Cloud Computing Introduction

CS516 – Cloud Computing

Computer Science Department

Maharishi International University

Maharishi International University - Fairfield, Iowa

Main concepts

- Cloud computing
  - Why do we need it?
  - What is it?
  - What are the benefits?
- Cloud services models
  - IaaS (Infrastructure as a Server)
  - PaaS (Platform as a Service)
  - FaaS (Function as a Service)

Evolution to the cloud

Issues with running an app in local

- Not reachable from the internet
- A resource is used up by other apps
- When you shot down the laptop, the app stops

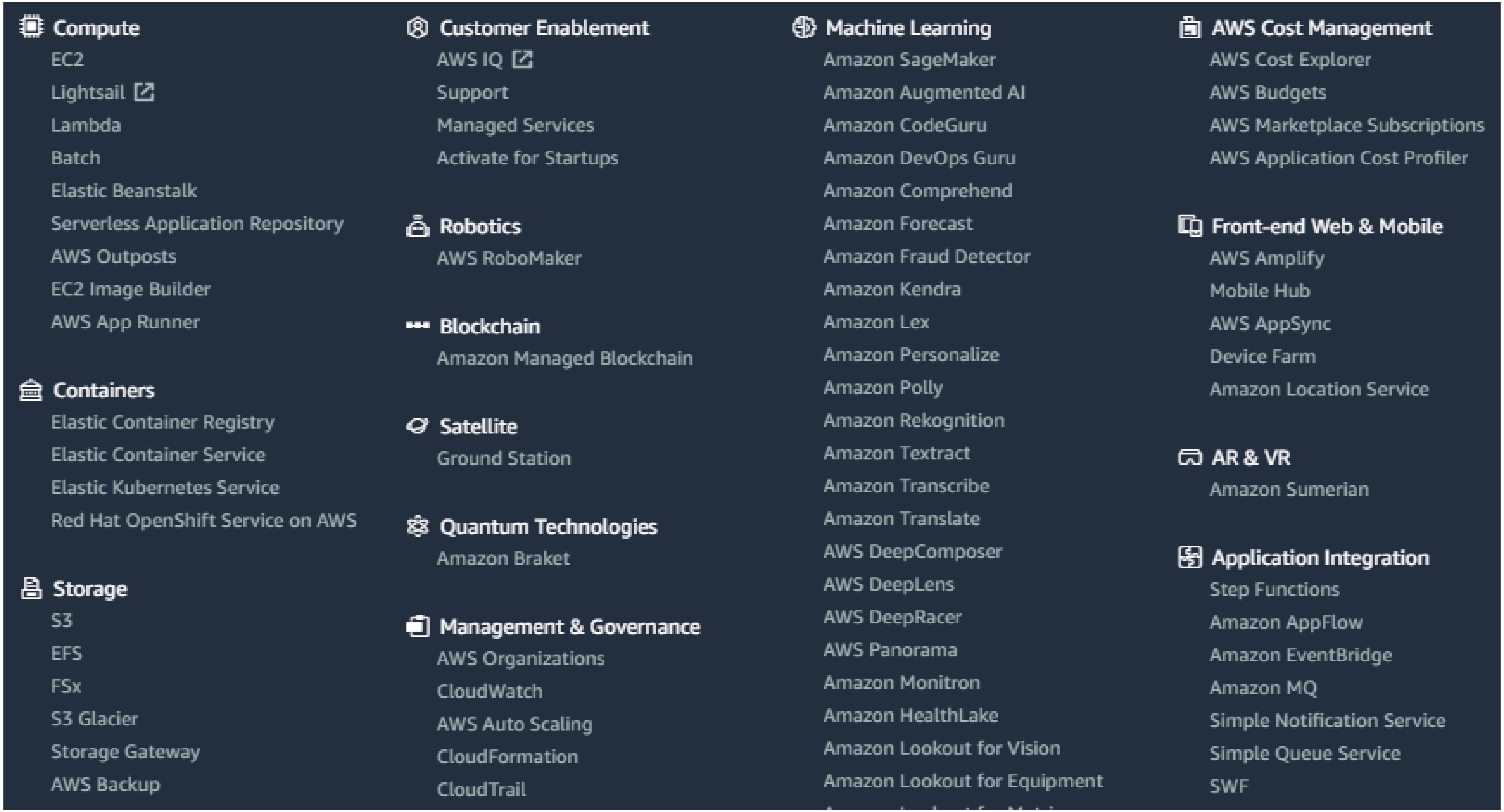
Running an app on the server

- Benefits: Reachable from the internet, the server is dedicated to the app only, running all the time 24x7.
- Issues: Not scalable. Not highly available. All work on you. App code grows which adds complexity. One day, it is impossible to add new features. All you do is fix bugs.

Running an app on the cloud server – Solves all issues above.

What is the Cloud Computing?

- Cloud Computing is the on-demand delivery of all types of resources as a web service such as computing, database, big data, AI, VR, IoT, blockchain, quantum technologies, robotics, satellite, you name it.
- Cloud Computing is a tool to build **evolvable** applications. The nature of life is to evolve (develop gradually from a simple to a more complex form).
- Think of the cloud as **software** that helps you build your app including all required components such as infrastructure, networking, storage, IP, CPU, memory, database, and so on. You use other software services to build your software.



Compute
- EC2
- Lightsail
- Lambda
- Batch
- Elastic Beanstalk
- Serverless Application Repository
- AWS Outposts
- EC2 Image Builder
- AWS App Runner

Containers
- Elastic Container Registry
- Elastic Container Service
- Elastic Kubernetes Service
- Red Hat OpenShift Service on AWS

Storage
- S3
- EFS
- FSx
- S3 Glacier
- Storage Gateway
- AWS Backup

Customer Enablement
- AWS IQ
- Support
- Managed Services
- Activate for Startups

Robotics
- AWS RoboMaker

Blockchain
- Amazon Managed Blockchain

Satellite
- Ground Station

Quantum Technologies
- Amazon Braket

Management & Governance
- AWS Organizations
- CloudWatch
- AWS Auto Scaling
- CloudFormation
- CloudTrail

Machine Learning
- Amazon SageMaker
- Amazon Augmented AI
- Amazon CodeGuru
- Amazon DevOps Guru
- Amazon Comprehend
- Amazon Forecast
- Amazon Fraud Detector
- Amazon Kendra
- Amazon Lex
- Amazon Personalize
- Amazon Polly
- Amazon Rekognition
- Amazon Textract
- Amazon Transcribe
- Amazon Translate
- AWS DeepComposer
- AWS DeepLens
- AWS DeepRacer
- AWS Panorama
- Amazon Monitron
- Amazon HealthLake
- Amazon Lookout for Vision
- Amazon Lookout for Equipment

AWS Cost Management
- AWS Cost Explorer
- AWS Budgets
- AWS Marketplace Subscriptions
- AWS Application Cost Profiler

Front-end Web & Mobile
- AWS Amplify
- Mobile Hub
- AWS AppSync
- Device Farm
- Amazon Location Service

AR & VR
- Amazon Sumerian

Application Integration
- Step Functions
- Amazon AppFlow
- Amazon EventBridge
- Amazon MQ
- Simple Notification Service
- Simple Queue Service
- SWF

Cloud is like a Lego. You got all the pieces to build something great. AWS has sophisticated services to build an app. Now they are focused on much bigger problems such as quantum, simulation, health care, supply chain, aerospace, and so on.

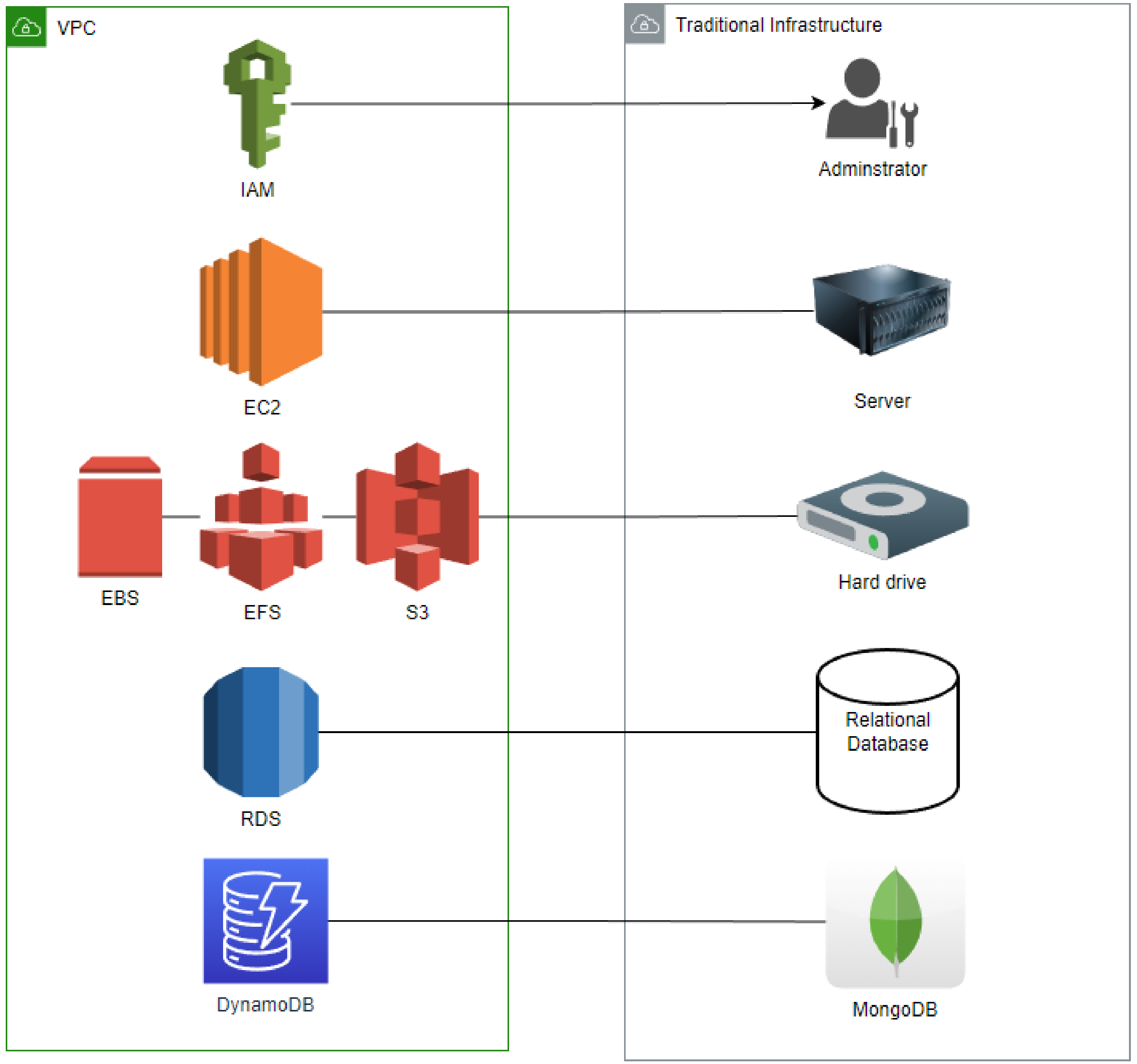
What are services?

A web service is an API. In the traditional world, we write an API in front of a database that does the CRUD. When you use the cloud, **the database itself is already an API** (Database as a Service). So, you don't have to build your own API. That is the power of the cloud! You have to change your mindset. **Instead of building your own, just use services out there**.

Web services are just HTTP endpoints (RESTful or SOAP). We can call and use AWS services in 3 ways.

1. AWS console – Calling AWS services from the web app. It is just a front-end app calling Amazon's web services when you hit buttons.
2. CLI - Calling the same AWS services from a computer terminal. You need to install AWS CLI and provide tokens. Great for quick experiments.
3. SDK – Calling the same AWS services from your application. For example, storing data in the database.

Cloud is not very different from traditional infrastructure.



Why the cloud?

From developer perspective:

- Do less and achieve more. Most work is done by the cloud provider. Developers only focus on business logic. No code or low code. For example, using AWS Simple Notification Service. You can send emails to clients without writing any code.

From business perspective:

- The cloud helps businesses save money. Running some workloads in the cloud is much cheaper in most cases. With the cloud, one developer can do work in one month that used to be done by a team of developers in months. Applications are high availability. So, customers are happy and no money loss.
- It gives businesses agility. Startups need to build an app in a month.

Benefits of the cloud

- **Do less and achieve more** – The cloud provider deals with technical problems. It lets developers focus on the application.
- **Cost-effective** – The cloud is cost-effective in most cases, especially when using serverless services.
- **Secure** – There are many security services that you can stack on top of your applications that protect against attacks at all layers. All data in transit and at rest is encrypted.
- **Performant** – The globe is in your hand with the cloud. You can serve users all over the world without losing performance. There are many services in the cloud that improve the performance of the application.
- **Reliable** – Because the app runs and data is stored in multiple data centers, even in multiple regions. That improves the high availability and fault tolerance of the application and the durability of the data. The cloud also helps your app to scale.

Benefits of the cloud

- **Agility** – Agility is crucial in business that gives advantages. You can deploy your application in multiple regions globally in minutes. There are also tools like Amazon Amplify that helps developers to build full-stack web and mobile applications in minutes.
- **You don’t have to guess capacity** – In the traditional infrastructure, you have to guess the server size that meets the need. But that could be too much or too low. If the server is too big, it will cost more. If the server is too small, the application goes down or gets slower due to full utilization. Cloud resources are elastic.
- **Built-in metrics** – Metrics are created along with the resource in the cloud. Metrics are useful information about the resource for monitoring and troubleshooting purpose. For example, when you create EC2 virtual machines in the AWS cloud, CPU utilization metrics are also created in the CloudWatch.

Models of Cloud Services

Non-cloud	IaaS	FaaS	SaaS
🟦 Application	🟦 Application	🟦 Application	🟨 Application
🟦 Runtime	🟦 Runtime	🟨 Runtime	🟨 Runtime
🟦 OS	🟦 OS	🟨 OS	🟨 OS
🟦 Hardware	🟨 Hardware	🟨 Hardware	🟨 Hardware
🟦 Networking	🟨 Networking	🟨 Networking	🟨 Networking
🟦 Building	🟨 Building	🟨 Building	🟨 Building

Infrastructure as a Service (IaaS)

IaaS means you rent a server from the cloud provider. You choose the operating system, memory, hard drive, and CPU size. You will receive a key pair to log in to your server after the instance is created (There is another way to login without key pair from AWS console which is recommended). Once the server is provisioned, you can do whatever you want in the server such as hosting a website you developed.

You still have a lot of work to do on your side. I recommended you utilize other cloud service models if you want to do less and achieve more.

The IaaS service in the AWS cloud is an EC2.

Platform as a Service (PaaS)

You don't know the cloud. And you've just got your code and want to run it in the cloud. Then use PaaS services. It will **provision** the underlying resources for you to run your code.

PaaS is where you can directly drag and drop your (backend) app and hit deploy. Then your app is available publicly.

You still have to look after the underlying assets, but you don't have to worry about provisioning of them.

The PaaS service in the AWS cloud is an Elastic Beanstalk. Under the hood, it utilizes the IaaS services such as EC2, Load Balancers, RDS.

Other PaaS providers are Heroku, Vercel, Digital Ocean and so on.

Function as a Service (FaaS)

FaaS allows customers to develop, run, and manage application functionalities **without the complexity of building and maintaining the infrastructure and servers**.

Building an application following this model is one way of achieving a **serverless** architecture and is typically used when building modern event-driven and microservices applications.

Serverless computing is a cloud computing execution model in which the cloud provider allocates machine resources on demand, taking care of the servers on behalf of their customers.

The FaaS service in the AWS cloud is a Lambda.

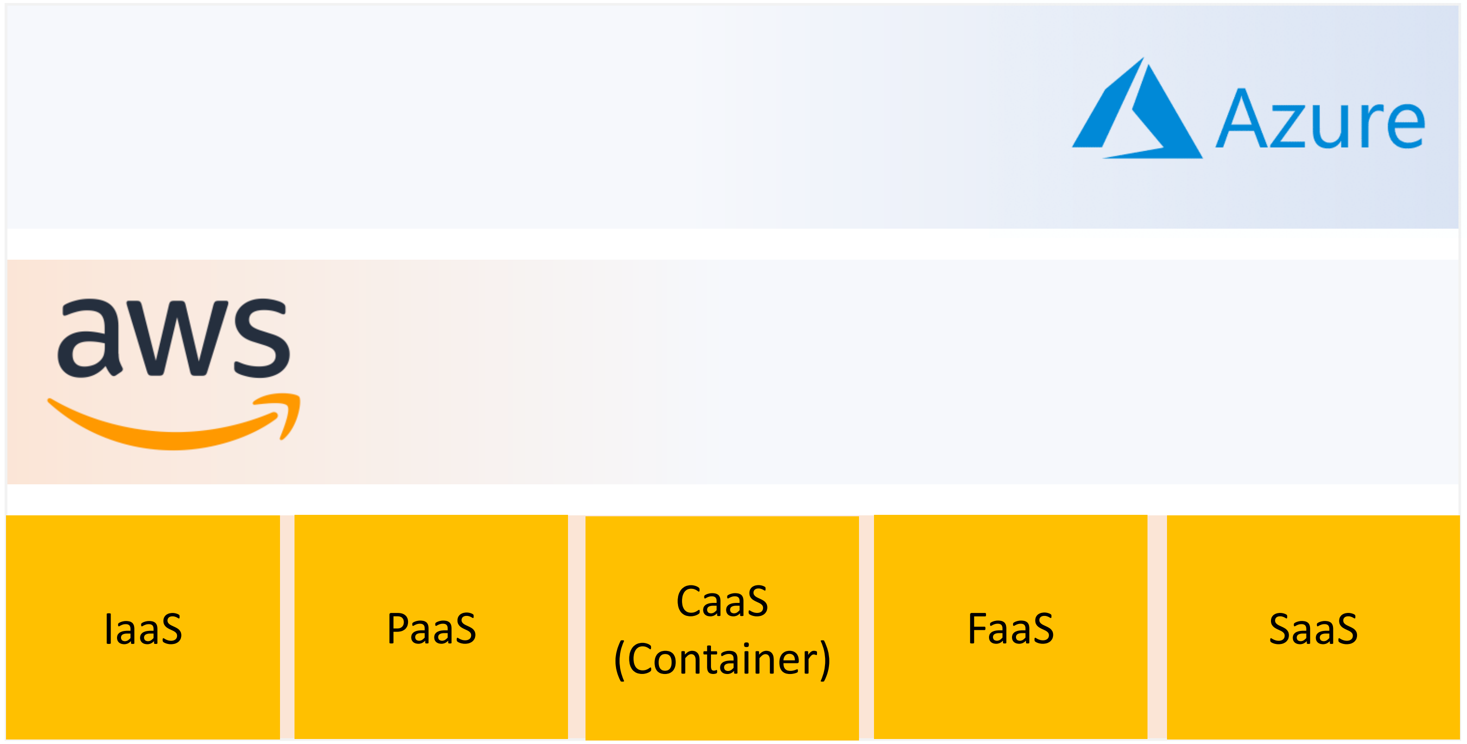
Software as a Service (SaaS)

SaaS is the app you develop and costumers use it via the internet. For example, Gmail, all you worry about is using the actual software, about creating messages, filtering spam filters. You're not worried about the underlying servers, how they are load balanced, high availability, DNS resolving etc.

SaaS is your app in the cloud running in the IaaS, PaaS, and/or FaaS models.

As a Service!

- **Container** as a service
- **Data** as a service
- **Desktop** as a service
- **Function** as a service
- **Infrastructure** as a service
- **Integration** as a service
- **Network** as a service
- **Platform** as a service
- **Security** as a service
- **Software** as a service



Container as a Service

Containerized deployments took over deployments on virtual machines. Because it is much lighter and faster to deploy apps. A Container as a Service model allows you to run containerized applications in the cloud.

The biggest benefit of this model is that containerized applications are platform-agnostic.

Docker is the most popular containerization technology. In AWS, there 2 ways to run containerized applications, on servers (ECS on EC2 or EKS) or serverless (ECS Fargate).

Cloud model differences

	IaaS (Infrastructure as a Service)	**CaaS (Container as a Service)**	**FaaS (Function as a Service)**
The app runs in	virtual machines	container	NA
You manage	a lot of things (networking, OS, library, environment, …)	a few things (your image)	only your business code
The app scales in	a couple of minutes	a minute	a second
The app costs	a lot (charges every minute + additional cost)	Depends if with a server or serverless	least expensive