UPPSALA
UNIVERSITET

Lecture-4: Distributed Computing Infrastructures (DCI) Part-I

Presenter: Salman Toor

Salman.Toor@it.uu.se

UPPSALA
UNIVERSITET

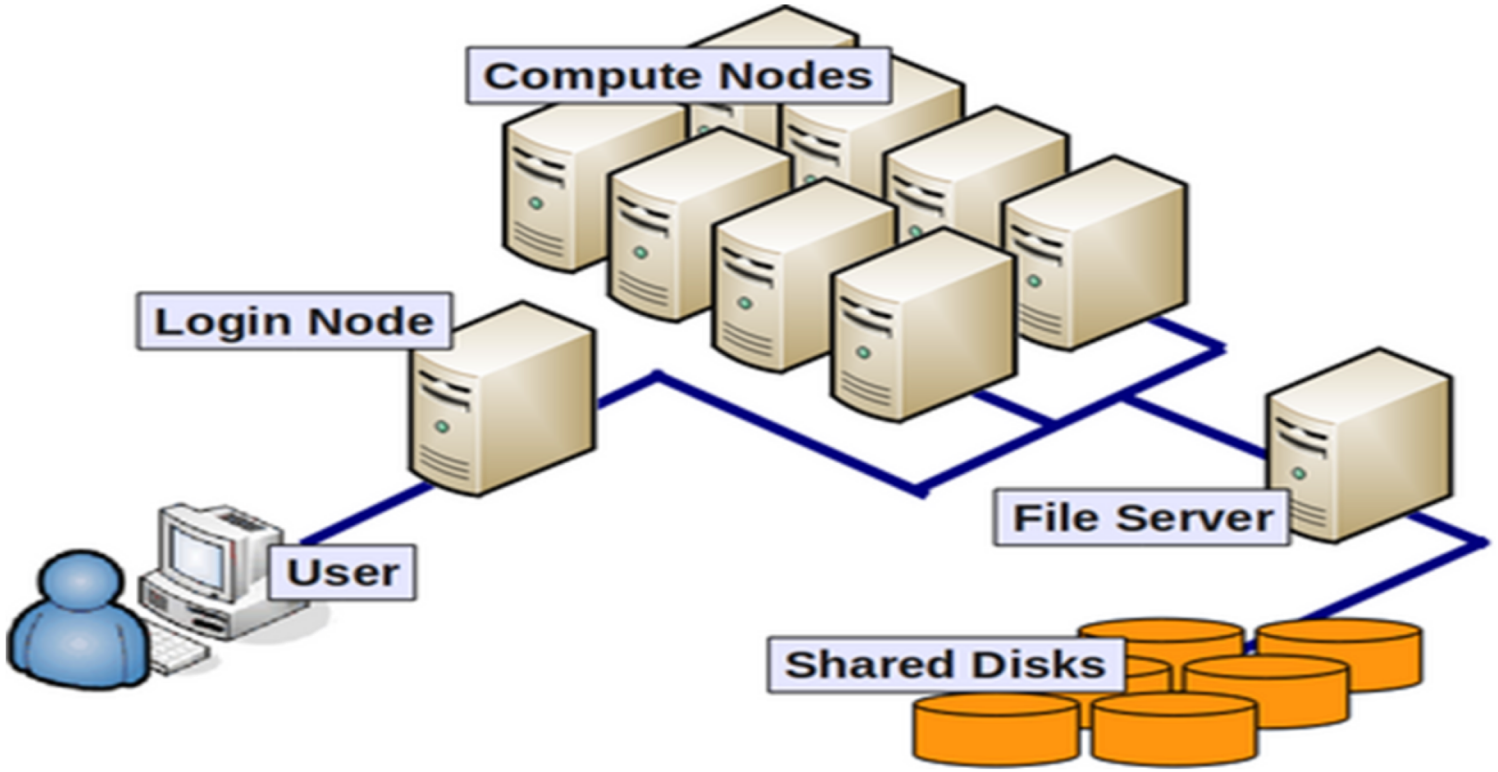
Distributed computing e-infrastructures (DCI)

- Broad term used for dedicated and public resources
  - Dedicated resources (clusters and specialized resources)
  - Public resources (public and community clouds)
- Offer compute, storage and network services

UPPSALA
UNIVERSITET

Cluster computing

Dedicated setup designed according to the local needs



https://www.educba.com/what-is-cluster-computing/

UPPSALA
UNIVERSITET

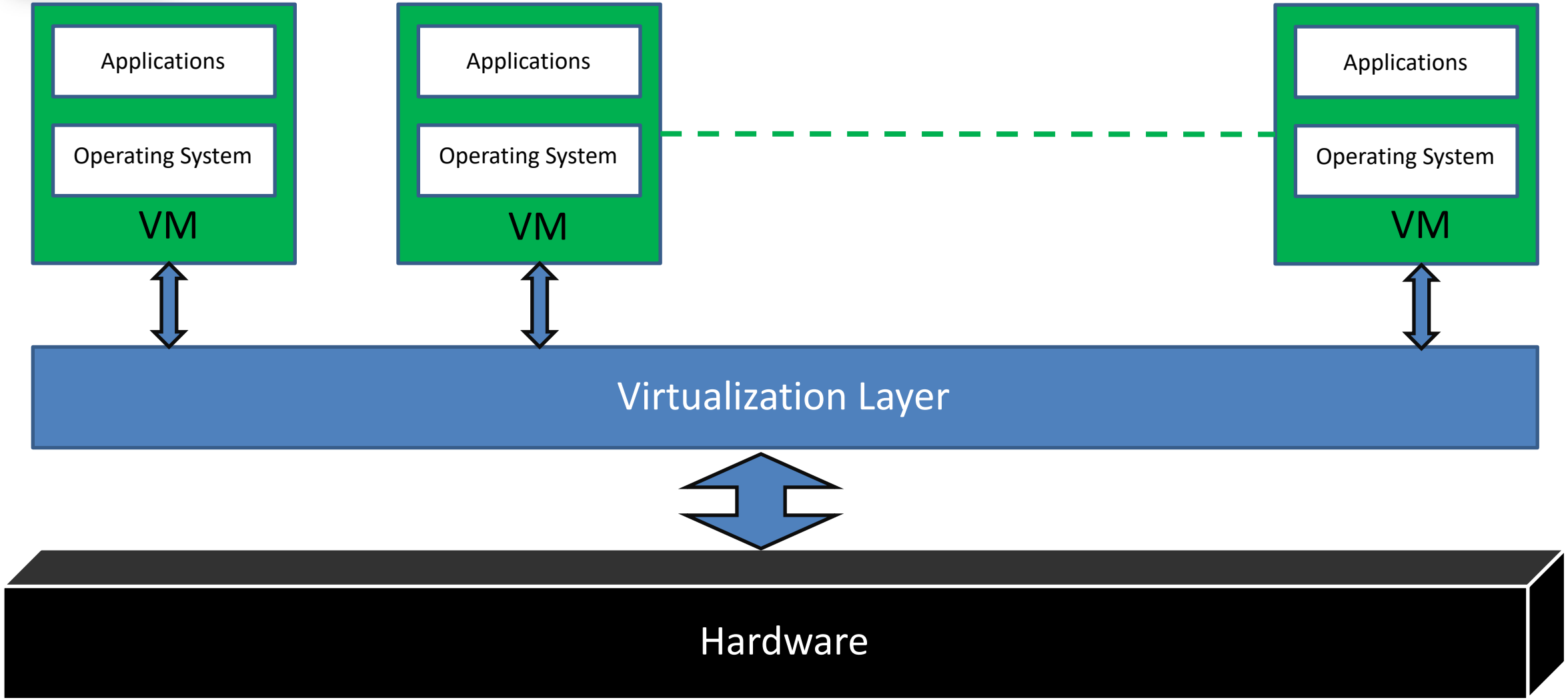
Cloud computing

Large-scale computing both for regular and specialized applications

https://csrc.nist.gov/publications/detail/sp/800-145/final

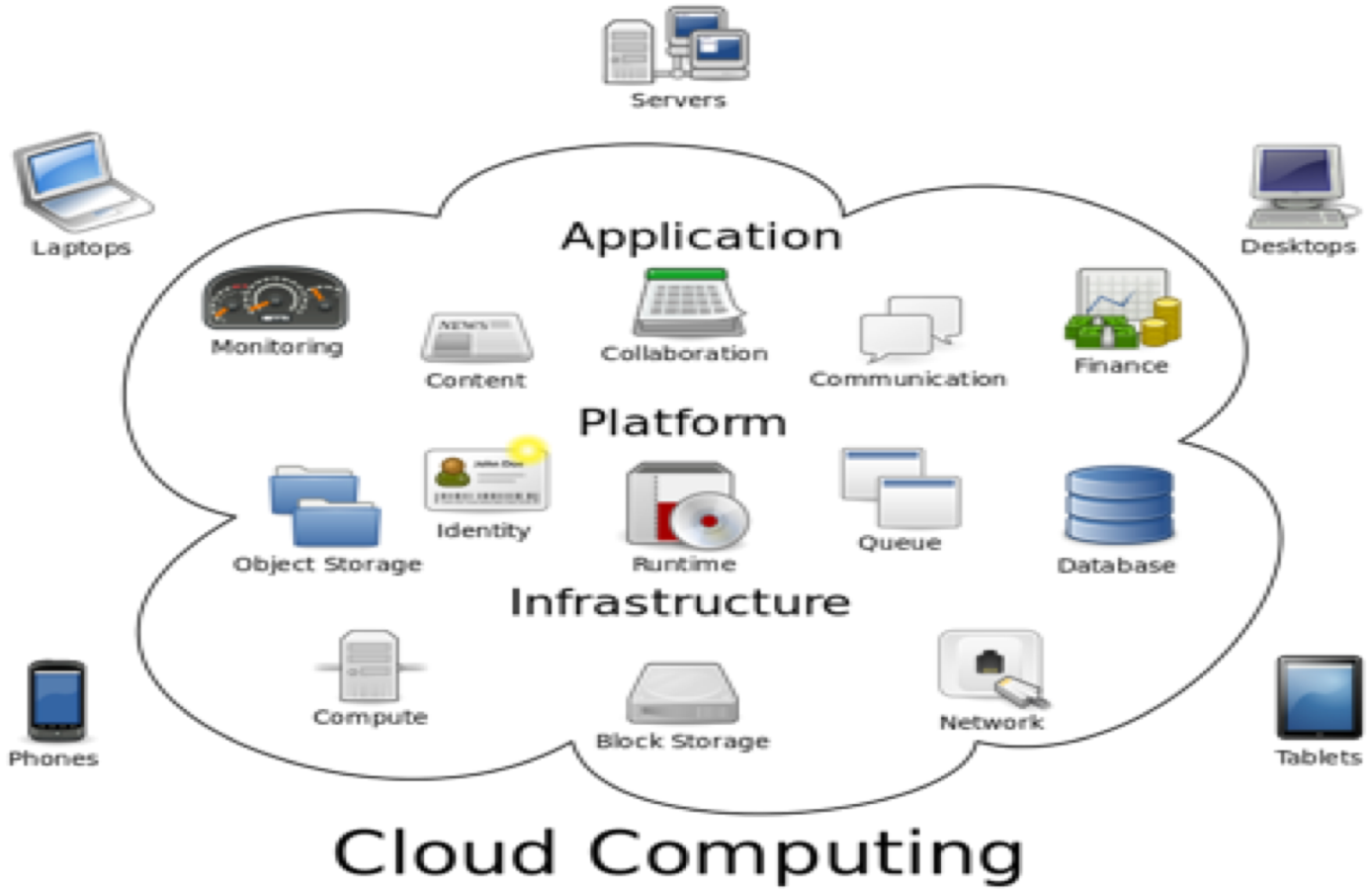
UPPSALA
UNIVERSITET

Basic design of e-infrastructures



UPPSALA
UNIVERSITET

What is Cloud Computing?

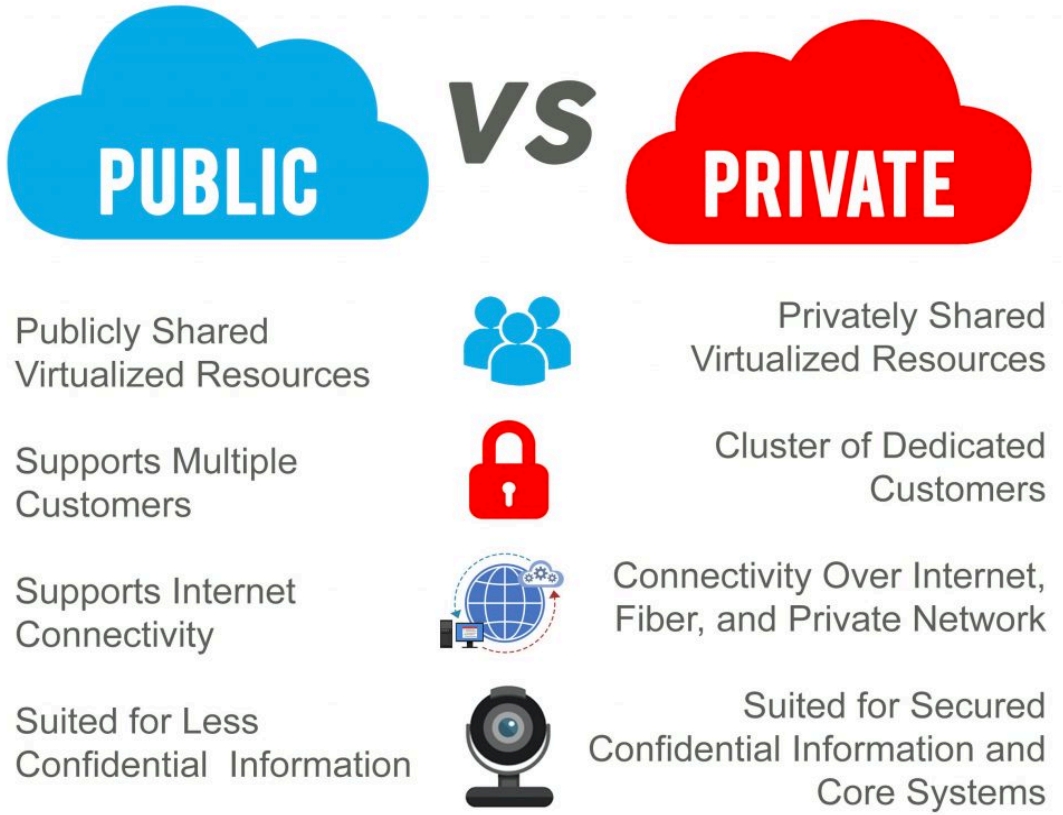


IT-infrastructure and services
“obscured by a a cloud”

UPPSALA UNIVERSITET

Deployment Models

- Public
- Private
- Community
- Hybrid



https://cloud.illinois.edu/types-of-cloud-computing-private-public-and-hybrid-clouds/

UPPSALA
UNIVERSITET

Features of current e-infrastructures

- On-demand availability of resources
- Pay-as-you-go model
- A platform with a range of services and tools
- Service level agreements (SLAs)

UPPSALA
UNIVERSITET

Cloud computing

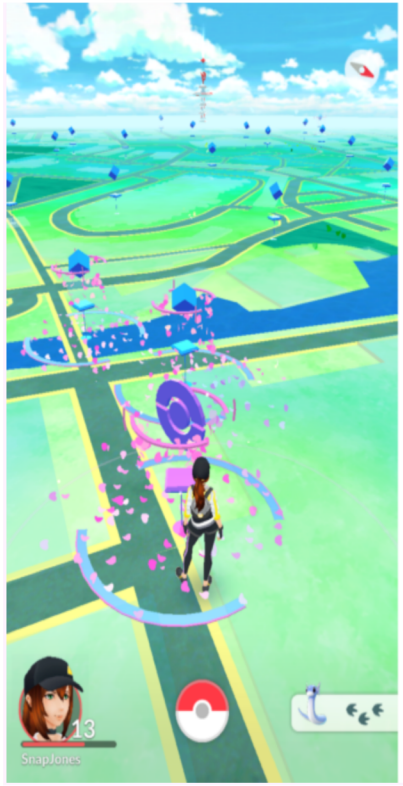
- A well-defined economic model
- Complete isolation, direct access and full control of allocated resources
- On demand resource allocation, No job queues!
- Loose coupling with the underlying resources
- “Standard” interface to interact with the cloud resources
- Orchestration of scalable services
- Minimal interaction with service providers

Are legacy applications portable to clouds?

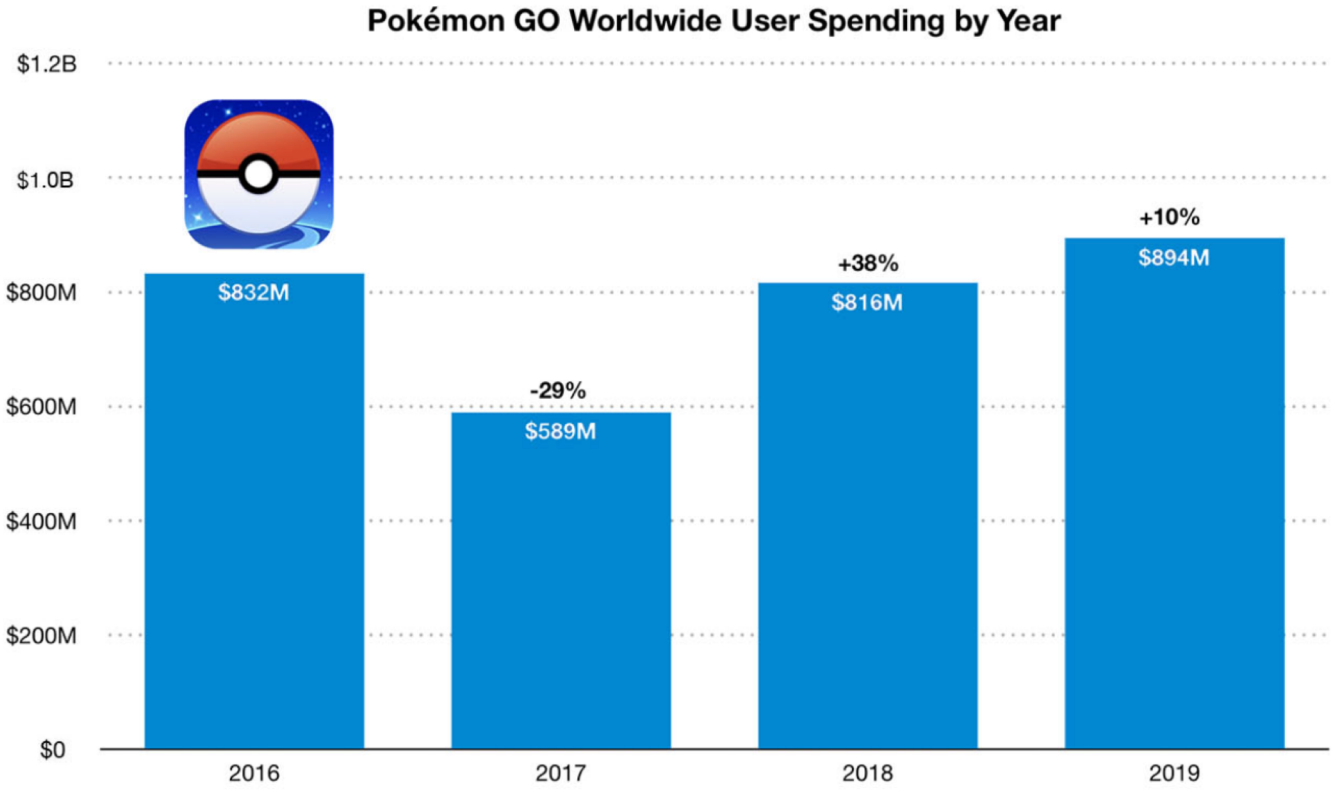
ANSWER: Yes

UPPSALA
UNIVERSITET

Small companies



Pokémon Go



https://arinsider.co/2020/01/20/2019-was-pokemon-gos-best-revenue-year-yet/

UPPSALA UNIVERSITET

Medium and large organizations

- Dropbox
- Google Apps (Gmail, Calendar, Drive …)
- Netflix
- Uber Cab Service
- Spotify
- etc.

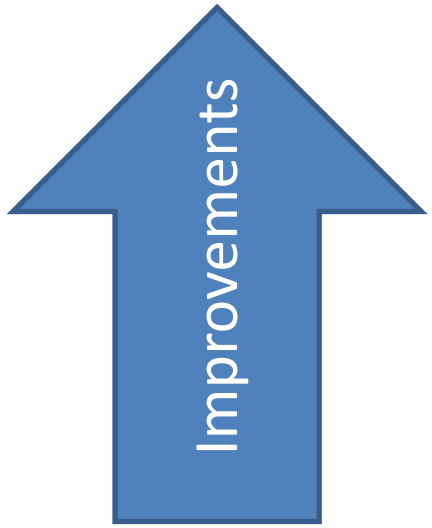
UPPSALA UNIVERSITET

Progress in last two decades

Cloud systems (Service-oriented, Agile software stack)

Level of abstraction

- Responsibilities
- Roles
- Business model
- …



Technological advancements

- Virtualization
- Communication tools
- Software designs
- …

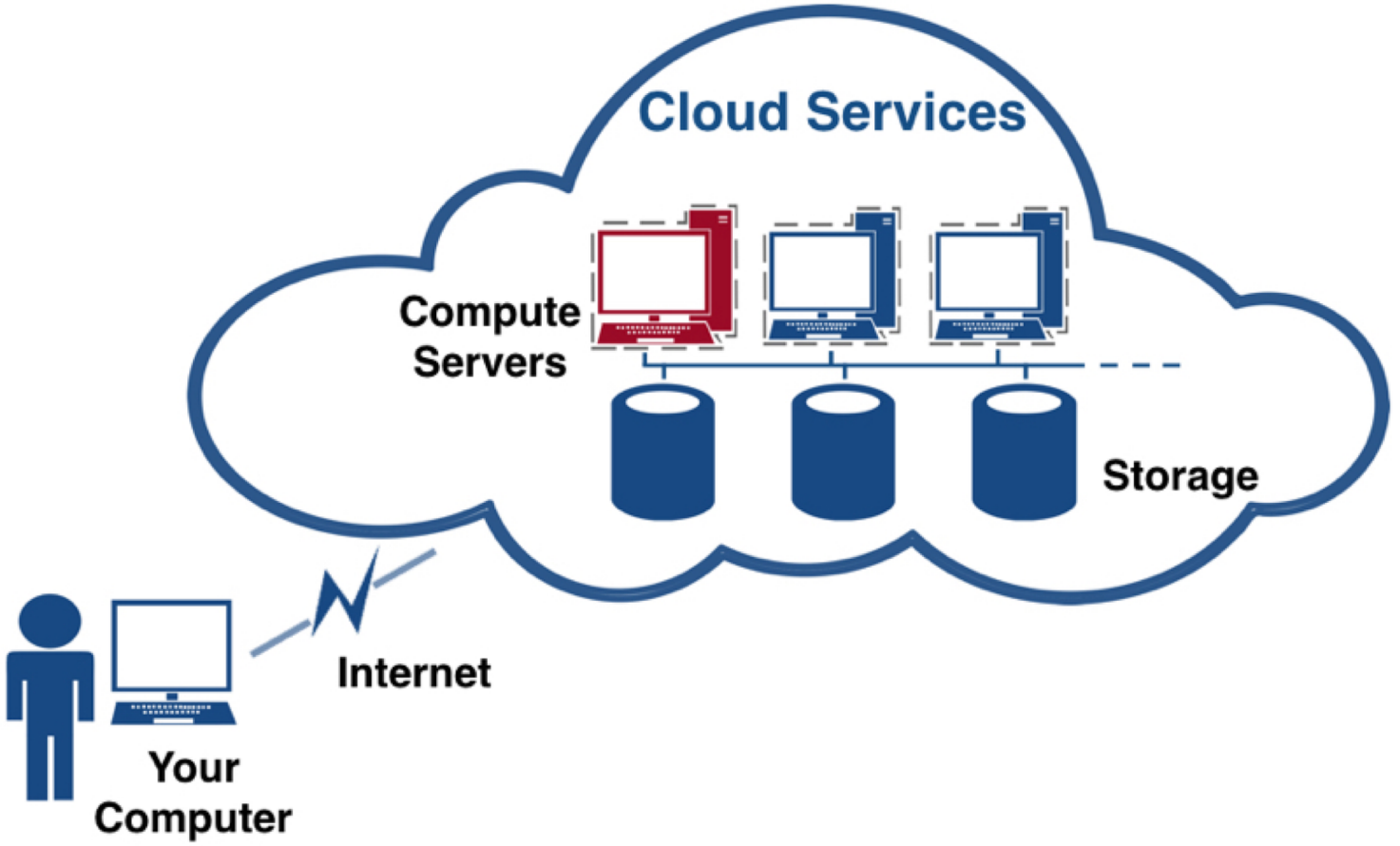
Dedicated clusters (Monolithic design)

UPPSALA
UNIVERSITET

Cloud 1.0
Raw virtual resources

UPPSALA
UNIVERSITET

Cloud 1.0



UPPSALA
UNIVERSITET

Cloud 2.0
Platforms and applications

UPPSALA
UNIVERSITET

Cloud 2.0

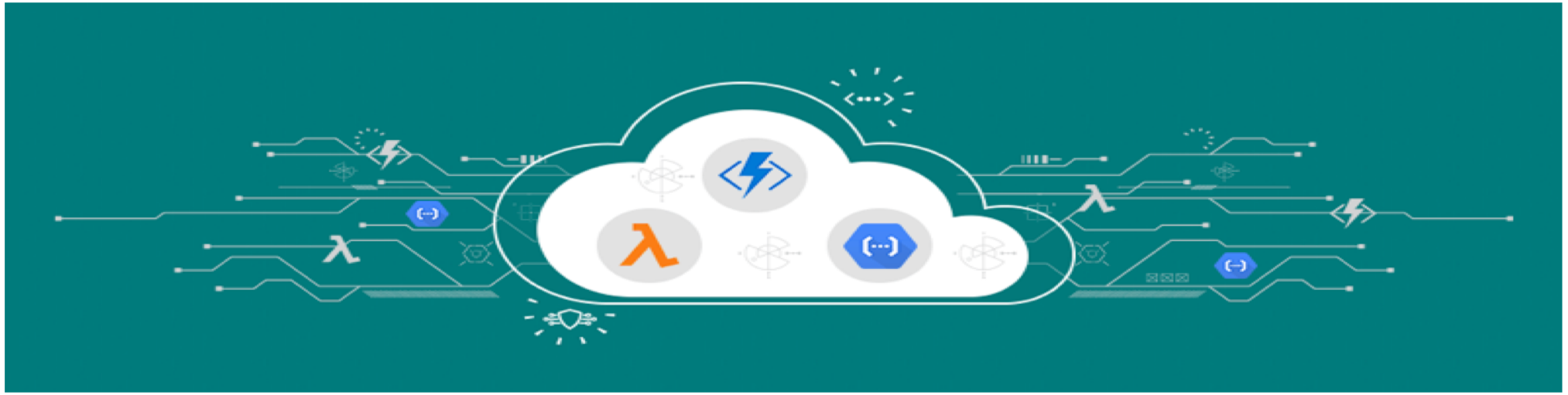
UPPSALA
UNIVERSITET

Cloud 3.0
Serverless architecture and AI

UPPSALA
UNIVERSITET

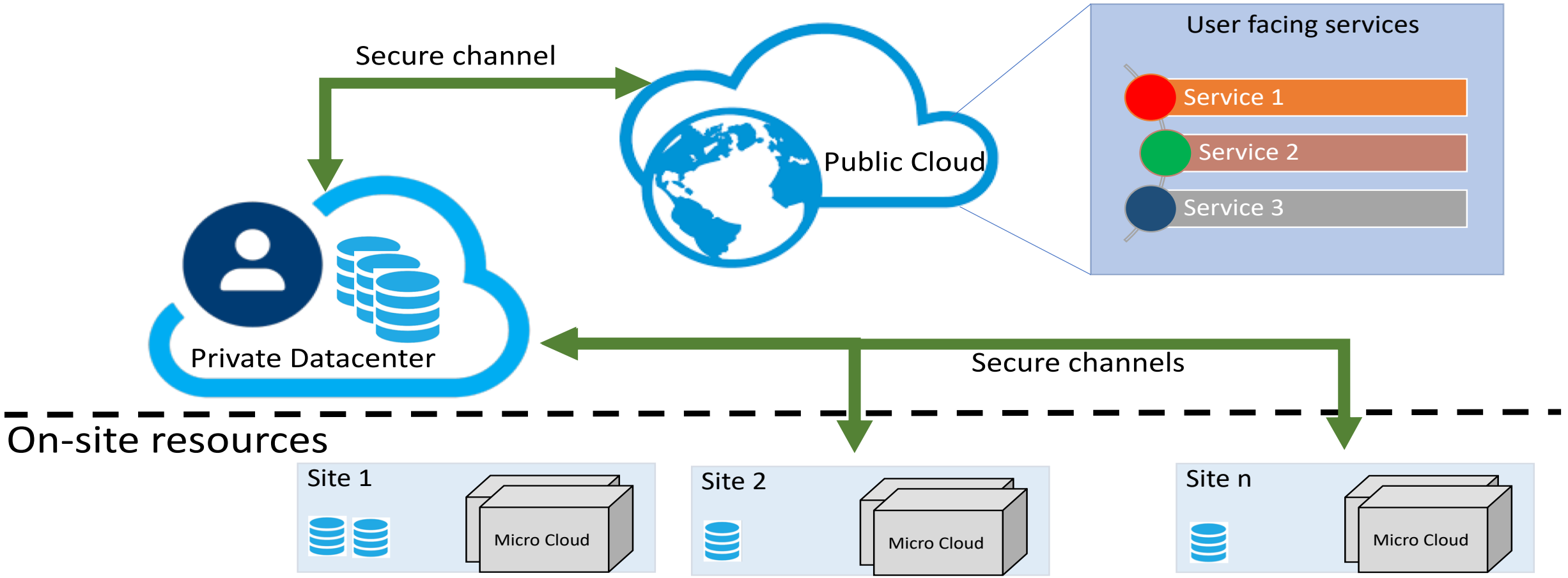
Cloud 3.0
Serverless architecture and smart services

- Amazon Lambda
- Google Cloud Functions
- Azure Functions
- OpenFaaS



UPPSALA UNIVERSITET

Hybrid infrastructure environment



UPPSALA
UNIVERSITET

Challenges for Application Experts and Service Providers

UPPSALA UNIVERSITET

Challenges

- Application design
- Vendor agnostic design

Target → Application Experts

- Data management
- Security and privacy
- Service level agreements
- High availability
- Efficient utilization of resources

Target → Application Experts and Service Providers

UPPSALA
UNIVERSITET

Virtualization

- Nutshell: The abstraction of available resources

- Definition:

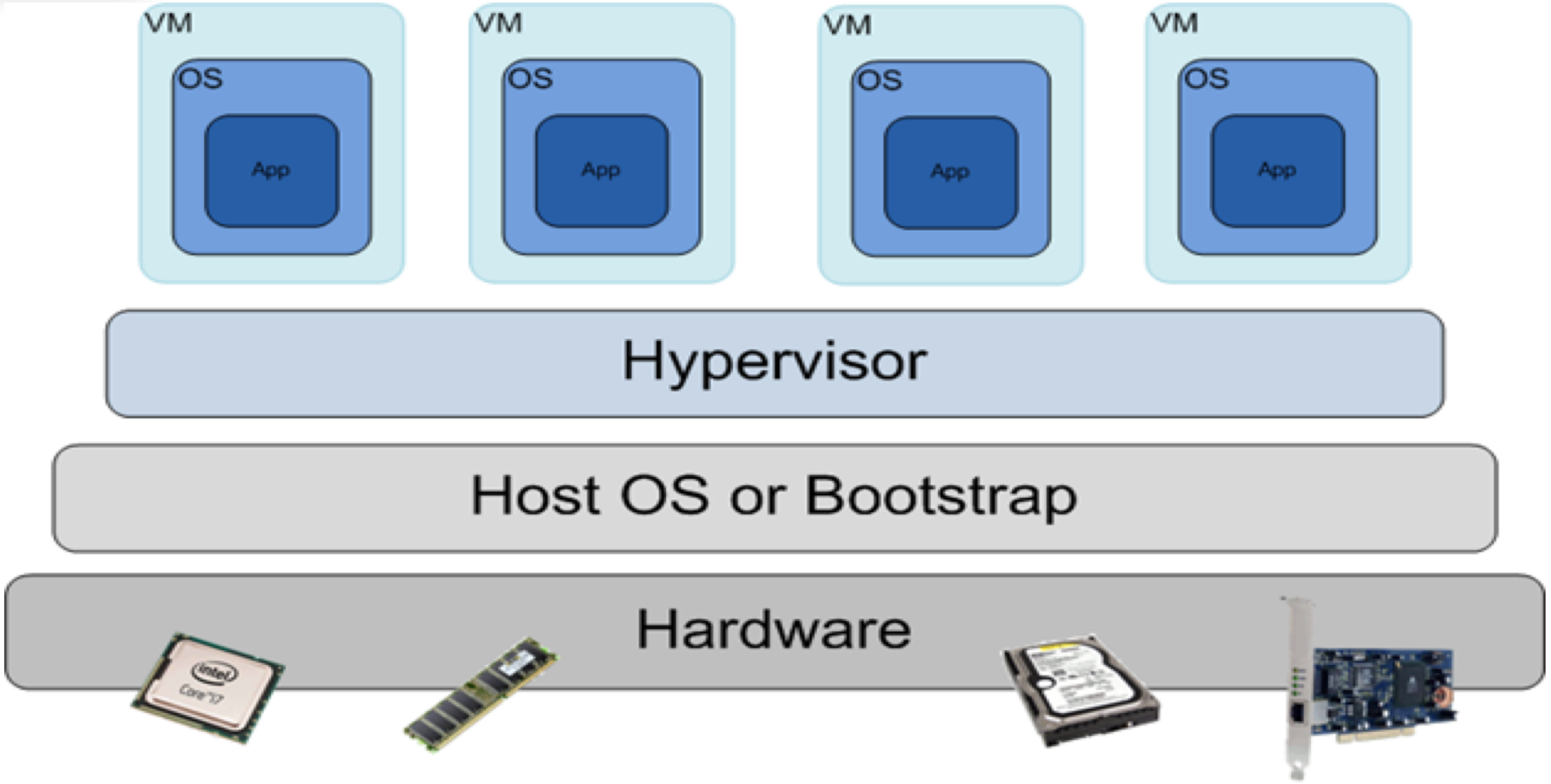
Virtualization technologies encompass a variety of mechanisms and techniques used to decouple the architecture and user-perceived behavior of hardware and software resources from their physical implementation.

- Whereas, resources can be either compute, storage, network..etc

http://www.computer.org/csdl/mags/co/2005/05/r5028.html

UPPSALA
UNIVERSITET

Virtualization
Basic illustration



http://blog.cloudpassage.com/2011/07/22/the-difference-between-virtualization-and-cloud-computing/

UPPSALA
UNIVERSITET

Beyond virtual machines

UPPSALA
UNIVERSITET

Containers

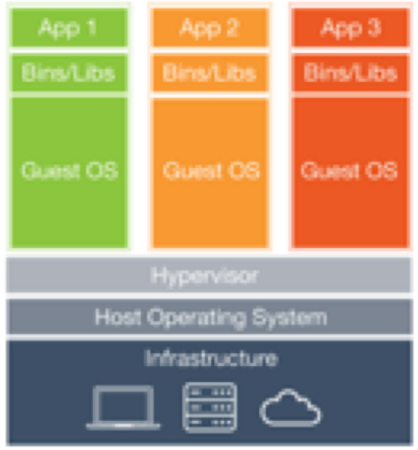
- OS level virtualization environment
- Kernel space is shared
- User space is separate for each Linux system (container)
- A lightweight alternative to Virtual Machines (VM)
- Shared same resources as host OS
- A simple model for packaging applications in Linux

https://linuxcontainers.org/

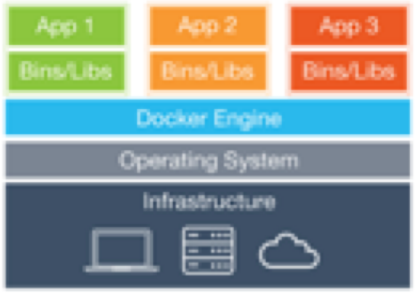
UPPSALA
UNIVERSITET

Docker

- Docker package an application together with all its dependencies in the container
- Guarantees that it will always run the same regardless of the environment
- Container based orchestration tool
- Docker Hub, container registry
- Open source



Virtual Machines



Containers

UPPSALA
UNIVERSITET

Images and formats

- Cloud images are customized disks images of OSes for private or public clouds

- Different formats are available:
  - raw: An unstructured disk image format (big in size)
  - vhd: VMware, Xen, Microsoft, VirtualBox, and others
  - vdi: Supported by VirtualBox, QEMU emulator.
  - iso: Archive format for the data contents of an optical disc
  - qcow2: Supported by the QEMU emulator that can expand dynamically and supports Copy on Write.
  - …

UPPSALA
UNIVERSITET

Contextualization

In cloud computing contextualization means providing customized computing environment

Or

Allows a virtual machine instance to learn about its cloud environment and user requirement (the ‘context’) and configure itself to run correctly

http://www.journalofcloudcomputing.com/content/4/1/17

UPPSALA
UNIVERSITET

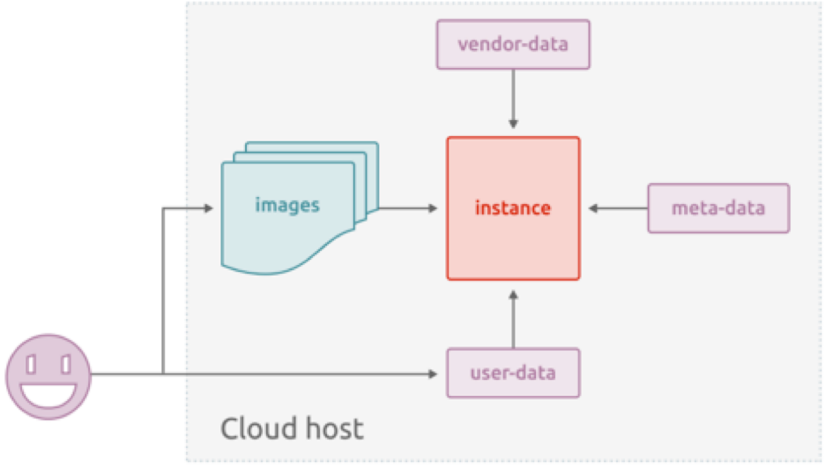
Contextualization

- Provide scalable solution
- No need to manage fat images
- Dynamic configuration
- Typically work in two layers
  - Meta-data : System information handled at cloud level
  - User-defined-data: User specific requirements/settings

UPPSALA UNIVERSITET

CloudInit

Cloud-init is the industry standard multi-distribution method for cross-platform cloud instance initialization. It is supported across all major public cloud providers, provisioning systems for private cloud infrastructure, and bare-metal installations.



https://cloudinit.readthedocs.io/en/latest/
https://www.lucd.info/2019/12/06/cloud-init-part-1-the-basics/

UPPSALA
UNIVERSITET

CloudInit

```
#cloud-config
apt_update: true
apt_upgrade: true
packages:
 - cowsay
 - python-pip
 - python-dev
 - build-essential
 - cowsay
byobu_default: system

runcmd:
 - echo "export PATH=$PATH:/usr/games" >> /home/ubuntu/.bashrc
 - source /home/ubuntu/.bashrc
 - pip install Flask
 - python /home/ubuntu/cowsay-app.py &
```

UPPSALA
UNIVERSITET

Orchestration

- Orchestration is a process of resource contextualization based on the automation available in the cloud systems.

- A process required for
  - rapid application deployment
  - scalability
  - management
  - high availability
  - agility

- Essential for large complex applications

- A process at the level of Platform as a Service (PaaS)

UPPSALA
UNIVERSITET

Orchestration

- Available tools

- Docker Compose https://www.docker.com/
- Kubernetes http://kubernetes.io/
- Ansible https://www.ansible.com/
- Heat https://wiki.openstack.org/wiki/Heat

UPPSALA
UNIVERSITET

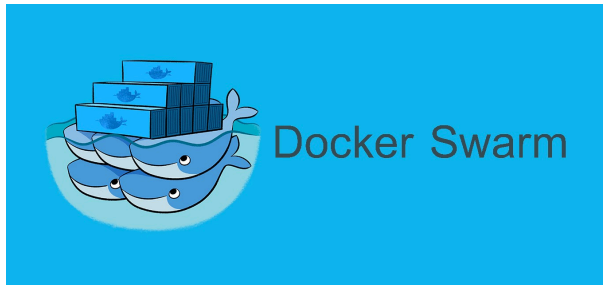
Containers

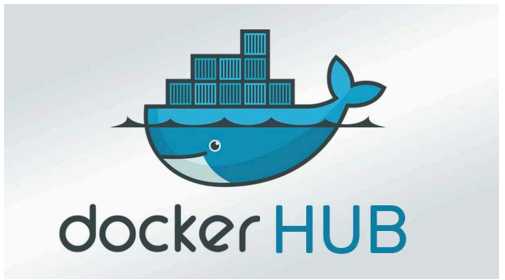
- OS level virtualization environment
  - Kernelspace is shared
  - Userspace is separate for each Linux system (container)
- A lightweight alternative to Virtual Machines (VM)
- Shared same resources as host OS
- A simple model for packaging applications in Linux.

UPPSALA
UNIVERSITET

Docker

- Docker package an application together with all its dependencies in the container
- Guarantees that it will always run the same regardless of the environment
- Container based orchestration tool
- Docker Hub, container registry
- Open source





UPPSALA UNIVERSITET

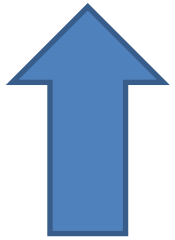
Infrastructure as Code (IaC)

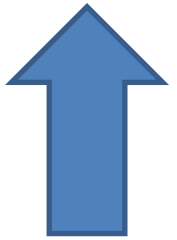
Infrastructure as code, also referred to as IaC, is an IT practice that codifies and **manages underlying IT infrastructure as software**. The purpose of infrastructure as code is to enable developers or operations teams to automatically **manage, monitor and provision resources**, rather than manually configure discrete hardware devices and operating systems.

Infrastructure resources	Operating systems and Host configurations	Application configuration
Compute	Linux	Application dependencies
Storage	Windows	Application configurations
Network	Access services	Data access control
High level services	Host level security	Application level security
		Application access

UPPSALA
UNIVERSITET

Infrastructure as Code (IaC)





Contextualization

Orchestration

UPPSALA
UNIVERSITET

Kubernetes

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

https://kubernetes.io/

UPPSALA
UNIVERSITET

Kubernetes

- Features
  - Automation
  - Scheduling
  - Self healing capabilities
  - Horizontal scaling
  - Load balancing
  - Environment consistency
  - Application centric management
  - Built-in services
  - Effective management of credentials
  - Application centric access control

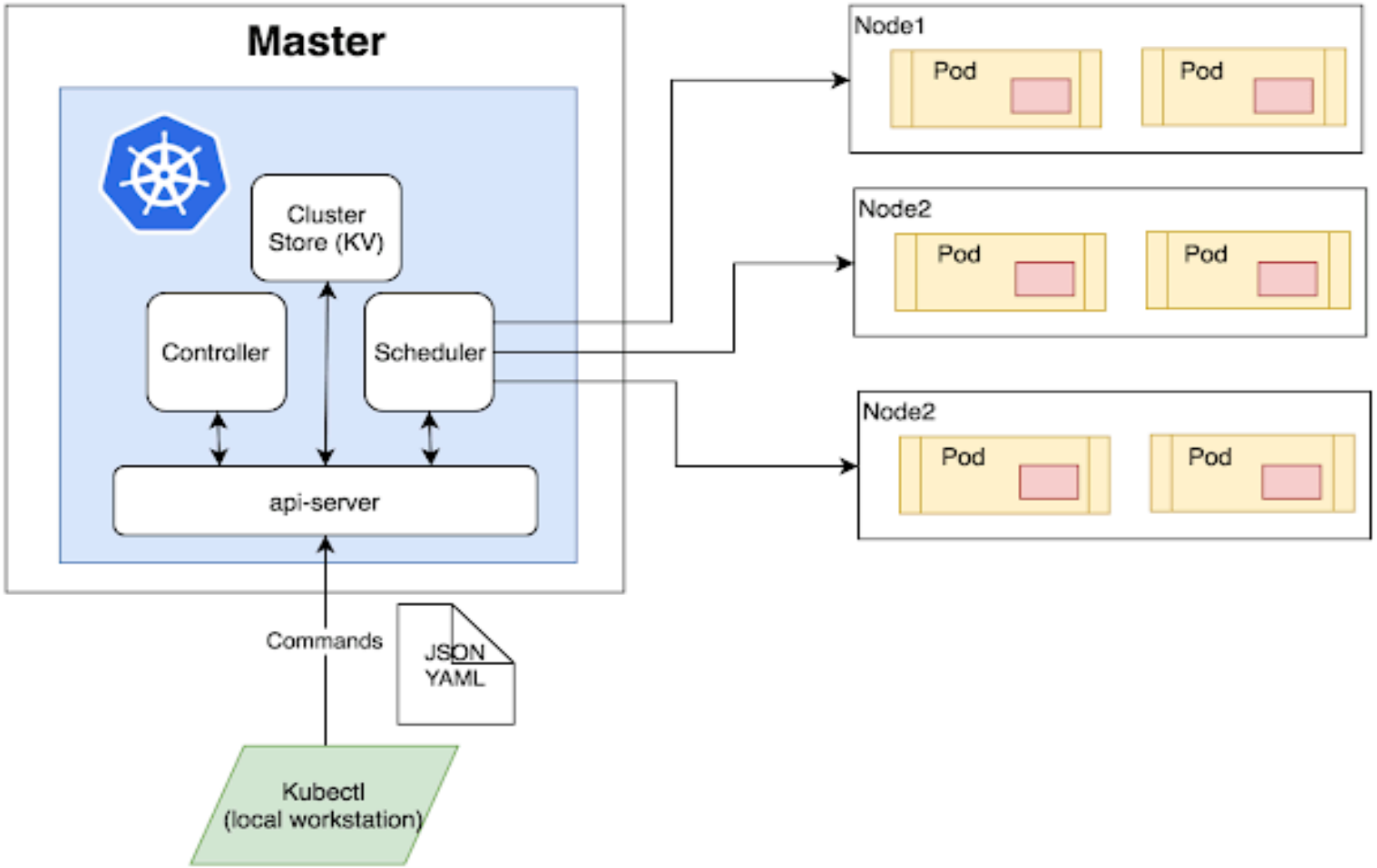
UPPSALA
UNIVERSITET

Kubernetes

- Basic concepts
  - **Cluster:** A collection of hosts aggregating their resources (CPU, RAM, Storage and Network)
  - **Master:** The main node of the cluster, forms control panel of Kubernetes
  - **Node:** A single host in the cluster, running essential services
  - **Namespaces:** A logical organization of the resources

UPPSALA
UNIVERSITET

Kubernetes Architecture



UPPSALA
UNIVERSITET

Kubernetes

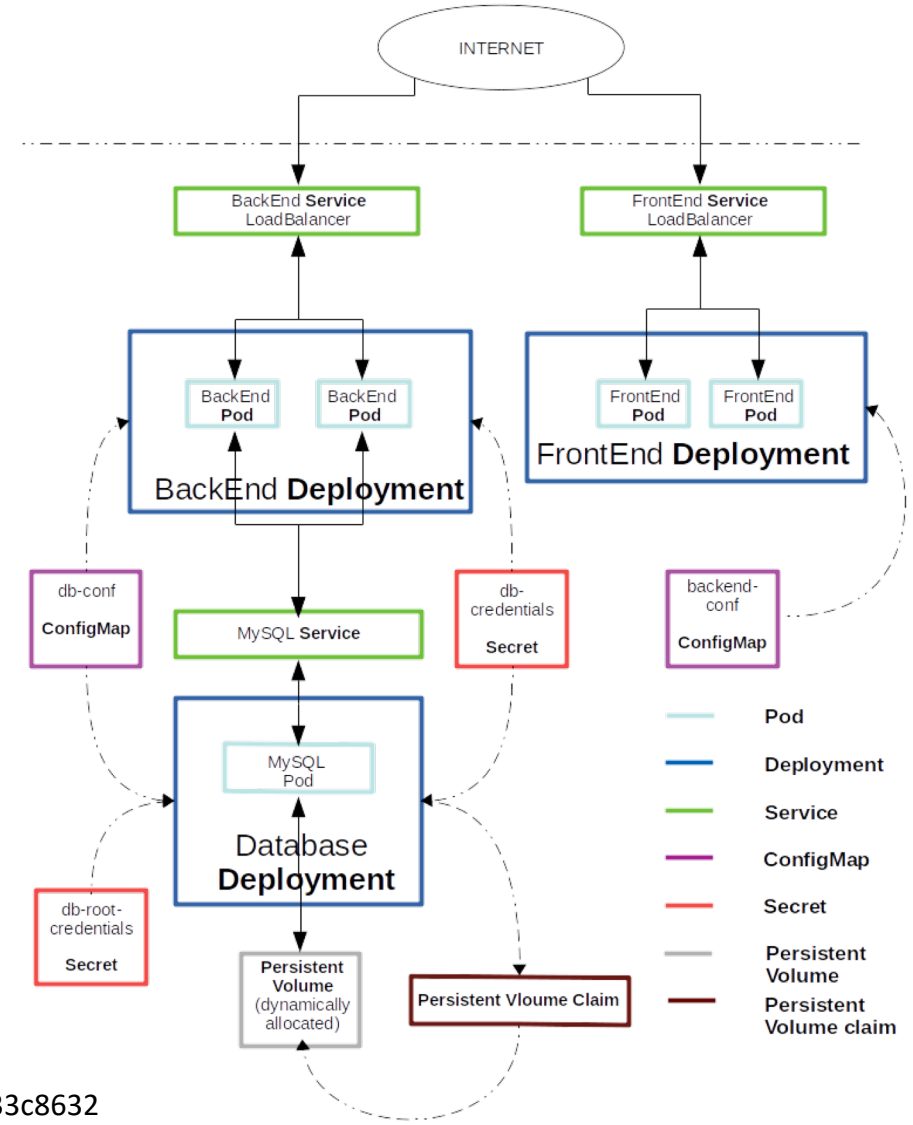
- Selected list of K8s objects
  - **Pods:** A group of one or more containers.
  - **Service:** An abstraction which defines a logical set of pods and a policy to access them. Every Kubernetes service has an IP address, but unlike the IP address of a pod, it is stable.
  - **Persistent-Volume and Persistent-Volume Claim:** A persistent-volume is a piece of storage in the cluster that has been provisioned to be used. Whereas a persistent-volume claim is a request by an application to consume the abstract storage resources declared through persistent volume.
  - **ConfigMap:** Another abstraction to decouple environment-dependent application-configuration-data from containerized applications.
  - **Secrets:** An object that contains a small amount of sensitive data such as a password, a token, or a key.
  - **Deployment:** An abstraction to represent the desired state of an actual deployment on Kubernetes

UPPSALA UNIVERSITET

Kubernetes Example

- Web application:
  - Database (MySQL server)
  - Backend (Java Spring Boot application)
  - Frontend (Angular app)

- Configuration settings:
  - Database, one replica
  - Backend, two replicas
  - Frontend, two replicas



https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

UPPSALA UNIVERSITET

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

Dockerfile

Frontend-dockerfile

```
FROM node:7.7-alpine

# install dependencies
ADD package.json /tmp/package.json
RUN cd /tmp && npm install

# Copy dependencies
RUN mkdir -p /opt/to-do-app && cp -a /tmp/node_modules /opt/to-do-app

# Setup workdir
WORKDIR /opt/to-do-app

RUN mkdir /opt/to-do-app/dist
COPY dist /opt/to-do-app/dist
COPY server.js /opt/to-do-app
COPY initialize.js /opt/to-do-app

# run
EXPOSE 8080
CMD ["sh", "-c", "node initialize && node server"]
```

Backend-dockerfile

```
FROM openjdk:10-jre-slim
RUN mkdir -p /opt/to-do-app/
COPY ./target/to-do-listEntity-app-0.0.1-SNAPSHOT.jar /opt/to-do-app/
WORKDIR /opt/to-do-app/
EXPOSE 8080
CMD ["java", "-jar", "to-do-listEntity-app-0.0.1-SNAPSHOT.jar"]
```

Upload the containers to dockerHub

```
# docker push image-name
```

UPPSALA
UNIVERSITET

Kubernetes

- Installation
  - Kubernetes: https://kubernetes.io
  - MiniKube: https://minikube.sigs.k8s.io/docs/start/
  - MicroK8s: https://microk8s.io/

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

UPPSALA UNIVERSITET

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

Kubernetes

backend-spring-application.yaml

```
spring:
  datasource:
    type: com.zaxxer.hikari.HikariDataSource
    hikari:
      idle-timeout: 10000
    platform: mysql
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
    url: jdbc:mysql://${DB_HOST}/${DB_NAME}
  jpa:
    hibernate:
      naming:
        physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

mysql-configmap.yaml

```
# Define 'Configmap' to store non-sensitive database configuration
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-conf  # name of ConfigMap, referenced in other files
data:
  host: mysql   # host address of mysql server, we are using DNS of Service
  name: to-do-app-db # name of the database for application
```

db-root-credentials-secret.yaml

```
# Define 'Secret' to store db-credentials
apiVersion: v1
kind: Secret
metadata:
  name: db-credentials # Name of the 'Secret'
data:
  username: dXNlcg==  # base64 encoded 'Secret' username
  password: ZGV2ZWxvcA== # base64 encoded 'Secret' password
```

db-root-credentials-secret.yaml

```
# Define 'Secret' to store 'root' user Credentials
apiVersion: v1
kind: Secret
metadata:
  name: db-root-credentials # Name of the Secret
data:
  password: bWFnaWM= # base64 encoded 'root' password
```

```
# kubectl apply -f <FILE_NAME>
```

UPPSALA
UNIVERSITET

Kubernetes

backend-spring-application.yaml

```
# Define a 'Service' To Expose mysql to Other Services
apiVersion: v1
kind: Service
metadata:
  name: mysql  # DNS name
  labels:
    app: mysql
    tier: database
spec:
  ports:
    - port: 3306
      targetPort: 3306
  selector:       # mysql Pod Should contain same labels
    app: mysql
    tier: database
  clusterIP: None  # We Use DNS, Thus ClusterIP is not relevant
---
# Define a 'Persistent Voulume Claim'(PVC) for Mysql Storage, dynamically provisioned by cluster
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim # name of PVC essential for identifying the storage data
  labels:
    app: mysql
    tier: database
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
---
```

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

UPPSALA
UNIVERSITET

Kubernetes

```
# Configure 'Deployment' of mysql server
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql
    tier: database
spec:
  selector: # mysql Pod Should contain same labels
    matchLabels:
      app: mysql
      tier: database
  strategy:
    type: Recreate
  template:
    metadata:
      labels: # Must match 'Service' and 'Deployment' selectors
        app: mysql
        tier: database
    spec:
      containers:
      - image: mysql:5.7 # image from docker-hub
        args:
          - "--ignore-db-dir=lost+found" # Workaround for https://github.com/docker-library/mysql/issues/186
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD # Setting Root Password of mysql From a 'Secret'
          valueFrom:
            secretKeyRef:
              name: db-root-credentials # Name of the 'Secret'
              key: password   # 'key' inside the Secret which contains required 'value'
        - name: MYSQL_USER # Setting USER username on mysql From a 'Secret'
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: username
```

UPPSALA
UNIVERSITET

Kubernetes

```
- name: MYSQL_PASSWORD # Setting USER Password on mysql From a 'Secret'
        valueFrom:
          secretKeyRef:
            name: db-credentials
            key: password
      - name: MYSQL_DATABASE # Setting Database Name from a 'ConfigMap'
        valueFrom:
          configMapKeyRef:
            name: db-conf
            key: name
      ports:
      - containerPort: 3306
        name: mysql
      volumeMounts:        # Mounting voulume obtained from Persistent Volume Claim
      - name: mysql-persistent-storage
        mountPath: /var/lib/mysql
    volumes:
    - name: mysql-persistent-storage # Obtaining 'vloume' from PVC
      persistentVolumeClaim:
        claimName: mysql-pv-claim
```

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

UPPSALA UNIVERSITET

backend-deployment.yaml

```
 # Define 'Service' to expose backend application deployment
apiVersion: v1
kind: Service
metadata:
  name: to-do-app-backend
spec:
  selector:  # backend application pod lables should match these
    app: to-do-app
    tier: backend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 8080
  type: LoadBalancer   # use NodePort, if you are not running Kubernetes on cloud
---
# Configure 'Deployment' of backend application
apiVersion: apps/v1
kind: Deployment
metadata:
  name: to-do-app-backend
  labels:
    app: to-do-app
    tier: backend
spec:
  replicas: 2    # Number of replicas of back-end application to be deployed
  selector:
    matchLabels: # backend application pod labels should match these
      app: to-do-app
      tier: backend
```

```
template:
  metadata:
    labels: # Must macth 'Service' and 'Deployment' labels
      app: to-do-app
      tier: backend
  spec:
    containers:
    - name: to-do-app-backend
      image: kubernetesdemo/to-do-app-backend   # docker image of backend application
      env:   # Setting Enviornmental Variables
        - name: DB_HOST   # Setting Database host address from configMap
          valueFrom:
            configMapKeyRef:
              name: db-conf  # name of configMap
              key: host
        - name: DB_NAME  # Setting Database name from configMap
          valueFrom:
            configMapKeyRef:
              name: db-conf
              key: name
        - name: DB_USERNAME  # Setting Database username from Secret
          valueFrom:
            secretKeyRef:
              name: db-credentials # Secret Name
              key: username
        - name: DB_PASSWORD # Setting Database password from Secret
          valueFrom:
            secretKeyRef:
              name: db-credentials
              key: password
      ports:
      - containerPort: 8080
```

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

UPPSALA UNIVERSITET

https://betterprogramming.pub/kubernetes-a-detailed-example-of-deployment-of-a-stateful-application-de3de33c8632

backend-configMap.yaml

```
# ConfigMap to expose configuration related to backend application
apiVersion: v1
kind: ConfigMap
metadata:
  name: backend-conf # name of configMap
data:
  server-uri: 34.66.207.42 # enternal ip of backend application 'Service'
```

frontend-deployment.yaml

```
# Define 'Service' to expose FrontEnd Application
apiVersion: v1
kind: Service
metadata:
  name: to-do-app-frontend
spec:
  selector: # pod labels should match these
    app: to-do-app
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 8080
  type: LoadBalancer # use NodePort if you are not running Kubernetes on Cloud
---
```

```
# 'Deployment' to manage of configuration of frontEnd Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: to-do-app-frontend
  labels: # pod labels should match these
    app: to-do-app
    tier: frontend
spec:
  replicas: 2  # number of replicas of frontEnd application
  selector:
    matchLabels:
      app: to-do-app
      tier: frontend
  template:
    metadata:
      labels: # Must match 'Service' and 'Deployment' labels
        app: to-do-app
        tier: frontend
    spec:
      containers:
      - name: to-do-app-frontend
        image: kubernetesdemo/to-do-app-frontend # docker image of frontend application
        env: # Setting Environmental Variables
          -  name: SERVER_URI # Setting Backend URI from configMap
             valueFrom:
               configMapKeyRef:
                 name: backend-conf # Name of configMap
                 key: server-uri
        ports:
        - containerPort: 8080
```

UPPSALA
UNIVERSITET

Complete K8s online course

YouTube Link:
https://www.youtube.com/watch?v=X48VuDVv0do&t=1s

UPPSALA
UNIVERSITET

Application-as-a-Package

UPPSALA
UNIVERSITET

Helm charts

- A package manager for Kubernetes
- Helm chart can contain any number of Kubernetes objects.
- Helm chart usually contains:
  - deployment
  - service
  - Ingress
  - Persistent Volume Claims
  - or any other Kubernetes object

https://helm.sh

UPPSALA
UNIVERSITET

Serverless architecture

Serverless architecture is a concept to build and run software and services without having to manage servers. In serverless architecture, the server will manage by provider and not by consumer.

https://blog.runcloud.io/understand-serverless-computing/

https://link.springer.com/chapter/10.1007/978-981-10-5026-8_1

UPPSALA
UNIVERSITET

FaaS: Function as a Service

- FaaS is a realization of the serverless architecture
- FaaS allows to execute code in response to events without the complex infrastructure typically associated with building and launching microservices applications.
- FaaS consists of lightweight functions that subdivide the application's logic and only executed based on generated events.

https://www.ibm.com/cloud/learn/faas

A Preliminary Review of Enterprise Serverless Cloud Computing (Function-as-a-Service) Platforms

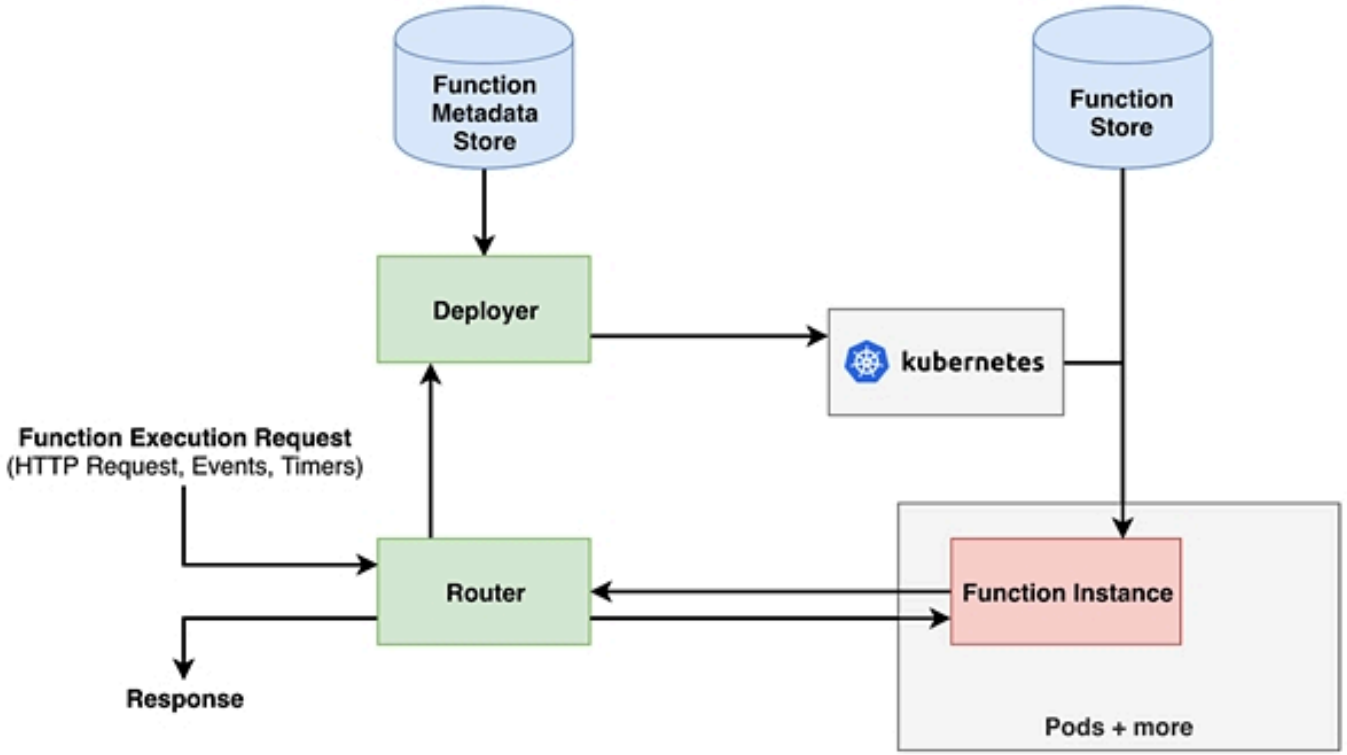
UPPSALA
UNIVERSITET

FaaS: Function as a Service

- Characteristics:
  - Rapid development
  - Event driven execution
  - Cost effective computing environment (Pay-as-you-*run*)
  - No resource selection, configuration and management
  - Better elastic behaviour

UPPSALA UNIVERSITET

FaaS: Function as a Service



The anatomy of the runtime of a FaaS platform.

https://www.infoq.com/articles/serverless-performance-cost/

UPPSALA
UNIVERSITET

FaaS: Function as a Service

- FaaS Platforms
  - Firecracker, AWS Lambda and AWS Fargate (Amazon)
  - Fission (Platform 9)
  - Fn (Oracle)
  - Knative/Kubernetes (Google)
  - OpenFaaS

https://firecracker-microvm.github.io/ https://fission.io/ https://fnproject.io/ https://knative.dev/ https://github.com/openfaas/faas