Redes en AWS

- Creación de redes y subneteo
- DNS
- Balanceo de carga
- WAF
- CDN
- Storage en la nube

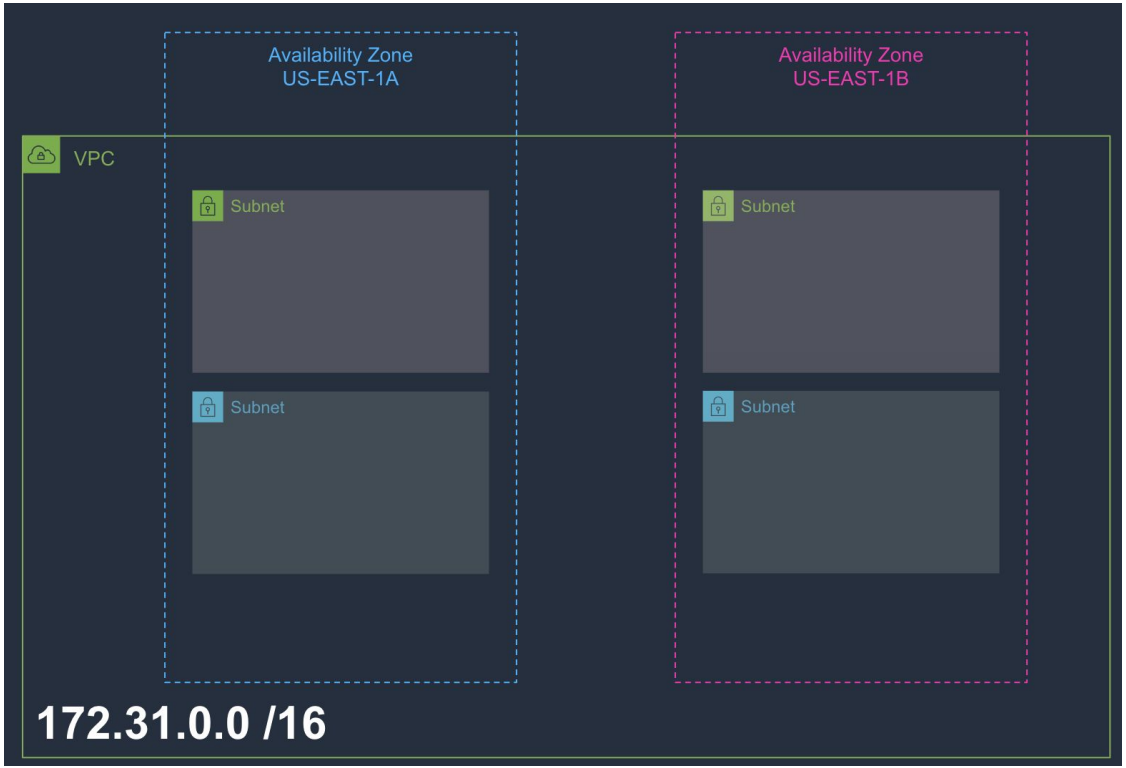
Creando una VPC conectada a Internet

1. Definir el rango de direcciones IP privadas y crear la VPC
2. Crear subnets en 1 o más availability zones (AZ) (una o mas subnets por AZ)
3. Crear una ruta a Internet (IGW)
4. Autorizar tráfico desde/hacia VPC

Es buena estrategia poner los elementos de nuestros clusters en distintas AZ, porque hay buena latencia entre ellas (estan conectados por fibra optica)

Elegir un rango de direcciones IP

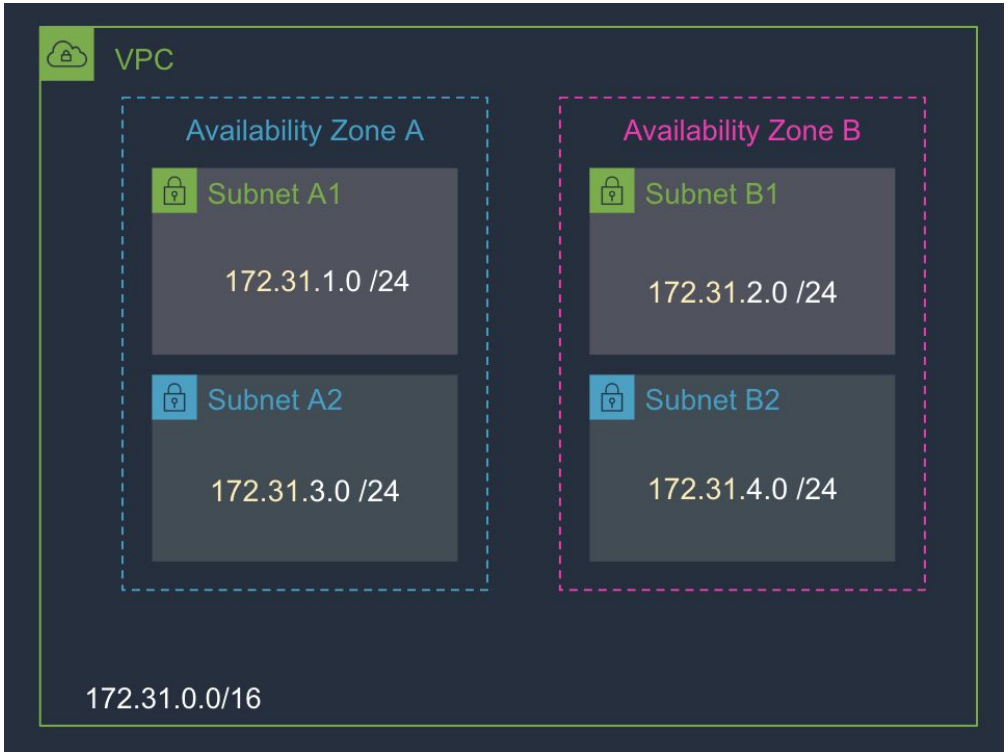
- Ejemplo VPC CIDR: 172.31.0.0/16



Conviene elegir el CIDR de forma tal que no haya ningun tipo de conflicto con otras redes que estamos accediendo.
Por ejemplo, si tenemos otro VPC o un datacenter on premise que queremos conectar con nuestro VPC, que no haya overlapping de IPs

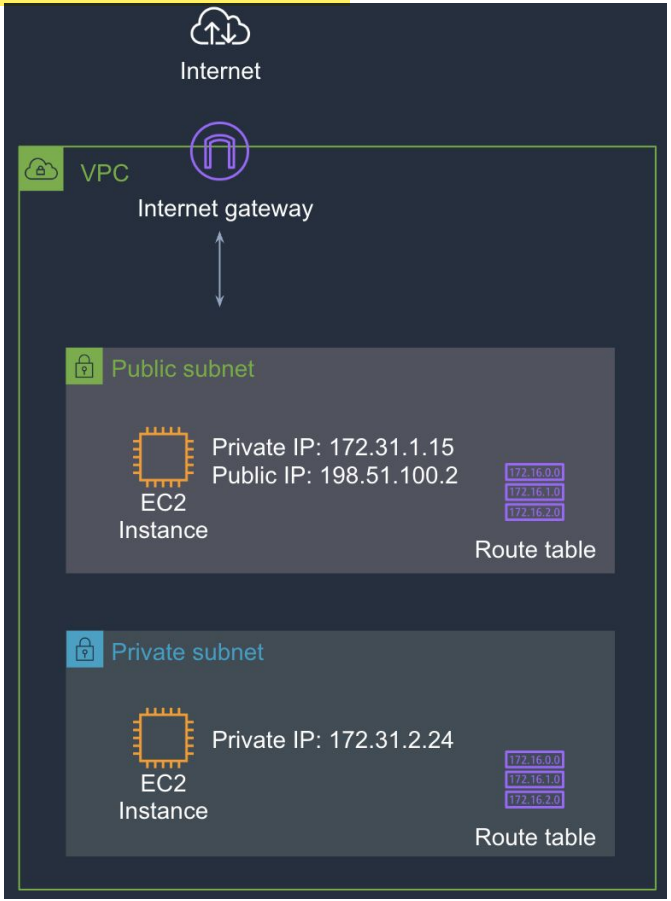
Crear subnets

- Definir una o más subnets para cada AZ



Crear una ruta a Internet

- Crear un Internet Gateway
- Crear una tabla de ruteo
- Colocar IGW como default gateway para las subnet públicas



Autorizar tráfico desde/hacia VPC

- Por default todo el tráfico es denegado
- Security groups (SG)
  - Nivel instancia
  - Stateful
- Network ACL
  - Nivel subnet
  - Stateless

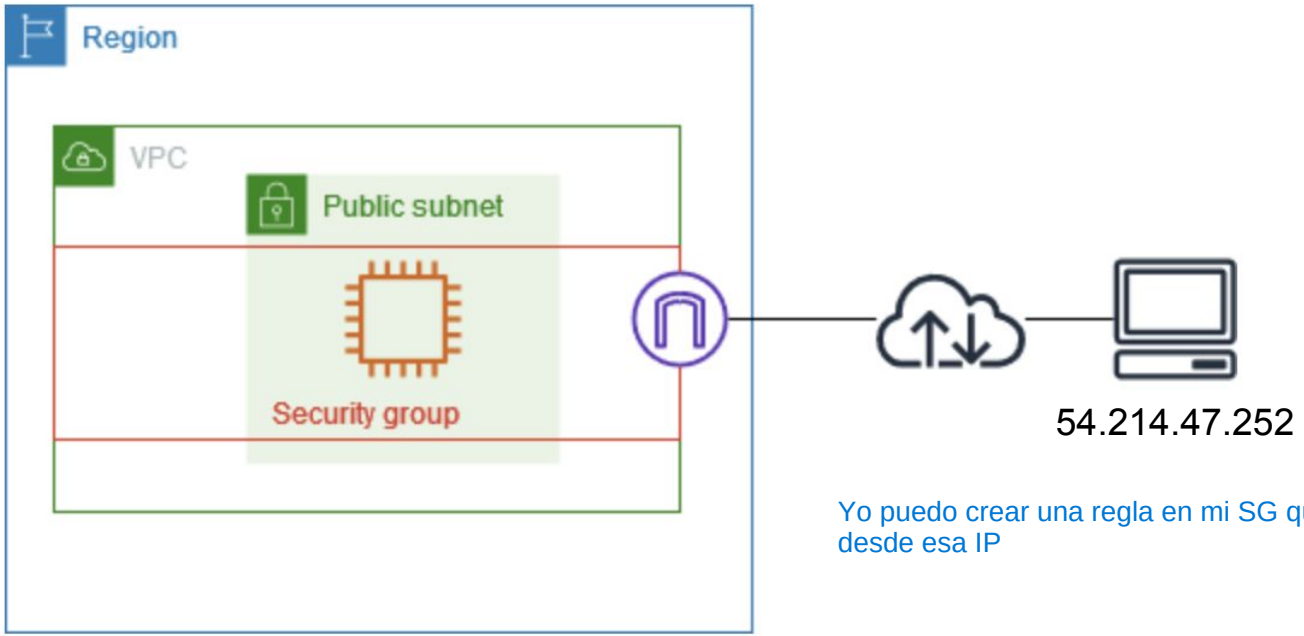
Por default cuando creamos la VPC no se permite ningun tipo de trafico, nosotros tenemos que crear reglas que permitan transitar este trafico.
Los SG son stateful porque si entra trafico que matchea una regla, automaticamente el trafico que conteste esa regla va a poder salir.
Por ejemplo, si yo habilito inbound puerto 22 (para SSH), no hace falta que tambien agregue el outbound.
Los network ACL van a nivel subnet, no por instancia. Estas son stateless, entonces tengo que habilitar todo inbound y outbound.

Security Groups

ENI: elastic network interface. Placa de red virtual. Los SG cuando decimos que funciona a nivel instancia, en realidad funciona a nivel ENI.

- Una instancia EC2 posee una o más ENIs
- Cada ENI tiene asociado uno o más SG (hay un ENI que se crea por default cuando yo lanzo una instancia)
- Un SG puede tener reglas de acceso “Inbound” y “Outbound”
- Solo se permiten reglas de tipo “Allow” Todo lo que no esta permitido es denegado. No puedo bloquear una IP especifica por ejemplo.

Security Groups - Ejemplo



Yo puedo crear una regla en mi SG que permita paquetes desde esa IP

Security Groups - Ejemplo

Inbound rules | Outbound rules | Tags

Inbound rules (6)

Manage tags | Edit inbound rules

Search

1

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-0caaa000b9571dc3b	IPv4	HTTPS	TCP	443	0.0.0.0/0
–	sgr-00fa9587d7abfc48d	IPv4	SSH	TCP	22	54.214.47.252/

Ejemplo: permito conexiones HTTPS de cualquier origen pero conexiones SSH solo de una IP en especial.
En la practica generalmente restringimos solo el trafico entrante. El trafico saliente se suele dejar en allow all.
Al tener una regla inbound SSH desde esa IP, no necesito establecer la regla outbound equivalente para el trafico de respuesta (porque el SG es stateful)

Network ACL

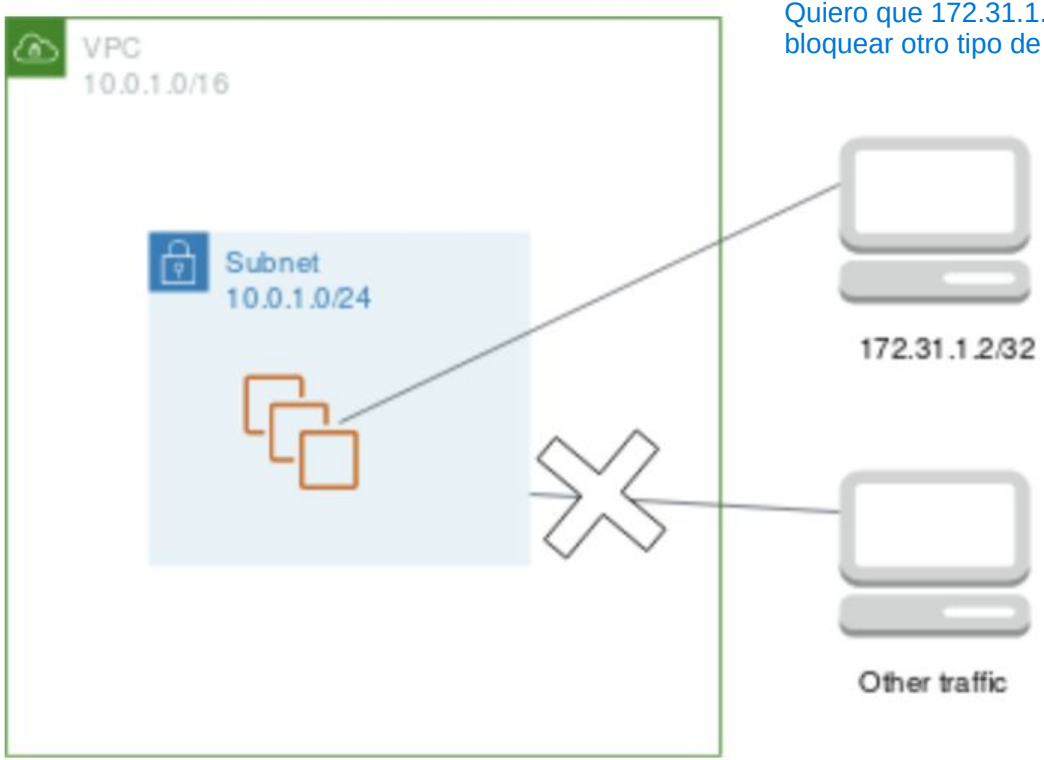
- Cada VPC se crea con un ACL default
- Se puede asociar un mismo ACL a varias subnets
- Cada subnet puede tener un único ACL y no mas
- Un ACL pueden tener múltiples reglas (inbound y outbound)
- Se permiten reglas de tipo “Allow” y/o “Deny”

(por ende, podemos hacer cosas con network ACL que no se pueden en un SG)

La idea de tener allow y deny, es hacer allow a un rango de IPs, y despues puedo hacer deny sobre subrangos mas chicos.

Network ACL - Ejemplo

Escenario para network ACL. Yo tengo una subnet. Quiero que 172.31.1.2 se pueda conectar pero bloquear otro tipo de trafico.



Network ACL - Ejemplo

A diferencia del SG, tengo que tener reglas outbound que matcheen las reglas inbound

Inbound

Rule #	Type	Protocol	Port range	Source	Allow/Deny	Comments
100	SSH	TCP	22	172.31.1.2/32	ALLOW	Allows inbound traffic from the remote computer.
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other inbound traffic.

Fijarse en el caso de SSH, que el trafico saliente SSH para contestar a la conexion inicial se permite en cualquier puerto que no requiere SUDO

Outbound

Rule #	Type	Protocol	Port range	Destination	Allow/Deny	Comments
100	Custom TCP	TCP	1024-65535	172.31.1.2/32	ALLOW	Allows outbound responses to the remote computer.
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other outbound traffic.

Capas de Seguridad en Redes en AWS

- **Capa 1: Security Groups**
  - Firewall virtual a nivel ENI
- **Capa 2: Network Access Control List (NACL)**
  - Objetivo: proteger instancias específicas dentro de una VPC
  - Ejemplo: bloquear una IP para que no acceda a la VPC
- **Capa 3: Específicas del proveedor (AWS)**
  - Firewall de DNS
  - WAF
  - Shield de DDoS

Route53

- Servicio de DNS administrado por AWS
- Features:
  - Registro directo de nuevos dominios
  - Resolución de nombres de dominio existentes
  - Health Checks

Podemos
- Registrar dominios nuevos
- Registrar un dominio que ya hayamos comprado
- Healthchecks. Chequeos que se hacen contra determinados dominios para ver si estan activos. Si no estan activos, se puede cambiar la resolucion del dominio por otro IP.

Route53 - Características

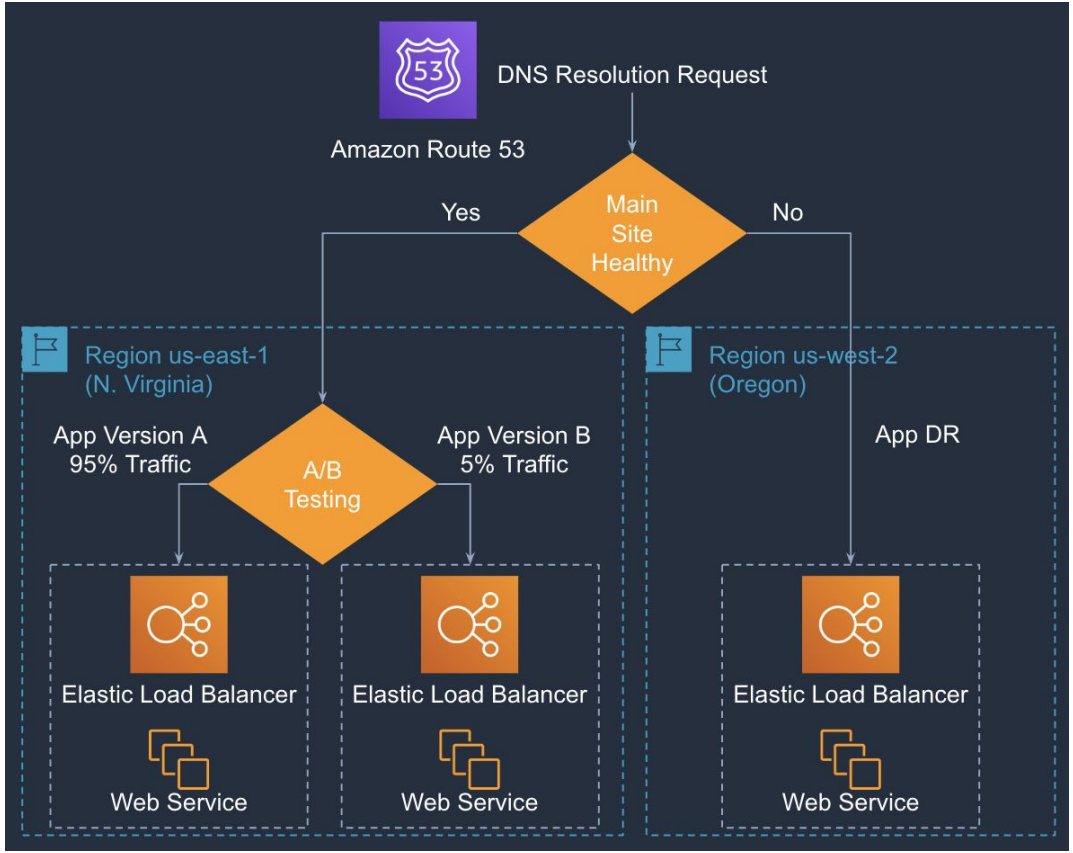
- SLA del 100%
- DNS Failover
- Resolución basada en reglas
- Logging de consultas DNS

Se supone que no cae nunca. Dado que el SLA es 100%, entonces si se llega a caer entonces AWS te hace un cheque devolviendote plata (porque un SLA es un contrato).
Aca las empresas te hacen una trampita diciendo por ejemplo que el servicio se considera caido despues de 15 minutos.

DNS failover es que puedo configurar que si un dominio resuelve a una IP que no esta funcionando, entonces puedo cambiarla por otra IP.
Ejemplo: mi server primario es A. Si se cae A, dale a los usuarios la IP de mi servidor secundario B (que generalmente es menos potente)

Puedo configurar reglas para cambiar la IP que contesta Route53 ante un dominio (por ejemplo de que region geografica viene)

Route53 - Ejemplo



El healthcheck puede ser chequear que funcione la conexion TCP, o puede ser mas complejo como una funcion que se fije si funcionan X servicios (por ejemplo si logro conectarme a la DB).

Puedo hacer traffic splitting. Por ejemplo si estoy deployando una nueva version de mi app, puedo dirigir el 5% del trafico a la nueva version, y el resto del trafico a la version vieja que ya se que funciona.
Esto lo hace el Elastic Load Balancer (que tambien puede tener healthchecks)

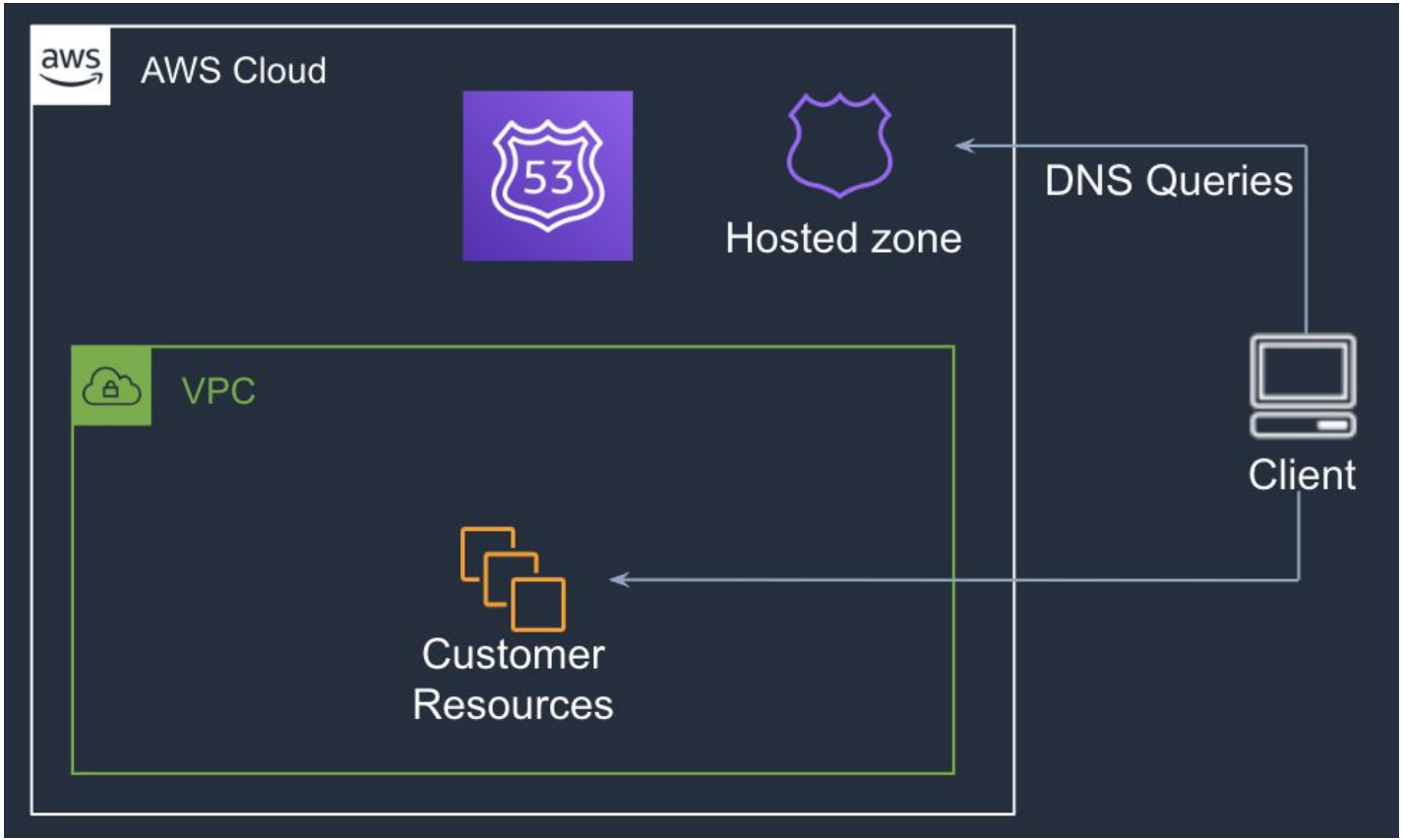
Hosted Zone

Ejemplo, voy a tener una hosted zone que se llama itba.edu.ar

- Es un contenedor de registros
- Tiene el mismo nombre que el dominio (ej: itba.edu.ar)
- Public Hosted Zones
  - Para resolución de nombres accesibles via Internet
- Private Hosted Zones
  - Para resolver nombres internos dentro de una VPC
  - Para obtener una respuesta de una private hosted zone debo preguntar desde una instancia EC2 dentro de la VPC

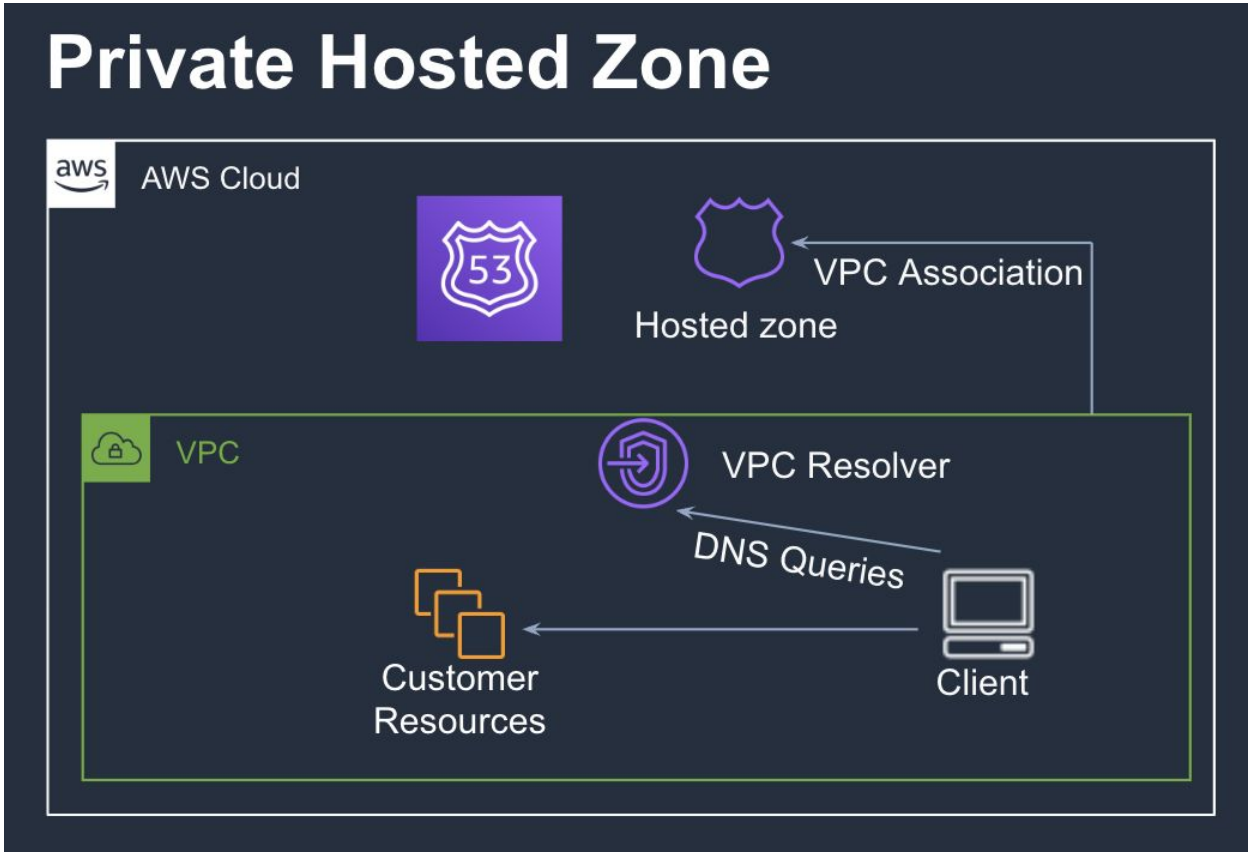
Public Hosted Zone

El cliente le pregunta a route 53 la IP de un servidor web por ejemplo. Route53 le contesta con la IP de un EC2 con un servidor web (que esta en una public subnet) montado en este ejemplo. El cliente ya puede acceder a la pagina.



Private Hosted Zone

El VPC resolver es el que se encarga de mandarle la consulta a Route53 cuando hago una consulta dentro de la VPC.
Para un sistema simple, podemos simplemente usar el archivo hosts para evitarnos configurar Route53.
Sin embargo, Route53 nos deja hacer cosas mucho mas complejas, como los healthchecks y traffic splitting.
Se puede configurar a quien se le da mas prioridad, si a Route53 o el archivo hosts del EC2.



DNS - Repaso

- Registro A
  - hostname -> IPv4
- Registro AAAA
  - hostname -> IPv6
- CNAME (para poner alias)
  - hostname -> otro hostname
- MX, etc…

  (mx para el server de correos)

Alias Records

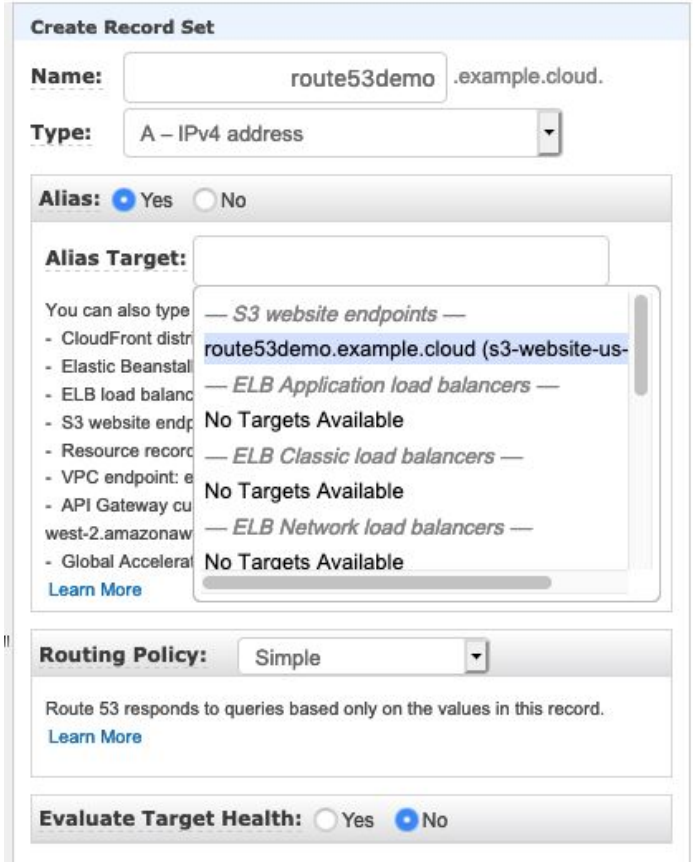
Por ejemplo, yo tengo la zona midominio.com. Yo no puedo hacer que esto resuelva a un CNAME. Pero si puedo hacer que resuelva a un alias. Entonces, puedo usarlo para apuntarlo a un Load Balancer por ejemplo.

- Tipo especial de registro en AWS (no es parte del estándar DNS)
- Apunta a un recurso de AWS o a otro registro (o a otro registro DNS)
- Los Alias pueden ser usados en la raíz de la zona (los CNAME no)
- Son resueltos por Route 53 internamente
- Se utilizan para apuntar directamente a cosas como:
  - un sitio web estático en un bucket S3
  - una distribución CloudFront
  - un Elastic Load Balancer
  - etc…

Alias Records

- Ejemplo: existe un registro Alias que apunta a un bucket S3. Si la dirección IP del bucket cambia, Route53 automáticamente empieza a responder con la nueva IP

En el ejemplo, se tiene un sitio estatico montado en un S3.
El punto es que si cambia el IP del servicio al que yo apunto (por ejemplo el S3), no hace falta que yo vaya a Route53 a cambiar la IP, sino que se cambia sola.
Esto me suma alta disponibilidad, porque el dominio siempre v aa resolver a ALGUNA INSTANCIA funcional de ese S3 con mi pagina.



Routing policies

- Controlan las respuestas de Route 53 a las consultas DNS
- Se utilizan en conjunto con los health checks para dirigir las consultas a destinos que estén funcionando correctamente

Routing policies

- Tipos de políticas:
  - Simple routing policy
  - Failover routing policy
  - Geolocation routing policy
  - Geo Proximity routing policy
  - Latency routing policy
  - Multivalue answer routing policy
  - Weighted routing policy

Tengo un healthcheck contra dos equipos. Si el primero responde le mando a ese, sino le mando al servicio failover.

Geolocation: si pregunto desde argentina me devuelve una IP. Si pregunto desde alemania tal vez me devuelve otra IP. Tal vez si pregunto desde francia elige no devolverme ninguna IP.
Geoproximity me devuelve la IP mas cercana a donde yo estoy preguntando.
En resumen, Geolocation son para reglas mas estaticas y Geoproximity por cuestiones de performance.

Siempre me devuelve la region que tiene menos latencia para conectarse conmigo

Tengo muchos registros para el mismo nombre y quiero que devuelva uno random

Queremos mandar el 95% del trafico a una instancia y el 5% a otra.

Elastic Load Balancing

- Un ELB distribuye el tráfico entrante hacia múltiples destinos
  - Instancias EC2, IPs, containers, etc.
- Monitorea el estado y rutea a destinos “sanos”
- Escala automáticamente dependiendo del tráfico entrante

Puedo usar un health check para definir a que instancia mando el trafico

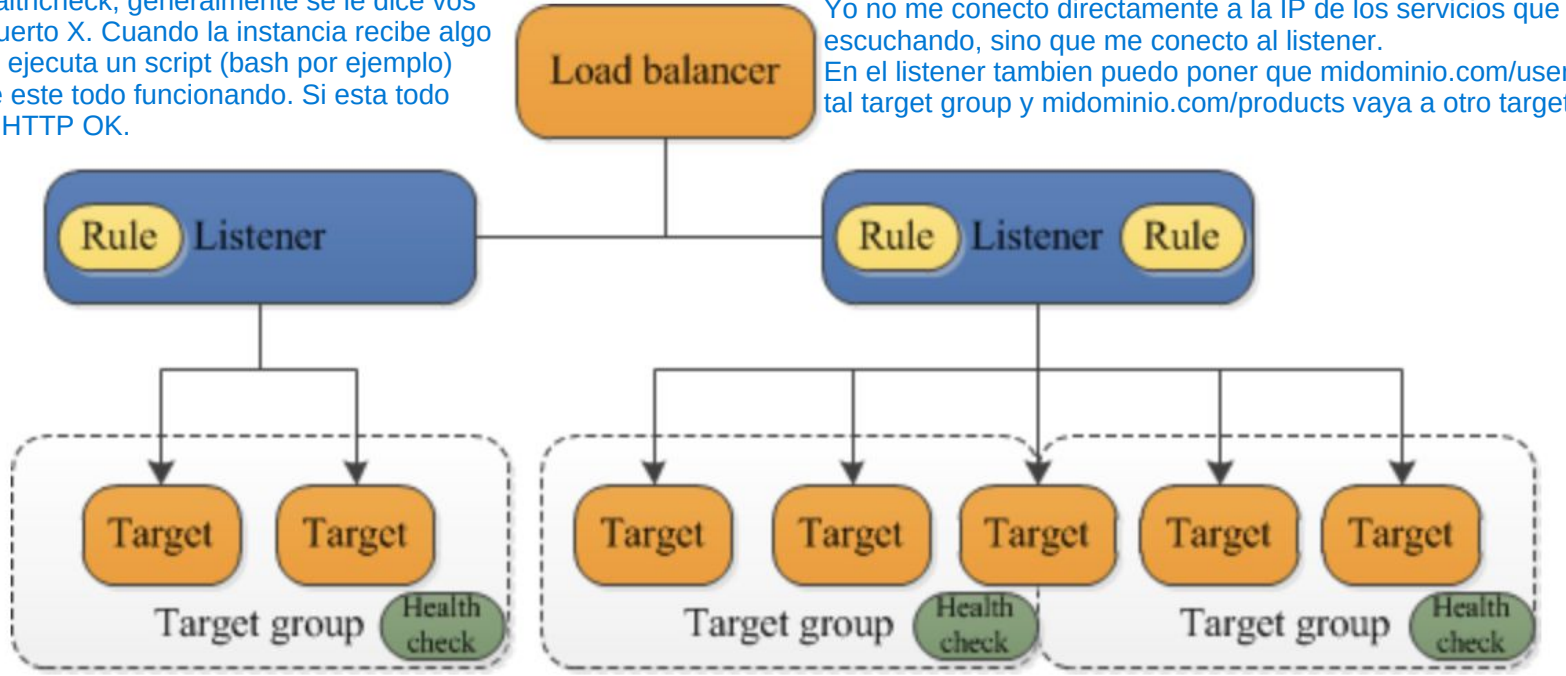
Elastic Load Balancing

Los ELB funcionan cross AZ. Puedo redirigir a distintas instancias que esten en distintas AZ.

Esto es tanto para ALB como NLB

Se puede tener listeners en distintos puertos de mi ELB. Cada listener puede tener distintos target group asociando, y se va haciendo un health check al target group.
Yo no me conecto directamente a la IP de los servicios que tengo escuchando, sino que me conecto al listener.
En el listener tambien puedo poner que midominio.com/users vaya a tal target group y midominio.com/products vaya a otro target group

Para lo del healthcheck, generalmente se le dice vos conectate al puerto X. Cuando la instancia recibe algo en ese puerto, ejecuta un script (bash por ejemplo) que se fija que este todo funcionando. Si esta todo bien devuelve HTTP OK.



Ejemplo: yo tengo hosteado un sitio web en HTTP y en HTTPS. Tengo un listener en el puerto 80 y otro en el 443.

Otro ejemplo: mis target son todos proxy, que rutean a diferentes bases de datos. Yo puedo crear un listener en el puerto 1024 (que se lo mando a la DB1, otro listener en el puerto 1025, que se lo mando a la DB2). Se usa para distribucion de carga en lectura y escritura.

Tipos de Load Balancer

- Application Load Balancer (ALB)
- Network Load Balancer (NLB)
- Classic Load Balancer (deprecated) no lo vamos a ver porque no se recomienda
- Gateway Load Balancer (3rd party appliances) es un appliance que NO es de AWS, por eso se dice 3rd party
  - Checkpoint, Fortinet, F5, etc.

Application Load Balancer (ALB)

- Capa 7
- Protocolos HTTP, HTTPS, gRPC
- Puede rutear en base al contenido del request
  - Ejemplo: redirigir determinadas URLs a destinos específicos
- Casos de uso típico: sitios web, microservicios

Por ejemplo, tengo una instancia contestando a itba.edu.ar/intranet y otra instancia contestando a itba.edu.ar/ceitba.
Esto se usa bastante para microservicios por eso.

ALB Sticky Sessions / Session affinity

- Permite redirigir al mismo cliente siempre a la misma instancia de backend
- El cliente debe soportar cookies
- Puede producir desbalanceo
- Caso de uso: que el usuario no pierda los datos del estado
- Alternativa: persistir el estado en una DB (persistir el estado en un Key-Value store y que nuestras sesiones sean Stateless)

Network Load Balancer (NLB)

(tal vez se podria llamar transport load balancer en vez de network)

- Capa 4
- Protocolos TCP, UDP, TLS
- Menor latencia que los ELB
- Caso de uso típico: balanceo de carga entre instancias EC2

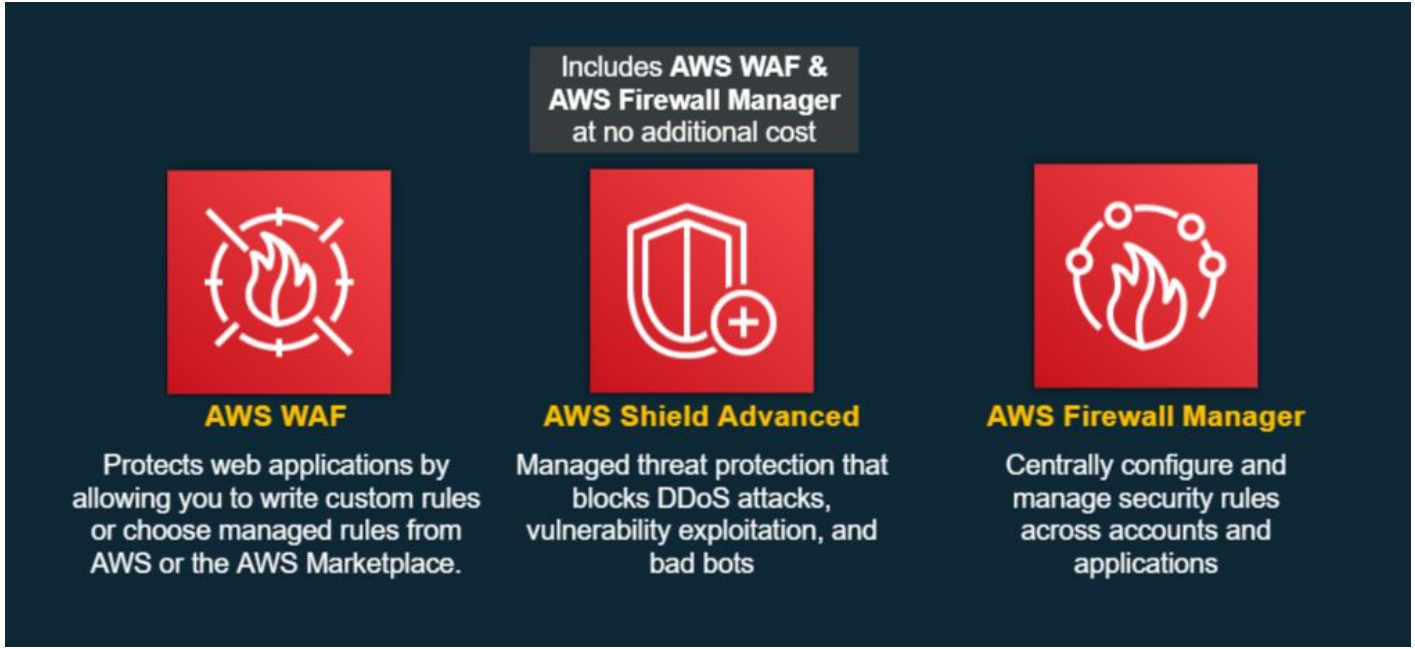
Aca esta mal el slide, tiene que decir ALB en vez de ELB.
Tiene menos latencia que un ALB porque no se tiene que ir inspeccionando los paquetes.

Por ejemplo, tengo un cluster en mi base de datos y a los clientes les doy una sola IP. Si se cae un nodo, el cliente sigue usando la misma IP y el NLB redirige la IP a otra instancia.

Un ALB esta en el orden de los 300/400 ms y un ELB en el de los 100ms. Esta diferencia es importante para una base de datos por ejemplo.

AWS - Seguridad

Cada cuenta que me creo en AWS trae sin costo adicional el WAF y el Firewall Manager. El Shield Advanced se paga aparte.



El WAF es para proteger mis aplicaciones web a traves de una serie de reglas.
Ademas, me puedo bajar sets de reglas predefinidos por AWS que sirven para evitar exploits especificos

El Shield Advanced es para protegerme de ataques DDOS

El firewall manager es cuando tengo muchas cuentas de AWS en mi organizacion, y quiero centralizar algunas reglas. Por ejemplo, quiero que ninguna instancia a nivel compañia pueda activar X puerto o conectarse a Y IP.

Niveles de protección

- AWS WAF
  - filtrar los request de acuerdo a reglas (ej: bloquear una IP)
  - protege contra exploits comunes de capa 7
  - no protege contra DDoS
- AWS Shield Standard
  - protección contra DDoS básicos (ej: SYN Flood)
- AWS Shield Advanced
  - protección contra ataques más sofisticados (es pago)

AWS Firewall Manager

- Permite administrar las reglas para todas las cuentas de una organización (que no son bypasseables)
  - Reglas comunes de seguridad (como reglas de SG)
  - Reglas de WAF
  - AWS Shield Advanced
  - Security Groups

Content Delivery Networks (CDN)

- Grupo de servidores distribuidos geográficamente para optimizar el tráfico web, al llevar el servicio más cerca del cliente

La idea es que todos mis clientes siempre se puedan conectar a mi sitio web con baja latencia, porque tienen un servidor cerca.

- Casos de Uso
  - Caché de sitios
  - Aceleración de APIs (ej: parte del procesamiento de la API ocurra cerca del cliente)
  - Objetos estáticos (imágenes, CSS, etc)
  - Streaming de video
  - Download de archivos grandes (cachear una ISO de un SO por ejemplo, que evita tener que elegir un mirror a mano)

En el regional edge cache se pueden cachear cosas grandes. En el edge location se cachean cosas mas pequeñas. Esto es porque los regional cache estan en regiones de AWS pero los edge no (AWS le puede subcontratar a algun ISP local)

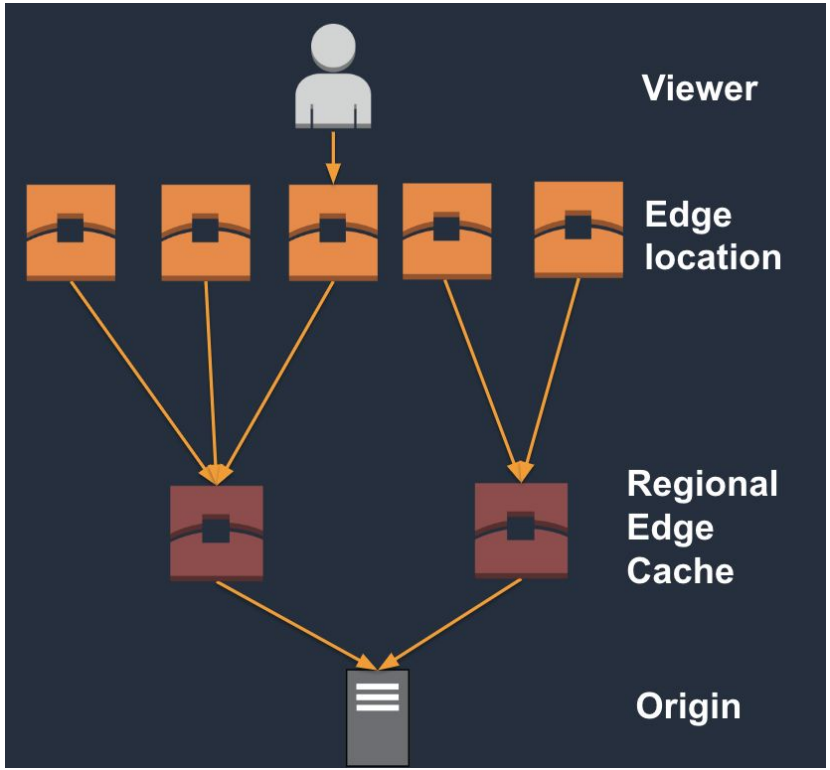
AWS CloudFront

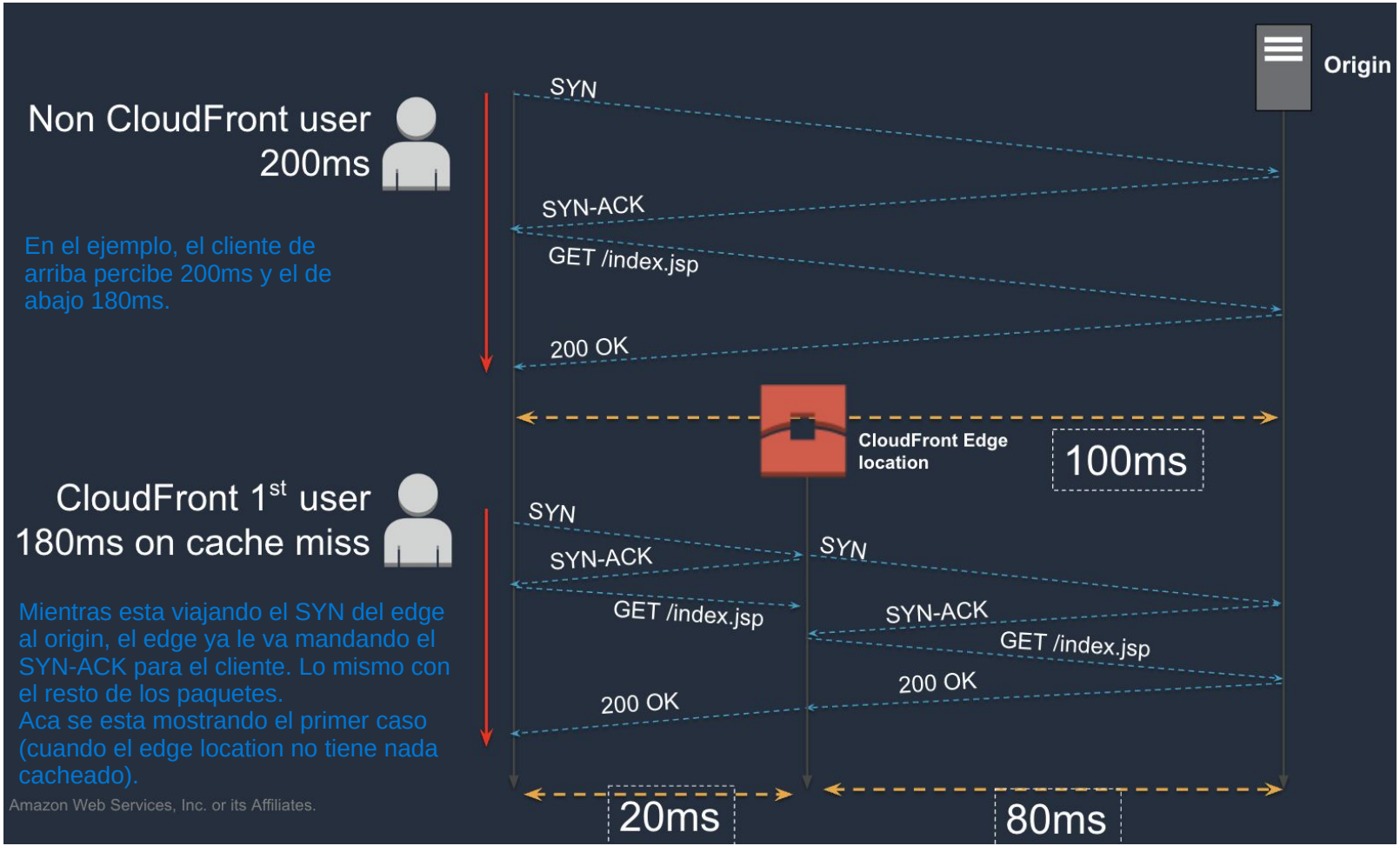
- Ruteo DNS inteligente basado en latencia y disponibilidad
- Edge locations basadas en un modelo PoP (puntos de presencia)
- Doble cache
- Protección contra DDoS
- WAF

(porque nos estamos conectando a server intermedios en vez de a origin)

- Control de acceso/URL firmadas

(que solo puedan acceder los que tienen URL firmadas)



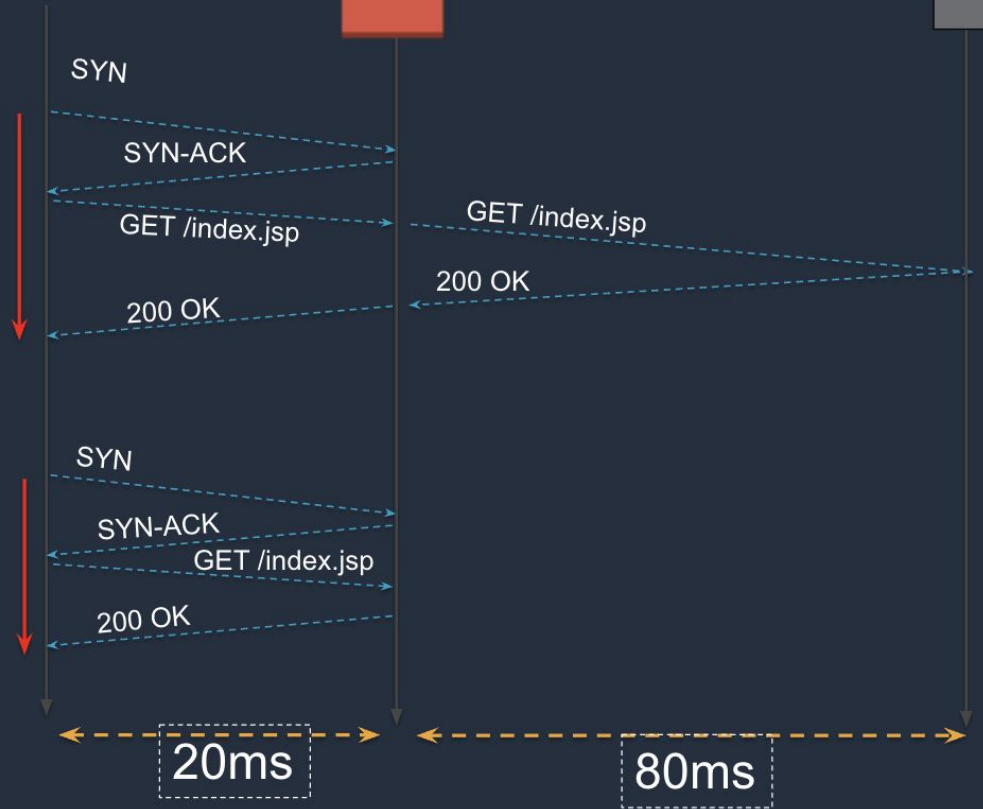


Cuando viene un segundo usuario, no tengo que hacer una conexion TCP con origin porque ya la tengo.
Igual todavia no tengo cacheado index.jsp entonces lo tengo que pedir.

CloudFront 2nd user
120ms on cache miss
2x acceleration

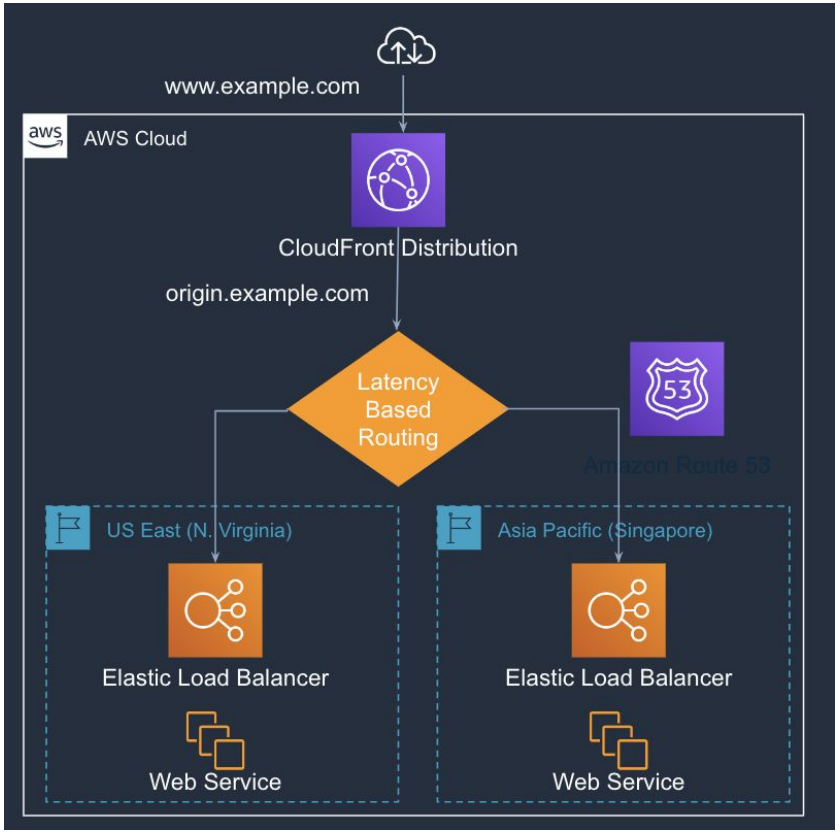
En el ejemplo de abajo, mi edge location ya tiene cacheado index.jsp, entonces ni siquiera hace falta conectarse con origin. El tiempo de conexion es mucho mas rapido que antes.

CloudFront 3rd user
40ms on cache hit
5x acceleration



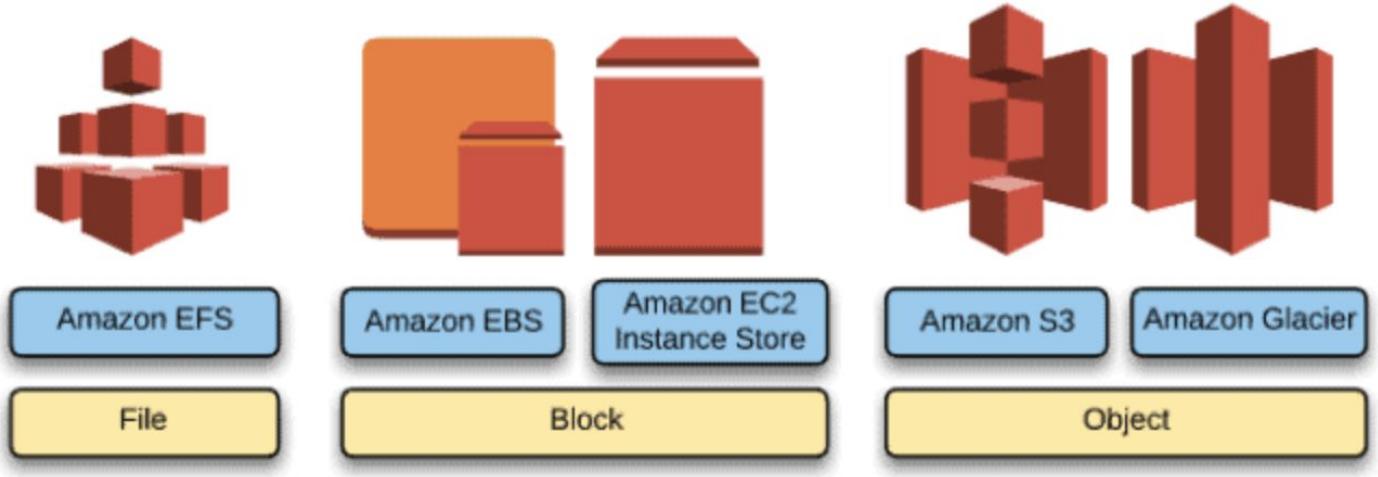
Aplicaciones multi-región

- Podemos combinar Route53 con CloudFront
- Se configura CloudFront con un ”custom origin”
- El custom origin apunta a un record set de Route53
- Usando Ruteo basado en latencia enviamos los request a la región más cercana



Una instancia de CloudFront cerca del usuario recibe el pedido, y de ser necesario va a buscar el dato al ELB de la region mas cerca que tenga. Si quiero agregar un ELB en una nueva region, con el registro Alias de Route53 la agrego y listo.

Storage en la nube



Elastic File System (EFS)

- Network file system as-a-service
- Escalable, elástico, alta disponibilidad
- Compatible con protocolo NFS
- Caso de Uso: montar un volumen compartido entre múltiples instancias EC2

Desde la consola de AWS me creo un filesystem que despues puedo montar desde mis EC2.
Usa el protocolo NFS.

Ejemplo: filesystem de backups. Todas las instancias van escribiendo ahi sus backups. En la practica se usa bastante poco, la mayoria guarda las cosas en S3.

Esta bueno para guardar archivos. No esta bueno para usar de base de datos, porque no se aprovecha del todo el cache de bloques de linux (porque el protocolo NFS no esta orientado a esto).
Cualquier instancia dentro del mismo VPC (por mas que sean distintas AZ) puede acceder al EFS

Elastic Block Storage (EBS)

Estos discos solo los puede acceder una instancia a la vez. Lo que si puedo hacer es desattachearlo y attachearlo a otro instancia. Pensarlo como un SSD normal.

- Discos virtuales
- Network-attached-storage (NAS)

El disco no esta local (fisicamente conectado), sino que esta en la red.

- Diferentes tipos disponibles
  - HDD (st1, sc1)
  - General Purpose SSD (gp2, gp3)
  - Provisioned IOPS SSD (io1, io2)

Posibilidad de garantizar SLA. El x% de nuestros requests van a tardar Y ms.

- Caso de Uso: disco dedicado a una instancia EC2

(ejemplo: volumen de una DB, webserver para guardar los archivos que la gente sube a una pagina web)

EC2 Instance Store

- Disco raiz (/) de una instancia EC2
- Dependiendo del tipo de instancia, algunas vienen con SSD
- Attacheado físicamente al host que aloja la instancia EC2 (son locales, entonces el acceso es rapido)
- Se borran al reiniciar la instancia Son discos efemeros
- Caso de Uso: datos temporales, cache, etc.

La latencia es un poco mas alta que un disco local.

Amazon S3

- Simple Storage Service
- Almacenamiento de archivos en la nube AWS
- Alta disponibilidad (99.5-99.99%)
- Redundante (durabilidad de 11 9's)
- Diferentes tiers
- Pueden definirse políticas de ciclo de vida para mover archivos automáticamente entre tiers

Amazon S3 - Standard tiers

- Express One Zone
  - mejor latencia

lo guarda en una AZ determinada. Tiene menor disponibilidad porque no tengo los datos replicados

- Standard
  - datos activos, acceso frecuente

Es el mas usado

- Infrequent Access
  - menor precio, mayor tiempo de acceso
- One Zone Infrequent access
  - más barato

Es todavia mas barato que infrequent access, y guarda en una sola AZ (sacrifico disponibilidad)

Amazon S3 - Glacier tiers

Se usa para guardar cosas por temas de compliance legal y quiero pagar lo menos posible (y lo mas probable es que no tenga que accederlos)

- Almacenamiento barato a largo plazo
- El acceso no es inmediato
- Leer los archivos requiere un proceso de desarchivado

  Puede tardar de minutos a horas
- Pago por cada acceso
- Útil para archivos de acceso muy poco frecuente
- Ejemplo: histórico de transacciones

Amazon S3 - Tiers