Name: Henrique Tsuyoshi Yara (OPUS-software)
Date: March 20, 2023

Figure: AWS logo

>>> Índice I

>>> **Índice II**

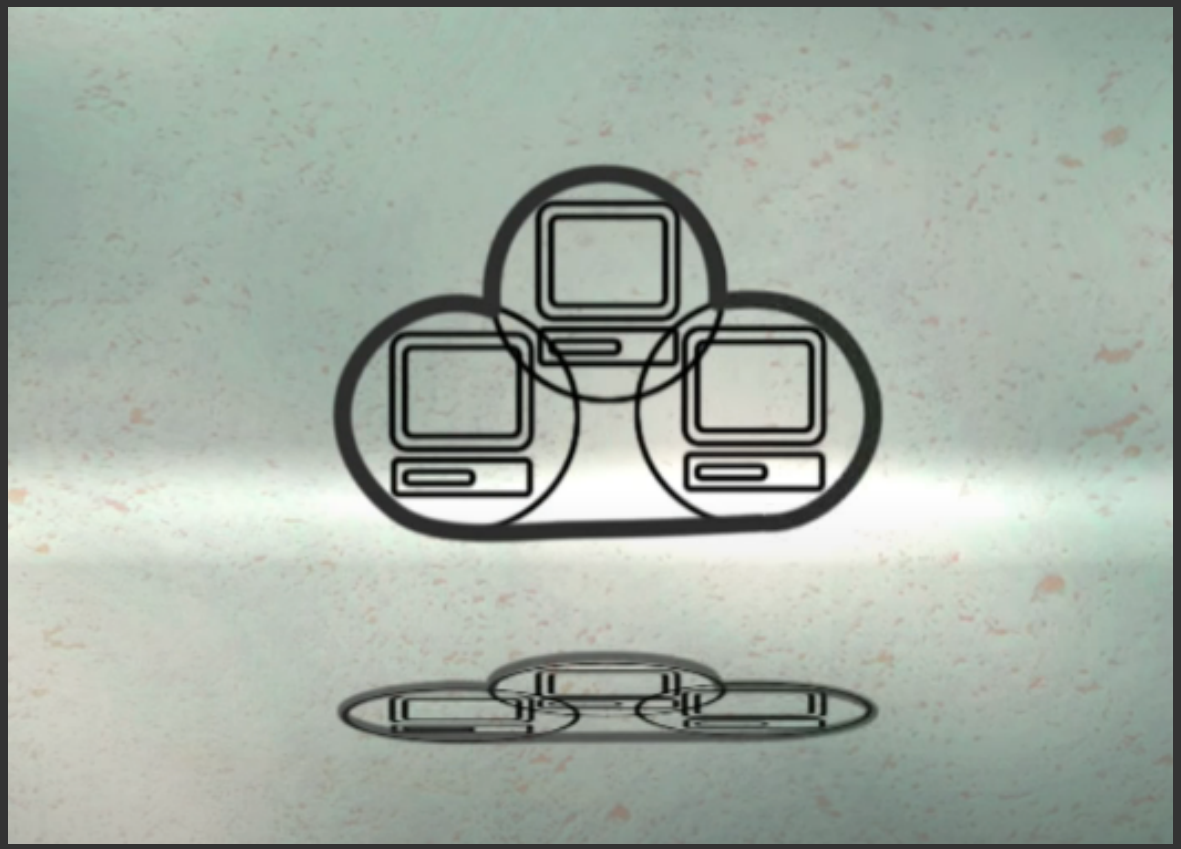
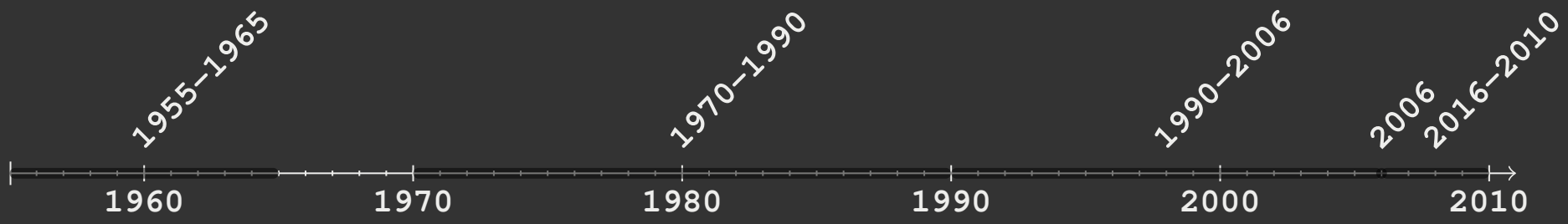


Figure: A cluster of servers drawn in a system diagram [1]

[1] Image source link

>>> Linha do tempo

História da computação em nuvem[4]



* (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

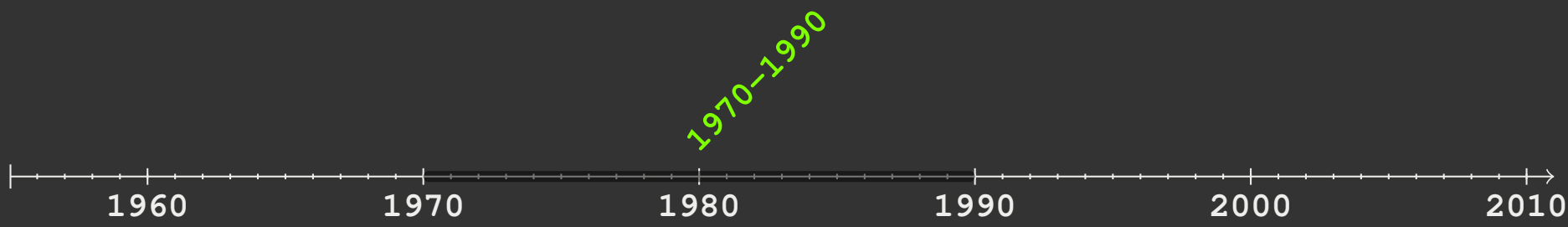
* 1942 - John Vincent Atanasoff construiu o computador ABC
* 1960 - Muito caro para aderir os computadores
  * Sala inteira para o servidor (Manter temperatura ideal, espaço, etc...)
  * Computador
  * Funcionários especializados
  * Problemas para adaptar o software

>>> 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 [1]
  * 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

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)

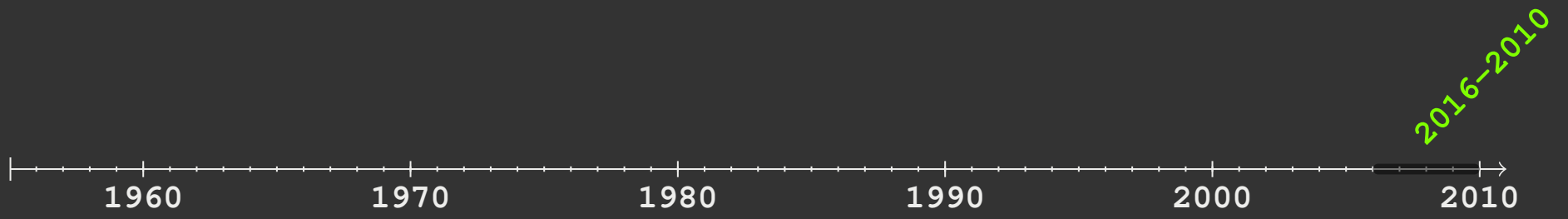
>>> Linha do tempo

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

>>> Virtualização

* Criação de infraestruturas virtuais a partir de uma estrutura física

* Conceito que reúne vários *softwares* e utiliza de virtualização
* Possui algumas características específicas:
  * Autoserviço sob demanda
  * Amplo acesso a rede
  * Pool de recursos
  * Rápida elaticidade
  * Serviços mensuráveis
* **Colocation:** alugar determinado espaço de um Data Center (refrigeração, instalação elétrica e segurança). No entanto, utiliza seu próprio equipamento. [2]

[2] Colocation

>>> 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

>>> Primeiros dias da computação em nuvem

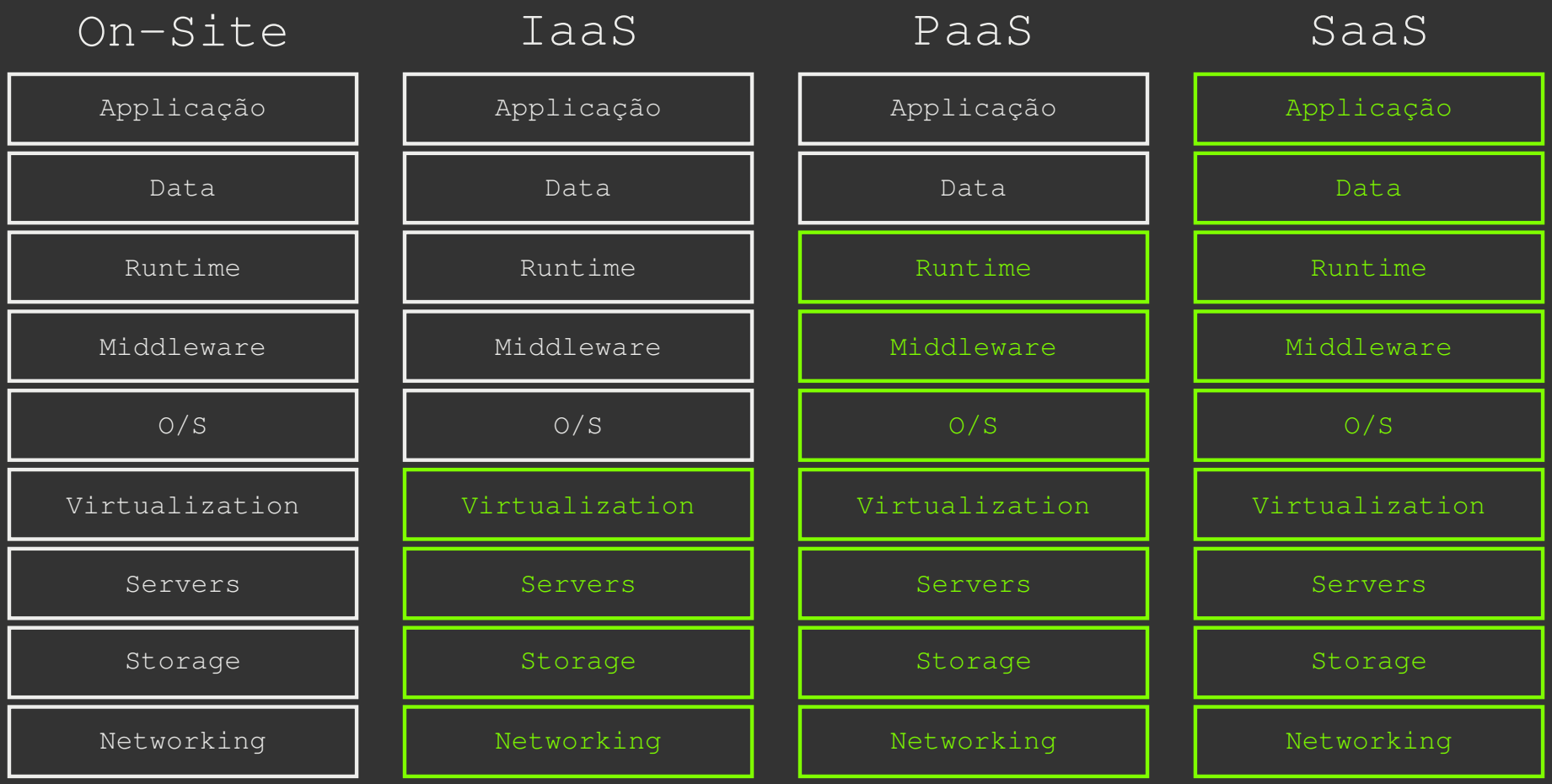


Figure: Imagem retirado do site da redhat[3]

[3] RedHat IaaS, PaaS e SaaS

>>> Tipos de nuvem

* Infraestrutura como Serviço (Iaas)
* Software como Serviço (SaaS)
* Plataforma como Serviço (PaaS)

>>> IaaS - Infrastructure as a Service

* O recurso fornecido ao consumidor é provisionar:
  * Processamento
  * Armazenamento
  * Comunicação de rede
  * Outros recursos de computação funcamentais nos quais o consumidor pode instalar e executar softwares em geral, incluindo sistemas operaionais e aplicativos
  * Possivelmente um controle limitado de alguns componentes de rede (*firewall*)

>>> PaaS - Plataform as a Service

* O recurso fornecido ao consumidor é instalar na infraestrutura na nuvem aplicativos criados ou adiquiridos pelo consumidor
* O consumidor tem controle sobre as aplicações instaladas e possívelmente configuraçoes de hospedagem de aplicações
* O consumidor não gerencia nem controla a infraestrutura na nuvem subjacente (Rede, servidores, sistemas operacionais, armazenamento ou mesmo recursos individuais da aplicação, com a possível exeção de configurações limitadas por usuário)

>>> SaaS - Software as a Service

* O recurso fornecido ao consumidor é o uso de aplicções do fornecedor executando em uma infraestrutura na nuvem
* As aplicações podem ser acessadas por vários dispositivos clientes através de interfaces leves ou ricas
* O consumidor não gerencia nem controla a infraestrutura na nuvem subjacente (Rede, servidores, sistemas operacionais, armazenamento ou mesmo recursos individuais da aplicação, com a possível exeção de configurações limitadas por usuário)

>>> Exemplos

Tipo	Serviço	Exemplos
IaaS	Rede virtualizada Armazenamento de dados Servidores Virtuais	AWS VPC, Azura Virtual Network AWS S3, Google cloud storage AWS EC2, Azure Virtual Machines
PaaS	Infraestrutura para desenvolvimento, implantação e execução de aplicativos Plataforma testes e gerenciamento de aplicações	Heroku, Google App Engine AWS Elastic Beanstalk
SaaS	Armazenamento Dados Editor de textos e planilha SIstema de Gestão de banco de dados	DropBox, Google Drive Gsuite e Office 365 AWS RDS, Google Cloud SQL

Table: Exemplos de serviços

>>> Modelos de implementação de nuvem

* Nuvem pública
* Nuvem privada
* Nuvem híbrida
* Nuvem comunitária

>>> Nuvem pública

* Provisionada para uso aberto ao público em geral
* Sua propriedade, gerenciamento e operação podem ser de:
  * Uma empresa
  * Uma instituição acadêmica
  * Uma organização do governo
  * Ou uma combinação mista
* Fica nas instalações do fornecedor
* **OBS**: Criar uma estrutura na Amazon e configurar usando VPN ou conexão direta continua sendo uma nuvem pública

>>> Nuvem privada

* Provisionada para uso exclusivo por uma únic organização composta de diversos consumidores
* A sua propriedade, gerenciamento de operação podem ser de:
  * A própria organização
  * Terceiros
  * Combinação mista
* Pode estar dentro ou fora das instalações da organização
* Nuvem privada não é organização e não precisa estar instalada localmente

>>> Nuvem comunitária

* Provisionada para uso exclusivo por uma determinada comunidade de consumidores de organizações que têm interesses em comum (missão, requisitos de segurança, políticas, observância de regulamentações)
* A sua propriedade, gerenciamento e operação podem ser de:
  * Uma organização
  * Mais de uma organizações da comunidade
  * Terceiros
  * Combinação mista
* Pode estar dentro ou fora das instalações das organizações participantes

>>> Nuvem híbrida

* Composição de duas ou mais infraestruturas na nuvem (**privadas, comunitárias** ou **públicas**) que permanecem entidades distintas
* São interligadas por tecnoogia padronizada ou proprietária que permite a comunicação de dados e portabilidade de aplicações (Transferencia de processamento para a nuvem para balanceamento de carga entre nuvens)

>>> 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
  * 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

>>> Pontos de presença

* Compõem as regiões
* Para verificar o status das zonas de disponibilidade/regiões ou recursos
  * Status AWS
  * 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

>>> 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:
    * Privada: recursos que não devem ser acessíveis pela internet
    * Pública: 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 conexões 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

Table: VPC NAT Gateway Vs NAT Instances on amazon EC2[4]

	VPC NAT Gateway	Instância NAT
Disponibilidade	Alta disponibilidade por padrão	Usa scripts para gerenciar falhas
Banda larga	Bursts de 10 Gbps	Se baseia na banda larga da instância
Manutenção	Feito pela AWS	Feito por você
Segurança	NACLs	Security groups e NACLs
Redirecionamento de portas	Não suportado	Suportado

[4] Comparison

>>> 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:
    * Usar 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

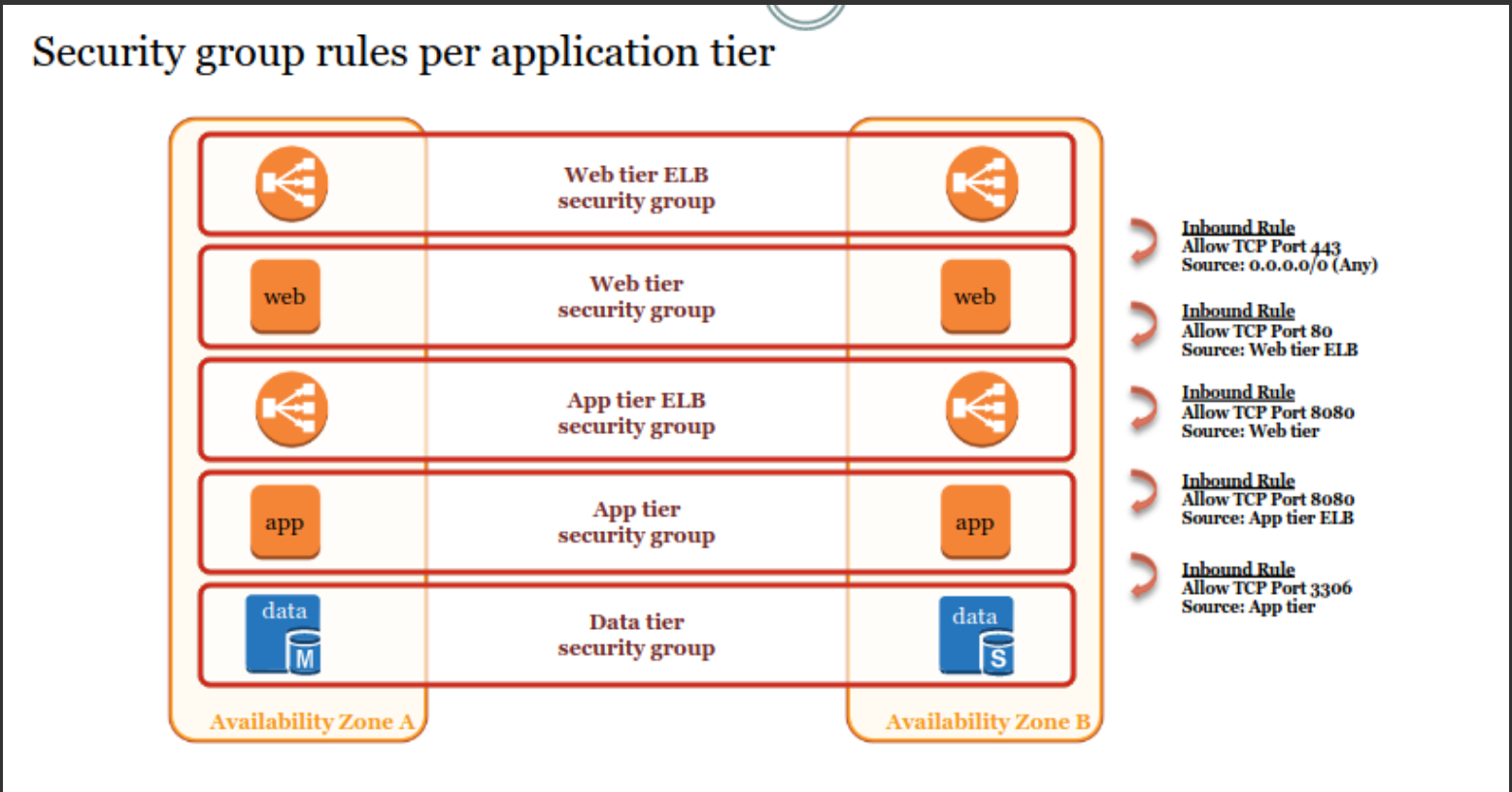


Figure: Security Group chaining diagram

>>> 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. Não mantém registro da conexão
* Aplica a todas as instâncias nas sub-redes

>>> NetworkACL II

* Cada regra vai ter uma prioridade
* É bom deixar um espaço entre cada regra para possíveis regras futuras (Ex: deixar 10 espaços entre cada regra)
* **OBS:** Liberar portas efêmeras (1024-65535). 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)
* Uso em:
  * Troubleshoot de conexões
  * Testar regras de acesso de rede
  * Monitorar tráfego
  * Detectar e investigar incidentes de segurança

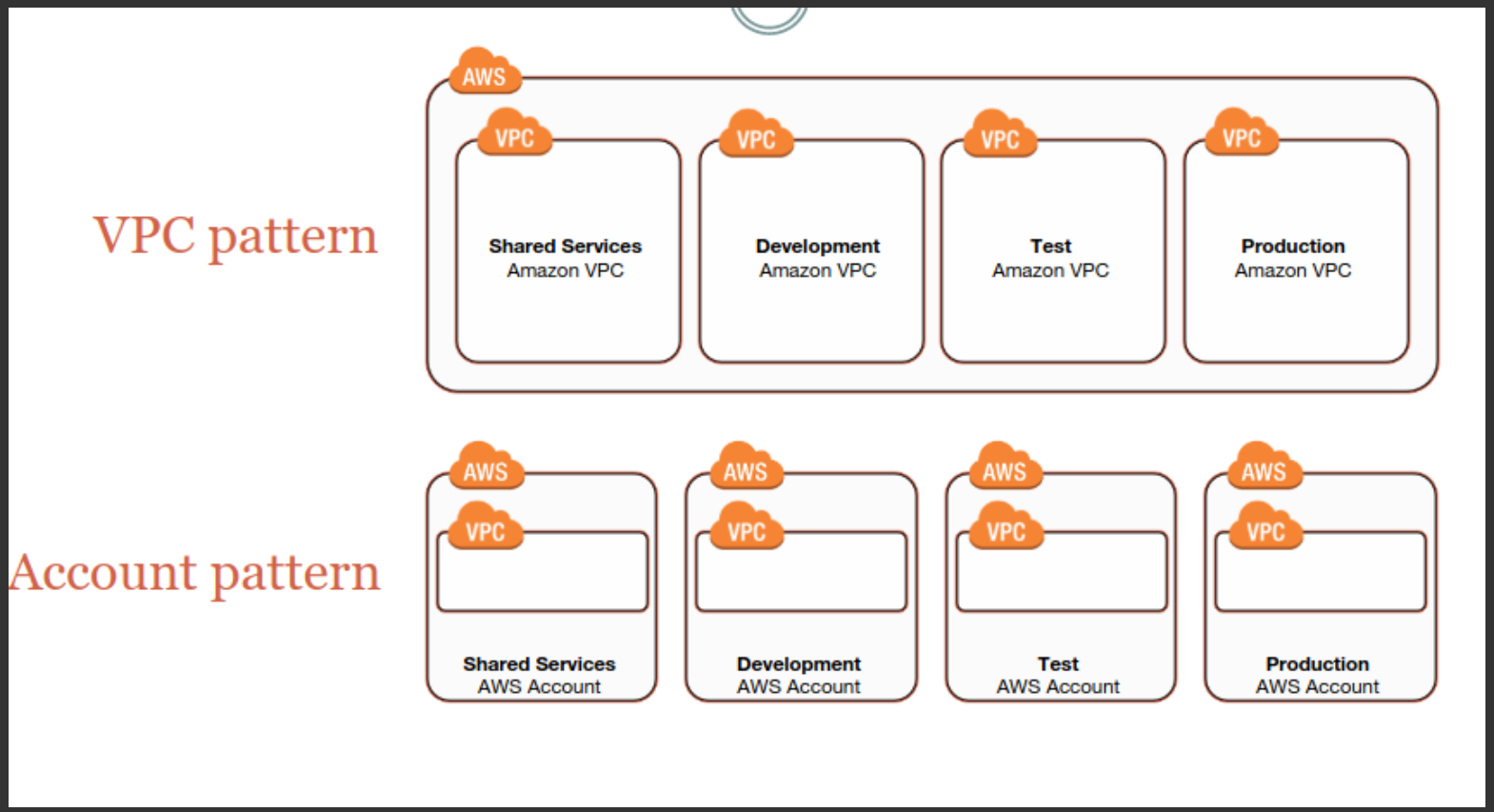


Figure: VPC pattern and Account pattern[3]

>>> 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)[5] 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[6] permite compartilhar VPCs com outras contas AWS
  * VPC Sharing Best Pratices

---

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

[6] VPC Sharing

* 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

* 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

* 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

* Machine learning, HPC, Gráficos, etc...

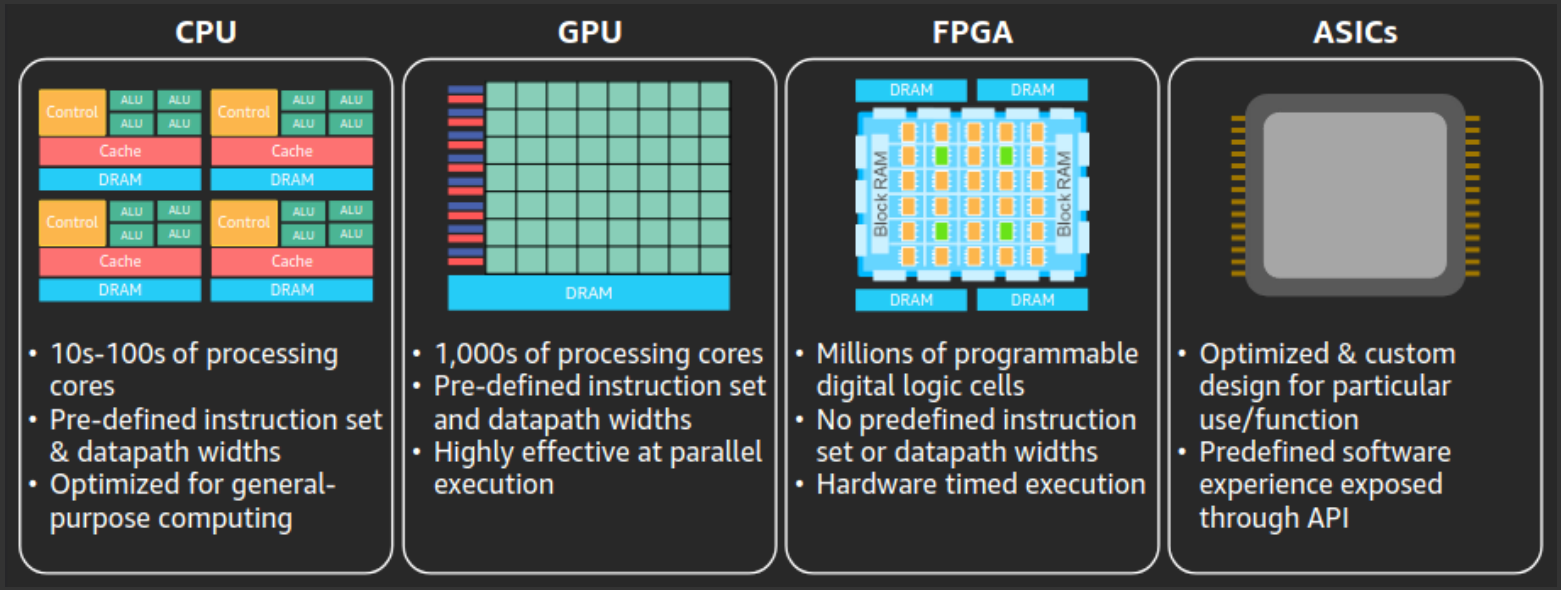


Figure: CPU vs GPU vs FPGA vs ASCIs[2]

>>> 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

* Escalar horizontalmente instâncias EC2
* Garante que o seu grupo vai ter a quantidade desejada de instâncias
* Pode aumentar a capacidade em um dia e horário específico
* Dynamic scaling define como escalar os recursos dependendo da mudança da demanda
* Pode usar o CloudWatch para aumentar o número de servers usando algum parâmetro definido
* Health checks

>>> Elastic IP

* Elastic IP é um endereço IPv4 público
* Elastic IP é alocado para a conta da AWS e será seu até que você o libere
* Mascarar a falha de uma instância ou software (remapeia rapidamente o endereço para outra instância na conta)
* É possível especificar o endereço IP elástico em um registro DNS para o seu domínio
* Cobrança:
  * Por hora quando um Elastic IP não está associado a uma instância em execução/encerrada ou a uma interface de rede não anexada
  * Será cobrado por qualquer endereço IP elástico adicional associado a uma instância

>>> 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
  * NO packet rewrite
* Layer 3 load balancer
  * Escalabilidade horizontal para appliances
  * Tolerancia de falhas para appliances
  * Transparencia de inserção de serviços
  * Possível comparilhar em diferentes VPCs e contas
  * Appliance as Service
* Appliance de terceiros
* Simplifica o deployment de appliance
* Conectar VPCs diferentes:
  * Fazer appliance ou segurança (Firewall)

Table: Fonte[7]

Recurso	Application Load Balancer	Network Load Balancer	Gateway Load Balancer	Classic Load Balancer
Tipo de balanceador de carga	Camada 7	Camada 4	Gateway da camada 3 + Balanceamento de carga da camada 4	Camadas 4/7
Tipo de destino	IP, instância, Lambda	IP, instância, balanceador de carga da aplicação	IP, instância	
Encerra fluxo / comportamento de proxy	Sim	Sim	Não	Sim
Listeners do protocolo	HTTP, HTTPS, gRPC	TCP, UDP, TLS	IP	TCP, SSL/TLS, HTTP, HTTPS
Acessível via	VIP	VIP	Entrada da tabela de rotas	

[7] Comparação dos load balancers

>>> Amazon Relational Database Service (RDS) I

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

>>> Amazon Relational Database Service (RDS) II

* Alta Escalabilidade
  * 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

>>> Amazon Relational Database Service (RDS) III

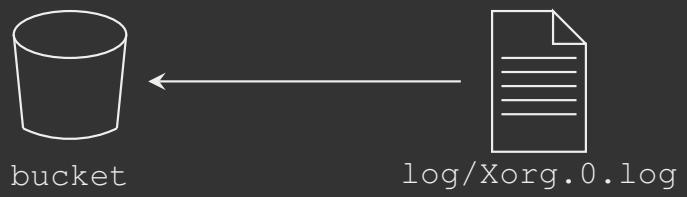
* Performace
  * Magnetic Storage (Compatibilidade)
  * Armazenamento de propósito geral (SSD): Burst de 3000 IOs (Broad range of database workloads)
  * IOPs provisionado (SSD): Operações de IO constantes (OLTP Database workloads)
* Segurança
  * Controle de acesso de rede (VPCs + IPsec VPN)
  * Encryption 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 (Simple Store Service) I

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



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

Figure: Endereço S3

>>> Amazon S3 (Simple Store Service) 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

* Monitorar recursos e aplicações em tempo real
* Ter todas as aplicações em um único dashboard
* Coletar métricas dos recursos e aplicações
* Criar alarmes que verificam métricas
    * Mandar notificações
        * Bills
        * IPAM
    * Fazer mudanças automáticas nos recursos
        * Auto scaling

>>> 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
- **OBS:** é possível forcar MFA por políticas customizadas ou IAM IC

>>> 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
  * Possível forcar MFA
  * Consegue gerenciar várias contas AWS
  * Pode importar ou usar hierarquias de outros provedores (Active directory)
  * Controlar acesso na cloud e aplicações na nuvem

>>> Referencias I

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

[2] 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).

[3] 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).

>>> Referencias II

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