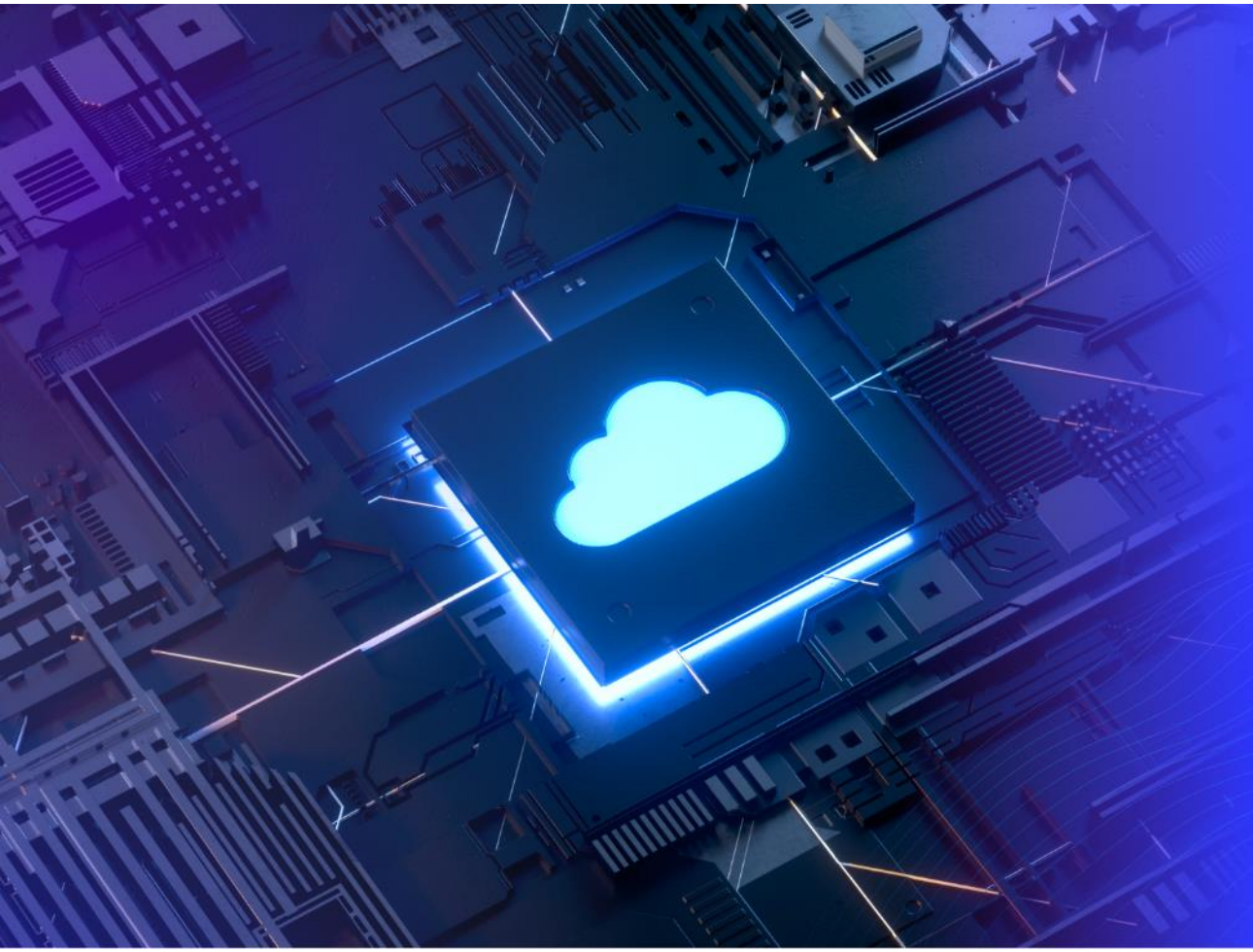
台灣積體電路製造股份有限公司
Taiwan Semiconductor Manufacturing Company, Ltd.



TSMC IT × NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

設計以及架構建置 K8S 自有叢集

資訊系統暨通訊服務處 系統建構一部 | 李青峰

March 16, 2022

AGENDA

Part 1: Why Kubernetes?

- What's the goal of K8s

Part 2: Kubernetes Resources

- Abstracted building blocks
- Hands-on lab

Part 3: Kubernetes Components

- How does it work
- On-Prem deployments

References

About me – 李青峰

2015-2017 / MComp, NTHU

Design & operate container orchestration platform in TSMC

- **2017-2018 /**
  - Build and operate Apache Mesos clusters

- **2019-2021 /**
  - Equipment edge computing cluster design
  - On-prem RedHat okd infrastructure technical architect

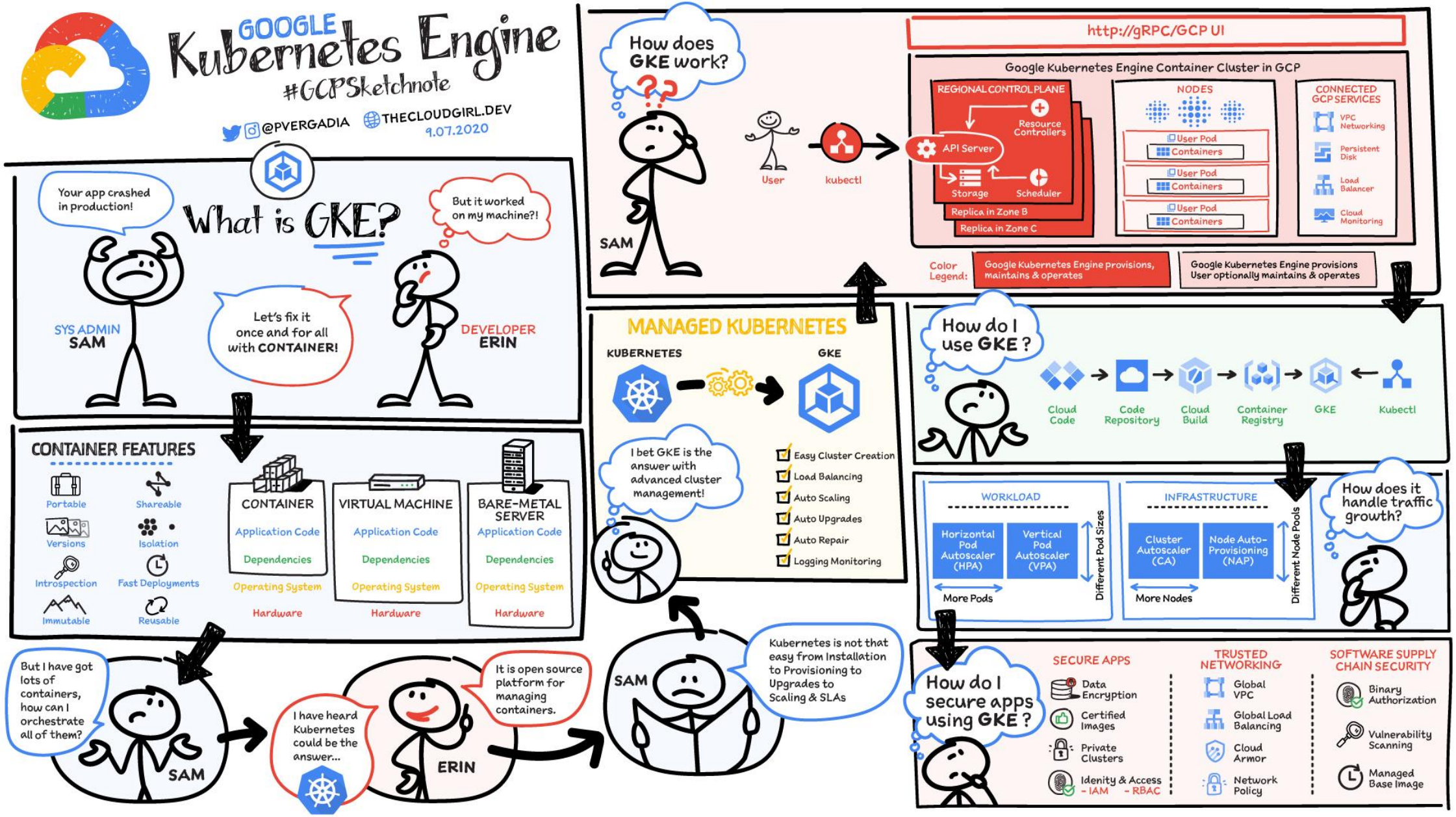
- **2021-today /**
  - Develop and build Kubernetes platform, specialized in network design

Goal

You will learn:

- What is Kubernetes (K8s)?
- Components of a K8s cluster
- Get familiar with K8s resources
- How to run apps on a K8s platform!

Part 1: Why Kubernetes (K8s)?

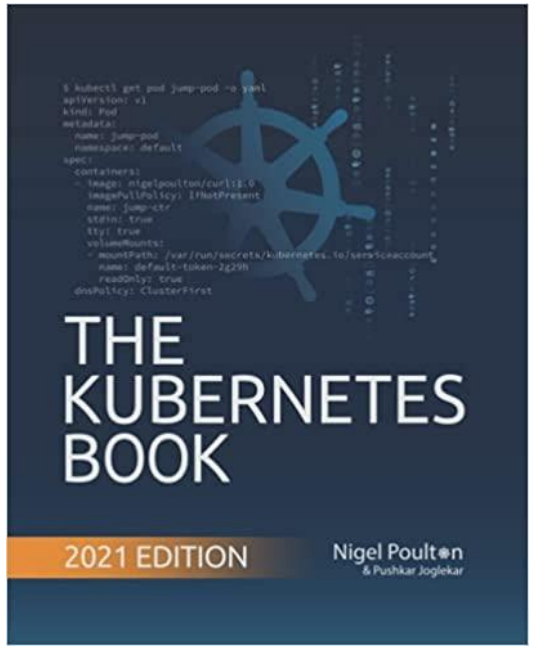


TSMC IT × NCTU CS
2022 CLOUD NATIVE
Development Best Practice

What is Kubernetes (K8s)

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

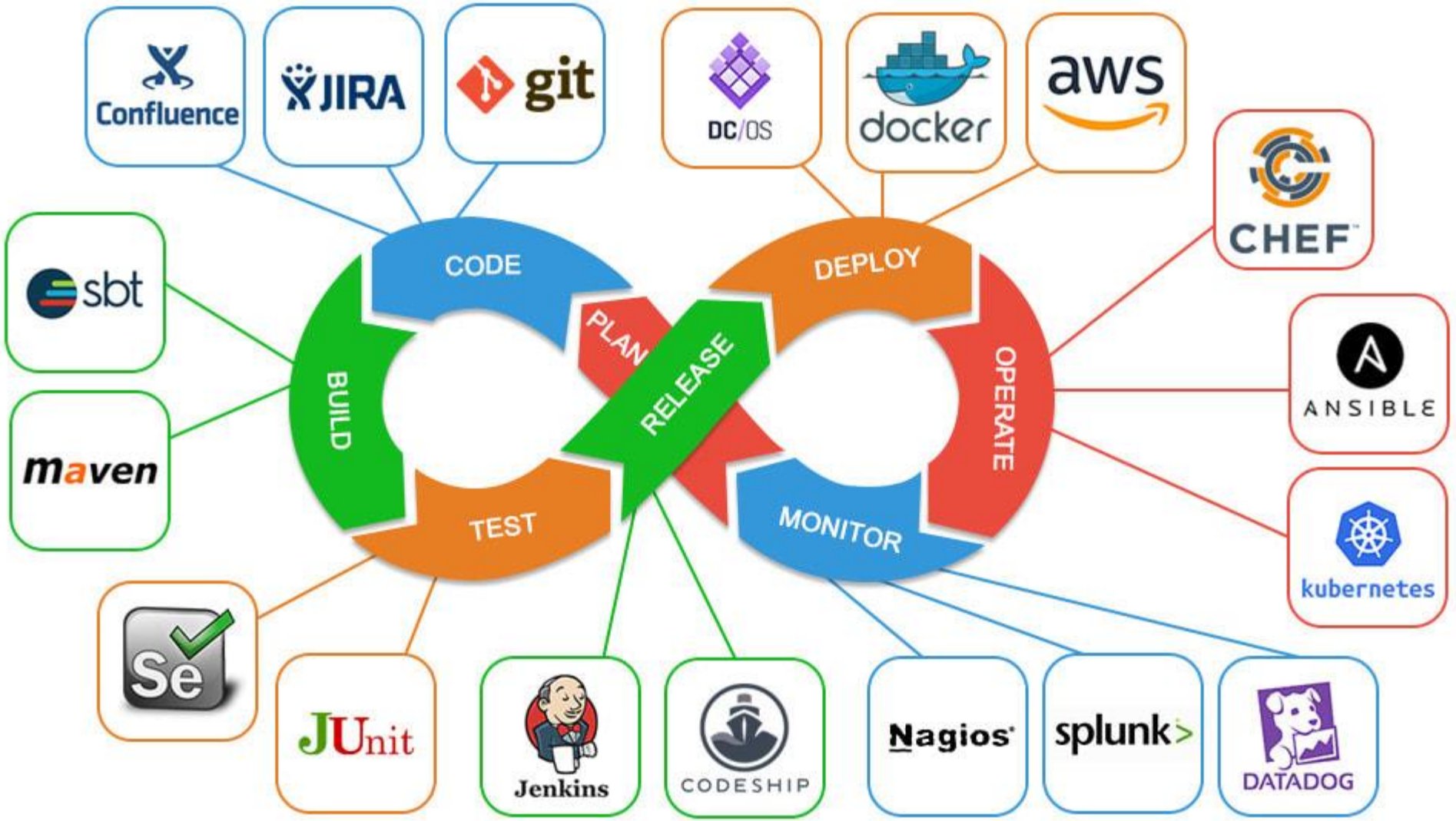
- **A container orchestration tool**
- **Production-Grade**
- **Written on Golang**
- **Developed by Google**
  - Borg (~2005) → Omega → Kubernetes (2014)



- https://kubernetes.io/
- https://www.cncf.io/blog/2019/08/19/how-kubernetes-works/

Key Goals of Kubernetes

- **Distribute containers in a logical & efficient way**
  - Automatic bin-packing
  - ▶ Maximize resource usage
- **Scale up (or down) easily**
  - Horizontal scaling
  - ▶ Adapt to demand
- **Keep applications running healthy & continuously**
  - Self-healing
  - ▶ Never go dark

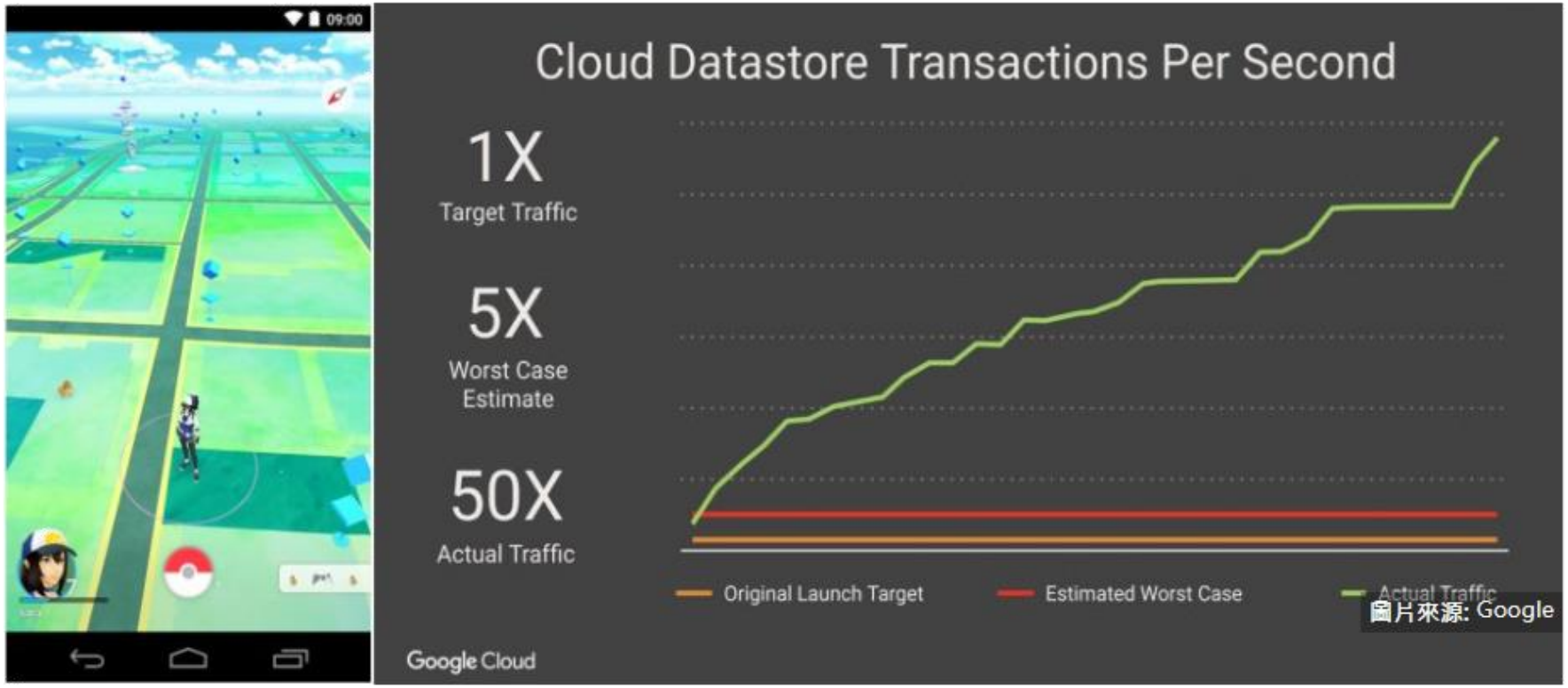


TSMC IT × NCTU CS
2022 CLOUD NATIVE
Development Best Practice

迎戰50倍爆量夢魘！ Pokémon遊戲打造GCE史上最大Kubernetes叢集

Niantic用Google的Cloud Datastore資料庫服務來儲存所有玩家資料，這是架構起Pokémon遊戲世界最主要的資料庫。但在遊戲上線第一天，不到15分鐘，Cloud Datastore每秒存取次數迅速從5倍、10倍，增加到了比預期多50倍的爆量流量。

文/ 王宏仁 | 2016-09-30 發表



The Need for a Container Orchestrator

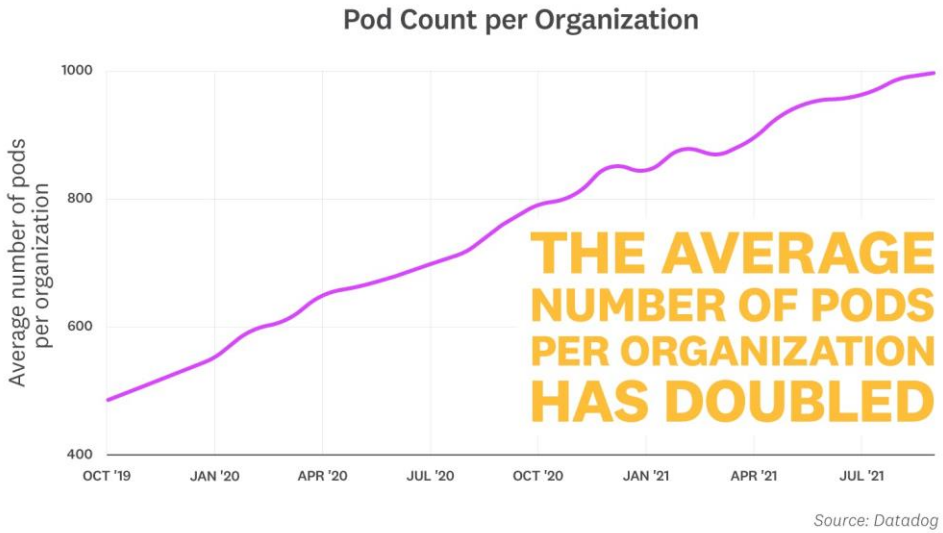
From Monolith to Microservices

- Speed, Scale, Agility
- Container is the ideal form for microservice deployments

Increased Usage of Containers

- An average organization using K8s today have thousands of containers running

Need a proper way to manage those thousands of containers!

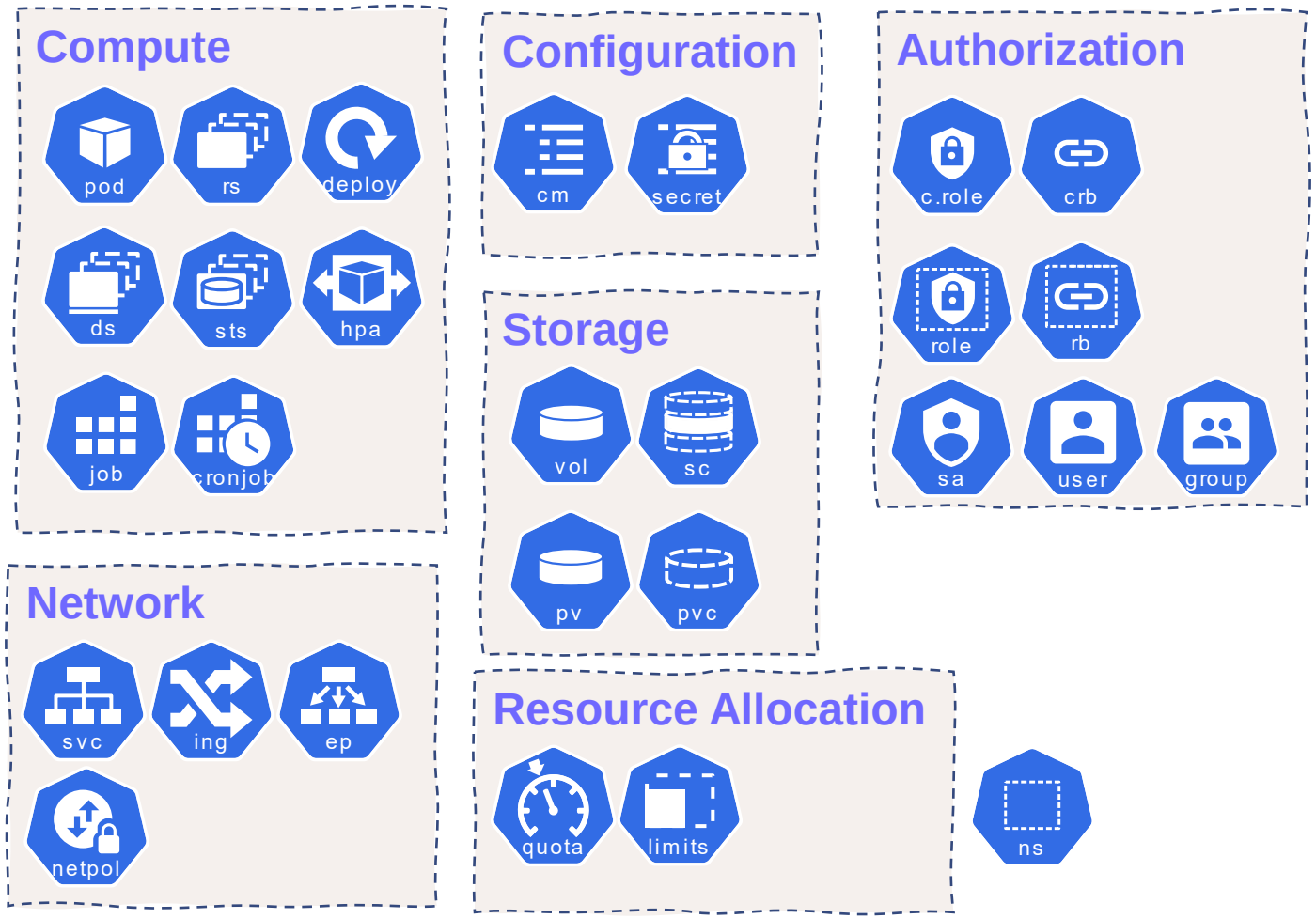


- https://www.datadoghq.com/container-report/

Part 2: Kubernetes Resources

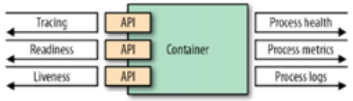
(Abstracted Infrastructure Building Blocks)

Commonly Used K8s Resources

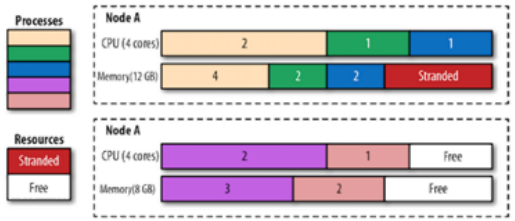


Top 10 Must-Know Design Patterns for Kubernetes Beginners

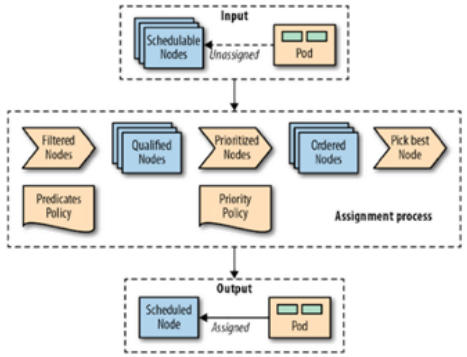
Foundational



Health Probe

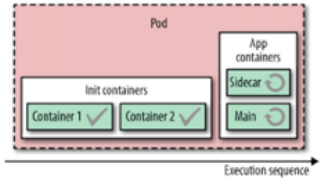


Predictable Demands

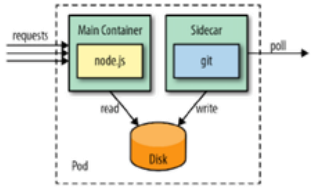


Automated Placement

Structural

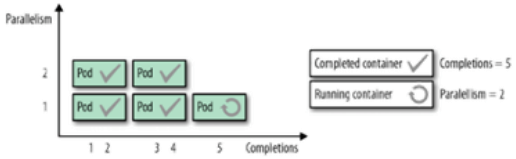


Init Container

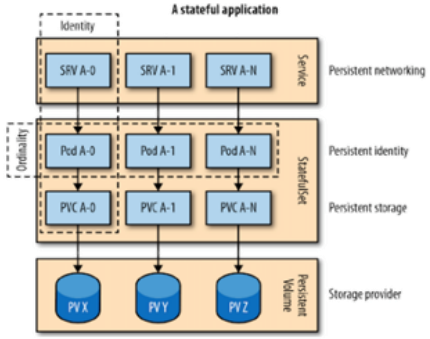


Sidecar

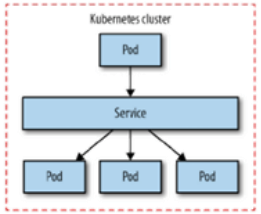
Behavioural



Batch Job

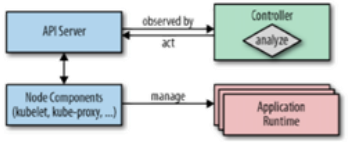


Stateful Service

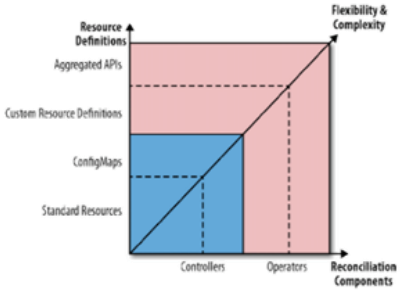


Service Discovery

Higher-level



Controller



Operator

https://developers.redhat.com/blog/2020/05/11/top-10-must-know-kubernetes-design-patterns

The “YAML” Configuration Files in K8s

The 3 Parts of an K8s Object Configuration File

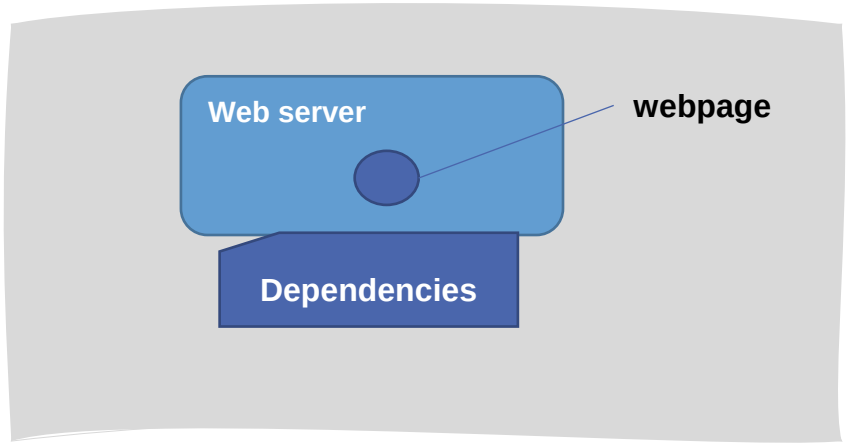
- **Metadata**
  - Namespace / Name / Labels / Annotations / …
- **Spec**
  - How it is defined
  - The desired states
- **Status**
  - How it is run
  - The current states

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
status:
  conditions:
  - ...
  hostIP: 192.168.0.113
  phase: Running
  podIP: 10.88.0.3
  podIPs:
  - ip: 10.88.0.3
  - ip: 2001:db8::1
  qosClass: Guaranteed
  startTime: "2022-02-17T21:51:01Z"
```

The Application

What are the essentials?

- **The application itself**
- **Supported by:**
  - The web server
  - Shared libraries
  - Various pieces of the OS



External microservices dependencies

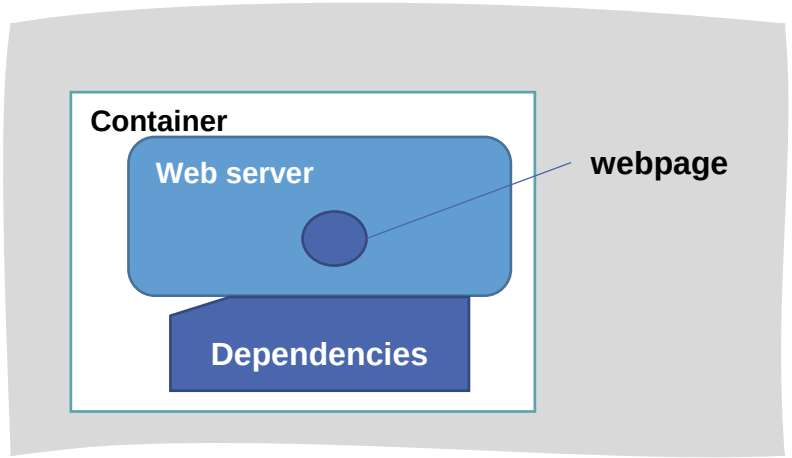
- **Database / Message Queues / …**

Containers

“A container image is a ready-to-run software package, containing everything needed to run an application”

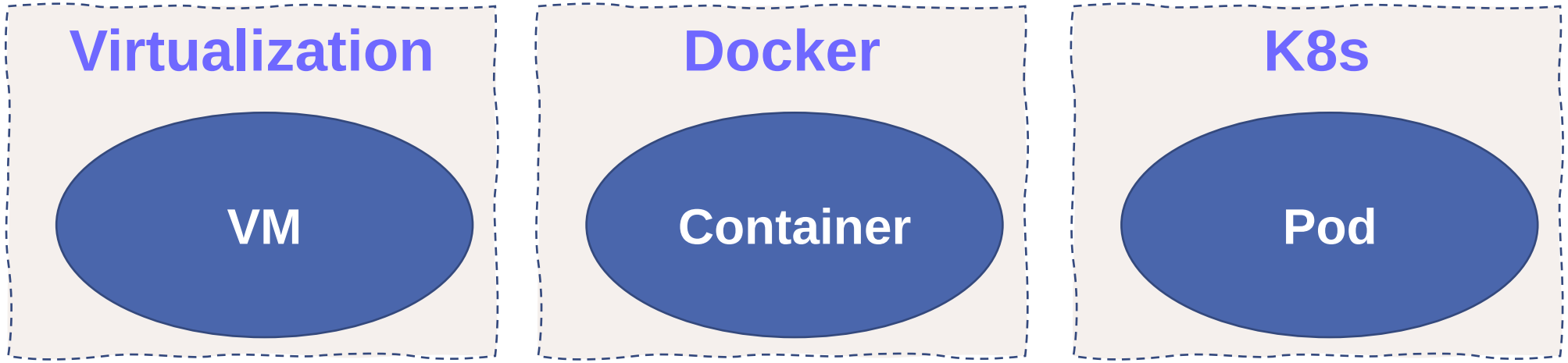
“A container instance is a container image bring to life”

- **Portable**
- **Fast Deployments**
- **Isolation**
- **Disposable**
- **Repeatable**



Pods

“Pods are the smallest deployable unit in a K8s cluster”



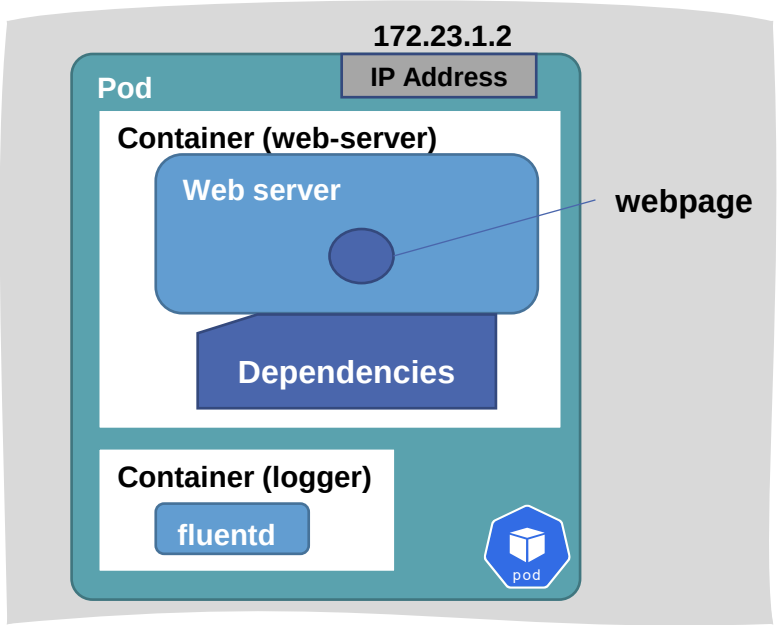
Pods

“Pods are the smallest deployable unit in a K8s cluster”

- **A pod can hold any number of containers**

1-container-per-pod	The most common use case
N-containers-per-pod	Co-located containers that are tightly coupled and need to share resources

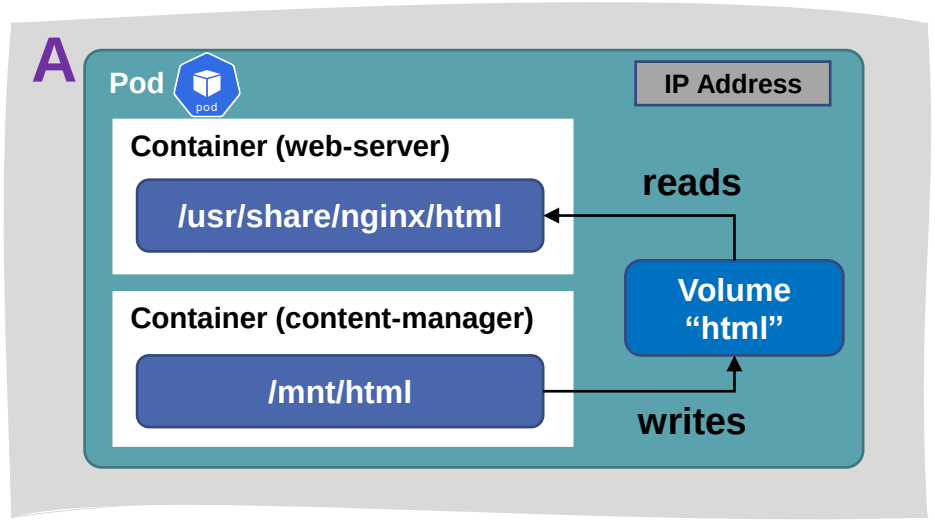
- **Why do we need “Pods” on top of containers?**
  - Support helper containers that assist a primary application
- **Uniquely addressable by an IP address**
  - New address on pod re-creation
- **They are stateless in themselves**
  - Volumes can be mounted for stateful applications

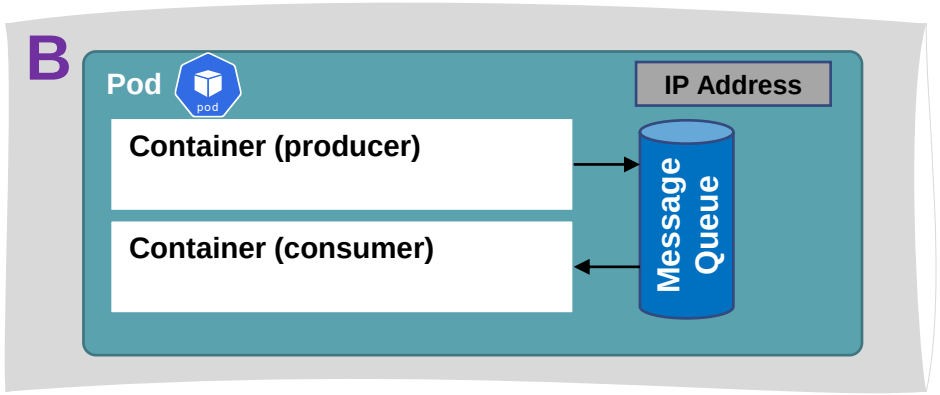


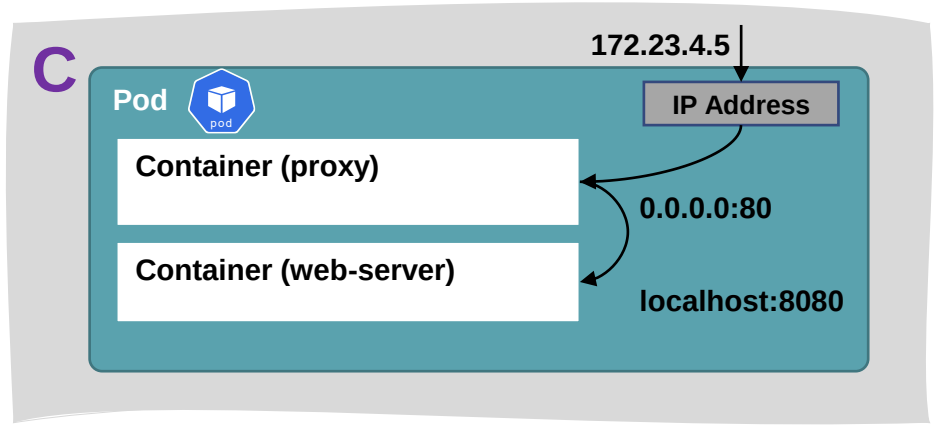
Pods: Inter-Container Communications

There are 3 common patterns:

A. **Shared volumes**
B. **Inter-process communications (IPC)**
C. **Network**



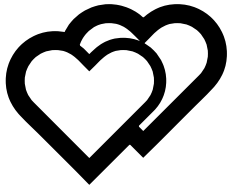




TSMC IT × NCTU CS
2022 CLOUD NATIVE
Development Best Practice

Pet vs Cattle

- Given a familiar name
- Taken to vet when sick
- Hugged



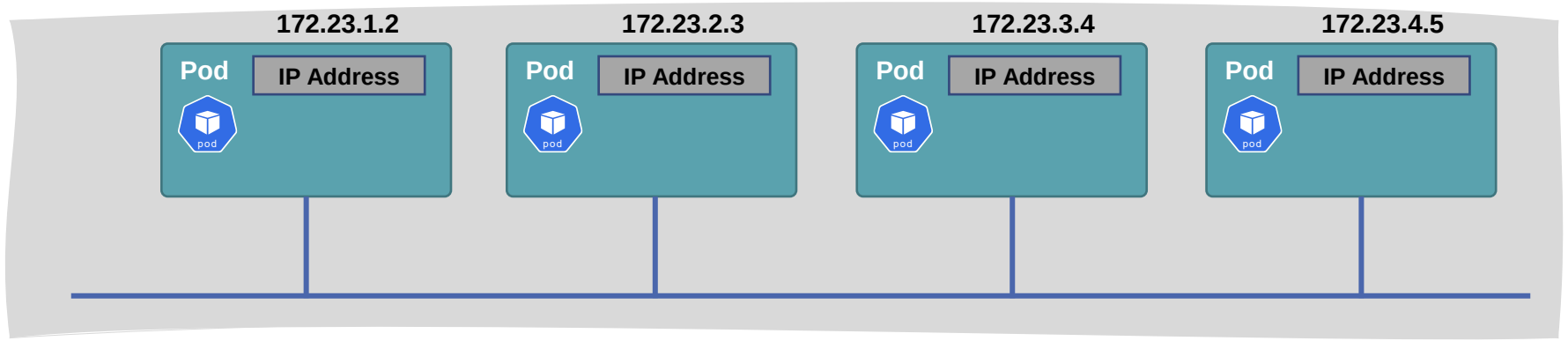
- Branded with an obscure name
- Shot when sick
- Eaten / Recycled

Lab 1: Create a *Pod*

Pod Networking Model

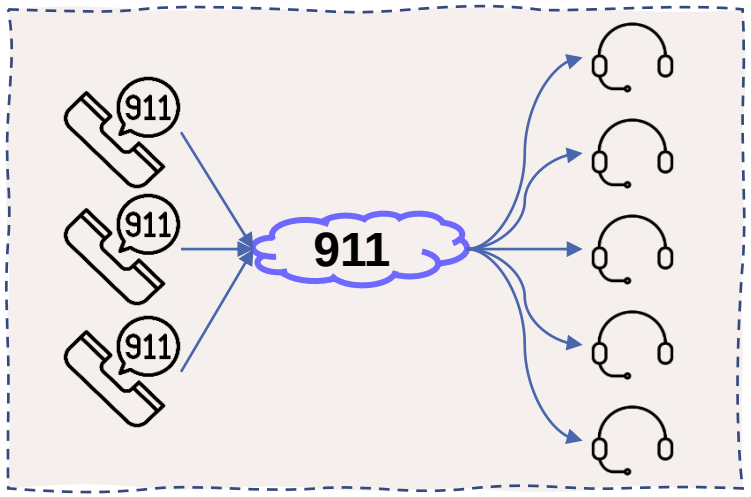
“All pods are able to communicate with each other on a flat, NAT-less network”

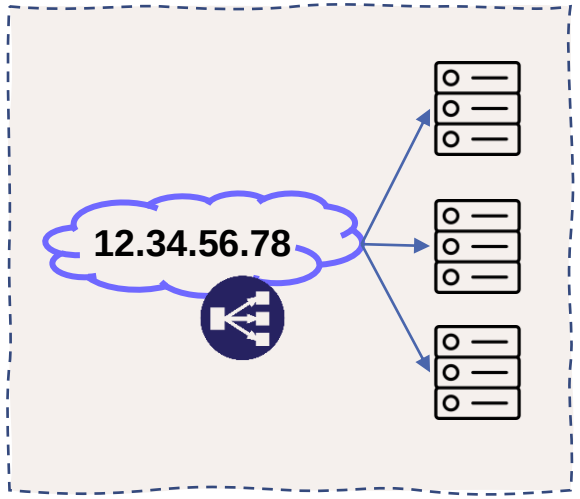
- **Connect to any other pods directly, regardless of where each pod are hosted**
- **K8s does not implement this, only specify CNI (Container Network Interface)**
- **Various network plugins**
  - Flannel
  - Calico
  - Cilium
  - …

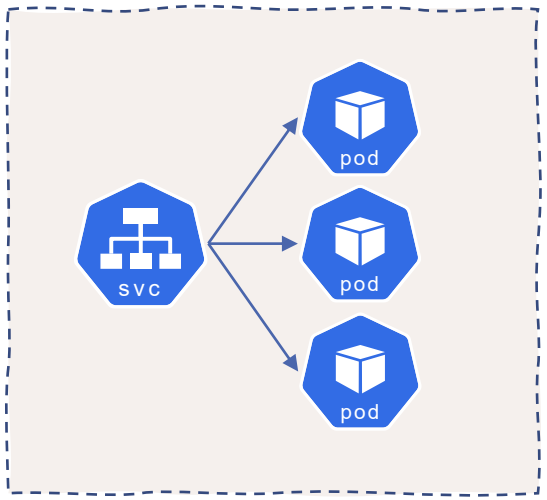


Service Discovery

Service-providing pods are volatile, how can we reliably refer to them?







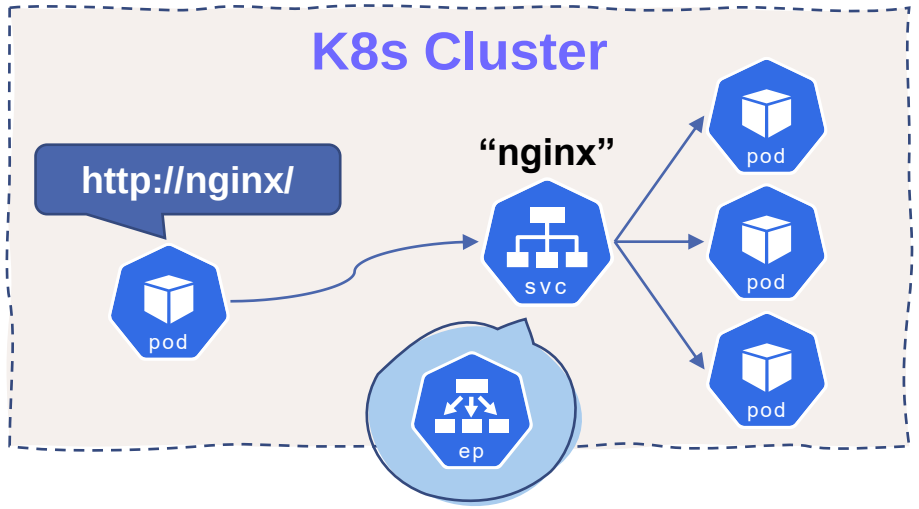
Services

“A persistent abstraction for a group of pods”

- Service can be referred to by its name
- Identify pods through pod label selectors
- Provides high availability & scalability through network means

- 3 types:

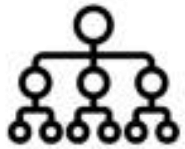
Type	Accessible from
ClusterIP	Within cluster
LoadBalancer	Externally via cloud provider’s load balancer service
NodePort	Externally via a dedicated port on all nodes

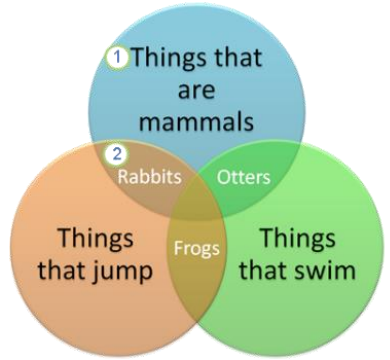


Labels & Selectors

Labels

- **Key/value pairs that attach to objects (Nodes, Pods, ReplicaSets, …)**
- **e.g.**
  - “stage”: “dev”
  - “division”: “icsd”
- **Multi-dimensional representation rather than hierarchical**
- **Can be identified efficiently with label selectors**





Label Selectors

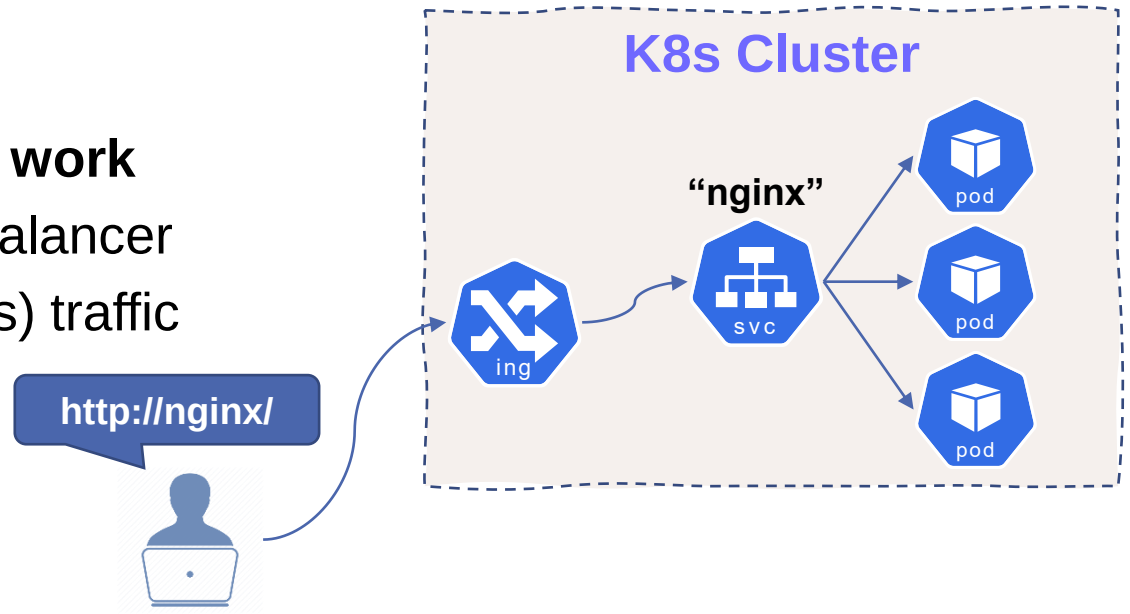
- **Identify a set of objects with their labels**
- **2 types of selectors requirements**

Type	Example
Equality-based	stage = production division != icsd
Set-based	stage notin (dev, uat) division exists

Ingresses

“Provides routing rules to allow external access to services inside K8s cluster”

- **External reachable URLs**
- **Load balancing**
- **SSL termination**
- **Name-based virtual hosting**
- **Require an ingress controller to work**
  - A Layer-7 traffic router / load balancer
  - Typically only forwards HTTP(s) traffic



Lab 2: Expose Pods with *service* & *ingress*

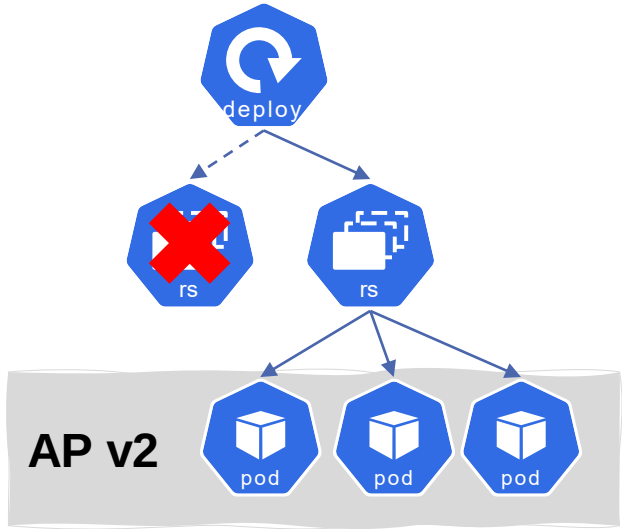
ReplicaSets / Deployments

ReplicaSets

- **Maintain a stable set of replica Pods running at any given time**
- **Link to underlying Pods via Pods' `metadata.ownerReferences` field**

Deployments

- **Provides declarative updates for Pods and ReplicaSets**
- **Scale out/in to accommodate more loads**
  - Triggered by `replicas` changes of a deployment
  - Underlying ReplicaSet will be updated
- **Declare new state of Pods**
  - Triggered by `podTemplate` changes of a deployment
  - A new ReplicaSet is created
  - Pod rollout between versions is controlled by `strategy`



Lab 3: Use *deployment* to manage pods

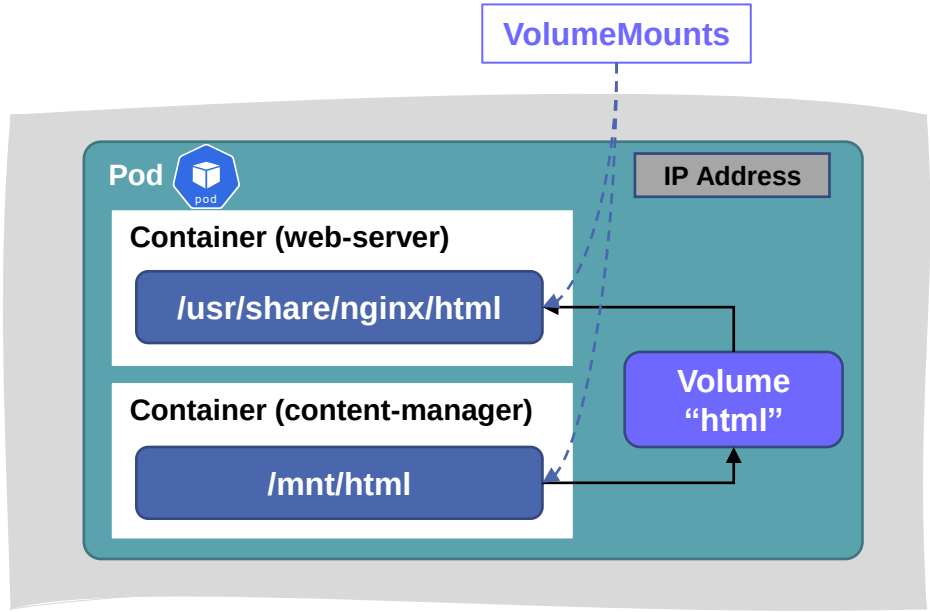
Volumes

“Disk resource which is described as *a part of Pod*”

- **Have the same lifecycle as the Pod**
  - unaffected by individual container restarts

- **Data in volume can be backed by various storage types:**

Backend	Type	Sharable within
Temp (Volatile)	emptyDir	Same pod
Local	hostPath (avoid!) Local	Other pods on the same host/node
Network	GlusterFS NFS gcePersistentDisk AWS EBS azureDisk ConfigMap Secret …	All other pods accessible to these storage



Namespaces

“A mechanism to organize, isolate resources within a cluster ”

- **Separate tenants / authorize access**
  - With RoleBindings!

- **Separate functions logically (virtual sub-clusters)**
- **Enforce resource management policies**
  - With ResourceQuotas / LimitRanges

- **Isolate networks**

  - With NetworkPolicy!

Part 3: Kubernetes Components

(What is it made up of?)

K8s Components

Node:

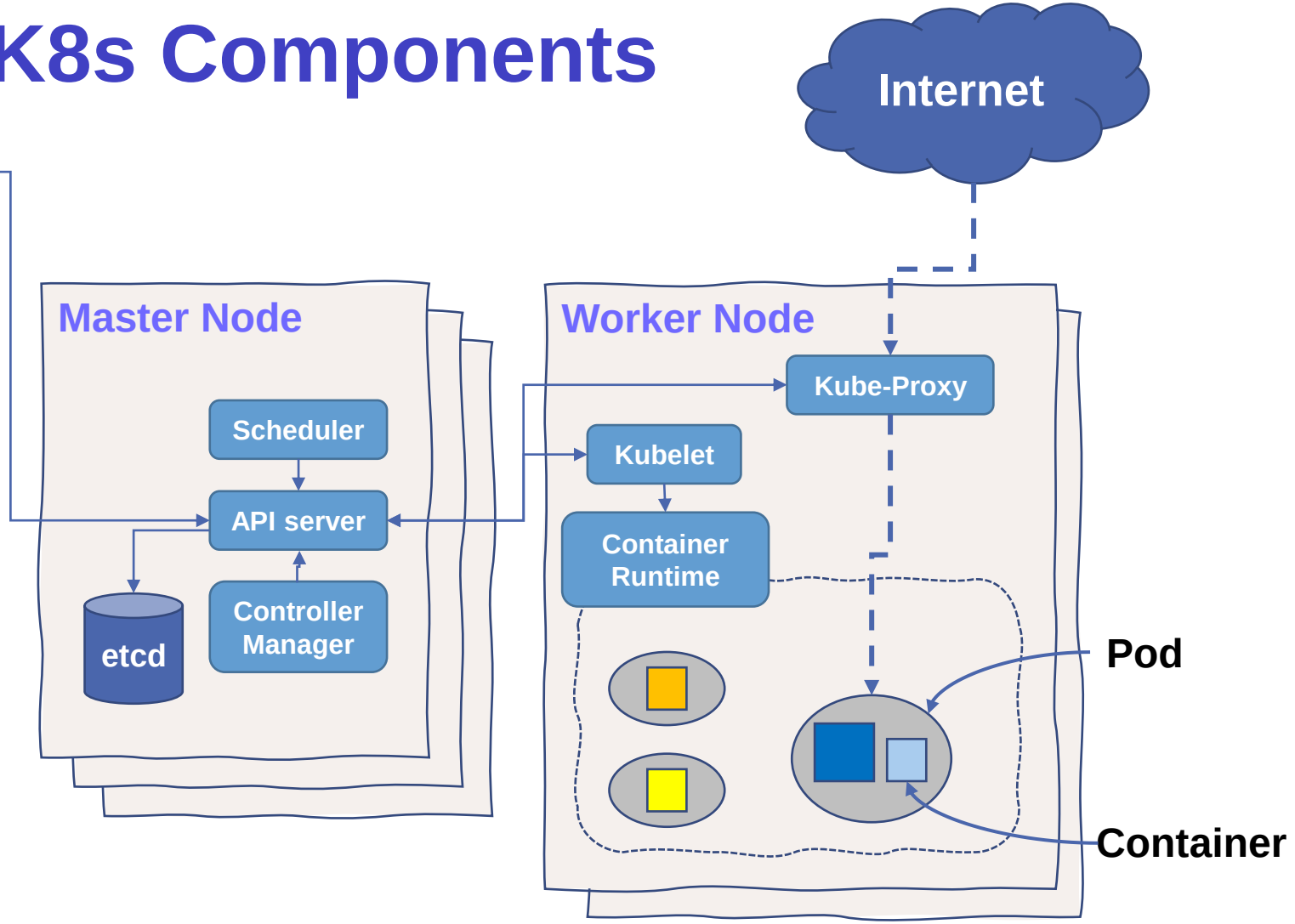
- A machine (VM, Host)

Master node:

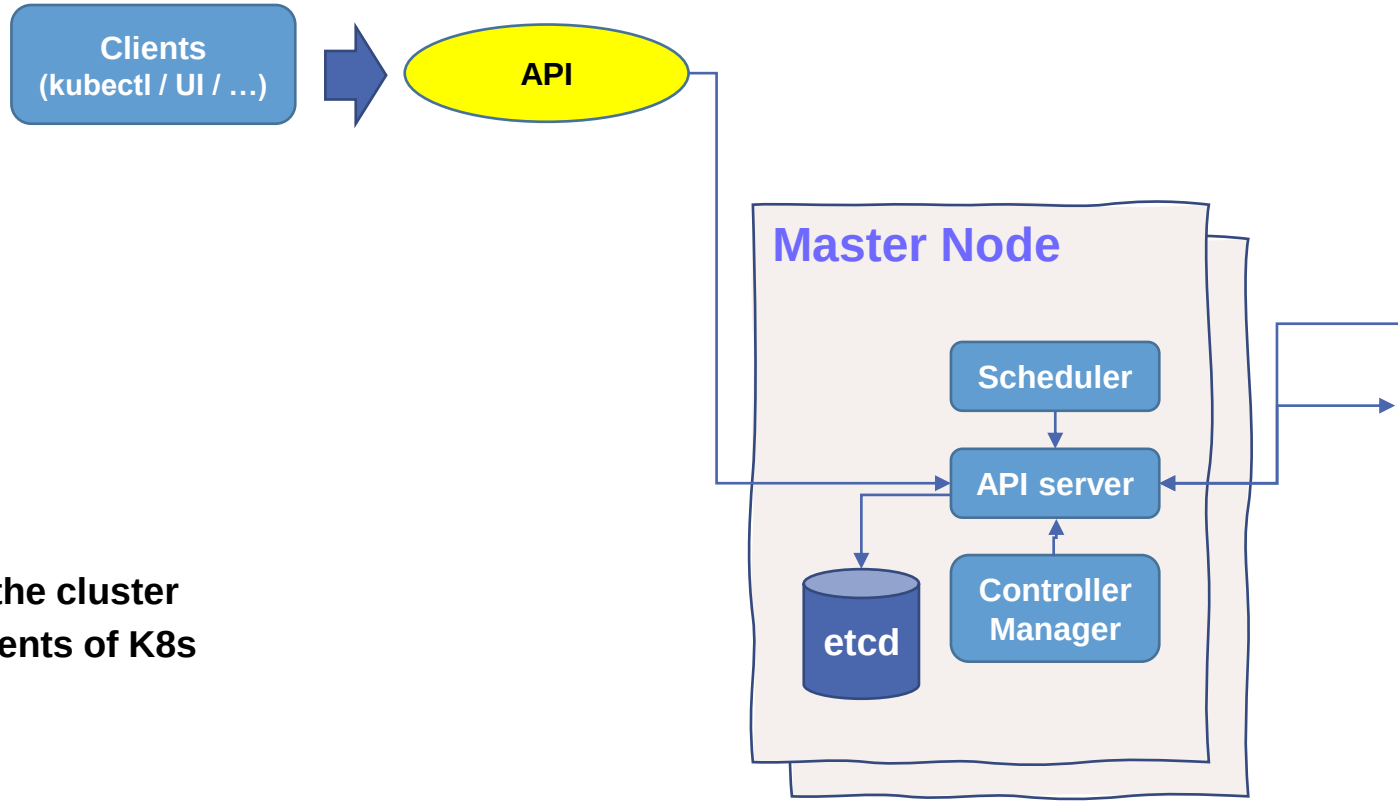
- Runs control plane components for the K8s cluster

Worker node:

- Runs all your applications!



Control Plane Components



API server:

- REST API for all K8s resources
- Store cluster-states in etcd

ETCD:

- Distributed key-value store
- Provides frontend to access shared state of the cluster
- SSOT (single-source-of-truth) for all components of K8s

Scheduler:

- Assign pods to nodes

Controller Manager:

- Distributed key-value store

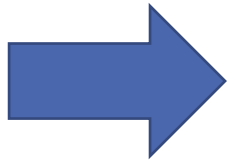
Controller Manager

“Each controller runs its control loop, watches the shared state from the API server and make changes attempting to move the current state closer to the desired state”

Core Controllers:

- **Deployment**
- **ReplicaSet**
- **CronJob**
- **…**

Current State



Desired state

To reduce complexity, core controllers are compiled into a single binary

→ **kube-controller-manager**

Worker Node Components

Kubelet:

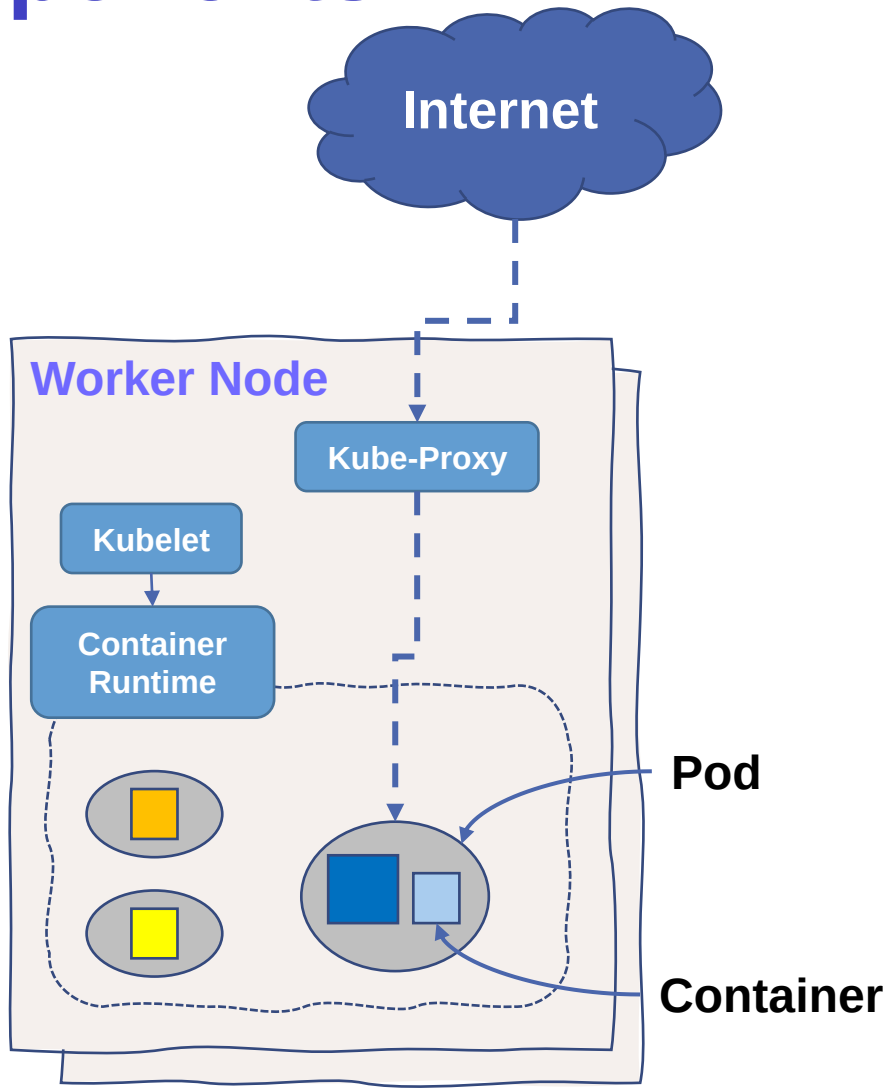
- Register its presence to api-server using “hostname”
- Manages container runtime through CRI (Container Runtime Interface)
- Takes PodSpec (primarily) from API server and ensure those containers are running & healthy
- Updates are reflected to the `status` section of Pod API objects

Container Runtime

- Actually set-up, execute and terminate the containers

KubeProxy

- Setup the networking stack on the node to allow external



Declarative Object Management Model

“When we write YAML, we describe the desired state of the object”

- **Declarative Model (desired-state driven)**
  - What images to use?
  - How many replicas?
  - Do replicas must run on different server racks

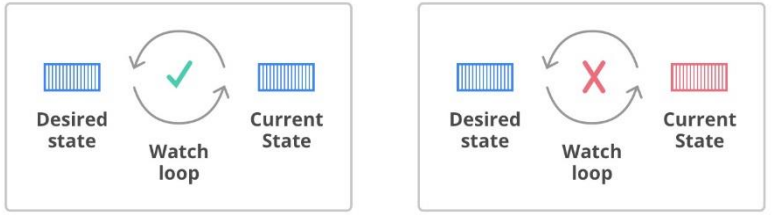
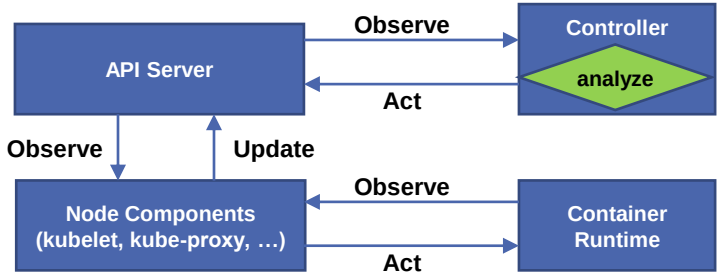


Image Credit: The Kubernetes Book

- **Components observe object states, and act on lower-level resources to achieve the goal.**
- **Eventually consistentcy**

An example:

#	Who	Observe	Action
0	Developer		Increase Deployment replicas by 1
1	Controller-manager	Deployment spec changed	Increase ReplicaSet replicas by 1
2	Controller-manager	# Pods are less then desired replicas	Generate a new pod from the template
3	Scheduler	Found an unscheduled new pod	Pick & assign a “node” for this pod
4	Kubelet	A new pod was assigned to my node	Create containers to match the pod spec



Building K8s On-Premise

Why?

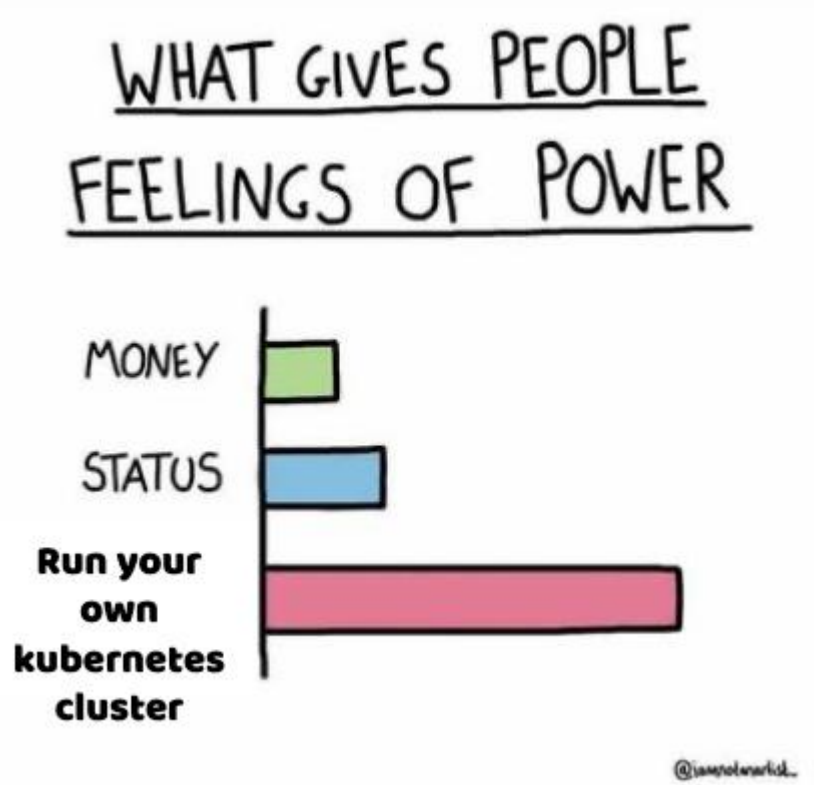
- Compliance & Data privacy
- Avoid lock-in
- Cost

Deployment Tools for K8s Installation

- Kubeadm
- Kops
- Kubespray

Why is it so *hard*?

- Networking
- Availability
- Persistent Storage
- Monitoring
- Upgrades



High Availability & Fault Tolerance

High Availability

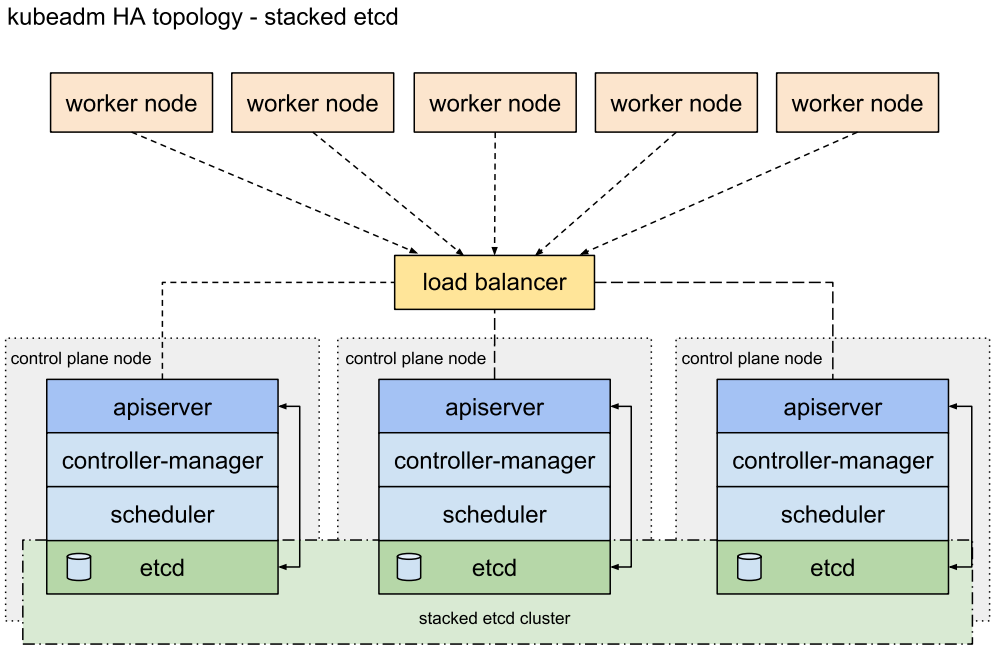
Fault Tolerance

Highly-Available Control Plane

“Worker node failures are covered, how about the control plane?”

- **To tolerate 1-node failure requires (at minimum):**
  - >= 2 instances of apiserver
  - >= 2 instances of controller-manager
  - >= 2 instances of scheduler
  - >= **3-member** etcd cluster

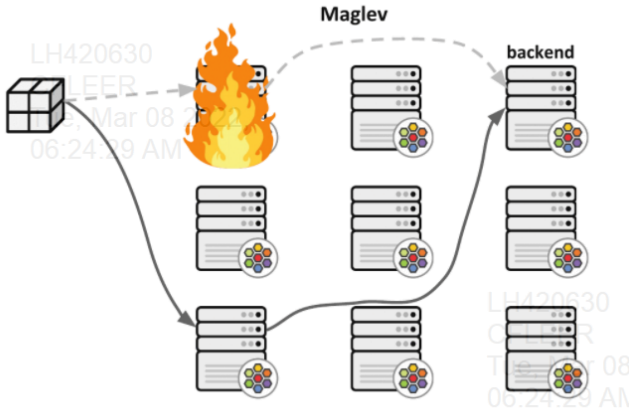
Avoid “split-brain” condition



Build for High Availability & Fault Tolerance

“Anything that can go wrong will go wrong”

- **Single host/node down (disk, memory, connectivity, …)**
  - ✔ K8s has it covered 😁
  - ⚠ On failure: Pod is re-created on another node



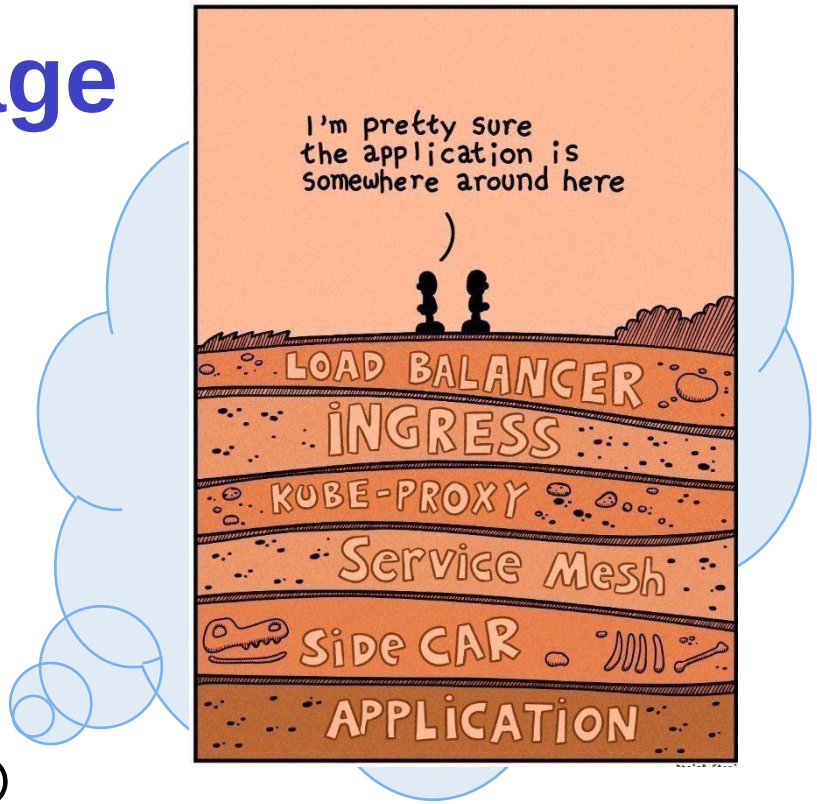
https://cilium.io/blog/2020/11/10/cilium-19

- **Load-balancer partial failure**
  - ✔ Use DSR (Direct Server Return) with Maglev load-balancing
  - ⚠ On failure: Nearly 100% of TCP connections are unaffected by load-balancer failures

- **Single datacenter failure**
  - ✔ 3-site architecture
  - ⚠ On failure: Loses 33% of capacity, but control plane is not affected!

- **Single control plane failure**
  - ✔ Multi-cluster with inter-connected network & service mesh
  - ⚠ On failure: Loses 50% of capacity, but we can switch to another control-plane and survive!

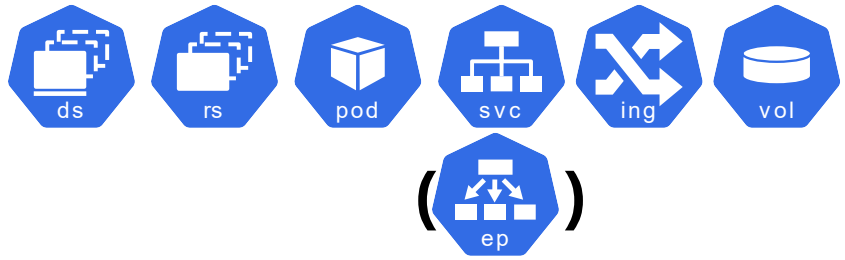
Takeaway Message



- IaC and Kubernetes is a natural paring!
  - Desired state driven
  - Reduce inconsistencies / human error
  - Faster / automated recovery
  - Make our life simpler! 😋
- Keep everything simple
  - Abstraction make things simple
  - Complex abstractions also make thing harder to trace 🙃
- Hope for the best, but prepare for the worst
  - K8s will try its best, but don't rely too much on it (https://k8s.af/)
  - Monitoring & design for failure are still important!

Homework Lab

- (60%) Design an application that make use of all these resources:



- (30%) Draw a graph describing how these resources are inter-connected in your app.

- (10%) Also try to make use of at least 2 other K8s resources types!

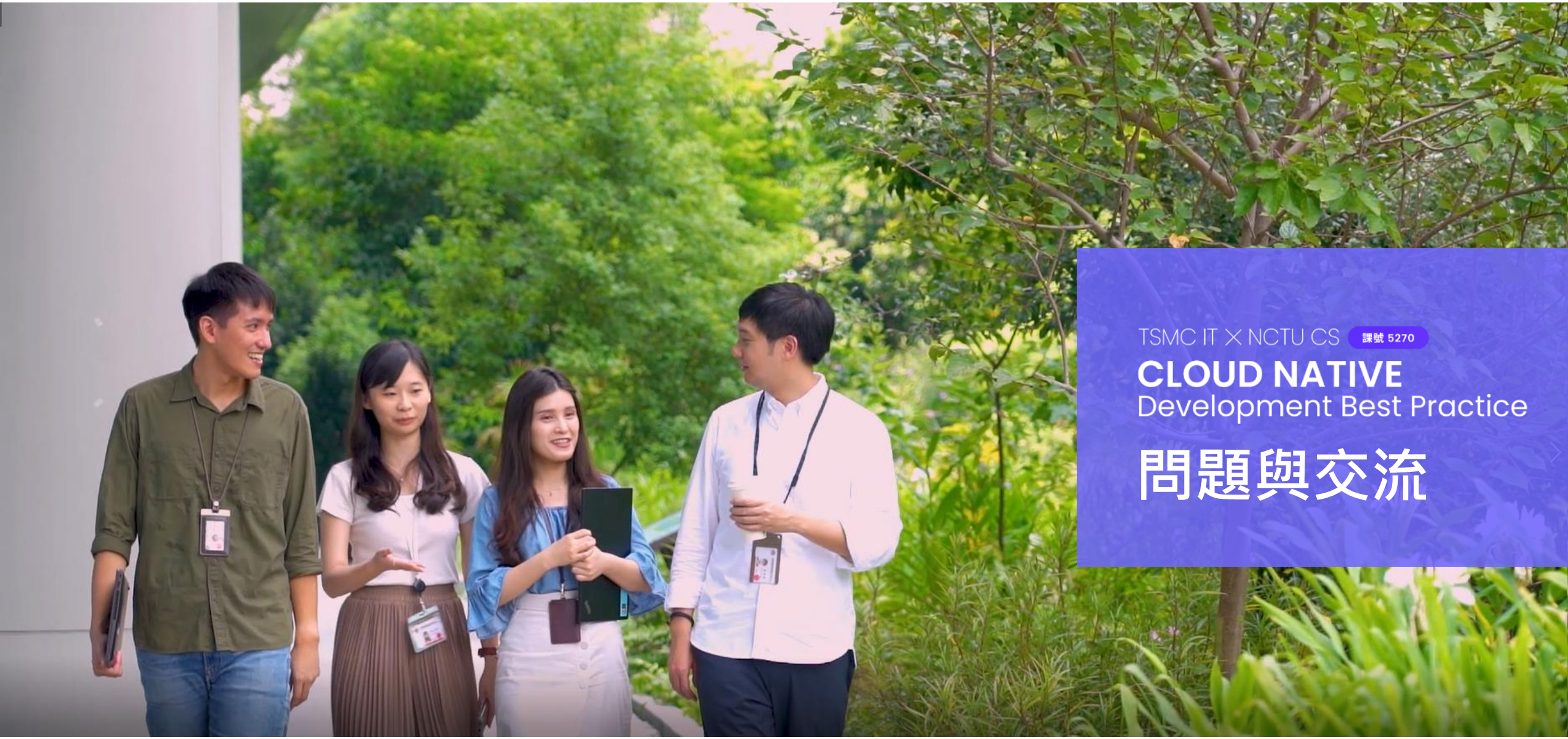
References

Phippy and friends: https://www.cncf.io/phippy/

Smooth Sailing with Kubernetes:
https://cloud.google.com/kubernetes-engine/kubernetes-comic

Building K8s (The hard way):
https://github.com/kelseyhightower/kubernetes-the-hard-way

- For those who are interested in how components are put together
- Or, maybe you want to get a taste of how production-grade K8s are built



TSMC IT × NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

問題與交流


TSMC IT X NCTU CS
課號 5270
CLOUD NATIVE
Development Best Practice
THANK YOU
FOR YOUR
ATTENTION