to move objects to Standard IA class 60 days after you created them or can move to Glacier for archiving after 6 months

Expiration actions: Helps to configure objects to expire (delete) after a certain time period and can clean up multipart uploads

When you choose the GLACIER storage class, your objects remain in S3. You cannot access them directly through the separate S3 Glacier service.

*Events:*

Event Notifications enable you to send alerts or trigger workflows. Notifications can be sent via SNS, SQS, or to a Lambda function (depending on the bucket location).

*Redirects:*

<https://docs.aws.amazon.com/AmazonS3/latest/dev/how-to-page-redirect.html>

*Secure:*

**Bucket policies**

You grant permission to an AWS account by the email address or the canonical user ID

**Access Logs** log to another bucker or even another account: logs requester, bucket name, request time, request action, response status, and an error code. Can be analyzed with Athena

Grant the Amazon S3 Log Delivery group write permission on the bucket where you want the logs saved. The target bucket must be in the same Region as the source bucket and must not have a default retention period configuration.

Ways to control access to buckets: IAM policy, ACL, bucket policy, and:

Query string authentication (Signed URL): allows them to share Amazon S3 objects through URLs that are valid for a predefined period of time

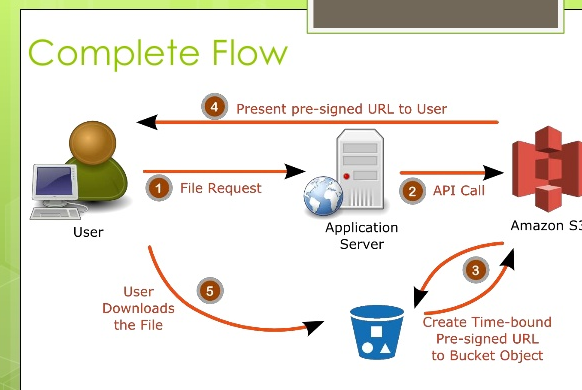
Use the ‘aws s3 presign’ command to generate pre signed URLs. Defaults to 3600 TTL and set the region. Then send link to user.

Examples:

Allow only logged-in users to download a premium video on your S3 bucket

Allow an ever changing list of users to download files by generating URLs dynamically

Allow temporarily a user to upload a file to a precise location in our bucket



*Transfer acceleration*

Fast, easy, and secure transfers of files over long distances. Leverages CloudFront globally distributed edge Locations

To use, need to enable transfer acceleration on the S3 bucket

*Pillar*

Operations: no operations needed

Security: IAM, Bucket Policies, ACL, Encryption (Server/Client), SSL

Reliability: 99.999999999% durability / 99.99% availability, Multi AZ, CRR

Performance: scales to thousands of read / writes per second, transfer acceleration / multi-part for big files

Cost: pay per storage usage, network cost, requests number

## Athena

Is an interactive query service that makes it easy to analyze data in S3, using SQL. It will work with “JSON", CSV, "Apache Parquet", "Apache ORC" amongst others, but "XML" is not supported

It is **serverless**, so there is no infrastructure to manage, and you pay only for the queries that you run

Use cases: Business intelligence / analytics / reporting, analyze & query VPC Flow Logs, ELB Logs, CloudTrail trails, etc...

*Pillar*

Operations: no operations needed, serverless

Security: IAM + S3 security. Query encrypted data stored in S3 and write encrypted results back to your S3 bucket

Reliability: managed service, uses Presto engine, highly available

Performance: queries scale based on data size

Cost: pay per query / per TB of data scanned

## SnowBall

*Snowball*

A device from amazon where u put your shit up to 80TB (72TB usable). Uses 256 bit encryption, can export from S3 or Import in to S3

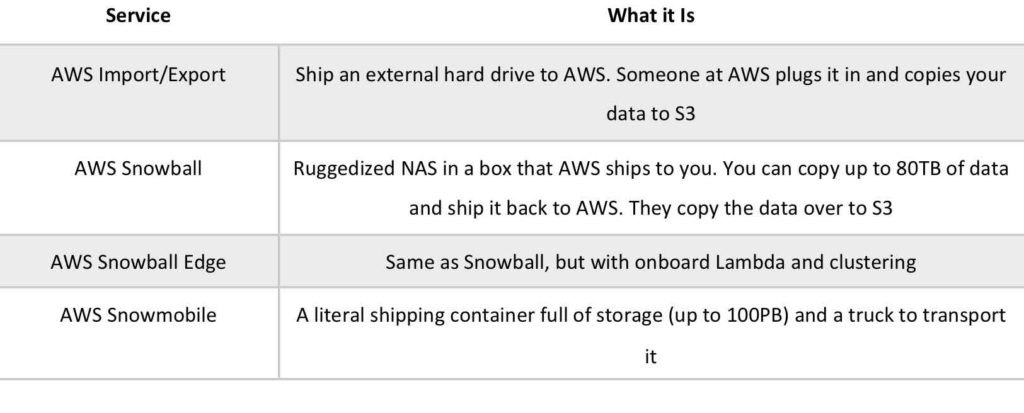
Edge:

100 TB (83TB useable) capacity with either:

Storage optimized 24 vCPU

Compute optimized 52 vCPU & optional GPU

Supports a custom EC2 AMI so you can perform processing on the go



1. Request snowball devices from the AWS console for delivery

2. Install the snowball client on your servers

3. Connect the snowball to your servers and copy files using the client

4. Ship back the device when you’re done (goes to the right AWS facility)

5. Data will be loaded into an S3 bucket

6. Snowball is completely wiped

7. Tracking is done using SNS, text messages and the AWS console

## Storage Gateway

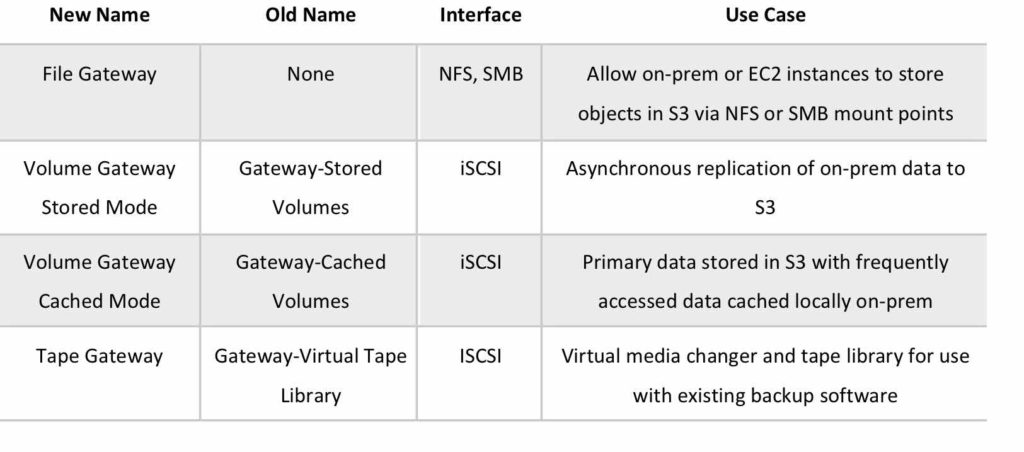
Connect on premise appliance to AWS you can download the storage GW software appliance. Transfers data to AWS over SSL by default

*File Gateway:* virtual on-premises file server (**NFS**), which enables you to store and retrieve files as objects in Amazon S3. Flat files only**. You can think of a file gateway as a file system mount on S3 on premise.**

*Gateway-stored*: volumes store your primary data **on premise**, while **asynchronously** backing up that data to S3 over SSLas EBS snaps. These volumes provide low-latency access to their **entire datasets**. Each volume can be up to 16 TB and mount them as iSCSI devices from your on-premises application servers

*Gateway-cached*: volumes allow you to utilize **S3 for your primary data**, while retaining some portion of it locally in a cache for frequently accessed data. You can create storage volumes up to 32 TBs (32 vols max) and mount them as iSCSI devices from your on-premises servers.

*Virtual Tape Lib*: collection of virtual tapes backed by s3 or glacier. Used for apps like NetBackup etc. With up to 10 virtual tape drives per gateway. Each virtual tape drive responds to the SCSI command set, so your existing on-premises backup applications will work without modification.



## EFS

Storage capacity is elastic, growing and shrinking automatically.

Uses NFSv4.1 protocol, so the applications and tools that you use today work seamlessly with EFS.

Pay only for the storage used by your file system but 3x more expensive than EBS.

EC2 instance uses security group to control access to EFS that allows NFS port with source being the SG itself

EFS file systems store data and metadata across multiple AZ in a region and can grow to petabyte scale, drive high levels of throughput, and allow massively parallel processing access from EC2 instances to your data.

Can be mounted from on-premises systems if using Direct Connect or a VPN connection

Performance mode:

General purpose (default)

Max I/O – used when thousands of EC2 are using the EFS

EFS file sync: sync from on-premise file system to EFS

EFS-to-EFS: automatically creates incremental backups of an EFS file system on a customer-defined schedule.

EFS now has Life-cycle management and Infrequent Access storage

# IAM

<https://685804732721.signin.aws.amazon.com/console>

EC2 Key pairs are NOT IAM objects. Don’t confuse with access keys

Access keys and secret are only used for API and CLI.

Attach policies to groups or roles

SO when we assign the role to the new EC2 instance (s3 admin). NO NEED TO RUN aws configure DONT NEED TO store sec creds

Roles easy to manage. IAM user, groups, roles are universal, use in any region.

E.g. need to change s3 access for 50 users: Use the IAM groups and add users as per their role to different groups and apply the policy to group

*AD integration and Federation*

Users outside of AWS assume identity provided access role. Federation assumes a form of 3rd party authentication LDAP, AD, etc.:

Know these docs and diagrams

<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml.html>

<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_enable-console-saml.html>

Paths in IAM have informative purposes only, thus no boundaries are enforced between users or groups based on their paths

If u want URL of the AWS IAM sign-in page to have a company name instead of the AWS account ID, can create an alias for his AWS account ID.

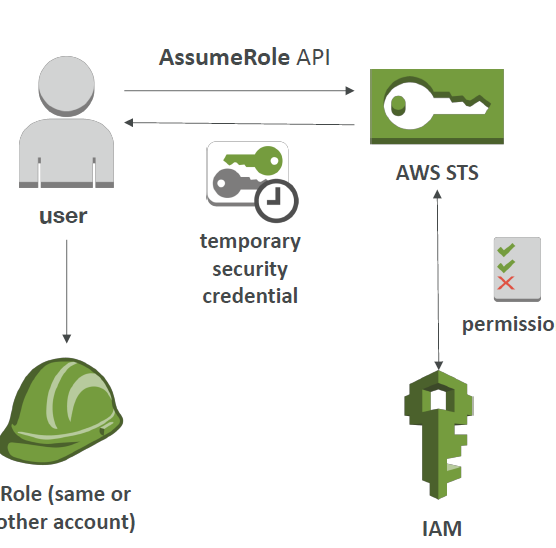
Each user in Amazon IAM can be part of no more than 10 groups.

Power user: access to all AWS services except the management of groups and users

Can have up to 5000 users per account

*Credential report*: CSV lists all users in your account and the status of their various credentials, including passwords, access keys, MFA devices, and signing certificates

*Cross Account*



# EC2

curl http://169.254.169.254/latest/meta-data/

curl <http://169.254.169.254/latest/user-data>

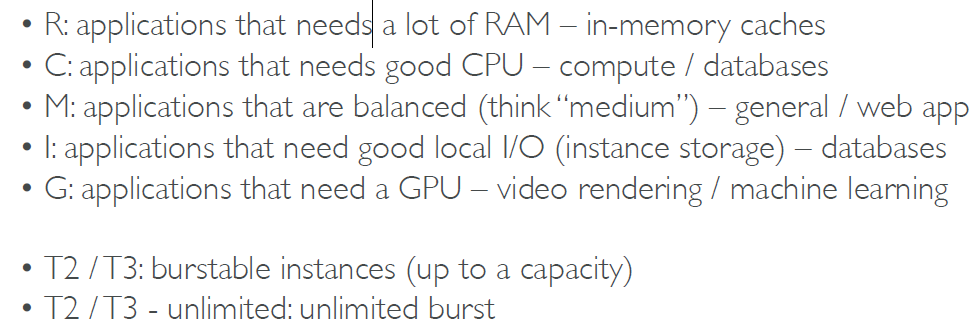
Hypervisor: Xen and now Nitro

Get charged for EIP if not associated with a running instance, or if it is associated with a stopped instance or an unattached network interface

Limit of 20 instances per region

## Instance types

<https://aws.amazon.com/ec2/instance-types/>



Memory Optimized also includes z-type, General purpose now includes “A” type

F, G, P for accelerated computing (GPU, enhanced networking- provides higher bandwidth, higher packet-per-second (PPS) performance, and consistently lower inter-instance latencies, EBS optimized)

T2/T3 instances are burstable – blows load and sputters. It’s unstable. Uses burst credits which accrue over time. Can see credit usage and balance in CloudWatch or the monitoring tab of the instance

<https://www.ec2instances.info/>

## Security Groups

SGs supports only allow rules inbound and outbound.

SG evaluates all rules before deciding whether to allow traffic.

SG are **stateful:** return traffic is automatically allowed, regardless of any outbound rules.

SGs can span across subnets and AZs

SGs operate at the instance level

**New SG by default all inbound traffic denied, outbound allowed**

## EBS

Virtual Network Drives block devices

Can attach a volume to any instance in the **same AZ**.

*Vol Types*

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html> **#STUDY THIS WELL**

General (**gp2**), 99.999% avail. The size affects the IOPS, You can see this when creating a gp2 EBS vol.1GiB - 16TiB. Provides 3 IOPS per GB up to 16,000 IOPS

Burstable up to a point until IOPS hits 3K (1TB). 16K IOPS for volume sizes greater than 5333 GiB

**You CAN’T modify IOPS of a GP2** only increase the size to get more IOPS

Provisioned IOPS SSD (**io1**) for IO intensive server. For applications that require sustained IOPS performance. When u need > 16K IOPS. 4GiB - 16TiB. Max IOPS 64K

50:1G is the maximum ratio of provisioned IOPS to requested volume size in GB

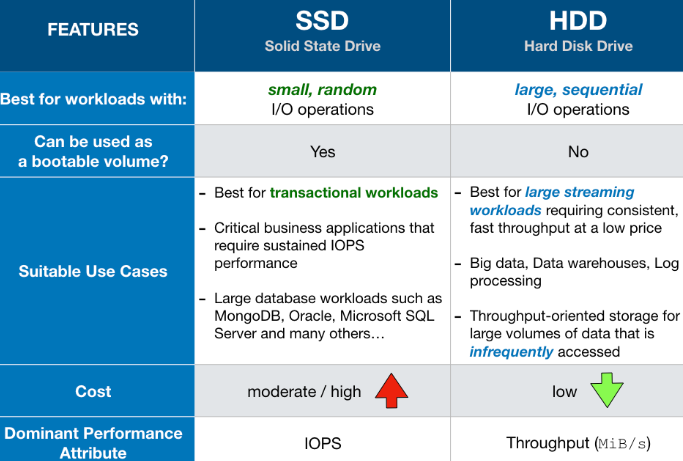
For instance, a 10 GiB volume can be provisioned with up to 500 IOPS. A volume 640 GiB in size allows up to 32,000 IOPS (50 × 640 GiB = 32,000). Can’t go past the max of 64K

Only GP2 and io1 can be used as boot volumes

Throughput Optimized HDD (**st1**): Use case big data, DW, log analyses.

Cold HDD (**sc1**): for large volumes of data that is infrequently accessed

EBS optimized: optimized configuration stack and provides additional, dedicated capacity for EBS I/O. certain instance types support it



*Vol Stuff*

Possible to Increase volume size, adjust performance, or change the volume type while the volume is in use

*Replication, Snapshot, Encryption*

When you create an EBS volume in an AZ, it is automatically replicated within that zone. When you create an EC2 instance, a **public** snapshot is created for the root volume.

Creating a volume manually does not create a snapshot by default.

Snapshot will **asynchronousl**y copy the data modified on the EBS volume. It does not copy the whole volume or all the data written to it, but just the modified data..

Snapshots are stored in S3 and you can copy it from one AWS region to another and **encrypt** while copying

Snapshot of a volume is available across all the AZ within a region and you can use an EBS snapshot to create new EBS volumes in any AZ in the region (you cannot encrypt while do this, see below)

Recommended to stop instance if creating snap of the root device but not mandatory.

EBS volumes restored by snapshots need recommended to be pre-warmed (using fio or dd command to read the entire volume)

Snapshots can be automated using Amazon Data Lifecycle Manager

Max 100,000 snapshots per account

You can use the default AWS KMS key or your own KMS key to encrypt

Create an **encrypted** volume from a snapshot of another encrypted volume.

Create an encrypted snapshot from an unencrypted snapshot by creating an encrypted copy of the unencrypted snapshot. You can then create an encrypted volume from this

Encryption is supported on all EBS volume types

To share an encrypted snapshot with another account: must share the CMK and use a custom CMK and grant permissions to the account or public

*RAID*

Amazon HATES RAID5/6 because parity writes cause bad performance.

RAID10 - striped and mirror, good performance

Snapshot of RAID array 3 ways to flush cache:

-freeze file system

-unmount it

-shut down the instance

## Instance store

Can’t stop/start the instance, reboot or terminate only. Cannot resize the device. You can specify the instance store vols **only when you launch an instance**

Root device for instance launched from the AMI is an instance store volume created from template stored in s3.

If host fails, u lose the data, can’t move the root device. Can share unencrypted snapshots only

Faster than EBS

Ideal for temporary storage of information that changes frequently, such as buffers, caches, scratch data

## AMI

If you create image from instance it will create a snapshot and stored in s3

AMIs are regional but you can copy to other regions

Must deregister AMI before deleting root device

Pricing is based on S3 storage

Cross Account Copy: You can't copy an encrypted AMI that was shared with you from another account. Instead, if the underlying snapshot and encryption key were shared with you, you can copy the snapshot while re encrypting It with a key of your own. You own the copied snapshot, and can register it as a new AMI.

You can't copy an AMI with an associated billingProduct code that was shared with you from another account. This includes Windows AMIs and AMIs from the AWS Marketplace. To copy a shared AMI with a billingProduct, launch an EC2 instance in your account using the shared AMI and then create an AMI from the instance

<https://aws.amazon.com/blogs/security/how-to-create-a-custom-ami-with-encrypted-amazon-ebs-snapshots-and-share-it-with-other-accounts-and-regions/>

## Instance Launch Types

There is a soft limit of 20 instances per region for new accounts.

*On demand* – charged by the hour with billing per second, after first minute

*Reserved* - 1 or 3 year term. Capacity reserve with up to a 75% discount from on demand.

It is possible to transfer a reserved instance from one Availability Zone to another but NOT region

You can terminate reserved instance to avoid any data transfer charges that the instance might incur and sell the reserved instance in the AWS Reserved Instance Marketplace to recuperate cost.

2 Types: *Standard and Convertible*: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/reserved-instances-types.html>

Convertible ~ 54% discount

Standard: instance family cannot be modified. You cannot exchange a Standard Reserved Instance, only modify it

Convertible can be exchanged during the term for another Convertible Reserved Instance with new attributes including instance family, instance type, platform, scope, or tenancy

*Scheduled Reserved*: purchase capacity reservations that recur on a daily, weekly, or monthly basis, with a specified start time and duration, for a one-year term. You reserve the capacity in advance, so that you know it is available when you need it. You pay for the time that the instances are scheduled, even if you do not use them.

Use Case: DB instances

*SPOT:* set bid price for what u want to pay and wait for the SPOT price. No guarantee. AWS automatically terminates the instance when the Spot price exceeds your maximum price.

If your instance is terminated by AWS in the FIRST hour, you won’t be charged for that hour. You are charged by the hour including partial hours. E.g. 90 minute usage at spot price of $0.04 would be $0.04 + $0.02 (30 minutes) = $0.06

2 minute warning if the instance is to be terminated to save any unsaved work. You can specify whether EC2 **should hibernate, stop, or terminate Spot Instances when they are interrupted**. The default is to terminate.

Use cases: Big Data, batch, non-prod work loads

*Scheduled:*

Allow you to reserve Amazon EC2 instances on a recurring schedule. You can purchase daily, weekly, or monthly reservations

## Dedicated Hosts and Instances

Instance runs on a dedicated Host, an isolated server with configurations that you can control. Complete isolation

You cannot change the tenancy of a default (shared) instance after you've launched it.

You can change the tenancy of a dedicated (instance vs host) instance by stopping and starting it: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/how-dedicated-hosts-work.html#moving-instances-dedicated-hosts>

Each VPC has a related instance tenancy attribute. You can't change the instance tenancy of a VPC after you create it.

You can consistently deploy your instances to the same physical server over time which enable you to use your existing server-bound software licenses and address corporate compliance and regulatory requirements.

*Dedicated Instance:* may share hardware within same account but not with other customers. No control over instance placement (may move around)

## Placement group

A placement group can span peered VPCs in the same Region. Placement groups can’t be merged. Names must be unique within your AWS account for the Region

Existing instance can now be moved to a placement group via CLI, or moved from one placement group to another, or removed from a placement group, given it is in the stopped state

*Clustered*

Logical grouping of instances in a Low-latency, 10 Gbps network. I.e. all nodes within the placement group can talk to all other nodes within the placement group at the full line rate of 10 Gbps

Clustered is all about performance not availability. Choose homogenous and high network performing instance types. Can’t use T class instance types for example for clustered placement.

Good for big data stuff

In case of a capacity error, stop and start all of the instances in the placement group, and try the launch again. This may migrate them to hardware that has the capacity

*Spread*

Placed on distinct (different) underlying hardware and span multiple AZ, and you can have a maximum of 7 running instances per AZ per group

Reduces the risk of failures that might occur when instances share the same underlying hardware and provide access to distinct hardware, and are therefore suitable for mixing instance types or launching instances over time. This is not for PERFORMANCE

*Partition*

Spreads instances across many different partitions (sets of racks) within an AZ. Scales to 100s of EC2 instances per group (Hadoop, Cassandra, Kafka)

## ECS/ECR

**ECR**:

A regional registry and repository that stores AWS managed and your own Docker container images and are stored in S3. The registry requires authentication and can be tied to IAM using policies to grant permissions to ECR resources and operations

Sent over HTTPS to the default registry URL and then encrypted at rest

ECR LC policies enable you to specify LC management of images in a repo

**ECS:**

ECS Core: Running ECS on user-provisioned EC2 instances

Fargate: Running ECS tasks on AWS-provisioned compute (serverless)

Container **service** to run, stop, and manage Docker containers in a cluster

No additional charge for ECS. You pay for AWS resources (e.g. EC2 instances or EBS volumes)

Grant Security privs to Docker containers: role granted to ECS Container Instances and role granted to an ECS Task

ECS cluster: region specific logical grouping of tasks or services. If you are running tasks or services that use the EC2 launch type, a cluster is also a grouping of EC2 container instances.

Contain tasks using both the Fargate and EC2 launch types. For tasks using EC2 launch type, clusters can contain multiple different container instance types, but each container instance may only be part of one cluster at a time

ECS tasks: Task definition is a parameter file written in json. Describes containers for your applications such as IAM task role, network, volumes, etc.

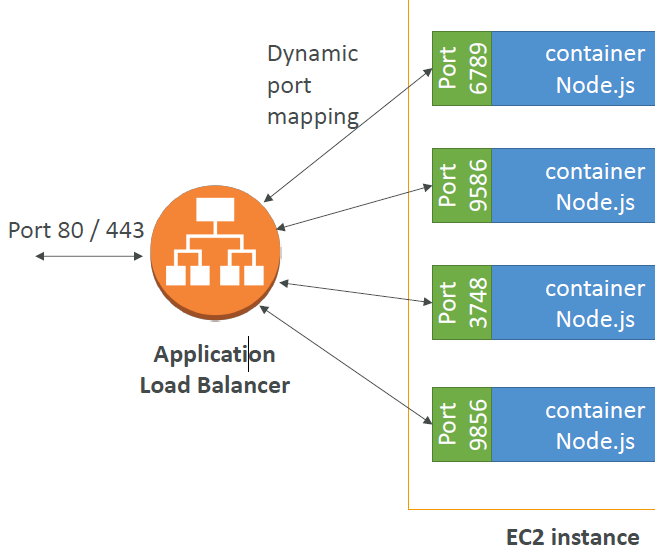
ALB has a direct integration feature with ECS called port mapping. Allows you to run multiple instances of the same application on the same EC2 instance.

Use cases:

Increased resiliency even if running on one EC2 instance

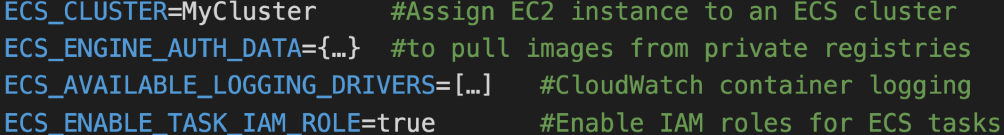
Maximize utilization of CPU / cores

Ability to perform rolling upgrades without impacting application uptime



Run an EC2 instance, install the ECS agent with ECS config file or use an ECS-ready Linux AMI (still need to modify config file). Can install the ECS container agent on any EC2 instance that supports the Amazon ECS specification

ECS Config file “/etc/ecs/ecs.config”



## Auto Scaling

You pay only for the AWS resources needed to run your applications and [Amazon CloudWatch](https://aws.amazon.com/cloudwatch/) monitoring fees.


   An illustration of a basic Auto Scaling group.
  

Scaling groups cannot span multiple regions only AZs in a region. You can create a launch config from an existing EC2 instance (instance settings -> Attach to AS). **You can't modify or disable a launch configuration after you've created it**. However, you can change which launch configuration is associated with an AS group. Specify launch configuration, **minimum # of instances**, and **max number of instances**. You can optionally specify a **desired capacity**, which is the number of instances that the group must have at all times. If you don't specify a desired capacity, the default desired capacity is the minimum number of instances that you specified.

If you are running at desired capacity of 3 and the maximum capacity of 3 and have alarms set at 60% CPU to scale out and your application is now running at 80% capacity **NOTHING** WILL happen

You can suspend AS processes if needed. Such as Launch, Terminate, notify. Etc.

..

Should have two policies, one for **scaling in** and one for **scaling out** for each event to monitor.

IAM roles attached to an ASG will get assigned to EC2 instances

*Scaling Options*

* Maintain – keep a specific or minimum number of instances running
* Manual – use maximum, minimum, or a specific number of instances
* Scheduled – increase or decrease the number of instances based on a schedule
* Dynamic – scale based on real-time system metrics (e.g. CloudWatch metrics)

*Scaling Policies:* CPU, Network, and custom metrics or schedule based

*Health Checks and Scaling*

By default, an Auto Scaling group determines the health state of each instance by periodically checking the results of the EC2 instance status checks but you can instead associate an ELB health check.

Health check grace period: The length of time that Auto Scaling waits before **checking** an instance's health status. The grace period begins when an instance comes into service

Simple scaling policies must wait for the **cooldown period** to expire after a scaling activity or health check replacement before they can respond to alarms that are breached.

**Cooldown** is the period of time which AS ignores scaling actions default is 300 secs (5 mins) and helps to ensure that your ASG doesn't launch or terminate additional instances before the previous scaling activity takes effect

If it is launching more than one instance, the cooldown period for each instance starts after that instance is launched. The group remains locked until the last instance that was launched has completed its cooldown period.

If your application is scaling up and down multiple times each hour, modify the ASG cooldown timers and the CloudWatch Alarm Period that triggers the scale in

Know how the **Default Termination** Policy works:

1. If there are instances in multiple Availability Zones, choose the AZ *with the most instances* and at least one instance that is not protected. If there is more than one AZ with this number of instances, choose the AZ with the instances that use the oldest launch configuration.

2. Determine which unprotected instances in the selected Availability Zone use the *oldest launch configuration*. If there is one such instance, terminate it.

3. If there are multiple instances to terminate based on the above criteria, determine which unprotected instances are *closest to the next billing hour*. (Helps you maximize the use of your EC2 instances and manage your EC2 usage costs.) If there is one such instance, terminate it.

4. If there is more than one unprotected instance closest to the next billing hour, choose one of these instances at random.

To use with ELB, remember to check “receive traffic from ELB “when creating AS group. Always choose Health Check type ELB. For ALB, choose “Target Groups” in the ELB section

Proactive Cycle: scale out based on peak periods

Proactive Event-based: scale out anticipation of increased demand

Demand: scale out based on metrics: CPU, network, etc. (horizontal) use monitoring service such as cloud watch

THE ELB SG must have port 80 open AND the instances launched in the AS group must have port 80 open to the public **or the ELB SG**. The SG is configured in the launch config

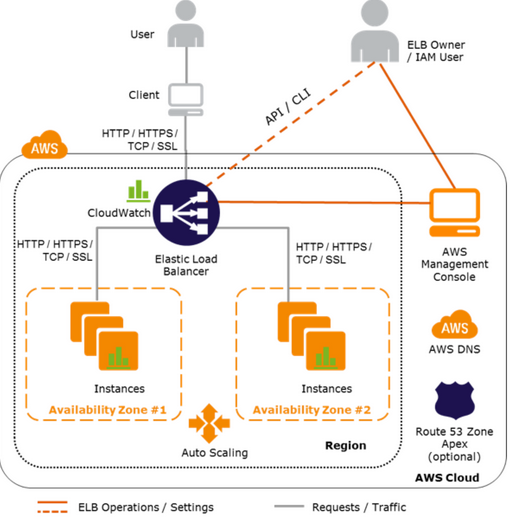
AS will automatically register targets in ELB target group if you are using AS with ELB

There is now AWS AS and EC2 AS: so can use with other service such as DynamoDB and Aurora

Example command:

as-put-scaling-policy scale-up-pol --auto-scaling-group my-as-group --adjustment=1 --type ChangeInCapacity --cooldown 300

Consist of an LC that defines instances that are created in response to demand and an AS group that defines when to use an LC to create new instances and in which AZ and ELB context they should be created in.



## ELB

Given a DNS host name – any requests sent to this host name are delegated to a pool of EC2 instances for multi AZ (Cross Zone load balancing to help route traffic evenly across all EC2 instances regardless of the AZ) and single region.

ELB gives you a single DNS name for addressing and AS ensures there is always the right number of healthy EC2 instances to accept requests

Because the set of IP addresses associated with a Load Balancer can change over time, you should never create an "A" record with any specific IP address.

If you want to use a friendly DNS name for your load balancer instead of the name generated by the ELB service, you should create an ALIAS record for the LoadBalancer DNS name

• 4xx errors are client induced errors

• 5xx errors are application induced errors

• Load Balancer Errors 503 means at capacity or no registered target

When you de register an instance, status will be “draining” to bleed off open requests

Listener Options: <https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/using-elb-listenerconfig-quickref.html>

Routing Algorithm

NLB does not uses a Round-Robin strategy

Classic uses a Round-Robin strategy for TCP listeners only

ALB 1st selects a target based on the routing rule, then uses a Round-Robin strategy to select a node

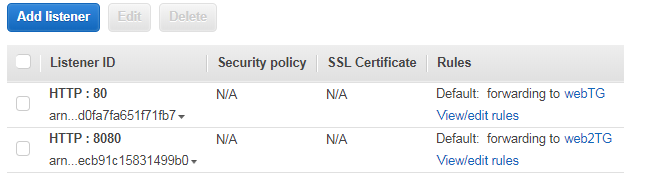
*ALB*

ALB (layer 7) allow you register targets in target groups, and route traffic to the target groups synchronously. With Classic Load Balancers, you register instances with the load balancer.

ALB has better app support. Supports HTTP/S and Web sockets. **Cross-zone load balancing is enabled default**

Better logging and monitoring. More expensive than classic

ALBs allow you to set up multiple target groups and route to them based on the path and/or hostname (path-based (e.g. /images or /orders) and host-based routing e.g. example.com)



Allows dynamic port mapping

The back end app never see the IP of the client directly: true IP of the client is inserted in the header X-Forwarded-For

Can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)

X-Forwarded-For request header takes the following form: X-Forwarded-For: clientIPAddress, previousRequestIPAddress, LoadBalancerIPAddress

Use Cases:

• Load balancing to multiple HTTP applications across machines (target groups)

• Load balancing to multiple applications on the same machine (ex: ECS)

• Load balancing based on route in URL

• Load balancing based on hostname in URL

Sticky Sessions: (CLB and ALB)

Session affinity to bind a user’s session to a specific application instance**. Same user goes to same instance on back end**

The cookie used for stickiness has an expiration date you control

ALB supports load balancer-generated cookies only. ALB Sticky sessions are enabled at the target group level

Use case: make sure the user doesn’t lose his session data

It is not fault tolerant, if an instance is lost the information is lost

Requires HTTP/HTTPS listener and does not work with TCP

Requires SSL termination on ELB as it users the headers for HTTPS

**Stickiness may cause imbalances.**

*NLB*

Network Load Balancers (Layer 4) allow you register targets in target groups, and route traffic to the target groups synchronously

• Forward **TCP** traffic to your instances

• Handle millions of request per seconds

• Support for static IP or elastic IP

• Less latency ~100 ms (vs 400 ms for ALB)

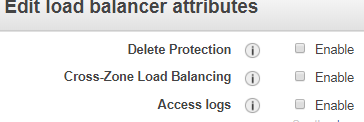
No X-forwarded-for, NLB will see the client

• Public facing: must attach Elastic IP – can help **whitelist** by clients

• Private facing: will get random private IP at time of creation

• **Has cross zone balancing but have to enable it (see below)**

• Has SSL termination



*Health checks*

Default checks every 30 seconds 10 times = 300 seconds (5 mins)

Unhealthy threshold: number of failures before declaring unhealthy. It will check the path that many times

Healthy threshold: # of checks before declaring healthy'

Response time out: TTL when receiving response from health check'

Health check interval: time between health checks so if you change the interval to 10 it cuts down the time to declare it dead

Change healthy threshold to 3 so it declares healthy faster

*SSL*

• Uses an X.509 certificate (SSL/TLS server certificate)

• Can manage certificates using AWS Certificate Manager

• Can create upload your own certificates alternatively

• HTTPS listener:

• You must specify a default certificate

• You can add an optional list of certs to support multiple domains

• Clients can use **SNI** (Server Name Indication) to specify the hostname to connect to. You can host multiple TLS secured applications, each with its own TLS certificate, behind a single load balancer. In order to use SNI, all you need to do is bind multiple certificates to the same secure listener on your load balancer. ALB will automatically choose the optimal TLS certificate for each client

Allows multiple domains to serve SSL traffic over the same IP address by including the hostname which the viewers are trying to connect to.

CloudFront delivers your content from each edge location and offers the same security as the Dedicated IP Custom SSL feature. SNI Custom SSL works with most modern browsers

• Ability to specify a security policy to support older versions of SSL / TLS (legacy clients)

For SSL termination at backend instances or support for Client Side Certificate use TCP for connections from the client to the ELB, use the SSL protocol for connections from the ELB to the back-end application, and deploy certificates on the back-end instances handling requests

Create a secure SSL listener and you will need an SSL cert:

<http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-add-or-delete-listeners.html>

*Access Logging:* After you enable access logging for ELB, it captures the logs and stores them in the S3 bucket that you specify as compressed files

# CloudFormation

The only required section in template is resource. Can see events for a deleted stack for up to 90 days. Templates are global

# CloudTrail

Captures AWS API calls and related events made by or on behalf of an AWS account and delivers log files to an S3 bucket that you specify (encrypted by default using S3-SSE). Optionally, you can configure CloudTrail to deliver events to a log group to be monitored by CloudWatch Logs. You can also choose to receive SNS notifications each time a log file is delivered to your bucket. Can be deployed with a single trail that is applied to all regions

# CloudFront

CDN: content cached at edge locations read and write. A distribution is a network of edge locations. CloudFront sends the distribution configuration to all the edge locations

After setup you can reference objects with the CDN URL prefix by object: <http://d6nw2h20x5scj.cloudfront.net/IMG_1792.JPG>

Can provide SSL encryption (HTTPS) at the edge using ACM and you can now use Field-Level Encryption to further enhance the security of sensitive data, such as credit card numbers or personally identifiable information (PII) like social security numbers

*Origin*

Origin is where the original content lives. Can be s3, ec2, ELB, route53 or an on premise server.

If a local copy of the file is not at an edge location, it will get a copy from the origin

Disabled distribution is no longer functional and Amazon stops billing it.

To remove an object from a cache, must invalidate it in the CloudFront console, API, or CLI

URL per edge location where it caches it where it has a TTL

Origin server can be configured to limit access protocols, caching behavior, add headers to the files to add TTL

You can clear cached objects but you will be charged

Objects are cached for 24 hours by default. How to manage content cache duration: <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html>

*OAI*

Origin Access Identity is a virtual user identity that is used to give the CloudFront distribution permission to fetch a private object from an S3 bucket

**If you setup OAI, then the s3 content can only be accessed via CloudFront via the policy allowing the OAI user**. **You can leave the bucket ACL/Policy to not be public**

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-restricting-access-to-s3.html>

*Signed URLs and Signed Cookies*

You can configure CloudFront to require user access to your files using either signed URLs or signed cookies. You then develop your application either to create and distribute signed URLs to authenticated users or to send Set-Cookie headers that set signed cookies on the viewers for authenticated users.

When you create signed URLs or signed cookies to control access to your files, you can specify the following restrictions:

An ending date and time, after which the URL is no longer valid.

Trusted signers (which AWS accounts can create signed URLs)

(Optional) The IP address or range of addresses of the computers that can be used to access your content.

CloudFront signed URL can only be created using the AWS SDK, so you have to code an application to verify users and generate these URLs

**If S3 can only be accessed through CloudFront, we cannot used self-signed S3 URLs**. If require that users always access S3 content using CloudFront URLs, not S3 URLs, click Yes to "Restrict Bucket

Access". This is useful when you are using signed URLs or signed cookies to restrict access to your content. Then you can use OAI

Use signed URLs for the following cases:

You want to use an RTMP distribution. Signed cookies aren't supported for RTMP distributions.

You want to restrict access to individual files, for example, an installation download for your application.

Your users are using a client (for example, a custom HTTP client) that doesn't support cookies.

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-urls.html>

Use signed cookies for the following cases:

You want to provide access to multiple restricted files, for example, all of the files for a video in HLS format or all of the files in the subscribers' area of a website.

You **don't want to change your current URLs**.

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-cookies.html>

There is also CloudFront Geo Restriction using whitelist and blacklists

# Lambda

You pay only for the compute time you consume - there is no charge when your code is not running.

Billing is based on both RAM reserved and the execution duration in 100ms units. Pay per calls and duration (increments of 100ms)

Compute service that runs your code in response to events and automatically manages the underlying compute resources for you

node.js, java, python, GO, ruby, .NET, and PowerShell

**Scales out automatically** and CHEAP

Can trigger other lambda functions, 1 event = x functions if functions trigger other functions

Global, can use to backup s3 bucket to bucket

*AWS Lambda Limits*:

Execution:

Memory allocation: 128 MB – 3008 MB (64 MB increments)

**Maximum execution timeout**: 300 seconds max 15 mins, **default is 3** seconds

Disk capacity in the “function container” (in /tmp): 512 MB

Concurrency limits: 1000 functions at a time

Deployment:

Lambda function deployment size (compressed .zip): 50 MB

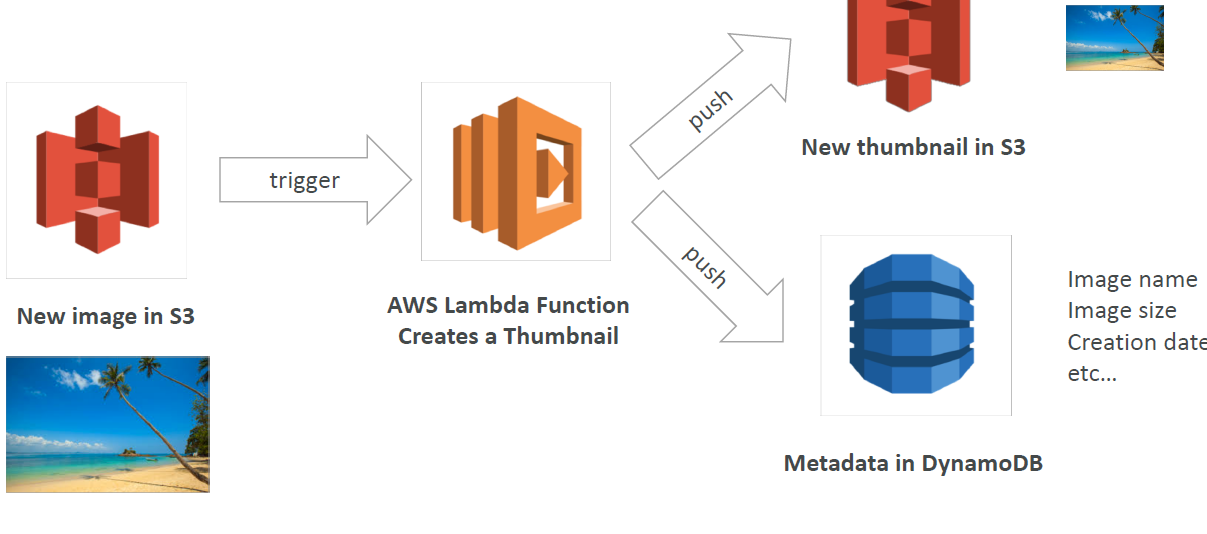
Size of uncompressed deployment (code + dependencies): 250 MB

Can use the /tmp directory to load other files at startup

Size of environment variables: 4 KB

Can use X-ray to debug lambda

Serverless Example



# CloudWatch

Choosing detailed monitor at ec2 creation switches from 5 mins to 1 mins but cost $

Dimension is an attribute of a metric (instance id, environment, etc.…).

Memory is not available in this because it grabs Hyper-V stats. Metrics exist only in the region in which they are created. Need 3rd party script for metrics: memory, disk space, swap utilization

However can use the **CloudWatch Agent**: <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html>

You get a CPU credit balance that builds up if your utilization is low

Can’t use IAM to control access to CloudWatch data for specific resources. For example, you can't give a user access to CloudWatch data for only a specific set of instances or a specific LoadBalancer.

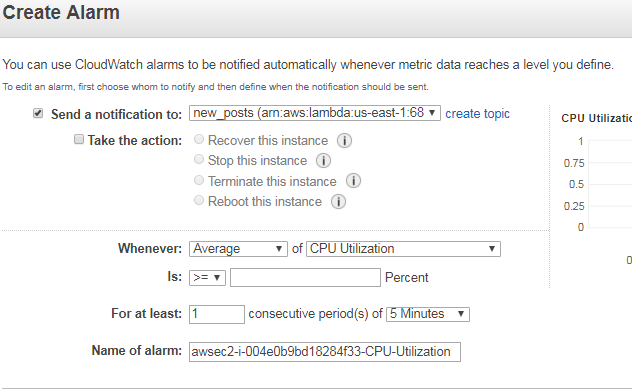
You can however specify which CloudWatch actions a user in your Account can perform. For example, you could create an IAM policy that gives only certain users in your organization permission to use GetMetricStatistics

CloudWatch stores metrics for terminated EC2 instances or deleted Elastic Load Balancers for 15 months and alarm history for 14 days.

*Alarms*

Alarm Actions:

* Recover the instance (only supported on specific instance types and can be used only with StatusCheckFailed System)
* Stop the instance (only applicable to EBS-backed volumes)
* Terminate the instance (cannot terminate if termination protection is enabled)
* Reboot the instance



*CloudWatch Logs/Agent*

Applications can send logs to CloudWatch using the SDK

CloudWatch Logs can go to:

Batch exporter to S3 for archival

Stream to ElasticSearch cluster for further analytics

Logs storage architecture:

Log groups: arbitrary name, usually representing an application

Log stream: instances within application / log files / containers

Can define log expiration policies (never expire, 30 days, etc...)

AWS CLI to tail CloudWatch logs

To send logs to CloudWatch, make sure IAM permissions are correct!

Security: encryption of logs using KMS at the Group Level

*Dashboards*

**Dashboards are global and can** include graphs from different regions

You can change the time zone & time range of the dashboards

You can setup automatic refresh (10s, 1m, 2m, 5m, and 15m)

*CloudWatch Events*

Source + Rule => Target

Schedule: Cron jobs

Event Pattern: Event rules to react to a service doing something

Ex: CodePipeline state changes!

Triggers to Lambda functions, SQS/SNS/Kinesis Messages

CloudWatch Event creates a small JSON document to give information about the change

*Custom Metrics*

Ability to use dimensions (attributes) to segment metrics

Instance.id

Environment.name

Metric resolution:

Standard: 1 minute

High Resolution: up to 1 second (StorageResolution API parameter) – Higher cost. Alarms can be triggered as often as 10 seconds

Use API call PutMetricData

Use exponential back off in case of throttle errors

# SNS

SNS has the following subscribers; Lambda, SQS, HTTPS, Email, SMS.  **Push** instant notifications to clients.  Pay as you go pricing push to apple, google, android, etc. Stored across multiple AZs. Notifications can be delivered over SMS, email, SQS, HTTP/S. Pub/sub = Publish/Subscribe

The event producer only sends message to one SNS topic. As many event receivers (subscriptions) as we want to listen to the SNS topic notifications

**Fan Out:**

Push once in SNS, receive in many SQS

Fully decoupled

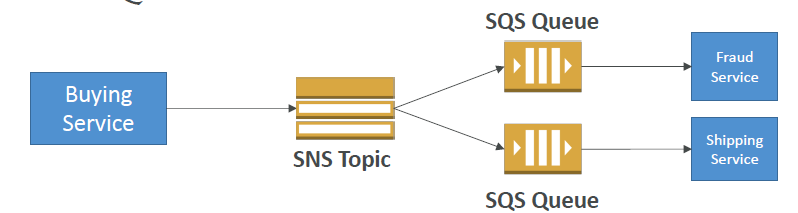
No data loss

Ability to add receivers of data later

SQS allows for delayed processing

SQS allows for retries of work

May have many workers on one queue and one worker on the other queue



# Route53

Limit of 50 domain names but can be increased

No charge to route Internet traffic to CloudFront, Elastic Beanstalk, ELB, or S3.

Zone apex ("root domain" or "naked domain") to be an "A Record,” cannot be a CNAME. A R53 alias similar to CNAME but can point to subdomains AND zone apex

Zone apex record to point to a load balancer: A record aliased to the load balancer DNS name

Alias resource record sets can save you time because Route 53 automatically recognizes changes in the resource record sets that the alias resource record set points to. Because the set of IP addresses associated with a LoadBalancer can change over time, you should never create an "A" record with any specific IP address. Best to create an alias record to the ELB name

If an alias resource record set points to a CloudFront distribution, a load balancer, or an S3 bucket, the time to live (TTL) can’t be set instead Route 53 uses the CloudFront, load balancer, or S3 TTLs.

Dig will show you the TTL left on a record

Every request you make to the Amazon Route 53 control API must be authenticated.

If you use a domain managed by another company just update the NS records for your domain to the Route 53 ones

--

***Routing Policies:***

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html>

*Simple*: one record with **multiple IP addresses**

*Failover*: route traffic to a healthy resource or to a different resource when the first resource is unhealthy. Active/Passive using health checks. Failover primary instance MUST be associated with a health check

*Geolocation*: you might want all queries from Europe to be routed to an ELB in the Frankfurt region. Routes traffic based on originating location to specified region. Always have a default location! Geolocation is based on national boundaries and will meet the needs well. Geoproximity is based on Latitude & Longitude

*Weighted*: example: if you want to send a tiny portion of your traffic to one resource and the rest to another resource, you might specify weights of 1 and 255. The resource with a weight of 1 gets 1/256th of the traffic and the other resource gets 255/256ths. You can gradually change the balance by changing the weights. If you want to stop sending traffic to a resource, you can change the weight for that record to 0.

*Latency*: e.g.: you have web servers in example.com in region Ireland and in Tokyo. When a user goes to example.com, Route 53 chooses to respond to the DNS query based on which region gives your user the lowest latency. *E.g. if the latency from the user in Singapore to Ireland improves, the user can be routed to Ireland.*

*Multivalue answer*: return multiple values (up to 8), such as IP addresses in response to DNS queries. You can specify multiple values for almost any record, but Multivalue answer routing also lets you check the health of each resource, so Route **53 returns only values for healthy resources (you can see this in a dig query)**. **It's not a substitute for a load balancer**, but the ability to return multiple health-checkable IP addresses is a way to use DNS to improve availability and load balancing.

---

In active-active failover, all the records that have the same name, the same type (such as A or AAAA), and the same routing policy (such as weighted or latency) are active unless Route 53 considers them unhealthy. Route 53 can respond to a DNS query using any healthy record.

*Health Checks*

Can have HTTP, TCP and HTTPS health checks (no SSL verification). Also look at all of the tabs and variables of existing health checks

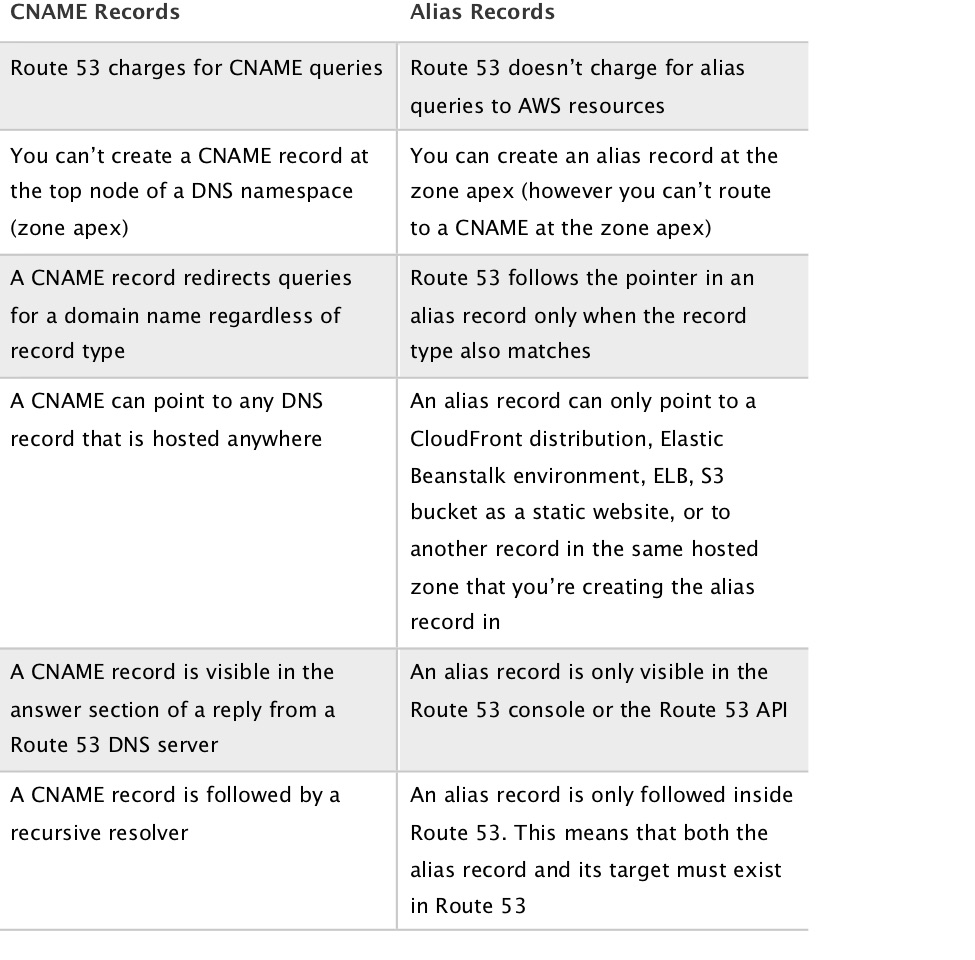
*CNAME vs ALIAS*

CNAME record maps a name to another name. It should only be used when there are no other records on that name. Can point to any DNS record hosted anywhere

ALIAS record maps a name to another name, but in **turn it can coexist with other records on that exact name**. Can only point to a CloudFront distribution, an Elastic Beanstalk env, S3 Site, an ELB **OR** records you already created for that zone!!

CNAME charges for queries, not so with Alias record.

Common to create an A record [www.urdomain.com](http://www.urdomain.com) then create an alias to that using the naked domain name so urdomain.com will point to www



# DB

## DynamoDB

For applications that need consistent, single-digit millisecond latency at any scale. Fully managed and supports both document and key-value store models. Its flexible data model, reliable performance, and auto scaling of throughput capacity makes it a great fit for mobile, web, gaming, ad tech, IoT

Creates the table across 3 AZs automatically and synchronously.

NoSQL: it is not as rigid as a relational database because you can easily add or remove rows or elements in your table/collection entry. It also has a more flexible schema because it can store complex hierarchical data within a single item which, unlike a relational database, does not entail changing multiple related tables

Pay only for what you use and there is no minimum fee. *3 pricing components*: provisioned throughput capacity (per hour), indexed data storage (per GB per month), data transfer in or out (per GB per month, however: no charge for transfer into DynamoDB, within a single region)

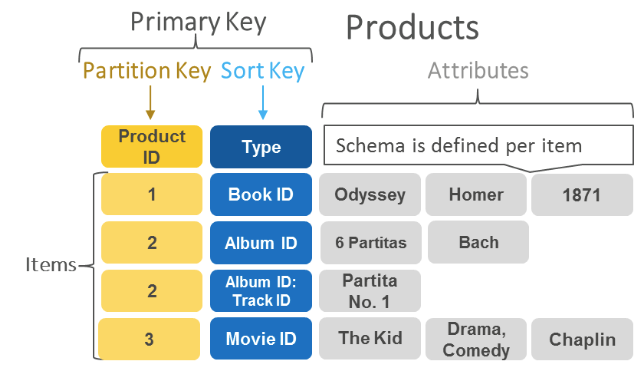
*Eventually Consistent Reads by default*: When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation. The response might include some stale data. If you repeat your read request after a short time, the response should return the latest data.

*Strongly Consistent Reads*: When you request, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful. A strongly consistent read might not be available if there is a network delay or outage.

Stores structured data in tables with an infinite number of rows, indexed by primary key, and allows low-latency read and write access to items

Maximum item size is 400 KB, which includes both attribute name binary length and attribute value lengths. The attribute name counts towards the size limit.

**Can only query on primary key, sort key, or indexes**



*Data Types:*

Scalar Types: String, Number, Binary, Boolean, Null

Document Types: List, Map

Set Types: String Set, Number Set, Binary Set

Tables do not have fixed a schema, so each data item can have a different number of attributes. The primary key can either be a single-attribute hash key or a composite hash-range key. Local secondary indexes provide additional flexibility for querying against attributes other than the primary key. Provides implicit item-level transactions for item put, update, delete, conditional operations, and increment/decrement.

*Global Tables and Cross Region Replication*: fully managed solution for deploying a multi-region, multi-master database. Performs all of the necessary tasks to create identical tables in these regions, and propagate ongoing data changes to all of them. Global tables need Dynamo Streams enabled

*Provisioned Throughput*

Table must have provisioned read and write capacity units

Read Capacity Units (RCU): throughput for reads ($0.00013 per RCU)

One *read capacity unit* represents one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB

The total number of RCU required depends on the item size, and whether you want an eventually consistent or strongly consistent read. For example, if your item size is 8 KB, you require 2 read capacity units to sustain one strongly consistent read per second, 1 read capacity unit if you choose eventually consistent reads, or 4 read capacity units for a transactional read request.

Write Capacity Units (WCU): throughput for writes ($0.00065 per WCU)

One *write capacity unit* represents one write per second for an item up to 1 KB

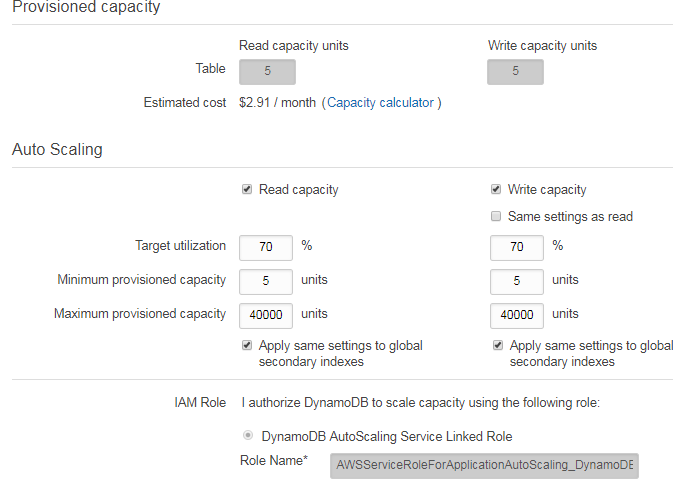
The total number of WCU required depends on the item size. For example, if your item size is 2 KB, you require 2 write capacity units to sustain one write request per second or 4 write capacity units for a transactional write request

Option to setup auto-scaling of throughput to meet demand

Throughput can be exceeded temporarily using burst credit

If burst credit are empty, you’ll get a ProvisionedThroughputException

It’s then advised to do an exponential back-off retry



On Demand

No capacity planning needed (WCU/RCU) - scales automatically

2.5x more expensive than provisioned capacity

Helpful when spikes are un-predictable or the application is very low throughput

*DAX (DynamoDB Accelerator)*

Seamless in memory cache for DynamoDB, no application rewrite

Writes go through DAX to DynamoDB

Micro second latency for cached reads & queries

Solves the Hot Key (hot partition) problem (too many reads)

5 minutes TTL for cache by default

Up to 10 nodes in the cluster

Multi AZ (3 nodes minimum recommended for production)

ProvisionedThroughputExceededException

*Streams*

Ordered flow of information about changes to items in a table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table. Whenever an application creates, updates, or deletes items in the table, Streams writes a stream record with the primary key attribute(s) of the items that were modified. A stream record contains information about a data modification to a single item in a table.

This stream can be read by Lambda, and we can use triggers to then do:

React to changes in real time (welcome email to new users)

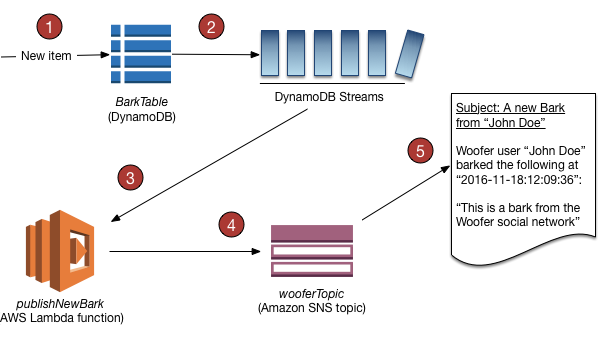
Analytics

Create derivative tables / views

Insert into ElasticSearch

Could implement cross region replication using Streams

Stream has hours of data retention



*Transactions*

All or nothing type of operations

Coordinated Insert, Update & Delete across multiple tables

Include up to 10 unique items or up to 4 MB of data

Good for storing JSON and web session data, index pointers to large data types (BLOB in SE for example), and supporting stateless web/app installations. **NOT good for big objects**

*Pillar*

Operations: no operations needed, auto scaling capability (**DynamoDB Auto Scaling**), serverless

Security: full security through IAM policies, KMS encryption, SSL in flight

Reliability: Multi AZ, Backups

Performance: single digit millisecond performance, DAX for caching reads, performance doesn’t degrade if your application scales

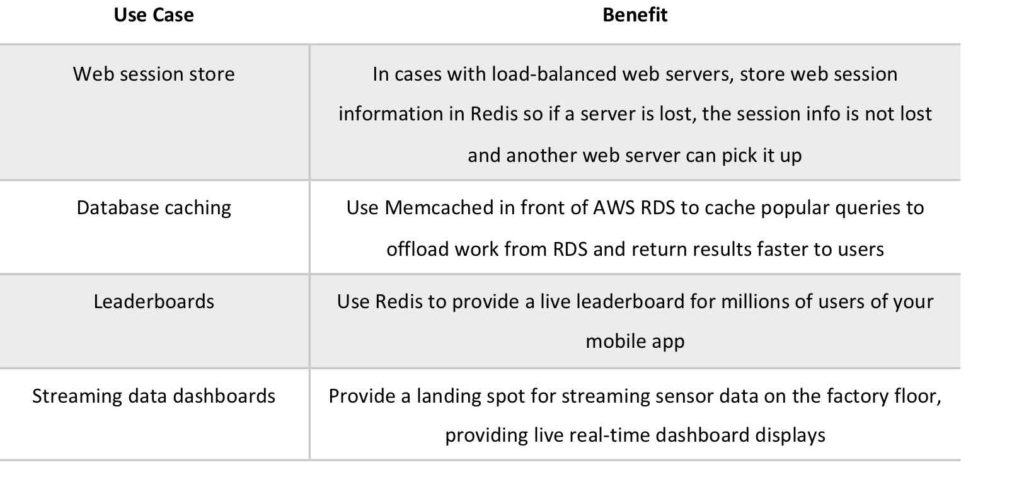
Cost: Pay per provisioned capacity and storage usage (no need to guess capacity – can use auto scaling). Priced based on provisioned throughput (read/write) regardless of whether you use it or not

## ElastiCache

Fully managed in memory cache, good for repetitive queries for many read-heavy application workloads or compute-intensive workloads.

Sits between and app and DB (RDS). Must provision an EC2 instance type

Good for STATELESS app and Key/Value store, Frequent reads, less writes, cache results for DB queries, store session data for websites, cannot use SQL.



*Memcached*

Not persistent and cannot be used as a data store

Supports large nodes with multiple cores or threads

**Scales horizontally, by adding and removing nodes**

*Redis*

Data is persistent and can be used as a datastore

Not multi-threaded

Scales by adding shards, not nodes

**Supports Multi-AZ**

Super Low Latency

Redis support Redis AUTH (username / password) which requires SSL in-flight encryption must enabled

Pillar

Operations: same as RDS

Security: AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, users (Redis Auth), using SSL

Reliability: can have a fully automated, fault tolerant implementation by enabling both **cluster mode and multi-AZ failover** using Redis

Performance: Sub-millisecond performance, in memory, read replicas for sharding, very popular cache option

Cost: Pay per hour based on EC2 and storage usage

## REDSHIFT

**OLAP**: does analytics on large amount of records. SQL based Data Warehouse.

Uses replication and continuous backups (default 1 day retention, 35 max) to enhance availability and improve data durability and can automatically recover from node and component failures

Single node or multi-node: leader node: manage client connect and receive query. Compute nodes: store data and perform query up to 128 nodes

Columnar data storage. Not like rows. Easier to compress. Doesn’t require indexes. 10x Faster than standard SQL DB. Can also improve performance for repeat queries by caching the result and returning the cached result when queries are re-run. Massive Parallel Processing: easy to load across nodes

Encrypted transit SSL. Encrypted at rest using AES-256

**Only available in 1 AZ.** Can restore snapshot to new AZ

Uses EC2 on the backend so you need to choose your instance type/size for scaling compute vertically, but you can also scale horizontally by adding more nodes to the cluster

Always keeps 3 copies: Original, replica on compute nodes, backup asynchronously on S3

Use Case- Analyze global sales data for multiple products:

 Store historical stock trade data

 Analyze ad **impressions and clicks**

 Aggregate gaming data

 Analyze social trends

 Measure clinical quality, operation efficiency, and financial performance in the health care space

A cluster subnet group allows you to specify a set of subnets in your VPC. When provisioning a cluster you provide the subnet group and Redshift creates the cluster on one of the subnets in the group.

Pillar

Operations: similar to RDS

Security: IAM, VPC, KMS, SSL (similar to RDS)

Reliability: highly available, auto healing features, **automated snapshots**

Performance: 10x performance vs other data warehousing, compression

Cost: pay per node provisioned, 1/10th of the cost vs other warehouses

## RDS

RDS instances are regional can be copied to different regions via snapshot or promoting read replica. Fully managed service: no patching or audit, all done by AWS

EC2 instance & EBS Volume type and size are provisioned on the back end but transparent

**OLTP: transaction record based queries**

Supports MySQL, MariaDB, PostgreSQL, Oracle, SQL Server

Can scale storage and change the storage type for all DB engines except MS SQL

By default, the maximum provisioned IOPS capacity on an Oracle and MySQL RDS instance (using provisioned IOPS) is 30,000 IOPS

Connection e.g.: mysql -h jimmydb.czlxgi6wj1oj.us-east-1.rds.amazonaws.com –P 3306 -u jimmy –p

Best Practice: instead of using the long RDS endpoint name, create internal zone and then create a CNAME to the endpoint: e.g. mysql.Internal -> CNAME wordpressdb.czlxgi6wj1o…

Default **DB subnet group** is created for your account by default with the default VPC ONLY

DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances.

Make sure create a DB subnet group in the same VPC as the EC2 SG then use the EC2 SG as the source when creating rule for DB Subnet Group

Upgrade DB instance: apply the changes immediately, choose Apply immediately. Choosing this option can cause an outage in some cases. For auto updates: an enable auto minor version upgrades. For modifications if you don't choose to apply changes immediately, the changes are put into the pending modifications queue. During the next maintenance window, any pending changes in the queue are applied

RDS automatically sends metrics to CloudWatch every minute for each active database: standard monitoring provides metrics ranging from CPU, memory, and disk usage, to more specific ones such as DB connections, I/O/latency/throughput

**RDS now has IAM authentication for MySQL and PostgreSQL:** you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token.

Network traffic to and from the database is encrypted using SSL

*Enhanced monitoring*: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.OS.html>

Walkthrough: creating DB instance and connecting:

1. Create Private SG in the VPC where your EC2s will connect. Add rule for mysql port with the EC2 instance SG as the source

2. Will ask for DB subnet grp, leave default OR if you choose a VPC that does not have one need to create it. (Each DB subnet group should have subnets in at least 2 Availability Zones)

You can create the DBSG in non-default VPC manually and it has to cover at least 2 AZ.  If you don’t, the RDS console will create when u launch RDS instance and will add all subnets in the VPC to it

3. Leave AZ no preference and launch it

4. Now you should be able to connect from EC2 instances from step1. Each DB instance creates a default snap shot, delete it after delete instance

**Pillar**

Operations: less operations, auto scaling storage. Small downtime for failover and maintenance, scaling in read replicas/ec2 instance/restore EBS implies manual intervention, application changes

Security: AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, authorizing users in DB, using SSL

Reliability: Multi AZ, highly available, Aurora Serverless option.

Performance: 5x performance. Up to 15 Read Replicas (only 5 for RDS). Depends on EC2 instance type, EBS volume type, ability to add Read Replicas. **Doesn’t auto-scale**

Cost: Pay per hour based on EC2 and storage usage. Possibly lower costs compared to Enterprise grade databases such as Oracle

*Multi-AZ*

Updates to your DB Instance are synchronously replicated across Availability Zones to the standby in order to keep both in sync and protect your latest database updates against DB Instance failure

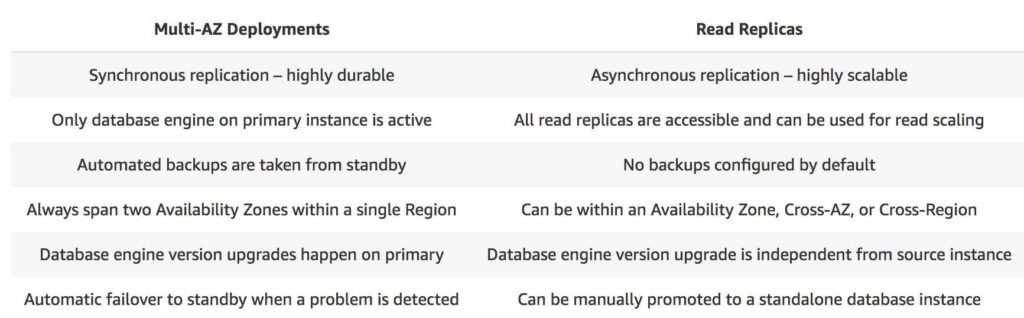
There is no charge to replicate data, Multi-AZ can be turned on after DB is created. Transaction success is returned only if the commit is successful both on the primary and the standby. In a fail over event, primary DB instance will switch over automatically to the standby replica and automatically changes the DNS record of the DB instance to point to the standby DB instance

*Read Replicas*

You can create a Read Replica of a given source DB Instance using the Management Console or the CreateDBInstanceReadReplica API. Once the Read Replica is created, database updates on the source DB Instance will be propagated to the Read Replica asynchronously (eventually consistent). You can create multiple Read Replicas (up to 5) for a given source DB Instance and distribute your application’s read traffic amongst them. Backups for replicas are not configured by default.

Can be cross AZ **and region** and can have 5 max RRs copies of a DB

Available [MySQL](https://aws.amazon.com/rds/mysql/), [MariaDB](https://aws.amazon.com/rds/mariadb/), [PostgreSQL](https://aws.amazon.com/rds/postgresql/) and [Oracle](https://aws.amazon.com/rds/oracle/)



With MySQL, MariaDB, PostgreSQL, and Oracle you can also set the read replica as Multi-AZ, allowing you to use the read replica as a DR target. When you promote the read replica to be a standalone database, it will already be Multi-AZ enabled

*Backup/Snapshots*

Restore DB Instance (snapshot): You are creating a NEW DB Instance (with new DNS endpoint) from a source DB Instance at a specified time. DB Snapshots are user-initiated.

By default and at no additional charge, RDS enables automated backups of your DB Instance with a 1 day retention period. Backup retention max 35 days.

Setting the backup retention period to 0 disables automated backups. The default backup retention period is seven days if you create the DB instance using the console

Automated backups occur **daily** during the preferred backup window. If the backup requires more time than allotted to the backup window, the backup continues after the window ends, until it finishes

Backup storage for each region is composed of the automated backups and manual DB snapshots for that region. Your backup storage is equivalent to the sum of the database storage for all instances in that region. Moving a DB snapshot to another region increases the backup storage in the destination region

*Encryption*

RDS encryption at rest is done via KMS and ONLY when the DB is created. Once the instance is encrypted, so are the backups, replicas, and snapshots.

SSL certificates to encrypt data to RDS in flight

To enforce SSL:

PostgreSQL: rds.force\_ssl=1 in the AWS RDS Console (Parameter Groups)

MySQL: Within the DB:

GRANT USAGE ON \*.\* TO 'mysqluser'@'%' REQUIRE SSL;

To connect using SSL:

Provide the SSL Trust certificate (can be download from AWS)

Provide SSL options when connecting to database

Encrypt unencrypted DB: unencrypted DB => snapshot => copy snapshot as encrypted => create DB from snapshot

*IAM DB Auth*

Works with MySQL and PostgreSQL. IAM database authentication token provides: SSL in transit, IAM central user management, use profile credentials specific to your EC2 instance to access your DB

## Aurora

Supports a maximum DB size of 64 TiB and scales automatically in 10G increments all other RDS DB types support a maximum DB size of 16 TiB

Can change the EC2 instance on back end at any time

Directly compatible with MySQL and PostgreSQL in fact you can create MySQL read replicas of Aurora DBs and vice versa

2 copies of your data contained in each AZ with min of 3 = 6 copies

Automated backups that do don affect DB performance and same with snapshots

Can handle the loss of 2 copies data without affecting writes and 3 without affecting reads. Storage is self-healing, auto scanned and repaired

Aurora database can be Global for DR or latency purposes and the read replicas can be global as well.

*Replicas*

Independent endpoints in an Aurora DB cluster, best used for scaling read operations and increasing availability up to 15 Replicas can be distributed across the Availability Zones that a DB cluster spans within a **single region**.

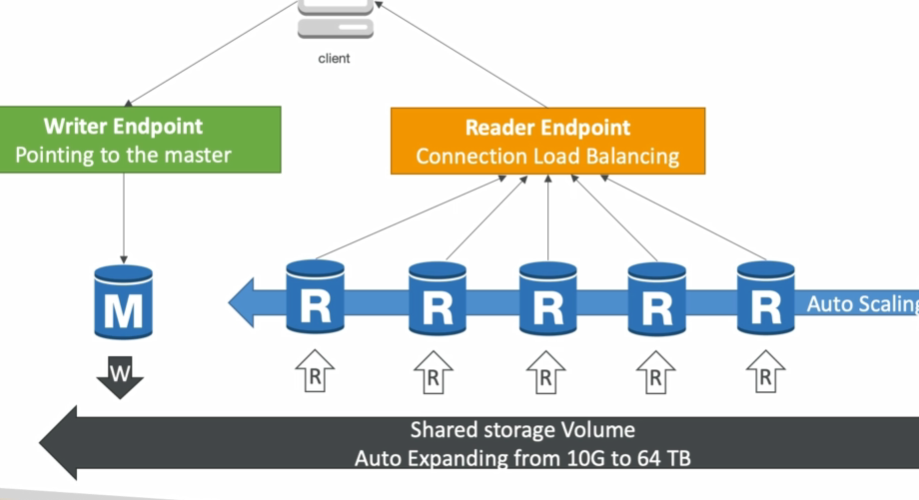
Can create Aurora replicas of other RDS instances such as MySQL. You can then promote that replica to be the master thereby migrating the DB to Aurora

*Aurora Global Database*

It is designed for globally distributed applications, allowing a single database to span multiple regions. It replicates your data with no impact on database performance, enables fast local reads with low latency in each region, and provides disaster recovery from region-wide outages

*Writer/Reader Endpoints*

You can see these endpoints after building the DB. 2 Different DNS names. You have to point to a writer endpoint to write and the read endpoint to read



# SQS

Ideal for multiple application components that communicate and coordinate their work in a loosely coupled and asynchronous manner.

Messages can contain up to 256kb of text in any format

delayMessages can be kept in que from 1 min to 14 days, default retention period is 4 days

*Message Producer*

Define Body

Add message attributes (metadata - optional)

Provide Delay Delivery (optional)

Get Back

Message identifier

MD5 hash of the body

*Message Consumer*

Poll SQS for messages (receive up to 10 messages at a time)

Process the message within the visibility timeout

Delete the message using the message ID & receipt handle

*Queue Types*

Standard: (Default) Unlimited transactions per second. Guaranteed to deliver message at least once but cannot guarantee it will not create duplicates and can’t guarantee order. It will make best effort to keep messages in order

Scales from 1 message per second to 10,000s per second **automatically**

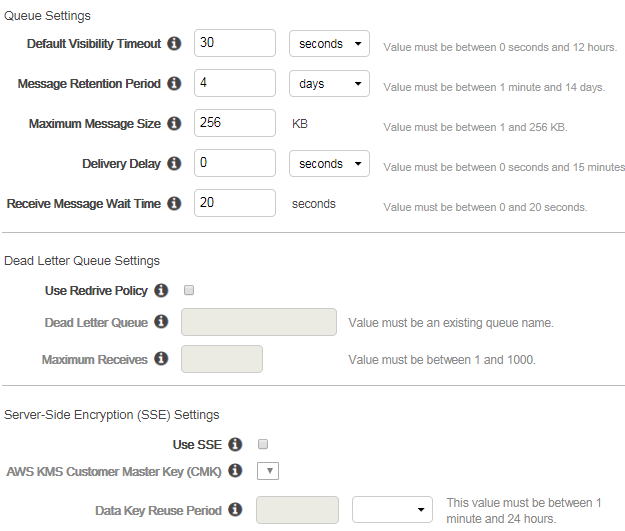
Delay Queue:

Delay a message (consumers don’t see it immediately) up to 15 minutes

Default is 0 seconds (message is available right away)

Can set a default at queue level

Can override the default using the **DelaySeconds** parameter



FIFO: guarantee order and no duplicates. Messages are delivered once and is there until consumed and deleted. Not in all regions yet

Support up to 300 messages per second (300 send, receive, or delete operations per second). When you batch 10 messages per operation (maximum), FIFO queues can support up to 3,000 messages per second

E.g. You have a number of image files to encode. In an SQS worker queue, you create an SQS message for each file specifying the command (jpeg-encode) and the location of the file in S3. A pool of EC2 instances running the needed image processing software does the following:

1. Asynchronously pulls the task messages from the queue

2. Retrieves the named file

3. Processes the conversion

4. Writes the image back to Amazon S3

5. Writes a “task complete” message to another queue

6. Deletes the original task message

7. Checks for more messages in the worker queue

*Visibility timeout*

The **visibilitytimeout** is theamount of time a message is **invisible** to other consumers in the queue after a reader picks up the message. The clock starts when a component first retrieves the message. During that time, the component processes and deletes the message. If your system doesn't call [**DeleteMessage**](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_DeleteMessage.html) for that message before the visibility timeout expires, the **message again becomes visible** to the [ReceiveMessage](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ReceiveMessage.html) calls placed by the components in your system and it will be received again. If a message should only be received once, your system should delete it within the duration of the visibility timeout. Default visibility timeout is **30 seconds** but can be increased to **12 hours max**

So if set too high (e.g. 15 minutes) and consumer fails to process the message, you must wait a long time before processing the message again

And if set too low (e.g. 30 seconds) and consumer needs time to process the message (2 minutes), another consumer will receive the message and the message will be processed more than once

Can also leverage the ‘**DelaySecond**s’ attribute: When a new message is added to the SQS queue, it can’t be processed by the consumer for a fixed period

Use **ChangeMessageVisibility API** to change the visibility while processing a message

Use **DeleteMessage API** to tell SQS the message was successfully processed

<https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html>

*Long Polling*

When the consumer instance polls, the SQS service will allow it to wait a certain time (**WaitTimeSeconds**) for one or more messages to be available before closing the connection.

The default SQS short polling returns immediately, even if the queue being polled is empty

SQS long polling polls periodically and doesn’t return a response until a message arrives in the queue or the long poll times out. The consumer can “wait” for messages in sense

Decreases the number of API calls made to SQS while increasing the efficiency and latency of your application

Enable long polling by setting the API ReceiveMessageWaitTime (**WaitTimeSeconds)** to a number > 0. Range is 1 to 20 seconds, higher the better

Short does not query all the servers that the SQS messages can reside on, so multiple queries of the queue may be needed to retrieve all messages in the queue.

*Dead Letter Queue*

If a consumer fails to process a message within the Visibility Timeout then the message goes back to the queue!

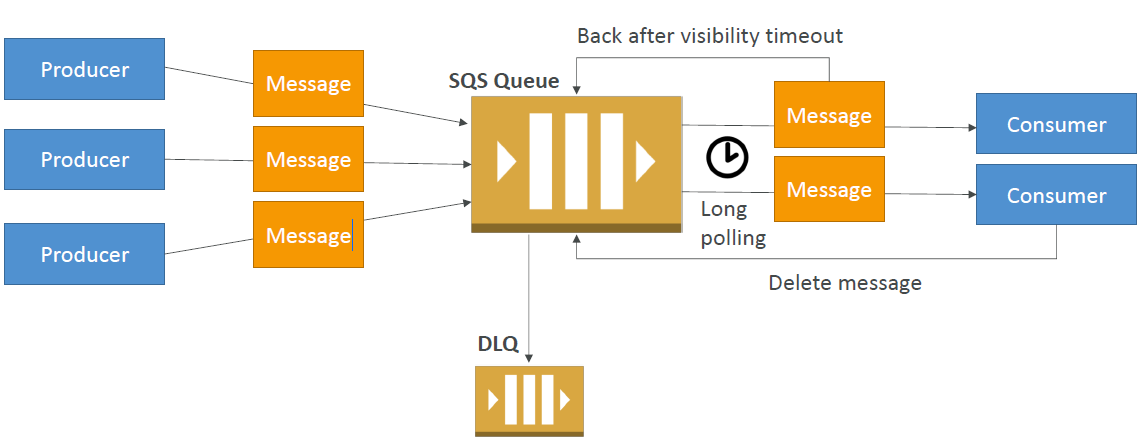
We can set a threshold of how many times a message can go back to the queue. It’s called a redrive policy

After the threshold is exceeded, the message goes into a dead letter queue (DLQ)

We have to create a DLQ first and then designate it dead letter queue

Make sure to process the messages in the DLQ before they expire!

*Overall Diagram*



# SWF

TASK oriented NOT message oriented.

Workers: app that can start a workflow. Like ordering from web site, they get tasks, process them, and return results

Deciders: control flow of activity. Verify address, cred card etc.

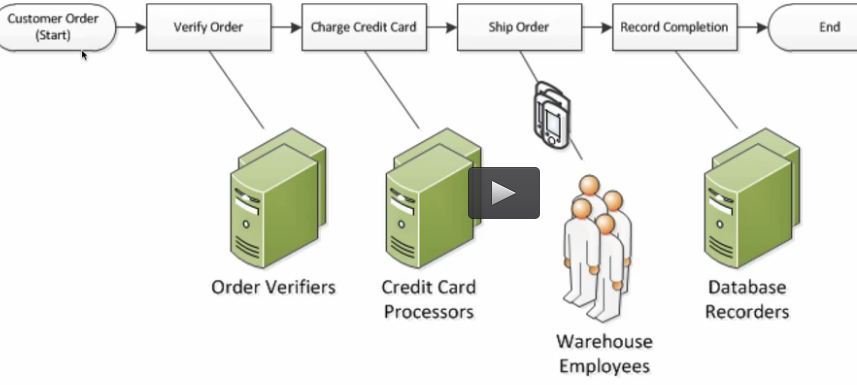
Decision tasks: sent to deciders in order for them to make decisions. Tells the decider the state of the work flow and the next thing to be done

SWF **ensures that a task is assigned only once and is never duplicated** and guarantees order and has retention of 1 year for workflow executions

Uses long polling

Can be used with on prem servers

SWF Domains: collection of related work flows (in json)



**Step Functions is recommended to be used for new applications, except:**

If you need external signals to intervene in the processes

If you need child processes that return values to parent processes

***AWS Step Functions*** provides serverless orchestration for modern applications. Orchestration centrally manages a workflow by breaking it into multiple steps, adding flow logic, and tracking the inputs and outputs between the steps. As your applications execute, Step Functions maintains application state, tracking exactly which workflow step your application is in, and stores an event log of data that is passed between application components. That means that if networks fail or components hang, your application can pick up right where it left off.

Use Case: automate the recurring tasks in your department such as patch management, infrastructure selection, and data synchronization to improve their current processes

# MISC

## EMR

Utilizes a hosted **Hadoop/Spark** framework running on the web-scale infrastructure of EC2 and S3. Launches all nodes for a given cluster in the **same Availability Zone**.

Supports Persistent and Transient cluster types

Cluster states are:

STARTING – The cluster provisions, starts, and configures EC2 instances.  
BOOTSTRAPPING – Bootstrap actions are being executed on the cluster.  
RUNNING – A step for the cluster is currently being run.  
WAITING – The cluster is currently active, but has no steps to run.  
TERMINATING - The cluster is in the process of shutting down.  
TERMINATED - The cluster was shut down without error.  
TERMINATED\_WITH\_ERRORS - The cluster was shut down with errors.

Use cases: log analysis, data processing, machine learning, and web indexing, big data

## Kinesis

*Streams*: Streaming Data from lots of sources (BIG data, ETL etc.). Data records are only accessible for a default of 1 day from the time they are added to a stream. Can increase to 7 days.

Stored in Shards (partitions). A shard is a unit of throughput capacity. Each shard ingests up to 1MB/sec and 1000 records/sec, and emits up to 2MB/sec. To accommodate for higher or lower throughput, the number of shards can be modified after the Kinesis stream is created using the API. Adding and removing shards = scaling

The data is consumed by something like EC2 -> S3, Redshift, or your own shit

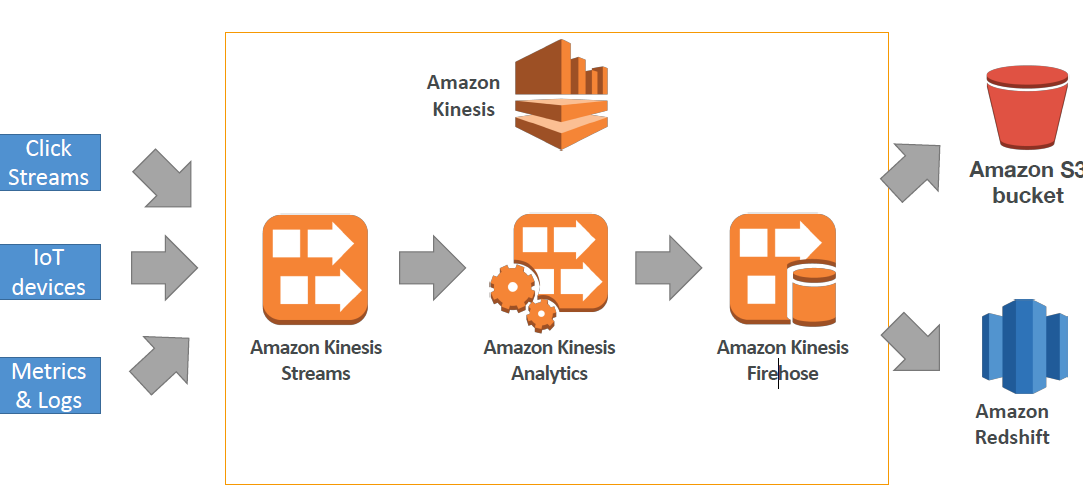
Once data is inserted in Kinesis, it can’t be deleted (immutability)

Provides ordering of records, as well as the ability to read and/or replay records in the same order

Replicated to 3 AZ

*Firehose*: data producers -> into firehose (no shards, it does it all 4 u), analyze the data (optional) and load into Redshift / Amazon S3 / ElasticSearch/ Splunk. Pay for data going into Firehose

*Analytics*: can run sql queries in firehose and streams then store in s3, redshift etc. Pay for actual consumption rate



NOTE: you use Streams OR firehose.

## Beanstalk

Free but you pay for the underlying instances AND must manage them going forward. Almost like CloudFormation

Managed service

Instance configuration / OS is handled by Beanstalk however you would need to manage the EC2 instances for patching etc.

Deployment strategy, capacity, load balancing etc. is configurable but performed by Elastic Beanstalk

Three architecture models:

Single Instance deployment: good for dev

LB + ASG: great for production or pre-production web applications

ASG only: great for non-web apps in production (workers, etc...)

Elastic Beanstalk has three components

Application

Application version: each deployment gets assigned a version

Environment name (dev, test, prod…)

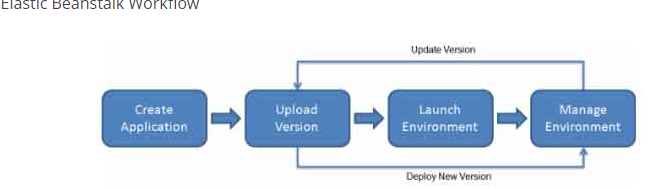
You deploy application versions to environments and can promote application versions to the next environment

Rollback feature to previous application version

Full control over lifecycle of environments

Integrates with RDS: provides connection info to your app via env vars for the DB

Support for all kinds of crap including Docker



More info: <https://tutorialsdojo.com/aws-elastic-beanstalk/>

## API Gateway

Collection of resources and methods that are integrated with back-end HTTP endpoints, Lambda function or other AWS services.

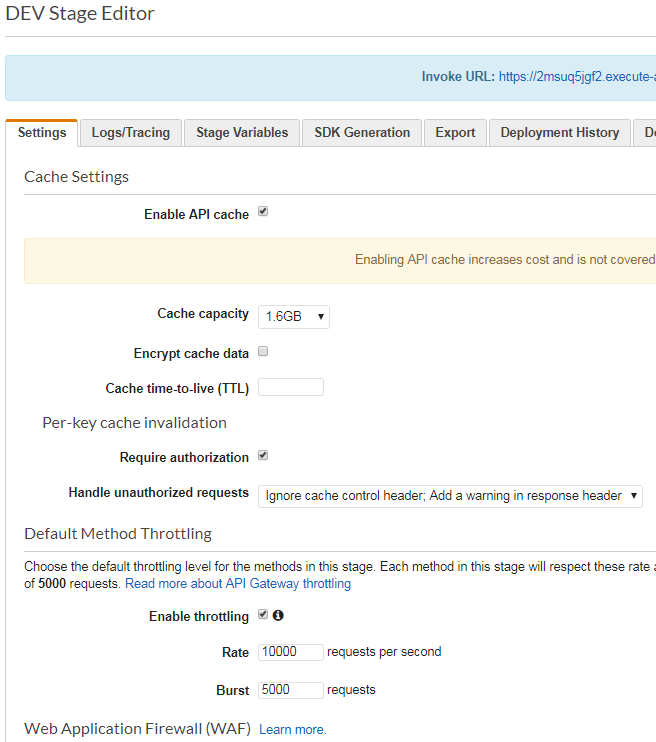
API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls and auto scales.

Gateway + Lambda it is the app-facing part of a serverless infrastructure.

Same origin policy: web app permits scripts contained in one web site to access data in a second, ONLY if they are in the same origin (domain) to prevent XSS attacks. Can use CORS to override this

You can add caching to API calls by provisioning an API Gateway cache and specifying its size in GB. The cache is provisioned for a specific stage of your APIs. This improves performance and reduces the traffic sent to your back end. Cache settings allow you to control the way the cache key is built and the TTL of the data stored for each method. API Gateway also exposes management APIs that help you invalidate the cache for each stage. Caching is available for REST APIs in API Gateway.

Throttling can be configured at multiple levels including Global and Service Call (method level). Throttling ensures that API traffic is controlled to help your backend services maintain performance and availability



Integration:

Outside of VPC:

AWS Lambda (most popular / powerful)

Endpoints on EC2

Load Balancers

Any AWS Service

External and publicly accessible HTTP endpoints

Inside of VPC:

AWS Lambda in your VPC

EC2 endpoints in your VPC

You pay only for the API calls you receive and the amount of data transferred out.

*Security*

IAM:

Great for users / roles already within your AWS account

Handle authentication + authorization

Leverages Sig v4

Custom Authorizer:

Great for 3rd party tokens

Very flexible in terms of what IAM policy is returned

Handle Authentication + Authorization

Pay per Lambda invocation

Cognito User Pool:

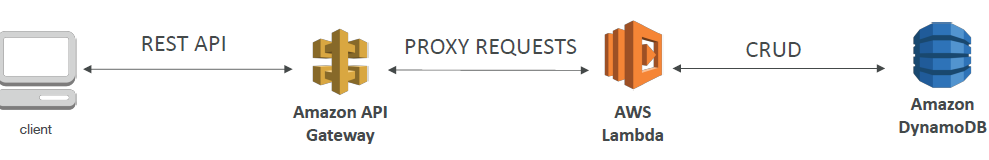
You manage your own user pool (can be backed by Facebook, Google login etc…)

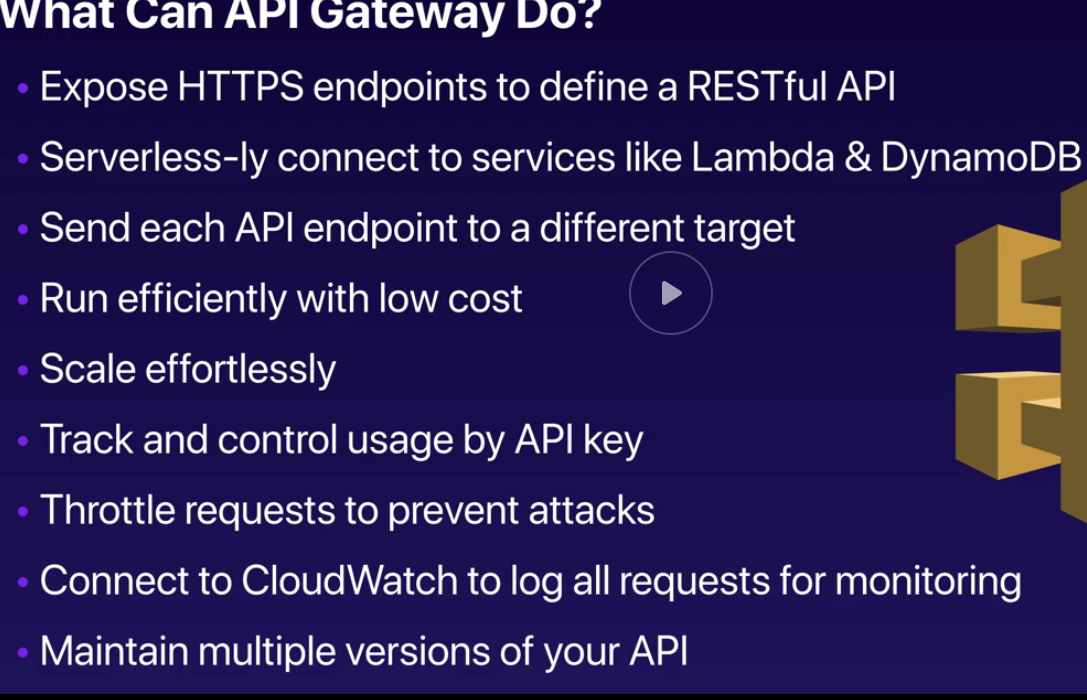
No need to write any custom code

Must implement authorization in the backend

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-api-gateway/>

<https://docs.aws.amazon.com/apigateway/api-reference/making-http-requests/>





## Cognito

*User Pools*

Create a **serverless** database of user for your mobile apps

Simple login: Username (or email) / password combination

Possibility to verify emails / phone numbers and add **MFA**

Can enable Federated Identities (Facebook, Google, SAML...)

Sends back JSON Web Tokens

**Can be integrated with API Gateway for authentication**

*Federated Identity Pools*

Goal:

Provide direct access to AWS Resources from the Client Side

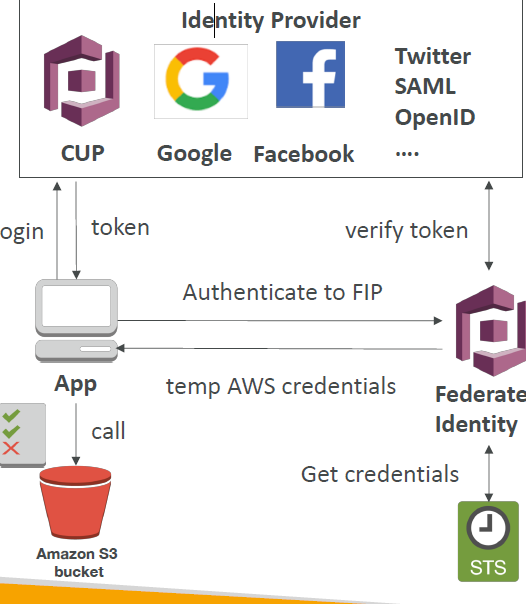
How:

Log in to federated identity provider or remain anonymous

Get temporary AWS credentials back from the Federated Identity Pool

These credentials come with a pre-defined IAM policy stating their permissions

Example: provide (temporary) access to write to S3 bucket using Facebook Login



## AD services

Simple AD is an inexpensive AD compatible service with common directory features. It is a standalone, fully managed, directory on the AWS cloud and is generally the least expensive option. It is the best choice for less than 5000 users and when you don’t need advanced AD features. **Does not integrate** with an on-premises Active Directory service

AD Connector is a directory gateway with which you can redirect directory requests to your on premise AD without caching any information in the cloud.

## ElasticSearch

Use ElasticSearch as a complement to another database you can search any field, even partially matches. **Search and indexing**

## KMS

Regional Service

The value in KMS is that the CMK used to encrypt data can never be retrieved by the user, and the CMK can be rotated for extra security

Encrypted secrets can be stored in the code / environment variables

If data > 4 KB, use envelope encryption

To give access to KMS to someone:

Make sure the Key Policy allows the user

Make sure the IAM Policy allows the API calls

Three types of Customer Master Keys (CMK):

AWS Managed Service Default CMK: free

User Keys created in KMS: $1 / month

User Keys imported (must be 256-bit symmetric key): $1 / month

Pay for API call to KMS ($0.03 / 10000 calls)

Encrypt Existing Requires migration for:

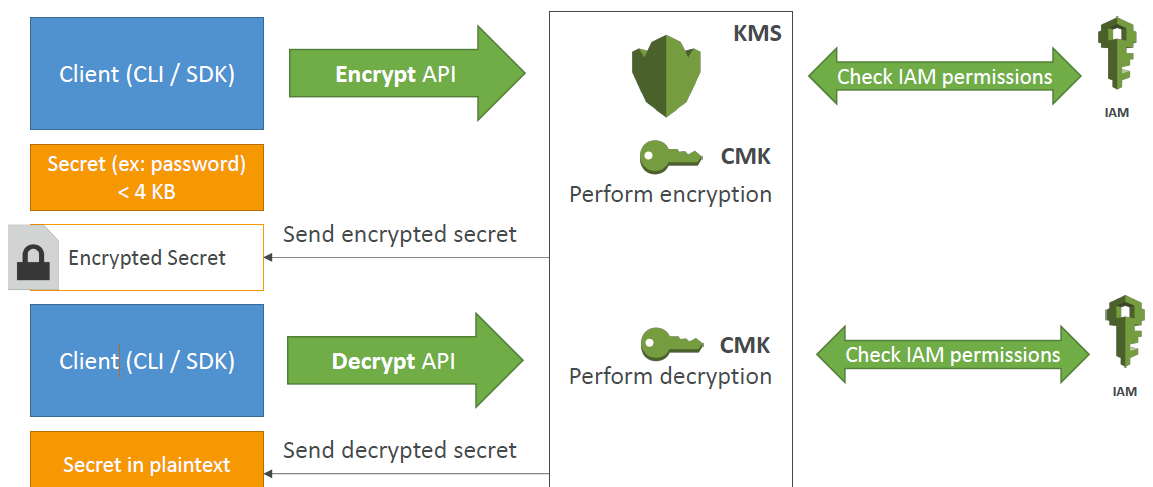
EBS Volumes

RDS databases

ElastiCache

EFS network file system

But not for S3



## Parameter Store

In the SM console

GetParameters or GetParametersByPath API

See Slide 388, 389

## Serverless Examples

Slides 315 - 345

# White Paper

<https://aws.amazon.com/architecture/well-architected/>

They now have a Well-Architected Tool

Best practices:

<https://aws.amazon.com/whitepapers/>

*Security Pillar:*

Automate responses to security events

Data protection: KMS, S3, etc.

Privilege management (IAM, STS, MFA,)

Infrastructure Protection (VPC, CloudFront, WAF)

Detective Controls (CloudTrail, Config, CloudWatch)

Response: IAM, CloudFormation, CloudWatch Events

*Reliability Pillar:*

Auto recover from failures and scale out to increase system availability

Foundations (IAM, VPC, Account Limits, Trusted Advisor)

Change Management (CloudTrail, AS, CloudWatch, Config)

Failure Management (CloudFormation, backup and recovery, S3 types, RT53)

*Performance efficiency Pillar:*

Compute and Monitor: (choose correct type, or lambda, monitor instances, demand match quantity (AS)

Database: read replicas, caching, proximity, monitor proximity and caching.

Space/Time trade off: CloudFront, ElastiCache, direct connect, RDS read replicas

*Cost Optimization Pillar:*

Budgets, Cost Explorer

Trade capex for operating exp. Economies of scale

Match supply to demand (AS, Lambda)

Cost effective resources: EC2 instance types, Trusted Advisor, Glacier

Expense aware (CloudWatch, SNS)

Optimizing over time (Trusted advisor, Billing Dashboard)

Use Tags

*Operational Excellence Pillar:*

Design Principles

Perform operations as code - Infrastructure as code

Annotate documentation - Automate the creation of annotated documentation after every build

Make frequent, small, reversible changes - So that in case of any failure, you can reverse it

Refine operations procedures frequently - And ensure that team members are familiar with it

Anticipate failure

Learn from all operational failures

Preparation: AWS config good tool to use for inventory of resources and AWS service catalog. CloudFormation

Operations: Codecomit, codedeploy, pipeline, SDKs, cloudtrail, CloudFormation, Cloudtrail

Evolve: CloudFormation, Codecomit, codedeploy, pipeline

Responses: CloudWatch alarms, SNS

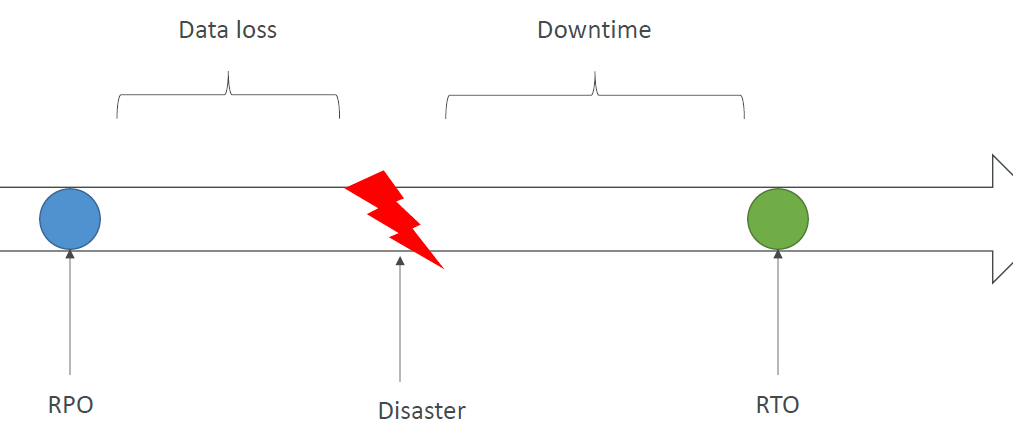
*Shared Responsibility:* https://aws.amazon.com/compliance/shared-responsibility-model/

*Disaster Recovery*

Different ways to use AWS as your DR site/strategy

*RTO* is the time it takes after a disruption to restore a business process to its service level and *RPO* acceptable amount of data loss measured in time before the disaster occurs

E.g. backup and restore has a high RPO cuz it takes too long



<https://tutorialsdojo.com/aws-well-architected-framework-disaster-recovery/>

<https://www.ecloudgate.com/Doc/DisasterRecovery_Overview>

Pilot light is a DR scenario in which a minimal version of an environment is always running in the cloud (like a heaters pilot lite).

Backup

EBS Snapshots, RDS automated backups / Snapshots, etc.…

Regular pushes to S3 / S3 IA / Glacier, Lifecycle Policy, Cross Region Replication

From On-Premise: Snowball or Storage Gateway

High Availability

Use Route53 to migrate DNS over from Region to Region

RDS Multi-AZ, ElastiCache Multi-AZ, EFS, S3

Site to Site VPN as a recovery from Direct Connect

Replication

RDS Replication (Cross Region), AWS Aurora + Global Databases

Database replication from on-premise to RDS

Storage Gateway

Automation

CloudFormation / Elastic Beanstalk to re-create a whole new environment

Recover / Reboot EC2 instances with CloudWatch if alarms fail

AWS Lambda functions for customized automations

Chaos

Netflix has a simian-army randomly terminating EC2

# Other

*CloudHSM*: Locating HSM appliances near your EC2 instances decreases network latency, which can improve application performance. It’s an appliance which u get to via ssh or rdp.

Can be used to generate, use and manage encryption keys in the cloud

*OpsWorks*= Chef and puppet

*AWS Config*: provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance

Can discover existing AWS resources, export a complete inventory of your AWS resources with all configuration details, and determine how a resource was configured at any point in time

*WAF/Shield*

*Type of condition you can add to add a rule:*

Cross-site scripting match conditions

IP match conditions

Geo match conditions

Size constraint conditions

SQL injection match conditions

String match conditions

Regular expression match conditions

AWS Shield is a managed DDoS protection service that safeguards applications running on AWS. It provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection

**Cheats**

CRR is async

Gateway stored uses async backup

Snapshot will asynchronously copy the data modified on the EBS volume

ALB/NLB route traffic to the target groups synchronously

DynamoDB creates table across 3 AZs automatically and synchronously

Redshift: backup asynchronously on S3

RDS multiAZ synchronously replicated across AZ

Read Replica: propagated to the replica async

SQS: async

# Solutions Diagrams