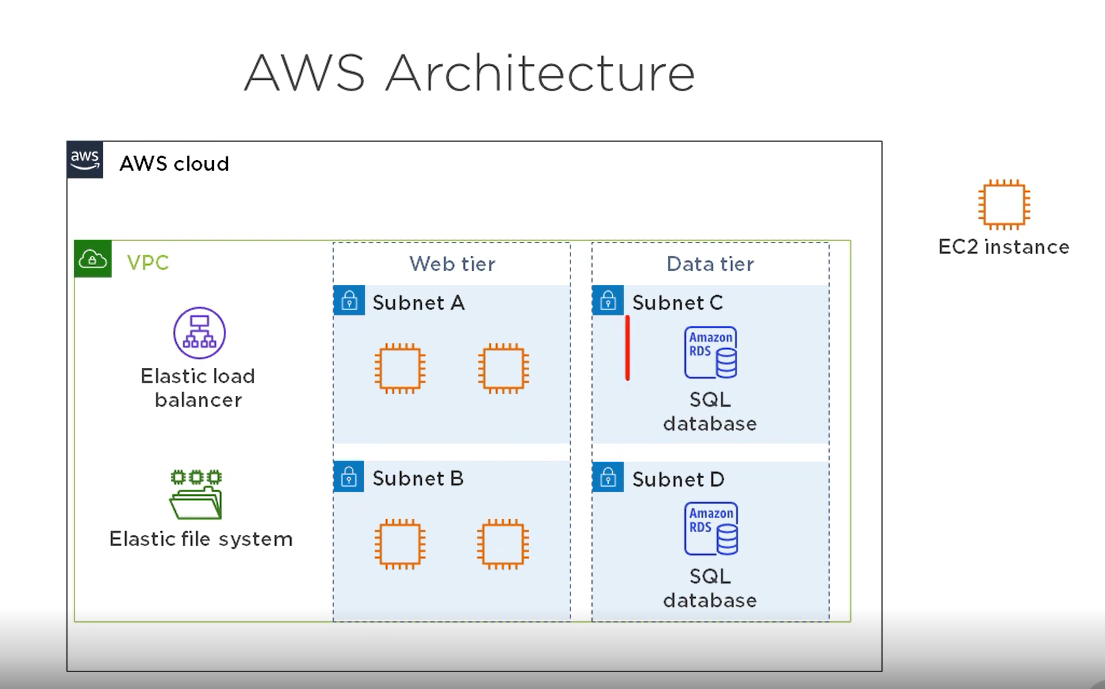
**Architecting for reliability on AWS**

Availability and annual downtime

* 99% - 3 days 15hrs
* 99.9% - 5hrs 45 mins
* 99.99% - about 1 hr
* 99.999% - 5 Mins

**Architecting for 99.9% availability**

Traditional app : Runs on win or linux, you can lift an shift it to aws without changing its code (eg: sql backed web application)



**Calculating Availability - EC2**

The service level agreement (SLA) for each service includes it annual availability. Availability for an eC2 Inst is 90% and failure rate of each inst is 10%

* Calculating avail-redundant instances
* multiply failure rate of 4 inst - 10% \* 10% \* 10% \* 10% = 0.01%
* Subtract product from 100% - 100% - 0.01% = 99.99%

**ELB -** will distribute traffic to the inst while continually monitoring them if 1 inst is down it will stop sending traffic to it.

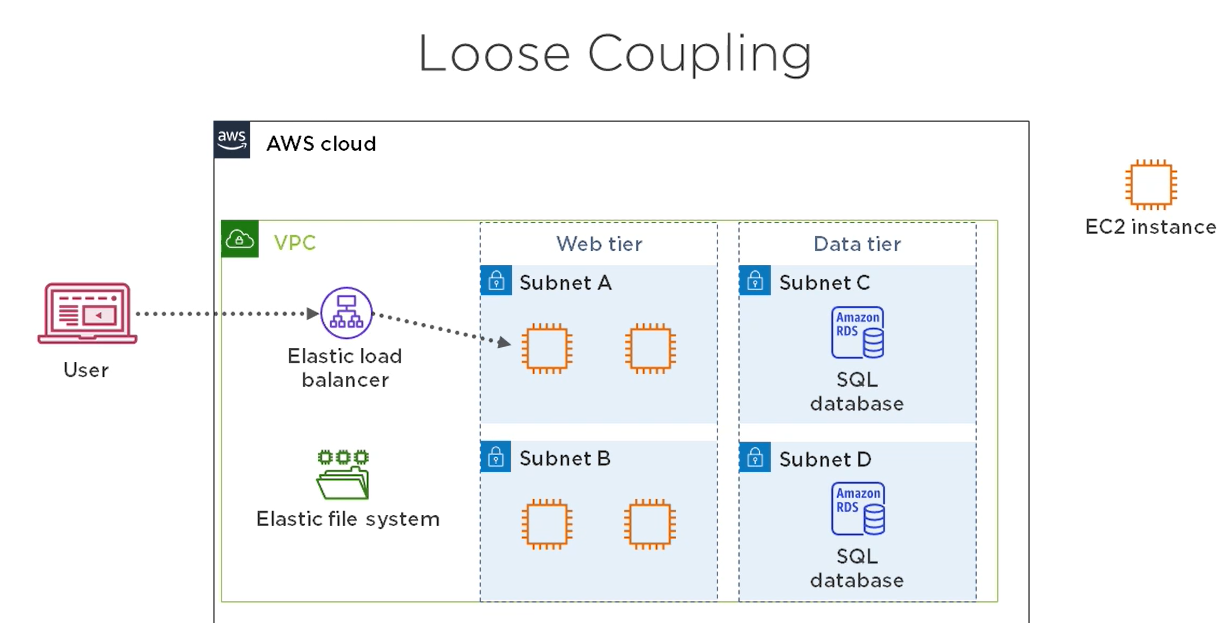
* Calculate avail - ELB
* ELB availability is 99.99%
* Multiply this by the availability of instances - 99.99% \* 99.99% = 99.998%

**EFS - Elastic file system**

* EFS avail is 99.99%
* 99.998 \* 99.99% = 99.97%

RDS - Relational Db service

* RDS multi-AZ availability is 99.95%
* 99.95% \* 99.97% = 99.92%

**Loose Coupling**   




**EFS -** Elastic services are always composed of redundant components. Elastic services are always loosely coupled with other resources

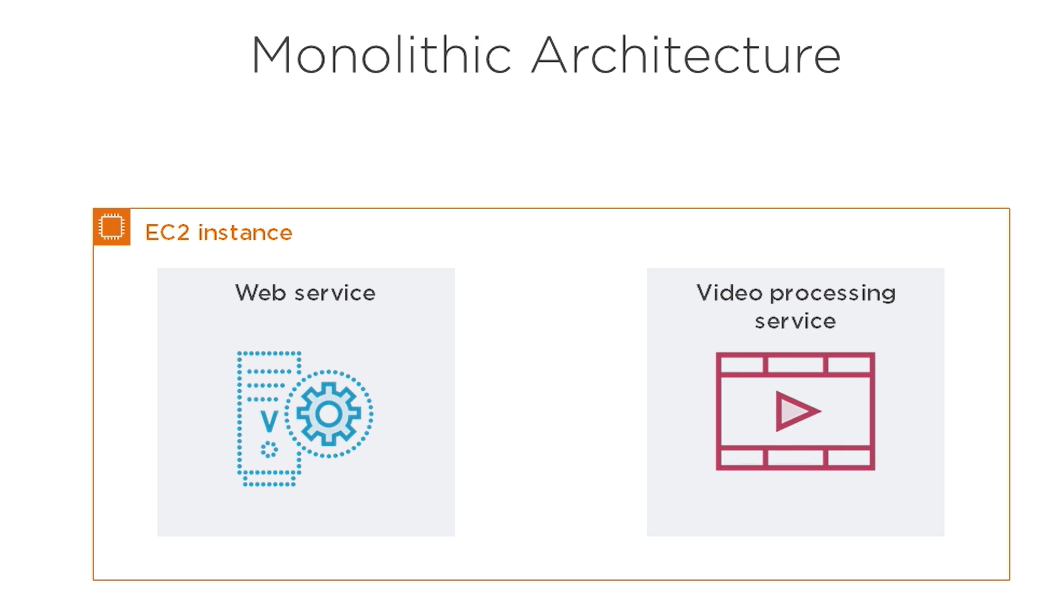
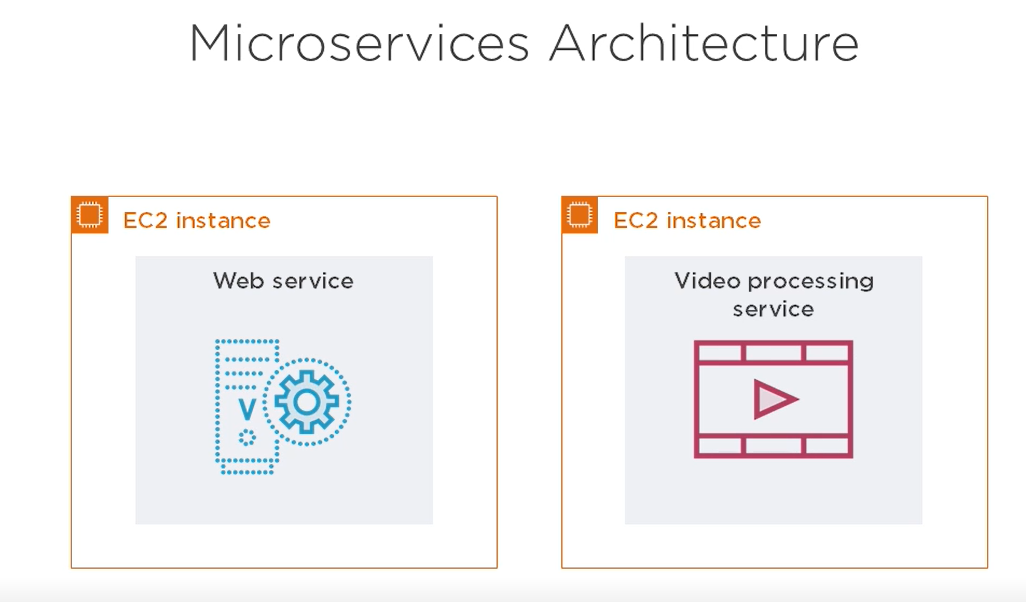
**Redundancy and loose coupling**

* Redundancy --> loose coupling --> availability

**loose coupling and performance :**

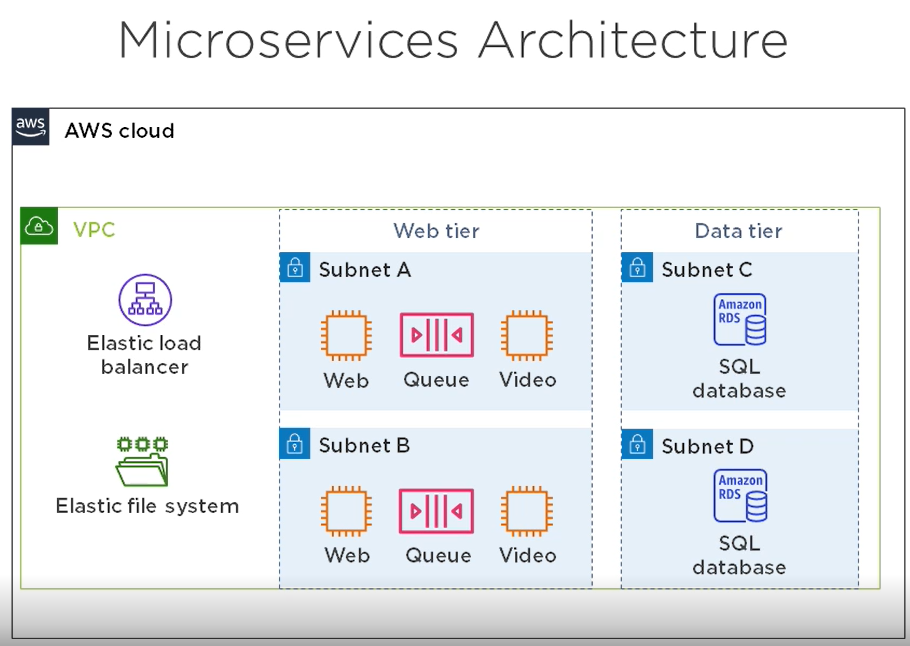
* Independently scale resources without impacting availability
* performance and availability are linked
* poor perform leads to low availability
* good perform helps maintain availability

**Simple Queue Service**

**** ****

**Benefits of micro services** - Easier updates, availability, scalability

**messaging queue using SQS**



**Containers**

* Behave like virtual machines
* Let you run multiple microservices on a single instance
* Processes running inside the container are isolated from the host

**Benefits of containers** - Ease of deployment ,Build an image, deploy it to instances, launch containers from it, Availability

**Docker**

* Image contains everything the service needs
* Launch containers from an image
* Like launching EC2 instances from an AMI

Elastic Container service : ECS clusters contain EC2 instances, can store containers in an elastic container registry(ECR)

**Cloud native applications**

* Depend on a cloud service that can't be deployed on-permises(Eg: SQS, S3, DynamoDB)
* Services instead of servers
* Availability not necessarily higher than traditional applications

**Lambda :**   
Managed serverless compute service,

* The was lambda works is you create a lambda function with you app code, tell lambda the language you app is written in. and simply call the function.
* Lambda supports many lang(C#, java, python, go, PowerShell)
* Availability is 99.95%

**S3** : unlimited file storage, static web hosting, avail is 99.9%

**DynamoDB**

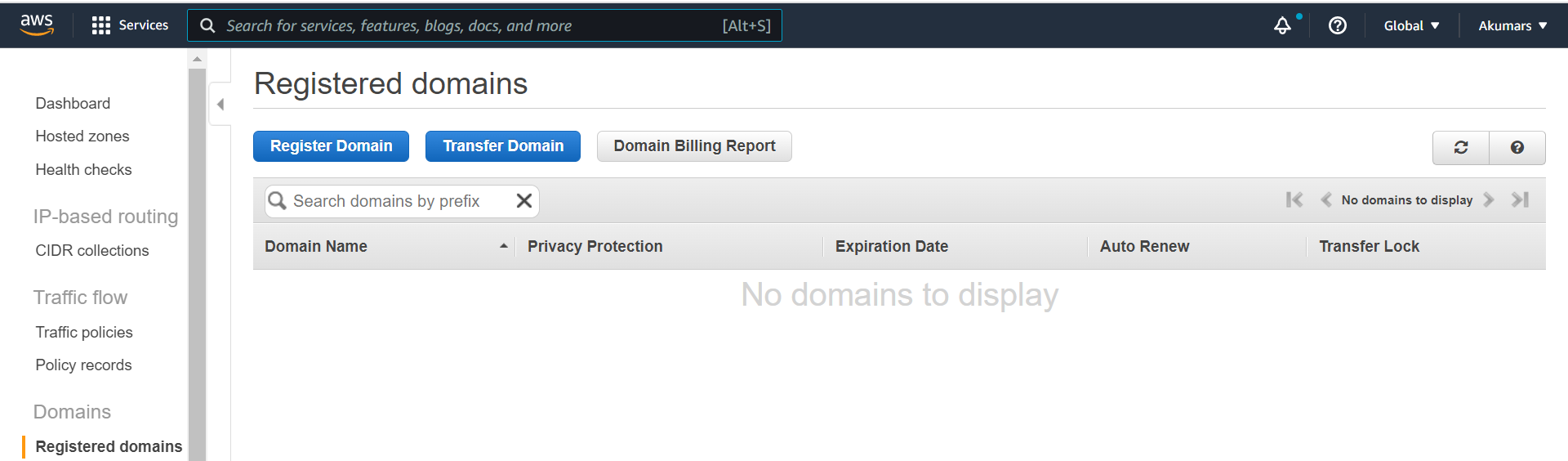
* Managed, nonrelational database
* Global tables feature replicates tables across multiple regions
* Availability with global tables is 99.999%

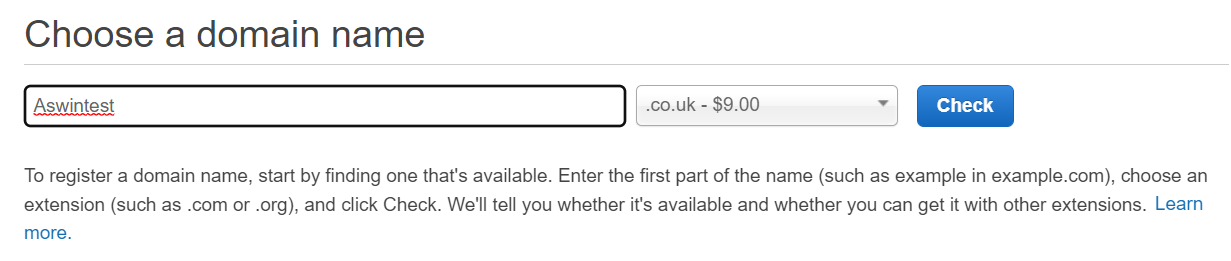
**Calculating Availability** : 99.95% \* 99.9% \* 99.999% = 99.85%

**Amazon Certificate Manager (ACM)**

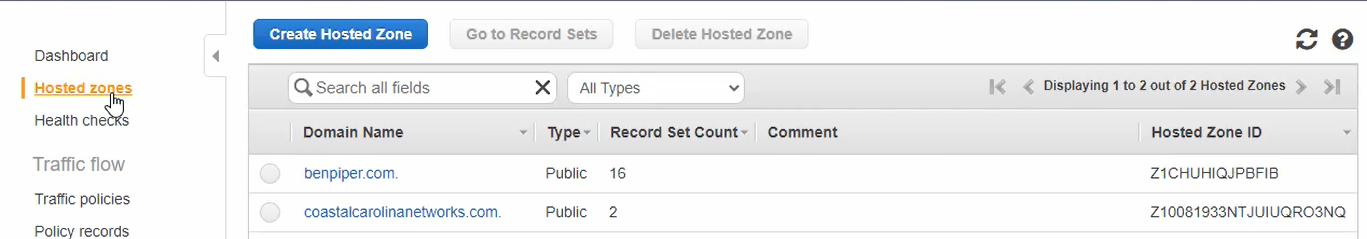
* Issues public TLS certificates for use with AWS services
* Before ACM issues a TLS certificate, you must verify that you control the DNS records for the domain

**Creating a TLS Certificate Using ACM**

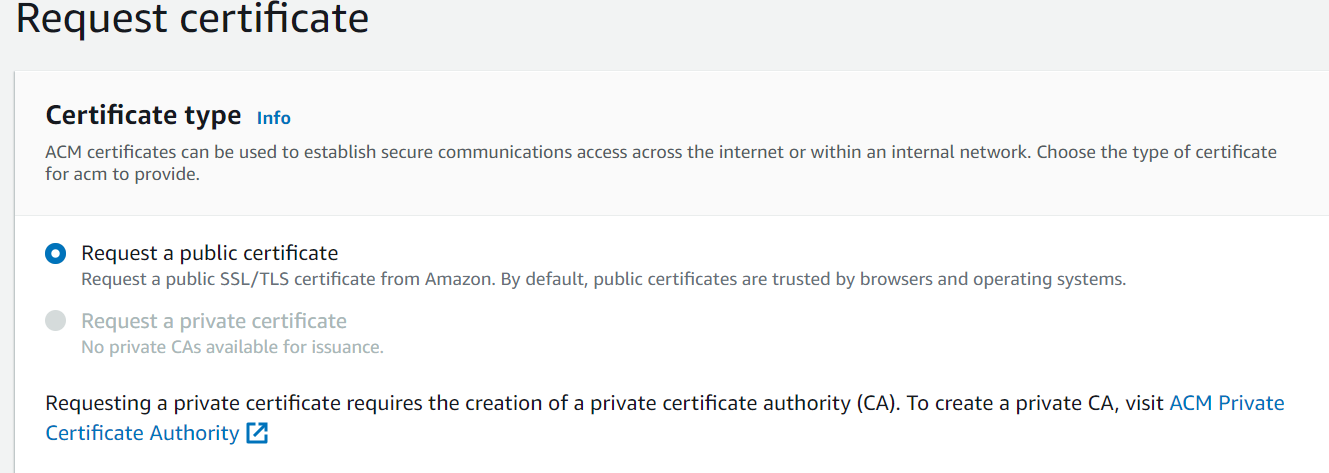
In route 53 Console  


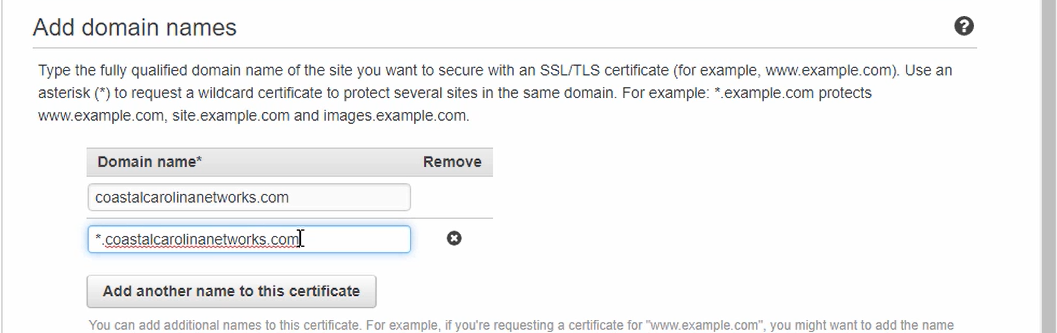


**When a domain is registered in route 53 it creates a public hosted zone.**

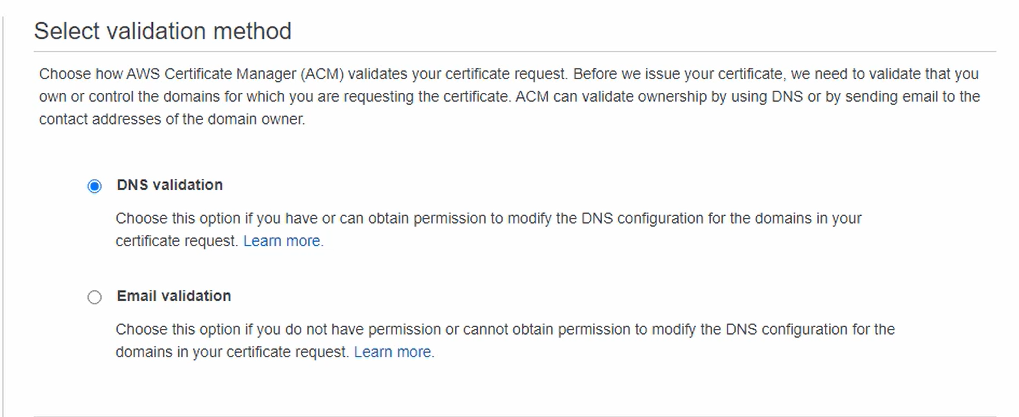


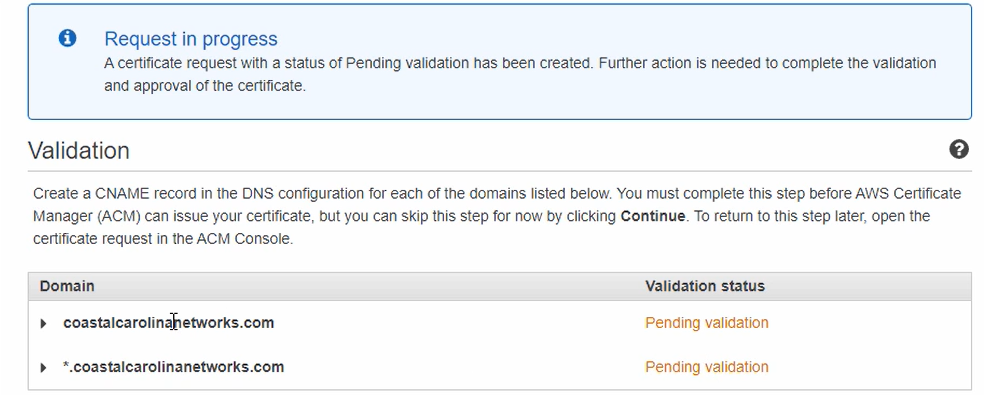
To request TLS certificate switch to ACM console  

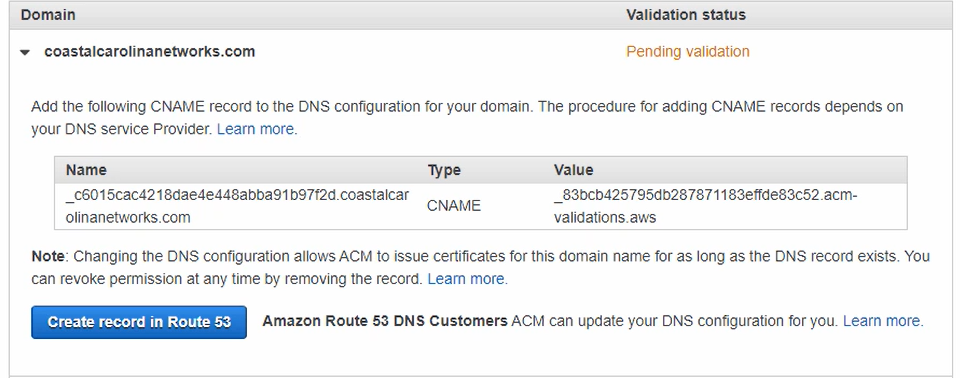

****

****

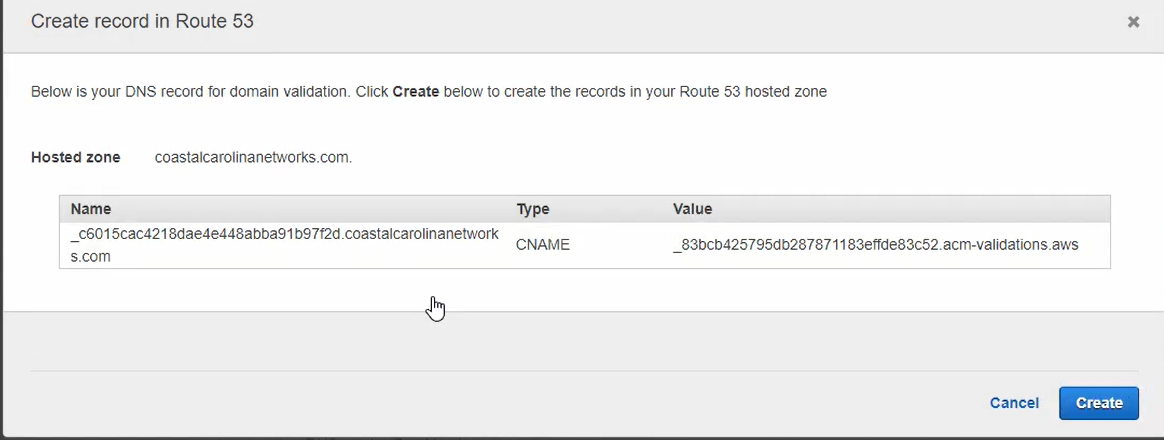
**Add another name ; ''\*" followed by domain name asterisk is going to make this a wildcard certificate so it can be vaild for any subdomain like www.**

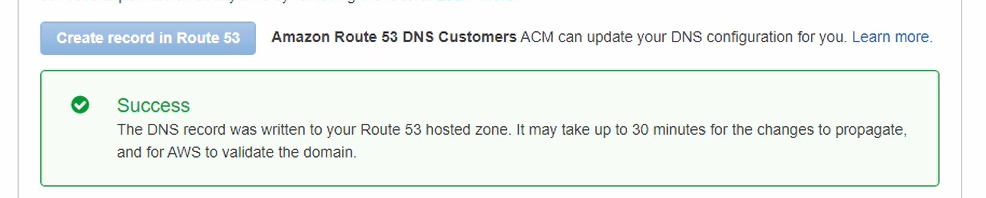
****

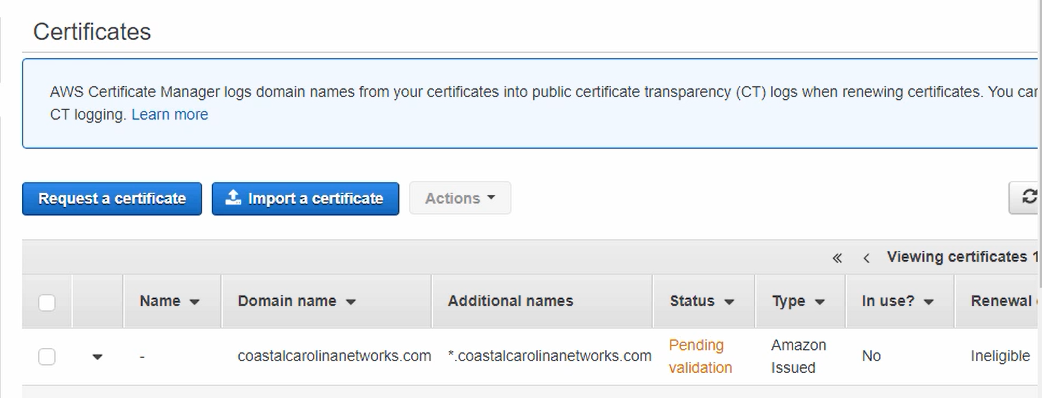
****

Expand the domain  


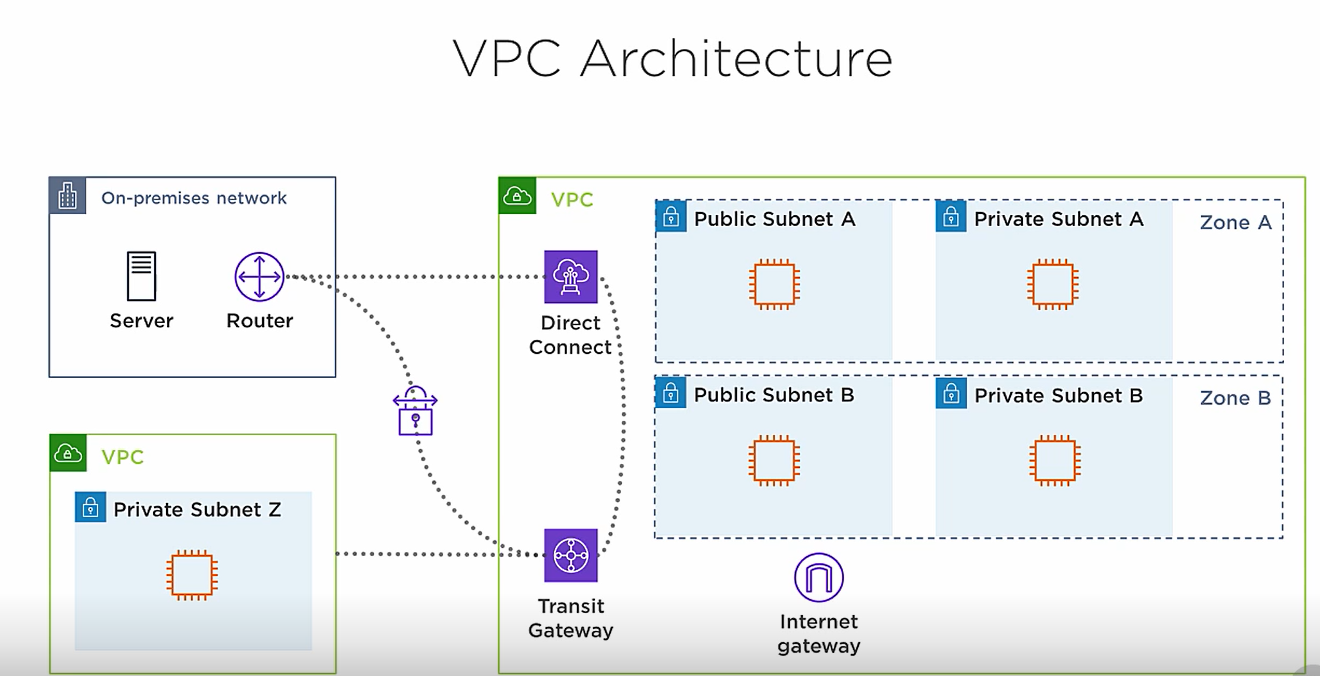
And select Create record in route 53



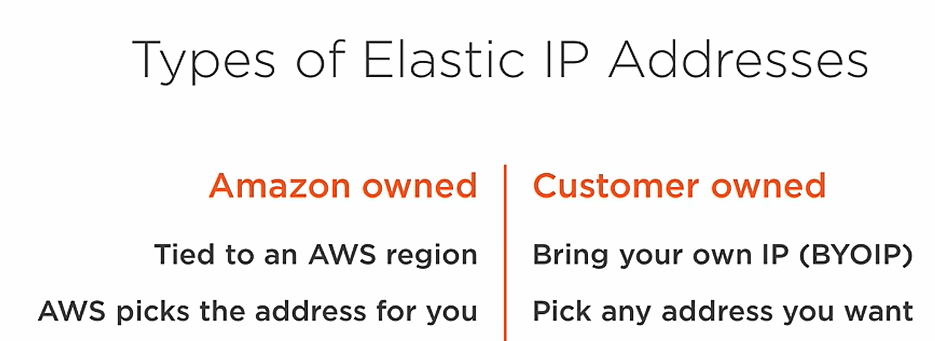




**VPC**

****

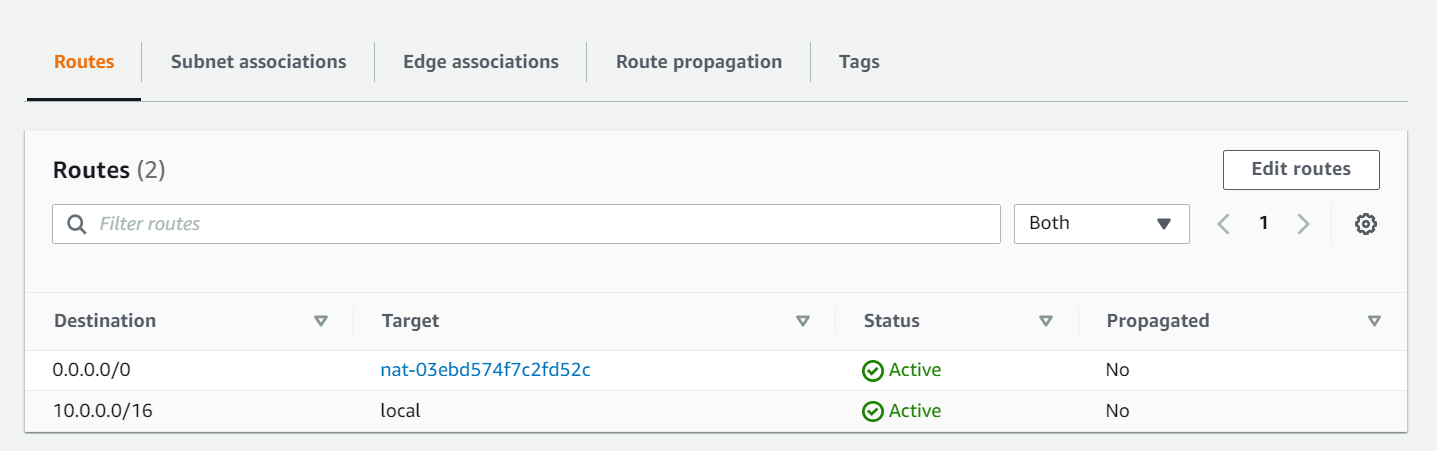
**Elastic Ip address**

* You allocate an EIP to your account and keep it as long as you want
* EIP allows an instance to retain the same public IP address
* EIP is bound to an ENI, which is attached to an instance
* You can move an EIP to different ENI ****

Inst in the private subnet are completely isolated from the internet. They cannot reach the internet and the internet cant reach them

This is where the NAT gateway comes in, it provides instances in the private subnet outbound only internet access.

Create a VPC with Public and private Subnet  
Tag private subnet with NAT Gateway



Instance in private subnet sens internet bound traffic to NAT gateway, Nat gateway sends traffic to internet gateway

To create Subnet in different Zone in the Subnet console select create Subnet

Configure AWS CLI   
$ aws configure

AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE

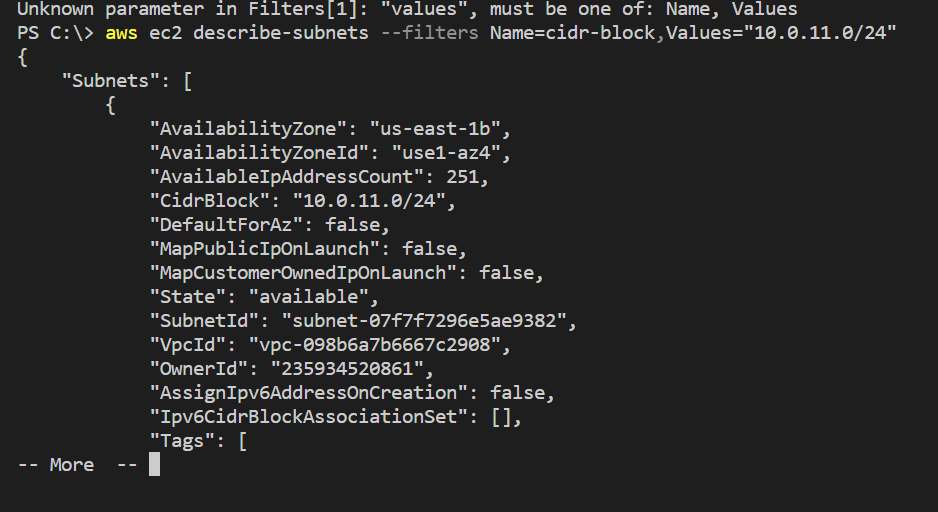
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

Default region name [None]: us-west-2

Default output format [None]: json

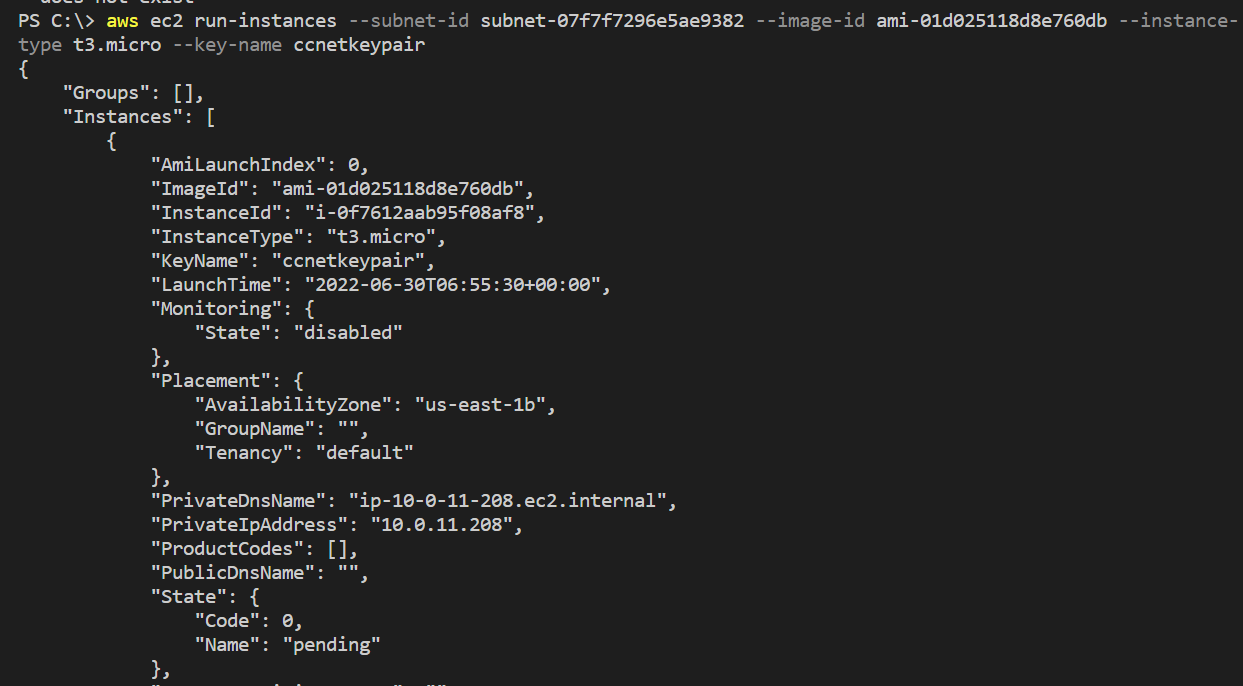
**Launching an Instance into a Public Subnet**To launch instances into public subnet B, Need subnet identifier

* **aws ec2 describe-subnets --filters Name=cidr-block, values="10.0.11.0/24"**

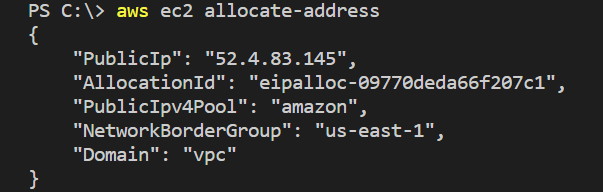
****

**Copy the subnet ID from subnet identifier  
  
launch a inst in that subnet**

* **aws ec2 run-instances --subnet-id subnet-07f7f7296e5ae9382 --image-id ami-01d025118d8e760db --instance-type t3.micro --key-name ccnetkeypair**

****

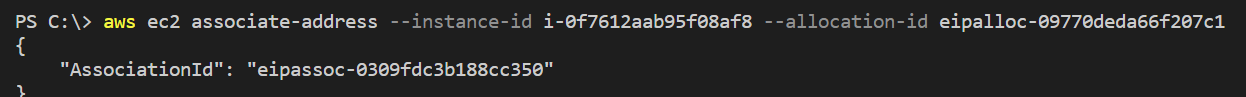
**Enter :🡪 aws ec2 allocate-address**



**Copy the instance ID : i-0f7612aab95f08af8**

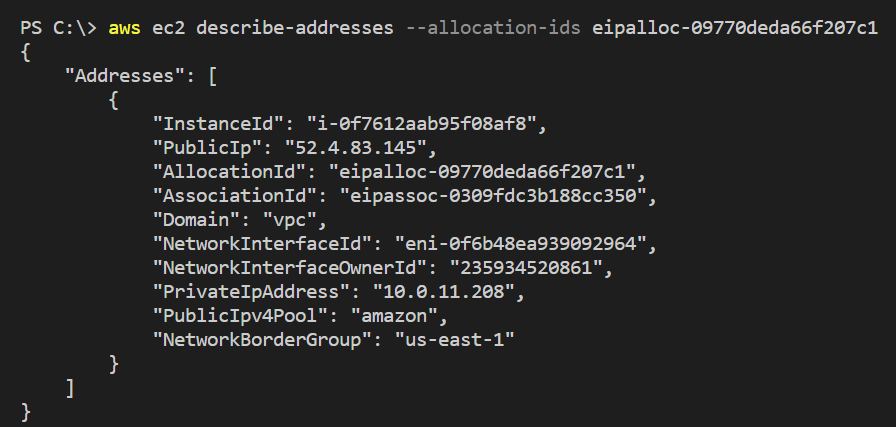
**Allocate another elastic IP address and associate to this instance**

* **aws ec2 associate-address --instance-id i-0f7612aab95f08af8 --allocation-id eipalloc-09770deda66f207c1**

****

**To view the Elastic ip associate to the inst**

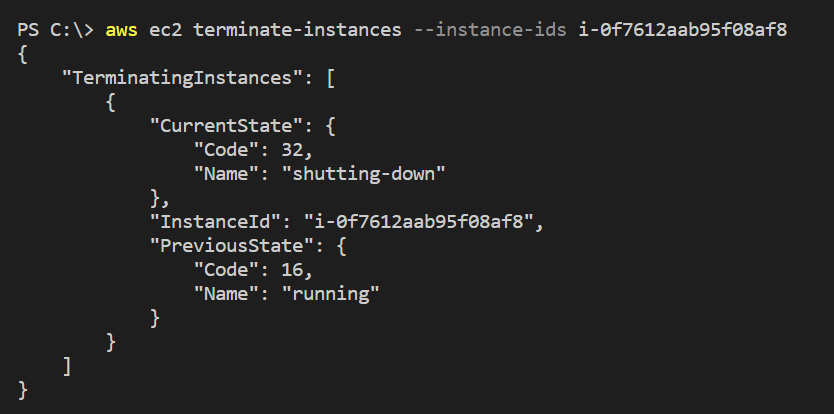
* **aws ec2 describe-addresses --allocation-ids eipalloc-09770deda66f207c1**

****

**AWS shield standard : Free service that detects against distributed denial of service(DDOS) attacks**

**To Terminate the Instance**

* **aws ec2 terminate-instances --instance-ids i-0f7612aab95f08af8**

****

**Release elastic IP**

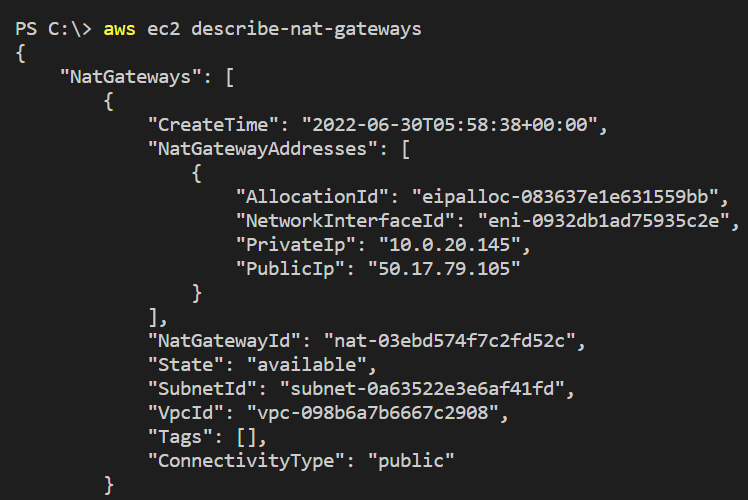
* **aws ec2 release-address --allocation-id eipalloc-09770deda66f207c1**

**Launching an Instance into a Private Subnet**

* **aws ec2 run-instances --image-id ami-01d025118d82760db --subnet-id [private-subnet-id] --instance-type t3.micro**

**List the NAT Gateways**

* **aws ec2 describe-nat-gateways**

****

**To Delete Nat gateway**

* **aws ec2 delete-nat-gateway --nat-gateway-id nat-03ebd574f7c2fd52c**

****

**Release Elastic IP**

* **aws ec2 release-address --allocation-id eipalloc-083637e1e631559bb**

****

**Connectivity Options**

Direct Connect Low latency connection to an AWS region. Bypass the internet

**Two types :** Dedicated, Hosted

* Direct connect dedicated connection :Physical connection hat terminates at a direct connect location. 1 or 10 Gbps
* Direct connect Hosted connection : Last-mile connection provided by a direct connect partner. 50 Mbps to 10 Gbps

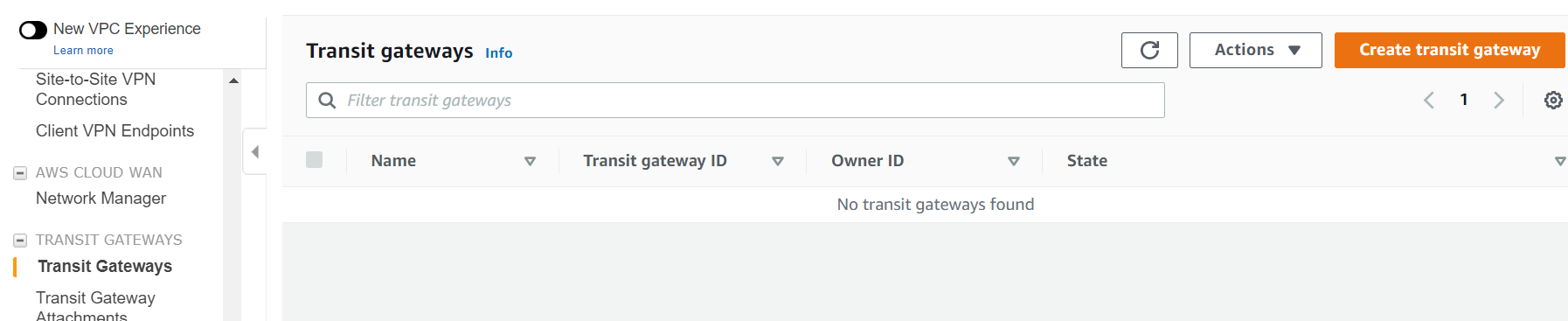
**Virtual private network :** Encrypted IPsec connection over the internet. Unpredictable latency.

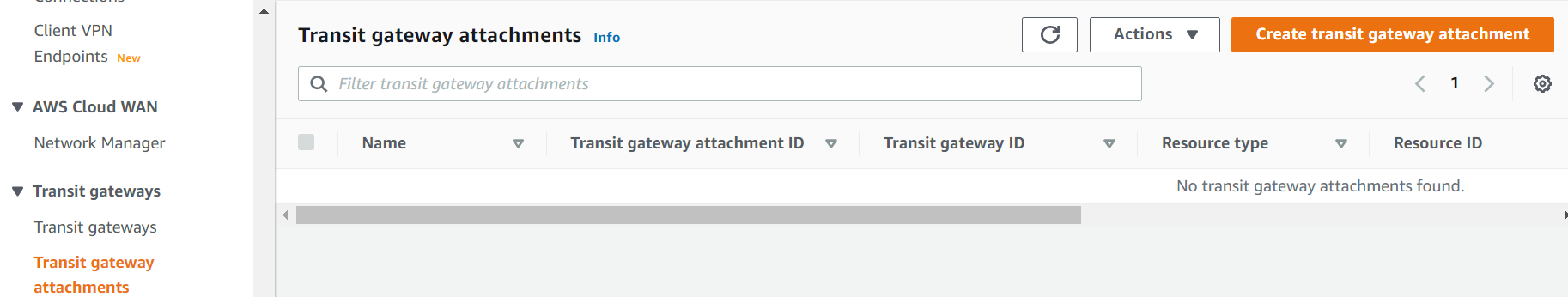
Two ways : Virtual private gateway, transit gateway

Virtual Private gateway : Enabled you to establish a VPN tunnel with only one VPC, Doesn't scale well

Transit gateway : Connect VPCs and on-premises networks, connects multiple VPCs together

**Connect VPCs using transit gateway**

****

****

**Automated Deployments with CloudFormation**

Cloud formation

**Template :**

* Jason or YAML document that describes AWS resources.
* Infrastructure as code
* used to create a stack

**Stack :**

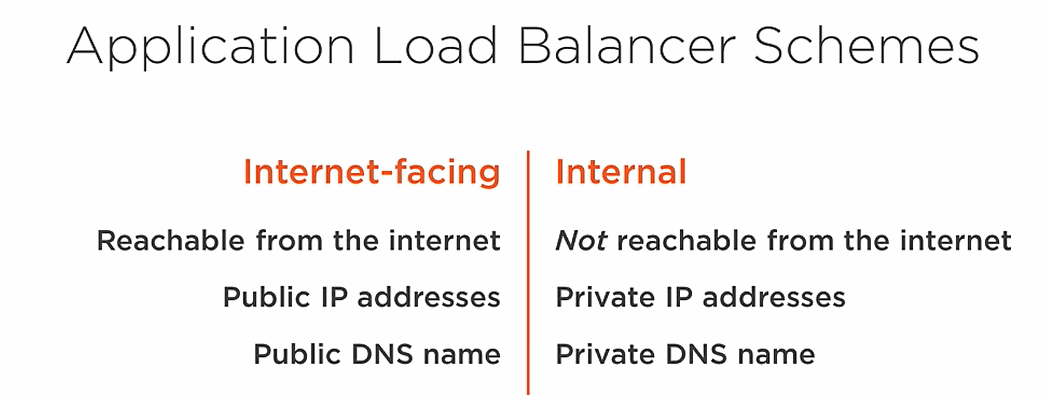
* Created by a template
* Collection of resources that you create, update, and delete as a single unit
* You can manually manage individual resources in a stack

**Managing Template**

* Different teams manage different resources
* Resources have different lifecycles
* Distributing resources across different stacks makes them easier to manage

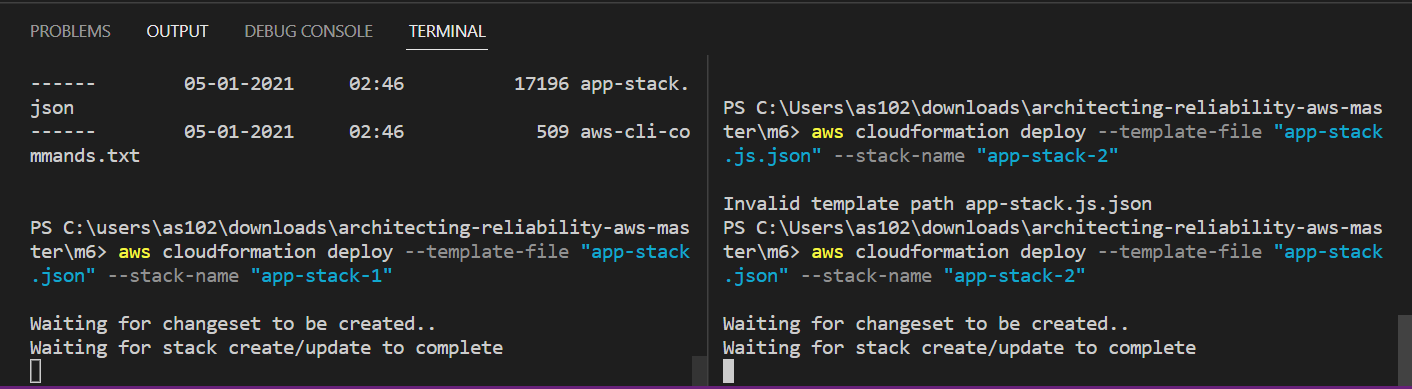
**Application Load Balancers**

* Supports HTTP and HTTPS traffic
* You can use any TCP port
* Listener receives connection from a client and proxies it to an instance in the target group
* Uses round-robin load balancing by default Can monitor health of instances

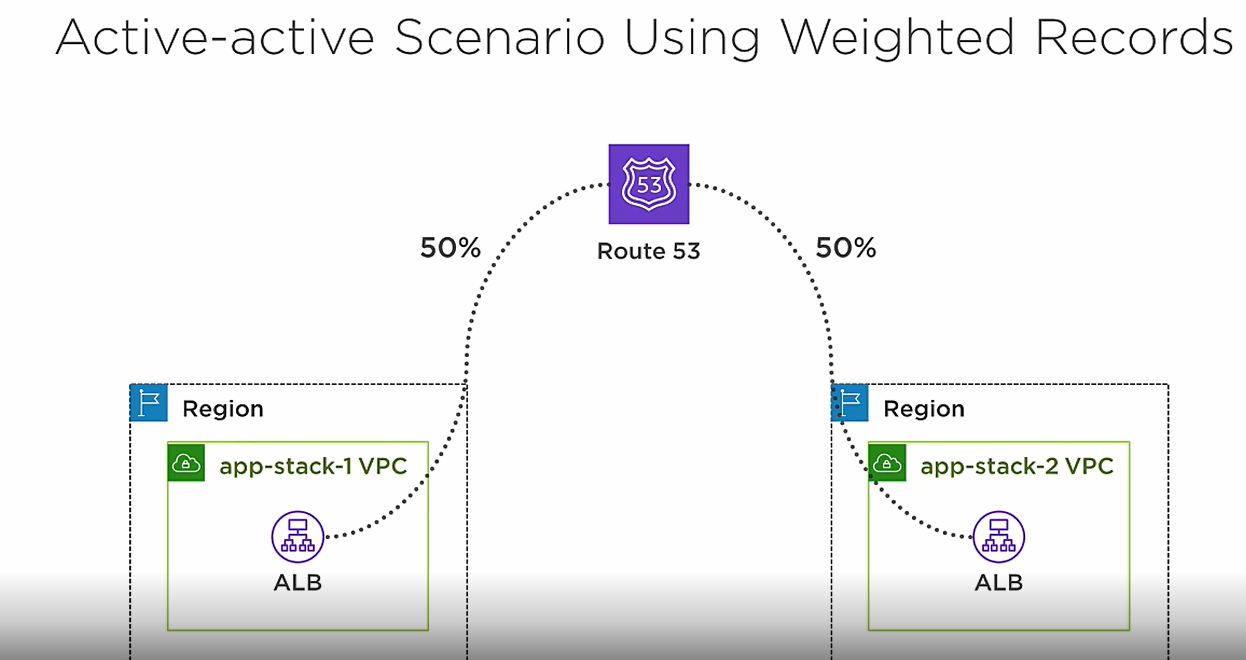


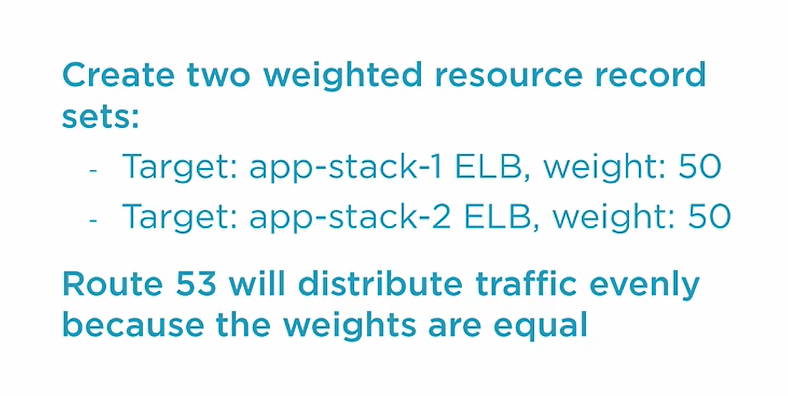
Deploying a Multi-region Application Using app stack

* **aws cloudformation deploy --template-file "app-stack.json" --stack-name "app-stack-1"**

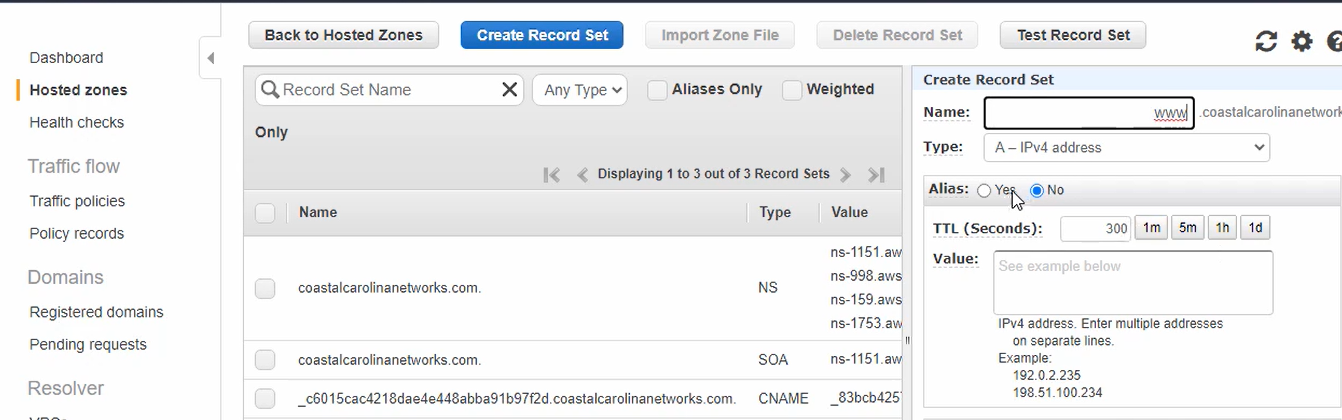
****

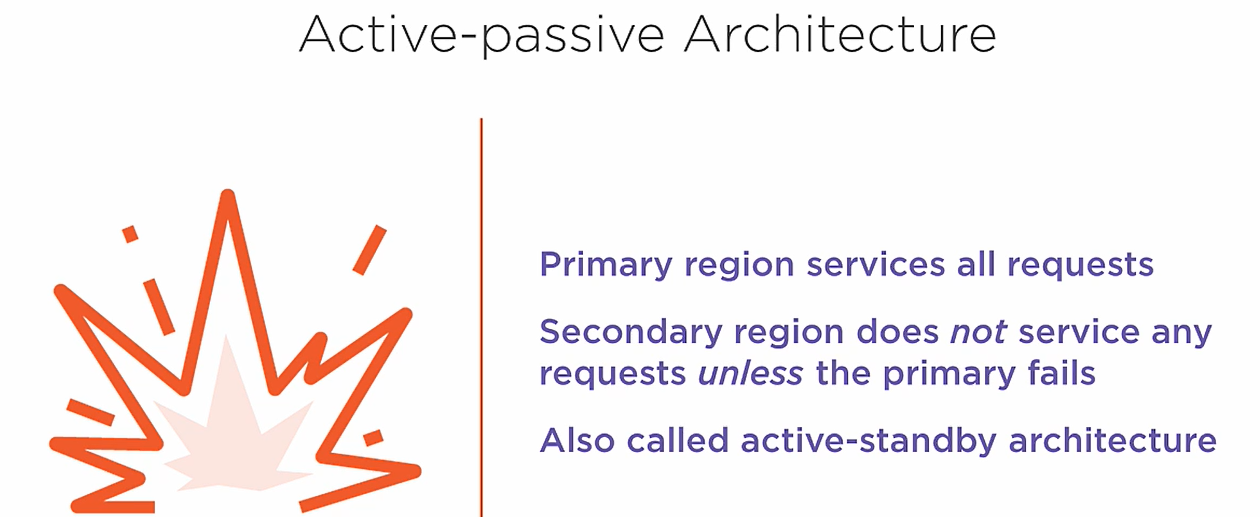
**Active-active Redundancy Using Weighted Resource Records**

****

**🡨 DEMO**

Route 53

****

****