Kubernetes Fundamentals

Shailendra Chauhan

Microsoft MVP, Founder & CEO – Dot Net Tricks

DotNetTricks

Agenda

- Introduction to Kubernetes
- Docker Swarm vs. Kubernetes
- Kubernetes Architecture
- Configuring Kubernetes
- YAML
- Pod
- Service
- Deploy ASP.NET Core App to Kubernetes

DotNetTricks

Introduction to Kubernetes

- Kubernetes is an open-source container management (orchestration) tool.
- Simplify container deployment, scaling of containers & container load balancing.
- Kubernetes can run on-premises bare metal, OpenStack, public clouds Google, Azure, AWS, etc.
- ***Note**: Kubernetes is not a containerization platform. It is a multi-container management solution.*

Kubernetes Features

- Automated Scheduling
- Self-Healing Capabilities
- Automated rollouts & rollback
- Horizontal Scaling & Load Balancing
- Offers enterprise-ready features
- Application-centric management
- Auto-scalable infrastructure

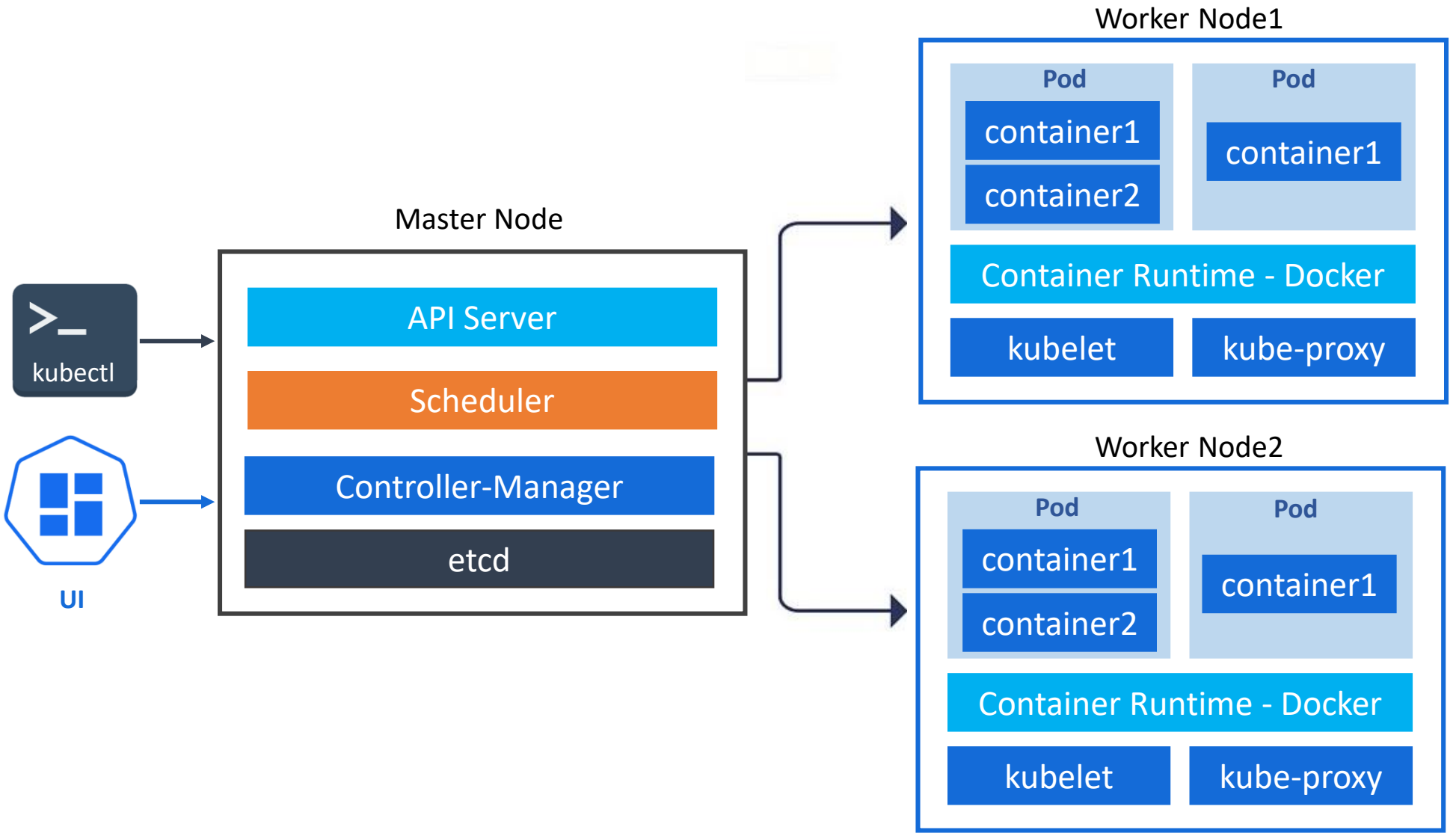
Kubernetes Disadvantages

- Kubernetes can be an overkill for simple applications
- Kubernetes is very complex and can reduce productivity
- The transition to Kubernetes can be complicated

Docker Swarm vs. Kubernetes

Parameters	Docker Swarm	Kubernetes
Scaling	No Autoscaling	Auto-scaling
Load balancing	Does auto load balancing	Manually configure your load balancing settings
Storage volume sharing	Shares storage volumes with any other container	Shares storage volumes between multiple containers inside the same Pod
Use of logining and monitoring tool	Use 3rd party tool like ELK	Provide an in-built tool for logging and monitoring.
Installation	Easy & fast	Complicated & time-consuming
GUI	GUI not available	GUI is available
Scalability	Scaling up is faster than K8S, but cluster strength not as robust	Scaling up is slow compared to Swarm, but guarantees stronger cluster state Load balancing requires manual service configuration
Load Balancing	Provides a built-in load balancing technique	Process scheduling to maintain services while updating
Updates & Rollbacks Data Volumes Logging & Monitoring	Progressive updates and service health monitoring.	Only shared with containers in same Pod Inbuilt logging & monitoring tools.

Kubernetes Architecture



Master Node Components

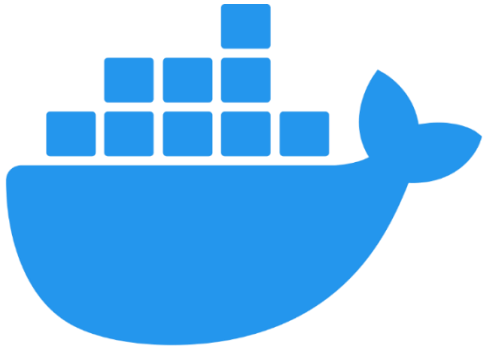
- **API Server -** The API server exposes a REST interface to the Kubernetes cluster. All operations against pods, services, and so forth, are executed programmatically by communicating with the endpoints provided by it.
- **Scheduler -** The scheduler is responsible for assigning work to the various nodes. It keeps watch over the resource capacity and ensures that a worker node's performance is within an appropriate threshold.
- **Controller manager -** The controller-manager is responsible for making sure that the shared state of the cluster is operating as expected.
- **Etcd -** A distributed key-value store that Kubernetes uses to share information about the overall state of a cluster. The master queries etcd to retrieve various parameters of the state of the nodes, pods and containers.

Worker Node Components

- **Kubelet**: gets the configuration of a Pod from the API server and ensures that the described containers are up and running.
- **Container Engine/Runtime**: Responsible for container management like pulling images, starting/stopping containers. Usually Docker is used for container runtime.
- **Kube-proxy**: Kube-proxy acts as a load balancer and network proxy to perform service on a single worker node
- **Pods**: A pod is a combination of single or multiple containers that logically run together on nodes

Configure Kubernetes

Minikube



Docker Desktop

Kubeadm

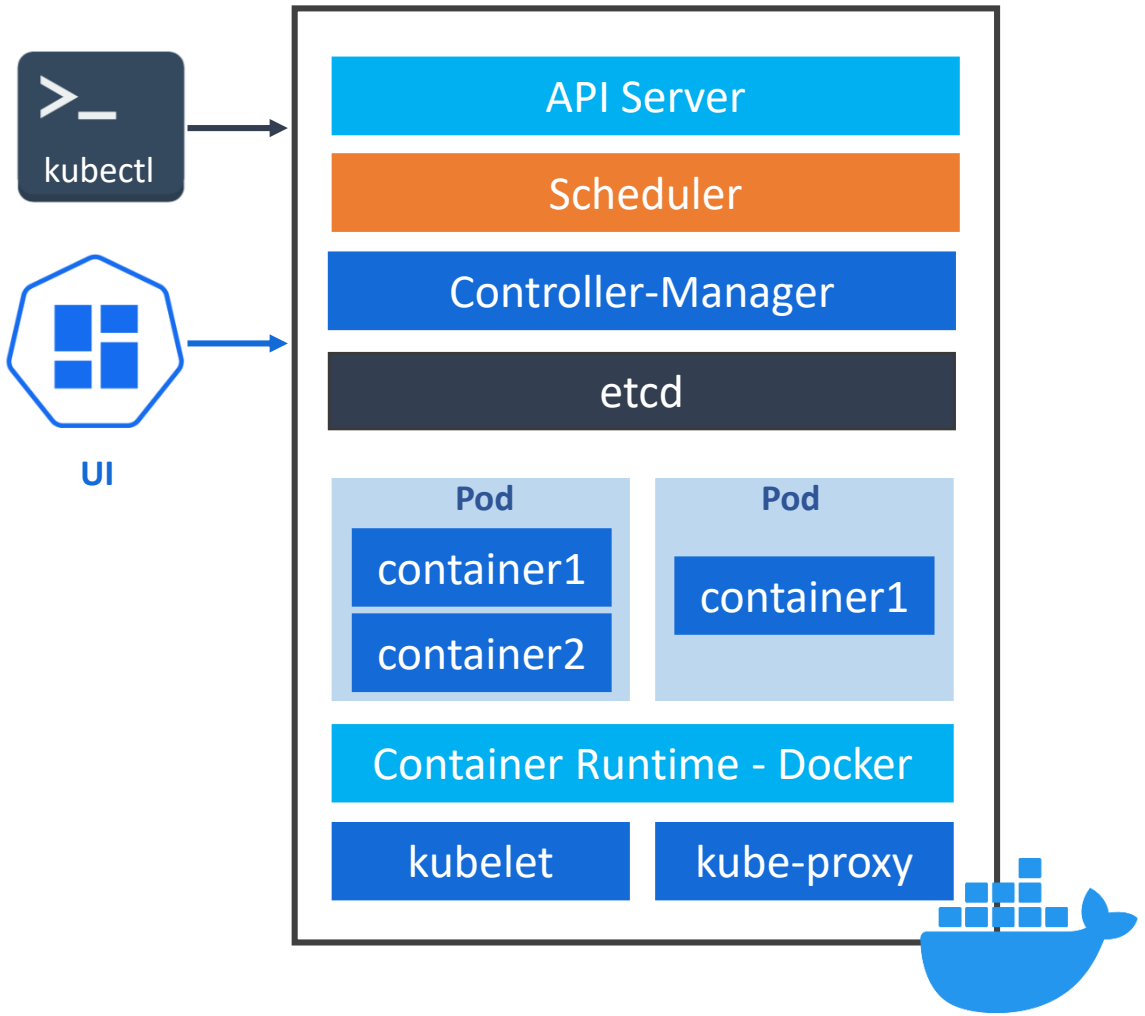
Cloud

Single Node Cluster

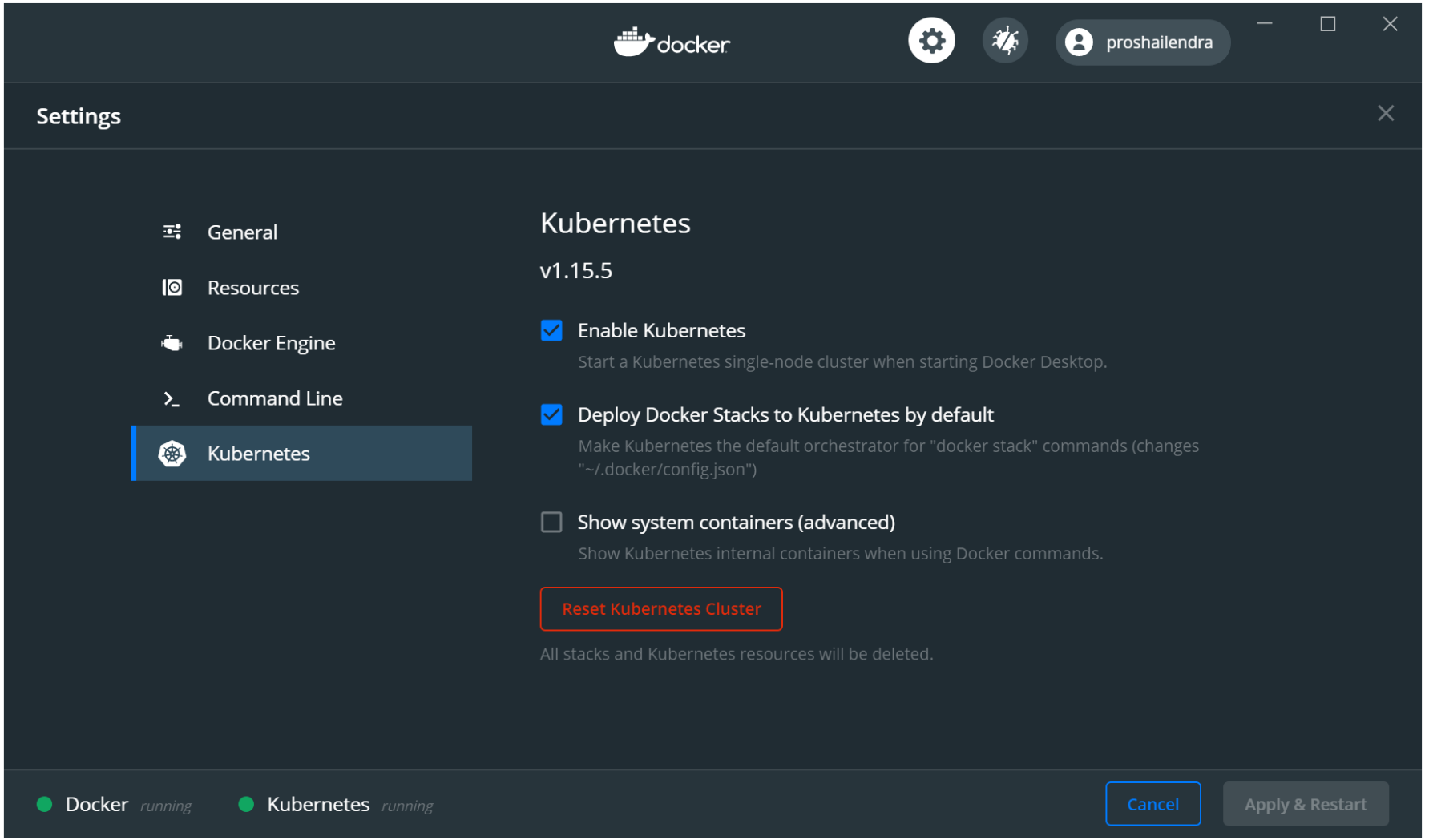
Multi Node Cluster

DotNetTricks

Single Node Cluster



Kubernetes Using Docker Desktop



DotNetTricks

Verify Kubernetes Setup

> kubectl help

> kubectl get nodes

> kubectl cluster-info

> kubectl config get-clusters

> kubectl config get-contexts

> kubectl config view

> kubectl config use-context docker-for-desktop

Configure Kubernetes GUI

- Verify Kubernetes Version

> kubectl version

- Deploy Dashboard GUI

> kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/alternative/kubernetes-dashboard.yaml

- Access Dashboard

> kubectl proxy

> http://localhost:8001/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard:/proxy/#!/overview?namespace=default

YAML

- YAML is a human friendly data serialization standard for all programming languages.
- Used for defining configuration or in an application where data is being stored or transmitted.
- YAML is introduced in 2001.
- Extension are .yml or .yaml
- YAML is case sensitive.
- YAML supports spaces instead of tabs.

XML vs. JSON vs. YAML

```xml
<Servers>
    <Server>
        <name>server1</name>
        <location>india</location>
        <status>active</status>
    </Server>
</Servers>
```

```json
{
    "Servers": [
        "Server": {
            "name": "server1",
            "location": "india",
            "status": "active"
        }
    ]
}
```

```yaml
Servers:
  - name: server1
    location: india
    status: active
```

DotNetTricks

YAML in Action

```yaml
integer: 25
string1: name
string2: "name"
string3: 'name'
float: 25.0
boolean: true
```

Basic: Datatype

```yaml
name: server1
location: india
status: active
```

Key/Value

```yaml
Servers:
  - server1
  - server2
  - server3
```

Array/List: Ordered

```yaml
Servers:
    name: server1
    status: active
    location: india
```

Dictionary/Map: Unordered

```yaml
Servers:
  - server1:
      location: india
      status: active
  - server2:
      location: usa
      status: active
```

Array/Dictionary/Key-value

DotNetTricks

YAML in Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    name: myapp
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

Pod

- A logical container with one or more application containers and some shared resources (volumes).
- Kubernetes uses pods to run your code and images in the cluster. Kubernetes works with Pods, rather than containers, so that containers in the same pod can be guaranteed to run on the same machine.
- Containers in the same pod share their networking infrastructure, storage resources, and lifecycle.

Pod Commands

> kubectl run my-pod --image=nginx --port=80 --labels="app=web" --generator=run-pod/v1

> kubectl get pods

> kubectl delete pod my-pod

> kubectl get pods --show-labels

> kubectl create -f pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
    name: my-pod
    labels:
       app: web
spec:
    containers:
      - name: my-cont
        image: nginx
```

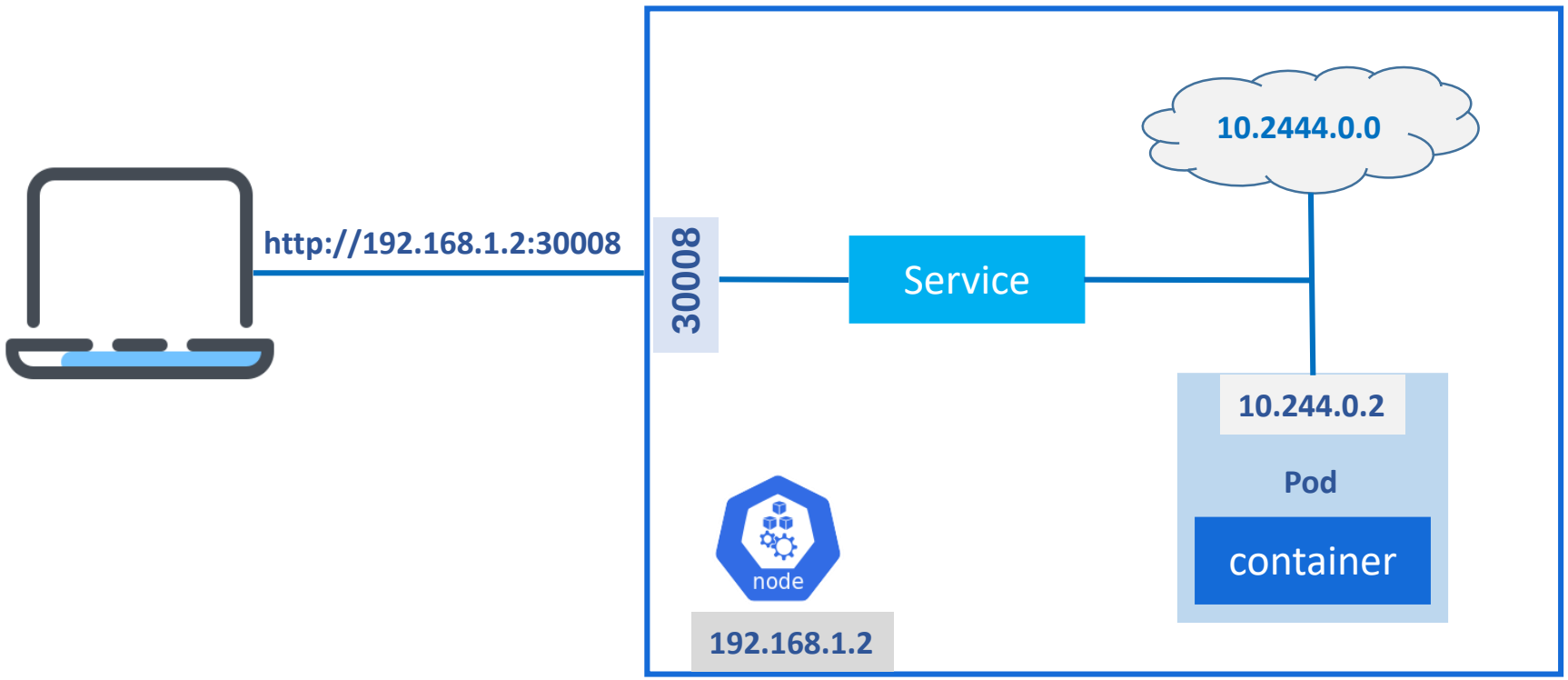
Pod States

- **Pending:** Pod has been created and accepted by the cluster, but one or more of its containers are not yet running.
- **Running:** Pod has been bound to a node, and all of the containers have been created. At least one container is running, is in the process of starting, or is restarting.
- **Succeeded:** All containers in the Pod have terminated successfully. Terminated Pods do not restart.
- **Failed:** All containers in the Pod have terminated, and at least one container has terminated in failure.
- **Unknown:** The state of the Pod cannot be determined.

Service

- A Service enables network access to a set of Pods in Kubernetes.
- Services select Pods based on their labels.
- When a network request is made to the service, it selects all Pods in the cluster matching the service's selector, and forwards the network request to them.

Service



Deploy ASP.NET Core App to Kubernetes

> docker build -t mvcapp:v1 .

> docker image ls

> kubectl apply -f pod.yml

> kubectl get pods

> kubectl apply -f service.yml

> kubectl get services