AWS CERTIFIED DEVELOPER - Apresentação

CURSO COMPLETO

CERTIFICAÇÃO DESENVOLVEDOR AWS

Dicas para este curso

- Faça anotações!
- Pratique o máximo que puder
- Defina a data que você quer passar no exame
- Quando estiver cansado, faça uma pausa
- Assista novamente as lições
- Você pode acelerar ou desacelerar os vídeos
- Não tenha pressa

Formato do Curso

- Lições teóricas com slides e diagramas
- Lições práticas para fortalecer o aprendizado
- Quizzes ao final de cada seção
- Simulados da prova ao final do curso

Exame AWS Certified Developer Exam (DVA-C01)

- Será avaliado seu conhecimento da plataforma AWS sobre a perspectiva de desenvolvedor.
- 2 tipos de questões:
  - Multipla escolha: 1 resposta correta e 3 erradas
  - Múltipla resposta: duas ou mais respostas corretas de um total de 5 ou mais
- Não há penalidade para pergunta não respondidas ou respondidas erradas
- Para passar no exame é necessário pontuação de pelo menos 720 (máximo 1000)

Peso de cada tópico

1: Implantação(Deployment)	22%
2: Segurança	26%
3: Desenvolvimento com serviços AWS	30%
4: Migração	10%
5: Monitoramento e solução de problemas	12%
Total	100%

- Implantação(Deployment): CICD, Beanstalk, Serverless
- Segurança: Cada serviço + uma seção dedicada somente a segurança
- Desenvolvimento: Serverless, API, SDK e CLI
- Migração: Entendendo serviços AWS para otimização de migração
- Monitoramento e solução de problemas: CloudWatch, CloudTrail, X-Ray

Avaliação do Curso

- Udemy vai pedir sua avaliação do curso logo no início
- Avaliação é muito importante para mim e para outros alunos que estão interessados no curso. Eu gostaria muito que você avaliasse esse curso
- Não avalie o curso até que você esteja pronto. Clique “pergunte depois”
- Nós vamos começar do básico. Fique a vontade para pular as lições se você tiver a certificação AWS solutions architect.

O que você vai aprender nesse curso


AWS EC2


ECR


ECS


BEANSTALK


LAMBDA


ROUTE 53


LOAD BALANCING


CLOUDFRONT


KINESIS


S3


RDS


DYNAMODB


ELASTIC CACHE


SQS


SNS


SWF


STEP FUNCTION


API GATEWAY


SES


COGNITO


SYSTEM MANAGER


IAM


CLOUDWATCH


KMS


CLOUDFORMATION


CLOUDTRAIL


CODECOMMIT


CODEBUILD


CODEDEPLOY


CODEPIPELINE


X-RAY

Serviços Essenciais AWS

IAM

AWS EC2

S3

LOAD BALANCING

ROUTE 53

RDS

ELASTIC CACHE

Ferramentas de desenvolvimento


aws cli


IAM


nodejs


python

Distribuição contínua, monitoramento e plataforma como código

BEANSTALK

CLOUDWATCH

KMS

CLOUDFORMATION

CLOUDTRAIL

CODECOMMIT

CODEBUILD

CODEDEPLOY

CODEPIPELINE

Fragmentação e Integração da Aplicação

SERVERLESS

SAM

DYNAMODB

API GATEWAY

COGNITO

LAMBDA

SEGURANÇA

IAM

SYSTEM MANAGER

KMS

ENCRYPTION SDK

VISÃO GERAL DOS OUTROS SERVIÇOS

ECR

ECS

CLOUDFRONT

STEP FUNCTION

SWF

SES

Fundamentos AWS - Parte 1

- Regions (Regiões)
- IAM
- EC2

Regions e Availability Zones

REGION: US-EAST-1 / sa-east-1
AVAILABILITY ZONE: US-EAST-1A / sa-east-1a
AVAILABILITY ZONE: US-EAST-1B / sa-east-1c
AVAILABILITY ZONE: US-EAST-1C

- AWS tem Regions em várias cidades ao redor do mundo. Ex. us-east-1
- Cada Region tem Availability Zones. Ex us-east-1a, us-east-1b, us-east1c.
- Cada Availability Zone é um data center isolado fisicamente dos outros data centers. O motivo é evitar que um desastre afete mais de uma Avalilability Zone
- Os Serviços AWS são separados por Regions com exceção de IAM e S3

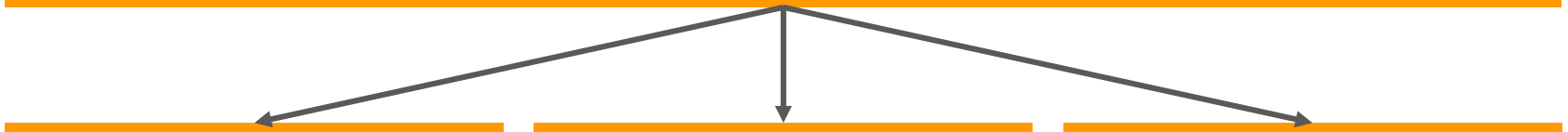
Introdução IAM

- IAM (Identity and Access management) = Identidade e Gerenciamento de acesso
- Muito importante para segurança na plataforma AWS
  - Users (usuário)
  - Groups (grupos)
  - Roles (atribuição)
- Root Account nunca deve ser usada ou compartilhada
- Aos usuários deve ser dada mínima permissão para executar tarefas requeridas
- Policy (regras) são escritas in JSON (Javascript Object Notation)

Introdução IAM

POLICIES

Define o que é permitido e o que não é permitido para as entidades abaixo



USERS

Normalmente atribuído a uma pessoa

GROUPS

Normalmente atribuído a times ou função

Ex: Desenvolvedores
Engenheiros

ROLES

Atribuído a recursos AWS.
Ex: EC2

Introdução IAM

- IAM é um serviço global
- Permissão é definida pela Policy escrita em JSON
- MFA (Multi Factor Authentication) pode ser configurado
- IAM tem Policies pré definidas (managed policies)
- Veremos IAM policies em detalhes mais pra frente
- Devemos dar aos usuários a mínima permissão que eles precisam para executarem suas tarefas.

IAM FEDERATION

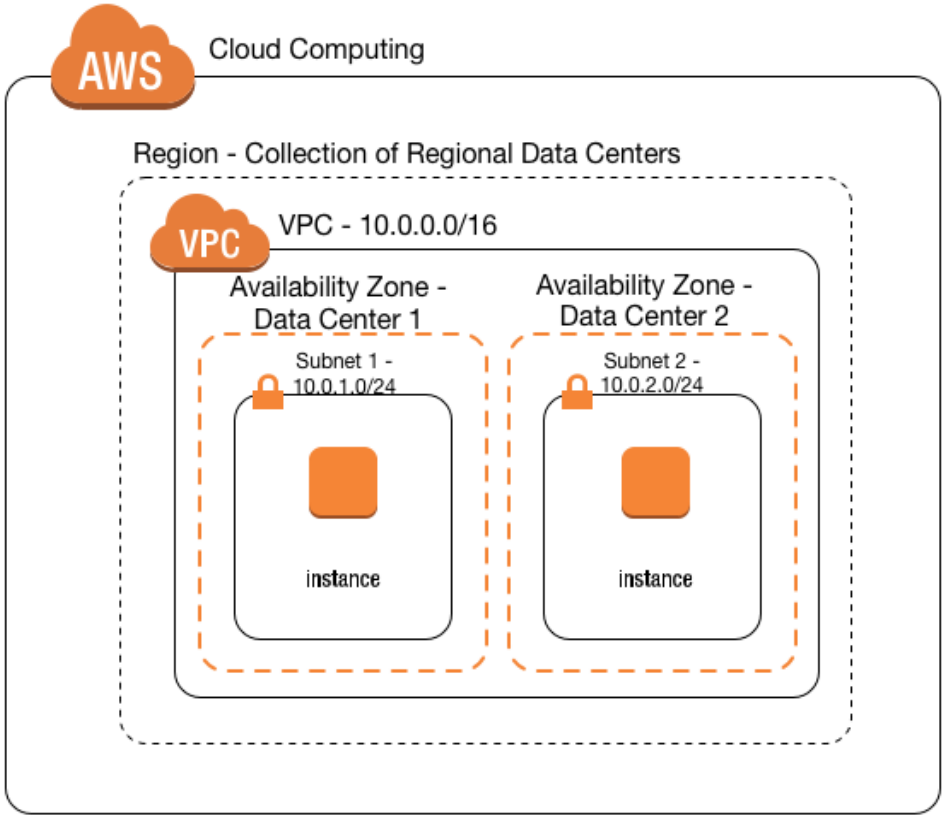
- Grandes empresas normalmente integram seu repositório de usuários com IAM
- Dessa forma os usuários pode acessar AWS usando a credenciais da empresa
- Identity Federation usa o padrão SAM (active directory)

IAM - NÃO ESQUECER !!!

- 1 user IAM por pessoa
- 1 role por aplicação
- Credenciais IAM nunca devem ser compartilhadas
- Nunca coloque suas credenciais no código da sua aplicação
- Nunca use a conta ROOT. Somente para a configuração inicial
- Nunca use credenciais da conta ROOT

O que é EC2 (Elastic Compute Cloud)

- Um dos mais populares serviços AWS
- Funcionalidades EC2:
  - Aluguel de Máquina Virtual ou Virtual Machine (EC2)
  - Armazenamento de dados em discos virtuais (EBS)
  - Distribuição de tráfego entre as Máquinas virtuais (ELB)
  - Auto Escalonamento using auto-scaling group (ASG)
- Entender EC2 é fundamental para entender como as nuvens funcionam



Prática: Criar uma Instancia EC2 rodando Linux

- Vamos criar nosso primeiro servidor virtual usando o console AWS
- Vamos conhecer os principais parâmetros de configuração EC2
- Vamos aprender como iniciar, parar e terminar nossa máquina virtual

Conectar com instancia EC2 usando SSH

- Para usuários Linux ou Mac
- Vamos aprender usar SSH para conectar com nossa máquina virtual usando linha de comando
- SSH é muito utilizado para conectar com computadores remotos, pois é uma forma segura de conexão se usado de forma correta.

SSH - PORT 22

INTERNET

MÁQUINA VIRTUAL - INSTANCIA EC2
COM IP ADDRESS PÚBLICO
RODANDO SISTEMA OPERACIONAL LINUX

Conectar com instancia EC2 usando SSH

- Para usuários Windows vamos usar a ferramenta grátis Putty
- Vamos aprender usar SSH para conectar com nossa máquina virtual usando linha de comando
- SSH é muito utilizado para conectar com computadores remotos, pois é uma forma segura de conexão se usado de forma correta.

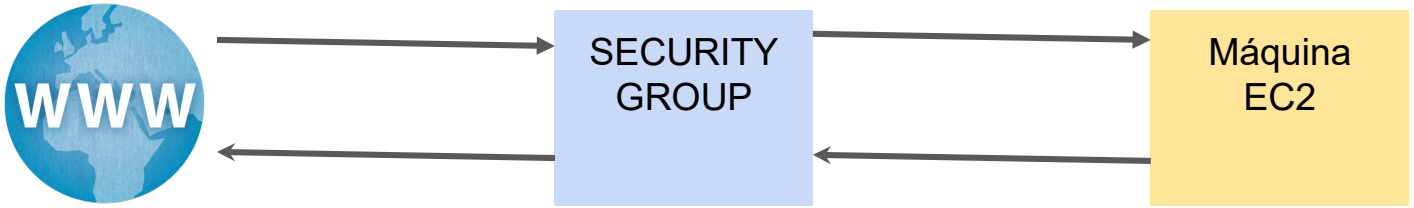
SSH - PORT 22

INTERNET

MÁQUINA VIRTUAL - INSTANCIA EC2
COM IP ADDRESS PÚBLICO
RODANDO SISTEMA OPERACIONAL LINUX

Introdução Security Groups

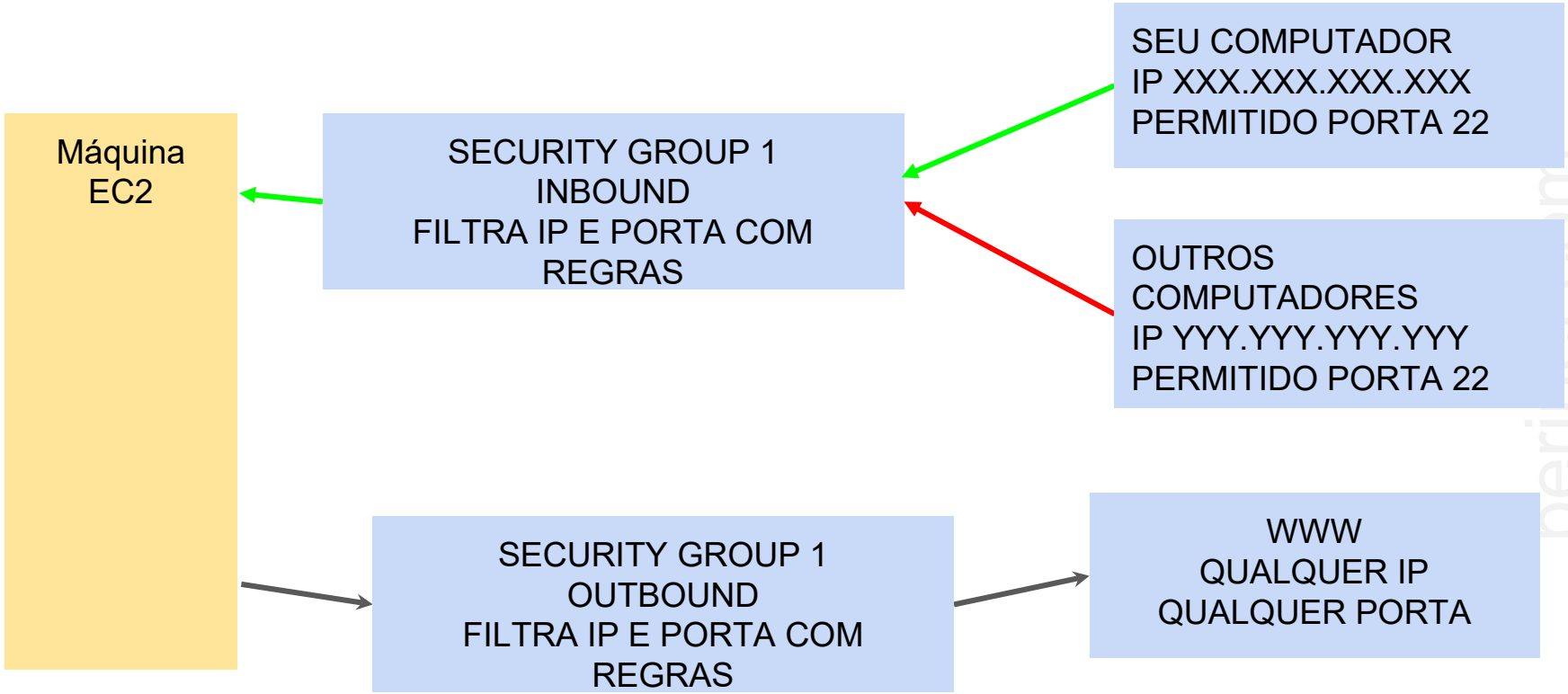
- Security group é o fundamento básico para segurança no AWS
- Security group determina o que é permitido entrar ou sair da maquina virtual EC2
- É o fundamento mais importante para solucionar problemas de rede no ambiente AWS
- Nessa lição nós vamos aprender a usar security groups para permitir tráfego entrando e saindo do EC2



Aprofundando em Security Groups

- Security Groups funcionam como um firewall para instancias EC2.
- Security Groups controlam:
  - Acesso a portas
  - Endereços IP permitidos (ipv4 e ipv6)
  - Controla entrada de dados para a instancia EC2
  - Controla saída de dados da instancia EC2

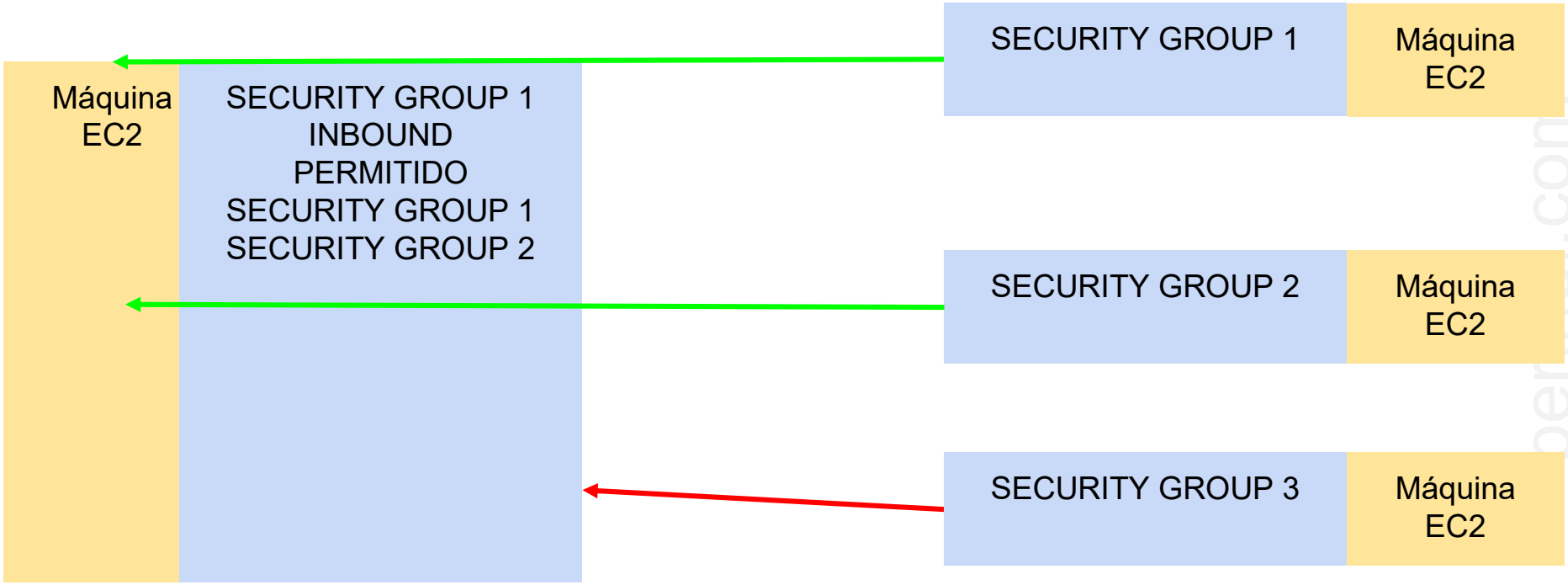
Diagrama Security Groups



security Groups - Importante Saber

- Pode ser associado a multiplas instancias
- Escopo: VPC
- Fica fora da instancia EC2. Tráfego é bloqueado antes de chegar a instância EC2
- Boa prática ter um security group só para acesso SSH
- Se voce não consegue acessar EC2 (time out) o problema é provavelmente Segurity Group
- Se o erro for Connection Refused, o problema NÃO é security group
- Todo tráfego de entrada é bloqueado por padrão
- Todo tráfego de saída é permitido por padrão

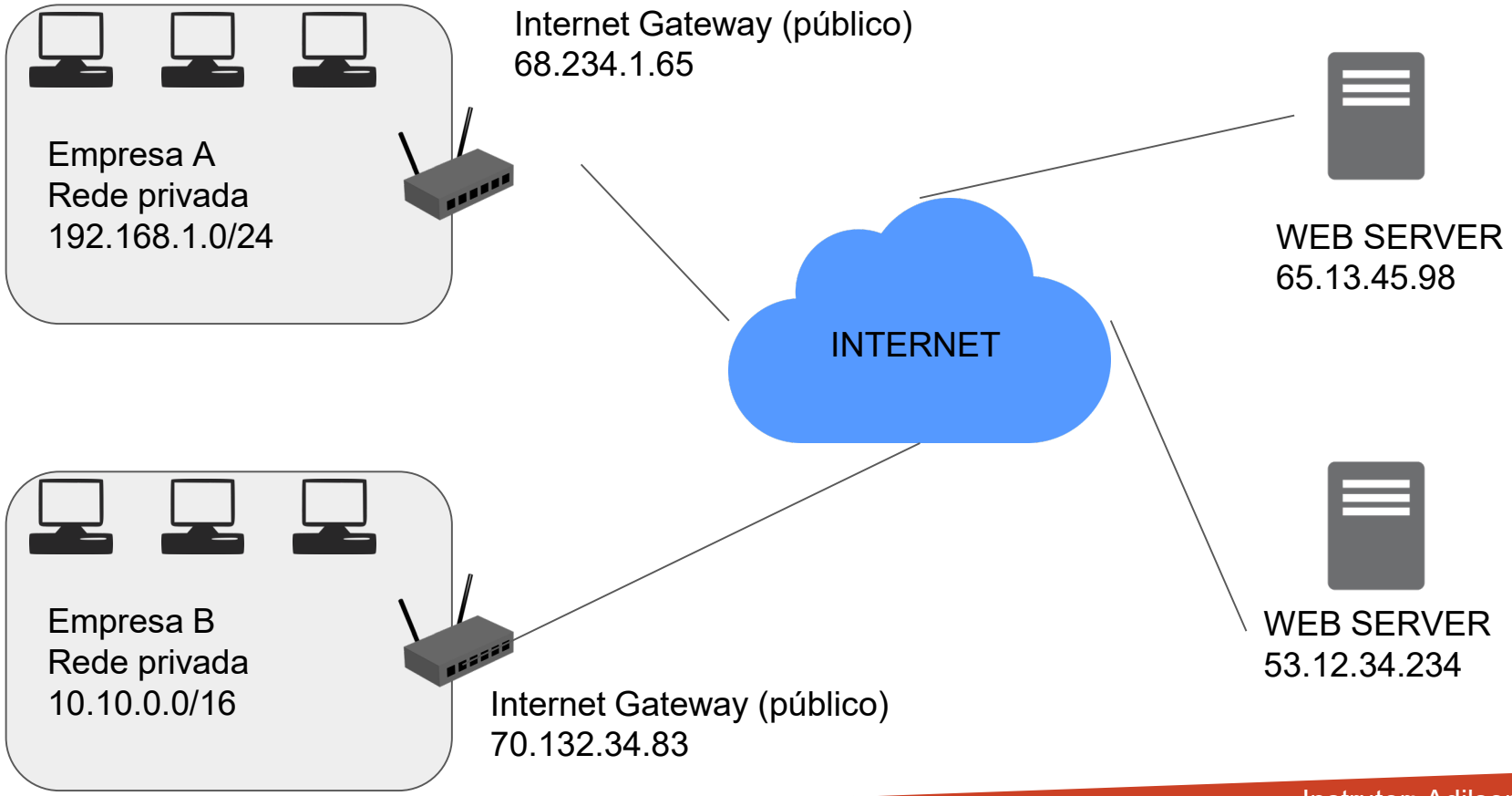
security Groups - referenciando a outro security group



ENDEREÇO IP PÚBLICO E PRIVADO (IPV4)

- Endereço IP pode ser IPv4 ou IPv6
- IPv4: 68.250.23.26
- IPv6: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
- Nesse curso nós só vamos utilizar IPv4
- IPv4 é o formato mais utilizado na Internet
- IPv6 é o mais novo formato e resolve o problema IoT (Internet of Things)
- IPv4 tem mais de 3.7 bilhões de endereços públicos
- Formato IPv4: [0-255].[0-255].[0-255].[0-255]
- Private IPv4: 10.0.0.0/8 [10.0.0.0 - 10.255.255.255]
  172.16.0.0/12 [172.16.0.0 - 172.31.255.255]
  192.168.0.0/16 [192.168.0.0 - 192.168.255.255]

IP PÚBLICO E PRIVADO - exemplo



ENDEREÇO IP PÚBLICO E PRIVADO - DIFERENÇAS

- IP Público
  - Significa que o endereço é identificado na internet
  - Endereço único em toda Internet
  - Pode ser facilmente geo-localizado
- IP Privado
  - Significa que o endereço só é válido na rede interna (casa ou escritório)
  - Endereço único na rede interna, mas duas empresas podem usar o mesmo endereço privado
  - Computadores acessam a Internet usando Internet Gateway
  - Somente um faixa de endereços podem ser usados como privado

ELASTIC IP

- Quando nós paramos uma Instância EC2, o endereço IP público pode mudar
- Se quisermos um endereço IP público fixo, precisamos de um Elastic IP
- Elastic IP é um endereço IPv4 público que você tem até que você delete
- Podemos anexar um Elastic IP a somente uma instancia EC2 de cada vez
- Podemos apontar o Elastic IP para outra instancia EC2 rapidamente
- Podemos ter até 5 Elastic IP em uma conta AWS. Podemos solicitar mais Elastic IP usando AWS ticket.

ELASTIC IP

- Tente não usar Elastic IP em produção (somente em desenvolvimento)
- Uso de Elastic IP demonstra uma arquitetura de baixa qualidade
- Use IP aleatório e registre um DNS
- Ou como veremos mais pra frente. Use Load Balancing. IP público não será necessário.

IP público x IP privado (IPv4) - AWS EC2 - Prática

- Por padrão, instâncias EC2 são criadas com:
  - IP privado para rede AWS interna
  - IP público, para rede web (Internet)
- Quando conectamos a EC2 usando SSH:
  - Não podemos usar o IP privado, pois não estamos na mesma rede
  - Temos que usar o IP público
- Se você parar sua instância EC2, seu IP público pode mudar

Instalar e iniciar servidor Apache na EC2

- Vamos usar nossa instancia EC2
- Vamos instalar o servidor Apache para exibir uma webpage
- Vamos criar um index.html para mostrar o hostname da nossa máquina

EC2 - User Data

- Podemos executar comandos automaticamente quando uma instância EC2 é criada usando EC2 User Data
- Esse script é executado somente uma vez quando a instância e iniciada pela primeira vez
- EC2 User Data é usado para automatizar o processo de boot tal como:
  - Instalar atualizações
  - Instalar aplicações
  - Baixar arquivos da Internet
  - Ou qualquer coisa que você quiser
- EC2 User Data é executada como usuário root

EC2 - User Data - Prática

- Vamos fazer com que nossa instância EC2 tenha o Apache instalado para mostrar uma simples web page
- Para isso nós vamos escrever um User Data Script
- O script vai ser executado na primeira vez que a instância EC2 for inicializada

EC2 - Tipos de Instâncias

- On Demand: Processos de curto prazo, paga somente pelo que usa
- Reserved: Processos de longo prazo (um ano ou mais)
- Convertible Reserved: Processo de longo prazo com flexibilidade de tipos
- Scheduled Reserved: Para processos com janela determinada
- Spot: Processos curtos, mais barato, pode ser terminada a qualquer momento
- Dedicated: Nenhum outro cliente vai compartilhar seu equipamento
- Dedicated Hosts: reserva o equipamento inteiro, com controle de Placement

EC2 - On Demand

- Pague somente pelo que usar (cobrado por segundo, depois do primeiro minuto)
- Tem o maior custo mas não é necessário pré-pagamento
- Sem contrato. Pode terminar a instância a qualquer momento
- Recomendado para aplicações de curto prazo e desenvolvimento

EC2 - Reserved

- Até 75% de desconto comparado com On Demand
- Pagamento adiantado
- Período de reserva: 1 ou 3 anos
- Reserva específica para determinado tipo de instância
- Recomendado para aplicações estáveis (ex: database)
- Convertible Reserved
  - Pode-se mudar o tipo de instância EC2
  - Até 54% de desconto
- Schedule Reserved
  - Funciona na janela de tempo determinada

EC2 - Spot

- Desconto de até 90% comparado com On-Demand
- Funciona tipo leilão. Você usa a sua instância até alguém pagar mais por ela.
- Preço varia de acordo com oferta e demanda
- Você recebe uma notificação de que alguém vai pagar mais pela instância e em 2 minutos sua instância será terminada
- Usada para batch jobs, Big Data analysis ou processos que aceitam interrupções
- Não deve ser usado para trabalhos críticos ou banco de dados

EC2 - Dedicated Hosts

- Equipamento dedicado para usar suas instancias EC2
- Total controle de onde colocar suas instâncias
- Acesso ao detalhes físico do equipamento
- Reservado por período de 3 anos
- Mais caro
- Usado para software que tem modelos de licensa complexos
- Ou empresas que tem regras rígidas sobre onde instalar seus servidores

EC2 - Dedicated

- Instâncias rodam em equipamento dedicado
- Pode compartilhar instâncias com outras que pertencem a mesma conta
- Sem controle sobre Placement (pode mover para outro hardware se for dado o comando Stop / Start

EC2 - Dedicated Instances vs Dedicate Hosts

Característica	Dedicated	Dedicated Hosts
Uso de equipamento dedicado	X	X
Cobrado por instância	X	
Cobrado por Hosting		X
Visibilidade de sockets, cores, host ID		X
Afinidade entre host e instância		X
Escolha do hardware onde lançar a instância		X
Escolha de Placement automática	X	X
Adicionar capacidade usando Allocation Request		X

EC2 - Qual tipo é o melhor?

- On Demand: Pode-se criar e terminar a qualquer momento e só é cobrado pelo tempo utilizado. Não tem desconto no preço.
- Reserved: Bom desconto se for feito um bom planejamento a longo prazo
- Spot: Leilão. Você pode usar sua instância até alguém pagar mais pelos recursos. Instância pode ser terminada a qualquer momento pela AWS
- Dedicated Hosts: Reserva do equipamento inteiro.

EC2 - Preço

- Preço de Instâncias (por hora) varia de acordo com os seguintes parametros:
  - Region
  - Tipo de Instância
  - On-Demand vs Spot vs Reserved vs Dedicated Host
  - Linux vs Windows vs Private OS (RHEL, WINDOWS SQL)
- Cobrado por segundo, mínimo de 60 segundos
- Cobrado também por armazenamento, transferência de dados, Endereço IP fixo, load balancing
- Não é cobrado quando a instância está parada

EC2 - Preço - Exemplo

- T2.small em sa-east-1 (São Paulo), custa $0.0372 por hora
- 10 segundos custa $0.0372/60 = 0.00062 (mínimo de 60 segundos)
- 60 segundos custa $0.0372/60 = 0.00062 (mínimo de 60 segundos)
- 30 minutos custa $0.0372*30/60 = $0.0186
- 1 mês custa $0.0372*24*30 = $26.784 (mês de 30 dias)
- Y segundos (Y > 60) custa $0.0372 * Y / 3600
- Para lista de preço atualizada, visite o link:
  - https://aws.amazon.com/ec2/pricing/on-demand/

O que é AMI (AMAZON MACHINE IMAGE)

- AWS tem imagens básicas como:
  - Ubuntu
  - Fedora
  - RedHat
  - Windows
  - Amazon Linux 2 …
- Essas imagens podem ser personalizadas usando EC2 User Data
- É possível criar sua própria imagem
- AMI é a imagem utilizada para criar instâncias EC2
- AMI pode ser montada para Linux ou Windows

Vantagens de utilizar AMI personalizado

- Por que usar AMI personalizado?
  - Instalação de pacotes necessários
  - Boot mais rápido
  - Máquinas configuradas com monitoramento
  - Segurança
  - Controle de manutenção e updates
  - Integração com Active Directory
  - Instalação de aplicação, para distribuição quando utilizado com auto-scaling
  - Usar AMI de outra pessoa otimizada para Banco de dados ou apps
  - AMI é montada para uma região

Visão Geral - Instâncias

- Instâncias tem 5 características:
  - CPU (tipo, fabricante, frequencia, número de núcleo, geração)
  - RAM (tipo, quantidade, geração
  - I/O (performance do disco, Otimização EBS)
  - Rede (Bandwidth e latencia)
- AWS oferece mais de 50 tipos de instâncias: https://aws.amazon.com/ec2/instance-types/
- R/C/P/G/H/I/F/CR especializado em RAM, CPU, I/O, Rede, GPU
- M é balanceado
- T2/T3 para burstable (Nitro)

Instancias Brustable (T2)

- AWS trabalha com o conceito de Burstable.
  - CPU tem desempenho normal
  - Quando a instância precisa de mais processamento, ela pode Brust sua capacidade para atender a demanda de processamento, quando isso acontece créditos brust começam a ser utilizados
  - Quando os créritos acabam, o CPU tem um desempenho muito ruim
  - Quando a carga de processamento volta ao normal, a instância começa a acumular créditos novamente

Instancias Brustable (T2)

- Instancias Brustable são ótimas para tratar cargas inesperadas
- Se a instância fica muito tempo sem créditos, é hora de mover para uma instância com mais poder de processamento

Instancias Brustable (T2)

Instance type	CPU credits earned per hour	Maximum earned credits that can be accrued*	vCPUs	Baseline performance per vCPU
t1.micro	6	144	1	10%
t2.nano	3	72	1	5%
t2.micro	6	144	1	10%
t2.small	12	288	1	20%
t2.medium	24	576	2	20%**
t2.large	36	864	2	30%**
t2.xlarge	54	1296	4	22.5%**
t2.2xlarge	81.6	1958.4	8	17%**
t3.nano	6	144	2	5%**
t3.micro	12	288	2	10%**
t3.small	24	576	2	20%**
t3.medium	24	576	2	20%**
t3.large	36	864	2	30%**

T2 unlimited

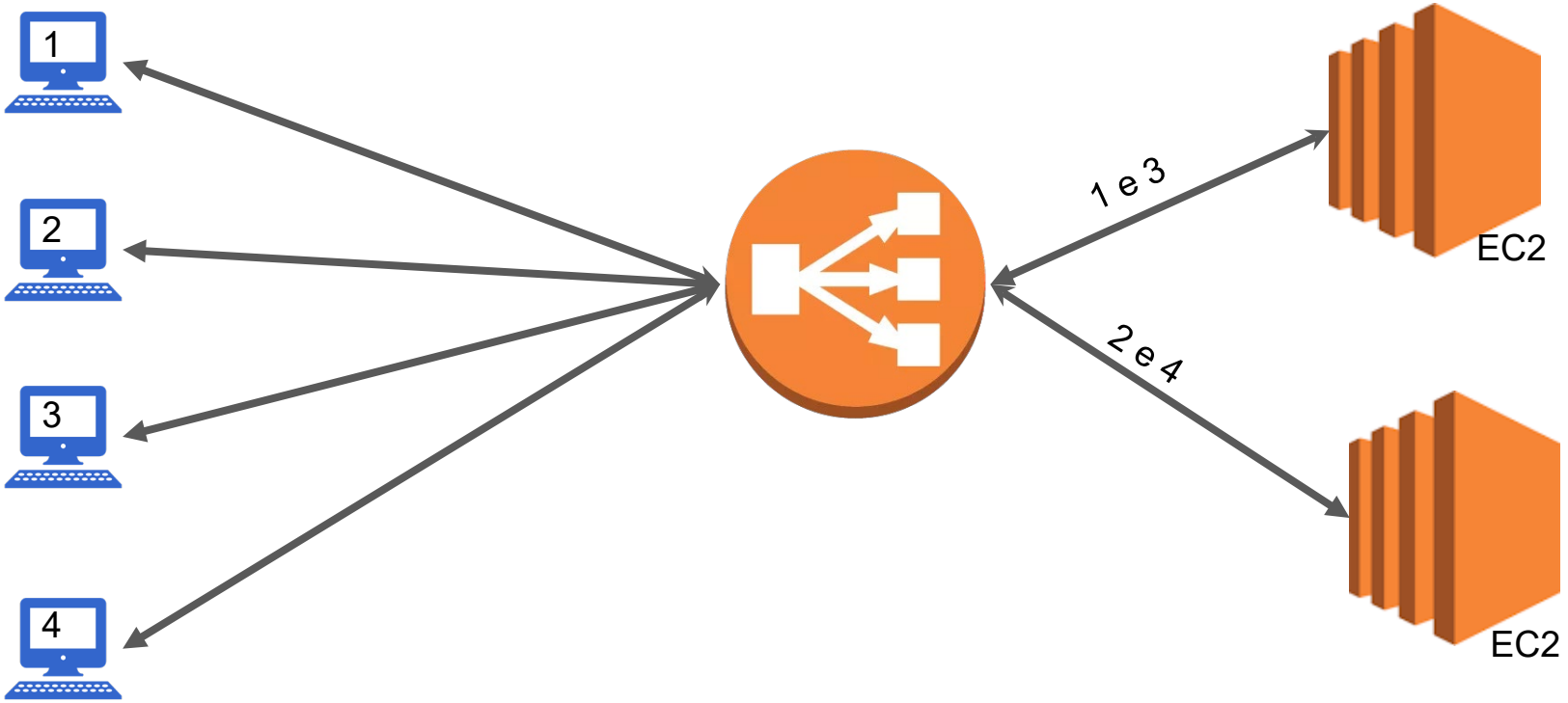
- É possível ter créditos brust ilimitados
- É cobrado extra se você utilizar todos os créditos porém sua instancia não perde performance
- TENHA CUIDADO. Pode custar caro se você não monitorar a utilização da CPU

EC2 - RESUMO

- Saber como usar SSH e mudar a permissão do arquivo .pem
- Saber como usar Security Groups
- Saber a diferença entre endereço IP público, privado e Elastic IP
- Saber usar User Data para Personalizar sua instância
- Saber que você pode criar AMI personalizada
- Instâncias EC2 são cobradas por segundo e podem ser criadas e deletada com alguns cliques

LOAD BALANCING

- Load Balancer são servidores que distribuem o tráfego entre as instâncias EC2



Load Balancer

- Distribuí o tráfego entre as instâncias
- Somente o DNS do Load Balancer é visto pelos usuários
- Para de enviar tráfego automaticamente para instâncias que apresenta problemas
- Verificar regularmente a saúde das instâncias
- Oferece SSL (HTTPS) para a sua aplicação
- Opção STICKNESS com cookies
- Alta disponibilidade entre availability zones
- Separa tráfego público de tráfego privado

Vantagens do uso de EC2 Load Balancer

- ELB (EC2 Load balancing) é gerenciado pelo AWS
  - AWS garante que vai estar sempre funcionando
  - AWS faz upgrades, manutenção, disponibilidade
- É mais barato criar seu próprio load balancer mas no final você vai ter muito mais trabalho para manter seu próprio Load Balancer
- É integrado com vários outros serviços AWS

Tipos de Load Balancer

- AWS oferece 3 tipos de Load Balancer
  - Classic Load Balancer: 2009 (Obsoleto: Não utilizar para novas aplicações)
  - Application Load Balancer: 2016
  - Network Load Balancer: 2017
- É possível configurar Load Balancer para rede pública(externa) ou privada(interna)

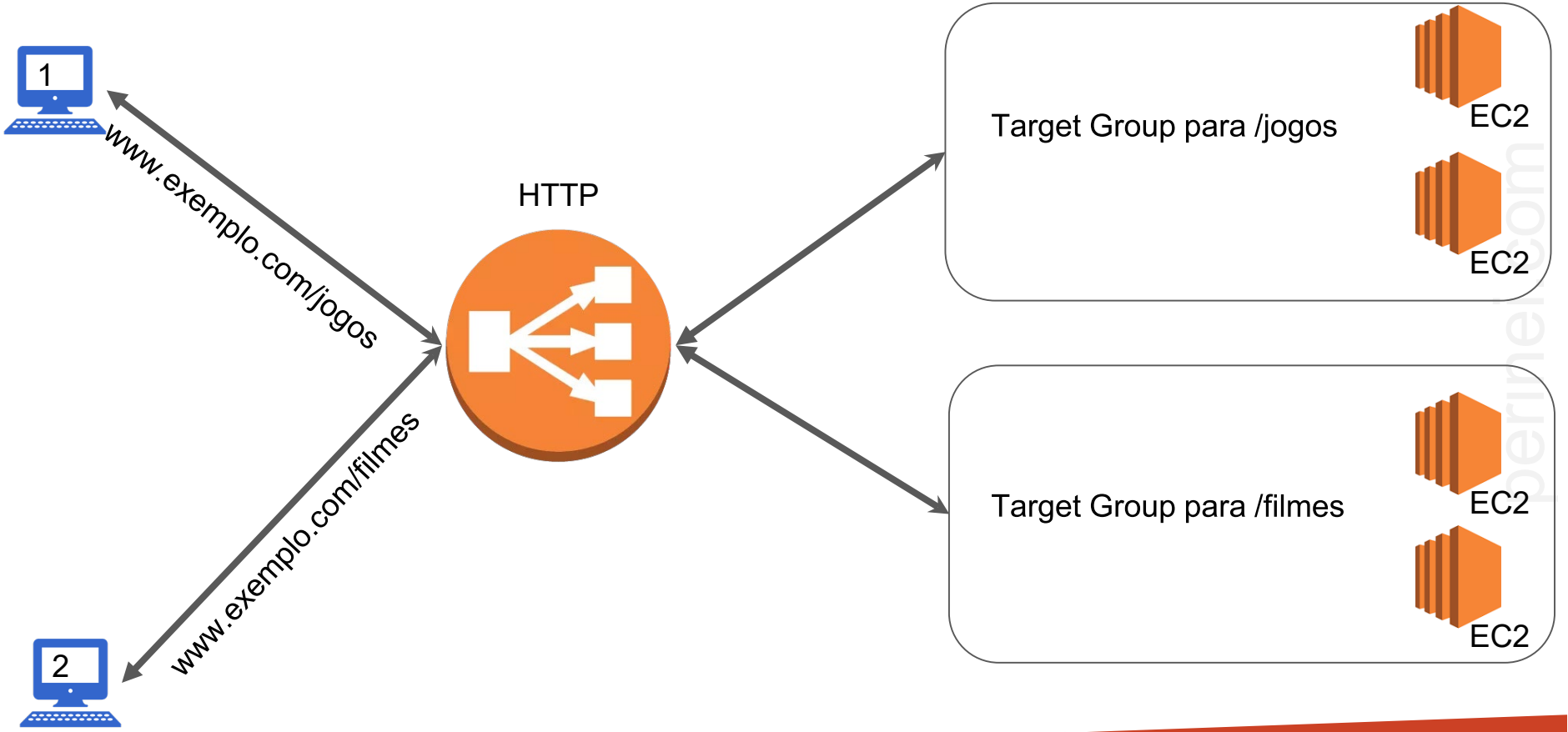
Load Balancer - Health check

- Health Check é crucial para o funcionamento do Load Balancer
- Health Check informa ao Load Balancer se a instância é capaz de responder a requisição.
- Health Check é executado em porta e caminho específico. Exemplo: /healthy
- Se a resposta não for 200 (OK) a instância é considerada deficiente e Load Balancer pára de enviar tráfego para essa instância

Application Load Balancer

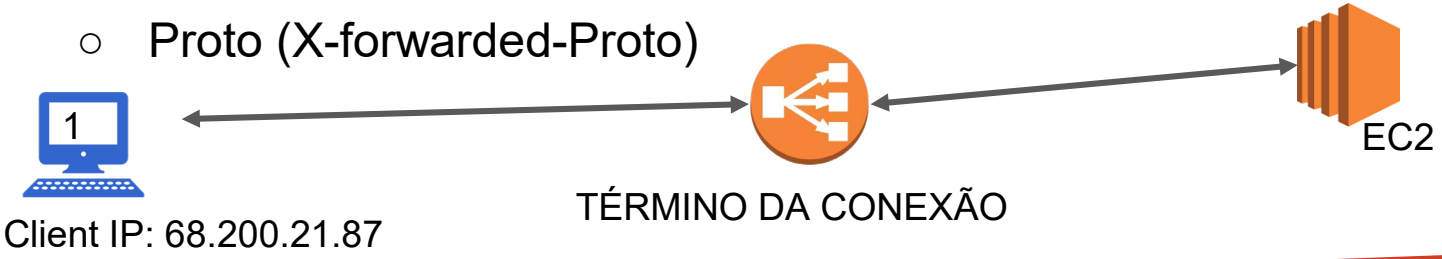
- Application Load Balancer (Layer 7) permite:
  - Load Balancing múltiplas aplicações HTTP em máquinas diferentes (Target Groups)
  - Load Balancing múltiplas aplicações HTTP na mesma máquina (containers)
  - Load Balancing baseado em rotas (URL)
  - Load Balancing baseado em hostname (URL)
- Application Load Balancer é excelente solução para aplicações rodando em containers
- Recurso Port Mapping para redirecionar para porta dinâmica

Application Load Balancer



Application Load Balancer

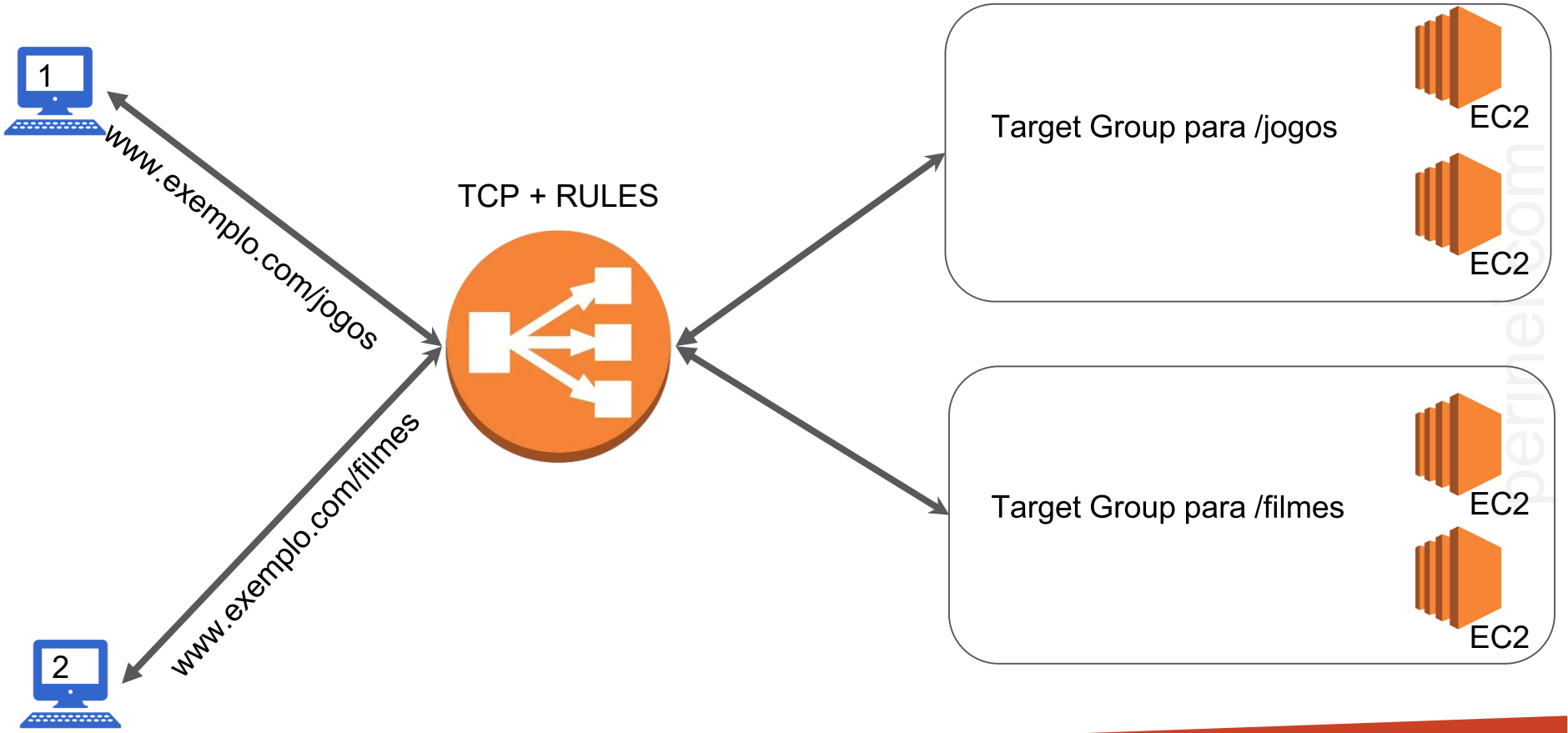
- Stickness pode ser habilitado no Target Group
  - Mesma requisição é direcionada para mesma instância
  - Stickness é gerenciado pelo Load Balancer, não pela aplicação
- ALB suporta HTTP/HTTPS and protocolo websockets
- A aplicação não vê o endereço IP da máquina cliente
  - O endereço IP do cliente é inserido no header X-Forwarded-for
  - Também é possível receber Port (X-Forwarded-Port) e
  - Proto (X-forwarded-Proto)



Network Load Balancer

- Network Load Balancer (Layer 4) permite:
  - Direciona tráfego TCP para instâncias
  - Capacidade para milhões de requisições por segundo
  - Suporta IP estático ou Elastic IP
  - Menor latência: ~100ms (vs 400ms for ALB)
- Network Load Balancer é usado em aplicações com alto volume de tráfego.
- Application Load Balancer é suficiente para a maioria das aplicações
- O processo de criação é o mesmo para Application e Network Load Balancing

Network Load Balancer



Load Balancer

- Classic Load Balancer: Não usar. Ultrapassado e limitado
- Application Load Balancer para HTTP/HTTPS e websockets
- Network Load Balancer para TCP
- CLB e ALB suportam certificados SSL
- Todos os Load Balancer são capazes de realizar healthy checks
- ALB pode rotear baseado em path e hostname
- ALB é excelente para ser usado com ECS(docker)

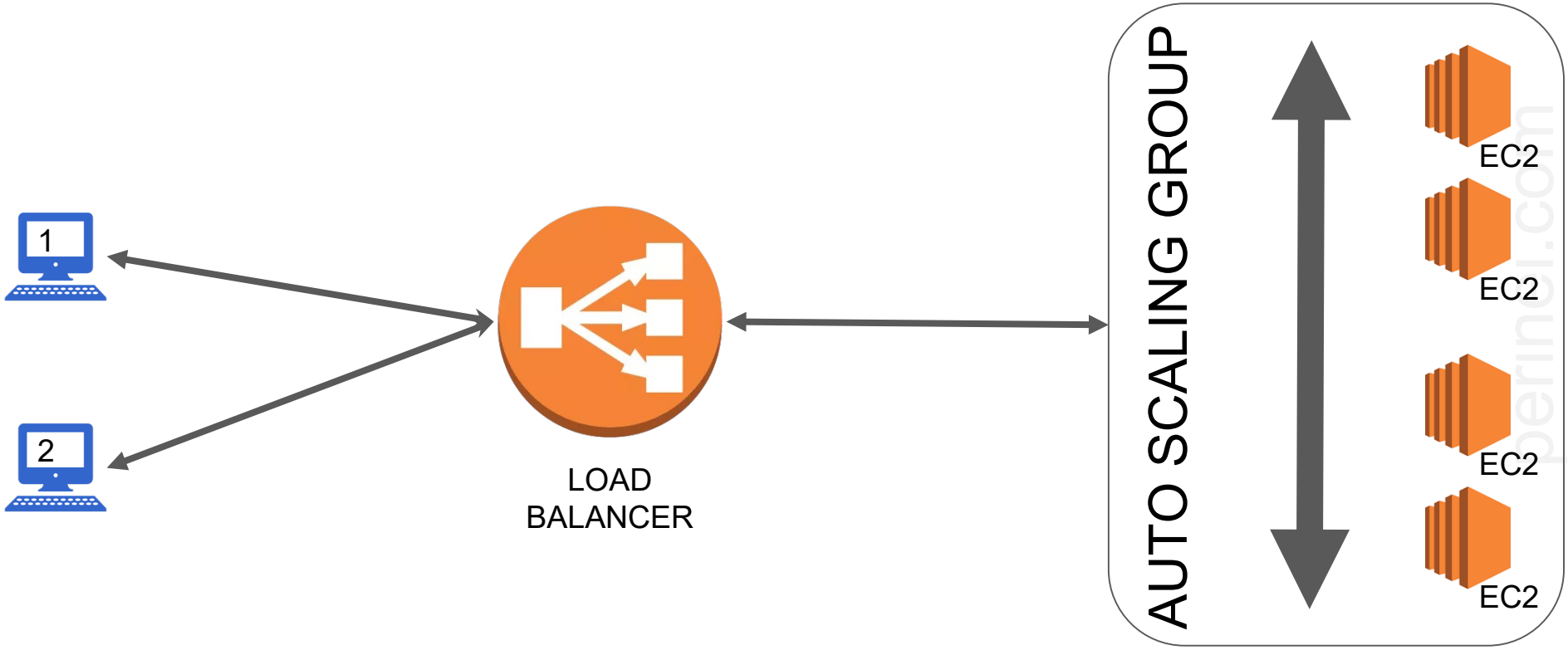
Load Balancer

- Todos os 3 Load Balancer (ALB, CLB and NLB) tem DNS. Não use endereço IP
- Load Balancer é escalável, mas não é muito rápido. Entre em contato com AWS para warm-up
- NLB (Network Load Balancer) pode ver IP dos clientes
- 4xx erros são gerados pelo cliente
- 5xx erros são gerados pela aplicação
    - Erro 503 significa: capacidade máxima atingida ou target não definido
- Se o Load Balancer não pode conectar com sua aplicação, verifique o Security Group

Auto Scaling Group

- Tráfego de um website ou aplicação variam
- Usando AWS é possível criar e destruir servidores rapidamente conforme a necessidade
- O objetivo do Auto Scaling Group é:
  - Scale out (criar instâncias EC2) para atender aumento da demanda
  - Scale in (terminar instâncias EC2) quando a carga diminuir
  - Garante que tenhamos um número mínimo e máximo de instâncias rodando
  - Automaticamente registra novas instâncias no Load Balancer

AUTO SCALING

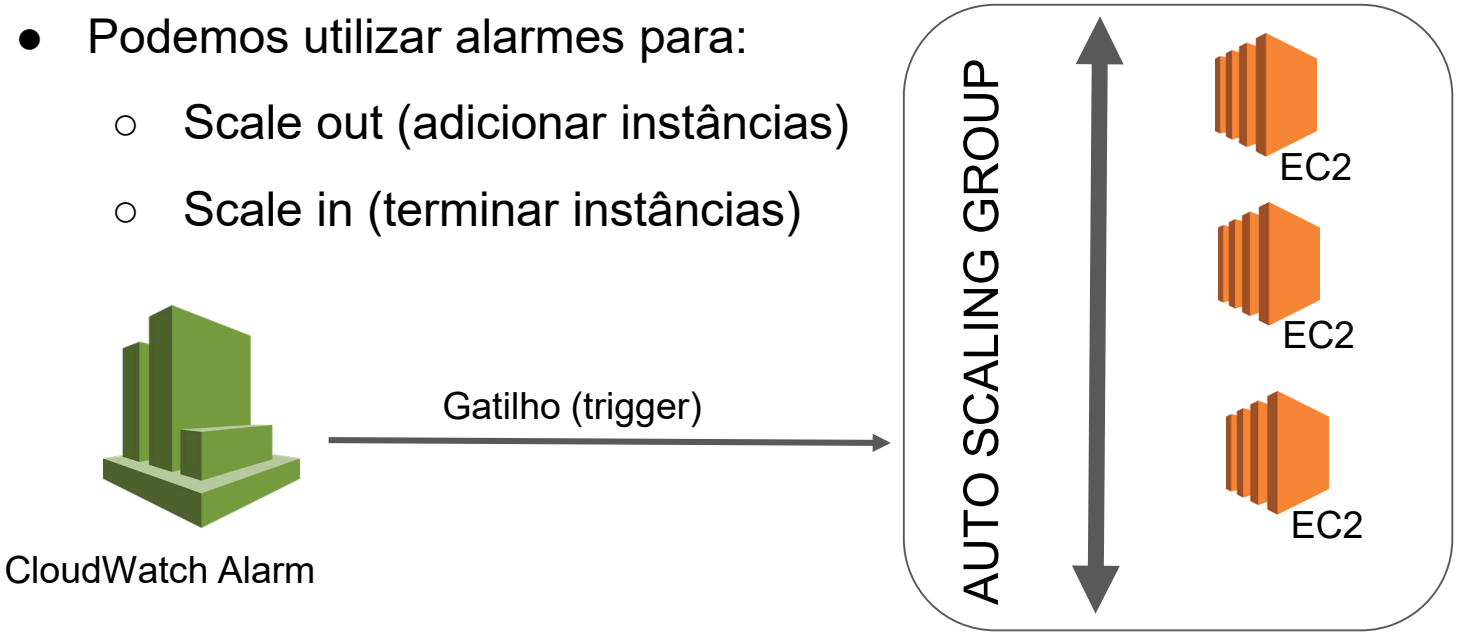


Auto Scaling Group: Atributos

- Launch Configuration
  - AMI + tipo de instância
  - EC2 USER DATA
  - EBS VOLUMES
  - SECURITY GROUPS
  - SSH KEY PAIR
- Tamanho mínimo / máximo e tamanho desejável
- Informação sobre Network e subnets
- Informação sobre Load Balancer
- Regras de escalonamento

Auto Scaling Alarms

- É possível escalonar um ASG baseado em alarmes do CloudWatch
- Alarmes monitoram um Metric (exemplo: uso da CPU)
- Podemos utilizar alarmes para:
  - Scale out (adicionar instâncias)
  - Scale in (terminar instâncias)



Auto Scaling: Novas Regras

- É possível definir regras gerenciadas diretamente por Instâncias EC2
  - Média de uso do CPU
  - Número de requisições ao LoadBalancer por Instância
  - Média de tráfego entrando
  - Média de tráfego saindo
- Essas regras fazem mais sentido e são mais fáceis de usar

Auto Scaling Custom Metric (personalizado)

- Podemos escalonar baseado em Custom Metric. Ex. no número de usuários conectados.
  1. Enviar Custom Metric da aplicação EC2 para CloudWatch usando PutMetric API
  2. Criar um Alarme CloudWatch para monitorar valores altos e baixos
  3. Usar Alarme CloudWatch como Scaling Policy para ASG

Auto Scaling: Resumo

- Auto Scaling pode ser acionado por uso de CPU, network e etc, ou pode ser acionado por Custom Metrics e também por agendamento.
- ASG usa Launch Configuration e podemos atualizar criando um novo Launch Configuration
- IAM Roles anexados ao ASG será atribuído a Instância EC2
- ASG é grátis. É cobrado apenas os recursos lançados pelo ASG
- Quando Instâncias EC2 que foram inicializadas pelo ASG apresenta problema, ASG inicializará outra para substituir.
- ASG pode terminar instâncias identificadas com não-saudáveis pelo Load Balancer

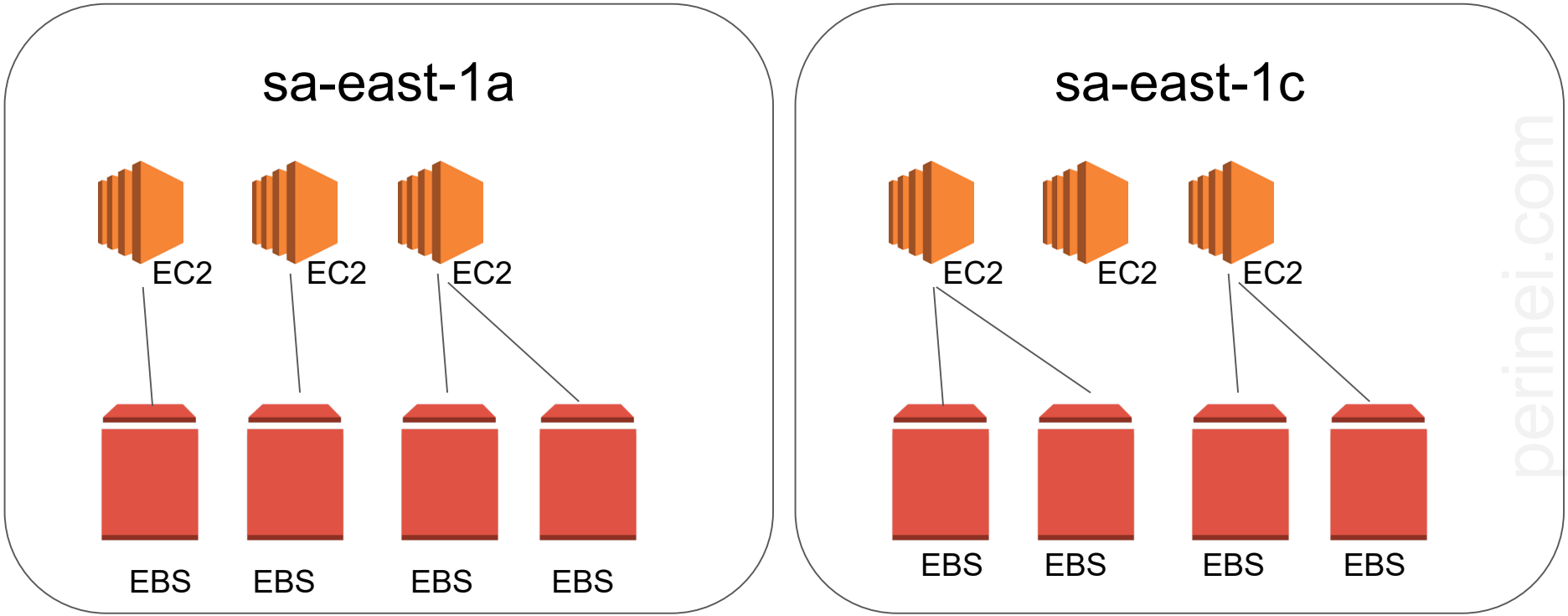
VOLUME EBS (Elastic Block Storage)

- Instância EC2 perde seu volume root quando são terminadas
- Instância EC2 podem terminar inesperadamente. AWS enviará um email avisando do ocorrido.
- Pode ser que você precise guardar os dados da Instância EC2
- EBS (Elastic Block Storage) é um drive de rede que pode ser anexado a instância
- Ele permite que os dados não se percam.

VOLUME EBS

- EBS é um drive de rede (não um drive físico)
  - EBS uma a rede AWS para acessar os dados. Isso significa que pode ser um pouco mais lento que um drive local
  - EBS pode ser desanexado de um EC2 e anexado a outro EC2
- EBS pertence a um Availability Zone
  - Um EBS criado em um Availability Zone não pode ser associado a um EC2 em outra Availability Zone
- Pode-se definir a capacidade (tamanho em GB e IOPS)
  - É cobrado pela capacidade escolhida
  - É possível aumenta a capacidade de um EBS depois de criado

VOLUME EBS



TIPOS DE VOLUME EBS

- EXISTEM 4 TIPOS DE VOLUME EBS
  - GP2 (SSD): General Purpose SSD. Meio termo entre preço em performance.

    Usado para vários tipos de aplicação
  - IOI (SSD): Alta performance, baixa latência e alto custo
  - STI (HDD): Baixo custo. Acesso frequente
  - SCI (HDD): Mais barato de todos. Acesso não frequente
- VOLUME EBS é definido por tamanho, Throughput, IOPS
- Nós só usamos GP2 até agora

MUDANDO O TAMANHO DE VOLUME EBS

- É possível somente aumentar o tamanho de um volume EBS
- Depois de aumentar o tamanho do volume é preciso reparticionar o volume

EBS Snapshot

- É possível fazer backup de Volume EBS usando Snapshot
- Snapshot só faz o backup do espaço utilizado. Se você fizer um Snapshot de um volume de 20GB que só tem 8GB de dados, O Snapshot terá 8GB
- Snapshot é usado para:
  - Salvar seus dados em caso de catástrofe
  - Migração:
    - Para um volume menor
    - Mudar o tipo de volume
    - Criptografar um volume

CRIPTOGRAFIA EBS

- Quando um Volume EBS Criptografado é criado:
  - Os dados são criptografados em repouso dentro do volume
  - Os dados em movimento entre o EC2 e o EBS são criptografados
  - Todos os Snapshots são criptografados
  - Todos os volumes criados do Snapshot são criptografados
- O processo é todo feito pela AWS. É totalmente transparente para o usuário
- Criptografia tem impacto mínimo na latência
- Criptografia EBS usa KMS (AES-256)
- Copiar um Snapshot permite que a cópia seja criptografada

EBS vs Instance Store

- Algumas tipos de EC2 não tem EBS como root volume
- Ao invés disso EC2 tem “Instance Store”
- Instance Store é fisicamente conectado a instância
- Vantagens:
  - Melhor performance
- Desvantagem:
  - Quando a instância é terminada, todos os dados são perdidos
  - Não é possível modificar o tamanho
  - Backup deve ser feito pelo usuário
- De uma forma geral: EBS deve ser adotado para a maioria das aplicações

EBS Resumo

- Volume EBS pode ser anexado somente a uma instância EC2 de cada vez
- EBS é limitada a Availability Zone onde foi criado
- Para migrar EBS de uma Availability Zone para outra, é preciso fazer um Snapshot do volume EBS e restaurar o Snapshot em outra Availability Zone
- Backup EBS usa parte do IO que você provisionou. Backups não devem ser feitos durante períodos de tráfego intenso, pois podem afetar a performance.
- Volume EBS root é terminado quando EC2 é terminado. É possível desabilitar essa função

Fundamentos AWS - Parte 3

- Route 53
- RDS
- ElastiCache
- VPC

AWS - ROUTE 53

- Route 53 é um DNS (Domain Name System) gerenciado pela AWS
- DNS é uma coleção de Registros (Records) e Regras (Rules) que auxiliam o cliente a encontrar o servidor desejado utilizando uma URL (Universal Resource Locator)
- Na AWS os Rocords mais comuns são:
  - A: URL para IPv4
  - AAAA: URL para IPv6
  - CNAME: URL para URL
  - Alias: URL para Recursos AWS

ROUTE 53 - Diagrama para A record

Request: http://www.exemplo.com

Return: IP: 68.10.234.7

Route 53

Request: http://www.exemplo.com
P: 68.10.234.7

HTTP Response: 200 OK

WebServer
68.10.234.7

AWS - ROUTE 53

- AWS Route 53 pode usar:
  - Pulic Domain Name que pertence a você
    - Meuendereçopublico.com
  - Private Domain Name que podem ser resolvidos pelas suas instâncias EC2 dentro da sua VPC.
    - App.empresa.internal
- Route 53 tem função avançada, por exemplo:
  - LoadBalancing (usando DNS - também chamado de Client Load Balancing)
  - Routing Policy:simple, failover, geolocation, geoproximity, latency and weighted
- Use Alias para Recursos AWS ao invés de CNAME (para melhor performance)

ROUTE 53

- Route 53 não é muito explorado na Certificação Desenvolvedor. Você precisa saber
  - A: URL para IPv4
  - AAAA: URL para IPv6
  - CNAME: URL para URL
  - Alias: URL para Recursos AWS
  - Use Alias para recursos AWS ao invés de CNAME

RDS - RELATION DATABASE SERVICE

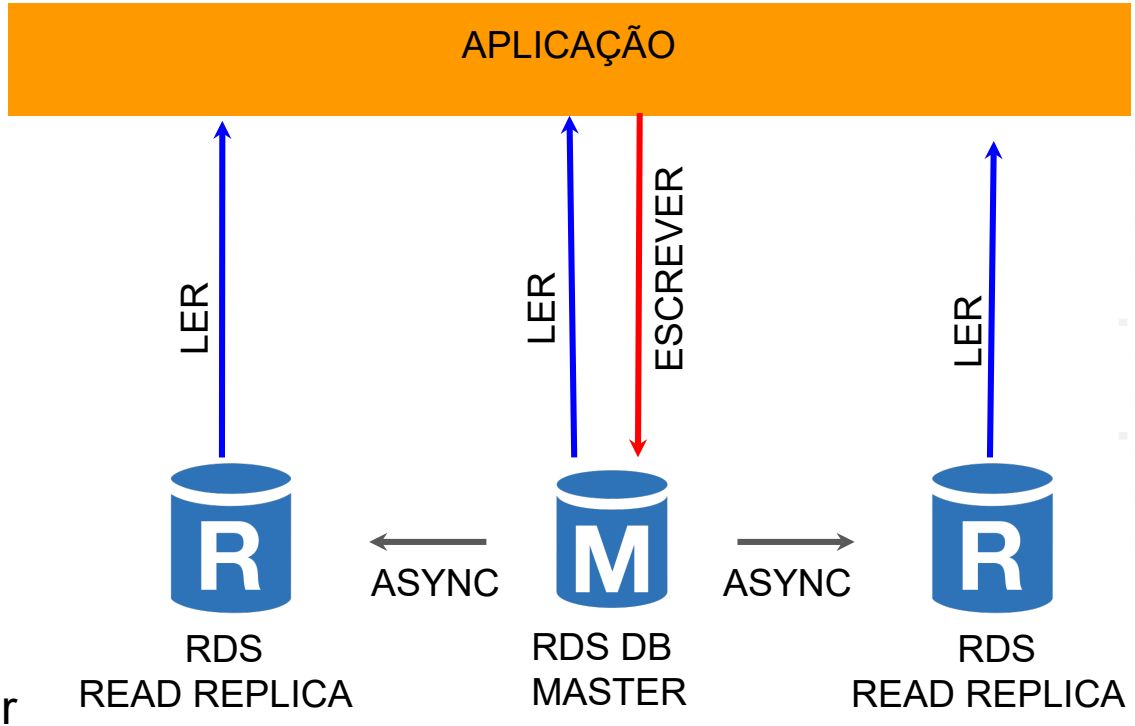
- RDS - Serviço de Banco de Dados Relacional
- RDS é um serviço gerenciado pela AWS para banco de dados que utilizam SQL para queries.
- RDS suporta os seguintes Banco de Dados:
  - Postgress
  - Oracle
  - MySQL
  - MariaDB
  - Microsoft SQL Server
  - Aurora (Banco de Dados desenvolvido pela AWS)

VANTAGEM DE USAR RDS vs EC2 e Banco de Dados

- Gerenciado pela AWS
- Patching é responsabilidade da AWS
- Backup contínuo. Banco de Dados pode ser restaurado para um timestamp
- Painel de Controle para monitoramento
- Read Replicas para otimizar leituras
- Opção de configuração Multi Availability Zone para recuperação após disastres
- Janela de manutenção para upgrades
- Escalonamento (vertical e horizontal)
- Não é possível SSH no RDS

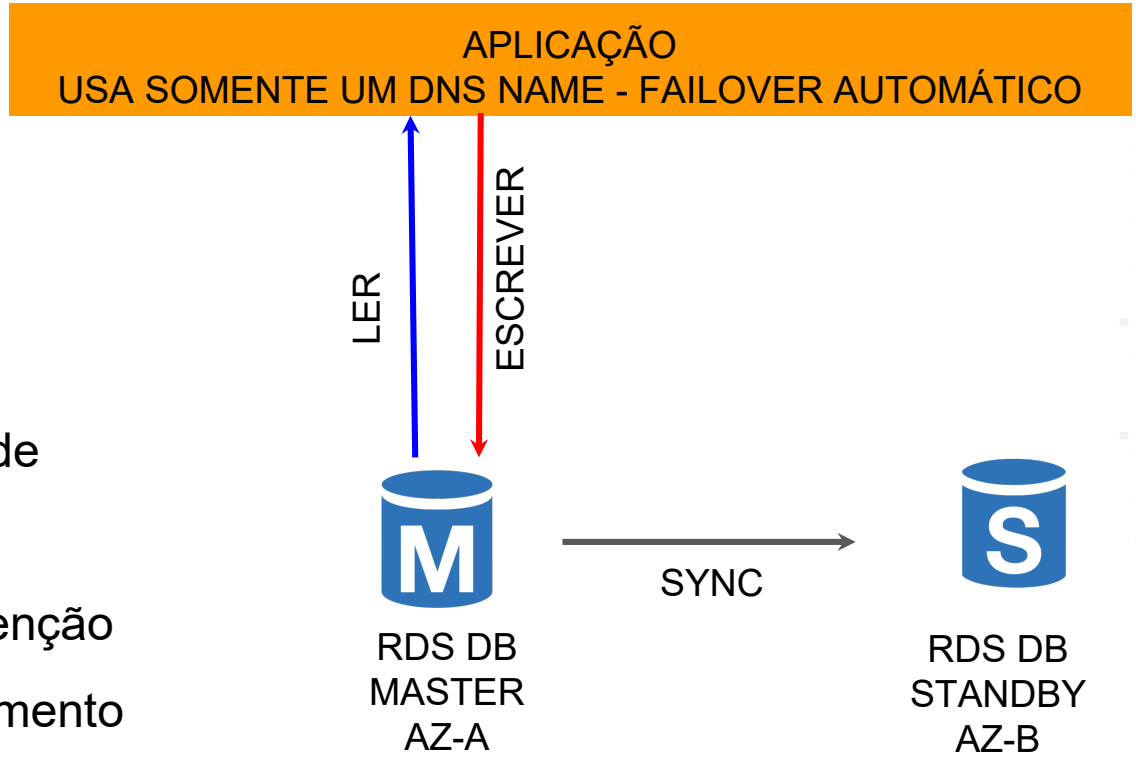
RDS - Read Replica

- Até 5 Read Replicas
- Mesma AZ
- Diferente AZ
- Diferente Region
- Réplica is ASYNC (eventually consistent)
- Réplica pode virar Master
- Aplicações devem atualizar String de Conexão para usar Read Replica



RDS - Multi Availability Zone

- Réplica é SYNC
- Um único DNS name.
- Automaticamente failover To Stand-by
- Aumenta disponibilidade
- Failover em caso de perda de AZ e rede
- Sem necessidade de intervenção
- Não é usado para escalonamento



RDS BACKUP

- Backup é habilitado automaticamente on RDS
- Backup automático
  - Snapshot diário completo do banco de dados
  - Registra log de todas as transações em tempo real
  - Possível recuperar o Banco de Dados em qualquer timestamp
  - 7 dias de backup (pode ser aumentado até 35 dias)
- Snapshot do Banco de Dados
  - Iniciado pelo usuário
  - Snapshot pode ser guardado por quanto tempo o usuário quiser

RDS e CRIPTOGRAFIA

- Criptografia em repouso com AWS KMS - AES-256
- Certificado SSL para criptografar banco de dados em trânsito
- Para forçar SSL:
  - PostgreSQL: rds.force_ssl=1 no console RDS (Parameter Group)
  - MySQL de dentro do Banco de Dados entre:
    - GRANT USAGE ON *.* TO ‘mysqluser’@’%’ REQUIRE SSL;
- Para conectar usando SSL:
  - Entre com o SSL trust certificate (pode ser feito o download no AWS)
  - Entre com a opção SSL quando conectar ao Banco de Dados

SEGURANÇA no RDS

- Banco de Dados RDS normalmente é criado em uma rede privada e não publica
- RDS usa security groups da mesma forma que EC2 para definir regras de acesso
- IAM Policies define quem pode gerenciar dados no RDS
- Usuário e Senha podem ser usados para logar no Banco de Dados
- IAM users também pode ser usados para autenticar (MySQL e Aurora)

RDS vs Aurora

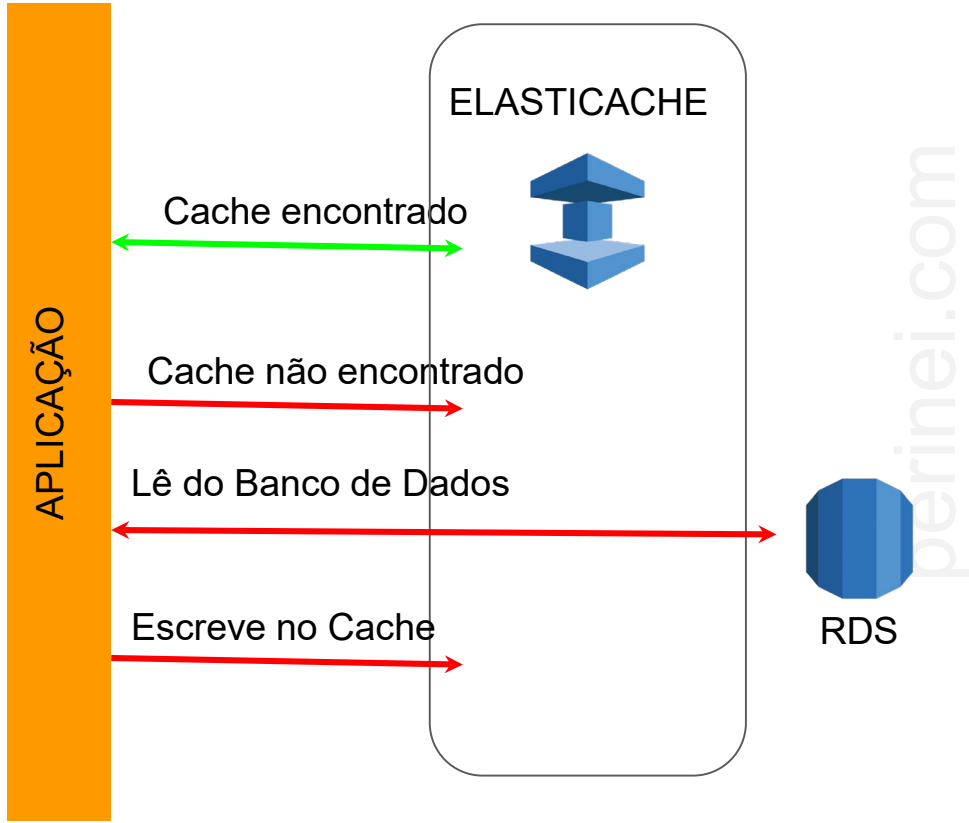
- Aurora é um Banco de Dados proprietário da AWS (não é opensource)
- Portgres e MySQL são suportados pelo Aurora (isso significa que os drivers Postgres and MySQL como se Aurora fosse Postgres ou MySQL)
- Aurora foi desenvolvido para as nuvens e AWS diz que é 5 vezes mais rápido que MySQL RDS e 3 vezes mais rápido que Postgres RDS
- Armazenamento Aurora cresce automaticamente em incrementos de 10GB, até 64TB
- Aurora pode ter até 15 Read Réplicas (sub 10ms) enquanto MySQL pode ter até 5
- Failover no Aurora é instantâneo. Usa HA nativo
- Aurora custa 20% mais que RDS, porém é mais eficiente

AWS ElastiCache

- ElastiCache é similar ao RDS. Oferece Banco de Dados gerenciados pela AWS
- Duas opções disponíveis: Redis e Memcached
- Caches são Banco de Dados na memória RAM, alta performance, baixa latência
- Ajuda a reduzir a carga no Banco de Dados principal para leitura intensa
- Ajuda a fazer sua aplicação Stateless
- Escrita escalona usando Sharding
- Leitura escalona usando Read Replica
- Multi AZ com Capacidade Failover
- AWS cuida da manutenção, patching, otimização, setup, configuração, monitoramento, recuperação de falha e backups

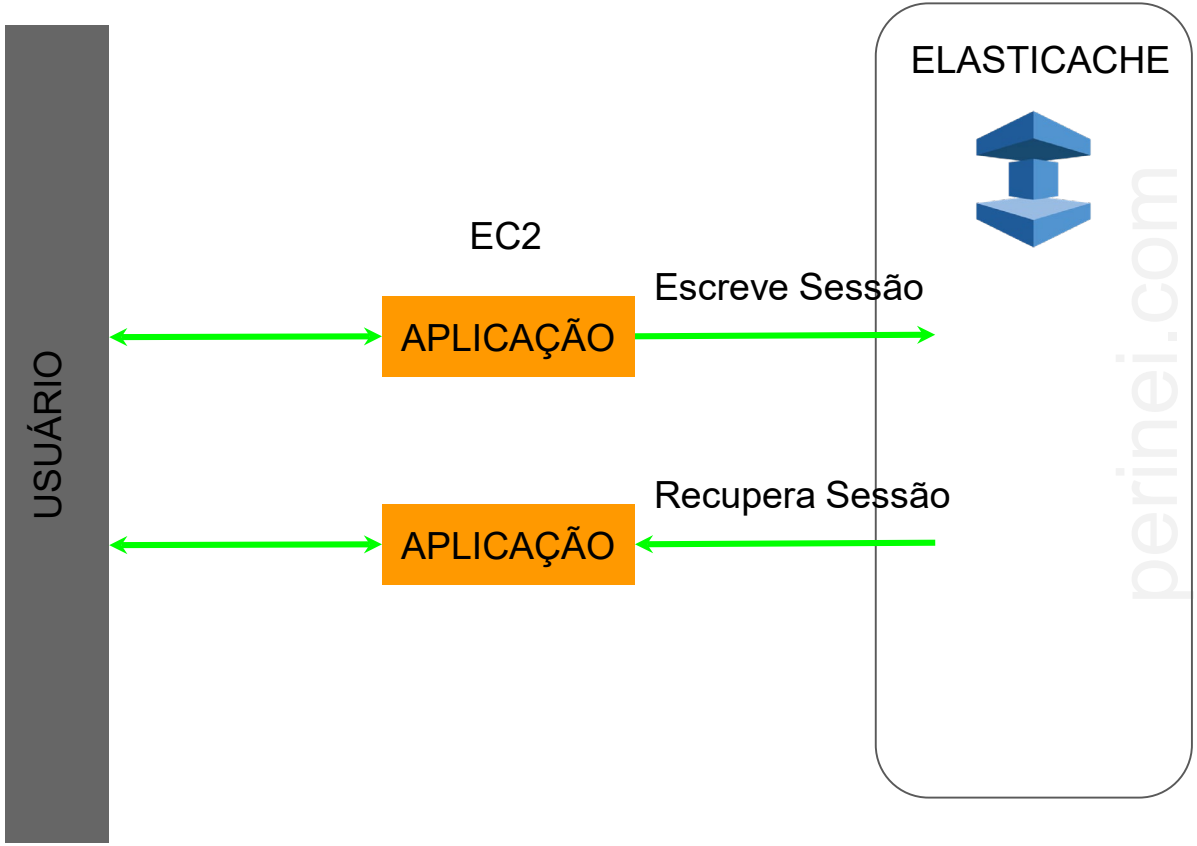
AWS ElastiCache

- A aplicação faz o query para o ElastiCache, se não estiver disponível, faz o query no Banco de Dados RDS e armazena no ElastiCache
- Ajuda a aliviar a carga no RDS
- Cache deve ter uma regra de validação para garantir que somente os dados mais recentes são utilizados



AWS ElastiCache - Guardar Sessão de Usuário

- Usuário loga na aplicação
- Aplicação escreve os Dados da sessão no ElastiCache
- Usuário acessa a Aplicação por uma Outra instância
- Instância lê os dados Do cache e verifica que o Usuário já está logado



REDIS

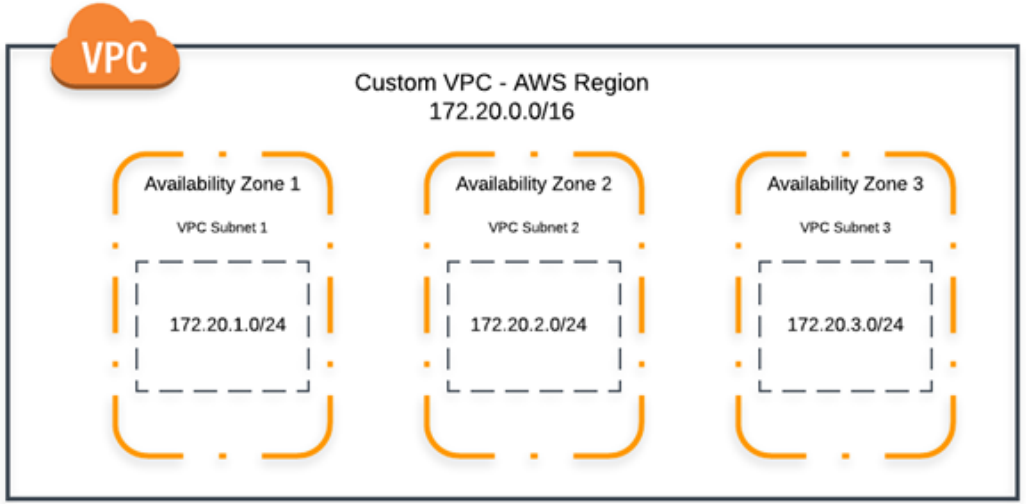
- Redis é um Banco de Dados do tipo Key-Value que armazena os dados na RAM
- Latência muito baixa (sub ms)
- Cache não se perde quando instância é reiniciada (persistente)
- Ideal para:
  - Sessão do Usuário
  - Alivia carga no Banco de Dados (ex. RDS)
  - Leaderboard (Placar para jogos)
- Multi AZ com failover automático para recuperação de desastres se você não quiser perder seus dados.
- Suporta Read Replicas

MEMCACHED

- Memcached armazena objetos na memória RAM
- Cache perde os dados se for reiniciado
- Leitura rápida dos objetos armazenados
- AWS não vai perguntar qual é o melhor

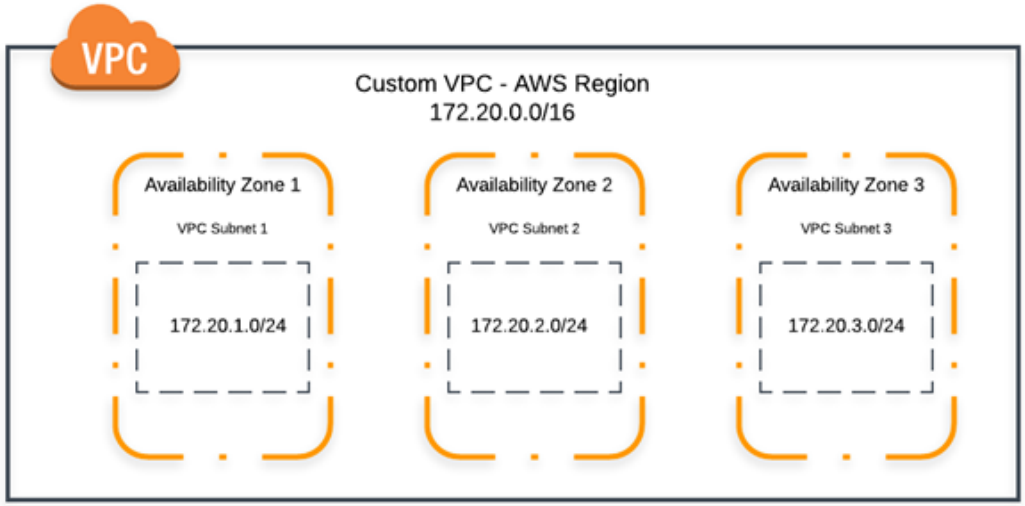
AWS VPC (Virtual Private Cloud)

- VPC é criado dentro de uma Region
- VPC contém subnets (redes)
- Cada subnet é associada a um AZ
- Subnet pode ser pública
- Subnet pode ser privada
- Pode-se ter várias subnets por AZ



AWS VPC (Virtual Private Cloud)

- Public Subnets podem conter:
  - Load Balancer
  - Static Websites
  - Files
  - Public Authentication Layers
- Private Subnets podem conter:
  - Servidores Web
  - Banco de dados



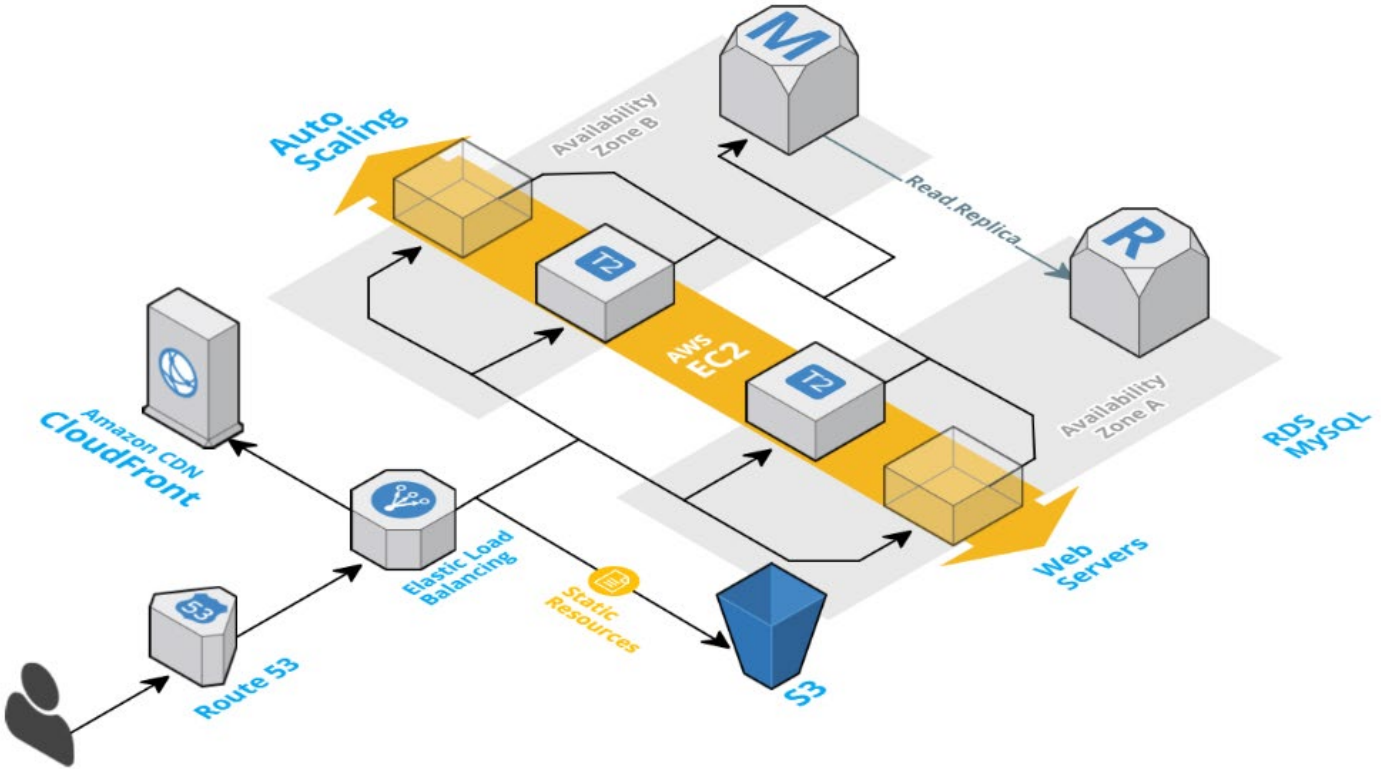
- Public e Private subnets podem comunicar-se, se estiverem no mesmo VPC

AWS VPC (Virtual Private Cloud)

- Todas as novas contas AWS tem um Default VPC
- É possível usar VPN para conectar com VPC

  (e acessar todos os endereços privados)
- VPC Flow Logs permite monitorar tráfego que entra e sai do VPC
- VPC pertence a uma conta e um Region
- Alguns recursos AWS podem ser criados dentro de um VPC, outros não
- É possível conectar VPCs usando Peer VPC (Para mesma conta ou outra conta)
- VPC não é muito cobrado na certificação desenvolvedor

Arquitetura de 3 níveis



S3 (SIMPLE STORAGE SERVICE)

- AWS S3 é um dos principais serviços
- Capacidade “infinita”
- Muitos websites usam S3 como backbone
- Muitos serviços AWS usam S3
- Vamos mergulhar fundo no S3

S3 - Visão Geral

- S3 permite armazenar objetos (arquivos) in “buckets” (diretório/pasta)
- Buckets precisam ter nome globalmente único
- Buckets são definidos a nível de Region
- Como nomear um Bucket
  - Não é permitido letra maiúscula
  - Não é permitido underscore “_”
  - Tamanho de 3 a 63 caracteres
  - Deve começar com letra minúscula ou número

S3 - Visão Geral - Objetos

- Objetos (files) tem Key. Key é o caminho completo
  - <meu_bucket>/meu_arquivo.txt
  - <meu_bucket>/minha_pasta/outra_pasta/meu_arquivo.txt
- Não existe o conceito de pasta dentro do bucket, mas a interface gráfica pode enganar você
- Característica dos objetos:
  - Tamanho máximo é 5TB
  - Se for fazer upload de arquivo maior que 5GB deve usar “multi-part upload”
- Metadata (text key / value pair - system ou user metadata)
- Tags (Unicode key / value pair - up to 10) importante para securança
- Version ID

S3 - Versioning

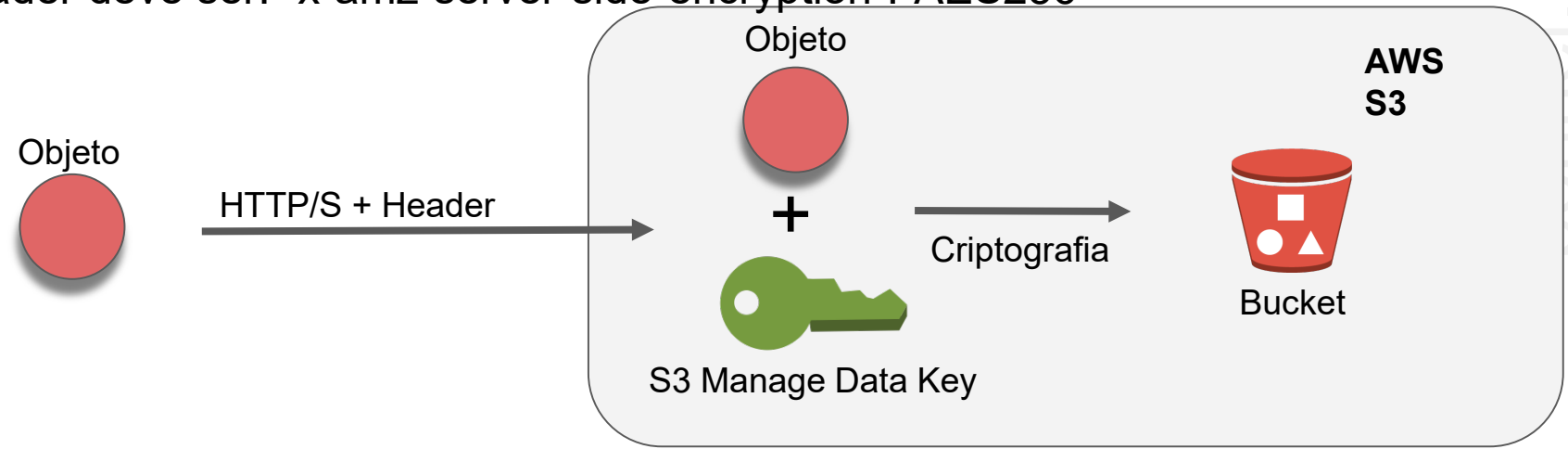
- É possível habilitar version para arquivos no S3
- É habilitado no nível Bucket
- Version: 1, 2, 3…
- Vantagens de utilizar version
  - Protege contra apagamento acidental
  - Fácil de restaurar versões anteriores
- Se version não estiver habilitado o arquivo terá version = “null”

S3 - Criptografia

- Existem 4 maneiras de criptografar objetos no S3
  - SSE-S3: Criptografa objetos usando Key gerenciada pela AWS
  - SSE-KMS: Key gerenciada pela Key Manager Service
  - SSE-C: Usuário gerencia a própria chave (key)
  - Client Side Encryption: Criptografia no Lado Cliente
- É importante saber qual método usar para cada situação

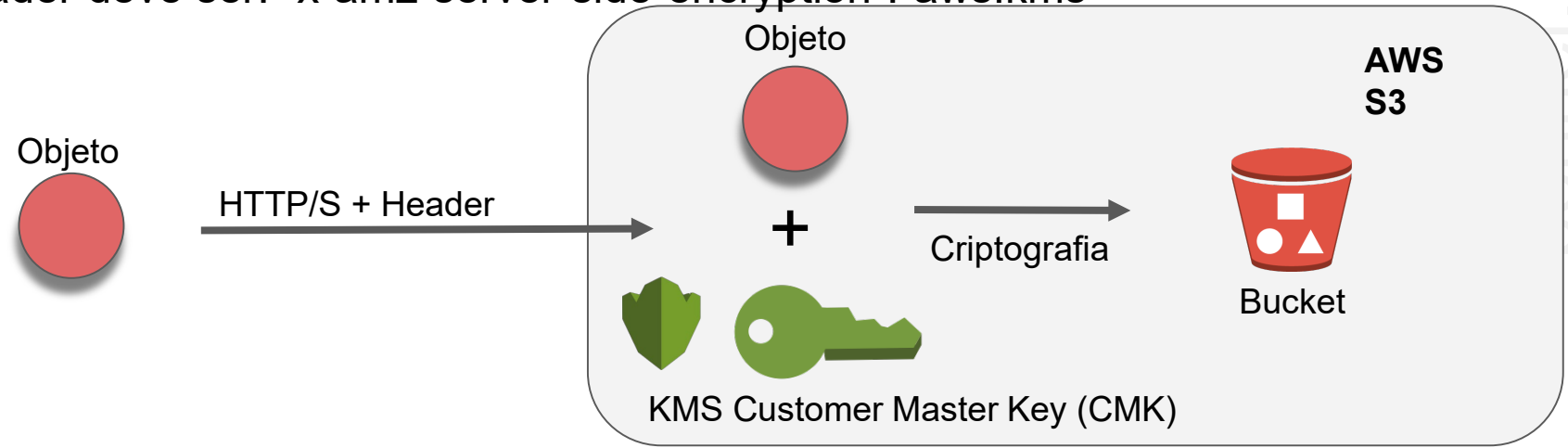
S3 - SSE-S3

- SSE-S3: Criptografa objetos usando Key gerenciada pela AWS
- Objeto é criptografado no lado Servidor
- Criptografia tipo AES-256
- Header deve ser: “x-amz-server-side-encryption”:”AES256”



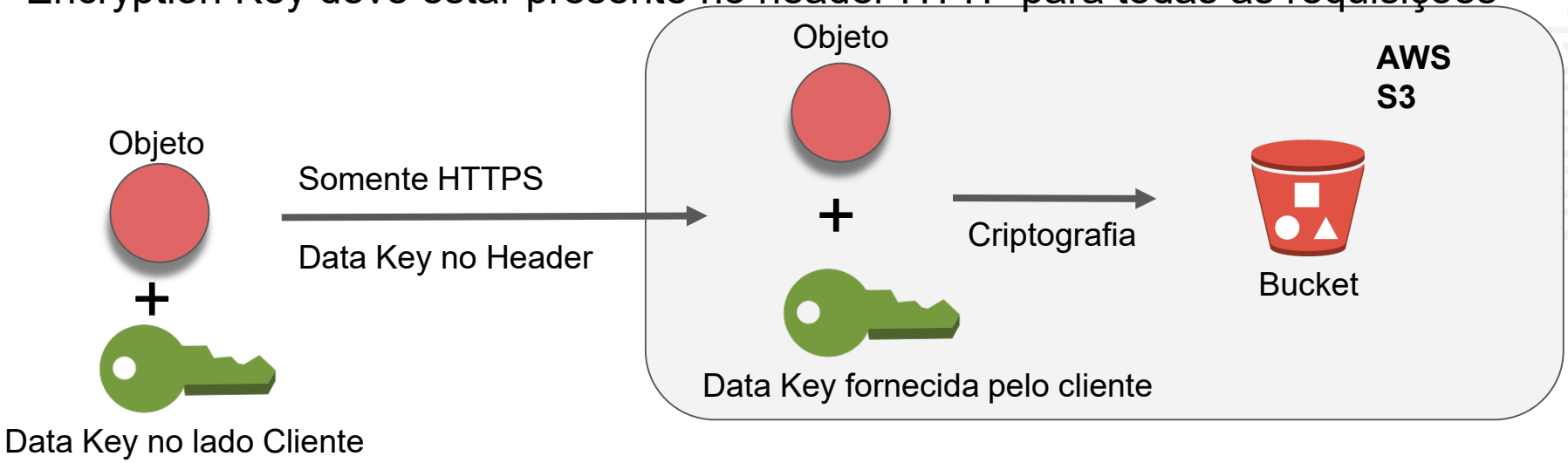
S3 - SSE-KMS

- SSE-S3: Criptografa objetos usando Key gerenciada pela AWS KMS
- Vantagens de usar KMS: Controlado pelo usuário + rastreamento para auditoria
- Objeto Criptografado no lado servidor
- Header deve ser: “x-amz-server-side-encryption”:”aws:kms”



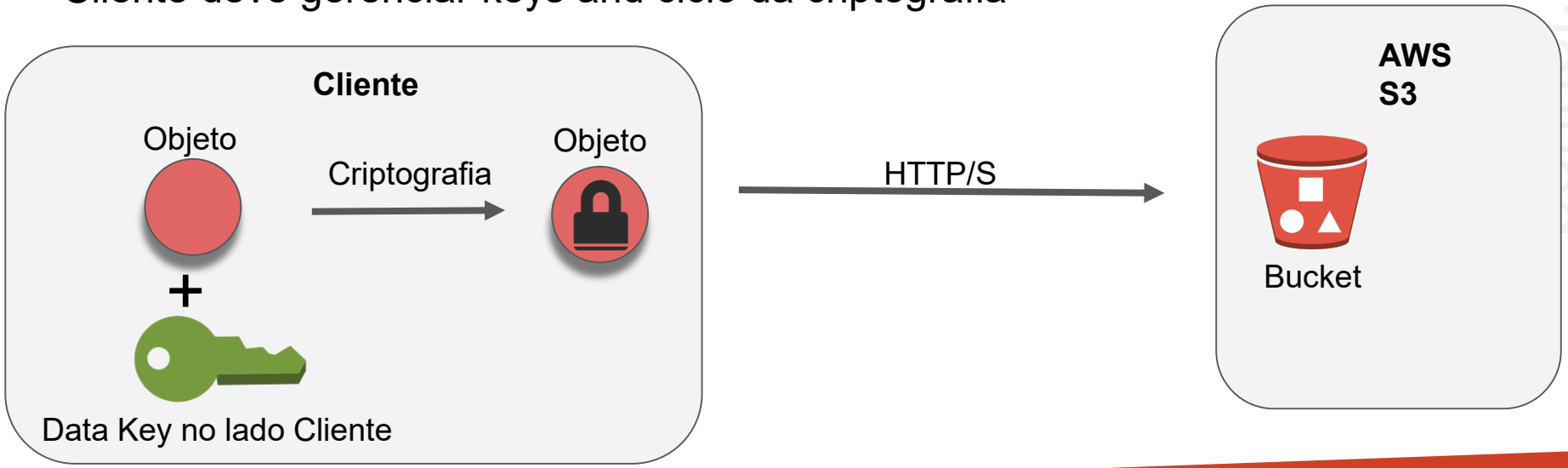
S3 - SSE-C

- SSE-C: Criptografa do lado servidor gerenciada pelo usuário fora do AWS
- S3 não armazena Encription Key fornecida pelo cliente
- É obrigatório o uso de HTTPS
- Encryption Key deve estar presente no header HTTP para todas as requisições



S3 - Criptografia do Lado Cliente

- Client Library como por exemplo Amazon S3 Encryption Client
- Cliente deve criptografar os dados antes de enviar para S3
- Cliente deve descriptografar dados depois de baixar do S3
- Cliente deve gerenciar keys and ciclo da criptografia



S3 - Segurança

- Usuário
  - IAM policies - Qual API é permitida para cada usuário
- Recursos
  - Bucket Policies - Regras para o bucket - Permite cross account
  - Object Access Control List (ACL) - permite controle mais granular
  - Bucket Access Control List (ACL - Menos comun

S3 - Bucket Policies

- JSON policies
  - Resource: Buckets and Objetos
  - Action: Allow(permitir) ou Deny(negar) API
  - Effect: Allow / Deny
  - Principal: Conta ou usuário para aplicar a policy
- Use Bucket policy para:
  - Permitir acesso público ao bucket
  - Forçar objetos para serem criptografados ao se fazer upload
  - Permitir acesso de outras contas (cross account)

S3 - Security

- Networking:
  - Suporta VPC endpoints (acesso a recursos utilizando endereço AWS)
- Logging and Auditoria:
  - Acesso ao S3 pode ser armazenado em outro Bucket
  - API calls pode ser registrados no AWS Cloud Trail
- User Security (Segurança nível usuário)
  - MFA (multi factor authentication) pode ser requerido em Buckets com Versioned habilitado, para deletar objetos
  - Signed URLs: URLs que são válidas somente por um período

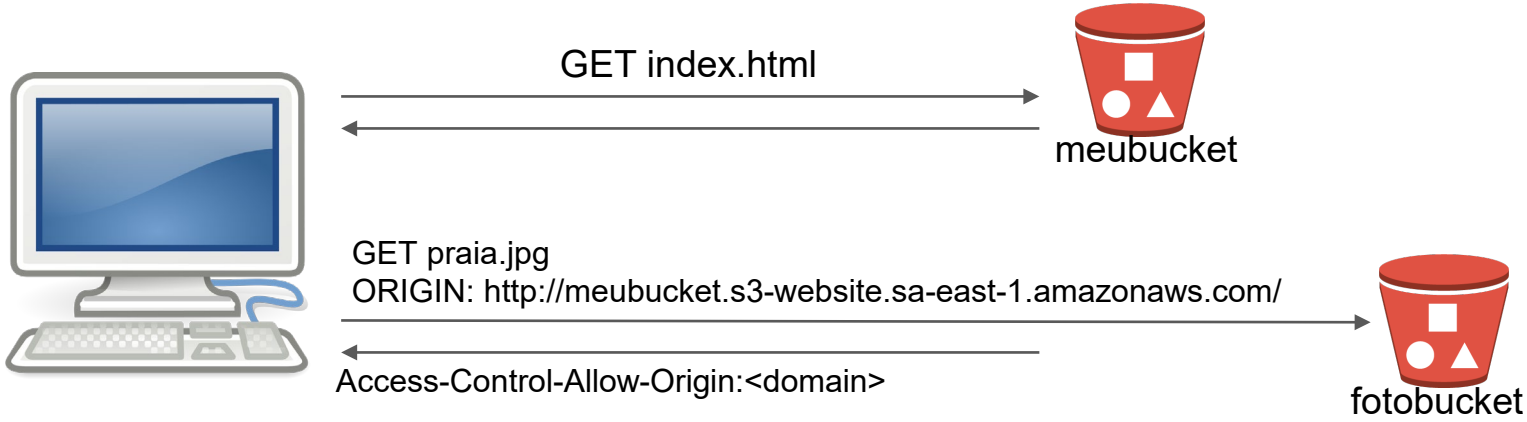
S3 - Websites

- S3 pode ser usado para hospedar websites estático
- O endereço será algo do tipo:
  - <nome-do-bucket>.s3-website-<AWS-region>.amazonaws.com

    ou
  - <nome-do-bucket>.s3-website.<AWS-region>.amazonaws.com
- Se você receber o erro 403 (forbidden)(proibido). Verifique se seu Bucket está habilitado para acesso público

S3 - CORS (Cross Resource Sharing)

- Se for solicitar dados de outro Bucket, é necessário habilitar CORS
- CORS permite limitar o número de websites que podem requisitar arquivos do S3 Bucket (usado para limitar custo)
- Pergunta frequente no exame:



S3 - Consistency Model

- Ler depois de escrever para PUTS de objetos novos
  - Assim que o objeto é escrito ele pode ser lido
  - Isso é verdade, com exceção se fizer um GET antes para ver se o objeto existe
- Eventual Consistency para DELETE e PUT para objetos existentes
  - Ao ler um objeto logo após fazer um update, pode ser retornado a versão antiga
  - Se um objeto for deletado, pode ser que ainda fique acessível por um curto período de tempo

S3 - Performance - Key Names

- Quando o número de transações por segundo é maior que 100 TPS a performance do S3 pode ser degradada
- Cada objeto vai para uma partição. É importante dividir os objetos entre as partições
- Era recomendado ter caracteres aleatórios na frente do key-name para melhor perfomance
  - <meu_bucket)/fe45_minha_pasta/my_arquivo1.txt
  - <meu_bucket)/ar33_minha_pasta/my_arquivo2.txt
- É recomendado nunca usar data como prefixo

S3 - Performance - Key Names

- Em 17 de julho de 2018, é possível escalonar até 3500 TPS para PUT e 5500 TPS para GET para cada prefixo
- Sendo assim não é mais necessário criar prefixos aleatórios para melhor performance

S3 - Performance

- Upload para arquivos maior que 5GB, use multipart upload:
  - Executa PUT em paralelo
  - Maximiza a capacidade da rede
  - Diminui o tempo para reenviar em caso de falha
- Use CloudFront como cache ao redor do mundo para otimizar leitura
- S3 Transfer Acceleration (Usa Edge Locations) - só precisa mudar o endpoint
- Se usar criptografia SSE-KMS, atente para o limite AWS de ~1000 download / upload por segundo

AWS - Developing

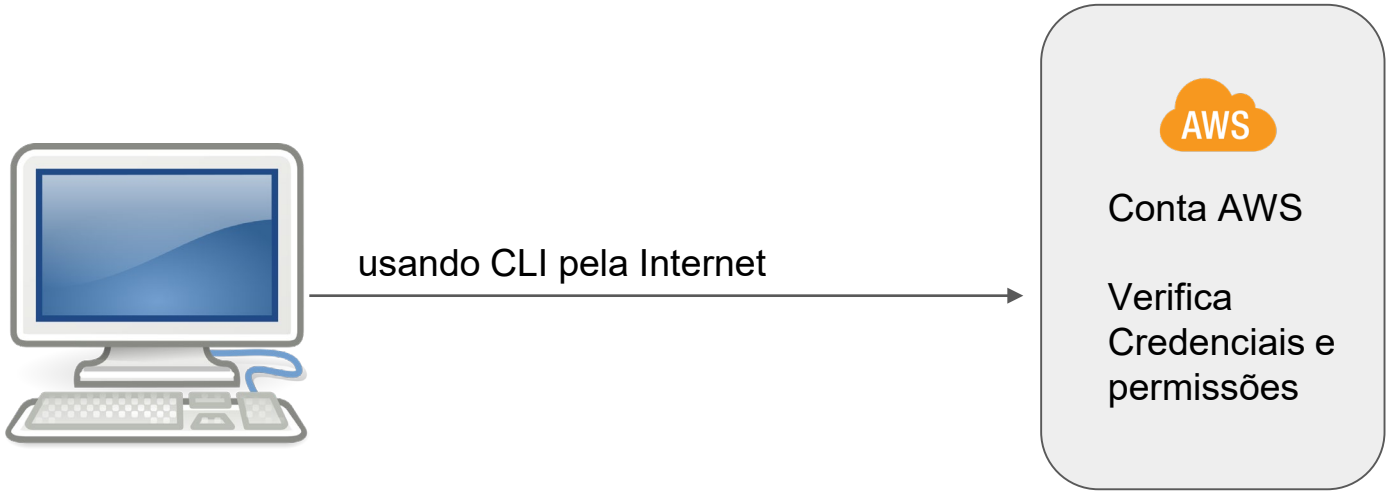
- Usar a plataforma AWS para desenvolvimento tem 2 componentes principais:
  - Como interagir com a plataforma AWS sem usar o console
  - Como interagir com serviços AWS tal como S3, DynamoDB e outros

AWS - Developing

- Formas de interagir com a plataforma AWS:
  - Usando AWS CLI (command line interface) no seu computador
  - Usando AWS CLI na instancia EC2
  - Usando AWS SDK (Software Developer Kit) no seu computador
  - Usando AWS SDK na instancia EC2
  - Usando AWS instance Metadata Service para EC2
- Vamos aprender a usar todas essas formas da melhor forma possível e usando a melhores práticas de segurança

AWS - Configuração do CLI

- Como configurar de forma correta o AWS-CLI
- Nunca forneça AWS Access Key e Secret Key para outra pessoa



AWS - Usando CLI em Instancia EC2 (Jeito CERTO)

- NUNCA DIGITE SUAS CREDENCIAIS DENTRO DE UMA INSTÂNCIA EC2
- IAM Role DEVE ser anexado ao EC2
- IAM Role pode ter um policy que diz exatamente o que EC2 pode fazer
- EC2 pode ser anexada somente a um IAM Role mas um Role pode ser anexada a várias instâncias EC2
- EC2 pode usar IAM sem nenhuma configuração adicional
- Esta é a melhor prática e você deve sempre usar EC2 com IAM Roles

AWS - CLI DRY RUN

- Usamos Dry Run para testar se temos permissão para executar comandos
- O comando não será realmente executado
- Cuidado. Alguns comandos podem custar caro se excutados. Por exemplo, criar uma instância com 64GB de RAM
- Alguns comandos CLI (não todos) pode conter --dry-run ao final para simular API calls

AWS - CLI STS Decode Erros

- Quando você executa um API call, você pode receber uma grande mensagem com erro
- Esse erro pode ser decodificado usando STS
- STS decode-authorization-message

AWS - EC2 Instance Metadata

- AWS EC2 Instance Metadata é uma importante ferramenta pouco conhecida pelos desenvolvedores de software
- Permite que uma instancia EC2 aprendam sobre si própria
- URL: http://169.254.169.254/latest/meta-data/
- É possível recuperar o nome do IAM Role pelo metadata, mas você NÃO pode recuperar o IAM policy
- Metadata: Informações sobre instancia EC2
- Userdata: Script inicial para instancia EC2

AWS - SDK

- Usado para acessar os comandos CLI de dentro da sua aplicação
- SDK (software development kit)
- AWS oferece SDKs para
  - Java
  - .NET
  - Node.js
  - Python (boto3 / botocore)
  - Go
  - Ruby
  - C++

AWS - SDK

- Precisamos usar SDK quando programamos Serviço AWS, tal como DynamoDB
- AWS CLI usa Python SDK (boto3)
- No exame você precisa saber quando usar SDK
- Vamos praticar SDK quando estudarmos Lambda
- Se você não especificar o Region. SDK usará us-east-1 como padrão

AWS - SDK - Segurança da Credencial

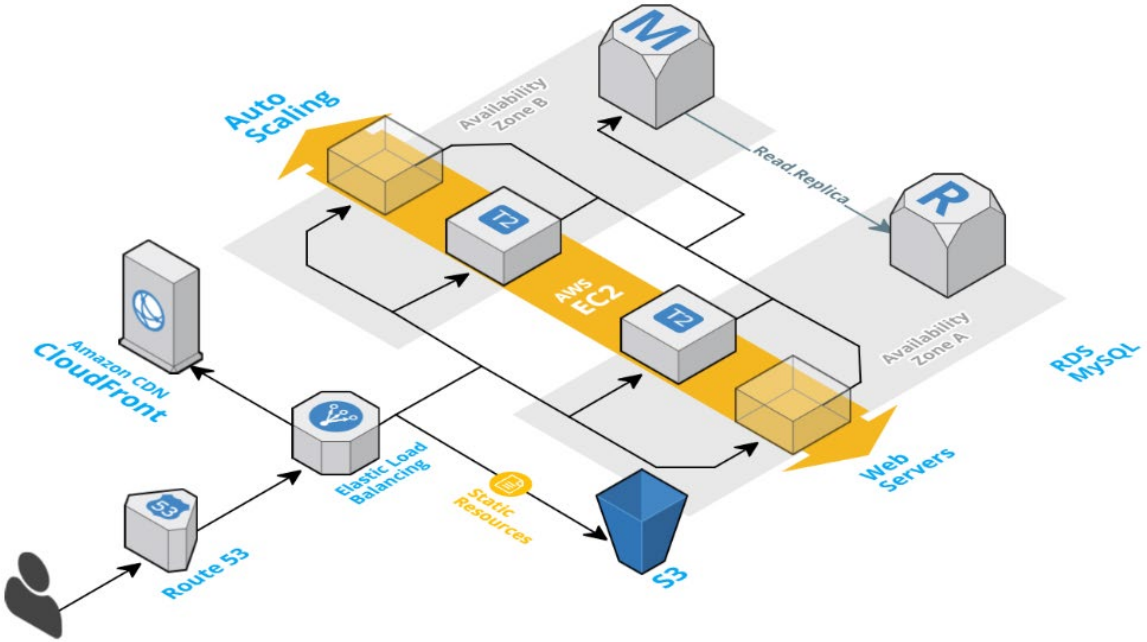
- É recomendado usar Default Credential Provider Chain
- Default Credential Provider Chain nos bastidores com:
  - Credenciais AWS ~/.aws/credentials (computador pessoal ou on premise)
  - Instance Profile Credentials usando IAM Role para EC2
  - Environment Variables
    - AWS_ACCESS_KEY_ID
    - AWS_SECRET_ACCESS_KEY
- Nunca coloque Credenciais AWS no seu código
- Melhores Práticas recomenda o uso de IAM role quando estiver trabalhando com Serviços AWS

Exponential Backoff

- Exponential Backoff
  - Qualquer API que falha devido a excesso de Calls, deve-se usar Exponential Backoff
  - Isso aplica-se para Rate Limited API
  - Pode ser feito usando SDK API calls
  - Tentativas:
    - 1 falhou espera 1 UT
    - 2 falhou espera 2 UT
    - 3 falhou espera 4 UT
    - 4 falhou espera 8 UT

AWS ElasticBeanStalk

- AWS ElasticBeanStalk
    - Distribuindo aplicações de forma segura e previsível



Dificuldades para o Desenvolvedor no ambiente AWS

- Gerenciar Infra-estrutura
- Deploying Code
- Configurar todos os banco de dados, Load Balancers e etc
- Preocupação sobre escalonamento

- A maioria das aplicações tem a mesma arquitetura (ALB + ASG)
- Tudo que o desenvolvedor quer é seu código rodando
- Consistência entre aplicações e ambientes diferentes

ElasticBeanStalk

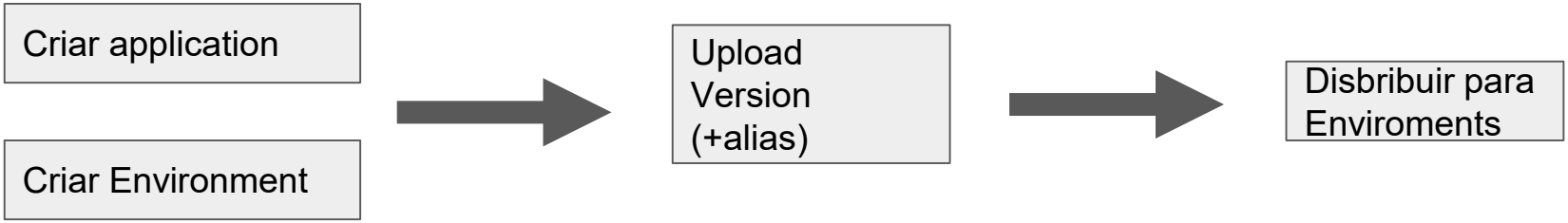
- É voltado para o desenvolvedor distribuir sua aplicação no AWS
- Utiliza componentes que nós já vimos como:
  - EC2
  - ASG
  - ELB
  - RDS
  - Outros
- Tudo em um só ambiente
- É possível mudar as configurações
- BeanStalk é grátis. Você paga pelos recursos que utilizar

ElasticBeanStalk

- Gerenciado pelo AWS
  - Configuração da Instância / Sistema Operacional gerenciado pelo BeanStalk
  - Deployment é configurável mas executado pelo BeanStalk
- Somente o código da aplicação é responsabilidade do desenvolvedor
- 3 tipos de arquitetura:
  - Single Instance Deployment (1 só instância): Ideal para testes
  - LB + ASG: Ideal para pré-produção e produção de aplicações web
  - ASG only: Ideal para não web apps (serviços)

ElasticBeanStalk

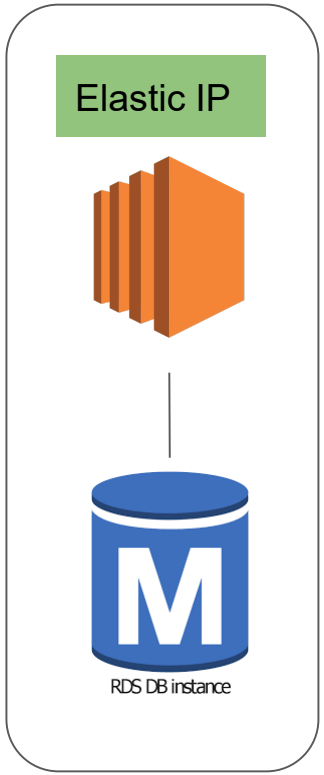
- ElasticBeanStalk tem 3 componentes:
  - Application
  - Application version: Cada deployment recebe uma versão
  - Environment name (dev, test, prod…): qualquer nome
- Application Versions são implantadas para Environments
- Rollback: restaura a aplicação anterior
- Controle total do ciclo de vida (lifecycle) dos environments

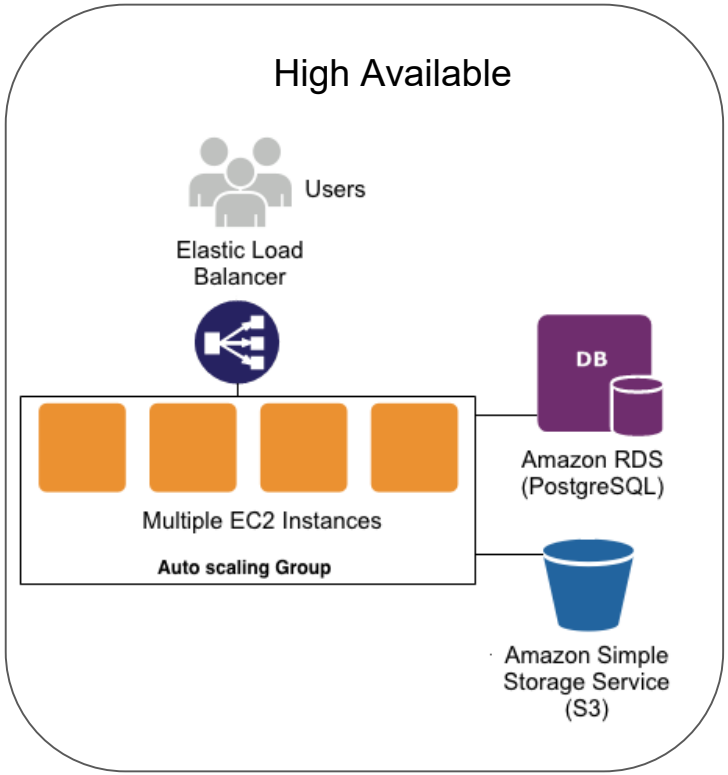


ElasticBeanStalk

- Suporta as seguintes plataformas:
  - GO
  - Java SE
  - Java com Tomcat
  - .NET (Windows Server com IIS)
  - Node.js
  - PHP
  - Python
  - Ruby
  - Packet Builder
  - Single Container Docker
  - Multicontainer Docker
  - Preconfigured Docker
  - Se a plataforma não for suportada, é possível criar uma Personalizada (avançado)

ElasticBeanStalk Deployment Modes





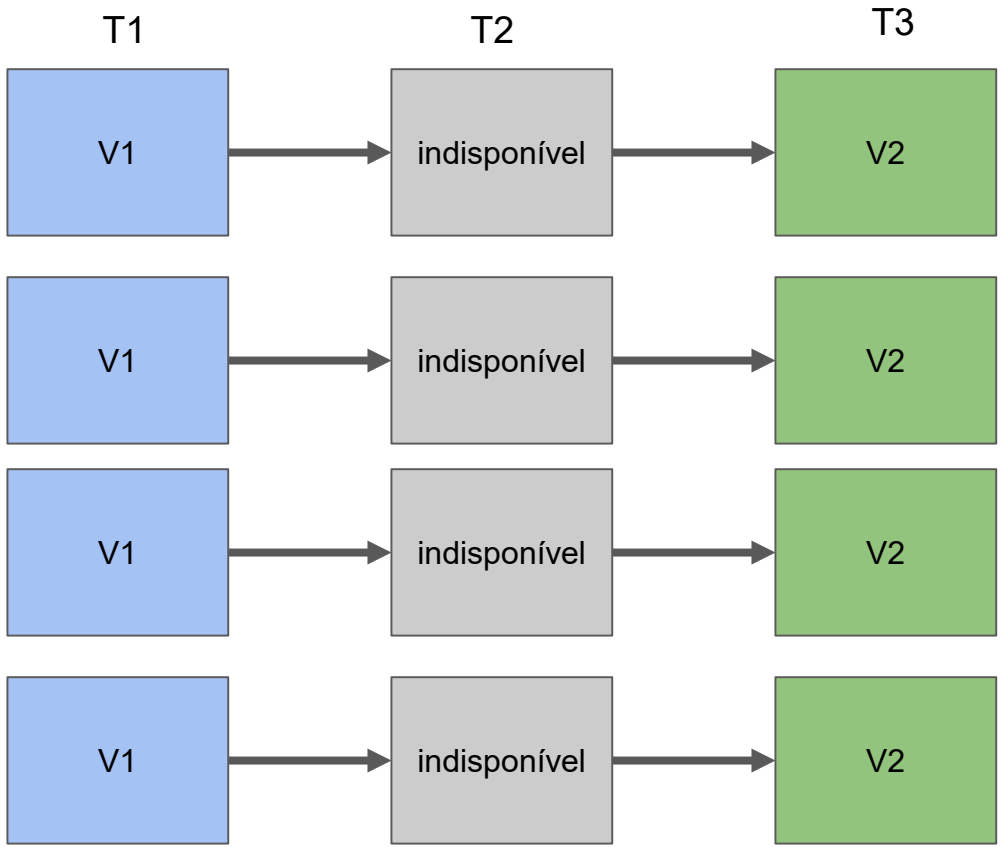
ElasticBeanStalk - Opções para atualizar sua aplicação

- All at once: Atualiza tudo de uma vez só. Aplicação fica indisponível durante atualização
- Rolling: Atualiza grupo de instancias (bucket), quando finalizar o primeiro bucket e a instancias são consideradas Healthy (saudável). BeanStalk começa a atualizar o próximo bucket
- Rolling with Additional Batches: similar ao Rolling porém inicia novas instâncias para instalar a nova versão (a versão antiga continua disponível)
- Immutable: Inicia novas instâncias em um novo ASG, publica a aplicação para as novas instâncias e então alterna o tráfego para as novas instâncias quando elas são consideradas saudáveis (healthy)

ElasticBeanStalk Deployment: All at once

- Publicação rápida
- Aplicação para de funcionar

  Durante publicação
- Ideal para desenvolvimento
- Não tem custo adicional



ElasticBeanStalk Deployment: Rolling

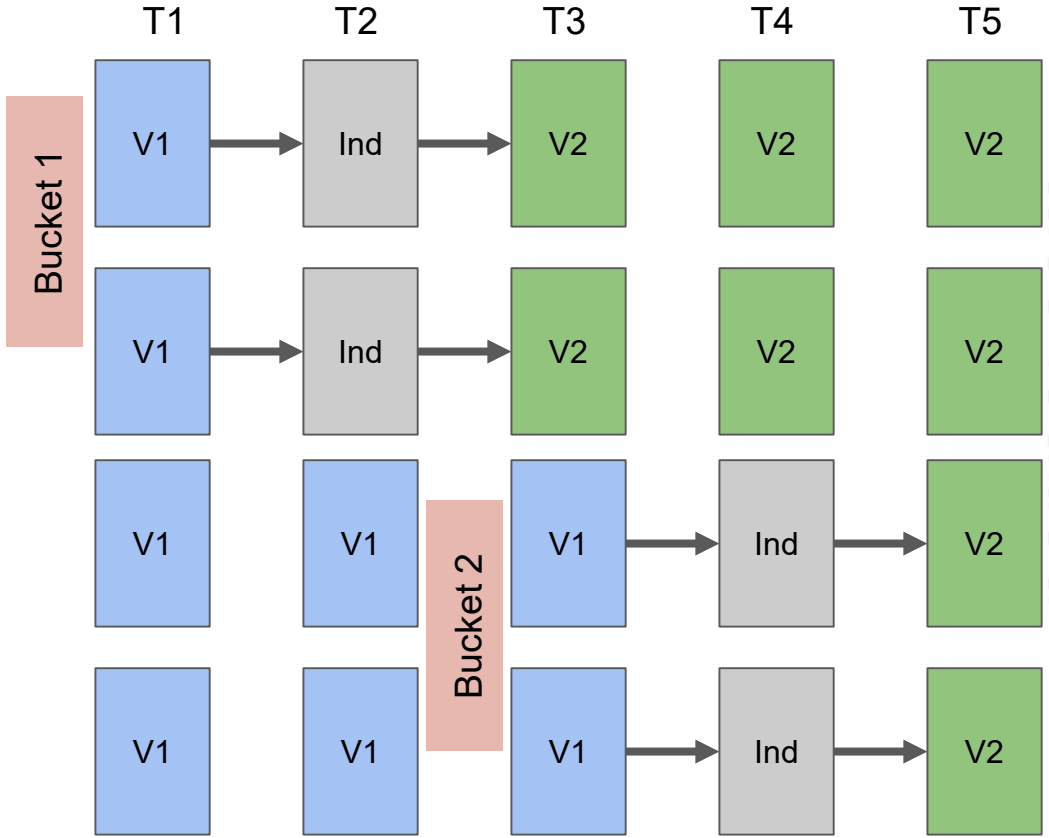
- Aplicação roda

  Com capacidade

  Menor
- Tamanho do Bucket

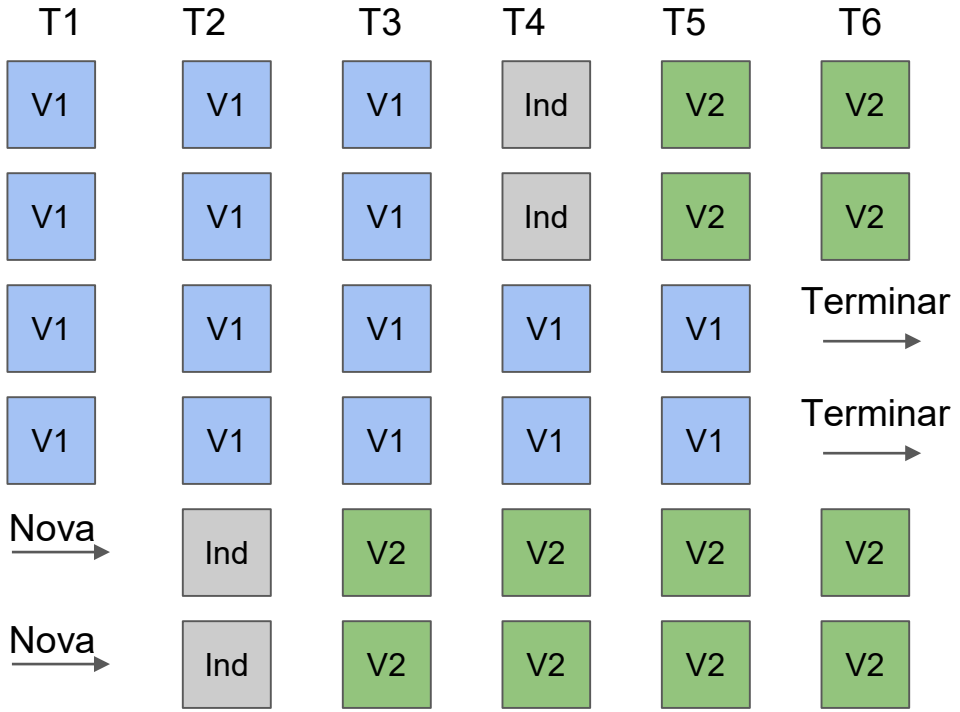
  Ajustável
- Aplicação roda duas versões

  ao mesmo tempo
- Não tem custo adicional
- Demora mais que All at Once



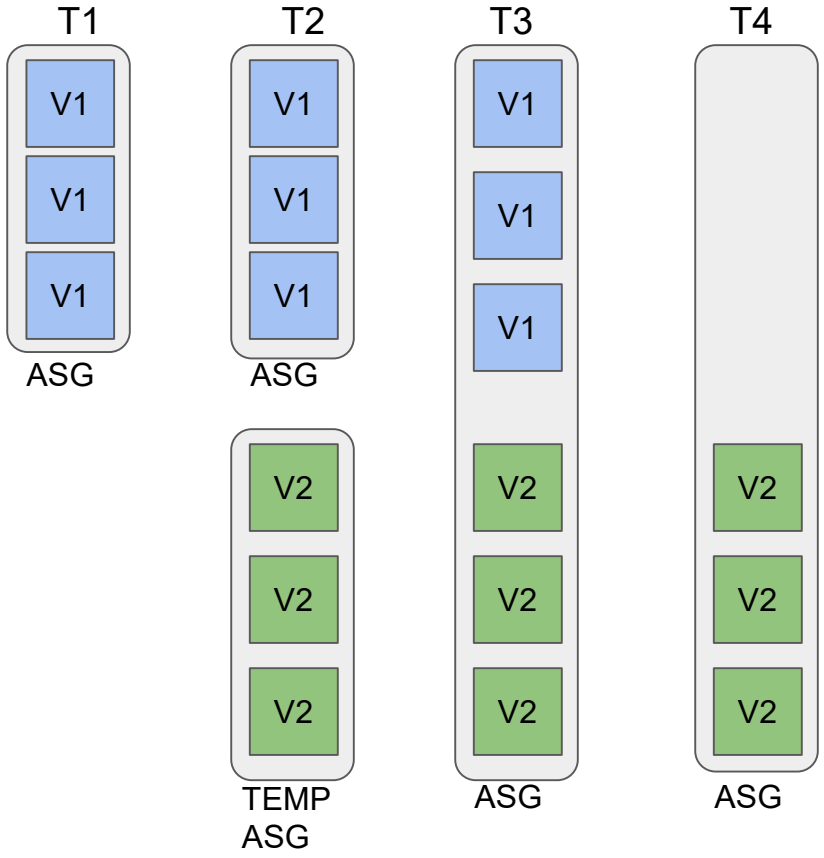
ElasticBeanStalk Deployment: Rolling with additional batches

- Aplicação roda Com capacidade Total
- Tamanho do Bucket Ajustável
- Aplicação roda duas versões ao mesmo tempo
- Instâncias adicionais são terminadas ao final.
- Demora mais.
- Ideal para Produção



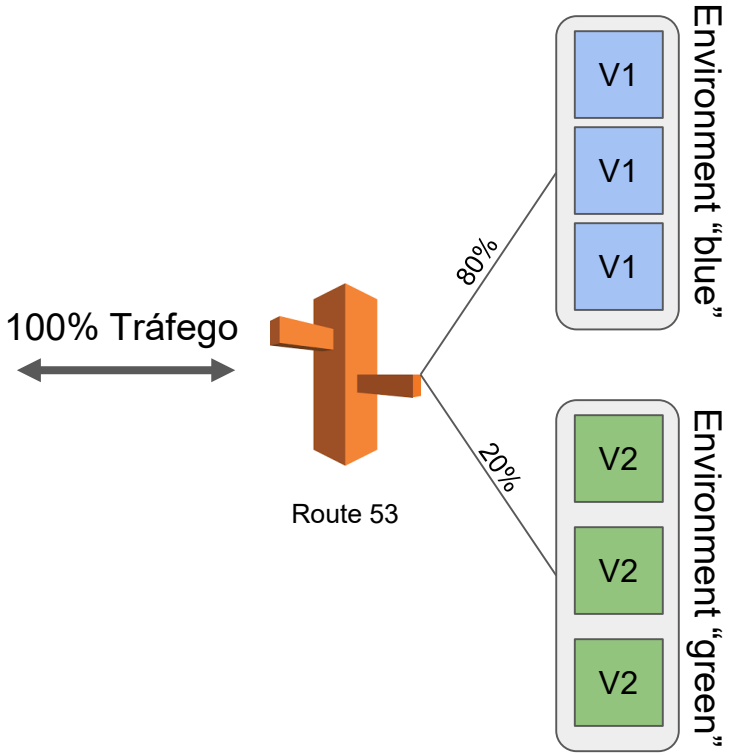
ElasticBeanStalk Deployment: Immutable

- Zero DownTime
- Novo código é publicado para um novo ASG
- Custo Maior
- Rápido Roolback em caso de falhas
- Ideal para produção



ElasticBeanStalk Deployment: Blue / Green

- Não é uma função do ElasticBeanStalk
- Cria um novo Enviroment and publica a V2.
- Novo environment (green) pode ser validado Roll Back se necessário
- Route 53 pode ser configurado com Weighted Policy para redirecionar parte do tráfego
- Use BeanStalk para alternar “swap URL” quando terminar com o teste do novo environment



ElasticBeanStalk - Deployment resumo

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

Deployment Methods

Method	Impact of Failed Deployment	Deploy Time	Zero Downtime	No DNS Change	Rollback Process	Code Deployed To
All at once	Downtime	🕒	X	✓	Redeploy	Existing instances
Rolling	Single batch out of service; any successful batches prior to failure running new application version	🕒 🕒 †	✓	✓	Redeploy	Existing instances
Rolling with additional batch	Minimal if first batch fails, otherwise, similar to **Rolling**	🕒 🕒 🕒 †	✓	✓	Redeploy	New and existing instances
Immutable	Minimal	🕒 🕒 🕒 🕒	✓	✓	Redeploy	New instances
Blue/green	Minimal	🕒 🕒 🕒 🕒	✓	X	Swap URL	New instances

ElasticBeanStalk - Extensions

- Deve ser feito upload do arquivo zip para o BeanStalk
- Configuração pode ser feita pelo console ou por arquivo
- Requerimento:
  - Pasta .ebextensions/ no root do código fonte
  - YAML/JSON format
  - Extensão .config (exemplo: logging.config)
  - Pode-se modificar alguns parametros usando: option_settings
  - Capacidade para adicionar recursos tal como RDS, ElastiCache, DynamoDB
- Recursos gerenciados pelo .ebextensions é apagado quando o environment é apagado

ElasticBeanStalk - CLI

- É possível instalar um CLI chamado EB CLI para trabalhar com ElasticBeanStalk usando CLI
- Comandos básicos:
  - eb create
  - eb status
  - eb health
  - eb events
  - eb logs
  - eb open
  - eb deploy
  - eb config
  - eb terminate
- Muito útil para automatizar publicação usando pipelines

ElasticBeanStalk - Como funciona

- ElasticBeanStalk utiliza CloudFormation
- Vamos estudar CloudFormation mais pra frente

ElasticBeanStalk - Mecanismo

- Descreve dependencies
  - requirements.txt para Python, package.json para Node.js
- Código deve ser em formato zip
- Arquivo zip é carregado para cada instância EC2
- Cada Instância EC2 resolve Dependencies (lento)
- Otimização em caso de longos Deployment: coloque Dependencies com Source Code (código fonte) para melhorar performance.

AWS CICD - Continuous Integration / Delivery

- Já aprendemos a criar recursos AWS manualmente
- Já aprendemos a interagir com AWS via CLI
- Como Deploy seu código usando Elastic BeanStalk
- Todos esses processos manuais fazem que erros sejam facilmente cometidos
- Vamos aprender agora como atualizar nosso código no repositório e o código será automaticamente deployed na Plataforma AWS da forma correta
  - Não esqueça de testar seu código antes do deployment
  - Criar Stages (dev, teste, prod)
  - Aprovação manual quando necessário
- Para ser um Desenvolvedor AWS é obrigatório saber AWS CICD

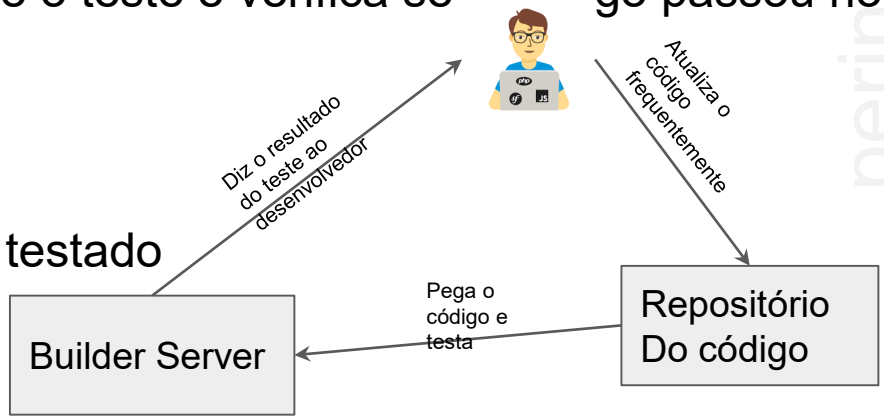
AWS CICD - Continuous Integration / Delivery

- Essa parte do curso é totalmente voltada para Deployment
- É a parte mais importante da certificação

- Vamos aprender:
  - AWS CodeCommit: Repositório para seu código
  - CodePipeline: automatizar nosso pipeline do código para ElasticBeanStalk
  - CodeBuild: Building e testando nosso código
  - AWS CodeDeploy: Deploy nosso código para grupo de instâncias EC2

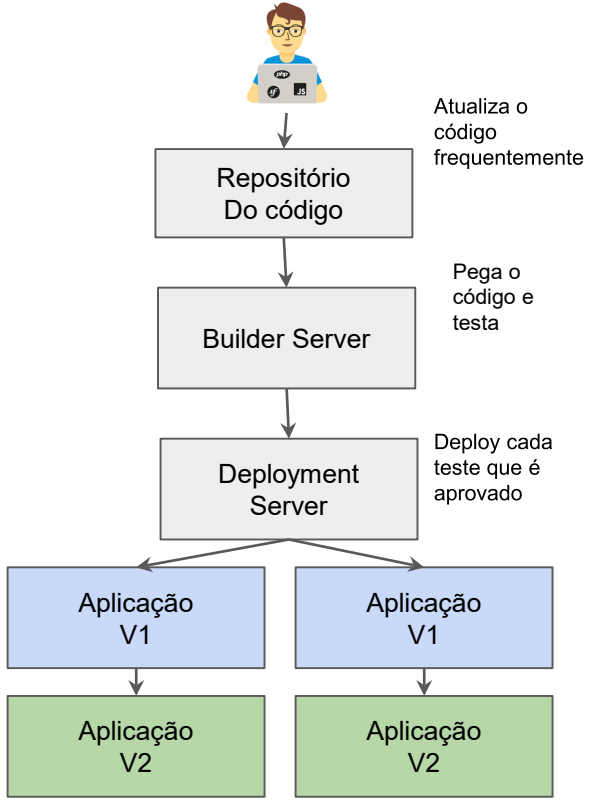
AWS CICD - Continuous Integration

- Desenvolvedores Push código para repositório, frequentemente GitHub ou CodeCommit.
- Build Server verifica/testa o código assim que é atualizado usando CodeBuild ou Jenkins
- O desenvolvedor recebe o feedback sobre o teste e verifica se o código passou no teste ou falhou
- Encontre bugs rapidamente e conserte
- Rápido deployment depois do código ser testado
- Deployment mais frequente
- Desenvolvedores mais felizes

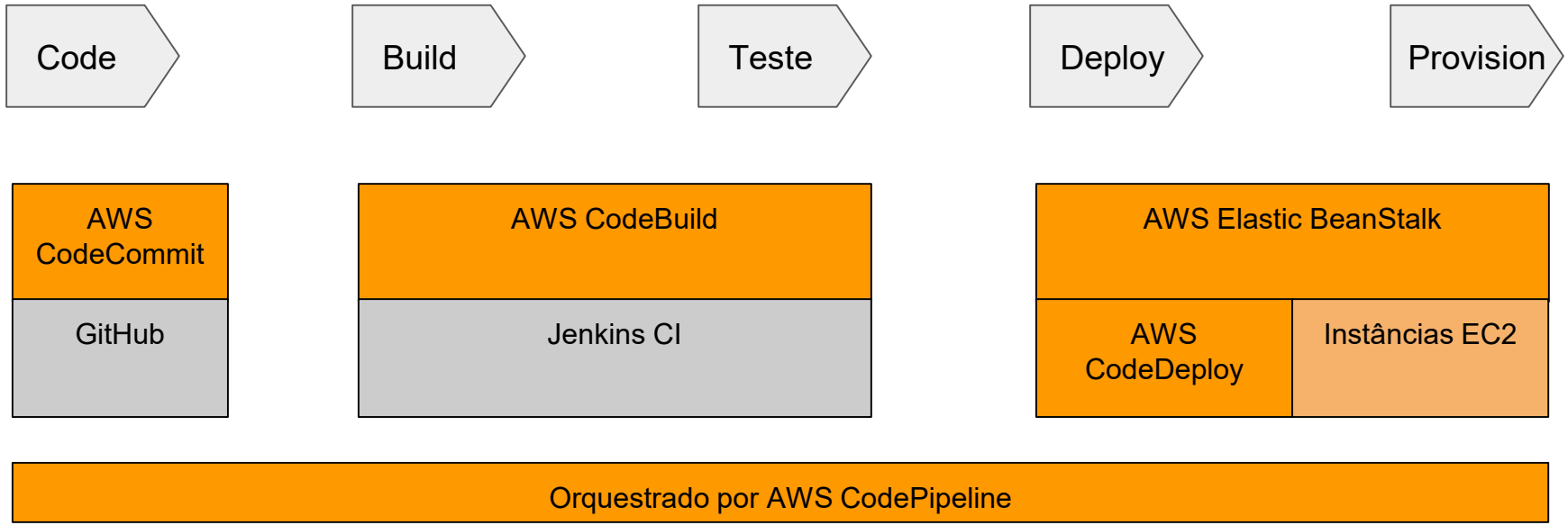


AWS CICD - Continuous Delivery

- Garante que o código será distribuído de forma confiável quando necessário
- Garante que Deployment pode ser frequente e rápido
- Muda o conceito de atualizar o código 1 vez por semana para 4 vezes por dia



Etapas CICD



AWS CodeCommit

- Version Control é a capacidade de entender and armazenar várias versões do código com a capacidade de Roll Back
- Habilitado usando Version Control System tipo Git
- Repositório Git pode ser armazenado em somente uma máquina, mas geralmente é armazenado em um repositório online
- Vantagens de repositório online:
  - Colaborar com outros desenvolvedores
  - Garante que é feito backup do código
  - Garante que está disponível e auditável

AWS CodeCommit

- Repositórios Git pode custar caro
- GitHub oferece repositórios público e privado grátis
- AWS CodeCommit:
  - Repositório Git privado
  - Sem limite de armazenamento
  - Gerenciado pela AWS
  - Alta disponibilidade
  - Código disponível somente na conta AWS (mais seguro)
  - Segurança (criptografia e controle de acesso)
  - Integrado com Jenkins / CodeBuild e outras ferramentas

AWS CodeCommit Security

- Interações são feitas usando Git
- Autenticação em Git:
  - SSH: Usuário AWS pode configurar SSH Keys no console IAM
  - HTTPS: Usando AWS CLI Authentication Helper ou gerando credenciais HTTPS
  - MFA (multi factor authentication) pode ser habilitado para segurança extra
- Autorização Git:
  - IAM Policies / Roles com permissão para acessar o repositório

AWS CodeCommit Security

- Criptografia
  - Repositório é automaticamente criptografado usando KMS
  - Criptografia em trânsito deve usar HTTPS ou SSH
- Cross Account Access:
  - Não compartilhe sua SSH Key
  - Não compartilhe sua Credencial AWS
  - Use IAM Role na sua conta AWS e use AWS STS com AssumeRole API

AWS CodeCommit vs GitHub

Similaridade:

- Repositório Git
- Code Review (pull requests)
- Pode ser integrado com AWS CodeBuild
- Suporta HTTPS e SSH

Diferenças

- Segurança
  - GitHub: GitHub Users
  - CodeCommit: AWS IAM
- Hospedagem
  - GitHub: hospedado pelo GitHub
  - GitHub Enterprise: hospedado no seu servidor
  - CodeCommit: Gerenciado e hospedado pela AWS
- Interface com usuário
  - GitHub: Oferece mais funções
  - CodeCommit: Oferece apenas o básico

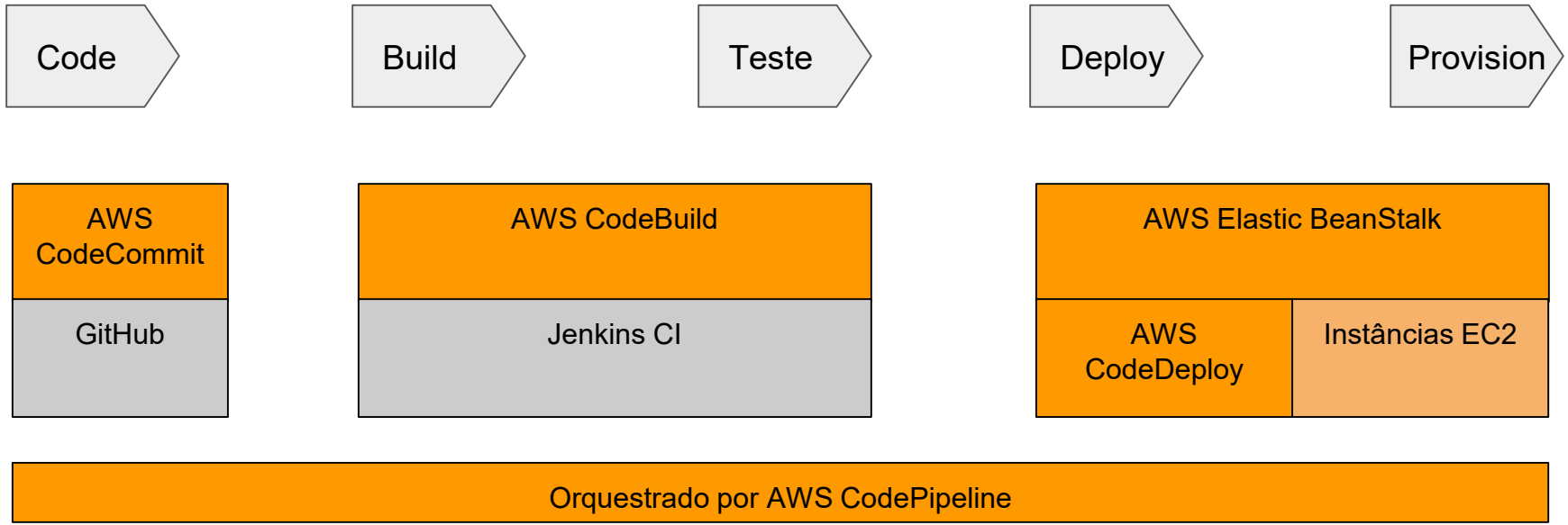
AWS CodeCommit Notifications

- Você pode habilitar notificações no CodeCommit usando AWS SNS (Simple Notification Service) ou AWS Lambda ou ainda AWS CloudWatch Event Rules
- Quando usar notificações SNS / AWS Lambda:
  - Branches deletados
  - Gatilho para atualização no Branch Master
  - Notificar Build System fora do AWS
  - Gatilho para AWS Lambda para realizar análise do código
- Quando usar CloudWatch Event Rules:
  - Gatilho para pull request updates (created, updated, deleted, commented)
  - CloudWatch Event Rules dispara um Tópico SNS

AWS CodePipeline

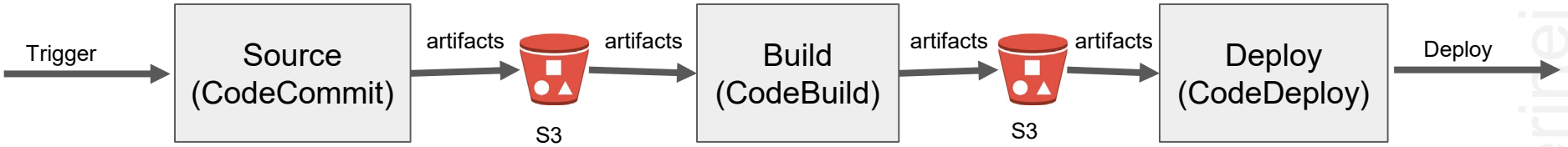
- Continuous Delivery
- É possível ver o fluxo paso-a-passo
- Source (onde está o código): GitHub, CodeCommit, Amazon S3
- Build: CodeBuild, Jenkins
- Load Test: ferramentas de terceiros
- Deploy: AWS CodeDeploy, beanStalk, CloudFormation, ECS
- Stage (estágio):
    - Cada estágio pode ter ações executadas de forma serial ou paralela
    - Exemplos de Stages: Build / Test / Deploy / Load Test…
    - Aprovação manual pode ser habilitada para qualquer Stage (estágio)

Etapas CICD



AWS CodePipeline Artifacts

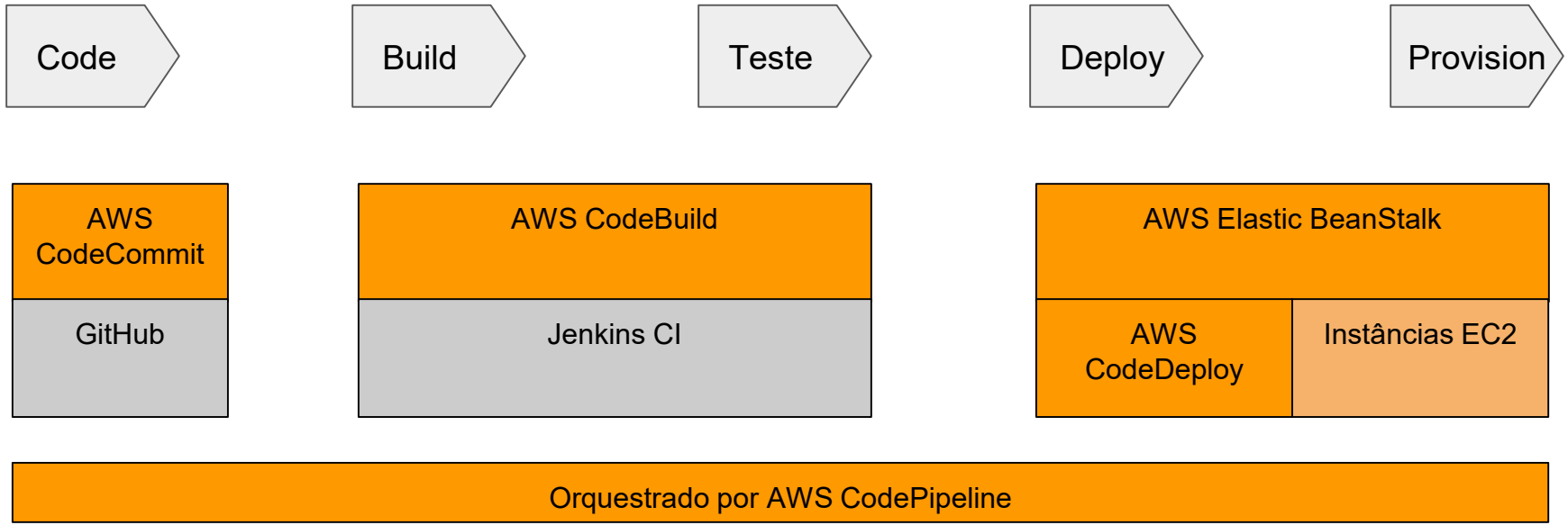
- Cada estágio do pipeline pode criar “artifacts”
- Artifacts são armazenados no S3 e repassados para o próximo estágio



AWS CodePipeline Troubleshooting

- Troubleshooting (Solução de Problema)
- Mudanças no CodePipeline Stage acontecem no AWS CloudWatch Events, que pode criar Notificações SNS
    - Possível criar eventos para falhas no pipeline
    - Possível criar eventos para Estágios Cancelados
- Se CodePipeline falhar em um estágio, seu pipeline para e você pode receber informações sobre o erro no console AWS
- AWS CloudTrail pode ser usado para auditar AWS API Calls
- Se Pipeline não puder executar uma tarefa, confira se IAM Service Role que está associado tem permissão suficiente (IAM Policy)

Etapas CICD



AWS CodeBuild

- Build Service gerenciado pela AWS
- Alternativa para usar outra ferramenta build, tal como Jenkins
- Continuous Scaling (Sem servidor para gerenciar ou provisionar)
- Cobrado somente pelo tempo que demora para completar o build
- Utilizar Docker nos bastidores
- Possibilidade de estender a capacidade usando nossa própria base de Docker
- Securança: Integração com KMS para criptografar os artifacts, IAM para Build, VPC para segurança de rede, CloudTrail para API calls logging

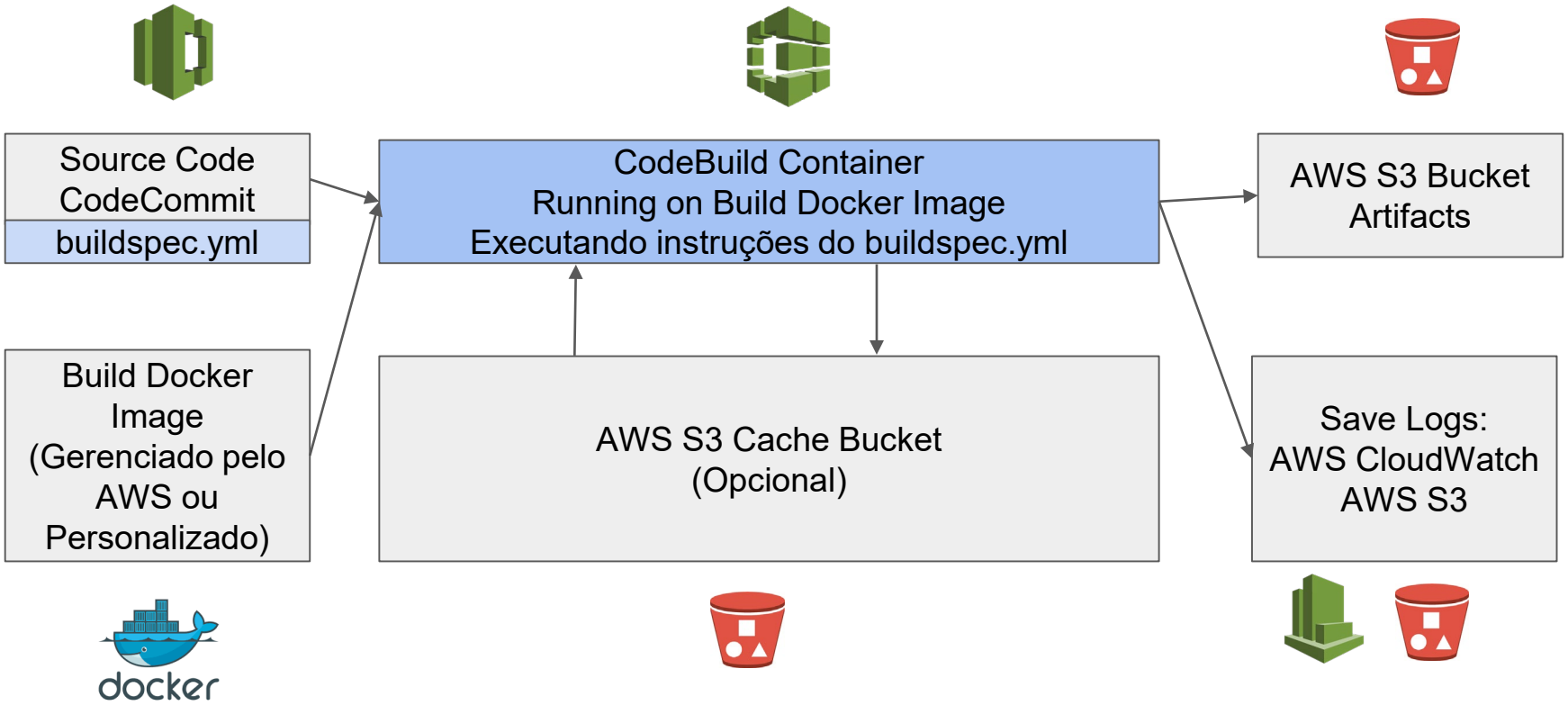
AWS CodeBuild

- Source Code pode ser do GitHub / CodeCommit / S3
- Instruções para o Build pode ser definidas no arquivo buildspec.yml
- Logs pode ser enviados para Amazon S3 e AWS CloudWatch Logs
- Metrics para monitorar estatísticas do CodeBuild
- Use CloudWatch Alarm para detectar falhas no Build e enviar notificações
- CloudWatch Events / AWS Lambda como Glue
- Notificações SNS
- Possibilidade de reproduzir CodeBuild localmente para troubleshoot em caso de erro
- Pipeline pode ser definido dentro do CodePipeline ou dentro do próprio CodeBuild

AWS CodeBuild - Ambientes suportados

- Java
- Ruby
- Python
- Go
- Node.js
- Android
- .NET Core
- PHP
- Docker: para usar qualquer ambiente que você quiser

Como AWS CodeBuild funciona



AWS CodeBuild - BuildSpec

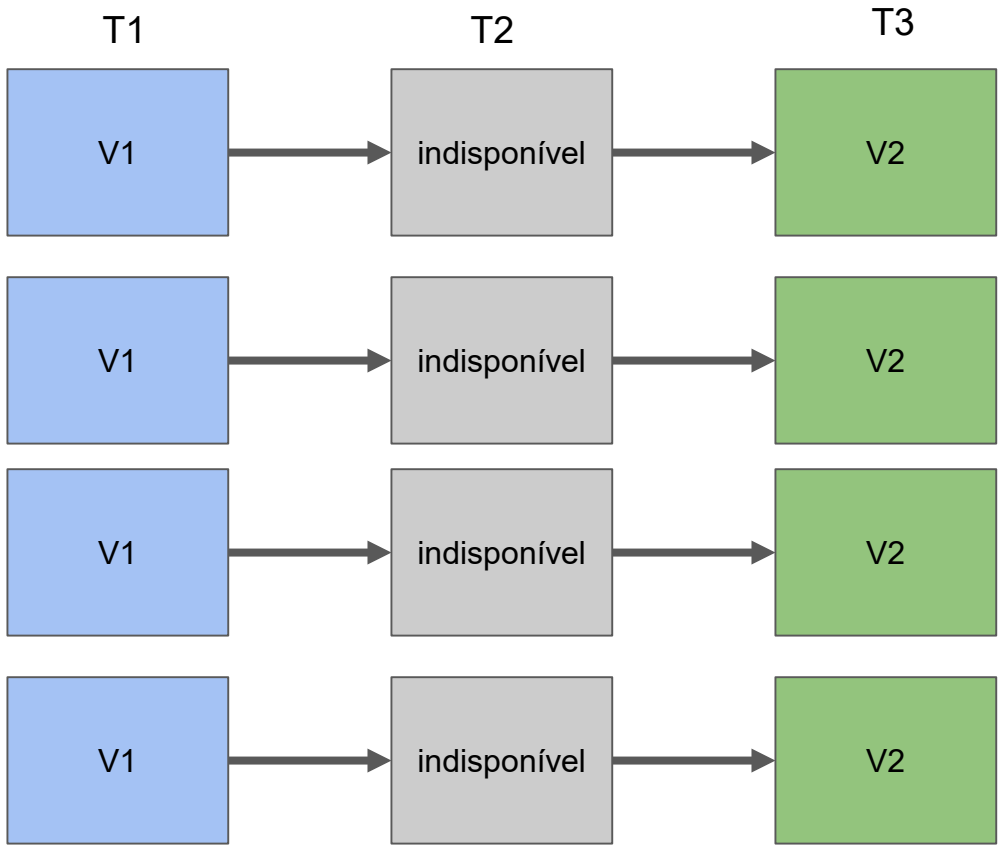
- Arquivo buildspec.yml deve estar no root do seu código
- Define as Environment Variables (Variáveis do Ambiente)
  - Plaintext variables
  - Secure Secrets: Utiliza SSM Parameter Store
- Phases (define os commandos a serem executados)
  - Install: instala dependencias que você vai precisar no seu Build
  - Pre Build: Comando final para ser executado antes do Build
  - Build: comando para construir o Build
  - Post Build: Etapa Final (exemplo: criar arquivo zip)
- Artifacts: O que vai ser atualizado no S3 (criptografado pelo KMS)
- Cache: Arquivos para por no Cache para acelerar Builds futuros

AWS CodeBuild - Local Build

- Em caso de precisar ir mais fundo para identificar problemas onde os logs não são suficientes
- É possível rodar CodeBuild Localmente no seu Desktop (depois de instalar Docker)
- Utilizar CodeBuild Agent

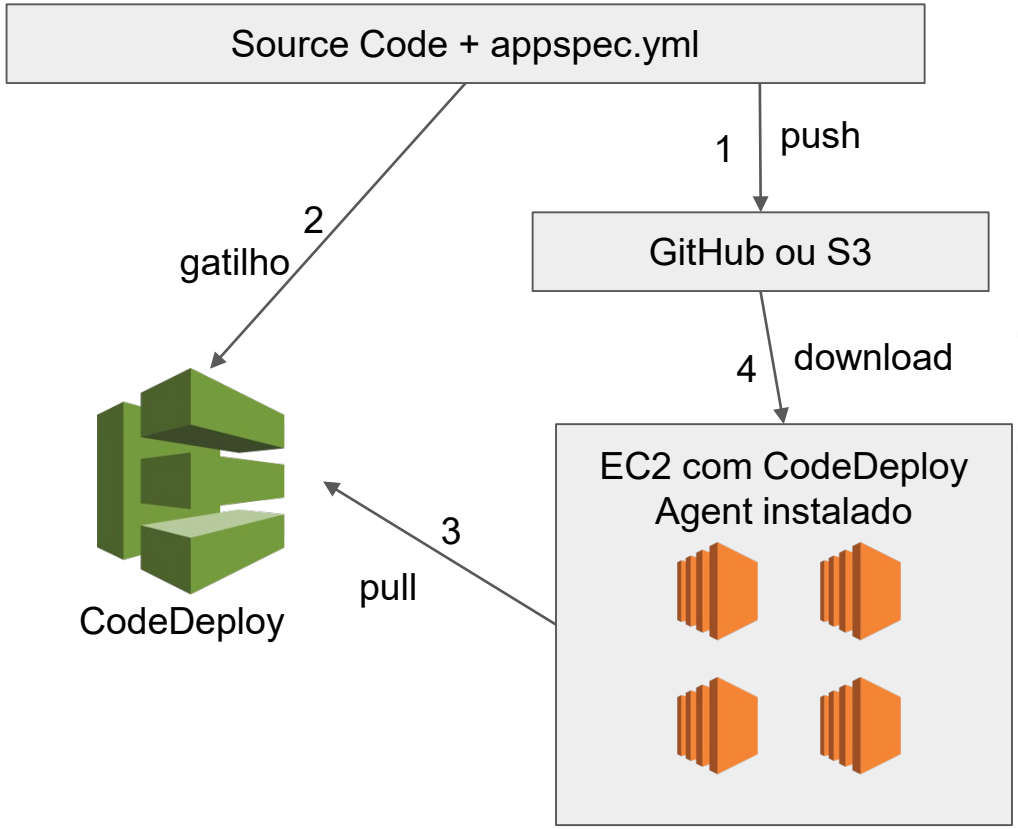
AWS CodeDeploy

- Deploy aplicação automaticamente para várias instâncias EC2
- Essas instâncias não são gerenciadas pelo BeanStalk
- Existem muitas formas de lidar com deployments usando ferramentas opensource (Ansible, Terraform, Chef, Puppet, etc…)
- Podemos utilizar AWS CodeDeploy que é gerenciado pela AWS



AWS CodeDeploy - passo-a-passo

- Instância EC2 ou Máquinas On Premise deve ter CodeDeploy Agent Instalado
- Agent monitora continuamente AWS CodeDeploy
- CodeDeploy envia appspec.yml
- Aplicação é baixada do GitHub ou S3
- EC2 executará as instruções recebidas
- CodeDeploy Agent reportará sucesso ou falha no deployment



AWS CodeDeploy

- Instância EC2 são agrupadas pelo Deployment Group (dev / test / prod)
- Várias opções para definir o tipo de deployment
- CodeDeploy pode ser integrado com CodePipeline e usar Artifacts
- CodeDeploy pode reusar ferramentas de configuração, funciona com qualquer aplicação, Integração com Auto Scaling
- Blue/Green deployment só funciona com EC2. Não funciona com On Premise
- Suporta AWS Lambda Deployment
- CodeDeploy não faz provisão recursos

AWS CodeDeploy - Componentes

- Application: nome único
- Compute Plataform: EC2/On-premise or Lambda
- Deployment Configuration: Regras do Deployment para Sucesso ou Falha
    - EC2/On-Premise: Pode-se especificar o número mínimo de EC2 saudável para deployment
    - AWS Lambda: Pode-se especificar como o tráfego será roteado para atualizar o Lambda Function
- Deployment Group: grupo de instância tagueada (TAGs)
- Deployment Type: In-Place or Blue/Green
- Application Revision: Código da aplicação + appspec.yml
- Service Role: Role para CodeDeploy realizar as tarefas necessárias
- Target Revision: Destino da Versão da Aplicação

AWS CodeDeploy - AppSpec

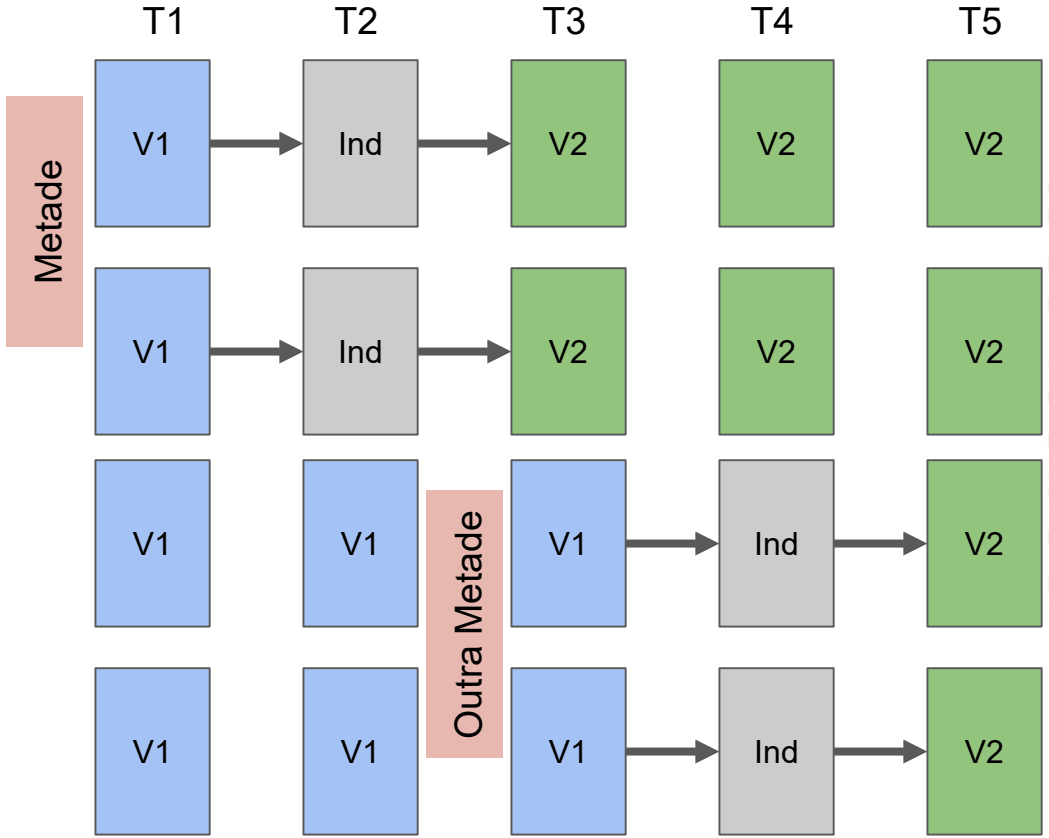
- File Section: Como copiar os arquivos do GitHub / S3 para o Destino
- Hooks: Conjunto de instruções para Deploy nova versão (hooks podem ter timeouts)
- Em ordem:
  - ApplicationStop
  - DownloadBundle
  - BeforeInstall
  - Install
  - AfterInstall
  - ApplicationStart
  - ValidateService: Importante

AWS CodeDeploy - Deployment Config

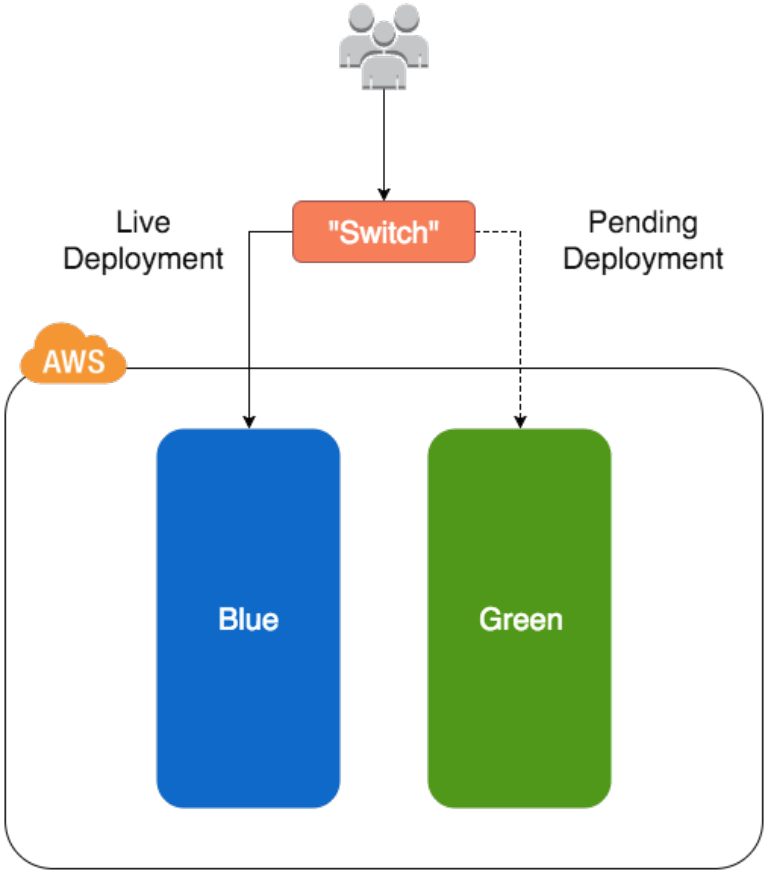
- Configs:
  - One at time: uma de cada vez. Se uma falhar o deployment para
  - Half at time: metade, 50%
  - All at Once: Rápido, Downtime, usado em desenvolvimento
  - Custom: personalizado
- Failures:
  - Instâncias ficam in “Failed State”
  - Novos deployments começam primeiro pelas instâncias em “Failed State”
  - Rollback: Deploy o código anterior ou habilita Rollback automático quando falha ocorrer
- Deployment Targets:
  - Grupo de Instâncias EC2 com Tags
  - Direcionado para um ASG
  - Mix of ASG/Tags para criar deployment segments
  - Personalização de scripts com DEPLOYMENT_GROUP_NAME Environment Variables

In Place Deployment - Half at a time

- Metade de cada vez



Blue Green Deployment



AWS CodeDeploy - LAB

- Criar EC2
- Instalar updates
- Instalar servidor apache
- Inicializar servidor apache
- Habilitar servidor apache
- Instalar codedeploy agent
- Anexar role para codedeploy agent
- Criar AMI personalizada
- Terminar instância EC2

AWS CloudFormation

- Infraestrutura como código
- Automatiza o processo de criar infraestrutura
  - Em outra região
  - Em outra conta AWS
  - Na mesma região, se tudo for deletado

AWS CloudFormation

- CloudFormation é uma forma declarativa de representar sua infraestrutura na AWS para a maioria dos recursos
- Em CloudFormation pode dizer coisas do tipo:
  - Crie um Security Group
  - Crie 2 instâncias EC2 usando esse Security Group
  - Crie 2 Elastic IP para estas 2 instâncias EC2
  - Crie um S3 Bucket
  - Crie um Load Balancer (ELB) na frente dessas Instâncias EC2
- Então, CloudFormation criará tudo que você pediu, na ordem correta. (Não é necessário especificar a ordem)

Vantagens de usar CloudFormation

- Infraestrutura como código
  - Nenhum recurso é criado manualmente (ótimo para controle)
  - Código pode usar controle de versão. Exemplo: Git
  - Mudanças na Infraestrutura são vista como código
- Custo
  - Cada recurso recebe um identificador, fácil de saber quando o stack custa
  - É possível estimar o custo usando CloudFormation Template
  - É possível programar seu ambiente de desenvolvimento para terminar às 17h e ser recriado às 9h

Vantagens de usar CloudFormation

- Produtividade
  - Possibilidade de destruir e criar recursos AWS dinamicamente
  - Diagrama da sua Infraestrutura é gerado automaticamente
  - Programação Declarativa. Sem necessidade de especificar a ordem
- Possível criar vários stacks para vários níveis (layer)
  - VPC stack
  - Network stack
  - App stack
- Não crie stacks do zero:
  - Use modelos disponíveis na web

Como CloudFormation funciona

- Templates precisam ser uploaded para S3 e então podem ser referenciados no CloudFormation
- Para atualizar o Template, não é possível editar o Template atual, é preciso fazer o upload de nova versão do Template.
- Stacks são identificados pelo nome
- Deletar um Stack. Deleta dos os Artifacts que foram criados pelo CloudFormation

Deploy a CloudFormation Template

- Manualmente:
  - Editando o Template no CloudFormation Designer
  - Usar o console para digitar os parametros
- Automaticamente:
  - Editar o Template usando arquivo YAML ou JSON
  - Usando CLI para Deploy Templates
  - Método recomendado quando se deseja automatizar sua plataforma

Estrutura do CloudFormation

Componentes dos Templates

1. Resources: Recursos AWS declarados no Template (Obrigatório)
2. Parameters: Entradas dinâmicas para seu Template
3. Mapping: Constantes para seu Template
4. Outputs: Referência aos Resources que foram criados
5. Conditionals: Lista de condições para criar Resources
6. Metadata

Template Helpers:

1. Reference
2. Functions

CloudFormation

- Você aprenderá como CloudFormation funciona
- Você aprenderá tudo que precisa pra responder as questões do exame
- O exame não pede que você escreva código em CloudFormation

CloudFormation - Exemplo

- Vamos criar uma instância EC2
- Vamos anexar um Elastic IP ao Instância
- Vamos adicionar dois Security Groups a instância EC2
- Não vamos focar na sintaxe do código agora
- Vamos estudar a estrutura do código mais tarde

CloudFormation - YAML

- YAML e JSON podem ser usadas com CloudFormation
- Suporta:
  - Key Value Pair
  - Nested Objects
  - Suporta Arrays
  - Strings em multiplas linhas
  - Pode incluir comentários

CloudFormation - Resource

- Resource é o centro do CloudFormation Template (Obrigatório)
- Representa os diferentes componentes da AWS que serão criados e configurados
- Resource podem referenciar outros Resources
- Existem mais de 224 tipos de Resources
- Resources Types Identifiers são escritos no formato:
    - AWS::nome-produto-aws::nome-data-type

CloudFormation - Parameters

- Parameters é a forma de entrarmos com dados para a criação de Resources
- Exemplos de uso de Parameters:
  - Reusar seus Templates dentro da Empresa
  - Quando algumas Entradas não podem ser definidas antecipadamente
- Parameters são muito importantes e previnem erros

CloudFormation - Parameter

- Quando usar?
- Faça a seguinte pergunta:
  - A configuração desse CloudFormation vai mudar no futuro, se a resposta for sim, use Parameters
- Não é necessário atualizar o Template para mudar seu conteúdo

```
Parameters:
  KeyName:
    Description: The EC2 Key Pair to allow SSH access to the instance
    Type: 'AWS::EC2::KeyPair::KeyName'
Resources:
  Ec2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      SecurityGroups:
        - !Ref InstanceSecurityGroup
        - MyExistingSecurityGroup
      KeyName: !Ref KeyName
      ImageId: ami-7a11e213
```

CloudFormation - Parameter Settings

- Parameters pode ser controlados pelos seguintes items:
  - Type:
    - String
    - Number
    - CommaDelimitedList
    - List<Type>
    - AWS Parameter (Ajuda a detectar valores inválidos. Valores são comparados com Valores na conta AWS)
  - Description
  - Constraints
- ConstraintDescription (String)
- Min/MaxLength
- Min/MaxValue
- Defaults
- AllowedValues (array)
- AllowedPattern (regexp)
- NoEcho (Boolean)

CloudFormation - Referenciar Parameter

- Função Fn:Ref é usada para referenciar parameters
- Parameters podem ser usados em qualquer lugar no Template
- Pode-se usar !Ref ao invés de Fn:Ref
- !Ref pode referenciar outros elementos dentro do Template

```
MyEIP:
  Type: "AWS::EC2::EIP"
  Properties:
    InstanceId: !Ref MyEC2Instance
```

CloudFormation - Concept: Pseudo Parameters

- AWS oferece Pseudo Parameters para qualquer CloudFormation Template
- Pode ser usado a qualquer momento e são habilitados por padrão

AWS::Region -> `sa-east-1`.

AWS::StackId -> `arn:aws:cloudformation:us-west-2:123456789012:stack/teststack/51af3dc0-da77-11e4-872e-1234567db123`.

AWS::StackName -> `teststack`.

AWS::URLSuffix - > `amazonaws.com`

CloudFormation - Mappings

- Mapping
  - Mappings são valores fixos dentro do CloudFormation Template
  - Muito útil para diferenciar dados entre Environments (dev vs prod), Regions, Tipos de AMI e outros
  - Todos os valores são Hardcoded no Template
  - Example

```
Mappings:
  RegionMap:
    us-east-1:
      "HVM64": "ami-0ff8a91507f77f867"
    us-west-1:
      "HVM64": "ami-0bdb828fd58c52235"
    eu-west-1:
      "HVM64": "ami-047bb4163c506cd98"
    ap-southeast-1:
      "HVM64": "ami-08569b978cc4dfa10"
    ap-northeast-1:
      "HVM64": "ami-06cd52961ce9f0d85"
```

CloudFormation - Mappings vs Parameters

- Mapping é ideal quando se sabe os valores que serão usados e esses valores pode vir de variáveis tais como:
  - Region
  - Availability Zone
  - AWS Account
  - Environment (dev vs prod)
- Permite um controle seguro sobre o Template
- Use Parameter quando os valores devem ser definidos pelo usuário

CloudFormation - Acessando Valores Mapping

- Fn:FindInMap
  - Retorna o valor (value) de uma key
  - !FindInMap [MapName, TopLevelKey, SecondLevelKey]

```
Mappings:
  RegionMap:
    us-east-1:
      HVM64: "ami-0ff8a91507f77f867"
      HVMG2: "ami-0a584ac55a7631c0c"
    us-west-1:
      HVM64: "ami-0bdb828fd58c52235"
      HVMG2: "ami-066ee5fd4a9ef77f1"
    eu-west-1:
      HVM64: "ami-047bb4163c506cd98"
      HVMG2: "ami-31c2f645"
    ap-southeast-1:
      HVM64: "ami-08569b978cc4dfa10"
      HVMG2: "ami-0be9df32ae9f92309"
    ap-northeast-1:
      HVM64: "ami-06cd52961ce9f0d85"
      HVMG2: "ami-053cdd503598e4a9d"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap
        - RegionMap
        - !Ref 'AWS::Region'
        - HVM64
      InstanceType: m1.small
```

CloudFormation - Outputs

- Output declara valores que podem ser importados por outros Stacks (Se você exporta-lo primeiro)
- Pode-se ver o Output no Console AWS ou usando AWS CLI
- Outputs são úteis quando você define uma rede no CloudFormation e, Output variáveis tal como VPC ID e Subnet ID
- É a melhor forma de colaborar usando Stacks. Cada Expert desenvolve um Stack e exporta as variáveis
- Não é possível deletar um Stack CloudFormation se um Output estiver sendo referenciado por outro Stack CloudFormation

CloudFormation - Output

- Criar um SSH Security Group como parte do Template
- Criar um Output que referencia este Security Group

```
Outputs:
    StackSSHSecurityGroup:
        Description: SSH Security Group para sua empresa
        Value: !Ref MinhaEmpresaSSHSecurityGroup
        Export:
          Name: SSHSecurityGroup
```

CloudFormation - Cross Stack Reference

- Criar um segundo Template que utiliza o Security Group criado pelo outro Template
- Usamos a função Fn::ImportValue
- Não é possível deletar um stack até que todas as referências a este stack seja também deletadas

```yaml
Resources:
  MySecureInstance:
    Type: AWS::EC2::Instance
    Properties:
      AvailabilityZone: sa-east-1
      ImageId: ami-b5erdrt4
      InstanceType: t2.micro
      SecurityGroups:
        - !ImportValue SSHSecurityGroup
```

CloudFormation - Output

- Criar um SSH Security Group como parte do Template
- Criar um Output que referencia este Security Group

```
Outputs:
    StackSSHSecurityGroup:
        Description: SSH Security Group para sua empresa
        Value: !Ref MinhaEmpresaSSHSecurityGroup
        Export:
          Name: SSHSecurityGroup
```

CloudFormation - Cross Stack Reference

- Criar um segundo Template que utiliza o Security Group criado pelo outro Template
- Usamos a função Fn::ImportValue
- Não é possível deletar um stack até que todas as referências a este stack seja também deletadas

```yaml
Resources:
  MySecureInstance:
    Type: AWS::EC2::Instance
    Properties:
      AvailabilityZone: sa-east-1
      ImageId: ami-b5erdrt4
      InstanceType: t2.micro
      SecurityGroups:
        - !ImportValue SSHSecurityGroup
```

CloudFormation - Conditions

- Conditions são usados para criar Resources ou Outputs baseado em uma condição
- Conditions pode ser qualquer condição que você quiser. Alguns exemplos:
  - Environment (dev / test / prod)
  - AWS Region
  - Any Parameter Value
- Cada Condition pode referenciar outros Conditions, Parameter Value ou Mapping

CloudFormation - como definir um Conditions

- Funções que podem ser utilizadas (lógica)
  - Fn::And
  - Fn::Equals
  - Fn::If
  - Fn::Not
  - Fn::Or

```
Conditions:
  CreateProdResources: !Equals [ !Ref EnvType, prod ]
Resources:
  EC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", AMI]
  MountPoint:
    Type: "AWS::EC2::VolumeAttachment"
    Condition: CreateProdResources
```

CloudFormation - Usando Conditions

- Conditions podem ser aplicados
  - Resources
  - Output
  - etc

```
Resources:
  MountPoint:
    Type: "AWS::EC2::VolumeAttachment"
    Condition: CreateProdResources
```

CloudFormation - Funções que você precisa saber

- Fn::Ref
- Fn::GetAtt
- Fn::FindInMap
- Fn::ImportValue
- Fn::Join
- Fn::Sub
- Condition Function(Fn::If, Fn::Not, Fn::And, Fn::Equals, Fn::Or)

CloudFormation - Fn::Ref

- Fn::Ref pode ser usado para referenciar:
  - Parameters: retorna o valor do Parameter
  - Resources: retorna o Physical ID do Resource (Ex. EC2 ID)
- Pode-se usar no formato !Ref

```
Outputs:
    StackSSHSecurityGroup:
        Description: SSH Security Group para sua empresa
        Value: !Ref MinhaEmpresaSSHSecurityGroup
        Export:
          Name: SSHSecurityGroup
```

CloudFormation - Fn::GetAtt

- Atributos são anexados aos recursos que você cria
- Exemplo: Saber qual é o Availability Zone de uma Instância EC2

```
Rosource:
  EC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: ami-246eta7
      InstanceType: t2.micro
```

```
NewVolume:
  Type: "AWS::EC2::Volume"
  Condition: CreateProdResources
  Properties:
    Size: 100
    AvailabilityZone:
      !GetAtt EC2Instance.AvailabilityZone
```

CloudFormation - Acessando Valores Mapping

- Fn:FindInMap
  - Retorna o valor (value) de uma key
  - !FindInMap [MapName, TopLevelKey, SecondLevelKey]

```
Mappings:
  RegionMap:
    us-east-1:
      HVM64: "ami-0ff8a91507f77f867"
      HVMG2: "ami-0a584ac55a7631c0c"
    us-west-1:
      HVM64: "ami-0bdb828fd58c52235"
      HVMG2: "ami-066ee5fd4a9ef77f1"
    eu-west-1:
      HVM64: "ami-047bb4163c506cd98"
      HVMG2: "ami-31c2f645"
    ap-southeast-1:
      HVM64: "ami-08569b978cc4dfa10"
      HVMG2: "ami-0be9df32ae9f92309"
    ap-northeast-1:
      HVM64: "ami-06cd52961ce9f0d85"
      HVMG2: "ami-053cdd503598e4a9d"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap
        - RegionMap
        - !Ref 'AWS::Region'
        - HVM64
      InstanceType: m1.small
```

CloudFormation - Fn::ImportValue

- Importa Valores que são exportados em outros Templates
- Para isso use Fn::ImportValue

```yaml
Resources:
  MySecureInstance:
    Type: AWS::EC2::Instance
    Properties:
      AvailabilityZone: sa-east-1
      ImageId: ami-b5erdrt4
      InstanceType: t2.micro
      SecurityGroups:
        - !ImportValue SSHSecurityGroup
```

CloudFormation - Fn::Join

- Join Values com Delimiter
  - !Join [delimiter, [comma-delimited list of values]]
- Cria “a:b:c”
  - !Join [“:”, [a, b, c]]

CloudFormation - Fn::Sub

- Fn:Sub ou !Sub é usado para substituir variáveis de um texto. Permite personalizar seu Template
- Exemplo: É possível combinar !Sub com References ou AWS Pseudo Variables
- String deve conter ${VariableName} e irá substituí-las

```
!Sub
  - String
  - { Var1Name: Var1Value, Var2Name: Var2Value}
```

CloudFormation - Conditions

- Funções que podem ser utilizadas (lógica)
  - Fn::And
  - Fn::Equals
  - Fn::If
  - Fn::Not
  - Fn::Or

```
Conditions:
  CreateProdResources: !Equals [ !Ref EnvType, prod ]
Resources:
  EC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", AMI]
  MountPoint:
    Type: "AWS::EC2::VolumeAttachment"
    Condition: CreateProdResources
```

CloudFormation - Rollback

- Stack Creation Fails:
  - Default: tudo volta como era antes (rollback). Tudo é registrado nos logs
  - Opção para desabilitar rollback e investigar o problema
- Stack Update Fails:
  - Stack automaticamente volta para o estágio anterior
  - Possibilidade de ver logs e saber o que aconteceu de errado

CloudFormation - FAQ

- Posso criar uma quantidade dinâmica de recursos
  - Não. não é possível. Resources devem ser declarados.
- Todos os serviços da AWS são suportados
  - Quase todos
  - É possível contornar essa limitação usando Lambda Custom Resources

AWS Monitoring, Troubleshooting e Audit

- Monitorar, Identificar Problemas e Auditoria
  - CloudWatch
  - X-Ray
  - CloudTrail

Monitoramento

- Já aprendemos como distribuir aplicações:
  - De forma segura
  - Automaticamente
  - Usando Infraestrutura como código
  - Usando os melhores componentes da AWS
- Aplicação foi distribuída e os usuários não ligam para como isso foi feito
- Usuário só querem saber que a aplicação funciona
  - Latência: Aumenta com o passar do tempo
  - Fora do ar: Experiência do usuário deve ser a melhor possível
  - Usuários reclamam da aplicação
  - Troubleshooting e prevenção
- Monitoramento Interno
  - Podemos antecipar uma falha
  - Performance e Custo
  - Trends
  - Aprender e Otimizar

Monitoramento na AWS

- AWS CloudWatch:
  - Metrics: Coleta e rastreia Key Metrics
  - Logs: Coleta, monitora, analiza e armazena arquivos log
  - Events: Envia notificações quando certos eventos acontecem no ambiente AWS
  - Alarmes: Reagem em tempo real a Metrics e Eventos
- AWS X-Ray:
  - Troubleshooting da aplicação para Performance e erros
  - Disbritubed Tracing of Microservices
- AWS CloudTrail:
  - Monitora API calls
  - Registra todas as mudanças nos recursos AWS feita pelos usuários

Monitoramento na AWS

- AWS CloudWatch:
  - Metrics: Coleta e rastreia Key Metrics
  - Logs: Coleta, monitora, analiza e armazena arquivos log
  - Events: Envia notificações quando certos eventos acontecem no ambiente AWS
  - Alarmes: Reagem em tempo real a Metrics e Eventos
- AWS X-Ray:
  - Troubleshooting da aplicação para Performance e erros
  - Disbritubed Tracing of Microservices
- AWS CloudTrail:
  - Monitora API calls
  - Registra todas as mudanças nos recursos AWS feita pelos usuários

AWS CloudWatch Metrics

- CloudWatch Disponibiliza Metrics para todos os serviços AWS
- Metric é uma variável monitorada (Utilização da CPU, Tráfego entrando na rede)
- Metrics pertencem ao namespaces
- Dimension é um atributo do Metric (Instance id, Environment, etc…)
- Até 10 Dimensions por Metric
- Metrics tem timestamps
- Possibilidade de criar um CloudWatch Dashboard com os Metrics

AWS CloudWatch EC2 Detailed Monitoring

- Metrics da Instância EC2 são capturados a cada 5 minutos
- Detailed Monitoring captura os dados a cada 1 minuto
- Use Detailed Monitoring se quiser uma resposta mais rápida do ASG
- AWS Free Tier permite que você tenha até 10 Detailed Monitoring Metrics
- Uso da memória RAM não é enviado para CloudWatch. Uso da memória RAM deve ser enviado de dentro da instância usando Custom Metric

AWS CloudWatch Custom Metrics

- Possibilidade de definir e enviar seu próprio Custom Metric para o CloudWatch
- Habilidade de usar Dimensions (attributes) para segmentar Metrics
  - Instance.id
  - Environment.name
- Metric Resolution:
  - Standard: 1 minuto
  - High Resolution: Até 1 segundo (StorageResolution API parameter) - Alto Custo
- Uso de API call PutMetricData
- Uso de Exponential Back Off em caso de Throttle Erros

AWS CloudWatch Alarms

- Alarmes são usados para acionar notificações para qualquer Metric
- Alarmes podem ser enviados para Auto Scaling, EC2 Actions, SNS notifications
- Várias Opções (sampling, %, max, min, etc…)
- Alarm States:
  - OK
  - INSUFFICIENT_DATA
  - ALARM
- Period:
  - Length of Time em segundos para capturar Metric
  - High Resolution Custom Metric: pode escolher somente 10 segundos ou 30 segundos

AWS CloudWatch Logs

- Aplicação pode enviar logs para o CloudWatch usando SDK
- CloudWatch pode coletar logs:
  - Elastic BeanStalk
  - ECS
  - AWS Lambda
  - VPC Flow Logs
  - API Gateway
  - CloudTrail
  - CloudWatch
  - Route53
- CloudWatch Logs podem ser enviados para
  - Batch Exporter para S3 Archival
  - Stream para ElasticSearch cluster para futura análise

AWS CloudWatch Logs

- CloudWatch Logs pode usar Filter Expressions
- Arquitetura de armazenamento de logs
  - Log Groups: Normalmente representa uma aplicação. Nome arbitrário
  - Log Stream: Instância dentro da aplicação / log files / containers
- Pode-se definir Log Expiration Policies (Never expire, 30 days)
- Usando AWS CLI podemos filtrar CloudWatch Logs
- Para enviar logs para CloudWatch, verifique que as permissões IAM estão corretas
- Segurança: Criptografia dos logs usando KMS a nível de Group

AWS CloudWatch Events

- Schedule: Cron jobs
- Event Pattern: Event Rules para reagir para um serviço fazendo alguma coisa
    - Exemplo: Mudança no Codepipeline State
- Trigger (gatilho) para Lambda Function, SQS/SNS/Kinesis Messages
- CloudWatch Event cria um arquivo JSON para dar informações sobre a mudança

AWS X-Ray

- Debugging em Produção:
  - Teste localmente
  - Adiciona log statements em todos os lugares
  - Re-deploy em produção
- Formato dos logs é diferente entre as plataformas usando CloudWatch (difícil de fazer análise)
- Debugging
- Sem visão comum de toda a arquitetura

AWS X-Ray

- Análise Visual da Aplicação

Vantagens do AWS X-Ray

- Troubleshooting Performance (Gargalo)
- Entendendo dependencies em arquitetura microservice
- Identificar Problemas com Pinpoint Service
- Revisar comportamento das requisições
- Encontrar erros e exceções
- Identificar usuários impactados

AWS X-Ray - Compatibilidade

- AWS Lambda
- Elastic BeanStalk
- ECS
- ELB
- API Gateway
- Instância EC2 ou qualquer outro servidor, inclusive on premise

AWS X-Ray - Utiliza Tracing

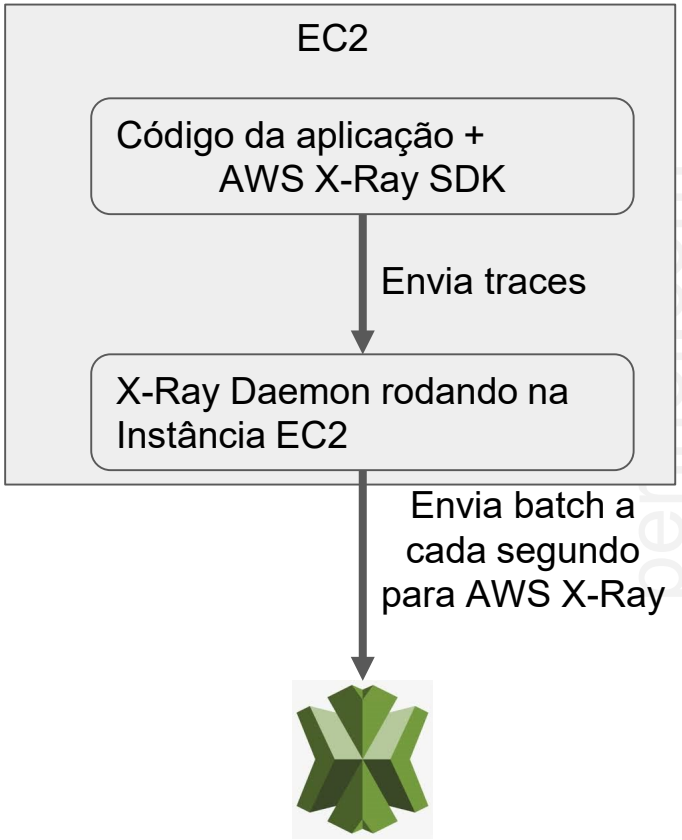
- Tracing é uma forma de monitorar um Request de um ponto ao outro
- Cada componente que trata um Request adiciona seu próprio Trace
- Trace é feito de segmentos (mais sub segmentos)
- Annotations pode adicionar Traces para incluir informações extras
- Capacidade Trace:
  - A cada Request
  - Sample Request (Exemplo: porcentagem ou taxa por minuto)
- Segurança X-Ray:
  - Autorização IAM
  - KMS para criptografia em repouso

Como habilitar AWS X-Ray

- No seu código (Java, Python, Go, Node.js, .NET) deve-se importar AWS X-Ray SDK
  - Pouca alteração no código
  - O SDK da aplicação captura:
    - Calls para Serviços AWS
    - Request HTTP/HTTPS
    - Database Calls (MySQL, PostgreSQL, DynamoDB)
    - Queue Calls (SQL)

Como habilitar AWS X-Ray

- Instalando X-ray daemon ou habilitando X-Ray AWS Integration
  - X-Ray daemon funciona como baixo nível UDP packet interceptor (Linux / Windows / Mac)
  - AWS Lambda / outros serviços AWS que já rodam X-Ray daemon
  - Cada aplicação deve ter permissão IAM para escrever dados no X-Ray



X-Ray Magic

- X-Ray coleta dados de vários serviços
- Service Map utiliza Segments e Traces
- X-Ray é gráfico, fácil entendimento

AWS X-Ray Troubleshooting

- Se X-Ray não estiver funcionando na instância EC2
  - Verifique se EC2 IAM Role the a correta permissão
  - Verifique se EC2 está rodando X-Ray Daemon
- Habilitar AWS Lambda:
  - Verifique se tem IAM execution role com Policy correta (AWSX-RayWriteOnlyAccess)
  - Verifique se X-Ray foi importado no código

AWS CloudTrail

- Governance, Compliance and Audit para sua conta AWS
- CloudTrail é habilitado por padrão
- Histórico de eventos e API calls feito na sua conta AWS pode ser recuperados utilizando
  - Console
  - SDK
  - CLI
  - AWS Services
- CloudTrail pode enviar logs para CloudWatch logs
- Se um recurso é deletado na AWS, procure o motivo no CloudTrail primeiro

Messaging

- Aplicações precisam comunicar com outras aplicações
- Existem duas formas de realizar essa comunicação entre aplicações

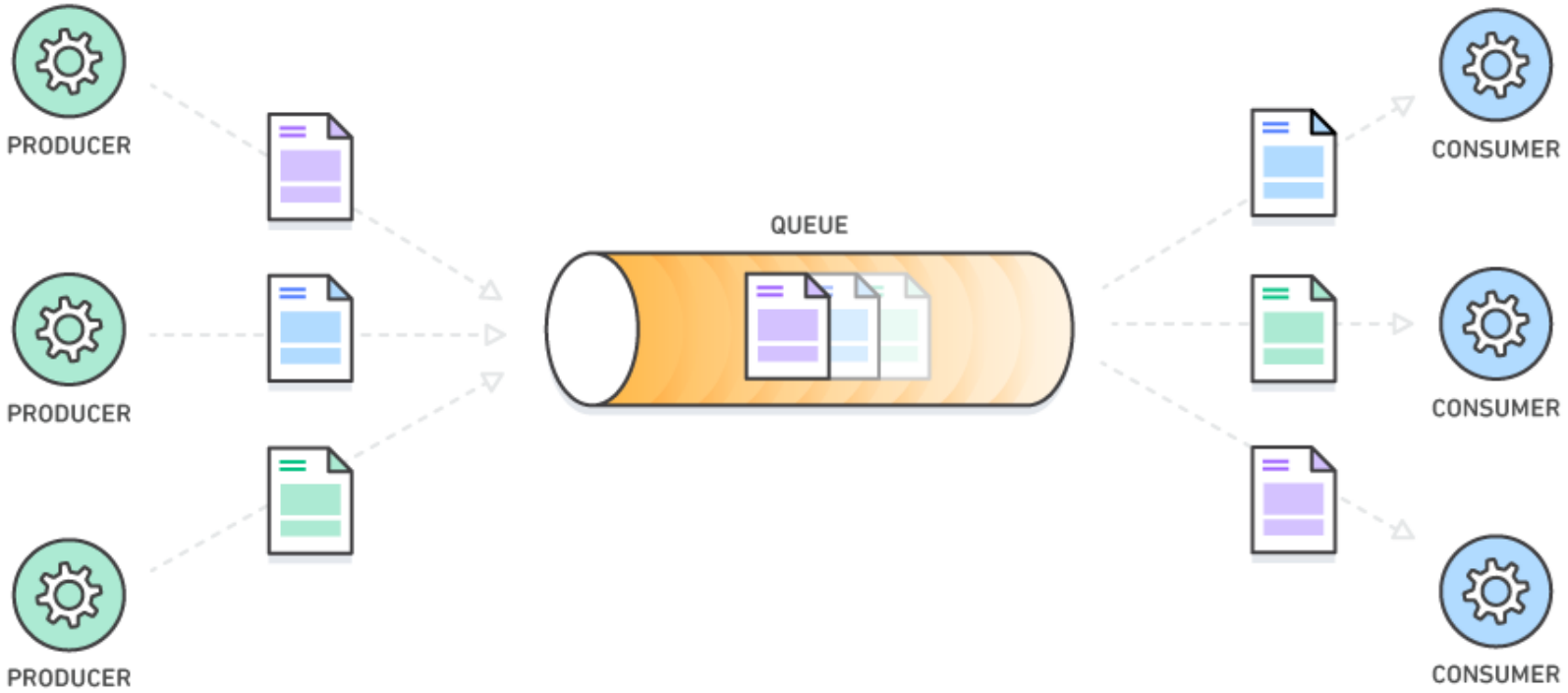
Synchronous - Aplicação-Aplicação

Asynchronous - Aplicação-Queue-Aplicação

Messaging

- Synchronous pode ser problemático entre aplicações quando ocorrem picos de tráfego
- Se sua aplicação normalmente encode 100 vídeos por hora, receber 10000 videos de uma vez?
- Nesse caso é melhor dissociar sua aplicação:
  - Usando SQS: modelo queue
  - Usando SNS: modelo pub/sub
  - Usando Kinesis: modelo streaming em tempo real
- Esses serviços podem escalonar independente da aplicação

AWS SQS

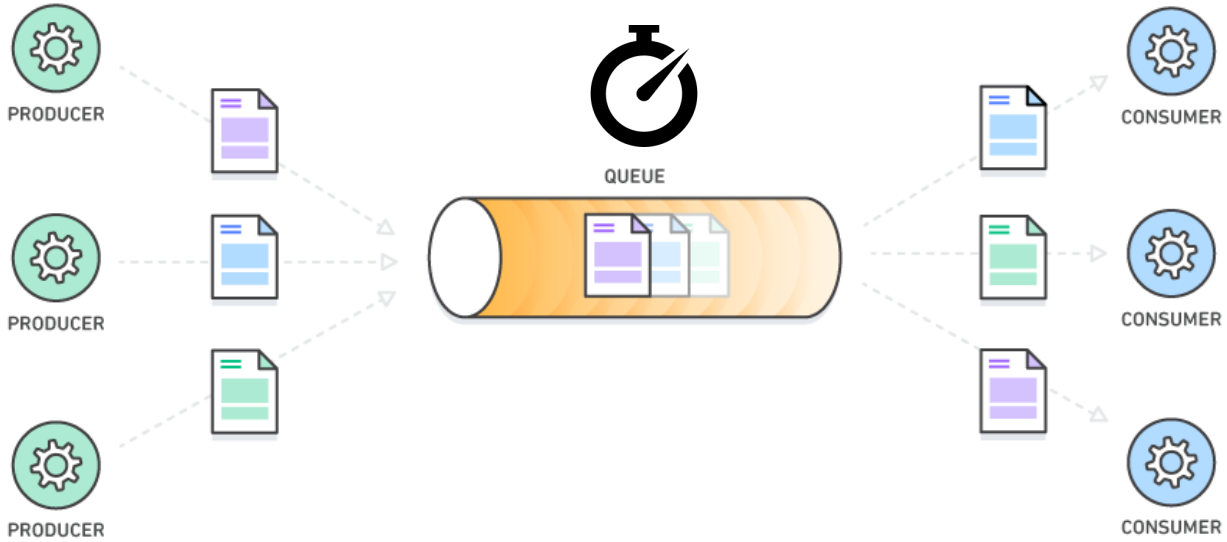


AWS SQS - Standard Queue

- Gerenciado pela AWS
- Escalona de 1 mensagem por segundo até 10.000 por segundo
- Default Retention of message (Armazenamento de mensagem): 4 dias (máx 14)
- Sem limite de quantas mensagens podem ficar no queue
- Latência menor que 10ms
- Escalonamento horizontal de consumidores
- Pode acontecer de ter mensagem duplicada
- Pode acontecer das mensagens serem consumidas fora de ordem (Best effort ordering)
- Tamanho máximo das mensagens é 256KB

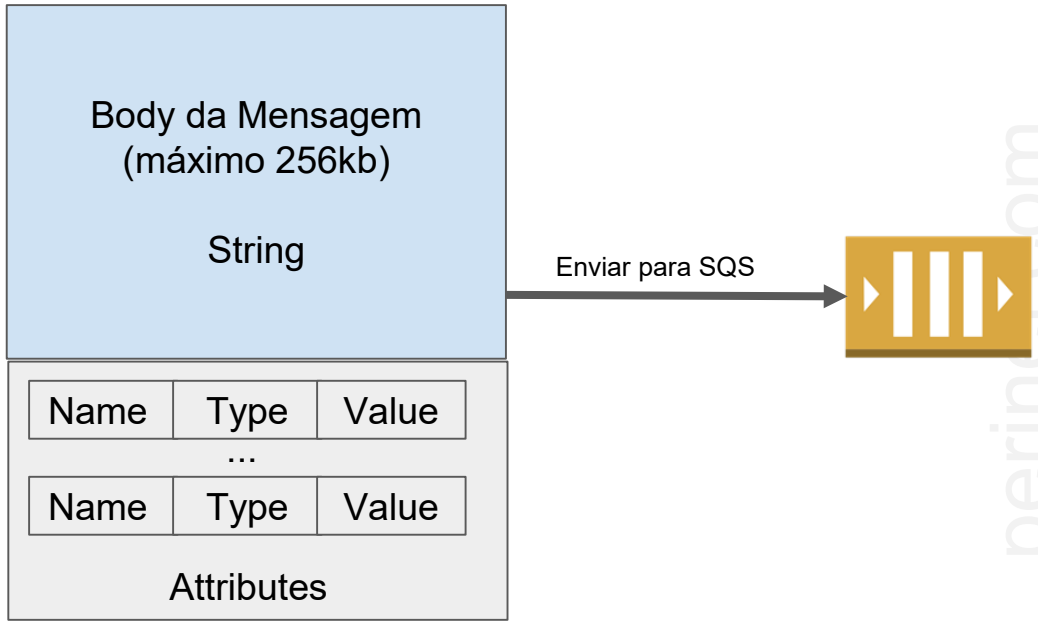
AWS SQS - Delay Queue

- Delay a message (consumidores não veem a mensagem imediatamente) máximo de 15 minutos
- Default é 0 segundos (mensagem fica disponível imediatamente)
- Default pode ser ajustado no queue level
- Default pode ser alterado usando DelaySeconds Parameters



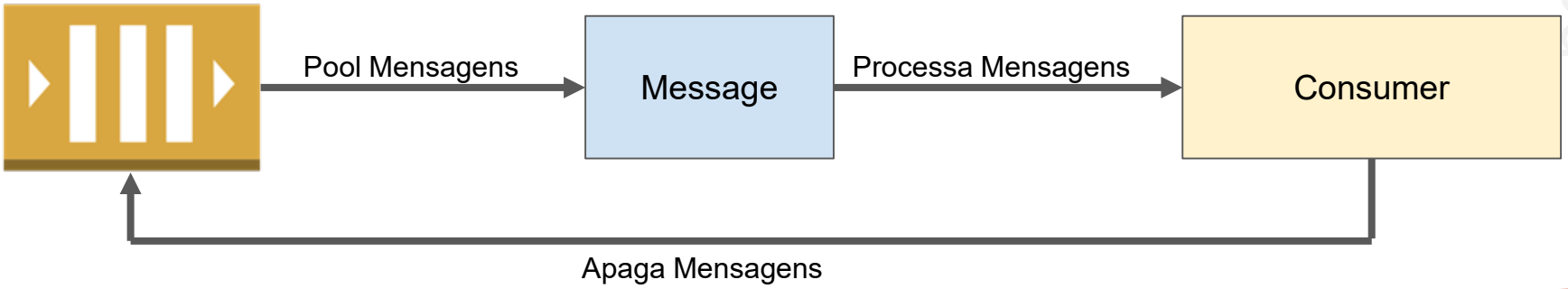
AWS SQS - Producing Messages

- Definição do Body
- Adicionar Messages Attributes (metadata é opcional)
- Opção Delay Delivery
- Retorna:
  - Message Identified
  - MD5 hash do Body



AWS SQS - Consuming Messages

- Consumers (Consumidores)
- Poll SQS para mensagens (recebe no máximo 10 mensagens por vez)
- Processa mensagens dentro do Visibility Timeout
- Deleta as mensagens usando Message ID e Receipt Handle

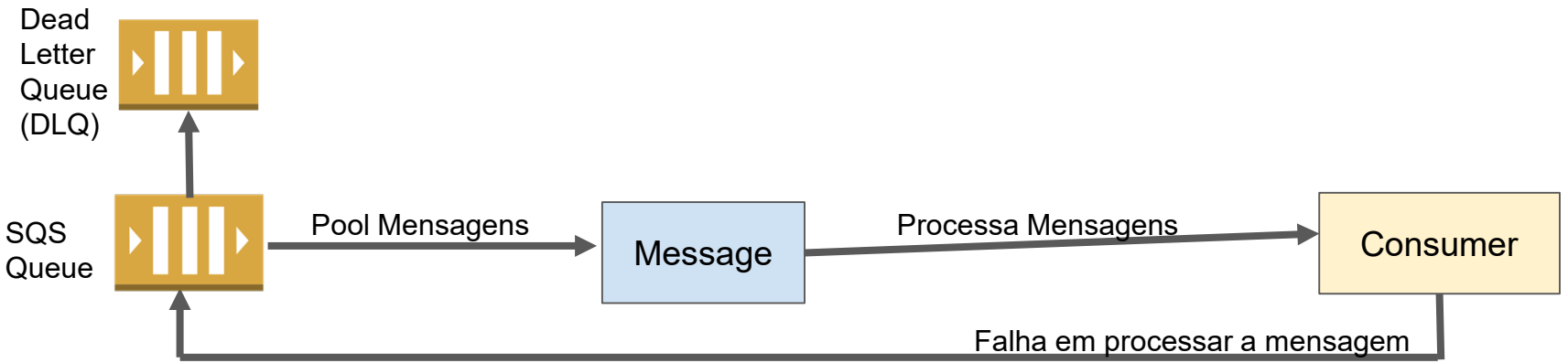


AWS SQS - Visibility Timeout

- Quando o consumidor Polls a Message do Queue, a mensagem fica invisível para outros consumidores por um período de tempo definido (Visibility Timeout)
  - Ajustável entre 0 e 12 horas (default: 30 segundos)
  - Se valor for muito alto (10 minutos) e o consumidor falhar ao processar a mensagem, é necessário esperar muito tempo para que a mensagem esteja disponível novamente.
  - Se valor for muito baixo (1 segundos) e o consumidor precisar de mais tempo para processar a mensagem, outro consumidor irá receber a mesma mensagem e a mensagem será processada 2 vezes
- ChangeMessageVisibility API muda a visibilidade enquanto processa a mensagem
- DeleteMessage API para avisar SQS que a mensagem foi processada

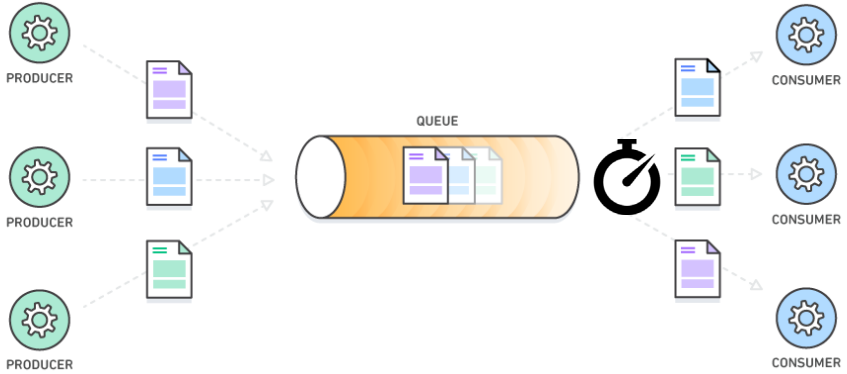
AWS SQS - Dead Letter Queue

- Se o consumidor falhar ao processar a mensagem dentro do Visibility Timeout, a mensagem volta para o Queue
- Podemos escolher quantas vezes uma mensagem pode voltar para o Queue - Redrive Policy
- Quando esse número é atingido, a mensagem vai para Dead Letter Queue (DLQ)
- Depois de criar DLQ, e apontar o SQS Queue para o Dead Letter Queue
- Processe a mensagem no DLQ antes que ela expire.



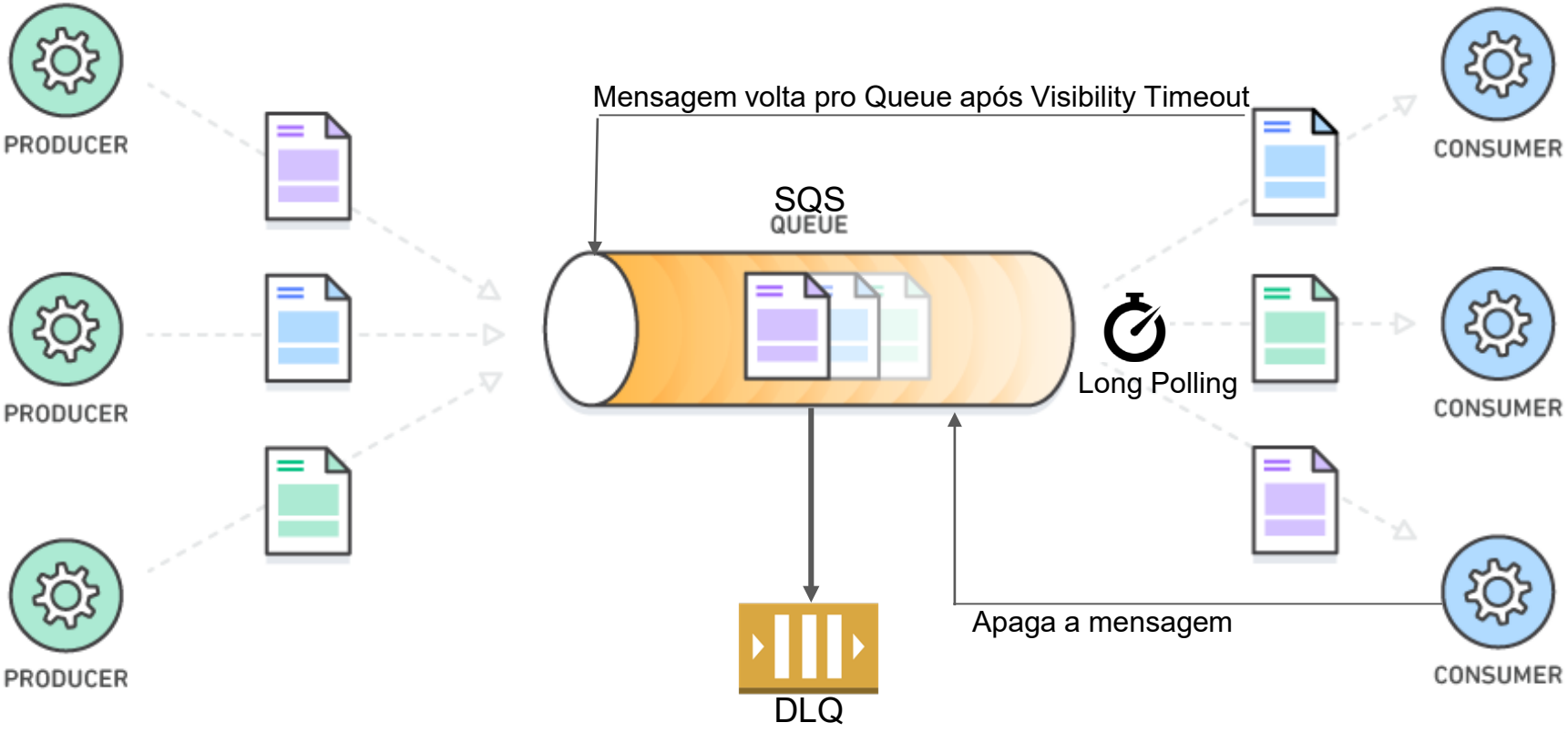
AWS SQS - Long Polling

- Quando o consumidor requisitar a mensagem do Queue, tem a opção de esperar por mensagens chegarem no Queue, se o Queue estiver vazio
- Essa técnica é chamada Long Polling
- LongPolling diminue o numero de chamadas a SQS API, e aumenta a eficiência da sua aplicação
- O Wait Time (tempo de espera) pode ser ajustado entre 1 segundo e 20 segundos (recomendado para a maioria dos casos)
- Long Polling pode ser habilitado no Nível Queue ou Nível API usando WaitTimeSeconds



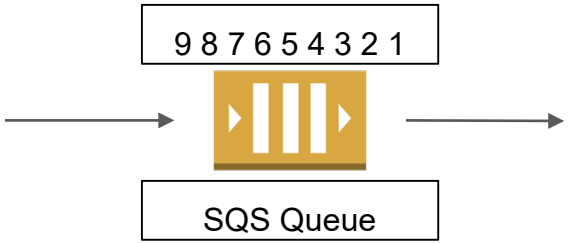
Consumidor espera até 20 segundos para que uma mensagem aparece no Queue antes de fazer uma nova chamada ao API

AWS SQS - Diagrama



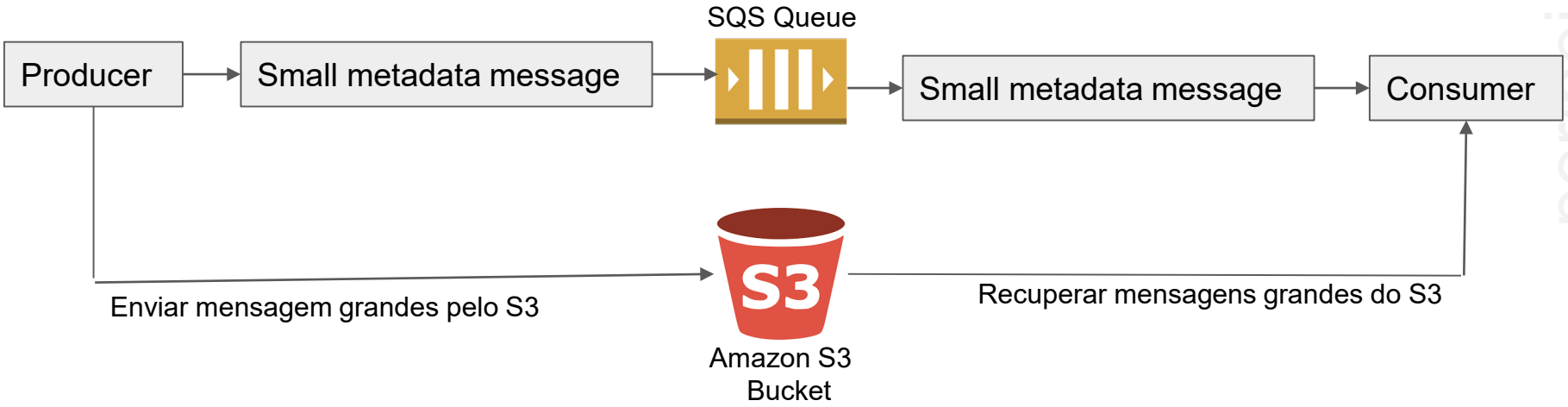
AWS SQS - FIFO Queue

- First In - First out – primeiro a entrar é o primeiro a sair (Não disponível em todas as regiões)
- Nome do Queue deve terminar com .fifo
- Baixo throughput (Até 3,000 por segundo com Batching (300/s sem Batching)
- Mensagens são processadas em ordem pelos consumidores
- Mensagens são enviadas somente uma vez
- Sem Delay por Mensagem (somente Delay por Queue)
- Possibilidade de fazer de-duplication baseado em conteúdo
- 5-minutos interval de-duplication usando “Duplication ID”
- Message Groups:
  - Possibilita agrupar mensagens por FIFO ordering usando “Message GroupID”
  - Somente um Worker pode ser designado por Message Group para que a mensagens sejam processadas em ordem
  - Message group é somente um Tag extra na mensagem



AWS SQS - Extended Client

- Limite da Mensagem 256KB, Como enviar mensagem maiores?
- Usando SQS Extended Client (Java Library)



AWS SQS - Security (Segurança)

- Criptografia em trânsido usando HTTPS endpoint
- Pode usar SSE (Server Side Encryption) usando KMS
  - Pode usar CMK (Customer Master Key)
  - Pode ajustar Data Key Reuse Period entre 1 minuto e 24 horas
    - Baixo e KMS API será usado com frequência
    - Alto e KMS API será usado com menos frequência
  - SSE somente criptografa o Body (corpo), não o metadata (message ID, timestamp, attributes)
- IAM policy deve permitir o uso do SQS
- SQS Queue Access Policy
  - Controle granular sobre IP
  - Controle sobre o tempo de requisição
- Nenhum VPC endpoint deve ter acesso a internet para acessar SQS

AWS SQS - Importante Saber

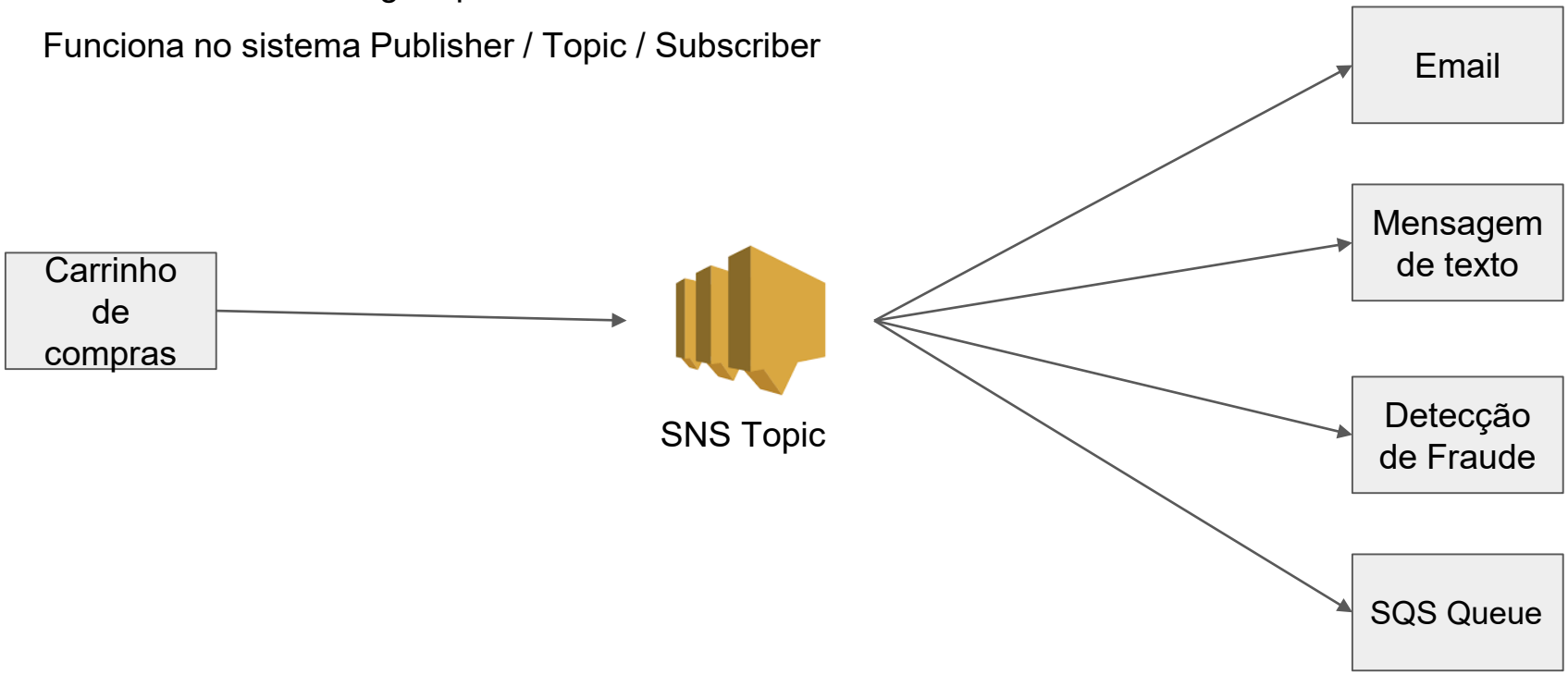
- CreateQueue
- DeleteQueue
- PurgeQueue
- SendMessage
- ReceiveMessage
- DeleteMessage
- ChangeMessageVisibility
- Batch API para SendMessage, DeleteMessage, ChangeMessageVisitility ajuda a reduzir seu custo

AWS SQS - Como usar

- Decouple a aplicação (Asynchronously)
- Buffer para escrever no Banco de Dados
- Tratar um quantidade grande de mensagens
- SQS pode sen integrado com Auto Scaling usando CloudWatch

AWS SNS

- Como enviar 1 mensagem para vários destinatários?
- Funciona no sistema Publisher / Topic / Subscriber



AWS SNS

- Producer somente envia mensagem para 1 SNS Topic
- Pode ter até 10.000.000 Subscribers por Topic
- Os Subscribers irão receber todas as mensagens enviadas(atenção para o novo recurso:filtrar mensagens)
- Limite de 100.000 Topics
- Subscribers podem ser:
  - SQS
  - HTTP / HTTPS
  - Lambda
  - Emails
  - SMS messages
  - Mobile Notifications

AWS SNS - Integração com vários produtos AWS

- Alguns serviços podem enviar dados diretamente para SNS
- CloudWatch (para alarmes)
- Notificação para Auto Scaling Groups
- Amazon S3 (bucket events)
- CloudFormation (Ex. Falhas no build)
- E muitos outros

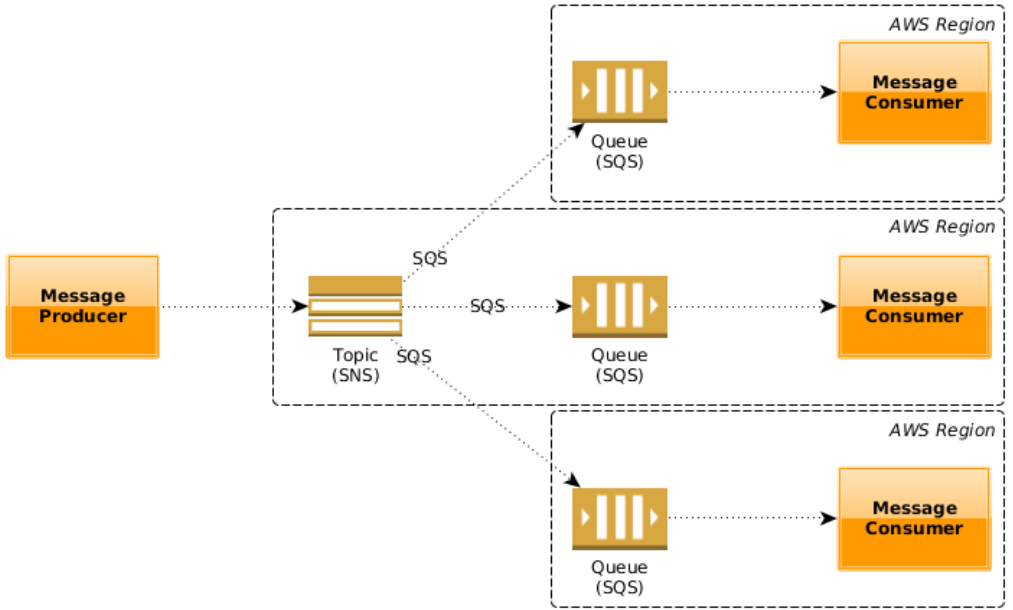
AWS SNS - Como Publicar

- Topic Publish (com seu servidor AWS, usando SDK)
  - Criar Topic
  - Criar Subscription
  - Publish o Topic

- Direct Publish (para mobile apps SDK)
  - Criar a plataforma da aplicação
  - Criar a plataforma Endpoint
  - Publicar para a plataforma Endpoint
  - Funciona com Google GCM, Apple APNS, Amazon ADM e outros

AWS SNS + SQS: Fan Out

- Publique uma vez usando SNS, receba em vários SQS
- Totalmente Dissociado
- Sem perda de dados
- Possibilidade de adicionar Subscriber a qualquer momento
- SQS permite processamento retardado
- SQS permite Retries of Work
- Pode ter vários Workers em um Queue e 1 só Worker no outro Queue

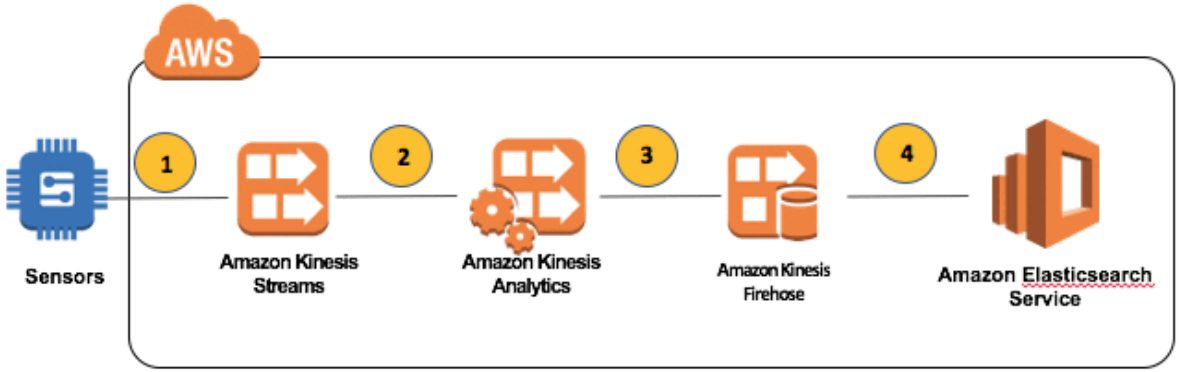


AWS Kinesis

- Kinesis é uma alternativa ao Apache Kafka
- Ideal para Log de Aplicação, Metrics, IoT, clickstream
- Ideal para Real-Time Big Data
- Ideal para Streaming Processing Frameworks (Spark, NiFi…)
- Dados são automaticamente replicados para 3 AZs

- Kinesis Streams: baixa latência, escalonável
- Kinesis Analytics: para análise em tempo real de streams usando SQL
- Kinesis Firehose: Carrega o stream no S3, Redshift, ElasticSearch...

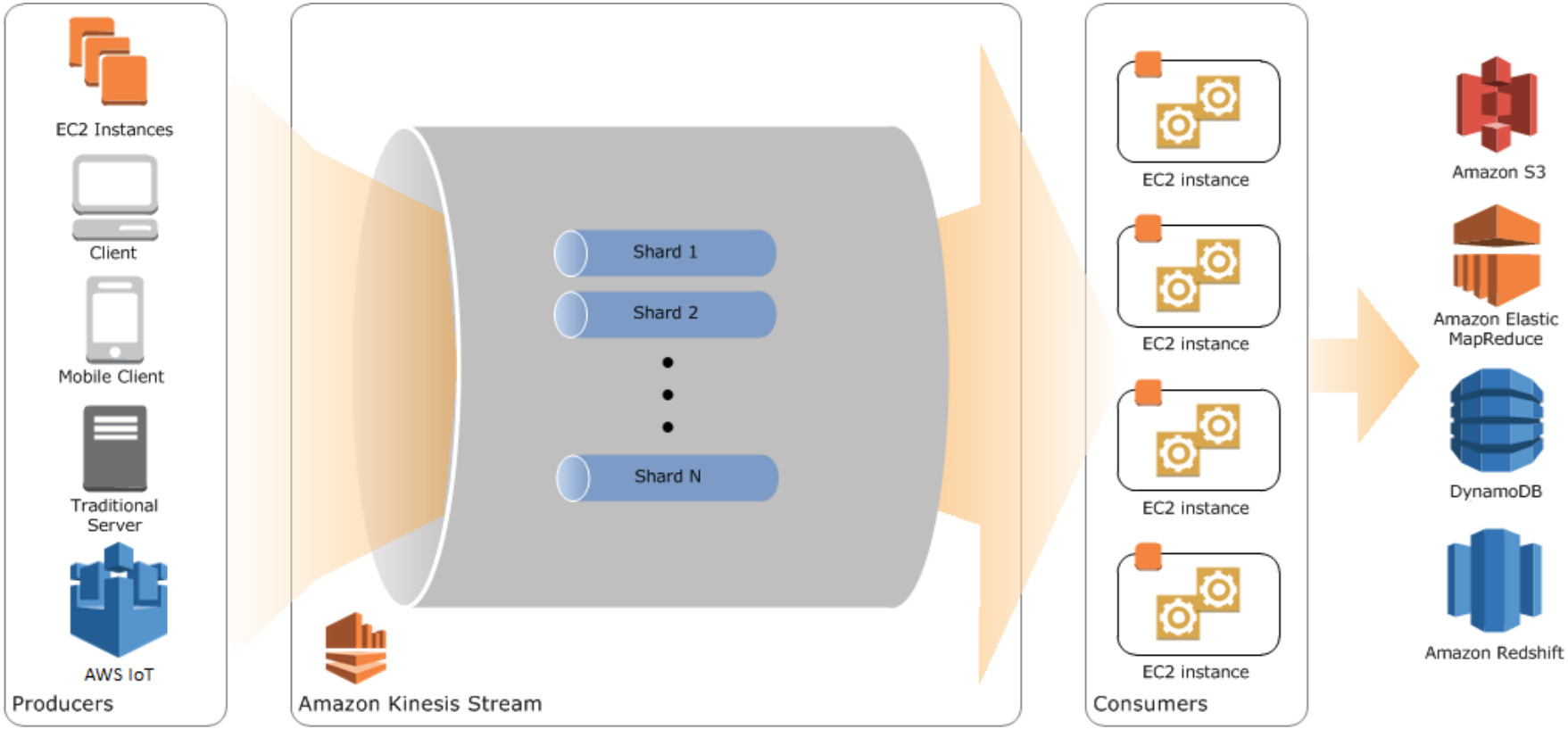
AWS Kinesis



AWS Kinesis Stream

- Stream são divididos em Shards / Partitions
- Retenção dos dados é de 1 dia mas pode ser estendido até 7 dias
- Habilidade de reprocessar / replay data
- Multiplas aplicações podem acessar o mesmo stream
- Processamento em tempo real e throughput escalonável
- Uma vez que os dados são inseridos no Kinesis, eles não pode ser apagados (Immutability)

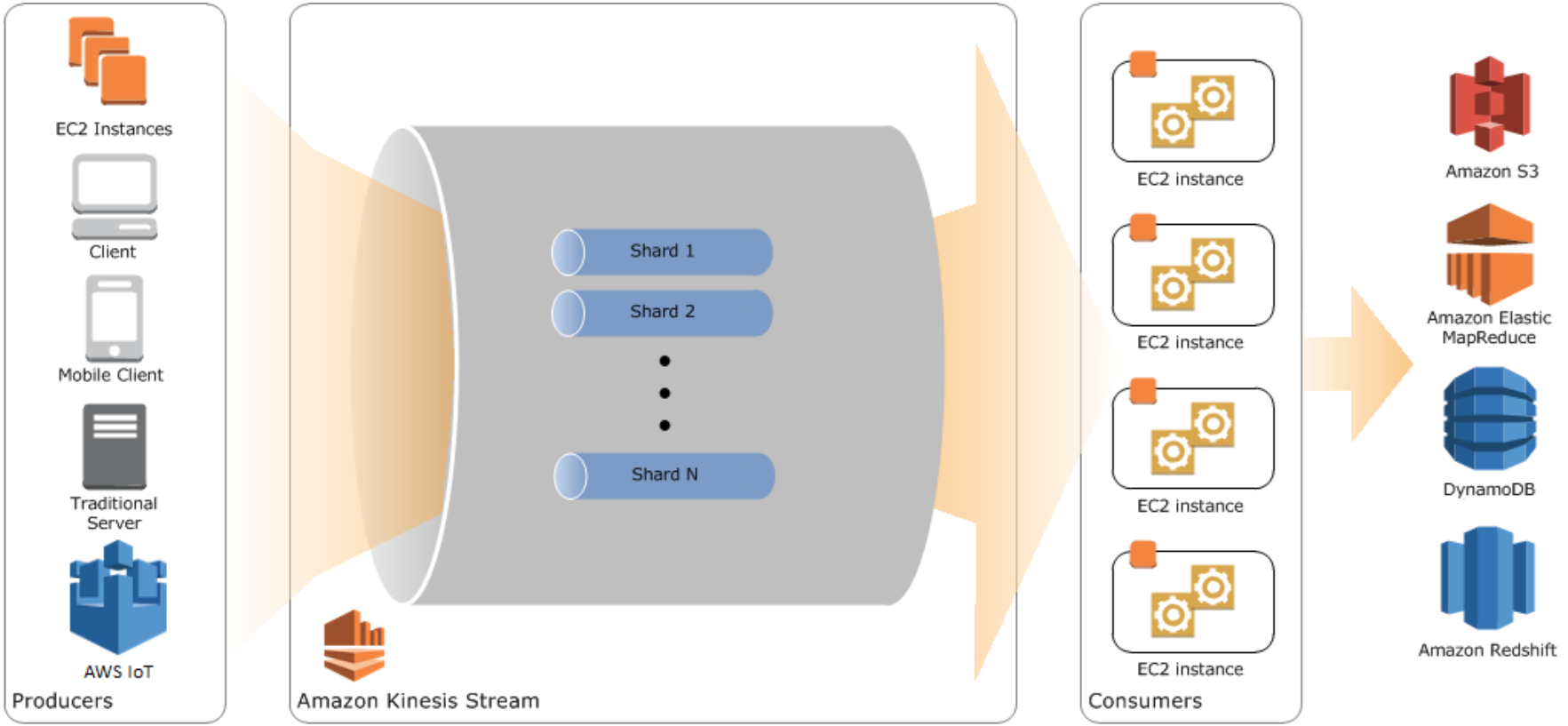
AWS Kinesis Stream



AWS Kinesis Stream Shards

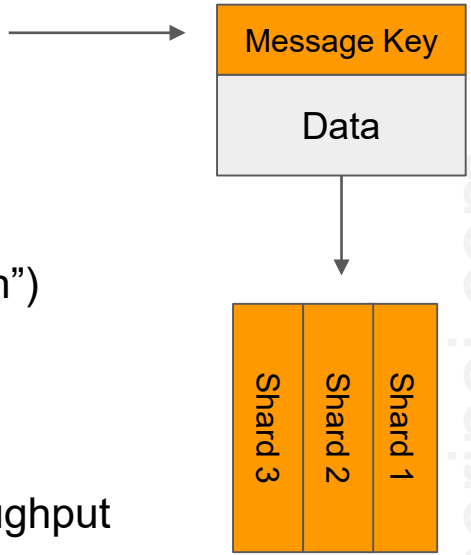
- 1 Stream é constituído por 1 ou mais Shards
- Capacidade por Shard
  - 1MB/s ou 1000 messages/s para escrita
  - 2MB/s para leitura
- Cobrado por Shard provisionado. Pode ter quantos Shards forem necessários
- Bathing ou por mensagem
- Número de Shards pode ser ajustável conforme necessidade
- Records são ordenados por Shard

AWS Kinesis Stream Shards



AWS Kinesis API - Put Records

- PutRecord API + Partition key é hashed para determinar Shard ID
- Mesma key vai para a mesma partition (Ajuda a ordenar as Keys)
- Mensagens enviadas recebem um número sequencial
- Escolha um Partition Key que seja distribuída (Previnir “hot partition”)
- Use user_id se tiver vários usuários
- Não use Estado_id se 90% dos usuários são da mesmo Estado
- Use Batching com PutRecords para reduzir custo e aumentar throughput
- ProvisionedThroughputExceeded se consumir acima do limite
- Podemos usar CLI, AWS SDK, or Producer Libraries de vários frameworks

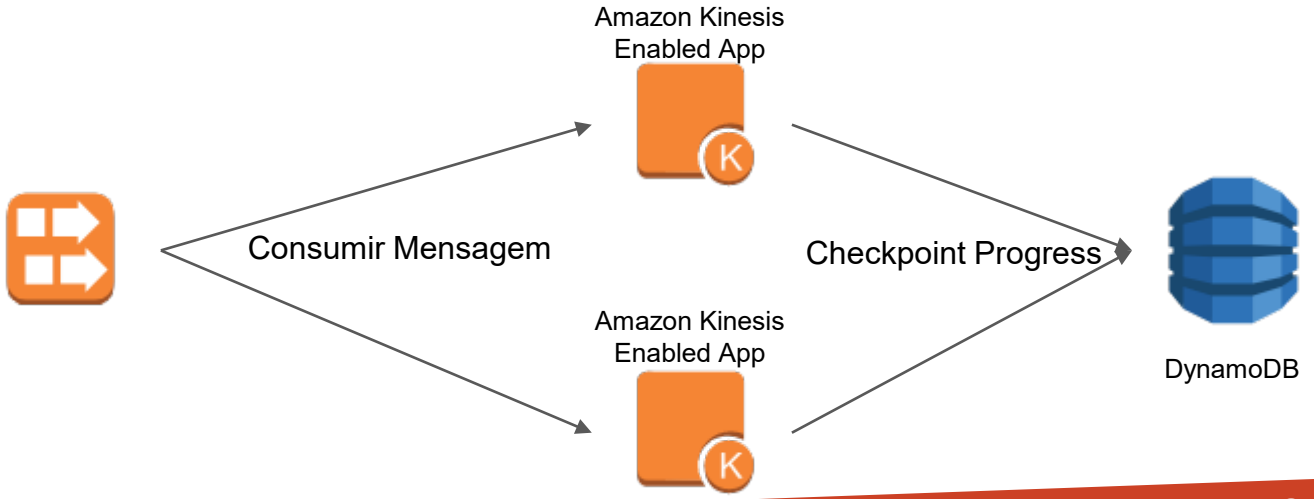


AWS Kinesis API - Exceptions

- ProvisionedThroughputExceeded Exceptions
- Ocorre quando é enviado mais dados do que o Shard suporta
- Verifique se Partition Key é distribuída de forma inteligente
- Solução:
  - Use Retries with backoff
  - Aumente o número de Shards
  - Escolha Partition Key de forma inteligente

AWS Kinesis API - Consumers

- Pode ser um consumidor normal (CLI, SDK e etc)
- Pode usar Kenesis Client Library (em Java, Node, Python, Ruby, .Net)
  - KCL usa DynamoDB para fazer checkpoint offsets
  - KCL usa DynamoDB para rastrear outros Workers e dividir o trabalho entre os Shards



AWS Kinesis Security

- Controle de Acesso / Autorização usando IAM Policies
- Criptografia em trânsito usando HTTPS endpoints
- Criptografia em repouso usando KMS
- Possibilidade de Criptografar/Decriptografar dados usando Client Side (mais complicado)
- VPC Endpoints disponível para Kinesis para acesso dentro do VPC

AWS Kinesis Data Analytics

- Análise em tempo real no Kinesis Streams usando SQL
- Kinesis Data Analytics:
  - Auto Scaling
  - Managed: Não é necessário provisionar servidores
  - Continuous: Tempo Real
- Cobrado somente pelo que é usado
- Pode criar Streams a partir de queries tempo real

AWS Kinesis Firehose

- Gerenciado pela AWS
- Quase Real Time (60 ms latency)
- Carrega dados dentro no Redshift / Amazon S3 / ElasticSearch / Splunk
- Escalonamento automático
- Suporta vários formatos de dados (cobrado por conversão)
- Cobrado pela quantidade de dados que passam pelo Firehose

SQS vs SNS vs Kinesis

SQS:

- Consumidor: pega dados
- Dados são deletados depois de serem consumidos
- Pode ter quantos Workers forem necessário
- Sem necessidade de provisionar capacidade
- Não garante ordem (exceção para FIFO)
- Retardo individual de mensagens

SNS:

- Envia dados para os subscribers
- Máximo de 10.000.000 subscribers
- Dados não persistem (se perdem se não forem entreguem
- Publisher / Subscribers
- Máximo de 100.000 tópicos
- Não há necessidade de provisionar capacidade
- Integrado com SQS para arquitetura fan-out

Kinesis:

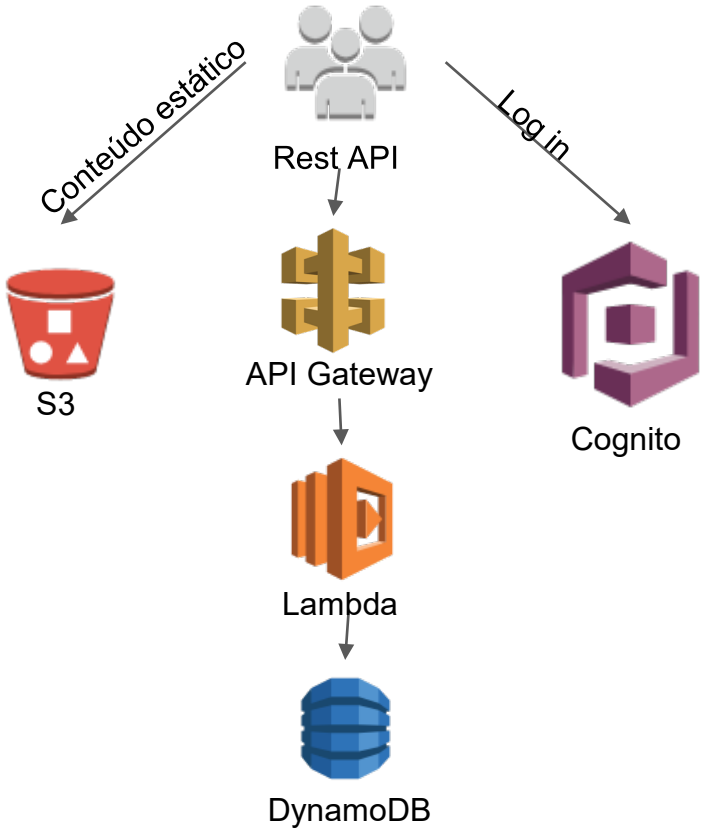
- Consumidor: pega dados
- Sem limite de número de consumidores
- Possibilidade de Replay dados
- Ideal para Real-Time Big Data, Análise e ETL
- Ordenamento ocorre a nível de Shard
- Dados expiram depois de X dias
- Deve-se provisionar capacidade

AWS Serverless

- O que é serverless?
- Serverless é um novo paradigma em que os desenvolvedores não precisam gerenciar servidores
- Simplesmente publique seu código
- Simplesmente publique sua Função
- Serverless começou com Lambda Function mas hoje em dia incluí, Banco de Dados, Mensagens, Armazenamento e outros
- Serverless não significa “Sem Servidores”. Significa somente que não gerenciamos ou provisionamos os servidores

AWS Serverless no AWS

- AWS Lambda e Step Functions
- DynamoDB
- AWS Cognito
- AWS API Gateway
- Amazon S3
- AWS SNS e SQS
- AWS Kinesis
- Aurora Serveless



AWS Lambda

EC2

- Virtual Server nas núvens
- Limitado pela RAM e CPU
- Servidor ligado 24 horas
- Para escalonar é necessário intervenção para adicionar mais servidores

---

Lambda

- Virtual Functions - sem servidor para gerenciar
- Limitado pelo tempo - máximo 15 minutos por requisição
- Executa somente quando é requisitado
- Escalonamento é automático

Benefício de usar AWS Lambda

- Preço simplificado:
  - Cobrado por requisição e tempo de processamento
  - Free Tier incluí 1.000.000 requisições e 400.000GB-segundos de tempo de processamento
  - Integração com vários outros serviços AWS
  - Integração com várias linguagens de programação
  - Monitoramento simplificado usando AWS CloudWatch
  - Até 3GB de RAM para executar suas funções
  - Aumento da RAM também aumenta performance da CPU e rede

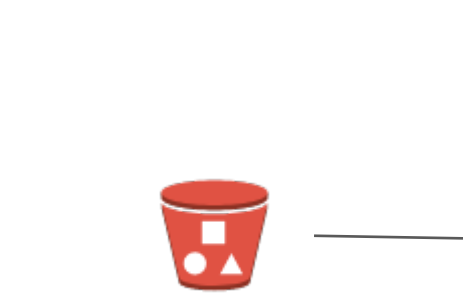
AWS Lambda: Linguagens suportadas

- Node.js (JavaScript)
- Python
- Java
- C# (.Net Core)
- Golang
- C# / Powershell

AWS Lambda: Serviços Integrados

- Principais Serviços Integrados
  - API Gateway
  - Kinesis
  - DynamoDB
  - S3
  - IoT
  - CloudWatch Events
  - CloudWatch Logs
  - SNS
  - Cognito
  - SQS

Exemplo de aplicação serverless



S3: Nova imagem

Lambda:
Função para criar
Thumbnail

S3: Novo Thumbnail

Metadata no DynamoDB:
Nome
Tamanho
Data

AWS Lambda: Preço

- Preço atualizado pode ser visto nesse link: https://aws.amazon.com/lambda/pricing/
- Cobrado por chamada a função:
- Primeira 1,000,000 requisições são grátis
- $0.20 por 1 milhão de requisições, depois disso ($0.0000002 por requisição)
- Cobrado também por tempo de execução (incremento de 100ms)
- 400.000 GB-segundos de tempo de computação por mês é grátis
- 400.000 segundos se a função utiliza 1GB RAM
- 3.200.000 segundos se a função utiliza 128 MB RAM
- Depois disso $1.00 por 600,000 GB-segundos
- Lambda em geral é um serviço muito barato. Por isso é tão popular

CONFIGURAÇÃO LAMBDA

- Timeout: Default é 3 segundos, máximo 15 minutos
- Environment Variables
- Memória alocada (128MB até 3GB)
- Pode ser colocada dentro de um VPC + Security Groups
- Deve-se associar IAM Role

AWS Lambda Concurrency e Throttling

- Concurrency: Até 1000 execuções podem acontecer ao mesmo tempo. Pode ser aumentado através de ticket ao Suporte AWS
- Possível ajustar “reserved concurrency”
- Cada chamada além do Concurrency Limit dispara um Throttle
- Comportamento do Throttle:
  - Synchronous Invocation: Retorna ThrottleError 429
  - Asynchronous Invocation: Tenta novamente e depois envia para DLQ

AWS Lambda Retries e DLQ

- Se a função invocada de forma assíncrona falhar, será feito uma nova tentativa
- Depois da segunda tentativa, será enviado para Dead Letter Queue
- DLQ pode ser SNS Topic ou SQS Queue
- O payload original é enviado para DLQ
- Essa é uma forma eficiente de detectar erros com sua função em produção, sem alterar o código
- Verifique se IAM execution Role está correta

AWS Lambda Logging, Monitoring e Tracing

- CloudWatch:
  - AWS Lambda logs são armazenadas no AWS CloudWatch Logs
  - AWS Lambda Metrics são exibidos no AWS CloudWatch Metrics
  - Verifique que sua Função Lambda tem IAM Role com Policy que permite escreve no CloudWatch
- X-Ray:
  - É Possível usar Trace com X-Ray
  - Habilitado na configuração Lambda (Executa X-Ray daemon)
  - Utilize AWS SDK no seu código
  - Certifique-se que Lambda Function tem a correta IAM execution Role

Limites do AWS Lambda

- Execução
  - Memory Allocation: 128MB - 3008MB (Incremento de 64MB)
  - Máximo tempo de execução: 15 minutos
  - Capacidade do disco (pasta /tmp) 512MB
  - Execuções simultâneas: 1000
- Deployment
  - Tamanho do arquivo Lambda Function: 50MB no formato .ZIP
  - Código + dependencies não podem ultrapassar 250MB
  - Pode utilizar a pasta /tmp para carregar outros arquivos no startup
  - Tamanho das Environment Variables: 4KB

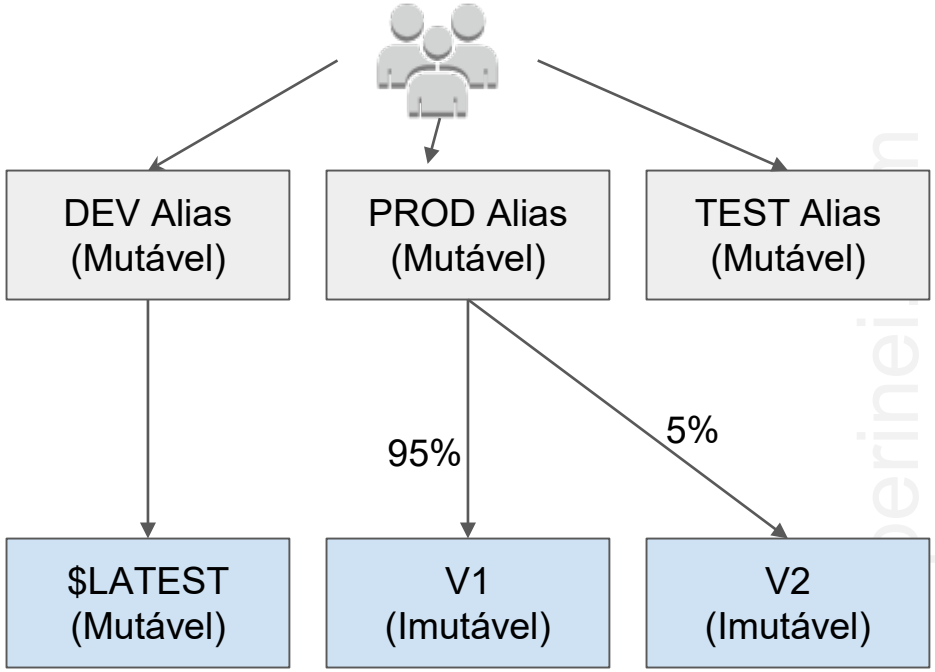
AWS Lambda Version

- Quando se trabalha com Lambda Function, trabalhos com $LATEST
- Quando estamos prontos para publicar um Lambda Function, criamos um Version
- Versions são imutáveis
- Versions: número da versão incrementa a cada nova versão
- Verson recebem seu próprio ARN (Amazon Resource Name)
- Version = código + configuração (nada pode ser mudado)
- Cada versão da Lambda Funciton pode ser acessada

$LATEST (editável)	V1 (não editável)	V2 (não editável)

AWS Lambda Aliases

- Aliases são “pointers” para versões da Lambda Function
- Nós podemos definir “Dev”, “Test”, “Prod” aliases e apontar para versões diferentes da Lambda Function
- Aliases são mutáveis
- Aliases permitem Blue / Green deployment usando weights para Lambda Function
- Aliases Habilitam configurações estáveis de triggers (gatilho) de eventos e destinos
- Aliases recebem seu próprio ARN



Lambda - Melhores Práticas

- Faça o trabalho pesado fora da função Handler
  - Conectar ao banco de dados
  - Inicializar AWS SDK
  - Pull Dependencies e banco de dados
- Utilize Environment Variables para:
  - Database Connection Strings, S3 bucket… não coloque esses valores diretamente no seu código
  - Senhas e outros dados sensíveis devem ser criptografados usando KMS
- Minimize seu o tamanho do Deployment package as necessidades da sua aplicação
  - Crie Microservices. Cada Lambda Function executa somente uma função
  - Não esqueça os Limites AWS Lambda
- Nunca faça uma Lambda Function chamar ela própria
- Não coloque sua Lambda Function dentro de um VPC, só se for extremamente necessário

Arquitetura Tradicional

- Aplicações Tradicionais utilizam Banco de Dados RDBMS
- Esses Bancos de Dados Utilizam a linguagem SQL Query
- Regras rígidas de como os dados devem ser modelados
- Possibilidade de fazer Join, Aggregations e Computations
- Escalonamento Vertical (Adicionar mais CPU, RAM, IO)

Banco de Dados NoSQL

- NoSQL significa Not Only SQL
- São Banco de Dados não relacionais
- São distribuído
- Não suporta Join
- Todos os dados usados no Query ficam em uma linha
- Não fazem Aggregation, exemplo: “SUM”
- Escalonamento Horizontal

- Quando comparamos SQL com NoSQL, um não é melhor do que o outro. A modelagem de dados é diferente e o modo de fazer query também é diferente

DynamoDB

- Gerenciado pela AWS, Alta Dispnibilidade com réplicas em 3 AZ
- Banco de Dados NoSQL
- Escalonável para quantidade muito grande de dados, banco de dados distribuído
- Milhões de requisições por segundo, trilhões de linhas, centenas de Terabytes de armazenamento
- Rápido e consistente em performance (Baixa latência)
- Integrado com IAM para segurança, autorização e administração
- Event Driven Programming com DynamoDB Streams
- Baixo Custo
- Auto Scaling

DynamoDB

- DynamoDB é feito de tabelas (table)
- Primary key é obrigatório e deve ser definido no momento que criamos a tabela
- Tabela pode ter um número infinito de itens (rows)
- Cada item tem Attributes (Pode ser adicionado a qualquer momento, pode ser nulo)
- Tamanho máximo do Item é 400KB
- Tipo de dados suportados:
    - Scalar Types: String, Number, Binary, Boolean, Null
    - Document Type: List, Map
    - Set Types: String Set, Number Set, Binary Set

DynamoDB - Primary Key

- Opção 1: Somente Partition Key (HASH)
- Partition Key deve ser única para cada item
- Partition Key deve ser diversificada para os dados serem distribuídos
- Example: user_id

Partition Key (unique)	Attribute	Attribute
user_id	nome	idade
sewr343	João	31
erio340	Maria	25

DynamoDB - Primary Key

- Opção 2: Partition Key + Sort Key
- A combinação deve ser única
- Dados são agrupados pela Partition Key
- Sort Key == Range Key
- Exemplo:

Primary Key		
Partition Key	Sort Key	Attribute
usuario_id	jogo_id	Resultado
sewr343	5324	ganhou
erio340	9674	perdeu

DynamoDB - Partition Key

- Vamos criar um banco de dados sobre Filmes
- Qual é a melhor Partition Key para maximizar a distribuição de dados?
  - filme_id
  - nome_do_produtor
  - protagonista
  - Idioma
- Filme_id é a melhor escolha

DynamoDB - Provisioned Throughput

- Tabela deve ter leitura e escrita provisionadas
- Read Capacity Units (RCU)
- Write Capacity Units (WCU)
- Opção de configurar auto-scaling para throughput
- Throughput pode temporariamente exceder o limite estabelecido usando “burst credit”
- Se o burst credit acabar, você receberá um “ProvisionedThroughputException”
- É Recomendado fazer um Exponential Back-Off Retry
- Novo Recurso: DynamoDB Read/Write capacity mode: On-demand

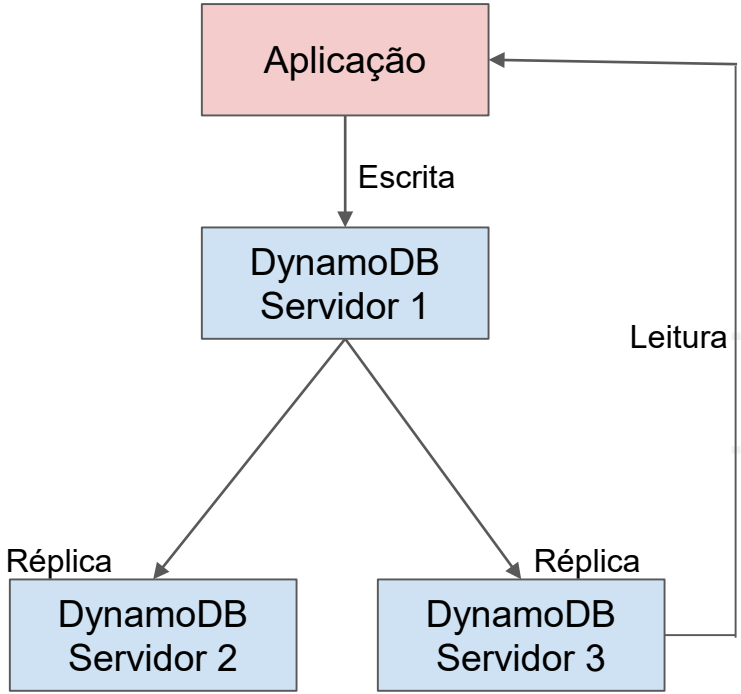
DynamoDB - Write Capacity Units (WCU)

- 1 Write Capacity Unit (WCU) representa uma escrita por segundo para 1 item até 1KB em tamanho
- Se o item for maior que 1KB, mais WCU serão consumidos

- Exemplo 1: escrever 10 itens de 2KB cada por segundo (10 * 2 = 20 WCU)
- Exemplo 2: escrever 6 itens de 4.5KB cada por segundo (5 * 6 = 30 WCU)
- Exemplo 3: escrever 120 itens de 2KB cada por minuto (2 * 120 / 60 = 4WCU)

Strongly Consistent Read vs Eventually Consistent Read

- Eventually Consistent Read: Se lermos um item logo após uma escrita, é possível lermos um valor inesperado devido a replicação
- Strongly Consistent Read: Se lermos um item depois de uma escrita, leremos o valor correto
- Por padrão (default), DynamoDB usa Eventually Consistent Reads, porém GetItem, Query e Scan tem um parametro chamado ConsistentRead que pode ser modificado para True (verdadeiro)



DynamoDB - Read Capacity Units (RCU)

- 1 Read Capacity Unit (RCU) representa 1 strongly consistent read por segundo ou 2 eventually consistent reads por segundo, para 1 item até 4KB
- Se o item for maior que 4KB, mais RCU serão consumidos

- Exemplo 1: 10 Strongly Consistent Read por segundo de 4KB cada
  - 10 * 4KB / 4KB = 10 RCU
- Exemplo 2: 16 Eventually Consistent Read por segundo de 12KB cada
  - (16 / 2) * (12 / 4) = 24RCU
- Exemplo 3: 10 Strongly Consistent Read por segundo de 6KB cada
  - 10 * 8 KB / 4 = 20 RCU (arrendondamos 6KB para 8KB)

DynamoDB - Partitions Internal

- Dados são divididos em partitions (partições)
- Partition Key passa por um algoritmo para saber para qual Partition deve ir
- Para computar o número de partições:
  - Capacidade: (TOTAL RCU / 3000) + (TOTAL WCU / 1000)
  - Tamanho: Tamanho Total / 10GB
  - Número de partições = CEILING(MAX(Capacity, Size))
- WCU and RCU são distribuídos igualmente entre as partições

DynamoDB - Throttling

- Se excedermos RCU or WCU recebemos um ProvisionedThroughputExceededException
- Razão:
  - Hot Key: um Partition Key sendo acessado muitas vezes
  - Hot Partitions:
  - Itens muito grandes: RCU e WCU dependem do tamanho do Item
- Solução:
  - Exponential Back-Off quando uma exceção é encontrada (recurso incluso no SDK)
  - Distribuir Partition Keys o máximo possível
  - Se for problema com RCU, podemos usar DynamoDB Accelerator(DAX)

DynamoDB - Escrevendo Dados

- PutItem - Escreve dados no DynamoDB (cria item ou substituí)
  - Consome WCU
- UpdateItem - Atualiza dados no DynamoDB (Atualiza parte dos atributos)
  - Possibilidade de usar Atomic Counters e incrementá-los
- Conditional Writes:
  - Escreve ou Atualiza somente se a condição for satisfeita
  - Ajuda com acesso simultâneo aos itens
  - Sem impacto na performance

DynamoDB - Apagando Dados

- DeleteItem
  - Deleta um item
  - Delete itens usando uma condicional
- DeleteTable
  - Apaga toda a tabela e seu conteúdo
  - Mais rápido que DeleteItem para apagar todos os itens

DynamoDB - Batching Writes

- BatchWriteItem
  - Máximo de 25 PutItem ou DeleteItem por Call
  - Máximo de 16MB de dados escritos
  - Máximo de 400KB de dados por Item
- Batching permite reduzir a latência, reduzindo o número de API Calls ao DynamoDB
- Operações são feitas em paralelo para melhor eficiência
- É possível que alguns item do Batch falhem, nesse caso é preciso tentar novamente usando Exponential Back-Off Algorithm

DynamoDB - Lendo Dados

- GetItem:
  - Leitura baseado no Primary Key
  - Primary Key = HASH ou HASH-RANGE
  - Eventually Consistent Read é o padrão
  - Opção do uso de Strongly Consistent Reads (consome mais RCU e pode demorar um pouco mais)
  - ProjectionExpression pode ser especificado para incluir somente certos atributos
- BatchGetItem:
  - Até 100 itens
  - Até 16MB
  - Itens são lidos em paralelo para minimizar latência

DynamoDB - Query

- Query retorna itens com base:
  - Valor do Partition Key (deve-se usar o operador “=”
  - Valor do SortKey (=, <, <=, >, >=, Between, Begin) - Opcional
  - FilterExpression para filtros (Filtro do lado Cliente)
- Retorna:
  - Até 1MB
  - Ou número de itens especificado em Limit
- Possível fazer paginação com o resultado
- Pode-se usar Query com Table, Local Secondary Index, ou Global Secondary Index

DynamoDB - Scan

- Utilizar a Scan e depois filtros é ineficiente
- Retorno de até 1MB. Usar paginação
- Consome muito RCU
- Pode-se diminuir o impacto usando Limit ou reduzir o tamanho do resultado
- Para melhor performance, use Parallel Scans:
  - Múltiplas Instâncias varrem múltiplas partições ao mesmo tempo
  - Aumenta o throughput e o RCU consumido
  - Limita o impacto do parallel Scans do mesmo jeito que é feito com Scans
- Pode-se usar ProjectionExpression + FilterExpression (Sem mudar o RCU)

DynamoDB - LSI (Local Secondary Index)

- Key Alternativo para tabela, local ao hash key
- Máximo de 5 Local Secondary Indexes por tabela
- Sort Key consiste de exatamente um Atributo
- O atributo que você escolher deve ser String, Number or Binary
- LSI deve ser definido na momento da criação da tabela

usuario_id	jogo_id	jogo_ts	Resultado	Duração
123jfu54	2546	“2019-04-29T15:36:52”	ganhou	5
fur85hr5	2145	“2019-04-03T17:46:50”	perdeu	25
r85uj9i4	9857	“2019-05-20T10:30:40”	perdeu	35

Partition Key

Sort Key

LSI

Atributos

DynamoDB - GSI (Global Secundary index)

- Para aumentar a velocidade dos Queries para não Key atributos, use Global Secondary Index
- GSI = Partition Key + Sort Key (opcional)
- Index é uma “nova” tabela e podemos projetar atributos a ela
  - A Partition Key e Sort Key da tabela original são sempre projetada (KEYS_ONLY)
  - Pode-se especificar atributos para projetar (INCLUDE)
  - Pode-se usar todos os atributos da tabela principal (ALL)
- Deve-se definir RCU / WCU para o índice
- Possibilidade de adicionar / modificar GSI (not LSI)

DynamoDB - GSI (Global Secondary Index)

usuario_id	jogo_id	jogo_ts	Resultado	Duração
123jfu54	2546	“2019-04-29T15:36:52”	ganhou	5
fur85hr5	2145	“2019-04-03T17:46:50”	perdeu	25

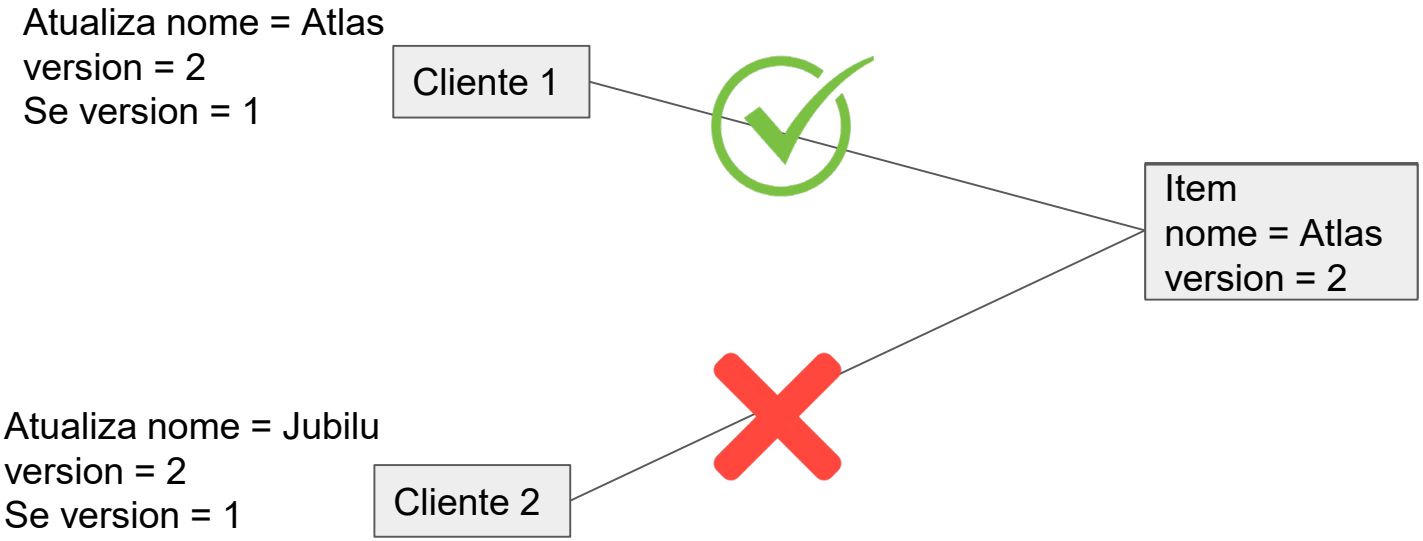
Partition Key | Sort Key | Atributos

usuario_id	jogo_id	jogo_ts	Resultado	Duração
123jfu54	2546	“2019-04-29T15:36:52”	ganhou	5
fur85hr5	2145	“2019-04-03T17:46:50”	perdeu	25

Atributos | Partition Key | Sort Key | Atributos

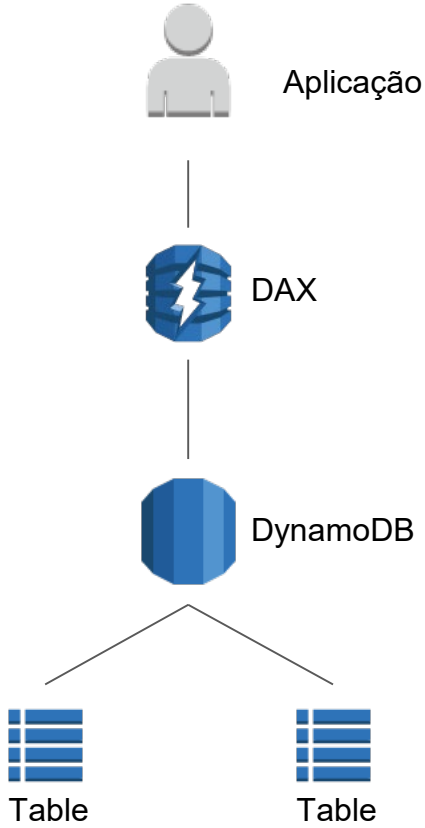
DynamoDB - Concurrency

- DynamoDB tem uma função chamada “Conditional Update / Delete”
- Isso garante que o item não foi modificado antes de ser alterado
- Isso faz DynamoDB Optimistic Locking / Concurrency Database



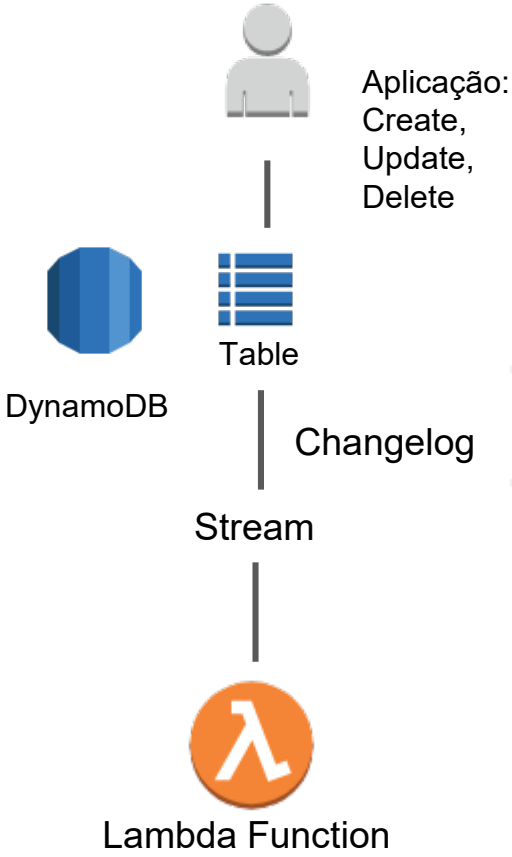
DynamoDB - DAX

- DAX = DynamoDB Accelerator
- Não é necessário nenhuma alteração na aplicação
- Escrita vai para o DAX e DAX escreve no DynamoDB
- Latência de Microsegundos para leituras e queries no Cache
- Resolve o problema de Hot Key (muitas leituras na mesma partição)
- 5 minutos TTL (Cache) padrão
- Até 10 nodes no cluster
- Multi AZ (mínimo de 3 nodes recomendado para produção)
- Securança (Criptografia em repouso com KMS, VPC, IAM, CloudTrail…)



DynamoDB - Streams

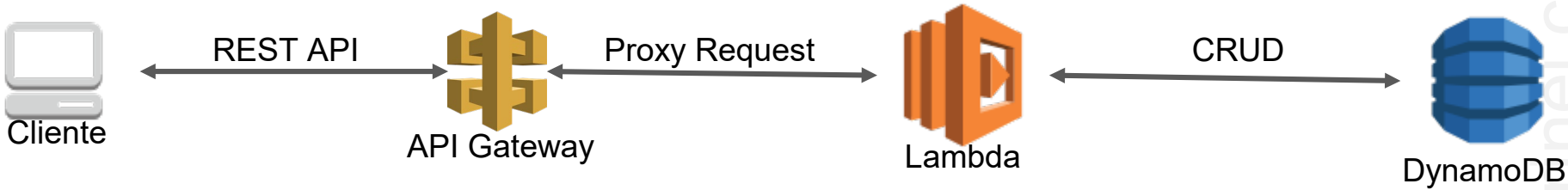
- Mudança no DynamoDB (Create, Update, Delete) pode iniciar um DynamoDB Stream
- Esse Stream pode ser lido pela AWS Lambda para:
  - Reagir a mudanças em tempo real (enviar um email de boas vindas)
  - Análise
  - Criar derivative tables / views
  - Insert no ElasticSearch
- Possível imprelemtar Cross Region Replication usando Streams
- Stream tem 24 horas de retenção



DynamoDB - Segurança e outros recursos

- Security:
  - VPC Endpoint disponível para acessar DynamoDB sem Internet
  - Acesso totalmente controlado pelo IAM
  - Criptografia em repouso usando KMS
  - Criptografia em transito usando SSL / TLS
- Backup e Restore
  - Restauração Point in time (igual RDS)
  - Sem impacto na performance
- Global Tables
  - Multi Region, Replicável, Alta Performance
- Amazon DMS pode ser usado para migrar para DynamoDB (Mongo, Oracle, MySQL, S3, etc…)
- É Possível instalar DynamoDB em uma máquina local para desenvolvimento

API Gateway (Application Programming Interface)



AWS API Gateway

- AWS Lambda + API Gateway: Sem Infraestrutura para gerenciar
- Gerencia Versões da API (v1, v2, v3)
- Gerencia diferentes Environments (dev, test, prod…)
- Gerencia Segurança (Autenticação e Autorização)
- Cria API Keys, gerencia Request Throttling
- Importação Swagger / Open API para definição rápida de APIs
- Transforma e Valida Requests e Responses
- Gera SDK e especificação da API
- Cache API Responses

API Gateway - Integração

- Fora do VPC
  - AWS Lambda (mais comum)
  - Endpoint na EC2
  - Load Balancers
  - Qualquer outro Serviço AWS
  - Acesso público através de HTTP
- Dentro do VPC
  - AWS Lambda no VPC
  - EC2 endpoint no VPC

API Gateway - Deployment Stages

- Alterar um API e salvar não significa que as alterações entrarão em vigor
- É preciso Deploy para as alterações terem efeito
- Isso pode causar alguma confusão
- Mudanças são Deployed para Stages (quantas você quiser)
- Exemplos de nomes para Stages (dev, test, prod)
- Cada Stage tem sua própria configuração
- É possível usar Rollback nos Stages

API Gateway - Stage Variables

- Stage Variables são como Environment Variables para API Gateway
- Use para mudanças frequentes na configuração do API
- Pode ser usados para:
  - Lamba Function ARN
  - HTTP endepoint
  - Parameter Mapping Templates
- Exemplo
  - Configure HTTP Endepoint
  - Passar configuração de parametros para AWS Lambda através de mapping templates
  - Stage Variables são passados para o objeto “context” na Lambda Function

API Gateway - Stage Variables e Lambda Aliases

- Criamos um Stage Variable para indicar o correspondente Lambda Alias
- Nosso API gateway irá automaticamente invocar a correta Lambda Function

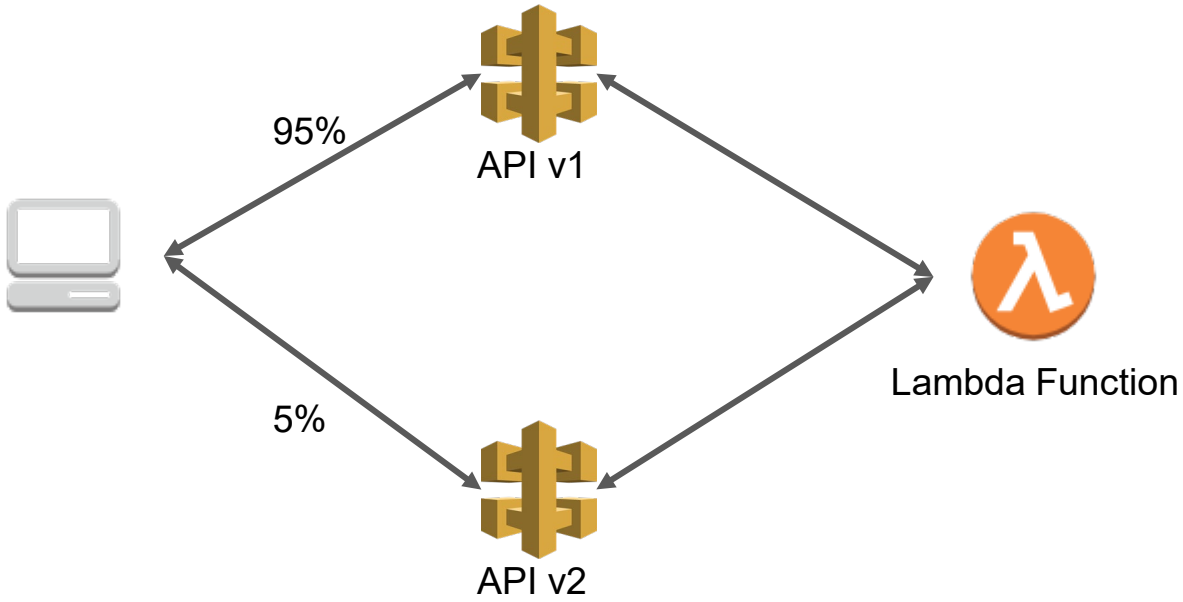
API Gateway

Lambda

Stages (variáveis)	Aliases	Versions
Prod (alias=PROD)	PROD	2
Test (alias=TEST)	TEST	4
Dev (alias=DEV)	DEV	$LATEST

API Gateway - Canary Deployment

- Possível habilitar Canary Deployment para qualquer Stage (Normalmente prod)
- Escolha o percentual de tráfego para cada API



Mapping Template

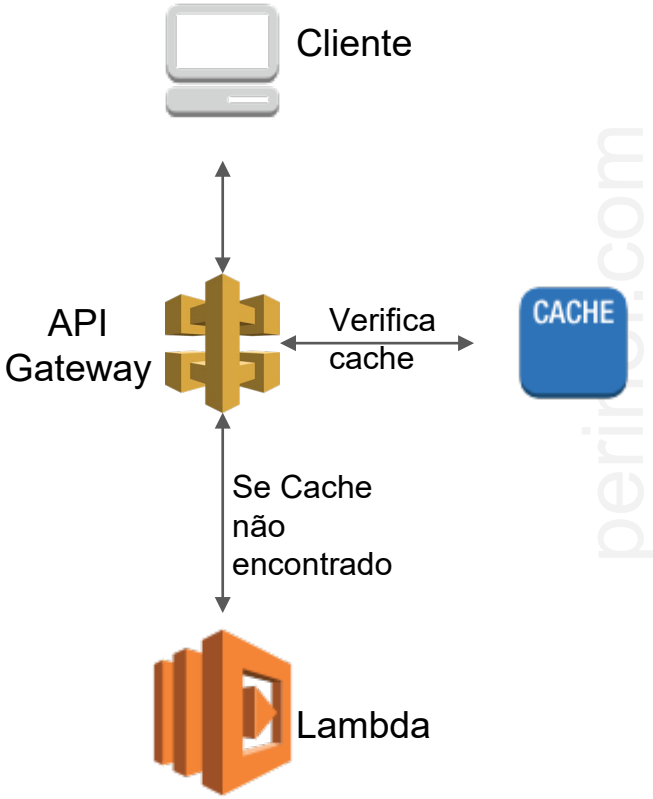
- Mapping Template pode ser usado para modificar Request / Response
- Renomear parametros
- Modificar body content
- Adicionar Headers
- Map JSON para XML para enviar para o backend ou para o cliente
- Use Velocity Template Language (VTL): para loop, if e etc
- Filtra resultado da saída (remove dados desnecessários)

AWS API Gateway Swagger / Open API spec

- Forma comum de definir REST API, usando definição da API como código
- Importa Swagger / OpenAPI 3.0 spec para API Gateway
  - Method
  - Method Request
  - Integration Request
  - Method Response
  - + AWS extensions para API Gateway e configuração de cada opção
- Pode exportar API como Swagger / OpenAPI spec
- Usando Swagger podemos gerar SDK para nossa aplicação

Caching API Response

- Caching reduz o número de chamadas feitas para o backend
- Default time TTL (time to live) é 300 segundos (mínimo: 0s, máximo 3600s)
- Cache são definidos por Stage
- Opção de criptografia
- Capacidade do cache entre 0.5GB e 237GB
- Possibilidade de alterar a configuração do cache para um método específico
- Habilidade de limpar o cache inteiro imediatamente
- Clientes podem invalidar o cache com o Header: CacheControl:max-age=0 (com a correta autorização IAM)



AWS API Gateway - Logging, Monitoring, Tracing

- CloudWatch Logs:
  - Habilita CloudWatch logging no nível Stage (com Log Level)
  - Capacidade de alterar as configurações por API (Ex. ERROR, DEBUG, INFO)
  - Log contém informação sobre Request / Response Body
- CloudWatch Metrics:
  - Por Stage
  - Possibilidade de habilitar Detailed Metrics
- X-Ray
  - Habilitar Tracing para pegar informação extra sobre Requests no API Gateway
  - X-Ray API Gateway + AWS Lambda oferecem uma solução completa

AWS API Gateway - CORS

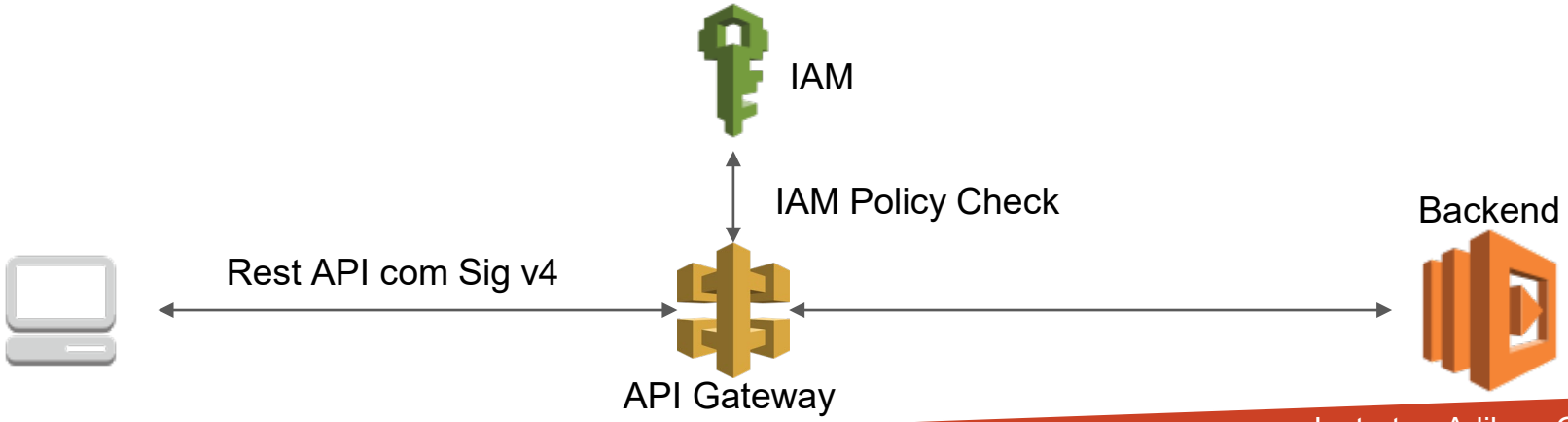
- CORS (Cross-Origin Resource Sharing) deve ser habilitado quando API recebe chamadas de outro domínio
- OPTIONS Pre-Flight Request deve conter os seguintes headers:
  - Access-Control-Allow-Methods
  - Access-Control-Allow-Headers
  - Access-Control-Allow-Origin
- CORS pode ser habilitado pelo console AWS

AWS API Gateway - Usage Plans e API Keys

- Possibilidade de limitar o uso da API
- Usage Plans:
  - Throttling: Configura a capacidade normal e Burst
  - Quotas: número de request por dia / semana / mês
  - Associado com API Stages
- API Keys:
  - Gera uma por cliente
  - Associado com Usage Plans
- Possibilidade de rastrear o uso da API Key

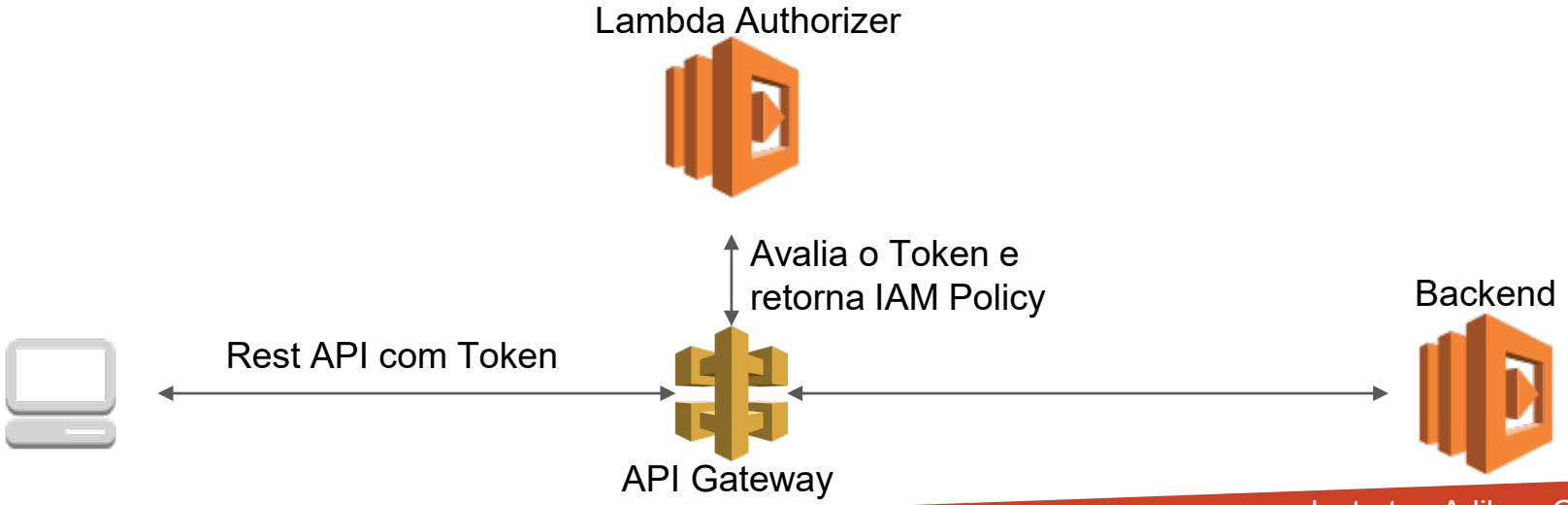
AWS API Gateway - Security

- Permissão IAM
  - Criar IAM Policy com a devida autorização e anexar ao User / Role
  - API Gateway verifica permisão IAM passada durante a chamada
  - Ideal para fornecer acesso dentro da infraestrutura AWS
  - Utiliza a capacidade “Sig v4” onde as credenciais IAM estão no Header



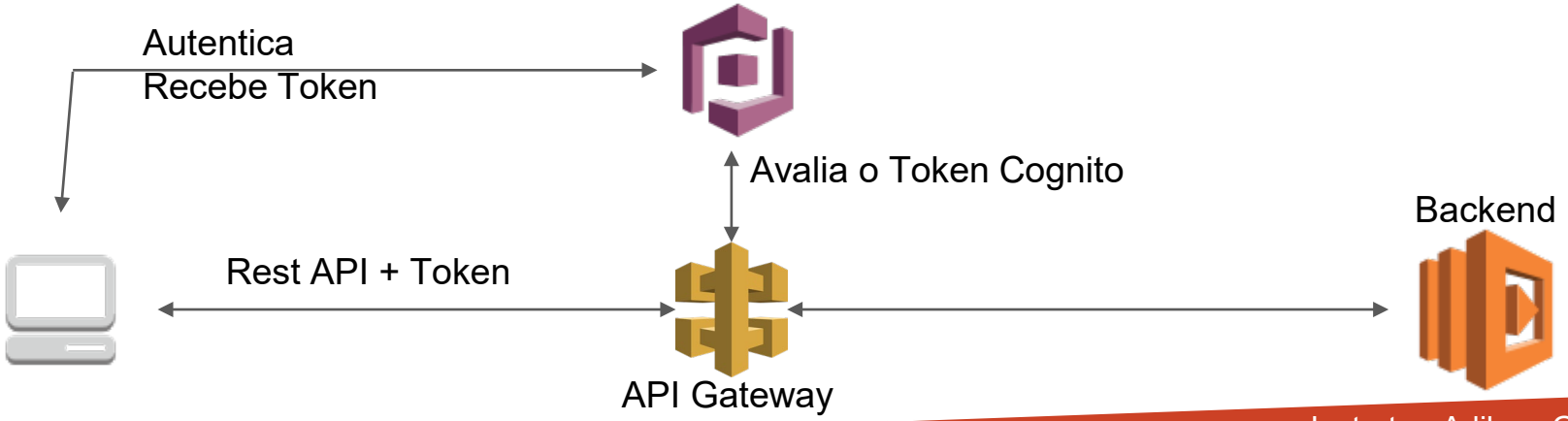
AWS API Gateway - Security

- Lambda Authorizer (Custom Authorizers)
- Usa AWS Lambda para validar o token no header que está sendo passado
- Opção para Cache do resultado da autenticação
- Lambda deve retornar uma IAM Policy para o usuário



AWS API Gateway - Security

- Cognito User Pools
  - Cognito gerencia User Lifecycle
  - API gateway verifica a identidade automaticamente do AWS Cognito
  - Não é necessário nenhum implementação
  - Cognito só trabalha na autenticação, não na autorização



AWS API Gateway - Segurança - Resumo

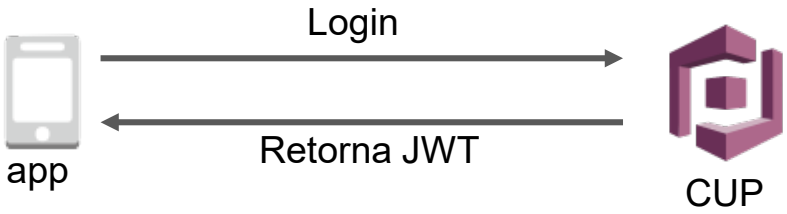
- IAM:
  - Ideal para users/roles que já existem na sua conta AWS
  - Gerencia autenticação e autorização
  - Utiliza Sig v4
- Custom Authorizer:
  - Ideal para tokens de aplicações de terceiros
  - Flexível sobre o que um IAM Policy retorna
  - Gerencia autenticação e autorização
  - Cobrado por chamada a Lamba Function
- Cognito User Pool
  - Gerenciado por você (pode usar Facebook, Google, Amazon)
  - Sem necessidade de escrever código
  - Deve-se implementar autorização no Backend

AWS Cognito

- Para dar uma identidade para nossos usuários, dessa forma eles podem interagir com nossa aplicação
- Cognito User Pools:
  - Sign in para app users
  - Integrado com API Gateway
- Cognito Identity Pools (Federated Identity):
  - Oferece Credenciais AWS para usuários para eles acessarem recursos AWS diretamente
  - Integrado com Cognito User Pools com um Identity Provider
- Cognito Sync:
  - Sincroniza dados do dispositivo com Cognito
  - Deve ser substituído por AppSync no futuro

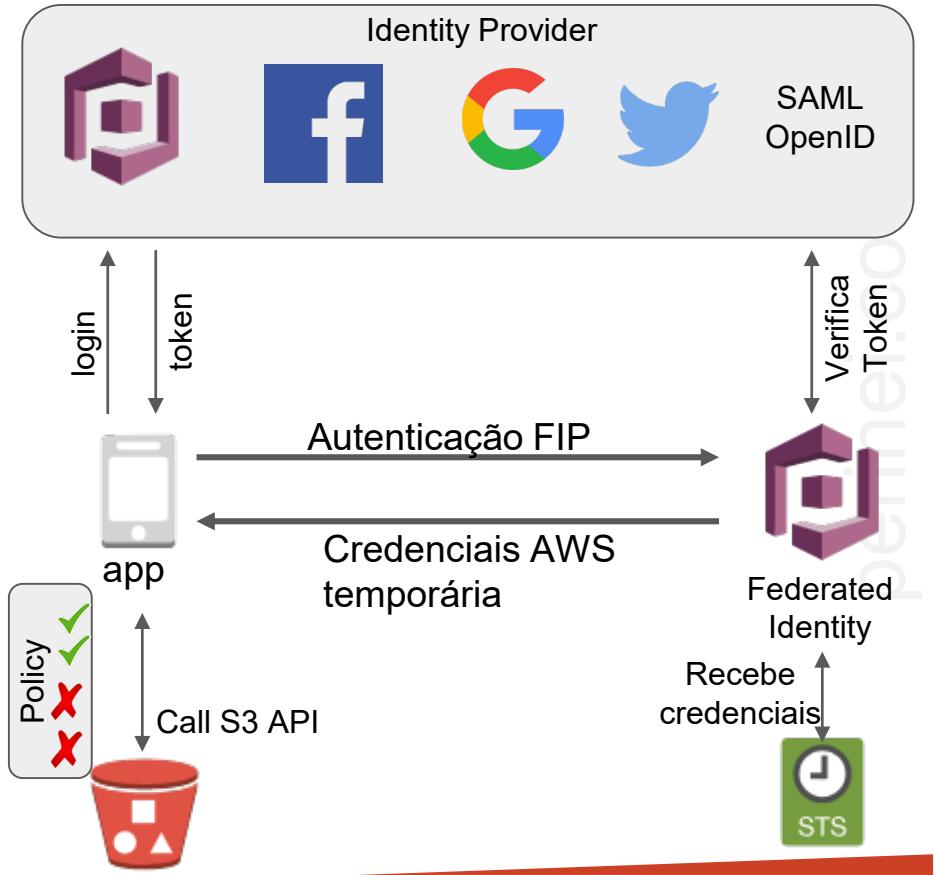
AWS Cognito User Pools (CUP)

- Cria um Banco de Dados Gerenciado pela AWS para ser usado pelo seu aplicativo móvel
- Simple login: Username (ou email) / senha/password
- Possibilidade de verificar email / telefone e adicionar MFA
- Pode-se habilitar Federated Identities (Facebook, Google, SAML…)
- Retorna um JSON Web Tokens (JWT)
- Pode ser integrado com API Gateway para autenticação



AWS Cognito – Federated Identity Pools

- Objetivo:
  - Fornecer ao Cliente acesso direto ao recursos AWS
- Como:
  - Log in no federated identity provider – ou permanece anonimo
  - Recebe credenciais AWS temporárias do Federated Identity Pool
  - Essas credenciais veem com um IAM Policy pré definida com permissões
- Exemplo:
  - Oferece acesso temporário para escrever no S3 Bucket usando o login do Facebook



AWS Cognito Sync

- Armazena preferencias, configurações e estado do app
- Sincronização entre vários dispositivos (Qualquer Plataforma - iOS, Android…)
- Capacidade de sincronização Offline (sincroniza quando dispositivo fica Online)
- Requer Federated Identity Pool no Cognito (Não no User Pool)
- Armazena dados no datasets (até 1MB)
- Máximo de 20 datasets para sincronização
- Não use Congnito Sync, Use AWS AppSync

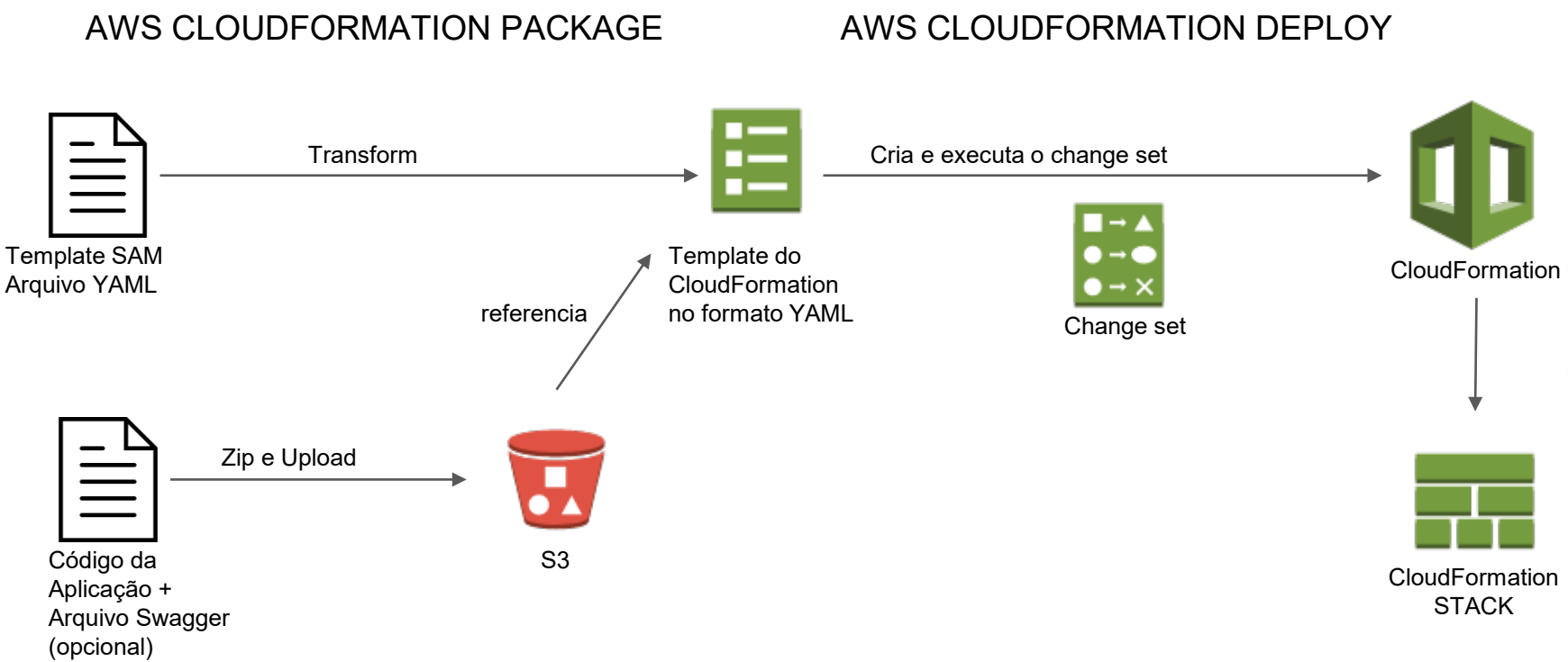
AWS Serverless Application Model (SAM)

- SAM = Serverless Application Model
- Framework para desenvolver e publicar aplicações serverless
- Configuração no formato YAML
- Gera complexos CloudFormation de um simples arquivo SAM YAML
- Suporta qualquer coisa do CloudFormation: Outputs, Mappings, Parameters, Resources…
- Somente 2 comandos para publicar no AWS
- SAM pode usar CodeDeploy para deploy Lambda functions
- SAM pode ajudar você a rodar Lambda, API Gateway, DynamoDB localmente

AWS SAM

- Transform Header indica que é um template SAM:
  - Transform: 'AWS::Serverless-2016-10 -31'
- Write Code
  - AWS::Serverless::Function (Lambda)
  - AWS::Serverless::Api (API Gateway)
  - AWS::Serverless::SimpleTable (DynamoDB)
- Package & Deploy:
  - aws cloudformation package / sam package
  - aws cloudformation deploy / sam deploy

AWS SAM – Como funciona



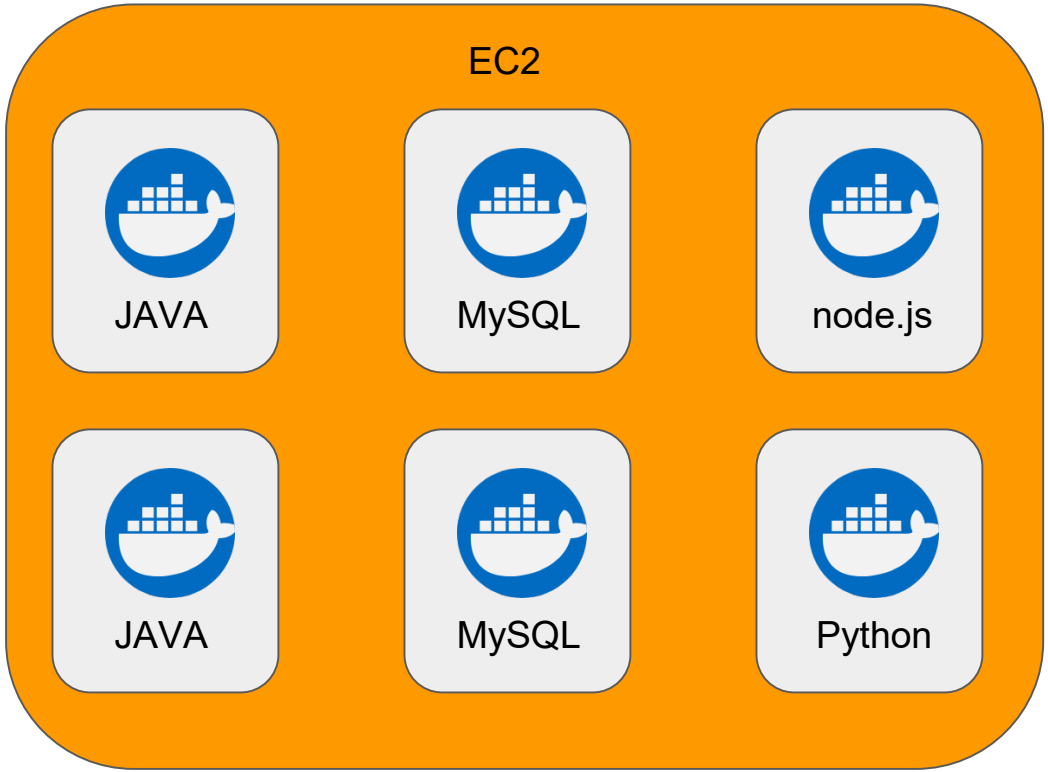
AWS ECS - Docker Containers in AWS

- Docker
- ECS
  - Cluster
  - Services
  - Tasks
  - Task Definition
- ECR
- Fargate

O que é Docker?

- Docker é um plataforma de desenvolvimento de software para implantar apps
- Apps são empacotados em Containers que podem rodar em qualquer system operacional
- Roda da mesma forma independente de onde está
  - Qualquer máquina
  - Sem problemas de compatibilidade
  - Comportamento previsível
  - Menos trabalho
  - Mais fácil de manter e deploy
  - Trabalha com qualquer linguagem, sistema operacional e tecnologia

Docker em um instância EC2

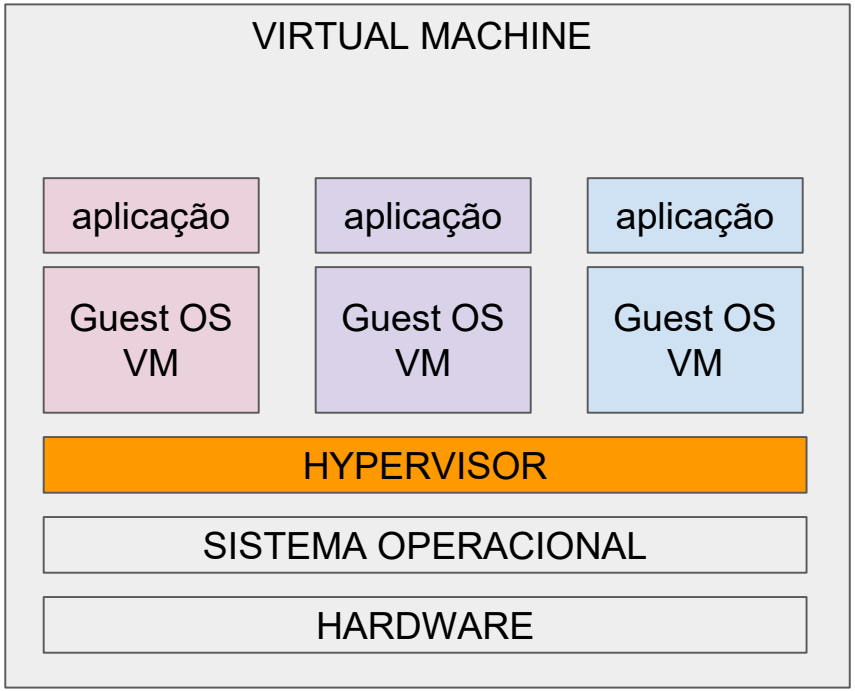


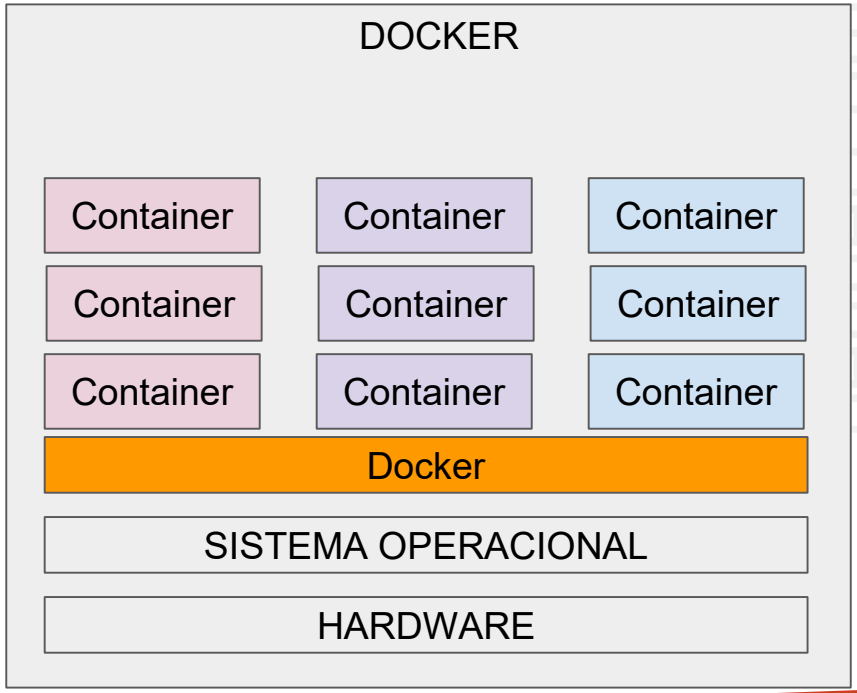
Onde Docker images são armazenadas?

- Docker images são armezenadas em Docker Repositories
- Público: Docker Hub https://hub.docker.com/
  - Encontre Images para várias tecnologias ou Sistema Operacional:
    - Ubuntu
    - MySQL
    - NodeJS
    - Java…
- Privado: Amazon ECR (Elastic Container Registry)

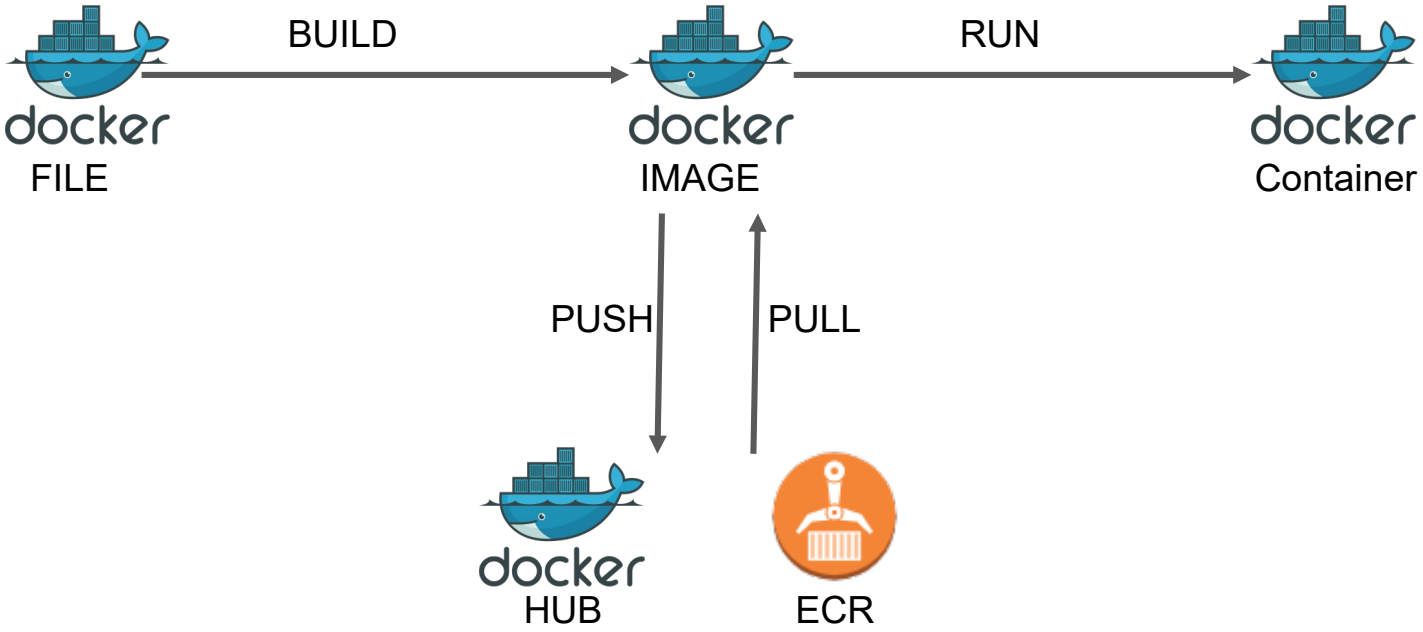
Docker X Virtual Machines

- Docker é um conjunto de tecnologias de virtualização
- Recursos são compartilhados pelo Host: Muitos Containers em um único servidor





Docker

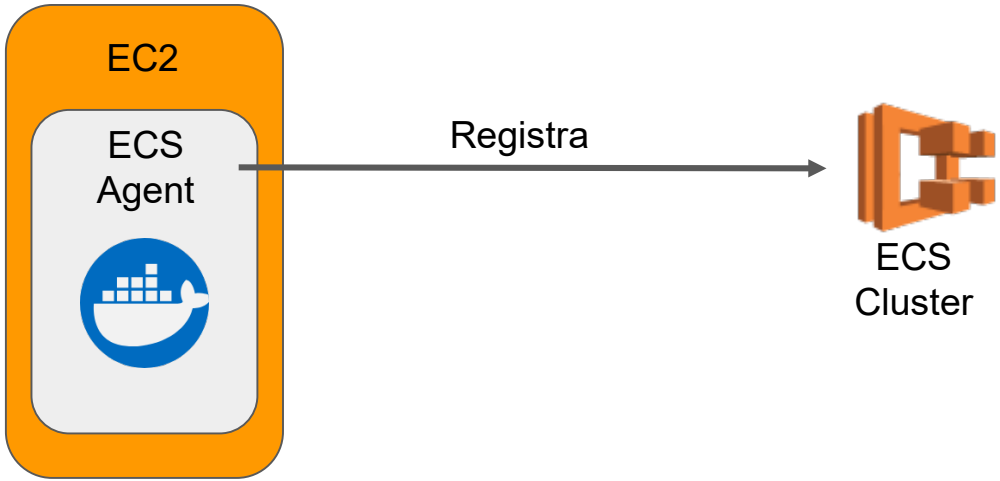


Gerenciamento de Docker Container

- Para gerenciar container, precisamos de uma Plataforma de Gerenciamento de Containers
- Temos 3 opções
  - ECS: Plataforma da AWS gerenciada pelo usuário
  - Fargate: Plataforma da AWS gerenciada pela AWS
  - EKS: Kubernetes (open source) gerenciado pela AWS

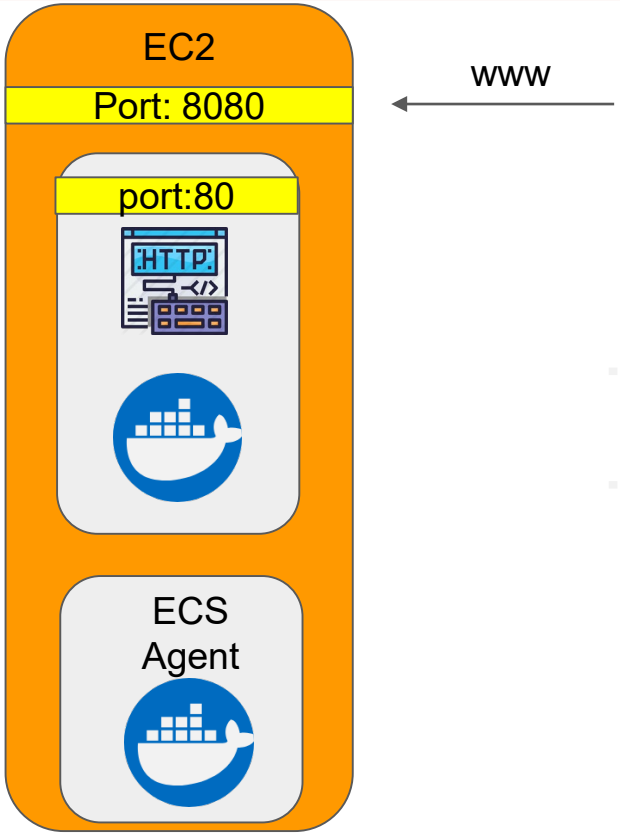
ECS Clusters

- ECS Cluster são grupos lógicos de EC2
- EC2 roda ECS agent (Docker container)
- ECS Agent registra a instância EC2 no ECS cluster
- EC2 roda um AMI especial, feito exclusivamente para ECS



ECS Task Definition

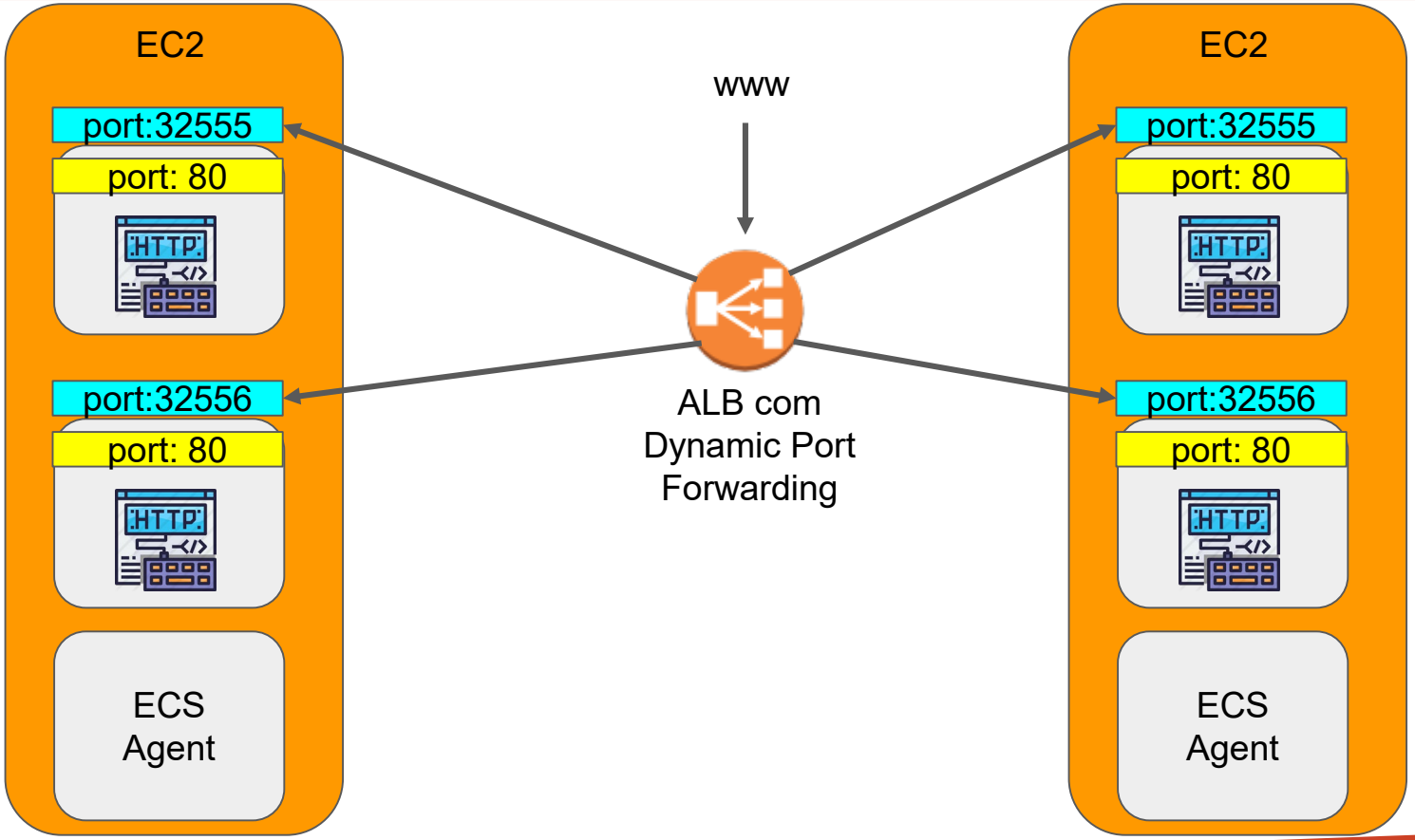
- Task Definition são Metadata no formato JSON que diz ao ECS como rodar Docker Container
- Contém informações cruciais sobre:
  - Image Name
  - Port Binding para Container e Host
  - Memória e CPU requerida
  - Environment Variables
  - Informações de Rede
  - IAM Role



ECS Service

- ECS Services define quantos Tasks devem rodar e como eles devem rodar
- Eles garantem que o número de Tasks definidas fique sempre rodando no Cluster de EC2
- Pode trabalhar junto com ELB, NLB, ALB se necessário

ECS Service com Load Balancer



ECR

- Até agora nós usamos Docker Image do Docker Hub (público)
- ECR é um repositório privado
- Acesso é controlado através de IAM (Policy)
- É necessário executar alguns comandos para Push e Pull
  - $(aws ecr get-login --no-include-email --region eu-west-1)
  - docker push 1234567890.dkr.ecr.us-east-1.amazonaws.com/imagename:latest
  - docker pull 1234567890.dkr.ecr.us-east-1.amazonaws.com/imagename:latest

Fargate

- Quando criamos nosso ECS Cluster, nós temos que criar nossas instâncias EC2
- Precisamos adicionar novas instância EC2 para escalonar
- Nesse caso nós gerenciamos a infraestrutura
- Com Fargate não é necessário gerenciar nada (É serveless)
- Não é necessário provisionar instância EC2
- Só precisamos criar Task Definitions e AWS criará Containers para nós
- Para escalonar, simplesmente ajuste o Task Number

ECS - Resumo

- ECS é usado para rodar Docker Containers
- Existe 3 formas de rodar Docker Containers na AWS:
  - ECS “Classic”: Necessário provisionar EC2
  - Fargate: ECS Serveless. Sem necessidade de gerenciar EC2
  - EKS: Kubernetes gerenciado pela AWS

ECS - Classic

- Instância EC2 devem ser criadas
- Devemos configurar o arquivo /etc/ecs/ecs.config com o nome do cluster
- EC2 deve rodar ECS Agente
- EC2 pode rodar Múltiplos Containers do mesmo tipo
- Deve-se especificar somente a porta do Container (não especificar a porta do Host)
- Deve-se usar Application Load Balancer com dynamic port mapping
- Security Group do EC2 deve permitir tráfego do ALB em todas as portas
- ECS tasks pode ter IAM Roles para executar ações na AWS
- Security Groups Operam no nível instância EC2, não no nível Task

ECR é usado para armazenar Docker Image

- ECR é integrado com IAM
- Push (Empurrar):
  - $(aws ecr get-login --no-include-email --region eu-west-1)
  - docker push 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
- Pull (Puxar):
  - $(aws ecr get-login --no-include-email --region eu-west-1)
  - docker pull 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
- “aws ecr get-login” generates a “docker login” command
- Se EC2 não conseguir fazer um Pull da Docker Image, verifique IAM

Fargate

- Fargate é Serverless (não precisar gerenciar EC2)
- AWS provisiona Containers para nós e associa a ENI
- Fargate Containers são provisionados pelo Container Spec (CPU / RAM)
- Fargate Tasks pode ter IAM Roles para executar ações na AWS

Integração ECS

- ECS é integrado com X-Ray
  - Para funcionar, X-Ray deve rodar como segundo container dentro do Task Definition
  - Use a Imagem: https://hub.docker.com/r/amazon/awsxray-daemon/
  - Mais Informações: https://docs.aws.amazon.com/xray/latest/devguide/xraydaemon-ecs.html
  - ECS é integrado com CloudWatch Logs:
  - É necessário configurar logging no nível Task Definition
  - Cada Container terá um Log Stream diferente

CLI

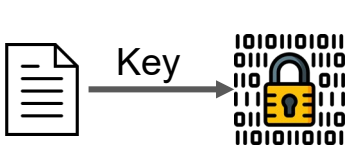
- Obter um “docker login” no ECR:
  - aws ecr get-login
- Pull:
  - docker pull 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
- Push:
  - docker push 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
- Create a service on ECS:
  - aws ecs create-service • Build a docker image: • docker build -t demo .

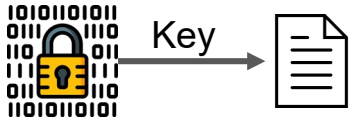
AWS Segurança e Criptografia

- KMS
- Encryption SDK
- SSM Parameter Store

Criptografia em trânsito (in transit / in flight)

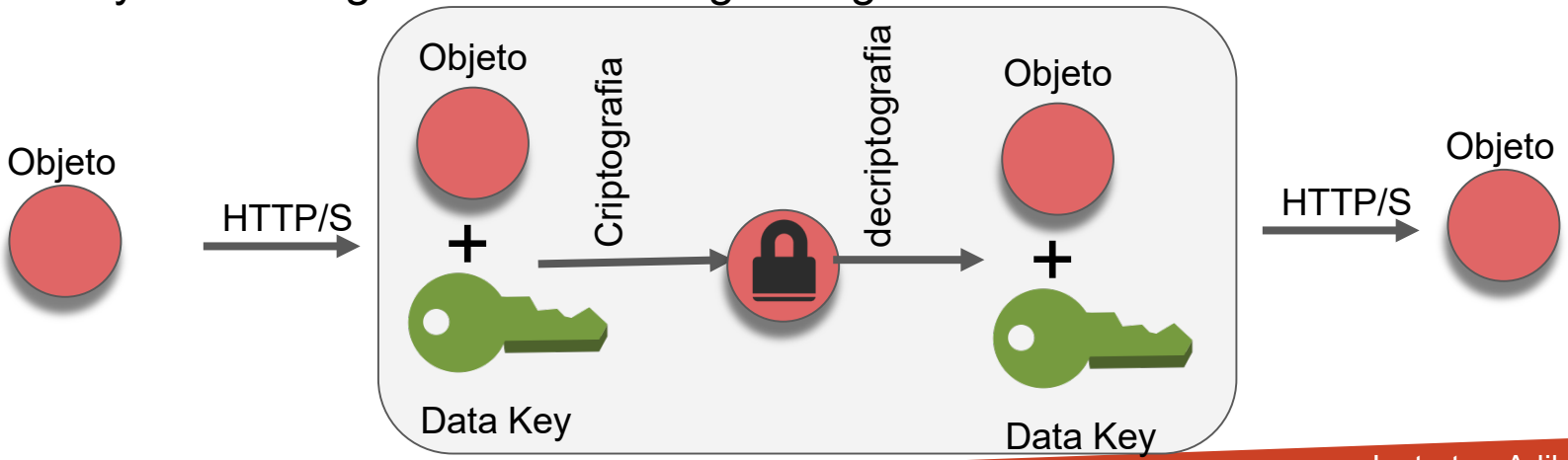
- Criptografia em Transito (SSL)
  - Dados são criptografados antes de enviar e descriptografados após o recebimento
  - Certificados SSL (HTTPS)
  - Criptografia em trânsito garante que não haverá Ataque Man In The Middle





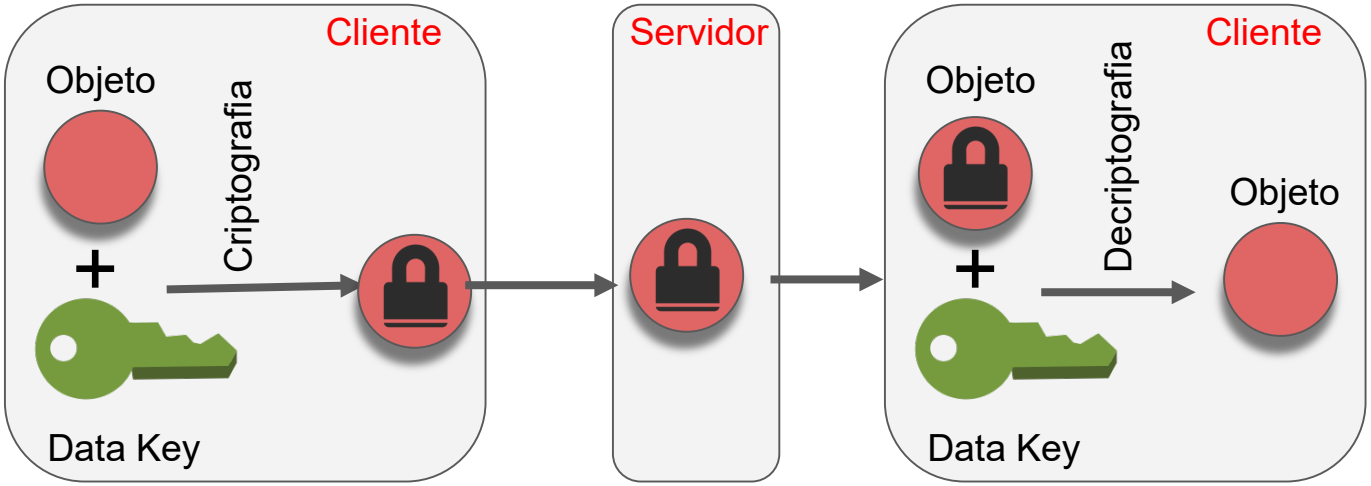
Criptografia do Lador Servidor em Repouso

- Criptografia do Lado Servidor em Repouso (at rest)
- Dados são criptografados depois de serem recebidos pelo servidor
- Dados são descriptografados antes de serem enviados
- Dados são armazenados criptografados usando uma Key, normalmente Data Key
- Key deve ser gerenciada em algum lugar e o servidor deve ter acesso a Key



Criptografia do Lado Cliente

- Criptografia do Lado Cliente
- Dados são criptografados pelo Cliente e nunca descriptografados no Servidor
- Dados são descriptografados pelo Cliente
- Servidor nunca deve ser capaz de descriptografar os dados
- Pode utilizar Envelope Encryption



AWS KMS (Key Management Service)

- KMS é a forma mais popular de criptografar dados na AWS
- Forma fácil de controlar acesso aos dados.
- Keys gerenciadas pela AWS
- Totalmente integrado com Autorização IAM
- Funciona nos bastidores para os seguintes serviços AWS
    - Amazon EBS: Criptografa volumes
    - Amazon S3: Criptografia do lado Servidor para objetos
    - Amazon Redshift: Criptografia de dados
    - Amazon RDS: Criptografia de dados
    - Amazon SSM: Parameter store
    - É possivel utilizar KMS via CLI / SDK

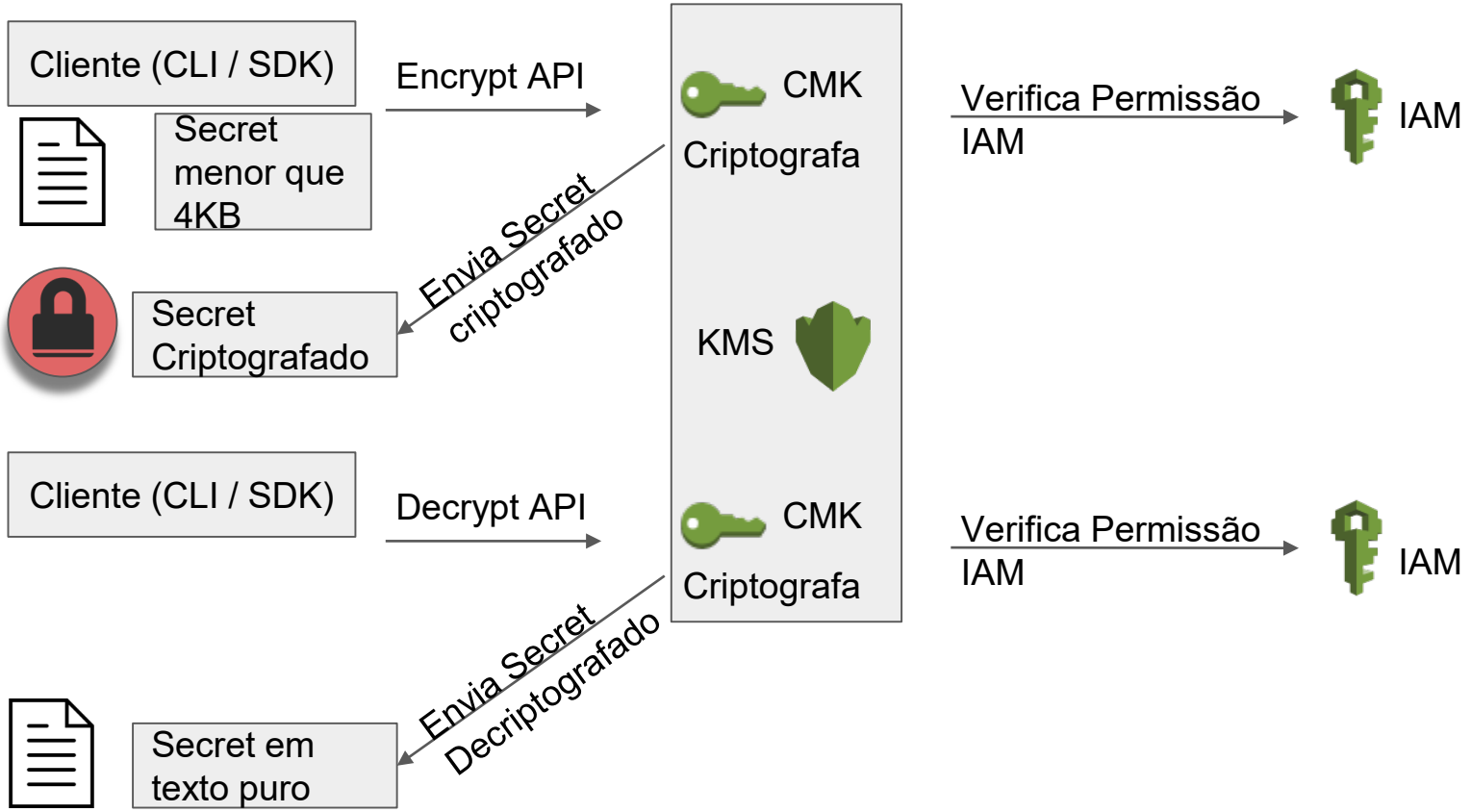
AWS KMS (Key Management Service)

- Sempre que quiser compartilhar informações sensíveis use KMS
  - Senha de Banco de dados
  - Credenciais para serviços externos
  - Private Key de certificados SSL
- No KMS, CMK é usada para criptografar dados e nunca pode ser lida pelo usuário, CMK pode ser alternada para aumentar a segurança
- Nunca coloque suas credenciais em texto puro, especialmente no seu código
- KMS pode criptografar no máximo 4KB por chamada
- Se os dados forem maiores que 4KB, use Envelope Encryption
- Para dar acesso ao KMS para outra pessoa:
  - Key Policy deve permitir acesso do usuário
  - IAM Policy deve permitir chamadas a API

AWS KMS (Key Management Service)

- Gerencia Keys e Policies
  - Create
  - Rotation Policies
  - Disable
  - Enable
- Possibilidade de auditar o uso da Key usando CloudTrail
- Existem 3 tipos de Customer Master Key (CMK):
  - AWS Managed Service Default CMK: Grátis
  - User Key Criada no KMS: $1 por mês
  - User Key importada (deve ser 256-bit symmetric key): $1 por mês
  - + Custo por API call no KMS: $0.03 / 10000 calls

Como KMS funciona?

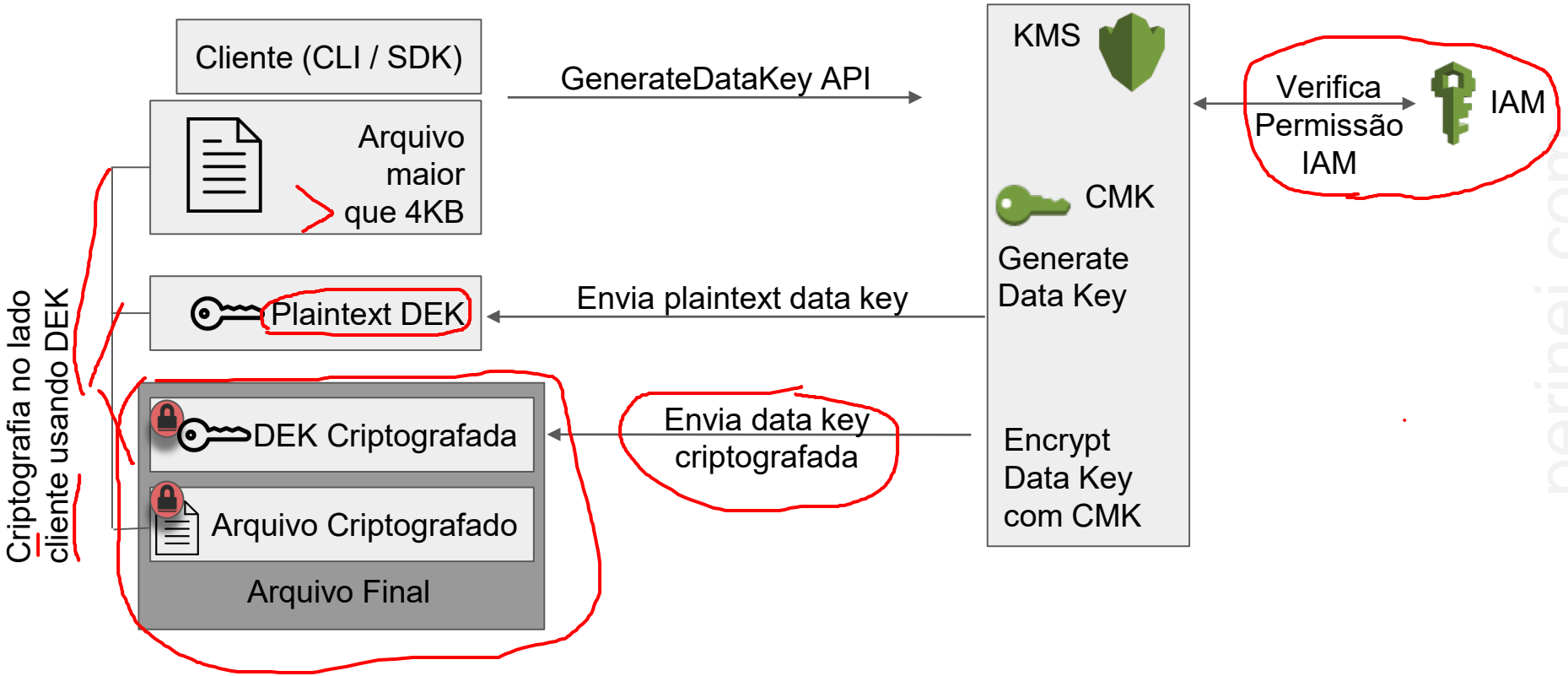


AWS Encryption SDK

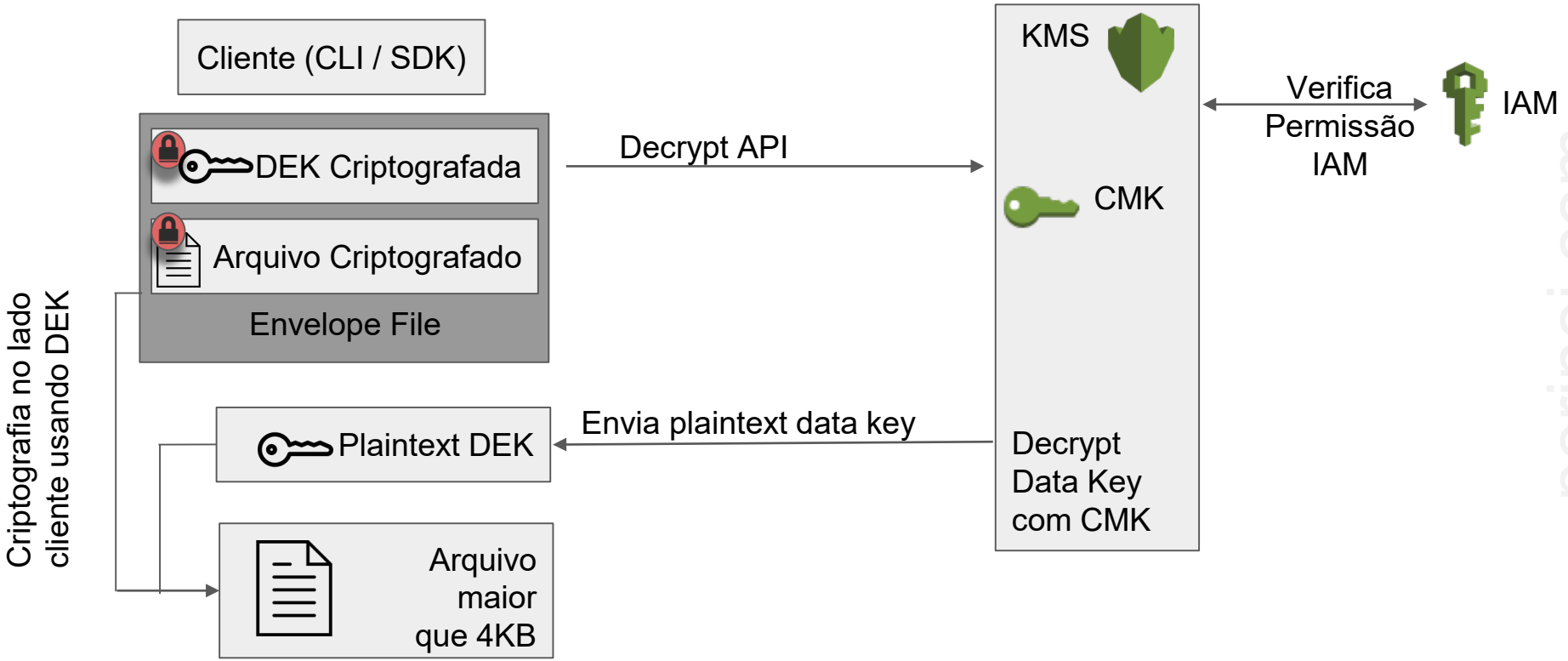
- Como fazer se você precisar criptografar mais de 4KB usando KMS?
- Para isso, precisamos usar Envelope Encryption
- Envelope Encryption dá um pouco de trabalho para implementar
- AWS Encryption SDK facilitar o uso de Envelope Encryption
- É diferente de S3 Encryption SDK
- Encryption SDK também existe como ferramenta CLI que nós podemos instalar

- Para a prova:
  - Tudo que tiver mais de 4KB deve ser criptografado usando Encryption SDK == Envelope Encryption = GenerateDataKey API

Envelope Encryption - GenerateDataKey API

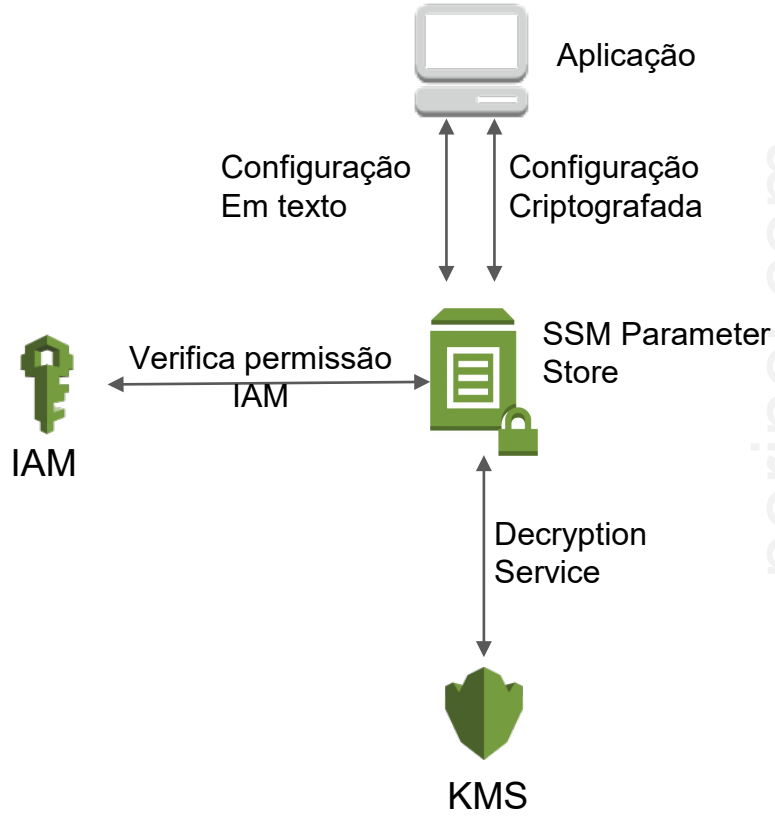


Decrypt Envelope Data



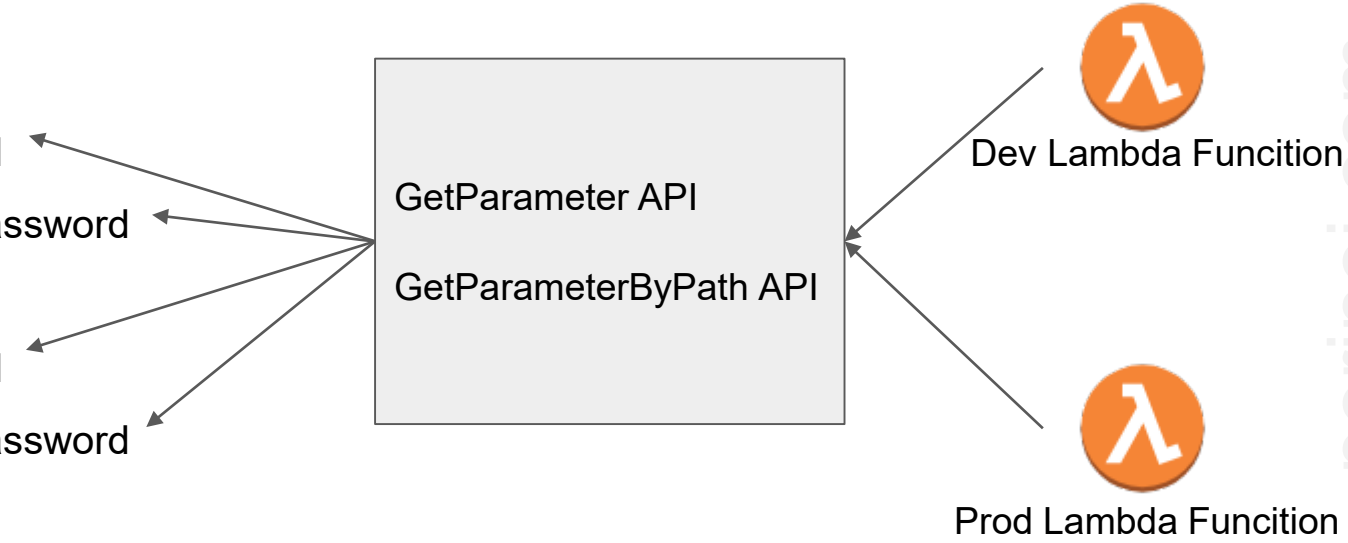
AWS Parameter Store

- Armazenamento Seguro para configuração e Secrets
- Criptografia usando KMS (opcional)
- Serverless, escalável, durável, SDK fácil de usar, grátis
- Rastreamento das Versões de configuração e Secrets
- Gerenciamento de configuração usando Path e IAM
- Notificações com CloudWatch Events
- Integração com CloudFormation



AWS Hierarquia Parameter Store

- /back-end/
  - app1/
    - dev/
      - db-url
      - db-password
    - prod/
      - db-url
      - db-password
  - app2/
- /front-end/



AWS Secret Manager

- É o mais novo serviço para armazenar Secrets
- Capacidade de forçar rotatividade das Secrets a cada X dias
- Geração automática de rotatividade de Secrets (Lambda)
- Integração com Amazon RDS (MySQL, PostgreSQL, Aurora)
- Secrets são criptografados usando KMS

IAM - Melhores Práticas

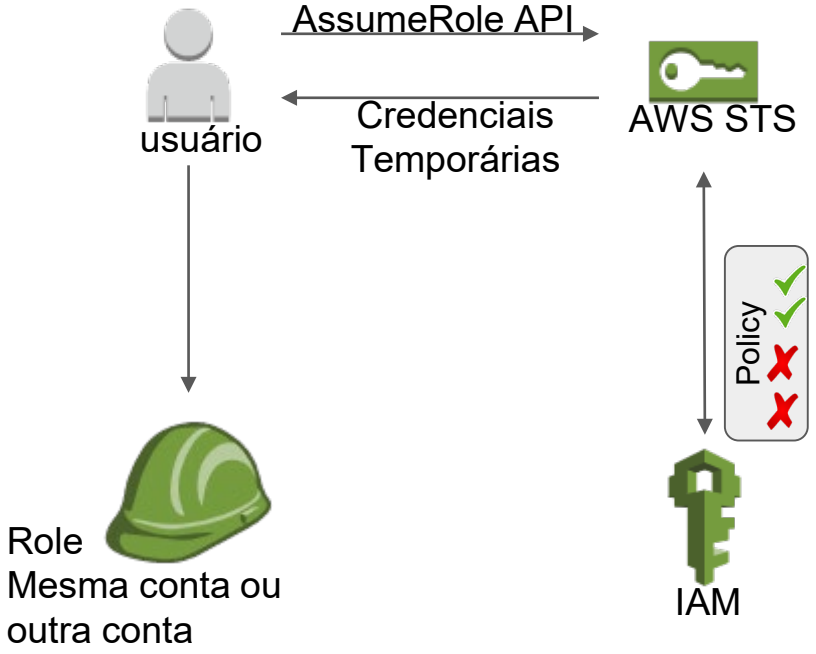
- Nunca use credenciais Root, Habilite MFA para conta Root
- Least Privilege
  - Cada Group / User / Role deve ter somente o mínimo nível de permissão que precisa
  - Nunca use “*” dentro de um Policy para acessar um serviço
  - Monitore API calls feitas pelos usuários no CloudTrail (principalmente Denies)
- Nunca coloque credenciais IAM key em um computador que não seja seu computador pessoal. Para EC2 instances use IAM Role
- Em servidores On Premise, a melhor prática é chamar STS para obter credenciais temporárias

IAM Roles - Melhores Práticas

- EC2 deve ter sua própria Role
- Lambda Function deve ter sua própria Role
- ECS Task deve ter sua própria Role (ECS_ENABLE_TASK_IAM_ROLE=true)
- CodeBuild deve ter sua própria Service Role
- Crie Least-Privileged Role para qualquer serviço que precise de um Role
- Crie Role por aplicação / Lambda Function (Não reutilize Roles)

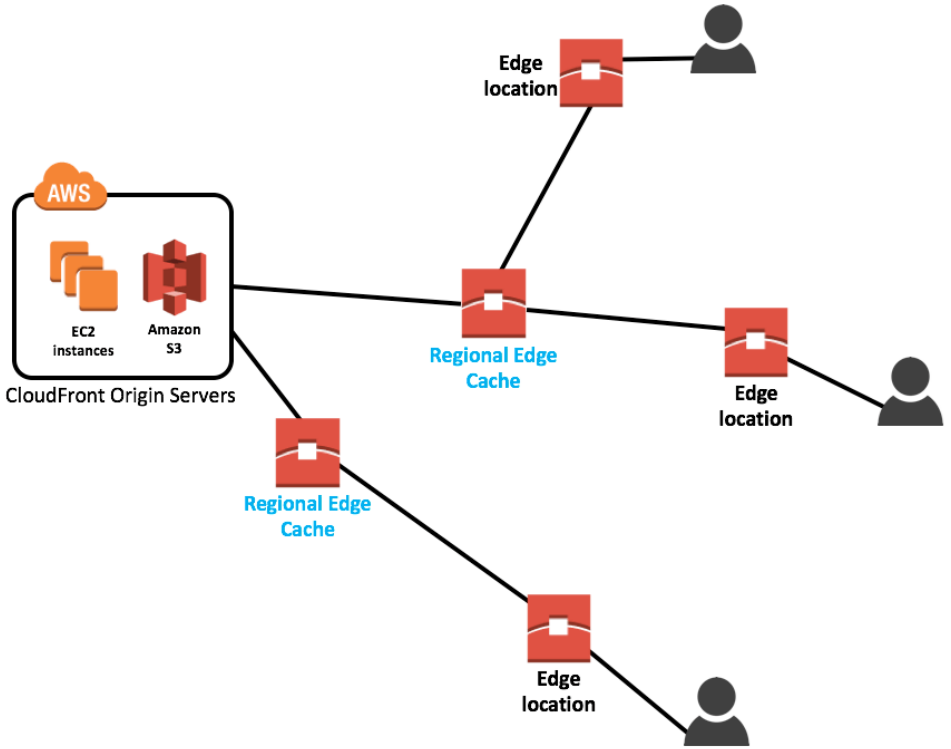
IAM - Melhores Práticas - Cross Account Acess

- Define IAM role para outra conta acessar
- Define que conta pode acessar essa IAM Role
- Utilize AWS STS (Security Token Service) para acessar as credenciais e assumir IAM Role que você tem acesso (AssumeRole API)
- Credenciais temporárias podem ser válidas pelo período de 15 minutos a 1 hora



AWS CloudFront

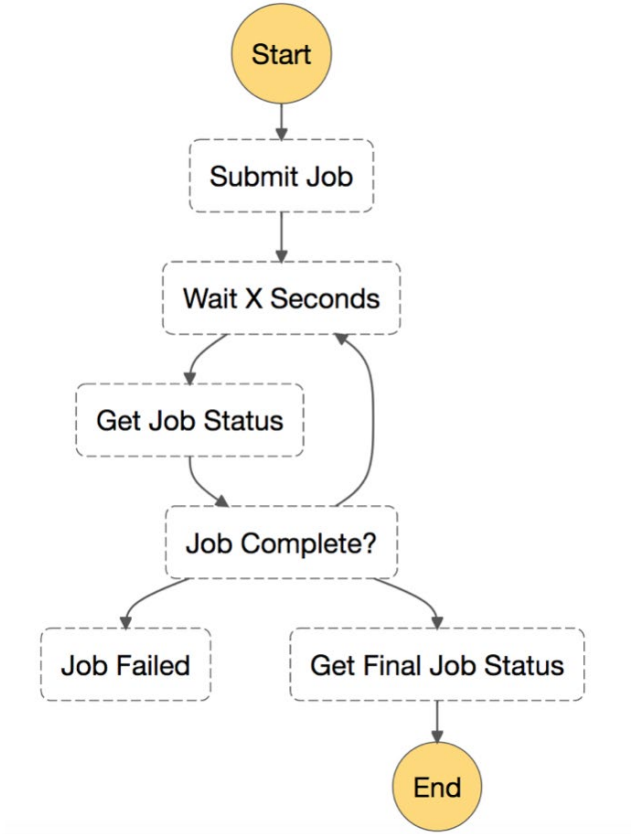
- CDN (Content Delivery Network)
- Melhora performance de leitura, conteúdo é Cached no Edge
- 169 pontos de presença globalmente (Edge Locations)
- É popular com S3 mas funciona com EC2, Load Balancing
- Ajuda a proteger contra ataques a rede
- Possibilidade de usar SSL (HTTPS) NO Edge usando ACM
- CloudFront pode usar SSL (HTTPS) para comunicar com sua aplicação
- Suporta protocolo RTMP (Video / Mídia)



AWS Step Function

- Cria um Workflow Visual para orquestrar suas Lambda Function
- Representa o fluxo como um JSON State Machine
- Recursos: Sequence, Parallel, Conditions, timeouts, error handling…
- Pode também ser integrado com EC2, ECS, On Premise Servers, API Gateway
- Tempo máximo de execução de 1 ano
- Possibilidade de implementar aprovação manual
- Exemplos:
  - Processamento de Dados
  - Aplicação Web

Step Function - Workflow Visual



AWS SWF - Simple Workflow Service

- Coordena trabalhos entre as aplicações
- Código roda no EC2 (não é Serverless)
- Tempo máximo de execução é de 1 ano
- Conceito de “activity step” e “decision step”
- Possibilidade de aprovação manual do “step”
- Exemplo: Carrinho de compras
- STEP FUNCTION é recomendado para ser usado para novas aplicações com exceção de:
    - Se você precisar de intervenção externa no processo
    - Se você precisar de Child Process que retorna valores para Parent Process

AWS SES - Simple Email Service

- Enviar email usando:
  - SMTP
  - AWS SDK
- Recebe email. Integração com:
  - S3
  - SNS
  - Lambda
- Integração com IAM para permitir enviar emails

AWS Database

- RDS: Banco de Dados, OLTP
  - PostgreSQL, MySQL, Oracle
  - Aurora + Aurora Serverless
  - Banco de Dados Provisionado
- DynamoDB: NoSQL
  - Gerenciado pela AWS, Key Value, Document
  - Serverless
- ElasticCache: Banco de Dados na memória RAM (In Memory)
  - Redis / Mencached
  - Função Cache

AWS Database

- Redshift: OLAP - Processamento analítico
  - Data Warehousing / Data Lake
  - Analytics Queries
- Neptune: Banco de Dados Gráfico
- DMS: Database Migration Service (Migrar banco de dados para AWS)

Limpando o ambiente

EC2 instances
Elastic Load Balancer
Launch Configurations
Auto Scaling Groups
Elastic IPs
Security Groups
EBS Volumes
Key Pairs
Elastic Beanstalk applications & environments
AWS Lambda functions
S3 objetos & buckets
RDS database
DynamoDB tables & indexes
ElastiCache cluster
Route53
API Gateway
Kinesis Streams
SNS Topics
SQS Queues
CodeCommit
CodeBuild
CodePipeline
CloudWatch Logs
CloudWatch Metrics
CloudWatch Dashboards
CloudWatch Alerts
CloudWatch Rules
CloudFormation Stacks
SSM Parameter Store
IAM User / Groups / Roles / Policies
ECS
ECR