



Stakeholders e Modelos de Serviço

Computação em Nuvem

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

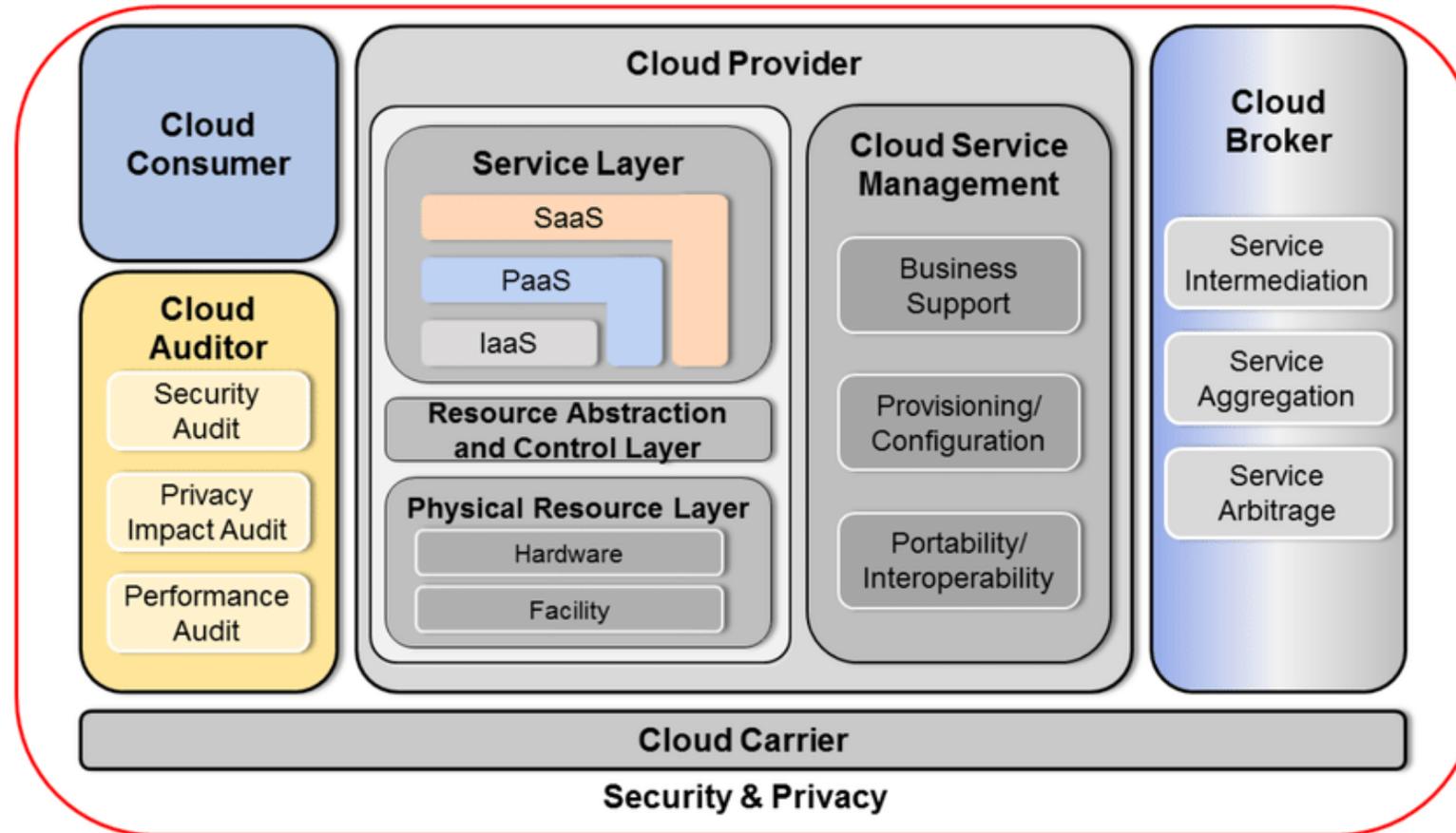
Objetivos de Aprendizado

- Entender o papel dos diferentes stakeholders na computação em nuvem.
- Compreender as diferenças fundamentais entre IaaS, PaaS, SaaS, XaaS e FaaS, identificando os níveis de controle, responsabilidade e cenários ideais para cada modelo.
- Avaliar critérios práticos (custo-efetividade, escalabilidade, compliance) que orientam a escolha do modelo de serviço mais adequado ao negócio ou aplicação específica.

Revisitando Conceitos

- **Computação como serviço.**
 - Entregues de maneira semelhante a utilidades tradicionais, como água, eletricidade, gás e telefonia.
 - Usuários acessam serviços com base nas suas necessidades, sem se preocupar com o local onde esses serviços são hospedados ou como são entregues.
- **Consumidor** delegam o **gerenciamento** da infraestrutura de IT.
 - Evita investimento pesado 'up-front'.
 - Pay-as-you-go
- **Provedores** se beneficiam da economia de **escala**.
 - Precisam otimizar hardware, espaço física, energia, refrigeração.
 - Infraestutura fortemente baseada em **virtualização** de recursos computacionais.

Arquitetura de Referência NIST



National Institute of Standards and Technology (NIST) Cloud Computing Reference Architecture

Papéis: Provedor

O provedor de nuvem é a entidade responsável por **fornecer** recursos de TI que são acessados remotamente por consumidores de nuvem.

- Recursos: armazenamento, processamento, e serviços de rede.
- Garante que os serviços sejam entregues conforme os [Acordos de Nível de Serviço \(SLAs\)](#) estabelecidos.

Papéis: Consumidor

O consumidor de nuvem é a organização, indivíduo ou usuário final que **utiliza** os serviços fornecidos pelo provedor de nuvem.

- O acesso aos recursos é feito por meio de interfaces programáticas, como APIs, ou por interfaces de usuário, como painéis de controle e aplicativos web.
- Deve **gerenciar a utilização** desses recursos de forma eficaz para obter o máximo benefício da nuvem.

Papéis: Proprietário

O proprietário de serviço de nuvem é a entidade que possui **legalmente** um serviço na nuvem, e este papel pode ser desempenhado tanto pelo provedor de nuvem quanto pelo consumidor de nuvem, dependendo de quem desenvolveu e implantou o serviço.

- Exemplo: uma empresa que desenvolve um software e o disponibiliza na nuvem pode ser tanto o consumidor (usando a infraestrutura de outro provedor) quanto o proprietário do serviço (pois detém os direitos sobre o software).
- A propriedade do serviço implica responsabilidades legais e de manutenção.

Papéis: Administrador

O administrador de recursos de nuvem é a pessoa ou organização encarregada de **gerenciar** os recursos de TI na nuvem, o que inclui a administração de serviços, servidores, redes, e quaisquer outros recursos associados.

- O administrador pode estar diretamente ligado ao provedor de nuvem, mas também pode ser um terceiro contratado pelo proprietário do serviço de nuvem para garantir a operação contínua e eficiente dos recursos na nuvem.
- Função: manter a disponibilidade, a segurança e o desempenho dos serviços oferecidos na nuvem.

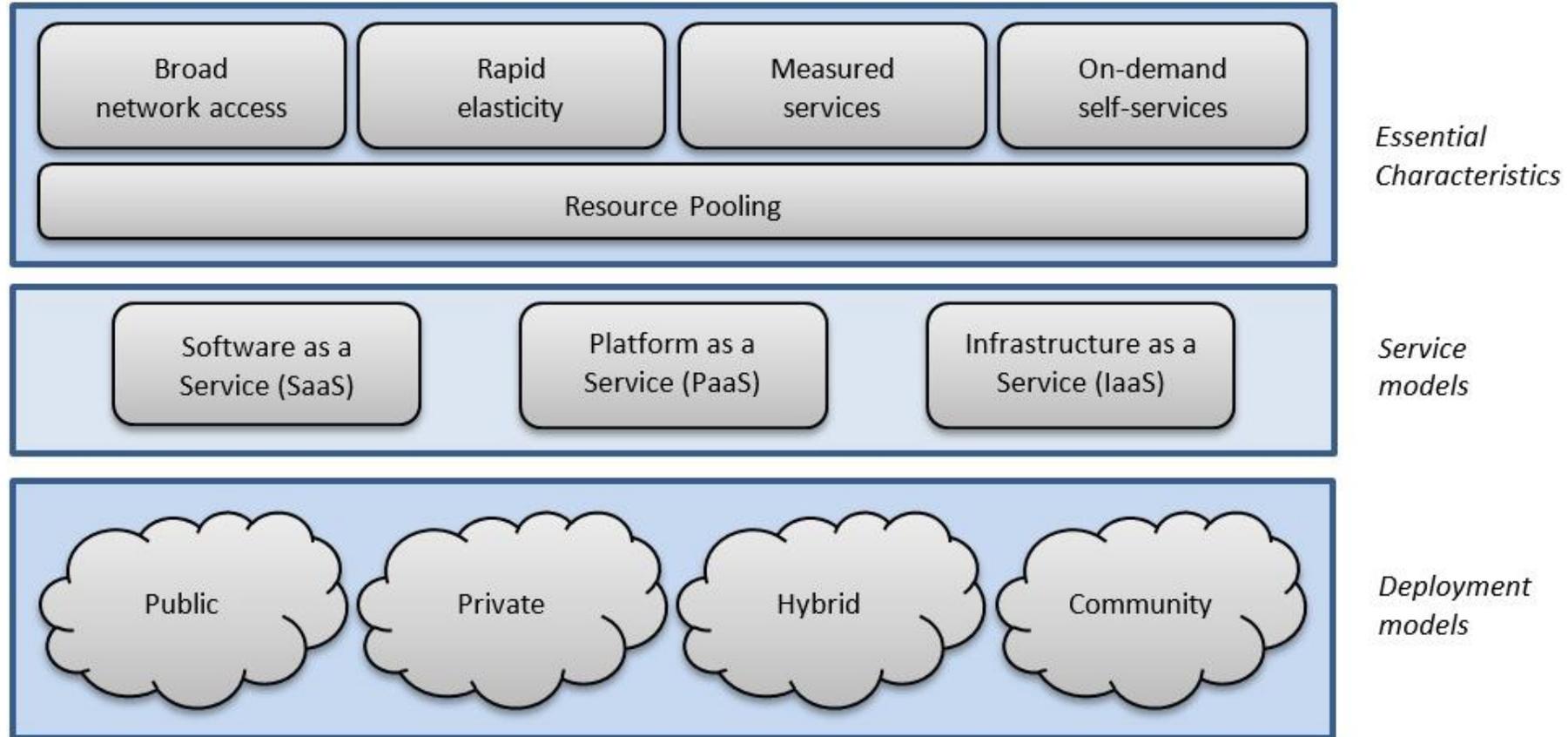
Outros papéis

- **Auditor de Nuvem**: realiza **avaliações** independentes da segurança, privacidade e desempenho dos serviços na nuvem, ajudando a **construir confiança** entre consumidores e provedores.
- **Broker de Nuvem**: atua como um **intermediário**, negociando e gerenciando a utilização de serviços entre consumidores e provedores, facilitando a interoperabilidade e a escolha do melhor serviço disponível.
- **Carrier de Nuvem**: fornece a **conectividade de rede** necessária para que consumidores possam acessar os serviços oferecidos na nuvem, desempenhando um papel crucial na infraestrutura.

Modelo de Despesas em Cloud

- **CapEx (Capital Expenditure) - Despesa de Capital:** Investimento inicial em ativos físicos (hardware, data centers) → Pagamento único ou amortizado ao longo do tempo.
 - **Vantagens:** Maior controle sobre o hardware, potencialmente menor custo a longo prazo (se bem gerenciado).
 - **Desvantagens:** Alto investimento inicial, risco de obsolescência tecnológica, necessidade de equipe especializada para manutenção.
- **OpEx (Operational Expenditure) - Despesa Operacional:** Pagamento recorrente por serviços e recursos utilizados ("pay-as-you-go") → Flexibilidade para aumentar ou diminuir a capacidade conforme a demanda.
 - **Modelo Cloud:** Aluguel de servidores virtuais, armazenamento, software, etc.
 - **Vantagens:** Baixo investimento inicial, escalabilidade sob demanda, foco no core business.
 - **Desvantagens:** Custo potencialmente maior a longo prazo se não otimizado, dependência do provedor de nuvem.

Cloud: Visão Geral



Fonte da imagem: [OER - IT Systems](#)

Modelos de Serviço (ou de Entrega)

- Descrevem maneiras de disponibilizar recursos de TI sob demanda na nuvem.
- Os pilares são: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS), Software como Serviço (SaaS), Função as a Service (FaaS) Anything as a Service (XaaS).
- Cada um confere ao usuário níveis variados de controle, flexibilidade e responsabilidade compartilhada entre cliente e provedor.
 - IaaS concede autonomia sobre os elementos básicos da **infraestrutura**.
 - PaaS entrega um ambiente pré-configurado para **desenvolvimento de aplicações**
 - SaaS disponibiliza **software completos** que podem ser usados diretamente pela nuvem.
 - FaaS permite que desenvolvedores **executem funções** de código em resposta a eventos, pagando apenas pelo tempo de execução.
 - XaaS engloba **qualquer tipo de serviço computacional** oferecido via nuvem, além dos tradicionais IaaS, PaaS e SaaS.

Infrastructure as a Service (IaaS)

- **Definição:** Provedor oferece recursos computacionais virtuais sob demanda.
- **Componentes principais**
 - Máquinas Virtuais (VMs) / Containers
 - Rede virtual (VPC, sub-redes)
 - Armazenamento de bloco/objeto
 - Balanceadores de carga
- **Quem controla?**
 - Cliente: Sistema operacional, middleware, aplicações, dados.
 - Provedor: Hardware físico, hypervisor, infraestrutura de rede.
- **Casos de uso**
 - Migração de workloads para a nuvem.
 - Ambientes de teste/produção com alta flexibilidade.
 - Big Data & HPC (ex.: clusters Spark).

Platform as a Service (PaaS)

- **Definição:** Plataforma de desenvolvimento e execução automatizada; provedor gerencia stack completo.
- **Componentes principais**
 - Runtime (Java, .NET, Node.js, Python)
 - Serviços de banco de dados (SQL/NoSQL)
 - Mensageria, filas, caches integrados
 - Ferramentas CI/CD, monitoramento
- **Quem controla?**
 - Cliente: Código da aplicação, lógica de negócio, dados.
 - Provedor: Infraestrutura subjacente, runtime, serviços de suporte.
- **Vantagens**
 - Rápida iteração (deploy em segundos).
 - Escalabilidade automática.
- **Caso de uso:** Micro-serviços gerenciados via containers (ex.: Azure App Service, Google App Engine).

Software as a Service (SaaS)

- **Definição:** Aplicação completa entregue via web; o cliente consome sem se preocupar com infra-estrutura ou plataforma.
- **Componentes principais**
 - Front-end web/mobile.
 - Backend completo, banco de dados e serviços integrados, além de APIs para integração.
- **Quem controla?**
 - Cliente: Dados do usuário, configurações, fluxos de trabalho.
 - Provedor: Aplicação inteira, infraestrutura, segurança.
- **Vantagens**
 - Zero manutenção operacional.
 - Atualizações automáticas e centralizadas.
- **Casos de uso:** CRM, ERP, colaboração (ex.: Salesforce, Office 365, Trello).
- **Exemplos populares:** Microsoft 365, Google Workspace, Dropbox Business.

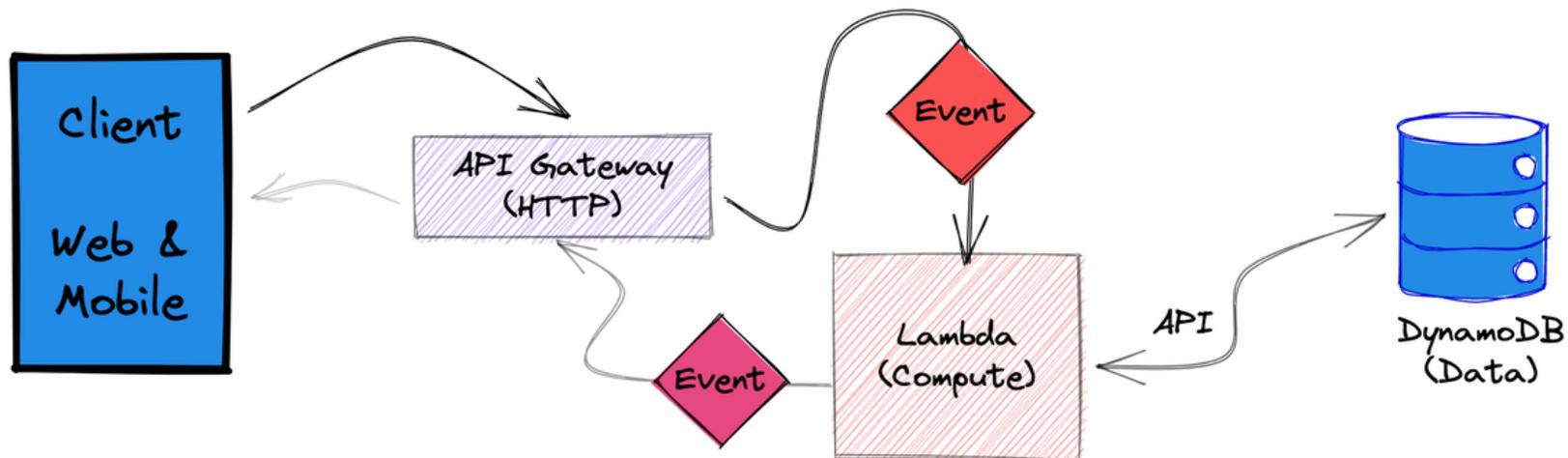
Function as a Service (FaaS)

- Modelo de computação **serverless** em que código é executado como funções independentes, acionadas por eventos.
- **Componentes-chave**
 - **Trigger**: Fonte de evento (HTTP, fila, timer, mudança no storage).
 - **Runtime**: Ambiente controlado pelo provedor (Node.js, Python, Java, Go).
 - **Billing**: Cobrança baseada em invocações e tempo de execução (milissegundos).
- **Quem controla?**
 - **Cliente**: Código da função, lógica de negócio, configuração de triggers.
 - **Provedor**: Provisionamento automático de hardware, sistemas operacionais e servidores web.
- **Casos típicos de uso**
 - Processamento de eventos em lote (logs, métricas).
 - Back-end para aplicações móveis/web (APIs RESTful leves).

Function as a Service (FaaS)

- **FaaS vs. Serverless**

- **Serverless** é um conceito amplo que abrange qualquer serviço (computação, armazenamento, BD, mensagens, API Gateway) onde gerenciamento e faturamento de servidores são invisíveis.
- **FaaS** é um subconjunto específico dentro do serverless, focado em computação baseada em eventos: código ou contêineres executam apenas quando disparados por eventos/solicitações.



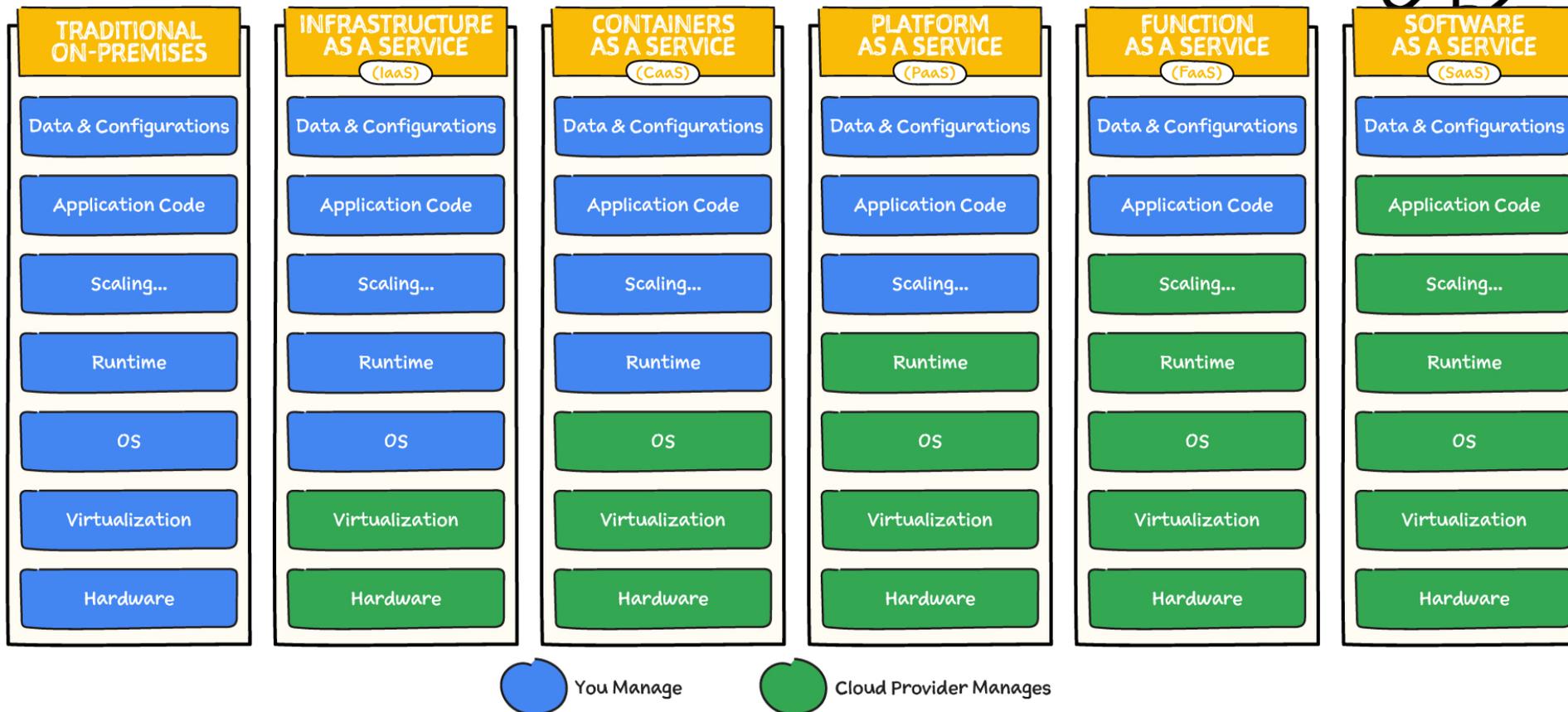
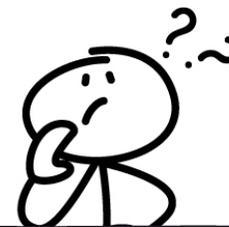
FaaS executa códigos em resposta a eventos, sem necessidade de gerenciar infraestrutura. Fonte da Imagem: [AWS](#)

Comparativo Resumido

Modelo	Controle do Cliente	Escalabilidade	Custo típico	Melhor para
IaaS	Alto (SO + apps)	Manual/Auto	CapEx → OpEx	Workloads customizados, legacy
PaaS	Médio (apps)	Auto	OpEx	Aplicações web/micro-serviços
SaaS	Baixo (dados)	Auto	OpEx	Soluções de negócio prontas
FaaS	Baixo (funções)	Instantânea	Por execução	Processamento event-driven



Wait... what is Cloud again?



Resumo de modelos de serviço na cloud. Fonte: [Google Cloud](#).

Resumo & Próximos Passos

- **Principais modelos revisados**
 - **IaaS** – infraestrutura virtual (VMs, redes, storage).
 - **PaaS** – plataforma gerenciada para desenvolvimento e deploy.
 - **SaaS** – aplicações completas acessíveis via browser/SDK.
 - **FaaS** – funções event-driven com escalabilidade automática.
 - **XaaS** – qualquer serviço especializado entregue na nuvem (DaaS, SECaaS, AaaS, etc.).
- Escolha o **modelo certo** com base em:
 - Nível de controle e customização necessário.
 - Padrões de carga e escalabilidade desejados.
 - Custo alinhado ao orçamento.
 - Requisitos regulatórios, segurança e governança.
- **Próxima aula**: Modelos de Deploy – Public, Private, Hybrid & Multi-Cloud.

Perguntas e Discussão

1. Quando um negócio deve optar por SaaS em vez de desenvolver sua própria aplicação (PaaS ou IaaS)? Quais são os trade-offs entre agilidade, controle e custo?
2. Como você abordaria a governança quando utiliza múltiplos provedores XaaS (ex.: DaaS + SECaaS) na mesma organização? Que métricas e políticas seriam cruciais para evitar silos de dados ou vulnerabilidades?
3. Em cenários de alta disponibilidade, qual modelo de serviço oferece a melhor combinação entre resiliência e simplicidade operacional? Considere FaaS versus PaaS em termos de failover automático e tolerância a falhas.
4. Como a escolha do modelo de serviço influencia a estratégia de backup e recuperação de desastres na nuvem? Diferencie abordagens entre IaaS (snapshots de VMs) e SaaS (replicação de dados nativa).