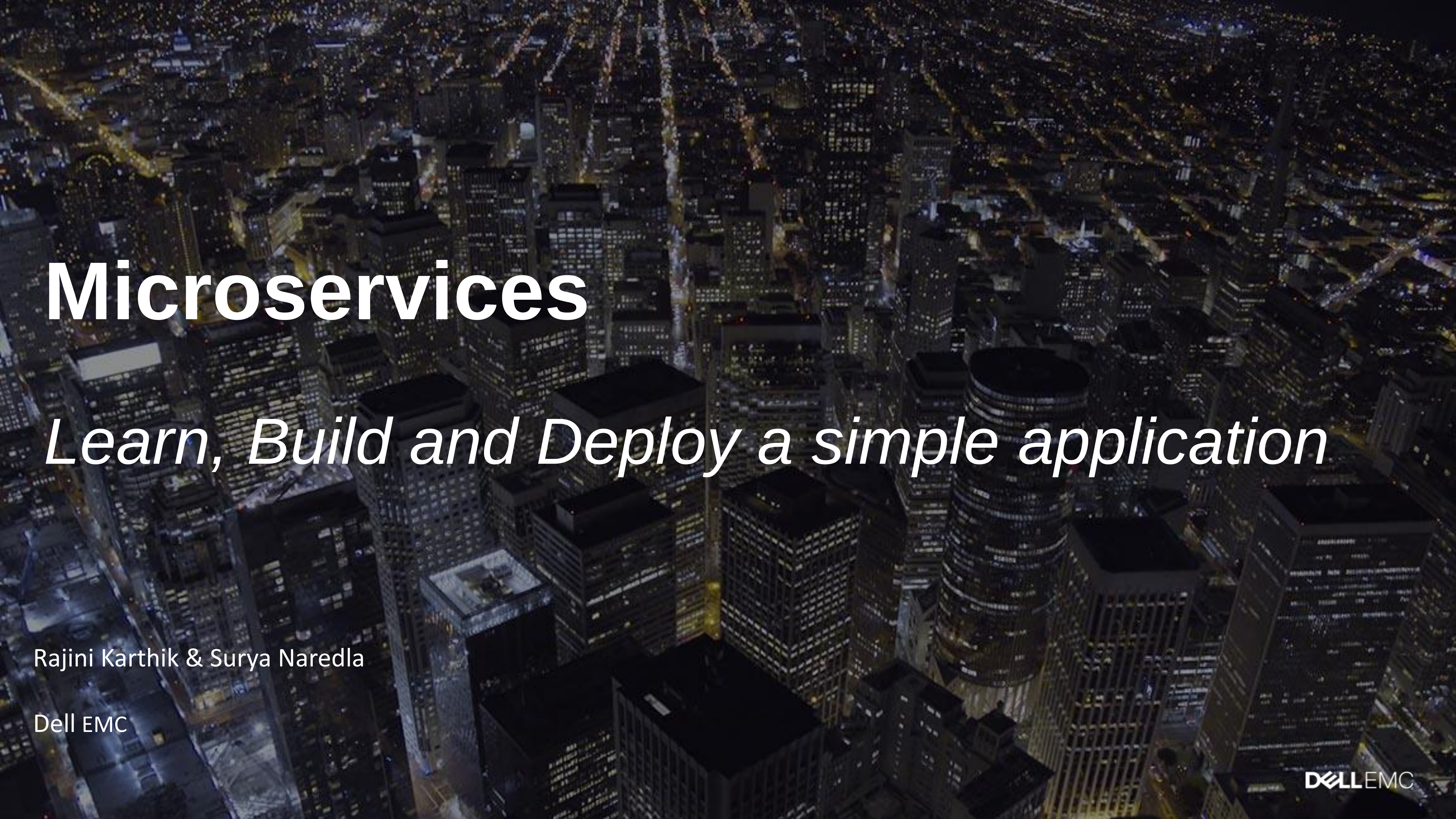


Microservices

Learn, Build and Deploy a simple application

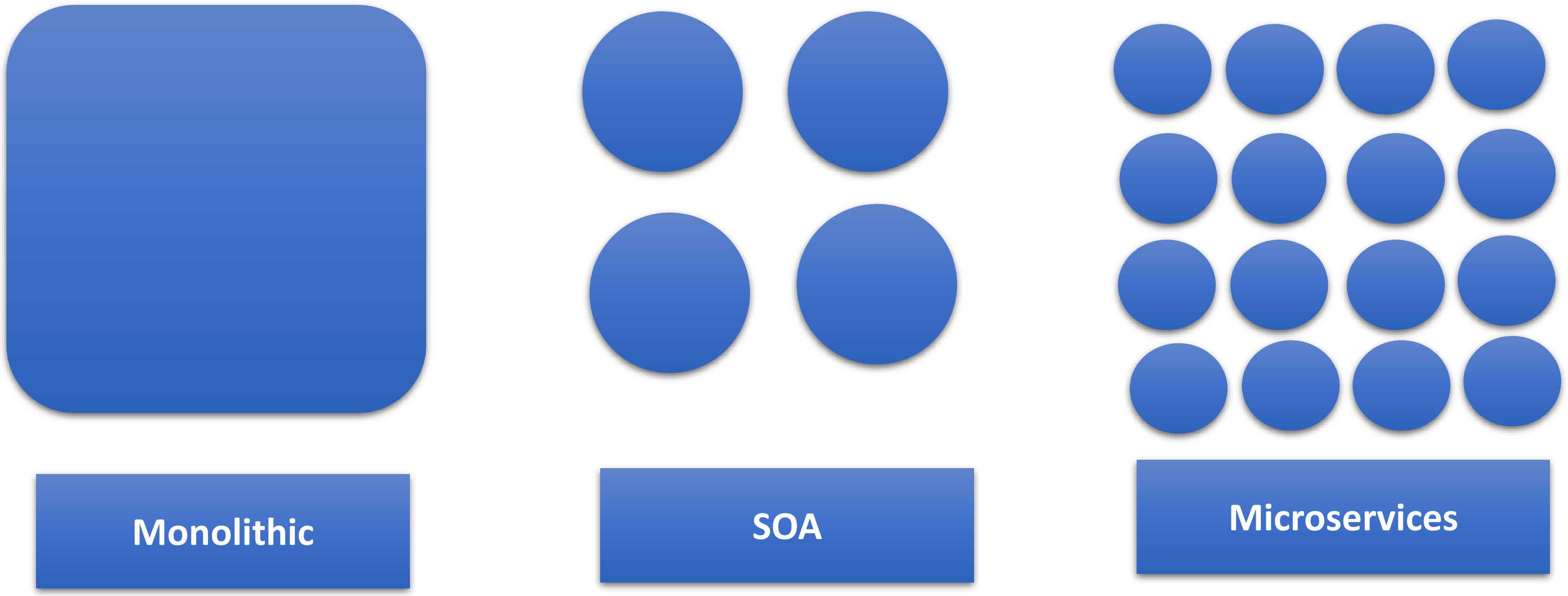
Rajini Karthik & Surya Naredla

Dell EMC

What is a Microservice?

- A software design pattern
- A collection of small autonomous services, modelled around a business domain
- Loosely coupled, independently deployable unit of code
- Separates business functions
- Communicate via well-defined APIs usually HTTPs
- Discoverable through some form of service discovery

Monolithic vs SOA vs Microservices



Why Microservices?

Challenges of Monolithic Architecture

Limited - Limitations due to size and complexity of the system

Flexibility – Cannot mix technologies and cannot adopt new technologies

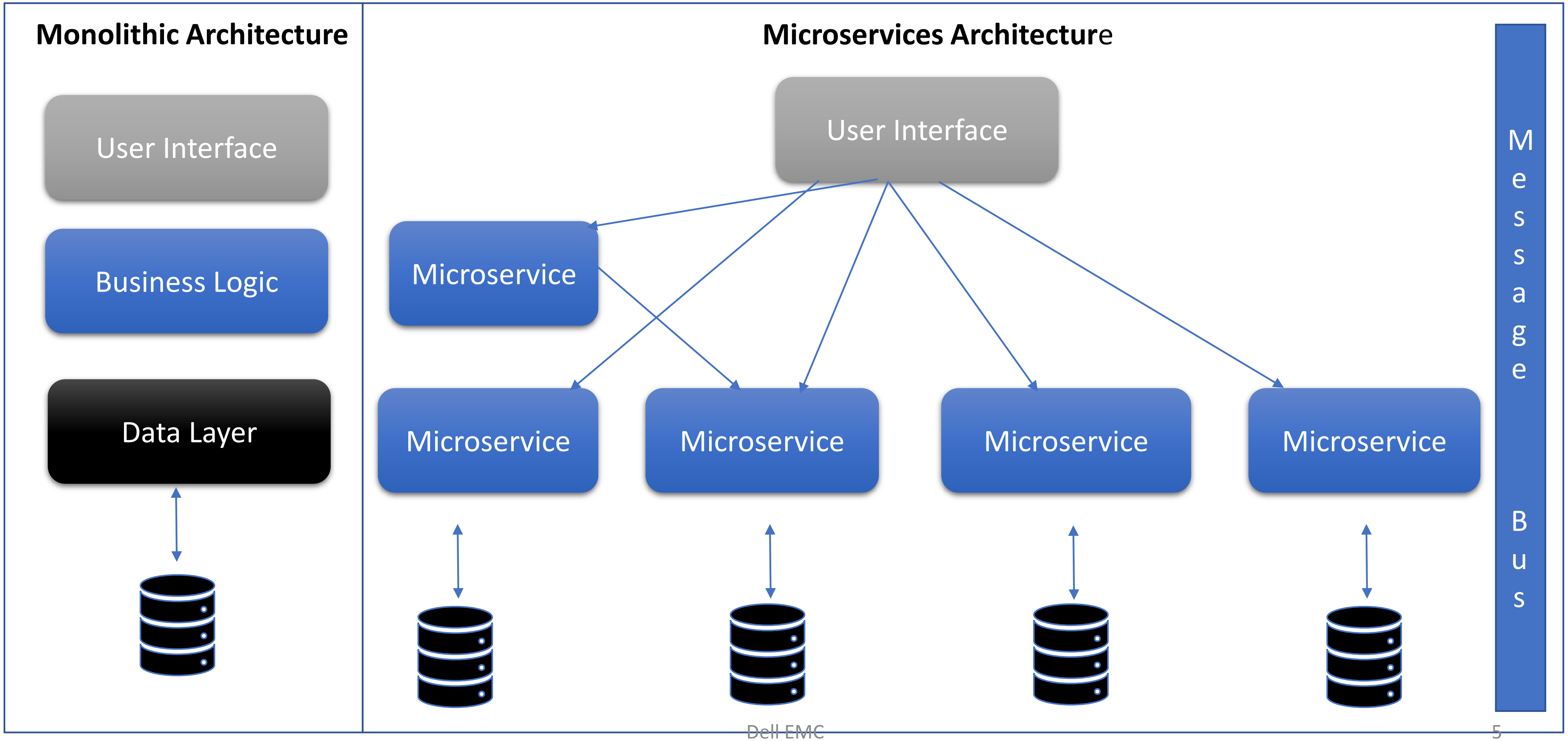
Reliability – Any change or bug will impact the whole system as they are tightly coupled

Scalability – Applications cannot be scaled easily based on individual features due to resource requirements

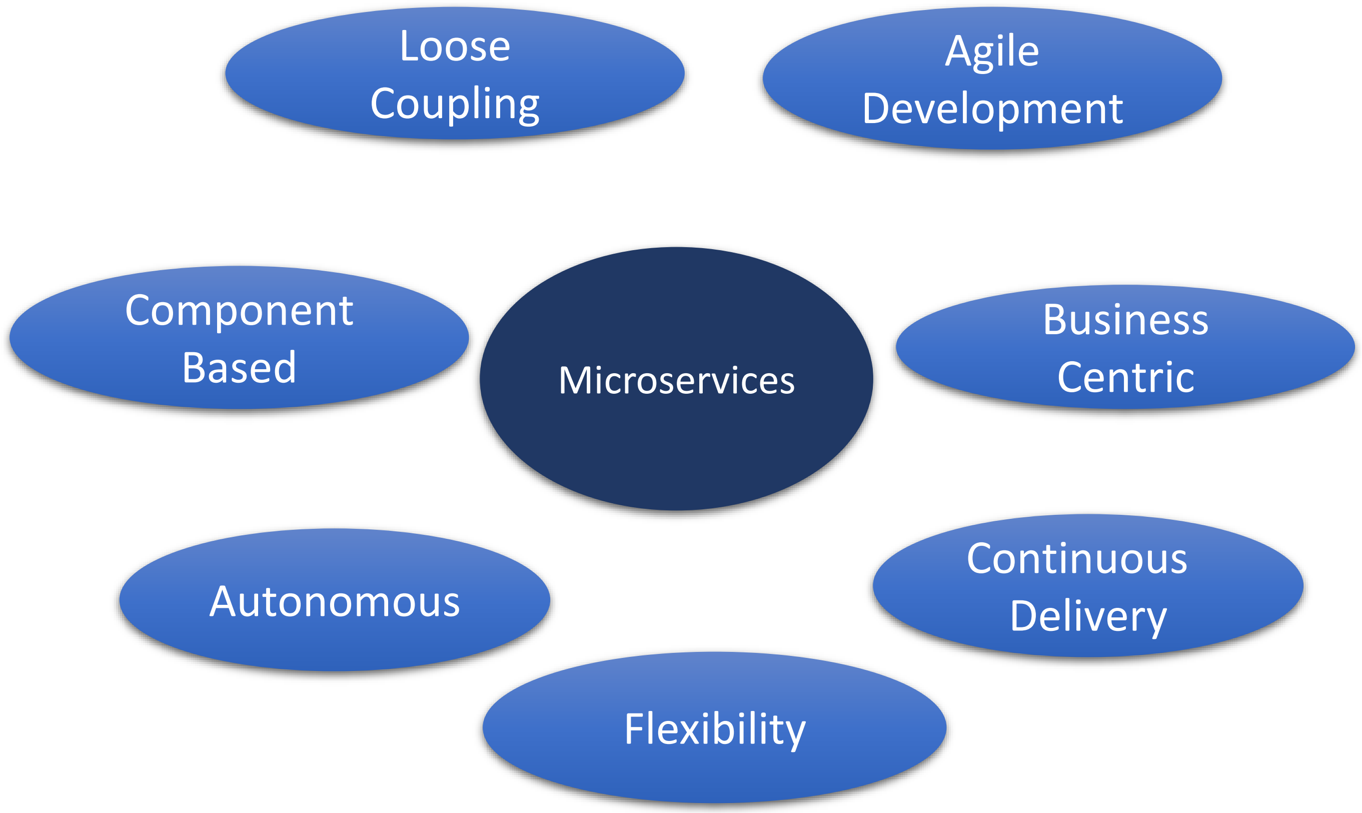
Development - Development takes time

Continuous Development – Multiple features cannot be built and deployed continuously and independently

Monolithic vs Microservices



Microservices Features



Pros & Cons of Microservices

Pros

- Development Flexibility
- Deployment Flexibility
- Service Isolation
- Technology Stack
- Component Based Scaling

Cons

- Planning
- Cost
- Component Integration
- Over Engineering
- Others
  - Eg- Troubleshooting, performance, code-sharing

Microservices on Containers vs. VMs

- **Low Cost**
  - For example OS License Cost
- **Performance and Efficiency**
  - CPU overhead
- **Reduced Size**
  - Container are smaller than VMs
- **Faster Execution**
  - Creation time

Microservices Architecture Considerations

- **Design** - API Gateway, Data Storage
- **Scalability** - Autoscaling
- **Availability** - Resource Constraints
- **Security** - Pod and Container Security
- **Deployment (CI/CD)** - Independent Deployment, updates and upgrades etc...

Container Orchestrators

docker SWARM

kubernetes

diego

Google Container Engine

MARATHON

Azure Container Service

RED HAT® OPENSHIFT Container Platform

kubernetes

Kubernetes

Overview

Why Kubernetes?

Supports microservices architecture through the service objects

Deployment, scaling and management of containerized applications and services

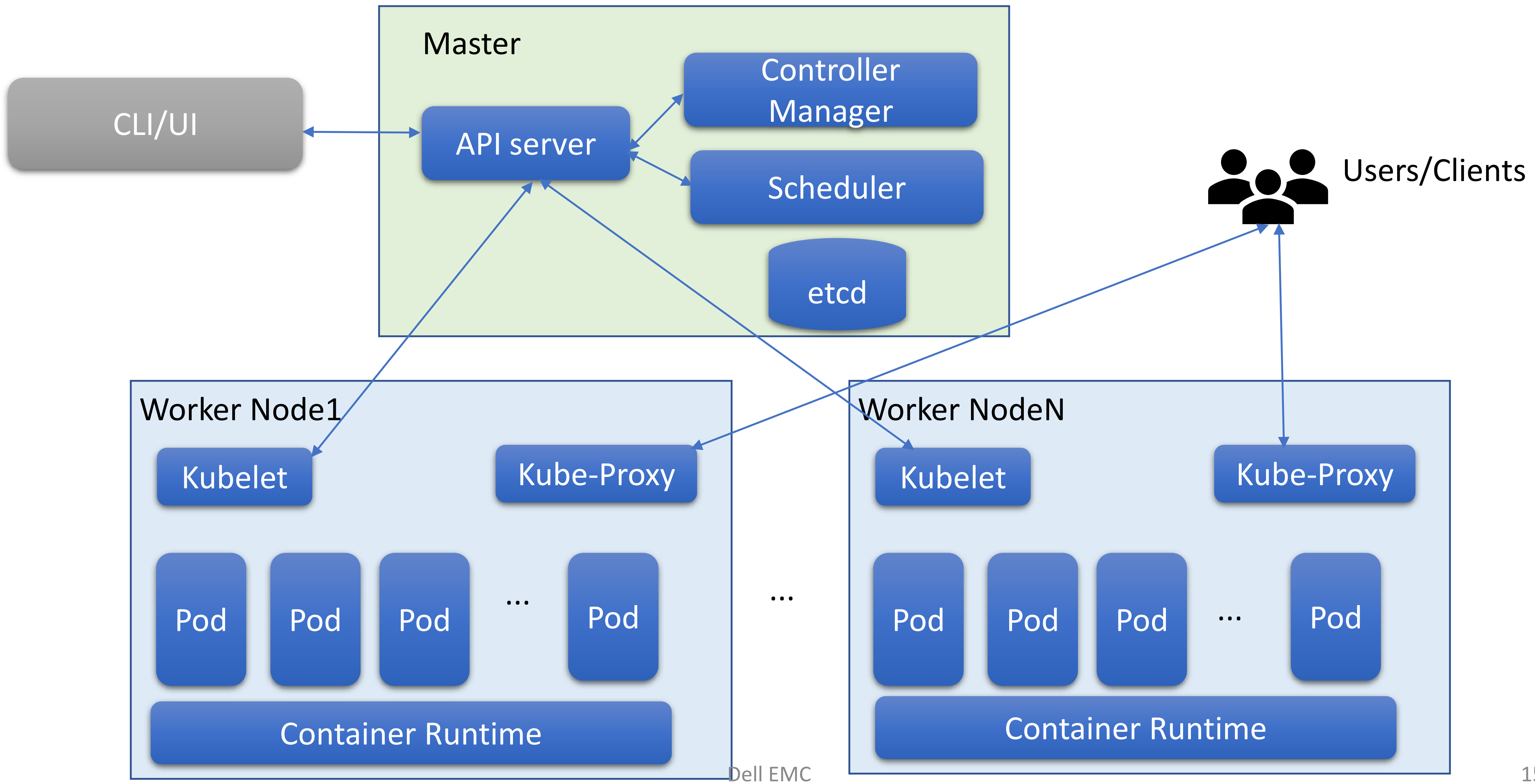
Kubernetes Terminology

1. **Cluster** - Network of containers
2. **Nodes** - Machines on which a cluster runs. Can be master or node
3. **Pod** - A group of one or more containers
4. **Service** - an abstraction which defines a logical set of Pods and a policy to access them
5. **ReplicaSet** - Ensures a given number of pods are running and available in the cluster at a given time
6. **Deployment** - descriptor of the desired state on creating and updating instances

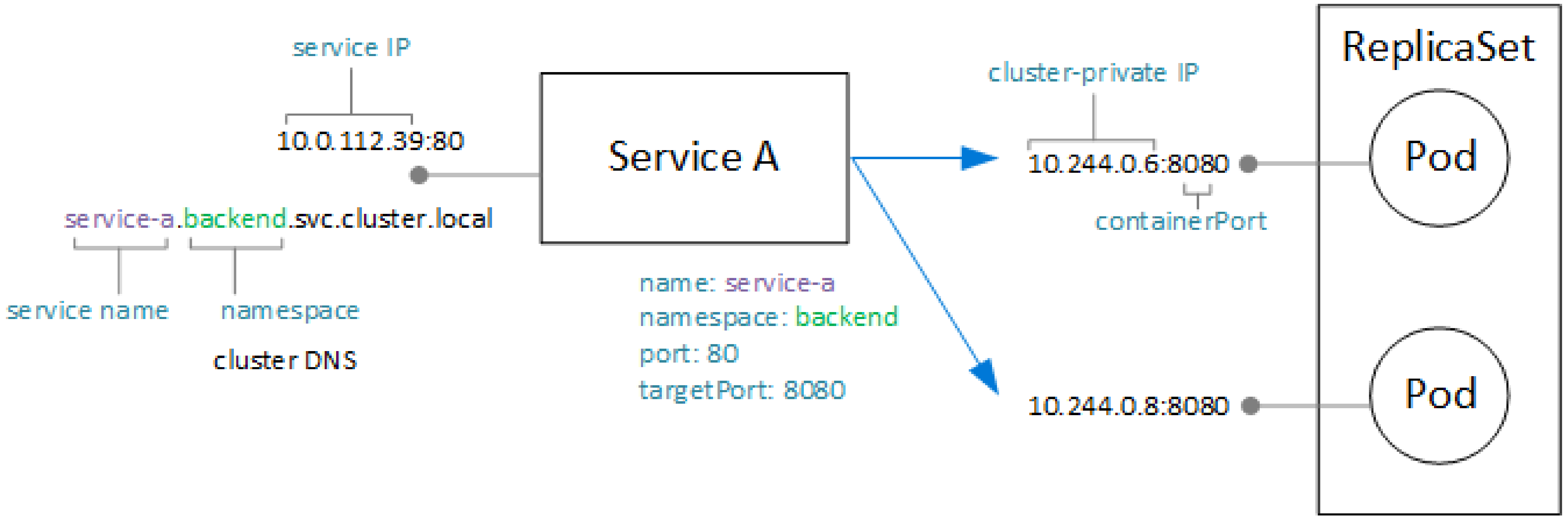
7. **Kubectl** - CLI tool for interacting with the Kubernetes cluster
8. **Volume** - sometimes-shared, persistent storage
9. **Master** -   Control plane entity responsible for managing the cluster (API Server, Scheduler, etcd (key-value store), controller-manager)
10. **Node** - Worker machine or VM in the cluster (kube-proxy, kubelet, container Runtime)

1. **Namespace** - Namespaces organize services within the cluster
7. **StatefulSets** - Maintain the state of applications beyond an individual pod lifecycle, such as storage

Kubernetes Architecture

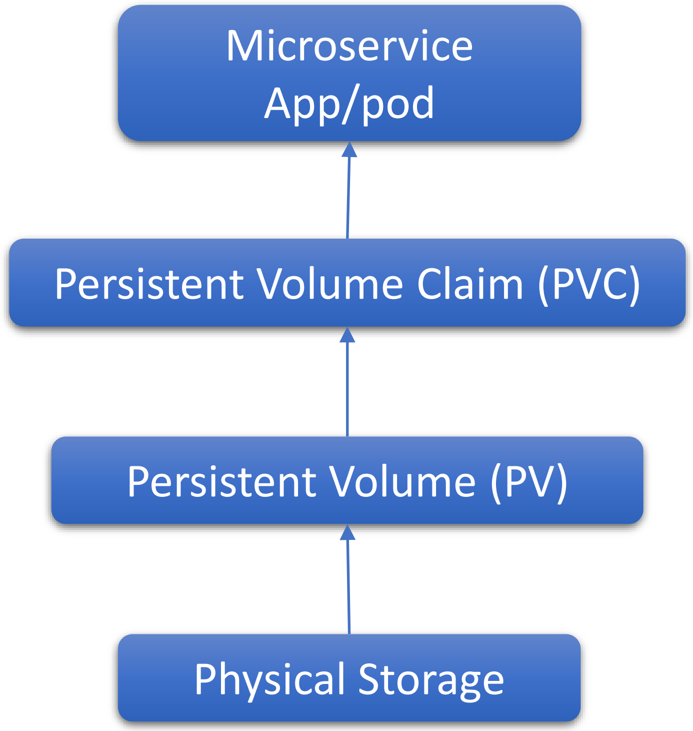


Kubernetes – Services and Pods



Reference: Microservices architecture on Azure Kubernetes Service from https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/microservices/aks

Kubernetes – Storage Primitives



- **PersistantVolume (PV)** – Administrator provisioned volumes, a virtual storage instance pointing to physical storage
- **PersistantVolumeClaim (PVC)** - A request for storage. PVCs consumes PVs

Kubernetes Service Providers

Minikube

An open-source tool that you can install in your local machine to use Kubernetes locally.

Google Kubernetes Engine (GKE)

Google's solution that manages production-ready Kubernetes clusters

Amazon Elastic Kubernetes Service (EKS)

Amazon's solution that manages production-ready Kubernetes clusters

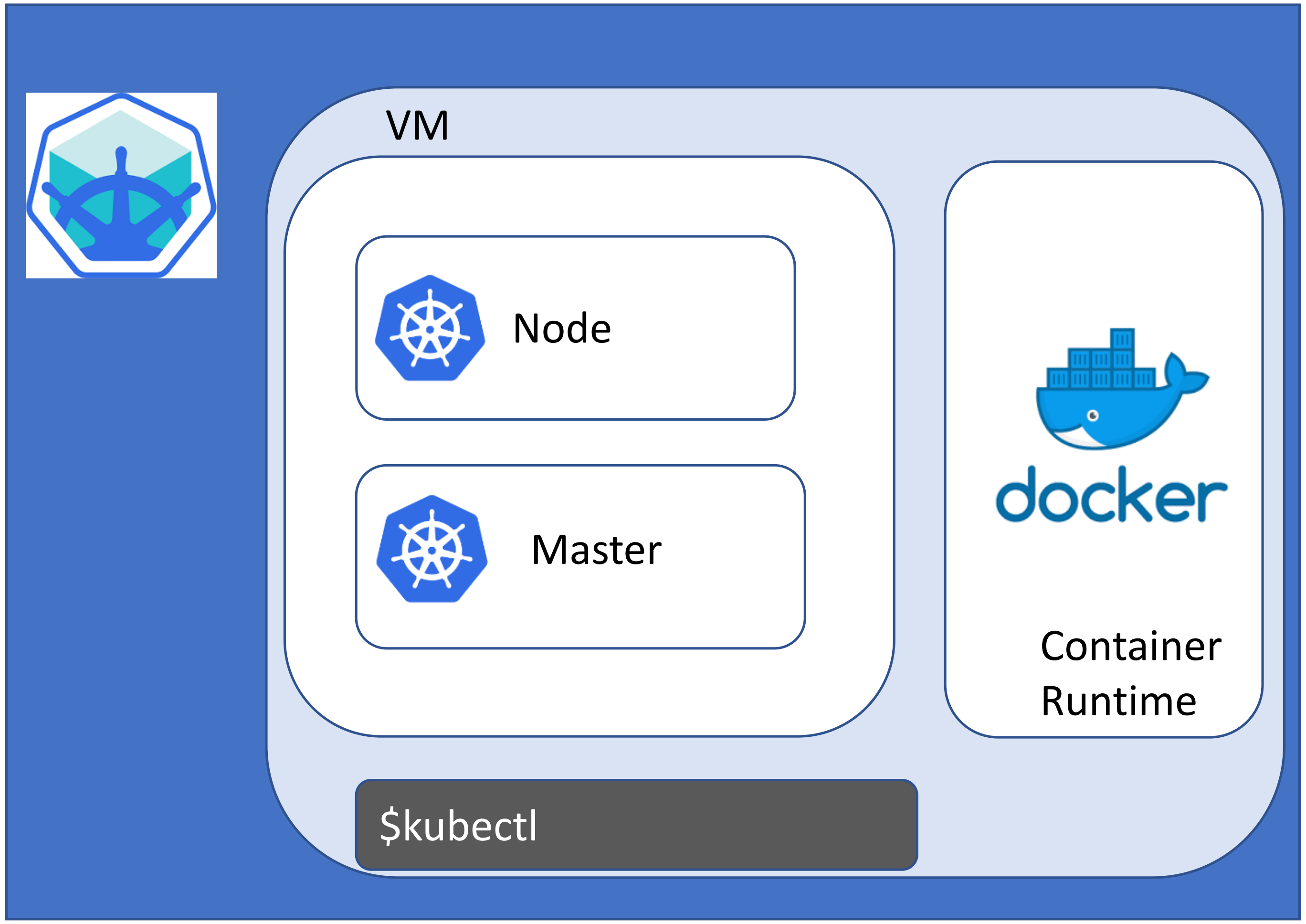
Azure Kubernetes Service (AKS)

Microsoft's solution that provides you managed, production-ready Kubernetes clusters.

OpenShift Kubernetes

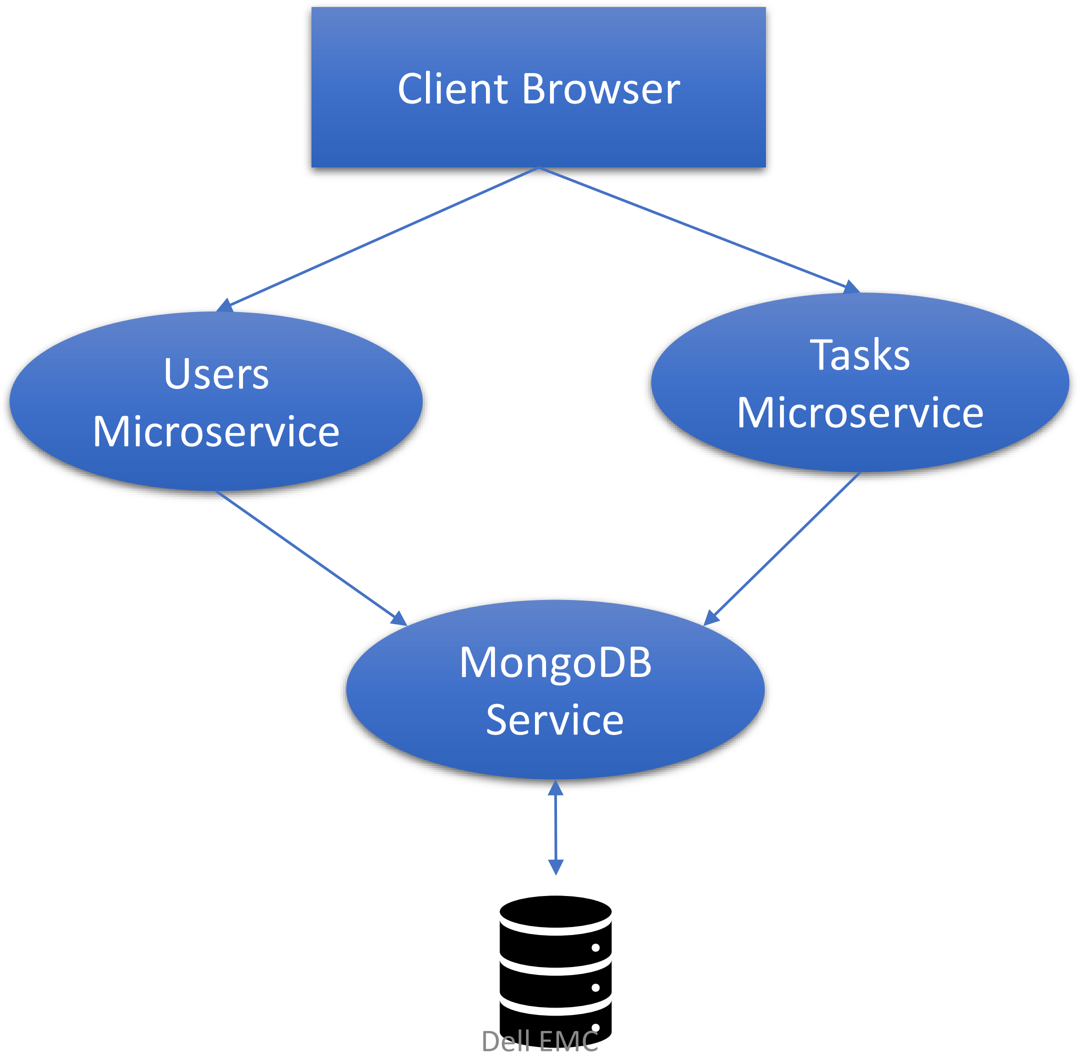
Red Hat's solution that handles Kubernetes clusters for you

Minikube



Demo Application

Application Architecture



Lab Logistics

GitHub:
https://github.com/rajinir/microservices-kubernetes-sample

Login:
Use the ssh private key in the Github, login into the instance provided to you using a ssh terminal (putty etc..)
https://github.com/rajinir/microservices-kubernetes-sample/tree/master/setup/id_rsa

Lab Section1
Instance, SSH and Setup

Lab Section2.
Deploy Minikube and Kubectl

Lab Section3
Minikube and KubeCtl

Lab Section4
Deploy MongoDB Pod

Lab Section5
Deploy Users-Tasks Sample Application

Lab Section6
Test the application

Tools & References

Ecosystem Tools

Prometheus - A monitoring solution for Kubernetes cluster

Istio - A tool that supports service deployment in kubernetes. It is used for connecting, monitoring and securing microservices

References

- https://www.microservices.com
- https://microservices.io/
- https://kubernetes.io/
- https://docs.docker.com/engine/reference/builder/
- http://eventuate.io/exampleapps.html

Thank you

Contacts

- IRC: rajinir
- Email: rajini_karthik@dell.com
- Email: surya_prabhakar@dell.com

DELL EMC