Aalto University
School of Science

Cloud Infrastructures for Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

This is not a cloud computing lecture but how cloud computing is important for big data platforms

Learning objectives

- Understand key cloud technologies
- Understand how cloud technologies empower big data platforms
- Understand cloud technologies enable us to acquire, utilize and manage resources for big data platforms

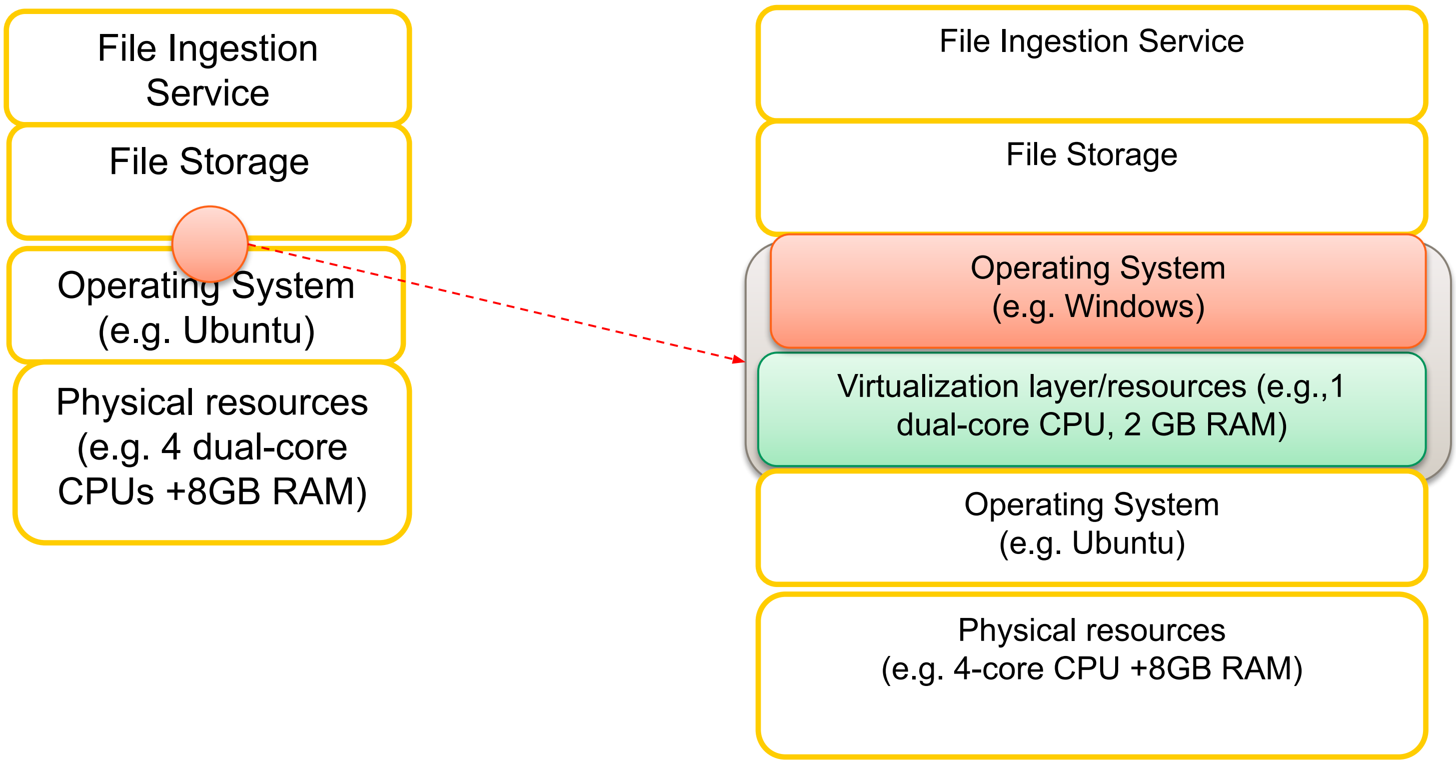
Service Model

- **Services offer well-defined interfaces for**
  - access resources: data, things, machines, and people
  - provide functions: ingestion, computation, sensing, analytics, inferences, etc.
  - offer diverse service level agreements (SLAs) for different types of business models (e.g.,pay-per-use and subscription)
- **Services are**
  - characterized by scalability, reliability, elasticity, etc.
  - provisioned in distributed systems of IoT, edge, cloud and HPC infrastructures, possibly a combination of different types of systems

Virtualization

- **Virtualization**
  - abstracts low-level compute, data and network resources to create *virtual version* of these resources
  - virtualization software creates and manages "virtual resources" isolated from physical resources
- **Virtualization is a powerful concept**
  - we can apply virtualization techniques virtually for everything!

Virtualizing physical resources



Main types of virtualization of infrastructures

- **Compute resource virtualization**
  - compute resources: CPU/GPU, memory, I/O, disks, etc.
  - "virtual machines"/containers
- **Storage virtualization**
  - resources: storage devices, hard disks, etc.
  - for usage and management of data storage
- **Network Function virtualization**
  - network resources: network equipment & functions
  - dynamically provision and manage network functions

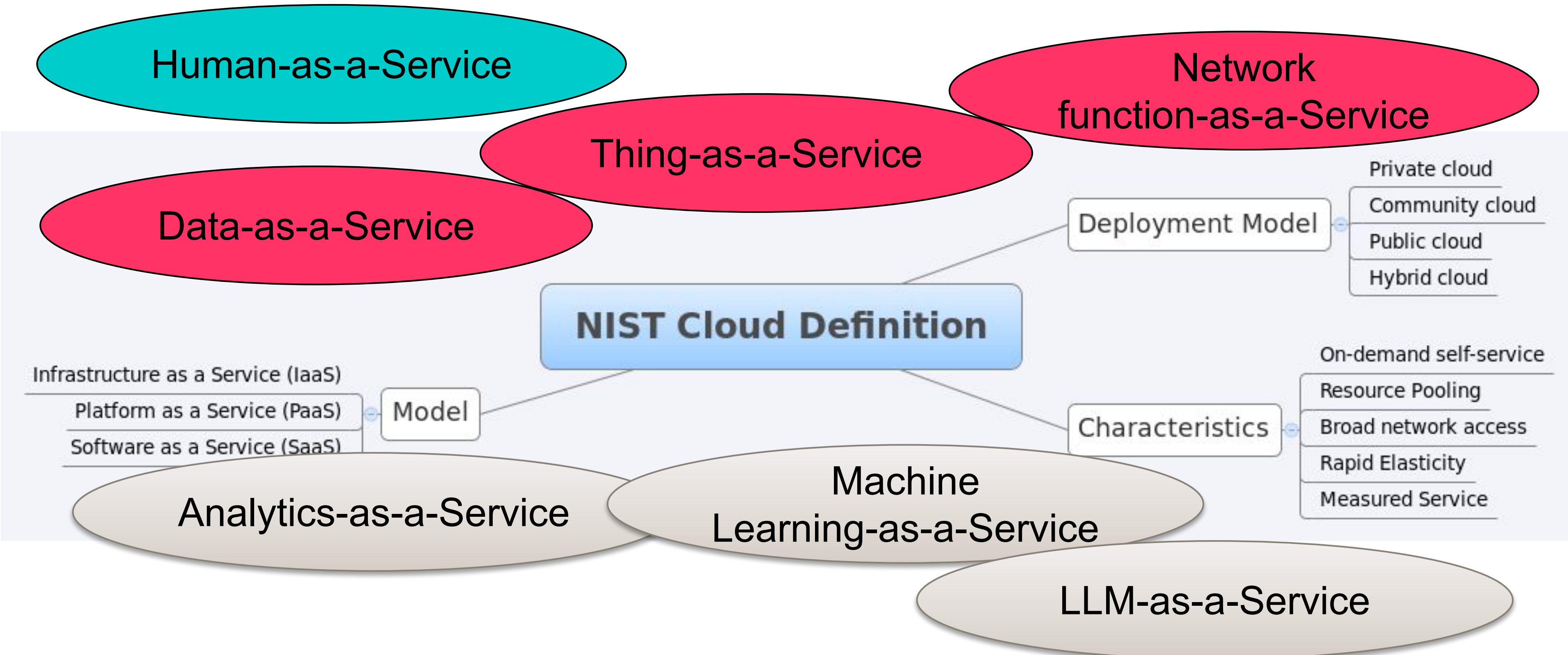
Cloud Computing

Original definition from NIST

"This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models."

Source: NIST Definition of Cloud Computing v15, http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

Cloud Computing



Cloud computing principles

- **"Cloud"**
  - not just data centers or public cloud infrastructures
- **For big data platforms: we need the "cloud mindset"**
  - apply cloud principles for developing and operating big data platforms
  - big data platforms can be in on-premise infrastructures empowered with cloud technologies!
  - part of the big data platforms can be in the edge with similar technologies, such as microservices and Kubernetes
  - both **multicloud and hybrid cloud** systems can be used used

Compute resource virtualization technologies

- **Physical compute resources for big data platforms**
  - individual physical hosts/servers (CPU, memory, I/O)
  - clusters, supercomputer, and data centers
- **At the low-level: two main streams**
  - hypervisor/Virtual Machine monitor
    - *Virtual machines (VirtualBox, VMWare, Zen, etc.)*
  - containterlization
    - *Containers (Linux Containers, Docker, Warden Container, OpenVZ, OCI based containers, etc.)*

Virtual infrastructural resources for big data platforms

- **For big data platforms: we leverage clusters/infrastructures of VMs/containers**
  - computing resources for core services and data
    - *e.g., data storage, data ingestion, data processing, and messaging*
- **On-demand resources for large-scale deployments**
  - compute nodes, storage, communication, etc.
  - virtual data centers work like a single distributed system
- **On-demand resources for elastic workload**
  - e.g., for data ingestion and analytics tasks

Example: OpenStack

- **A Big Data Platform can be built based on OpenStack (or similar)-based data center**

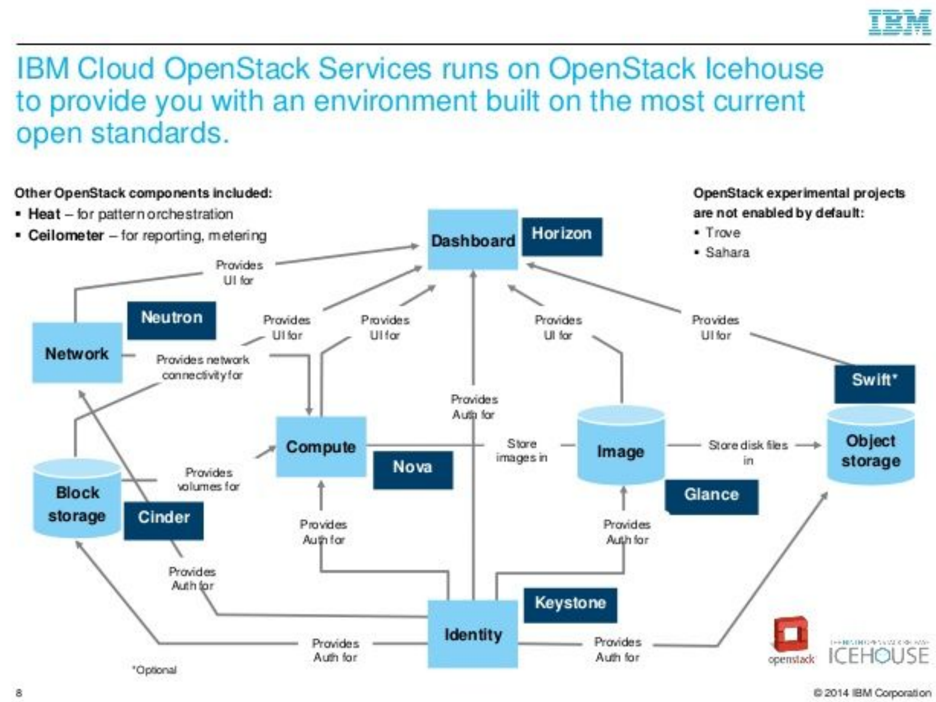


Figure source: http://www.slideshare.net/OpenStack_Online/ibm-cloud-open-stack-services

Example: Kubernetes

- Support Docker, rkt, runc, etc.
- Act as a resource orchestration and management for big data platforms

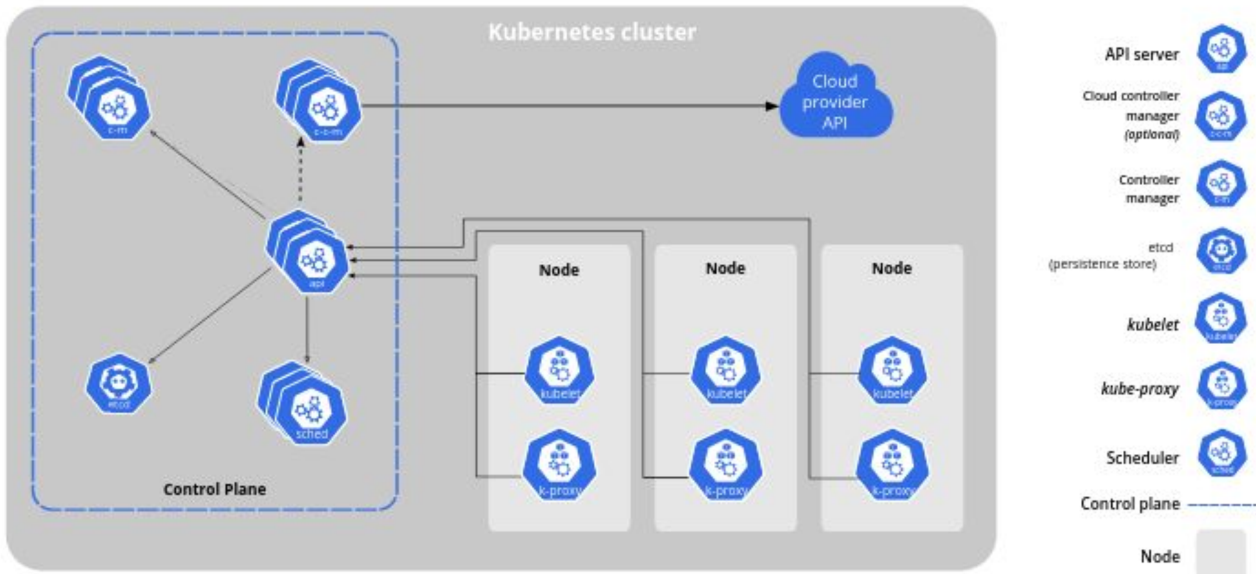


Figure source: https://kubernetes.io/docs/concepts/architecture/cloud-controller/

Example: storage virtualization

- **Low-level storage**
  - e.g., VMware Virtual Machine File Systems
- **High-level, e.g., database**
  - MySQL Cluster + auto-sharding

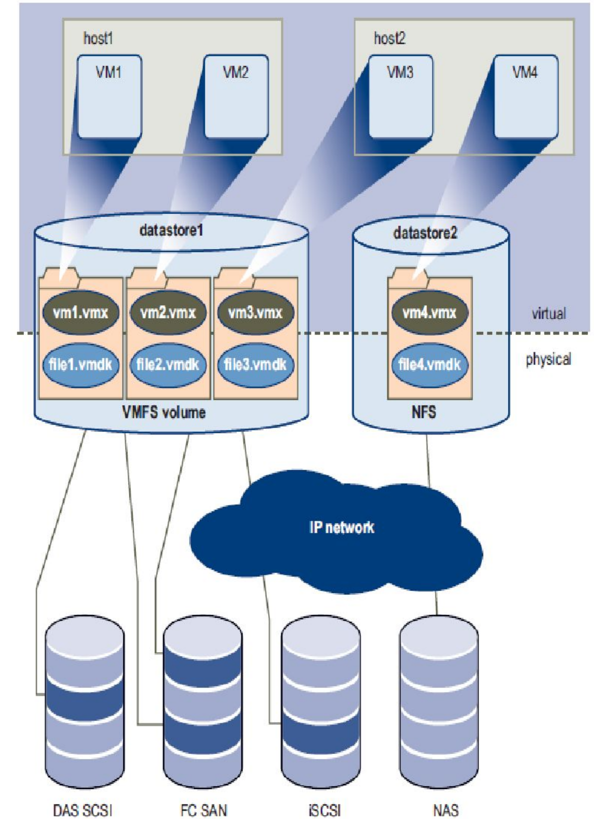


Figure source:
https://www.vmware.com/pdf/vi_architecture_wp.pdf

Can be used as the Data Storage layer for Data Lake or Data Services

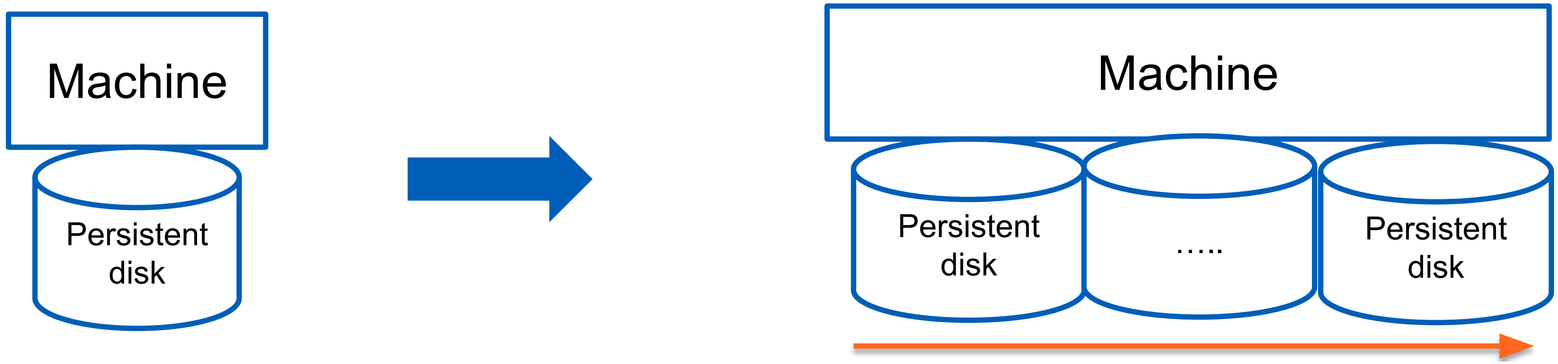
#1: Enabling managed services

Many options for resource provisioning in big data platforms

- **Managing *infrastructural resources* for big data platforms must be easier with virtualization**
  - different types of storage: block storage and file storage
  - many CPU/memory configurations: single core to many cores
  - suitable for different workloads
- **Different SLA offerings**
  - reliability, security, performance, maintainability ...
- **Elasticity**
- **Globalization support: important for many businesses**
  - design with multiple clouds and hybrid clouds

On-demand storage provisioning

- **Big data requires big storage which can be changed on-demand!**



Dynamic numbers and sizes

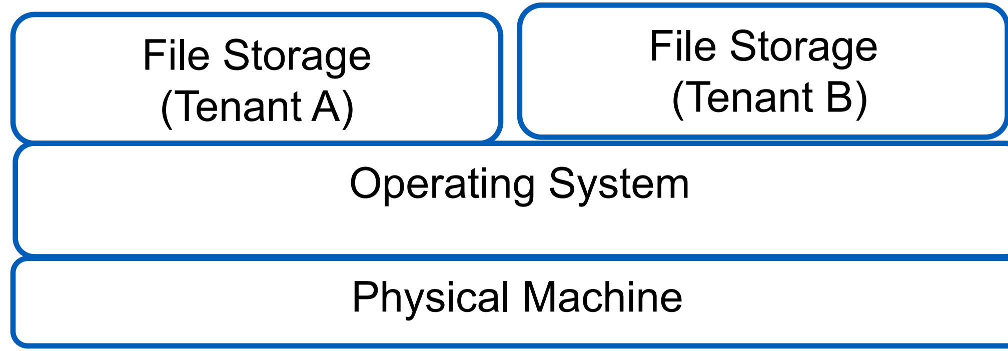
- **Examples of dynamic configurations:**
  - Google persistent disks: different types of disks, can have 128 disks and max 257 TiB for a single VM

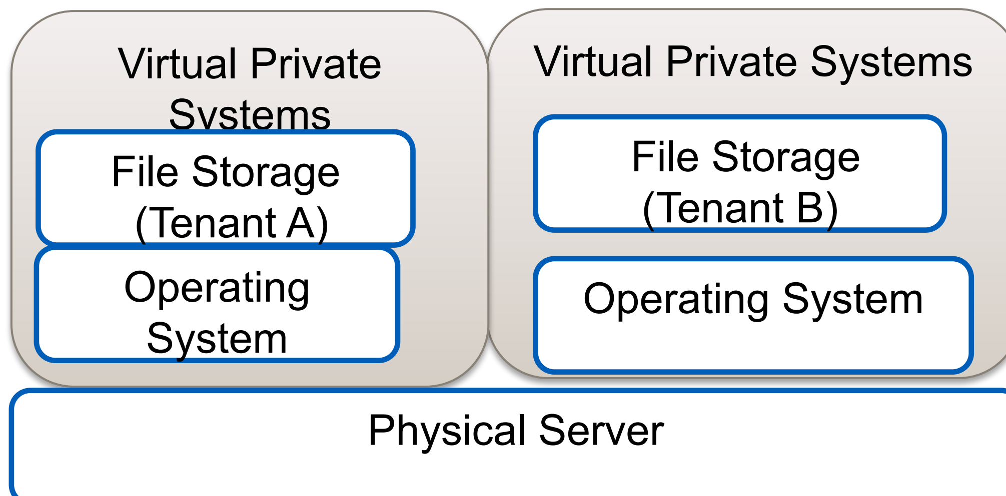
Flexible computing capabilities

- **Different workloads and programming models in big data platforms need flexible computing resource provisioning**
  - Storage, Data Ingestion, and Analysis
- **Examples**
  - small data ingestion/analysis jobs
    - *with pandas, DuckDB, scikit-learn (few cores)*
  - large-scale MapReduce/Spark ⇒ clusters of VMs/containers
  - Machine Learning with TensorFlow ⇒ TPU (Tensor Processing Unit)
- **Cloud technologies easily enable different computing capability configurations**

Security improvement

- Tenant's service isolation while platforms support multiple tenants
- Virtual private instances for security and performance of tenant's data!





#2: Achieving fault tolerance, performance and elasticity

Easing Replication Management

Passive (Primary backup) model:

- FE (Front-end) can interface to a Replication Manager (RM) to serve requests from clients.
- E.g., in MongoDB

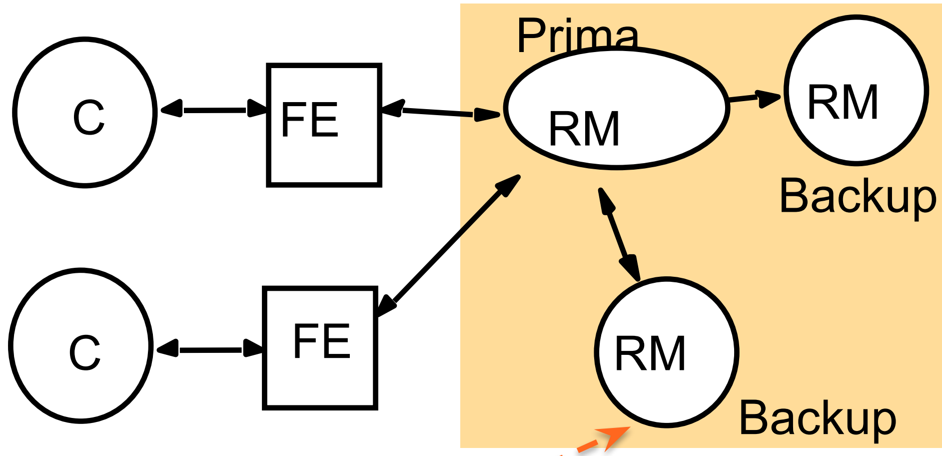


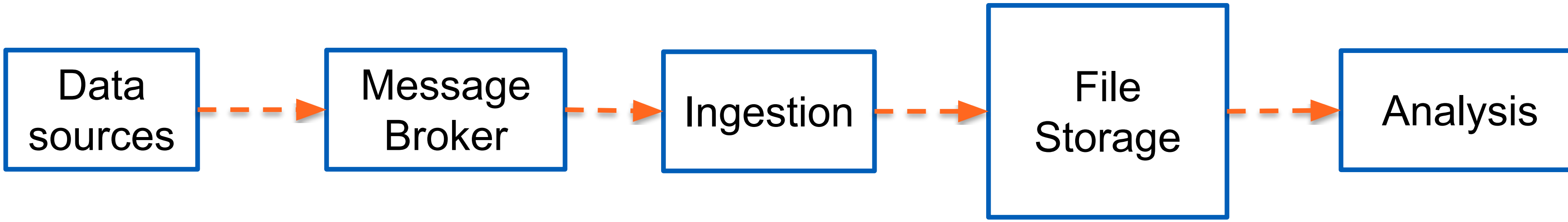
Figure source: Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5

Easy to deploy, globalize, manage and replace RM using cloud resources

High availability and performance

- **Cost/optimization**
  - elasticity, hot deployment, etc.
  - cloud bursting with hybrid cloud (combining private/on-premise + public resources)
  - edge-cloud continuum
- **Improving service performance in incident management**
  - e.g., spend time to fix a machine or just quickly relaunch a new one (and fix the old one later) ?

Scaling in every place of the data pipelines

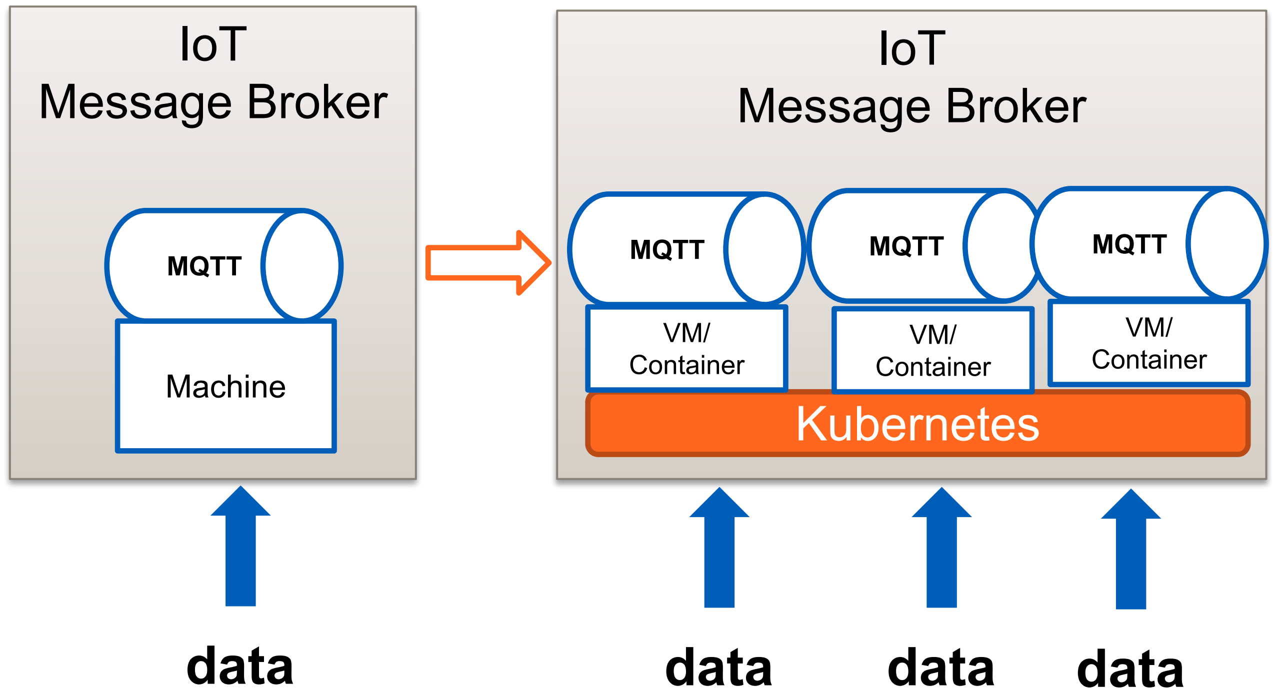


- **Scaling**
  - disk spaces for file storage
  - resources for data ingestion
  - resources for data analysis

Happen at different times and locations

Scaling middleware nodes

- Increase the number of brokers when more data arrive
- Provide dedicated brokers on-demand



Example: scaling compute nodes for data analysis

Monitoring Jobs VM Instances Configuration Web Interfaces

Name	Role	
thebasecluster-m	Master	SSH
thebasecluster-w-0	Worker	
thebasecluster-w-1	Worker	
thebasecluster-w-2	Worker	
thebasecluster-w-3	Worker	

Equivalent REST

4 nodes

On-demand change

Name	thebasecluster
Region	europe-north1
Zone	europe-north1-a
Autoscaling	Off
Scheduled deletion	Off
Enhanced flexibility mode	Off
Master node	Standard (1 master, N workers)
Machine type	n1-standard-2 (2 vCPU, 7.50 GB memory)
Primary disk type	pd-standard
Primary disk size	500 GB
Worker nodes	6
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)

Monitoring Jobs VM Instances Configuration Web Interfaces

Name	Role	
thebasecluster-m	Master	SSH
thebasecluster-w-0	Worker	
thebasecluster-w-1	Worker	
thebasecluster-w-2	Worker	
thebasecluster-w-3	Worker	
thebasecluster-w-4	Worker	
thebasecluster-w-5	Worker	

6 nodes

#3: Living in the world of Microservices & DevOps

Microservices

- **Many components for data storage, data processing and ingestion**
  - microservices can be used to design components of big data platforms
  - in particular: services serving data requests and services for storing data in the platform
- **Big data platforms provide features for other microservices**

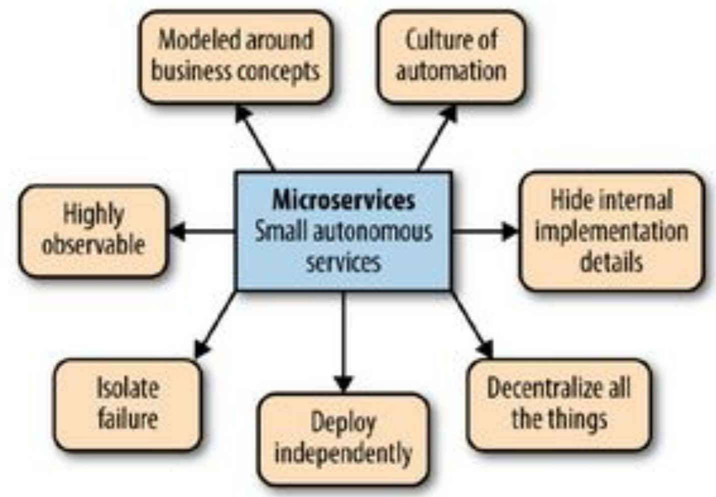


Figure source: Sam Newman, Building Microservices, 2015

Examples

Microservices for both services using big data platforms and components of data platforms

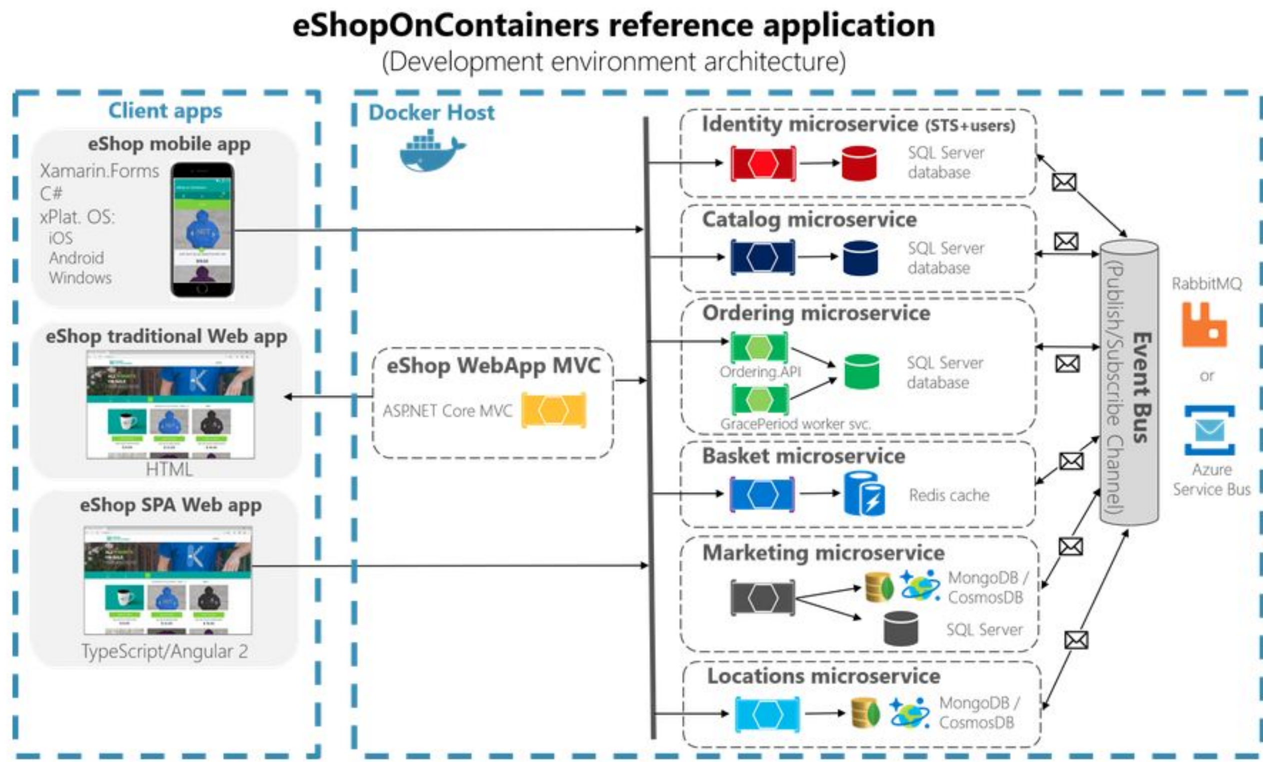


Figure source:
https://blogs.msdn.microsoft.com/dotnet/2017/08/02/microservices-and-docker-containers-architecture-patterns-and-development-guidance/

DevOps

- Close the gap between development/test environment and real/production environments
- Simplify testing, emulating real environments, etc.

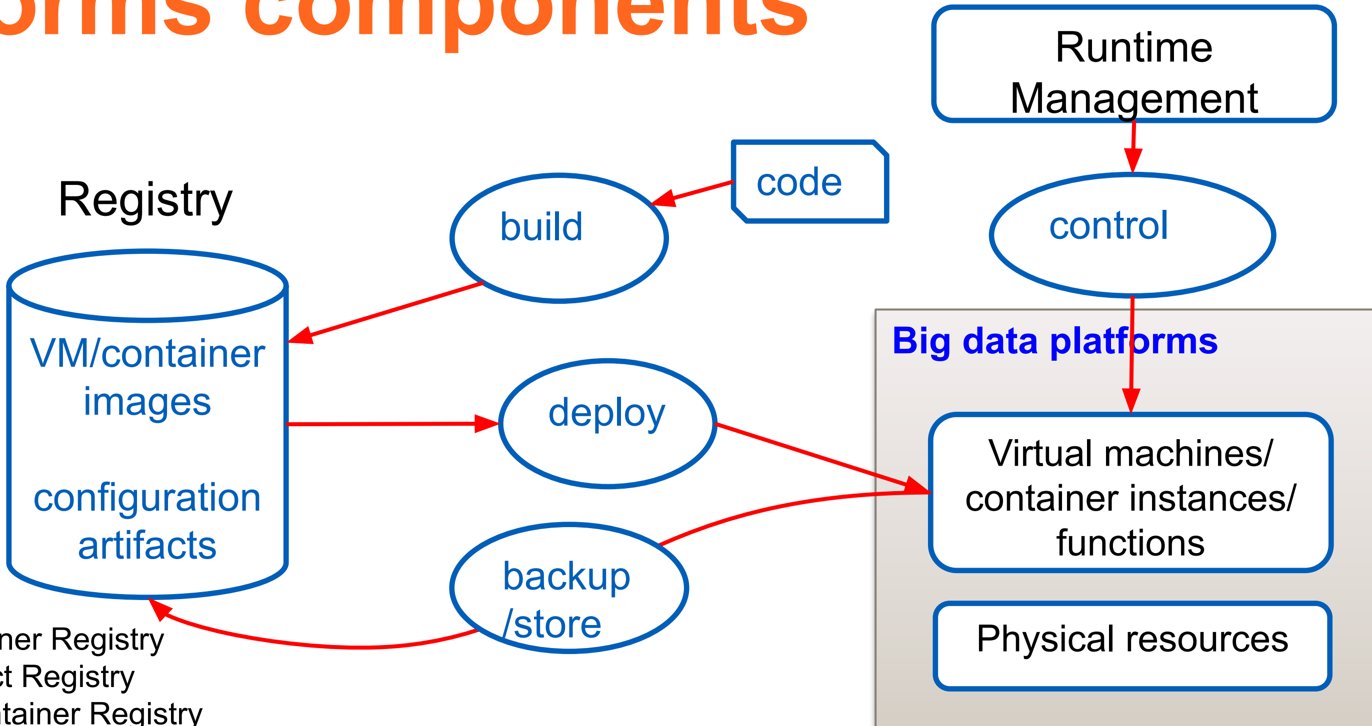
DevOps for big data platforms are part of software systems under DevOps. However, many tasks have to deal with requirements from big data analytics and services!

For example: testing a data pipeline (data intensive test cases) != testing an integrated set of software components

Tools and frameworks and providers for infrastructural resources provisioning and service deployment:

Chef, Vagrant, Terraform, Amazon, Google, Microsoft, OpenStack, OpenShift, …

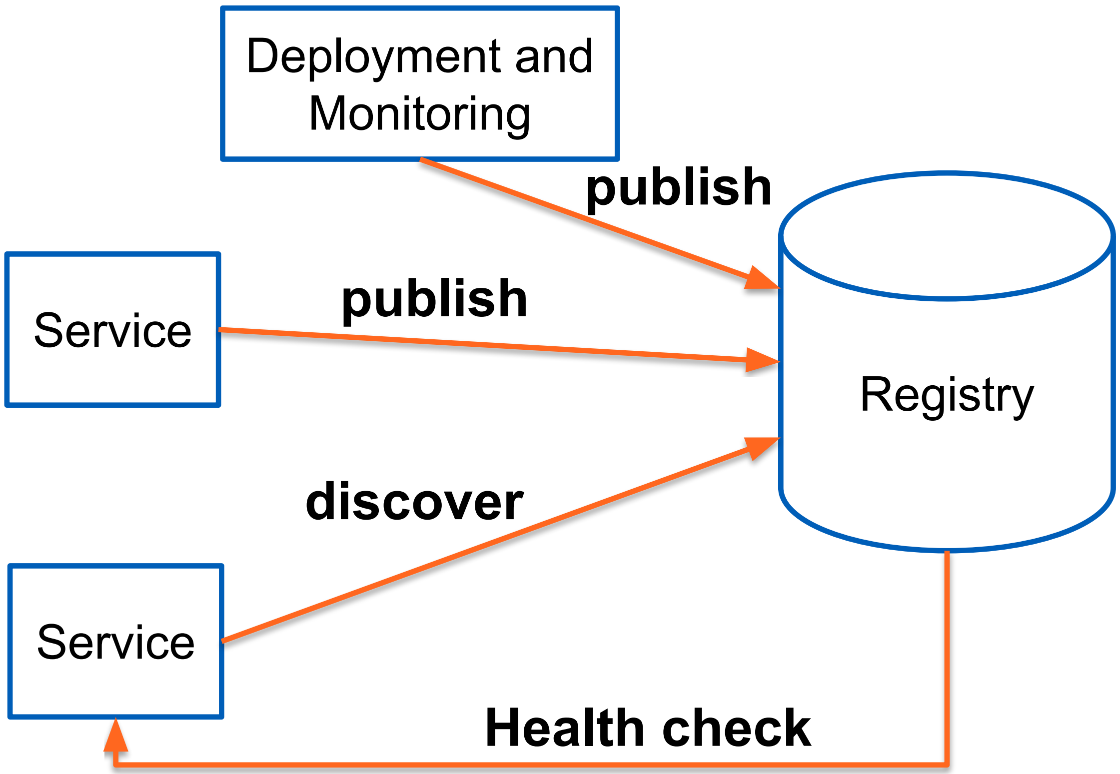
Provisioning and management of platforms components



Google Container Registry
Google Artefact Registry
AWS EC2 Container Registry
Azure Container Registry
Etc.

#4: Service Discovery

Service Discovery principle



- **Key requirements**
  - fast
  - consistent
  - secure
  - cross data centers
  - simple APIs

Distributed coordination

- **A lot of algorithms, etc.**
  - Paxos family
- **Well-known in the cloud**

TABLE 4. PATTERNS OF PAXOS USE IN PROJECTS

		Usage Patterns								
Project	**Consensus System**	**SR**	**LR**	**SS**	**BO**	**SD**	**GM**	**LE**	**MM**	**Q**
GFS	Chubby			✓				✓	✓	
Borg	Chubby/Paxos	✓				✓		✓		
Kubernetes	etcd						✓		✓	
Megastore	Paxos		✓							
Spanner	Paxos	✓								
Bigtable	Chubby						✓	✓	✓	
Hadoop/HDFS	ZooKeeper	✓						✓		
HBase	ZooKeeper	✓		✓			✓		✓	
Hive	ZooKeeper			✓					✓	
Configerator	Zeus								✓	
Cassandra	ZooKeeper					✓		✓	✓	
Accumulo	ZooKeeper		✓	✓					✓	
BookKeeper	ZooKeeper						✓		✓	
Hedwig	ZooKeeper						✓		✓	
Kafka	ZooKeeper						✓	✓	✓	
Solr	ZooKeeper							✓	✓	✓
Giraph	ZooKeeper		✓		✓				✓	
Hama	ZooKeeper				✓					
Mesos	ZooKeeper							✓		
CoreOS	etcd					✓				
OpenStack	ZooKeeper					✓				
Neo4j	ZooKeeper			✓				✓		

Notes from the paper: “server replication (SR), log replication (LR), synchronization service (SS), barrier orchestration (BO), service discovery (SD), group membership (GM), leader election (LE), metadata management (MM) and distributed queues (Q)”

Source: Ailidani Ailijiang, Aleksey Charapkoy and Murat Demirbasz , Consensus in the Cloud: Paxos Systems Demystified, http://www.cse.buffalo.edu/tech-reports/2016-02.pdf

Technology choices: ZooKeeper

- https://zookeeper.apache.org/
- Support service discovery, configuration information and distributed synchronization
- Centralized registry service
- Data is organized into a shared hierarchical name space
  - small data size
- Highly available and reliable

ZooKeeper Service

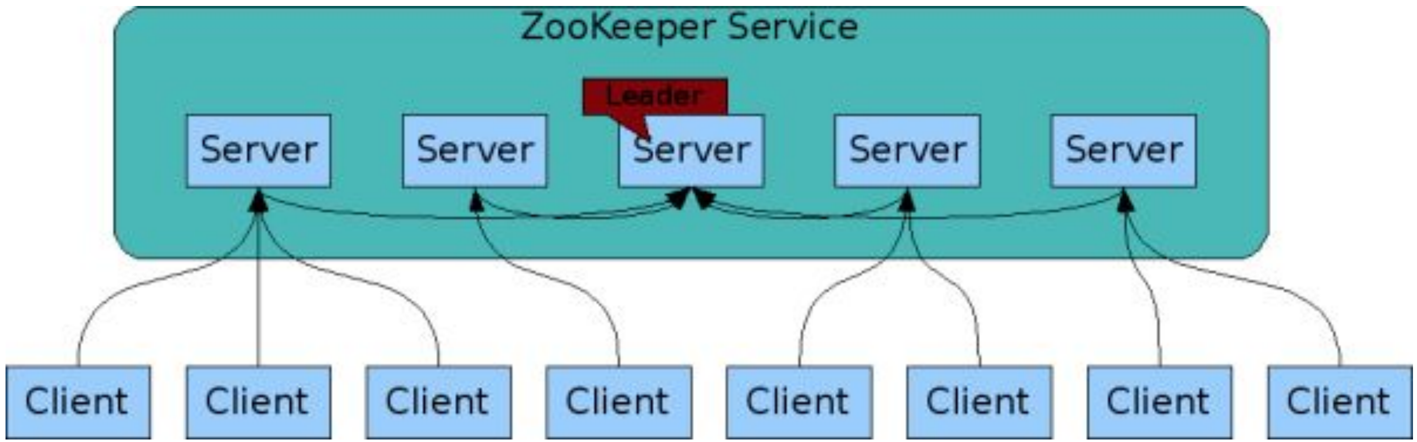


Figure source: https://zookeeper.apache.org/doc/r3.4.10/zookeeperOver.html

ZooKeeper data -- znodes

- **Data nodes called znodes**
- **Missing data in a znode ⇒ problems with the entity that the znode represents**
- **Persistent znode**
  - /path deleted only through a delete call
- **Ephemeral znode, deleted when**
  - the client created it crashed
  - session expired

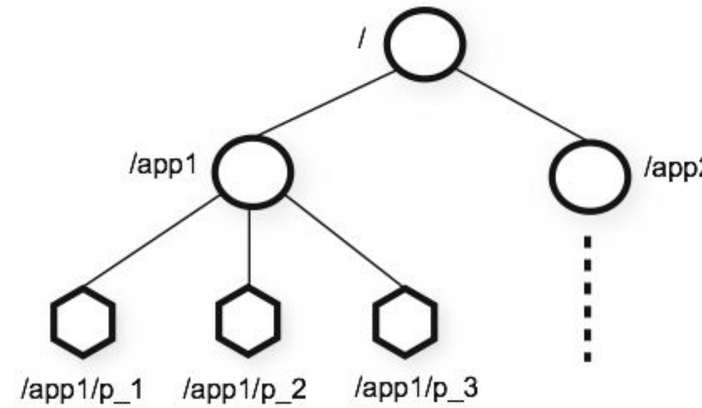


Figure source:
https://zookeeper.apache.org/doc/r3.4.10/zookeeperOver.html

Technology choices: Consul

- **https://www.consul.io**
- **Cross data centers**
- **End-to-end service discovery**
    - include health check

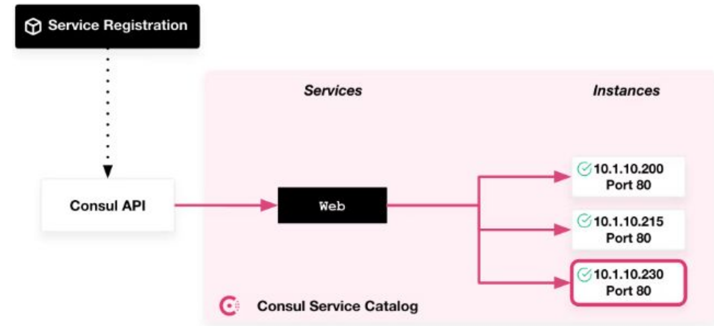


Figure source:
https://developer.hashicorp.com/consul/docs/concepts/service-discovery

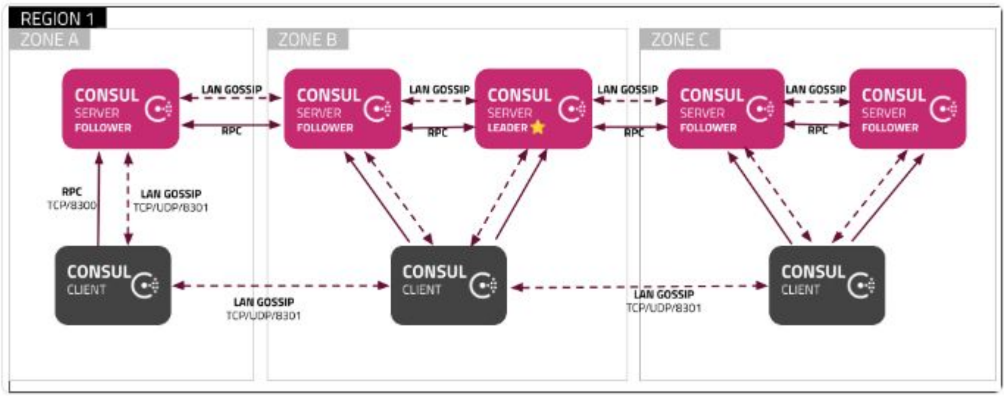


Figure source:
https://developer.hashicorp.com/consul/tutorials/production-deploy/reference-architecture

Technology choices: etcd

- **Consistent, distributed key-value store**
- **Allow monitor changes of keys/directories**
  - enable reactive actions based on changes
- **Widely used for**
  - service discovery and state/configuration management
  - distributed key locking
  - e.g. in Kubernetes

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io