Kubernetes - Day 02

Learning Objectives

- Kubernetes basics
- Introduction to YAML
- Pods
- Pod create steps
- Pod Lifecycle
- Pod and Container status

Running Applications in Kubernetes

- Pattern to follow for running applications on a Kubernetes cluster:
  - Write the application as small independent microservices
  - Package each microservice in its own container
  - Wrap each container in its own Pod
  - Deploy Pods to the cluster via controllers such as;
    - Deployment
    - StatefulSet
    - DaemonSet
    - Job
    - Cronjob
    - Namespace
    - Service

Declarative model & Desired state

- **Desired state** is at the very heart of Kubernetes
  - Declared in the application manifest file
  - POST(ed) to the API server
  - Stored in the cluster store
  - Implemented on the cluster
  - Watch loops make sure the current state of the application doesn't vary from t
    state

Kubernetes Clients

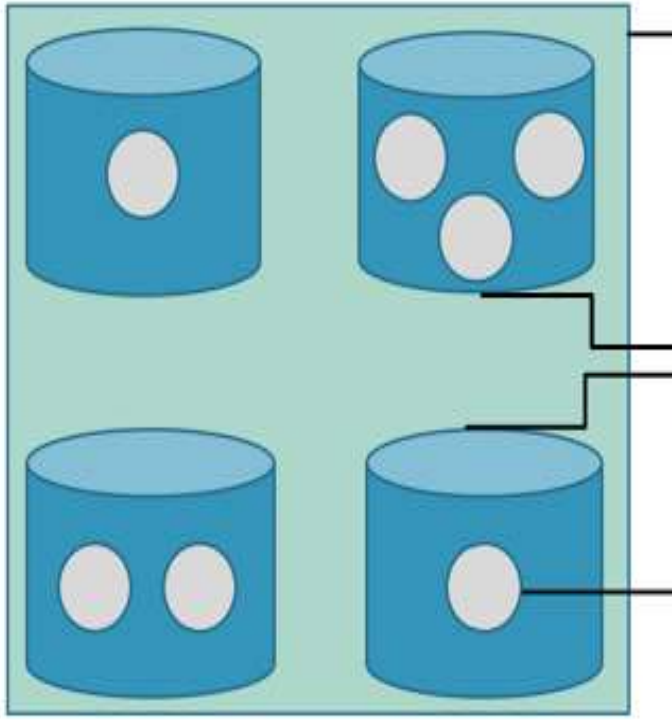
- kubectl:
  - A command line tool used for creating/deleting/retrieving Kubernetes resource
  - Converts user-friendly commands into the JSON payload required by the API ser
  - To POST commands, it references a config file for cluster / API server endpoint
- API client libraries:
  - Available for programming languages, used to interact with a Kubernetes cluste programmatically.
  - Provide a more flexible way to interact with Kubernetes resources and can be u custom automation and integration tools
- Dashboard:
  - web-based Kubernetes user interface to deploy, troubleshoot your containerize manage the cluster resources
  - https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

Kubernetes objects and resources

- Objects are persistent entities and represent the status (actual state) of the cluster
  - Eg: Pod, Service, Volume, Deployment
- A "record of intent" (desired state), Kubernetes system will constantly work to ensure t object exists
- Resource is an endpoint in the Kubernetes API that stores a collection of API objects of
  - Eg: the built-in pods resource contains a collection of Pod objects.

What is a Pod

- Pod
  - Smallest object in K8s
  - Organize application into pods
  - Environment for containers
  - Kubernetes runs containerized apps
  - Cannot run a container directly on a Kubernetes cluster – containers must always run inside of Pods
  - At high level, a Pod is just a wrapper that allows a container to run on a Kubernetes cluster



Features of Pod

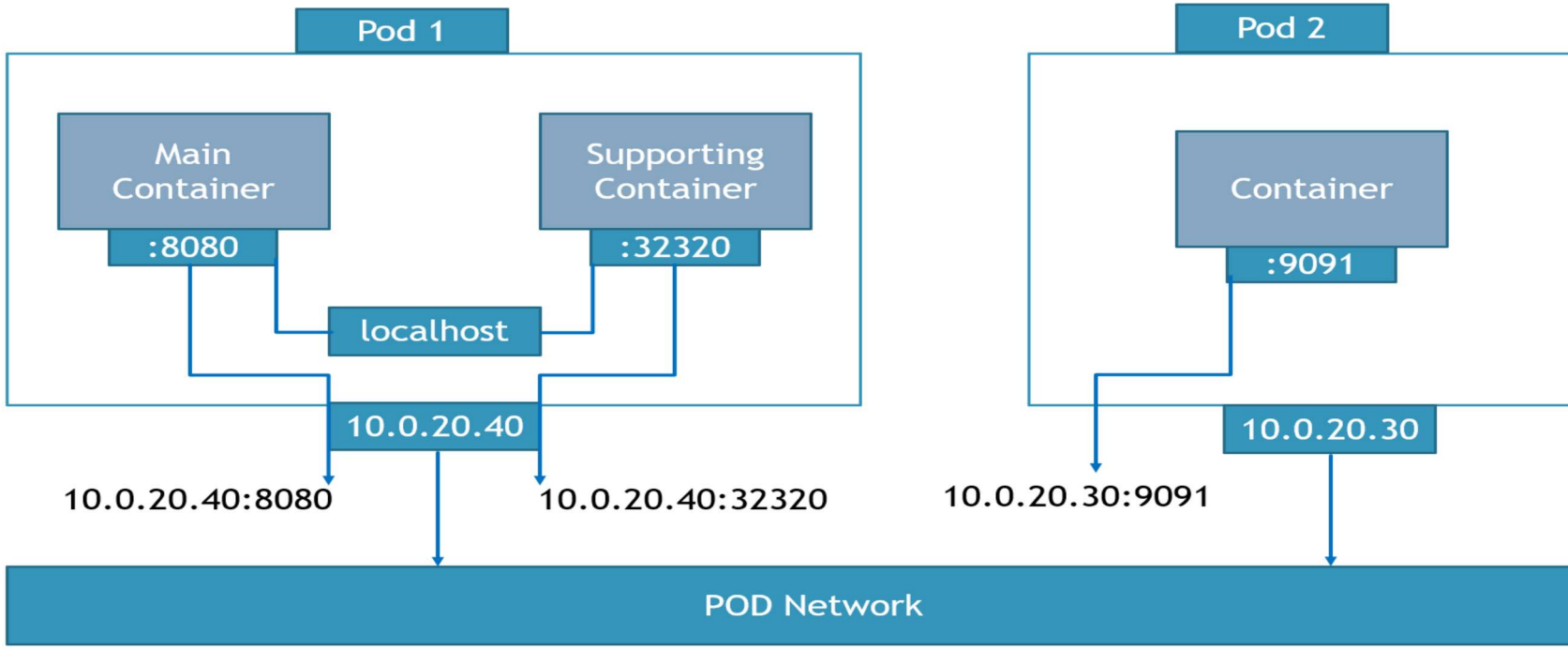
- Pods augment containers
  - Labels and annotations
  - Restart policies, Resource requests and limits
  - Probes, Affinity and anti-affinity rules
  - Termination control, Security policies

- Pods assist in scheduling
  - Pods are also the minimum unit of scheduling in Kubernetes.
  - If you need to scale your app, you add or remove Pods.
  - You do not scale by adding more containers to an existing Pod

- Pods enable resource sharing
  - Provide a shared execution environment for one or more containers.
  - Includes things such as:
    - Shared filesystem, network stack (IP address and ports…), memory, volumes

Single and Multi-container Pods

- Pods in a Kubernetes cluster are used in two main ways:
  - Pods that run a single container.
  - Pods that run multiple containers that need to work together
    - Application composed of multiple co-located containers that are tight coupled and need to share resources



Shared volumes

- Multiple containers in a pod access shared data using Kubernetes Volume
- Uses a directory on the host that is shared with all containers within a Pod
- EmptyDir
  - Enables data to survive container restarts, but these volumes have the same lifetim
- HostPath
  - Volumes are not deleted when the pod is terminated, data stored in them can pers
    the lifetime of the pod
- Use case:
  - Multi-container Pod with a shared Volume having one container writes logs or other
    shared directory, and the other container reads from the shared directory

YAML in Kubernetes

- What is YAML?
  - Yet Another Markup Language or YAML ain't markup language
  - Data serialization language that is often used for writing configuration files
  - Popular because it is human readable and easy to understand
  - Used in conjunction with other programming languages

- Why is YAML in k8s?
  - Convenience: Adding parameters to the command line is not required
  - Maintenance: Can be added to source control to track changes
  - Flexibility: Create much more complex structures than on CLI

YAML details

- YAML Basic Rules – Syntax
  - YAML files extension is .yml or .yaml
  - YAML is case sensitive
  - YAML does not allow the use of tabs instead it uses 2 spaces
  - List members are denoted by a leading hyphen (-)
  - Associative arrays are represented using the colon space (:) in the form key: value

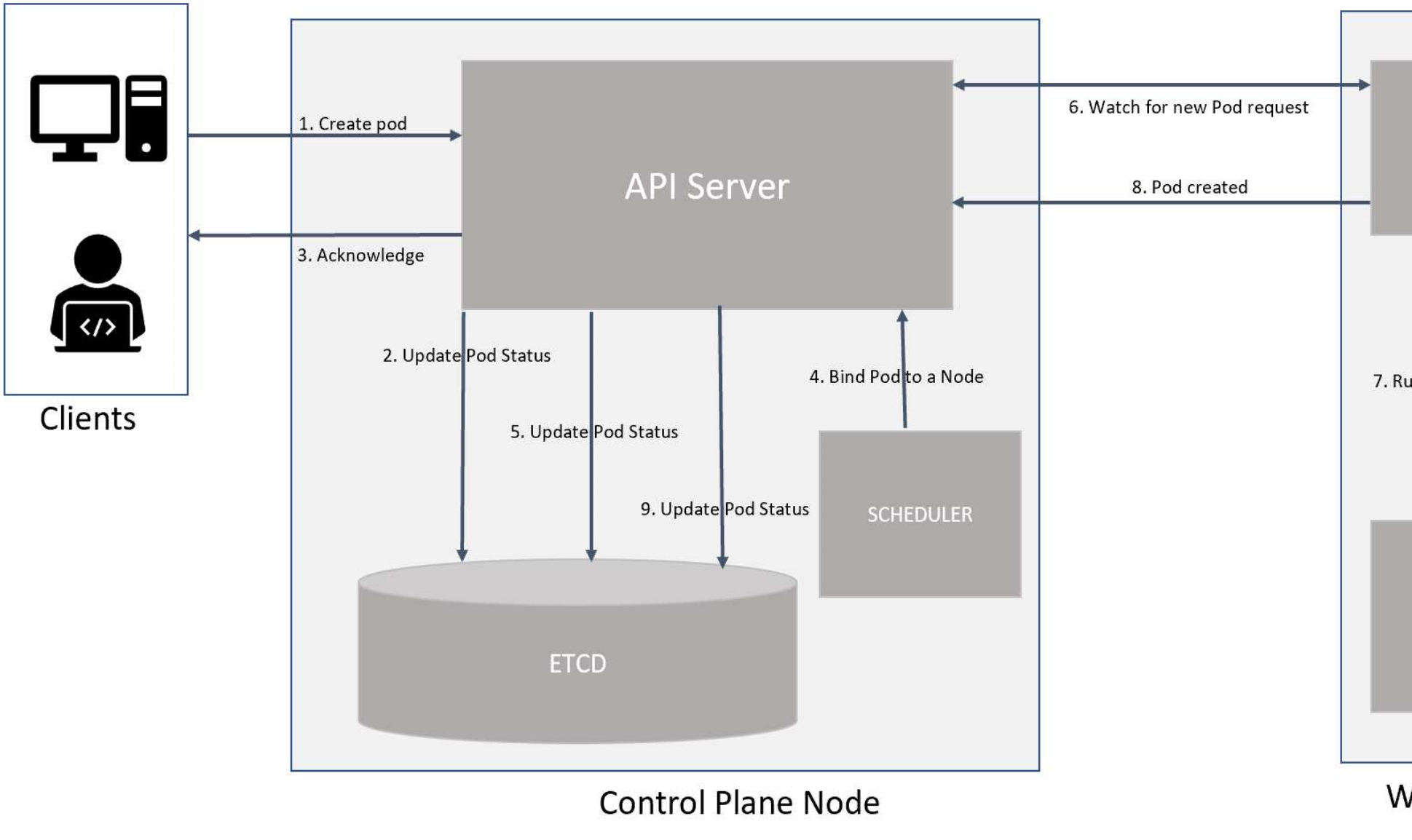
Yaml Example – Pod Manifest

```
1  apiVersion: v
2  kind: Pod
3  metadata:
4    name: nginx
5  spec:
6    containers:
7    - image: ng
8      name: ngi
```

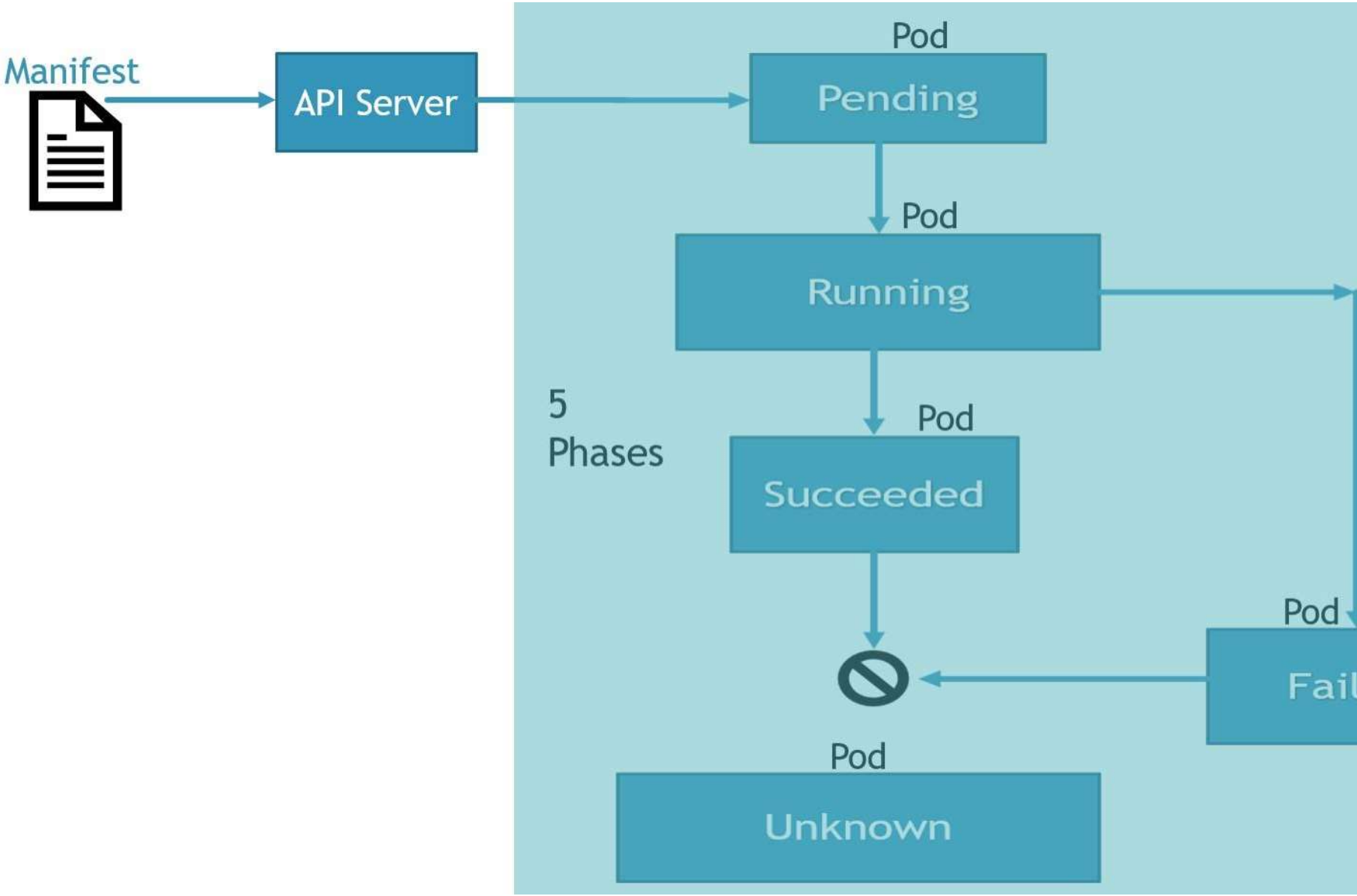
Pod Manifest

- top-level elements of yaml:
  - **apiVersion**: Specifies API group and the API version
    - Pods are in the core API group which is special, as it omits the API group name, describe them in YAML files as just v1
    - Examples:
  - **kind**: type of object being deployed
  - **metadata**: identify the object in the cluster and create loose couplings between di objects
  - **spec**: define the containers that will run in the Pod

Create Pod

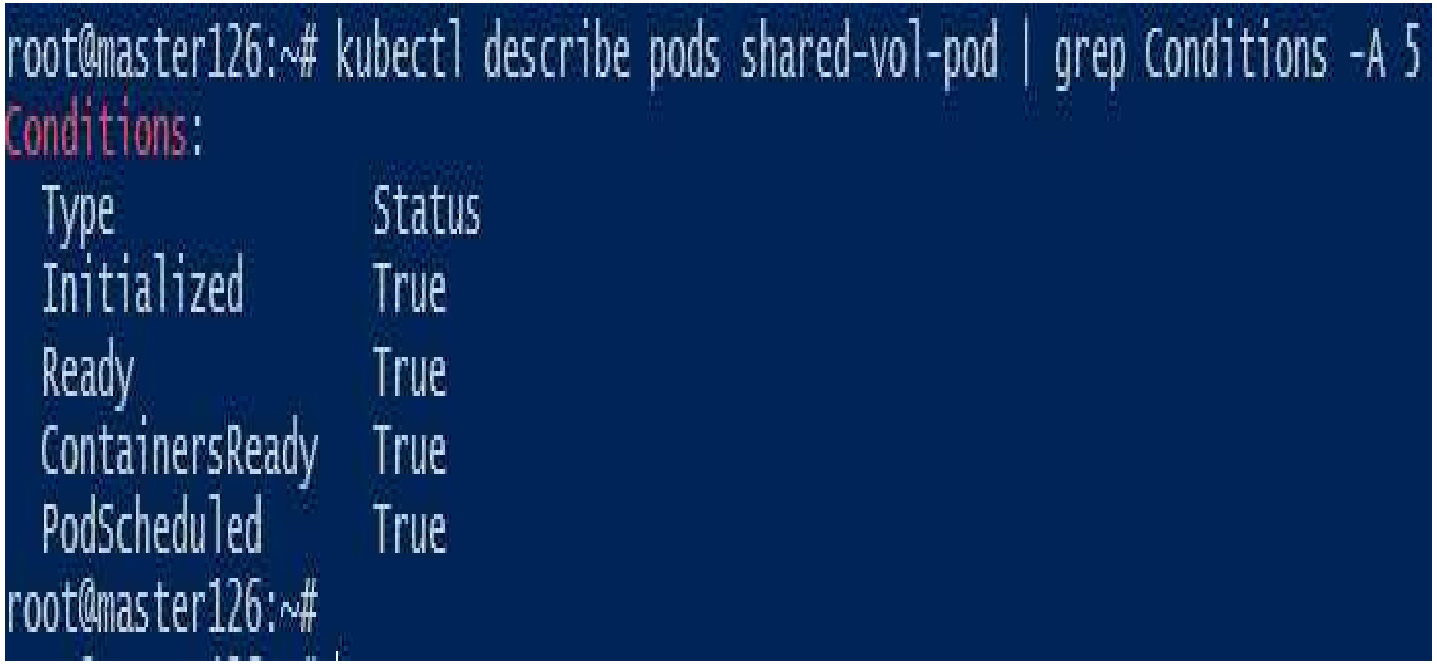

Clients
1. Create pod
3. Acknowledge
API Server
2. Update Pod Status
5. Update Pod Status
9. Update Pod Status
4. Bind Pod to a Node
SCHEDULER
ETCD
Control Plane Node
6. Watch for new Pod request
8. Pod created
7. Ru
W

Pod Lifecycle



Pod Conditions

- Pod-Status has an list of Pod-Conditions through which the Pod has or has not passed:
  - **PodScheduled**: the Pod has been scheduled to a node.
  - **ContainersReady**: All containers in the Pod are ready.
  - **Initialized**: All init containers have started successfully.
  - **Ready**: Pod is able to serve requests and should be added to the load balancing pools of all m Services.



Container State

- 3 possible container states:
  - Waiting
  - Running
  - Terminated