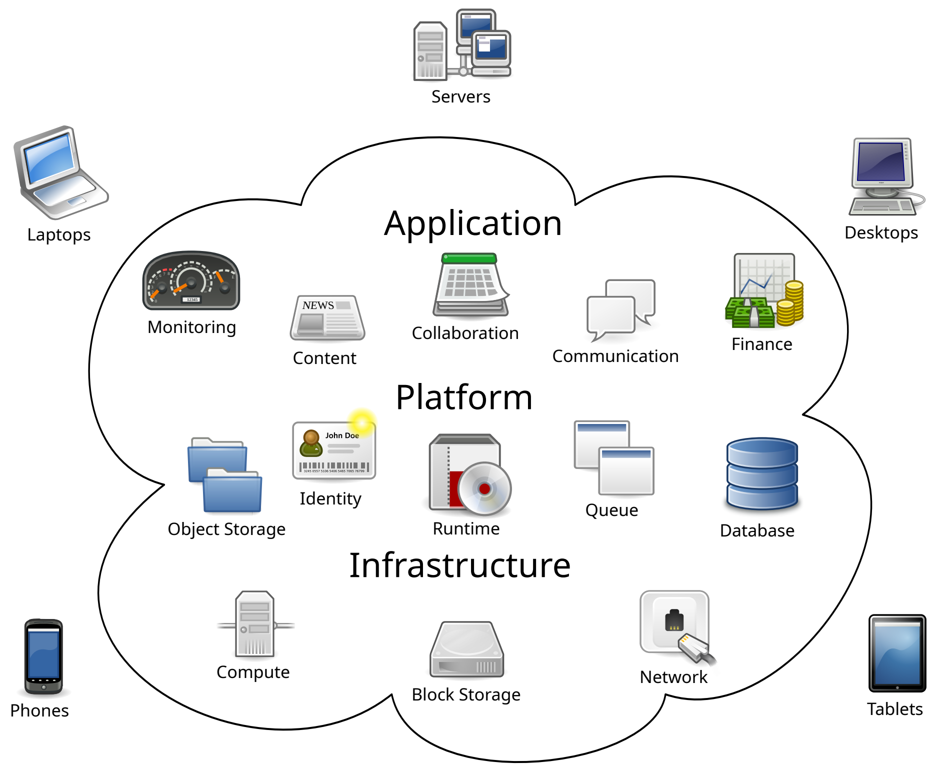
Orchestration

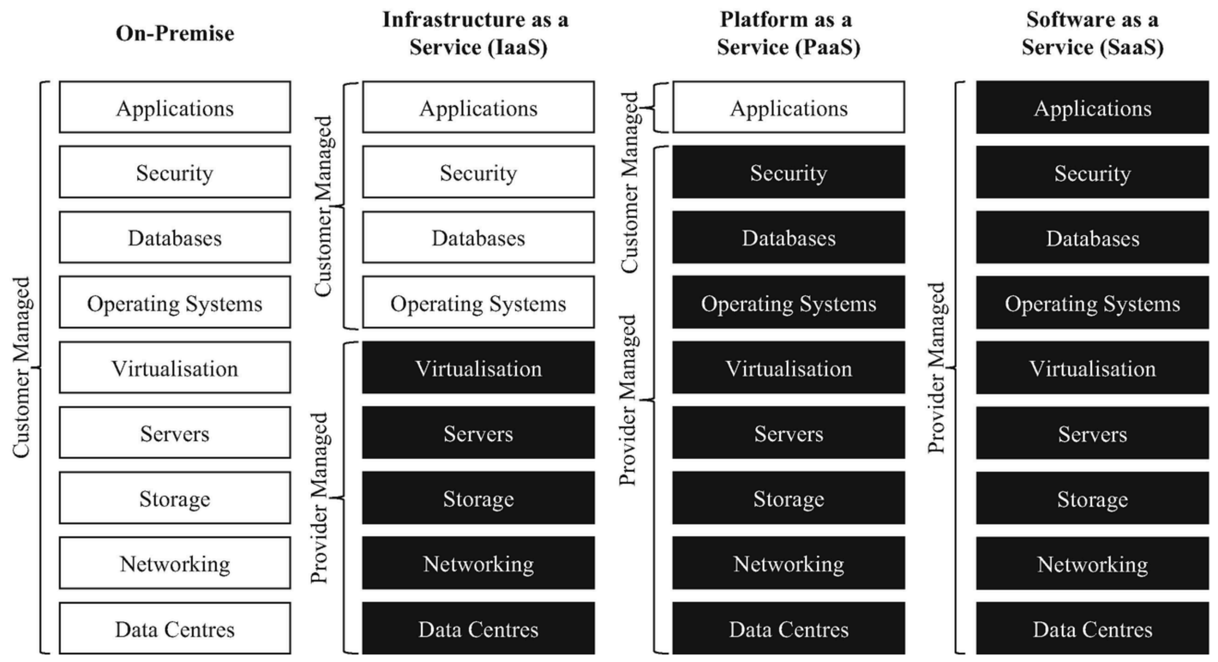
The Cloud



https://commons.wikimedia.org/wiki/File:Cloud_computing.svg

As A Service

Orchestration is necessary for On-Premise Installation, IaaS, and PaaS.



https://commons.wikimedia.org/wiki/File:Comparison_of_on-premise,_IaaS,_PaaS,_and_SaaS.png

Function as a Service (FaaS): provider runs individual functions in response to events.

Cloud Computing

> Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services.

> [Developers with innovative ideas] need not be concerned about over-provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under-provisioning for one that becomes wildly popular, thus missing potential customers and revenue.

> Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1000 servers for one hour costs no more than using one server for 1000 hour.

Michael Armbrust, et al. 2009. Above the Clouds: A Berkeley View of Cloud Computing.

When to use it

- Varying demand: traffic spikes, viral growth, seasonal patterns
- Batch processing: machine learning training
- Experimentation: spin up environments quickly and cheaply
- Startups: no upfront capital for infrastructure needed
- Global players: serve users across geographies

Job Scheduling

Stands in contrast to cloud computing.

Schedule long-running tasks onto a fixed number of machines in compute clusters and grids.

- limit resources used by individuals
- maximize overall resource utilization
- manages batch jobs in a queue

Examples: SLURM (on our university cluster), PBS, TORQUE, ...

This stands in contrast to managing a cloud of machines that continually offer services.

Virtualization

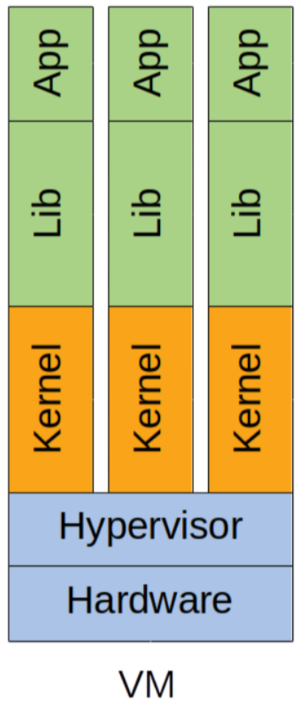
Hypervisor: runs directly on hardware and hosts multiple operating systems.

Advantage: isolation (separate kernels)

Disadvantage: heavy (size in GBs, boot time)

Examples: Xen, KVM, VMware, Hyper-V, ...

Enables Infrastructure as a Service.



Tom Goethals, et al. 2018. Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications.

Containerization

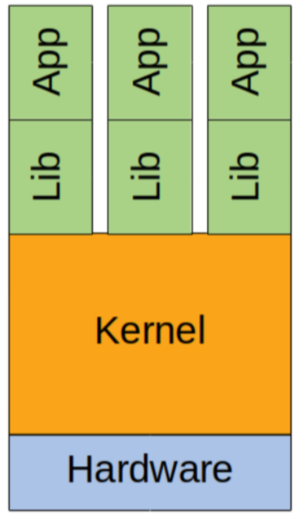
Containers: isolated processes that run on the same kernel but have their own filesystem and network, and limited resources.

Advantage: lightweight (size in MBs)

Disadvantage: weaker isolation (shared kernel)

Examples: Docker, LXC, containerd, CRI-O, ...

Enables Platform as a Service.



Container

Tom Goethals, et al. 2018. Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications.

Unikernel

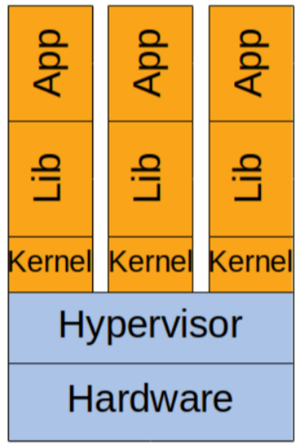
Unikernel: application compiled together with operating system components necessary to run on a hypervisor.

Advantage: very fast boot (time in ms)

Disadvantage: emerging technology

Examples: MirageOS, OSv, Unikraft, Rumprun, …

Area of active research.



Unikernel

Tom Goethals, et al. 2018. Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications.

Kubernetes (K8s)

https://kubernetes.io/

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Released by Google in 2014.

Configuration Language

Infrastructure as code.

Imperative

Lists steps to execute.

```
docker run nginx
docker scale nginx --replicas=3
docker update nginx --image=nginx:1.16
```

Declarative

Describes desired state.

```yaml
spec:
  replicas: 3
  image: nginx:1.16
```

Kubernetes Overview

Cluster

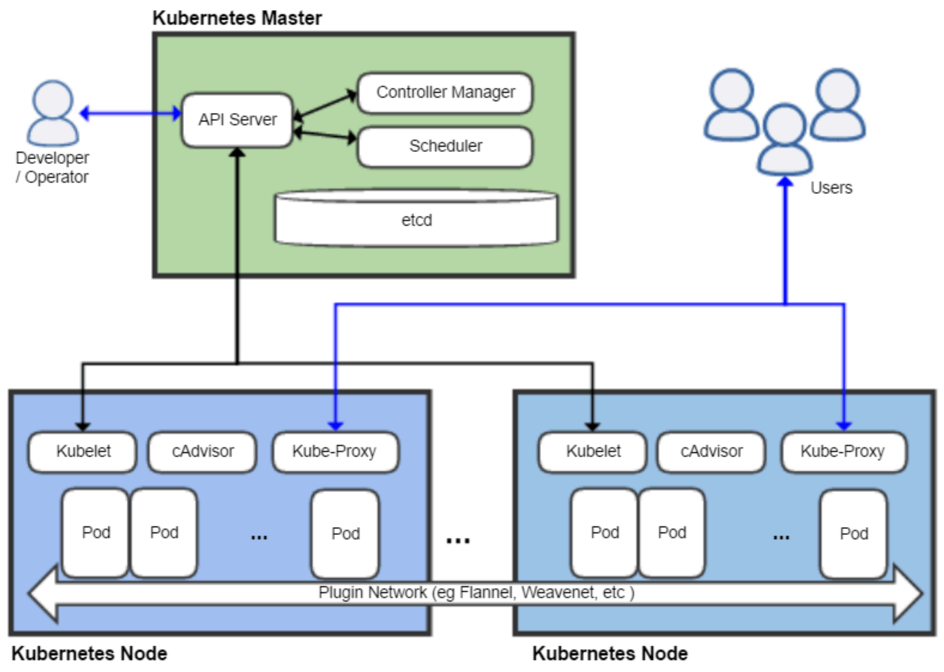
A set of machines managed by Kubernetes and treated as a single system.

Node

A single physical or virtual machine in the cluster.

Pod

Unit of deployment, contains containers that share a network and storage.



demo/basics/

Controllers

Desired state: specified by user and stored.

Observed state: continually reported and also stored.

Controllers reconcile observed state with desired state whenever there is a difference.

That's where the name Kubernetes comes from.

```
$ kubectl get pods -o name
pod/nginx-66686b6766-lnbsc
pod/nginx-66686b6766-lwwj9
pod/nginx-66686b6766-q7n8x
```

```
$ kubectl delete pod nginx-66686b6766-lnbsc
```

```
$ kubectl get pods -o name
pod/nginx-66686b6766-bl9tj
pod/nginx-66686b6766-lwwj9
pod/nginx-66686b6766-q7n8x
```

demo/delete_pod

Virtual Network

A software-defined network that provides the illusion of a single, unified network, independent of the underlying physical network.

Implemented with overlay networks: encapsulate packets to traverse underlying physical networks.

In Kubernetes:

Each Pod has a unique IP address independent of node structure.

Any Pod can reach any other Pod directly via its IP address.

Containers in the same Pod can communicate via localhost.

demo/network/

Service

A stable name for one or more locations of potentially changing servers.

Example: your phone contains a mapping from your friends' names to their numbers.

Common mechanisms:

- Configuration Files: hard-coded list of service locations.
- Environment Variables: inject service locations at startup.
- Domain Name System (DNS): map service name to one or more IP addresses.
- Service Registry: centralized database of service locations.

Service in Kubernetes

> In Kubernetes, a Service is a method for exposing a network application that is running as one or more Pods in your cluster.

Every service automatically has:

- a virtual IP that never changes
- a DNS name `my-service.my-namespace.svc.cluster.local`
- a shorthand DNS name `my-service` from within the same namespace
- environment variables injected into Pods that start after the Service

A DNS server runs in the cluster and queries a distributed database `etcd` for the current Service to Pod mapping.

demo/service/

Load Balancing

How to distribute requests for a single service across multiple servers.

Common approaches:

- DNS round-robin: DNS server returns one of multiple addresses to client.
- Client-side: client has list of servers and picks one for example randomly.
- Proxy server: dedicated server between client and servers.

Layer 4 (transport layer): decide based on IP address and port.

Layer 7 (application layer): decide based on HTTP headers, path, cookies, etc.

Load Balancing in Kubernetes

Service Load Balancing

Splits requests to a service among multiple Pods.

Randomly chooses one of the Pods by default. Layer 4.

Gateway Load Balancing

Splits requests to a URL among multiple Services.

Programmable rules based on path, hostname, headers. Layer 7.

demo/balancing/

Summary and Outlook

Orchestration is a wide and deep topic.

Next week: Repetition.