# Architecture 101:

## Encryption

Encryption is the process of taking plane text and converting to cipher text and converting cipher text into plain text. Plain text and cipher text can be text image or any other data.

Encryption is generally used an algorithm and one or more keys, it’s generally used to encrypt data at **rest** (when local files/data are encrypted)or **in** **transit**.

### Symmetrical

Here the same key is used for both encryption and decryption, hence transferring the key is the challenge here.

### Asymmetrical

Here there will be two keys, **public key** used for encryption and **private key** used for decryption. Vendor can generate and send the public key over anything as it doesn’t make any sense standalone. The party will always hold the private key to decrypt the message.

So private key will never become public sharing.

## RPO vs RTO

These are the parts of disaster recovery

### RPO

**Recovery Point Objective** where it decides how much business it can tolerate to lose in terms of time. This is the maximum time between a failure and last successful backup

### RTO

**Recovery Time Objective** is the amount of time a system can be down. How long a solution takes to recover.

# AWS Architecture 101:

## AWS Global Infrastructure

### Region and Availability Zone

AWS have **Region** and **Availability Zone**, Data stored in a AWS region will not leave the region boundary because the data handling laws/policy could vary from region to region.

Region is identified by region-code and region-name.

Region contains multiple Availability Zone (AZ) are the data centers which are separated and isolated networks. A failure in on AZ won’t impact other which makes region fault tolerant.

AZs in same region are connected with redundant, high speed, low latency network connections.

Most AWS service runs within AZs. Some series operates from one AZ, while other replicate between AZs. Some service allows us to choose the AZ where some doesn’t.

### Edge Locations

Edge locations/Points of presence are small pockets of AWS compute, storage and networking close to major populations and generally used for edge computing and content delivery.

## AWS Well-Architected Framework

This is a set of best practice guidelines principles to architect AWS structure for our need. This framework provides some principles to follow.

**Security, Reliability, Performance Efficiency, Operational Excellence, Cost Optimization.**

Refer related white paper:

<https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf>

## Elasticity

With vertical scaling based on fluctuation of demand our capacity may not meet most of the time server either could be over burden or underutilized and with huge capacity resource wastage on non-peek time not avoidable.

With horizontal scaling, instead of a high capacity server we will have multiple low capacity server chained, but here too the demand to capacity need of underutilization and over burden can’t be avoided.

Where comes **Elasticity** which is an automated version of horizontal scaling where based on the demand hour capacity will be increased or decreased. Elasticity provides flexibility to scale in or scale out to very maximum to complete minimum.

Elasticity helps to achieve maximum cost optimization and performance efficiency.

# AWS Product Fundamentals

## S3 Simple Storage Service

S3 is a **global** object storage platform that can be used to store objects in form of text files, photos, audios, videos, large binaries and other object type.

The region for s3 is Global region. The basic unit of S3 is **bucket.** The bucket name should be unique globally.

Anything we will store inside S3 bucket will be replicated across the AZs in the region. Bucket is a flat store, so even we create a folder inside bucket still it’s an object got mapped.

Object (key value, key is name value is the data) will have a unique name inside an object. Object can have size till 5TB.

S3 is great for storage and also for internet access of our data stored.

S3 can’t be mounted to server as network drive or mount point. We shouldn’t attach S3 buckets to server as disk or block storage.

### S3 quick notes

* Bucket name should be globally unique.
* Minimum of 3 maximum of 63 chars – no uppercase or underscore.
* Must start with a lowercase or number but can’t formatted as an IP address.
* Default 100 buckets per account and hard 1,000 buckets limit via support request.
* Unlimited object and capacity inside a bucket.
* The objects key is a name and value is the data.
* An object size from 0 to 5TB.

## Introduction to CloudFormation

AWS CloudFormation is an automation product available in AWS as Infrastructure as Code (**IoC**) product. We can create, manage and remove infrastructure using template JSON or YAML.

So we can create a template to automate multiple redundant tasks of infrastructure create and management. Ex for Dev, QA so on.

### The three building blocks of CloudFormation

**Template**

Is a JSON or YAML contains logic resources and configurations

**Stack**

Stacks are created and modified based on template, which also can be used to change and update a stack.

**Physical Resources**

Stack takes logical resources from template and create, update and delete physical resources in AWS.

### Quick tips

* A template can create up to 200 resources.
* Change in logical resource update or replace the physical resources with some disruption.
* New logical resource cause new physical resource.

### Useful resources

Github file:

<https://github.com/linuxacademy/content-aws-csa2019/tree/master/lesson_files/01_aws_sa_fundamentals/introduction_to_cloudformation>

CloudFormation template:

<https://s3.amazonaws.com/cloudformation-templates-us-east-1/WordPress_Single_Instance.template>

AWS CloudFormation resource reference:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>

# Identity and Access Management (IAM)

## IAM Essentials

IAM is the primary service that handles authentication and authorization within AWS environment i.e. AWS api.

IAM controls access to AWS service via **policies** that can be attached to **users, groups** and **roles.** Users are given long term credentials to access AWS resources.

Roles allow for short-term access to resources when assumed, using temporary access credentials.

### ARN – Amazon Resource Name

ARN is a notation in AWS which can unique identify any resource under any user any region.

**Format of ARN:**  arn:partition:service:region:account-id:

Ex: arn:aws:s3:::myamazingcatpics/truffs.jpeg

arn:aws:ec2:us-east-1:1246863355:instance/\*

**Note:** Fields with :: omits the value and \* is a wildcard match

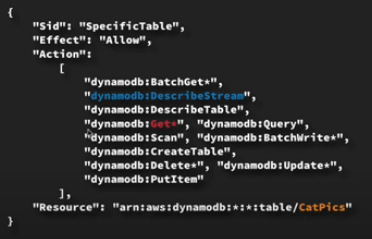
## IAM Policies

An IAM Policy (Policy Document) is known as an **identity policy** when attached to an identity or a **resource policy** when attached to a resource. They have no effect until they attached to something.

A policy document is a list of statements:



A statement matches a request to AWS. Requests are matched based on their **Action** which are API calls. A given statement results in an **Allow** or **Deny** for the request.



**Exam tips: IAM policies**

* If a request isn’t explicitly allowed, it’s implicitly (default) denied.
* If a request is explicitly denied, it overrides everything else.
* If a request is explicitly allowed, it’s allowed unless it’s explicitly denied
* Remember **DENY -> ALLOW -> DENY**
* Customer-managed policies are flexible but required administration.
* Inline and managed policies can apply to users, groups and roles.

## IAM Users

IAM users are type of IAM identity suitable for long-term access for a known entity (human, service, application).

Principals authenticate to IAM user either with a user name and password or access keys.

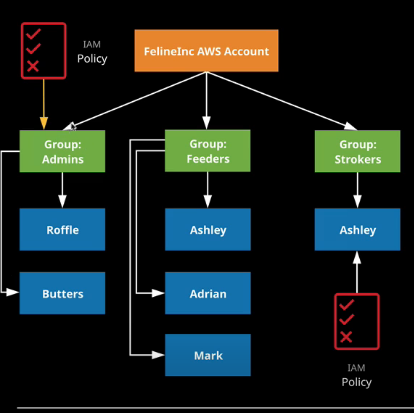
**Exam tips:**

* Hard limit 5,000 IAM users per account.
* 10 group membership per IAM user.
* Default maximum 10 policies per user.
* No inline limit, but you can’t exceed 2048 characters for all inline policies on a IAM user.
* 1 MFA per user.
* 2 access keys per user.

## IAM Groups

IAM Group is a collection of IAM users. Groups allows easier administration over sets of IAM users. Inline and managed policies can be applied to groups that flow on to members of that group.

Groups are not true identity, they can’t be a principal in a policy, so they can’t be used in resource policies.



**Exam Tips:**

* Groups are admin feature to group IAM users.
* Groups can contain many IAM users and users can be in many groups.
* IAM inline policies can be added to IAM groups and these flows into IAM users under that group.
* Managed IAM policies can be attached and flow onto IAM users who are members.
* Group is not the true identity and can’t be referenced from resource policies.
* Groups have no credential.

## IAM Access Keys

Access keys are pair of values used by applications, SDKs or the AWS command line to authenticate to AWS.

Access key contains two parts **access key ID** and **secret access key.** The access key ID is the public part of the key and is stored by AWS once generated.

The secret access key is private part should be stored by user and never be shared and neither can be changed if lost.

An IAM user is the only identity that uses access keys. They are allowed two sets. They can be created, deleted, enabled and disabled.

Access keys can’t be used to login to console, and they don’t expire. If anyone finds a set of access keys, they have access to the permission of the IAM users to which they belongs.

An user can only have 2 access keys.

## IAM Roles

IAM roles are **assumed** by another identity allowed in the **trust policy** – IAM user, AWS service, another AWS account, web identity or even anonymous identity. When a role assumed the Security Token Service (**STS**) generates a **time-limited** set of access keys (temporary security credentials). These access keys have permission defined in permissions policy. IAM roles have no long term credentials (access keys or user name or password).

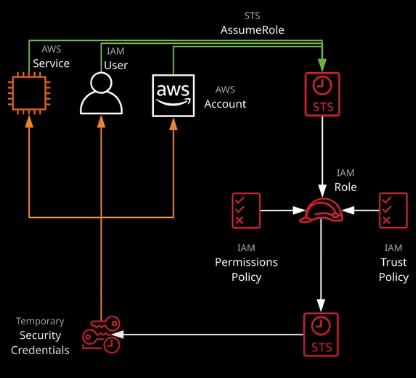
**Trust policy** controls which identities can assume the role.

**Permissions** policy defines the permissions provided.

Example Scenario:

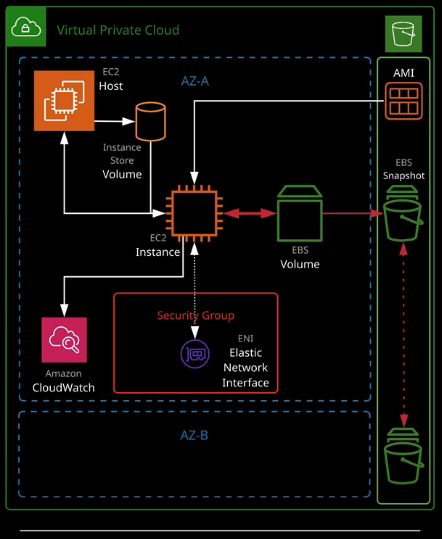
* Company merger
* AWS service access
* “Break glass” style extra access
* Cross-account access
* Web identity federation

Architecture (How IAM role works)



# Server-Based Compute (EC2) Fundamentals

## EC2 Architecture



## Instance type and size

**EC2 instances** are grouped into families, which are designed for a specific broad type of workload. The type decides a certain set of feature where size decides the level of work load they can cope with.

The current EC2 families are **general purpose, compute optimized, memory optimized, storage optimized** and **accelerated computing.**

**Instance type** includes

* **T2** and **T3:** Low cost instance types that provides burst capability.
* **M5:** For general workloads
* **C4:** Provides some more capable CPU
* **X1** and **R4:** Optimize large amount of fast memory
* **I3:** Delivers fast IO
* **P2, G3** and **F1:** Delivers GPU and FPGAs

**Instance sizes** include nano, micro, small, medium, large, x.large, 2x.large and larger.

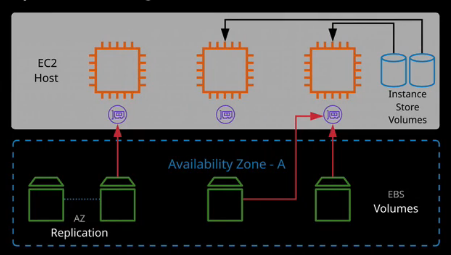
**Special cases**

* **“a”** for AMD cpu
* **“A”** for ARM based
* **“n”** for higher speed networking
* **“d”** for NVMe storage

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

## EC2 Storage architecture

**Instance store volumes** are the storage attached to EC2 host, so if the host fails or changed the storage is lost. But this store is fastest as this is internal to host. It can be used as temporary store volume but nor resilient and non-persistence.



### EBS

**EBS** is a network storage based product we can create volume attach to the instances. EBS we can use only in a particular AZ. If we need to move between AZs, we need **snapshot.**

**EBS** volume is persistent, can be attached and removed from EC2 instances and are replicated with in single AZ.

* **Sc1** is a lowest cost, infrequent access and can’t be used as boot volume.
* **St1:** low cost, throughput intensive and can’t be used as boot volume.
* **Gp2:** Default, balance of IOPS/MiB/s – burst pool IOPS per GB.
* **Io1:** High performance, can adjust size and IOPS separately.

To protect against **AZ failure,** EBS snapshots (to S3) can be used. Data is replicated across AZs in the region and optionally internationally.

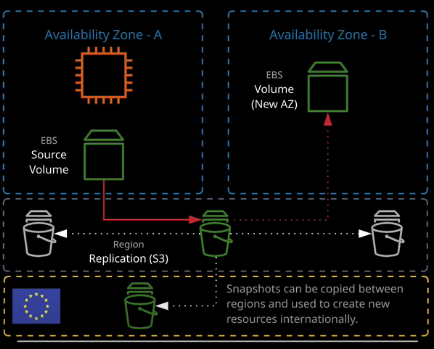
## EBS Snapshots

**EBS Snapshots** are a **point-in-time backup** of an EBS volume stored to S3. The initial snapshot is a **full copy** of the volume. Future snapshots are only the data changes since the last snapshot.

Snapshots can be used to create new volumes and great way to **move** and **copy** data between AZs. When creating snapshot of root/boot volume it is recommended the instance is powered off or disks are flushed.

Snapshots can be shared between AWS accounts or even publicly.

Snapshots can be copied between regions, shared and automated using Data Lifecycle Manager (DLM)



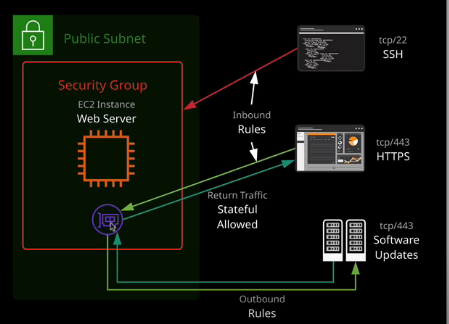
## Security Group

Security groups are the software firewall that can be attached to network interfaces product in AWS. By default every EC2 install is created with ENI (Electric Network Interface) to which **security group** is attached.

Security groups each have inbound rules and outbound rules. A rule allow traffic **to** and **from** a source (IP, Network, named AWS entity) and protocol.

Security group have a hidden default/implicit deny rule but can’t explicitly deny traffic.

They are stateful, means any traffic allowed in/out the return traffic also allowed. Security group can reference AWS resources, other security groups and even themselves.



## Instance metadata

Instance metadata is data relating to the instance that can be accessed from **within the instance itself** using a utility capable of accessing HTTP (ex. curl) and using the URL

<http://169.254.169.254/latest/metadata>

The instance metadata is a way that scripts and application is running on EC2 can get **visibility of data** they would normally need API calls for.

## EC2 Notes

* When we stop and start EC2 instance the physic host changes so does the hardware and **public -ip.**
* When we restart the EC2 instance the public-ip doesn’t change as physical host is same.

# Virtual Private Cloud

## VPC and Subnets

VPC in AWS provides network isolation. It is a private data center inside the AWS platform.

By default VPC can’t talk to anything outside of it unless configured.

VPC are regional (can’t span across regions) highly available and can be connected to your data center and corporate network.

VPC is isolated from other VPC by default.

VPC and subnet max /16 (65,536 IPs) and minimum /28 (16 IPs)

VPC subnets can’t span AZs (1:1 mapping)

VPC is regional based product.

Certain IPs are reserved in subnet.

There are 2 types of VPC exists in every AWS account created.

* Default VPC
* Custom VPC

**Default VPC** by default created with every AWS account. There are certain services in AWS which requires default VPC to run.

It is configured using *16 CIDR block (172.31.0.0/16). A /20* public subnet in each AZ allocating a public IP by default.

Attached internet gateway with main route table sending all IPv4 traffic to internet gateway using a 0.0.0.0/0 route.

A default DHCP option set is attached.

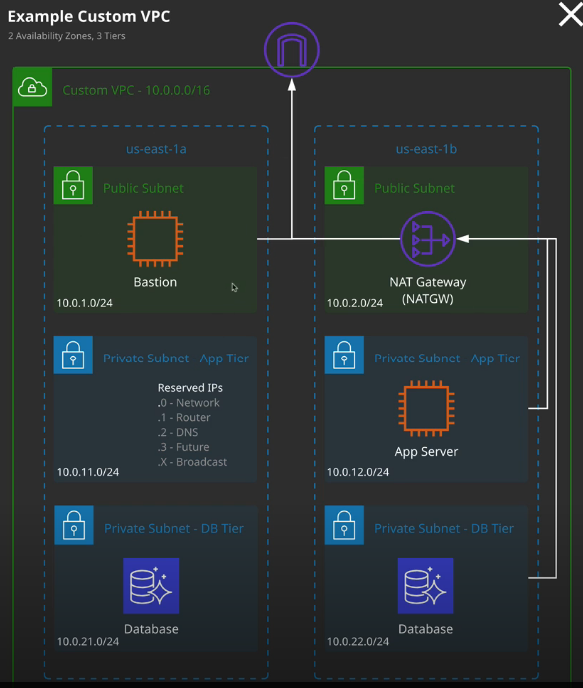
SG: Default – all from itself, all outbound

NACL: Default – allow all inbound and outbound

## Part 2

When we create a VPC that’s not enough because services don’t go directly into a VPC.

A subnet is an isolated local network and VPC can have many subnet. Subnet is created inside a AZ. So they can’t span across AZs. But an AZ can have many subnets.



In every single subnet there will be 5 reserved IPs.

DHCP is a protocol that allows resources inside a network to auto configure their IP addresses. It is associated with VPC.

## Routing and Internet gateway

Every VPC has a virtual routing device called VPC router. VPC router moves traffic between subnet and also outside VPC to public network on configuration.

VPC router is controlled using route table. Route table is container for one or more routes. When we create a VPC it creates a main route table for that VPC which is allocated to all subnets inside that VPC by default. **A subnet must have only one route table.**

A route table controls what VPC router does with traffic leaving from subnet.

An Internet Gateway is created and attached to a VPC (1:1). It can route traffic for public IPs to and from the internet.

The subnet exists inside VPC is of 2 types: **public** and **private**

Every route table has a **local** router, which matches the CIDR of the VPC and lets the traffic be routed between subnets.

A route contains a destination and target. Traffic is forwarded to target if destination matches.

If multiple routes apply, the most specific is chosen(highest prefix). A /32 is chosen before /24 and a /16.

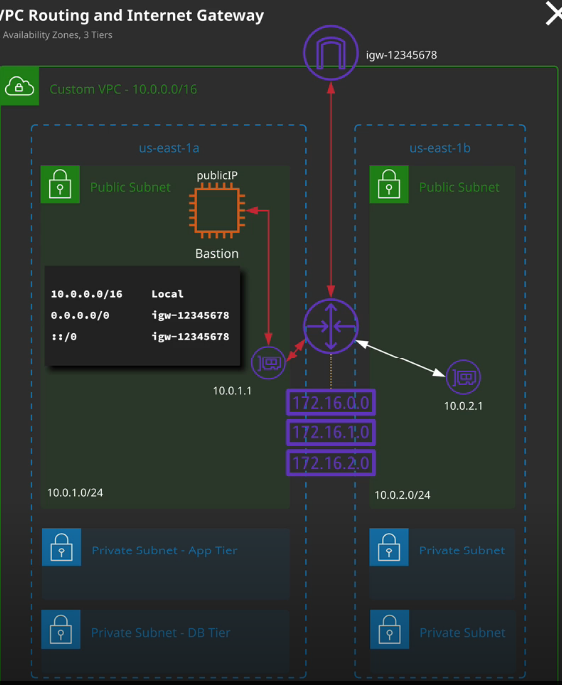
Default routes (0.0.0.0/0 iv V4 and ::/0 V6) that match any traffic that already not matched.

A subnet can be made public, 3 steps we need to do to make it so.

* Configure the subnet to allocate public IP to its resources.
* Create an Internet gateway and attach it to VPC (IGW is a highly available product by design, we don’t have to worry about performance and failure). It will do SNAT i.e. translate private IP to public IP.
* Add routes. We need to tell the VPC router to forward any IP traffic destined for internet through the IGW (Internet Gateway) - To that better to create a new route table instead updating the default route table for future switching purpose. We should add the subnet to route table we created for public access.

Till now what we have configured is static route. But even we can configure dynamic route (route propagation) which happens through BGP (Border Gateway Protocol)

BGP is a protocol to advertise and learn routes from other router.



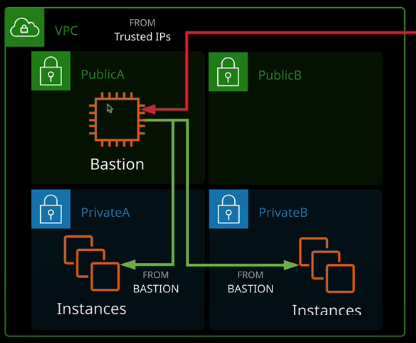
## Bastion Host / Jump Box

It’s a host that sits at the perimeter of a VPC. It functions as entry point to the VPC for trusted admins.

If the instances are running on private and we want occasionally connected in to either for software updates or configuration changes then rather than having private resources available to public internet we can have single Bastion host in public subnet which is accessible and by connecting to that Bastion host from there connect to the internal resources.

Bastion host generally connected to via SSH or RDP (windows)

Bastion host must be kept updated and security hardened and audited regularly.

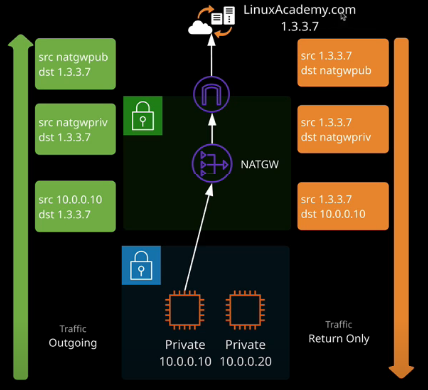


## NAT, NAT Instance and NAT Gateway

There could be a situation where resources inside VPC needs some level of out going internet access may be for software updates.

Network Address Translation(NAT) is a method of remapping source IPs or destination IPs of packet. It can be used in a number of ways.

* **Static NAT** A private IP is mapped to a public IP (what IGW do)
* **Dynamic NAT** A range of private addresses are mapped onto one or more public IP(s) (used by home router and NAT gateways)

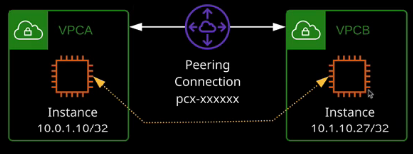
****

NAT Gateways needs to be created inside public subnet and needs to given Elastic IP address which will be used to connect through IGW to any public address.

NAT Gateways are not highly available by default that means if any subnet fails then the NAT Gateway associated to it will Fail. To make it highly available we need to create NAT Gateway in **each** **public subnet** in the VPC**.**

# Advanced VPC

## VPC Peering

If we decides to establish direct communication between VPC we need to do VPC peering.

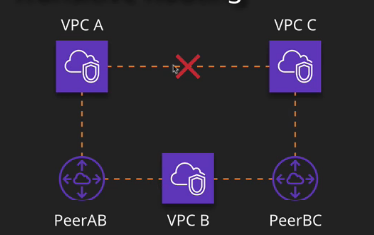
Services can communicate using private Ips from VPC to VPC

VPC peering can span AWS accounts and even regions (with some limitations)

Data is encrypted and transmitted using AWS global backbone.

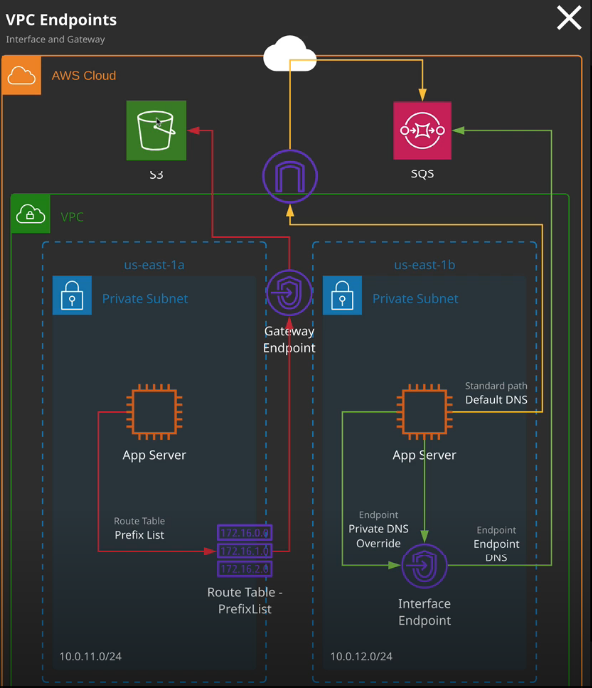
VPC peering used to link two VPC at layer 3.

### Important limitation and considerations

* VPC CIDR block can’t overlap
* VPC peers connects two VPCs – routing is not transitive
* Routes are require at both side.
* NACL and SG can be used to control the access
* SGs can be referenced but not cross region-code
* DNS resolution to private IPs can be enabled, but it’s a setting needed at both side.

## VPC Endpoint

VPC endpoint are the gateway objects created with in a VPC. They can be used to connect to AWS public services without the need for the VPC to have an attached internet gateway and be public.

**VPC Endpoint Types:**

* Gateway endpoints: can be used for Dynamo DB and S3
* Interface endpoints: can be used for everything else (eg. SNS, SQS etc)