>>> >>> AWS

Name: Henrique Tsuyoshi Yara (OPUS-software)
Date: February 1, 2023

Figure: AWS logo

>>> Índice I

>>> **Índice II**

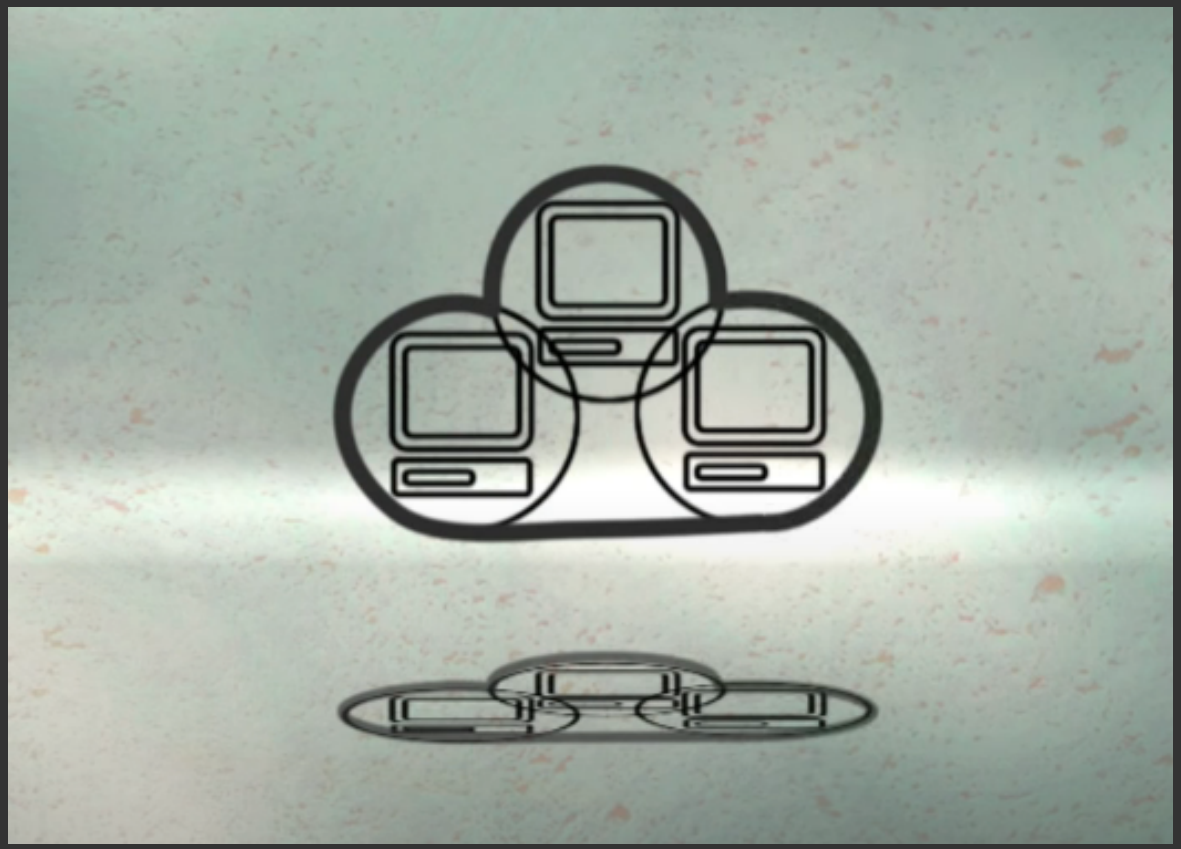
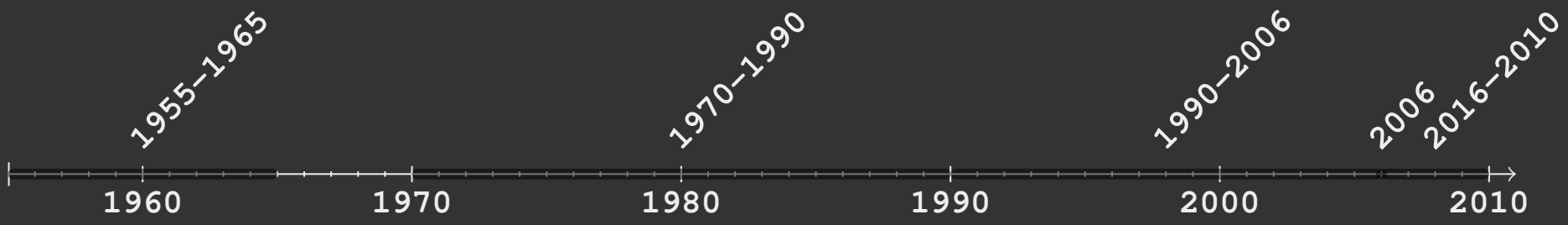


Figure: A cluster of servers drawn in a system diagram [1]

[1] Image source link

>>> Linha do tempo

História da computação em nuvem[6]



* (1955–1965) Problemas na infraestrutura de TI
* (1970–1990) Hypervisors e a internet
* (1990–2006) Internet para todos
* (2006–2006) Precipitation
* (2006–2010) Primeiros dias da computação em nuvem

Problemas na infraestrutura de TI (1955-1965)

>>> Problemas na infraestrutura de TI I

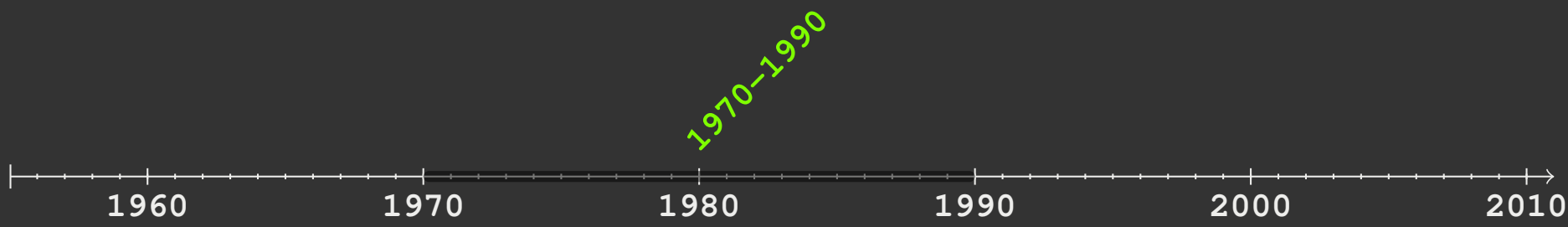
* 1942 - John Vincent Atanasoff construiu o computador ABC
* 1960 - Muito caro para aderir os computadores
  * Sala inteira para o servidor (Manter temperatura ideal, espaço, etc...)
  * Computador
  * Funcionários especializados
  * Problemas para adaptar o software

>>> Problemas na infraestrutura de TI II

* Empresas com poder aquisitivo conseguiram aderir os computadores
* 1961 - John MacCharty fez uma palestra no MIT
  * Computação poderia ser vendida como água ou eletricidade [2]
  * Mas seria necessário de uma grande evolução tecnológica

>>> Linha do tempo

Hypervisors e a internet
(1970–1990)



>>> Hypervisors e a internet I

* Diminiur os custos da adoção do computador
  * Múltiplos usuários podem compartilhar o mesmo armazenamento e o poder de processamento da CPU
* 1970 - Nasceu a tecnologia da virtualização
  * Um host pode ser particionado em VMs
  * Cada parte pode rodar um código independente

>>> Hypervisors e a internet II

* 1983 - A internet nasceu
  * Projeto ARPANet: comunicação de professores de universidades
* Arquitetura cliente-servidor conectados por cabos (Internet)
* O número de servers cresceram junto com as páginas web
  * Os servers se moveram para datacenters

>>> Linha do tempo

Internet para todos
(1990-2006)

>>> Internet para todos

* Empresas precisavam de datacenters maiores
* Problemas:
  * Ociosidade dos servers (off-season)
  * Escalabilidade manual e precisava de dispositivos físicos
    * Difícil acompanhar o crescimento da aplicação
  * Atualização e manutenção manuais
  * Grande distância dos servidores (UX ruim)

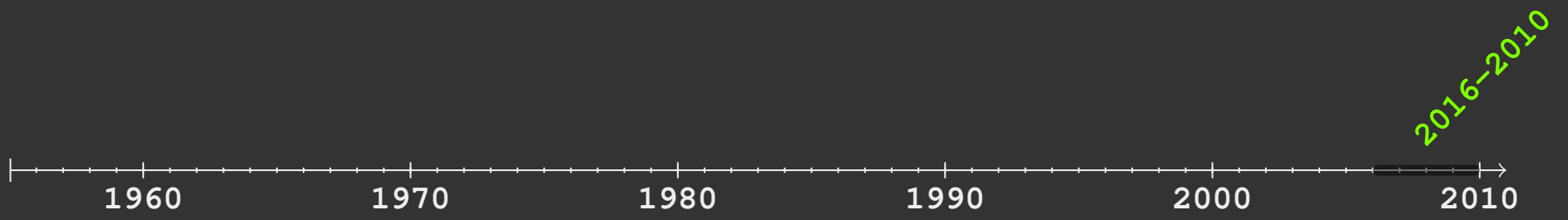
>>> Linha do tempo

Precipitation
(2006)

>>> Precipitação I

* Grandes players (*Google, Amazon, eBay, etc...*)
  * *Próprio data center*
  * *Centenas/milhares de servers de alta qualidade no mundo inteiro*
  * *Começar a alugar essas máquinas*
* *2006 - "Cloud Computing" introduzino como forma de aluguel de poder computacional*
* *AWS - primeira provedora de cloud*
* *Sonho de John McCharty foi realizado depois de 50 anos*

Primeiros dias da computação em nuvem
(2006-2010)



>>> Primeiros dias da computação em nuvem I

* Por 6 anos a AWS estabeleceu seu monopólio na área de nuvem
* Computação em nuvem → empresa focam no desenvolvimento
  * Os dados são criptografados e seguros
  * Dados são armazenados com redundância em nuvem
  * Escalabilidade
  * De fácil deploy
  * Alta disponibilidade

>>> Primeiros dias da computação em nuvem

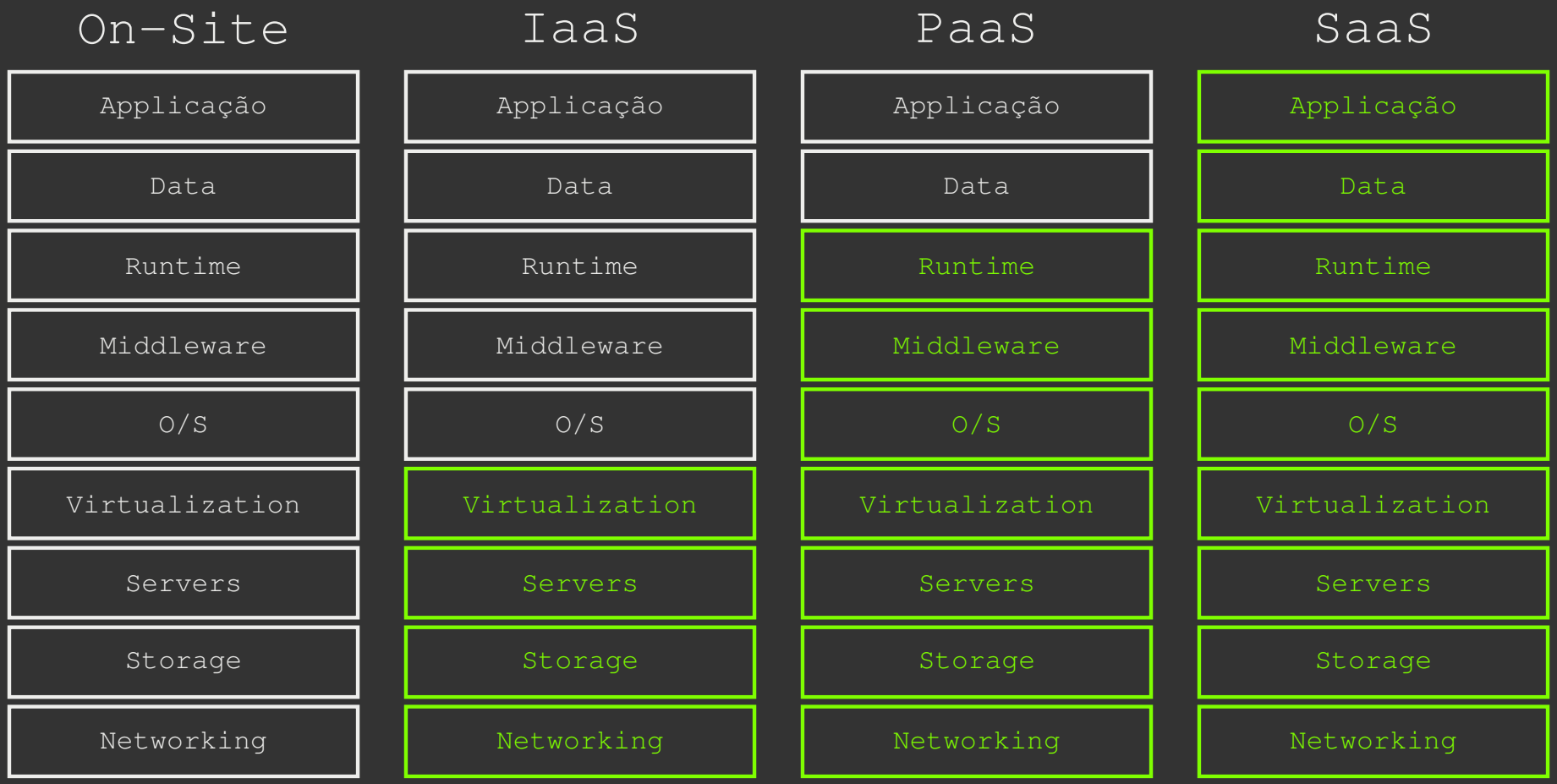


Figure: Imagem retirado do site da redhat[2]

[2] RedHat IaaS, PaaS e SaaS

>>> O que é cloud para o NIST

* Um modelo para habilitar o acesso por rede a um conjunto compartilhado de **recursos de computação** e precisa ser:
  * Ubíquo (Pode ser encontrado em todos os lugares)
  * Conveniente
  * Sob demanda
* **Recursos de computação:** Redes, servidores, armazenamento, aplicações e serviços
* Esses recursos devem ser provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços.

>>> 5 Características

* Conceito que reúne vários *softwares* e utiliza de virtualização
* Possui algumas características específicas (NIST):
    * Autoserviço sob demanda
    * Amplo acesso a rede
    * Pool de recursos
    * Rápida elaticidade
    * Serviços mensuráveis

>>> Auto-serviço sob demanda

* O consumidor pode providionar por conta própra **Recursos de computação**
* Não necessita da intervenção humana dos provedores de serviço

>>> Amplo acesso por rede

* Os **Recursos de computação** estão disponíveis através da rede
* São acessados através de mecanismos padronizados que promovem o uso por dispositivos, clientes leves ou ricos de diversas plataformas (Smartphones, tablets, laptops ou desktops)

>>> Agrupamento de recursos

* Os **Recusos de computação** do provedor são agrupados para atender a múltiplos consumidores em modalidade multi-inquilinos (Recursos físicos e virtuais diferentes dinamicamentes atribuídos e reatribuídos conforme a demanda dos consumidores)
* Há uma certa independência de localização geográfica, uma vez que o consumidor em geral não controla ou conhece a localização exata dos recursos fornecidos
* Mas pode ser capaz de especificar a localização em um nível de abstração mais alto (país, estado, datacenter)

>>> Elasticidade rápida

* Os recursos podem ser provisionados e liberados elasticamente, em alguns casos automaticamentes, para rapidamente aumentar ou diminuir de acordo com a demanda
* Para o consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ser ilimitados e podem ser alocados em qualquer quantidade e a qualquer tempo

>>> Serviços mensurado

* Os sistemas na nuvem automaticamente controlam e otimizam o uso dos recursos através de medições em um nível de abstração apropriado para o tipo de serviço (como armazenamento, processamento, comunicação de ree e contas de usuário ativas)
* A utilização de recursos pode ser monitorada, controlada e informada, gerando transparência tanto para o fornecedor como para o consumidor do serviço utilizado

>>> Origem AWS I

* 2006 - Amazon Web Services começou a oferecer infraestrutura de TI como forma de serviços web
  * Low Cost (Pay-as-you-go)
  * Agility and Instant Elasticity
  * Open and Flexible
  * Secure (PCI DSS Level 1, ISO27001, etc...)

>>> Origem AWS II

* Instância (2006):
    * 1.7 GHz Xeon Processor
    * 1.75 GB of RAM
    * 160 GB of local disk
    * 250 Mbps network bandwidth
* Instância (2019):
    * 4.0 GHz Xeon Processor (z1d instance)
    * 24 TiB of RAM (High Memory instances)
    * 60 TB of NVMe local storage (I3en.metal instances)
    * 100 Gbps network bandwidth

>>> Formas de acesso

* **Console:** Gerenciar a infraestrutura e os recursos da aws com uma interface web
* **SDK:** Simplifica o uso dos serviços da AWS provendo bibliotecas para os desenvolvedores
  * Tem suporte para: Java, .NET, C++, PHP, etc...
* **CLI:** Controla múltiplos serviços usando a linha de comando e é possível automatizar usando scripts

>>> Recursos

* Cada recurso vai ter um **Amazon Resource name** (Identificador único)

>>> Free Tier

* São recursos que podem ser usadas de graça na Amazon

>>> Calculadora

* É utilizada para calcular o custo total de algum recurso
  * Calculadora antiga
  * Calculadora nova

>>> Regiões

* Cada região tem um preço diferente
* Uma região é composta de zonas de disponibilidade
* Algumas regiões podem ter mais serviços que outras
* **OBS:** É bom saber se juridicamente a gente pode armazenar os dados fora do Brasil
  * Regiões e zonas de disponibilidade
  * Serviços regionais
* **OBS:** Tráfegos entre zonas de disponibilidade ou regiões podem acabar sendo cobrados

>>> Zonas de disponibilidade

* Compõem as regiões
  * Serviços regionais
* **OBS:** Tráfegos entre zonas de disponibilidade ou regiões podem acabar sendo cobrados

>>> Status AWS

* Para verificar o status das zonas de disponibilidade/regiões ou recursos
  * Status AWS
* **OBS:** Tráfegos entre zonas de disponibilidade ou regiões podem acabar sendo cobrados

>>> Pontos de presença

* Pontos de cache utilizado pela AWS (É possível usar *CNDs*)

>>> Virtual Private Cloud (VPC)

* VPCs são usadas para criar redes virtuais
* VPCs podem usar CIDR entre /16 até /28
* Cada região tem uma VPC padrão
    * Não é recomendado usar
* VPCs são isoladas entre si
    * Podem ser configuradas para se comunicarem
* **VPC wizard** tem algumas configurações pré-definidas de VPC
* Lembrar de verificar e configurar:
    * DHCP options set
    * DNS resolution
    * DNS hostname

>>> Conexões VPN

* **AWS Hardware VPN:** Conexão da VPC para uma rede remota usando IPsec hardware VPN
* **AWS Direct Connect:** Conexão privada dedicada da VPC para uma rede remota
* **AWS VPN CloudHub:** Múltiplos **AWS Hardware VPN** pelo VPC permitindo comunicação entre muitas redes remotas
* **Software VPN:** Conexão VPN usando uma instância EC2 rodando um software VPN

>>> Subnets I

* Subnets são uma parte da rede inteira
  * A rede pode ser dividida em subnets
  * Uma subnet pode ser dividida em subnets
* Cada subnet é como uma rede separada

137.207.32.2

Network ID Host ID

Figure: Subnet Addresses

>>> Subnets II

* Subnets são aplicadas em AZs
* Subnet:
    * Pública: recursos que não devem ser acessíveis pela internet
    * Privada: recursos que devem ser acessíveis pela internet

>>> NAT Gateway I

* Instâncias dentro de subnets privadas podem conectar com serviços fora da VPC, mas instâncias de fora não podem iniciar ocnexões com essas instâncias
  * Public: Instâncias em subnets privadas podem conectar com a internet
  * Private: Instâncias em subnets privadas podem conectar com outros VPCs
* Cobrança por hora de uso e quantidade em GBs de dados processados

>>> NAT Gateway II

Table: VPC NAT Gateway Vs NAT Instances on amazon EC2

	VPC NAT Gateway	
Availability	Highly available by default	Use s
Bandwidth	Bursts to 10 Gbps	Based on
Maintenance	Managed by AWS	
Security	NACLs	Sec
Port forwarding	Not supported	

>>> Internet Gateway

* Permite a comunicação do seu VPC com a internet
* São escaláveis horizontalmente, redundantes e tem alta disponibilidade por padrão
* Libera a entrada e a saída de determinado **Route Table**
* Não tem custo

>>> Route table

* Determina para onde o tráfico de rede é roteado
* Associa as **subnets**
* Se a **Route table** não tiver uma rota default ela não está pública
* Apenas uma route table por subnet
* Boa prática:
    * User route tables customizadas para cada subnet roteamento mais granularizado para os destinos

>>> Security Groups I

* Firewall virtual para controlar entrada e saída de tráfico (1 ou mais instâncias)
* Pode ser aplicado a um CIDR ou outro security group para situações de autoscaling
* Apenas regras de liberação
* Stateful: se o tráfego de entrada é permitido, a resposta de saída não precisa ser inspecionada/localizada e vice versa
* Por padrão todas os tráfegos de saída são permitidos
    * Modificar a saída traz complexidade para a aplicação e não é recomendada (Apenas se for preciso por compliance)

>>> Security Groups II

* Grande parte das empresas criam security groups para cada camada da aplicação

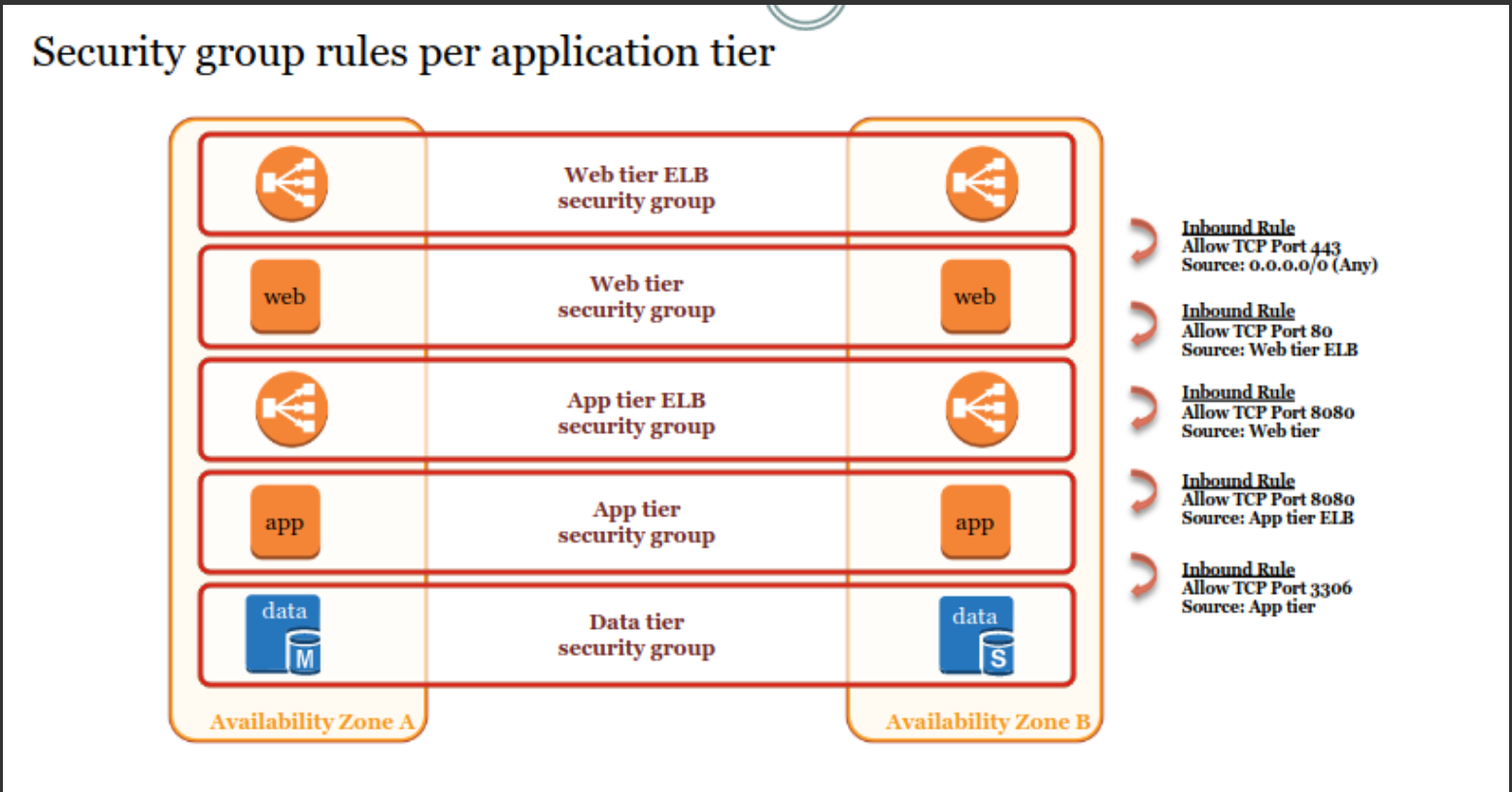


Figure: Security Group chaining diagram

>>> NetworkACL I

* Regra de segurança da rede (Como se fosse um *firewall*)
* Regras de liberação e negação
* Stateless: o tráfego de retorno deve ser explicitamente permitido pelas regras
* Aplica a todas as instâncias nas sub-redes
* São Firewalls virtuais opcionais que controlam a entrada e a saída e uma subnet
* stateless: Allow all incoming/outgoing traffic by default and use stateless rules to allow or deny traffic. "Stateless rules” inspect all inbound and outbound traffic and do not keep track of connections.
* Cada regra vai ter uma prioridade

>>> NetworkACL II

* É bom deixar um espaço entre cada regra para possíveis regras futuras (Ex: deixar 10 espaços entre cada regra)
* **OBS:** É bom liberar portas efêmeras (1024-65535). São usadas para comunicações de saída através do protocolo de rede TCP/IP

>>> Amazon VPC Flow Logs

* Captura detalhes do tráfico na VPC (Aceito e recusado)
* Pode ser habilitado para PVCs, subnets e ENIs
* Uso em:
    * Troubleshoot de conexões
    * Testar regras de acesso de rede
    * Monitorar tráfego
    * Detectar e investigar incidentes de segurança

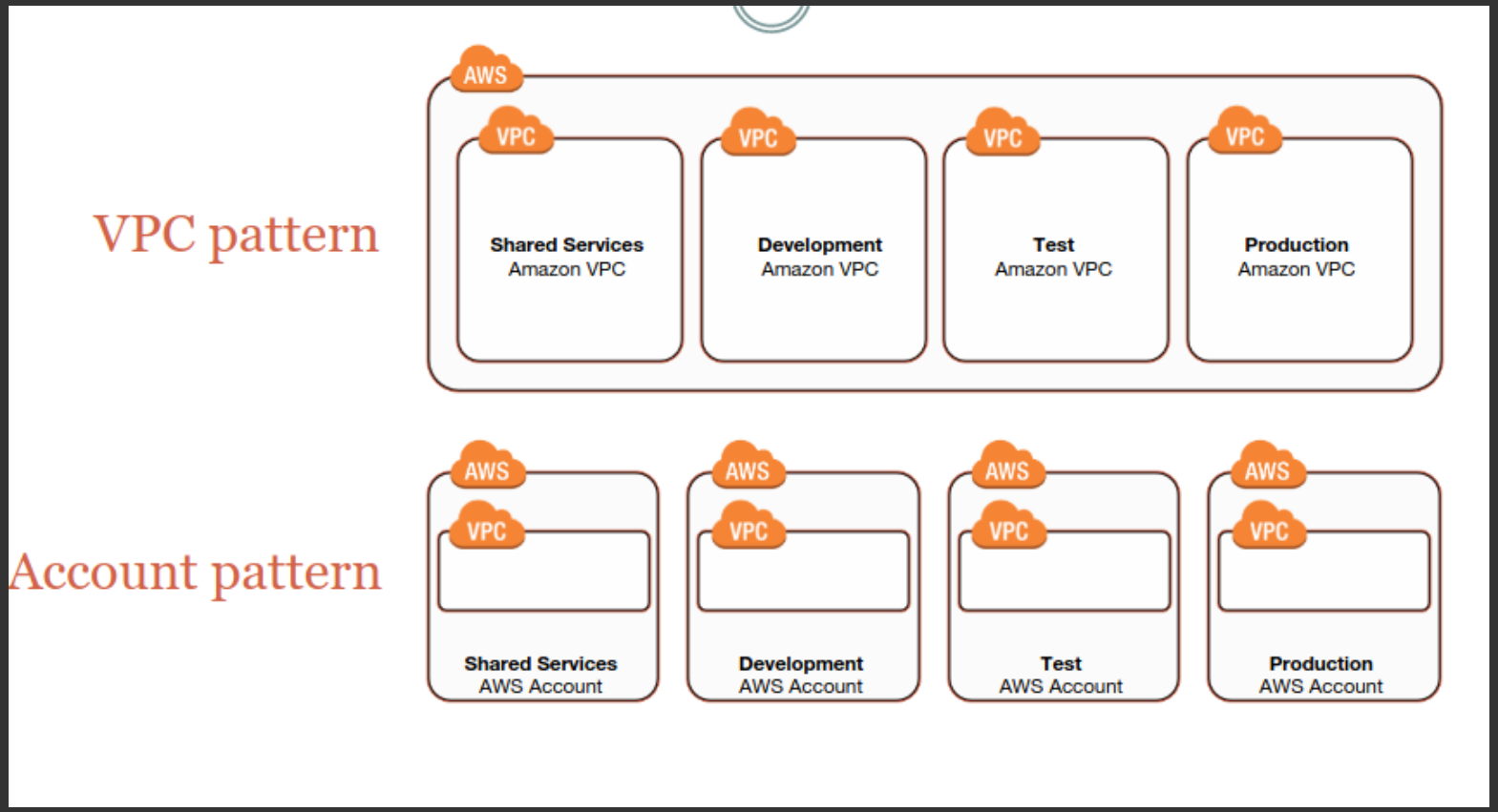


Figure: VPC pattern and Account pattern[5]

>>> AWS Infrastructure Patterns II

* Como escolher um padrão?
    * Complexidade da empresa e isolamento dos workloads
    * Uma equipe de TI?
        * Multi-VPC
    * Muitas equipes de TI?
        * Multi-account
    * Alto isolamento de workload
        * Multi-account

>>> Network Optimization I

* Tamanho da VPC
  * Evitar alocar /16 endereços IP padrão para todas as subnets
  * Alguns recursos precisam de IPs livres
    * Ex: Load balancer precisa de 8 IPs livres)
  * IPAM (VPC IP Address Manager)[3] para gerenciar os IPs nas redes
    * OBS: IPAM pode ser usado no CloudWatch (Verificar se os endereços IPs estão acabando ou overlay de VPC)

>>> Network Optimization II

* Quantas subnets por VPC?
  * Pelo menos 1 subnet por VPC
  * Aplicação em várias AZs = pelo menos uma subnet por AZ
    * OBS: Quando uma subnet é colocada em uma AZ não é possível mudar
* Compartilhar VPC ou criar uma VPC nova para o workload?
  * Times em diferentes contas da AWS, não precisam necessariamente usar diferentes VPCs
    * VPC Sharing[4] permite compartilhar VPCs com outras contas AWS
  * VPC Sharing Best Pratices

---

[3] Network Address Management and Auditing at Scale with Amazon VPC IP Address Manager

[4] VPC Sharing

* Grande parte dos serviços da AWS não usam VPC
  * Para esses serviços a VPC não consegue trazer nenum isolamento fora AAAAAAAAAAAAAAA
  * O tráfico de rede entre regiões da AWS atravessam a rede global da AWS por padrão
  * Amazon S3 e DynamoDB oferecem VPC endpoints para conectar sem atravessar a internet pública

>>> Directing Traffic To Your VPC

* To enable access to or from the Internet for instances in a VPC subnet, you must:
* Attach an Internet gateway to your VPC
* Ensure that your subnet's route table points to the Internet gateway
* Ensure that instances in your subnet have public IP addresses or Elastic IP addresses
* Ensure that your NACLs and security groups allow the relevant traffic to flow to and from your instance

* Oferece instâncias
* Podemos escolher: Processador, SO, Armazenamento, Redes, etc...

Figure: Tipo das instâncias

* São de uso geral: Web/App servers, Gaming servers, Dev/Test Environments, etc...

>>> Instâncias de propósito geral

* **Instâncias M5**: Equilíbrio entre memória, poder computacional e velocidade de rede
  * Proporção de memória para vCPU é de 4:1
* **Instâncias T3**: Tem uma linha base de performace da CPU e tem a possibilidade de passar a linha base (acumulando crédito ou pagando)
  * Usado para workloads que não usam a CPU constantemente.
* **Instâncias A1**: Workloads que precisam escalar em múltiplos cores, rodar instruções ARM, etc...

>>> Instâncias Memory-intensive workloads

* Uso em: Banco de dados de alta performace, Análise de Big Data, Cache de memória, etc...
* **R5 Instances**: Workloads que processam data sets grandes em memória
  * Proporção de memória para vCPU é de 8:1
* **X1/X1e Instances**: Proporção de memória para vCPU é de 16:1 e 32:1
* **High memory instances**: Certificado para rodar SAP HANA
  * Possui 6 até 24 TB de memória

>>> Instâncias Compute-intensive workloads

* Uso em: High-perf computing (HPC), Multiplayer Gaming, Video encoding, etc...
* **C5 Instances:** Alta performace por um preço baixo
  * Proporção de memória para vCPU é 2:1
* **z1d Instances:** ALta performace em uma única thread. Processador mais rápido em nuvem de 4.0 GHz
  * Proporção de memória para vCPU é de 8:1

>>> Instâncias Storage-intensive workloads

* Uso em:
  * Alta operações de I/O. Ex: High-perf databases, No SQL databases, etc...
  * Muito armazenamento. Ex: Big Data, Kafka, Log processing...
* **Instâncias I3/I3en**: Otimizadas para operações de I/O com pouca latência
* **Instâncias D2**: Custo baixo por armazenamento e suporta alta taxas de transferências
* **Instâncias H1**: Aplicações de custo baixo que usam altas transferências de dados e acesso sequencial para grandes Data Sets.
  * Mais vCPUS e memória por TB que o D2

>>> Workloads de computação acelerada

* Uso em: Machine learning, HPC, Gráficos, etc...)

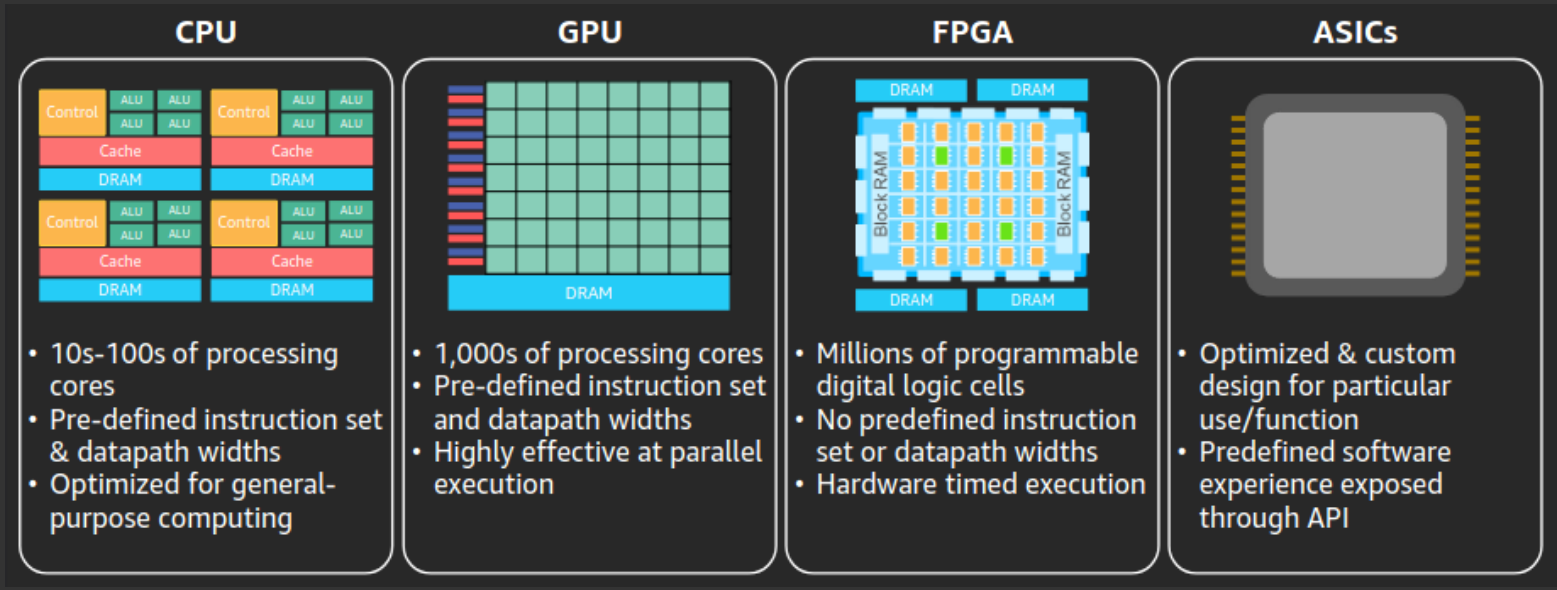


Figure: CPU vs GPU vs FPGA vs ASCIs[3]

>>> Instâncias de computação acelerada

* **Instâncias P2/P3:** GPU (deep learning training, HPC, etc...)
* **Instâncias G3/G4:** GPU (renderização 3D, codificação de vídeo, etc...)
* **Instâncias F1:** FPGAs programáveis (processamento de imagem, computação financeira, etc...)
* **Instâncias Inf1:** Alta performace e custo baixo para machine learning
  * Integração com ML frameworks (TensorFlow, PyTorch, etc...)

>>> Instâncias Bare Metal

* Feito para workloads que não são virtualizados ou precisam de tipos específicos de hypervisors ou tem licenças que restrigem o uso de virtualização

>>> Amazon Machine Images (AMIs)

* Amazon Maintained
  * Imagens de Windows e Linux
  * Recebem Updates pela amazon em cada região
  * Amazon Linux 2 (5 anos de suporte)
* Marketplace Maintained
  * São gerenciados e mantidos pelos parceiros da AWS
* Your Machine Images
  * AMIs que foram criadas de instâncias EC2
  * Podem ser privadas, compartilhadas com outras contas ou publicadas na comunidade

>>> Amazon EBS

* Blocos de armazenamento como serviço
* Escolher o armazenamento e computar baseado no seu workload
* Pode colocar ou retirar de uma instância
* Volumes magnéticos ou baseados em SSD
* Suportam snapshots de um bloco modificado
* Dados criptografados por padrão em volumes EBS
* Fast Snapshot Restore (FSR)
* Rede mais otimizada para EBS em instâncias C5/C5d, M5/M5d, R5/R5d

>>> Security Group

* Firwall virtual para controlar a entrada e saída de tráfego em instâncias EC2

>>> Autoscaling groups

* Permite escalar horizontalmente as instâncias EC2
* O amazon EC2 auto scaling garante que o seu grupo vai ter a quantidade desejada de instâncias
* É possível configurar o Auto Scaling group para aumentar a capacidade em um dia e horário específico
* O dynamic scaling define como escalar os recursos dependendo da mudança da demanda
* É possível usar o CloudWatch para aumentar o número de servers usando algum parâmetro definido
* Health checks

>>> Classic Load Balancer

* Suporte para EC2-Classic
* Suporte para TCP e SSL
* Suporte para sticky sessions usando cookies gerados pela aplicação
* Redireciona as requisições para instâncias registradas
* Tem health-checks

>>> Application Load Balancer

* Funciona na camada de aplicação do modelo OSI (HTTP, HTTPS, gRPC)
* É possível adicionar regras para poder redirectionar as requisições de forma mais precisa
* Health checks podem ser feitos em grupos de instâncias
* Benefícios em relação ao clb:
  * Path conditions (URL)
  * Host conditions (Host field in http header)
  * HTTP header conditions
  * Multiplas aplicações em um EC2 (Bom para mircoserviços)

>>> Netowrk Load Balancer

* Funciona na camada de rede (TCP, UDP, TLS)
* Foward TCP traffic
* High performance
* Support static / Elastic IP
* Latency 100 ms (400 ms ALB)

>>> Gateway Load Balancer

* Gateway + Load Balancer
  * Next-hop in route table
  * NO packet rewrite
* Layer 3 load balancer
  * Provide horizontal scale to appliances
  * Fault tolerance for appliances
  * Insert services transparently
  * Share across differente VPCs and accounts
  * Provide appliance as a service
* get package of IP and use to part of appliance
* Thirt party appliance
* Simplify applicance deployment
* Conectar VPCs diferentes:
  * Fazer appliance ou segurança

Table: Fonte[5]

a	a

[5] Comparação dos load balancers

>>> Amazon Relational Database Service (RDS) I

* Banco de dados relacional gerenciado pela AWS, facilita:
  * Hardware provisioning
  * Setup
  * Patching
  * Backups
* Fácil de administrar
  * Fácil de instânciar e pronto para uso em poucos cliques
  * Possível escolher Poder computacional e memória
    * Standard (General purpose)
    * Memory Optimized (Memory intensive application)
    * Burstable performance (Burst CPU usage)
  * Automatic patching
  * Recomendações de melhores práticas (Engines, Armazenmento, Tipos de instâncias, Redes, etc...)
* Alta Escalabilidade

>>> Amazon Relational Database Service (RDS) II

    * Verticalmente (vCPU, Armazenamento e Mem)
    * Réplicas de leitura
* Disponível e durável
    * Replica os dados em múltiplas instâncias
    * Snapshots: Feito por usuários e armazenados em buckets
    * Backups automáticos: São feitos automaticamentes em buckets
    * Point-in-time recovery: Possível recuperar seus dados em pouco tempo
    * Host replacement: Substituição automática em caso de falha de hardware
    * Multi-AZ Deployments: É possível criar réplicas em vários AZs
* Performace
    * Magnetic Storage (Compatibilidade)

>>> Amazon Relational Database Service (RDS) III

- Armazenamento de propósito geral (SSD): Burst de 3000 IOs (Broad range of database workloads)
- IOPs provisionado (SSD): Operações de IO constantes (OLTP Database workloads)

* Segurança
  - Controle de acesso de rede (VPCs + IPsec VPN)
  - Encryption at rest e Encryption at transit

>>> Amazon Relational Database Service (RDS) IV

* Disponibiliza
    * Amazon Aurora
    * PostgreSQL
    * MySQL
    * MariaDB
    * Oracle
    * SQL Server
* AWS Database Migration Service
    * Migra bancos de dados existentes para o Amazon RDS

>>> Amazon S3 I

* Amazon Simple Store Service
* Armazena objetos:
  * Dados
  * Metadados (Pares chave-valor)
* Cada objeto tem um identificador único (Global)
* Pode habilitar versionamento
* Objetos não são deletados (são marcados como deletado)

http://**bucket**.S3.<region>.amazonaws.com/**Xorg.0.log**

Figure: Endereço S3

>>> Amazon S3 II

* Storage classes
    * S3 Standard
    * S3 Intelligent - Tiering
    * S3 Standard - IA
    * S3 One zone - IA
    * S3 Glacier Instant Retrieval
    * S3 Glacier Flexible Retrieval
    * S3 Glacier Deep Archive

>>> S3 Features

* **S3 Storage class analysis:** Descobrir padrões de acesso
* **S3 Lifecycle policy:** Quando os dados devem ser transferidos para outro tipo de storage class
* **S3 Cross-Region Replication (CRR):** Replicar buckets entre diferentes regiões
* **S3 Same Regnio Replication:** Replicar buckets na mesma região
* **S3 Object Lock:** Não permite que o bucket seja apagado
* **S3 Inventory:** Lista de objetos e status da criptografia
* **S3 Batch operations:** Copiar objetos de buckets, colocar tags, modificar acesso, etc...
* **S3 Select:** Aumenta a performace da query e reduz os custos

>>> S3 Access Management

* Buckets são privados por padrão
    * O dono do recurso da acesso para os outros
* Resource-based policies
    * Bucket policies: escrito em Json e permite/bloqueia usuários em determinado bucket
    * Access Control Lists (ACLs) usa xml e define quem tem permissão para usar o bucket
* User policies (IAM)

>>> Cloud Watch

* Permite:
  * Monitorar recursos e aplicações em tempo real
  * Coletar métricas dos recursos e aplicações
  * Criar alarmes que verificam métricas. Podem:
    * Mandar notificações
    * Fazer mudanças automáticas nos recursos

>>> Cloud Watch

* CloudWacth Logs insights
* CloudWacth Logs
* CloudWacth Alarms
* Auto scaling
* AWS Integration
* CloudWacth Events

>>> IAM

* Identity and Acess Management
* Boas práticas:
    * Habilitar MFA
    * Não usar o **root**
    * Grupos para atribuir permissões
    * Política de senhas do IAM
* OBS: É universal, funciona em todas as regiões

>>> IAM - Users

* **Programmatic access**
* Ativa acesso por key ID e secret access key:
    * AWS API
    * CLI
    * SDK
* **AWS Management Console access**
* Interface web

>>> IAM - Tags

* Identificar serviços
* Relatório de faturamento baseado em *Tags*
* **OBS:** Até 50 tags por serviço

>>> IAM – Políticas I

* Criar grupos com permissões para os usuários
* Não usar permissões diretamente nos usuários
* Permissões mais específicas são mais fortes
    * Permissão de usuário prevalece contra permissão de grupo)
* Políticas de senha

>>> IAM - Políticas II

* Políticas de acesso
    * As políticas definidas por um arquivo Json
    * Existem olíticas prontas
    * Criar políticas customizadas
    * Políticas podem ser atribuídas em usuários/grupos

>>> Funções/Roles

* Dar permissões para:
    * Recursos
        * Ex: Dar permissão para uma instância acessar um bucket
    * Outras contas AWS
    * Federações do SAML 2.0
    * Identidade web (Login Google, amazon, etc...)

>>> Relatórios de acesso

* Relatórios de credenciais
  * Lista de todas as credenciais geradas
* Access Analyzer: Gera um relatório de políticas pra a gente ver o que precisa ser modificado. é possível arquivar, resolver, etc...

>>> IAM Identify Center

* Expande as capacidades da AWS IAM
* Provê um lugar centralizado para administrar os usuários e seus acessos para contas AWS e aplicações cloud
  * Workforce identities
  * Application assignments for SAML applications
  * Identity Center enabled applications
  * Multi-account permissions
  * AWS access portal

>>> IAM - Users

* a

>>> Usando Spot (com Coiled)

* Spots podem salvar muito dinheiro [7], mas você precisa:
  * Encontrar máquinas Spots o suficiente na sua região
  * Lidar com seu desaparecimento

>>> Escolher AZ e Região [1]

* Verificar se na sua zona tem Spots o suficiente
  * Usar o Spot placement score para ver qual região ou az tem mais chance de suprir as necessidades do usuário
* Deixar todos os Spots em uma zona só (Evitar custos de transferência de dados por zona)

>>> Fallback

* Se os Spots não forem o suficiente é possível colocar instâncias *On-demand para suprir as necessidades*
  * *Spot: Vai alocar apenas Spots*
  * *Spot-with-fallback: Vai alocar Spots e vai alocar on-demand caso seja necessário*
  * *On-demand: Se quiser que tudo seja estável*

>>> Burstable Instances

* Vai existir cobrança se usar mais CPU do que foi definido
* Caso a instância use menos CPU do que foi definido a instância vai juntar créditos de CPU para usar

>>> Outros

* EBS Disks
* Network traffic for EC2 instances
* Nat Gateways

>>> AWS Network Optimization Tips

* Read this [4]

>>> Referencias I

[1] Coiled. *AWS Cost Explorer Tips and Tricks*. URL: https://blog.coiled.io/blog/aws-cost-explorer-tips/?ck_subscriber_id=2013077611 (visited on 01/20/2023).

[2] Simson Garfinkel. *The Cloud Imperative*. URL: https://www.technologyreview.com/2011/10/03/190237/the-cloud-imperative/ (visited on 01/20/2023).

[3] Chetan Kapoor. *AWS re:Innvet*. URL: https://github.com/ahmedtariq01/Cloud-DevOps-Learning-Resources/blob/main/AWS%20Learning/AWS%20EC2%20Foundations.pdf (visited on 01/20/2023).

>>> Referencias II

[4] Bentzen M., Looney B., and Kumar R. *AWS Network Optimization Tips*. URL: https://aws.amazon.com/blogs/networking-and-content-delivery/aws-network-optimization-tips/?ck_subscriber_id=2013077611 (visited on 01/20/2023).

[5] Frisby R. and Mcgibney J. *AWS Web Services - Virtual Private Cloud (VPC)*. URL: https://github.com/ahmedtariq01/Cloud-DevOps-Learning-Resources/blob/main/AWS%20Learning/AWS%20VPC%20for%20Beginners.pdf (visited on 01/20/2023).

[6] Ankit R Sanghvi. *History of Cloud Computing*. URL: https://www.cohesive.so/blog/the-history-of-cloud-computing (visited on 01/20/2023).

[7] Nat Tabris. *Save Money with Spot*. URL: https://www.coiled.io/blog/save-money-with-spot (visited on 01/20/2023).