|  |
| --- |
|  |
| Ver a imagem de origem  Certifield Developer Associate |

**IAM = Indentity Acess Management:**

E o local onde você consegue criar Usuários, grupos e permissões

* **Multifactor Autentication:** Logar em um site, normalmente você tem seu usuário e sua senha, Se os desenvolvedores tiverem apenas o user e o password, ficará um buraco de segurança bem grande, e aí onde habilitamos o MultiFactory Autentication

User, Pass, Token, Auth com isso depois de inserir o usuário e a senha ele vai pedir um código de autenticação que abrimos o app, e colocar o código de autenticação.

Eleva ao topo a segurança

* **Identity Federation:**  Se na sua empresa você já tem um AD (Active Directory) Você conseguira fazer um link e os usuários, vão utilizar esta base de dados para autenthicar na AWS.
* **Permissions: (Granular)** Você consegue aplicar permissões a usuários (Users) específicos, Exemplo tenho um usuário que queira criar maquinas EC2, mais somente do tamanho Small e que tenha no Max. 12 GB de Memória RAM, Max. 1 Terabyte de Disk.
* **Passwords:** Podemos criar uma política de rotatividade da senha, que ela expira em 3 meses, ou a cada 6 meses, e tenque ter um grau de dificuldade especifico. Consigo integrar todos os serviços ou **Services** (ou seja, o mesmo usuário poderá dar permissões ao S2 ao S3 DB e assim por diante.
* **Compartilhamento de Acesso:** consigo compartilhar o acesso da plataforma com outros usuários utilizando o IAM criando usuários pra cada um (recursos da plataforma com outros usuários) **Shared**
* **PCI DSS Complance:** Você tem um website, ele está hospedado na AWS, se você configurou ele inteiro com códigos dentro da AWS e agora eu preciso receber pagamentos via cartão de credito. (Visa e Mastercard) Você tem a garantia que os dados do seu cliente não vão ser compartilhados e estão seguros (boas práticas)
* **Criação de Grupos de usuários (Groups)** Dentro do IAM, podemos criar os Grupos de usuários (Users) que trabalham com: Databases, S3, EC2, e etc. Criando grupos eu consigo organizar melhor os meus usuários dentro da organização (empresas)
* **ROLES:** Regras que você associa, tanto pra usuários (Users), quanto para Groups, Regras de Acesso, e regras de serviço.

**Por exemplo:** Imagine que você tem uma web Server e este servidor hospeda uma página, e essa página tem um link com um Banco de Dados (Data-base), onde o Usuário que está acessando a página (Client) ele consegue acessar a página dele, e acessar as informações!

Só que o formato da página está no Web Service, e as Informações estão no Banco de Dados.

Conseguintemente, eu tenho 02 serviços aqui: um chamado EC2, e o outro DB e o serviço de EC2 tenque acessar o serviço de DB eles são 02 Instancias de serviços diferentes com uma frequência muito grande.

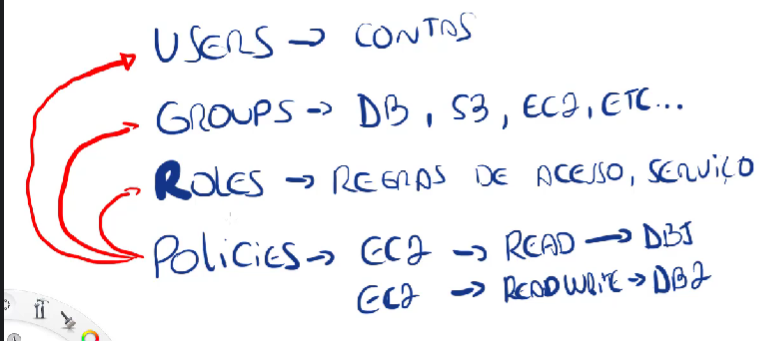
Conseguintemente ele tenque ter Regras (Roles) ou permissões, eu tenho que aplicar uma regra, olha o EC2 pode acessar o DB.

**Então:**

São regras de acesso a serviços que pode ser aplicada a usuários, quanto pra grupos e serviços também.

* **Polices:** O que esse usuário deve fazer, então eu crio uma police e falo: olha esse servidor EC2, ele pode acessar somente em Leitura (Read) Data-base I, esse EC2 tem Read-Write e pode acessar o Data-Base II e por ai vai.

**A Police você pode aplicar a:**



Resumindo:

Você cria os usuários, você divide eles em grupos de usuários, associa regras (Roles), e você define dentro das políticas o que eles podem fazer, consigo gerenciar 100%!

Laboratórios:

Hands On IAM Groups:

User Groups -> A melhor forma de criar e separar os departamentos ou funções e por grupos.

Posso segmentar os grupos, por funções ou Departamentos.

Interessante criar por funções, serão criados 02 Servidores, um para Administrador de Banco de Dados DBA, e outro para Servidores.

2 Grupos, um primeiro grupo para quem trabalha com server (EC2), e o Outro DataBases

Criando os grupos eu vou adicionar as Polices, e depois os Users com as minhas regras.

## EC2 INTRO

* Marquinas Virtuais ou (VMs)
* EBS-only (Elastic Block Storage)

Tipos de maquinas de acordo com sua categoria:

**General Purpose.** Medim, large, xlarge, 2xlarge, 4xlarge, metal

**Computer Optimized**

**Memory Optimized**

**Accelerated Computing GCPU**

**Storage Optimized**

**Instance Features**

**Mersuaring Instance Performace**

Bursty – colocar acima do que está especificado na máquina.

# **General Purpose:**

Me general purpose eu tenho maquinas virtuais para uso geral.

Types: A1, T3, T3a, T2, M5, M5a, M5n, M5zn, T4, Mac.

Observation: instance storage: NVMe: Now Volaty Memory express Solid State Drive

Get example:

A1 VirtualCPU Memory Storage networking Perfomance (Gbps)

2 4Gib EBS-Only up to 10 (Gbps)

# **Computer optimazed**

Se você tem Servidores, onde fazemos procuras, parte computacional um pouco maior, (marchine learning). Quantidade maior de cálculos.

**EBS – Bandwidth:** Quantidade de Banda para acessar o Disco.

Get example:

C5.large vCPU Memory Gib instance Storage networking Bandwidth(GB)

2 4 EBS-Only Up to 10

**MEMORY OPTIMAZED**

Dedicado para Servers para Armazenar Data-base com Hight Memory, focando bem a parte de memória nestes tipos de Instances.

**Name Processors Memory (GiB) Instance Storage (GB) networking Bandwidth**

**u-3tb1.56xlarge 224 3,072 EBS-Only 50 (Gbps)**

# **Accelerated Computing (Graphics Processor Unit)**

Instance GPUs vCPU Mem (Gib) GPU Memory (Gib) Networking performace

P2.xlarge 1 4 61 12 Hight

# **Storage Optimazed**

Tipos de instancia voltados especificamente para Espaço em disco e armazenamento dedicado.

Get exemple:

Instance size vCPU Memory (GiB) Instance Storage networking Bandwidth (GB)

Im4gn.xlarge 2 8 1 x 937 AWS nitro SSD Up to 25Gbps

**Instance Features**

Existem especificamente 02 tipos de maquinas, são maquinas que tem **Fixed Performace Instances**, se a máquina tem por exemplo: 5 processors, ela chega ao máximo de 5 processadores e para por lá.

Se você utiliza uma máquina Burstable Performace Instance, por exemplo uma máquina T3, a perfomacer dela pode ir um pouco acima do que está na especificação (Specific)

**Measuring Instance performance**

Como você mede a performance de cada máquina em CPus e em memória.

**EC2 Pricing:**

Valores das Maquinas Virtuais EC2 temos 02 métodos de cobrança, On-demand e Free-Tier

O método de cobrança free-tier você tem 12 meses (free), normalmente as maquinas com a instancia T2. micro por exemplo.

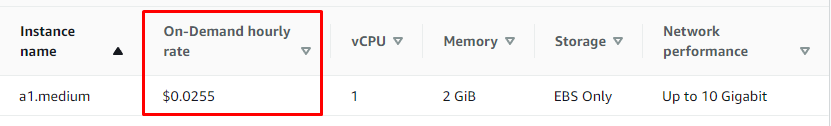
As maquinas ON-Demand são todas as outras maquinas que tem um valor:

Hoje em dia, a forma de cobrança, e através de per-second-billing feita em segundos.

* Pode ser por horas, se você escolher uma instancia reservada ou uma Instancia Spot (Dedicate Host) or Spot Instance.

Há AWS cobra por mês fechado, ou seja, o mês tem 750 horas então um exemplo de uma máquina ON-demand:

Get example:



Esse exemplo de instancia EC2 vai custar: 0.0255 X 750 Hours = US$ 19.12 Dollars.

Today date Dollars USD 21/03/2022 US$ 1.00 = R$ 5.04

US$ 19.12 X BRL5,04 = 96,36 Centavos em Real Brasileiro (R$)

Maquina EC2 no final do mês ela vai custar em BRL R$ 96.36

Free-tier.

On-demand influência nos valores $$ location (region), Operation System (S.O, Type de EC2 (small, large, xlarge)

[EC2 On-Demand Instance Pricing – Amazon Web Services](https://aws.amazon.com/ec2/pricing/on-demand/)

Custos de On-Demand Amazon AWS

# Entendendo Volumes Elastic Block Store ( EBS ):

Virtualização as Vms e você pegar uma máquina física, e dividir ela em cada parte.

Da mesma forma que virtualizo uma máquina física, consigo virtualizar um disco:

**EBS EBS**

**EBS EBS**

A AWS consegue pegar os discos físicos, e virtualizar esses discos.

Automaticamente, quando adquirimos um Volume EBS da Amazon, ela tem uma cópia desse volume idêntica. Em outra região.

# Variaveis do Sistema EBS:

HDD -> Dentro do tipo HD, temos 02 subcategorias: ST1 Thowput otimizado. SC1 Cold Disc

SSD -> Dentro do tipo SSD, temos IO1 (hyperFormace) e o GP2 (General Porpose Two)

Utilizamos um termo chamado IOPS que significa: Input Output Operation per Second

Significa o que: quantas operações ele consegue escrever no disco por segundo.

Temos também o **Throughput** que é a quantidade de Megabits por segundo que você consegue escrever no disco.

O Max IOPS / Instance:

# Entendendo Load Balances

Normalmente a AWS Prover 02 tipos de Load Balancers que são aplication networking e o clássico que e chamado de Elastic Load Balancer!

Porem vamos ver um conceito: o que é um Load Balancer:

Imagine que você está acessando de um computador, a sua requisição e traduzida através de um servidor DNS, ele passa qual endereçamento IP esta aquela requisição daquele site que vc está querendo.

O conteúdo que você quer está dentro de uma solução AWS, a empresa que detem o site, tem sua estrutura dentro da AWS, ela tem os servidores web:

Cliente -> DNS -> sua requisição traduzida Load Balancer -> Server1, Server2, Server3

A partir do momento que a sua requisição e apontada a um IP, esse IP de DNS não e o IP do servidor! Ele e o endereçamento IP do LOAD BALANCER

Temos o IP de Origem: 89.7.4.1 public e ele vai chegar ao endereçamento de IP 171.1.120

Quando ele chegar no LOAD BALANCER, o que ele vai fazer: ou dividir o trafego, ou ele vai enviar para o servidor que tem aquele recurso. (cada server tem seu endereçamento IP)

BALANCIAMENTO DE CARGA EM SERVIDOR COM CONTEUDO IGUAL.

A AWS prevê 03 tipos de Load Balancers:

* **Application Load Balancer:**

Ele sempre trabalha na camada de aplicação, então ele e responsável pelos protocolos de HTTP/HTTPS e trabalha na **Layer 7**. Ele consegue identificar a origem e o destino desse pacote. (consegue fazer o balanceamento porque ele sabe o tipo de trafego que está passando por lá)

* **Networking Load Balancer:**

Ele e mais pra parte de Networking Layer, parte de TCP e ele trabalha dentro da camada 4 (Layer 4)

Ele consegue filtrar portas, ele consegue filtrar origem e destino, mas ele para na camada 4 (Layer 4)

* Classic Load Balancer (Elastic Load Balancer)

Fazia o Http/Https -> Layer 4 e o Layer 7

Quando a sua requisição chega no Load Balancer, ele envia sua requisição para o server, como eles sabem que ele deve voltar para o servidor de origem.

Os Load Balancers utilizam um sistema chamado X-forward-for-header.

No cabeçalho da requisição Http/Https tem um campo aonde ele preenche esse campo com o endereçamento IP de Origem: 84.7.4.1 ele consegue identificar a origem e devolver

Https\_Header\_Field;

* Gateway Load Balancer

Seu cliente faz uma solicitação ao seu aplicativo. O Load Balancer recebe a solicitação com base nas configurações da tabela de rotas definidas em sua VPC, Internet Gateway ou Transit Gateway. O balanceador de carga roteia solicitações para um grupo de destino que consiste em uma frota escalável de dispositivos (por exemplo, firewalls, sistemas de inspeção profunda de pacotes, sistemas de filtragem de URL etc.) para processar fluxos de tráfego. O dispositivo virtual processa o tráfego e o encaminha de volta ao balanceador de carga ou descarta o tráfego com base em sua configuração. Esse tipo de balanceador de carga atua como uma ligação entre a origem e o destino. O balanceador de carga encaminha o tráfego para seu destino.

Leitura Complementar de Load Balancers:

O aplication Load balancers opera no nível de camada 7 (Layer 7) roteando o trafego a destinos e instancias do EC2, contêineres endereços IP e funções de Lambda de acordo com o conteúdo da solicitação.

Ideal para Load Balancers avançados de trafego HTTP e HTTPS, o application Load Balancer oferece um roteamento avançado de solicitações e entrega de arquiteturas de aplicativos modernos, incluindo microserviços e aplicativos baseados em contêiners.

O Application Load Balancer simplifica e aprimora a segurança dos aplicativos, garantindo que as cifras e protocolos SSL/TLS mais recentes sejam sempre usados.

Principais Recursos

--Balanceamento de Carga da camada 7 (Layer 7)

Você pode usar o Load Balancer é compatível com a terminação de HTTPS entre os clientes e o Load Balancer. O Application Load Balancer também oferece gerenciamento de certificados SSL por meio da AWS Identity and Acess Management (IAM) e do AWS Certificate Manager para políticas de segurança definidas!

--Server Name Indicator (SNI)

Indicação de Nome de Servidor, e uma extensão do protocolo TLS quer permite que o cliente indique o nome de host a ser conectado no início do handshake do TLS.

O Load Balancer pode apresentar vários certificados pelo mesmo listerner seguro, o que permite oferecer suporte a vários sites seguros, usando um único listerner seguro. Os Application Load Balancers também oferecem suporte a um algoritmo inteligente de seleção de certificado SNI. Se o nome de host indicado por um cliente corresponder a vários certificados, o Load Balancer determinará o melhor certificado a usar considerando os fatores, incluindo os recursos do cliente.

--Endereço IP como destinos

É possível balancear a carga de qualquer aplicativo hospedado na AWS ou no local usando os endereços IP dos back-ends aplicativo como destinos. Isso permite fazer o balanceamento da carga para um back-end de aplicativo hospedado na mesma instancia.

Cada aplicativo hospedado na mesma instancia pode ter um grupo de segurança associado e usar a mesma porta.

Você também pode usar endereços IP como destinos para balancear a carga de aplicativos hospedados localmente (por meio do Direct Connect ou de uma conexão VPN) em VPCs emparelhadas e no EC2-Classic (usando o ClassicLink). A capacidade de balancear carga entre recursos na AWS local, ajuda a executar migrações, de bursting ou failover para a nuvem.

--Funções de Lambda como destinos

O Application Load Balancer é compatível com a invocação das funções do Lambda para atender as solicitações HTTP(S), permitindo que os usuários acessem os aplicativos sem servidor de qualquer cliente HTTP, incluindo navegadores da Web. Você pode registrar funções Lambda como destino para um Load Balancer e utilizar o suporte de regras de roteamento baseado em conteúdo para rotear solicitações para funções de Lambda diferentes.

Você pode usar um aplication Load Balancer com um endpoint HTTP comum para aplicativos que usam servidores e computação sem servidor.

Você pode criar um site inteiro usando funções de Lambda ou combinar instancias do EC2, contêineres, servidores locais e funções de Lambda para criar aplicativos.

--Alta Disponibilidade

Um aplication Load Balancer exige que você especifique mais de uma zona de disponibilidade (Availability zone). Você pode distribuir o trafego de entrada entre os seus destinos em várias zonas de disponibilidade.

Um aplication Load Balancer escala automaticamente sua capacidade de gerenciamento de solicitação de resposta ao trafego de entrada de aplicativos.

--Recursos de Segurança

Ao utilizar um Amazon Virtual Private Cloud (VPC), você pode criar e gerenciar grupos de segurança associados ao Elastic Load Balancing para disponibilizar opções adicionais de rede e segurança.

Você pode configurar um aplication Load Balancer voltado para internet ou criar um Load Balancer interno (não voltado para a internet)

--Roteamento Baseado em Conteúdo

Se o aplicativo for composto por vários serviços individuais, um aplication Load Balancer poderá rotear uma solicitação para um serviço com base no conteúdo da solicitação

--Roteamento Baseado em Host

Você pode rotear uma solicitação de cliente baseada no campo Host do cabeçalho de HTTP. Dessa forma, e possível rotear do mesmo Load Balancer para vários domínios.

--Roteamento Baseado em caminho

Você pode rotear uma solicitação de cliente baseada no caminho de URL do cabeçalho HTTP.

--Roteamento baseado em método HTTP

Você pode rotear uma solicitação de cliente baseada no valor de qualquer cabeçalho HTTP padrão personalizado

--Roteamento baseado em parâmetros de sequência de caracteres de consulta

Você pode rotear uma solicitação de cliente baseada na sequência de caracteres de consulta ou de parâmetros de consulta.

--Roteamento baseado em CIDR de endereço IP de origem

Você pode rotear uma solicitação de cliente baseada em CIDR do endereço IP de origem onde a solicitação for proveniente

--Suporte a aplicativos Conteinerizados

O Application Load Balancer oferece suporte aprimorado a contêineres, executando o Load Balancing entre várias portas na mesma instancia do Amazon EC2.

A estreita integração com o Amazon EC2 Container Service, disponibiliza uma oferta de contêineres gerenciada. O ECS permite que você especifique uma porta dinâmica na definição de tarefas do ECS, disponibilizando ao contêiner uma porta não utilizada quando programado a instancia do EC2. O programador do ECS adiciona automaticamente a tarefa ao Load Balancer usando essa porta.

--Suporte ao HTTP/2

O HTTP/2 e uma nova versão do Hypertext Transport Protocol (HTTP), que usa uma única conexão multiplexada para permitir várias solicitações sejam enviadas na mesma conexão. Ela também compacta os dados de cabeçalho antes de envia-los em formato binário e oferece compatibilidade com conexões SSL para os clientes

--Suporte ao WebSocket

As sticky session são um mecanismo usado para rotear solicitações do mesmo cliente para o mesmo destino. O Application Load Balancer oferece suporte ao sticky session usando cookies gerados por Load Balancers.

Se você habilitar as sticky sessions, o mesmo destino recebera a solicitação e você poderá usar o cookie para recuperar o contexto da session ou da sessão. Esse recurso de sticky sessions, ou stickiness, e definido por grupos de destinos

--Monitoramento operacional

O Amazon CloudWatch relata métricas do Application Load Balancer como número de solicitações enviadas para o Load Balancer e armazenar os Logs no Amazon S3 para análise posterior.

Os logs são compactados e têm uma extensão do arquivo gzip. Os logs compactados economizam espaços de armazenamento de largura de banda usada para transferências, além de serem uteis para os diagnósticos de falhas de aplicativos e a análise de trafego web.

--Registro de Log

Você pode usar o registro de logs de acesso para registrar todas as solicitações enviadas para o Load Balancer e armazenar os logs no Amazon S3 para análise posterior. Os logs são compactados e tem uma extensão de arquivo gzip. Os logs compactados economizam espaço de armazenamento e largura de banda usada em transferências, além de serem uteis para o diagnóstico de falhas de aplicativos de analises de trafego web.

Você também pode usar o AWS CloudTrail para registrar as chamadas de API Application Load Balancer na sua conta e entregar arquivos de log. O histórico de chamadas API permite executar analises de segurança, rastrear alterações de recursos e realizar auditorias conformidade.

--Proteção contra Exclusão

Você pode habilitar o recurso de proteção contra exclusão em um Application Load Balancer para impedir que ele seja excluído acidentalmente.

--Solicite o Monitoramento

O Application Load Balancer injeta o “X-Amzn-Trace-Id” um novo cabeçalho HTTP identificador personalizado, a todas as solicitações enviadas para o Load Balancer. O monitoramento de solicitações enviadas ao Load Balancer.

O monitoramento de solicitações permite rastrear uma solicitação por meio seu ID único conforme a solicitação passa pelos vários serviços que compõem sites e aplicativos distribuídos.

É possível usar o identificador de monitoramento exclusivo para descobrir qualquer problema de performance ou atraso na pilha de aplicativos com a mesma granularidade de uma aplicação individual.

--Firewall de aplicativos web

Você já pode usar o AWS WAF para proteger aplicativos web em Application Load Balancers. O AWS WAF é um firewall de aplicativos web que ajuda a proteger esses aplicativos contra exploits comuns na web, que podem afetar a disponibilidade do aplicativo, comprometer a segurança ou consumir recursos me excesso.

--Modo de início lento com algoritmo de balanceamento de carga

O Application Load Balancer aceita um algoritmo de balanceamento de carga round-robin. Além disso, o Application Load Balancer aceita o modo de início lento com algoritmo Round Robin que permite que você adicione novos destinos sem sobrecarrega-los com uma enxurrada de solicitações.

Com o modo de início lento, os destinos são ativados antes de aceitarem as partes de solicitações que são cabíveis a eles, com base em um período de aumento na quantidade especificado por você.

O início muito lento e muito útil para aplicativos que dependem do cache e precisam de um período de ativação antes de poderem responder a solicitações com performance ideal.

--Autenticação de Usuários

Você pode descarregar a funcionalidade de autenticação dos seus aplicativos no Application Load Balancer. O Application Load Balancer autenticará com segurança os usuários, conforme eles acessarem os aplicativos na nuvem.

O Application Load Balancer e totalmente integrado ao Amazon Cognito, o que permite que usuários finais façam autenticação por meio de provedores de identidade social como Google, Facebook e Amazon, e por meio de provedores de identidade empresarial, como Microsoft Active Directory via SAML ou qualquer provedor de identidade (IdP) em conformidade com o OpenID connect. Se você já tiver uma solução personalizada de IdP compatível com o OpenID connect, o Application Load Balancer também poderá autenticar usuários empresariais ao estabelecer conexão diretamente com seu provedor de identidade.

--Redirecionamentos

O Application Load Balancer pode redirecionar uma solicitação recebida de um URL para outro. Isso inclui a capacidade de redirecionar solicitações HTTP para solicitações HTTPS, o que permite cumprir metas de conformidade para navegação segura, alcançar uma melhor posição nas pesquisas e obter uma alta pontuação de SSL/TLS para o site.

Também é possível usar redirecionamentos para enviar os usuários a um site diferente, como de uma versão antiga de um aplicativo para a nova.

--Resposta Fixa

O Application Load Balancer pode controlar quais solicitações de cliente são atendidas pelos aplicativos. Isso permite responder a solicitações recebidas com códigos de erro HTTP e mensagens de erro personalizadas do próprio Load Balancer, sem enviar a solicitação para aplicativo.

--O que é Networking Load Balancer?

O Networking Load Balancer opera no nível de conexão (camada 4) ou Layer 4, roteando conexões a destinos e instancias do Amazon EC2, microserviços e contêineres em uma Amazon Virtual Private Cloud (VPC) de acordo com os dados do protocolo IP. Ideal para o balanceamento de carga de trafego TCP e UDP, Networking Load Balancer e capaz de processar milhões de solicitações por segundo, mantendo latências ultrabaixas.

O Networking Load Balancer e otimizado para gerenciar padrões de trafego súbitos e voláteis usando um único endereço IP estático por zona de disponibilidade (Availability zone).

Ele e integrado a outros serviços populares da AWS, como Auto Scaling, Amazon EC2 Containers Service (ECS), Amazon Cloud Formation e AWS Certificate Manager (ACM)

--Principais Recursos

Load Balancing baseado em conexões

Você pode fazer o balanceamento de carga de trafego TCP e UDP, roteando conexões de destinos como instancias do Amazon EC2, microserviços e contêineres

--Alta disponibilidade

O networking Load Balancer e altamente disponível. Ele recebe o trafego de clientes e distribui entre destinos na mesma zona de disponibilidade.

Além disso, o Load Balancer monitora a integridade dos destinos registrados e garante que o roteamento seja feito apenas para destinos íntegros.

Quando o Load Balancer detecta um destino com problemas de integridade, cessa o roteamento de trafego para esse destino e roteia o trafego para os demais destinos íntegros.

Se todos os destinos de uma zona de disponibilidade tiverem problemas de integridade e você tiver configurado destinos em outra zona de disponibilidade, O Networking Load Balancer fara automaticamente um failover para rotear o trafego para destinos íntegros nas outras zonas de disponibilidade

--Alto Throughput

O Networking Load Balancer foi criado para assimilar o crescimento de trafego e pode fazer Load Balancing de milhões de solicitações por segundo. Além disso, também absorve padrões de trafego súbito voláteis

--Baixa Latência

O Networking Load Balancer oferece latências extremamente baixas para aplicativos que dependem da latência

--Preservação de endereço IP de Origem

O Networking Load Balancer preserva o IP de origem do lado do cliente, o que permite que o back-end veja o endereço IP desse cliente. Esse endereço pode ser usado em procedimento adicional pelos aplicativos

--Suporte a IP Estático

O Networking Load Balancer fornece automaticamente um IP estático por zona de disponibilidade (sub-rede), fornecendo assim o seu próprio IP Fixo.

--Transferência de Carga TLS

O Networking Load Balancer suporta o termino da sessão TLS do cliente.

Isso permite que você descarregue tarefas de termino de TLS para o Load Balancer, preservando o endereço IP de origem para os seus aplicativos back-end. Você pode escolher entre políticas de segurança predefinidas para seus listerner TLS, a fim de atender aos padrões de conformidade e segurança. Pode ser usado o AWS Certificate Manager (ACM) ou AWS Identity and Acess Management (IAM) para gerenciar os certificados do servidor.

Você pode usar o SNI (Indicador de Nome do Servidor) para atender a vários sites seguros por meio de um único listerner TLS. Se o nome do host cliente corresponder a vários certificados, O Load Balancer selecionara o melhor deles, que será usado com base em um algoritmo de seleção inteligente

--Verificações de Integridade

O Networking Load Balancer oferece suporte a verificações de integridade de destinos de rede e aplicativos. A integridade da rede e baseada na resposta geral do destino de trafego normal. Se o destino responder lentamente ou não responder novas conexões, O Load Balancer marcará esse destino como não disponível.

As verificações de integridade dos aplicativos também podem ser usadas para obter mais detalhes. A sondagem periódica de uma URL especifico em um determinado destino, permite integrar a integridade do aplicativo real. Uma visibilidade completa das verificações de integridade e possível motivos de falha também e disponibilizada por meio de “códigos de Motivo” na API do Networking Load Balancer e nas métricas da Amazon CloudWatch anexadas as verificações de integridade de destinos. Esse recurso proporciona diagnósticos rápidos e deputação avançada.

--Failover de DNS

Se não houver destinos íntegros registrados no Networking Load Balancer ou se os nós do Networking Load Balancer em uma determinada zona apresentarem problemas de integridade, o Amazon Route53 direcionara o trafego para os nós de Load Balancer em outras zonas de disponibilidade.

--Integração com serviços da AWS

O Networking Load Balancer e integrado a outros serviços da AWS, como Auto Scaling, Elastic Containers Service (ECS), Cloud Formation, Elastic BeanStalk, CloudWatch, Config, Cloudtrail, CodeDeploy e AWS Certificate Manager

--Conexão TCP de Longa Vida

O Networking Load Balancer oferece suporte a conexões TCP de longa vida, ideais para aplicativos do tipo WebSocket.

--Suporte a APis centralizadas

O Networking Load Balancer usa a mesma API que o Application Load Balancer. Dessa forma, você pode trabalhar com grupos de destinos, verificações de integridade e usar o Load Balancer em várias portas na mesma instancia EC2 para oferecer suporte a aplicativos Conteinerizados.

--Monitoramento e auditoria Robustos

O Amazon CloudWatch relata métricas do Networking Load Balancer. O CloudWatch fornece métricas com Active Flow Count, Heath host Count, New Flow Count e Porcess Byte, entre outras. O Networking Load Balancer também e integrado ao AWS Cloudtrail rastreia as chamadas da API ao Application Load Balancer

--Registros em Logs aprimorados

Você pode usar o recurso de logs de fluxo para registrar todas as solicitações enviadas ao Load Balancer. Os logs de fluxo, capturam informações sobre o trafego IP e percorre as interfaces de rede na VPC, os dados de Logs de fluxo são armazenados usando o Amazon CloudWatch Logs.

Quando os listerner TLS são adicionados ao Networking Load Balancer, e possível optar por ativar os Logs de acesso, que serão enviados para o bucket do S3 de sua preferência. Os Logs de acesso capturam informações sobre todas as conexões TLS recebidas por seu listerner, proporcionando visibilidade dos Hand shakes TLS bem-sucedidos e com falha.

--Isolamento Zonal

O Networking Load Balancer foi criado com arquiteturas de aplicativos em uma única zona. Se algo na zona de disponibilidade falhar, executaremos automaticamente o failover para outras zonas de disponibilidade integras.

Recomendamos que clientes configurem os Load balancers e os destinos em várias Avaibility Zones para obter alta disponibilidade. No entanto, O Networking Load Balancer pode ser habilitado em uma única zona de disponibilidade para oferecer suporte a Arquiteturas que exigem isolamento Zonal.

**Load balancers: Heath Check**

**Ping Protocol** [HTTP]

**Ping Port** 80

**Ping Path** [Index.html]

*Advanced Details*

**Response Timeout** O tempo de espera de resposta para o Heath Check!

Significa, que ele vai enviar uma requisição e se ele não receber uma resposta em 2 segundos ele vai contar aquela requisição com a falha!

**Interval** O tempo entre os Heath Checks de quanto em quanto segundos e pra ele ir lá e verificar no servidor se ele está disponível ou não.

**Unheathy threshold** Quantos TimeOuts são necessários, pra falar que aquele servidor caiu, se eu receber 2 TimeOuts de HeathCheck, significa que caiu.

**heathy threshold** quando ele voltar, quanto tempo eu preciso aguardar? 10 vezes

Route 53

Serviço AWS para DNS, se você precisa trabalhar com Domain Name Server, você pode utilizar o DNS.

O que o ROTA 53 pode fazer, primeiro ele e responsável pelas resoluções de endereçamento IP e necessário um endereçamento IP pra fazer essa tradução, dentro do Route53, ele não trabalha de forma isolada.

Por exemplo: se você registra o seu domínio e hospeda o seu site, em uma região nos EUA por exemplo, a replicação dessas informações ela e por toda a AWS Cloud.

Você terá uma redundância, de todas as localizações, e você tem garantido por eles 100% SLA (Availability) quando os serviços de DNS, estiverem dentro da **Rota53**, essas informações são passadas para todos os outros servidores de todas as regiões, de todas as **Availability zones**, se você tem um usuário no Brasil, e o registro foi feito nos EUA, essa replicação e passada ao Brasil também.

E quem faz essa requisição, ele não precisa ir em todos os servidores de DNS até chegar naquele registro. Todos os registros estão em todos os servidores.

Voce tem uma disponibilidade de SLA de 100% que a própria **AWS** garante.

**Ponto Importante:**

Redundância de todas as localizações:

E que dentro do Rota 53 você pode habilitar que nós chamamos de **Geo\_location,** então você pode filtrar, quem acessa o seu site ou quem não acessa o seu site através de Geo\_Location

**Failover**

Temos esse sistema, se você tem 02 servidores web: o servidor 1.1.1.1 e o servidor: 2.2.2.2, através de regras de DNS, você pode apontar para um servidor, e se ele se tornar indisponível, ele automaticamente passa para os outros.

E claro que tem todo um sistema de cache, que as vezes pode demorar algumas horas, mais o que a rota 53 faz isso automaticamente, e você não precisa ficar adicionando

**Domain Name Servers:**

DNS –

Domain (Domínio) é nada mais nada menos que o nome do seu site.

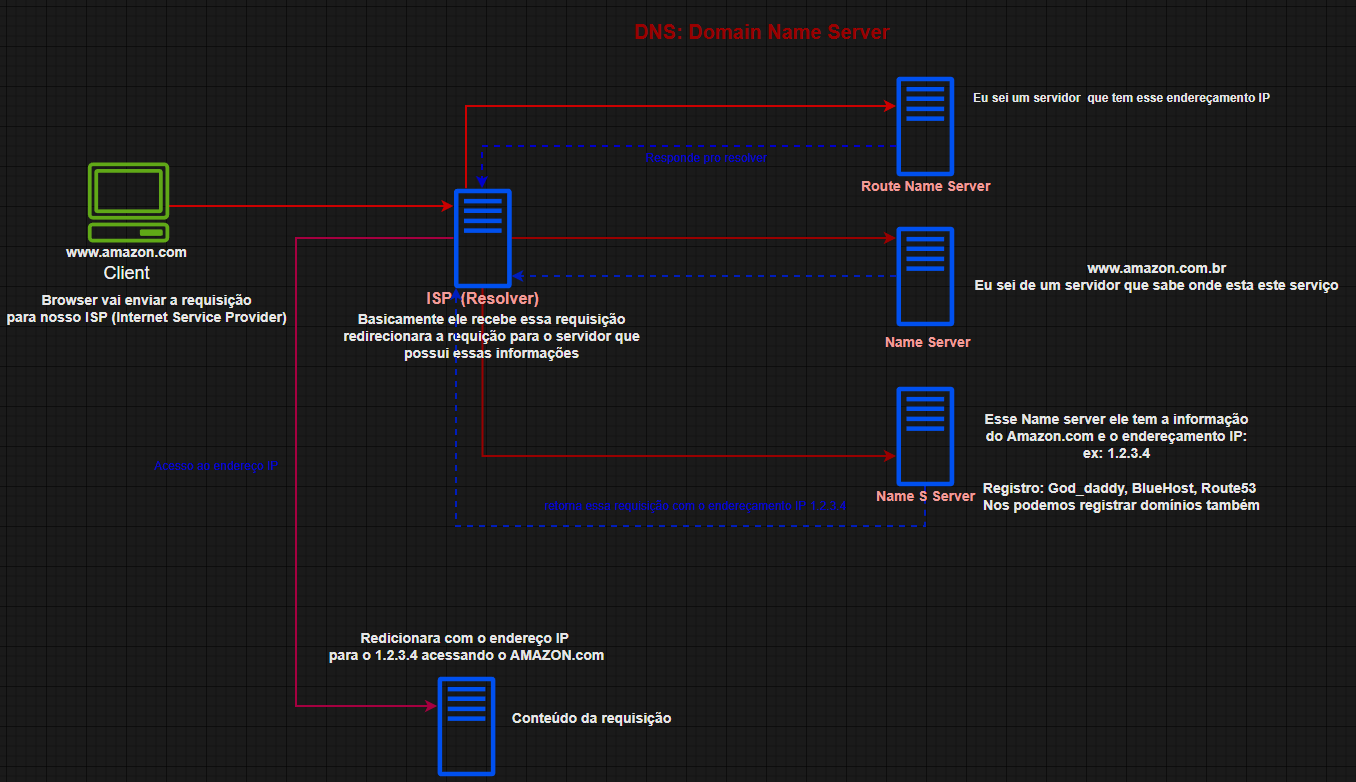
Porém existe uma diferença muito grande de alguém querer acessar o seu site, e chegar nos arquivos onde estão hospedados o seu site!

Basicamente, o Domain que pode ser [www.amazon.com.br](http://www.amazon.com.br) para o número IPv6: 192.168.1.1 seja ele IPv4 e IPv6

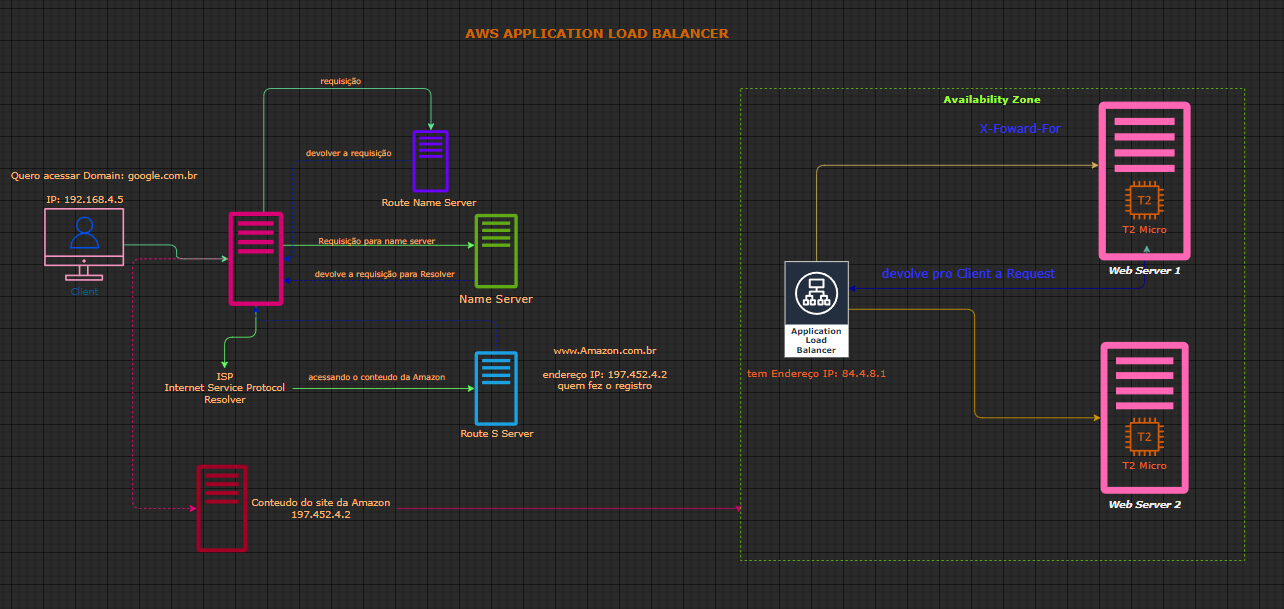
Como isso tudo funciona:

O computador: O browser vai enviar a requisição, para os registros de DNS, normalmente um PC ele está com 2 DNS apontados. Normalmente ele aponta para um grupo de servidores, vamos dar o nome de Resolver, normalmente este Resolver ele está dentro do ISP (Internet Service Provider) que é seu provedor.

Exemplificando no Grafico o DNS:



Com Load Balancers:



Registros

Visualizar um registro chamado Host A (pode ser definido como) e um tipo de registro dentro da arvore de DNS

Tipos de Registros:

* Host A (A ou AAAA) o que e o Host A, e um tipo de registro onde dentro da arvore de DNS, O site completo ele está apontado ao endereçamento IP: 1.1.1.1 (aonde está os arquivos desse site em especifico)

O A e para IPv4(A) e p IPv6(AAAA) esse registro pode ser usado para internos como por exemplo: você tem o servidor para AD server 1, esse endereçamento IP192.168.10.1 (supondo que seja o servidor de Active Directory)

* Aliás (Cname) OU Canonical Name, ele e muito utilizado em conjunto com o Host A:

Digamos que eu queira um Subdomínio (subdomain) ele está apontado para 1.1.1.1

* Mail Exchange: Registros de Email, onde está o servidor de e-mail, então colocamos através do MX.amazon.com, você também consegue colocar um número aqui como prioridade 5 por exemplo.

Eu posso ter vários servidores de e-mail, com prioridades diferentes:

Mx.amazon.com 5

Mx.amazon.com 10

Então toda vez que eu visualizar um registro MX, significa que os Email serão enviados e encaminhados para aquele servidor em Especifico

* SRV (Service Record) Se você tem algum serviço, e precise que as aplicações aprendam sobre este serviço

Quando você tem um SRV apontado para o endereçamento IP (SRV) ele indica qual o serviço está rodando: LDAP, ele informa também o que e o Target: Endereçamento IP, informa a Port: 8080, e a Priority (prioridade) qual e a Menor (onde e o primeiro a ser utilizado)

-Start of Authority (SOA) por Zona, normalmente ele e o Name Server (Primary Name Server)

Dentro do Soa: nós temos informações por exemplo como o Name Server, Qual e o administrador que registro aquele domínio (se a informação não estiver como privado) SN (Serial Number) que toda vez que você altera um registro de DNS esse número aumenta por 1, e lá dentro do SOA, tem as informações de Refresh Time, atualiza as informações e tabelas com os seus Records!

* Name Server: NS

Nós temos nada mais nada menos que as informações dos Records, e a melhor origem de todas as informações que eles podem estar, basicamente ele armazena o Start of Authority o (SOA)

* Pointer (PTR)

Ele basicamente faz o oposto que o Host A faz, para um registro de nome, ele dá o endereçamento IP, o Pointer pega o endereçamento IP (1.1.1.1) e converte para um domínio (name)

**Lab: Route 53**

DNS management: Através dela eu consigo gerenciar as minhas regras de DNS, que são os DNS contidos no DNS Records: Host A, Cname, MX, SRV, NS, Pointer

Traffic management: Consegue gerenciar o trafego, qual trafego vai pra que lado, conseguimos aplicar as políticas (policys) de peso, simple Router, eiteh Router, Failover (pra encontrar o servidor que está disponível)

Availability Monitoring: Onde Monitora onde os servidor estão ativos e qual não estão!

Register Domain: Onde registramos nossos domínios.

Configurando Topologias com Servidores Web.

100% baseado em como aumentar sua disponibilidade do seu site, ou da aplicação web.

Vamos montar 03 servidores:

Em localizações diferentes a que eu estou:



Com as 03 maquinas Virtuais Criadas, vamos coletar o endereçamento IP das Regions de London, Paris, Sidney

London Ipv4 Public: **18.130.106.37**

Paris Ipv4 Public: **35.180.42.206**

Sidney Ipv4 Public: **13.55.212.177**

Vamos tentar acessar através do endereçamento IP public.

Com os 03 Sites Funcionando.

Trabalhando com métricas no Route53

Simple Routing: vou enviar uma requisição de DNS para cada servidor (sendo que cada usuário vai acessar um servidor diferente de acordo com o DNS)

Failover Routing: Quando um se torna indisponível ele faz o giro pro outro.

GeoLocation Routing: Rotear usuários baseados na sua localização ´pra poder encontrar o servidor mais próximo, e esse servidor mais próximo se não estiver disponível, ele vai pro outro como segundo caminho.

GeoProximity Routing (Traffic Flow only) Ele vai pelo recurso, pelo servidor mais próximo.

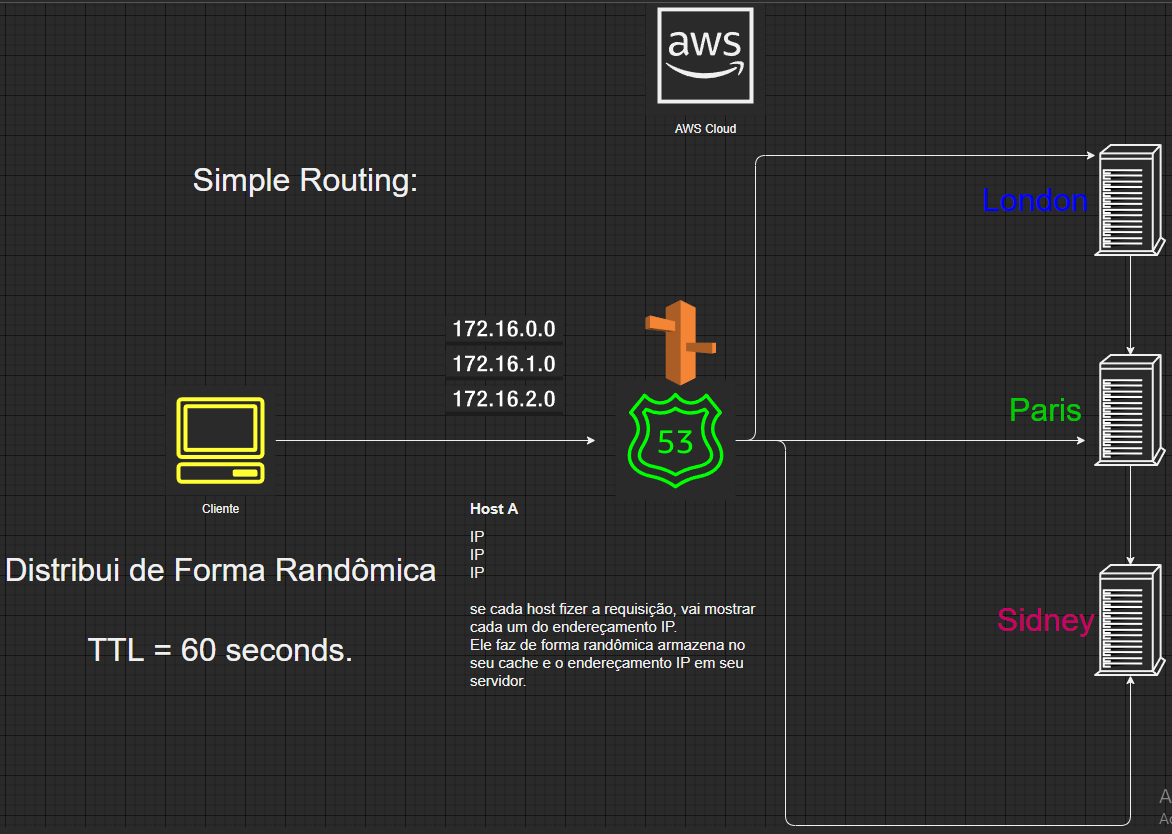
Latency-Based Routing: Ele vai pela Latencia, de acordo com a latência, se esse servidor está próximo, ou tem uma latência menor ele vai pela menor

Multivalue Awser Routing: Se pode passar vários endereçamentos IP para mesma URL

Weigthted Routing: Um peso, pra esse servidor vou enviar 80% trafego, e o outro 20% do trafego.

O TTL (Time to Live) Quando o cliente enviar pro DNS e o ROUTE53 vai falar assim, pra você chegar neste endereçamento IP 18.10.106.37, quando ele passar essa informação, o Time to Live quanto tempo em segundos, essa informação passada para o cliente, vai ser armazenada no cache do cliente.

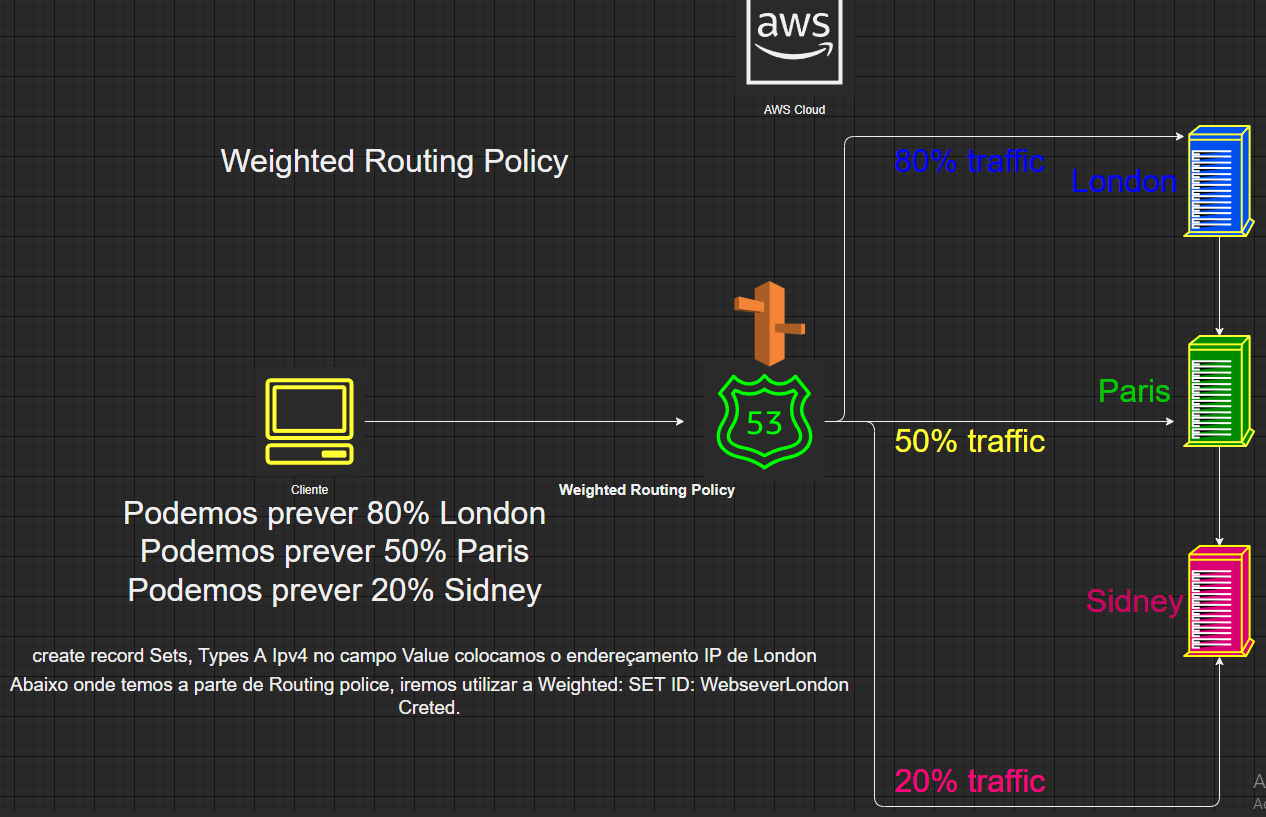
Create the Roles.

Simple Routing Policy

WEIGHTED ROUTING POLICY:

Vamos tirar a Policy anterior, indo em Delete Records Set

Criaremos uma Nova Policy Create Records Set

Não esqueça de alterar as TTL (Time to Line de cada Server) nas regras de Políticas. d

**Weigthted Routing Policy com Health Check!**

Temos a opção de criar um health Check, caso algum dos servidores se tornem indisponíveis, suponhamos que o server de London se torne indisponível, tem a funcionalidade de Setarmos um Health Check para verificar isto.

**Associate with Health Check: Yes.**

Configure to Health Check:

**Name:** HC-London

**What Monitor:** Endpoint

**Monitoring and Endpoint:**

**Specify:** IP Adress

**Protocol:** HTTP

**IP Adress: 18.130.106.37**

**Host Name: WWW.CLUBECLOUDBRASIL.COM**

Port.: 80

Path: INDEX.HTML

**Create Heath Check**

Depois que criarmos os 03 Health Check dos 03 Servidores, faremos a associação deles dentro do DNS Records

Derrubando o Server Sidney na EC2 (Stop), e verificando o nosso Health Check, definindo as configurações de Interval e as demais. Ele vai estar Unheath

**Latency Routing Policy**

**Se temos, 03 servidores, e queremos que os usuários sempre acessem os servidores mais próximos: através do Latency Routing Policy.**

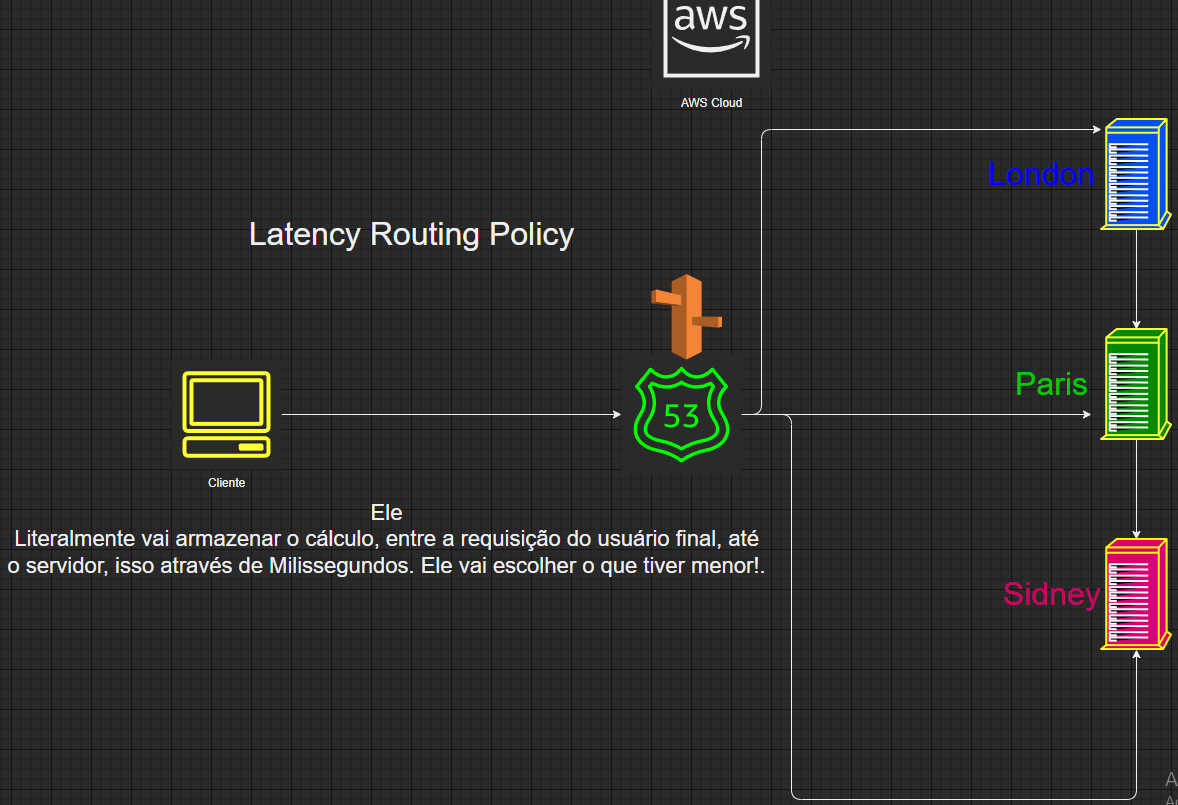
Ele Literalmente vai armazenar o cálculo, entre a requisição do usuário final, até o servidor, isso através de Milissegundos. Ele vai escolher o que tiver menor!

Utilizando o CMD, iremos bater o ping 18.130.106. 37 -t

Security Group Edit inbound Rules, criar um ICMP Custom ICMP 4 echo Request

**Security Group Edit inbound Rules, criar um ICMP Custom ICMP 4 echo Request**

**Fazer a mesma coisa com os Security Groups e adicionar o ICMP pra todos**.



Criaremos o Primeiro Records:

Routing policy: Latency

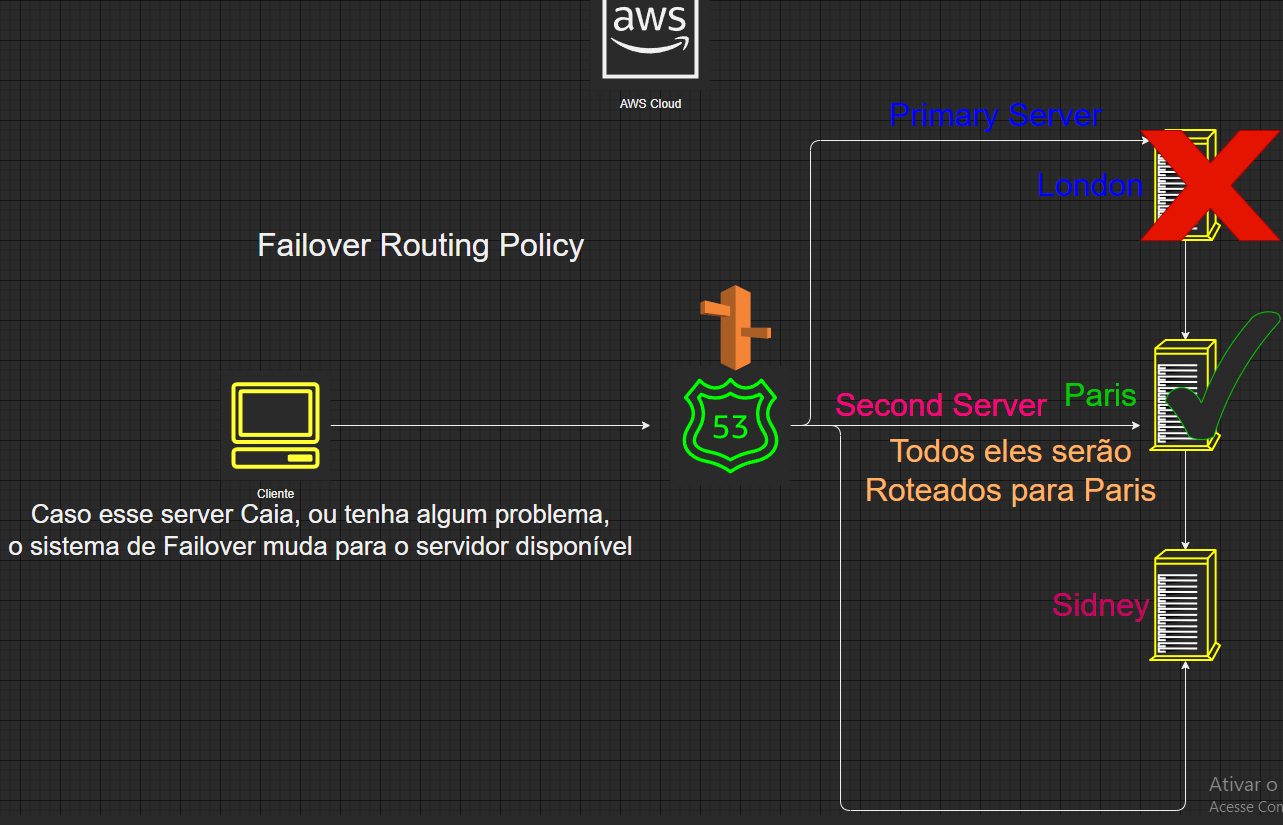
Region: eu-West-2

Set ID: London

Habilitando o Health Check em Londres que já estava criado anteriormente no Laboratório. Ele ira verificar a Menor latência assim como fizemos com o ping, utilizando o endereçamento IP (Para isso precisaremos mudar no Security Group e criar um ICMP 4 e deixar com echo Request

**Failover Routing Policy**

Os usuários que fizerem a requisição: Request para a Route53 para o domínio clubecloudbrazil.com



Create Record Sets no AvaibilityLondon com um TTL de 1 minuto.

**Value: 18.130.106.37** cole o endereçamento IP de London.

**Routing policy: Failover**

**Failover Types:** Primary or Secondary?

**Set ID:** Primary

Create Record Sets no AvaibilityLondon com um TTL de 1 minuto

**Value:** **35.180.42.206**

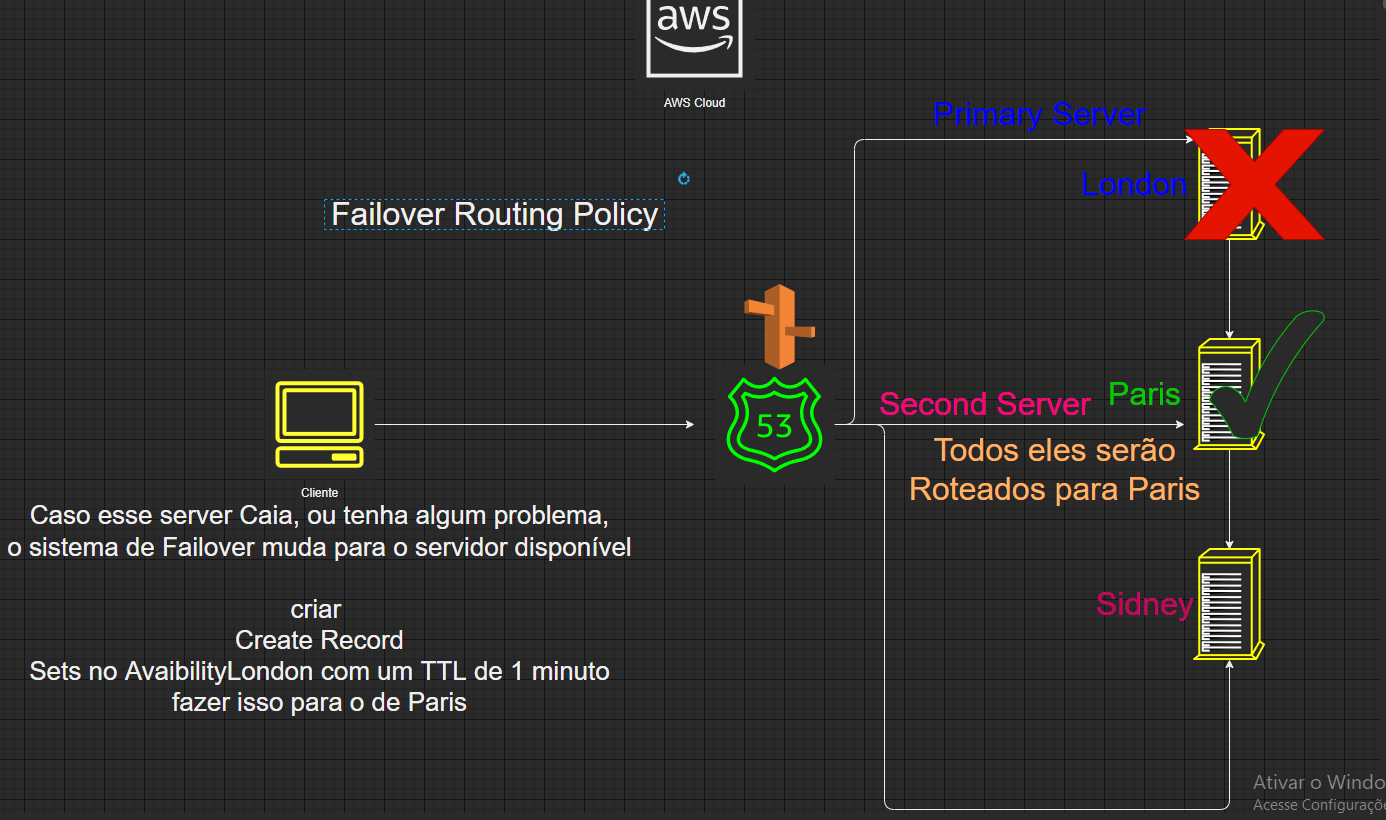
Associate with Heal Check: Yes

Health Check Associate: HC-London

Create novo Records Sets, para Paris, com TTL 60 seconds e Failover Types: Primary or Secondary?

Set ID: Secondary

Heath Check Associate: HC-Paris

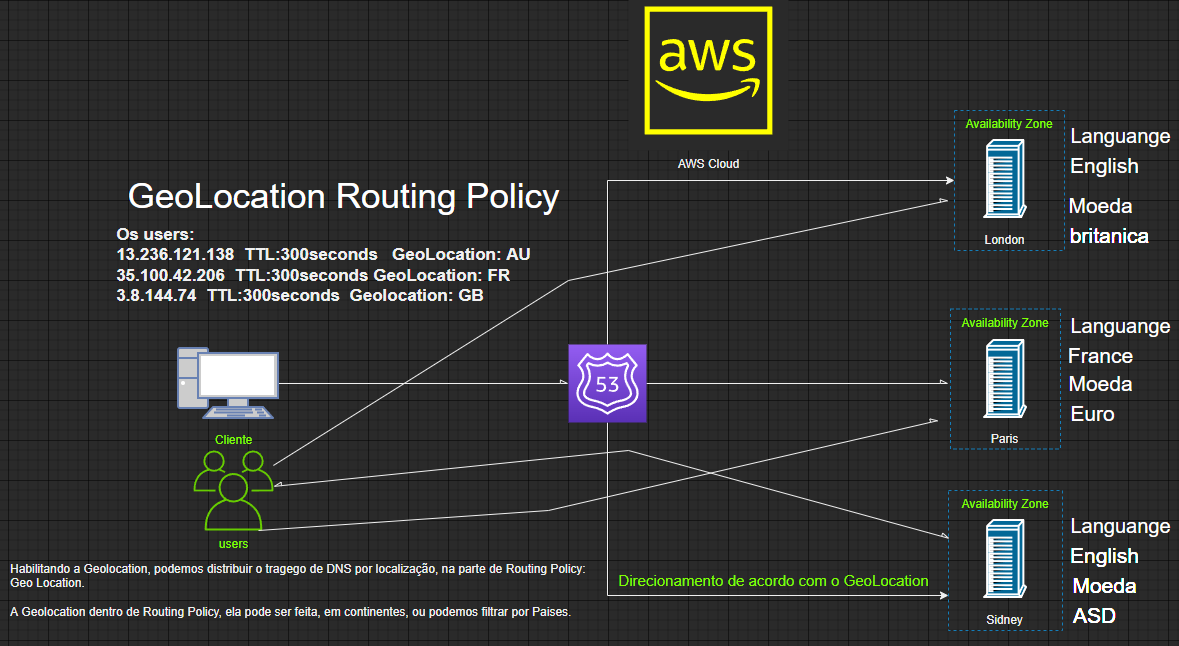


**GeoLocation Routing Policy**

Habilitando o Geolocation, podemos distribuir o trafego de DNS por localização, na parte de Routing policy: GeoLocation

A geoLocation dentro de Routing Policy, ela pode ser feita, em continentes ou podemos filtrar por países.

**Criação de 03 Geo\_location: uma para cada pais, mas funcionando e esse e o cenário que posso fazer de acordo com a nossa área**

**.**

**GeoProximity Routing Policy**

Ele funciona basicamente por Traffic Flow: significa saber exatamente onde o trafego está passando, e podemos adicionar politicas no meio do caminho para esse trafego!

Ele e baseado em localização de GeoLocation por GeoProximity, podemos fazer que usuários de determinada região, sejam roteados para aquela região, e podemos fazer também aplicar um failover, dentro da mesma regra: Então fazemos 02 tipos de Policy, dentro da Mesma regra. Podemos fazer várias Policy.

Criando um Traffic Policy:

Policy Name: Europe

Version:

Version Description:

Ele abre uma janela, possibilitando-me trabalhar com o Flow do meu Trafego, por onde meu tráfego chega, e como eu posso tratar o trafego.

Escolhendo a GeoProximity Role, posso trabalhar inicialmente com duas regiões, sendo que a primeira região eu falo pra ele: eu quero rotear esse trafego para um destino, a partir da regra de proximidade, eu posso passar as coordenadas de latitude e longitude, ou eu posso escolher as coordenadas:

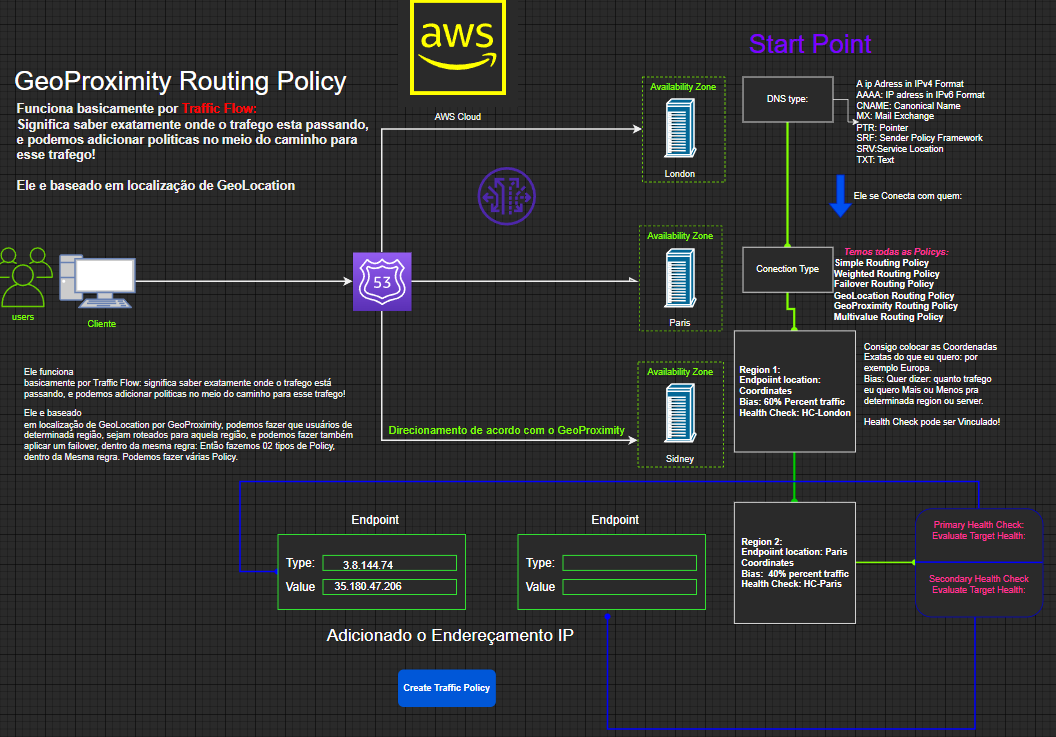
Europe London.

Bias (significa o que: quem bater nesta de London, eu quero mais trafego, saindo, ou menos trafego saindo.

Se o trafego for ligado direto, ao servidor de Londres, ele pode ser ligado diretamente a este servidor:

Pra ficar ainda mais sofisticado: posso ainda adicionar um Failover Roles: dizendo que de acordo (Baseado) com o Health Check eu vou apontar para o endereçamento IP.

Ou baseado no Health Check de Paris eu quero que ele saia a um EndPoint de Paris.



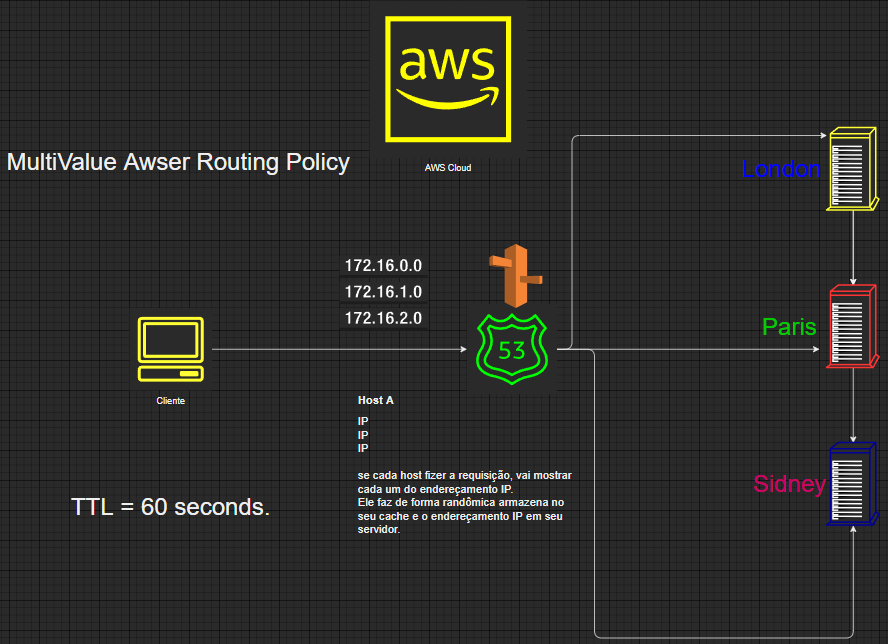
Esta policy tem um Custo de 50$ **Dólares** por Mês, esse tipo de Policy ela e cobrada.

Remova o tipo de Policy.

**E a habilidade que você tem de visualizar o Traffic Flow e setar regras com Politicas**

**Multivalue Awser Routing Policy**

Entendendo bem o Simple Routing Policy, eu criando os Heath Check consigo setar o Multivalue Awser Routing Policy



**Amazon S3**

**Simple Storage Service**

O Amazon S3 foi criado para armazenar arquivos, tipos de arquivos: MP4, Docs., Jpeg, etc... qualquer tipo de arquivo você pode guardar dentro do Storage S3.

* Existe um Limite máximo de armazenamento de 5 TB. (por arquivo)
* Storage ele é Unlimited: Não tem Limite, você pode armazenar a quantidade de arquivos que você tiver dentro da S3, você pagará por Gigabytes.

Discos dentro da Amazon S3 se chamam **Buckets. (discos, ou cestas) poderá criar suas folders.**

Importante!

Quando você cria uma Bucket, você precisa dar um nome pra essa Bucket, e o nome precisa ser exclusivo mundialmente, quando você cria a sua Bucket isto gera um DNS, e conseguintemente uma URL exclusiva para sua Bucket.

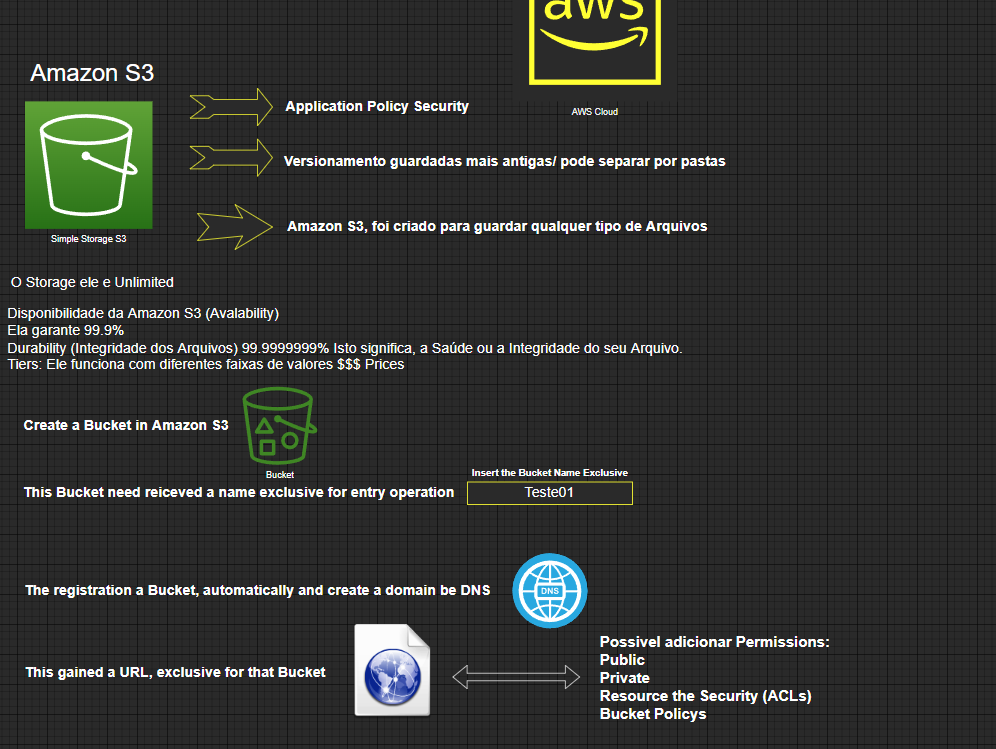
Exemplo de uma Bucket: https://s3-us-west-2.amazonaws.com/nome\_da\_bucket/folder/nome\_do\_arquivo.Doc

Disponibilidade da Amazon:

* **Avalability**: Ela garante 99.9% de Avalability.
* (Integridade) **Durability:** 99.9999999% - Significa, a saúde ou a integridade do seu arquivo.
* Ele funciona com diferentes **Tiers**, significa diferentes preços $$
* **Versioning** (você pode fazer o upload de um arquivo hoje, mudar o arquivo amanhã no dia seguinte, no terceiro dia se faz outra alteração do arquivo. (Você consegue armazenar em pastas diferentes, ou em folders diferentes) ele vai colocando uma **TAG**, ou uma **Label** que é uma identificação e vai falando qual arquivo e de qual versão!
* **Encryption:** Todos os seus arquivos você deve habilitar os arquivos, que se tornem 100% seguros, outros deveram decriptar seus arquivos para acessa-los.
* **ACL**: Acess Listes, você pode aplicar a um arquivo, se você tem um arquivo de vendas.xls quem pode ter acesso a esse arquivo: somente o grupo ou departamento de vendas, ou somente 04 vendedores do departamento de vendas, você consegue fazer isso através e utilizando as ACLs
* **Bucket Policie:** Por exemplo, você cria um Bucket do depto de vendas: Eu posso aplicar uma policy nessa Bucket, permitindo somente o grupo de vendas, todos os arquivos que estiver lá dentro, eles podem ser acessados pelo pessoal que está dentro do grupo de vendas V.

**Mais não pode ser associado ao pessoal do Financeiro X**

Nota: Se você quiser criar camadas de segurança, ou camadas de complexidade, mas ela acaba ainda de armazenar arquivos na sua conta AWS!



Componentes

Componentes da estrutura S3:

Object Base, ele e baseado em objetos, baseado em arquivos, quando você faz um upload de um arquivo, ele sempre faz a leitura de um arquivo, e depois a escrita do arquivo.

Instancia – Standard (CloudFronted) – ele pega esse seu arquivo e envia para outras regiões, para os outros datacenters, para outras Zonas de Avalability, dependendo do tipo de armazenamento que você usa.

Componentes da Estrutura do S3:

1° Key - Significa que ele irá armazenar baseado no nome do arquivo gera uma key

2° Value – E considerado para Amazon os dados do arquivo (conteúdo do arquivo)

3° Versioning – Version, essa versão pode salvar uma planilha e dentro ela coloca v1.

4° Metadata – Campo reservado dentro do S3, que ele armazena dos dados, informações que são colocadas no topo do arquivo (TAG) local onde o arquivo está!

5° ACLs – Ela e relacionada ao arquivo em Si

6° Bucket Policy – que e relacionada a própria Bucket.

S3 Tiears – Pricing:

Existem 04 tipos de camadas de cobrança que significam o seguinte cada:

* S3 Standard: Jeito padrão de armazenamento que a Amazon tem garante 99,9 Avalibility(disponibilidade) e 99,999999999 Durability (podem ser acessados em tempo real)

Quando subimos para uma região ele e replicado no mínimo 02x Avalability Zones.

Cobrado por GB

* S3 IA: Infrequent Acess: vai armazenar, e não vai acessar com tanta frequência, toda vez que você acessa o arquivo, você paga uma pequena (fee)$$, também e armazenado em duas Avalability Zones
* S3 One Zone: Cobrado uma pequena fee$$ por cada acesso, mas ele só está em 01 Avalability Zone, arquivos que você acessa em uma frequência menor ainda.

* S3 Glacier: Ele e pra Backup, ele e muito barato, só que esse acesso essa (fee $$) por cada acesso se chama **Retrieval** ele tem um intervalo, ou pode demorar de 3 a 5 horas para o arquivo estar disponível.

Como a Amazon faz a cobrança:

1°- Primeiro você e cobrado pelo Storage: Não importa a **Tiear** que você tenha, você vai pagar por GB gib.

2°- dependendo do Tiear que você escolhe, não sendo o Standard, você paga por request, paga por acesso, peer by Acess.

3°- Você paga também por Storage Management Price (TAGs), local do arquivo, nome de um departamento você paga pelo gerenciamento.

4° - DTP – Data Transfer Princing: (Transferência de Region para Region)

As Vezes você tem uma máquina EC2 em E - West, e você guarda a instancia S3 em U-West, e por alguma razão eu crio outra máquina EC2, com uma instancia S3 e as duas estão na E-West e a segunda está na E-West.

* Na primeira, você vai pagar mais que a segunda, pois você está pagando **Data Transfer Princing.**

5° Transfer Acelleration: Quando você habilita CDN (Contact Delivery Networking) ou quando falamos (CloundFront)

Nota: **Http 200 e o código de confirmação e o código de confirmação para sucesso de operações que você faz dentro da Bucket S3**

AWS CLi

Instalando e configurando a AWS Comand line Interface, depois de fazer o download, será necessário executar um passo a passo para acessarmos a Nuvem AWS:

Para verificar a Version da AWS instalada na sua Máquina, utilizamos o comando dentro do CLI

aws --Version

* 1° Para Iniciar a conectividade entre o seu PC e a Nuvem AWS, através do comando:

- Aws configure.

Ele vai pedir a Acess Key ID:

Depois que ele pedir a acess key ID, ele vai pedir uma Secret:

1° User e uma 2° Password.

2° Criando um User no IAM, chamado rep1, isto irá gerar um acess key e secret key.

3° Depois de Inserir estas informações ele irá pedir a Region: us-west-2

4° Por ultimo ele irá pedir o tipo de saída das informações escolha entre: text, json, table

Pronto, estamos conectados a AWS.

CROSS – REGION - REPLICATION

Dentro do S3, precisamos criar uma nova bucket, com o nome de andrepanizzabkp.

Esta nova bucket, será criada com as Permissions padrão, sem acesso externo public Not Make Public.

Note: que iremos criar a Bucket andrepanizzabkp em outra Region: Frankfurt.

Dentro da Bucket andrepanizza, iremos acessar e entrar na guia Replication:

Pois devo apontar essa Bucket: AndrePanizza sendo a Replicação da Bucket andrepanizzabkp.

Vamos em Management: e clicaremos em Replication.

* Não adicionamos nem uma regra para replicação, e depois em Add Roles.
* Por pré-requisito, precisamos habilitar o sistema de Versioning para a Bucket

Set Source: Quem e a Origem da Bucket: andrepanizza. Ou posso filtrar por Tags.

Você terá a opção de replicação dos Objetos utilizando a AWS encripted KMS

Destination da Bucket: andrepanizzabkp.

Posso Alterar o Storage Class, significa eu posso guardar esses arquivos em um outro tipo de Storage(Tiears)

Andrepanizza – Standard $$$ Region: **us-ohio EUA**

Andrepanizzabkp – Não está usando nada Region: **Fankfurt**

Posso estar adicionando uma Tiears

Regras: E necessário criar uma regra. Para uma bucket acessar a outra:

IAM role

Create new role

Rules Name: s3-Replication

Esta regra vai estar habilitada ou desabilitada: essa regra e permitindo o acesso de uma Bucket para Outra Bucket

Save.

***Cross-region Replication Sucessfully***

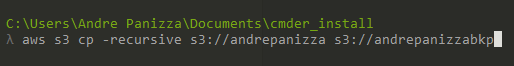
A regra da AWS e a seguinte, quando você realiza a Sincronização da Replicação, ele não copia os arquivos já existentes, ele cópia de agora em diante.

Vou ter que forçar via linha de comando que ele copie os arquivos já existentes, de uma bucket pra outra utilizando a CLI

Inserindo o comando para forçar a replicação dos arquivos já existentes:

aws s3 cp –recursive [bucket de origem] [bucket de destino]

aws s3 cp –recursive s3://andrepanizza s3://andrepanizzabkp



Pro exame:

* Se na minha bucket principal, eu deletar um arquivo, ele não deleta o arquivo na bucket de replicação: o Cross-Replication não funciona para delete files.

Porque: Essa e a forma com que a AWS construiu o sistema de Cross-Replication do S3, se você faz replicação de arquivos, e que você quer proteger os arquivos originais, com uma cópia, não faz sentido então acidentalmente, remover o principal, e a mesma ação for lá na cópia e remover

Nota: Se você quiser remover totalmente as duas Buckets, origem e destino [Os dois]

Criação de VPC

CIDR – e o bloco de Ip que você vai fazer para essa VPC – O Bloco de IP dessa rede.

Classes A = 10.0.0.0

Classe B = 172.16.0.0

Classe C = 192.168.0.0

Por exemplo se criarmos o bloco de CIDR 10.0.0.0/32 sua máscara de sub-rede será:

255.255.255.255

-> /8/12/16/24/32 Normalmente na AWS utilizamos mais /16/24.

Lembrando que temos 4 Octetos que variam: 192.0.0.0.0 todos esses octetos variam.

Lembrando que a AWS reserva 5 Ips, um de networking, reservado, e Broadcasting então se tivermos 32 Ips, menos 5 teremos 27 Ips disponíveis.

10.0.0.0/24 seria ideal para redes públicas, pois não queremos muitos Ips nas redes públicas. 256 Ips seriam o suficiente, então na hora de definirmos nossas subnets utilizaremos como CIDR 10.0.0.0/16 e para nossas subnets com 256 Ips, usaremos 10.0.0.0/24 e para próxima subnet 10.0.1.0/24.

Então criamos nossa VPC-Demo com o CIDR: 10.0.0.0/16

E as Subnets:

PrublicSubnetA Az: us-VirginiaA IpCIDR: 10.0.0.0/24

PlublicSubnetB Az us-VirginiaB IpCIDR: 10.0.1.0/24

Isso irá garantir uma quantidade de IP Public de 256 em cada SubnetPublic que queremos.

Subnets Privadas:

PrivateSubnetA Az: us-VirginiaA IpCIDR: 10.0.16.0/20

PrivateSubnetB Az: us-VirginiaB IpCIDR: 10.0.32.0/20

Que contem 4.096 Ips em cada faixa de Subnet Private. Lembre-se que ele pega a faixa 02, e o octeto 0.0 variam de 0-255.

Para fazer o calculo do basta se basear pelo /24 que possui 256 Ips.

/23 = 512, /22 = 1048, /21 = 2092, 20 = 4088 Ips e assim sucessivamente.

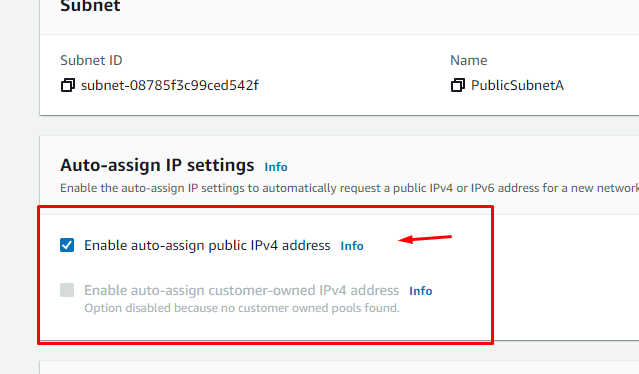
Se pegarmos um /16 teremos 65.536 Ips

Tendo nossas Subnets definidas e nossa Demo-VPC:

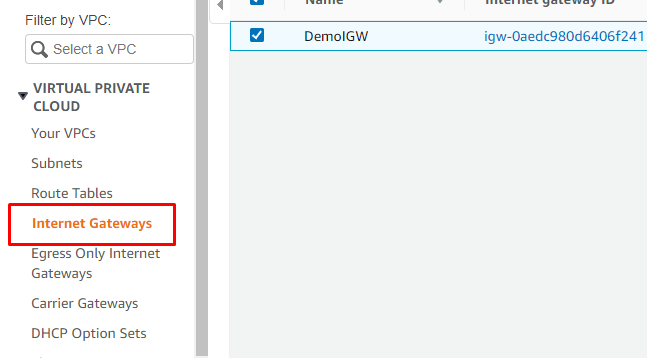
Criamos nossa Instancia EC2, e na parte de Settings vinculamos nossa VPC e a PrublicSubnetA:



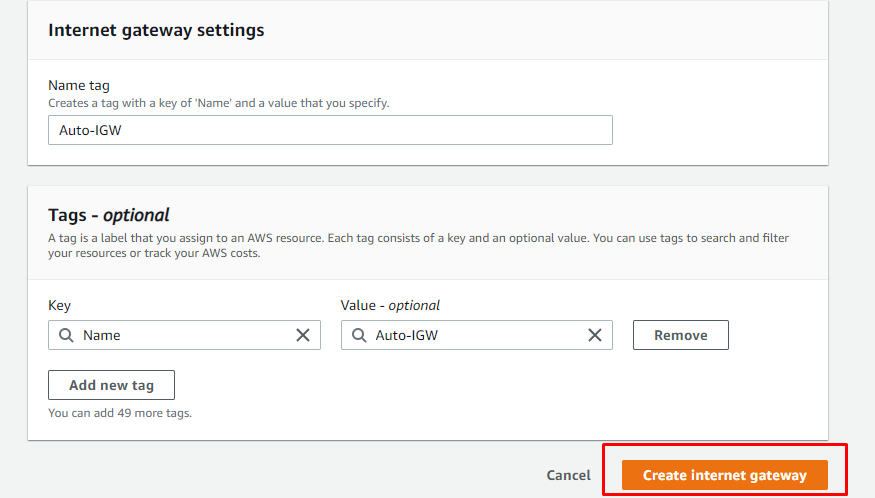
**Lembramos** que precisamos habilitar o Auto-Assigned Public IP da PrublicSubnetA e B



Ok isto parece bom, precisamos agora, fazer a nossa EC2 conversar com a internet, para que isso ocorra precisamos criar um Internet-Gateway:

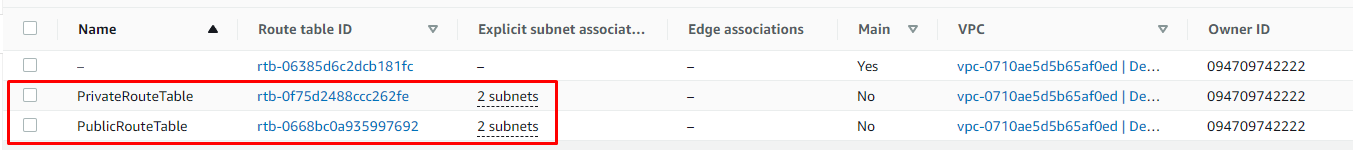


Criamos nosso Internet-Gateway:



Depois da criação do nosso Internet-gateway, precisamos criar as nossas Rotas, para separar as Subnets Publicas das Privadas:

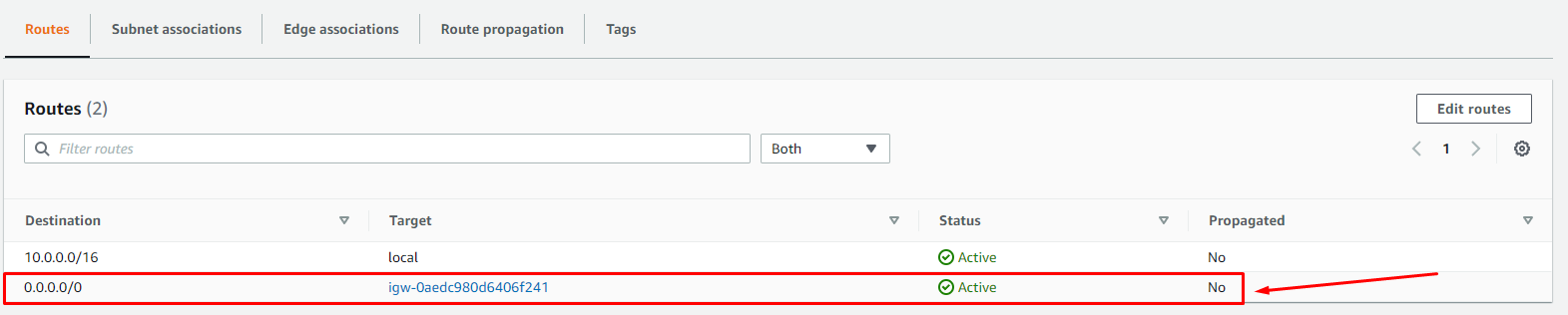
Então iremos criar 02 Routers Tables



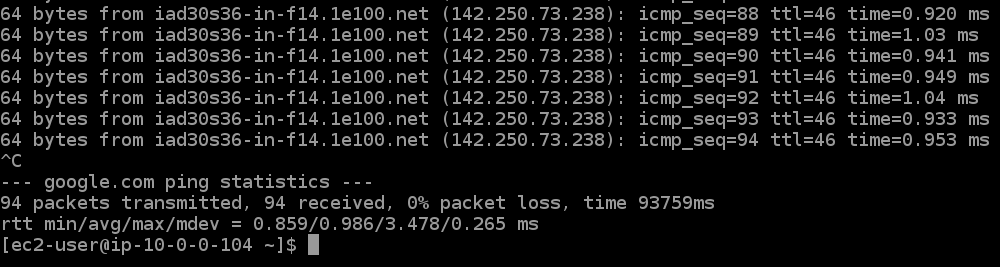
Com as Rotas Criadas, vamos Associar as Subnets Public as Publics e as Privates as Private dessa forma.



Com as 04 Subnets Associadas as Routes, Note, que agora podemos apontar nosso Internet-Gateway a internet, já que dentro da subnet temos o apontando o bloco CIDR, 1.0.0.0/16 Local.



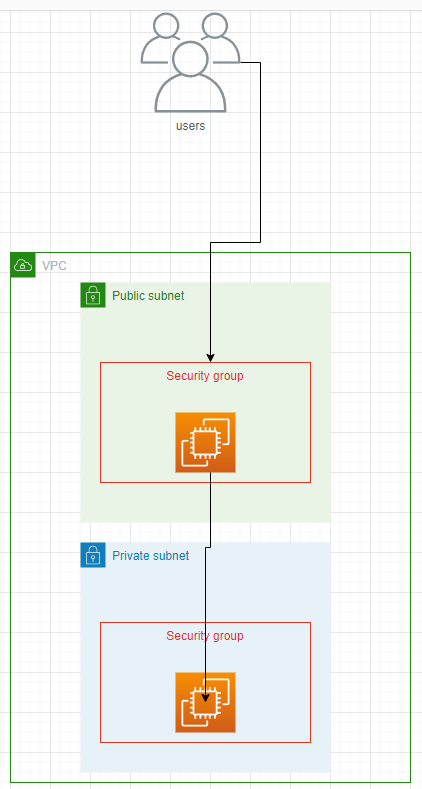
Então apontamos para 0.0.0.0 e o alvo será nosso Internet-Gateway que acabamos de criar.

Note que agora nossa instancia EC2 poderá acessar normalmente a Internet e Pingar um endereço do google.com

Get Starter with Bastions Host:

Portanto um Bastion Host e uma Instancia EC2 em uma sub-rede publica

E o que devemos fazer, e editar as regras do Security-Group da Instancia nas sub-redes Privadas, e o que temos que fazer e que os usuários façam SSH diretamente no Bastion Host EC2 da subnet publica, e a partir do Bastion Host, iremos fazer SSH na Instancia EC2 do segmento Privado:



A um conceito de Lúpulo, para saltar do host Bastião para as subnets privadas

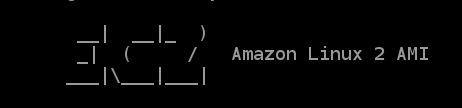
Vamos criar mais uma instancia EC2, e nesta instancia quando formos defini-la utilizaremos o Security-Group

Name: Private-SG

Description: Lauching-Wizard-Created-Security-Group

Port: SSH Custom [sg-0b6d25c9d81645c1e](https://us-east-1.console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroup:groupId=sg-0b6d25c9d81645c1e)

Depois review to Lanch, e crie a máquina.



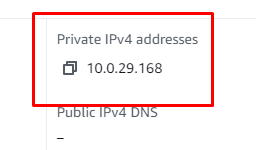
Precisamos adicionar a nossa DemoKeyPair.pem que criamos a nosso Bastion Host, para podermos ter acesso a Sub-rede Privada.

Abriremos com o Nano ou o vim.

Copiamos e colamos nossa DemoKeyPair.pem para o nano e depois salvamos.

Chmod 0400 DemoKeyPair.pem

ssh [ec2-user@10.0.29.168](mailto:ec2-user@10.0.29.168) -i DemoKeyPair.pem



E temos nossa conexão com a Ec2 Privada:



Se tentarmos pingar o google.com por exemplo, isto não será possível, pois a instancia privada não tem acesso a internet.

NAT = Networking Address Translation

Conversão de endereço de Rede.

Essas Instancias NAT, permitem que instancias EC2 em subnets privadas se conectem a Internet.

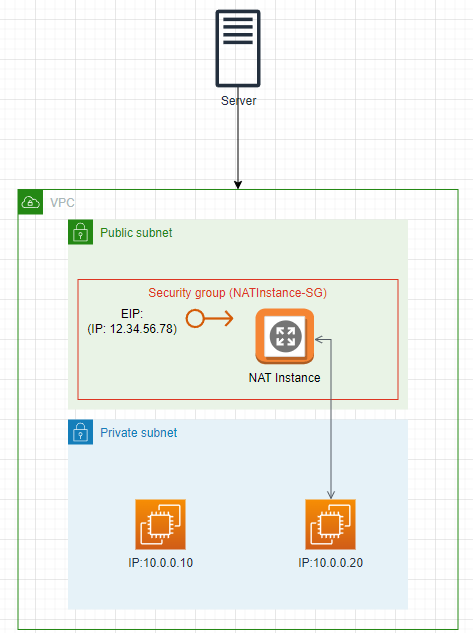
Para fazer isso, a Instancia NAT deve ser iniciada em uma Subnet Publica e, em seguida, conectada a Sub-rede Privadas

A instancia devera ter um Elastic IP fixo anexado a ela.

Então temos um servidor público com IP: 35.170.60.41 e queremos acessa-los de nossas sub-redes privadas ok.

Então como estabelecemos a conectividade das sub-redes privadas com o servidor?

Bom para fazer isso, vamos lançar uma Instancia NAT, com seu próprio grupo de segurança em uma subnet Public, então o que vamos fazer e anexar um Elastic IP a essa Instancia NAT, e então vamos editar a tabela de rota da subnets privadas, para a sub-rede publica para poder enviar o trafego das Instancias EC2 entre as duas subnets para a Instancia NAT.



Ele terá que passar pelo SG-Nat Instance, como você pode ver esse pacote e reescrito pela Instancia Nat, e isso que eles fazem, eles reescrevem os Pacotes de rede, e o IP foi alterado.

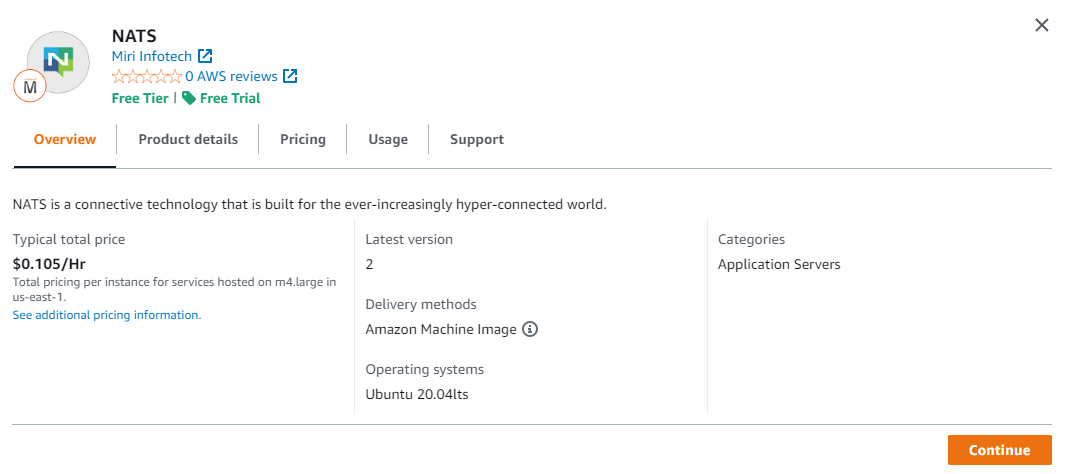
OBS: Por isso que a Verificação de origem devem ser desabilitadas na Instancia EC2

Então teremos uma tabela de rotas que permitem que as instancias privadas, falem com a Instancia NAT

Criando Uma NAT Instance:

Vamos criar um, NAT Appliance:

Com a Instancia NAT-Appliance:



Nome da Minha Instancia: Nat-Instancie.

Na Aba de Security-Group:

Security-Group name: nat-Instance-sg

Description: Launch-wizard-2 created 2021

SSH TCP 22 LOCAL

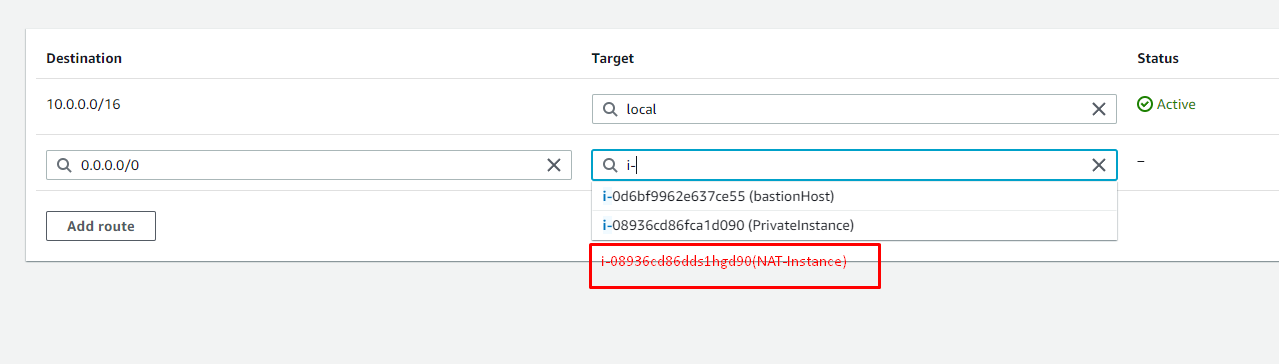
HTTP TCP 80 10.0.0.0/16

HTTPS TCP 443 10.0.0.0/16

Depois vamos em Networking, Change Source/Destination Check

Clique me Stop

Na PrivateRouteTable, na opção Edit Inbound, adicione uma rota especifica:



Depois, vai no Security-Group do NAT-Instance, e adicone a ADD Rule All ICMP – Ipv4 custom 10.0.0.0/16

Assim nos teremos acesso a internet na Instancia Privada.

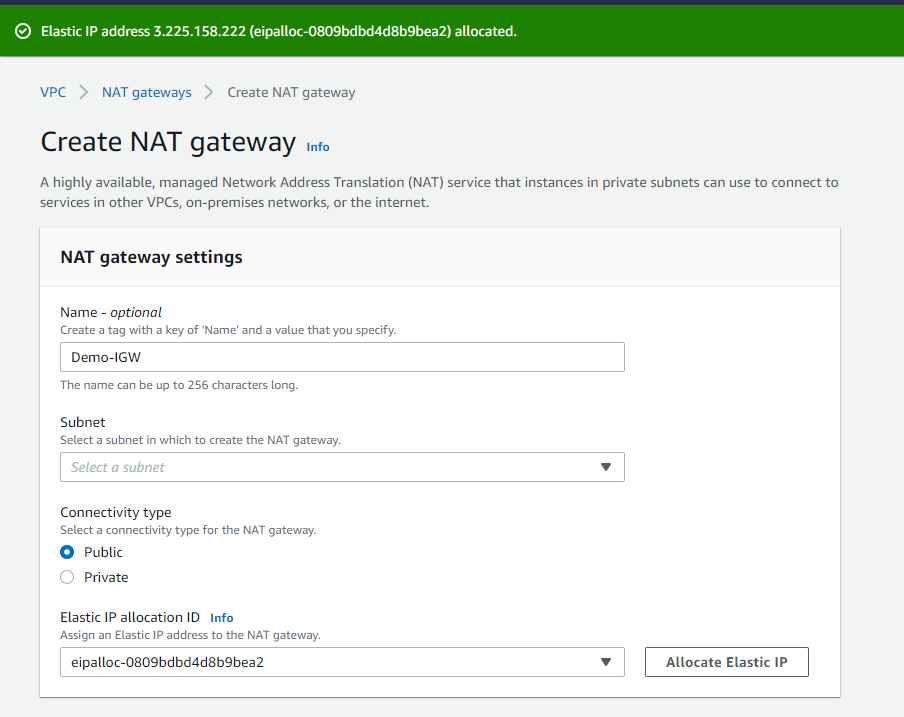
Utilizando um Ping. Google.com

Nota: A diferença entre uma Subnet Public que tem acesso a internet através de um Internet-Gateway. As Subnets Private, não possuem acesso a internet, por isso que deve se criar e configurar um NAT Gateway ou um NAT Instance, para que a Subnet Private, tenha acesso a internet.

Conceito de NAT Gateways

Gateway Networking Adress Translate

Criando um Nat gateway:



Editei as Rotas novamente removendo o NAT Instance, na PrivateRouteTable. E adiciona o NAT Gateway criado na Public na sua PrivateRouteTable para o Private ter acesso.

Isso parece muito bom, criamos um Nat Gateway dentro da AZ e apontamos com a PrivateRouteTable para a internet gateway para que a nossa subnet privada tenha acesso a internet.

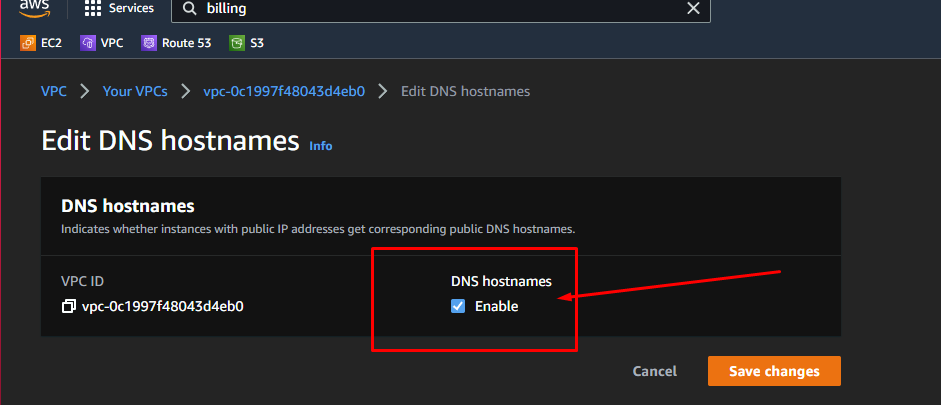
Porem precisamos criar um segundo NAT gateway em outras Avalability Zones, e em seguida editar as Rotas para torna-lo altamente disponível

DNS Resolutions In VPC

EnableDnsSupport:

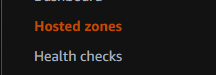
Então vamos falar sobre as duas configurações de DNS, que podemos encontrar no VPC, o primeiro e a solução DNS chamada ENA EnableDnsSupport, isto e pra saber se o resolvedor DNS

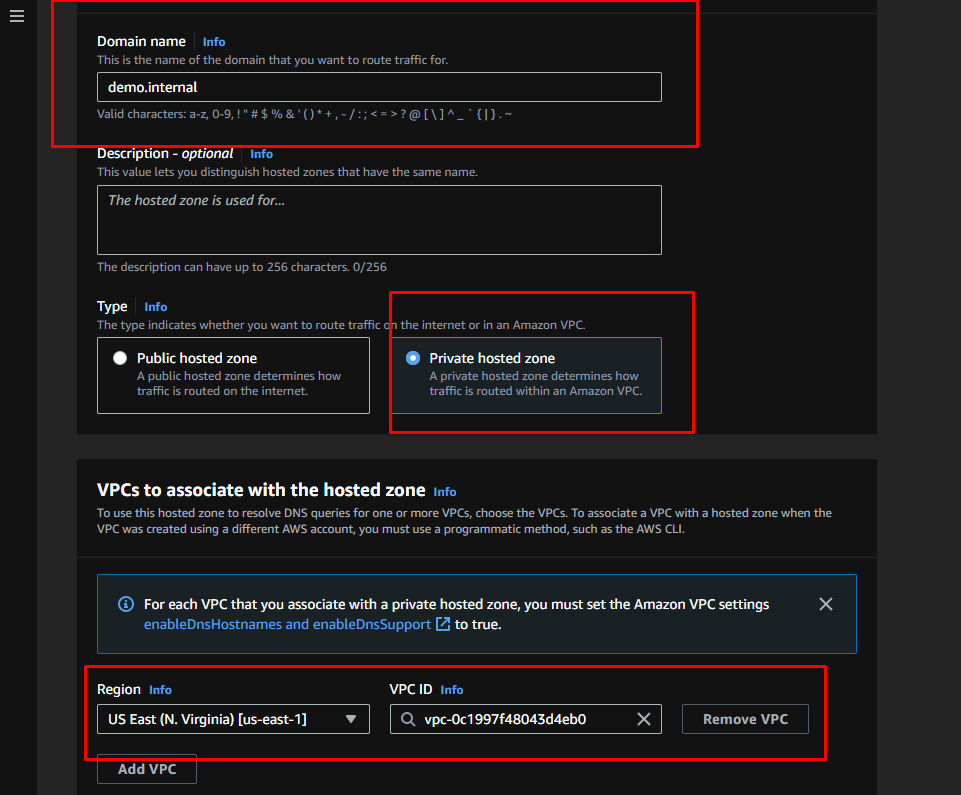
Para habilitar as Resoluções de DNS, vou entrar na minha VPC clicar com o botão direito e ir em Edit DNS hostnames.

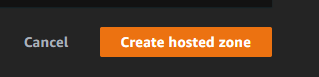


Habilitar DNS Hostnames.

Agora podemos ir para Route53:



**Habitar e Criar uma Host Zone com essas configurações:**



Precisaremos configurar o Records, e adicionar.

NACL CONFIGURAÇÃO:

-Grupos de segurança ACL e NACL de Redes:

-Exemplo de uma sub-rede, e saberemos quando você tem um instancia EC2 que anexamos um grupo de segurança (Segurity Group) a ela!

Mais existe um nível de proteção na subnet que ainda não vimos.

Que e a ACL ou a NACL de rede.

Vamos dar um exemplo para entendermos de vez o que é uma NACL de rede:

-Portanto uma solicitação vai para sua instancia EC2, agora está acontecendo apartir de uma perspectiva de rede?

1° Bem, primeiro a solicitação vai chegar ao NACL antes de ir para as subnets, então haverá algumas regras de entrada, no NACLs que serão definidas, e se a solicitação não for permitida, ela não será enviada, e se for permitido, então vai entrar, certo?

Então o NACL não tem estado, então veremos o que isso significa em um segundo portanto, a primeira solicitação passa pelo NACL e, em seguida, chega dentro das subnets e passa pelas regras de entrada do grupo de segurança, certo.

Então, sabemos como isso funciona, então portanto, novamente, se a solicitação não for permitida explicitamente, ela será negada.

Agora algo sobre o grupo de segurança: é que eles tem estado.

Portanto, lembre-se de que os NACLs não tem estado e os grupos de segurança tem estado!

Então o que isso significa?

* Isso significa que as solicitações passam pelas regras de entrada de segurança e chega a Instancia EC2, a instancia EC2 responderá com qualquer resposta que haja para a perspectiva do aplicativo.
* Então a saída será aceita automaticamente pelo nível do grupo de segurança.

Isso ocorre porque o grupo de segurança tem estado, isso significa que tudo o que é aceito também pode sair.

Portanto aqui, não a regras sendo avaliadas, e as regras de saída do grupo de segurança não são importantes neste exemplo.

Portanto, agora que sabemos que a saída do nível do grupo de segurança e sempre permitida porque tem estado, o que ainda acontece?

Bem, a NACL não tem estado, e sem estado e, portanto, como não tem estado, as regras de saída do NACLs serão avaliadas, E novamente, se eles não estiverem passando, o pedido não passara.

* Por isso, isto e uma solicitação de Entrada!

Agora vamos passar pelo mesmo cenário, mas para solicitação de saída:

Portanto, o grupo de segurança desta vez, então a instancia EC2 está fazendo uma solicitação de **saída**

Está fazendo um pedido para o exterior, a instancia tenta se conectar primeiro acessando o google.com, e, portanto, as primeira regra que são avaliadas são as regras de saída do Security Group.

Então, o trafego e permitido sair da instancia EC2 para a web?

* Então se as regras forem boas e a solicitação for permitida, a solicitação também passará pelas regras de saída NACL, portanto eles são avaliados.

Em seguida a solicitação chega em google.com, e ele volta para os serviços da Web da Amazon e, obviamente, o NACL não tem estado e, portanto, as regras de entrada do NACL são avaliadas.

Nota: A NACL padrão tem esta especificidade: aceita tudo, de entrada e saída com as subnets, as quais está associado: (Ele permite que tudo saia e entre)

Default NACL for VPC that supports IPV4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RULES#** | **TYPE** | **PROTOCOL** | **PORT RANGE** | **SOURCE** | **ALLOW** / **DENY** |
| **100** | **All ipv4 Traffic** | **all** | **all** | **0.0.0.0/0** | **ALLOW** |
| **\*** | **All ipv4 Traffic** | **all** | **all** | **0.0.0.0/0** | **DENY** |

Note for Exam: Mais se o Exame mencionar para você que existe um NACL padrão, saiba que, por padrão este NACL o desassociara das subnets, e se esta NACL estiver associada a subnets, desculpe, ela permitira que tudo entre e tudo saia

Ephemeral Ports:

* Portanto quando você tem um cliente e um servidor se conectando, eles devem usar a porta.

Portanto temos o endereço IP e as Portas, e os clientes se conectam ao servidor na porta definida!

Portanto sabemos por exemplo, que a porta HTTP 80, e a HTTPS 443 ok. E SSH e 22 e assim por diante.

O cliente se conecta a uma porta definida, exceto se os clientes precisarem de uma resposta do servidor

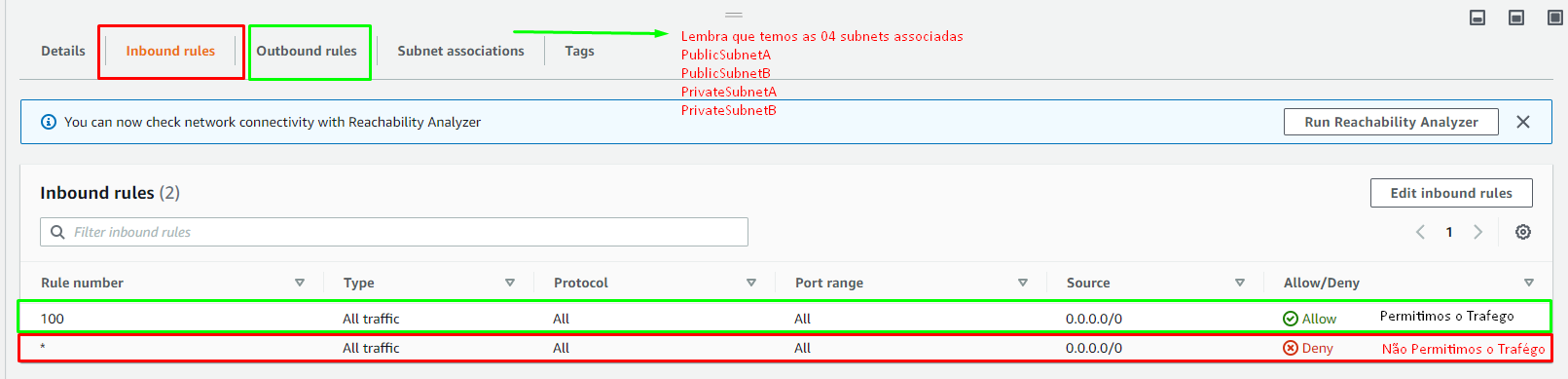
Portanto o servidor também precisa se conectar aos clientes para enviar uma resposta (response)

* E O CLIENTE NÃO TEM NENHUMA PORTA ABERTA POR PADRÃO!

Então o que ele faz quando o cliente está se conectando a um servidor, o cliente abre portas especificas nele mesmo e essa porta e efêmera, porta que durará tanto quanto a conexão entre o clientes e o servidor.

HANDS ON NACL:

Criando as NACL e por padrão uma NACL Default ela aceita todas as conexões (aceita tudo)



(100) –

( \* ) -

Nota: As NACLs por padrão sempre têm todo trafego de entrada e saída autorizado, e o padrão será associado a todas as Subnets você criar!

Seguindo:

Vou iniciar um servidor HTTP para me conectar a ele

1° Vamos nos conectar e irei instalar o HTTPD:

Sudo yum install -y httpd

2° Em seguida vamos iniciar o Web Server

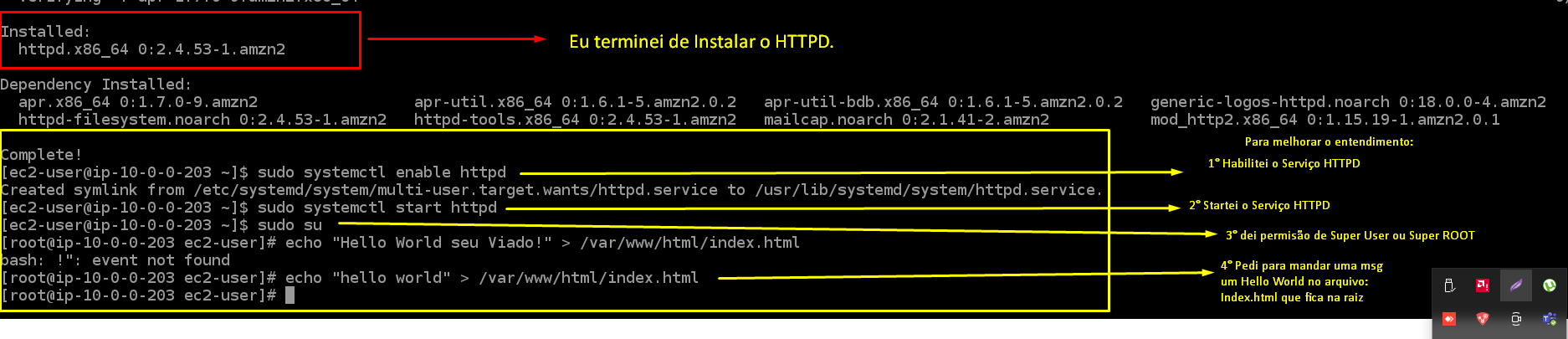
Sudo Systemctl enable httpd

Sudo Systemctl start httpd

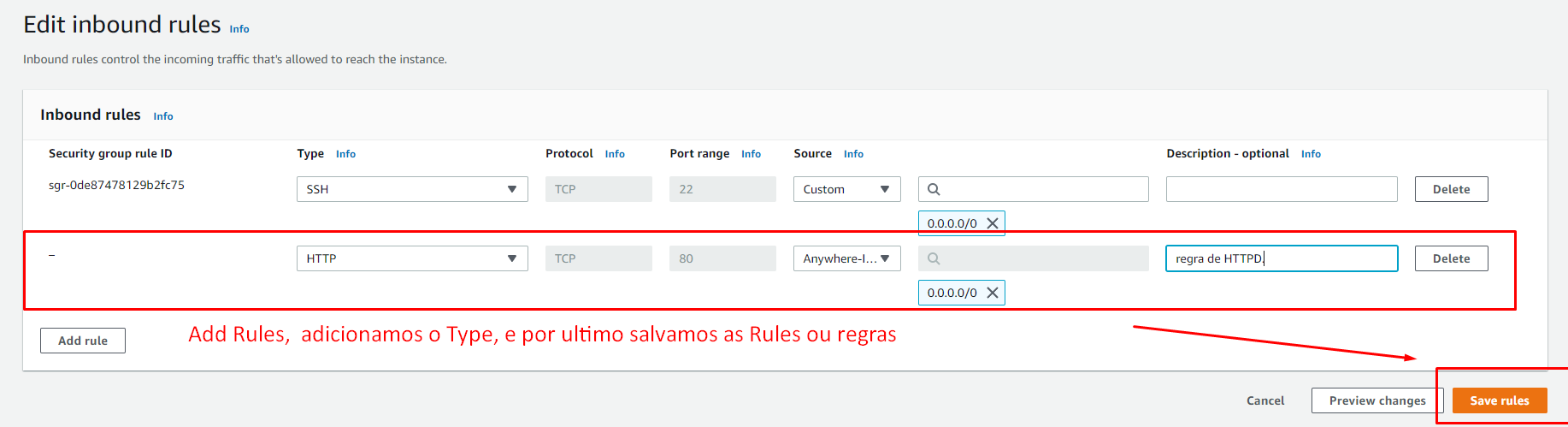
Sudo su

Echo “Hello World” > /var/www/html/index.html

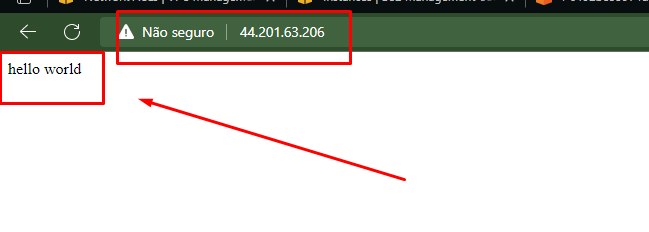
3° Agora outra coisa que precisamos e garantir que o host, tenha esta instancia publica com HTTP habilitado para o Security Group.



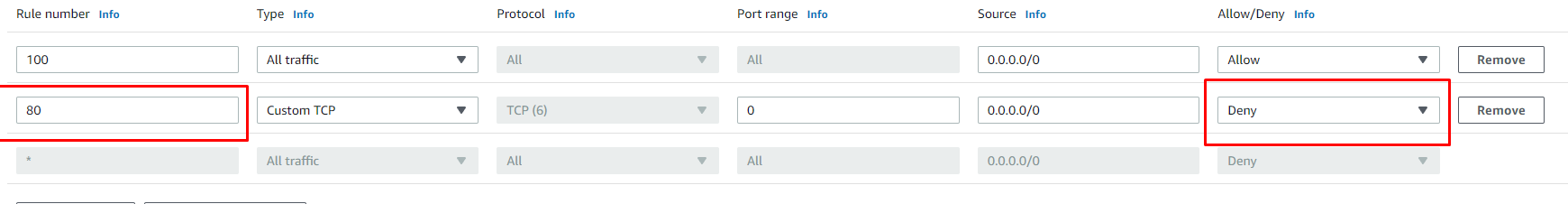
4° editamos o SG



Colamos a URL e teremos o resultado



Negando a conexão da URL



VPC - Reachability Analyzer

Analisador de acessibilidade da VPC

exemplo: Na parte de Pricing e cobrado 0.10 centavos de Dólar para criar um Reachability Analyzer

Peering – Connect to VPC

Então o que queremos e juntar nossas peças, duas peças para ser mais exato.

-Criando 02 VPC:

-Demo-VPC

-Default-VPC

E para isso vamos fazer primeiro, que as peças não estão conectadas:

Vamos criar uma instancia, com as configurações padrão, utilizando a Default-VPC, e depois lançaremos a Instancia.

Com a Instancia Criada chamada: DefaultVPCInstance, e o que eu quero fazer e me conectar e garantir que esse padrão em alguma VPC possa se conectar a instancia no meu BastionHost.

Se dermos uma olhada no:

-BastionHost: Private IPv4: 10.0.0.72

-DefaultVPCInstance: Private IPv4: 172.31.36.159

Diferentes eles são.

Então conectaremos as 02 Instancias, BastionHost e DefaultVPCInstance.

Para conectarmos, precisamos criar uma conexão aparente, acessamos Peering Connections.

Created Peering Connections:

Name: DefaultPeeringConnection.

Precisamos selecionar a VPC Local, ou selecionar a DemoVPC

Agora vamos selecionar outro para se conectar, Select Another VPC to Peering, setaremos a VPC DefaultVPC.

Created Peering Connections.

Endpoints – VPC

Vamos considerar AmazonDemoDB como um serviço, podemos utilizar o serviço público da AWS!

Mais esses serviços, como você sabe, eles residem nas nuvens da AWS, portanto deve haver uma maneira melhor de acessa-los, que você acessa de forma privada. E a resposta nossos pontos de Extremidade VPC (Endpoints)

Portanto os EndPoints VPC são pontos de extremidade privados dentro de sua VPC, que permitem que você inicie uma conexão privada com os serviços da AWS.

Aqui por Exemplo, aqui em minhas subnets privadas, minha EC2 pode se conectar ao EC2 endpoint VPC, CloudWatch ou Amazon S3, ou que você quiser usando um Endpoint VPC.

Mais tudo isso sem atravessar a internet pública.

Então, como isso funciona?

Bom se considerarmos também o Endpoint VPC, eles são chamados de link privado, eles são alimentados por links privados, vamos considerar o SNS novamente.

Portanto temos duas maneiras de acessar o Amazon SNS apenas para ter certeza de que realmente você entende.

1° O primeiro é passar por um NAT gateway, então de uma subnet privada a um Nat gateway, a um Nat Gateway de internet, ao Amazon SNS, o que custa muito dinheiro, porque o Nat Gateway tem um custo por hora e então você precisa configura-lo e tudo passa pela internet pública.

Portanto definitivamente não há o melhor desempenho de rede ou performance.

2° O segundo e criar um Endpoint VPC e, em seguida conectar sua instancia EC2, com os Endpoints VPC, até o Amazon SNS e todo o trafego permanece privado na rede da AWS.

Portanto, todos os serviços na AWS têm uma URL publica certo?

Mais os pontos de extremidades da AWS, permitiram que você se conecte a esses serviços usando uma rede privada em vez de usar uma internet pública.

Eles são redundantes e escalam horizontalmente, portanto, não há preocupações de escalonamento, e eles eliminam toda a necessidade de ter um gateway de internet ou Nat gateway para acessar os serviços da AWS, o que pode representar uma boa economia de custos.

Em caso de problemas com os pontos de extremidade (endpoints) VPC, pode precisar certificar de que a resolução de configuração de DNS em sua VPC está ativada e também verificar as tabelas de rota, em seu VPC porque você deve modifica-los.

Portanto existem 02 tipos de Endpoints na AWS:

-interface Endpoints (Nós de Extremidade)

Portanto, os pontos de extremidade da interface (interface Endpoints) te forneceram um ENI, um ENI e uma interface de rede elástica, daí o nome pontos de extremidade interface. (E um ENI e um endereço IP privado)

Então esse endereço de IP será um ponto de entrada em um serviço AWS, ENI você pode Anexar um Security Group e o ENI support a maioria dos serviços da AWS

Gateway Endpoints (Nós de extremidade do gateway)

Agora, terminais de gateway, provisione um gateway, e esses gateways são um poucos mais complicados, porque devem ser usados como um destino em uma tabela de rota.

Você precisa se lembrar, eles suportam Amazon S3 e DynamoDB, portanto os endpoints do gateway, os endpoints precisam ser roteados das subnets privadas, e em seguida para o Amazon S3 ou o DynamoDB

Introdução ao CloudFront CDN

**CloudFront CDN (Content Delivery Networking)**

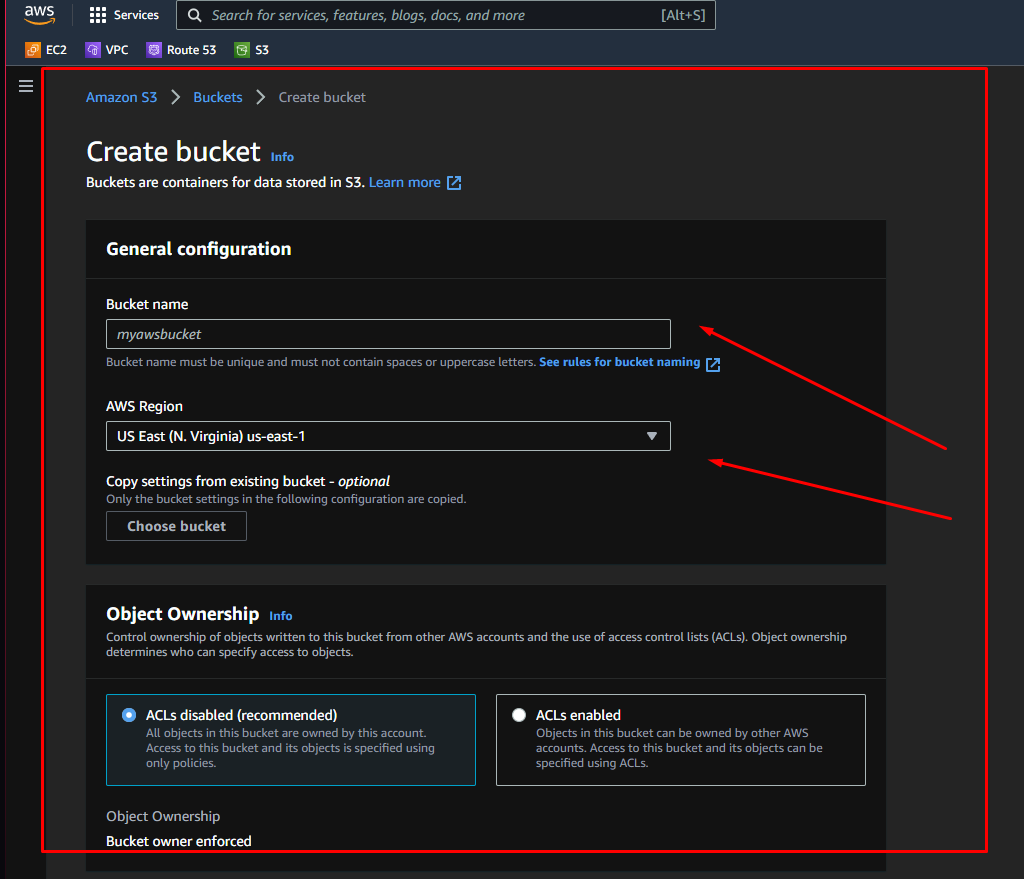
**Se tiver um vídeo hospedado na Avalability zone, São Paulo Brasil, e usuários querem acessar o conteúdo em diversos outros países, isto pode gerar um delay.**

**Melhor situação e fazer o upload do seu arquivo, na sua Avalability Zone, e ele ser replicado para as outra Avalability Zones, e também para as Edge Location, cidades menores, distantes das Avalability Zones, para que possam acessar também.**

**-a partir do acesso, e feito um cache, e esse cache tem uma validade, que nós chamamos de Lifetime.**

Hands On CDN CloudFront:

CDN – Criar uma Bucket:



Vamos adicionar um arquivo de Vídeo.

Adicionar o serviço de CloudFront para replicar esses dados para todas as Edge Location.

Esse serviço vai estar disponível nas edge location que nos selecionarmos na CloudFront, fazendo com que usuários em outras localidades possam assistir o vídeo com um delay bem mais baixo.

Em Networking Cloud Front:

Clicamos em Create Distribuition.

Existem 02 tipos RTMP e Web

O Web: quando você vai hospedar sites, tanto static, ou dynamic com uma velocidade maior para os usuários!

O RTMP: e quando você vai armazenar arquivos de Mídia dentro de uma Bucket.

RTMP Settings:

1 - **Origin Domain Name:** (Aonde esta, qual bucket vc quer que faça parte dessa RTMP)

2-**Restrict Bucket Acess:** quando você quer restringir URLs do CloudFront, tem uma URL padrão.

Se habilitar sim, ele vai pedir para criar uma nova restrição para essa bucket, e apontar para criar uma nova identidade para restrição dessa Bucket.

3 - **Price Class:** Onde quero distribuir os Videos:

- **Use all Edge Locations (Best Performance)**

- **Use only North America and Europe**

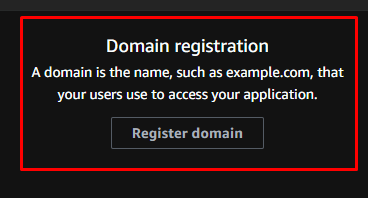
- **Use North America, Europe, Asia, Middle West, and África**

4 - **Alternate Domain Names** (Cnames):

Posso alterar o nome da URL para torna-la um pouco mais amigável

Como por exemplo: **cdn.images.andrepanizza.com**

Para que eu possa fazer isso eu preciso Logar no meu Register, onde que eu registro o meu domínio:

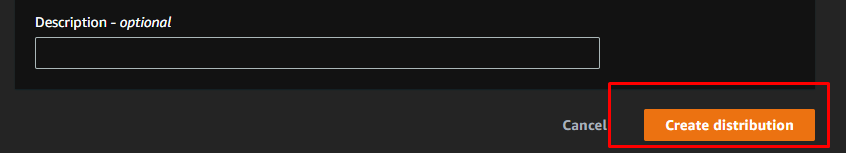
 Na Router 53

E criar um Cname e apontar pra Amazon.

5 – **Logging:** Significa que ele vai armazenar todas as requisições que os usuarios fizerem no vídeo.

Bucket for logs: andrepanizza.MP4 nome da Bucket

**6 - Distribuition State:** Irei associar o conteúdo com a distribuição, toda vez que ele for requisitado ele vai criar um registro na parte de requisição.



Hands On Configurando Web CDN

Conceitos de Web CDN -

CONCEITO DE SERVELESS

Conceito básico de Serveless:

**Leitura Complementar sobre Serveless:**

**O que Significa sem Servidor?**

Sem servidor e a arquitetura nativa da nuvem que permite transferir mais das suas responsabilidades operacionais á AWS, aumentando a agilidade e a inovação. A arquitetura sem servidor permite criar e executar aplicativos e serviços sem preocupação com os servidores

KMS – ENCRIPTYON

Introdução ao KMS (key management servisse) Encription:

Criar Keys, criar usuários, que podem utilizar essas Keys, fazer o sistema de Encription e Description em qualquer tipo de arquivo, ou alguns serviços da AWS.

1. Se você tiver o usuário 1 ele apenas poderá gerenciar (manager Keys)
2. Criar o usuário 2 ele consegue fazer a encriptação e a decriptação

A partir dai o que os usuários conseguem fazer, eles conseguem pegar dados de as vezes um arquivo de texto, ou uma bucket do S3.

Ele consegue pegar esses dados, encriptar com o algoritmo da AWS e mostrar um arquivo e um conteúdo totalmente Encriptado, seguro (o mesmo usuário consegue também decriptar)

Por exemplo:

Você tem Volumes EBS – Você consegue encriptar com o Key Management Service, você tem sua bucket com S3, você também consegue encriptar, RDS você consegue encriptar com o KMS.

Qualquer tipo de dado em quase todos os serviços da Amazon AWS que podem receber a Encriptação pra garantir que esses dados estão seguros na cloud e em transição entre o usuário final e a cloud, você pode utilizar o Key Management Service (KMS)

Hands On KMS Key Management Service.

Vamos criar um cenário onde vamos encriptar e decriptar usuários dentro do KMS, o laboratório vai ser dividido em duas partes.

- Onde na primeira parte, vamos criar 02 usuários – usuário 1, e o usuário 2 dentro do IAM

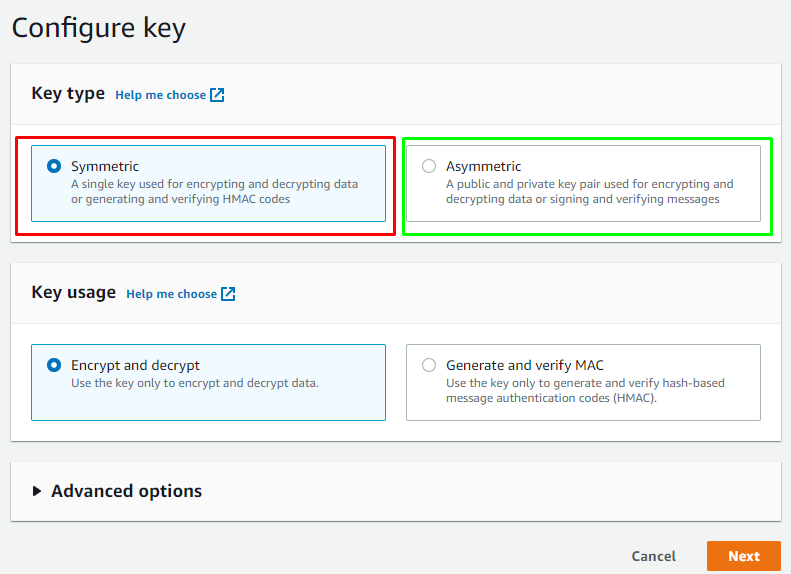
-Vamos jogar esses dois usuários dentro de um Grupo e esse grupo vai se chamar: Encription Group (Os dois Usuários terão acesso de ADM) Somente Programático.

Depois vamos dar privilégios a esses usuários:

-Dentro do KMS, eu consigo falar com que 1 usuário (Manager create and delete Keys) (ADM) e o outro Encripta e decripta (usuário 2 encrypter and decripter)



Agora com os usuários com permissão administrator e o Group KMS Encryption criados como a Policy Attachment: FullAcessAdministrator, eu preciso dar acesso ao user 1 (de Manager) e o user 2 (de Encrypted/Decrepted)

No serviço de KMS: 

-**Customer managed keys** – e criar uma Key.

**Nesta tela aparecerão 2 opções:**

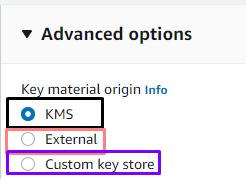
Você pode criar um Key Simetrica e Keys Asimetricas.

A Key Simetrica (Symmetric) e a key que você utiliza 01 key tanto para encriptar, quanto para decriptar contéudo.

A Key Asimetrica (ASymmetric) você usa uma Key publica e uma Privada, uma para **encriptar/decriptar** e outra para verificar ou assinar.

Vamos continuar com a criação da chave Symmetric por questões de Simplicidade.

Em Advanced Options:

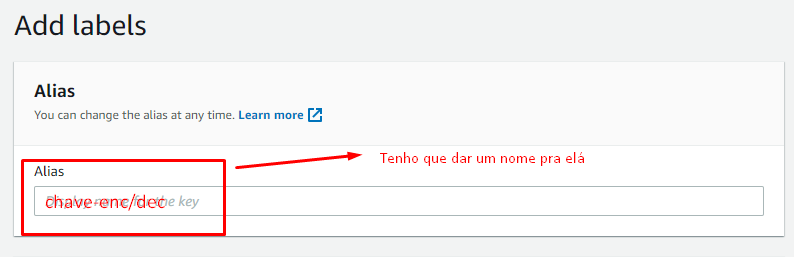


KMS – Você vai criar uma Key utilizando o key management service.

External - Você pode conectar a uma outra conta

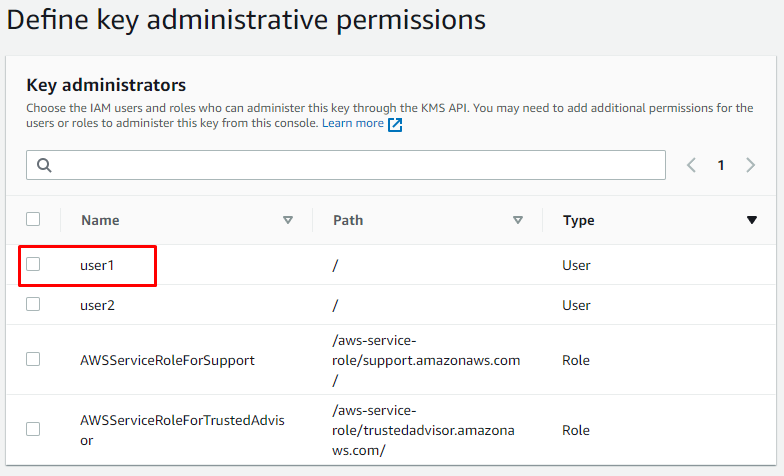
Custom Key Store (Cloud HSM) – E um sistema privado que você consegue exportar as suas Keys





* Darei o nome da minha chave, como Encripted e decripted
* Posso criar uma tag, ou adicionar uma descrição e next.





Agora ele está pedindo para escolher um usuário com regras de administrador ou seja:

Quem vai conseguir criar, remover, invalidar chaves, habilitar ou desabilitar chaves.

* User 1



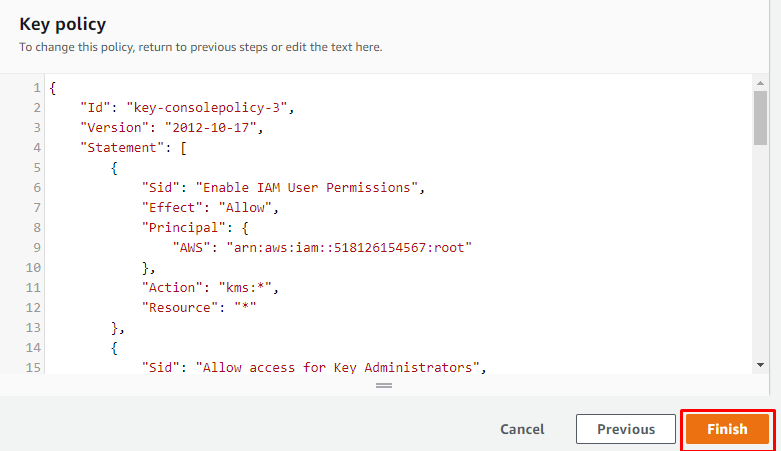
A próxima pergunta e: Selecione o usuário IAM que pode Encriptar, que nesse laboratório e o User 2

* User 2



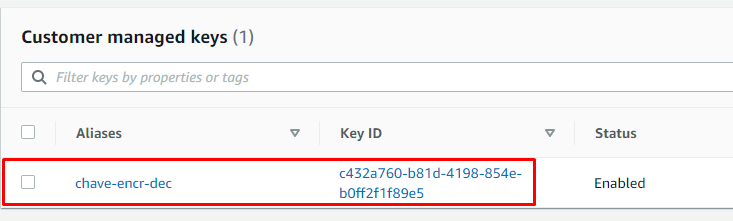
NOTA: Nunca mantenha por questões de segurança o mesmo usuário conseguindo fazer as duas tarefas.

Vai te mostrar o tipo de Encriptação:

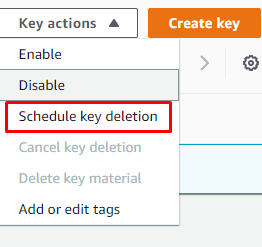


Finished.

Temos aqui:



Podemos agendar o delete, que e no mínimo **7 dias**



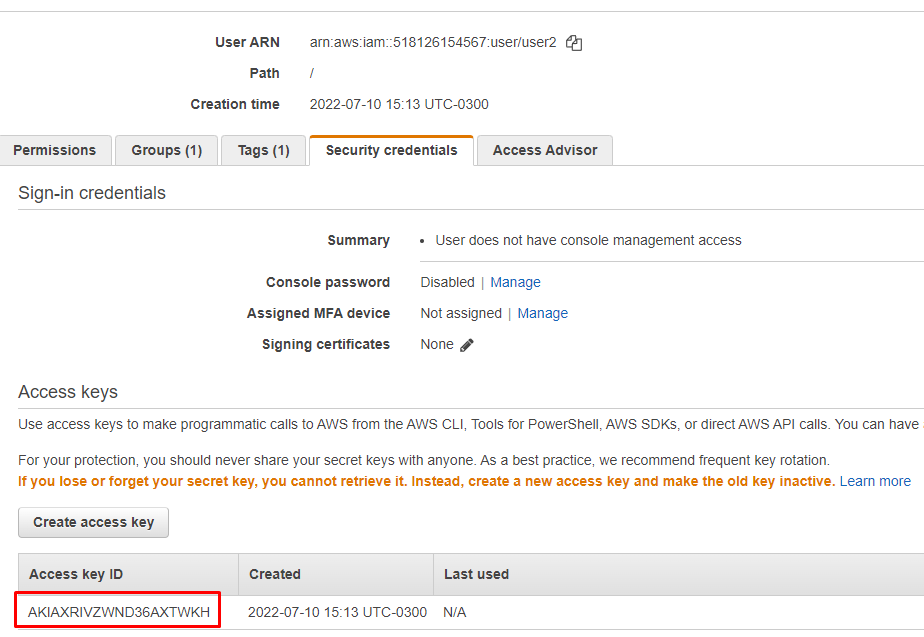
Hands On KMS – Parte 2

-Agora vamos criar uma máquina virtual instancia EC2 para verificarmos se o KMS funciona, como primeiro passo criaremos a máquina:

Depois que criamos a máquina, entramos como super user (Sudo su)

E depois usamos o comando aws configure:

Nota: Para conseguir a acess Key: Entre no IAM – User 2 Security Credentials



Precisaremos da Acess Key e da Security Key

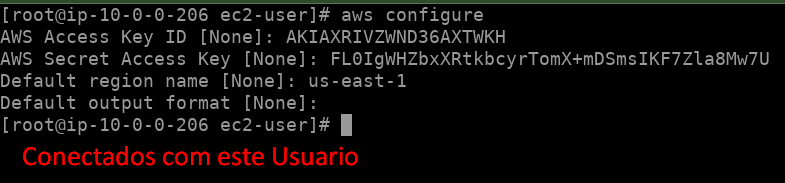
E a secret key do User 2 –



Location Region: us-east-1

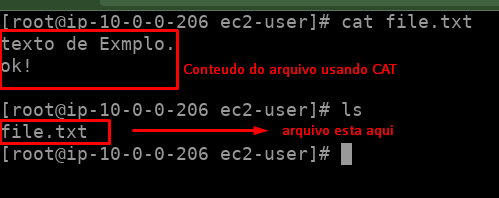
Default output format: none (nada)

Estamos conectados:



Primeiramente vamos criar um arquivo e colocar um texto dentro.

Nome desse arquivo seria file.txt



Com os arquivos criados, iremos fazer uma serie de comandos que estão nos recursos da Aula:

1 - aws kms encrypt --key-id XXXXXXXXXXX --plaintext fileb://file.txt --output text --query CiphertextBlob | base64 --decode > encryptedfile.txt

2 - aws kms decrypt --ciphertext-blob fileb://encryptedfile.txt --output text --query Plaintext | base64 --decode > decryptedfile.txt

3 - aws kms re-encrypt --destination-key-id XXXXXXXXXXXX --ciphertext-blob fileb://encryptedfile.txt | base64 > newencryptedfile.txt

**Onde no lugar desses XXXXXXXX iremos colocar as informações:**

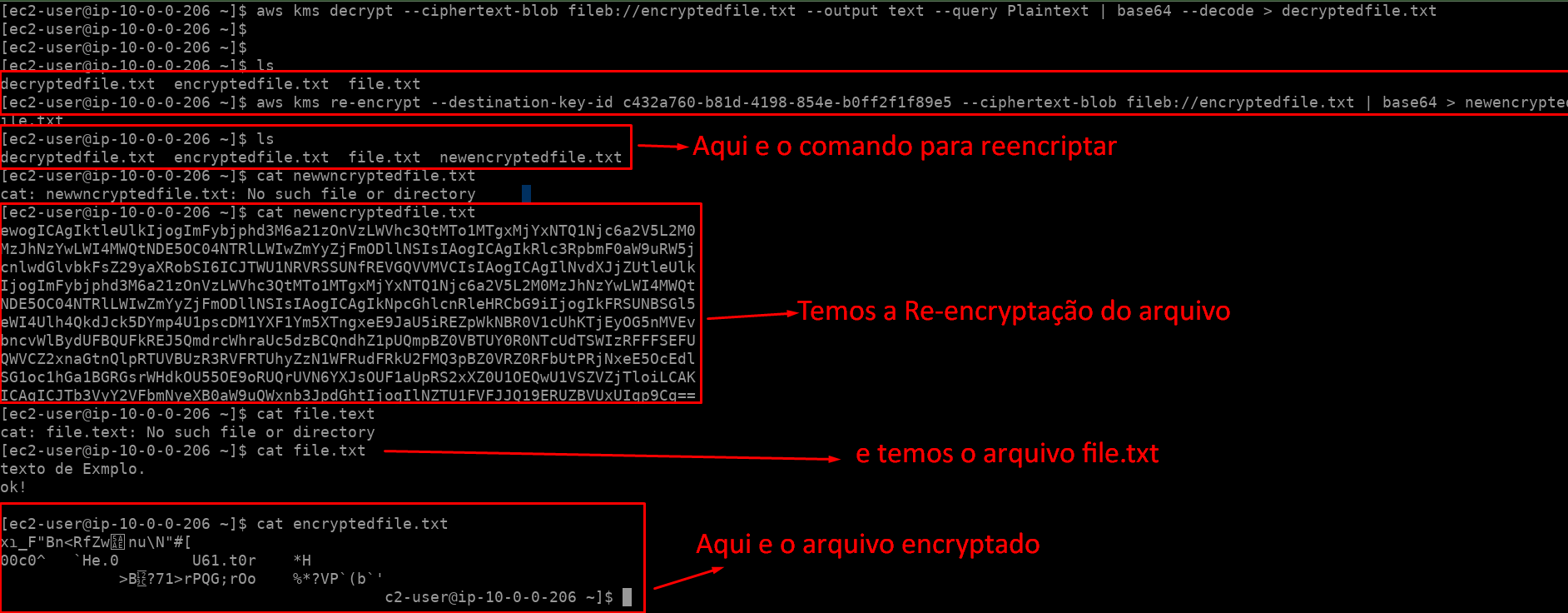
1. Primeiro ele está pedindo a Key ID



aws kms encrypt --key-id c432a760-b81d-4198-854e-b0ff2f1f89e5--plaintext fileb://file.txt --output text --query CiphertextBlob | base64 --decode > encryptedfile.txt

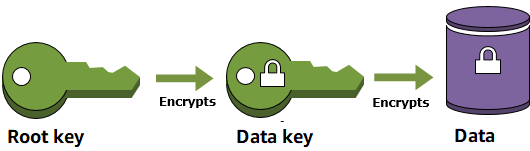
1. Vou colar na última:

aws kms re-encrypt --destination-key-id c432a760-b81d-4198-854e-b0ff2f1f89e5--ciphertext-blob fileb://encryptedfile.txt | base64 > newencryptedfile.txt



KMS Envelope:

O que fizemos neste laboratório:



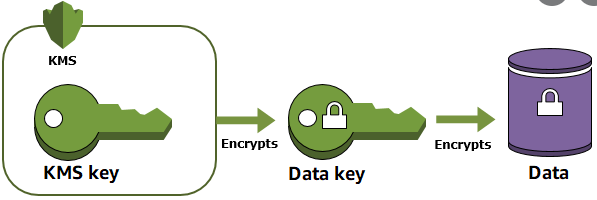
Foi pegar nossa Master Key, Encrypt os dados e transformar em algo extremamente seguro.

**A Master key e utilizada para Encrypted e Description**

**CMK – Costumer Master Key – ela utiliza campos como:**

* **Metadata**
* **Key ID**
* **Creation Date**
* **Key State**

**Dentro do Customer Master Key.**



Proteger esta master Key, literalmente encrypta a Master Key, com um algoritmo de encriptação.

Então o sistema que você utiliza para encrypter os dados, só que estamos utilizando para encrypter toda a master Key, o que chamamos de AWS KMS Envelope

SERVICES – SQS:

Simple Queue Service

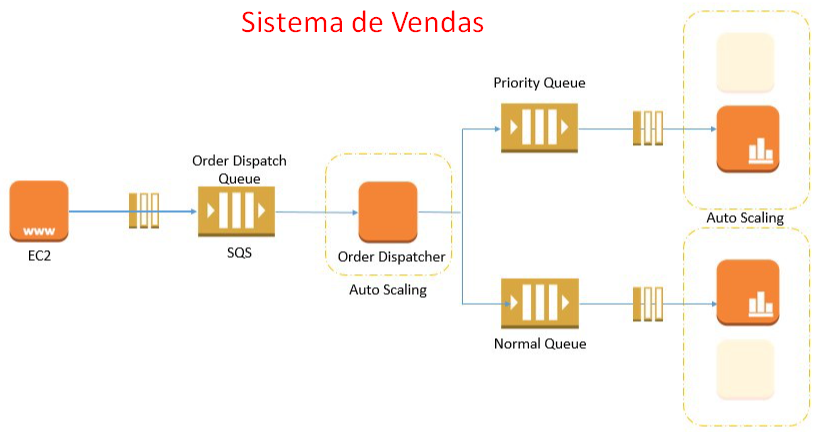
Simples fila Serviço -> fila.

* Armazena mensagens para que dispositivos, computadores, ou processos ou aplicações, possam ler estas mensagens e executar estas mensagens.

Onde as mensagens são enfileiradas, os processos são enfileirados (o que fazer e enfileirado)

E um sistema de armazenamento de filas de processos que devem ser executados.

Apartir do momento que o usuário faz uma compra em um web server, e a aplicação do web server armazena essa informação em uma Queue (No SQS).



SQS – TYPES:

O SQS tem 02 tipos de montagens de filas:

1° Tipo de Montagem de fila que o Queue utiliza e: STANDARD:

Standard – Queue usa HI-Throughput - Unlimited Transaction Peer Seconds

* Ele tenta entregar na ordem no que foi chegado, ele garante que no mínimo uma mensagem ou uma instrução vai ser entregue, no mínimo 1 vez.

Ele tenta manter a Ordem com que a mensagem chegou, não e garantido,

2° Tipo de Montagem de fila que e o: FIFO.

FIFO – Queue (First-in-First-out)

* Ela garante, One Time Processing – A mensagem vai ser processada apenas uma vez, aqui não existe duplicidade, ele limita a 300 Transactions Peer Seconds (TPS) Quase Unlimited 300 Transactions peer Seconds

SQS – TIPS:

SQS – SNS – SIMPLE NOTIFICATION SERVICES

Simple Notification Service:

Este sistema e um sistema de notificação, de serviço da aws:

Push – Mobile, SMS, Text message, Email, HTTPs – Site que contêm um sistema de logs.

Você paga por Pay-you-Go –

Você não precisa criar um alerta, pra quem vai ser enviado o alerta, vc cria um tópico e quem vai receber os alarmes daquele tópico.

CLOUD FORMATION:

## - Cloud Provider: foco da AWS são as empresas.

## - Ela criou uma grande quantidade de serviços:

Se eu quiser montar um servidor Virtual, eu preciso montar uma máquina EC2.

Se eu quiser armazenar arquivos, eu posso utilizar um serviço chamado S3 Buckets.

Se eu quero trabalhar com domínio (criar meu domínio e associar meu domínio a um servidor Web Route53)

O gerenciamento dessa estrutura estava bem caótico e bem complexo, então a AWS desenvolve um serviço chamado CloudFormation.

O CloudFormation: Nada mais e a forma que você tem de criar um código.

Template: Essa template, subimos a template na plataforma de CloudFormation, a template faz toda a instrução que estão lá dentro via código.

Preciso criar a seguinte estrutura:

2 X -EC2

2 X -Elastic IP – Pro endereçamento IP não ficar mudando quando a Máquina for reiniciada

2 X Buckets S3

1 X Load Balancer

Através de Código e Linha de comando utilizando YAML, e JSON da forma Declarativa:

Como se fosse de maneira declarativa, como se fosse uma lista de compras.

Benefícios de Utilização de CloudFormation:

Toda parte de infra, e baseada em códigos, benefícios, controle e gerenciamento. (automática)

Você pode salvar versões do seu código: consegue gerenciar esse tipo de versões;

Custo X Infra = Todas as suas implementações utilizando CloudFront – recebe uma TAG (referente ao tipo de serviço que você está subindo)

Toda vez que você cria um script via CloudFormation, ele te gera o Custo Estimado.

Você pode adicionar uma redução de custo dentro sua topologia, com o sistema de adição e redução de serviços.

Exemplo:

5 Servidores = 5 X EC2 – Essas maquinas funcionam enquanto esses funcionários estão dentro da empresa.

Das 07:00hs até as 18:00hs, O período das 20:00hs até as 06:00hs do outro dia, ninguém ta usando esses servidores.

Podemos então criar um script pra destruir todos os servidores: 20:00hs da noite

Podemos criar outro Script pra Criar todos os Servidores as 08:00hs da manhã desse dia

Então você vai passar a utilizar na madrugada 0% de recursos da AWS e gastando nada do seu Orçamento.

Qual Editor devo utilizar pra fazer toda parte de Codificação para CloudFormation:

* YAML
* JSON

Editor: Swagger

Aviso sobre cobrança CloudFormation

- CloudFormation e Gratuito, porém os Serviços que você vai subir vc acaba pagando por eles.

- Se você subir algum Script que esteja fora do free Tier elegible, lembre-se de Deletar esse serviço depois que finalizamos

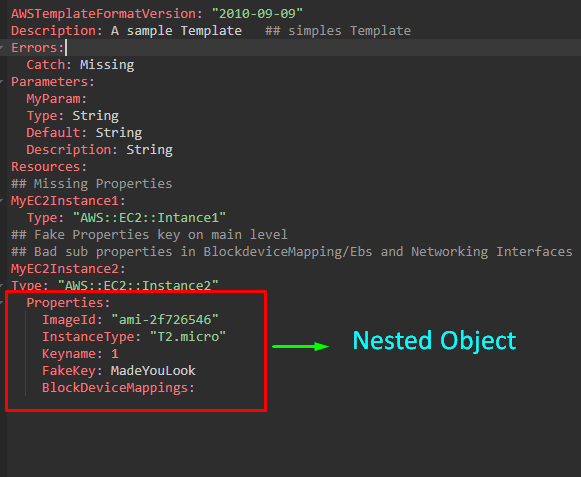
Deletamos o Stack Inteiro, que ele vai acabar todo o restante!

A Linguagem YAML

Simplicidade da linguagem de programação YAML:

1 – Definimos o Objeto que vai ser utilizado, declarando o que quero dentro da Linguagem de Programação, existem pontos aqui chamados de Key e Value.

Quando declaramos alguma coisa por exemplo:



Temos um Ninho de Objetos em Properties, como se fosse arranjos:

-As propriedades da Instancia: Instance2

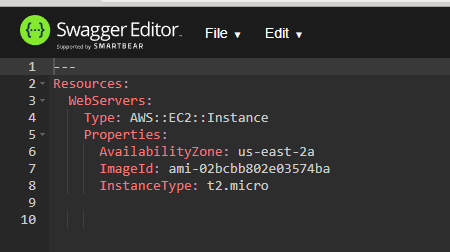
Existe um outro ponto que nos chamamos de Arrays (múltiplos elementos)



Hands On CloudFormation

Nos recursos da Aula iremos fazer o Download dos arquivos: server-Ec2

Entendendo a estrutura:



Resources = Recursos:

webServers = ou Grupo (Posso dar o nome que eu quiser para esse grupo)

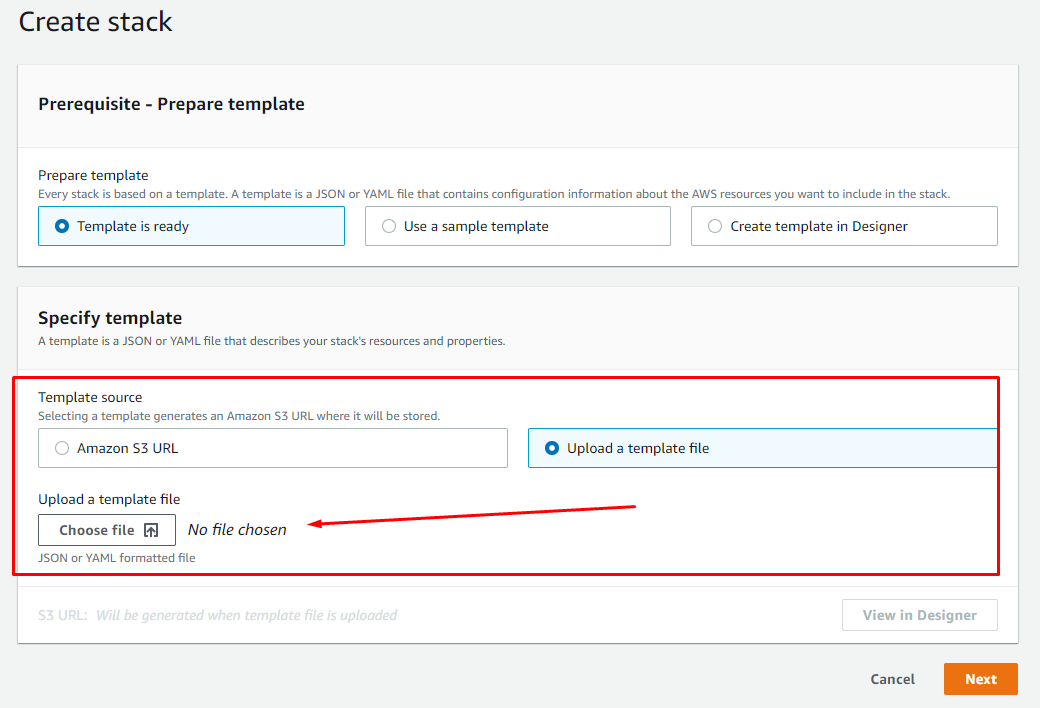
Type = que será AWS maquinas EC2 Instancias (deve ser sempre separado por duplo 2 pontos

Propriedades = Quais são as propriedades dessa maquina, ela vai estar na Availability zone us-east-2ª

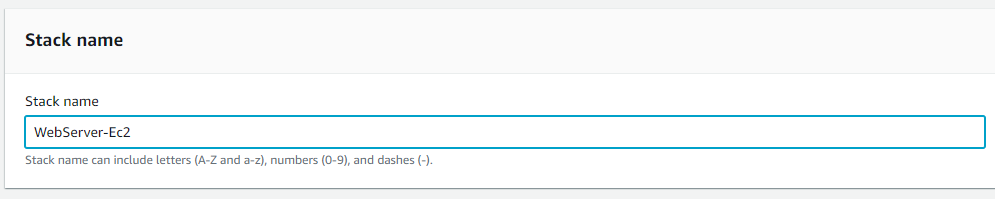
ImageId = Imagem da Máquina, disponível na seleção do tipo (Linux, Ubuntu, Windows, Mac)

InstanceType = Tipo de Instancia (T2. Micro, T3. xlarge, T4. xlarge) qual tipo gostaria de utilizar.

Depois que terminarmos de digitar todos os comandos, iremos salvar o arquivo como YAML.



Depois de Criar o Stack, Stack-Name, e depois created Stack.



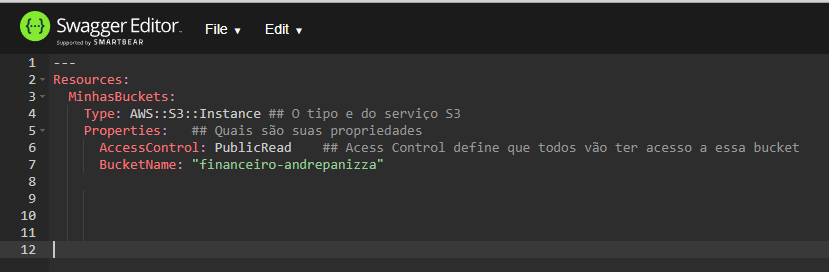
Depois você terá uma serie de configurações que poderá ser feita aqui:



Created Stack.

E sua máquina será provisionada dentro da CloudFormation.

Provisionando S3 bucket



Se caso alguma palavra contiver letras maiúsculas, ele irá fazer o Rollback

Introdução ao Kinesis:

O serviço da AWS que gerencia conteúdo, gerado como websites, como aplicativos, plataformas

Como ele funciona:

-Kinesis trabalha 100% Versão Streaming Data (vídeo) que está relacionado.

Exemplo bem pratico:

Você tem uma Origem, ou seja, um Source, essa Origem envia para vários Locais, esses locais conseguem passar para esses locais um certo tipo de Trafego.

Que vai ser replicado por outros Nodes (Nós)

Outro Exemplo:

Você tem um Website:

-Esse website faz vendas Online:

Os usuários que tem acesso a sua URL, eles vão acessar seu Websites:

1° Eles podem fazer o seu cadastro. (nome, Email, Senha) Vamos armazenar essas informações no DynamoDB

O website, está armazenado num Servidor EC2.

2° Eles podem listar seus produtos

Quando ele Lista os Produtos – Pegar essa lista de produtos, e entender através da navegação do Site, pelo cliente 1 ele ta navegando e lista produtos de uma categoria X por exemplo:

Criar uma lista para que os clientes que mais visualizam esta categoria, podemos fazer uma venda bem especifica.

3° Eles podem também comprar produtos:

Vai gerar um perdido, essa venda vai ter um número, endereço pra entrega, produto, e o valor do produto.

O que este website está gerando aqui e a Streaming de dados.

E esse tipo de Streaming de Data e onde o Kinesis consegue te ajudar, porque ele faz o seguinte:

Ele armazena as informações, e os outros aplicativos e outras maquinas EC2, podem capturar esse tipo de informação, e podem utilizar ou processar.

Utilização:

* Você pode utilizar ele pra jogos, dispositivo IOT, Uber utiliza

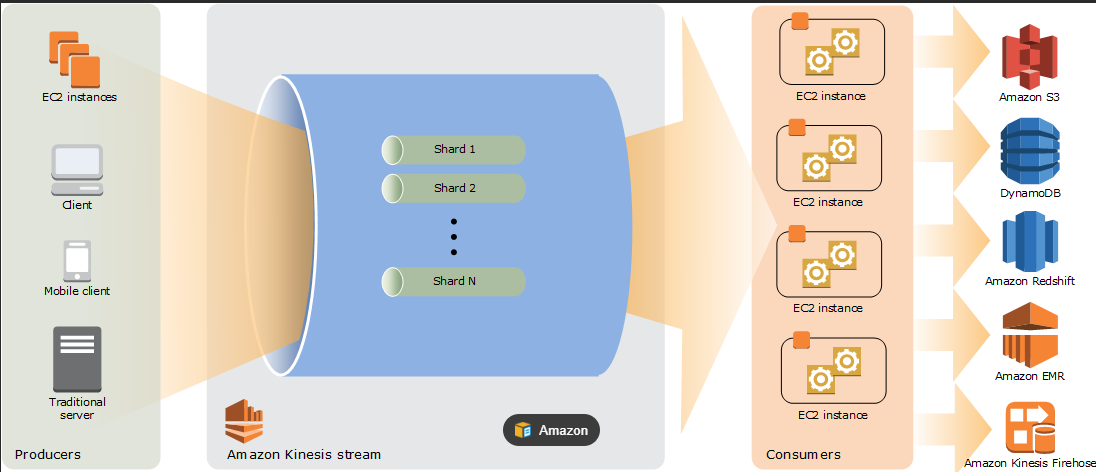
Toda essa movimentação do usuário aqui dentro do Site, nos podemos chamar de Streaming.

Nota: Existem 03 Tipos de Streaming dentro do Kinesis:

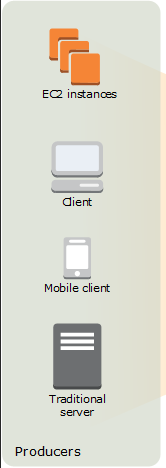
* Kinesis Stream
* Kinesis Firehouse
* Kinesis Analitics

Kinesis Tips:

Vejamos o Primeiro Kinesis onde ele pode ajudar a minha aplicação e o meu projeto:



Basicamente temos 02 Lados, onde o Amazon Kinesis Fica no Meio

 Este lado, esquerdo e aonde estão os producers.

Os Producers, e quem vai produzir os dados, quem vai alimentar o Kinesis, qualquer tipo de fonte que gere dados contínuos, ele pode ser utilizado como producer.

No nosso exemplo, a Instancia EC2, o website está hospedado lá e os usuários estão fazendo os processos.

O Kinesis streaming como ele faz:

Como se fosse um Mega HD com blocos dentro desse HD, os blocos são chamados de Shard.

 então são blocos de armazenamento onde informações vão ser armazenadas, por um período chamado Período de Retenção (retency Peoriod)

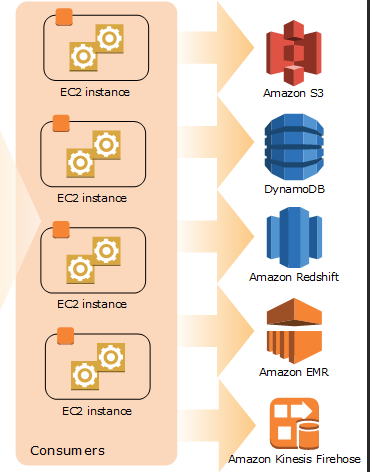
* O período de retenção pode ser de até 24hs até 168hs, que e 7 dias.

Então ele armazena esses dados, mantém esses dados disponíveis, que foram alimentados pelos producers (você pode mudar de acordo com a sua aplicação)

* Os Consumer começam a consumir esses dados, hoje cada Shared, ele consegue suportar 05 Transactions peer Second = ele consegue deixar fazer 5 leituras por segundo dentro de cada Shared.

Cada Shard consegue armazenar no máximo em uma velocidade de 2Mb por segundo, 1000 entradas por segundo para leitura.

Consumer:



* Elas podem ser maquinas EC2, que estão lendo os dados, processando os dados, depois de Lidos, podemos armazenar em uma Bucket S3, ou em Tabelas DynamoDB, ou enviar para o Kinesis FireHouse



Kinesis FireHose:

Ele e Parecido com o Kinesis Stream, porém ele não deixa disponível para o Consumer puxar a informação,

Então você tem alguém gerando os dados, o que o Kinesis Firehose vai fazê-lo pega esses dados e Armazena automaticamente essa Stream de Dados, dentro de uma Bucket S3 ou dentro de uma Redshift, ou dentro do Amazon Elastic Search

Conseguintemente se você tiver uma necessidade de visualizar (View) esses dados, você consegue visualizar através da Bucket S3, porque esses dados serão consumidos e processados dentro da Bucket.

Você não precisará do consumer, ele não aparece.

O firehose – coloca todos os dados dentro do Amazon S3 – e uma aplicação especifica acessa essa bucket S3 com as devidas permissões de acesso e policys e consome esse recurso.

Kinesis analitycs:

Coloca todos os outros juntos em um lugar só, pegando os exemplos anteriores, você sempre tem um Source (Origem) de Streaming. Sempre tem dados sendo populados dentro do seu Site.

Só que, como eu analiso esses dados aqui dentro, como eu posso puxar informação desses dados e colocar em gráficos em analise, pra saber quanto está crescendo, ou qualquer outro tipo de métrica que você queira:

Ai que vai entrar o Kinesis Analitycs, ele faz o SQL Querys, então ele faz perguntas tanto no Kinesis Stream, quanto o FireHose para identificar informações, de como as vezes, qual a quantidade de pedidos tem esse site, usuários, pedidos, clientes, todas as métricas.

Gerando todas as métricas ele pode enviar para uma interface e armazenar:

-S3 – Armazenar os Resultados

-Redshift

-ElasticSearch

Nota: A utilização do Analitycs, ela vem depois que o Kinesis, tanto Stream, quanto FireHose já estão funcionando, porque eles podem alimentar o Kinesis analitycs e você consegue tirar varias Metricas do que esta acontecendo.



HANDS ON Kinesis LAB:

Basicamente, nos vamos ter uma Straming Data, que vai alimentar um servidor EC2 T2.Micro, e essa maquina EC2 vai ter um website, vai ter um gráfico um chart, ele vai ficar movendo como se fosse um gráfico de processamento ou tempo de resposta.

E essa maquina EC2 ela vai alimentar o Kinesis que nos vamos habilitar o formato Stream, e esse Kinesis vai armazenar os dados dentro de uma tabela do DynamoDB.

-A própria AWS disponibiliza uma Template tudo isso já formado, onde podemos executar essa template dentro do CloudFormation, bem simples.

Criação da Stack CloudFormation:



Não se esqueça, dentro da Stack, deletar ela, pois ele irá deletar todas as tabelas e maquinas EC2 que foram criadas usando o CloudFormation

INTRODUÇÃO - ElasticBeanstalk:

Ok vamos falar sobre Elastic Benstalk, agora imagine o seguinte cenário para falarmos de uma forma bem interessante:

Imagine que eu User, tenho a necessidade de subir um servidor web então a primeira tarefa e acessar a EC2 e criar o servidor virtual.

Fazer a instalação do segmento de Developer para que este website funcione (WebServers).

Utilizando outro cenário onde temos um código, com alguns cliques eu subo um serviço, o serviço já cria o webServers de acordo com seu código.

Qualquer atualização que precise fazer no seu código, você pode utilizar uma aplicação em separado para ela atualizar o código, sem que tenha que acessar o servidor, fazer upload do código, atualizar as linhas de programação do site.

O ElasticBeanstalk e um serviço da AWS, que gerencia e realiza deployment de qualquer aplicação com as plataformas que são disponíveis na AWS hoje em dia.

E que para que esse deployment aconteça, o que o ElasticBeanstalk faz basicamente e:

Acessa o serviço, e fala: olha eu quero que você crie um servidor que nós vamos chamar de Enviroment, deployment a PHP suporte:

O que o Elastic BeanStalk vai fazer, ele vai abrir uma seção, criar uma maquina EC2, vai criar todas as permissões no IAM, vai criar uma bucket S3, para que depois você possa jogar seu arquivo Zip no próprio BeanStalk.

Ele faz o upload do próprio BeanStalk para a bucket S3, faz o unzip, joga os arquivos que estão lá dentro dentro do servidor com a aplicação PHP funcionando.

Leitura Complementar ElasticBeanstalk:

- O que é o Elastic BeanStalk:

O Amazon Web Service (AWS) consiste em mais de cem serviços, cada um dos quais expõe uma funcionalidade. Embora a variedade de serviços ofereça flexibilidade para você decidir como vai gerenciar a infraestrutura da AWS, pode ser um desafio descobrir quais serviços usar e como provisiona-los.

Com o Elastic Beanstalk, possível implantar, gerenciar rapidamente aplicativos na nuvem AWS sem precisar conhecer a estrutura que executa esses aplicativos.

O elastic beanstalk automaticamente gerencia os detalhes de fornecimento de capacidade, balanceamento de carga, fornecimento de capacidade, e monitorar o status do aplicativo.

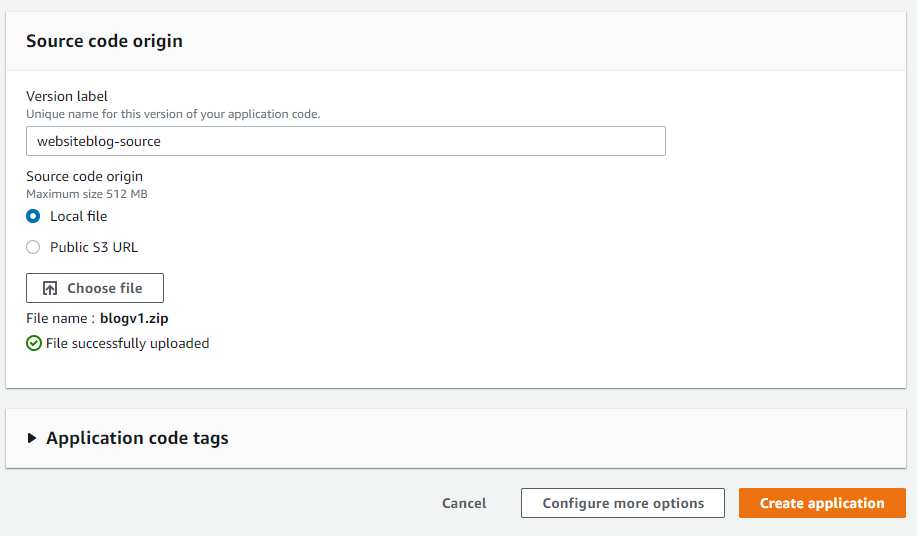
O elastic Beanstalk oferece suporte a aplicativos escritos desenvolvidos em GO, Java, .NET, Node.js, PHP, Python e Ruby. Quando você implanta o aplicativo, o ElasticBeanstalk cria uma versão da plataforma compatível selecionada, e provisiona um ou mais recursos da AWS: Como Instancias EC2 para executar o aplicativo.

Voce pode interagir com o Elastic Beanstalk com o Console de gerenciamento da AWS, a AWS Comand line Interface (AWS CLI), ou **(eb**), uma CLI de alto nível desenvolvida especificamente para o Elastic Beanstalk.

HANDS ON ElasticBeanstalk

Laboratório Elastic Beanstalk:

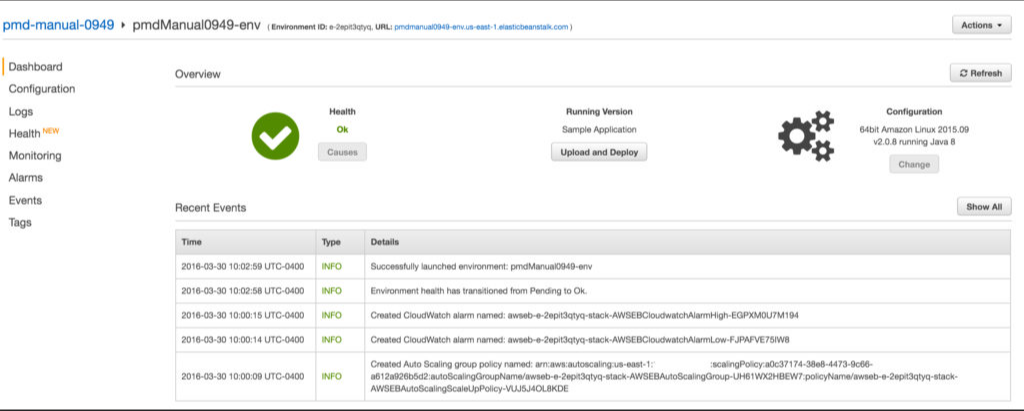
Depois de subir o arquivo da version1:



Clicamos em Create Application.

A aplicação irá ser construída e as Permissions no IAM serão criadas junto com uma Bucket S3.

Após o termino da criação da Application, utilizando o Elastic Beanstalk, nossa tela será este aqui:



-Será abordado cada tópico deste:

-DASHBORD: exibe as informações do dashboard

-CONFIGURATION:

Na parte de Configuração, onde e a parte que vc pode mexer um pouco mais:

SOFTWARE: alterar o **Document root**, **Memory limit**

INSTANCES: conseguimos gerenciar as instancias, quero mudar para a instancia, através do **Evironment type** (**single instance** or **Load Balancer**, podemos mudar a instancia através do **Instance Type.**

CAPACITY: conseguimos gerenciar as capacidades

LOAD BALANCER

ROLLING UPDATES AND DEPLOYMENTS

SECURITY

MONITORING

MANAGER UPDATES

-LOGS: As atualizações e tudo mais e não tenho nada ainda

-HEALTH: Consigo visualizar se tem algum erro, ou tem alguma tarefa em espera para ser executada.

-MONITORING: Consigo visualizar a quantidade de CPU utilizada na maquina EC2, quantidades de bytes que entraram e saíram, ou se meu serviço de Benstalk para o meu Enviroment, se ele já caiu, ou entrou em outro status de erro.

ALARMS: Eu consigo também criar alarms, pra caso, o meu serviço entre em erro, e eu receba um e-mail, ou um SMS

MANAGER UPDATES: Eu consigo gerenciar os UPDATES

EVENTS: Consigo visualizar os Eventos, que foi o passo a passo de tudo o que aconteceu

TAGS: E gerenciar as minhas Tags também.

Hands On Elastic Beanstalk – Parte 2

Agora temos a necessidade de atualizar o nosso blog, ou o nosso site com o arquivo:

Depois disso eu subo meu arquivo da Version 02 no Benstalk fazendo o Upload dele Choose Your File.

Depois que o deploy desse arquivo e feito, precisamos entender como funciona a parte de Deployment e onde está acontecendo esse deployment.

Modify Rolling Update and Deployments

Eu posso mudar minha política de deployment:

Documentação das políticas de Deployments:

https://doc.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html

**Deployment Policy: Posso mudar**

**All at Once** – Exemplo se ela está em 10 instancias EC2, ela vai atualizar o código nestas 10 instancias ao mesmo tempo (meu site vai ficar fora do ar)

**Rolling –** Olha, dessas 10 pega 5 instancias atualiza e depois pega as outras 05 e atualiza, quer dizer que se eu tenho 10 Instancias e eu tenho muito trafego no site, o que será que acontece se eu tirar metade (processamento alto consumo)

**Rolling with additional** bash – Se eu tenho as minhas 10 Instancias, ele vai criar mais 05 atualizar nessas 05 novas, e derrubar as anteriores, demora mais, porém não tem DownTime.

**Immutable** – Ele pega as 10 instancias, ele cria outras 10 Instancias, atualiza o código, e depois vira o tráfego para esses 10 instancias. (se algo der errado, ele consegue retornar para as 10 instancias antigas, sem DownTime, sem Mudanças de DNS.

**Blue/Green** – Ele cria um novo grupo de servidores EC2, por ser Blue/Green, você vai virar o trafego com Load Balancer e não com o Elastic Beanstalk, virando o trafego com o Load Balancer você precisa tirar de uma DNS, e colocar na outra, teremos que fazer o SWAP de URL.

Introdução ao CI and CD

* CI continuous Integration
* CD Continuous Delivery / Deployment

Basicamente são excelentes práticas de como se criar o código, testar o código, aprovação, testar, reaprovação e depois deployment:

Todas essas etapas, são que se seguidas a risca, e a linha você consegue fazer implementações sem DownTime, você implementa os Updates, sem que ninguém perceba e o sistema não se torne indisponível.

Workflow – significa, quais são os passos que devem ser executados primeiro, para garantir desde a escrita do código até o seu deployment.

Existem ferramentas que podem te ajudar:

Passo a Passo o que seria em boas praticas o que seria utilizar o Continuous Integration, aonde o nosso primeiro ponto aqui, e o que nos chamamos de Code Repository.

CI – Continuous Integration

* GIT – Existe um serviço dentro da AWS que faz com que vc tenha um GitHub dentro da AWS, que você mantenha versões para teste, atualizadas, aprovação ou seja um repositório de código.

As vantagens são, muitos desenvolvedores conseguem codificar, conseguem corrigir bugs no sistema, conseguem compartilhar o código com os outros.

* Management System – Ele vai receber, o código através do GIT, e de uma forma automatizada, ele vai construir esse código dentro do sistema, ou dentro da aplicação que vc está rodando.

Se ele construir dentro da aplicação, e tiver algum erro, como eu vou identificar este erro.

A identificação desse erro, ele vem aqui na nossa terceira Etapa:

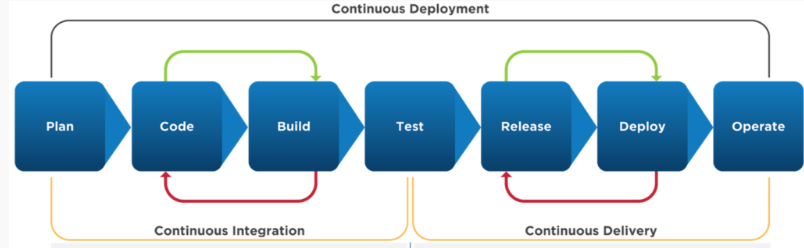
* Test da Aplicação – E aonde você pode rodar, também códigos de forma automatizada, testar a aplicação, a nova codificação, os novos comandos e códigos, e verificar se ela foi aprovada ou se algum código foi aqui reprovado. (O grande foco aqui e o teste, por menor que ele seja), por uma variação que seja em uma linha de código, pra que depois ele possa entrar em produção.

CD – Continuous Delivery/Deployment

* Se na sua aplicação você optar por um frame que, tem como objetivo o Continuous Delivery: Significa que manualmente, alguém vai ter que realizar o teste, e a execução dentro do Enviroment real, ou que nós chamamos de **staging**

Ambiente de teste para jogar depois para a aplicação Final.

* Se você fizer o continuous deployment, ele e totalmente automatizado onde ele vai ler os testes se foram ou não aprovados, e se todos foram aprovados ele vai jogar tudo para a versão final (live)



Code Repository:

Existe um serviço dentro da AWS, que se chama CodeCommit, e o serviço de GIT que vamos ter dentro da AWS, onde vc pode armazenar, compartilhar e ter versões diferentes sempre em direção da versão final de um código

Sistema de gerenciamento – Management System se chama CodeBuild, que vai fazer a construção dos códigos, dentro do sistema, e a parte de desenvolvimento, tanto de delivery como deployment temos um serviço chamado CodeDeploy. Para cada passo fora os testes, nós temos um serviço.

CODE COMMIT -> CODE BUILD -> CODEDEPLOY -> TEST

Tudo isso pode ser gerenciado com um outro serviço AWS chamado **CodePipeline**:

Introdução ao CodeCommit

E um serviço de Code Control, onde nós podemos armazenar os códigos e vários desenvolvedores podem armazenar os seus códigos, e os códigos também são numerados com as suas versões.

Conseguimos fazer o tracking, checar o código mais atual e o código mais antigo, e armazenar versões, gerenciar melhor o seus Updates, e o colaboration aonde vários desenvolvedores em cima do código master, ser enviado para aprovação e depois fazer o merge (todas as versões se tornam uma master)

* Existe um sistema de Branch, quando você cria um repositório tem inicialmente um master code:

File 01.php

File 02.php

Eu vou criar outra branch, e essa branch eu vou copiar o arquivo número 02 para ela – atualizarei.

Depois faço o merge para o repositório Master.

Hands On CodeCommit:

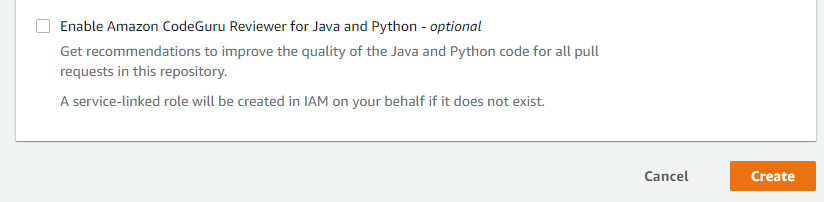
1 PASSO - Vamos criar um repositório no CodeCommit

2 PASSO – Depois vamos criar um file.01.php dentro do repositório.

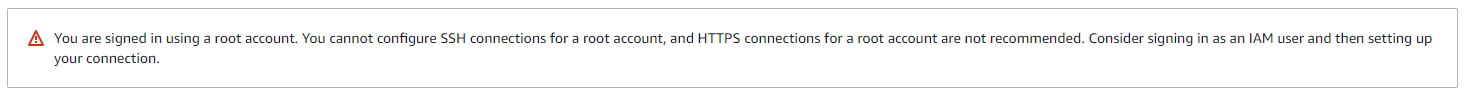
3 – PASSO – Vamos criar uma Branch com o meu nome, e fazer alterações no código.

4 – PASSO – Depois que eu fizer o CodeCommit faça o Merge, levar as minhas configurações para ambiente real.

Se você estiver programando em Java ou em Python, a AWS tem um serviço de revisão que e o CodeGuru:



Tem está mensagem de erro que deve ser comentada:



Estamos utilizando a conta root, a conta principal não e uma conta de usuário com privilégios na plataforma da AWS.

É por ser uma conta principal, como que eu vou identificar quem são as branchs, quando eu estiver utilizando por exemplo, os softwares de GIT para fazer o upload diretamente da sua maquina:

Você não consegue configurar conexões SSH e HTTPs para root account.

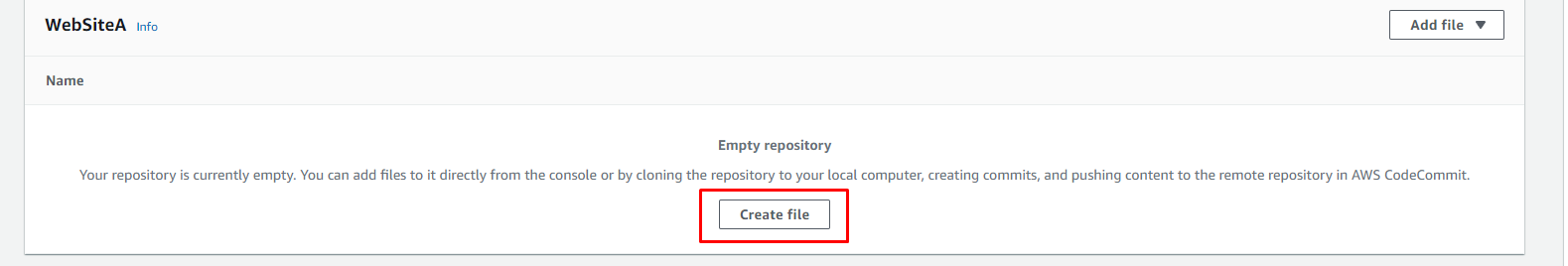
Mais aí e só criar um account dentro do seu desenvolvedor dentro do IAM aplicando políticas e dando privilégios ao CodeCommit a plataforma da AWS, e quando ele acessar o repositório e criar a Branch ela vai ser associada ao usuário dele IAM.

## Basicamente, existem 02 formas de criar este arquivo:

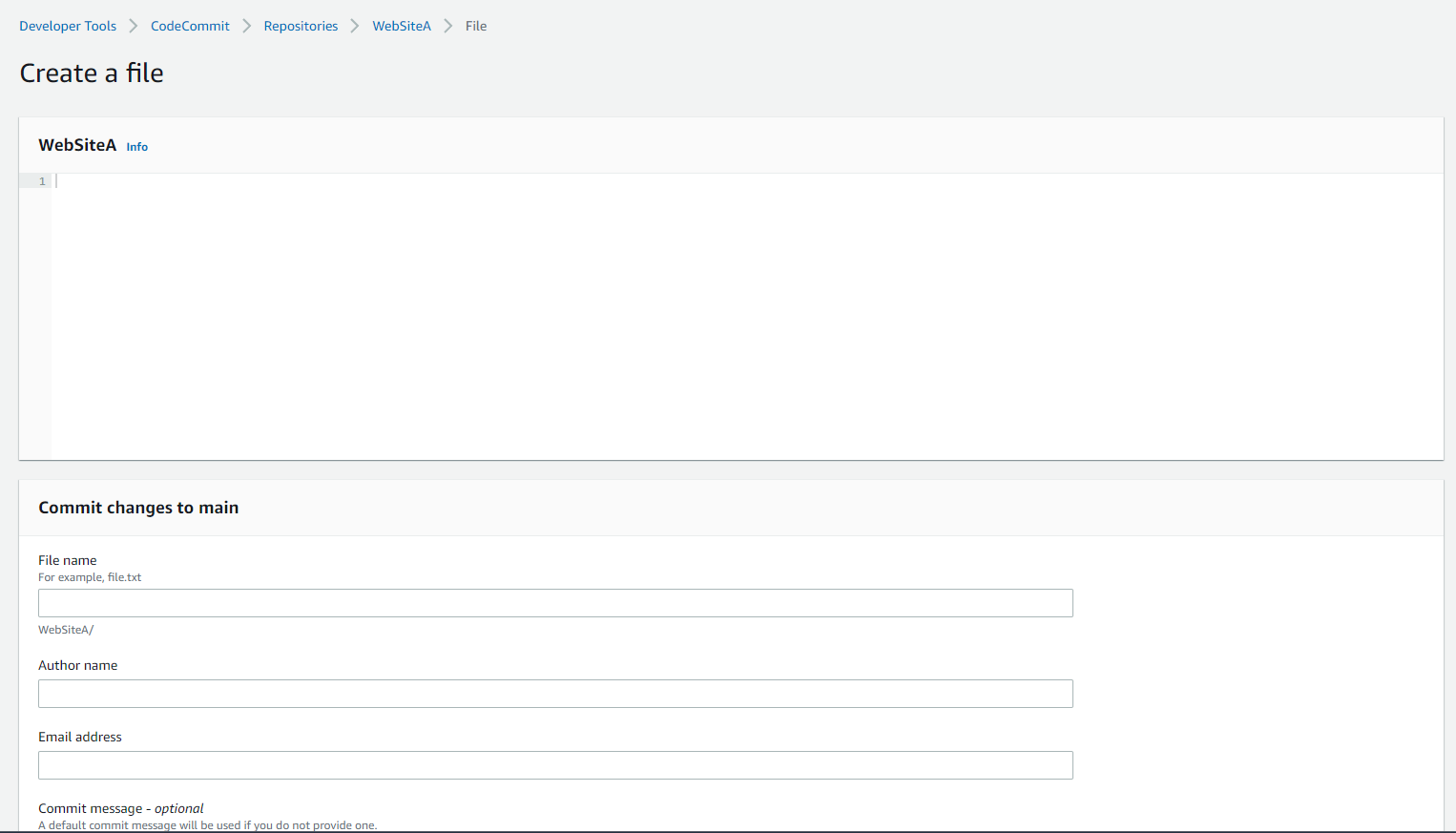
* 1° A própria Pagina da AWS
* 2° e via software chamado GIT (open Source) se conectar a conta da AWS.

Importante: Todas as Interações entre o software VScode e o git com o CodeCommit são encriptadas em SSH e são em HTTPS seguro.

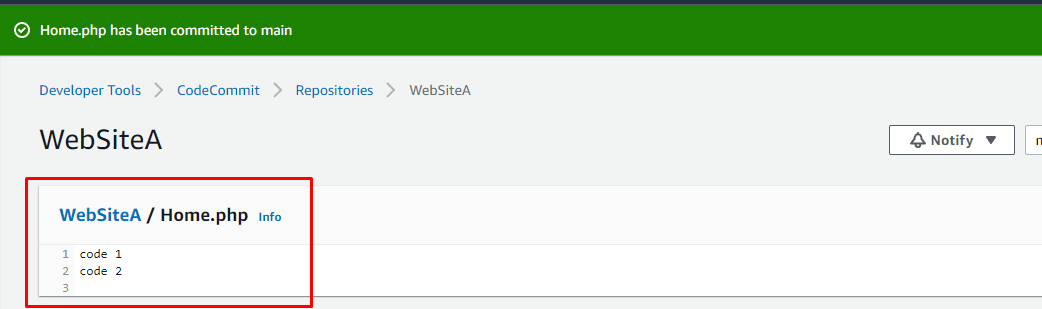
**Criando Via própria pagina da AWS teremos a nossa linha de comando:**



Dessa forma:

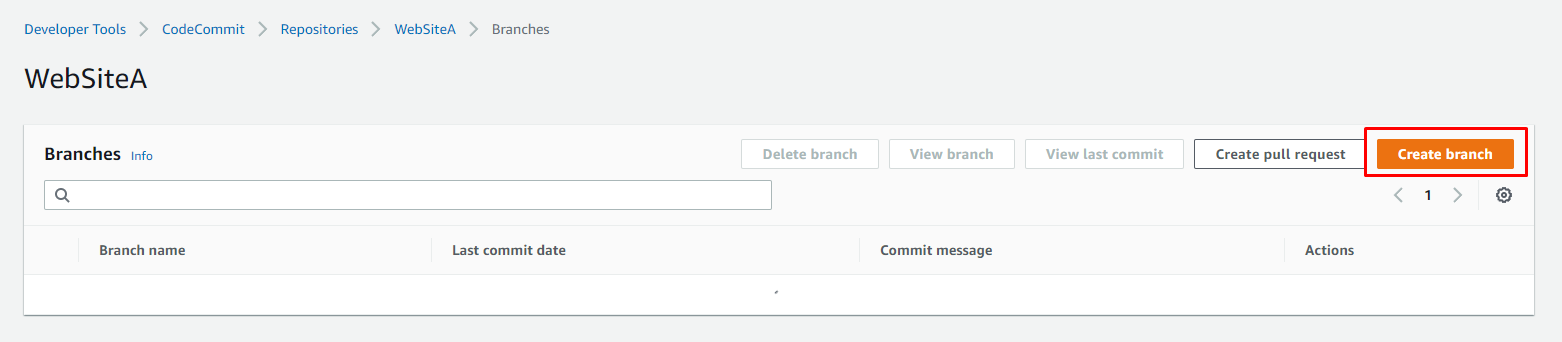


Criando nosso primeiro arquivo chamado Home.php:

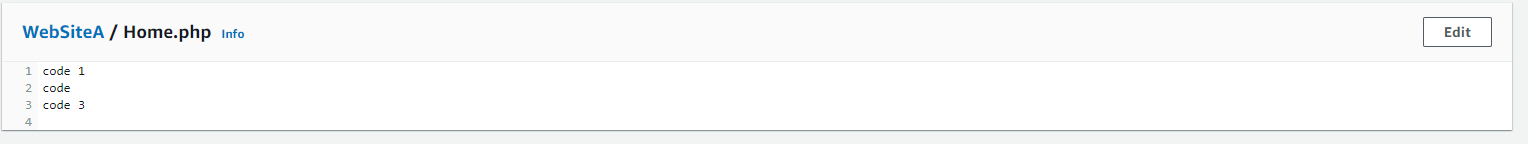


Este código está dentro da Master Branch:

Agora eu entro em branchs e crio uma branch nova:



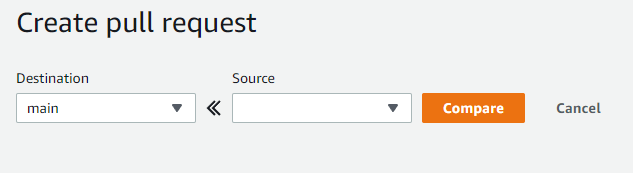
Depois de criar a breach faço alterações no meu código:



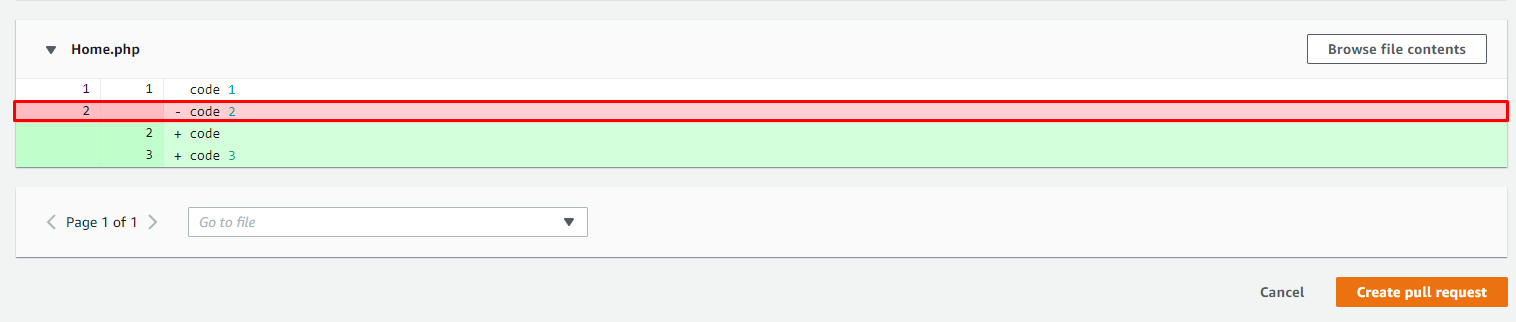
Depois que as alterações forem feitas eu volto na Branch Master e faço o Merge:

E vou criar um **PULL REQUEST**

**Create PULL REQUEST ele irá puxar a informação de alguém:**

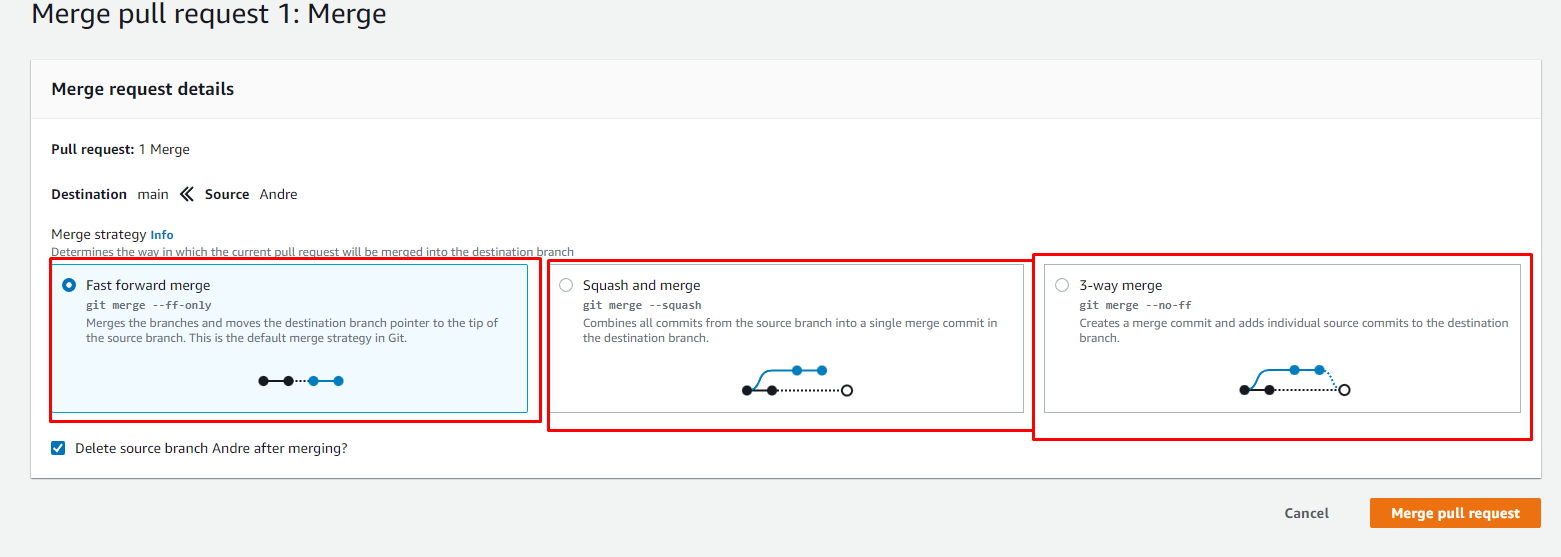


Clicando em Copare, ele irá mostrar o que foi alterado no código:



Se eu estiver satisfeito com as minhas changes, eu posso dar um nome a esse Merge, e dar uma descrição

Clicando em create pull request, ele me pergunta que o merge será do tipo Manual a aprovação então seguindo com os tipos de Aprovação:

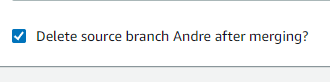


Fast Forward Merge – Onde ele Subscreve todos os Arquivos

Squash and Merge - Onde ele combina todos os commits do Source com o Destination e faz um por um.

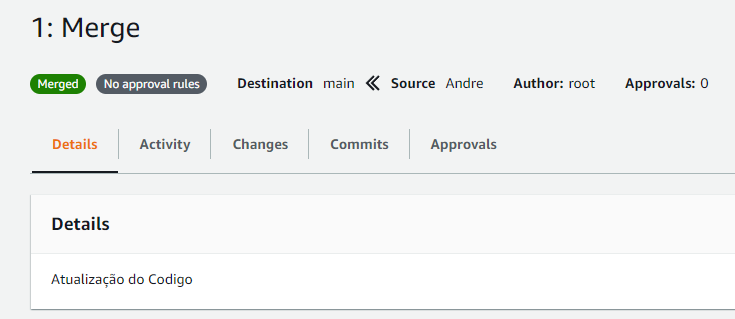
3 – Way-Merge – Onde ele pega e adiciona origem individual a partir do destino.

NOTA: Se deixar esta caixa marcada, ele vai deletar a Branch depois do Merge.

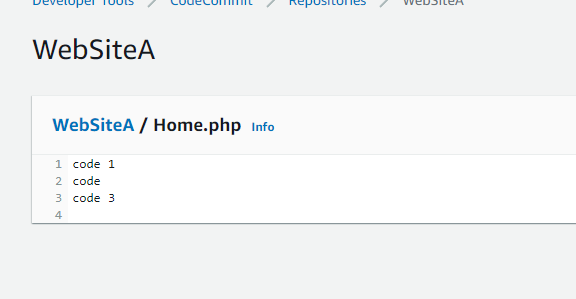


Clico agora em Merge Pull Request:

Esse São so Detalhes:



Resultado final do meu Merge:



Code Deploy

Serviço da AWS, que basicamente ele foi criado para automatizar o desenvolvimento, ou seja a instalação da aplicação tanto em maquinas EC2, ou maquinas que você tenha na sua empresa, ou sistemas ON-Premisse, ou podemos fazer isso também em Lambda Functions.

Criamos um Deploy, ele e mantido através de arquivos, e existem códigos e scripts lá dentro e tudo que cria tudo dentro do Code Deploy (Grande vantagem, ele reduz qualquer tipo de erro manual)

O code Deploy ajuda no quesito de instalação de bibliotecas ou atualizações tanto dentro de uma EC2, ou em outras aplicações da sua empresa.

Evitando erros e o processo rápido.

Integrações com outras ferramentas de CI e CD:

Por exemplo:

* Jeankins
* GitHub
* Atlassant
* AWS Pipeline
* Assinble
* Puppet
* Chef

Todas essas ferramentas o Code Deploy consegue fazer a Integração.

Depois que a integração estiver feita, você precisa escolher um dos dois tipos de fazer o Deployment:

02 Tipos:

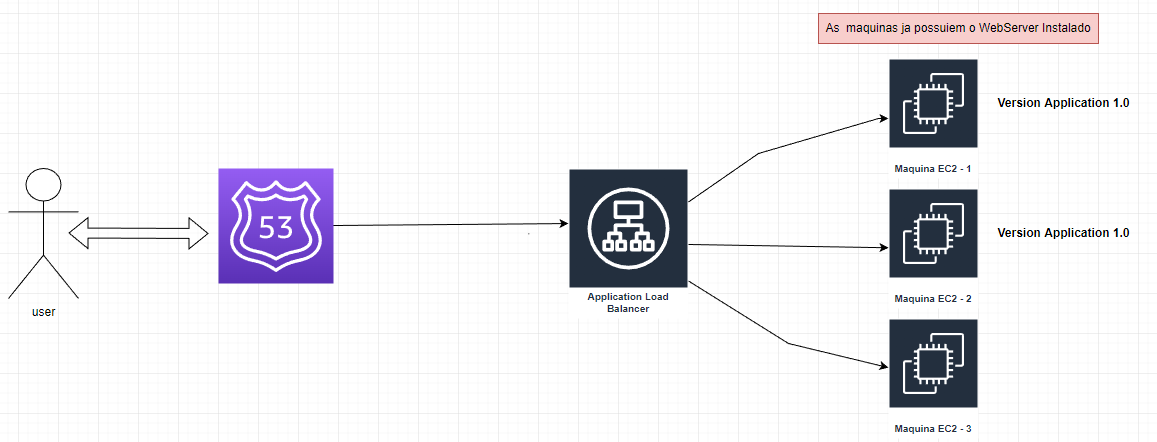
1° O que nos temos como definição de In-Place

2° Tipo Blue/Green

Quando você finaliza a integração, ele vai te perguntar como você quer realizar o Deploy, a execução do seu projeto como ela vai ser:

In-Place:

Note que para funcionar em um Auto-Scaling, seria bom colocar um Load Balancer:



Rodando o Codeplay, você vai nomear esses três servers ou o auto-Scaling-Group, dizendo: Olha eu tenho a versão 2.0 do software – **CODEDEPLOY,** o code Deploy vai começar a Instalar a versão 2.0 nos meus servers.

Quando você escolhe ele não acontece, sem o Down-Time da Aplicação ele vai ficar fora do ar por algum tempo durante o Deploy ok.

Ele primeiro derruba a primeira máquina, atualiza para a versão 2.0, e depois sobe a máquina.

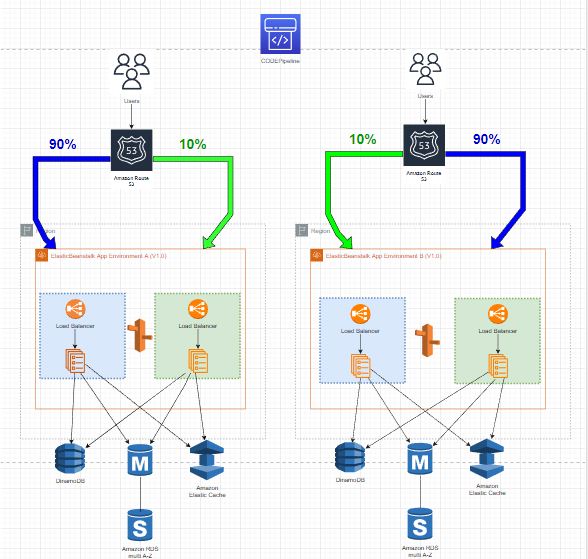
Os recursos dentro da sua instancia de Auto-Scaling vai ser reduzida, você pode até falar no load Balancer, não envia mais para a Instancia 1, envia pra 2 e 3.

E assim teremos um pouco mais de trabalho pra derrubar servidores remanejando o trafego via load balancer, para determinadas instancias, a fim de atualizar a versão.

Blue/Green

Podemos ilustrar neste gráfico:

Você cria uma nova instancia, se eu tenho 03 maquinas EC2, e instala toda a sua aplicação de versão 2.0 nessas maquinas, no Load Balancer, você vira o trafego para o outro lado, qual a grande vantagem, instantaneamente, quem atualiza a pagina e o DownTime e zero.



E se alguma coisa der errado, você vai lá no Load Balancer e Vira o trafego de volta.

Termos:

Vamos criar um Grupo dentro do **Code Deploy**, ele irá se chamar deployment Group:

Nada mais nada nada menos você fala quantas instancias EC2 você vai instalar, Deployment configuration, quais são as regras, se ele falhar e pra fazer o que, se ele obtiver sucesso e pra fazer o que.

Appsec file – são as Instruções que você passa para o CodeDeploy, o que vai fazer primeiro.

1° atualiza o servidor EC2

2° atualiza o HTTPS

3° Copia a pagina Index para um bucket S3 pra raiz desse servidor

4° Pausa o servidor, para que reinicie a máquina EC2 para que essa máquina esteja disponível para acesso.

Revisão: teste se está funcionando ou não.

Hands On Lab CodeDeploy

Criação de um WebServers, bucket S3 vai ter os meus arquivos de Script:

Scripts e páginas HTML, ele vão ter instruções de instalar o webServers, copiar os arquivos para a Instancia EC2, iniciar o servidor.

Dentro da minha Bucket, vai ter o Index.html, vai existir na Bucket.

Montar uma web server dentro da Bucket e Atualizar via Code Deploy.

1° Coisa que vamos fazer e criar Regras IAM:

O serviço de Deploy vai ter que ter acesso a uma Bucket S3 e ter acesso a Instancia EC2.

-Acesso Full acess a S3 e EC2