Kubernetes Quickstart Workshop

Discover core Kubernetes concepts as we learn about different resource types while investigating the modern infrastructure problems they solve. This workshop features guided theory and interactive elements with a live cluster to explore how to engineer and operate applications in Kubernetes.

Let’s talk about…

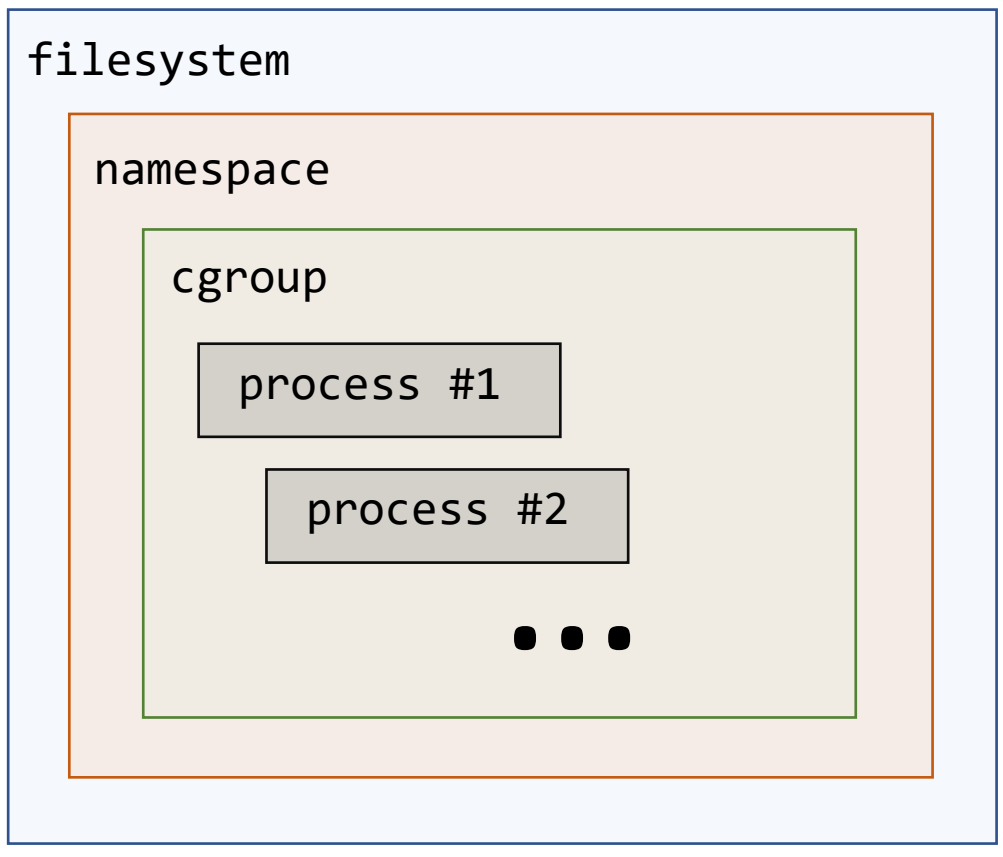
- What are containers?
  - I’ve got 99 ~~problems~~ containers
- Kubernetes
  - Control Plane
  - Cloud native WordPress
  - Running Containers [Pods, Deployments, Sets]
  - Storage [Persistent Volumes & Claims]
  - Application Config [ConfigMaps]
  - Application Secrets [Secrets]
  - Networking [Services]
- Tooling

What are containers?

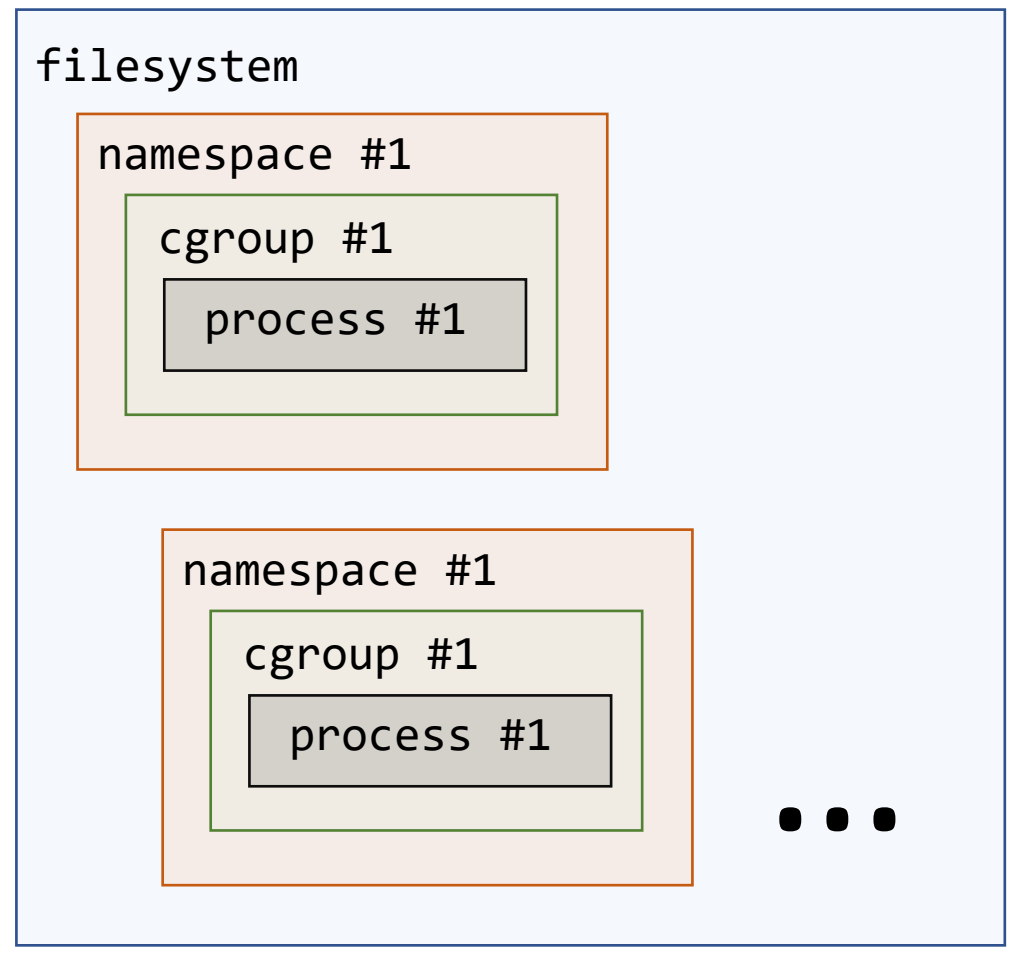
An Isolated Process Sandbox.

- Process(es) that share the same:
  - **Process Namespace**
    - Scoping access to only those resources in that namespace
  - **Process cgroup**
    - Restricting resource consumption by process
- Lifecycle coupled

Typical Unix Filesystem



Contained processes on Unix Filesystem



Images are like Onions

Executable code that creates a container.

- Layers of dependencies
- Immutable
- Commonly built with **Dockerfile**
  - Programmatic image building

Dockerfile example

```
1 | FROM ubuntu:20.04 # Base                    <- Base image
2 | RUN apt-get update -y \                     <- Install pre-requisites
3 | && apt install -y python3 g++ make
4 | WORKDIR /app                                <- Define working directory
5 | COPY . .                                    <- Copy files from build machine to image
6 | RUN yarn install –production                 <- Execute arbitrary command/script
7 | CMD [“node", "src/api.js"]                   <- Start my process
8 | EXPOSE 8080                                 <- Tell container runtime the port to
                                                    listen for traffic
```

What are containers?

I’ve got 99 ~~problems~~ containers

Containers help:

- Make our application runtime portable and independent
- Provide light-weight isolation

Container Images help:

- Package application for hosting/development/sharing
- Identify/encapsulate exact dependencies

Dockerfile (or other image build script) help us:

- Codify/automate image building

Kubernetes

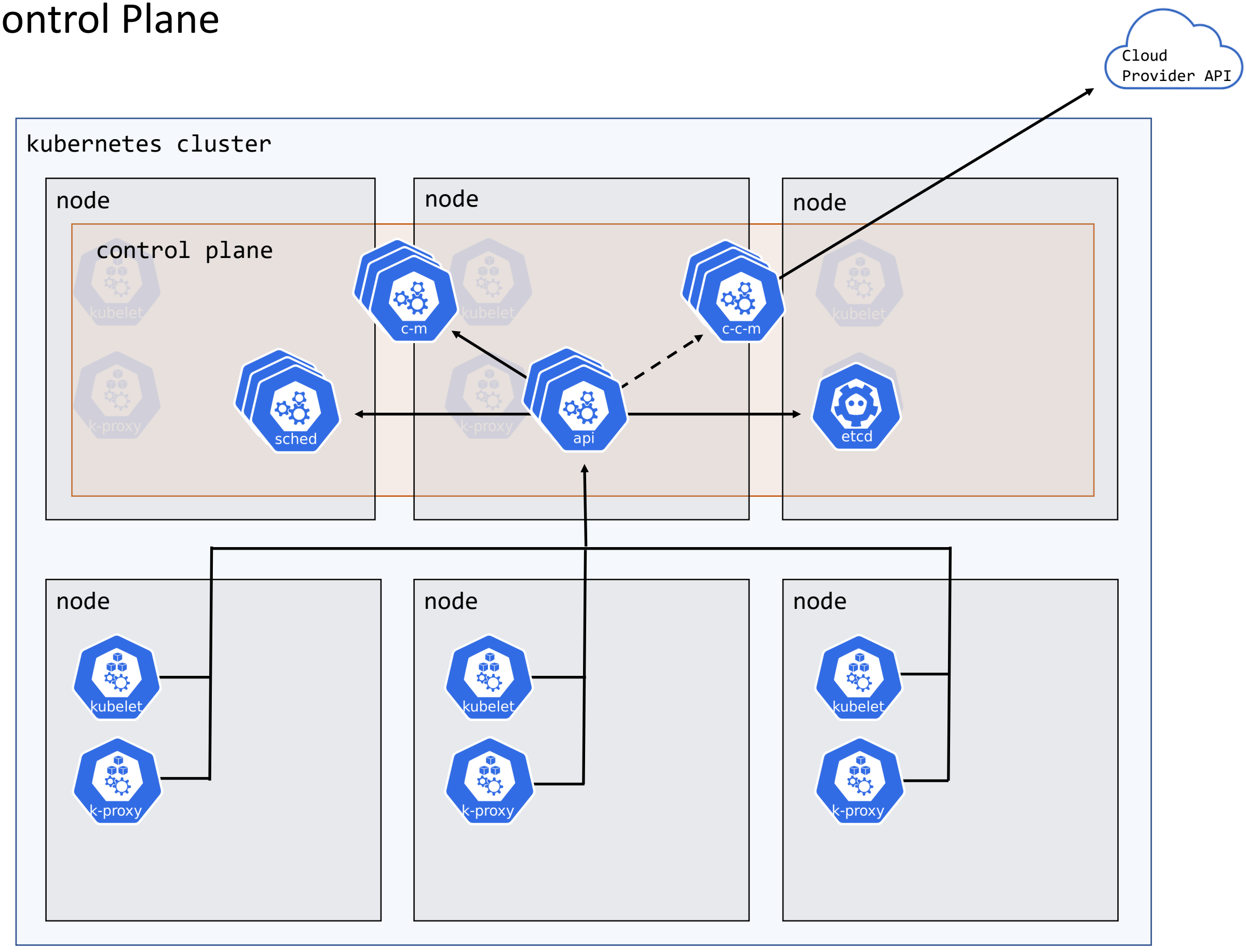
Open-source system for automating deployment, scaling, and management of containerized applications.

Kubernetes Features include:

- Automated rollouts and rollbacks
- Service discovery and load balancing
- Storage orchestration
- Secret and configuration management
- Automatic bin packing
- Batch execution
- IPv4/IPv6 dual-stack
- Horizontal scaling
- Self-healing
- Designed for extensibility

Kubernetes

Control Plane



**legend on next page*

Kubernetes

Control Plane

kube-apiserver
Exposes Kubernetes API. Front-end of Control Plane.

etcd
Kubernetes cluster data store.

kube-scheduler
Intelligently runs pods on nodes.

kubelet
Agent running on each node, ensures containers are running in pods.

kube-controller-manager
Control loop, enforces cluster state.

kube-proxy
Agent running on each node, maintains network rules facilitating pod networking.

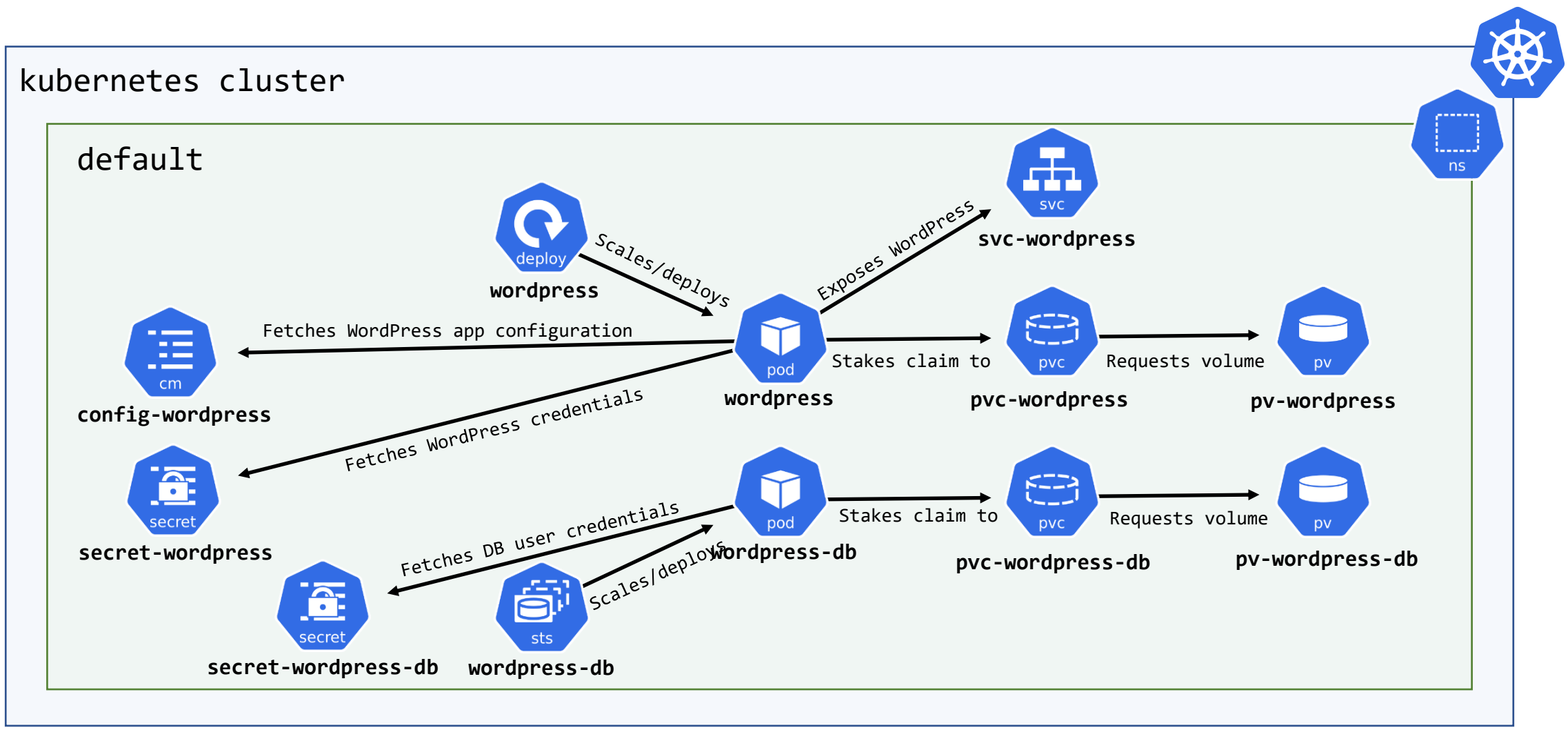
cloud-controller-manager
Links cluster to cloud provider API. Optional.

Kubernetes

Cloud native WordPress

open-source content management system written in PHP and paired with a MySQL or MariaDB database

Let’s explore the different components in a cloud-native WordPress architecture.



Kubernetes

Running Containers | **Pods, Deployments, Sets**

Pod 

- Workload; one or more containers with shared compute, network, and storage
- Schedules on a node and runs to completion (or indefinitely)
- Mostly mutable
- Multiple instances of a pod are managed by some **Controller**

Deployment 

- Desired state for one or more pods
- Abstracts some `set`

Stateful Set 

- Maintains a consistent identity for pods and their resources

Kubernetes

Storage | **Persistent Volumes & Claims**

Persistent Volume 

- Unit of storage
- Provisioned in one of two ways:
  - **Static** – created by a Cluster Administrator, identify some real, existing storage
  - **Dynamic** – created automatically as dictated by a **StorageClass** when requested

Persistent Volume Claim 

- Claim to a unit of storage
- Binding with different access modes
  - e.g. `ReadWriteOnce, ReadOnlyMany, ReadWriteMany,`

Kubernetes

Application Config & Secrets | **ConfigMaps & Secrets**

Config Map 

- Non-confidential key/value store
- Can be consumed in pods as:
  - Command line arguments
  - Configuration flat files
  - Environment variables
- Data in a ConfigMap must be:
  - UTF-8 strings if defined using `data`
  - Base64-encoded strings defined using `binaryData`
  - <1MB in size
- Decouple application config from containers

Secret 

- Sensitive key/value store
- Data in a Secret must be:
  - Base64-encoded strings defined using `data`
  - UTF-8 strings if defined using `stringData`
- Otherwise, akin to `ConfigMaps`

Kubernetes

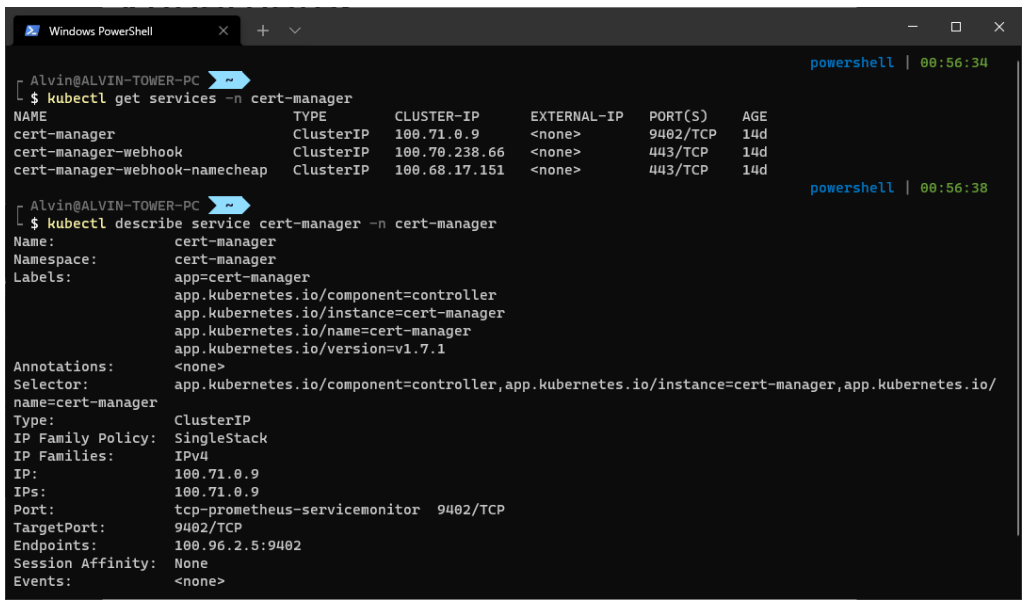
Networking| **Services**

Service 

- Exposes a set as a network service
  - Pods come and go, and with them their IP
  - Services are routable internally to the cluster via DNS
- Targets pods by `label`
  - `port` - arbitrary port for traffic inbound from node
  - `targetPort` - actual port used by app container, routes traffic from `port` to `targetPort`
  - `name` - unique string to identify port
- Services can abstract application(s) outside of the cluster
- **Service types**
  - `NodePort` – Exposes port on node
  - `LoadBalancer` – External load balancers
  - `ExternalName` – Routes to service by DNS (CNAME)

Tooling

kubectl | **Command-line utility**



k8s Lens | **Operations/IDE**