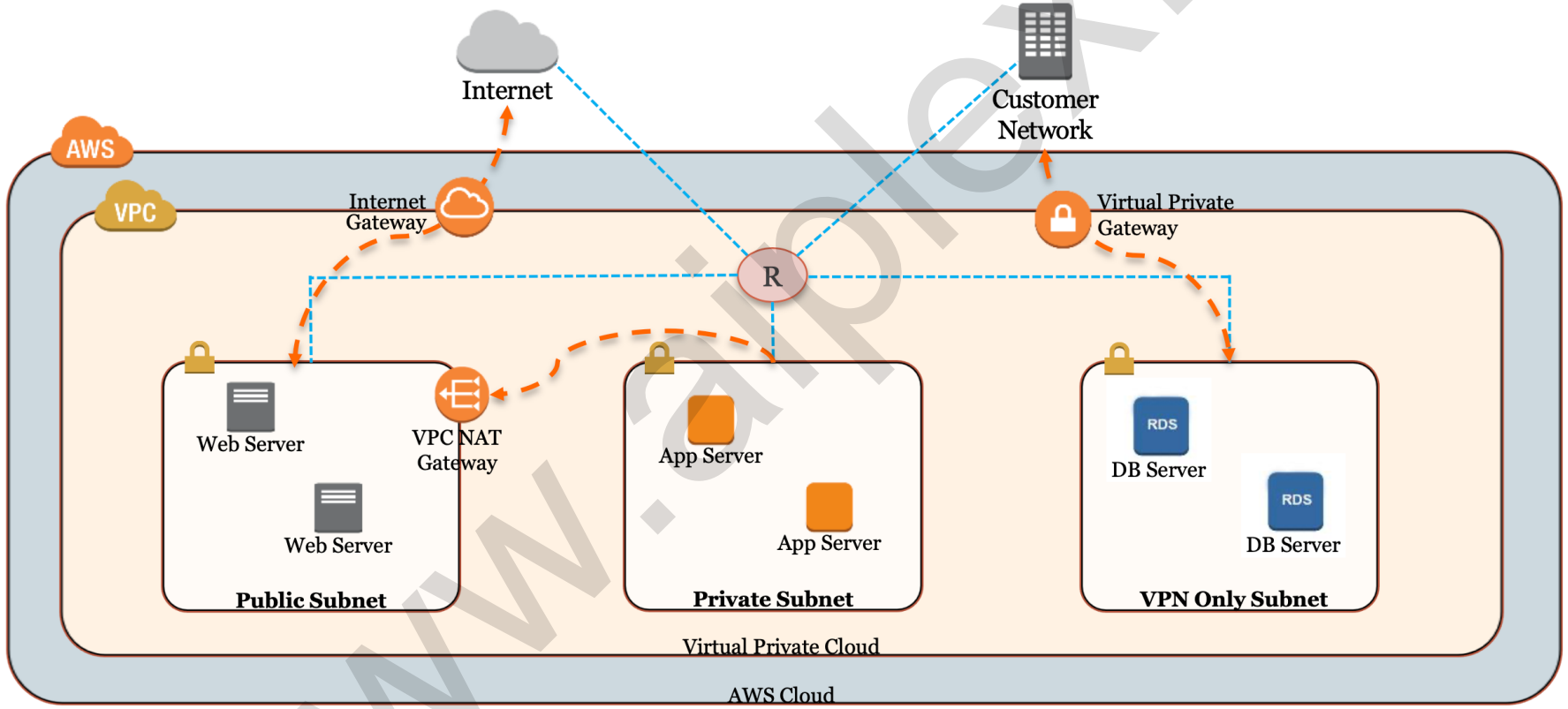
AWS SERVICES

DO FOLLOW ME ON
INSTAGRAM/
TWITTER/
TELEGRAM

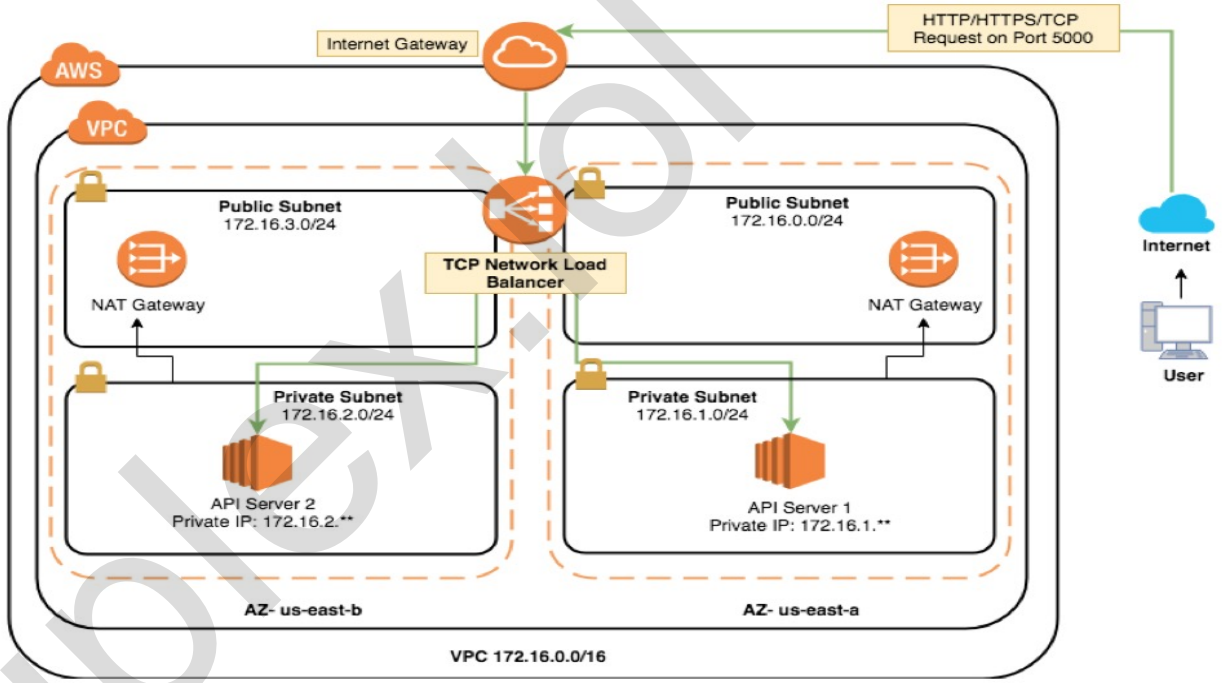
SINGAM4DEVOPS

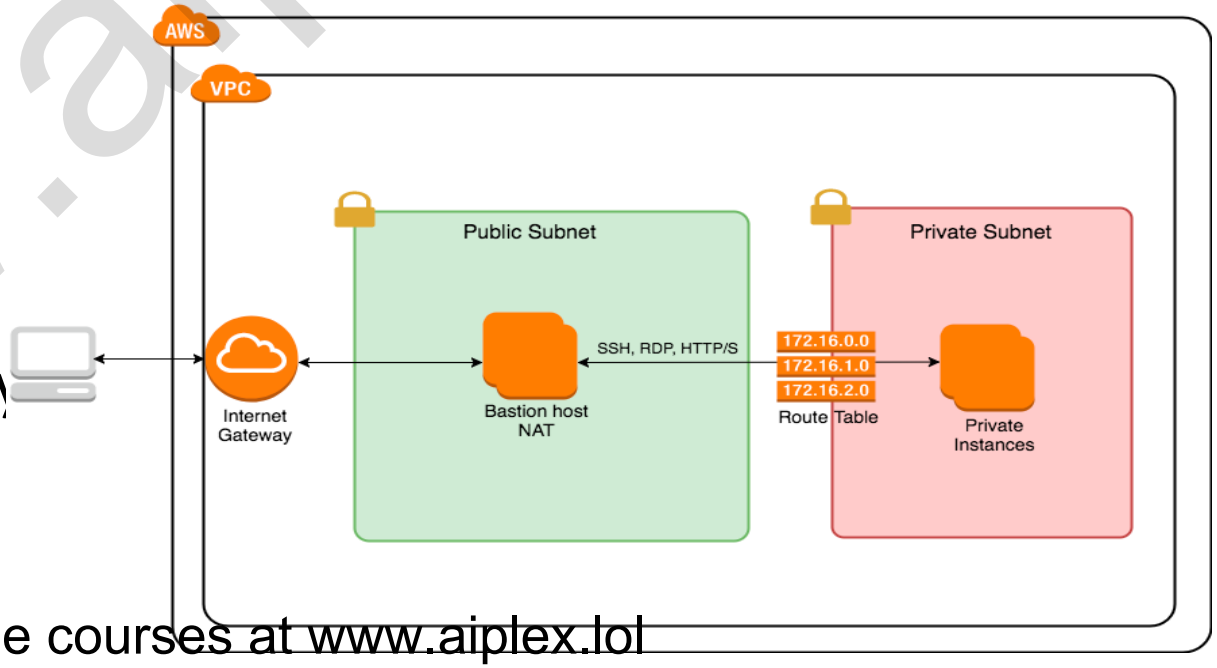
VPC



SUBNETS

- **Public subnets**
- Include a routing table entry to an Internet gateway to support
- inbound/outbound access to the public Internet.
- **Private subnets**
- Do not have a routing table entry to an Internet gateway and are not directly accessible from the public Internet.
- Typically use a "jump box" (NAT/proxy/bastion host) to support restricted, outbound-only public Internet access.

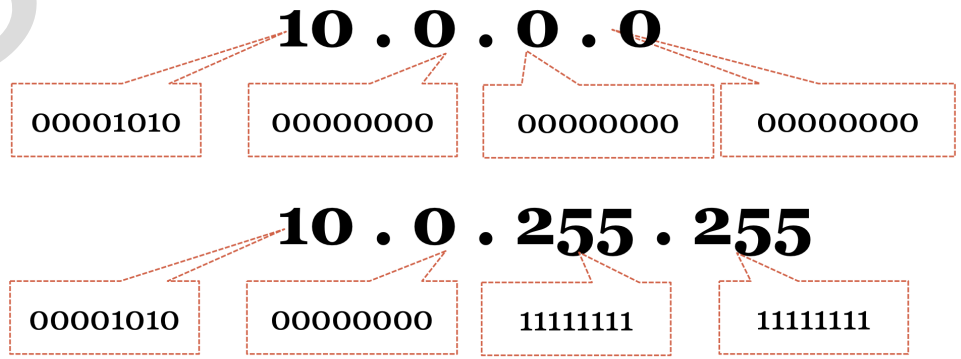




CIDR SETUP

- When you create your VPC, you specify its set of IP addresses with CIDR notation
- **Classless Inter-Domain Routing** (CIDR) notation is a simplified way to show a specific range of IP addresses
- Example: 10.0.0.0/16 = all IPs from 10.0.0.0 to 10.0.255.255

Every set of 4 digits in an IP address represents a set of 8 binary values (8 bits).



- 255.255.0.0=11111111 11111111 00000000 00000000
  255.255.254.0 = 11111111 11111111 11111110 00000000

- 16 bits that can have a maximum value of 1111111111111111 =65,535

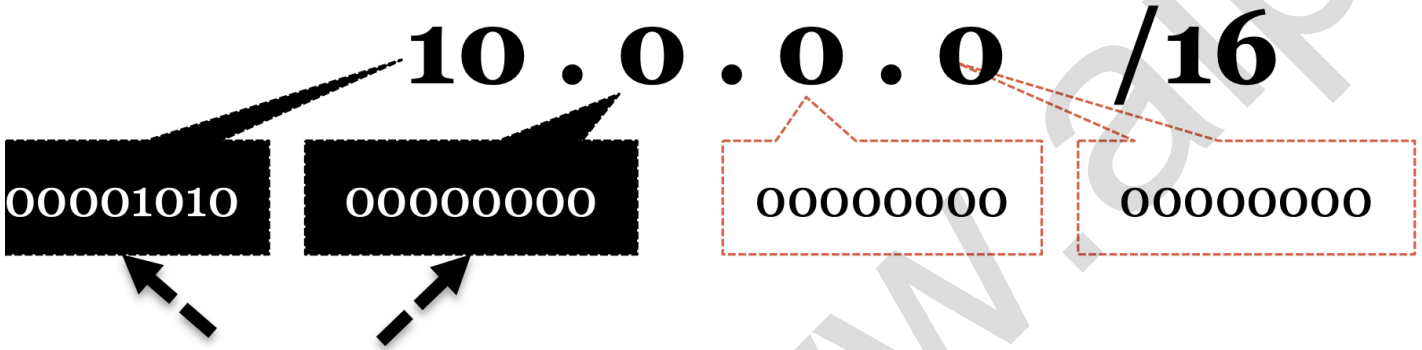
Short Form	Full Form	Maximum #Machines
/8	/255.0.0.0	16,777,215
/16	/255.255.0.0	65,535

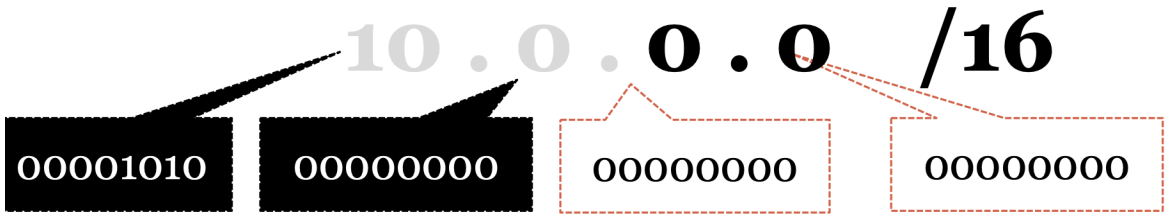
We are fixing the place value /8 means one value /16 means two places

/16 ALL YOU HAVE TO KNOW ABOUT

- The 16 in the CIDR notation example represents how many of those bits are "locked down" and cannot change.

The unlocked bits can change between 1 and 0, allowing the full range of possible values.

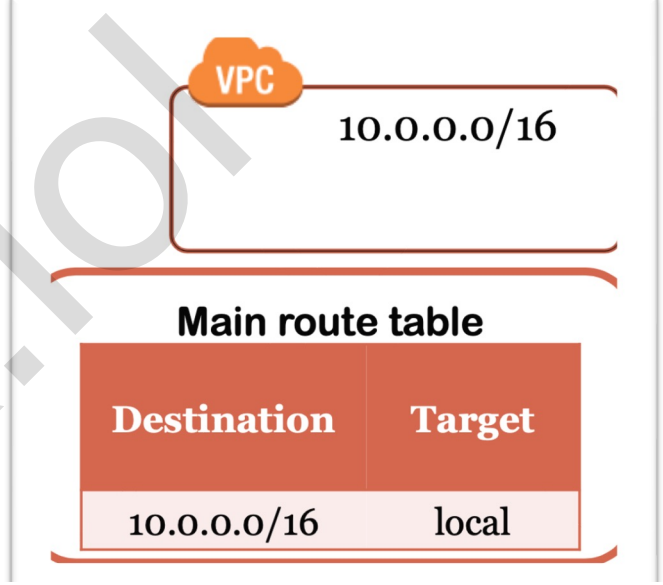


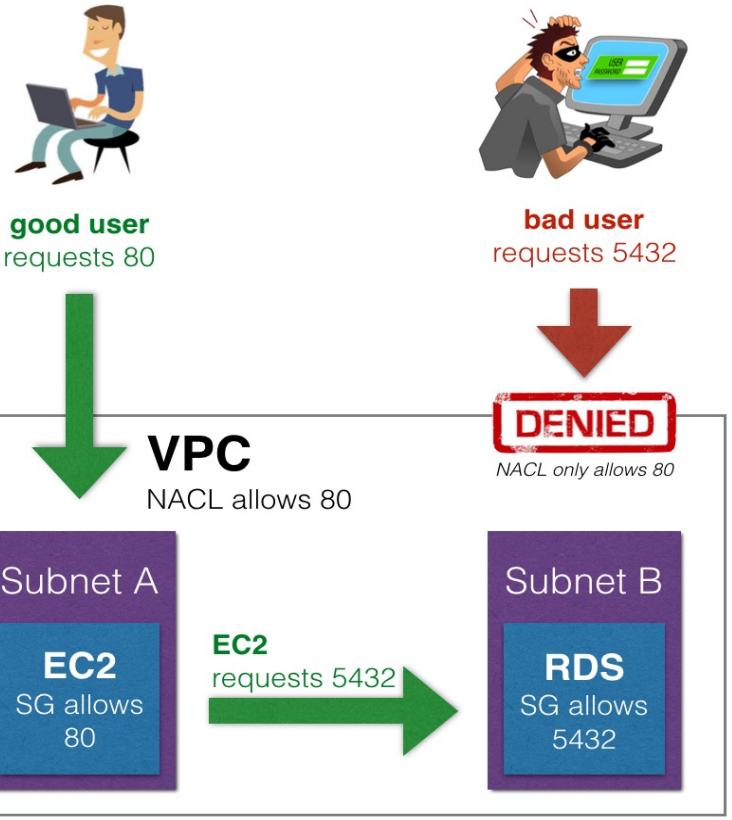


Directing Traffic Between VPC Resources

Security Group	Network Access Control List
Acts as a firewall for associated Amazon EC2 instances	Acts as a firewall for associated subnets
Controls both inbound and outbound traffic at the instance level	Controls both inbound and outbound traffic at the subnet level
You can secure your VPC instances using only security groups	Network ACLs are an additional layer of defense.
Supports allow rules only	Supports allow rules and deny rules
Stateful (Return traffic is automatically allowed, regardless of any rules)	Stateless (Return traffic must be explicitly allowed by rules)
Evaluates all rules before deciding whether to allow traffic	Evaluates rules in number order when deciding whether to allow traffic, starting with the lowest numbered rule.
Applies only to the instance that is associated to it	Applies to all instances in the subnet it is associated with
Has separate rules for inbound and outbound traffic	Has separate rules for inbound and outbound traffic
A newly created security group denies all inbound traffic by default	A newly created nACL denies all inbound traffic by default
A newly created security group has an outbound rule that allows all outbound traffic by default	A newly created nACL denies all outbound traffic default
Instances associated with a security group can't talk to each other unless you add rules allowing it	Each subnet in your VPC must be associated with a network ACL. If none is associated, the default nACL is selected.
Security groups are associated with network interfaces	You can associate a network ACL with multiple subnets; however, a subnet can be associated with only one network ACL at a time.

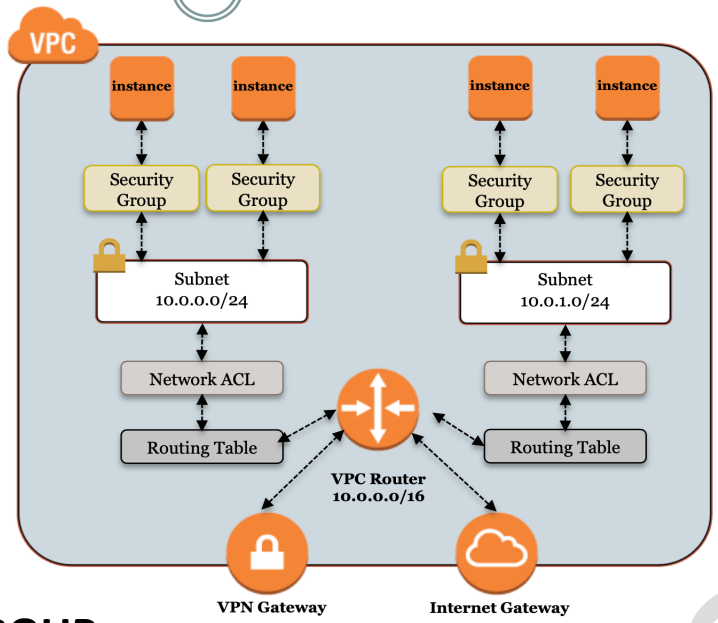
Tutorials Dojo



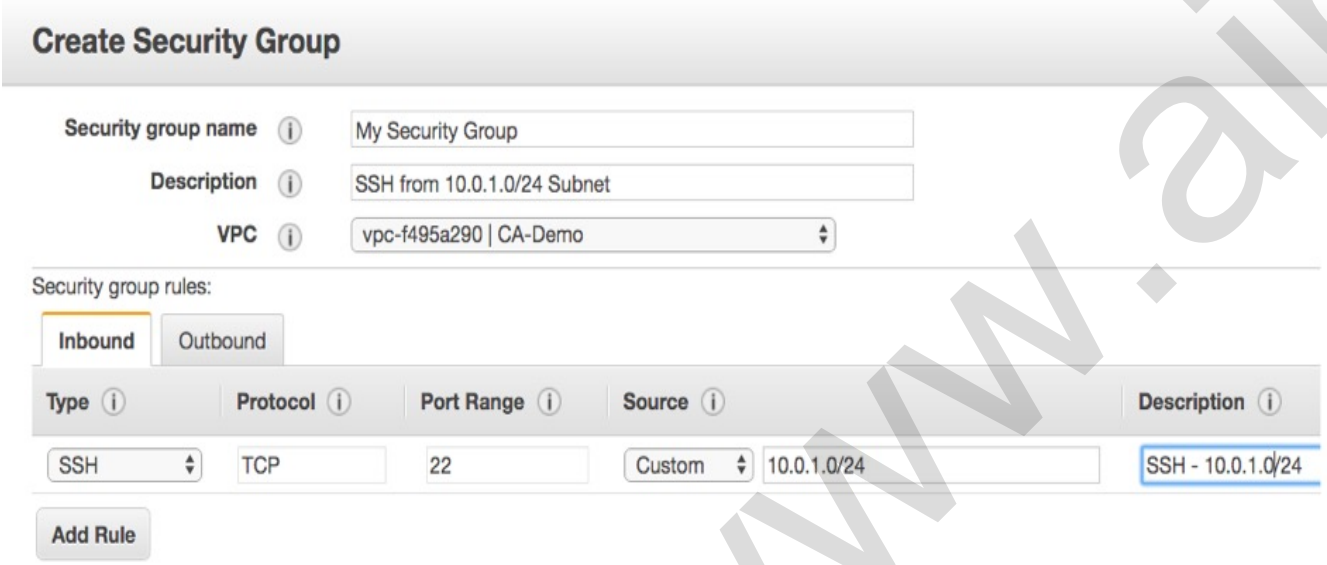


Security Layer of AWS

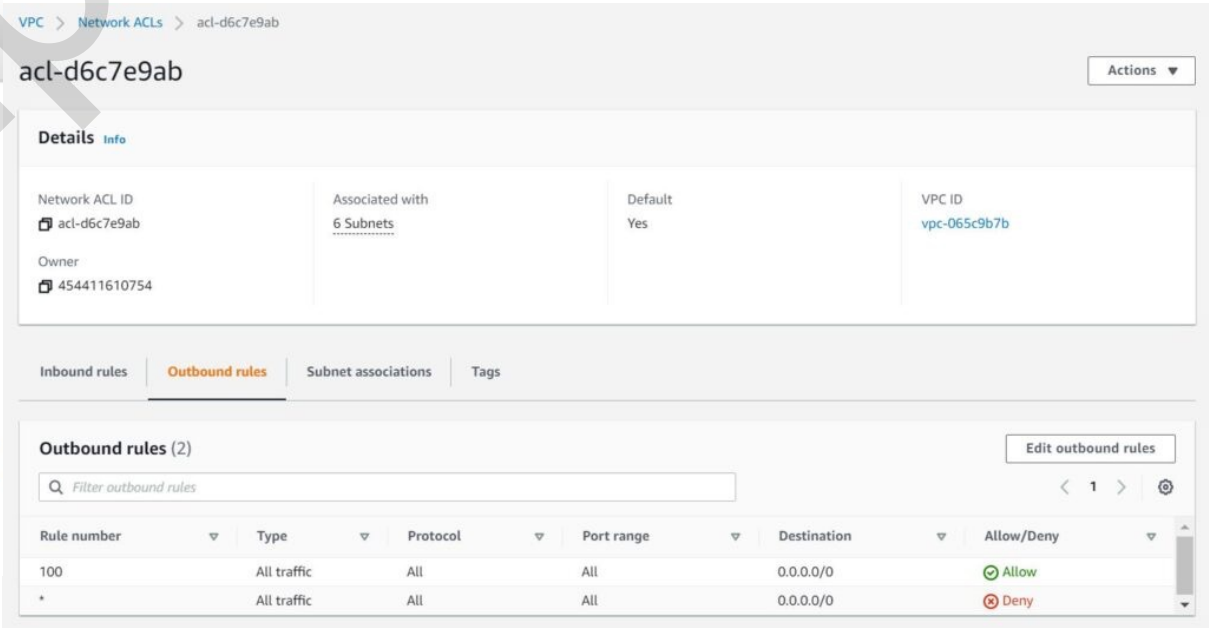
- Security groups
- Network access control lists (ACLs)
- Key Pairs



SECURITY GROUP

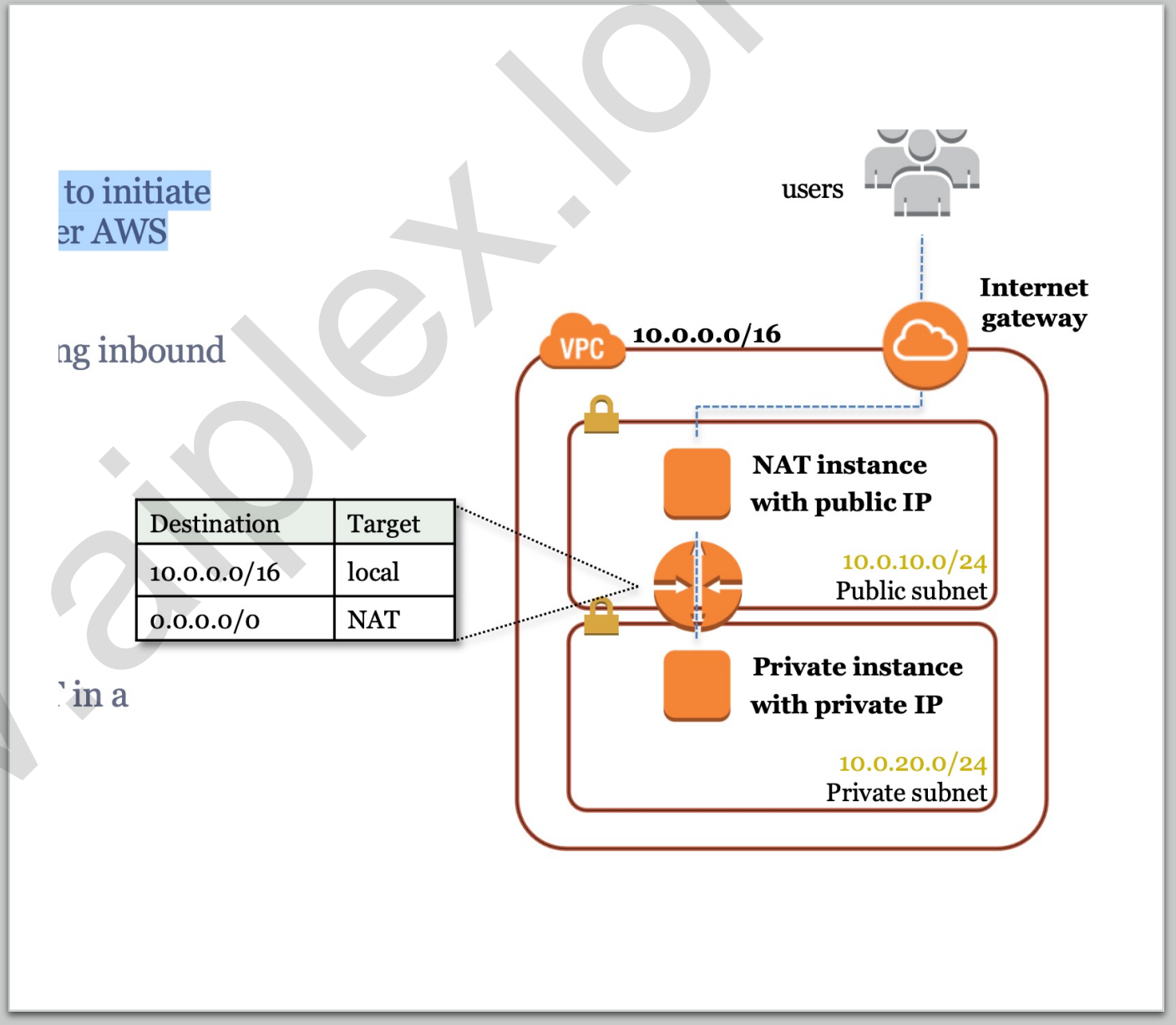


NACL

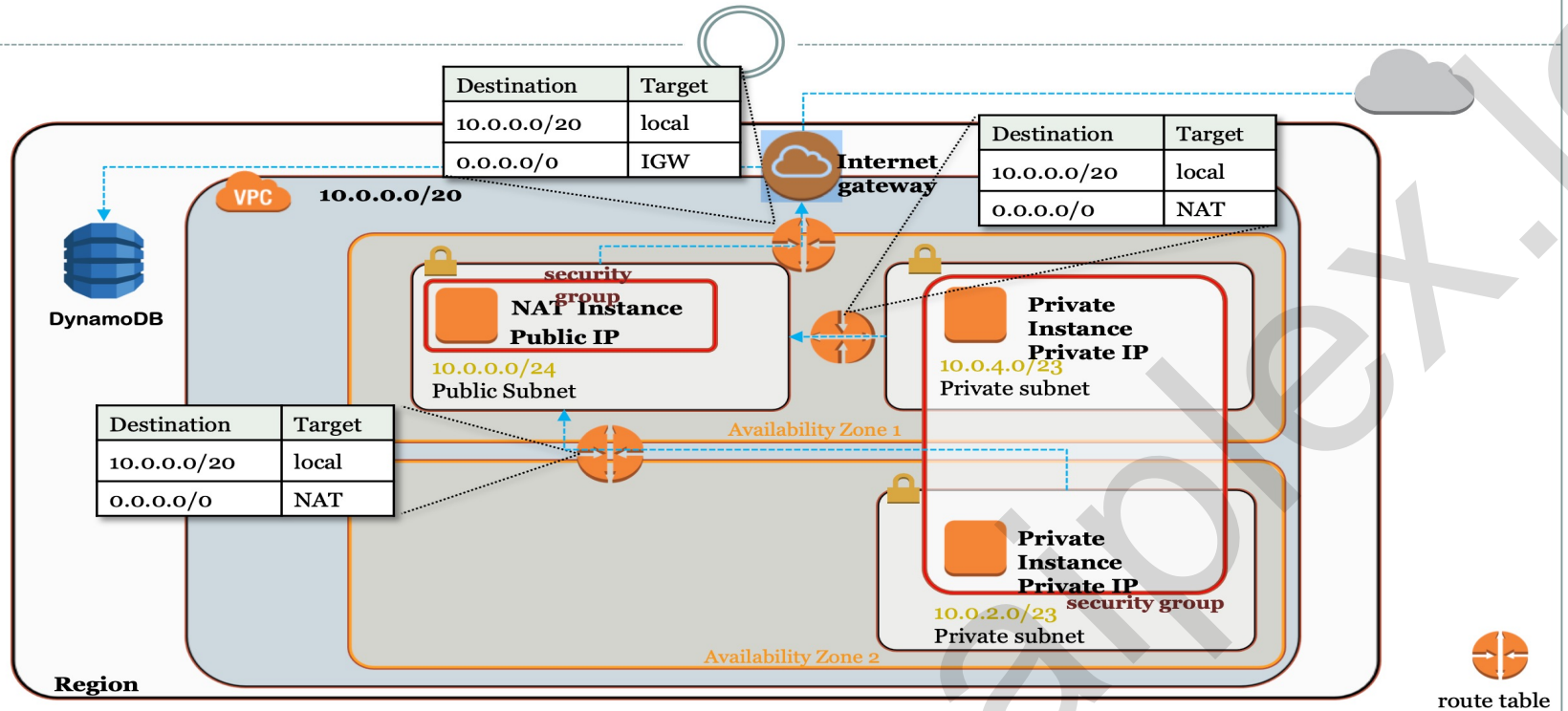


Network Address Translation services

- Enable instances in the private subnet to initiate outbound traffic to the Internet or other AWS services.



Subnets, Gateways, and Routes



The destination for the route is 0.0.0.0/0 , which represents all IPv4 addresses.

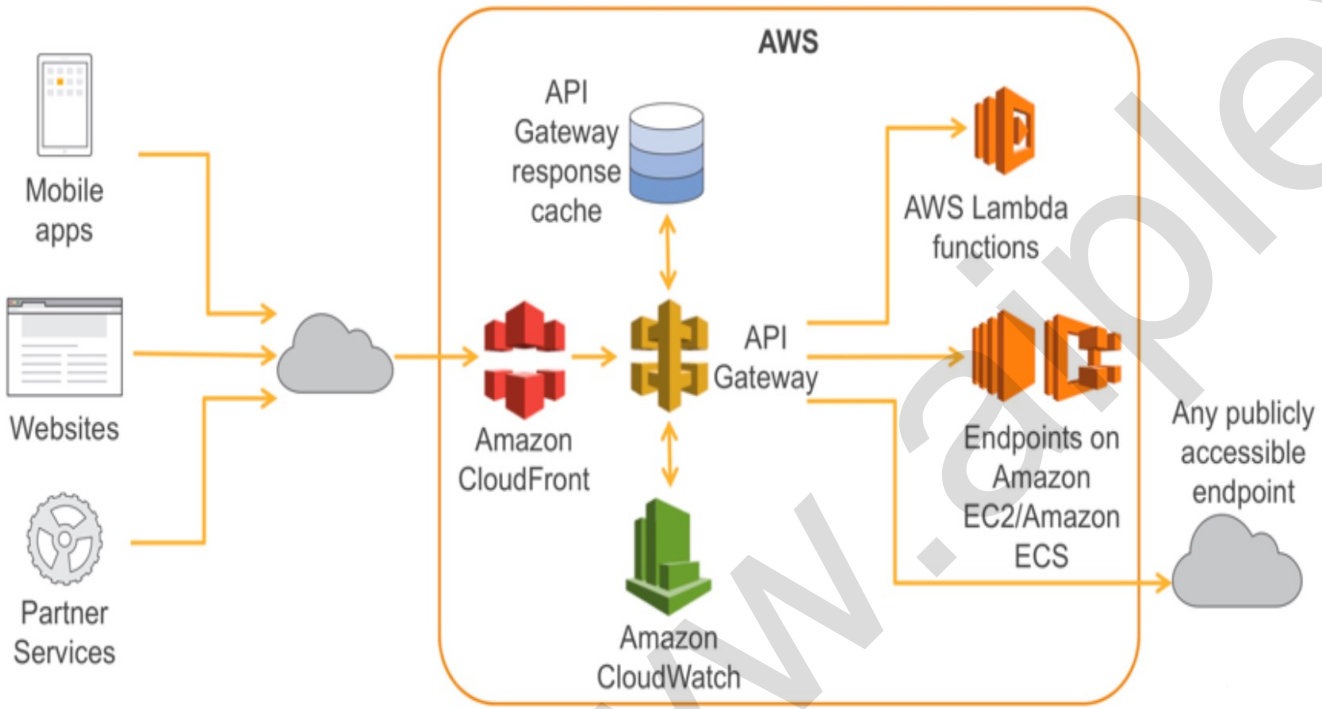
The target is the internet gateway that's attached to your VPC

LOCAL → SUBNET
DESTINATION → VPC, NAT Gateway, Virtual PrivatE Gateway, VPC endpoint

RT1 -> VPC HIT → ROUTE TO PUBLIC SUBNET

Destination => IP address/CIDR range .
Target => Where you want to send the traffic for the specified destination (e.g. if the destination is my local subnet, mention target as "local")
The Internet gateway is one of the targets (e.g. routing traffic to the internet). Other options for the target would be
NAT Gateway
Virtual Private Gateway
VPC endpoint
VPC peering connection etc. depending on your architecture

AWS API GATEWAY

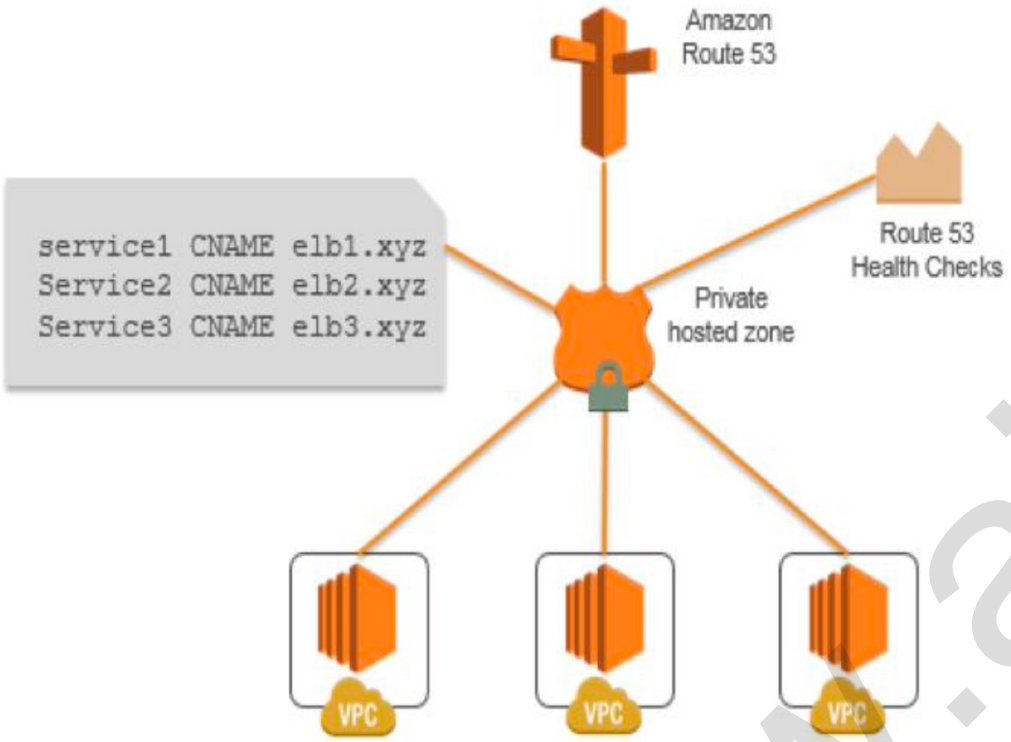


Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services.

API Gateway, helps us to create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications.

AWS ROUTE 53



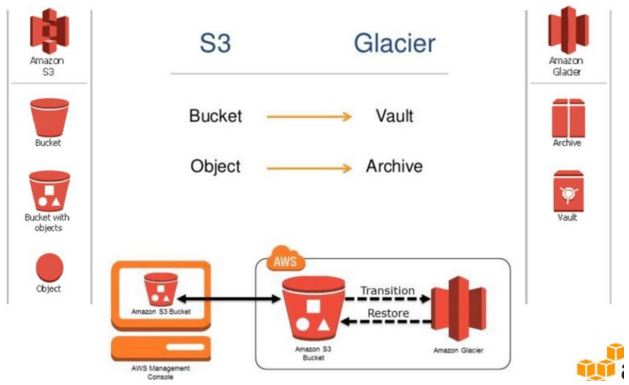
AWS Route 53 connects requests to the infrastructure running in AWS. These requests include AWS ELB, Amazon EC2 instances, or Amazon S3 buckets.

AWS Route 53 can be easily used to configure DNS health checks, continuously monitor your applications' ability to recover from failures, and control application recovery with Route 53 Application Recovery Controller.
AWS Route 53 traffic flow helps to manage traffic globally via a wide variety of routing types including latency-based routing, geo DNS, weighted round-robin, and geo proximity

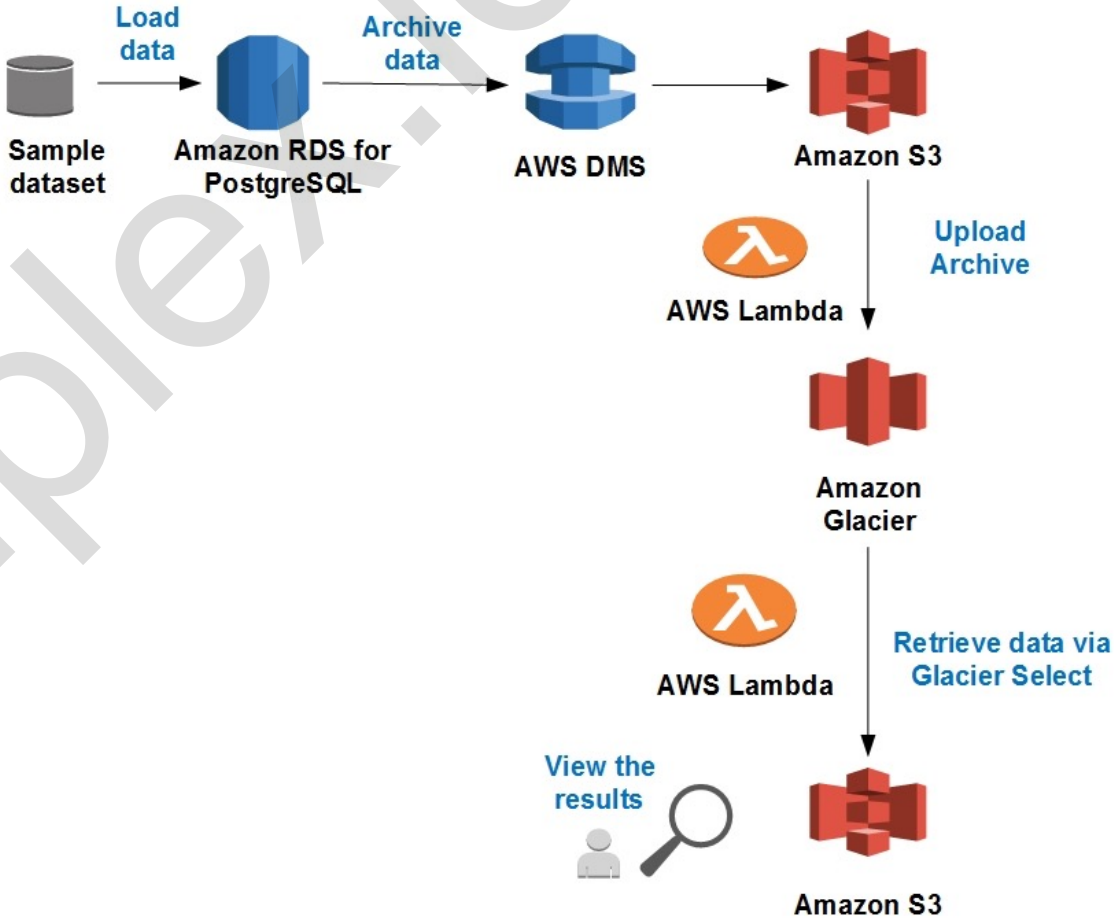
1) latency-based routing - user latency can be reduced by serving requests from the region where network latency is the lowest
2) geo DNS - Geolocation routing can be used to send traffic to resources based on the geographical location of users
3) weighted round-robin - Two regions servers are there then we can divide the traffic based on percentage [East 30% and West 70%]

Log analysis with Amazon Elasticsearch Service and Kibana

S3 Glacier Data Flow



Storage class	Designed for	Availability Zones	Min storage duration	Min billable object size	Monitoring and automation fees	Retrieval fees
Standard	Frequently accessed data	≥ 3	-	-	-	-
Intelligent-Tiering	Long-lived data with changing or unknown access patterns	≥ 3	30 days	-	Per-object fees apply	-
Standard-IA	Long-lived, infrequently accessed data	≥ 3	30 days	128KB	-	Per-GB fees apply
One Zone-IA	Long-lived, infrequently accessed, non-critical data	≥ 1	30 days	128KB	-	Per-GB fees apply
Glacier	Archive data with retrieval times ranging from minutes to hours	≥ 3	90 days	40KB	-	Per-GB fees apply
Glacier Deep Archive	Archive data that rarely, if ever, needs to be accessed with retrieval times in hours	≥ 3	180 days	40KB	-	Per-GB fees apply



Application Load Balancer

An Application Load Balancer (ALB) only works at layer 7 (HTTP).
It has a wide range of routing rules for incoming requests based on host name, path, query string parameter, HTTP method, HTTP headers, source IP, or port number.

** ELB only allows routing based on port number.

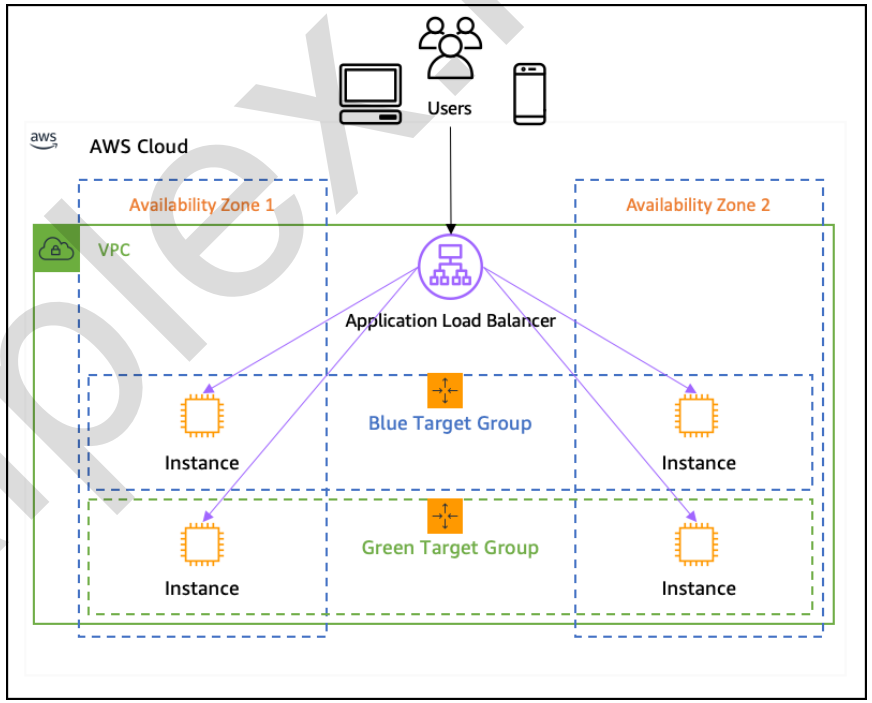
ALB can route requests to many ports on a single target.
Plus, ALB can route requests to Lambda functions.

ALB further supports Server Name Indication (SNI), which allows it to serve many domain names.

There is a limit, however, to the number of certificates you can attach to an ALB, namely 25 certificates plus the default certificate.

ALBs are typically used for web applications.

If you have a microservices architecture, ALB can be used as an internal load balancer in front of EC2 instances or Docker containers that implement a given service.



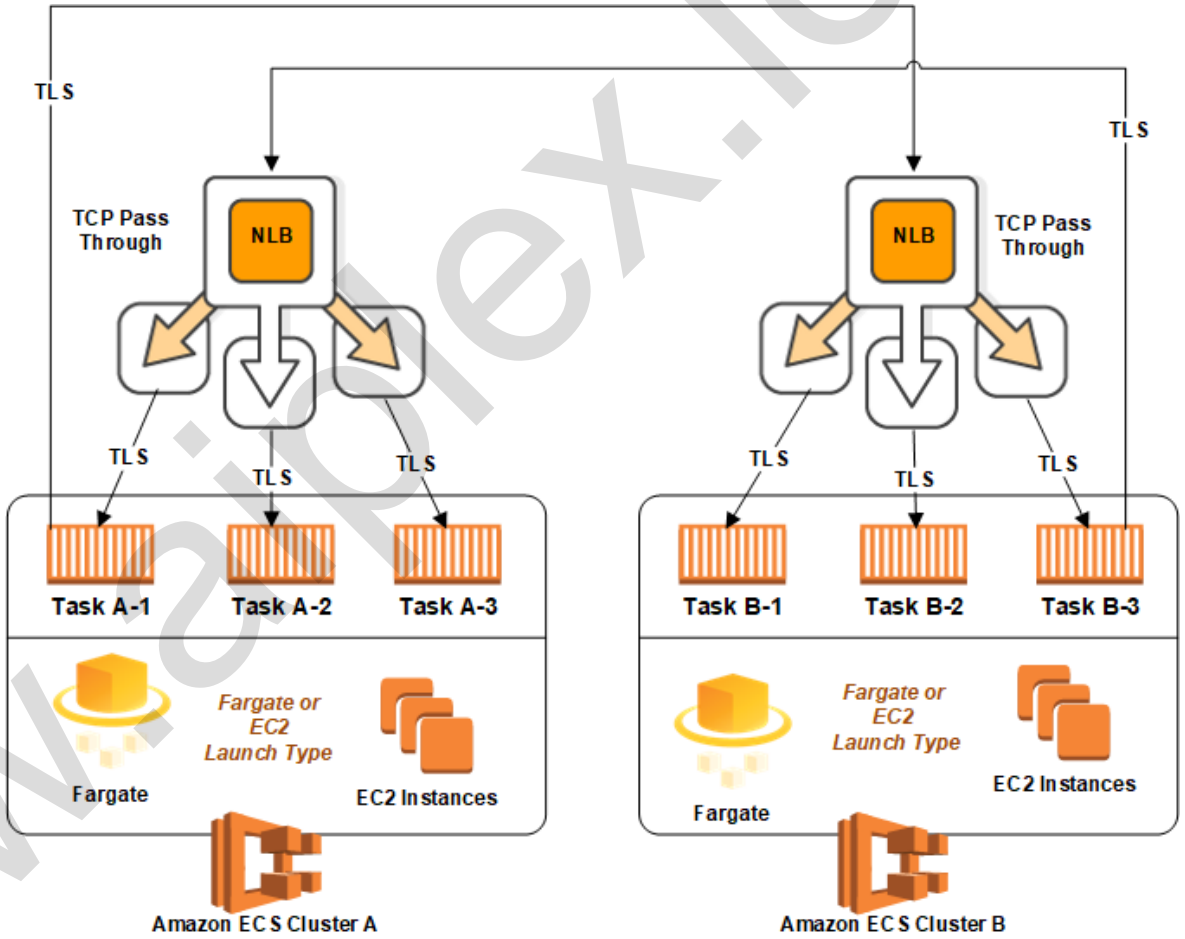
Network Load Balancer

- A Network Load Balancer (NLB) works at layer 4 only and can handle both TCP and UDP, as well as TCP connections encrypted with TLS.

- It has a very high performance

- It uses static IP addresses and can be assigned Elastic IPs—not possible with ALB and ELB.

- NLBs would be used for anything that ALBs don't cover. A typical use case would be a near real-time data streaming service (video, stock quotes, etc.) Another typical case is that you would need to use an NLB if your application uses non-HTTP protocols.



Comparison of AWS LB's

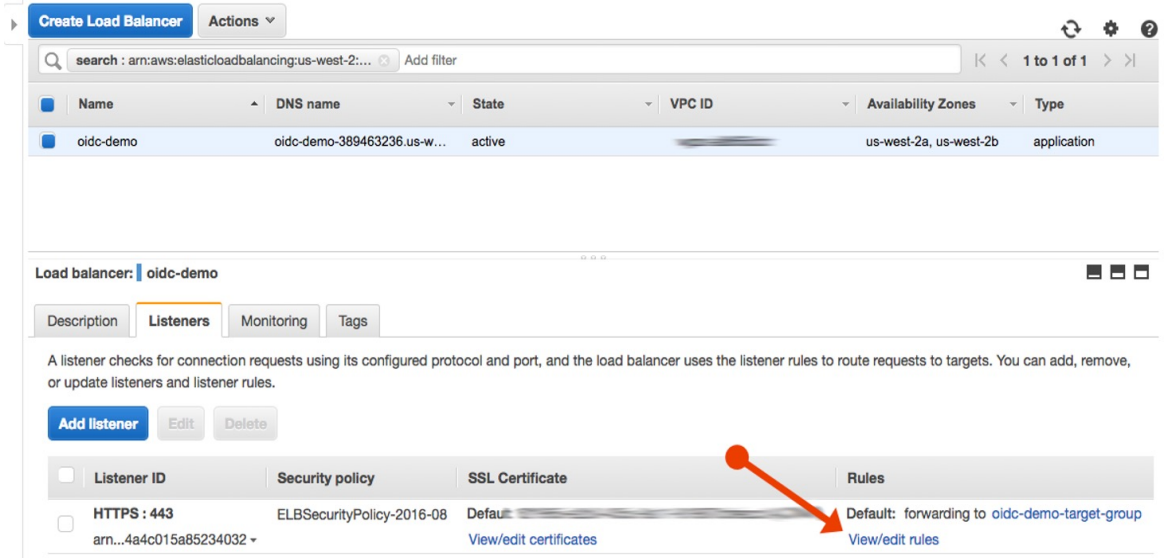
	ALB	NLB	ELB
Basic load balancing features			
Balance load between targets	Yes	Yes	Yes
Perform health checks on targets	Yes	Yes	Yes
Highly available	Yes	Yes	Yes
Elastic	Yes	Yes	Yes
TLS Termination	Yes	Yes	Yes
Performance	Good	Very high	Good
Send logs and metrics to CloudWatch	Yes	Yes	Yes
Layer 4 (TCP)	No	Yes	Yes
Layer 7 (HTTP)	Yes	No	Yes
Running costs	Low	Low	Low

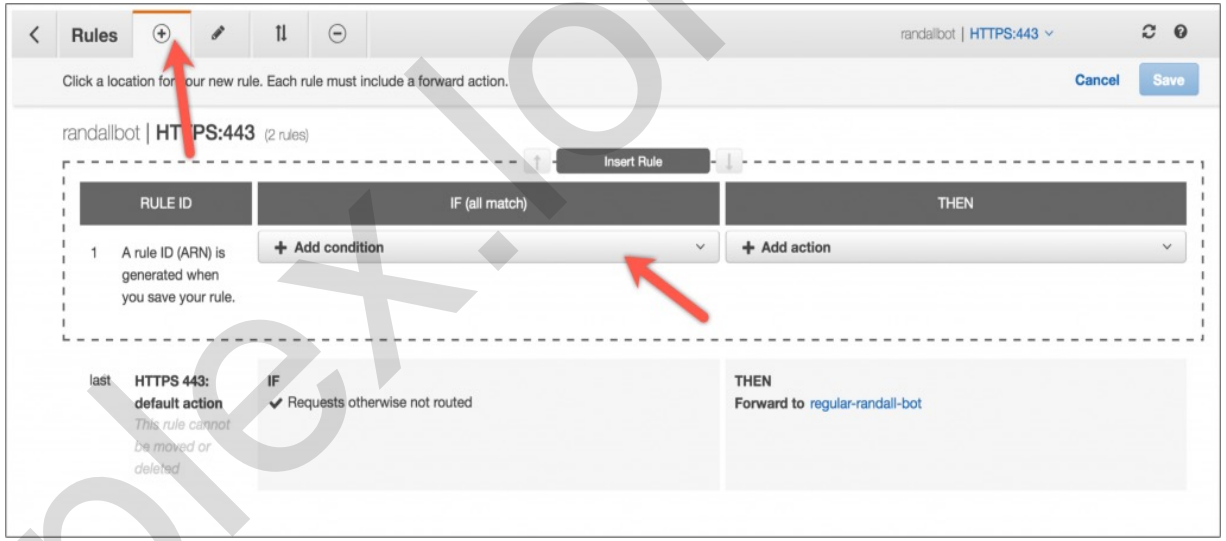
Layer 4 load balancing operates at the intermediate transport layer, which is responsible for delivering messages regardless of the content. Layer 4 load balancers simply forward network packets to and from the upstream server without bothering to inspect what's in them.

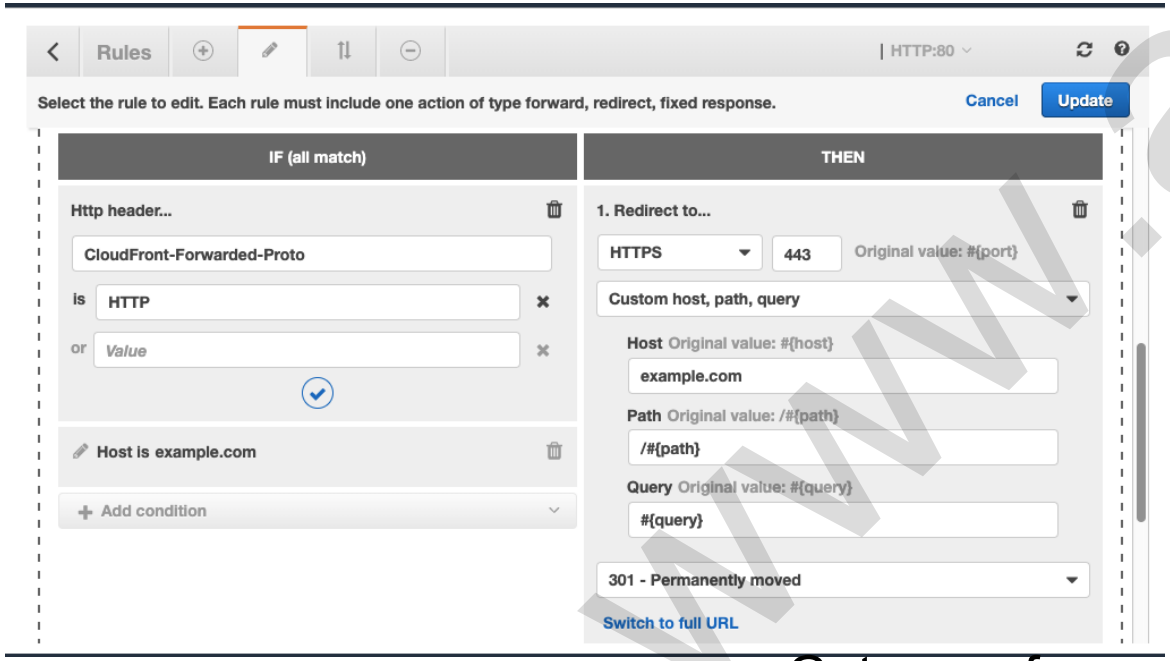
Layer 7 load balancing operates at the high-level application layer, which is responsible for the actual content of the message. Layer 7 load balancers route network traffic in a more complex manner, usually applicable to TCP-based traffic like HTTP. Unlike Layer 4, a Layer 7 load balancer terminates the network traffic and reads the message within. It makes a decision based on the content of the message. After which, it makes a new TCP connection to the selected upstream server and writes the request to the server.

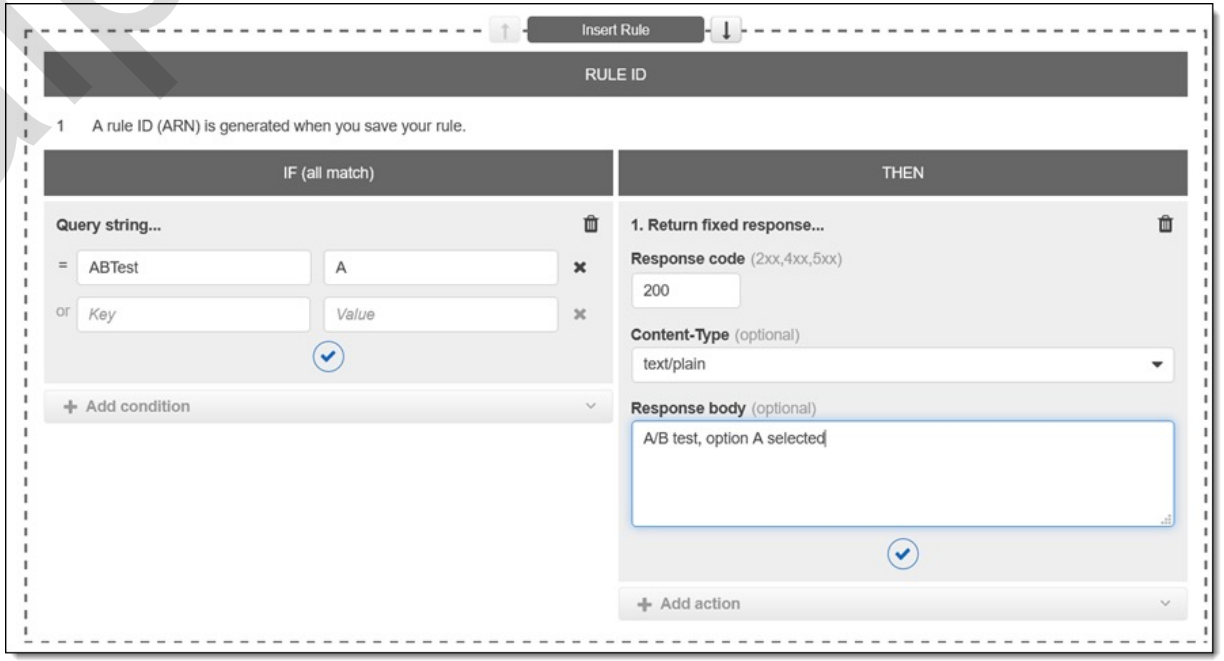
The decryption/encryption of TLS traffic is done at LB end rather then the application servers, which helps you optimize the performance of your backend application servers while keeping your workloads secure.

Application Load balancer Dashboard









DJANGO PROJECT LAB

- Create a repo in aws ECR name - > django-app
- IMPORTANT - Once the repo is create change the 600735812827.dkr.ecr.us-west-1.amazonaws.com BELOW VALUES IN COMMANDS TO THE REPO OF YOURS
- aws ecr get-login-password --region us-west-1 | docker login --username AWS --password-stdin 600735812827.dkr.ecr.us-west-1.amazonaws.com
- cd app/
- docker build -t 600735812827.dkr.ecr.us-west-1.amazonaws.com/django-app:latest .
- docker push 600735812827.dkr.ecr.us-west-1.amazonaws.com/django-app:latest
- Change the docker_image_url_django in VARIABLES.TF file with your current repo name
- Change the file paths in iam.tf and variables.tf file
- Go to terraform folder and hit this below command
- ssh-keygen -f california-region-key-pair
- terraform init
- terraform plan -out terraform.out
- terraform apply "terraform.out"
- pip install boto3 click

- export AWS_ACCESS_KEY_ID=""
- export AWS_SECRET_ACCESS_KEY=""
- export AWS_DEFAULT_REGION="us-west-1"
- cd deploy folder
- Run command in deploy folder - python3 update-ecs.py --cluster=production-cluster --service=production-service
- terraform destroy

DO FOLLOW ME
ON
INSTAGRAM/
TWITTER/
TELEGRAM
SINGAM4DEVOPS