AWS Certified Solutions Architect Associate - SAA-C02

GET TO THE LAST POINT EDITED: fn+shift + F5

Exam Blue Print:

- 130 Minutes in Length
- 65 Questions (this can change)
- Multiple Choice
- 70% to pass. Results are between 100 and 1000 with 720 of passing score.
- Certification valid for 3 years.
- Scenario based questions.

Partner Program: To be part of the AWS partner program, which means you can create a consulting company to provide AWS services to your customers, you need to have a certain amount of certified profesionals.

There are 3 type of partner programs: Select, Advanced and Premier, and you will need the following amount for each one of them:

Partner	Practitioner Certs	Associate Certs	Professional/ Specialty Certs
Select	2	2	0
Advanced	4	4	3
Premier	10	10	10

So if you want to be an Advanced partner for instance, you will need 4 Practitioners, 4 Associate certs and 3 professional certs.

Identity and Access Management 101

101 Means that is an introductory topic for a lecture. This is used in UK and the States.

IAM:

- Centralized control of the aws account
- Shared access to your AWS account.
- Granular Permissions (you can access to this serivce but not to that service)
- Identity Federation (including Cognito [Facebook, Linkedin, Google, etc], or AD). You can log in into the AWS console using the same user and password used in your windows or your facebook, etc account.
- Multifactor Authentication.
- Provide temporary access to users/devices and services where necessary (AIM Roles)
- Allows to set up your password rotation policy.
- Integrates with many AWS services.
- Supports PCI DSS Compliance. Compliance framework. When you are taking credit card details you need to be compliant with the framework.

Key Terms:

- **Users:** End Users such as people, employees of an organization, etc.

- **Groups:** A collection of users. Each user in a group will inherit the permissions of the group. For instance you may have a group with only access to S3. You may have another group to access EC2. If the user is in that group it will inherit the permissions of that group.

- **Policies:** Are made up of Policy documents. Are made in JSON and give permissions as to what a User/Group/Role is able to do.

- **Roles:** You create a role and assign it to an AWS resource. Is a way to allow one part of AWS to do something with another part.You may give a EC2 instance permissions to write files into a S3 bucket, etc. Roles isues that are valid for short durations.

AIM Dashboard: Services -- Security, Identity, & Compliance -- AIM

There you can see the link to send your users to log in into the console. This link includes the Account ID, the user will need to introduce his user and password.

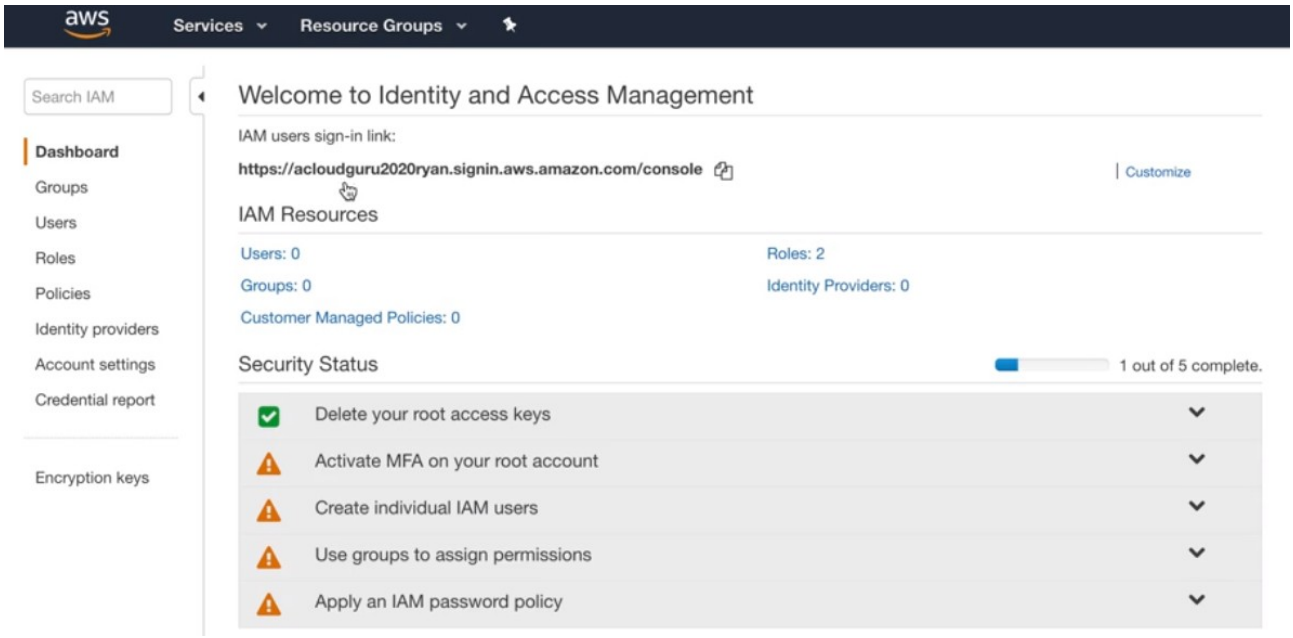
https://082396398680.signin.aws.amazon.com/console

You can customize this link too by creating an alias. This is DNS change so the alias will need to be unique in the world:

https://testing1010.signin.aws.amazon.com/console

Root Account: Is the email address that your first signed up with. With the root account you are in God mode, you are able to do anything in AWS.

<u>When you sign in for the first time you are recommended to do the following steps.</u>



- **Enable the 2FA:**
  Go to My Security Credentials (Top - Right, click on your user name) -- Activate MFA for your root account, click on Manage MFA -- Choose one of the 3 options, the app, the security key or other hardware.

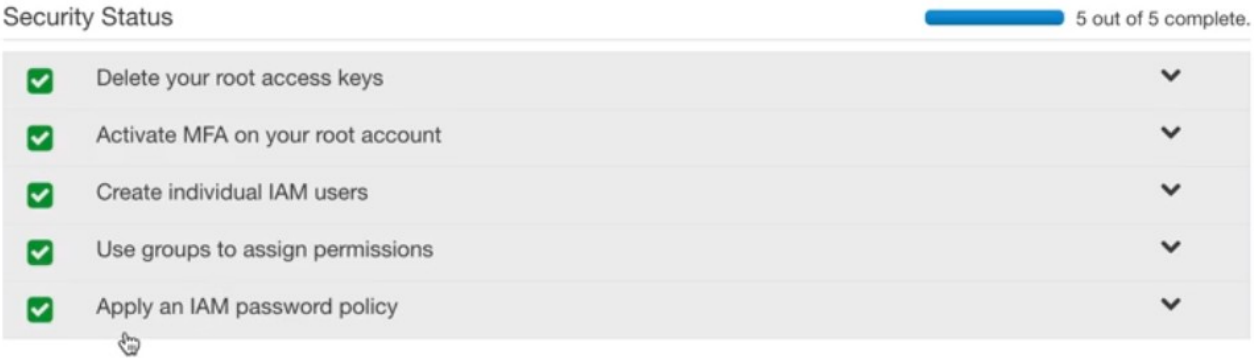
  If you choose to use an authenticator app is advisable to take a picture of the QR code and store it in a safe place. So if you lose your phone with the authenticaror app inside you will be able to enable it again by scaning the picture.

- **Create individual AIM users:**
  Click on Manage Users -- Add user -- Select AWS access type (where this new user will access): Programmatic (creates key id and security key id) and/or aws management console. Permissions: Create a Group, name it Developers and give it the permissions you want and hit Create Group, hit next until you create the user.

  IMPORTANT: Store the access key and secret key generated, you will need them to log in into the servers. You won't be able to retrieve that keys so you will lose it if you don't store them after creating the user. You can email them to the user too.

- **Apply an IAM Password Policy:**
  Manage Password Policy and enable the options you want, as much as better.

After applying everyting we have to see the following:



Change User Passwords:
If for some reason you need to change the user passwords, for the console or CLI access go to: Users -- Select your user -- Security Credentials

There you can enable/disable or change the console password, generate new access keys, or enable and manage the 2FA.

Roles:
Go to Roles -- Create role -- AWS service, EC2 -- search and select s3 full access (s3f) -- role name: S3_Admin_Access -- create role

Create a Billing Alarm:

Go to CloudWatch -- Billing -- click the Create alarm in the center of the page, DON'T use the Create alarm in the top righ (specific alarm) -- You have to see a page that shows a graphic and several options to configure

You can get the same page using the Create alarm in the top righ doing -- Create alarm (top right button) -- Select metric -- Billing -- Total Estimated Charge -- tic USD -- Select metric

Once in the Specify metric and conditions -- Define the threshold value to 10 USD -- you can define the Period in which cloudwatch will check the billing alarm, by default is 6 hours -- hit Next when you are done.

Select an SNS topic: Create new topic -- name it as *Billing_Alarm* -- Email endpoint that will receive the notification… → add there your email (you will receive one email there and you will need to confirm that you want to subscribe to it) -- Create topic

Once subscribed (by confirming the email) hit next -- Name and description,Define a unique name: *Billing Alarm* and Alarm description: *Send me an email if my bill goes over $10*

You will be redirected to a preview page, hit Create alarm.

You will appear in the billing dashboard and you will see your new created alarm. Will appear as INSUFFICIENT for a while because you choose to check it automatically every 6 hours.

If you didn't confirmed by email yet you will see Actions as pending, confirm the email and subscribe to it. Then go back to the console and hit the refresh button, you will see that now is not showing the pending message.

And that's it, you will receive an alarm everytime your bill goes over $10.

Exam Tips:
How can you get automatic notifications if the bill goes over $X?
By creating a billing alarm in cloudwatch, wich uses an SNS topic.

S3 101

Provides a secure, durable, highly-scalable object storage.

Object-based storage. Flat files, documents, pictures, movies, etc.

Data is spread across multiple devices and facilities.

Files can be from 0 Bytes to 5TB and there is unlimited storage.

Files stored in Buckets (folders).

S3 is universal namespaced, Bucket names must be unique globally.

Names are global but you are storing buckets locally:
If you create a bucket in the default Region, East Virginia, the bucket will have the domain name:
https://my_bucket.s3.amazonaws.com

If you create the bucket in another region the domain name will be:
https://my_bucket.eu-west-1.amazonaws.com → if you are in Ireland Region for example.

Exam Tips:
-What code you receive back uploading a file? HTTP 200 code.
-S3 is object based storage, not suitable to install an OS or a database.
-You can turn on MFA Delete → to confirm via MFA if you want to delete a file.
-Read S3 FAQs

Objects consists in:

- Key: The object name.
- Value: The data, made up of a sequence of bytes.
- Version ID: Important for versioning.You have a file with a name, like: my_file. If you upload another file with the same name S3 will keep both and you will be able to go back to the previous version.
- Metadata: Data about data you are storing. Example this object belongs to Finance.
- Subresources:
  - Access Control Lists
  - Torrent
  - etc

Data Consistency model for S3:

- Read after Write consistency for PUTS of new Objects: You upload a file to S3 you will be able to read it inmediatelly.

- Eventual Consistency for overwrite PUTS and DELETES (can take some time to propagate): If you overwrite a file you will be able to access the old object for a few seconds. After that few seconds you will access to the new file. If you delete it you will see it for a few seconds to, is not immediate. The changes can take a little bit of time to propagate.

S3 Guarantees:

- Built for 99.99% availability
- AWS Guarantee 99.9% availability
- AWS Guarantee 11 x 9s durability for S3 information.

S3 Features:

- Tiered Storage Available
- Lifecycle Management. Automatic migration of objects to other S3 Storage Classes
- Versioning
- Encryption
- MFA Delete
- Secure your data using ACL and Bucket Policies

S3 Storage Classes: https://aws.amazon.com/s3/storage-classes/

All of them using Availability Zones ≥3 except S3 One Zone - IA

- **S3 Standard:** 99.99% Availability. 11x9s Durability.

- **S3 - IA**, Infrequently Accessed: Less frequently accessed data but requires rapid access to it. Low storage fee but you are charged a retrieval fee.

- **S3 One Zone - IA**: Lower cost and only one AZ. Data will be lost if the AZ is destroyed.

- **S3 - Intelligent Tiering:** Introduced in 2018. Uses AI to move data to the most cost-effective access tier without performance impact.

- **S3 Glacier:** Archive data. Secure, durable and low cost. Retrieve data configurable between minutes to hours. You can pay a fee to retrieve data faster.

- **S3 Glacier Deep Archive**: Cheapest option to store data. Retrieve data up to 12 hours

S3 Charges:

- Storage
- Requests
- Storage Management Pricing
- Data Transfer Pricing
- Transfer Acceleration: Uses Edge Locations. Stores data close to the end users lowering the latency.
- Cross Region Replication Pricing: You can automatically replicate a bucket in another region for high-availability and dissarter recovery. As soon as you upload an object this is automatically replicated in the other region.

Create an S3 Bucket:

Create a default bucket: Storage - S3 - Create Bucket - Add unique domain name - Create Bucket.

Upload an object: Select your new bucket - Object tab, Upload - leave all default - Upload

The bucket and the image won't be reachable on inet.

Make Object public: Go to the bucket - Permissions - Block public access (bucket settings) , Edit - Untic block all public access - Save and confirm.

Now select the object you want to make public - Actions, Make public.

Go to the object Details, you will see the Object URL there, like
https://teest1234569.s3.amazonaws.com/carlos_enrique.png

Exam Tips:

- Buckets names shares common namespace (Domain). You can not have the same bucket name as someone else, have to be unique for the whole world.
- In the Management console you see the buckets globally but you have to store them in a particular AZ.
- Cross Region Replication, CRR: You can replicate the content of one bucket to another bucket in another Region automatically.
- You can change the Storage class and encryption of your objects on the fly.
- Remember the storage classes.
- Transfer Acceleration: Uses Edge Locations and aws backbone network to get the object close to the end user in the other side of the world. Example: We upload a large file in London and we want the people of New Zealand to access it with low latency. You use TA to cache the file in an edge location close to NZ.
- Restrict Buckets Access:
  - Bucket Policies: Applies across the whole bucket.
  - Object Policies: Applies to individual files.
  - AIM Policies to User/Group: Applies to users and groups.

S3 Pricing Tiers:

S3 Charges:

- Storage
- Requests and Data Retrievals
- Data Transfer Pricing
- Storage Management & Replication Pricing
- Transfer Acceleration: Uses Edge Locations. Stores data close to the end users lowering the latency.
- Cross Region Replication Pricing: You can automatically replicate a bucket in another region for high-availability and dissarter recovery. As soon as you upload an object this is automatically replicated in the other region.

S3 Storage Cost

- **S3 Standard:**
  - First 50 TB / Month: $0.023 per GB
  - Next 450 TB / Month: $0.022 per GB
  - Over 500 TB / Month: $0.021 per GB

- **S3 - Intelligent Tiering:**
  - Exacly the same that S3 Standard but you have access to S3 IA Tier which is always $0.0125 per GB, which is half of the price of s3 standard.
  - Monitoring and Automation cost, $0.0025 per 1000 objects. (Every 1000 objects you have to pay this)

- **S3 - IA**:
  - All storage / Month: $0.0125 per GB

- **S3 One Zone - IA**:
  - All storage / Month: $0.001 per GB

- **S3 Glacier:**
  - All storage / Month: $0.004 per GB

- **S3 Glacier Deep Archive**:
  - All storage / Month: $0.00099 per GB

Exam Tips: Which Tier of S3 should I use for a given scenario.

Order from expensive to chepeast:

1. S3 Standard
2. S3 - Infrequent Access
3. S3 - Intelligent Tiering → Uses S3 Standard and S3 - IA
4. S3 One Zone - IA → Risk of losing data if the AZ dies.
5. S3 Glacier
6. S3 Glacier Deep Archive

S3 Security and Encryption:

By default all newly created buckets are private. You can set up access control to your buckets using:

- Bucket Policies → applied on Buckets
- Access Control Lists → applied on Objects

Buckets can be configured to create access logs which logs all requests made to th S3 buckets.This can be sent to another bucket and even another bucket in another account.

Types of Encryption: Exam Tip

- Encryption In Transit: When the browser uses https. Achieved by SSL/TLS. Encrypted between my computer and the server.

- Encryption At Rest: Is where we encrypt the data that will be stored.

    - Server Side: Where AWS helps you to encrypt the object:

        - S3 Managed Keys - **SSE-S3** (Server Side Encryption). AWS manages the keys

        - AWS Key Management Service, Managed Keys - **SSE-KMS**. Managed by me and AWS together.

        - Server Side Encryption with Customer Provided Keys - **SSE-C**. You give AWS your keys, you can encrypt your S3 objects as well.

    - Client Side: You encrypt the object and then upload it to S3.

Encrypt an Object:
S3 - Select the bucket you want - Select the object you want to encrypt - Server-side encryption settings - Edit - Amazon S3 key (SSE-S3) - Save

Encrypt a Bucket: → Much more efficient
Select the bucket you want to encrypt - Properties - Default encryption, Edit - Enable, Amazon S3 key (SSE-S3) - Save changes

Now all objects that we put into this bucket will be encrypted.

S3 - Versioning:

Exam Tips:

- Stores all versions of an object (including all writes and even if you delete an object)
- Great backup tool.
- Once enabled Versioning can NOT be disabled, only suspended. You have to remove the bucket and create a new one to see it as disabled. If you make it suspended the new objects won't be versioned but the old objects will keep it's versions.
- Integrates Lifecycle rules
- It comes with MFA Delete capability, which can be used to provide an additional layer of security.

Create a bucket and upload an object:
Create the bucket as public and enable the Bucket Versioning.

Upload an object several times:
Create a notepad called Document.txt with the text: This is version 1.0

Upload it to the new bucket.

Modify the text to: This is Version 2.0 and upload it

Tick the List versions button, you will see both documents there, the lastly updated is the current document and the first document uploaded is the oldest version.

You have to make each document public to be reachable through internet. Every new version will need to be made public.

Delete an Object:
With the List versions unticket, you only can see the last object and not the old versions, click on Delete.

Type delete to confirm it.

This is NOT deleting the object, just made it dissapear from the list of objects.

Tick the List versioning and you will see it as Delete marker. Delete is hiding the current version from your view but it is still there marked as Delete marker.

Restore an object:
Select the Delete marker object and hit Delete. This will delete the Delete marker and will make the object available again.

Now you can see the document through internet again, the text will show:
This is version 3.0. Last version

Suspend Versioning:
This prevents the creation of new objects but preserves the old ones.

Delete objects:
You have to go one by one, each version of the object.

Tick the version you want to delete and hit Delete button -- confirm typing the text permanently delete -- Delete objects

You can tick all the versioned objects to delete them all together.

Lifecycle Management with S3

Exam Tips:

- What lifecycle is: Automates moving your objects between the different storage tiers.
- It allows you to delete objects to after X time.
- Can be used in conjunction with versioning, to move old versions of the objects.

Go to the bucket created in the previous lecture -- Management -- Create lifecycle rule:

Add a name: Main Lifecycle Rule

Choose a rule scope:

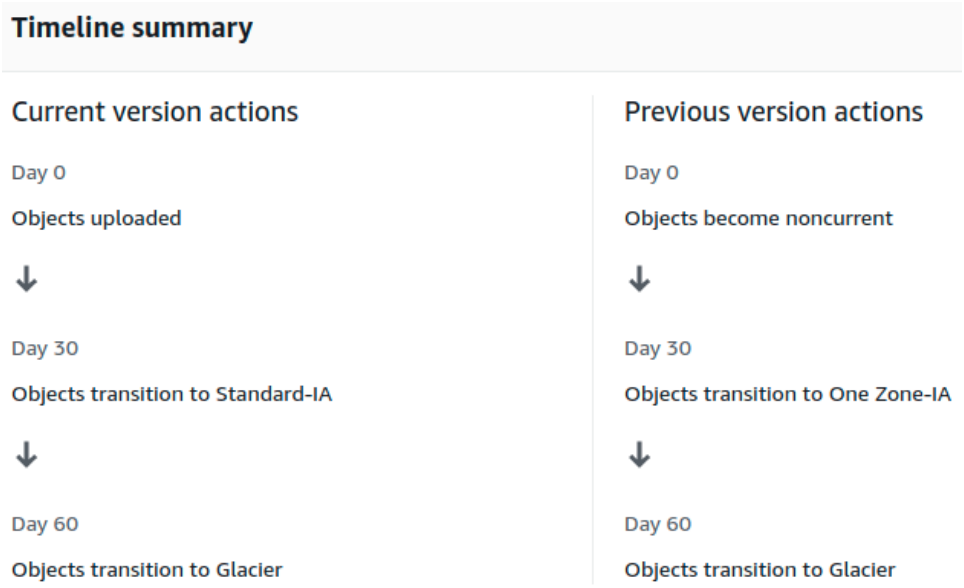
- Limit the scope of this rule using one or more filters
- This rule applies to all objects in the bucket (will ask you to ack)

Lifecycle rule actions:

- Transition current versions of objects between storage classes: Move the object to another storage class in X days.
- Transition previous versions of objects between storage classes: Move an old version of the object to another storage class in X days.
- Expire current versions of objects
- Permanently delete previous versions of objects: Delete an old version after X days
- Delete expired delete markers or incomplete multipart uploads
- When a lifecycle rule is scoped with tags, these actions are unavailable.

Transitions: You can add more transitions, example: After 30 days move the current version to Standard-IA and after 60 days move it to Gracier.

At the end you have a a Timeline summary informing of the rules created:



You can see, edit, disable, delete the rule going to Management in the bucket.

S3 Object Lock & Glacier Vault Lock

You can use S3 Object Lock to store objects using a write once, read many (**WORM**) model.
It helps you to prevent objects to being deleted or modified for a fixed amount of time or indefinitely.

You can use it to meet regularoty requirements that require WORM storage, or add extra layer of protection against object changes and deletion.

Modes:

- Governance Mode: You can grant some users special permissions to delete, alter an object or alter the retention settings. The rest of the users can not do anything.

- Compliance Mode: No one, not even the Root, can modify or delete the object version.It's retention mode **can NOT be changed** and it's retention period can not be shortened by no one, ensuring that the object version won't be deleted or modified during the **Retention Period.**

Retention Periods:
Is the time that an object version will be protected against overwritten/delete actions. You set up the retention period.

At the end of the RP the object version can be overwritten or deleted unless you also placed a Legal Hold on the object version.

When you place a RP, AWS S3 stores a timestamp in the object version's metadata to indicate when the RP expires.

Legal Holds:
Prevents an object version from being overwritten or deleted. Like the RP.

However a Legal Hold doesn't have an associated retention period and remains in effect until removed the legal hold.

Legal Holds can be placed and removed by any user who has the s3:PutObjectLegalHold permission.

S3 Glacier Vault Lock: Is a way of locking objects into Glaciar similar to S3 Object Lock.

You can lock a Vault Lock policy with WORM for example and lock the policy for future edits. Once locked the policy can no longer be changed.

Exam Tips:

- Use S3 Object Lock to store objects using a WORM model, write once, read many.
- Object locks can be on individual objects or applied across the bucket as a whole.
- Object locks ome in two models:
  - Governance mode: The object can be modified if you have special permissions.
  - Compliance mode: No one can modify the object. You habe to wait utill the Retention Perion set is over.
- S3 Glaciar Vault Lock: To lock Objects inside glacier tier.

S3 Performance

How can we design our applications to maximize their performance with S3.

Think that you can access S3 several times at the same time, in parallel, is NOT like a network storage server that the access is a serial connection.

So you can spread your upload or download requests in parallel in order to gain speed.

S3 Prefixes:

A prefix is a subfolder. bucket/**folder1/sub1**/file.png → folder1/sub1/ is the prefix.
A prefix is anything between the bucket and the file. You can have as many prefix you need.

This is good to improve performance if you divide into different prefix a heavy file or folder.
S3 has an extremely low latency, by default 100-200 ms. You get the first byte at this speed.

You can achieve a high number of requests per second per prefix:

- 3,500 PUT/COPY/POST/DELETE
- 5,500 GET/HEAD

Example:

If you have a heavy file and you spread it into 4 prefix evenly, like:

bucket/**folder1**/file.png
bucket/**folder2**/file.png
bucket/**folder3**/file.png
bucket/**folder4**/file.png

you can achieve 22,000 requests per second for GET and HEAD

S3 - KMS Limitation:

If you have SSE-KMS encryption on your objects you may be impacted by the KMS limits.

- Uploading a file it calls the GenerateDataKey KMS API
- Downloading a file it calls the Decrypt KMS API

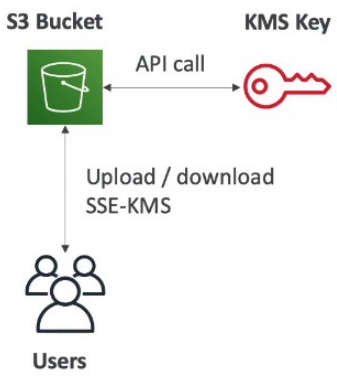
These two requests will counts towards the KMS quota

By default KMS have a quota on number of requests per seconds and based on the region can be 5500, 10000, 30000 reqs/s

You can NOT change that quota.

If you pass this quota you will be throttled. So you need to know if KMS will block your performance in S3.

The quotas are pretty big for normal usage but you still need to know if you have many files in high usage in your S3 buckets.



S3 Performance, how to optimize it: UPLOADS

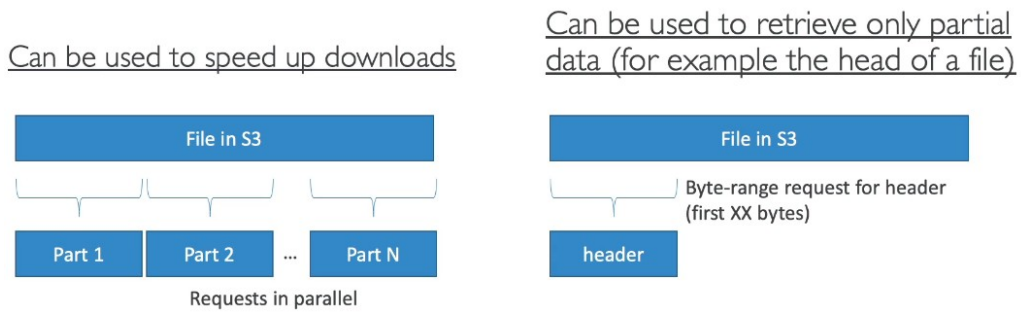
- **Multi-Part upload:** Parallelize uploads, which will speed up the transfer to maximize the bandwithd. You do this through AWS CLI.
    - Recommended for files up to > 100 MB
    - Required for files over > 5 GB.

  Example: We have a big file, we will divide it in parts using AWS CLI, each of this parts will be uploaded in parallel to S3. Once all the parts are uploaded S3 will put all toguether again.

- **S3 Transfer Acceleration (upload only):** Move the file you want to the closest edge location of the end user. Compatible with multi-part upload.

GETing files: DOWNLOADS

- **S3 Byte-Range Fetches:** Parallelize GETs by requesting specific byte ranges for your files. Better resilience in case of failures.
    - Can be used to speed up downloads.
    - Can be used to retrieve only partial data (for example the head of a file)



S3 Select and Glacier Select:

S3 Select:
Enables your application to retrieve data from an object by using simple SQL expressions.

By doing this you can achieve drastic performance increases in many cases, you can get as much as 400% improvement.

Esentially is pulling your data using SQL.

Example: Let's say that we have a .csv file inside a .zip as an object in S3.

- Without S3 Select to get the data inside this .zip we need to download the file, decompress it, open the .csv and process the entire .csv to get the data we need.

- With S3 Select we just write a simple SQL query which directly goes in and grabs the data, only downloading the data that we need. This means you're dealing with an order of magnitude less data, which improves the performance.

Gracies Select:
Is very similar to S3 Select.

High regulated industries, like financial services, heathcare, and others, write data directly to Gracier to satisfy compliance needs.

They use Glacier to save storage costs.

Glacier Select allows you to run SQL queries against Glacier directly.

Exam Tips:

- S3 Select is used to retrieve only a subset of data from an object using simple SQL expressions.

- Get data by rows or columnns using simple SQL expressions.

- Save money on data transfer and increase speed.

AWS Organizations & Consilidated Billing:

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

The root aws account is the master account, the one that manages the rest of accounts.

The best practices are to use the root/master account for billing only and without delpoying any resources in there.

AWS Organizations allows you to have multiple aws accounts and be able to centrally manage them.

Consolidated Billing: AWS Organizations provides consolidated billing.

This works automatically when you create an aws organization with several accounts on it.

In AWS the more than you use the LESS than you pay.

If you have several accounts into aws organizations, aws will agreagate all the accounts bills in one.

This way you will have more discounts in your final bill rather than having all the aws accounts separated.

Example: You have 5 aws accounts that are using a lot of S3 space. If you include them into aws organizations the consolidated billing will see the 5 accounts as only 1 and will sum the space they are using in S3.

Let's say each account uses 10 TB of space. You will have a better discount if one single account uses 50 TB of space (the sume of all the single accounts 10 TB) rather than having 5 accounts paying for 10TB of space separatelly.

Each month aws charges the root/management account for all the members accounts inside the aws organization in a consolidated bill.

Advantages:

- One bill per aws account
- Very easy to track charges and allocate costs
- Volum pricing discount

Exam Tips:

- Always enable multi-factor authentication on root account
- Always use strong and complex password on root account
- Paying account should be used for billing purposes only. Do not deploy resources on it.
- Enable/Disable AWS services using Service Control Policies (SCP) either on OU or on individual accounts.

Sharing S3 Buckets Across Accounts:

Exam Tips:

3 Different ways to share S3 buckets across accounts

- Using Bucket Policies & AIM (applies across the entire bucket). Programmatic Access Only.
- Using Bucket ACLs & AIM (individual objects). Programmatic Access Only.
- Cross-account AIM Roles. Progammatic AND Console access.

Cross Region Replication:

- You can replicate the content of a bucket into another Region.

- Use cases: You have log files in your origin region and you want to keep them into the target region as a backup. You can block the access to the target bucket to avoid those logs to be modified.

- The previous content of the origin bucket won't be replicated to the target bucket.
  Only if you upload a new object to the origin bucket it will be replied to the target bucket.

- Permissions: The teacher says that the permissions are not traveling to the target bucket. So if you have read permissions in one object in origin, those permissions won't be in the target. But I was able to upload a image with read permissions and the target was having the same ones. Looks like is replicating the permissions.

Exercise: Replicate one S3 object into another Region.

-- Create a bucket in your region, you can call it test-for-cross-replication-us-east1 and add an object inside, like an .txt saying hello. Make all public (untick *Block all public access* box) and enable versioning control.

-- Create the destination bucket: Call it → test-for-cross-replication-target-bucket-tokyo
Chose the Tokyo region. Make all public. Leave everyone else as default and create it.

-- Go to the your region bucket (...us-east-1)-- Management -- **Replication rules**, Create replication rule:

- Replication rule name: replicatetojapan
- IAM role: create a new role
- Source bucket: This rule applies to all objects in this bucket
- Destination: Chose a bucket in this account -- Browse S3, select the tokyo bucket. Enable bucket versioning.
- Destination storage class: Change it if you want.
- Leave the rest as default and hit Save.

-- Add a new version of the .txt file into the origin bucket, you will see that the file is appearing in the target bucket. For images will take a minute or so.

Exam Tips:

- Versioning must be enabled on both, the source and the destination buckets.
- Files in an existing bucket are not replicated automatically.
- All subsequent updated files will be replicated automatically.
- Delete markers are not replicated.
- Deleting individual versions or delete markers will not be replicated.

WHAT IS DELETE MARKER → when you delete something and is still there, check this

S3 Transfer Acceleration:

Exam Tips:

The users uploads their big files to the closest edge locations.

Then aws utilizes the CloudFront Edge Network to transfer your uploads to S3, providing much more speed.

You can use a distinct URL to upload directly to an edge location, like mydomain.s3-accelerate.amazonaws.com

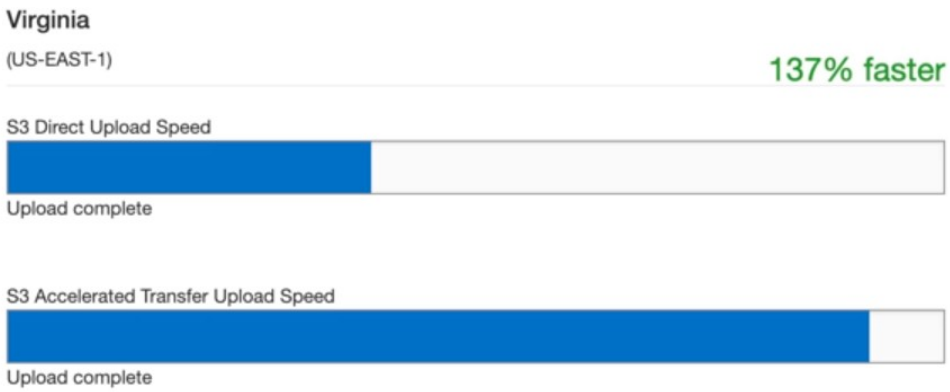
S3 Transfer Acceleration Tool: To compare the speed you get when you upload an s3 object with or without transfer acceleration into different regions.

Your browser uploads a s3 object in multiple plarts to different regions.

In the image below we uploaded a s3 object to Virginia from London.

The first upload is directly to S3 and the speed is lower.

The second one is using transfer acceleration and the speed is much higher, 137% much faster.



AWS DataSync:

Is a way of syncronizing your data.

Seamlessly and securily connects to S3, AWS EFS and AWS FSx for win servers to copy data and metadata to and from AWS.

Allows you to move large amounts of data into aws.

Typically you install the DataSync agent on a server that connects to your NAS or file system and then copy data to aws and Write data from aws.

Automatically encrypts your data and accelerates transfer over the wide area network.

Performs automatic checks in-transit and at-rest.

Exam Tips:

- Used to move large amounts of data from on-prem to AWS
- Used with NFS and SMB compatible file systems.
- Replication can be done hourly, daily, or weekly.
- Install the DataSync agent to star the replication (typically on prem).
- Can be used to replicate EFS to EFS.

CloudFront:

Is a CDN, Content Delivery Network.

Delivers webpages and other web content to an user based on:

- The geographic locations of the user.
- The origin of the webpage.
- A content delivery server.

CloudFront: Can be used to deliver your entire website, including dynamic, static, streaming, and interactive content using a global network of edge locations.

Requests to your content are automatically routed to the nearest edge location, so the content is delivered with the best possible perfomance.

Edge locations: Where the content will be catched. This is separated to an AWS Region/AZ.

Origin: The origin of all the files that the CDN will distribute. Can be an S3 Bucket, an EC2 instance, an Elastic Load Balanced, or Route53.

Distribution: Name given to the CDN which consists of a collection of Edge Locations.

How this works:
We have our servers in London.

A user in China wants to watch a video. This user will query to the closest Edge Location.

If the Edge Location does not have this video will download it from London to the Edge Location in China and will be cached there, is gonna be catched during a TTL which is in seconds, we can configure it.

The first user will need to wait until the video is downloaded to the Edge Location but the nexts users won't need to wait that much time because will be able to see the cached video from the China Edge Location.

Type of Distributions:

- Web Distribution: Typically used for Websites
- RTMP - Used for Media Streaming. This is getting discontinued in 2020/2021

Edge Locations are not just READ only, you can WRITE to them too. Example: put an object on to them.

You can clear chached objects but you will be charged for it. Example: You upload a new video but your users are still getting the old video, you can clear the cache but you will be charged for it.

Create a CloudFront Distribution:

We have to have a bucked and an object there, like a picture.

Creating a Distribution:
Go to: Network & Content Delivery -- CloudFront -- Create Distribution -- Web, Get Started -- Origin Domain Name: Select the URL of your bucket -- Leave the rest as default -- Create distribution.

Creating or dissabling (you need to dissable before deleting) the Distribution will takes time, minutes or even one hour.

Options to configure when creating the Distribution (may be asked in the exam):

Restrict Bucket Access: Yes if you want your users to access the content using the CloudFront URLs, not the S3 ones. This is useful wheb you are using signed URLS or signed cookies to restrict your content.

TTL: Time to live of the cached object in the edge location.

Restrict Viewers Access (Use Signed URLs or Signed Cookies): Media companies like Netflix want to make sure that your payed to have access to that content. You can restrict access using signed urls or signed cookies.

AWS WAF Web ACL: You can put a web application firewall.

Once the Distribution is created:

Get the Domain Name: d29jz5cqpdpiwa.cloudfront.net → in this example

Go to S3 -- pick the name of an object in the bucket and add it at the end of the Domain Name, like:

d29jz5cqpdpiwa.cloudfront.net/no_permissions_mountain.jpeg

You will access the image using CloudFront and should do it faster.

Now we know that is working, the CloudFront Distribution is loading objects from S3 using the generated URL.

Modify your Distribution settings:

Go to Network & Content Delivery -- CloudFront -- Tick your distribution,Distribution Settings -- Invalidations -- Create Invalidation: you can invalidate a particular path of the S3 bucket if you want, this means that what you have invalidated is no longer going to be in the edge locations. Example: You pushed some data and you see that is not showing up correctly, you can invalidate it.

Delete a Distribution:
Tick the one you want to delete -- Disable -- Yes, Disable -- This will take some time.

Once Disabled do the same, tick the Distribution and hit Delete.

Exam Tips:

The full CloudFront previous lesson.

CloudFront Signed URLs and Cookies:

Scenario: You have a site in which you want to restrict access to media content, premium content. Only subscribed users are allowed to access your videos. Websites like Netflix, or e-learning platforms for example.

You will need to use signed URLs or signed Cookies to achieve this.

You have to configure CloudFront to only allow Signed URLs or Cookies.

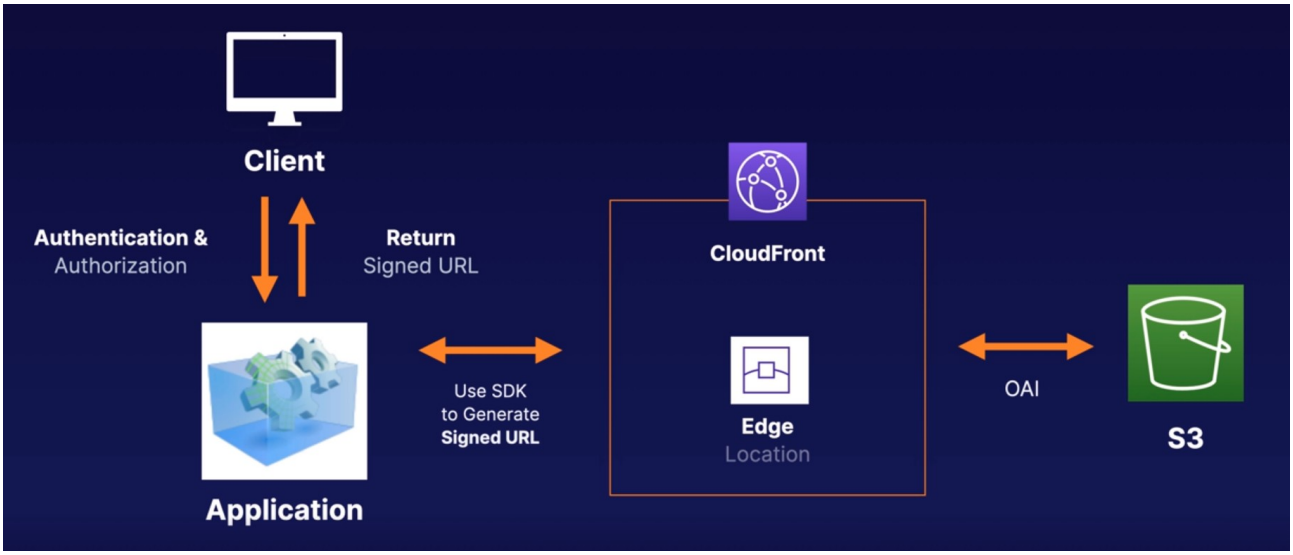
You can use:

- **Signed URLs:** A signed url is for individual files → 1 file = 1 URL
- **Signed Cookies:** A signed cookie is for multiple files → 1 cookie = multiple files

Signed URL Policies: To create a signed url or cookie we have to attach a policy, which can include:

- URL expiration
- IP ranges
- Trusted signers (which aws account can create signed URLs)

How this works:



1. We have our CloudFront Distribution with our Edge Locations.

2. That's connecting into our backend, our Origin, which in this case is S3. Which is using an OAI (Origin Access Identity) to do that authentication. The users will only be able to access CloudFront, not the S3, EC2, ELB, Route53.

3. The Client logs in into the Application by entering an username and password (Authentication & Authorization).

4. The app uses the CloudFront SDK to generate a Signed URL or Cookie and is returning that back to the Client.

5. The Client can then use CloudFront to access the file in S3/EC2/ELB/etc

CloudFront Signed URL features:

- Can have different origins. Does not have to be EC2, could be S3, ELB,etc.
- When you generate that Key-pair (signed URL), is account wide and managed by the root user
- Can utilize all CloudFront caching features
- Can filter by date, path, IP, address, expiration, etc.

S3 Signed URL features:

- When you are not using CloudFront where people have direct access to S3.
- Issues a request as the AIM user who creates the presigned URL.
- You have the same permissions as the AIM user who created the the signed url.
- S3 signed url have a limited lifetime.

CloudFront signed URL vs S3 signed URL:

If the user needs access to a single S3 bucket is better to grant him access with a S3 signed URL
If the user needs to access to other services is better to use a CloudFront signed URL.

Exam Tips:

- Use signed url/cookies when you want to secure content so that only the people you autorize are able to access it.
- Signed URL for individual files. 1 file = 1 URL
- Signed Cookie is for multiple files. Multiple files = 1 Cookie.
- If the Origin is EC2 use CloudFront.
- If the Origin is S3 and you only got a single file use S3 signed URL

Snowball:

What is Snowball:
Is a petabyte-scale data transport solution that uses secure appliances to transfer large amount of data into and outo AWS.

Why to use it:
Addresses common challenges with large-scale data transfers including high network costs, long transfer times, and security concerns.

Transfering data with snowball is simple, fast, secure, and can be as little as one-fifth the const of high speed internet.

Security:
Uses multiple layers of security, including tamper-resistant enclosures, 256 bit encryption, and TPM (trusted platform module).

Once the data has been transferred, AWS performs a software erasure of the snowball appliance.

Sizes:
50 TB, 80 TB and 100 TB (Snowball Edge)

Snowball Edge:
Comes with onboard storage and computing capabilities.

A part for transfer data in/to AWS you can use Snowball Edge as temporary storage tier for large datasets, or to support local workloads in remote ofline locations.

Is like to have a mini AWS at your disposal.

Snowmobile:
Is an Exabyte-scale data transfer service used to move extremely large amounts of data to AWS. You can tranfer up to 100PB.

You can transfer an entire Datacenter if you need.

Snowmobile is a container pulled by a semi-trailer truck.

Exam Tips:

- What Snowball is
- You can import to S3
- You can export from S3

Storage Gateway:

What is it:
Hybrid service that connects on-prem software appliance with cloud based storage to provide a seamless and secure connection between on-prem IT environment and AWS storage infrastructure.

The service enables you to securily store data to the AWS cloud for scalable and cost-effective storage.

Storage gateway is a virtual or physical device in your datacenter that is going to replicate your data into AWS.

Exam Tips:

- **File Gateway**: A way to storing flat files on S3. This is NFS/SMB.
- **Volume Gateways:** Uses iSCASI. Is a way of storing copies of your hard disks or virtual hard disk in the cloud.
    - Stored Volumes: Entire Dataset is stored on site and is asynchronously backed up to S3 in the form of a EBS snapshot.
    - Cached Volumes: Entire Dataset is stored on S3 and the most frequently accessed data is cached on site.
- **Gateway Virtual Tape Library:** Is a virtual tape library. You can get physical storage gateway appliances.

Athena vs Macie:

Ahtena: Interactive query service which enables you to analyse and query data located in S3 using standard SQL.

Allows you to turn S3 into a giant DB. You can query S3 using standard SQL.

- Serverless, nothing to provision, pay per query / per TB scanned
- No need to set up complex Extract/Transform/Load (ETL) processes
- Works directly with data stored in S3

Use Cases:

- Can be used to query log files stored in S3. e.g. ELB logs, S3 access logs, etc
- Generate busines reports on data stored in S3
- Analyse AWS costs and usage reports
- Run queries on click-stream data

Macie: Security service which uses Machine Learning and NLP (Natural Language Processing) to discover, classify and protect sensitive data stored in S3.

- Uses AI to recognise if your S3 objects contain sensitive data such as PII
- Dashboards, reporting and alerts
- Works directly with data stored in S3
- Can also analyse CloudTrail logs
- Great for PCI-DSS (if you take credit card payments on your website) and preventing ID theft

PII: Personal Identifiable Information.

- Personal data used to establish an individual's identity.
- This data could be exploited by criminals, used in identity theft and financial fraud.
- Home address, email address, Social Security Number.
- Passport number, driver's license number.
- D.O.B, phone number, bank account, credit card number.

Exam Tips:

Athena:

- Interactive query service
- Allows you to query data located in S3 using standard SQL
- Serverless
- Commonly used to analize log data stored in S3

Macie:

- Macie uses AI to analize data in S3 and helps identify PII
- Can also be used to analyse CloudTrail logs for suspicious API activity
- Includes dashboards, reports and alerting
- Great for PCI-DSS compliance and preventing ID theft

Creating a static website on S3: Bucket policy

Grant public reading permissions for all the objects inside a bucket:

S3 -- Bucket permissions -- Bucket policy -- paste this code adding the name of the bucket.

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Sid":"PublicReadGetObject",
    "Effect":"Allow",
    "Principal": "*",
    "Action":["s3:GetObject"],
    "Resource":["arn:aws:s3:::<MY_BUCKET>/*"]
  }]
}
```

With this policy you will grant reading permissions to everybody on internet.
Won't be necessary to grant permissions one by one to the objects.

/* → after MY_BUCKET, this means that the policy applies to all the objects inside my bucket.

This policy works with the bucket URL not with the url's of each object.
To get the bucket url go to bucket properties, at the bottom.

EC2:

EC2 101 Elastic Compute Cloud:

01/26/2021

Exam Tips:

EC2: Elastic Compute Cloud. Web serivice that provides resizable compute capacity in the cloud. AWS EC2 reduces the time required to obtain and boot a new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.

Pricing types:

- On demand
- Reserved
- Spot instances: If terminated by aws you won't be charged for a partial hour. If you terminate it you will be charged for any hour in which the instance ran.
- Dedicated

Launch an instance:

- Termination protection is turned off by default, you must turn it on.
- On an EBS-backed instance, the default action is for the root ebs volume to be deleted when the instance is terminated.
- EBS root of your default's AMI can be encrypted. You can use bit locker or others to encrypt it to and you can do it when creating the AMI in the console or by api.
- Additional volumes can be encrypted.

Security Groups:

Security Groups: Are like the local firewall. Let's you to open and close in/out ports.

Statefull: Security groups are Statefull. Means that if you enable something for the inbound, outbound is enabled automatically for that port.

Example: I have a website and as inbound rule I have opened the port 80.

If I delete the Outbound rule, by default allowing all traffic, I will still be able to access the website and send request from the out of the server.

NACL: Network access control lists are stateless. You have to create both inbound and outbound rules to allow traffic.

Exam Tips:

- All Inbound traffic is **blocked** by default.
- All Outbound traffic is **allowed** by default.
- Changes to Security Groups take effect immediately.
- You can have any number of EC2 instances within a security group.
- You can have multiple security groups attached to EC2 instances.
- Security Groups are STATEFUL. If you create an inbound rule allowing traffic in, that traffic is automatically allowed back out again.
- You can not block specific IP addresses using Security Groups, instead use Network Access Control Lists.
- You can NOT specify DENY rules, only allow them. NACL (VPC) allows this.

EBS 101:

EBS: Elastic Block Store. Esentially is a virtual hard disk.

Provides persistent block storage volumes for use with EC2 instances in the aws cloud.

Each EBS volume is replicated within its AZ to protect you from component failure, offering high availability and durability.

Flavours:

- General Purpose (SSD)
- Provisioned IOPS (SSD). For really fast imputs/oputputs per second. Databases
- Throughtput Optimised Hard Disk Drive. Physical HD (magnetic). Big Data
- Cold Hard Disk Drive (magnetic). File servers
- Magnetic. Legacy.

Use Cases:

Solid-State Drives (SSD)			Hard disk Drives (HDD)		
Volume Type	**General Purpose SSD**	**Provisioned IOPS SSD**	**Throughput Optimized HDD**	**Cold HDD**	**EBS Magnetic**
Description	General purpose SSD volume that balances price and performance for a wide variety of transactional workloads	Highest-performance SSD volume designed for mission-critical applications	Low cost HDD volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads	Previous generation HDD
Use Cases	**Most Work Loads**	**Databases**	**Big Data & Data Warehouses**	**File Servers**	**Workloads where data is infrequently accessed**
API Name	gp2	io1	st1	sc1	Standard
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB	1 GiB-1 TiB
Max. IOPS**/ Volume	16,000	64,000	500	250	40-200

EBS Volumes and Snapshots:

The ec2 instance and the EBS have to be in the same AZ.

You can change the size or even the volume type, like from gp to io on the fly, is NOT necessary to shut down the server, but will affect it anyway so is recommended to do it our of hours.

How to move one EBS from one AZ to another:

1. Find the root device volume, where the OS is installed, is the one that have a snapshot.

2. Create a snapshot: Volumes -- Actions -- Create Snapshot -- Name it and Create Snapshot. Will take a bit of time to be created. You can find it in EBS -- Snapshots

3. EBS -- Snapshots -- Actions -- Create Image -- Name it, Virtualization type: leave Hardware-assisted virtualization -- create

4. Images -- AMIs -- You can see there your image. Hit Launch, you can launch an instance as usual using your image.

5. Now in the configure instance details you can change the AZ and run the instane in the one that you need.

Step 3: When you select the virtualization type you have two options, the one by default, Hardware-assisted virtualization (hvn) and Paravirtual.

If you choose paravirtual you will have less instances types to select when you create an instance from that image so is better to select hvn. For example, with pv you can not select t2.micro

Copy the Snapshot / Image into another Region:

EBS -- Snapshots -- Actions -- Copy AMI -- Destination Regions, select the one you want
The same for images.

Therefore you can send a snapshot/image to another region and then there you can choose in which AZ you want to install it.

Terminating instances:
When you terminate an instance the root EBS volume will terminate too, but by default the other volumes that may have that instance won't be terminated, you will need to eliminate them manually. Or you can specify to terminate them altogether with the instance when you create the volumes.

Exam Tips:

- Volumes exists on EBS. EBS is like a virtual HD.
- Snapshots exists on S3. Snapshot is like a photo of the disk.
- Snapshots are point in time copies of volumes.
- Snapshots are incremental, this means that only the blocks that have changed since your last snapshot are moved to S3.
- If this is your first snapshot, it may take some time to create.
- Creating a snapshot for an ebs that serves as root device you should stop the instance before.
- You can take a snap when the instance is running.
- You can create AMI's from snapshots.
- You can change EBS volume sizes on the fly, include changing the size and the storage types.
- Volumes will ALWAYS be in the same AZ than the instance.
- We can move an EC2 volume from one AZ to another, take a snap, create an AMI from the snap and use that AMI to launch the intance in another AZ.
- You can move EC2 volume from region by doing the same than above + copying the AMI to the other Region.

AMI Types (EBS vs Instance Store):

Exam Tips:

- Instance Store Volumes are sometimes called Ephemeral Storage.
- Instance store volumes cannot be stopped. If the underlying host fails, you will lose your data.
- EBS backed instances can be stopped. You will not lose the data on this instance if it is stopped.
- You can reboot both, you won't lose your data.
- By default, both ROOT volumes will be deleted on termination. However, with EBS volumes, you can tell AWS to keep the root device volume.

Why Instance Store is usefull:

It is faster than EBS. You can use it as temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content.

Instance Store is physically attached to the hardware the EC2 is running on, so it can better performance than EBS, which is network attached.

Where to find Instance Store:
When you are going to launch a new EC2 instance, go to Community AMI's -- Root device type, tick *Instance store*

You will have less Intances types to select, the lowest one is c1.medium.

When you are in the step 4 Add Storage you will see in Volume Type that is showing Instance Store 0, normally you have threre EBS.

ENI vs. ENA vs. EFA:

ENI - Elastic Network Interface: Essentially a virtual network card.

EN - Enhaced Networking: Uses single root I/O virtualization (SR-IOV) to provide high-performance networking capabilities on supported instance types.

EFA - Elastic Fabric Adapter: A network device that you can attach to yout EC2 instance to accelerate High Performance Computing (HPC) and machine learning applications.

ENI: 1 ENI card allows you to:

- A primary IPv4 from the ip addresses range from your VPC
- One or more secondary private IPv4 from the range of your VPC
- One Elastic IP (persistent public IP) per private IPv4 address.
- One public IP IPv4
- One or more IPv6
- One or more security groups
- A MAC address
- A source/destination check flag
- A description

When to use ENI:

- Create a management network, like to access via ssh to the server
- Use network and security appliances in your VPC
- Create dual-homed instances with workloads/roles on distinct subnets. You may have your production network and your database network and you may segregate them by using multiple ENI.
- Create a low budget high availability solution.

Enhanced Networking: On supported instances types only.

- Uses single root i/o virtualization (SR-IOV) to provide high performance networking capabilities on supported instances types. SR-IOV is a method of device virtualization that provides higher i/o performance and lower CPU utilization when compared to traditional virtualized network interfaces.

- Enhaced networking provides higher bandwidth, higher packet per second (PPS) performance, and consistently lower inter-instance latencies. There is no additional charge for using enhaced networking.

- Use where you want good network performance.

Can be enabled using:

- Elastic Network Adapter (ENA): The better one. Supports network speeds up to 100 Gbps for suppported instances types.

- Virtual Function (VF) Intel 82599 interface: The old one. Support network speeds up to 10 Gbps for supported instance types. Used on older instances.

In any scenarion question, you provably want to choose ENA over VF.

Elastic Fabric Adapter:

- You can attach it to the ec2 instance to accelerate High Performance Computing (HPC) and machine learning apps.

- EFA provides lower and more consistent latency and higher throughput than TCP transport traditionally used in cloud based HPC systems.

- EFA can use OS-bypass. OS-bypass enables HPC and machine learning apps to bypass the operating system kernel and to communicate directly with the EFA device. It makes it a lot faster with a lot lower latency. Not supported with Windows, only Linux.

Exam Tips: In the exam you will be given different scenarios and you will be asked to choose whether you should use:

ENI, Elastic Network Interface:
For basic networking. Perhaps you need a separate management network or a separate logging network at a low cost. In this scenario you will use multiple ENIs for each network.

EN, Enhaced Network:
For when you need speeds between 10 Gbps and 100 Gbps. Anywhere you need reliable, high throughput.

EFA, Elastic Fabric Adaptor:
To accelerate High Performance Computing (HPS) and Machine Learning applications or if you need to do an OS by-pass.

Encrypted Root Device Volumes & Snapshots:

AWS let's you to create encrypted root device volumes when you create the instance. This is a new feature, but before this was not possible.

Encrypt root volume in an already running instance:
The volume was not encrypted when the instance was created.

Go to Elastic Block Store -- Volumes -- Select the one you need -- Create a snapshot of the volume

Once created the snapshot you will see that is not encrypted either.
In Elastic Block Store, Snapshots go to -- Actions -- Copy -- change description -- tick Encryption this snapshot and hit copy.

Create an Image: EBS -- Snapshot -- select the just copied one -- Actions, Create Image -- Create

You will have now the AMI encrypted to create the instance with it.

Exam Tips:

- Snapshots of encrypted volumes are encrypted automatically.
- Volumes restored from encrypted snapshots are encrypted automatically.
- You can share snapshots but only if they are unencrypted.
- These snapshots can be shared with other AWS accounts or made public.
- You can encrypt root device volumes upon creation of the EC2 instance.

Process to encrypt a root device:

1. Create a Snapshot of the unencrypted root device volume.
2. Create a copy of the snapshot and select the encrypt option
3. Create an image/AMI from the encrypted snapshot
4. Launch the EC2 instance with the encrypted AMI.

Spot Instances and Spot Fleets:

Spot Instances allow you to take advantage of unused EC2 capacity in the AWS cloud.

Spot instances are available at 90% discount compared to On-Demand prices.

How to use it:

Decide your maximum Spot price. The instance will be provided so long as the Spot price is BELOW your maximum Spot price.

The hourly Spot price varies depending on capacity and region.

If the Spot price goes above your maximum, you have two minutes to choose whether to stop or terminate your instance.

Prevent termination for a while:
You can prevent to stop your Spot instance from being terminated even if the Spot price goes over your max Spot price. You can set **Spot blocks** for between one or six hours.

Pricing History:
You can see the price for different AZ which will help you to select the cheapest one.

Go to: Instances -- Spot Requests -- Pricing history

Use Cases:
You can use them for various stateless, fault tolerant, or flexible applications, such as big data, containerized workloads, CI/CD, web servers, high-performance computing (HPC), and other test and development workloads.

Applications that DON'T need to be online all the time.

Use Cases:
- Big Data and analytics
- Containerized workloads
- CI/CD and testing
- Web services
- Image and media rendering
- Hih-performance computing

Applications that DON'T need to be online all the time.

Not good for:
- Persistent workloads
- Critical jobs
- Databases

How to terminate Spot instances:

Spot Fleets:
A Spot Fleet is a collection of spot instances and, optionally, On-Demant Instances.

A Spot Fleet attempts to launch the number of Spot Instances and On-Demand Instances to meet the target capacity you specified in the Spot Fleet request.

The request for Spot Instances is fullfilled if there is available capacity and the maximum price you specified in the requests exceeds the current Spot price, as long as the Spot price is below your maximum Spot price then is going to launch Spot instances.

The Spot Fleet also attempts to maintain its target capacity fleet if your Spot Instances are interrupted, so will relaunch those instances.

Launch Pools:
Spot Fleets will try and match the target capacity with your price restraints.

1. Set up different launch pools. Define things like EC2 instances type, operating system, and AZ

2. You can have multiple pools, and the fleet will choose the best way to implement depending on the strategy you define.

3. Spot fleets will stop launching instances once you reach your price threshold or capacity desire.

Spot Fleets Strategies:

- capacityOptimized: The Spot Instances come from the pool with optimal capacity for the number of instances launching.

- lowestPrice: The Spot Instances come from the pool with the lowest price. This is the default strategy.

- diversified: The Spot Instances are distributed across all pools.

- InstancePoolsToUseCount: The spot instances are distributed across the number of spot instances pools you specify. This parameter is valid only when used in combination with lowestPrice.

Exam Tips:

- Spot Instances can save up to 90% of the cost of On-Demant instances.
- Useful for any type of computing where you don't need persistent storage.
- You can block Spot Instances from terminating by using Spot block.
- A Spot Fleet is a collection of Spot Instances and, optionally, On-Demant instances.

EC2 Hibernate:

When we start our EC2 instance, the following happens:

- Operating system boots up

- User data script is run (bootstrap scripts). Script that runs when the instance is first booted. It may be a script that installs nginx or mysql or to change the hostname or doing other configuration tasks.

- Applications start, like nginx, mysql, etc (can take some time.)

EC2 Hibernate:

When you hibernate an instance, the OS is told to perform hibernation (suspend-to-disk).

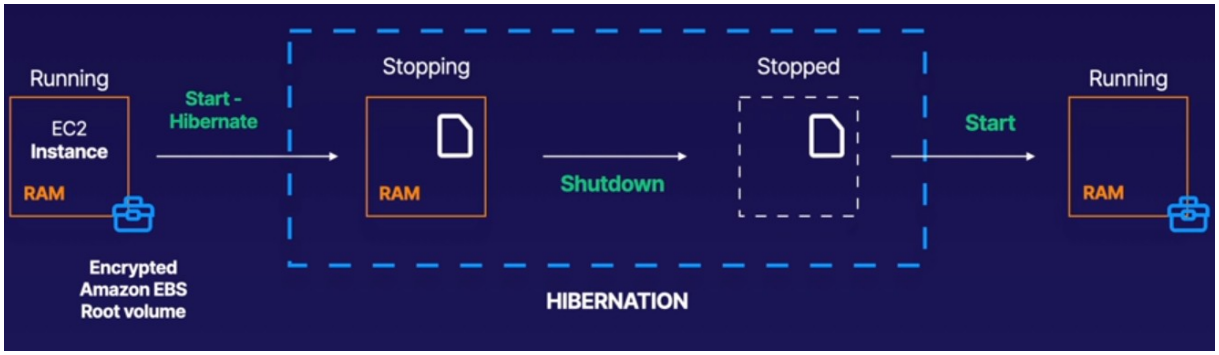
Hibernation saves the content from the instance memory (RAM) to your EBS root volume.

We persist the intance's EBS root vollume and any attached EBS data volumes.

This makes the booting process a lot faster to load up.

When you start your instance from hibernation:

- The EBS root volume is restored to its previous state.
- The RAM contents are reloaded.
- The process that were previously running on the instance are resumed.
- Previously attached data volumes are reattached and the instance retains its instance ID.



We have an instance with its EBS and RAM.

We start the hivernation process, which stops out the instance, saves the ram down into the EBS volume.

Then shutdown and stop.

When we start it is gonna take the RAM from the EBS and load it back into the RAM. So we don't need to do things like restart the OS or start out applications, etc.

Useful for: With hibernate the instance boots much faster. The OS is not need to reboot because the in-memory state (RAM) is preserved.

1. Long-running processes
2. Services that take time to initialize.

What do you need to enable it:
When creating the instance,in the: Step 3: Configure Instance Details, you have to tick the enable hibernation box.

Have in mind that the EBS root volume will need enough space to store the RAM contents and my applications. Increase the size to 25 GB.

To use hibernation the root volume must be encrypted.

How to use it:
First go to your running instace and run the *uptime* command to see the last time the instance was booted, for example: up 3 min

Then go to the aws console -- Instance State -- Hibernate instance.

To start it: Instance State -- Start

Now go to the instance CLI and run *uptime*, you will see that it has not been rebooted, you will see: up 3 min ore more.

Hibernation is not restarting the instance.

Exam Tips:

- Hibernate preserves the in-memory RAM on persistent storage (EBS)
- Mush faster to boot up because you do not need to reload the operating system.
- Instance RAM must be less than 150 GB
- Instance families include: C3, C4, C5, M3, M4, M5, R3, R4, and R5, T2, etc.
- Available for Windows, Amazon Linux2 AMI and Ubuntu.
- Can NOT be hibernated for more than 60 days.
- Available On-Demand instances and Reserved Instances.

CloudWatch 101:

What is it:
CloudWatch is a monitoring service to monitor your AWS resources, as well as the applicatons that you run on AWS. Monitores performance.

Can monitore things like:

- Compute: EC2 Instances, Autoscaling Groups, ELB, Route53 Health Checks.

- Storage & Content Delivery: EBS Volumes, Storage Gateways, CloudFront

CloudWatch on EC2: Host level metrics consists of:

- CPU
- Network
- Disk
- Status Check

Don’t confuse it with CloudTrail:

CloudTrail is a kind of CCTV for the AWS environment. It records AWS management console actions and API calls.

Everytime you create a bucket, an instance, etc, you are making an API call to AWS and this is all recorded using CloudTrail.

You can go in and identify which users and accounts called AWS, the source IP address of the calls and when where made.

CloudWatch monitores performance
CloudTrail monitores API calls in the AWS platform.

Exam Tips:

- CloudWatch used to monitor performance.
- Can monitor most of AWS as well as your applications that runs on AWS.
- By default will monitor events every 5 minutes.
- You can have 1 minute intervals by turning on detailed monitoring. (have extra cost)
- You can create CloudWatch alarms which trigger notifications.
- CloudWatch is all about performance. CloudTrail is all about auditing.

Configure CloudWatch:

Enable it:
CloudWatch comes by default for the 5 minutes monitoring, but we can change it to 1 minute by doing this.

Launch an instance and in the Step 3: Configure Intance Details tick enable cloudwatch detailed monitoring to monitor every 1 minute instead of 5. **This have additional charges** you can leave it unticket to monitor every 5 min.

Once the instance is running go to Status checks you have to see it running.

In Monitoring you will see the host level metrics.

You can NOT see there things like RAM utilization or how muc disk storage left. To enable them you have to do a custom metric not covered in the SAA certification, this is on the sysops admin cert.

Set up an alert: To get an email when something hits the limit we stablished.

Go to Management & Governance -- CloudWatch -- Alarms, In alarm -- Create alarm -- Select metric -- EC2 -- Per-Instance Metrics -- Select your instance ID the metric name that you are looking for, CPUUtilization in this example -- Select metric:

Name it: CPUUtilization-90%

Conditions: Greater/Equal 90

Additional Configuration -- Datapoints to alarm: 1 out of 1 → The first 1 is one minute, because is what we have configured. In the second 1 we are saying that if the condition happens one time you must to trigger the alarm.

If the number were 2, the alarm would be triggered at the second time, so the instance reaches the 90% of cpu utilization for the first time but doesn't triggers any alarm.
If it reaches the limit a second time then the alarm will be triggered.

Next -- Create new topic -- leave the name by default -- Add your email -- Create topic -- Next -- Give it a name -- Next -- Create alarm

Could ask you to confirm your email addres, do it.

If you are not using the instance you could have Insufficient data, just turn on the instance and wait for a while.

Force the instance to trigger the alarm:
Get into the instance and create a loop to increase the cpu utilization: while true; do echo; done

This is gonna max out the cpu and will trigger the alarm sending us the email.

Management & Governance - CloudWatch:
Create Dashboards:
Go to Management & Governance -- CloudWatch -- Create dashboard, name it and select what you want, this is for the sysops cert.
You only have to remember that you can create dashboards Globaly or by region.

Logs:
Allows us to do performance logging.

Events:
You can set up events.

Exam Tips:

- Standar Monitored = 5 min.
  Detailed monitoring = 1 min.

- With cloud watch you can:
  - Create Dashboards, local or global, to see what is happening with your AWS env.
  - Alarms: Allows you to set alarms that notify you when particular thresholds are hit.
  - Events: CloudWatch Events help you to respond to state changes in your aws resources.
  - Logs: Helps you to aggregate, monitor and store logs

- CloudWatch monitores performance. CloudTrail monitors API calls in the aws platform.

AWS CLI:

Create AIM user and group:
Go to AIM -- Users -- Add user, user name: MyEC2User -- tick programatic access -- Next
Create group, name it as AdminGroup -- give it AdministratorAccess -- Create group, next, next, create user.

Remember: Download or save the secret access key and acess key id when your user has been created. If you close that window you won't be able to recover the credentials again.

If you lose your secret access key you can generate a new one from Users -- Security credentials

Create instance:
Take amazon linux -- t2.micro -- leave all by default -- create a new keypair if necessary -- launch
Change permissions to 400 for the new key and get inside the server.

Configure AWS CLI:
In amazon linux aws cli is already installed, but if you run it will tell you that you need to configure the credentials.

- Type: aws configure
- Add the access key id and the secret key from the previous steep.
- Region: the one where the instace is: us-east-1
- Default output: press enter

Now you are able to access aws from the instance via cli, to test it type: aws s3 ls
or create a bucket: aws s3 mb s3://test23234234234 → the name don't have to exist previously.

[root@cloudwatch ~]$ **aws s3 ls**
2021-03-22 10:50:53 test23234234234

Edit/Modify your credentials: vi ~/.aws/credentials
You will see your aws access keys there.
You have ~/.aws/config to. There you have the region.

Using access keys is not secure. Is much better to use AWS Roles.

Exam Tips:

- You can interact with AWS from anywhere in the world just by using the CLI.
- You will need to set up access in AIM
- Command themselves are not in the exam, but some basic commands will be useful to know for real life.

AWS Roles:

Create a new role:
IAM -- Roles -- Create Roles -- Choose use cases: EC2 -- Next Permissions -- Attach a policie: AministratorAccess -- Next, Next -- Role name: AdminAccess -- Create Role

Delete old credentials:
Go via cli to the instance created in the previous exercise -- Remove the credentials there:
rm -rf ~/.aws

Attach the role to the instance:
Select your instance -- Actions -- Security -- Modify IAM role -- select the role created previously: AdminAccess -- Save

Refresh the page. You should be able to see in Security the IAM Role there.
If you click on the name of the role you will be able to see it and modify it if necessary.

Back on the server:
Via cli, your command will work, like aws s3 ls.

Now we have admin access through this server, and without aws credentials, ~/.aws is not there so no one can take these credentials and do evil things.

Exam Tips:

- Roles are more secure than storing your access key and secret access key on individual EC2 instances.
- Roles are easier to manage. If you have a fleet with 1000 instances and you want to modify the credentials you have to do it one by one. Roles instead can be changed easier and faster.
- Roles can be assigned to an EC2 instance after it is created using both the console & command line.
- Roles are universal, you can use them in any region.

Using Bootstrap Scripts:

Are a way to automate the AWS deployment. Is a way to add command lines when your EC2 instance boots.

You can do updates, you can install software, basically anything that you can do in an CLI.

Create an instance:
amazon linux, t2.micro -- Give it the IAM role created previously -- in User data you paste or type the bootstap script, which is a bash script:

#!/bin/bash → allways the shebang
yum update -y
yum install httpd -y
service httpd start
chkconfig httpd on → to start httpd after a server reboot.
cd /var/www/html
echo “<html><h1>Hello to my webpage</h1></html>” > index.html
echo "<html><h1>Hello Cloud Gurus </h1></html>" > index.html
aws s3 mb s3://sfgsdfgsdfgs3453453543 → create a bucket
aws s3 cp index.html s3://sfgsdfgsdfgs3453453543 → copy index.html into the new bucket

From the teacher:
#!/bin/bash
yum update -y
yum install httpd -y
service httpd start
chkconfig httpd on
cd /var/www/html
echo "<html><h1>Hello Cloud Gurus Welcome To My Webpage</h1></html>" > index.html
aws s3 mb s3://YOURBUCKETNAMEHERE
aws s3 cp index.html s3://YOURBUCKETNAMEHERE

Instance Metadata

Get the bootstrap script of an instance:
Get into the instance and type: curl http://169.254.169.254/latest/**user-data** > bootstrap.txt

It will paste on bootstarp.txt the script created in the previous lesson.

Get the metadata of the instance:
You can get a lot of information of the instance, like the IPs, security group, network, etc.

Just type: curl http://169.254.169.254/latest/**meta-data**/ and hit enter

```
[root@server ~]# curl http://169.254.169.254/latest/meta-data/
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
events/
hibernation/
hostname
iam/
identity-credentials/
instance-action
instance-id
instance-life-cycle
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/
```

Cool things to do:

You can use the bootstap script to, for example get the public IP, sent it to a file, copy it to s3, then trigger a lambda function in order to store the IP in a db.

Exam Tips:

- Metadata is used to get information about an instance (such as public IP)
- This is the command to get it: curl http://169.254.169.254/latest/**meta-data**/
- To get the bootscript ran when creating the instance do:
  curl http://169.254.169.254/latest/**user-data**/

EFS:

Elastic File System: File storage service for AWS EC2 instances.

EFS is easy to use and provides a simple interface that allows you to create and configure file systems quickly and easily.

With EFS storage capacity is elastic, growing and shrinking automatically as you add and remove files, so your apps have the storage that they need when they need.

Is similar to EBS but with EBS you can not share the same volume to 2 different instances at the same time, when with EFS you can do it.

LAB: Create 2 instances with apache and use EFS to store the web.

The two instances will go to EFS to take the html code. This centralizes the code in EFS instead of having to deploy the html to the 2 servers.

Create the EFS:
Storage -- EFS -- Create file system -- leave default (vpc and regional), Create or Customize:

Create → you create the EFS straigh away.

Customize → you can specify a lot of things:
- Encryption
- Enable backups
- Lifecycle
- Performance mode
- Throughput mode
- Tags

Customize, Next -- Network: You can choose the VPC and the AZ in which you want to replicate the EFS, by default will be replicated in all the AZ of the region.

Next, Policy option: You can create a custom security policy or choose the common policies, like prevent root access and so.

Review and Create → will take 5 minutes to create the EFS

Launch the instances:
2 amazon linux, t2.micro, change the number of instances to 2, add the following to the bootstrap script:

```
#!/bin/bash
yum update -y
yum install httpd -y
service httpd start
chkconfig httpd on
yum install -y amazon-efs-utils
```

Add the instances to your custom web-DMZ security group and leave the rest by default and launch the instances.

Security Group:
We have the EFS in the default security group and the instances in the web-DMZ security group.

We have to communicate both by opening ports like this:

Allow NFS protocol into default sg:
Default security group: Inbound rules -- Edit inbount rules -- Add rule, NFS (port 2049), select the web-DMZ security group by tying sg -- Save

Get into the ec2 instances:
You should see that apache has been installed correctly by looking at /var/www/html
If the folder /www/html is there means that httpd has been installed properly.

Mount EFS into ec2 instances:
Get into the mount helper:
Storage -- EFS -- select your EFS -- hit Attach (upper-right) -- copy the command of the mount helper:

sudo mount -t efs -o tls fs-b3300306:/ efs

-o tls → the files will be encryped in transit

Go to the servers and modify the folder to

sudo mount -t efs -o tls fs-b3300306:/ /var/www/html

We've changed the default efs directory for /var/www/html

It will take a few seconds, won't show any success message.

Check if works:
Go to one of the servers and create an index.html basic file:
<html><body><h1>Hi, first website using EFS.</html></body></h1>

Now you will see the same index.html file in both servers, EFS is working properly

Exam Tips:

- Supports the NFSv4 protocol
- You only pay for the storage you use (no pre-provisioning required)
- Can scale up to the petabytes
- Can support thousands of concurrent NFS connections
- Data is stored across multiple AZ's whithin a region.
- Read After Write Consistency.

Amazon FSx for Windows and Amazon FSx for Lustre:

Amazon Fsx for Windows File Server:
It's a Windows file server. Designed to use Windows applications like SQL server, Active Directory, IIS, Sharepoint, etc.

Provides a fully managed native Microsoft Windows file system so you can easily move your Win-based applications that require file storage to AWS.

Amazon Fsx is built on Windows Server.

Runs Windows Server Message Block (SMB)-based file services.

Supports AD users, access control lists, groups and security policies, along with Distributed File System (DFS) namespaces and replications.

Differences with EFS:

- FSx works with SMB, EFS with NFS.
- Amazon DO NOT support EC2 instances running Windows to connect to EFS. EFS it's linux only.

Amazon FSx for Lustre: (pronunced: laster)
Fully managed fs that is optimized for compute-intensive workloads, such as:

- High-performance computing
- Machine learning
- Media data processing workflows
- Electronic design automation (EDA)
- Big Data

You can launch and run a Lustre file system that can process massive data sets at up to hundreds of gigabytes per second fo throughput, millions of IOPS, and sub-millisecond latencies.

Differences with EFS:

- Designed specifically for fast processing workloads like the ones specified above.
- Launches a file system that provides sub-millisecond access to your data like specified above.

Exam Tips: In the exam youo'll be given different scenarios and asked to choose whether you should use an EFS, FSx for windows or FSx for Lustre:

- **EFS:** For linux instances and linux-based applications. When you need distributed, highly resilient storage.

- **Amazon FSx for Windows:** When you need centralized storage for Windows-based applications such SQL server, IIS, and other Microsoft applications

- **Amazon for Lustre:** When you need high-speed, high-capacity distributed storage. This will be for apps that do Hight Performance Compute (HPC), financial modeling, etc. FSx for lustre can store data directly on S3

EC2 Placement Groups:

Is a way of placing the EC2 instances.

There are 3 different types of placement groups:

1. Cluster Placement Group
2. Spread Placement Group
3. Partitioned

Clustered Placement Group: Is a grouping of instances instances within a single AZ.

Are recommended for applications that need low network latency, high network throughput, or both.

Only certain instances can be lauched in to a cluster placement group.

Basically you are putting very close your instances to have better network performance.

Spread Placement Group: Is the opposite, you don't want to place your instances in the same hardware (rack, network, power supply, etc).

The instances can be in the same AZ but they are not sharing resources in order to avoid a cascade fail.

This is recommended for applications that have a small number of critical instances that should be kepts separate from each other.

We are talking about single instances each, so each single ec2 instance is separated with each other in order to avoid to have 2 instances sharing resources.

Partition Placement Group: Similar to Spread placement group, except that you can have multiple EC2 instances within a partition.

When you are using partition placement group amazon divides each group into logical segments called Partitions, and Amazon EC2 ensures that each partition within a placement group is on its own set of racks without sharing hardware resources.

This allows to isolate the impact of hardware failure whithin your application.

Exam Tips:

- **Clustered Placement Group:** Low Network Latency / High Network Throughput. EC2 instances as close as possble. Always in the same AZ in the same Region
- **Spread Placement Group:** Individual Critical EC2 instances. Must be on separate pieces of hardware. Can be span in multiple AZ.
- **Partitioned Placement Group:** Multiple EC2 instances. Can be used in clusters like: Hadoop,Kafka, and Casandra. Can be span in multiple AZ.
- The name you specify for a placement group must be unique within your AWS account.
- Only certain types of instances can be lauched in a placement group: Compute Optimized, GPU, Memory Optimized, Storage Optimized.
- AWS recommends homogenous instances within clustered placement groups. All the instances should be the same kind.
- You can't merge placement groups.
- You can move an existing instance into a placement group. Before you move the instance, the instance must be in stopped state. You can move or remove an instance using CLI or SDK, NOT the console yet.
- There is no charge for creating a placement group.

High Performance Compute / HPC:

With cloud computing it's easier to get started with HPC.

You can create large number of resources in almost no time. You only pay for the resources you use, and once finished, you can destroy the resources.

HPC use cases:

- Genomics
- Finance and financial risk modeling
- Machine learning
- Weather prediction
- Autonomous driving
- etc

Services we can use to achieve HPC:

- **Data Transfer:** How to move our data into AWS
    - Snowball, Snowmobile: When you load all your data there and you send it to AWS. You don’t have to do it via Internet.
    - AWS DataSync: Install a vm in your DC and store the data in our dc or push it to AWS (S3, EFS, FSx, etc)
    - Direct Connect: Dedicated line from our DC to AWS which can reduce network cost, increases bandwidth throughput, and provide a more consistent network experience than internet based connections.

- **Compute and Networking:**
    - EC2 instances that are GPU or CPU optimized.
    - EC2 fleets (spot instances or spot fleets)
    - Placement groups (cluster placement groups)
    - Enhanced networking: Uses SR-IOV to provide high I/O and lower CPU utilization, higher PPS, low latencies, etc. Use when you want good performance. Have no extra cost.
    - Elastic Network Adapters: Choose Elastic Network Adapter (ENA)
    - Elastic Fabric Adapters: This is explained in the ENI vs. ENA vs. EFA section

- **Storage:**
    - Instance-attached storage:
        - EBS: Scale up to 64,000 IOPS with Provisioned IOPS (PIOPS)
        - Instance Store: Scaling to millions of IOPS, low latency
    - Network storage:
        - S3
        - EFS
        - FSx for Luster: HPC optimized distributed file system. Millions of IOPS, also backed by S3

- **Orchestration & Automation:**

    - AWS Batch: Run easily and efficiently hundreds of thousands of batch computing jobs. Supports multiple node parallel jobs allows you to run a single job that spans multiple instances. You can easily schedule jobs.

    - AWS ParallelCluster: Open soure cluster management tool that makes easy to deploy and manage HPC clusters on AWS. Uses IFS as a code. Automate creation of VPC, subnet, cluster type, and instance types.

Exam Tips:

We can achieve HPC on AWS through:

- Data transfer:
    - Snowball, Snowmobile
    - AWS DataSync
    - Direct Connect

- Compute & Networking:
    - GPU or CPU optimized instances
    - EC2 Fleets (spot instances or spot fleets)
    - Placement groups (cluster pg)
    - Enhaced Networking SR-IOV
    - Elastic Network Adapters (ENA)
    - Elastic Fabric Adapters (OS-bypass)

- Storage:
    - EBS
    - Instance Store
    - S3
    - EFS
    - FSx for lustre

- Orchestration & Automation:
    - AWS Batch
    - AWS ParallelCluster

AWS WAF / Web Application Firewall:

OSI layer 7 aware firewall. Allows you to control access to your content.

Monitores http and https (layer 7) requests that are forwarded to Amazon CloudFront, an Aplication Load Balancer or API Gateway.

Also lets you control access to your content.

You can see things like query string parameters. It can see the actual information that your are sending to your webservers.

Query String Parameters:
http://acloud.guru**?id=1001&name=ryan**

The web address is http://acloud.guru which ends with the top level domain name.

?id=1001&name=ryan are the query string parameters which it's passing variables back to your webserver.

So this variable is called id and is equal to 1001 → id=1001

This variable is called name and is equal to ryan → name=ryan

WAF can see this query string parameters because is a layer 7 fw. Wich is better than the hardware firewall that can only see until the layer 4.

What can you do:
You can configure conditions such as what IP addresses are allowed to make this request or what query string parameters need to be passed for the request to be allowed.

The application load balancer or CloudFront or API Gateway will either allow this content to be received or to give a HTTP 403 Status Code.

Basic level behaviours:

1. Allows all requests except the ones you specify.

2. Block all requests except the ones you specify.

3. Count the requests that match the properties you specify. (Pasive mode)

Extra protection WAF provides:

- IP addresses that request originate from.
- Contry that request originate from.
- Values in request headers.
- Strings that appear in requests, either specific strings or strings that match regex patterns.
- Length of requests.
- Presence of SQL code that is likely to be malicious (SQL injection)
- Presence of a script that is likely to be malicious (XSS)

Exam Tips:
In the exam you will be given different scenarios and you will be asked how to block malicious IP addresses:

- Use WAF
- Use Network ACLs

Lab: Using AWS Tags and Resource Groups and AWS Config

Lab

What are we going to do:

- We will search for resources, Resource Groups & Tag Editor, with the tags Module: Starship Monitor and Module: Warp Drive.
- We are going to create two resource groups, one for each Module and we will add the resources found before.
- We will take an AMI from one of the 4 instances that we have, this will create a different AMI ID.
- We will use AWS Config to see if the resources inside the groups created before have the new created AMI.
- This will show that some of the resources have the wrong AMI, helping us to locate them. This is good for auditing resources and compliance.

Set up AWS Config: Provides a detailed view of the resources of your aws account and how are they configured, etc. Good for audits.

Management & Governance -- Config -- 1-click setup -- Confirm -- View dashboard

EC2: The 4 servers are already created by the lab.

Create an AMI of one of the instances, doesn't matter which one.

EC2 -- Instances (running) -- right click in one of them -- Image and templates -- Create image -- give it a name: Base -- Create image (will take 5 minutes)

AMIs:
Images -- AMIs -- select the one you did (Base) -- Launch -- leave all as default -- Add Tags: Name Test Web Server -- Configure Secruty Groups: select cfst security group (created by the lab) -- Launch -- Proceed without key pair.

Tag Editor:
Management & Governance -- Resource Groups & Tag Editor -- Tagging -- Tag Editor -- Resources types: find the resources you've created (ec2 and s3) AWS::EC2:Instance , AWS::S3::Bucket -- Search resources -- you will see 5 instances and 4 buckets

Do a search:

Filter resources: type the instance name, like: mod. 1 → which stands for: Mod. 1 Web Server A

You will see two instances appearing:
Mod. 1 Web Server A

Mod. 1 Web Server B

Tick the box to select them all -- Clear the filters (the two instances will be still selected)

Search for the s3 buckets by the name of one of them: moduleone -- tick it and clear filters again.

Now you have two instances and one s3 bucket selected between all the resources.

Manage tags of selected resources -- Add tag, key: Module , tag value: Starship Monitor -- Review and apply tag changes -- Apply changes to all selected

Do another search: This time for module 2

Search resources (ec3 and s3 Resource types are already there form the last search) -- Filter resources: mod. 2 -- select both ec2 instances and clear filters -- search for: moduletwo and select it, clear the filters.

Manage tags of selected resources -- Add tag, key: Module , tag value: Warp Drive -- Review and apply tag changes -- Apply changes to all selected

Create resource group: By using the recently created tags
Resources -- Create Resource Group -- Grouping criteria, Tags: select Module and Starship Monitor -- Preview group resources -- all the resouces with tag Module: Starship Monitor will appear, 2 ec2 and 1 s3 in this example

Create a group name: For the recently searched resources
Group details -- Group name: Starship-Monitor -- Create group.

Now we have a tag with Module: Starship Monitor.

Lets to it again for Warp Drive: Create Resouce Group -- Tags: Module , Warp Drive -- Preview group resources -- Group name: Warp-Drive -- Create group

Now we have two resouces groups, go to Saved Resouces Groups to see them all.

Any new resources created with the tags Module Starship Monitor or Warp Drive will be added automatically to these resouces groups.

AWS Config: This will search into the tags created previously for the ones that have the wrong AMI.

Get into ec2 to pick an AMI up. You can go to AMIs or to the server you've created before, Test Web Server, and pick the AMI ID, ami-0sd02340ed → for example, won't be this id.

Into AWS Config -- Rules -- Add Rule -- Add AWS managed rule -- search for: ami, select the first one: approved-amis-by-id, Next -- Trigger, Tags: Module Starship Monitor -- Parameters: amiids, paste the AMI ID copied on the previous steep, Next -- Add rule. Is gonna take a few minutes to the rule to be evaluated.

This is gonna create a rule that will search resources with the module tag set to: Starship Monitor and it’s gonna try to make sure that the AMI ID of that resource is set to the value, and if not is gonna tell us about it.
IMPORTANT: The rules will only be evaluated when the rules changes, so try that reboot all the 5 instances of this exercise.

Once the servers are back, refresh the AWS Config console, in Rules you will see 2 Noncompliant resource(s)

Mod. 1 Webserver A
Mod. 1 Webserver B

They were created before creating the AMI we are looking for.
The Tag will match: Module: Starship Monitor, but the AMI won’t match the AMI, therefore will show us that discrepance.

Lab: Using EC2 Roles and Instance Profiles in AWS

Databases on AWS:

Databases 101:

Relational Databases: Think of a traditional spreadsheet, like microsoft excel or libreoffice calc.

Using the excel example, you have file.xls.

file.xls is the database and inside we have our data, spread in Tables, Rows and Fields (Columns).

ID	First Name	Surname	Gender
1	Ryan	Kroonenburg	M
2	John	Adams	M
3	Julia	Clark	F
4	Danielle	Dustagheer	F

Relational Databases on AWS:

- SQL Server
- Oracle
- MySQL Server
- PostgreSQL
- Aurora
- MariaDB

RDS has two key features:

- Multi-AZ for Dissaster Recovery
- Read Replicas for performance

Non Relational Databases: DynamoDB. Let's you to add new field for particular Document. If you do the same in a RDS you will affect the whole table.

- Collection = Table
- Document = Row
- Key Value Pairs = Fields

Data Warehousing: Used for business intelligence. Tools like Jaspersoft, Cognos, SQL Server Reporting Services, Oracle Hyperion, Sap Business Warehouse.

Used to pull in very large and complex data sets. Usually used by management to do queries on data (such as current performance vs targets etc)

OLTP vs OLAP: Both differs in terms of type of queries you will run:

- OLTP: Online Transaction Processing: Much more concrete. Example: You want to know all the information of the order 2120230. So the query pulls up a row of data such as Name, Date, Address to deliver to, Delivery Status, etc.

- OLAP: Online Analytics Processing: Much more complicated. Example: The COO wants to know the Net Profit for EMEA and Pacific for the Digital Radio Product. The queries will pull in large numbers of records, like Sum of radios Sold in EMEA, sum of Radios sold in Pacific, unit cost of radio in each region, sales price of each radio, sales price - unit cost.

Redshift: Data Warehousing databases use different type of architecture both from a database perspective and infrastructure layer. Redshift is AWS Data Warehousing solution.

ElasticCache: Web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud.

The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.

ElastiCache supports two open-source in-memory caching engines:
- Memcached
- Redis

Exam Tips:

- RDS (OLTP)
  - SQL
  - MySQL
  - PostgreSQL
  - Oracle
  - Aurora
  - MariaDB

- DinamoDB (NoSQL)

- Red Shift (OLAP): Bussines Intelligence or Data Warehousing

- ElastiCache: To speed up performance of existing databases (frequent identical queries)
  - Memcached
  - Redis

Exam Tips: Taken from the Lab

- RDS runs on virtual machines. You DON'T have any access into those instances.
- You can NOT log in to these operating systems.
- Patching of the RDS OS and DB is Amazon's responsibility.
- RDS is NOT Serverless. Except Aurora, the rest is running on instances managed by Amazon.
- Aurora Serverless IS serverless.

RDS: Backups, Multi-AZ, and Read Replicas:

RDS Backups:
- Automated backups.
- Database Snapshots.

Automated Backups:
- Allow you to recover your database to any point in time within a "retention period".
- The retention period can be between one and 35 days.
- Automated backups will take a full daily snapshot and will also store transaction logs throughout the day.
- When you do a recovery, AWS will first choose the most recent daily back up, and then apply transaction logs relevant to that day.
- This allows you to do a point in time recovery down to a second, within the retention period.
- Automated backups are enabled by default.
- The backup data is stored in S3 and you get free storage space equal to the size of your db. So if you have a RDS instance of 10GB, you will get 10GB worth of storage.
- Backups are taken within a defined window.
- During backup window, storage I/O may be suspended while your data is being backed up and you may experience elevated latency.

Database Snapshots:
- Are done manually (they are user initiated).
- They are stored even after you delete the original RDS instance, unlike automated backups.

Restoring Backups:
- Whenever you restore either an Automatic Backup or a manual Snapshot, the restored version of the database will be a new RDS instance with a new DNS endpoint.

Encryption at Rest:
- Encryption at rest is supported for Mysql, Oracle, SQL Server, Postgres, MariaDB & Aurora.
- Is done by using KMS
- Once your RDS instance is encrypted, the data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots.

Multi- AZ: Multi-AZ is for Disaster Recovery only. Is not primarily used for improving performance. For performance improvement, you need Read Replicas.

- Multi-AZ allows you to have an exact copy of your production database in another AZ.
- AWS handles the replication for you, so when your production database is written to, this write will automatically be synchronized to the stand by database.
- In the event of planned database maintenance, DB Instance failure, or an AZ failure, Amazon RDS will automatically failover to the standby so that database operations can resume quickly without administrative interaction.
- Available for all the RDS databases except for Aurora because is already fault tolerance by it's own architecture.

Read Replica: Used for scaling, not for Disaster Recovery.

- You can have an instance having all the writes and the other replicas doing the readings from the ec2 instances. You can also have a Read Replica from a Read Replica to.
- Read replicas allow you to have a read-only copy of your production database.
- This is achieved by using Asynchronous replication from the primary RDS instance to the read replica.
- You use read replicas primarily for very read-heavy database workloads.
- Must have automatic backups turned on in order to deploy a read replica.
- You can have up to 5 read replica copies of any database.
- You can have read replicas of read replicas, but watch our for latency.
- Each read replica will have its own DNS end point.
- You can have read replicas that have Multi-AZ.
- You can create read replicas of Multi-AZ source databases.
- Read replicas can be promoted to be their own databases. This breaks the replication.
- You can have a read replica in a second region.

How to create a read replica: This will have impact in the production DB, so do it in a maintenance window.

Go to Services - Database - RDS - Modify - Availability & durability: tick create a standby instance - Continue - Will warn you that this change will have performance impact - tick apply inmediately.

Hit the refresh button to see that the status is in "Modifying", give it a few minutes to finish. At the end you can go to the tab Configuration and see that Multi-AZ has been turned to "Yes"

Fail over to the replica:
Select the db that have a read replica - hit Actions, Reboot - tick Reboot with failover, this will reboot the production db and will fail over it to another in another AZ.

Turn on Read Replicas: The backups must be turned on, if not do:
Select the db and hit Modify - change the Backup retention period to 35 days - Continue - Apply inmediately, will take a while to apply the changes.

Now you can hit Actions - Create read replica - give it a name: acloudguru-readreplica - leave the rest by default - Create read replica. Will be modifying the primary and creating the replica, will take a while.

Promote a read replica to primary:
Select the read replica - Actions - Promote

Exam Tips:

RDS Backups:
- Automated backups.
- Database Snapshots. (manualy)

Read Replicas:
- Can be Multi-AZ
- Used to increase performance
- Must have a backup turned on
- Can be in different regions
- Creating a read replica does not cause any downtime for the app. The replication is asynchronous and will increase a bit the cpu of the primary while creating it.
- Can be MySQL, postgres, mariadb, oracle, aurora
- Can be promoted to primary, this will break the Read Replica. This have to be done manually and will take a big amount of time.

Multi-AZ:
- Used for Disaster Recovery
- You can force a failover from one AZ to another by rebooting the RDS instance.

Encryption at Rest:
- Encryption at rest is supported for Mysql, Oracle, SQL Server, Postgres, MariaDB & Aurora.
- Is done by using KMS
- Once your RDS instance is encrypted, the data stored at rest in the underlying storage is encrypted, as are its automated backups, read replicas, and snapshots.

DynamoDB:

AWS no SQL solution.

- Fast and flexible NoSQL database.
- For all applications that need consistent, single-digit millisecond latency at any scale.
- Fully managed db.
- Supports document and key-value data models.
- Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Basics of DynamoDB:
- Stored on SSD
- Spread across 3 geographically distinct data centers.
- Eventual Consistent Reads (Default)
- Strongly Consistent Reads

Eventual Consistent Reads:
Consistency across all copies of data is usually reached within a second. Repeating a read after a short time should return the updated data. (Best Read Performance)

Example, if you have an app and it's writing to a dynamodb table and then you want to read that data that's been written to that table a couple of seconds later you will do it to able.

Strongly Consistent Reads:
A strongly consistent read returns a result that reflects all writes that received a successful response prior to the read.

If your app needs to access the data written one second or less.

Exam Tips:

- Stored on SSD
- Spread across 3 geographically distinct data centers
- Eventual Consistent reads (default) one second or more
- Strongly consistent reads. One second or less.

Advanced DynamoDB:

DynamoDB Accelerator (DAX):

- Fully managed, highly available, in-memory cache.
- 10x performance improvement.
- Reduces request time from milliseconds to microseconds -even under load.
- No need for developers to manage caching logic
- Compatible with DynamoDB API calls.

Why do you need cache data:
With the traditional application and cache you'd have a cache sitting off to the side from the application.

This can be a traditional cache like Redis or Memcached.

As developer you have to code your app to take advantage of that cache, and this can be expensive and it's yet another thing to manage.

And these caches don't typically support writes just reads.

With dynamodb you can get into a situation where you have too many reads and your requests get throtled.

Meaning you'll get some errors when you try to read from dynamodb when it's under load.

What we can do is to introduce DAX into your app.

You make NO code changes into your app and DAX sits between your app and dynamodb.

Unlike the traditional caches, dax has what's called a write through cache, so you'll have read and write performance improvement.

If you send your writes to dax they flow directly into dynamodb, responding in microseconds.

Also designed for HA failing over to another AZ in case of fail in the original one.

Takes care of all this automatically.

Transactions:

- Multiple all-or-nothing operations
- Financial transactions
- Fulfilling orders
- Two underlying reads or writes - prepare/commit
- Up to 25 items or 4 MB of data.

Transaction: In the case of financial transactions when you want to transfer money from one account to another.
You can NOT debit one account without crediting another account, both have to happen simultaneously or nothing can happen.

DynamoDB performs to underlying reads or writes every item in the transaction.
One to prepare the transaction and one to commit the transaction.

On-Demand Capacity:

- Pay per request pricing
- Balance cost and performance
- No minimum capacity
- No charge for read/write (unlike provisioned capacity)- only charges storage and backups
- Pay more per request than with provisioned capacity
- Use for new product launches

In contrast to the default provisioned capacity on-demand offers, simple pay per request pricing for read and write request, so you only pay for what you use.

Balance cost and performance: If the traffic hits a new peak dynamodb adapts rapidly to accommodate the workload.

You pay more per request than provisioned capacity, so on demand is good for new product launches where you are not exactly sure what your consumptions is going to look like.

Once your app is in steady state and you can predict the traffic and consumption you will want to switch from on-demand to provisioned capacity.

On-Demand Backup and Restore:

- Full backups at any time, via API call or via Console
- Zero impact on table performance or availability
- Consistent within seconds and retained until deleted
- Operates within same region as the source table. You can NOT do backups and restores across regions.

Point-in-Time Recovery (PITR):

- Protects against accidental writes or deletes
- Restore to any point in the last 35 days
- Incremental backups
- Not enabled by default

- Latest restorable point: five minutes in the past

Streams:
DynamoDB Streams captures a time-ordered sequence of item-level modifications in any dynamodb table and stores this information in a log for up to 24 hours.

Applications can access this log and view the data items as they appeared before and after they were modified, in near-real time.

A stream is an ordered flow of information about changes to items in a dynamodb table.
When you enable a stream on a table, dynamodb captures information about every modification to data items in the table.

Provides a stream of inserts, updates, and deletes to the dynamodb table items.

Where to apply this:
- Cross-region replication, which dynamodb implements automatically (global tables)
- Messaging or notification applications.
- Aggregation or filtering.
- Analytical reporting
- Archiving
- Auditing
- Search, etc

Combined with Lamda functions for functionality like stored procedures.

Global Tables: Managed Multi-Master, Multi-Region Replication.
You replicate your data from one region to another that we choose.

- Globally distributed applications
- Based on DynamoDB streams
- Multi-region redundancy for Dissaster Recovery or HA
- No application rewrites
- Replication latency under one second

To enable global tables we will need to enable dynamodb streams before.
Will take a while to be replicated in the other region when you enable it.

Once the tables are replicated in the other region if you add a new item it will appear in the other region in seconds.

Database Migration Services (DMS):
You can migrate different db products from your prem, or EC2, or RDS by using DMS.

You configure the logic into DMS and it is doing the rest automatically.

During the migration the source db remains completely operational during the migration.

Until now DynamoDB is not a supported source db. You can migrate TO (target) dynamo but not FROM (source).

Security:

- Encryption at rest using KMS
- Site-to-site VPN
- Direct Connect (DX)
- IAM policies and roles
- Fine-grained access
- CloudWatch and CloudTrail
- VPC endpoints connect our ec2 via private IP addresses without exposing dynamodb to inet.

Redshift:

Fast and powerfull, fully managed, petabyte scale data warehouse service in the cloud.

OLAP transactions.

Uses different type of architecture both from a database perspective ifr layer.

Type of configurations:

- Single Node (160GB)

- Multi-Node
  - Leader Node (manages client connection and receives queries)
  - Compute Node (store data and perform queries and computations). Up to 128 Compute nodes.

Advanced Compression:
Columnar data stores can be compressed much more than a row-based ones because similar data is stored sequencially on disk.

Redshift employs multiple compression techniques and can often achieve significant compression relative to transactional relational data stores.

In addition, redshift doesn’t require indexes or materialized views, and so uses less space than traditional relationa database systems.

When loading data into an empty table, redshift automatically samples your data and selects the most appropiate compression scheme.

Massively Parallel Processing (MPP):
Automatically distributes data and query load acorss all nodes.

Redshift makes it easy to add nodes to your data warehouse and enables you to maintain fast query performance as your data warehouse grows.

Backups:

- Enabled by default with a 1 day retention period.
- Maximum retention period is 35 days.
- Redshift always attempts to maintain at least three copies of your data (the original and replica on the compute nodes and a backup in S3)
- Can also asynchronously replicate your snapshot to s3 in another region for DR.

Prices:

- Compute nodes hours (total number of hours you run across all your compute nodes for the billing period. You are billed for 1 unit per node per hour,so a 3-node data warehouse cluster running persistently for an entire month would incur 2,160 instance hours. You will not be charged for leader node hours; only compute nodes will incur charges)

- Bakups
- Data transfer (only within VPC, not outside it)

Security:

- Encrypted in transit using SSL
- Encrypted at rest using AES-256 encryption
- By default Redshift takes care of key management
  - Manage your own keys through HSM
  - KMS

Availability:

- Currently only available in 1 AZ
- Can restore snapshots asynchronously to new AZs in the event of an outage

Exam Tips:

- Redshift is used for busines intelligence
- Available in only 1 AZ
- Backups:
  - Enabled by default with a 1 day retendion period
  - Maximum retention period is 35 days
  - Redshift allways attemps to maintain at lease three copies of your data (the original and replica on the compute nodes and a backup in S3)
  - Can also asynchronously replicate your snapshots to s3 in another region for DR

Aurora

Amazon Aurora: is a MySQL and PostgreSQL compatible relational db engine.

Combines the speed and availability of comercial db with the simplicity and cost efectiveness of open source db.

Benefits:
Provides up to 5 times better performance than MySQL and three times better than PostgreSQL at a much lower price point, whilst delivering similar performance and availability.

Things to know:

- Start with 10GB, scales in 10GB increments to 64TB (Storage Autoscaling)

- Compute resources can scale up to 32vCPUs and 244GB of Memory.
- 2 copies of your data is contained in each AZ, with minimum of 3 AZ. 6 copies of your data

Scaling Aurora:

- Is designed to transparently handle the loss of up to two copies (AZ) of data without affecting databases write availability and up to three copies without affecting read availability.

- Aurora storage is also self-healing. Data blocks and ddisks are continously scanned for errors and repaired automatically.

Types of Aurora Read Replicas:

- Aurora Replicas (currently 15)
- MySQL Read Replicas (curently 5)
- PostgreSQL (currently 1)

Backups:

- Automated backups are always enabled on Aurora db. Backup do NOT impact database performance.
- You can take snapshots, which has NO impact on performance.
- You can share Aurora snapshots with other aws accounts.

Aurora serverless:

Aurora serverless is an on-demand, autoscaling configuration for the MySQL and PostgreSQL compatible editions of Aurora.

An aurora serverless db cluster automatically starts up, shuts down and scales capatity up or down based on your application's needs.

Aurora serverless provides a relative simple, cost-effective option for infrequent, intermittent, or unpredictable workloads.

Exam Tips:

- 2 copies of your data are contained in each AZ, with minimum of 3 AZ. 6 copies of your data

- You can share Aurora Snapshots with other AWS accounts

- 3 types of read replicas available. Aurora Replicas, Mysql Replicas and PostgreSQL replicas. Automated failover is only available with Aurora Replicas.

- Aurora has automated backups turned on by default. You can also take snapshots with Aurora. You can share these snapshots with other aws accounts.

- Use Aurora Serverless if you want a simple, cost-effective option for infrequent, intermittent, or unpredictable workloads.

Elasticache:

What is Elasticache:
Is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud.

The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.

Example: You have a very popular website and you want to store the most common queries to the cache.

Then the next time that a user want to use one of the most common queries will find them in the cache instead of the db, releasing workload and being served faster because the cache uses memory instead of disks.

Engines supported:
- Memcached
- Redis

Memcached vs Redis:
Memcached if you want a really simple cache to offload your db. Able to scale horizontaly and you get multithreaded performance.

Redis for more complex things like Advanced data types, ranking and sorting datas, pub/sub capabilities, persistence, multi-AZ, backup and restore capabilities.

Exam Tips:
- Use Elasticache to increase database and web application performance.
- Redis is Multi-AZ
- You can do backups and restore of Redis.

Database Migration Service (DMS):

What is DMS: Is a way to migrating db.

Cloud service that makes it easy to migrate relational db, data warehouses, NoSQL db, and other types of data stores.

You can use DMS to migrate your data into the AWS cloud, between on-premises instances (through an AWS cloud setup), or between combinations of cloud and on-premises setups.

How it works:
At most basic level, AWS DMS is a server in the aws cloud that runs replication software.

You create a source and target connection to tell dms where to extract from and load to.

Then you schedule a task that runs on this server to move your data.

DMS creates the tables and associated primary keys if they don't exist on the target.

You can pre-create the target tables manually or you can use AWS Schema Conversion Tool (SCT) to create some or all of the target tables, indexes, views, triggers, etc.

Type of DMS migrations:

- Supports homogenous migrations: The source and the target db are the same, example: migrating from Oracle to Oracle.

- Supports heterogeneous migrations: You can migrate from one db engine to another, example: from SQL Server to Aurora

Sources	Targets

• On premises and EC2 instances databases:
  ◦ Oracle, SQL Server, MySQL, MariaDB,PostgreSQL, SAP, MongoDB, Db2
• Azure SQL Database
• Amazon RDS (including Aurora)
• Amazon S3
	• On premises and EC2 instances databases:
	  ◦ Oracle, SQL Server, MySQL, MariaDB,PostgreSQL, SAP, MongoDB
	• RDS
	• Redshift
	• DynamoDB
	• S3
	• Elasticsearch service
	• Kinesis Data Streams
	• DocumentDB

Migration:

Homogeneous migration: Source → EC2 instance running DMS → Target

Heterogeneous migration: Source → EC2 instance running DMS and SCT → Target

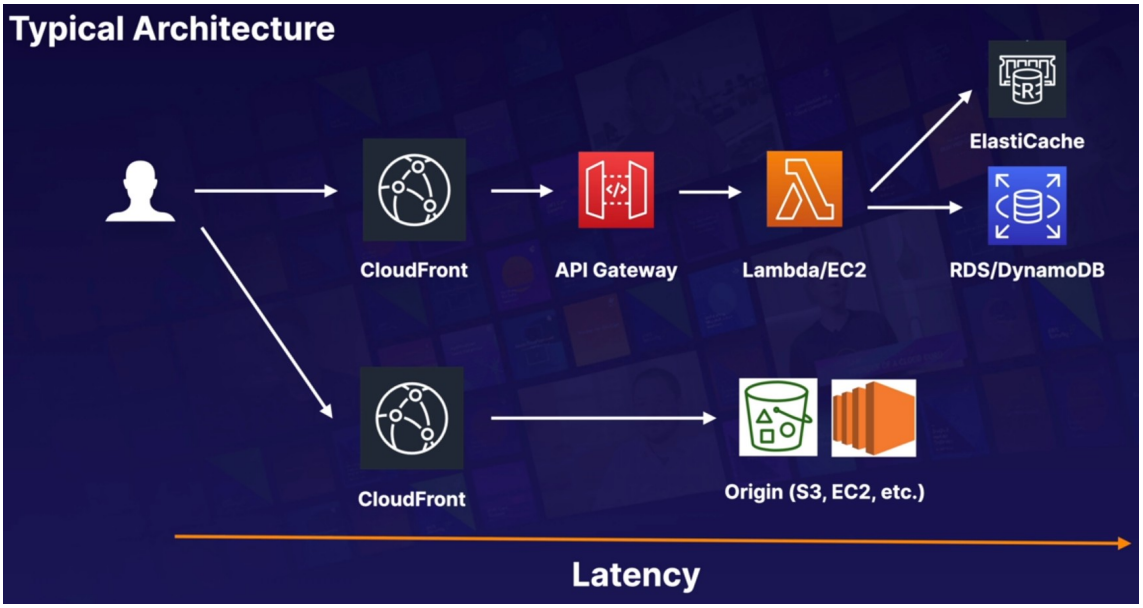
Exam Tips:

- DMS allows you to migrate DBs from one source to AWS
- The source can be on premises, inside AWS or another cloud provider.
- You can do homogeneous migrations (same engines).
- If you do a heterogeneous migration, you will need the AWS Schema Conversion Tool (SCT)

Catching Strategies:

Which services have caching capabilities:

- CloudFront
- API Gateway
- ElastiCache → Memcached and Redis
- DynamoDB Accelerator (DAX)



The more cache you put in front of your application the lower the latency is for your end user

Exam Tips: Catching is a balancing act between up-to-date, accurate information and latency, we can use the following serivces to cache on aws:

- CloudFront
- API Gateway
- ElastiCache → Memcached and Redis
- DynamoDB Accelerator (DAX)

EMR Overview:

Amazon EMR: Elastic Map Reduce. Solution for **Big Data** hosted on AWS.

Industry leading cloud big data platform for processing vast amounts of data using open source tools such as Apache Spark, Apache Hive, etc.

With EMR you can run petabyte scale analysis at less than half the cost of traditional on premises solutions and over 3 times faster than standard apache spark.

Components: The central component is the cluster.

A cluster is a collection of ec2 instances. Each instance in the cluster is called a node. Each node has a role within the cluster, refered to as the node type.

EMR also install different software components on each node type, giving each node a role in a distributed application like Apache Hadoop.

Node types:

- **Master Node:** Node that manages the cluster. The master node tracks the status of tasks and monitors health of the cluster. Every clulster has a master node.

- **Core node:** A node with software components that runs tasks and stores data in the Hadoop Distributed File System (HDFS) on your cluster. Multi-node clusters have at least one core node.

- **Task node:** A node with software components that only runs tasks and does not store data in HDFS. Task nodes are optional.

Logs: You can persist your logs to S3 and you have to do it when you set up the cluster.

The logs are stored in the master by default. To avoid losing the logs you can configure the cluster to periodically archive the log files stored on the master node to S3.

This ensures the log files are available after the cluster terminates, whether this is through normal shutdown or due to an error.

Amazon EMR archives the log files to S3 at five-minutes intervals.

Exam Tips:

- EMR is used for big data processing
- Consists on a master node, a core node and (optionally) a task node
- By default, log data is stored on the master node
- You can configure replication to S3 for all log data from the master node, this can only be configured when creating the cluster for the first time.

Advanced IAM:

AWS Directory Services:

- Family of managed services
- Allow you to connect AWS resoures with on premises AD
- Standalone directory in the cloud
- Allows users to access AWS resources with their existing corporate credentials. You can also log in into AWS console using the same corporate credentials.
- Single sign on (SSO) to any domain joined EC2 instance

Active directory:

- On premises directory used by most enterprises.
- Hierarchical database of users, groups, computers organized in trees and forests.
- You can apply group policies to help you manage users and devices on a network.
- Is based in LDAP and DNS.
- Supports Kerberos, LDAP, and NTLM authentication.
- Highly available

Services:

AWS Managed Microsoft AD: This managed service provides:

- AD domain controllers (DCs) running in real Windows Servers.
- By default you get 2 DCs for HA each on it's own AZ.
- These DC are reachable by applications in your VPC.
- You can add additional DC to increase availability or transaction rates.
- You have exclusive access to the DC, no other aws user have access to these DC, you can be confident about the security.
- You can extend the AD to your on premises using AD Trust.
- Is the best choice if you have more than 5000 users.

Shared responsability:

AWS:

- Will do a multi AZ deployment for you.
- Patching, monitor, recover DC
- Instance rotation (checking that you are in the latest version of the software)
- Backup operations: Snapshot and restore

Customer:

- Users, groups, Group Policy Objects (GPO)
- Standard AD tools
- Scale out DCs
- Trusts (resource forest)

- Manage Certificate authorities (LDAPS)
- Federation

Simple AD: The ligth version of Managed Microsoft AD.

- Standalone managed directory in the cloud
- Used to support basic AD features
- Sizes:
  - Small: <= 500 users
  - Large: <= 5000 users
- Easier to manage EC2 instances when you want to use your existing corporate credentials to log in into these ec2 instances rather that provisioning users and passwords on these instanes or manage keys.
- Linux workloads that need LDAP
- Does NOT support trusts (can't join on premises AD)

AD Connector:

- Directory gateway (proxy) for on-premises AD
- Avoid caching information in the cloud.
- Allow on premises users to log in to AWS using AD
- Join EC2 instances to your existing AD domain.
- Scale across multiple AD Connectors.

Differences between the 3 services:

- **Managed Microsoft AD:** Hosted on the AWS cloud. Feature-rich managed AD. The best choice when you have more than 5000 users

- **Simple AD:** Hosted in the AWS cloud. Inexpensive AD compatible service with the common directory features. The best option for up to 5000 users.

- **AD Connector:** Connects your AD in your premises to AWS. The best option to use your AD with AWS services.

Cloud Directory:

- Directory based store for developers
- Multiple hierarchies with hundreds of millions of objects
- Use cases: org charts, course catalogs, device registries.
- Fully managed service.

Amazon Cognito User Pools:

- Managed user directory for SaaS application.
- Sign-up and sign-in for web or mobile
- Works with social media identities. Example: You can log in into a SaaS application using your facebook, google or amazon credentials.

Exam tips: For the exam you have to distinguish which services are AD compatible and which are not

- AD compatible:
  - Managed Microsoft AD
  - AD Connector
  - Simple AD

- Non AD compatible:
  - Cloud Directory
  - Cognito user pools

IAM Policies:

Amazon Resource Name (ARN): Uniquely identifies any resource in AWS

Content of an ARN: arn:partition_name:service:region:account_id:

Begin with:

-- arn is always arn
-- partition_name: There are different partitions, aws | aws-cn (China)
-- service: s3|ec2|rds, etc
-- region: us-east-1 | eu-central-1, etc.
-- account_id: twelve digits number.

End with:

resource
resource_type/resource
resource_type/resource/qualifier
resource_type/resource:qualifier
resource_type:resource
resource_type:resource:qualifier

Examples:

-- arn:aws:iam::123456789012:user/mark → two colon (::) because IAM is global. user is the resource_type and mark is the resource

-- arn:aws:s3:::my_awesome_bucket/image.png → ::: because all s3 objects are globally unique so they have their own unique identifier.

-- arn:aws:dynamodb:us-east-1:123456789012:table/orders → table (resource type) orders, (resource)

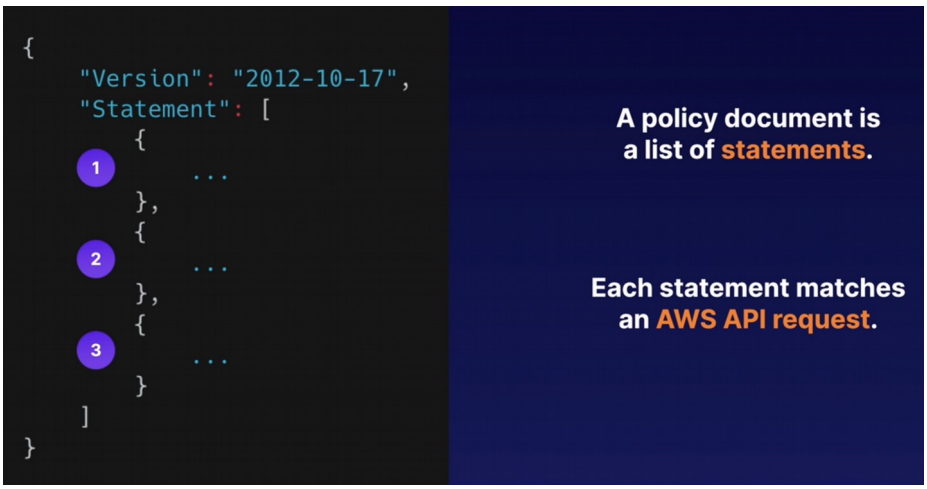
-- arn:aws:ec2:us-east-1:123456789012:instance/* → * all the resources

IAM Policies: JSON document that defines permissions.
- Identity policy: Attached to user, group or role
- Resource policies: Attached to resources like s3 buckets, SQS queues, KMS keys, etc.

IAM policies must be attached to an identity or resource, otherwise it won't have any effect.

List of statements: IAM policies are simply structured as a list of statements.

Basic format of IAM policy:



-- JSON statement always starts with the version number 2012-10-17. Helps AWS to identify the structure of the document.

-- [Array /List.] → Inside we indicate the statements.

-- {Statement} → Each statement matches an AWS API request. We have 3 statements in the list above.

-- API request: Any action that you perform on AWS. When you start an ec2 instance, create a table in dynamo db or get an object in s3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGet*",
                "dynamodb:DescribeStream",
                "dynamodb:DescribeTable",
                "dynamodb:Get*",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWrite*",
                "dynamodb:CreateTable",
                "dynamodb:Delete*",
                "dynamodb:Update*",
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
        }
    ]
}
```

Sid: Human readable string that tells you what this statement is for.

Effect: Allow or Deny.

Action: Have the form → Service_name: Action_name

The * means that any API request that start with that particular string, BatchGet, Get, etc.

Resouce: The resource the Action is against.

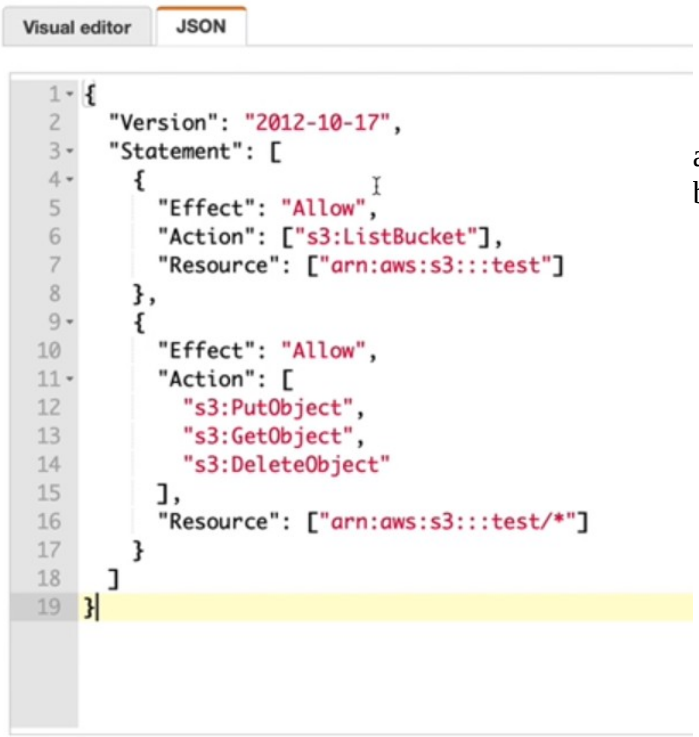
Creating and Applying an IAM policy:

Security, Identity, & Compliance -- IAM -- Policies, there are two types of policies:

- AWS managed policies: Created by AWS for you for convenience and are denoted by the orange box. Not editable by you but you can use as many as you like.

- Customer managed policies: Created by us.

-- Create policy -- JSON, paste the policy:



1st statement: allow to list the bucket "test"

2nd statement: Allow to upload, download and delete everything that is in the "test" bucket

-- Review policy -- Give it a name -- Create policy

Attach the policy to a role:
The policy by itself has no effect, you have to attach it to a role.

IAM Dashboard -- Roles -- Create role -- AWS service -- ec2 -- Next: Permissions -- Search for the policie created previously -- Next, Next, name it and Create role.

Now this policy have effect. If you attach this role to any ec2 instance it will be implicitly granted access or deny access based on what the policy contains.

You can attach one or more policies to the Role you've created before, hit Attach policies -- search the one you need, a managed one by aws for instance -- Attach policy

Exam tips:

- Any permissions that are not explicitly allowed are implicitly denied
- A policy explicitly deny that overwrite anything else in any other policies.Denies will always overwrite.

- Only attached policies have effect.
- When you have several policies attached to either an identity or resource AWS will joins all those policies together when it performs its evaluation.
- Understand AWS managed policies vs customer managed policies.
-

Permission Boundaries: Used to delegate admninistration to other users.

Prevent privilege escalation or unnecessaryly broad permissions.

AWS supports permission boundaries for IAM entities.

Users or Roles permissions boundaries is an advanced feature for using a managed policy to set maximum permissions that an identity based policy can grant to an IAM entity.

Permissions and entities permissions boundary allows it to perform only the actions that are allowed by both its identity based policies and its permission boundaries.

In other words, it controls the maximum permissions an IAM policy can grant.

Use cases:
- Developers creating roles for lamda function
- Application owners creating roles for EC2 instances
- Admins creating ad hoc users.

How to apply it:
We have an user with the AdministratorAccess policy attached. But we want to limit his access to just DynamoDB.

Into the user click on Permissions boundary -- Set boundary -- search the policy you want, DynamoDBFullAccess -- Set boundary.

Now the user even having full admin permissions will only be able to work with the DynamoDB ones.

Resource Access Manager (RAM):

What is it: RAM allows resource sharing between accounts.

Which resources can I share with RAM:
- App Mesh
- Aurora
- CodeBuild
- EC2
- EC2 image builder
- License manager
- Resource group
- Route 53

Example:

We have 2 aws accounts, account1 and account2.

In account1 we have an AuroraDB and we want to share it with account2. We can grant access using RAM.

Into account1:
Go to Security, Identity, & Compliance -- Resource Access Manager -- give it a name: Aurora Cluster -- select the resource to share, Aurora db clusters -- tick the box of your aurora db -- Pincipal: Allow external account: add the account2 id-- Create resource Share

You will see the new resource share called Aurora Cluster as status active.

But if you click into this new resource account you will see that the shared principals is still associating.

RAM works by sending an invitation from account1 to account2. So you have to go to acccount2 to accept that invitation.

Into account2:
Go to Security, Identity, & Compliance -- Resource Access Manager -- Shared with me -- you will see a pending invitation, click on it and accept resource share.

Now if you go to RDS and refresh you will see the aurora database shared from the account1.

AWS Single Sign-On (SSO):

What is this: Service that helps centrally manage access to AWS accounts and business applications.

3rd party applications like:

- Atlassian
- Office 365
- G Suite
- Salesforce
- Dropbox
- Github
- Slack
- etc

SAML 2.0-enabled applications: Security Assertion Markup Language
Standard for login users into applications based on their sessions on another context.

One context may be my Microsoft AD environment and other context being bussines applications like G Suite or Office 365.

SAML allows you to log in into your G suite application using the Microsoft AD context.

All sign on activities are recorded on CloudTrail.

Exam Tip:

- If you see in the exam SAML you have to choose the answer that contains single sign on (SSO) on it.

Route 53

DNS 101:

What is DNS: Is like a phone book (yellow pages)

DNS is used to convert human friendly domain names into an internet protocol (IP) address.

Is a way of looking up a domain name and getting up an IP address.

Is like when you to look up for someones name in the yellow pages and you get the phone number.

Top Level Domains (TLD):

Is the last word in a domain name, the one that comes after the period (.)

Top Level Domain:

- .com
- .edu
- .gov
- .es
- .cat

Second Level Domain:

- **.co**.uk → .co is the second level domain
- **.gov**.uk → .gov is the second level domain
- **.com**.au → .com in the second level domain here.

Internet Assigned Numbers Authority (IANA):
Is the authority that controlls the Top Level Domain Names. It does it in a root zone database which is a db with all available tld.

You can view this db here: https://www.iana.org/domains/root/db

Domain Registars:
The domain registars are authorities that can assign domain names directly under one or more tld.

All domains must be unique, to achieve that the domains are registered with InterNIC, a service of ICANN, which enforces uniqueness of domain names across the internet.

Each domain name is registered in a central database know as WhoIS database.

Popular domain registars are: Amazon, GoDaddy.com, 123-reg.co.uk, etc.

Start of Authority Record (SOA):
The SOA record stores information about:

- Name of the server that supplied the data for the zone
- The administrator of the zone
- The current version of the data file
- The default number of seconds for the time to live file on resouce records

NS Records (Name Server):
They are used by TLD servers to direct traffic to the Content DNS server which contains the authoritative DNS records.

Example:

1. We have the domain: hellocloudgurus.com → The user type into his browser the domain. The browser doesn't know the IP for that domain.
2. The browser goes for the TLD server, querying it for the authoritative DNS records saying: hey I've got this domain, hellocloudgurus.com, and I need to know the IP address.
3. The TLD server doesn't contains the IP address. It contains the top level domain, the ns record, ns.awsdns.com in this example.
4. We then query the NS records.
5. The NS records will give us the Start Of Authority (SOA)
6. Inside the SOA is where we find the DNS records

DNS records: Consist on different thigs:

- **A records:** The fundamental type of DNS record. The A stands for Address. Is used by a computer to translate the name of the domain to an IP address. Is like the yellow pages, we have the name and is gonna give us the number.

- **TTL:** Time To Live. The time, in seconds, that a DNS record is cached on either the Resolving Server or the users own local PC. The lower the ttl, the faster changes to DNS records take to propagate throughout the internet.

- **CNAME:** Canonical Name. Can be used to resolve a domain name to another. Example: You may have a mobile website for acloud.guru, like m. acloud.guru, and mobile. acloud.guru to the same address. Instead of using different IP addresses you just need to map one to the other with CNAME. Like to see the IP of m.acloud.guru to acloud.guru

- **Alias Records:** Used to map resource records sets in your hosted zone to Elastic Load Balancers, CloudFront distributors, or S3 buckets that are configured as websites. Example: You can mape www.example.com to another target DNS name, elb1234.elb.amazonaws.com

- CNAME vs Alias:

  A CNAME can NOT be used for naked domain names (zone apex record).

  You can NOT have a CNAME for http://acloud.guru, it must be either an A record or an Alias.

Another explanation:
Alias Records have special functions that are not present in other DNS servers. Their main function is to provide special functionality and integration into AWS services.

Unlike CNAME records, Alias can also be used at the Zone Apex, where CNAME records cannot.

Alias Records can also point to AWS Resources that are hosted in other accounts by manually entering the ARN

Exam Tip:

- ELBs do NOT have pre-defined IPv4; you resolve to them using a DNS name
- Understand the difference between an Alias and a CNAME.
- On the exam, given the choice, always choose an Alias over a CNAME.
- SOA records (Start Of Authority)
- NS → name server
- A → address
- CNAMES
- MX → mail exchange
- PTR → reverse of an A record, gives you the name from an IP

Register a Domain Name:

Exam Tip:

- You can buy domain names directly with AWS.
- It can take up to 3 days to register depending on the circumstances.

Routing Policies Available on Route53:

The following policies are available at route53:

- Simple Routing
- Weighted Routing
- Latency-based Routing
- Failover Routing
- Geolocation Routing
- Geoproxymity Routing (Traffic Flow Only)
- Multivalue Answer Routing

Simple Routing Policy:
If you choose the simple routing policy you can only have one record with multiple IP addresses.

If you specify multiple values in a record, Route 53 returns all values to the user in a random order.

So if you have 3 IPs in an A record and you paste the domain name in the browser you will get randomly one of the 3 IPs. Once you got one you will get the same IP all the time until you refresh your local DNS or you change the ttl in route53 (in this case you will need to wait a few minutes to see the changes applied)

Weighted Routing:
Allow you to split your traffic based on different weights assigned.

For example, you can set 10 % of your traffic to go to US-EAST-1 and 90% to go to EU-WEST-1

So you create several A records, one for each public IP. Then you assing to each record a weight like:

AZ on Sydney: 20
AZ on Ireland: 30
AZ on Ohio: 50

The 50% of the traffic will go to Ohio, the 30% to Ireland and the 20% to Sydney.

Health Checks:
You can set health checks on individual record sets.

If a record set fails a health check it will be removed from Route53 until it passes the health check.

You can set up SNS notifications to alert you if a health check is failed.

Latency-based Routing:
Allows you to route your traffic based on the lowest network latency for your end user (ie, wich region will give them the fasted response time)

To use latency based routing, you create a latency resource record set for the ec2 or lbs resource in each region that hosts your website.

When route53 receives a query for your site, it selects the latency resource record set for the region that gives the user the lowest latency.

Route 53 then responds with the value associated with that resource record set.

Failover Routing Policy:
Are used when you want to create an active/passive set up. For example, you may want your primary site to be in eu-west2 and your secondary Dissaster Recovery Site in ap-southeast-2

Route 53 will monitor the health of your primary site using a health check.

A health check monitors the health of your end points.

Geoloaction Routing Policy:
Lets you choose where your traffic will be sent based on the geographic location of your users (ie the location from which DNS queries originate)

For example, you might want all queries from Europe to be routed to a fleet of ec2 instances that are specially configured for your european customers, These servers may have the local language of your european customers and all prices are displayed in euros.

Geoproximity Routing (traffic flow only):

Lets route53 route traffic to your resources based on the geographical location of your users and your resources. You can also optionally choose to route more traffic or less to a given resource by specifying a value, know as a bias.

A bias expands or shrinks the size of the geographic region from which traffic is routed to a resource.

To use geoproximity routing, you must use Route 53 traffic flow.

Multivalue Answer Policy:
Lets you configure route53 to return multiple values, such as IP addresses for your webservers, 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 route53 returns only vallues for healthy resources.

This is similar to simple routing however it allows you to put health checks on each record set.

VPC:

Tips for the exam:
It will have 10 questions about vpc.

You should be able to create a vpc by memory the day before the exam to pass it.

What is VPC: Basically is a Virtual datacenter in the cloud.

Amazon Virtual Private Cloud lets you provision a logically isolated section of the AWS cloud where you can launch aws resources in a virtual network that you define.

You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

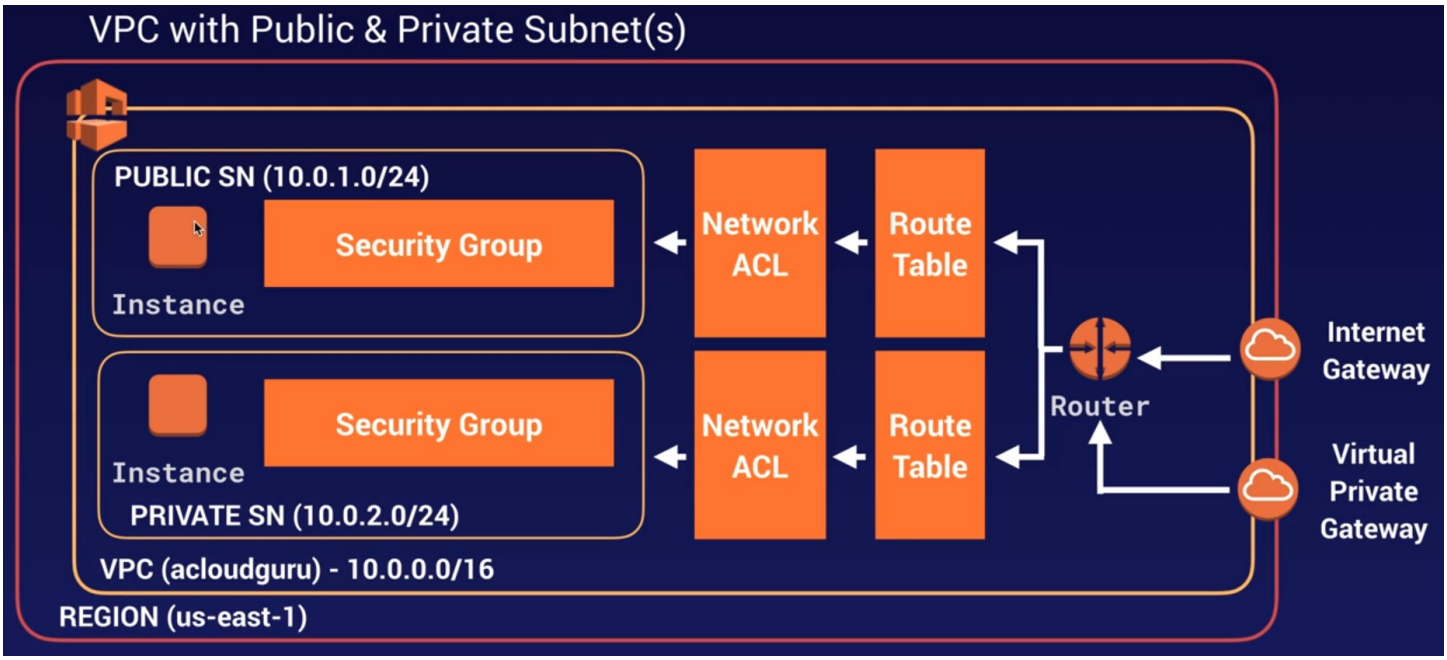
You can easily customize the network for your private cloud.

Example:
You can create a public facing subnet for your web servers that has access to internet.
And place your backend services such as DBs or app servers in a private facing subnet with no Internet access.

You can leverage multiple layers of security, including security groups and network access control lists, to help control access to ec2 instances in each subnet.

VPN:
Additionally you can create a Hardware Virtual Private Network connection between your corporate datacenter and your VPC and leverate the aws cloud as an extension of your corporate datacenter.

Diagram:



- The red line represents a region., us-east-1
- Inside the region we have the vpc, acloudguru 10.0.0.0/16
- Outside the vpc we have two ways of connecting:
  - Internet gateway
  - Virtual private gateway
- Both connections goes to a Router
- The router directs traffic to different route tables
- The route table directs traffic to Netwok ACL, first line of defense, also acting like firewalls. NACL are STATELESS and can block specific IP addresses on the nacl.
- Then move over the Security Group which STATEFULL.
- There we have two different subnets:
  - Public: Internet is accesible for any ec2 instances in the public subnet.
  - Private: ec2 instances can not access the internet on their own.
    - You can still connect into these ec2 instances but the way you do that is ssh to a server in the public subnet (called jump box or bastion host) and from there ssh to the server in the private subnet.

Private IPs:
10.0.0.0 - 10.255.255.255 → /8
172.16.0.0 - 172.31.255.255 → /12
192.168.0.0 - 192.168.255.255 → /16

What can we do with a VPC:

- We can launch instances into a subnet of your choosing.
- Assign custom IP addresses ranges in each subnet
- Configure route tables between subnets.
- Create internet gateways and attach into our VPC
- Much better security control over your aws resources.
- Instance security groups
- Subnet access control lists ACLS

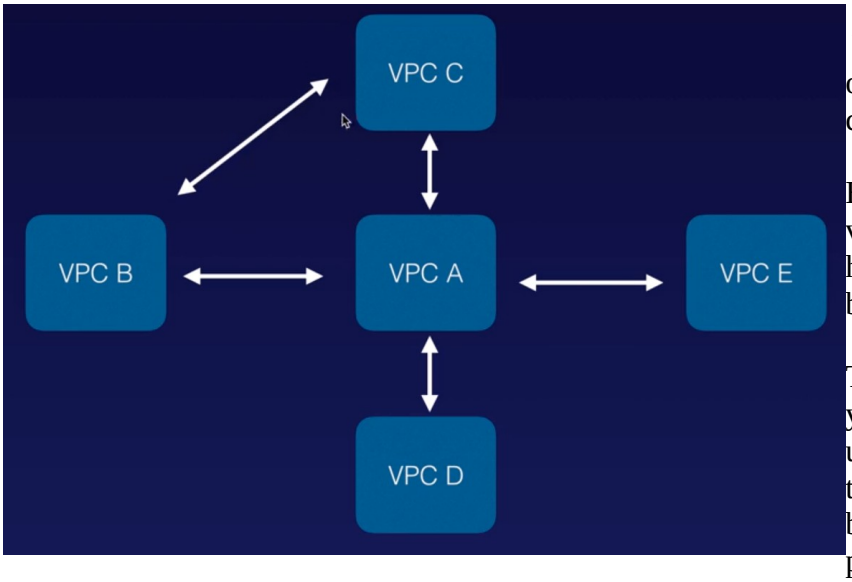
Default VPC vs Custom VPC:

- Default is user friendly, allowing you to immediately deploy instances.
- All Subnets in default VPC have a route out to the internet.
- Each ec2 instance has both a public and private IP address.

VPC Peering:

- Allows you to connect one VPC with another via a direct network route using private IP addresses.
- Instances behave as if they were on the same private network
- You can peer VPCs with other aws accounts as well as with other VPCs in the same account.
- Peering is in a start configuration: ie 1 central VPC peers with 4 others. NO TRANSITIVE PEERING.
- You can peer between regions.

Transitive Peering:



You can go from vpc a to the other vpcs because have a direct connection.

But you can not go from vpc b to vpc c using vpc a as a link. You have to have a direct connection between b and c.

This is No Transitive Peering, you can not peer from B to C using A as intermediary, you have to have a direct connection between B and C like in the picture.

Exam Tip:

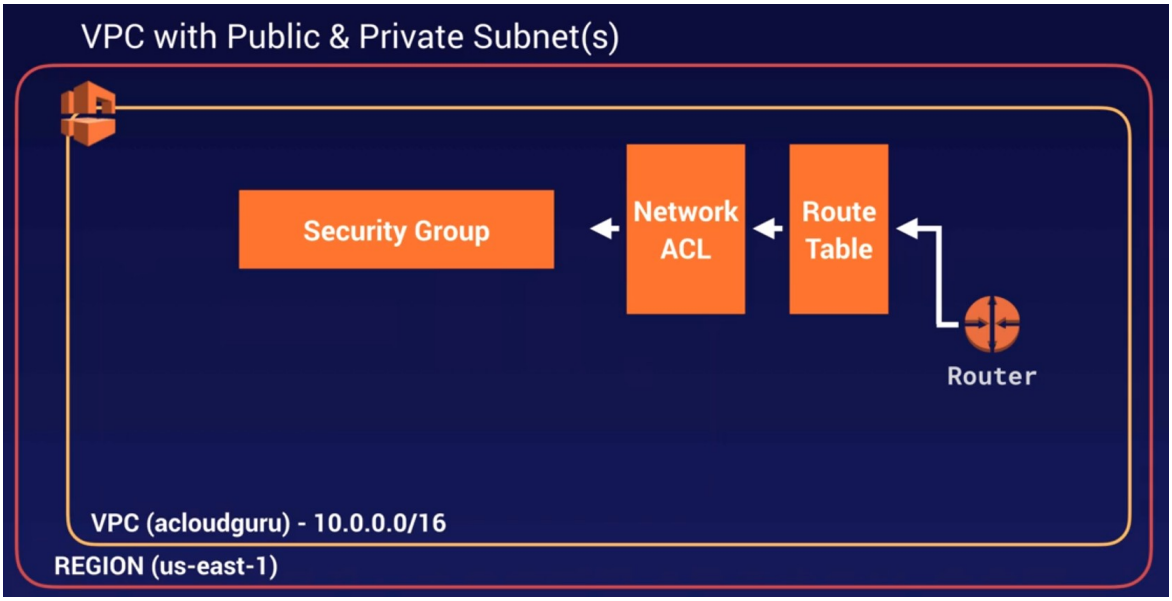
- Think of a VPC as a logical datacenter in AWS

- Consists of:
  - IGWs (Internet gateways) or Virtual Private Gateways
  - Route tables
  - Network Access Control Lists (NACL)
  - Subnets
  - Security Groups.
- 1 Subnet = 1 Availability Zone. You can NOT have the same subnet in different AZ, but you can have multiple subnets in the same AZ.
- Security Groups are Statefull; NACL are Stateless. With NACL you can add Deny rules as well as Allow rules, and when you open up a port on Inbound it does NOT automatically open up a port in the Outbount, you have to go there and open up manually.
- NO TRANSITIVE PEERING

Creating a VPC:
Tenancy -- Select default

When you create a VPC is automatically creating a Route Table, a NACL and a Security Group.

This is how a VPC looks like:



Create a subnet:
Select the VPC ID of the vpc created in the previous steep, acloudguru.

AZ: Select the one you want. Have in mind that everybody is used to pick the az A or az B, leaving the others unused. Amazon detected that and is randomizing this by changing the AZ for each customer, example:

If the aws account, companyA , uses us-east-2a to place his services and the aws account companyB places its services in the same us-east-2a this doesn't means that both companys are going to be in the same datacenter.

To avoid overload the us-east-2a because everybody tends to select the A or B, Amazon randomizes which AZ will be in us-east-2a, so even both companies are in us-east-2a each one of them could be in a different DC.

Subnet name: 10.0.1.0 - us-east-1a

Give it the IP 10.0.1.0/24

And create. Create another called 10.0.2.0 - us-east-1b, give it the IP 10.0.2.0/24

Now you have two subnets. By default doesn't have a Public IP.

Amazon reserved IPs:
Notice that we have 251 available IPs instead of 254.

Amazon by default reserves to it self 3 IPs.
One for the VPC router
One for DNS
One for future use
Then we have the network ip and the broadcast one.

Get public IP to a subnet:
Select the one you need -- Actions -- Modify auto-assin IP settings -- tick Auto-assign IPv4

Internet Gateway:
Until now our network doesn't have an internet gateway, go to:

Internet Gateways -- Create internet gateway -- name it: MyIGW, Create -- Actions, Attach to VPC -- selec the vpc, Attach

You can only create one IGW per VPC.

Route Tables:
Create a new route table, call it MyPublicRoute and select the vpc.

Now we have two route tables, the one created when we've created the vpc, and the last one that will be used for public internet.

The first one is tagged as Main. When you create a VPC by default is assigning the route table created with it as Main.

Edit the public route to create a new one. Tick the public route -- Routes -- Edit routes -- Add route -- 0.0.0.0/0 , Target: Internet Gateway

Do the same for Ipv6:
::/0 and IGW, save routes

Any subnet associated with this subnet will automatically become public.

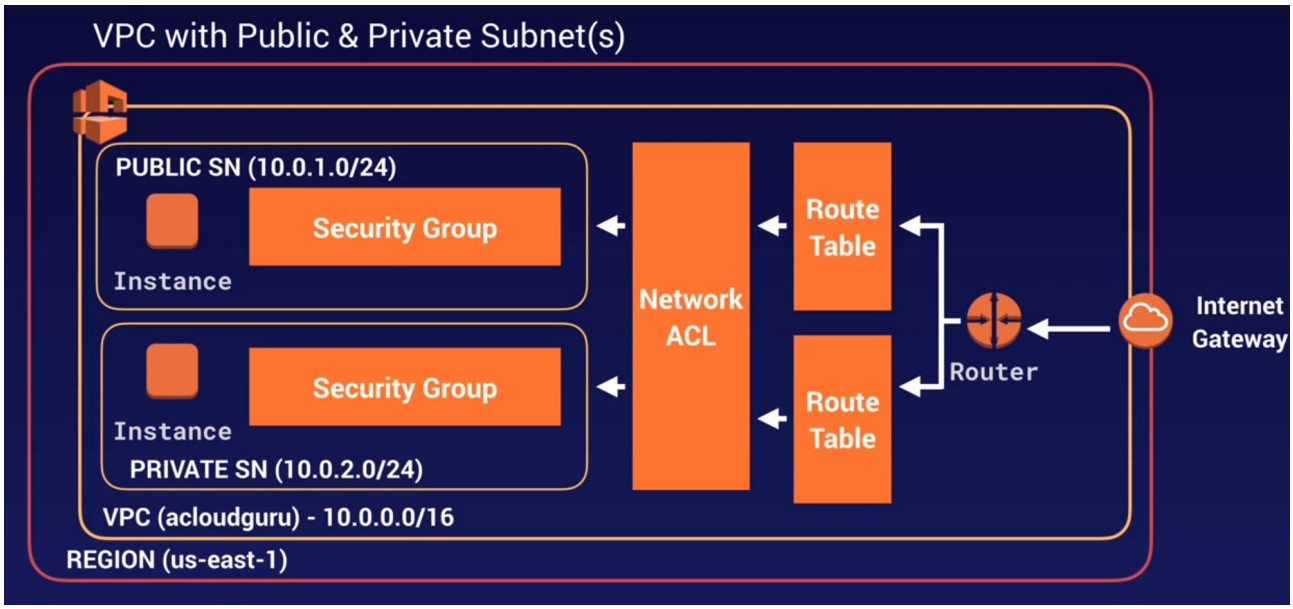
<u>Subnet Associations:</u>
Selected MyPublicroute go to Subnet Associations -- Edit subnet associations , select the first subnet, save

EC2:
Launch an instance, in the network section do:
Network: select the VPC you've created
Subnet: Select the public one

Create a new security group opening 80 and 22 to the world

Launch another instance. Pick your vpc network. Pick the private subnet. Autoassign public ip will be disabled.

Pick the default security group.



Exam Tip:

- When you create a VPC a default Route table, NACL and default security group is created.
- It won't create any subnets, not will it create a default internet gateway
- us-east-1a in your aws account can be a completely different AZ from another aws account. The AZ are randomized
- Amazon always reserve 5 IP within your subnets
- You can only have 1 internet gateway per VPC
- Security groups can't span VPC

Let's communicate both instances:
Both are in separated subnets, one in public the other in private. We want to connect both with ssh.

Create a security group for the private instance. Call it MyDBSG -- select the vpc created by us -- rules: icmp -- source 10.0.1.0/24, do the same for http, https, ssh and mysql/aurora

Now change the security group for that instance: EC2 -- instances -- actions -- networking -- change security group -- select MyDBSG

Now from the public ec2 instance you must be able to ping the private, and ssh to (by copying the private key in the public server, don't do that, this is only for learning, in the real life we use bastion servers).

Private server, how to get updates:

If you go to the private server you won't have inet connection, therefore no updates, new software installs, etc.

You can do this with nat instances and nat gateways.

NAT Instances & NAT Gateways

Allows you to enable ec2 instances in private subnets to still be able to go out and to download software.

In order to do that we need a way to communicate to our inet gateway.

What we are NOT going to do is to make the subnets public.

To do that we will create nat instances and nat gateways. 99% of the time in real life we will use nat gateways.

Difference between Nat Instances and Nat Gateways:

- NAT Instance: Is a simple EC2 instance
- NAT Gateway: Is a hightly available Gateway that allows you to have your private subnets to have internet access without becoming public.

NAT Instances: Let's create one

Go to EC2 Instances -- Launch instances -- Community AMIs -- type: nat, select the first one -- free tier -- change the Network to our VPC -- Subnet: select the public subnet, which will give you a public IP for the instance -- tag it as: Name NAT_instance -- security group: select the public one (22 and 80) -- select to use the general purpose ssd -- launch.

Disable source and destination checks:
Nat instances must work as gateway and to do that we need to disable the source and destination checks.

Select the instance -- Actions -- Networking -- Source / destination check -- tick stop and save

Enable the private instance to talk with the Nat instance:
Go to VPC -- route tables -- select the private one -- add route: 0.0.0.0/0 , target: Instance, select nat _instance -- save

Now the private subnet can go to internet by using the nat_instance as a gateway.

You will be able to ssh from the public server to the private, and you will be able to get internet from in the private server.

Problem with nat instance:
Is a single instance that can easily get overwelmed. Is a single point of failure.

If you delete the nat instance and you take a look to the route table, it will appear as black hole.

NAT Gateways:
The highly available option.

Create one:
In VPC – Nat gateways -- create nat gateway -- subnet: select the public subnet -- Allocate elastic Ip -- create it

Edit the main route table -- add route: 0.0.0.0/0, target: nat gateway -- save

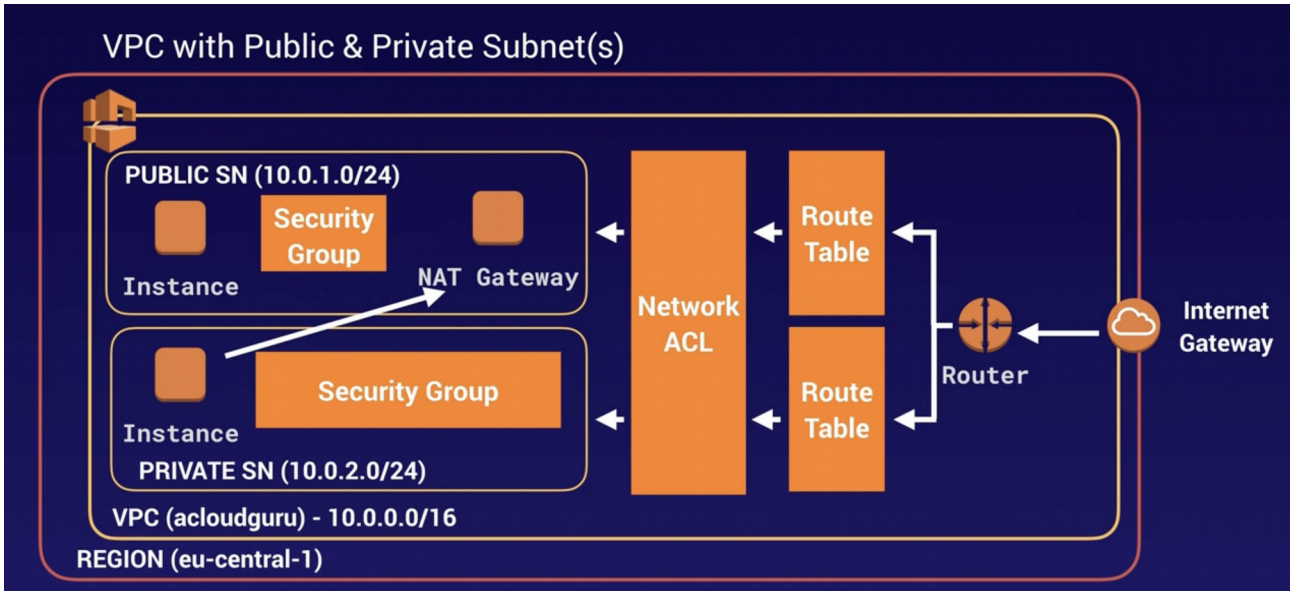
Test it:
Once the nat gateway is created, will take a while, you must be able to use internet in your private instance.

Exam Tip:

- NAT Instances:
  - When creating NAT instance, disable source/destination check on the instance
  - NAT instances must be in a public subnet
  - There must be a route out of the private subnet to the nat instance, in order for this to work
  - The amount of traffic that NAT instances can support depends on the instance size. If you are bottlenecking, increase the instance size
  - You can create high available using Autoscaling Groups, multiple subnets in different AZ, and a script to automate failover.
  - Behind a security group

- NAT Gateways:
  - Redundant inside the Availability zone
  - Prefered by the enterprise
  - Starts at 5Gbs and scales currently to 45Gb
  - No need to patch
  - No associated with security groups
  - Automatically assigned a public ip
  - Remember to update your route tables
  - No need to disable source/destination checks.
  - If you have resources in multiple AZ and they share one NAT gateway, in the event that the NAT gateways AZ is down, resources in the other AZ lose internet access. To create an AZ independent architecture, create a NAT gateway in each AZ and configure your routing to ensure that resources use the nat gateway in the same az.

Here we can see the NAT Gateway:



NACL, Network Access Control Lists:

Exam Tip:

- Your VPC automatically comes with a default NACL, and by default it allows all the inbound and outbound traffic.
- You can create custom NACL. By default, each custom NACL denies all inbound and outbound traffic until you add a rules.
- Each subnet in your VPC must be associated with a NACL. If you don't explicitly associate a subnet with a NACL, the subnet is automatically associated with the default NACL
- Block IP addresses using NACL, not security groups
- You can associate a NACL with multiple subnets; however, a subnet can be associated with only one NACL at a time. When you associate a NACL with a subnet, the previous association is removed.
- Network ACL contains a numbered list of rules that is evaluated in order, starting with the lowest numbered rule.
- NACL have separated inbound and outbound rules, and each rule can either allow or deny traffic.
- NACL are Stateless; responses to the allowed inbound traffic are subject to the rules for outbound traffic (and viceversa)

Custom VPC & ELBs:

When you provision an Elastic Load Balancer you need to provision 2 subnets, is not going to work if you don't do that.

VPC Flow Logs:

What are vpc flow logs:
Is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC.

Flow log data is stored using CloudWatch Logs.

After you've created a flow log, you can view and retrieve its data in CloudWatch logs.

Its a way to store all your traffic in your vpc.

Can be created at 3 levels:

- VPC level
- Subnet level
- Network interface level

How to create flow logs:
Go to Maintenance and Governance – CloudWatch – Logs – Create log group: VPCFlowLogs –

Go to VPC – Your VPCs – select the one you want – Actions – Create flow log – filter: All (to log accepted and denied traffic) – Destination: Sent it to CloudWatch – Destination log group: select the one created in CloudWatch (VPCFlowLog) – Set up permissions (to create a new IAM role) – Role name: FlowLogsRole – Allow

This will kick you out from the previous flow logs creation, open the old tab if you can, you will be in the creation page with the data filled, just select the IAM role recently created (FlowLogsRole) – Create

See the logs:
Maintenance and Governance – CloudWatch – Logs – select VPCFlowLogs and that's it.
You can force to have logs by visiting your website.

Exam Tip:

- You can NOT enable flow logs for your VPCs that are peered with your VPC unless the peer VPC is in your account. This means that you can have logs between peered VPC but have to be in the same AWS Account.
- You can tag flow logs
- After you've created a flow log, you cannot change its configuration; example, you can not change the IAM role after the flow log creation.
- Not all IP traffic is monitored:
  - Traffic generated by instances when they contact the Amazon DNS server. If you use your own DNS server, then all traffic to that DNS is logged.
  - Traffic generated by a windows instance from Amazon Windows license activation
  - Traffic to and from 169.254.169.254 for instance metadata
  - DHCP traffic
  - Traffic to the reserved IP address for the default VPC router.

Bastion Host:

What is it:
Is a safe instance in a public subnet that have the only purpose of being accessed via ssh or rdp from outside the VPC.

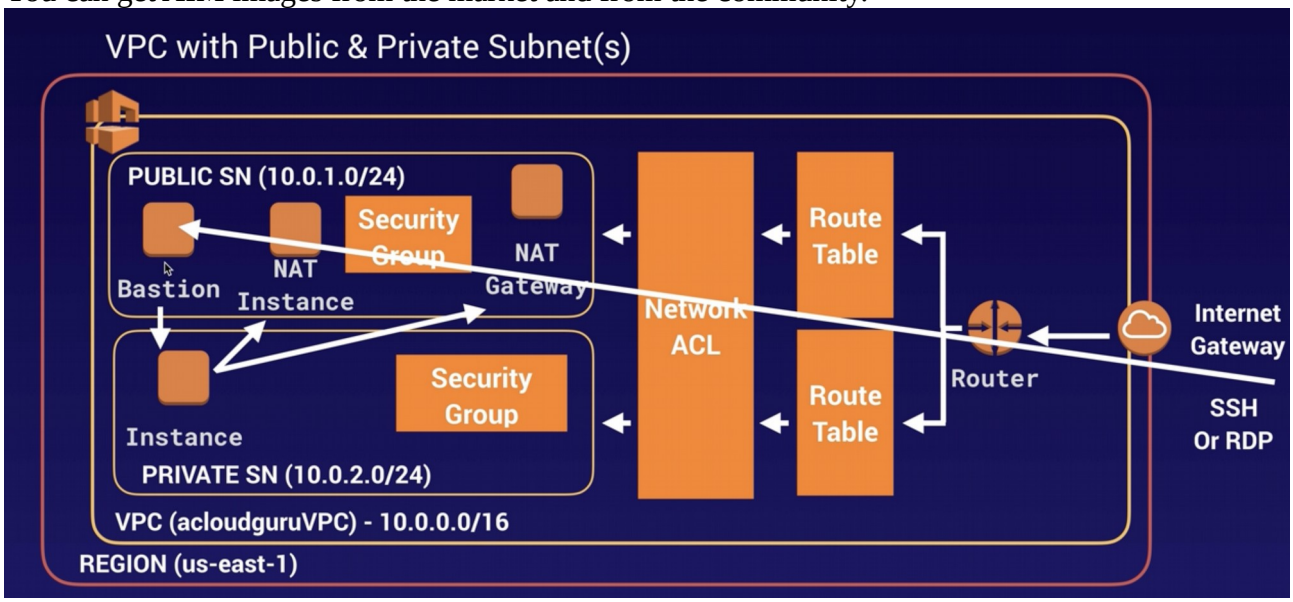
Generally hosts a single application, for example a proxy server, and all the other services are removed or limited to reduce the threat to the instance.

Schema:
The instances in the private network have access to internet using the NAT Instance or the NAT Gateway.

We reach the Bastion instance passing all the steps from outside the VPC, so the internet gatewaym the router, the routing table, the NACL, and the security group.

Then the bastion server forwards the connection through rdp or ssh to the instances in the private network.

You can get AIM images from the market and from the community.



Exam Tip:

- A NAT Gateway or a NAT Instance is used to provide internet traffic to instances in private subnets
- A Bastion is used to securely administer ec2 instances using ssh or rdp. Can be called Jump Boxes to.
- You can not use Nat Gateway as a Bastion host.

Direct Connect:

What is it:
Cloud service solution that makes it easy to establish a dedicated network connection from your premises to AWS.

With it you can establish private connectivity between aws and your office, datacenter, etc, which in many cases can reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than internet based connections.

Schema:
We have the AWS Region. Then the Direct Connect location, that there are a lot spread over the world (could be a cage in your datacenter)

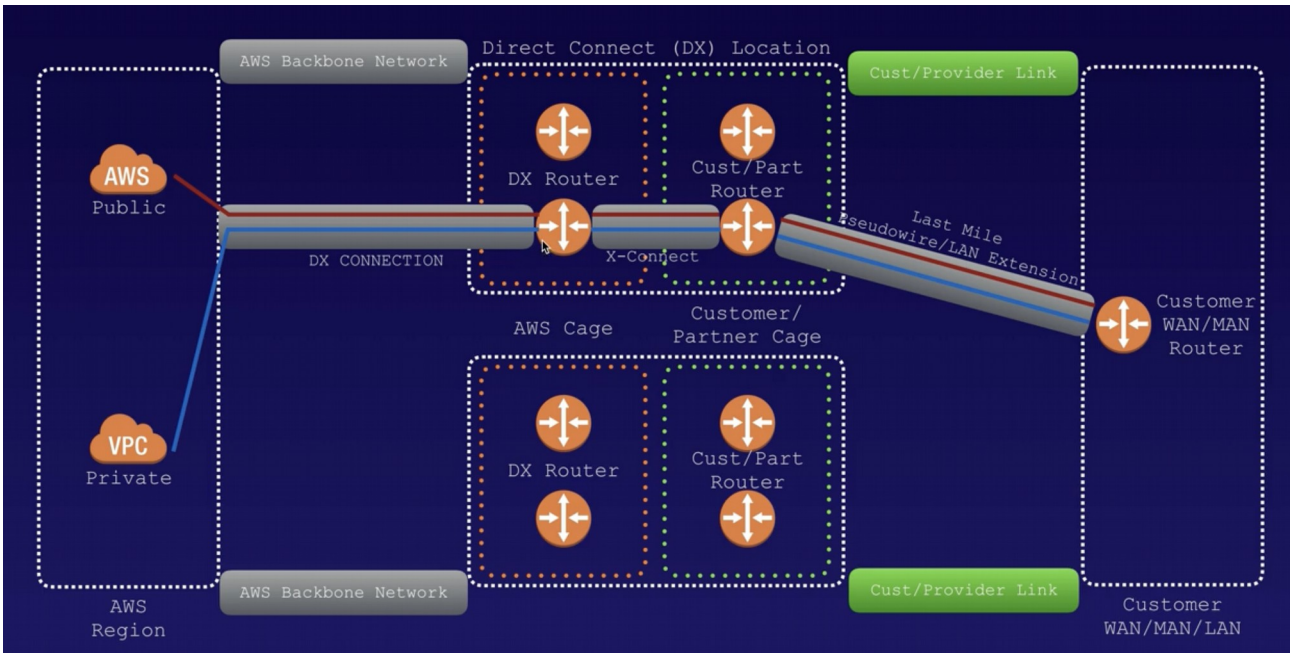
Inside that cage AWS have their own routers. In the same location the customer have a cage with their own routers to.

Then you have a direct connection from your dc to the direct connect location and form it there is a backbone network connecting with aws.

The connection between aws cage and the customer cage is made with a cross connect, essentially is a network cable that connect aws routers in the cage with the customer routers in its own cage to.

From the direct connect location you connect to the customer DC with a dedicated line, MPLS, etc.

You never use internet, you are always in private lines.

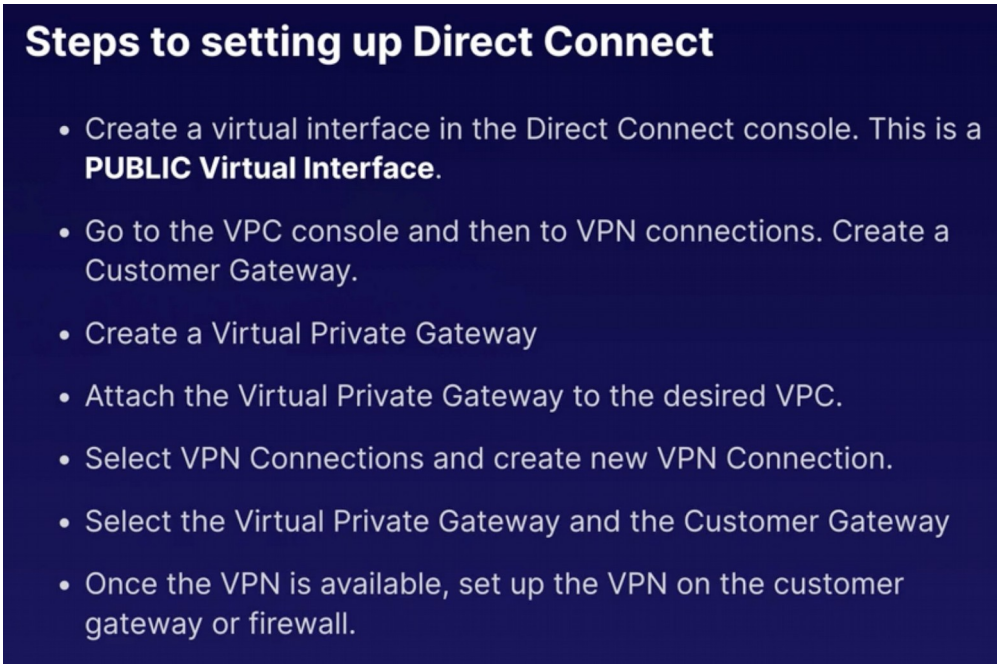


Exam Tip:

- Directly connects your datacenter to AWS
- Useful for high throughput workloads (ie lots of network traffic)

- Or if you need stable and reliable secure connection

Exam Tip: **Setting up Direct Connect:** **Video**



Global Accelerator:

What is it:
Service in which you create accelerators to improve availability and performance of your applications for local and global users.

Directs traffic to optimal endpoints over the aws global network. This improves the availability and performance of your internet applications that are used by a global audience.

By default GA provides you with two static IP addresses that you associate with your accelerator.

Alternatively, you can bring your own.

Easy explanation:
So instead of using internet and having to jump from one router to another owned by different providers until you reach the destination, with GA you connect to the nearest edge location and from there your connection will use the aws network, being much faster and efficient.

Global Accelerator vs CloudFront:
AWS Global Accelerator and Amazon CloudFront are separate services that use the AWS global network and its edge locations around the world.

CloudFront improves performance for both cacheable content (such as images and videos) and dynamic content (such as API acceleration and dynamic site delivery).

Global Accelerator improves performance for a wide range of applications over TCP or UDP by proxying packets at the edge to applications running in one or more AWS Regions.

Global Accelerator components:

- **Static IP addresses:** GA provides 2 IPs by default

- **Accelerator**: Directs traffic to optimal endpoints over the aws global network to improve the availability and performance of your internet applications. Each accelerator includes one or more listeners.
- **DNS Name:** GA assigns to each accelerator a DNS nam like: a12345678890abcdef.awsglobalaccelerator.com. That points to the IP addresses that ga assigns to you. Depending on the case you can use your accelerators IP or dns name to route traffic to your accelerator, or setup the dns records to route traffic using your own custom domain name.

- **Network Zone:** Services the static IP addresses for your accelerator from a unique IP subnet. Similar to an aws AZ, a network zone is an isolated unit with its own set of physical infrastructure.

  When you configure an accelerator , by default, GA allocates two IPv4 addresses for it. If one IP address from the network zone becomes unavailable due to its address blocking by certain client networks, or network disruption, client application can retry on the healthy static IP address from the other isolated network zone.

- **Listener:** Processes inbound connections from clients to GA, based on the port (or port range) and protocol you configure. Global accelerator supports both TCP and UDP protocols.
  Each listener has one or more endpoint group associated with it, and traffic is forwarded to endpoints in one of the groups.
  Your associate end point groups with listeners by specifying the Regions that you want to distribute traffic to. Traffic is distributed to optimal endpoints within the endpoint groups associated with a listener.

- **Endpoint Group:** Each endpoint group is associated with a speciffic AWS region. Enpoint groups include one or more endpoints in the region.

  You can increase or reduce the percentage of traffic that would be otherwise directed to an endpoint group by adjusting a setting called a traffic dial. The traffic dial lets you easily do performance testing or blue/green deployment testing for new releases across different regions.

- **Endpoint:** Can be network load balancers, application load balancers, ec2 instances, or elastic IP addresses.

  An application load balancer endpoint can be internet facing or internal. Traffic is routed to endpoints based on configuration options that you choose, such as endpoint weights.

  For each endpoint, you can configure weights, which are numbers that you can use to specify the production of traffic to route to each one. This can be useful, for example, to do performance testing within a region.

Exam Tip:

- Global accelerator is a service in which you create accelerators to improve availability and performance of your applications for local and global users.

- You are assigned two static IP addresses, or you can bring your own.
- You can control traffic using traffic dials. This is done within the endpoint group.

VPC Endpoints:

What is this:
You can connect your VPC privately with other aws services by using a private link.

Not necessary to use direct connect, internet gateway, nat device or vpn.

Instances in the VPC does NOT require public IPs to communicate with resources in the service.

Traffic between the vpc and the other services does NOT leave the aws network.

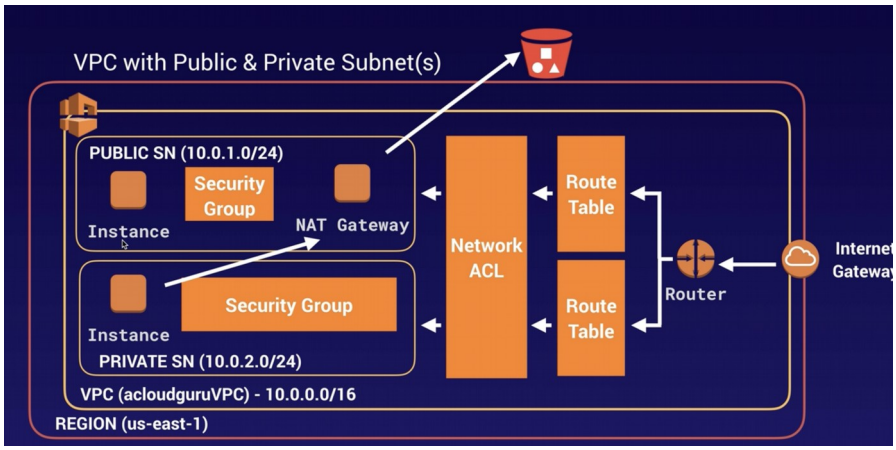
Virtual devices:
Endpoints are virtual devices. Horizontally scaled, redundant, and highly available vpc components. You can communicate your vpc with the aws services without imposing available risks or bandwidth constraints on your network traffic.

Types of Endpoints:

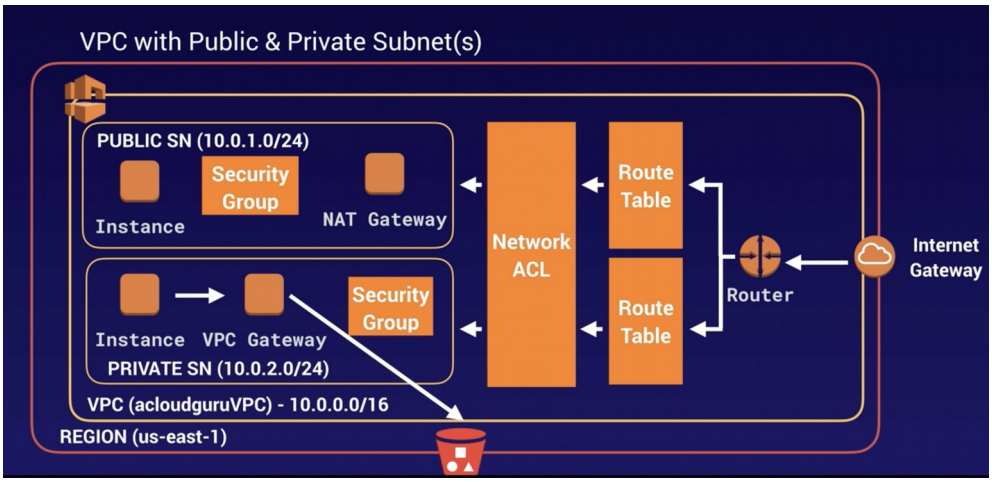
- **Interface Endpoints:** Is a Elastic Network Interface (ENI) with a private IP that serves as an entry point for traffic destined to a supported service. Basically attach an ENI to an ec2 instance and you will connect internally to the services that interface endpoint supports.

- **Gateway Endpoints:** Similar to interface endpoints. Currently supporting S3 and DynamoDB.

Before VPC Endpoint:



With VPC Endpoint:



How to configure it:

VPC – Endpoints – Create Endpoint – AWS Services – select S3 – select the route table – full access policy, create.

You will need to wait for a while until the new route is set up in the route table.

Now you can go to the ec2 instance in the private vpc, REMEMBER, you will need always to type the region where you are in, example: aws s3 ls --region us-east-2

Exam Tip:

- VPC endpoint: enables you to connect your private vpc to supported aws services.
- Powered by PrivateLink. Doesn't requires a VPN, Direct Connect, Internet Gateway or NAT device.
- There are two types:
  - Interface Endpoint
  - Gateway Endpoint: S3, DynamoDB

AWS PrivateLink:

Sharing Applications Across VPCs:
You can share applications between different VPCs by:

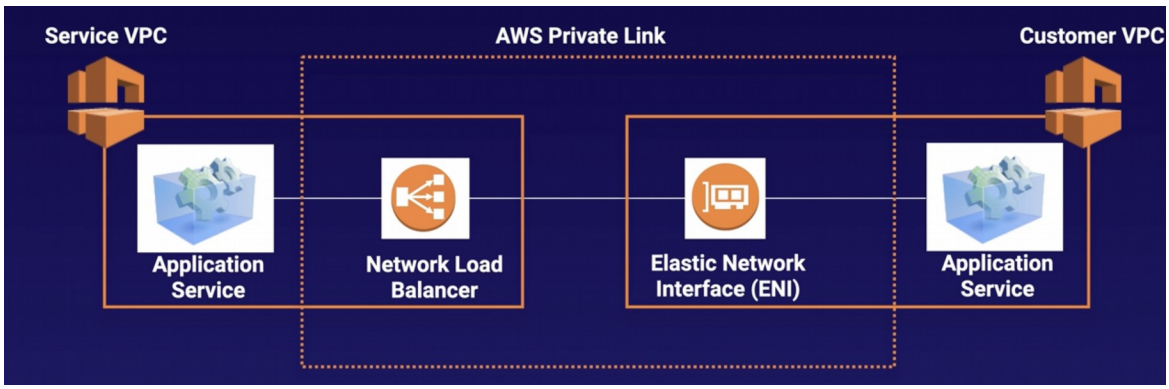
- **Open the VPC up to the internet:**
  - Security considerations, everything in the public subnet is public.
  - A lot more to manage.
- **Use VPC peering:**
  - You will have to create and manage many different peering relationships. The whole network will be accessible. This is not good if you have multiple applications within your vpc.

PrivateLink: To avoid the previous scenarios, aws invented PrivateLink.

This the best way to expose a service VPC to tens, hundreds or thousands of customer VPCs.

Doesn’t require VPC Peering, no route tables, NAT, IGW, etc.

Requires a Network Load Balancer on the service VPC and an ENI on the customer VPC.



Exam Tips:

- If you see a question asking about peering VPCs of tens, hundreds or thousands of customer VPC, think on AWS PrivateLink
- Doesn’t require VPC peering, no route tables, NAT, or IGW.
- Requires a Network Load Balancer on the service VPC and a ENI on the customer VPC.

AWS Transit Gateway:

Exam Tips:

- Allows you to have transitive peering between thousands of VPC and on-premises data center.
- Works on a hub-and-spoke model
- Works on a regional basis, but you can have it across multiple regions.
- You can use it across multiple AWS accounts using RAM (Resource Access Manager)
- You can use route tables to limit how VPC talk to one another.
- Works with Direct Connect as well as VPN connections.
- Supports IP multicast (not supported by any other aws service)

AWS VPN CloudHub:

Exam Tips:

- If you have multiple sites, each with own VPN connection, you can use AWS VPN CloudHub to connect those sites toghether.
- Hub-and-spoke model
- Low cost, easy to manage
- It operates over the public internet, but all traffic between the customer gateway and the AWS VPN CloudHub is encrypted.

AWS Network Costs:

Exam Tips:

- Use private IPs over public IPs to save on costs. This then utilizes the AWS backbone network.
- If you want to cut all network costs, group your ec2 instances in the same AZ and use private IPs. This will be cost free but have in mind that you will have a lot of single point of failure issues if the AZ goes down or something similar.

HA Architecture:

What is a Load Balancer:
Physical or virtual device that balances the network load across multiple webservers.

There are three types:

- **Application Load Balancer:**
  - Best suited for http / https traffic.
  - Layer 7 load balancer, application aware (means that, for example, can see if you changed the language in your browser to french, then will send you the french webservers). Can see inside the application.
  - Intelligent. You can create advanced request routing, sending specified requests to specific webservers.

- **Network Load Balancer:**
  - Best suited for load balancing of TCP traffic where extreme performance is required.
  - Layer 4 (connection level)
  - Capable of handling millions of request per second, while maintaining ultra-low latencies.
  - Static IP Address

- **Classic Load Balancer:**
  - Legacy LB.
  - Can load httl/https apps and use layer 7 specific features like X-Forwarded and sticky sessions, but is NOT application aware, doesn't doit at layer 7.
  - You can also use layer 4 load balancing for apps that rely in TCP.
  - Cheaper.

Errors in Load Balancing:

- **Classic Load Balancers:** If your app stops responding , the ELB (Classic) responds a 504 error (Gateway timeout). This means that the app is having issues. This could be either at the

webserver layer or at the DB layer. Identify where the app is failing and scale it up or out where possible.

X-Forwarded-For-Header:
Is a way to keep the public IP of the client (the user) that is connecting the app from his home.

When you want to access the application first you will connect to the load balancer.
Then the lb will connect with the proper ec2 by using the internal IP of the lb and the ec2.

Therefore for the application, the private IP of the load balancer will be the IP which the end user have, which is not true.

To have the public IP of the user the app have to look to the X-Forwarded-For-Header to get the public IP.

Exam Tips:

- Application Load Balancer (More intelligent, no Static IP, only domain name provided)
- Network Load Balancer (If you want performance, it have a Static IP)
- Classic Load Balancer (cheap, no Static IP, only domain name)
- 504 Error means the gateway has timed out. Means that the application is not responding within the idle timeout period.
  - Troubleshoot the application to see if the problem is with the webserver or the db.
- X-Forwarded-For: If you need the public IP of your end user look for the x-forwarded-for header.

Creating a Load Balancer:

Health check settings:
Classic:
Response Timeout: 2 → time to wait when a response from the heath check
Interval: 5 → amount of time between health checks
Unhealthy threshold: 2 → num. of consecutive health ch. failures before declaring ec2 unhelthy.
Healthy threshold: 3 → num. of consecu. health ch. success b4 declaring ec2 healthy.

Target group:
Healthy threshold: 2
Unhealthy threshold: 3
Timeout: 5
Interval: 6

Exam Tips:

- Instances monitored by ELB as:
  - InService
  - OutOfService
- Health checks check the instance health by talking to it.
- LB have their own DNS name. You are never given an IP address.
- Read ELB Faq for classic lb.

Advanced Load Balancers Theory:

Sticky Sessions:
Classic LB routes each request independently to the registered ec2 instance with the smallest load.

When you want to bind a user's session to a specific EC2 instance. This ensures that all requests from the user during the session are sent to the same instance.

You can enable sticky session to Application LB, but traffic will be sent at the Traffic Group Level rather than individual ec2 instances.

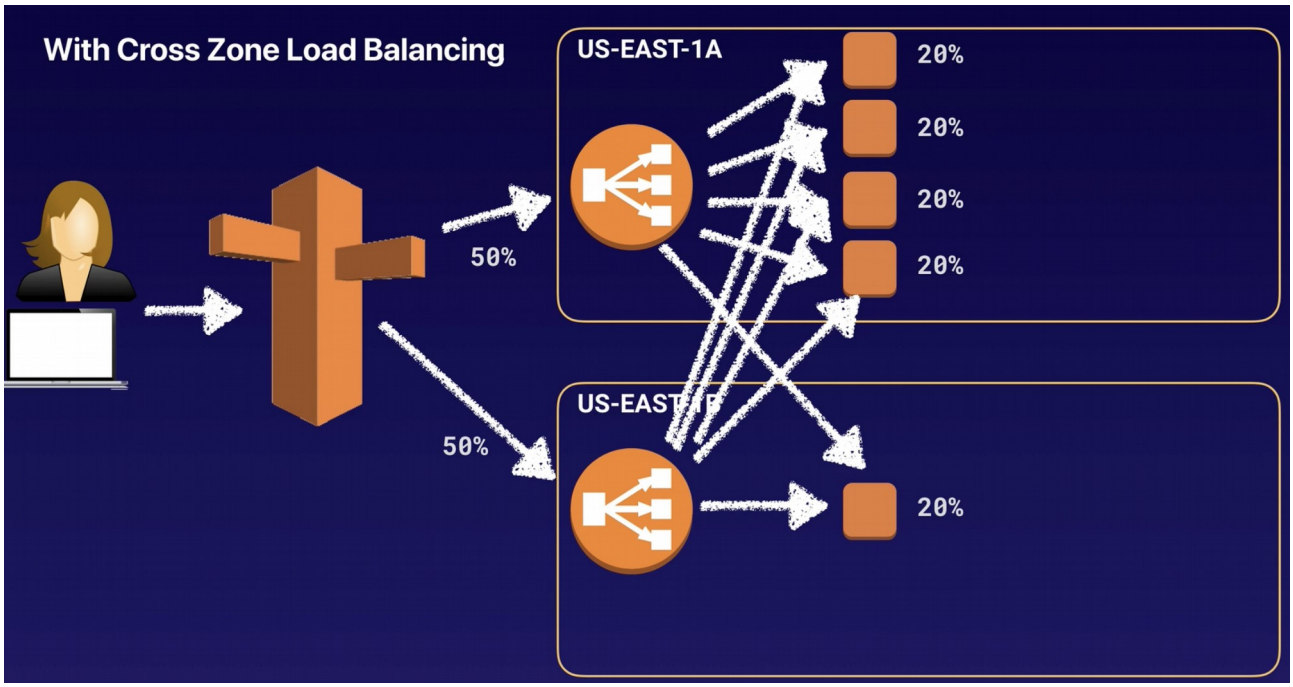
Example: When the application is saving something locally into the instance, like placing different bets into a bet slip. If you don't have the bets saved in the same instance you will lose them once the LB sends you to another instance.

Exam scenario: You have 2 instances in the same AZ. The users requests are going only to one of the instances. To fix this you disable sticky sessions.
The scenario could be testing wheter or not you are going to enable sticky sessions or dissable it.

If the scenario tells that the instance is writing to an ec2 instance local disck you need to enable sticky sessions.

Cross Zone Load Balancing:

If you have a route53 rule to split the 50% or traffic between 2 AZ and in one az (us-east-1b) you have only one instance, cross zone lb can send internally traffic from us-east-1b to 1a to avoid to overload the single instance in the 1b zone.



Exam scenario: You have ec2 instances in 2 AZ. You are using route 53 to send all the traffic to one of the AZ.

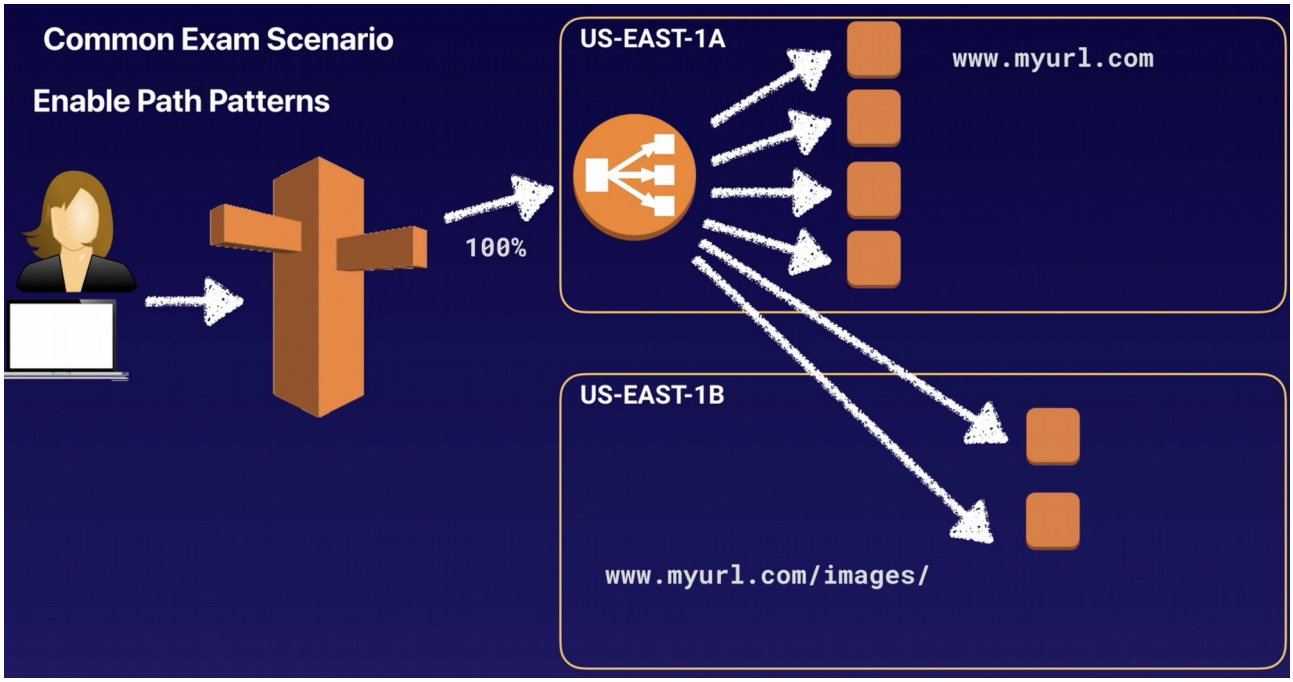
You should enable cross-zone load balancing. That way the instances in the AZ that don't receive traffic will receive it.

Path Patterns:

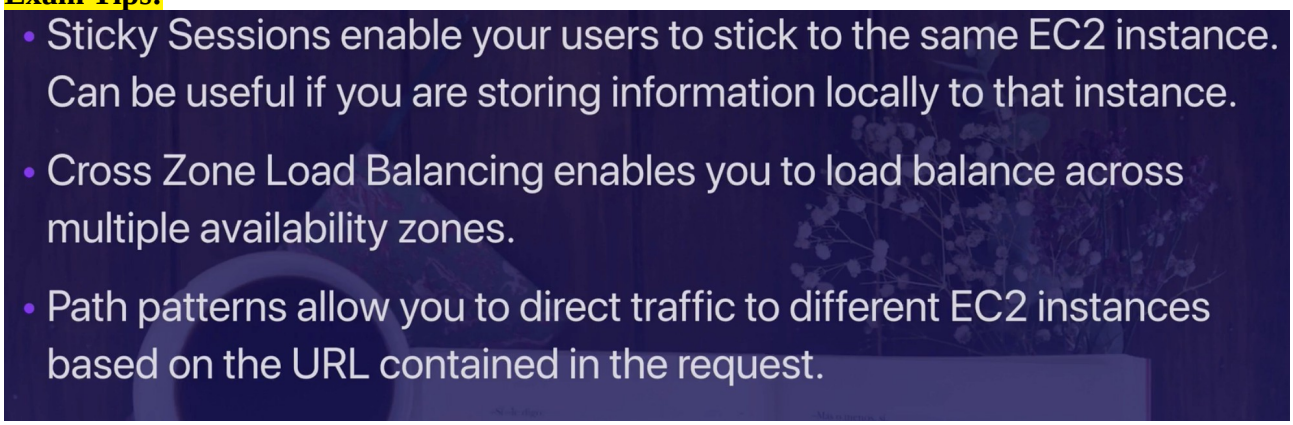
Short explanation:
You have instances in 2 AZ.
You send 100% of traffic to one az, to the url myurl.com
But you have the images and other media files in the instances in the other az.
To achieve this you can use Path Patterns to send all requests for images to the instances in the az b.

Exam scenario: EC2 instances in two AZ. You have Route 53 configured to send 100% traffic of www.myurl.com to one AZ. But you want to send www.myurl.com/**IMAGES/** to the instances on the other AZ.

To do that enable Path Patterns.



Exam Tips:



Auto Scaling:

Auto Scaling has 3 components:

1. **Groups**: Logical component where you are gonna put your instances in. Webserver group or application group, Database group, etc.

2. **Configuration Templates:** The instructions on what ec2 instances to launch, the size, instance type, memory, storage, AMI, SG, key pair, block device mapping.

3. **Scaling Options:** Provides several ways for you to scale your autoscaling groups. Dynamic Scaling: Configure it to scale based on the occurrence of specified condition. Scale based on schedule

Scaling Options:

1. Maintain current instance levels at all times.
2. Scale manually
3. Scale based on schedule
4. Scale based on demand
5. Use predictive scaling

Maintain current instance levels at all times:
You can configure Auto Scaling to maintain a specific number of running instances.

To maintain the current instance levels, auto scaling performs a periodic health check on running instances within an autoscaling group.

If one instance is NOT healthy, auto scaling terminates it and spins up a new one.

Scale manually:
You specify the change in the maximum, minimum or desired capacity of your auto scaling group.

Scale based on schedule:
Scaling actions are performed automatically as a function of time and date.
Useful when you know exactly when to increase or decrease the number of instances in your group.

Scale based on demand:
Using scaling policies. Let's you define parameters that control the scaling process.

Example: You have 2 instances and you want the CPU utilization of the autoscaling group to stay around 50% when the load on the application changes.

Predictive Scaling:
Is predicting based in your previous performance when you are going to need scaling options.

Exam Tips:

- Maintain current instance levels at all times.
- Scale manually
- Scale based on schedule
- Scale based on demand
- Use predictive scaling

Launch Configurations and Auto Scaling Groups:

Launch Configuration:
To create an Auto Scaling group first create a Launch Configuration.

Is quite similar to launch an EC2, asking you to select the AMI, the SG, the storage, key pair,etc.

Doing this does NOT do anything, is not launching nothing, just creating the configuration that you want to launch.

To apply this configuration you need to create the Scaling Group using that launch configuration.

Create an Autoscaling Group:
If you just created the launch configuration is gonna let you to create the autoscaling group straight away.

HA Architecture:

Plan for failure: Everything fails. Everything. You should always plan for failure.
Amazon's Cloud Monkey

Exam question: You need always 6 instances running and must be HA. You also must be able to recover from an AZ failure, which is the best HA architecture and the most cost effective.

The correct choice is 3 AZ with 3 instances each, 9 instances in total, because if one AZ fails you will have the 6 instances that you need anyway.

Exam Tips:

- Always design for failure
- Use Multiple AZ and multiple Regions where ever you can
- Know the difference between Multi-AZ and Read Replicas for RDS
  - Multi-AZ for disaster recovery and Read Replicas for performance
- Know the difference between Scaling out and Scaling up:
  - Scaling Out: Where you use Scaling Groups, we add additional instances
  - Scaling Up: Where we increase the resources inside an ec2 instance (scaling vertically)
- Read the question carefully and always consider the cost element
- Know the different S3 storage classes.

Building a Fault Tolerance WP Site:

You need to create:
2 S3 buckets
Cloudfront (will take a bit of time to create)
Security Groups (create webDMZ, pt 80 and 22 open and link it to the rds SG)
RDS → mysql (will take a bit of time to create)
IAM, create role for S3
EC2 instance (with terminal access)

[…] Lab not finished. It had several chapters.

Exam Tips:

- You can reboot a server to force it to balance to the other servers.
- CloudFormation:
  - Is a way to completely scripting your cloud environment
  - Quick Starts is a bunch of CloudFormation templates already built by AWS Solultions Architects allowing you to create complex environments very quicly.

Elastic Beanstalk:

You can get 1 or 2 questions about it in the exam.

How is different from CloudFormation:

- CloudFormation is very scripted. Is all using json, you can create masive templates and deploy resources at scale but is for advanced AWS users.

- Elastic Beanstalk is aimed for developers that have no idea about AWS that they want to provision a wordpress or joomla website or something like that and don't want to go and learn CloudFormation. They just want to press a few buttons and having the website deployed.

EB let's you to chose between a few popular environments (php, docker, python, tomcat, etc).
You just select the environment, upload your code (or get sample code) and creates everything by it self.

Exam Tips:

- You can quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. You simply upload your app, and EB automatically handles the details of capacity provisioning, load balancing, scaling, and app health monitoring.

High Availability with Bastion Hosts:

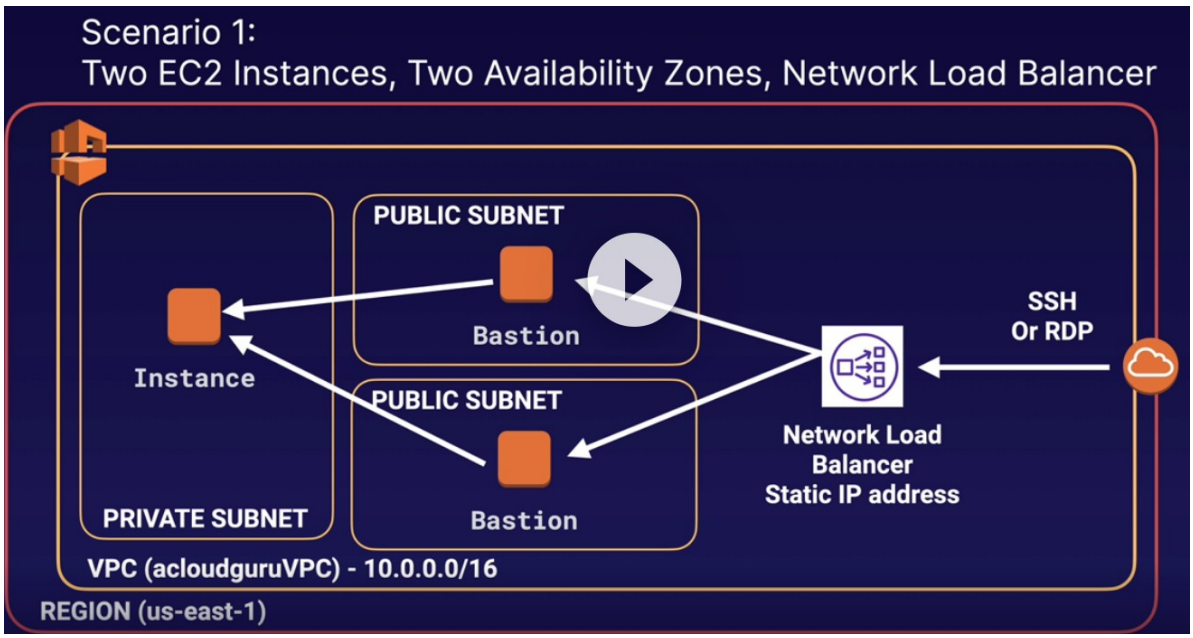
A bastion host is an instance in a public vpc that you are using to access instances in private vpc.

Is a jump instance. You can not access the private vpc from outside so you jump to the bastion host, who is in a public vpc and is allowed to access the private vpc.

You access via ssh or rdp.

There are two scenarios:

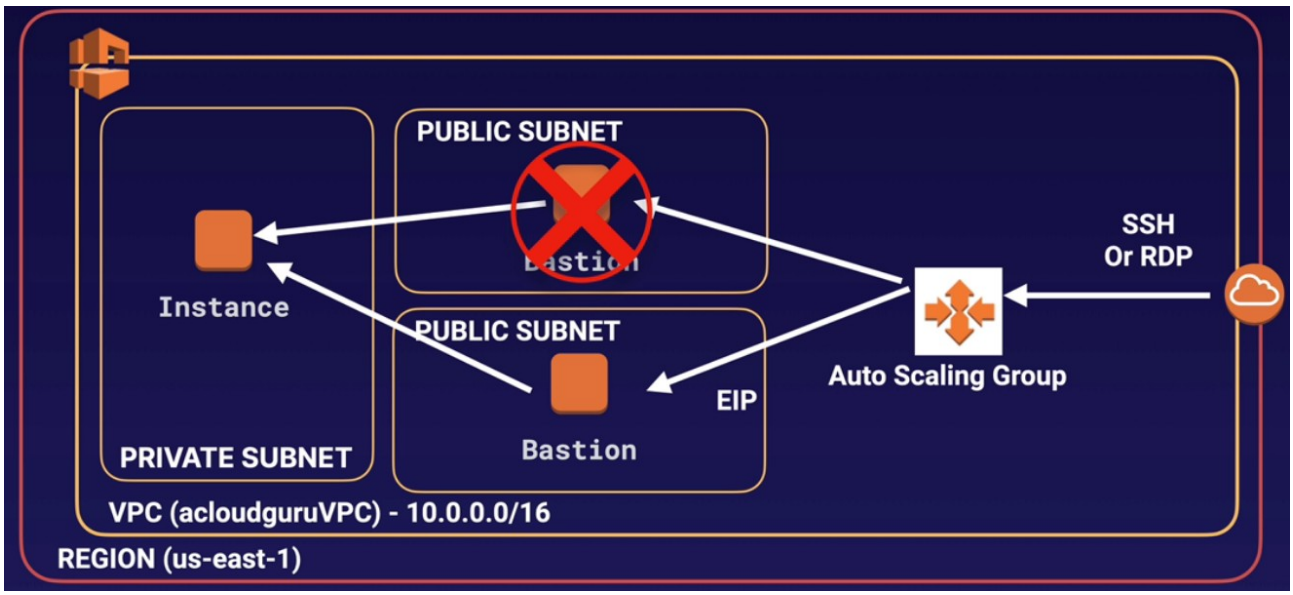
Scenario 1, the expensive one, using a Network Load Balancer because ssh or rdp works with layer 4. (Application load balancer won't work in this scenario)



Scenario 2, cheapest option for dev or test environments.
We have a private vpc with the instance in it, a public vpc (with an elastic ip address EIP) with the bastion and an Auto Scaling Group with minimum and maximum of 1.

If the bastion falls the ASG will provision another bastion in another subnet, you can use a user data script to take over the EIP and then you'll be able to ssh or rdp the new bastion.

You will have some downtime while the bastion is privisioned in another public subnet by the Auto Scaling Group and also incur a little more downtime depending on how long your health check takes to find that is unhealthy.



Exam Tips:

- Two hosts in two separate AZ. Use a Network Load Balancer with static IP and health checks to fail over from one host to the other.

- Can't use an Application Load Balancer, as it is layer 7 and you need to use layer 4.

- One host in one AZ behind an Auto Scaling Group with health checks and a fixed Elastic IP. If the host fails, the health check will fail and the ASG will provision a new EC2 instance in a separate AZ. You can use a user data script to provision the same EIP to the new host. This is the cheapest option, but is not 100% fault tolerant.

On-Premises Strategies with AWS

You need to be aware of what high-level AWS services you can use on-premises for the exam:

- Database Migration Services:
  - Allows you to move databases to and from AWS
  - Might have your DR environment in AWS and your on-premises environment as your primary.
  - Works with most popular database technologies, such as Oracle, Mysql, Dynamodb, etc.
  - Supports homogenous migrations, example: from Oracle to Oracle or heterogeneous migrations, example from sql server to aurora, for example.

- Server Migration Service:
  - SMS supports incremental replication of your on-premises servers in to AWS.
  - Can be used as a backup tool, multi-site strategy (on-premises and off-premises), and a DR tool.

- AWS Application Discovery Service:
  - Helps enterprise customers plan migration projects by gathering information about their on-premises data centers.

    - You install the AWS Application Discovery Agentless Connector as a virtual appliance on Vmware vCenter.
    - It will then build a server utilization map and dependency map of your on-prem environment.
    - The collected data is retained in encrypted format in an AWS Application Discovery Service data store. You can export this data as a CSV file and use it to estimate the Total Cost of Ownership (TCO) of running on AWS and to plan migration to AWS.
    - This data is also available in AWS Migration Hub, where you can migrate the discovered servers and track their progress as they get migrated to AWS.

- Using VM Import/Export:
    - Migrate existing applications in to EC2.
    - Can be used to create a DR strategy on AWS or use AWS as a second site.
    - You can also use it to export your AWS VMs to your on-prem data center.

- Download Amazon Linux 2 as an ISO:
    - Works with all major virtualization providers, such as Vmware, Hyper-V, KVM, VirtualBox, etc.

Exam Tips:

- Database Migration Services (DMS)
- Server Migration Services (SMS)
- AWS Application Discovery Service
- VM Import/Export
- Download Amazon Linux 2 as an ISO

Applications:

SQS: Is the oldest service in AWS.

Amazon SQS is a web service that gives you access to a message queue that can be used to store messages while waiting for a computer to process them.

It's a distributed queue system that enables web service applications to quickly and reliably queue messages that one component in the application generates to be consumed by another component.

A queue is a temporary repository for messages that are awaiting processing.

Using Amazon SQS, you can decouple the components of an application so they run independently, easy message management between components.

Any component of a distributed application can store messages in a fail-safe queue.

Messages can contain up to 256 KB of text in any format. Any component can later retrieve the messages programmatically using the Amazon SQS API.

The queue acts as a buffer between the component producing and saving data, and the component receiving the data for processing.

This means the queue resolves issues that arise if the producer is producing work faster than the consumer can process it, or if the producer or consumer are only intermittently connected to the network.

Types of queue:

- Standard queues (default)
- FIFO queues

Standard Queues: Offered as the default one by aws.
Let's you have a nearly-unlimited number of transactions per second. Standard queues guarantee that a message is delivered at least once.

Occassionally (because of the highly-distributed architecture that allows high throughput), more than one copy of a message might be delivered out of order.

However, starndard queues provide best-effor ordering witch ensures that messages are generally delivered in the same order as they are sent.

FIFO queues: First in First out
The order in which messages are sent and received is strictly preserverd and a message is delivered once and remains available until a consumer processes and deletes it; duplicates are not introduced into the queue.

FIFO also support message groups that allow multiple ordered message groups within a single queue.

FIFO are limited to 300 transactions per second (TPS), but have all the capabilities of standard queues.

Exam Tips:

- SQS is pull-based, not push-based. You have to have an EC2 instance pulling the messages out of the queue.
- Messages are 256 KB in size or less.
- Messages can be kept in the queue from 1 minute to 14 days; the default retention period is 4 days.
- SQS guarantees that your message will be processed at least once.
- SQS long polling is a way to retrieve messages from your Amazon SQS queues. While the regular short polling returns immediately (even if the message queue being polled is empty), long polling doesn't return a reponse until a message arrives in the message queue, or the long poll times out.
- Any time you see a scenario based question about decoupling your infrastructure - think SQS.
- Visibilty timeout: Amount of time that a message is invisible in the SQS queue after a reader picks up that message.

  For instance you have a message in the queue and the EC2 instance picks it up, the visibility timeout is the length of time that the message is gonna be invisible in the queue until the EC2 instance processes the job.

  Once the job is processed by the ec2 instance the message is deleted from the queue.

  If the job is not processed within that time, the message will become visible agian and another reader will process it. This could result in the same message being delivered twice.

  To avoid that you can increase the visibility timeout.

  Visibility timeout maximum is 12 hours.

Simple Workflow Service - SWF

Is a way of combining your digital environment (code, apps, etc) along with manual tasks (human beings).

Example: The amazon.com web. You go to the web for a product, you purchase it by paying with your card. Until now all has been managed by code, algorithms, applications, etc.

But then a human being in the amazon warehouse have to pick your order, pack your product, add a label and post it to you. This tasks are managed by human beings.

Exam Tips:

- SWF vs SQS:
  - SQS has retencion period of up to 14 days; with SWF, workflow executions can last up to 1 year.
  - SWF presents a task-oriented API, whereas SQS offers a message-oriented API.
  - SWF ensures that a task is assigned onlyu once and is never duplicated. With SQS, you need to handle duplicated messages and may also need to ensure that a message is processed only once.
  - SWF keeps track of all the tasks and events in an application. With SQS, you need to implement your own application-level tracking, especially if your application uses multiple queues.
- SWF Actors: Workflow Starters: An application that can initiate (start) a workflow. Could be your e-comerce website following the placement of an order, or a mobile app searching for bus times.
- Deciders: Control the flow of activity tasks in a workflow execution. If something has finished (or failed) in an workflow, a Decider decides what to do next.
- Activity Workers: Carry out the activity tasks.

Simple Notification Service, SNS:

What is SNS: Web service that makes it easy to set up, operate, and send notifictions from the cloud.

It provides developers with a highly scalable, flexible, and cost-effective capability to publish messages from an application and immediately deliver them to subscribers or other applications.

Push Notifications: Allows you to do push notifications to Apple,Google, Fire OS, and Windows devices, as well as Android devices in China with Baidu Cloud Push.

SQS Integration: Besides pushing cloud notifications directly to mobile devices, SNS can also deliver notifications by SMS text message or email to Simple Queue Service (SQS), or to any HTTP endpoint.s

Topics: SNS allows you to group multiple recipients using topics. A topic is an “access point” for allowing recipients to dynamically subscribe for identical copies of the same notification.

One topic can support deliveries to multiple enpoint types, for example, you can group together IOS, Android and SMS recipients.

When you publish once to a topic, SNS delivers appropiately formatted copies of your message to each subscriber.

SNS Availability: To prevent messages from being lost, all messages published to SNS are stored redundantly across multiple availability zones.

Exam Tips:

- SNS Benefits:
  - Instantaneous, push-based delivery (no polling)
  - Simple APIs and easy integration with applications
  - Flexible message delivery over multiple transport protocols
  - Inexpensive, pay-as-you-go model with no up-front costs
  - Web-based AWS management Console offers the simplicity of a point-and-click interface.

- SNS vs SQS:
  - Both messaging systems in AWS
  - SNS is push
  - SQS polls (pulls)

Elastic Transcoder:

What is Elastic Transcoder: Media transcoder in the cloud.

A way to convert media files from their original source format in to different formats that will play on smartphones, tablets, PCs, etc.

Provides transcoding presets for popular output formats, which means that you don’t need to guess about which settings work best on particular devices.

You pay based on the minutes that you transcode and the resolution at which you transcode.

Exam Tips:

- Elastic transcoder is a media transcoder (converter) in cloud. It converts media files from original source format in to different formats that will play on smartphones, tablets, PCs, etc.

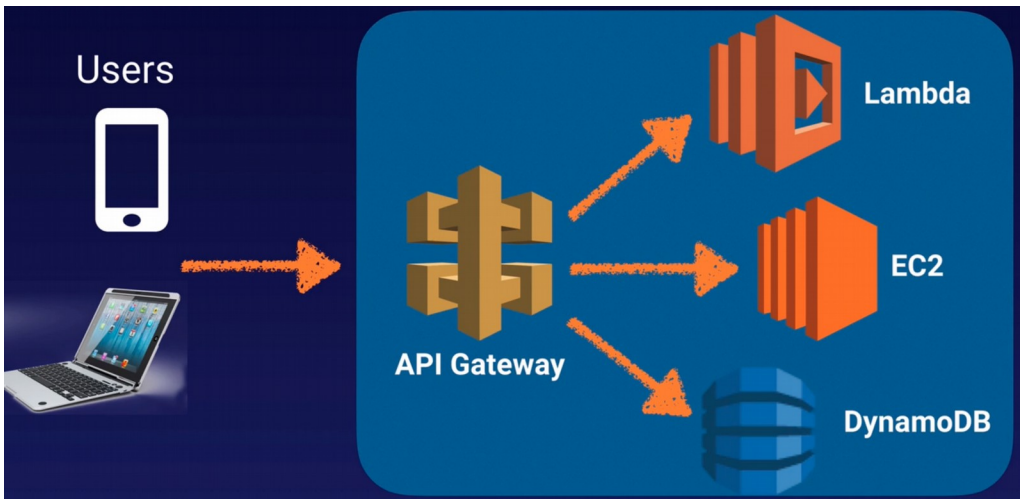
API Gateway **5 or 10 questions will come in the exam**

What is API Gateway: Fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale.

Is like a front door, like a load balancer in some regards in that its going to distribute traffic across multiple sources but you don't need EC2 instances behind it.

API Gateway is basically a doorway into your AWS environment and typically you'd use it to communicate to Lambda functions. You can use it to communicate to EC2 and other things like DynamoDB.

How it works:



The users do a call to our API Gateway.
The API Gateway is basically the front end or the front door to our whole AWS environment.
Depending on the call you can pass through Lamda, ec2 instance, dynamodb.

API Gateway options:

- Exposes HTTPS enpoints to define a RESTful API
- Serverless-ly connect to services like Lamda & DynamoDB
- Send efficiently with low cost
- Scale effortlessly
- Track and control usage by API key
- Throttle requests to prevent attacks
- Connect to CloudWatch to log all requests for monitoring
- Maintain multiple versions of your API

How to configure it:

- Define an API (container)
- Define Resources and nested Resources (URL paths)
- For each Resource:
  - Select supported HTTP methods (verbs)
  - Set security

- Choose target (such as EC2, Lambda, DynamoDB, etc)
- Set request and response transformations.

How to deploy it:

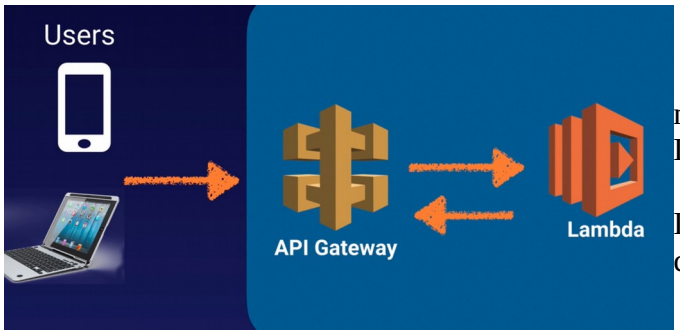
- Uses API Gateway domain, by default
- Can use custom domain
- Now supports AWS Certificate Manager: free SSL/TLS certs

<u>To understand for the exam:</u>
API Catching: You can enable API caching in API Gateway to cache your endpoint's response.

Whith caching, you can reduce the number of calls made to your endpoint and also improve the latency of the request to your API.

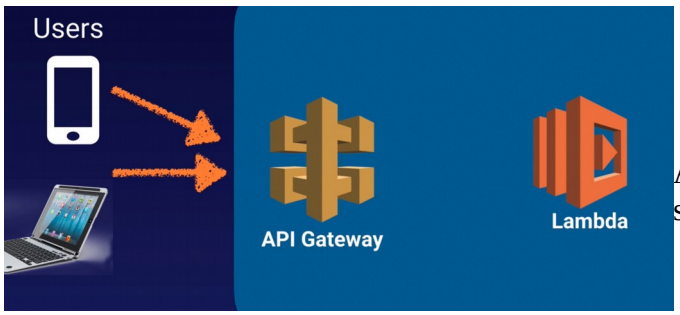
When you enable caching for a stage, API Gateway caches responses from your endpoint for a specified time-to-live (TTL) period, in seconds.

API Gateway then responds to the request by looking up the endpoint reponse from the cache instead of making a request to your endpoint.



User 1 makes request to our api gateway, maybe its a GET request that it sends that on to Lamda.

Lambda goes ahead and gets a response, that is cached in the api gateway



Now a second user makes the same request.

Api gateway have that response already cached so doesn't go to Lamda.

Same Origin Policy: In computing, the same-origin policy is an important concept in the web application security model.

Under the policy, a web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin.

This is done to prevent Cross-Site-Scripting (XSS) attacks.

- Enforced by web browsers
- Ignored by tools like PostMan and curl.

CORS: Is one way the server at the other end (not the client code in the browser) canrelax the same-origin policy.

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources (e.g. fonts) on a web page to be requested from another domain outside the domain from which the first resource was served.

CORS is a way of a web in one domain being able to talk to a web page in another domain.

How it works:

- Browser makes an HTTP OPTIONS call for a URL (OPTIONS is an HTTP method like GET, PULL, and POST)
- Server returns a response that says: “These other domains are approved to GET this URL”
- Error - “Origin policy cannot be read at the remote resource?” You need to enable CORS on API Gateway.

Exam Tips:

- Remember what API Gateway is at a high level
- API Gateway has caching capabilities to increase performance
- API Gateway is low cost and scales automatically
- You can throttle API Gateway to prevent attacks
- You can log results to CloudWatch
- If you are using Javascript/AJAX that uses multiple domains with API Gateway, ensure that you have enabled CORS on API Gateway
- CORS is enforced by the client.

Kinesis: (Pronunced kenises)

What Streaming data is:
Is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of kilobytes.)

- Purchases from online store (like amazon.com)
- Stock Prices
- Gaming data (as the game plays)
- Social network data
- Geospacial data (like uber.com)
- iOT sensor data

What is Kinesis: Is a platform on aws to send your streaming data to.

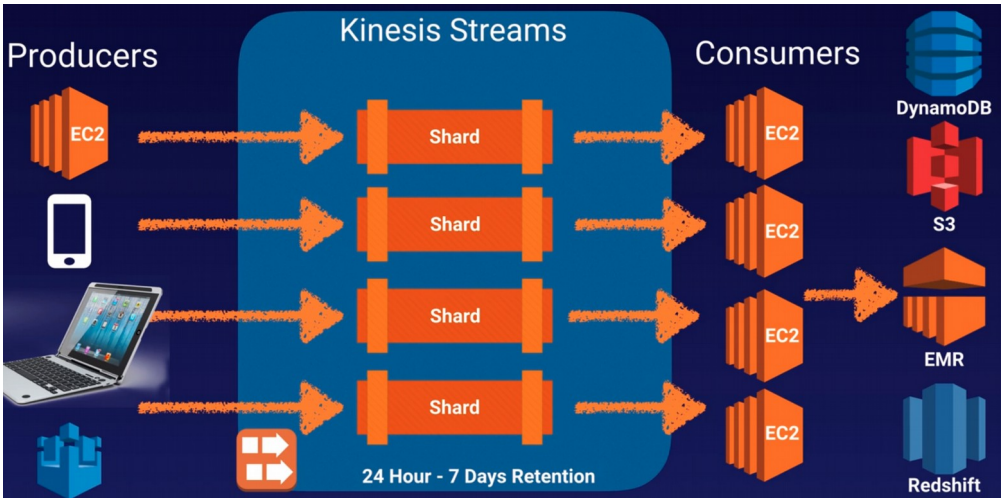
Kinesis makes it easy to load and analyze streaming data, and also providing the ability for you to build your own custom applications for your business needs.

Type of Kinesis:
There are 3 different types of Kinesis, and you need to know all of them and what are they used for.

- Kinesis Streams
- Kinesis Firehose
- Kinesis Analytics

Kinesis Streams: Allows you to persistently store your data from 24 hours to 7 days while your data consumers go through and do something with that data.

Once the data consumers analyzed that data they might store the analytics somewhere in like dynamodb, or s3, or EMR, or Redshift, etc.

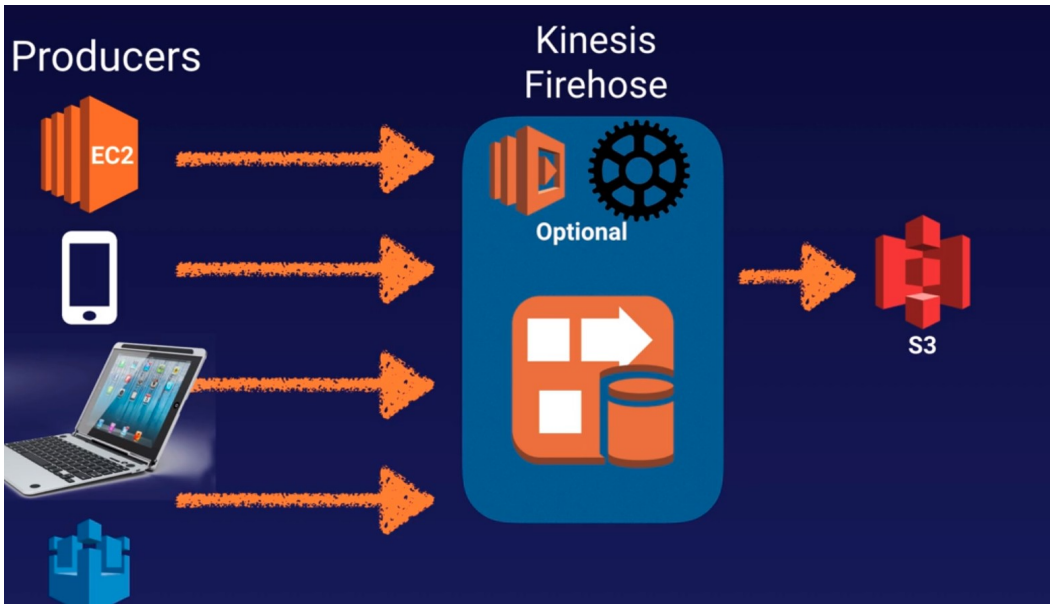


Kinesis streams consist of Shards:
When you put your data into kinesis streams is all gonna be stored in individual shards.

A shard accepts 5 transactions per second for reads, up to a maximum total data read rate of 2 MB per second and up to 1000 records per second for writes, up to a maximum total data write rate of 1MB per second (including partition keys)

The data capacity of your stream is a function of the number of shards that you specify for the stream. The total capacity of the steram is the sum of the capacities of its shards.

Kinesis Firehose: There is no data persistence. You have to do something with the data as soon as it comes in.

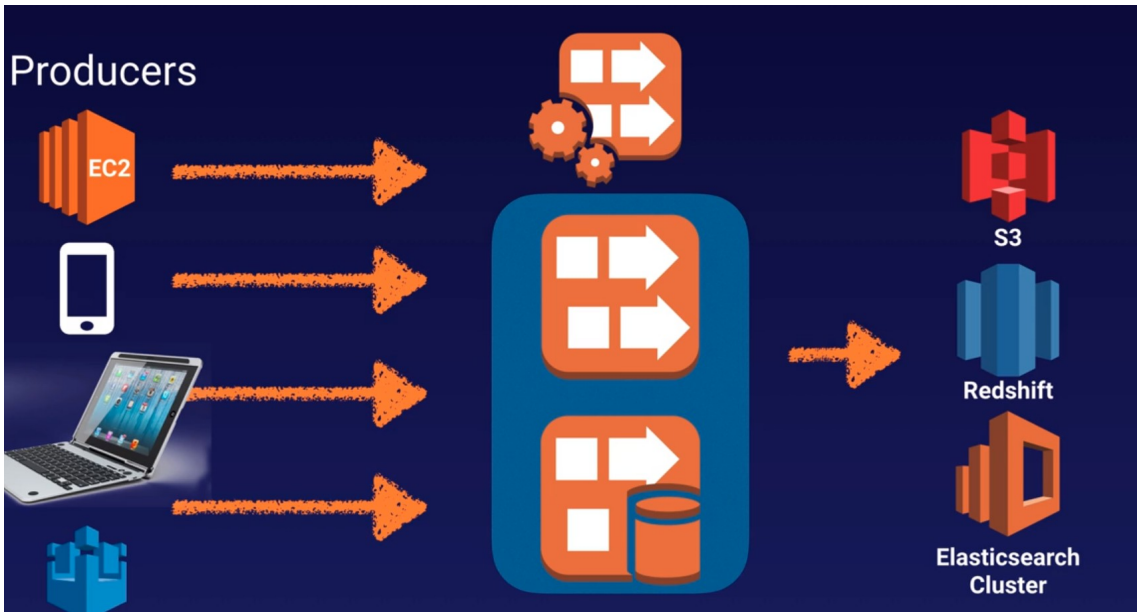


When the producers send data to kinesis firehose there is no persistent storage, the data have to be analized as it comes in. Is optional to have a lambda function inside the kinesis firehose.

As soon as the data comes in, it triggers a lambda function, the lambda function runs a particular set of code and outputs somewhere safe, like s3, elastic search cluster, etc.

Kinesis Analytics: Works with kinesis streams and kinesis firehose and can analyze the data on the fly, inside either service.

And then it goes in and stores this data either on s3, redshift, or elastic search cluster.



Exam Tips:

- Know the differences between Kinesis Streams and Kinesis Firehose. You will be given scenario questions and you must choose the most relevant service.
  - Kinesis Streams: to persistently store data (24 hours - 7 days)
  - Kinesis Firehose: NO persistent data, have to be analized straight away.
- Understand what Kinesis Analytincs is.
  - Analyses the data inside both kinesis streams and firehose.

Web Identity Federation / Amazon Cognito:

What is it: Lets you give your user access to AWS resources after they have successfully authenticated with a web-based identity provider like Amazon, Facebook, or Google.

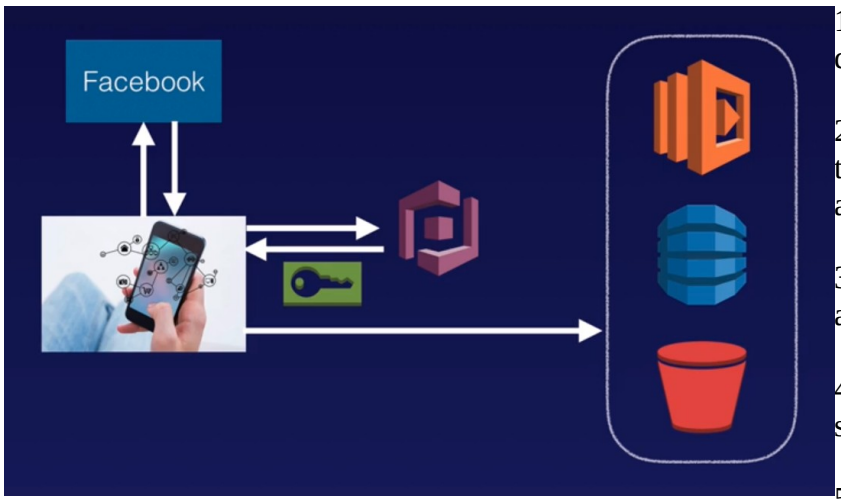
Following successful authentication, the user receives an authentication code from the Web ID provider, which they can trade for temporary AWS security credentials.

Short explanation: Web identity federation is when you try to sign in to a website using your Facebook credentials, google credentials, your amazon credentials, etc.

Amazon Cognito: Provides web identity federation service.

- Sign-up and sign-in to your apps
- Access for guests users
- Acts as an Identity Broker between your application and Web ID providers, so you don’t need to write any additional code.
- Synchronizes user data for multiple devices (if you change your email,phone, etc, will be replicated in the other devices)
- Recommended for all mobile applications aws services.

Use cases: The recommended approach for web identity federation using social media accounts like facebook.



1 We have a website with lambda dynamo and s3

2 Our user (via phone) what to use those resources. First thing to do is authenticate with facebook.

3 Facebook gives the user the authentication token.

4 Then the authentication token is sent to Cognito.

5 Cognito will respond and grant access to your aws environment, depending on what level of access you permit.

6 Then you will be able to do things like execute lambda functions, store data in dynanodb tables or store images on s3, etc.

Cognito brokers between the app and facebook or google to provide temporary credentials which map to an IAM role allowing access to the required resources.

No need for the application to embed or store aws credentials locally on the device and it gives users a seamless experience across all mobile devices.

Cognito User Pools:
User Pools are user directories used to manage sing up and sing in functionalyty for mobile and web applications.

User can sign in directly to the user pool, or using facebook, amazon or google.

Cognito acts as an Identity Broker between the identity provider and AWS.

Successfully authentication generates a JSON Web token (JWTs)

User Pools vs Identity Pools:

- Identity Pools: Enable provide temporary AWS credentials to access aws services like s3 or dynamodb. Is the actual granting of your access to the aws resource.
- User Pools: Things like your email address, your password, etc.

Event Processing Patterns:

Event driven applications: Many solutions architects build event driven applications where one or more aws services will automatically perform work in response triggered by other aws services.

This architectural pattern can make services more reusable, interoperable, and scalable.

Event-Driven Architecture: Publish/Subscribe (Pub/Sub) Messaging
In modern cloud architecture applications are decoupled into smaller, independent building blocks that are easier to develop, deploy, and maintain.

Publish/Subscribe or Pub/Sub Messaging provides instant event notifications for these kind of distributed applications.

The pub/sub model allows messages to be broadcast to different parts of the system asynchronously.

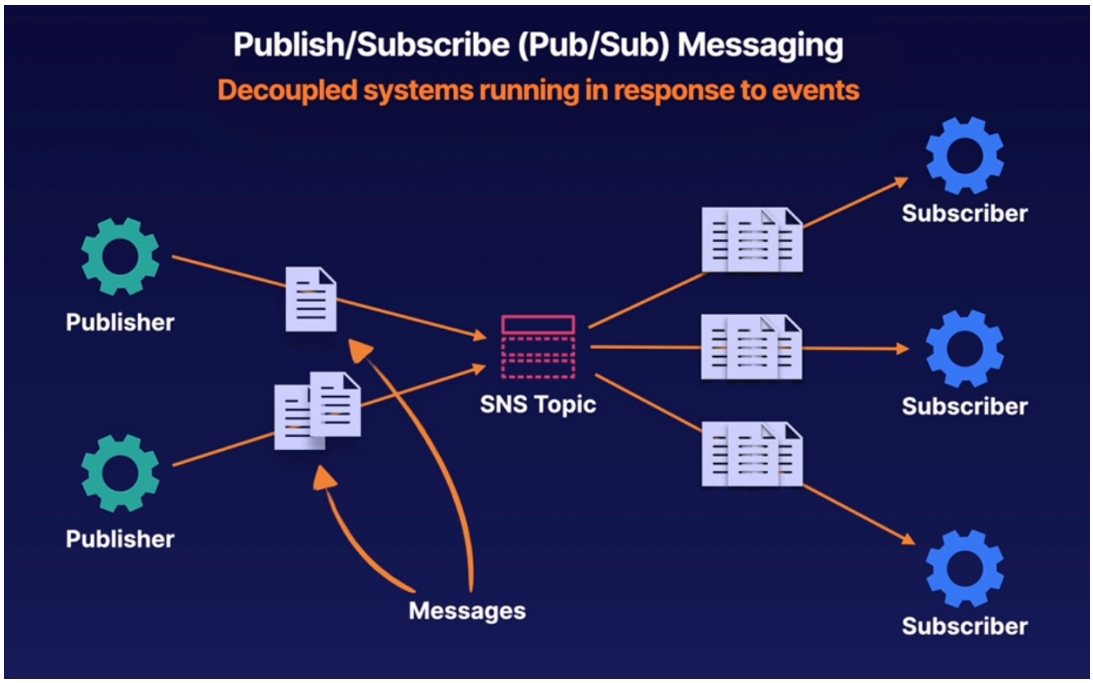
Central to this architecture is an SNS message topic.

This provides a mechanism to broadcast asynchronous event notifications and endpoints that allow other AWS services to connect to the topic in order to send and receive those messages

To broadcast a message, a component called Publisher simply pushes a message to the topic.

The Publisher can be your own application or one of many AWS services that can publish messages to SNS topics.

All services that subscribed to the topic will instantly receive every message that is broadcast.



The subscribers to the message topic ofter perform different functions and can each do something different with the message in parallel.

The publisher doesn’t need to know who’s using the information that is broadcasting and the subscriber don’t need to know who the message comes from.

Dead-Lettet Queues (DLQ):

This is analogous to something called Dead Letter Office which is a facility within a postal system where undeliverable mail is processed.

Sometimes messages are inspected for potential redelivery,or sometimes it'll just stay there forever. Unclaimed.

We have 3 services in AWS that use DLQ:

- **SNS:** Messages published to a topic that fail to deliver are sent to an SQS queu; held for further analysis or reprocessing.

- **SQS:** Messages sent to SQS that exceed the queue’s maxReceiveCount are sent to a DLQ (another SQS queue).

  For example, when a source queue has a redrive policy with max receive count of say five, and the consumer of the source queue receives a message six times without ever deleting it, SQS will move the message to the dead letter queue.

- **Lambda Functions:** Result from failed asynchronous invocations; will retry twice and send to either an SQS queue or SNS topic.

  In Lambda, dead letter queues store the messages that resulted in failed asynchronous executions of your Lambda function and execution can result in error for several reasons. Your code might raise an exception, timeout or run out of memory.

  The runtime executing your code might encounter an error and just stop.
  Your function might hit its concurrency limit and be throttled.

  Regardless of the archetype, when this kind of error occurs,your code might have run completely partially or not at all.

  If the function returns an error Lambda attempts to run it two more times before sending the message to the dead letter queue. So in other words,messages in the dead letter queue mean your normal error handling and retries have all failed.

How DLQs work: Begginer usse case for learning about serverless applications on AWS.

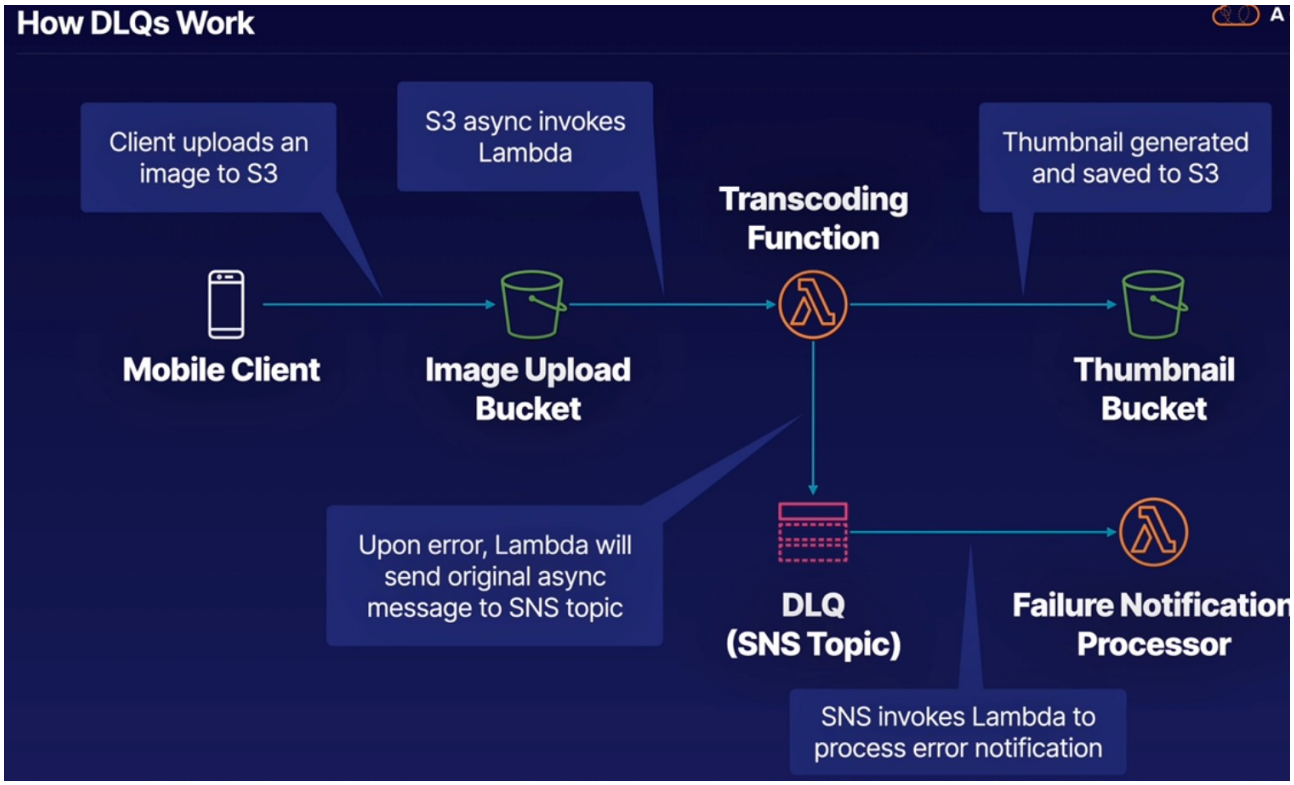
Creating thumbnails from images dropped into an S3 bucket.

Let's say we have a mobile client and our client application uploads an image to an S3 bucket. In response to that image, upload S3 will asynchronously invoke a Lambda function.

This transcoding function will generate a thumbnail sized image and save that to S3. This transcoding function can also be configured to send any transcoding failures to an SNS topic.

The SNS topic will receive the original asynchronous message sent from S3 to the Lambda function.
SNS will invoke another Lambda function to process any kind of error notifications from here.

You could perform some further investigation as to the cause of the failure.

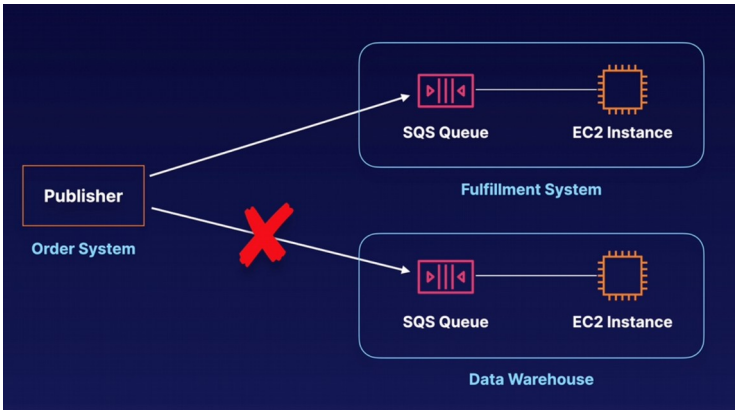


Fanout Pattern:

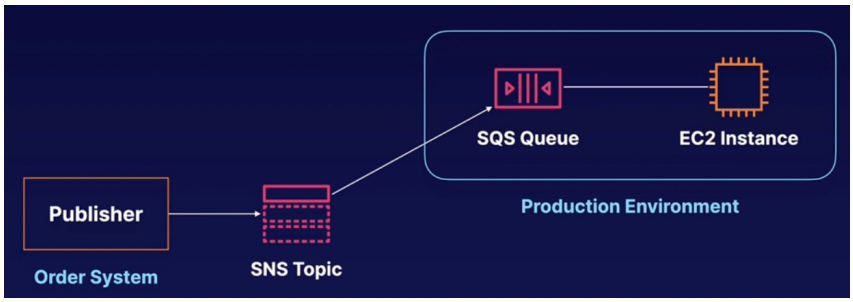
Example: Sample e-commerce application where we have an order system which needs to send messages to both a fulfillment system and a data warehouse.

We can send a message to SQS queue, number one in the fulfillment system,and another message to SQS queue. Number two in the data warehouse,and that can work, but it's not very reliable.

What if our order system crashes or loses network connectivity before sending a message to queue number two,our data warehouse won't receive that message and we can't have that.

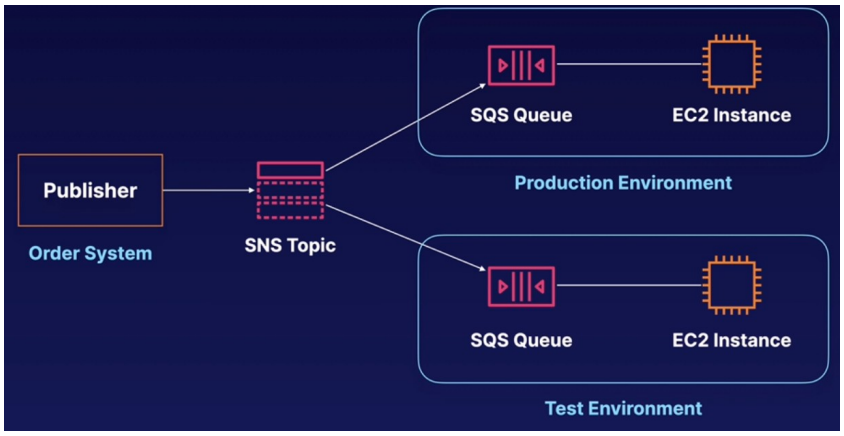


Rather than communicating directly with the SQS queues,the fan out pattern is when a publisher sends an SNS message first to a topic, and that message is replicated and pushed to multiple SQS queues.



The SQS queues that are subscribed to that topic would receive identical notifications for the new order.

Another way to use this Fanout pattern is to replicate data sent to your production environment. With your test environment,you can subscribe another queue to the same topic for new incoming orders.



Then by attaching this new queue to your test environment,you could continue to improve and test your application using the same data sent to your production environment.

S3 Event Notifications:

Let's say you have an S3 bucket and you want to initiate processing on new objects as they arrive, or maybe you want to capture information about the object and log it for tracking or security purposes.

The S3 event notification feature enables you to receive notifications when certain events happen in your bucket, notifications can be delivered to SQS queues, SNS topics, or Lambda functions.

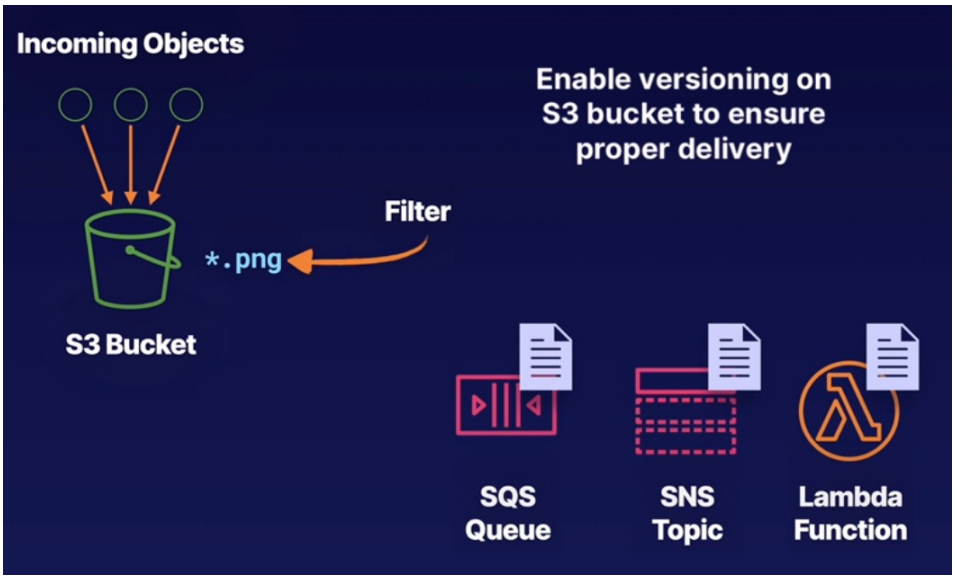
So for example, if objects are uploaded to a bucket, a message will be sent to multiple subscribers or destination simultaneously.

You can also filter what objects you want to receive notifications about.

For example, you can limit notifications to just PNG files, for example, by specifying a wild card. Now in practice these event notifications, aren't perfect.

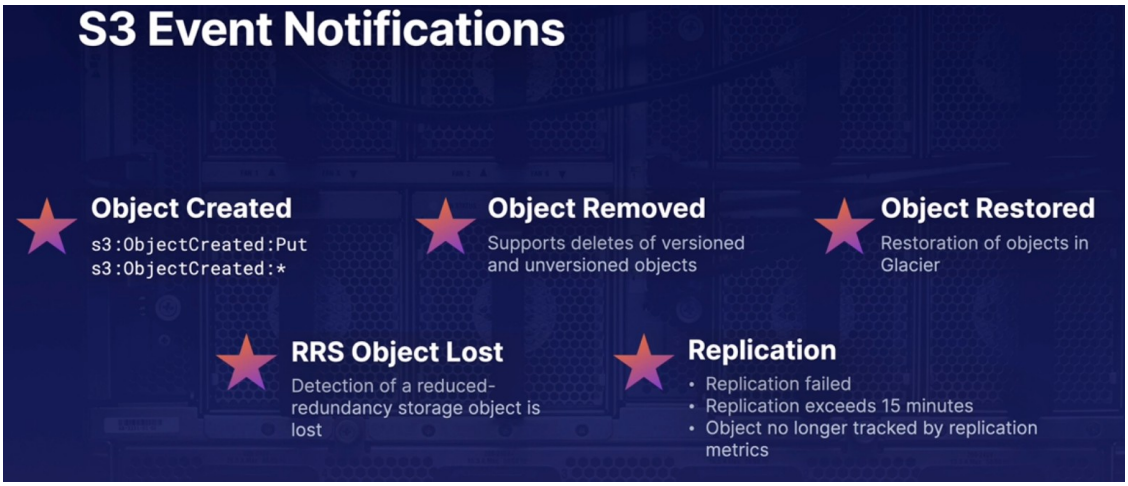
Sometimes you'll miss a notification here or there to get around this.

You'll want to enable versioning to ensure that you get a notification for every successful.



Right now, there are a number of different S3 events that can send event notifications:

- First are the Object created events. These are events like the S3 put object event.
- Object removed events. This supports deletes of versioned and, and versioned objects.
- Object restored: Notification when objects are restored from Glacier.
- RRS Object lost: Notifications of when objects from reduced redundancy, storage are lost and finally, we can get replication event notifications.
- Replication: We can be notified when replication fails when replication exceeds 15 minutes, or when we have an object that's no longer tracked by replication metrics.



Exam Tips:

- Understand the pub/sub pattern - facilitated by SNS
- You need to know which aws services support DLQ. SNS, SQS, Lambda
- Understand the Fanout Pattern and SNS his role in it.
- How S3 notifications work. Which events trigger; which services consume.

Scaling EC2 Using SQS exercise:

Script to generate messages in an SQS queue:

```
#!/usr/bin/env python3

import argparse
import logging
import sys
import uuid
from time import sleep

import boto3
from botocore.exceptions import ClientError

parser = argparse.ArgumentParser()
parser.add_argument("--queue-name", "-q", default="Messages", help="SQS queue name")
parser.add_argument("--interval", "-i", default=0.1, help="timer interval", type=float)
parser.add_argument("--message", "-m", help="message to send")
parser.add_argument("--log", "-l", default="INFO", help="logging level")
args = parser.parse_args()

if args.log:
    logging.basicConfig(format="[%(levelname)s] %(message)s", level=args.log)
else:
    parser.print_help(sys.stderr)

sqs = boto3.client("sqs")

try:
    logging.info(f"Getting queue URL for queue: {args.queue_name}")
    response = sqs.get_queue_url(QueueName=args.queue_name)
except ClientError as e:
    logging.error(e)
    exit(1)

queue_url = response["QueueUrl"]

logging.info(f"Queue URL: {queue_url}")

while True:
    try:
        message = str(uuid.uuid4())
        logging.info("Sending message: " + message)
        response = sqs.send_message(QueueUrl=queue_url, MessageBody=message)
        logging.info("MessageId: " + response["MessageId"])
        sleep(args.interval)
    except ClientError as e:
        logging.error(e)
        exit(1)
```

Script to process the messages sent by the previous script:

```
#!/usr/bin/env python3

import logging
import time

import boto3
from botocore.exceptions import ClientError

QUEUE_NAME = "Messages"

logging.basicConfig(format="[%(levelname)s] %(message)s", level="INFO")
sqs = boto3.client("sqs")

try:
    logging.info(f"Getting queue URL for queue: {QUEUE_NAME}")
    response = sqs.get_queue_url(QueueName=QUEUE_NAME)
except ClientError as e:
    logging.error(e)
    exit(1)

queue_url = response["QueueUrl"]
logging.info(f"Queue URL: {queue_url}")

logging.info("Receiving messages from queue...")

while True:
    messages = sqs.receive_message(QueueUrl=queue_url, MaxNumberOfMessages=10)
    if "Messages" in messages:
        for message in messages["Messages"]:
            logging.info(f"Message body: {message['Body']}")
            time.sleep(1)  # simulate work
            sqs.delete_message(
                QueueUrl=queue_url, ReceiptHandle=message["ReceiptHandle"]
            )
        else:
            logging.info("Queue is now empty")
```

Security:

Reducing Security Threats:

Bad actors:

- Typically automated processes
- Content scrapers
- Bad bots
- Fake user agent
- Denial of service (DoS)

Benefits of preventing bad actors:

- Reduce security threats
- Lower overall costs

What can you use:

- Network Access Control List (NACL): to deny all traffic from a certain IP for example.
- Host based firewal, firewalld, iptables, ufw, on linux, Windows firewall.

Application Load Balancer (ALB): Here the bad actor's incoming connection with terminate at the alb it self, so the EC2 instances behind the alb will be completelly unaware of the bad actor IP, a host based firewall will be ineffective in this case.

One additional measure you can take is to allow only the ALB security group access to the EC2 security group. However this won't completelly block the traffic from the ALB originated from the bad actor. We still have to use a NACL in this case.

With ALB you can use a Web Application Firewall (WAF) to block IPs, know attacks like XSS, etc.

Another scenario: Using Network Load Balancer (NLB)
The traffic doesn't terminate at the NLB. It passes through it directly to your EC2 instance, the client IP, the IP of that bad actor is visible from end to end.

You can use a firewall block on the ec2, but is better to block at the NACL. A WAF (web application firewall) could be possible as well.

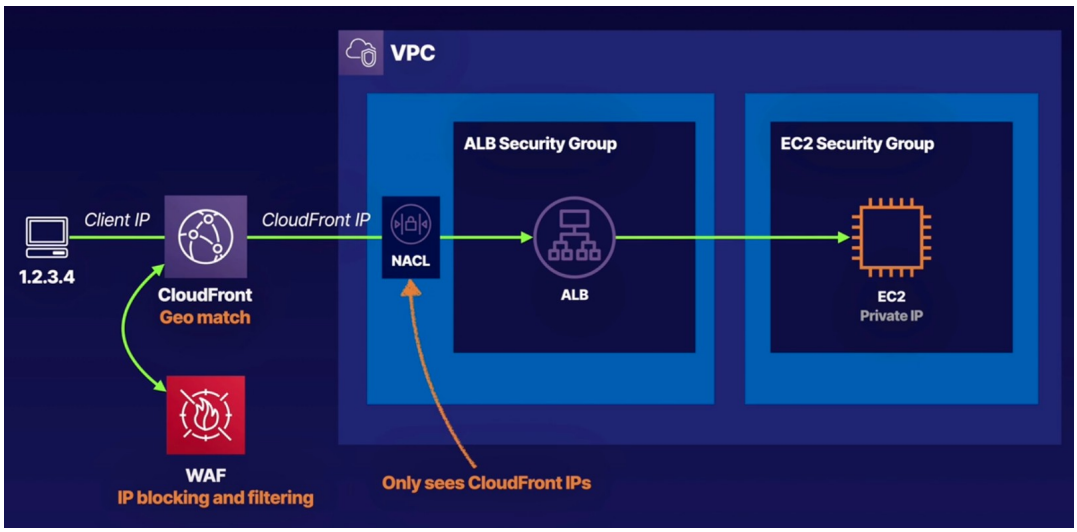
When to use WAF or NACL:
WAF works at layer 7 (application level) and NACL does it at 4.

If you want to block IP or ranges of IP NACL does the trick but attackers uses multiple IPs so could be hard to be constantly blocking new IPs.

IF you are operating a public web application you better use WAF.

Cloudfront:
If you are using cloudfront on your public web application you can also attach WAF to cloudfront.

Here we have the same problem if we have Cloudfront in front of our NACL. NACL will see only the IP of Cloudfront so would be ineffective to block the attaker IP.



In this cases you want to attach a WAF to your Cloudfront distribution and use the IP blocking and filtering options.

Additionally if you are getting abuses from a particular country you can use Cloudfront geomatch feature to block that country's traffic all together.

Key Management Services (KMS):

What is KMS:

- Regional secure key management and encryptionand decryption used to encrypt your data.
- Manages **Customer Master Keys (**CMKs), cmk is a logical representation of the key. It's a pointer or reference to some undrlying cryptografic material.

  The CMKs you create exist in a region and never leave that region or KMS at all.
- Ideal for S3 objects, database passwords and API keys stored in System Manager Parameter Store.
- CMKs can encrypt and decrypt data up to 4 kb in size.
- Integrated with most AWS services.
- You pay per API call. API like listing your keys, encrypting data, decrypting, etc.
- Supports audit capability using CloudTrail - logs delivered to S3.
- FIPS 140-2 Level 2 service. FIPS is a US government computer security standard used to approve cryptographic modules. Level 2 means that you just have to show evidence of tampering.
- Level 3 is CloudHSM. Level 3 have even more stringent security mechanisms than level two.
- If you see something in the exam about level 2, probably have something to do with KMS.

Types of CMK: Customer Master Keys

Type	Can View	Can Manage	Dedicated to My Account
Customer Managed	Yes	Yes	Yes
AWS Managed CMK	Yes	No	Yes
AWS Owned CMK	No	Yes	No

AWS Managed CMK: Are free. Used by default if you pick encrypton in most aws services. Only that service can use them directly.

You can track the usage of an aws managed cmk, but the lifecycle and permissions fo the key are managed on your behalf by KMS.

Customer Managed CMK: Only you can create. Allows key rotation. Controlled via key policies and can be enabled/disabled.
Cryptographic best practices encourages the rotation of the keys.

AWS Owned CMK: Used by AWS on a shared basis across many accounts.
You typically won't see these. You can not use,view or audit any of these.

Symmetric vs Asymmetric CMKs:

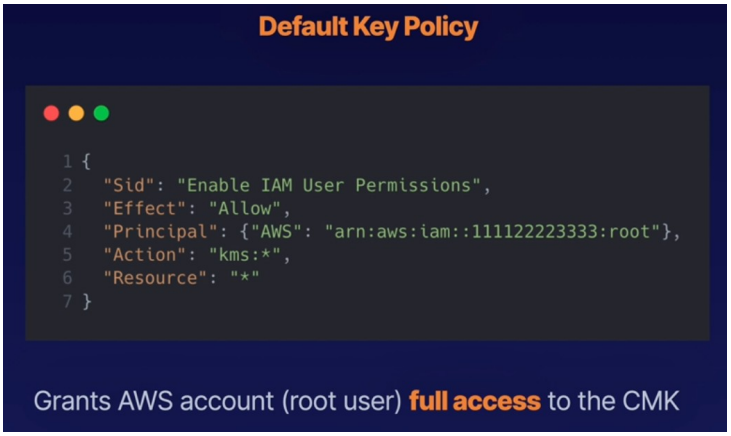
Symmetric:

- Same key used for encryption and decryption
- AES-256
- Never leaves AWS unencrypted
- Must call KMS APIs to use
- AWS services integrated with KMS use symmetric CMKs
- Encrypt, decryot and re-encript data
- Generate data keys, data key pairs, and random byte strings.
- Import your own key material.

Asymmetric:

- Mathematically related public/private key pair. You can give the public key to anyone even if is not trusted. The private key must to be keep secret. This is how SSL certificates work.
- Based in RSA and ellipti-curve cryptography (ECC). ECC is newer and considered better that RSA.
- Private key never leaves AWS unencrypted.
- Must call the KMS APIs to use private key
- Download the publick key and use outside AWS.
- Used outside AWS by users who can't call KMS APIs
- AWS services integrated with KMS do not support asymmetric CMKs.
- Sign messages and verify signatures.

The below key has one policy statement that gives the aws account, the root user, that owns the CMK full access to the CMK and enables IAM policies in the account to allow access to the CMK.



Warning: Pay close attention to the key policies attached to your keys. If you accidentally delete the statement or change its permissions, such that the account, the root user can't access the key you'll have to contact AWS support to regain access. So be very careful.

Example policy:

We grant a role called EncryptionApp access to a number of acctions on this key.

CloudHSM, Hardware Security Modules:

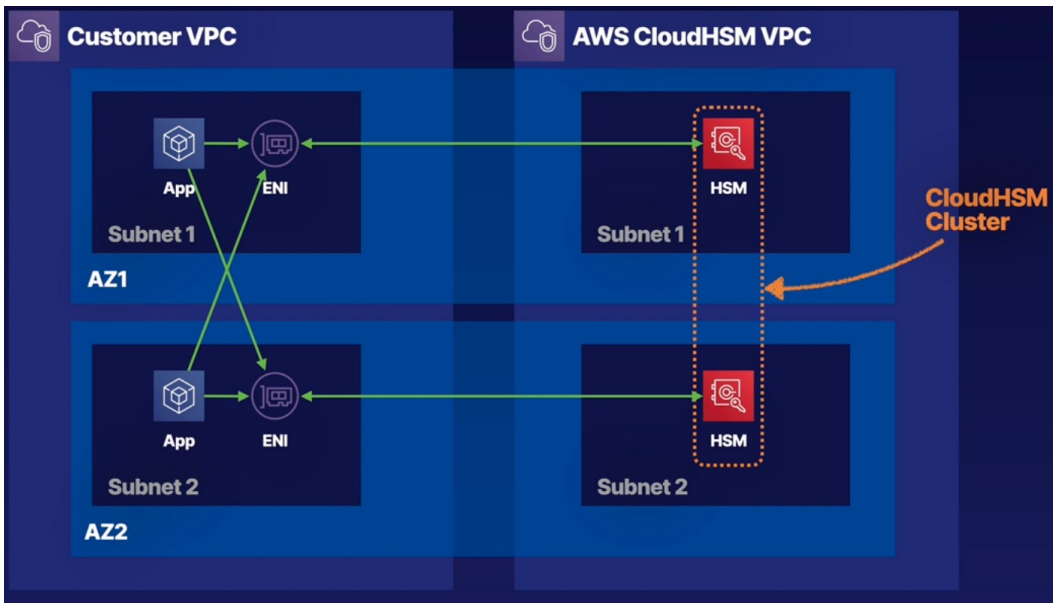
What is it:

- Dedicated hardware security module (HSM)
- FIPS 140- Level 3 → if you see this in the exam the answer is always CloudHSM
- Level 2 is KMS
- Manage your own keys → this is the difference between KMS
- No access to the AWS-managed component
- Runs within a VPC in your account
- Single tenant, dedicated hardware, multi-AZ cluster

- Industry-standard APIs. No AWS APIs
- PKCS#11
- Java Cryptography Extensions (JCE)
- Microsoft CryptoNG (CNG)
- Keep your keys safe - irretrievable if lost.

CloudHSM Architecture:

CloudHSM have to operate in its own vpc. Then it projects ENI (Elastic Network Inerfaces) into the VPC of your choosing. This is how your application communicates with the HSM cluster.



HSM is not HA by default. You have to explicitly provision HSM across AZ like in the above diagram.

Ideally you'll wanna place one HSM per subnet in each AZ with a minimum of 2 AZ, as AWS recommends.

Exam Tips:

- Regularory compliance requirements
- FIPS 140-2 Level 3 is CloudHSM

System Manager Parameter Store:

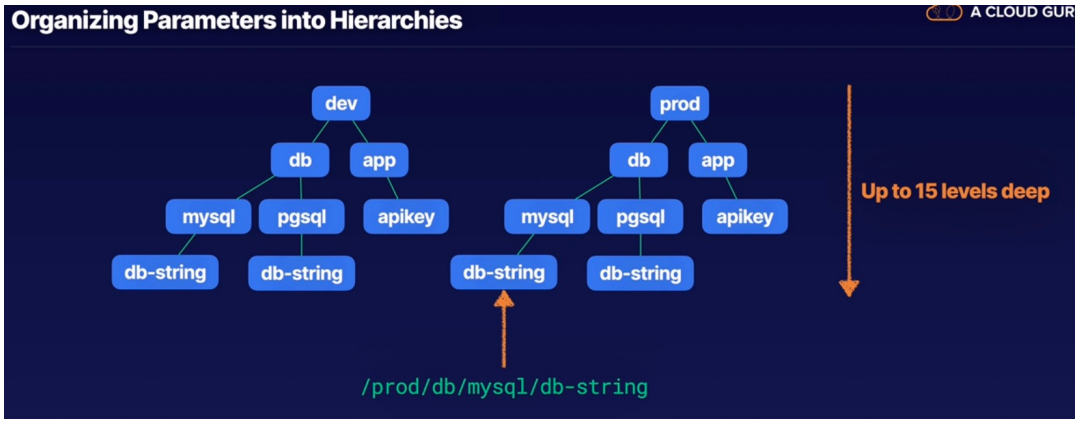
What is it: It's a serverless scalable and high performance system for storing data and secrets, like passwords, db connection strings, etc.

If you run a platform with thousands of instances that are being created and destroyed constantly you need a place where to store all the passwords for those instances. You could leave it in and S3 bucket, on git or in other places but this is not safe. That's when this service comes in.

- Is a component of AWS System Manager (SSM)
- Secure serverless storage for configuration and secrets:

    - Passwords
    - Database connection strings
    - License codes
    - API keys
- Values can be stored encrypted (KMS) or plaintext
- Separate data from source control
- Store parameters in hierarchies
- Track versions
- Set TTL to expire values such passwords

Store parameters in hierarchies:



You can retrieve data from the whole prod tree for example, or only from /prod/db/mysql/**db-string** for example.

You can grant permissions to a single piece of information, or to the whole area.
Or grant access to teams to the whole environments, or just to prod, dev or test, for example.

You have 15 levels deep.

Integration with CloudFormation:

With Parameter store you can easily integrate it with CloudFormation templates.

```
1  Parameters:
2    LatestAmiId:
3      Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
4      Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'
5
6  Resources:
7    Instance:
8      Type: 'AWS::EC2::Instance'
9      Properties:
10       ImageId: !Ref LatestAmiId
```

In line 4 we have the AMI.
Line 10 we have the ec2 instance resource.

So there is no need to create and manage complex mappings like before the Parameter Store was introduced.

Secret Manager:

What is it:

- Similar to System Manager Parameter Store
- Charge per secret store and per 10.000 API calls. So takes care of the **cost** of the service
- Automatically rotate secrets. Full key rotation integration with RDS.
- Apply the new key/password in RDS for you
- Hability to generate random secrets, parameter store can not do it.

AWS Shield:

What is it: Help to protect agains DDOS attacks.

AWS Shield tiers:

- **AWS Shield Standard:**
  - Automatically enabled for all customers at no cost
  - Protects against common layer 3 and 4 attacks.
    - SYN/UDP floods: Every UDP packet received have to be replied. If you are flooded with them the regular connection to the server will be imposible.

      SYN floods work similar to UDP but they work on TCP, making the server to wait for an answer but no one is coming. It leaves a huge number of connections half open waiting for handshakes that never complete.

    - Reflection attacks: UDP attack where the source IP address of packets is spoofed, where the victim will end up receiving a large volume of response packets it never requested and Shield has absolutely proven effective.

- **AWS Shield Advanced:**
  - Enhanced level of protection but at $3000 per month per AWS organization.
    - Enhanced protection for EC2, ELB, CloudFront global accelerator, and Route 53
  - Busines and Enterprise support customers get 24x7 access to the DDoS Response Team (DRT)
  - DDoS cost protection:

Web Application Firewall, WAF:

What is it: Web application firewall that lets you monitor HTTP(S) requests to CloudFront, ALB, or API Gateway.

- Control access to content: Based on rules that you specify WAF directs your service to respond to requests either with the requested content or an error.

- You do this by configuring filtering rules to allow/deny traffic. Type of filtering:
  - IP addresses
  - Query string parameters, like https://example.com/over/there?name=ferret → question marks, ampersands, equal signs,etc, these are name value pairs.

    You can configure WAF to allow or deny them and it can also protect you agains SQL query injection attacks.

- If the WAF block traffic it returns in HTTP 403 error code, which is the forbidden code.

How does WAF works:
- **Allow** all requests, except the ones you specify.
- **Block** all requests, except the ones you specify.
- **Count** the requests that match the properties you specify.
  - Request properties:
    - Originating IP address
    - Originating country
    - Request size
    - Values in request headers, example block http requests without headers
    - Strings in request matching regex patterns
    - SQL code (injection)
    - XSS

AWS Firewall Manager: Service integrated with WAF. Centrally configure and manage firewall rules across an AWS Organization.

Using AWS Firewall Manager you can:
- Deploy WAF rules for:
  - ALB
  - API Gateway
  - CloudFront distributions
- You can deploy AWS Shield Advanced protections:
  - ALB
  - ELB Classic
  - Elastic IPs (EIP)
  - CloudFront distributions
- Enable security group for EC2 and ENIs

Serverless:

Lambda:

What is Lambda: Is a compute service where you can upload your code and create a Lambda function.

AWS Lambda takes care of provisioning and managing the servers that you use to run the code. You don't have to worry about operating systems, patching, scaling, etc.

The basics: Lambda is the ultimate abstraction layer. The whole physical path to lambda that AWS manages is:
Datacenters - Hardware - Assembly Code/Protocols - High Level Languages - Operating Systems - Application Layer/AWS APIs - AWS Lambda

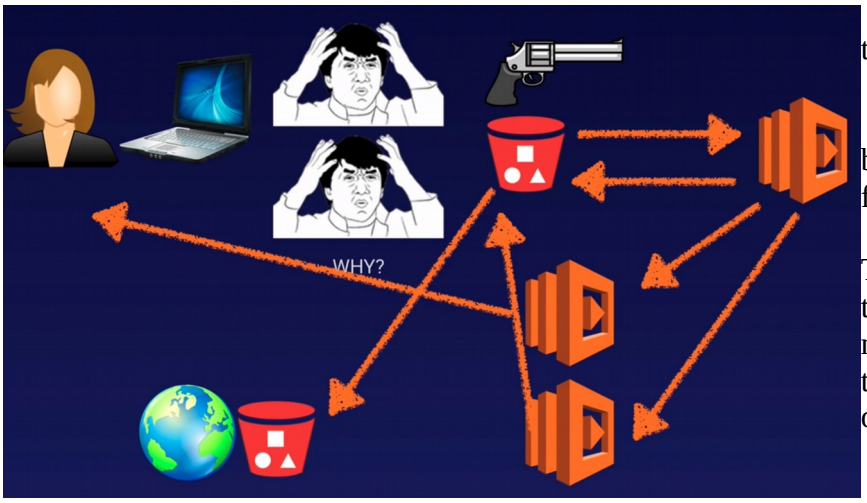
Amazon manages all that, you only need to worry about your code.

Where to use Lambda:

- As event-driven compute service where AWS Lambda runs your code in response to events. These events could be changes to data in an Amazon S3 bucket or an Amazon DynamoDB table, etc.
  Those event-driven compute services are called triggers.

- As a compute service to run your code in response to HTTP request using Amazon API Gateway or API calls made using AWS SDKs. This is what A Cloud Guru uses.

How it works:



We have our user and she wants to create a meme.

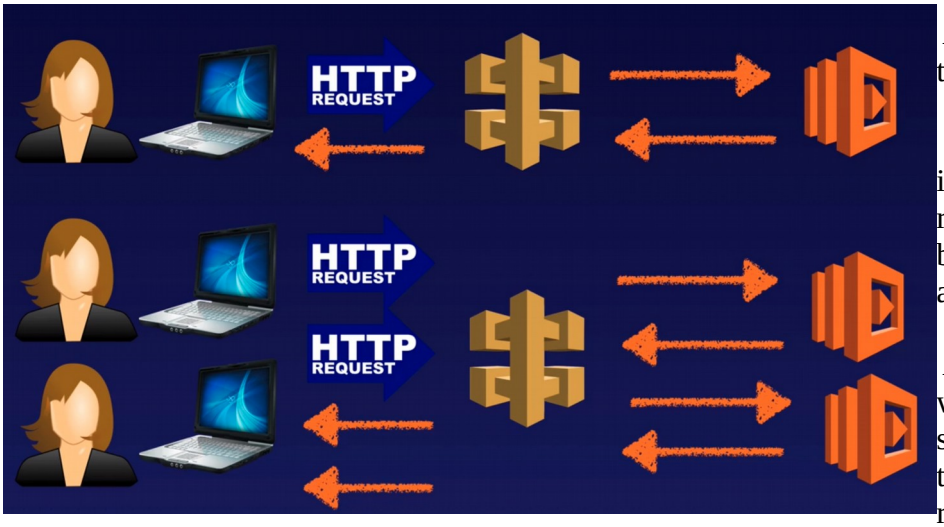
She uploads the meme to an S3 bucket which triggers a Lambda function.

The lambda function will take that meme, will take the metadata as well, and it will put the word "Why" for example, over the meme.

Lambda can trigger another Lambda function for example to tell the user that this is your meme.
Can trigger another lambda function to store that meme back in an S3 bucket.
And you can use Cross Region replication to store that meme somewhere in the world.

The other way it works: As a compute function

Our user sends an HTTP request to API Gateway.



API Gateway proxies that to Lambda.

Lambda will run the code in response to that HTTP request, and then send it back to that API Gateway and sent it back to the user.

A cool thing about this way Lambda scales, let's say we have two users with two separate HTTP requests, both will hit API Gateway, and both will trigger the same lambda function but it will be separately run, isolated from each other and returning both back to the user.

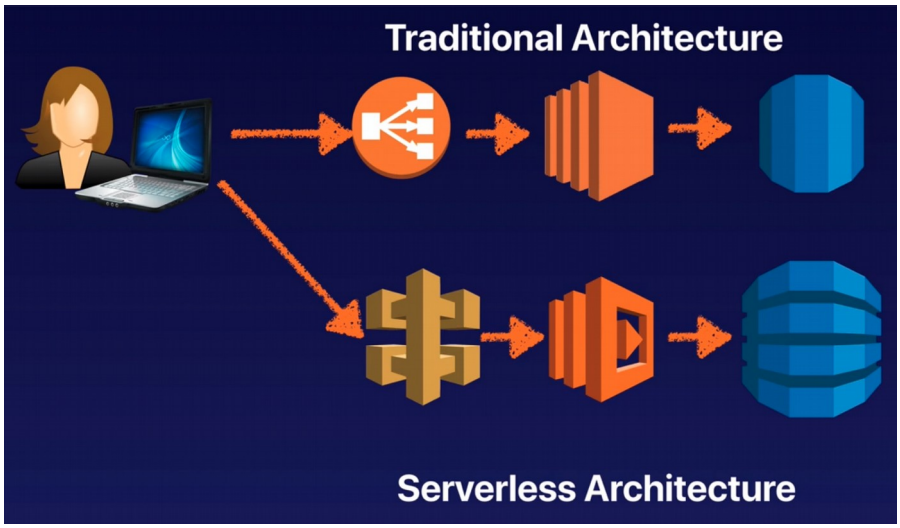
Lambda is a bit different from auto scaling. With autoscaling you might just spread it across two webservers.

With lambda if you have a million users hitting your API Gateway, that will then go on and trigger a million different lambda functions, that's how well scales.

Traditional vs Serverless Architecture:

Traditional Architecture:
The user sends a requests -- hits Route 53 -- then goes out to the ELB -- the LB sends it on to your webservers -- then the web servers may communicate with your RDS or database server -- then sends back a response to the user.

You rely on virtual machines, OS, LB, etc.



Serverless Architecture: The user sends a response to the API Gateway -- that then sends a response to Lambda -- which can write to DynamoDB, Amazon Aurora Serverless -- then it send its back to the user.

Languages supported by Lamda:

- Node.js
- Java
- Python
- C#
- Go
- Powershell

Lambda Pricing:

- Number of requests: First 1 million requests are free. $0.20 per 1 million requests thereafter.

- Duration: Duration is calculed from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 100ms.
  The price depends on the amount of memory you allocate to your function.
  You are charged $0.00001667 for every GB-second used.

Is very cheap. A Cloud Guru have arround 1 million users and they pay $400 a month aprox. Doing the same level of compute with EC2, they'd probably be spending upwards $100,000 per month.

It build on duration. If the function runs for 100 ms then you're going to be charged a certain amount, if it runs 200 ms it charges a little more.

It charges by the number of requests and the duration of those requests.

Why is Lambda cool:

- No servers, no sysadmins, patching OS, no worries about why the server crash, etc.

- Continuosly scales, anytime you make an invocation to API Gateway, that will trigger a separate lambda function.

- It's very cheap compared with traditional architecture.

Lambda in action:
Every time you speak with Alexa, Amazon Echo, etc, you are actually speaking to Lambda.

Exam Tips:

- Lambda scales out (not up) automatically
- Lambda functions are independent, 1 event = 1 function
- Lambda is serverless
- Know what services are serverless. (Have in mind that, like with RDS, even though you don't have to worry about the OS, aws takes care of it, so still have to be patched, doing maintenance, etc.)
  - Aurora Serverless
  - S3
  - Lambda
  - DynamoDB
  - API Gateway

- Lambda functions can trigger other lambda functions, 1 event can = x functions if functions trigger other functions.
- Architectures can get extremely complicated with Lambda, AWS X-ray allows you to debug what is happening.
- AWS X-ray helps you to debug your serverless applications.
- Lambda can do things globally, you can use it to back up S3 buckets to other S3 buckets etc.
- Know your triggers, what can trigger lambda, what can not trigger it.

Serverless Application Model, SAM:

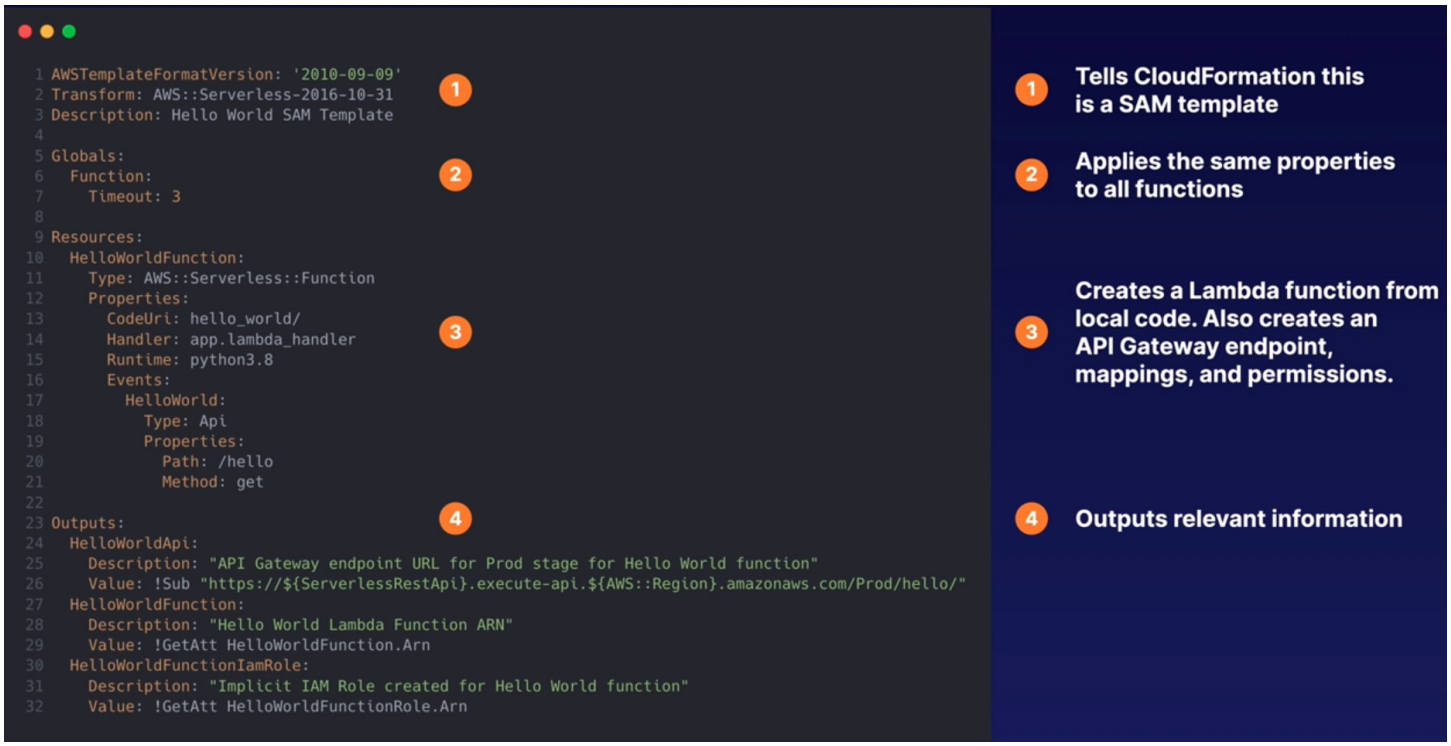
What is SAM: CloudFormation extension optimized for serverless applications.

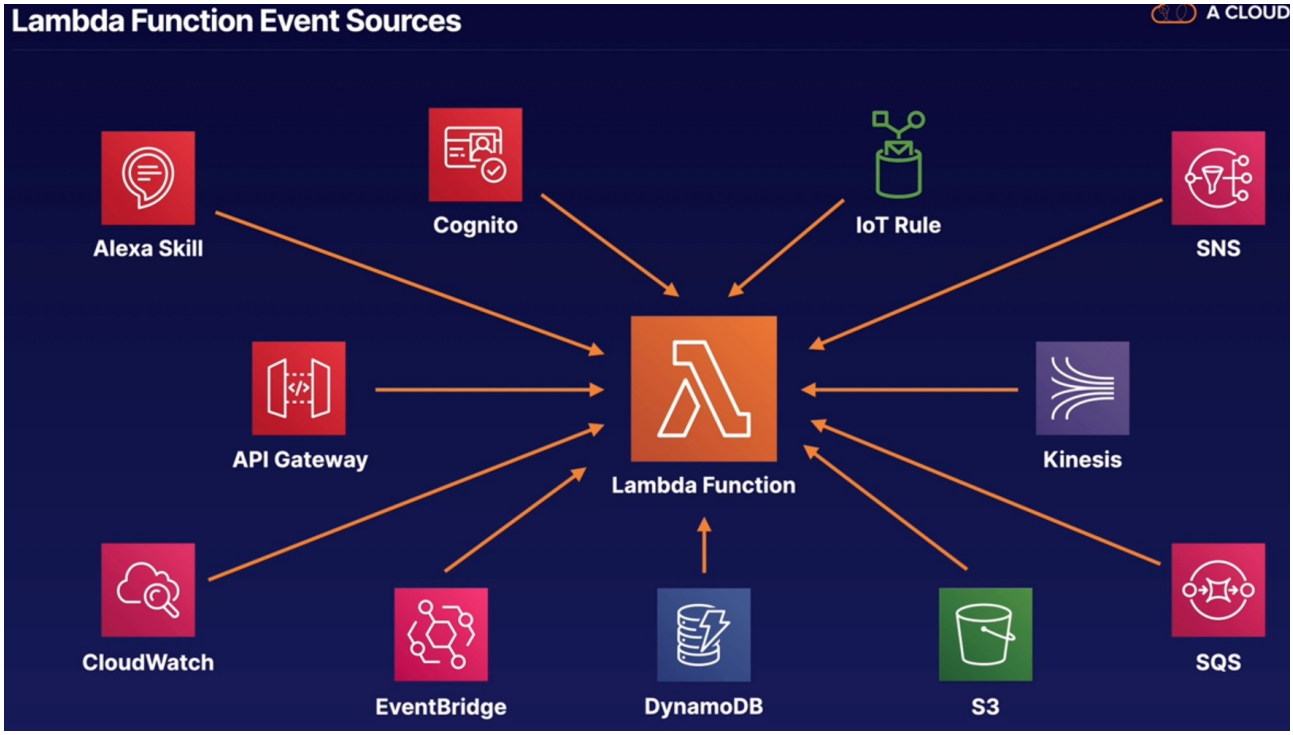
- On top of CloudFormation it defines a number of new resources types:
  - Functions
  - APIs
  - tables
  - dynamoDB
  - etc

- With just a few lines of configuration you can define the application you want and model it using SAM.

- Support anything CloudFormation supports

- Run serverles applications locally





Elastic Container Service, ECS:

What is ECS:

- Managed container orchestration service.
- Create clusters to manage fleets of container deployments.
- ECS manages EC2 or Fargate instances
- Schedules containers for optimal placement
- Define rules for CPU and memory requirements
- Monitors resource utilization
- Deploy, update, roll back
- Free scheduler and orchestration components or the cluster. You pay for the ec2 or fargate resources
- VPC, security groups, EBS volumes
- ELB
- CloudTrail and CloudWatch

ECS Components:

- **Cluster:** Logical collection of ECS resources, either ECS EC2 instances or fargate instances

- **Task Definition:** Defines your application. Similar to a Dockerfile but for running containers in ECS. Can contain multilple containers.

- **Container Definition:** Inside a task definitio, it defines the individual containers a task uses. Controls CPU and memory allocation and port mappings.

- **Task:** Single running copy of any containers defined by a task definition. One working copy of an application (e.g., DB and web containers)

- **Service:** Allow task definitions to be scaled by adding tasks. Defines minimum and maximum values.

- **Registry:** Storage for container images, elastic container registry (ECR) or docker hub. Used to download images to create containers.

Fargate: Serverless compute engine for containers that work with ECS and EKS.
- Eliminates need to provision and manage servers
- Specify and pay for resources per application
- Works with both ECS and EKS
- Each workload runs in its own kernel
- Isolate and security.
- Choose EC2 instead if:
  - Compliance requirements
  - Require broader customization
    Require access to GPU

EKS: Elastic Kubnernetes Services
- K8s is open source software that lets you deploy and manage containerized applications at scale.
- Same toolset on premises and in cloud
- Containers are grouped in pods
- Like ECS, supports both EC2 and Fargate
- Why use EKS:
  - Already using K8s
  - Want to migrate to AWS your K8s cluster

ECR: Elastic Container Registry and Docker Hub
- Managed Docker container registry
- It let's you store, manage, and deploy images
- Integrated with ECS and EKS
- Works with on-premises deployments
- Highly available
- Integrated with IAM
- Pay for storage and data transfer, similar to S3

ECS + ELB:
- Distribute traffic evenly across tasks in your service
- Supports ALB, NLB, CLB
- Use ALB to route HTTP/HTTPS (layer 7) traffic
- Use NLB or CLB to route TCP (layer 4) traffic
- Supported by both EC2 and Fargate launch types
- ALB allows:
  - Dynamic host port mapping
  - Path based routing
  - Priority rules
- ALB is recommended over NLB or CLB

Security:

To respect the Least Privilege security approach is better to use the Task role, where you split into different roles the access to the reuces, instead of all accessing everithing like in the ec2 instance role.