Regiões e AZs

AZs são DC distintos
IAM e S3 são soluções Globais

IAM

Permissão e gerenciamento de acessos

A recomendação é que o usuário tenha sempre o menor privilégio possível.

- 1 user IAM por pessoa
- 1 rolê por aplicação, não reutilizar
- IAM nunca deve ser compartilhado
- O código não deve ter credenciais
- Por padrão, qualquer usuário criado não possui nenhuma permissão
- Usuário root, tem todas as permissões. Nenhuma permissão pode ser excluída, assim como o usuário.

Roles

- Roles são criadas e então assumem a permissão para a execução/requisição de serviços.
- Usada para conceder uma permissão, sem que seja necessário criar uma credencial permanente.
- As permissões são delegadas para uma role.

Policies

- Documentos que definem permissões
- Podem ser aplicadas em usuários, grupos e roles
- São escritas em json
- Todas as permissões são negadas por padrão
- A política mais restritiva é a que será aplicada.
- Policy simulator, é usado para testar as políticas
- Condições podem ser usadas para aplicar condições lógicas (ex. Endereço IP de origem)

Métodos de autenticação

1. Access Key
2. User and password
3. STS, token para acesso

EC2

São máquinas virtuais.
Limite de 20 instâncias on demand por conta.
Limite de 20 instâncias reservadas por conta.

Quando está criando uma EC2, às principais opções são:

- Network (vpc)
- Subnet
- Público IP (auto assign)
- IAM role
- Shutdown behavior

- Termination protection
- Monitoring
- Tenancy
- EBS
    - Tamanho
    - Tipo
    - Delete on terminator
    - Encriptação
- Tags
- SGs

Billing

- On demand
    - Pago pelo uso
    - Bom para dev/teste
- Spot
    - 90% de desconto em relação a on demand
    - Ideal para aplicações stateless
    - Pode rodar qualquer tipo de aplicação
    - O cliente é informado 2 minutos, para tomar uma ação antes da instância ser desligada
- Reservado
    - Mais barata que a on demand
    - Opções de reserva são de 1 ou 3 anos
    - Pagamentos upfront são mais baratos

	Standard	Convertible
Terms	1 year, 3 year	1 year, 3 year
Average discount off On-Demand price	40% - 60%	31% - 54%
Change AZ, instance size, networking type	Yes, via ModifyReservedInstance API or console	Yes, via ExchangeReservedInstance API or console
Change instance family, OS, tenancy, payment options	No	Yes
Benefit from price reductions	No	Yes

Tabela comparativa entre os planos.

On-Demand	Reserved	Spot
No upfront fee	Options: No upfront, partial upfront or all upfront	No upfront fee
Charged by hour or second	Charged by hour or second	Charged by hour or second
No commitment	1-year or 3-year commitment	No commitment
Ideal for short term needs or unpredictable workloads	Ideal for steady-state workloads and predictable usage	Ideal for cost-sensitive, compute intensive use cases that can withstand interruption

- Dedicado hosts
    - Servidor físico dedicado
    - Hardware dedicado
    - Bom para requisitos regulatórios
    - Bom para tipos de licenciamento
    - Opção mais cara de servidor
- Dedicada instâncias
    - Servidor virtual em um hardware dedicado
    - Não fornece acesso ao servidor físico

- Hardware pode ser compartilhado com outra instância da mesma conta
- Curso adicional de 2 dólares por região

Tabela comparativa

Characteristic	Dedicated Instances	Dedicated Hosts
Enables the use of dedicated physical servers	x	x
Per instance billing (subject to a $2 per region fee)	x	
Per host billing		x
Visibility of sockets, cores, host ID		x
Affinity between a host and instance		x
Targeted instance placement		x
Automatic instance placement	x	x
Add capacity using an allocation request		x

Forma de cobrança

- As horas parciais de instância consumidas são cobradas com base no uso da instância.
- As instâncias são cobradas quando estão em execução - precisam ser interrompidas ou encerradas para evitar o pagamento.
- Cobrança por hora ou segundo (por segundo apenas com instâncias do Linux).
- Os dados entre instâncias em diferentes regiões são cobrados (entrada e saída).
- As taxas regionais de transferência de dados se aplicam se pelo menos uma das seguintes condições for verdadeira, mas são cobradas apenas uma vez para uma determinada instância, mesmo se ambas forem verdadeiras:
  - A outra instância está em uma zona de disponibilidade diferente, independentemente do tipo de endereço usado.
  - Os endereços IP públicos ou elásticos são usados, independentemente da zona de disponibilidade em que a outra instância esteja.

Tipos de instâncias

Category	Families	Purpose/Design
General Purpose	A1, T3, T3a, T2, M5, M5a, M4	General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads
Compute Optimized	C5, C5n, C4	Compute Optimized instances are ideal for compute bound applications that benefit from high performance processors
Memory Optimized	R5, R5a, R4, X1e, X1, High Memory, z1d	Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory
Accelerated Computing	P3, P2, G4, G3, F1	Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating-point number calculations, graphics processing, or data pattern matching
Storage Optimized	I3, I3en, D2, H1	This instance family provides Non-Volatile Memory Express (NVMe) SSD-backed instance storage optimized for low latency, very high random I/O performance, high sequential read throughput and provide high IOPS at a low cost

SSH - Linux

A permissão do arquivo .pem precisa ter permissões de leitura apenas pelo dono do arquivo. Necessário alterar a permissão via chmod 0400

SSH - Windows

Precisa converter a chave para .ppk

SG

Inbound e Outbound

Controla

- Porta
- Origem
- Entrada
- Saída
- Ipv4 e IPv6

1. SG é por VPC. Não pode reutilizar em VPCs diferentes.
2. Erros de time out, são erros de SG.
3. Erros de conexão recusada, não são erros de SG
4. Outbound por padrão libera tudo

Elastic IP

- 5 por conta (padrão)

User Data

- Executa no primeiro boot
- Sempre roda com permissões de root
- Limite de 16KB
- Dados não são criptografados

Instance metadata

- 169.254.169.254

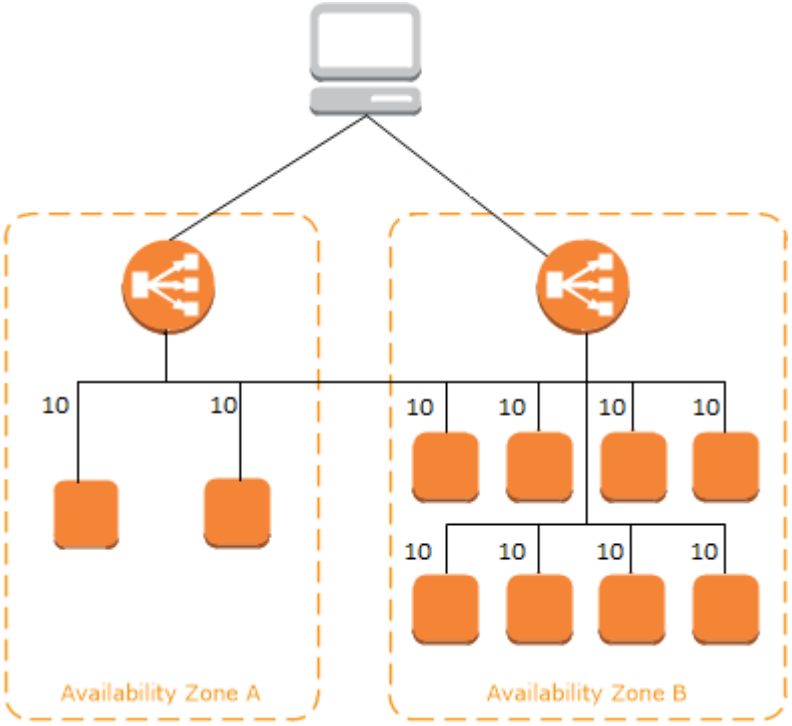
Load Balancear

- Health check
- Http - https
- Stickiness (manter sessão)
- HA entre AZs diferentes
- 100% gerenciado pela AWS
- Usa security group para segurança

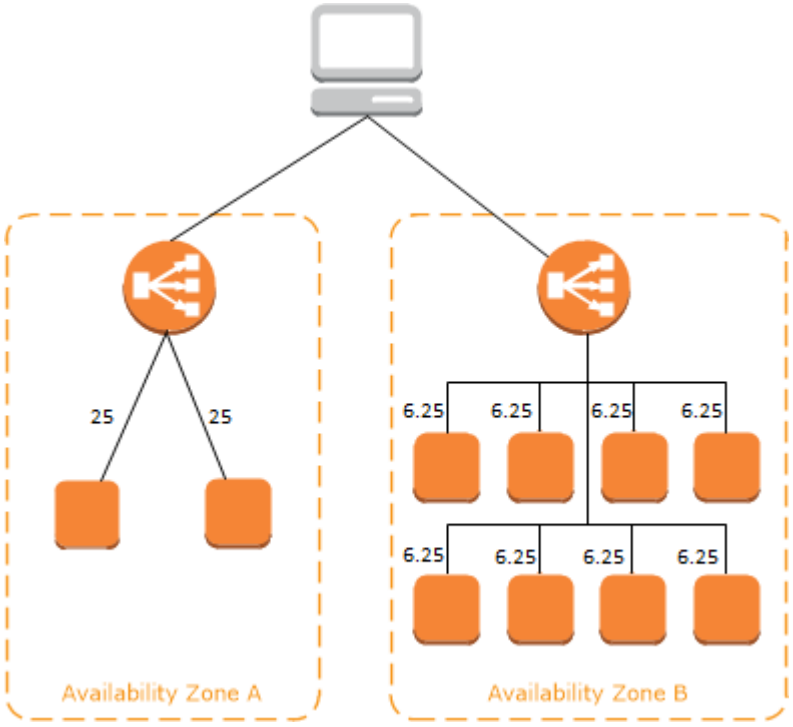
Tipos

- CLB (obsoleto)
- ALB - camada 7
- NLB - camada 4

- Health Check resposta OK - 200
- Balanceamento de carga entre zonas
    - Se o balanceamento de carga de zona cruzada estiver habilitado, cada um dos 10 destinos receberá 10% do tráfego. Isso ocorre porque cada nó do balanceador de carga pode rotear seus 50% do tráfego do cliente para todos os 10 destinos.



- Se o balanceamento de carga de zona cruzada estiver desativado:
    - Cada um dos dois destinos na Zona de disponibilidade A recebe 25% do tráfego. Cada um dos oito destinos na Zona de Disponibilidade B recebe 6,25% do tráfego.



- Os balanceadores de carga de aplicativos, o balanceamento de carga entre zonas está sempre ativado.
- Com os balanceadores de carga de rede e balanceadores de carga de gateway, o balanceamento de carga de zona cruzada é desabilitado por padrão. Depois de criar o balanceador de carga, você pode habilitar ou desabilitar o balanceamento de carga de zona cruzada a qualquer momento.

ALB

- Camada 7
- Máquinas diferentes ou não
- Rota baseada em (uri)
- Hostname
- Port mapping (dinâmico)
- Web socket
- X-forwarded-for (para aparecer o IP do cliente na aplicação)
- X-forwarded-proto
- X-forwarded-port
- Latência de 400ms

NLB

- Camada 4
- Suporta IP estático
- Suporta elastic IP
- Menor latência (100ms)

A escalabilidade do load balancer é lenta. Pode ser solicitado um warm up para a Aws

Erros no LB

- 4xx - gerado pelo cliente
- 403 - 'Forbidden' error. You configured an AWS WAF web access control list (web ACL) to monitor requests to your Application Load Balancer and it blocked a request.
- 500 - 'Internal server' error. There are several reasons for their error: A client submitted a request without an HTTP protocol, and the load balancer was unable to generate a redirect URL, there was an error executing the web ACL rules.
- 503 - 'Service unavailable' error. This error in ALB is an indicator of the target groups for the load balancer having no registered targets.
- 504 - Gateway timeout' error. Several reasons for this error, to quote a few: The load balancer failed to establish a connection to the target before the connection timeout expired, The load balancer established a connection to the target but the target did not respond before the idle timeout period elapsed.
- Erro de conexão, validar SG

ASG

- Scale in
- Scale out
- Garante número mínimo de EC2
- Registra EC2 no load Balancer

Launch configuration

- Customizar o lançamento de EC2
- Ami
- Tipo de instância

- EC2 userdata
- EBS volume
- SG
- Ssh key par

vPC

- vPC é definida por região
- Uma vPC default é criada em cada região e em cada AZ
- A vpc default contém todas as subnets públicas
    - Possuem auto associação de ipv4
    - Na tabela de roteamento, existe uma rota para o internet Gateway
- Composta de:
    - Subnets
    - Internet Gateway
    - vPC
    - Nat Gateway
    - Hardware vpn connection
    - Customer Gateway
    - Router
    - vPC endpoint
    - Peering connection
    - Egress para ipv6
- 5 ips são reservados para aws

Sg são stateful
Nacs são stateless

S3

- Bucket não cria hierarquia
- 100 por conta
- Cada objetivo recebe um Key

Colocar a imagem comprando as opções de S3

CLI

S3

- cp
- mb
- ls
- rb

EC2

- Não usar credenciais
- Usar role
- 1 EC2 = 1 role
- 1 role = n EC2
- 1 role = n políticas

As políticas precisam ser o mais granulares possíveis. Para testar, tem o policy simulator.

Dry run

Testa o comando para ver se tem permissão.

Exemplo:
Aws ec2 --dry-run

Para decodificar a msg do dry run, é necessário usar o sts-decode decode erros

Exemplo:
Aws sts decode -authorization -message --encode-message

Metadata

- Informações sobre a ec2
- Não é script de instalação
- Permite obter dados sobre EC2
- Http://169.254.169.254/latest/metadata
- Necessário incluir / no fim da chamada

SDK disponíveis

- Java
- .Net
- J
- Node.js
- Python
- Go
- Ruby
- C++

Ao usar SDK, a melhor prática é ter uma role para acesso.

Exponential Backoff

- Primeira tentativa falha, 1 Unidade de tempo
- Segunda tentativa falha, 2 Unidade de tempo
- Terceira tentativa falha, 4 Unidades de tempo
- Quarta tentativa falha, 8 Unidades de tempo

Isso deve ser usado quando ocorre uma falha na chamada da API.

Profiles

Aws configure --profile
Aws S3 ls (default)
Aws S3 ls --profile nome-profile (acessar outra conta)

Beanstalk

O Elastic Beanstalk é compatível com aplicações desenvolvidas em Go, Java, .NET, Node.js, PHP, Python e Ruby.

Versões compatíveis da plataforma

Todas as versões atuais da plataforma estão listadas em Plataformas compatíveis com Elastic Beanstalk no guia Plataformas do AWS Elastic Beanstalk.

Docker
Docker de vários contêineres
Docker pré-configurado
Go
Java SE
Tomcat
.NET Core no Linux
.NET no Windows Server
Node.js
PHP
Python
Ruby

Aplicativo
Uma aplicação do Elastic Beanstalk é uma coleção lógica de componentes do Elastic Beanstalk, incluindo ambientes, versões e configurações de ambiente. No Elastic Beanstalk, uma aplicação é conceitualmente semelhante a uma pasta.

Versão do aplicativo
No Elastic Beanstalk, uma versão da aplicação se refere a uma iteração rotulada específica do código implantável de uma aplicação Web. Os aplicativos podem ter várias versões e cada uma delas é única.

Em um ambiente em execução, é possível implantar qualquer versão do aplicativo já carregada no aplicativo ou fazer upload e implantar imediatamente uma nova versão do aplicativo.
Você pode fazer upload de várias versões do aplicativo para testar diferenças entre uma versão do seu aplicativo Web e outra.

Ambiente
Um ambiente é um conjunto de recursos da AWS que executam uma versão do aplicativo. Cada ambiente executa somente uma versão do aplicativo por vez, no entanto, você pode executar a mesma versão ou diferentes versões do aplicativo em vários ambientes ao mesmo tempo.

Nível do ambiente
O nível de ambiente designa o tipo de aplicação que o ambiente executa e determina quais recursos são provisionados pelo Elastic Beanstalk para oferecer suporte a ele. Um aplicativo que atende a solicitações HTTP é executado em um nível de ambiente de servidor da web. Um ambiente de back-end que extrai tarefas de uma fila do Amazon Simple Queue Service (Amazon SQS) é executado em uma camada de ambiente de operador.

Configuração de ambientes
Uma configuração de ambiente identifica um conjunto de parâmetros e configurações que definem como um ambiente e seus recursos associados se comportam.

Configuração salva
Uma configuração salva é um modelo que você pode usar como um ponto de partida para a criação de configurações exclusivas de ambiente.

Plataforma
Uma plataforma é uma combinação de um sistema operacional, tempo de execução da linguagem de programação, servidor Web, servidor de aplicações e componentes do Elastic Beanstalk.

.ebextensions

Tipos de Deploy

Supported deployment policies

Deployment policy	Load-balanced environments	Single-instance environments	Legacy Windows S environments†
All at once	✓ Yes	✓ Yes	✓ Yes
Rolling	✓ Yes	✗ No	✓ Yes
Rolling with an additional batch	✓ Yes	✗ No	✗ No
Immutable	✓ Yes	✓ Yes	✗ No
Traffic splitting	✓ Yes (Application Load Balancer)	✗ No	✗ No

- permite full controle sobre os recursos
- paga o que provisionar
- uma aplicação possui
    - várias versões
    - vários ambientes (dev/prod)
- dois tipos
    - web servers
    - workers - para trabalhar com sqs, por exemplo
- tipos de deploy:
    - all at once - nova versão de uma única vez, simultaneamente em todas as instâncias
        - bom para dev
        - rápido o deploy
        - atualiza tudo simultaneamente
        - causa indisponibilidade
        - não tem custo adicional
    - rolling - atualiza um conjunto de instâncias, depois vai para outro grupo
        - aplicação fica com duas versões
        - ambiente fica com capacidade reduzida, durante a atualização
        - não é ideal para ambientes sensíveis
        - sem custo adicional
        - muito tempo para deploy
    - rolling with additional batch - cria novas instâncias, depois elimina um grupo e assim por diante
        - aplicação continua com a capacidade completa
        - permite determinar o tamanho do grupo
        - baixo custo adicional
        - muito tempo para deploy
        - bom para ambientes produtivos
        - aplicação fica com duas versões em execução
    - immutable - cria novas instâncias em um novo ASG, depois que estiver tudo OK direciona o tráfego para o novo ASG

        - sem downtime
        - alto custo, pois duplica as instâncias
        - deploy longo
        - rollback rápido
        - bom para produção
    - b/g - cria um novo stage e faz o deploy
        - pode integrar com route53 para testar o novo stage
        - sem downtime
        - pequeno custo adicional
- RDS, para uso em produção é recomendável criar o RDS de forma separada e parametrizar os parâmetros
    - RDS_HOSTNAME
    - RDS_PORT
    - RDS_DB_NAME
    - RDS_USERNAME
    - RDS_PASSWORD

CI/CD

Code	Build/test	Deploy/provisioning
Codecommit Github	Codebuild Jenkins(CI)	Beanstalk Codedeploy

O code pipeline é um orquestrador.

Codecommit

- Controle de versão - Git
- Vantagens
    - Colaboração
    - Bkp de código
    - Disponibilidade
    - Auditável
- Repositórios são privados
- Sem limite de armazenamento
- Gerenciado pela aws
- Alta disponibilidade
- Código disponível na conta Aws
- Criptografia e controle de acesso
- Íntegra com Jenkins e codebuild
- Autenticação
    - Ssh
    - HTTP/https
    - Mfa
- Autorização
    - Role
    - IAM policies
    - Suporta apenas políticas baseadas em identidade, não políticas baseadas em recursos.
- Criptografia
    - Kms
    - HTTPS
    - Cross accounts via role com sts
- Notificações
    - Sns
    - Lambda

    - Triggers
        - Branch deletada
        - Update Master
        - Build system fora da aws
        - Gatilho para análise de código
    - Cloudwatch
        - Event rule para sns
        - Gatilho para atualizações de solicitação de pull (criadas / atualizadas / excluídas / comentadas).
        - Confirme eventos de comentário.
        - As regras de eventos do CloudWatch vão para um tópico do SNS.

Codepipeline

- CD
- Fluxo passo a passo
- Source
    - Github
    - Codecommit
    - S3
- Build: codebuild/Jenkins
- Load tests
- Deploy
    - Codedeploy
    - Beanstalk
    - Cloudformation
    - Ecs
- Stage
    - Serial/paralelo
    - Build/test/deploy
    - Aprovação: manual
- Conceitos
    - Pipeline
        - Workflow que escreve como ocorre o avanço entre as etapas do processo
    - Artefatos
        - Arquivo ou mudança que será trabalhada ou por ações ou por estágios da pipeline
        - Em cada estágio podem ser criados novos artefatos
        - São passados para o próximo estágio ou armazenados no S3
    - Estágios
        - Etapas que a pipeline passa
        - Podem ter ações sequenciais ou paralelas
        - Podem ter aprovações manuais
    - Ações
        - Executam algo nos artefatos
    - Transições
        - Processo entre um estágio e outro

Codebuild

- Gerenciado pela aws
- Alternativa ao jenkins
- Cobrado pelo tempo de build

- Usa docker para executar a rotina
- Segurança
    - Kms
    - IAM build
    - Vpc
    - Cloudtrail
- Arquivo de instruções é o buildspec.yml
- Log
    - S3
    - Cloudwatch
- Métricas para monitorar
- Cloudwatch alarm para detectar falhas e enviar notificações
- Sns
- Suporta
    - Java
    - Ruby
    - Python
    - Go
    - Node.js
    - Android
    - . Net core
    - Php
    - Docker (para outros personalizados)

Buildspec.yml

- Fica na raiz do repositório
- Define variáveis
    - Plain text
    - Secure Secrets (ssm - parameter store)
- Phases
    - Install - instala dependências
    - Prebuild - antes do processo de build
    - Build - execução do processo de build
    - Postbuild - exemplo: cria um arquivo .ZIP
- Artifacts
    - Gerado no S3 com kms
- Cache
    - Objetivo acelerar builds futuros

CodeStar

CodeStar fornece uma interface de usuário unificada, permitindo que você gerencie facilmente suas atividades de desenvolvimento de software em um só lugar.

Cada projeto AWS CodeStar vem com um painel de gerenciamento de projeto, incluindo um recurso integrado de rastreamento de problemas desenvolvido pelo Atlassian JIRA Software.

O cenário requer um conjunto de ferramentas de desenvolvimento unificado e menciona a colaboração entre os membros da equipe, sincronização e gerenciamento centralizado do pipeline de CI / CD, que será CodeStar em vez de CodePipeline ou CodeCommit.

- Suporta
    - Java,
    - JavaScript,

    - PHP,
    - Ruby,
    - Python.
- Colaboração entre os times de desenvolvimento
- Suporta vários ide
- Integra com
    - CodeCommit.
    - CodeBuild.
    - CodeDeploy.
    - CodePipeline.
    - CloudWatch
- É gratuito

Cloud9

- Ide baseado em Browser
- Disponibiliza ferramentas integradas e um Shell

Codedeploy

- Ferramenta de deploy
- deploy no EC2
    - Faz o blue/green
- Deploy na lambda
    - Faz o deploy atualizando a versão da lambda
    - Pode escolher como irá ocorrer o canary release em relação a distribuição do tráfego
        - Linear
        - Ou completo
- Deploy no ecs
    - Faz o deploy das tarefas no ecs
    - Faz BLUE/Green
    - Faz a redistribuição do tráfego das tarefas antigas para as novas tarefas criadas
    - A tarefa antiga é encerrada
- semelhante ao Ansible, Chef, etc
- pode usar em ambientes on premise (para isso precisa de um agente)
- arquivo de configuração appspec.yml
- source, pode ser git, s3
- permite agrupar instâncias pelo Deployment Group (DEV/TEST/PROD)
- Blue/Green só funciona com EC2
- Suporta lambda deployment
- Pode ser integrado ao codepipeline
- Tipos de deploy
    - In-place
        - Interrompe a aplicação atual, e cria uma nova
        - Ecs e lambda, não usa essa técnica
    - BLUE/Green
        - Cria uma nova versão
        - Faz o deploy nesta nova versão
        - Efetua uma validação/checagem
        - Após estar ok, ocorre o direcionamento para a nova versão
        - Não funciona com instância On premise
- componentes:
    - application: nome único
    - Compute plataform: EC2/On premise/lambda

    - deployment configuration: regra de sucesso ou falha
        - ec2/local: número mínimo de instâncias saudáveis
        - lambda: recebimento de tráfico
    - deployment group: grupo de instâncias com uma tag
    - deployment type:
        - in place
        - blue/green
    - Application /revision: código da aplicação + appspec.yml
    - service role: role necessário para o deploy (precisa ter permissão)
    - target revision: destino da versão da aplicação

appspec.yml

- O nome do arquivo AppSpec para uma implantação EC2 / On-Premises deve ser appspec.yml. O nome do arquivo AppSpec para uma implantação Amazon ECS ou AWS Lambda deve ser appspec.yaml
- File Section: como copiar os arquivos origem -> destino
- hooks: conjunto de instruções para deploy (pode ter um time out)
- sequência de execução:
    - application stop
    - download bundle
    - beforeinstall
    - install
    - afterinstall
    - applicationstart
    - validateservice

- Config
    - One at time: 1 a 1, falhou para.
    - Half of time: 50%
    - All at once: rápido, bom para dev
    - Custom: personalizados
- Failures
    - Instância fica "failures status"
    - Novo deploy começa na instância com falha
    - Rollback: deploy código anterior ou algum alternativo
- Deployment target
    - Ec2 com tags
    - Asg
    - Mix de asg/tags para criar segmentos de deploy
    - Personalização de Scripts com Deployment_group_name com variáveis de ambiente

Cloudformation

- IaC
- Automação
- Forma declarativa para criar recursos
- Não precisa especificar a ordem para criar os recursos
- Aumenta a produtividade
- Template fica armazenado no S3
- Novas versões = novo upload para o S3
- Possui
    - Resource - é o que você vai criar

    - Parameter - variáveis parâmetros que serão utilizados
    - Mapping - constantes utilizadas
    - Output - saída com os dados que foram criados. Ideal para ser usado como entrada em outros templates
    - Conditionals - condições
    - Metadata

- Resource
    - Obrigatório
    - Formato:
        - Aws::nome-profile::data-type
        - Aws::ec2::instance
- Parameter
    - Reutilizar parâmetros definidos pelo usuário
    - Tipos
        - String
        - Number
        - Comma delimitar list
        - List <type>
        - Awa parameter
    - Description
    - Constraints
    - Exemplos
        - Fn:ref = !Ref
        - Pseudo parameter
            - Aws::region (sa-east-1)
            - Aws::stackid (arn….)
- Mapping
    - Valores fixo
    - Exemplo definição
        - Mappings:
            - Regionmap
                - us-east-1
                    - "IAM": "1234"
                - us-weast-1
                    - "IAM": "321"
        - Exemplo de uso
            - ImageId:FindInMap
                - Regional
                - !Ref 'aws::region"
                - IAM
- Output
    - Declara o valor para ser usado por outra stack
    - Não pode deletar uma stack, se o valor estiver sendo utilizado
- Condition
    - Cris uma condição
    - 5 funções
        - Fn::and
        - Fn::Equals
        - Fn::if
        - Fn::not
        - Fn::or
- Transform
    - Usado para Sam
- Funções importantes no cloudformation
    - Fn:: Ref - faz referência
    - Fn::GetAtt - obtém informações sobre um atributo
    - Fn::FindInMap - retorna o valor de uma key
    - Fn::ImportValue - importa saída de uma outra stack

    - Fn::Join - concatenar valores
    - Fn::Sub - substitui texto
    - Condition Functions - condições
- Rollback
    - Falha: tudo volta como era
    - Padrão é estar habilitado
    - Pode ser desabilitado
- Stacksets
    - Para depois em múltiplas contas

Monitoring

Cloudwatch (metrics)

- Variáveis monitoradas
    - Cpu
    - Memória
- Dimension, atributos de metric
    - Instance ID
    - Limite de até 10
- Tudo tem timestamp
- Período de coleta
    - Padrão: 5 minutos
    - Detail monitoring: 1 minuto
- Memória RAM não é enviado nenhum dado
    - Para monitorar, precisa usar custom metric e enviar os valores
- Custom metric
    - Padrão 1 minuto
    - High resolution de 1 segundo
    - Usa API call PutMetricData
    - Recomenda-se o uso do exponential Backoff

Cloudwatch (alarm)

- Aciona notificações para qualquer métrica
- Pode enviar para
    - Asg
    - EC2
    - Sns
- Opções de alarmes
    - %
    - Max
    - Min
    - Sampling
- Tipos de states
    - Ok
    - Alarm
    - Insufficient_data
- Período
    - Tempo (expresso sempre em segundos)
    - High resolution
        - 10 segundos
        - 30 segundos

Cloudwatch (logs)

- App envia log via sda
- Coleta logs de serviços
- Pode enviar logs para
    - S3
    - Elk
- Pode usar filtros
- Log group
- Log stream: log file
- Política de expiração
- A aplicação precisa ter permissão para enviar log
- Criptografia com kms

Cloudwatch (events)

- Schedule cron jobs
  Event Pattern
- Trigger

CloudTrail

- Habilitado por padrão
- Ideal para
    - Governança
    - Compliance
    - Auditoria
- Histórico de eventos e API calls
- Pode enviar log para o cloudwatch
- dados armazenados são todos criptografados com KMS
- os dados com os logs são publicados a cada 15 minutos
- Dois tipos
    - aplicado a todas as regiões
        - o CloudTrail registra eventos em cada região e entrega os arquivos de log de eventos do CloudTrail para um bucket S3 que você especificar. Se uma região for adicionada após a criação de uma trilha que se aplica a todas as regiões, essa nova região será incluída automaticamente e os eventos nessa região serão registrados.
    - aplicado a uma região
        - registra os eventos apenas naquela região. Em seguida, ele entrega os arquivos de log de eventos do CloudTrail para um bucket do Amazon S3 que você especificar. Você só pode criar uma trilha de região única usando o AWS CLI. Se você criar trilhas únicas adicionais, pode fazer com que essas trilhas entreguem arquivos de log de eventos do CloudTrail para o mesmo bucket do Amazon S3 ou para buckets separados. Esta é a opção padrão quando você cria uma trilha usando o AWS CLI ou a API CloudTrail.
- Cloud trail para organization

    - Se você criou uma organização no AWS Organizations, também pode criar uma trilha que registrará todos os eventos de todas as contas da AWS nessa organização. Isso é conhecido como trilha de organização. As trilhas da organização podem se aplicar a todas as regiões da AWS ou uma região. Os rastreamentos da organização devem ser criados na conta de gerenciamento e, quando especificados como aplicáveis a uma organização, são automaticamente aplicados a todas as contas de membros da organização. As contas dos membros poderão ver a trilha da organização, mas não podem modificá-la ou excluí-la. Por padrão, as contas-membro não terão acesso aos arquivos de log da trilha da organização no bucket do Amazon S3.

Xray

- Ideal para debug em prod
- Visualização fim a fim
- Análise visual
- Tshoot (identificar gargalo)
- Dependências em arquitetura de microservices
- Revisa o comportamento
- Encontra erros e exceções
- Identifica usuários impactados
- Compatível com:
    - Lambda
    - Ecs
    - Elb
    - API Gateway
    - Ec2 ou server on premise
- Tracing (ponto a ponto)
- Para usar, é preciso usar SDK na aplicação
- Linguagens
    - Node
    - Nada
    - C#
    - Python
    - Go

Messaging

- Síncrono (ap1 - ap2)
- Assincrono (ap1 - queue - ap2)
    - Sqs
    - Sns
    - Kinesis

Sqs

Produtor -------- queue --------- consumidor

- Standard queue

- 1 msg/segundo até 10.000 msg/segundo
- Retenção é de 4 dias
- Retenção máxima de 14 dias
- Sem limite de msg na fila
- Latência menor que 10ms
- Escalonamento horizontal
- Tamanho máximo de 256 kb
- Delay queue
    - A msg fica disponível para ser consumida após 15 minutos
    - O padrão é 0 segundos
- Producing mensagem
    - Body
    - Attributes (opcional) key/value
    - Opção de delay delivery
    - Retorno
        - Message identified
        - MD5 hash body
- Consuming mensagem
    - Máximo de 10 msg por vez
    - O processamento sempre ocorre dentro do visibility time out
        - Exemplo: depois de 30 segundos a msg volta para o final da fila e fica novamente disponível no pool para ser consumida
    - O consumidor precisa deletar a msg
        - Message Id
        - Receipt handle
- Visibility timeout
    - Tempo padrão é de 30 segundos
    - Valor pode ser ajustado entre 0 e 12 horas
    - A aplicação pode solicitar mais tempo para processar a msg, para isso usa o ChangeMessageVisibility via API
    - A aplicação usa DeleteMessage via API para informar que a mensagem foi processada e pode ser apagada
- Dead letter queue
    - Ocorre quando o consumidor falha ao processar a msg dentro do visibility time out, e como consequência a msg volta para a queue
    - Pode ser definido a quantidade de vezes que uma msg pode voltar para o queue via Retrive Policy
    - Quando esse número é atingido, a msg vai para o dlq.
    - Uma vez que a msg foi para o dlq, ele deve ser processado nesta fila.
- Long polling
    - O consumidor faz a chamada no pool, e fica aguardando por um determinado tempo as msg chegarem, caso a fila esteja vazia.
    - Aumenta a eficácia da aplicação
    - Faz menos consultas a fila
    - O tempo pode variar entre 1 a 20 segundos
    - When the wait time for the ReceiveMessage API action is greater than 0, long polling is in effect. The maximum long polling wait time is 20 seconds. Long polling helps reduce the cost of using Amazon SQS by eliminating the

number of empty responses (when there are no messages available for a ReceiveMessage request) and false empty responses (when messages are available but aren't included in a response).

- Filas FIFO
    - Nome da fila deve terminar em .fifo
    - Tem baixo throughput
        - Até 3000/segundos
    - O consumo das mensagens ocorre na ordem de chegada
    - Sem delay por msg
    - MessageGroup
        - Não aceita msg duplicadas no mesmo
        - Agrupa msg usando MessageGroupID
        - Somente 1 worker por MessageGroup
        - MessageGroup é uma TAG
- Extended Client
    - Usado para mensagens maiores que 256kb
    - Usa um cliente Java para isso
    - Usa sqs + S3
- Segurança com sqs
    - Https
    - Sse com kms
        - Somente criptografa o body
    - Usa IAM policy
- Notações importantes
    - CreateQueue
    - DeleteQueue
    - Purge (fila toda)
    - SendMessage
    - ReceiveMessage
    - DeleteMessage
    - ChangeMessageVisibility
    - Batch API
        - SendMessage
        - DeleteMessage
        - ChangeMessageVisibility

Sns

Origem ---------- sns ---------- vários assinantes

- 10000000 assinantes por tópico
- 10000 tópicos
  Assinantes
    - Sqs
    - Http/https
    - Lambda
    - E-mail
    - Sms
    - Mobile notifications

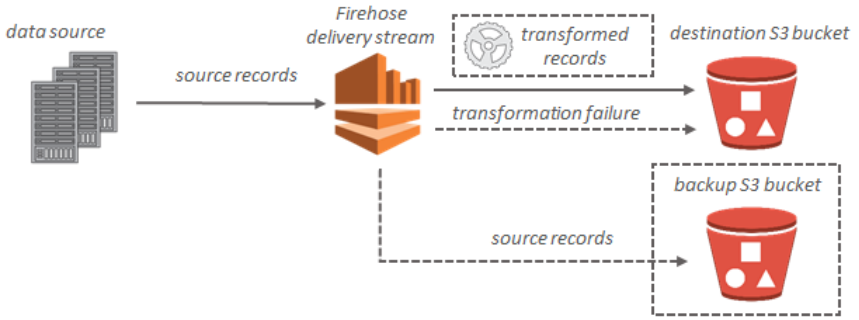
Kinesis

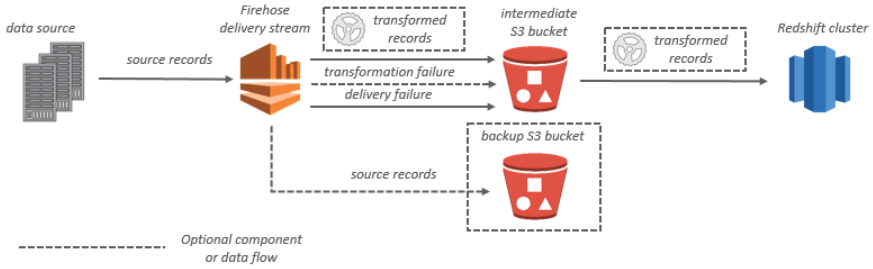
- Replica dados para 3 AZs
- Usado para
    - Log

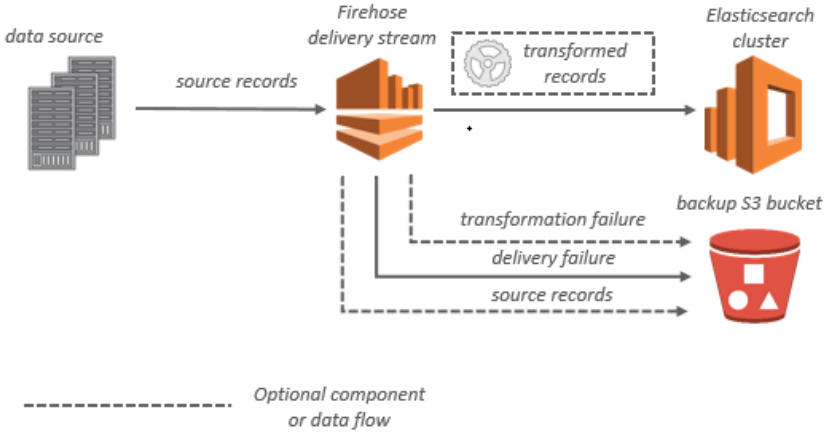
    - Big Data
- Kinesis stream
    - Baixa latência
    - Escalonável
- Kinesis analytics
    - Análise em tempo real
    - Usa sql
- Kinesis firehouse

Kinesis Firehouse

- Carrega o stream para o S3, redshift, elk, etc
- 60 ms de latência
- Suporta vários tipos de dados
- Pode invocar sua função Lambda para transformar os dados de origem de entrada e entregar os dados transformados aos destinos. Você pode habilitar a transformação de dados Kinesis Data Firehose ao criar seu fluxo de entrega.
- Os dados transformados são enviados do Lambda para o Kinesis Data Firehose. O Kinesis Data Firehose então o envia para o destino quando o tamanho do buffer de destino especificado ou o intervalo de buffer é alcançado, o que ocorrer primeiro.







Kinesis stream

- Dividido em suares/partitions
- Retenção dos dados de 1 a 7 dias
- Pode reprocessar o dado
- Vários aplicativos podem acessar o mesmo stream
- Processamento em tempo real
- Throughput escalonável
- Dado inserido, vira um dado imutável
- Shards
    - 1 stream = 1 ou + shards
    - 1mb/s ou 1000 msg por escrita
    - 2mb/s para leitura
    - Cobrança é por shared provisionado
    - Pode ter N shards
- Um aplicativo Kinesis Data Streams típico lê dados de um fluxo de dados como registros de dados
- Você pode enviar os registros processados para painéis, usá-los para gerar alertas, alterar dinamicamente preços e estratégias de publicidade ou enviar dados para uma variedade de outros serviços da AWS
- Como o tempo de resposta para o recebimento e processamento de dados é em tempo real, o processamento normalmente é leve.
- Alguns cenários de uso:
    - Ingestão e processamento acelerado de registro e alimentação de dados
    - Métricas e relatórios em tempo real
    - Análise de dados em tempo real
    - Processamento de fluxo complexo
    - Um uso comum é a agregação de dados em tempo real seguida pelo carregamento dos dados agregados em um armazém de dados ou cluster de redução de mapa.

Kinesis Analytics

- Processar e analisar dados de streaming usando SQL padrão.
- Quando usar:
    - Gerar análises de séries temporais
    - Feeds de painéis em tempo real
    - Create real-time metrics

Lambda

- Limite de 15 minutos para processamento
- Escala de forma automática
- é executada a partir de um evento
- a invocação pode ser:
    - síncrona: é executada e aguarda um retorno (é chamada passando o parâmetro Invoke)
    - assíncrona: é executada e não precisa aguardar o retorno (é chamada passando o parâmetro Event)
- Pode ter até 3gb de RAM
- mais memória, mais CPU
- Time out 3s máximo 15 minutos
- 1000 execuções simultâneas
- DLQ

    - Função falha, nova tentativa
    - Depois da segunda tentativa vai para DLQ
    - DLQ são
        - Sns
        - Sqs
    - Payload original é enviado para o DLQ
    - Boa forma para detectar erros
- Aliás
    - aponta para uma versão específica
    - pode ser alterado, ao contrário da versão
- Version,
    - várias vers]oes do código
    - ARN único por versão
    - $LATEST
    - 1
    - 2
- Cobrança é baseado na quantidade de invocação e tempo de processamento e memória
- handler
    - é o que é executado quando você executa a lambda
- layers
    - permite carregar bibliotecas ou códigos adicionais, separando do código principal
    - os arquivos ficam disponíveis em /opt
    - máximo de 5 layers

Step Functions

- orquestrador serverless
- pode ser usado com lambda e outros serviços AWS
- State Machine
    - é o workflow
- Tasks
    - é um fluxo de trabalho
    - cada fluxo de trabalho é um estado
- Standard workflows
    - pode executar por até 1 ano
    - possui auditoria
    - histórico de execução
    - debug visual
- Express workflow
    - pode executar por 5 minutos

RDS

- ACID (Atomicity, Consistency, Isolation and Durability)
- Online Transaction Processing (OLTP)
- Banco de dados transacional
- banco de dados estruturado
- backup automatizado
- push-button para autoscaling, replicação e redundância
- serviço gerenciado pela AWS
- Suporta
    - Aurora
    - mysql
    - mariaDB
    - Oracle
    - SQL Server
    - PostgreSQL

- Scalu up
    - Compute
    - storage
- Storage não diminui de tamanho
- tipo de Storage pode ser alterado, exceto para MSSQL
- Para muitas ações, é necessário criar uma nova instância a partir de um snapshot. esta instância já deve conter as configurações novas
- escalar storage, pode ter degradação de performance
- escalar compute, pode ter downtime
    - geralmente você pode optar por fazer a manutenção na janela de manutenção que o banco já tem configurado
- MultiAZ e Read réplicas

Multi-AZ Deployments	Read Replicas
Synchronous replication – highly durable	Asynchronous replication – highly scalable
Only database engine on primary instance is active	All read replicas are accessible and can be used for read scaling
Automated backups are taken from standby	No backups configured by default
Always span two Availability Zones within a single Region	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Database engine version upgrades happen on primary	Database engine version upgrade is independent from source instance
Automatic failover to standby when a problem is detected	Can be manually promoted to a standalone database instance

- Aurora
    - MySQL e PostgreSql
    - mysql -> cross-region
    - multi-master
    - serverless

Elasticache

- Redis e Memcache
- diminui a latência
- pode ser usado na frente do RDS para aumentar a performance
- usado para dados de sessão, assim como dynamoDB
- gerenciado pela AWS
- é acessado apenas via VPC
- Lazy Loading
    - carrega os dados para o cache, apenas quando necessário
    - evita encher o cache com dados não solicitados
    - com dados no cache, retorna para aplicação. caso os dados não estejam no cache ou tenham expirado, retornam null para a aplicação
    - a aplicação deve buscar os dados e guardar as informações no cache para a próxima consulta
    - dados em cache podem ficar obsoletos se nenhum parâmetro de TTL for configurado
- pode ter cache permitindo que o dado sempre esteja atualizado

Feature	Memcached	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Data persistence	No	Yes	Yes
Data types	Simple	Complex	Complex
Data partitioning	Yes	No	Yes
Encryption	No	Yes	Yes
High availability (replication)	No	Yes	Yes
Multi-AZ	Yes, place nodes in multiple AZs. No failover or replication	Yes, with auto-failover. Uses read replicas (0-5 per shard)	Yes, with auto-failover. Uses read replicas (0-5 per shard)
Scaling	Up (node type); out (add nodes)	Single shard (can add replicas)	Add shards
Multithreaded	Yes	No	No
Backup and restore	No (and no snapshots)	Yes, automatic and manual snapshots	Yes, automatic and manual snapshots

Dynamodb

- Banco nosql
    - suporta key-value
        - key é o nome do dado
        - value é o conteúdo em si
    - estrutura de documento
        - json
        - html
        - xml
- Possui 3 réplicas uma em cada AZ
- cross-region replication é opcional
- Escalável sem downtime
- Baixa latência
- dados são armazenados em SSD
- Suporta milhões de requisições por segundo
- IAM, gerencia todo o controle de acesso
    - Segurança
    - Autorização
    - Administração
    - suporta, identity-based policies:
        - anexando uma política de permissão a um usuário ou grupo
        - anexando uma política de permissão a uma role (cross-account)
    - não suporta políticas baseadas em recursos
    - a tabela é o recurso primário, index, stream são recursos adicionais a tabela e podem ter permissão específica
        - cada recurso possui seu próprio ARN
- Event com dynamodb stream
- Baixo custo
- Auto scalling é automático
- Composto por
    - Tabelas
        - Pode ter infinitas linhas
    - Primary Key que é obrigatório
    - Cada linha (item), possui
        - Atributos
        - Tamanho máximo de 400kb
- Tipo, provisioned
    - Read (rcu) precisa ser definido

    - Write (wcu) precisa ser definido
    - Nesse caso, Auto scalling podem ser usado quando o limite é atingido
- Wcu
    - 1wcu = 1 item de até 1kb por segundo
    - Itens maiores que 1kb, precisam de mais de 1wcu de consumo
    - Exemplos
        - 10 itens de 20kb cada
            - 10*2 = 20wcu
        - 6 itens de 4,5kb
            - 6*5 = 30wcu
        - 120 itens de 2kb por minuto
            - 2*120/60 = 4wcu
- Leitura Eventualmente consistente - padrão
    - A leitura ocorre após a escrita. Pode ocorrer de trazer informações antigas
- Leitura fortemente consistente
    - A leitura sempre trará as informações mais atuais do dado
    - pode ter uma latência maior que a leitura eventualmente consistente
    - não suporta Global secundário index
    - usa mais capacidade computacional para processar a requisição
    - usa o parâmetro
        - --consiste-read=true
- GetItem, query e Scan possuem um parâmetro (consistente read) que se definido para true, a leitura será fortemente consistente
- Rcu
    - toda a chamada na API para ler um dado, faz uma requisição de leitura
    - as requisições podem ser:
        - consistentes
        - eventualmente consistente
        - transacionais
    - 1rcu = 1 leitura forte ou 2 leituras eventualmente, por segundo para um item de até 4kb
    - Itens maiores que 4kb, precisam de mais rcu
    - Exemplos
        - 10 leituras fortes por segundo de 4kb
            - 10*4/4 = 10rcu
        - 16 leituras eventualmente por segundo de 12kb
            - (16/2)*(12/4) = 24rcu
- Partition
    - Dados são divididos em partições de storage e replicados.
    - tudo é gerenciado pela AWS
- Primary Key
    - é um atributo único. exemplo User ID
    - o valor de uma partition key gera um hash que irá determinar a partição, ou seja o local físico onde o dado será armazenado
- Composite Key
    - é uma combinação de partition key + sort key (chave composta: User ID + CPF)
- Throttling
    - Se exceder o limite, a msg "Provisioned throughtput exceeded exception
        - Possíveis causas

            - Hot key: partition Key com muitos acessos
            - Hot partition: itens muito grandes de rcu e/ou wcu
        - Soluções
            - Exponential Backoff
            - Distribuir partition Key o máximo possível
            - Se o problema for com rcu, pode-se usar dax
- Exemplos de chamadas na API
    - PutItem
    - UpdateItem
    - Conditional writes
    - DeleteItem
    - DeleteTable
    - BatchWriteItem
    - GetItem
    - ProjectionExpression
    - BatchGetItem
        - 100 itens
        - 16mb
    - Query
        - mais eficiente que o Scan
        - encontra um item na tabela, baseado na primary key e em algum atributo
    - Scan
        - retorna 1mb ou o limite
        - Consome muito rcu
        - Quanto mais filtros e ineficiente
- Local Secondary Index
    - Key alternativa para a tabela
    - Máximo de 5lsi por tabela
    - Sortkey consiste de 1 tabela
        - String
        - Number
        - Binary
    - Lsi deve ser definido na criação da tabela
    - não pode ser modificado, removido, adicionado depois da criação da tabela
- Global Secondary Index
    - Aumenta a velocidade das consultas para atributos que não são Key
    - Gsi = partition key + sort Key (opcional)
        - Gsi é uma nova tabela
        - Precisa definir rcu e wcu
    - Pode ser criado e modificado a qualquer momento
- Concurrency
    - Garante que o item não foi modificado antes de ser alterado
- Optimistic Locking
    - é uma estratégia que garante que dois acessos simultâneos não criem conflitos
        - update e delete ao mesmo tempo
- Dax
    - Accelerator
    - Escrita vai para o dax, e o dax escreve o dado
    - Latência de microsegundos
    - Resolve problema de hotkey (muitas leituras na partição)
    - 5 minutos de ttl (cache)
    - Máximo de 10 nodes por cluster

    - MultiAz (3 nodes por produção)
    - não atua com escrita, apenas leitura
    - diferença entre elasticache
        - DAX não tem lazy loading
        - DAX não modifica a aplicação
        - DAX menos gerenciamento
        - elasticache suporta vários bancos de dados
- Stream
    - Acionado por
        - Create
        - Update
        - Delete
    - Stream é lido por uma lambda
    - 24 horas de retenção
    - Pode implementar crossRegionReplication usando stream
- DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table and stores this information in a log for up to 24 hours. Applications can access this log and view the data items as they appeared before and after they were modified, in near-real time.
- You can also use the CreateTable or UpdateTable API operations to enable or modify a stream. The StreamSpecification parameter determines how the stream is configured:
- StreamEnabled — Specifies whether a stream is enabled (true) or disabled (false) for the table.
- StreamViewType — Specifies the information that will be written to the stream whenever data in the table is modified:
- KEYS_ONLY — Only the key attributes of the modified item.
- NEW_IMAGE — The entire item, as it appears after it was modified.
- OLD_IMAGE — The entire item, as it appeared before it was modified.
- NEW_AND_OLD_IMAGES — Both the new and the old images of the item.
- In this scenario, we only need to keep a copy of the items before they were modified. Therefore, the solution is to enable DynamoDB streams and set the StreamViewType to OLD_IMAGES.
- In general, Scan operations are less efficient than other operations in DynamoDB. A Scan operation always scans the entire table or secondary index. It then filters out values to provide the result you want, essentially adding the extra step of removing data from the result set.
- If possible, you should avoid using a Scan operation on a large table or index with a filter that removes many results. Also, as a table or index grows, the Scan operation slows. The Scan operation examines every item for the requested values and can use up the provisioned throughput for a large table or index in a single operation. For faster response times, design your tables and indexes so that your applications can use Query instead of Scan. (For tables, you can also consider using the GetItem and BatchGetItem APIs.)
- Additionally, eventual consistency consumers fewer RCUs than strong consistency. Therefore, the application should be refactored to use scan APIs with eventual consistency.
- Cálculos
    - Read capacity unit (RCU):

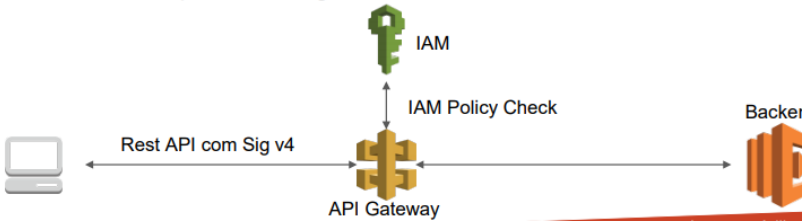
- Each API call to read data from your table is a read request.
- Read requests can be strongly consistent, eventually consistent, or transactional.
- For items up to 4 KB in size, one RCU can perform one strongly consistent read request per second.
- Items larger than 4 KB require additional RCUs.
- For items up to 4 KB in size, one RCU can perform two eventually consistent read requests per second.
- Transactional read requests require two RCUs to perform one read per second for items up to 4 KB.
- For example, a strongly consistent read of an 8 KB item would require two RCUs, an eventually consistent read of an 8 KB item would require one RCU, and a transactional read of an 8 KB item would require four RCUs.
- Write capacity unit (WCU):
- Each API call to write data to your table is a write request.
- For items up to 1 KB in size, one WCU can perform one standard write request per second.
- Items larger than 1 KB require additional WCUs.
- Transactional write requests require two WCUs to perform one write per second for items up to 1 KB.
- For example, a standard write request of a 1 KB item would require one WCU, a standard write request of a 3 KB item would require three WCUs, and a transactional write request of a 3 KB item would require six WCUs.
- To determine the number of RCUs required to handle 3 strongly consistent reads per/second with an average item size of 7KB, perform the following steps:
  - 1. Determine the average item size by rounding up the next multiple of 4KB (7KB rounds up to 8KB).
  - 2. Determine the RCU per item by dividing the item size by 4KB (8KB/4KB = 2).
  - 3. Multiply the value from step 2 with the number of reads required per second (2x3 = 6).
  - To determine the number of WCUs required to handle 1 standard write per/second, simply multiply the average item size by the number of writes required (7x1=7).

DynamoDB Feature	Benefit
Serverless	Fully managed, fault tolerant, service
Highly available	99.99% availability SLA - 99.999% for Global Tables!
NoSQL type of database with Name / Value structure	Flexible schema, good for when data is not well structured or unpredictable
Horizontal scaling	Seamless scalability to any scale with push button scaling or Auto Scaling
DynamoDB Streams	Captures a time-ordered sequence of item-level modifications in a DynamoDB table and durably stores the information for up to 24 hours. Often used with Lambda and the Kinesis Client Library (KCL)
DynamoDB Accelerator (DAX)	Fully managed in-memory cache for DynamoDB that increases performance (microsecond latency)
Transaction options	Strongly consistent or eventually consistent reads, support for ACID transactions
Backup	Point-in-time recovery down to the second in last 35 days; On-demand backup and restore
Global Tables	Fully managed multi-region, multi-master solution

API Gateway

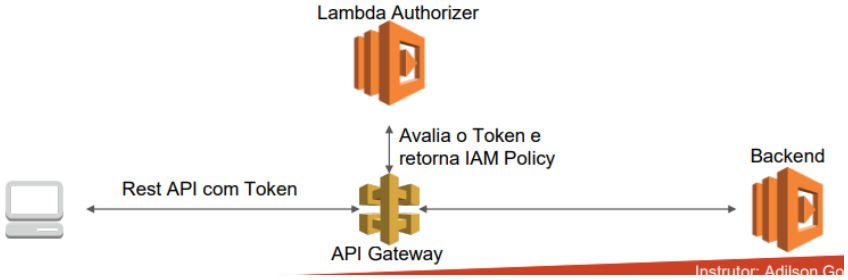
- Gerenciado pela aws
- Gerência versões de aplicações
- Gerencia ambientes (dev, hom, prod)
- Gerência segurança
    - Autenticação
    - Autorização
- Permite o uso de api Key
- Gerência request throttling
- Importa de swagger/openapi
- Transforma/valida requisições e respostas
- Gera sdk
- Cache para api response
- Íntegra com
    - Fora da vPC
        - Lambda
        - Endpoint EC2
        - Lb
        - IP público http
    - Dentro da vPC
        - Lambda
        - EC2
- Stages de deployment
    - Dev, hom, prod
    - Altera, salva e faz o deploy
- Stage variables
    - Variáveis do api gateway
- Mapping template
    - Usado para modificar request/response
        - Renomear parâmetros
        - Modificar body content
        - Adicionar header
        - Json para XML
        - Remoção de dados desnecessários
- Cache
    - Ttl 300 segundos - padrão
    - Ttl 0 - sem cache
    - Ttl 3600 segundos - máximo
    - Definido por stage
    - Pode ser criptografado
    - Capacidade é de 500mb a 237gb
    - Pode ser limpo por completo
    - Para invalidar o cache, pode-se usar:
        - Header cachecontrol.max-age=0 (necessário ter iam configurado)
- Cors
    - Precisa ser habilitado
- UsagePlan
    - Plano para cobrança de uso
        - Quotas
        - Api stage
        - Throttling
- Segurança
    - Sigv4

- 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 Heac



  - Lambda autorizer
    - (anteriormente conhecido como um autorizador personalizado) é uma função Lambda que você fornece para controlar o acesso à sua API. Um autorizador Lambda usa estratégias de autenticação de token de portador, como OAuth ou SAML. Antes de criar um autorizador API Gateway Lambda, você deve primeiro criar a função AWS Lambda que implementa a lógica para autorizar e, se necessário, autenticar o chamador.

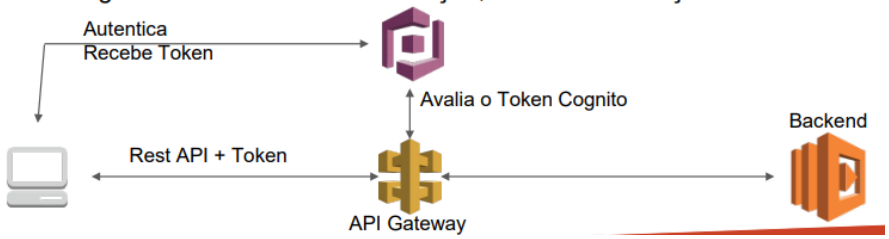
- Lambda Authorizer (Custom Authorizers)
- Usa AWS Lambda para validar o token no header que está sendo passado
- Opção para Cache do resultado da autenticação
- Lambda deve retornar uma IAM Policy para o usuário



  - Cognito

Cognito User Pools

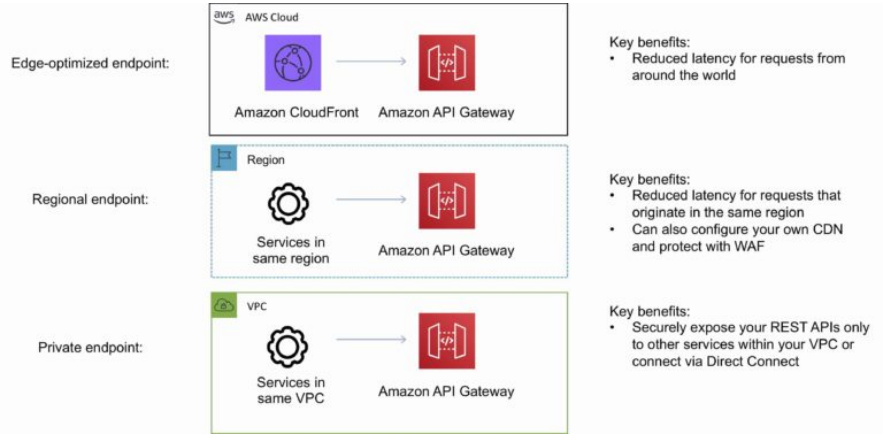
  - Cognito gerencia User Lifecycle
  - API gateway verifica a identidade automaticamente do AWS Cognito
  - Não é necessário nenhum implementação
  - Cognito só trabalha na autenticação, não na autorização

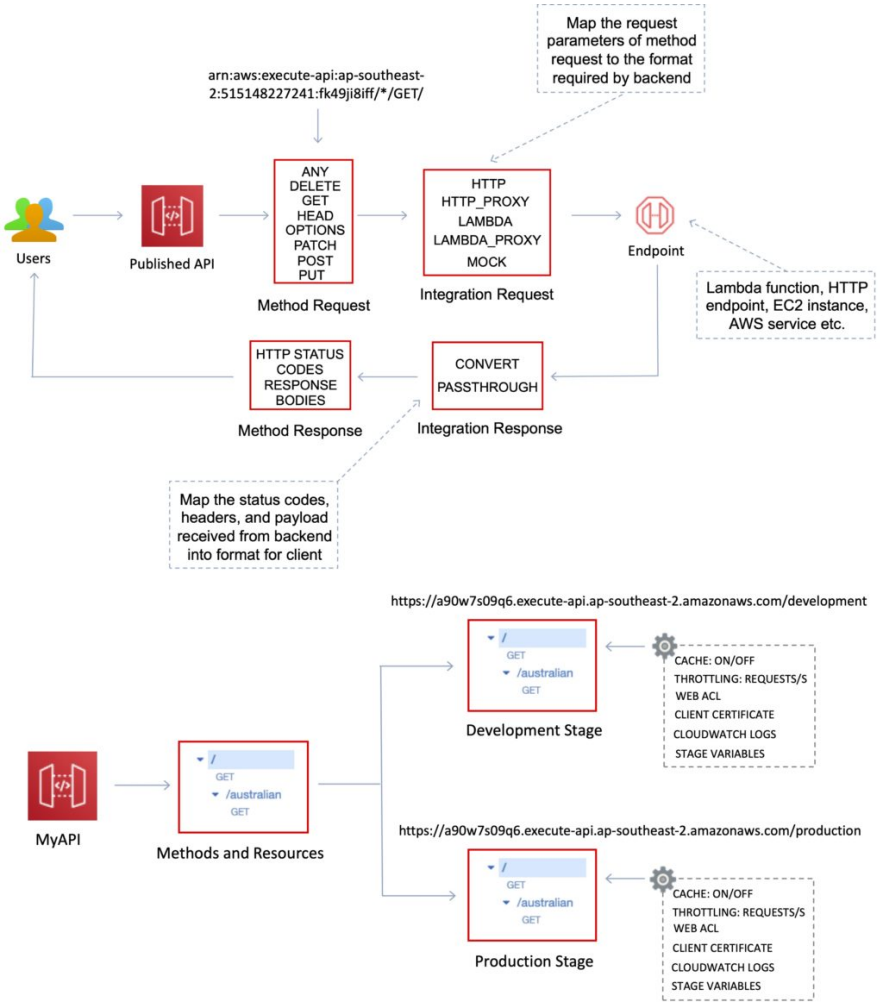


  - Userpool
    - Só autentica
    - Não autoriza
  - A Lambda authorizer (formerly known as a custom authorizer) is an API Gateway feature that uses a Lambda function to control access to your API.

- A Lambda authorizer is useful if you want to implement a custom authorization scheme that uses a bearer token authentication strategy such as OAuth or SAML, or that uses request parameters to determine the caller's identity.
- When a client makes a request to one of your API's methods, API Gateway calls your Lambda authorizer, which takes the caller's identity as input and returns an IAM policy as output.
- There are two types of Lambda authorizers:
  - A token-based Lambda authorizer (also called a TOKEN authorizer) receives the caller's identity in a bearer token, such as a JSON Web Token (JWT) or an OAuth token.
  - A request parameter-based Lambda authorizer (also called a REQUEST authorizer) receives the caller's identity in a combination of headers, query string parameters, stageVariables, and $context variables.
  - For this scenario, a Lambda authorizer is the most secure method available. It can also be used with usage plans and AWS recommend that you don’t rely only on API keys, so a Lambda authorizer is a better solution.

API Gateway Feature	Benefit
Support for RESTful APIs and WebSocket APIs	With API Gateway, you can create RESTful APIs using either HTTP APIs or REST APIs
Private integrations with AWS ELB & AWS Cloud Map	With API Gateway, you can route requests to private resources in your VPC. Using HTTP APIs, you can build APIs for services behind private ALBs, private NLBs, and IP-based services registered in AWS Cloud Map, such as ECS tasks.
Metering	Define plans that meter and restrict third-party developer access to APIs
Security	API Gateway provides multiple tools to authorize access to APIs and control service operation access
Resiliency	Manage traffic with throttling so that backend operations can withstand traffic spikes
Operations Monitoring	API Gateway provides a metrics dashboard to monitor calls to services
Lifecycle Management	Operate multiple API versions and multiple stages for each version simultaneously so that existing applications can continue to call previous versions after new API versions are published
AWS Authorization	Support for signature version 4 for REST APIs and WebSocket APIs, IAM access policies, and authorization with bearer tokens (e.g. JWT, SAML) using Lambda functions.

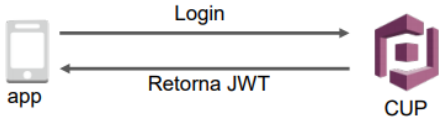




Cognito

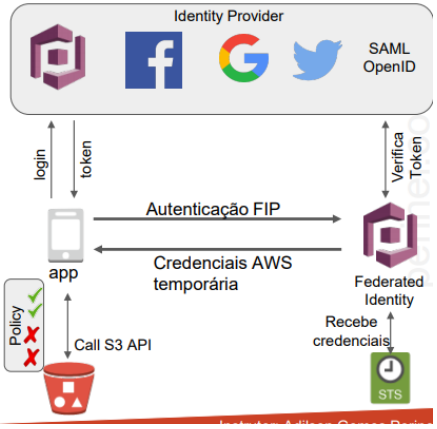
- Userpool
  - Sign in para aplicações
  - Pode ser integrado com api gateway
  - Gerenciado pela aws
  - Login simples
  - Pode ter mfa
  - Pode federar com Google, Facebook, saml, etc
  - Retorna um jwt (json Web token)

- 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



- Identity pool
  - Oferece credencial para acesso a recursos
  - Pode ser integrado com user pool
  - Fornece ao cliente, acesso.
  - Recebe credenciais aws, temporárias
  - Essa credencial, vem com um iam policy definida
  - Exemplos: gravar dados no s3

- 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



- Cognito Sync
  - Faz o Sync de dados do dispositivo com o cognito
  - Será substituído pelo appsync

Route53

- Dns público
- Dns privado
  - Relacionado a uma vpc
- Faz health check, para o destino
- Alias é um tipo de entrada de dns exclusiva da aws
- Políticas
  - Simples
  - Failover - usa health check (stand by)
  - Geolocation - ex. Europa, América. Idioma. Copyright
  - Geoproximidade - região mais perto
  - Latência, sempre a menor
  - Multivalue
  - Weighted, conforme o % determinado

ECS

- velocidade para usar docker rapidamente
- gerenciado pela AWS
- paga as EC2 e storage
- Cluster - grupo de EC2 instances
- container instance - EC2 que executa o ECS agent
- Grupo de EC2
- Em cada EC2, tem um agente
- Precisa de uma ami específica
- Task definition - blueprint que descreve como o docker será executado
  - Metadata em json
  - Define:
    - Imagename
    - Port binding
    - Vou
    - Memória
    - Rede
    - Iam roles
    - Variáveis de ambientes
- Task - um contêiner em execução, com base na task definition
  - pode ter iam role para acesso a outros recursos
  - recomendado é ter role específica
- Serviços
  - Quantidade de tasks

    - Trabalha junto com alb, nlb
- Service auto scaling
- capacity provider trabalha junto com auto scaling
- estratégia de depoy:
    - binpack: minimiza o número de instâncias em uso
    - random: substitui as tasks de forma randômica
    - spread: inicia as tasks com base em um valor especificado
        - instance id
        - az
- task placement
    - distinctInstance: coloca cada task em uma instância diferente
    - memberOf: coloca cada task com base no parâmetro especificado
- fargate
    - serverless
    - docker image somente do ecr e docker

Ecr

- Repo privado
- Acesso controlado via iam
- Comandos
    - Aws ecr get-login --region xxx
    - Docker build -t meuapp .
    - Docker tag meuapp:latest idconta.ecr.aws/meuapp:latest
    - Docker push idconta.ecr.aws/meuapp:latest

SAM

- faz o deploy de recursos, usando uma linguagem simplificada
- deploy:
    - lambda
    - dynamodb tables
    - api gateway
    - s3
- usa um arquivo yaml com as instruções para deploy
- o template tem que iniciar com Transform: AWS::Serverless:2016-10-31

Segurança

- Criptografia
    - Em trânsito
        - Certificado ssl
    - Lado do servidor
        - Usa uma key
        - Essa Key precisa ser gerenciada
    - Lado do cliente
        - Dados são enviados criptografados
- Kms
    - Gerencia chaves de criptografia na aws
    - Serviço gerenciado pela aws
    - Funciona com quase todos os serviços da aws
    - Padrão de 4kb por chamada para criptografia
    - Acima de 4 kb, precisa usar envelope encryption

        - Ocorrendo no lado do cliente
    - Gerencia keys e policies
    - Usa cmk para Chaves
- Parameter store
    - Usado para conf de Secrets
    - Pode usar kms
    - Possui rastreabilidade de versões
    - Integrado com cloudwatch
    - Organiza as informações de forma hierárquica

System manager

AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. It is highly scalable, available, and durable.

You can store values as plaintext (unencrypted data) or ciphertext (encrypted data). You can then reference values by using the unique name that you specified when you created the parameter.

There are no additional charges for using SSM Parameter Store. However, there are limit of 10,000 parameters per account

- Access Advisor feature on IAM console -
    - ajuda a identificar as funções não utilizadas, o IAM relata o carimbo de data / hora usado pela última vez que representa quando uma função foi usada pela última vez para fazer uma solicitação AWS. Sua equipe de segurança pode usar essas informações para identificar, analisar e, em seguida, remover com segurança funções não utilizadas. Isso ajuda a melhorar a postura de segurança de seus ambientes AWS. Além disso, removendo funções não utilizadas, você pode simplificar seus esforços de monitoramento e auditoria, concentrando-se apenas nas funções que estão em uso.
- IAM Access Analyzer
    - ajuda a identificar os recursos em sua organização e contas, como buckets do Amazon S3 ou funções IAM, que são compartilhados com uma entidade externa. Isso permite que você identifique o acesso não intencional aos seus recursos e dados, o que é um risco à segurança.
- Amazon Inspector
    - é um serviço automatizado de avaliação de segurança que ajuda a melhorar a segurança e a conformidade dos aplicativos implantados na AWS. O Amazon Inspector avalia automaticamente os aplicativos quanto à exposição, vulnerabilidades e desvios das melhores práticas.