BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to DevOps

Sonika Rathi

Assistant Professor
BITS Pilani

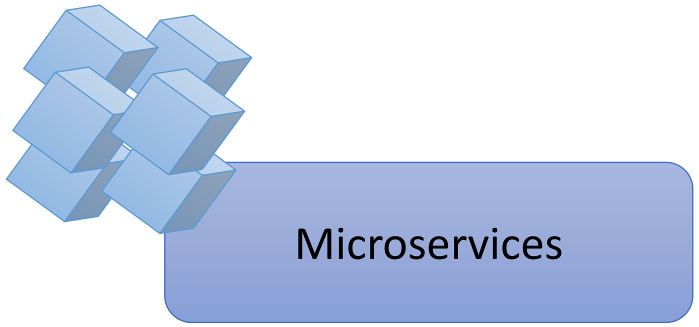
Agenda

Microservices and Current Trends

- Introduction to Microservices
  - Need of Microservices for DevOps
  - Benefits of Microservice and DevOps
- Introduction to Kubernetes
  - Kubernetes Architecture
  - Kubernetes Benefits
- AWS Lambda
  - Function as a Service [FaaS]
  - Serverless Computing
  - AWS Lambda - Success Story
  - AWS Lambda Benefits

Microservices

Introduction



Software development technique

A type of the service-oriented architecture (SOA); that structures an application as a collection of loosely coupled services

It improves modularity

Enables small autonomous teams to develop, deploy and scale their respective services independently

Allows the architecture of an individual service to emerge through continuous refactoring

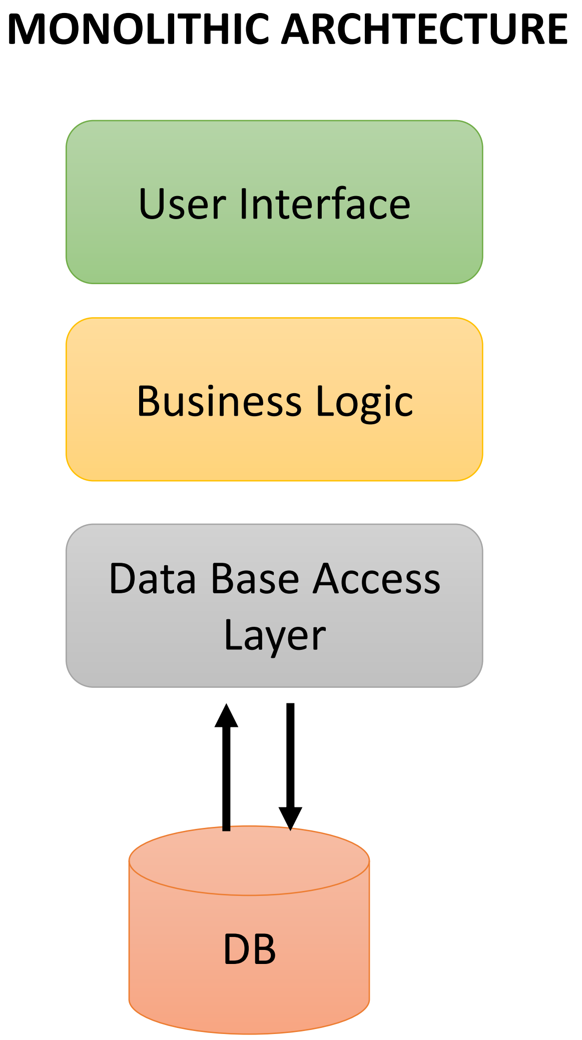
Microservices

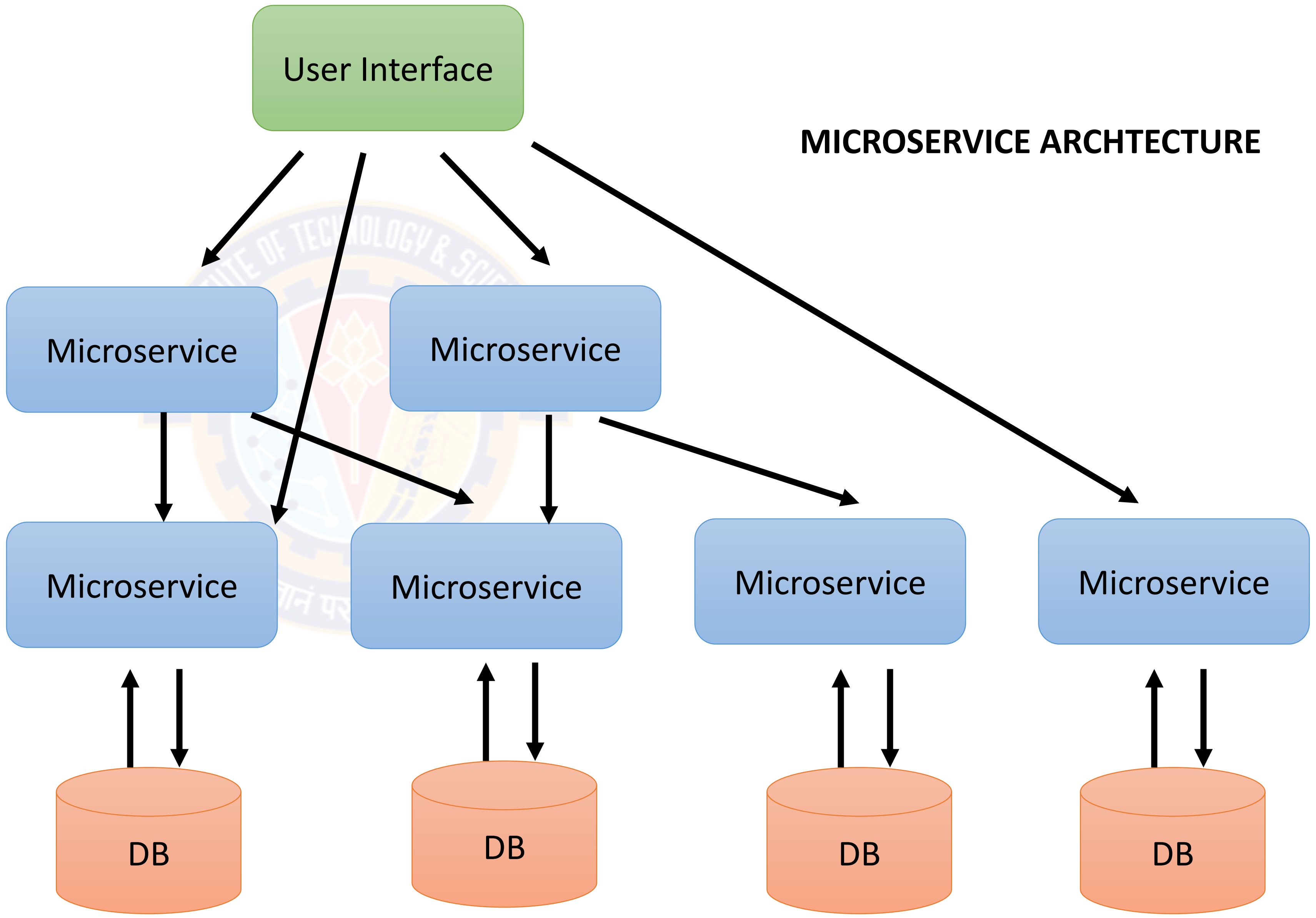
Introduction

Who

Uber
Netflix
Amazon
Ebay
Gilt
Tesla

Microservice Architecture





Microservices

Lets compare

Monolithic – Software Development	Microservices – Software Development
Traditional Software Development [Waterfall]	New Approach to Software Development [Agility]
Large Team Working on single complex Application	Services are encouraged to be small by handful Developers
No Single Developer understand entire Application	Developers understand the Services
Limited Reuse is realized	Use of REST API – reused by other services
Scaling is Challenge	Services can be scaled as they exists Independent
Challenge for Operational Agility	Services and Teams become Agile
Single Development Stack	Autonomous Service Development Stacks

Microservices

How does it works

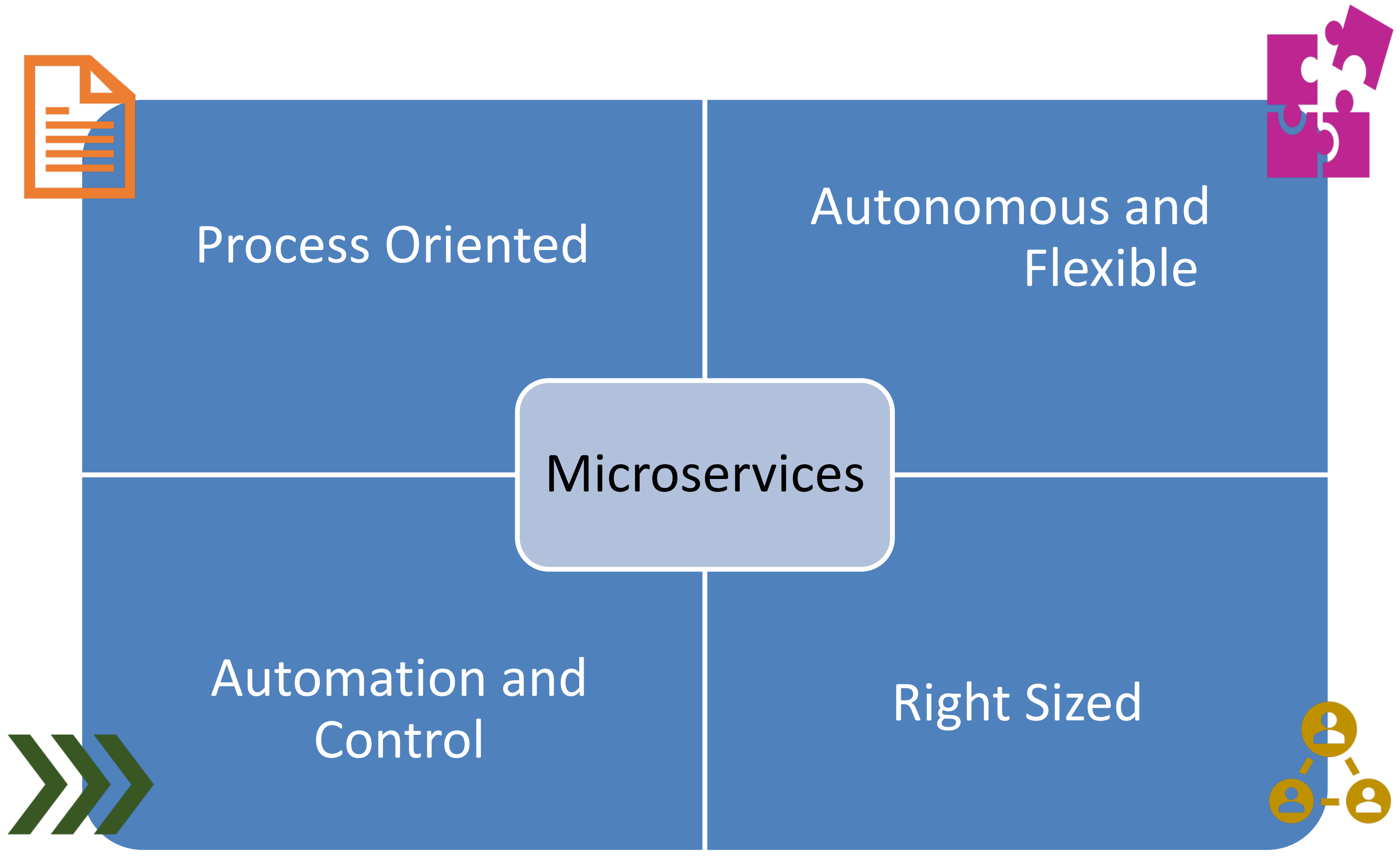
The Idea is to break down applications into smaller, independent services

That will not dependent upon a specific coding language

Using Microservices Ideology divide large & complex applications in to smaller building blocks

Microservices

Characteristics of Microservices



Microservices and DevOps

Need of Microservices for DevOps

- DevOps promotes small, more empowered and autonomous teams, along with automation and measurements
- It has great potential to be a fantastic improvement in the way IT and business work together
- Challenging Area:
  - Setup Pipeline for new automation tooling
  - The mindset of Architects

Microservices and DevOps

Microservices enabling DevOps

- With small autonomous DevOps teams, the world will start producing small autonomous components, called microservices
- It makes sense to produce deployable components for a business function
- This will help in speed improve five to ten times by moving to the cloud and using a high productivity platform
- DevOps and microservices are more or less inseparable, they can hardly exist in separation

Microservices and DevOps

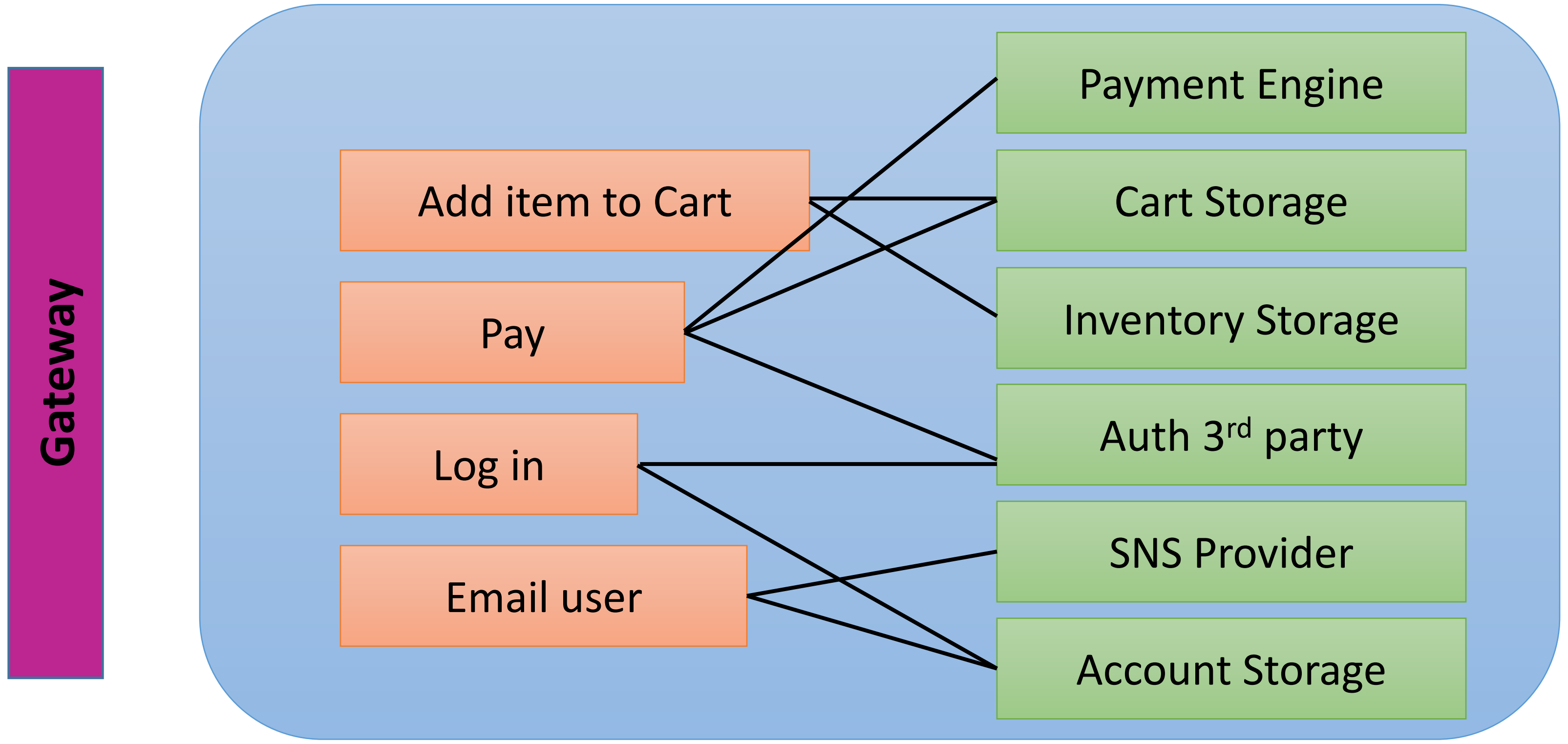
Benefits

Benefits

- Deployability
- Reliability
- Availability
- Scalability
- Modifiability
- Management

Microservices Example

Lets understand with example

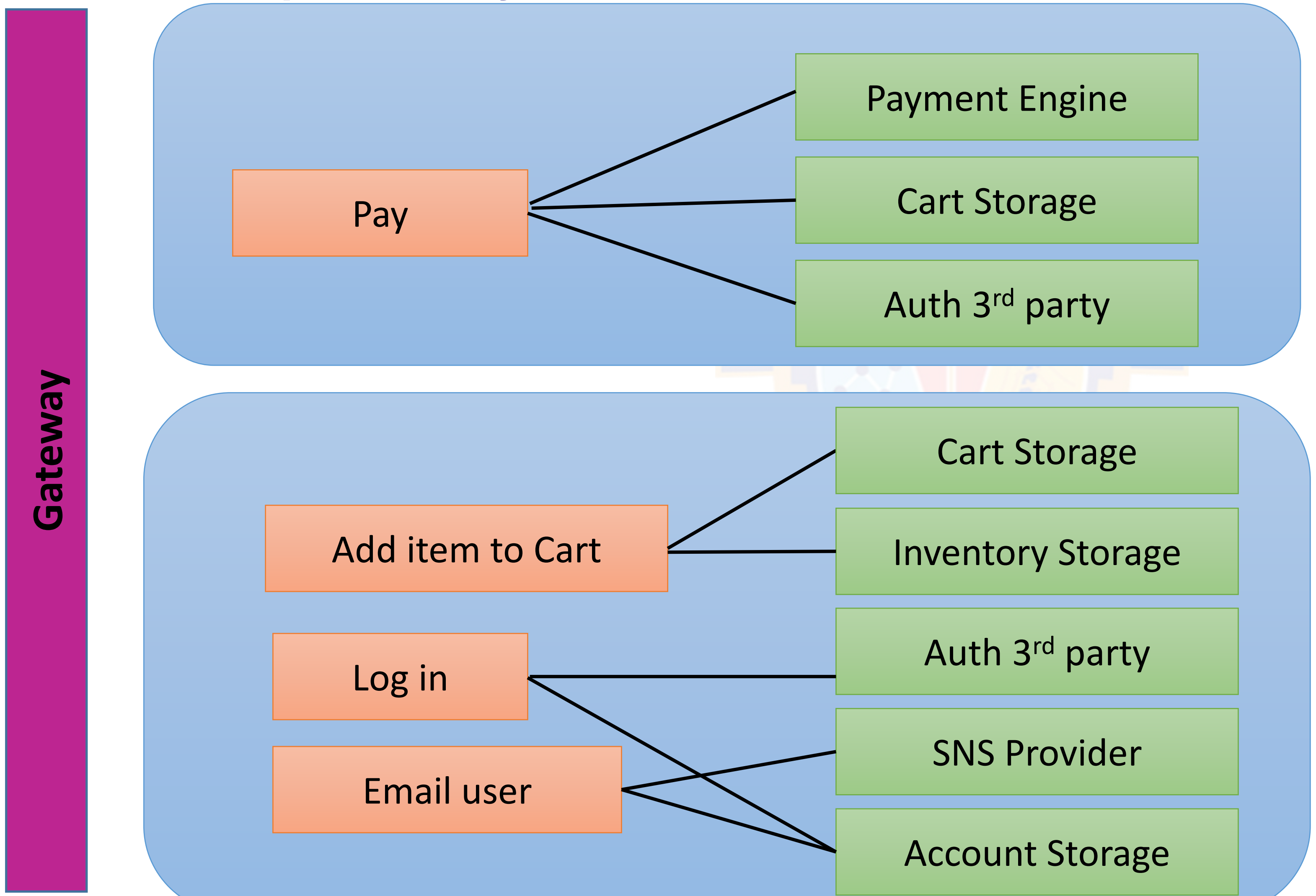


🟠	API endpoints
🟢	External calls to other infrastructure / Services
🔵	Container

Monolith application example

Microservices Example

Our Goal is to split out Payments service from the monolith



- 🟠 API endpoints
- 🟢 External calls to other infrastructure / Services
- 🔵 Container

Kubernetes

Introduction

Kubernetes

Kubernetes is an open source system for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications

Hosted by the Cloud Native Computing Foundation (CNCF)

Symbol K8 - is an abbreviation derived by replacing the 8 letters “ubernete” with “8”

Kubernetes

What?

Kubernetes

A system for running and coordinating containerized applications across a cluster of machines

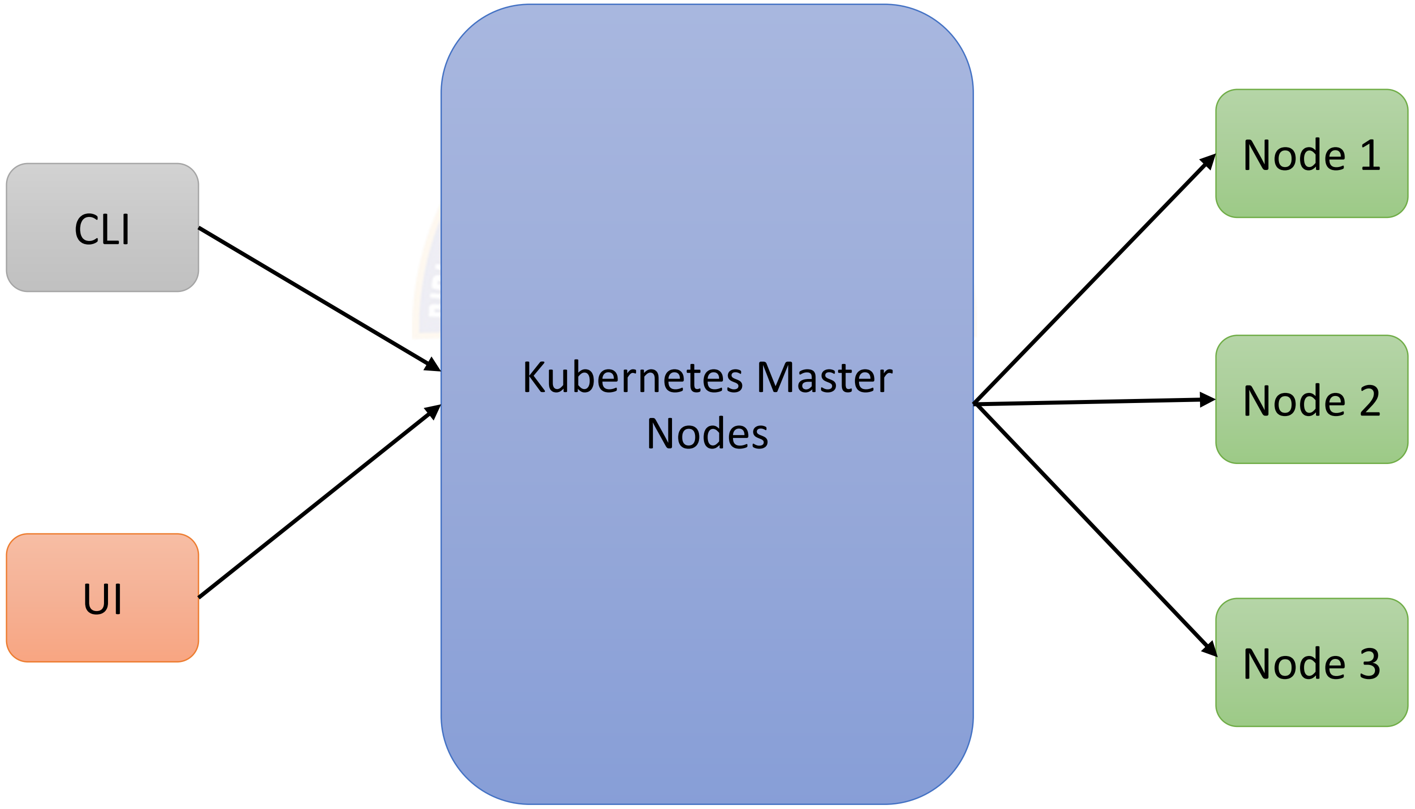
Platform designed to completely manage the life cycle of containerized applications and services

Kubernetes helps with methods that provide

- Predictability
- Scalability
- High availability

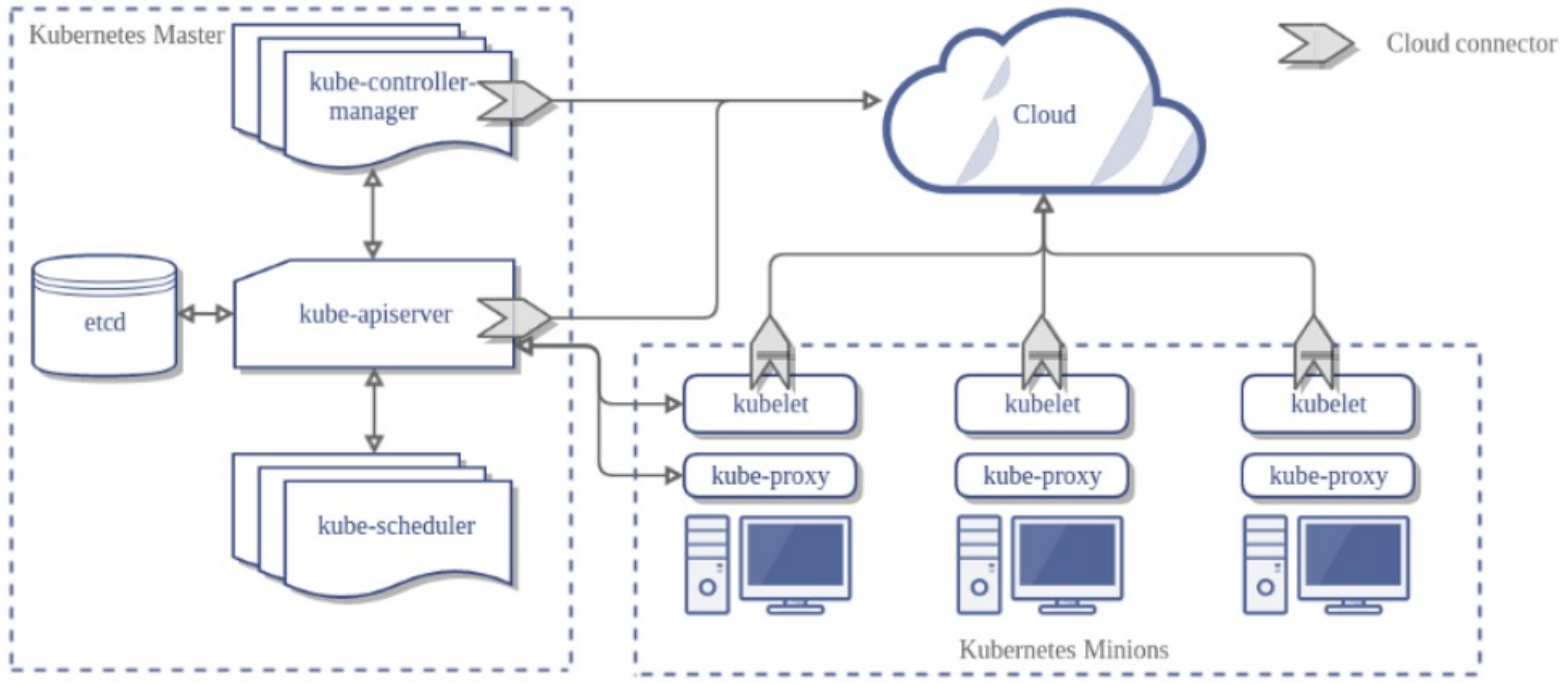
Kubernetes

Components



Kubernetes

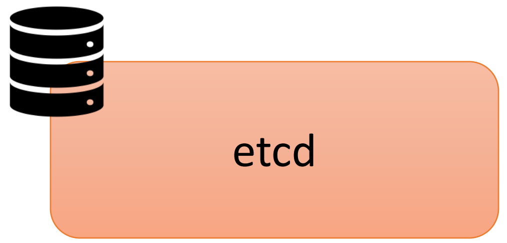
Architecture



Kubernetes Components

Kubernetes Master Server Components

- Kubernetes master server acts as the primary control plane for Kubernetes clusters
- Overall the components on the master server work together to accept user requests, determine the best ways to schedule workload containers, authenticate clients and nodes, adjust cluster-wide networking, and manage scaling and health checking responsibilities
- These components can be installed on a single machine or distributed across multiple servers



Globally available configuration store

Distributed key value store

Configured to span across multiple nodes

etcd can be configured on a single master server or, in production scenarios, distributed among a number of machines

Kubernetes Components

Kubernetes Master Server Components

kube-apiserver

Main Management Point

It allows a user to configure Kubernetes' workloads and organizational units

It is also responsible for making sure that the etcd store and the service details of deployed containers are in agreement

The API server implements a RESTful interface

kube-controller-manager

It manages different controllers that regulate the state of the cluster, manage workload life cycles, and perform routine tasks

A replication controller ensures that the number of replicas defines for a pod machine

When a change is seen, the controller reads the new information and implements the procedure that fulfills the desired state

Kubernetes Components

Kubernetes Master Server Components

kube-scheduler

The process that actually assigns workloads to specific nodes in the cluster is the scheduler

The scheduler is responsible for tracking available capacity on each host to make sure that workloads are not scheduled in excess of the available resources

Kubernetes Components

Kubernetes Node Server Components

- In Kubernetes, servers that perform work by running containers are known as nodes
- Node servers have a few requirements that are necessary for :
  - Communicating with master components
  - Configuring the container networking
  - Running the actual workloads assigned

The first component that each node must have

This requirement is satisfied by installing and running Docker

Responsible for starting and managing containers

Runs the containers defined in the workloads submitted to the cluster

Kubernetes Components

Kubernetes Node Server Components

kubelet

The main contact point for each node with the cluster group

Responsible for relaying information to and from the control plane services

Interact with the etcd store to read configuration details or write new values

The kubelet service communicates with the master components to authenticate to the cluster and receive commands and work

kube-proxy

To manage individual host subnetting and make services available to other components

This process forwards requests to the correct containers

It can do primitive load balancing

Kubernetes Objects

Kubernetes Pods

- One or more tightly coupled containers are encapsulated in an object called a pod
- Generally represents one or more containers that should be controlled as a single application
- Consist of containers that operate closely together, share a life cycle, and should always be scheduled on the same node
- Share their environment, volumes, and IP space
- Consist of a main container that satisfies the general purpose of the workload and optionally some helper containers that facilitate closely related tasks

Kubernetes

Benefits

Benefits

Portable

100% open source

Allows easy container management

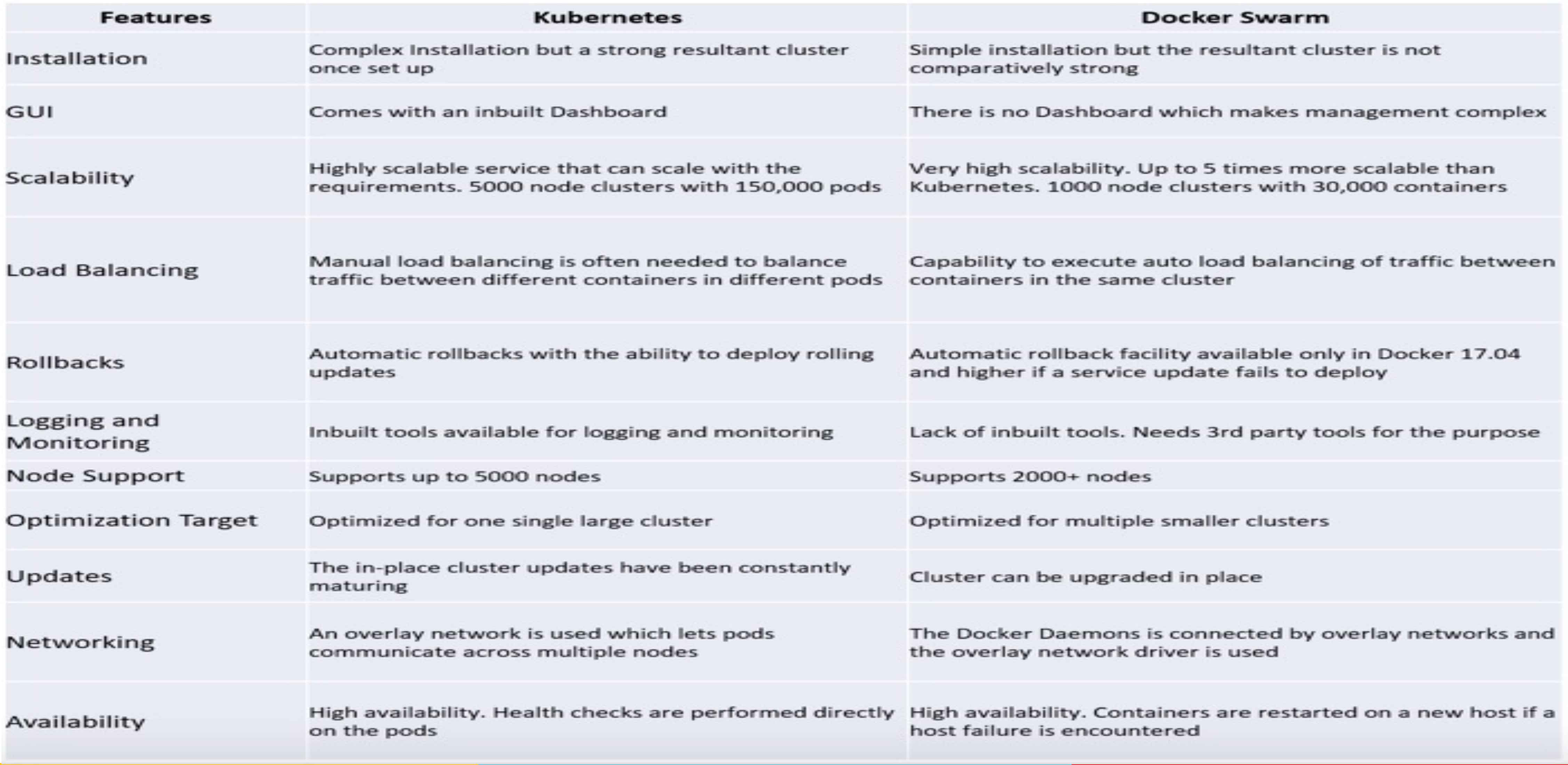
Allows Workload Scalability

Supports High Availability

Efficient

Kubernetes II Docker Swarm

Lets compare



Features	Kubernetes	Docker Swarm
Installation	Complex Installation but a strong resultant cluster once set up	Simple installation but the resultant cluster is not comparatively strong
GUI	Comes with an inbuilt Dashboard	There is no Dashboard which makes management complex
Scalability	Highly scalable service that can scale with the requirements. 5000 node clusters with 150,000 pods	Very high scalability. Up to 5 times more scalable than Kubernetes. 1000 node clusters with 30,000 containers
Load Balancing	Manual load balancing is often needed to balance traffic between different containers in different pods	Capability to execute auto load balancing of traffic between containers in the same cluster
Rollbacks	Automatic rollbacks with the ability to deploy rolling updates	Automatic rollback facility available only in Docker 17.04 and higher if a service update fails to deploy
Logging and Monitoring	Inbuilt tools available for logging and monitoring	Lack of inbuilt tools. Needs 3rd party tools for the purpose
Node Support	Supports up to 5000 nodes	Supports 2000+ nodes
Optimization Target	Optimized for one single large cluster	Optimized for multiple smaller clusters
Updates	The in-place cluster updates have been constantly maturing	Cluster can be upgraded in place
Networking	An overlay network is used which lets pods communicate across multiple nodes	The Docker Daemons is connected by overlay networks and the overlay network driver is used
Availability	High availability. Health checks are performed directly on the pods	High availability. Containers are restarted on a new host if a host failure is encountered

AWS Lambda

Function as a Service [FaaS]

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers
- “serverless” computing
- You pay only for the compute time you consume - there is no charge when your code is not running
- Lambda functions can write state to external data stores via web requests
- External Data store can be of AWS or Any
  - AWS Solutions: S3, Dynamo, and Redshift
  - Other Solutions: PostgreSQL, Cassandra and Kafka

AWS Lambda : Success Story The Seattle Times

Challenges

After maintaining on-premises hardware and custom publishing software for nearly two decades, The Seattle Times sought to migrate its website publishing to a contemporary content management platform

To avoid the costs of acquiring and configuring new hardware infrastructure and the required staff to maintain it, the company initially chose a fully managed hosting vendor

But after several months, The Times' software engineering team found it had sacrificed flexibility and agility in exchange for less maintenance responsibility

As the hosted platform struggled with managing traffic under a vastly fluctuating load, The Seattle Times team was constrained in its ability to scale up to meet customer demand

AWS Lambda : Success Story The Seattle Times

Why Amazon Web Services?

To address these core scalability concerns, The Seattle Times engineering team considered several alternative hosting options, including self-hosting on premises, more flexible managed hosting options, and various cloud providers

The team concluded that the available cloud options provided the needed flexibility, appropriate architecture, and desired cost savings. The company ultimately chose Amazon Web Services (AWS), in part because of the maturity of the product offering and, most significantly, the auto-scaling capabilities built into the service

The Seattle Times deployed its new system in just six hours. The website moved to the AWS platform between 11 p.m. and 3 a.m. and final testing was completed by 5 a.m. — in time for the next news day

AWS Lambda : Success Story The Seattle Times

End Result

With AWS, The Seattle Times can now automatically scale up very rapidly to accommodate spikes in website traffic when big stories break, and scale down during slower traffic periods to reduce costs

“Auto-scaling is really the clincher to this,” Seattle Times says. “With AWS, we can now serve our online readers with speed and efficiency, scaling to meet demand and delivering a better reader experience.”

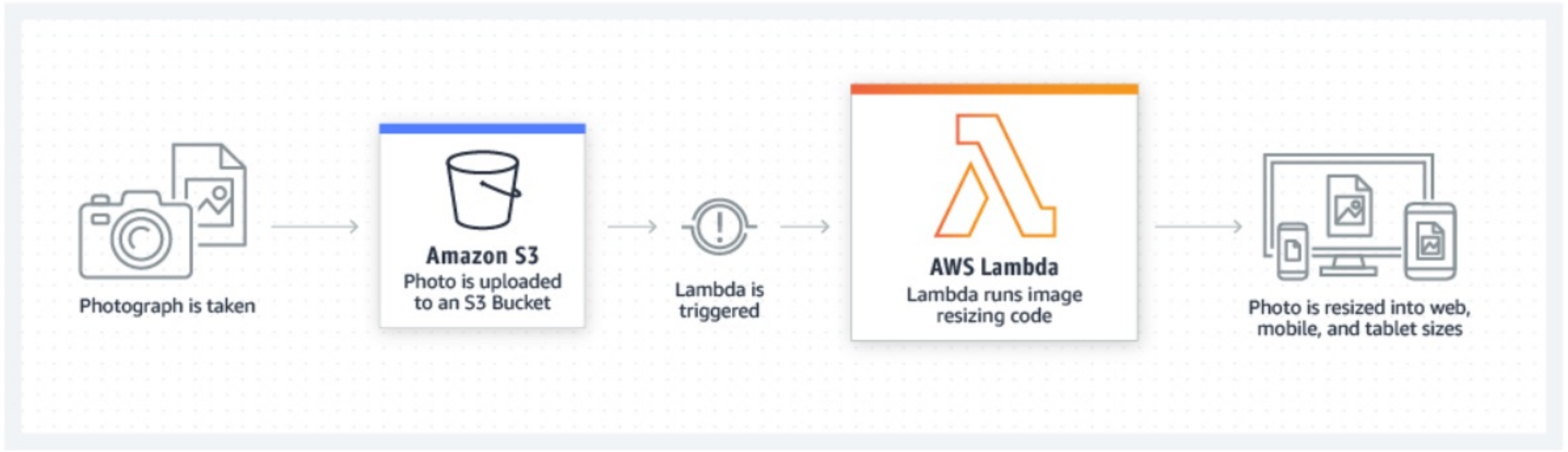
"AWS Lambda provides us with extremely fast image resizing,” Seattle times says.

“Before, if we needed an image resized in 10 different sizes, it would happen serially. With AWS Lambda, all 10 images get created at the same time, so it’s quite a bit faster and it involves no server maintenance.”

AWS Lambda : Success Story The Seattle Times

How it worked

Reference Architecture: Sample Code


Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

References

External

- **Docker :** https://docs.docker.com/
- **Microservices :**
- https://www.mendix.com/blog/the-microservices-you-need-for-devops/
- https://www.mulesoft.com/resources/api/what-are-microservices
- **Kubernetes :** https://kubernetes.io/docs/concepts/
- **Kubernetes in 5 mins by VMware:** https://www.youtube.com/watch?v=PH-2FfFD2PU
- **AWS Lambda:** https://www.youtube.com/watch?time_continue=1&v=eOBq__h4OJ4

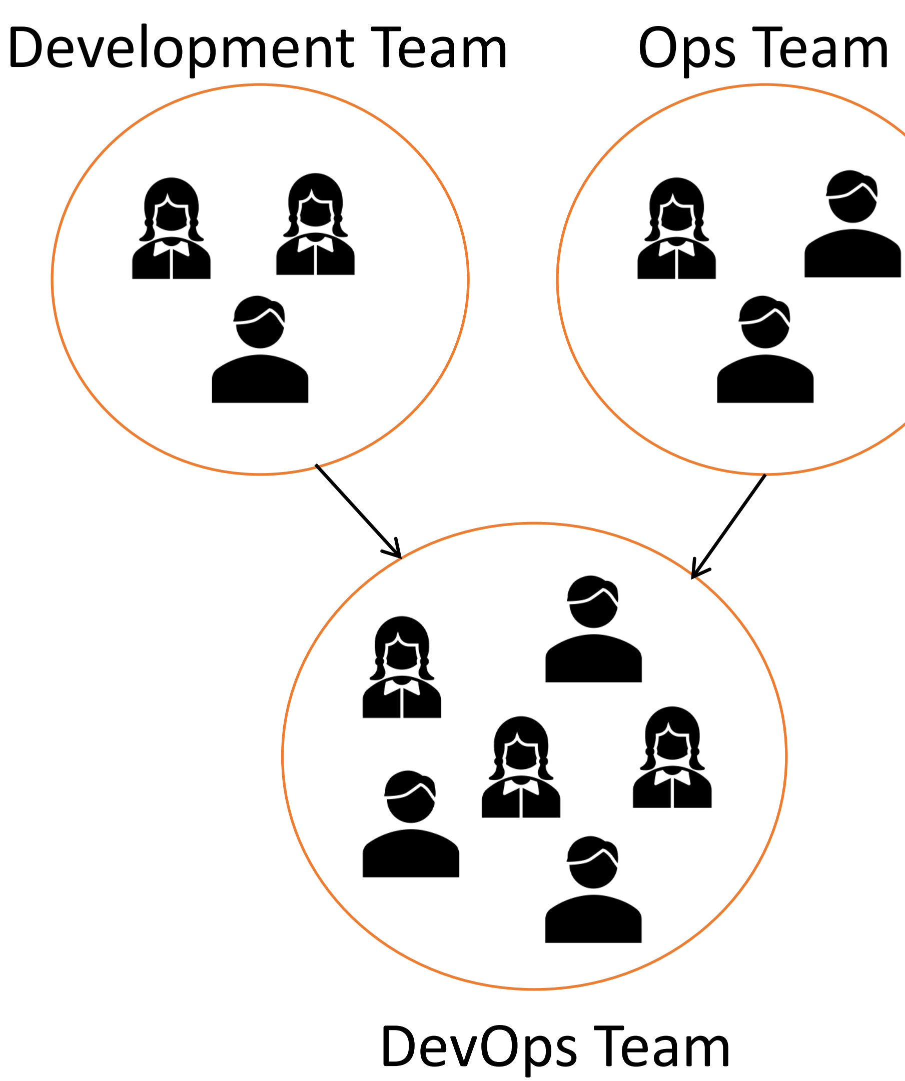
SRE & DevOps

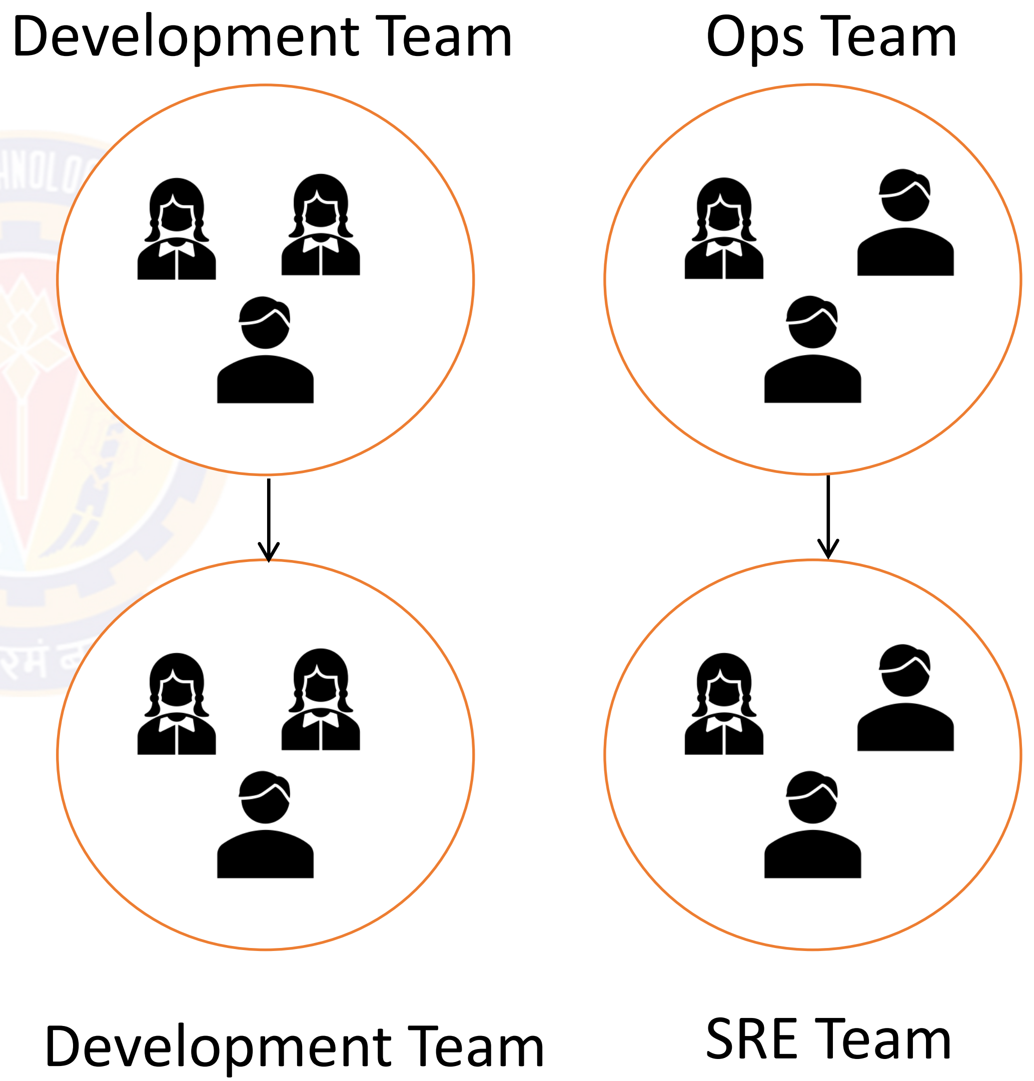
Two sides of same coins

	SRE	DevOps
Essence	Set of practices & metrics	Mindset and culture of collaboration
Coined	2003, by Ben Treynor, @ Google	2009, by Patrick Debois
Goal	Bridge the gap between Dev & Operation	Bridge the gap between Dev & Operation
Focus	Site availability & Reliability	Continuity & Speed, Early time to market, stability
Team Structure	Site reliability engineers with ops and development skills	Wide range of role Product owners, developers, QA engineers, SREs etc.,

DevOps & SRE

Big Picture





Q&A

Thank You!

In our next session: