Kubernetes Advanced

kubernetes

Session #01
Review Main Concepts

kubernetes

Session Contents

- Kubernetes Architecture
- Kubectl
- Pods
- Namespaces
- Deployments
- Services
- Ingress

kubernetes

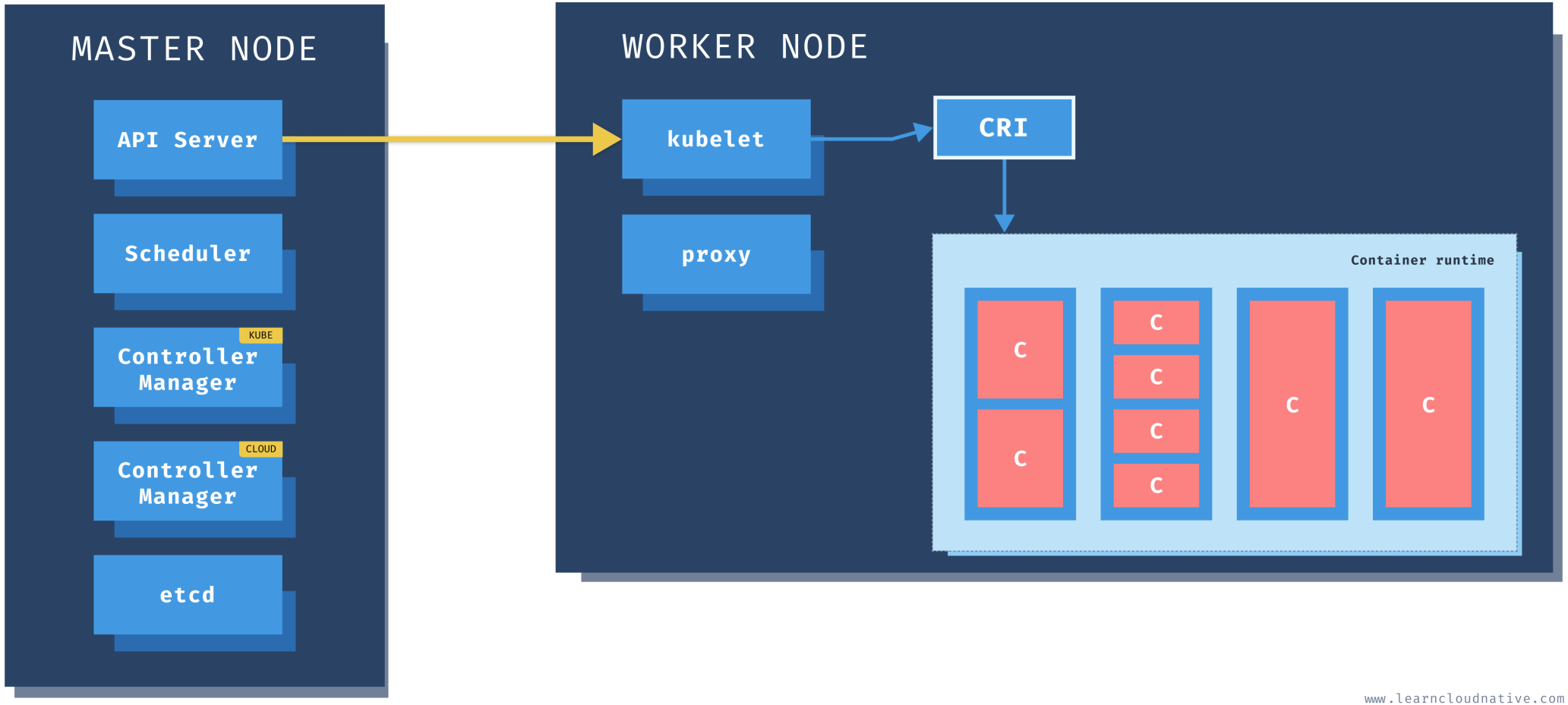
Kubernetes Architecture

Kubernetes Cluster

- A Kubernetes cluster consists of a set of machines (physical or virtual), called nodes
- Control Panel or Master Node(s)manages the cluster and nodes
- Node(s) or Worker Node(s) runs containerized workloads
- Nodes can be heterogeneous (small, large, GPUs, Linux, Windows, etc.)

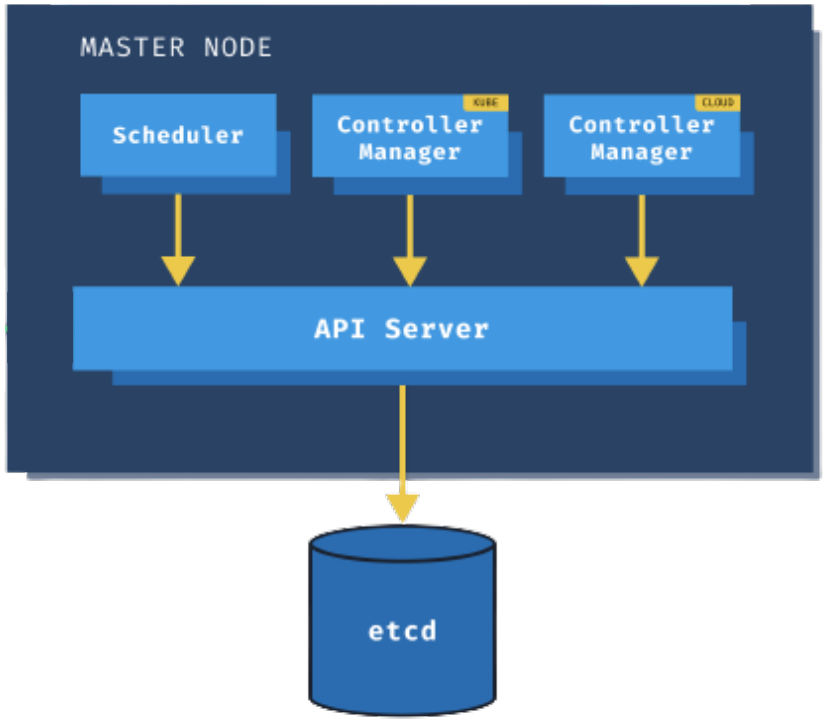
kubernetes

Kubernetes Cluster



Control Plane Components

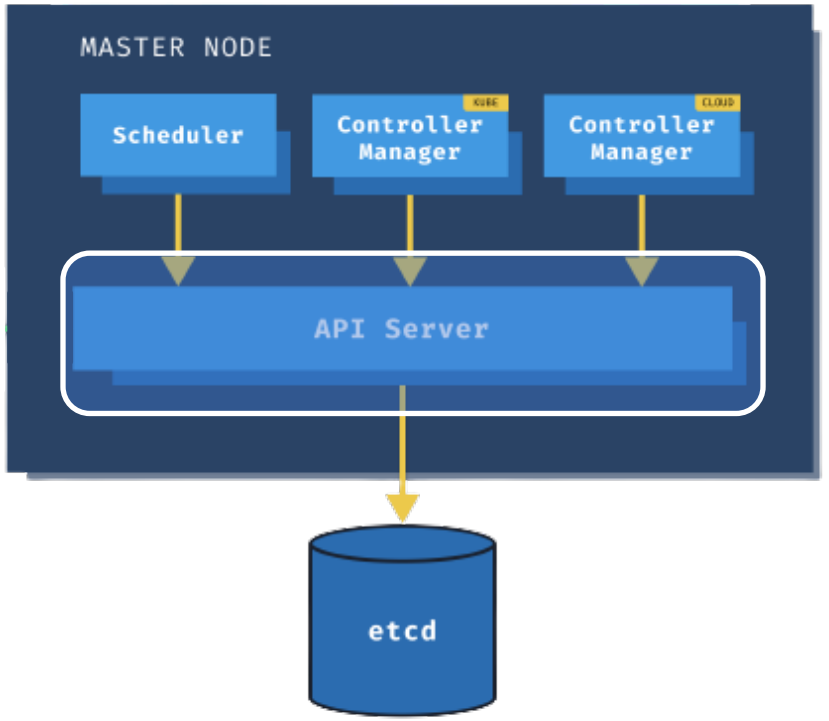
- Make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied)
- Control plane components can be run on any machine in the cluster. For simplicity, start all control plane components on the same machine, and do not run user containers on this machine.



kubernetes

Control Plane: API Server

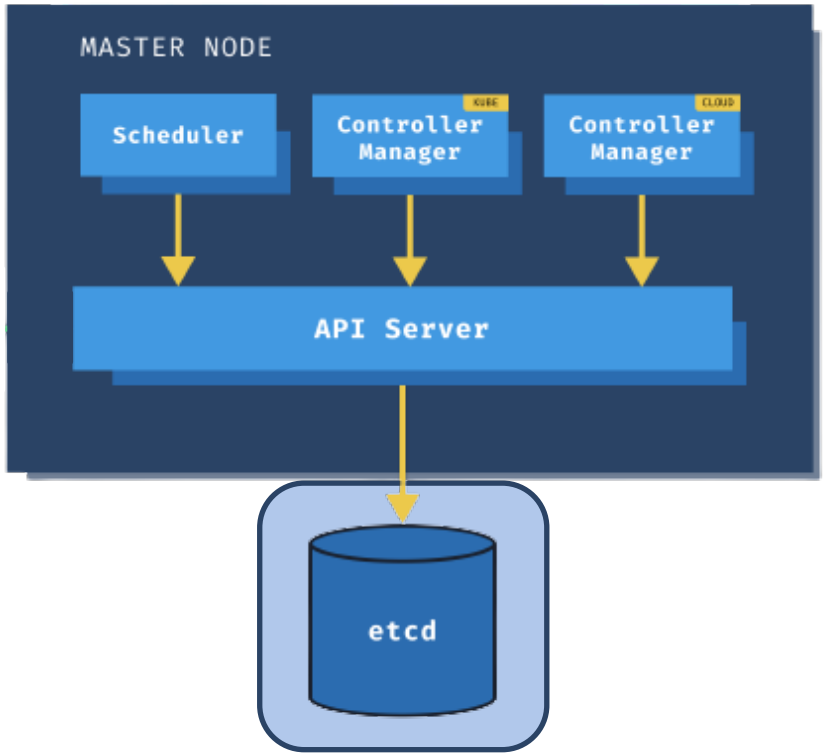
- Component that exposes the Kubernetes API
- The API server is the front end for the Kubernetes control plane
- Every request to manage your cluster should use API Server
- Main implementation of a Kubernetes API server is kube-apiserver.
- Scale horizontally, deploying more instances and balance traffic between those instances



kubernetes

Control Plane: etcd

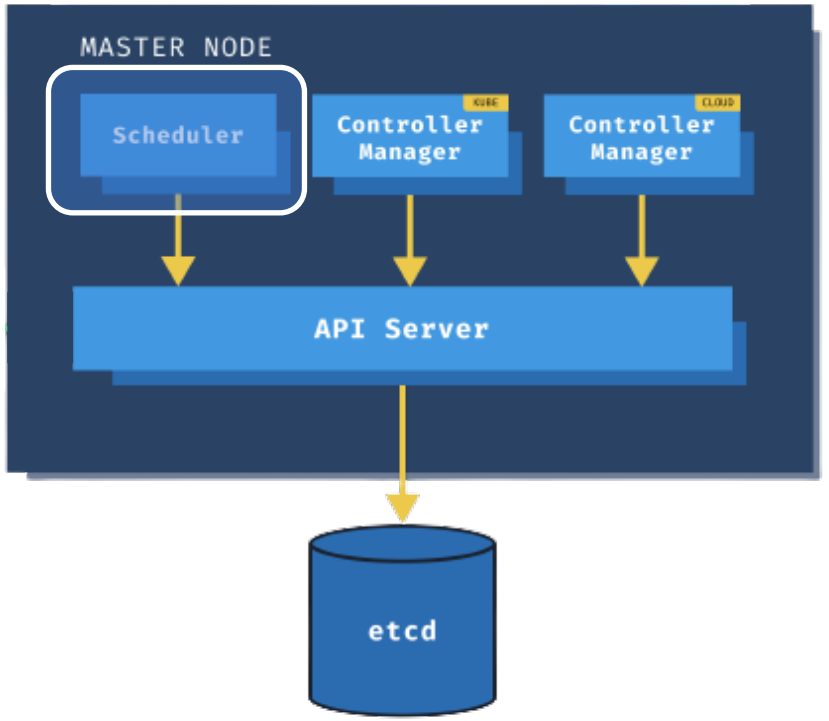
- Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data
- Used as cluster backing store because of simplicity and performance
- Don’t have a backup plan implementation on the cluster, you need to do it manually



kubernetes

Control Plane : Scheduler

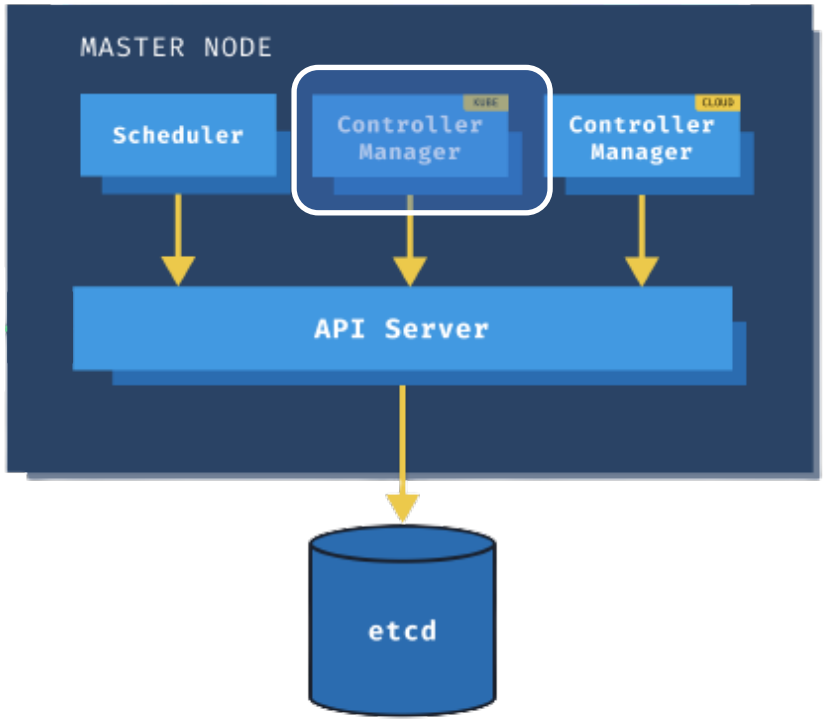
- Component that watches for newly created Pods with no assigned node, and selects a node for them to run on
- Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications



kubernetes

Control Plane : Kube Controller Manager

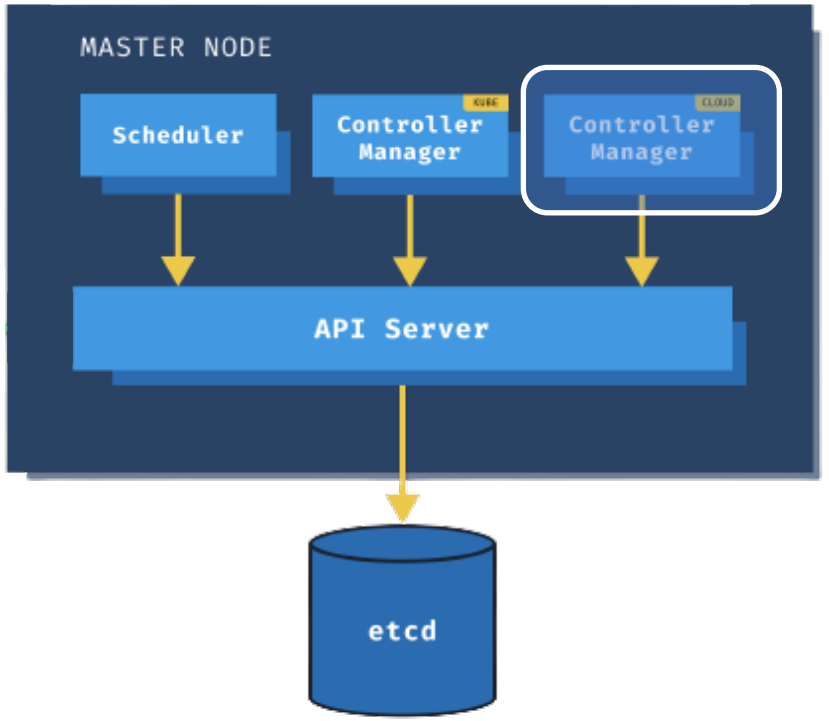
- Component that runs controller processes
- Each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process
- Some types of these controllers are:
  - Node controller: Responsible for noticing and responding when nodes go down.
  - Replication controller: Responsible to run the number of Pods defined on each ReplicaSet
  - ServiceAccount controller: Create default ServiceAccounts for new namespaces



kubernetes

Control Plane : Cloud Controller Manager

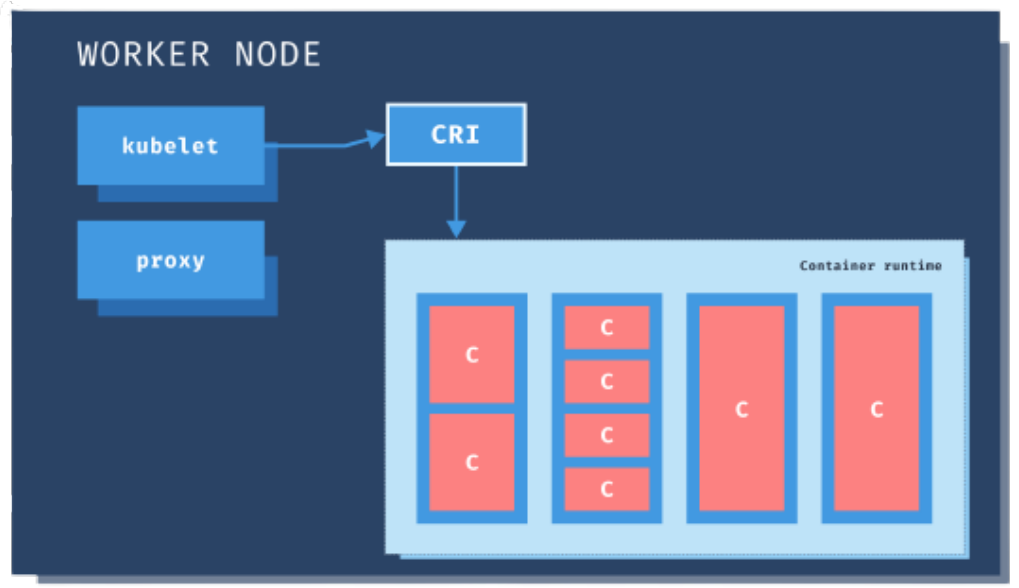
- Component that embeds cloud-specific control logic
- Let you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster
- If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.



kubernetes

Node Components

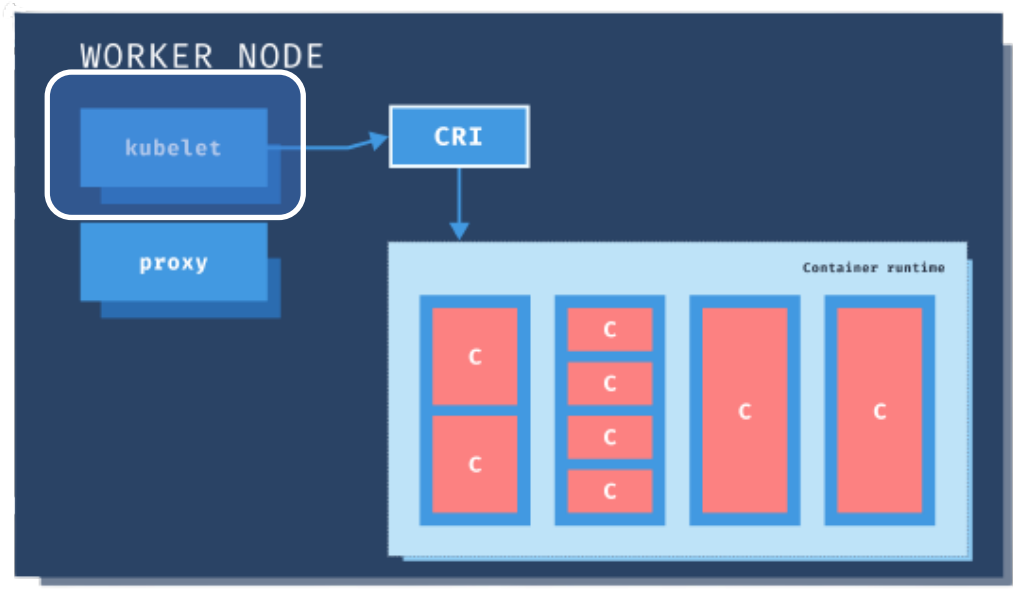
- Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment
- These components needs to run on a machine to be considered part of the Kubernetes cluster



kubernetes

Node: Kubelet

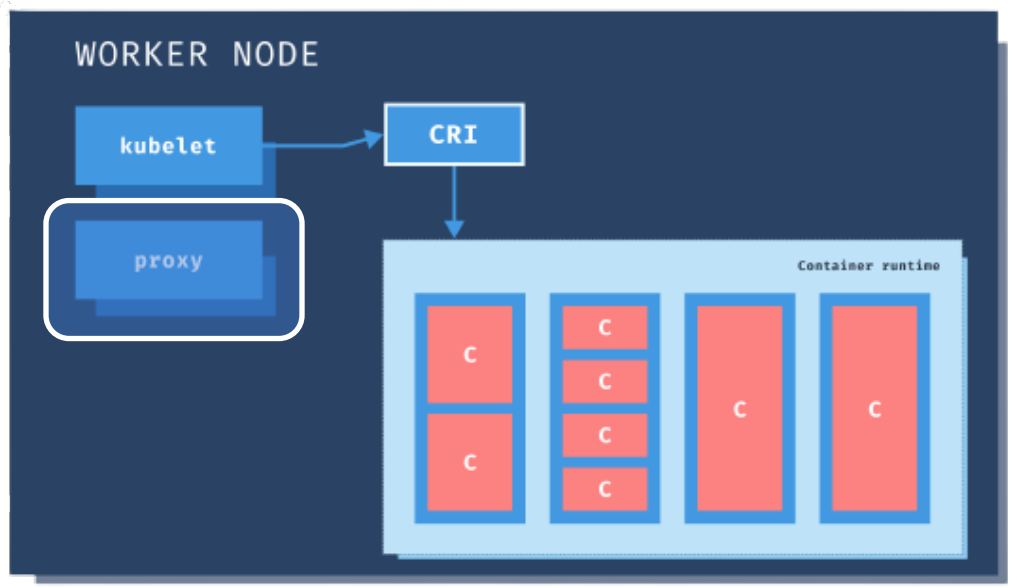
- An agent that runs on each node in the cluster and makes sure that containers are running in a Pod
- Takes a set of PodSpecs and ensures that the containers described in those PodSpecs are running and healthy
- Doesn't manage containers which were not created by Kubernetes



kubernetes

Node: Kube Proxy

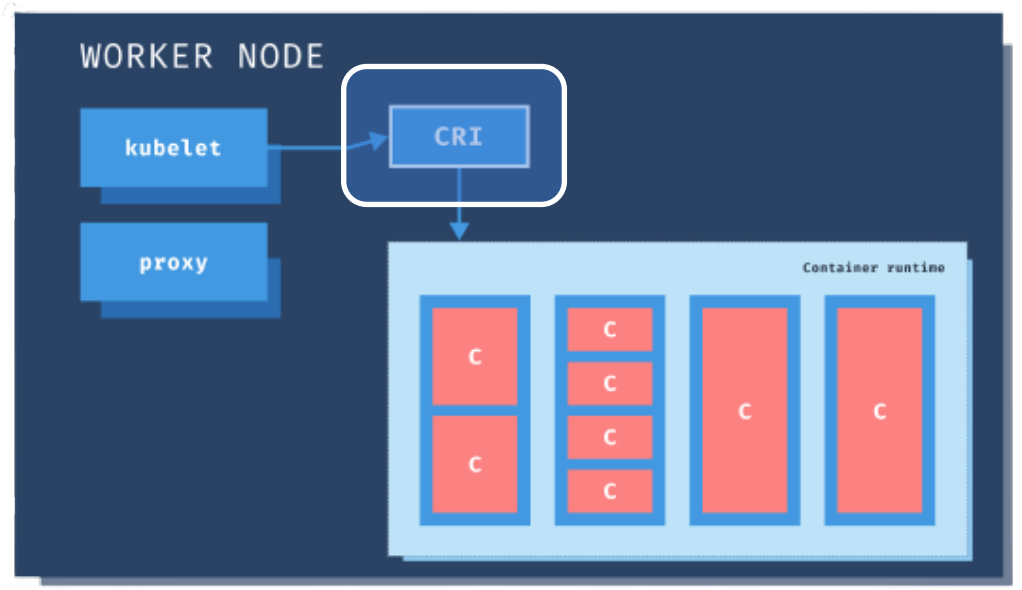
- Network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept
- Maintains network rules on nodes to allow network communication to your Pods from network sessions inside or outside of your cluster
- kube-proxy uses the operating system packet filtering layer



kubernetes

Node: CRI

- The container runtime is the software that is responsible for running containers
- Kubernetes supports container runtimes such as containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface)



kubernetes

Kubectl

CLI Tooling

- Preferable way to interact with Kubernetes Clusters is using CLI (Command-Line Interface) tools
- Motivation
  - Allow to automatize interactions
  - Needs much less resources
  - Small attach surface

kubernetes

API Server

- API Server is a “simple” REST API application, you can manage a cluster by making REST calls to the API Server
- Example HTTP Request: `GET /api/v1/namespaces/default/pods/{name}`
- Some challenges
  - Manage REST requests
  - Authn/authz
- Kubernetes have a CLI tool to make this interaction easily: kubectl

kubernetes

Kubectl

- API Server is a “simple” REST API application, you can manage a cluster by making REST calls to the API Server
- Example HTTP Request: `GET /api/v1/namespaces/default/pods/{name}`
- Some challenges
  - Manage REST requests
  - Authn/authz
- Kubernetes have a CLI tool to make this interaction easily: `kubectl`
- First challenge, how to pronounce? ‘kubectl’: The definitive pronunciation guide

kubernetes

Kubectl: How to use?

```
kubectl [command] [type] [name] [flags]
```

- `command`: Operation that you want to perform on one or more resources, for example `create, get, describe, delete.`
- `type`: Resource type. Case-insensitive and can specify the singular, plural, or abbreviated forms.
- `name`: Case-insensitive name of the resource. If the name is omitted, details for all resources are displayed
- `flags:` Optional flags. For example, `-o` allow to specify output type of the commands

kubernetes

Kubectl: Examples

- List all nodes

```
kubectl get nodes
```

- Get more details on node node01

```
kubectl describe node node01
```

- List all nodes with more details

```
kubectl get node -o wide
```

- List all nodes and labels

```
kubectl get node -show-labels
```

kubernetes

Kubectl: How to reference resources?

- You can perform an action on several resource using only one command even on resources from different types
- Resources from same type

```
kubectl get pod pod-01 pod-02
```

- Resources from different types

```
kubectl get pod/pod-01 node/node01
```

kubectl Cheat Sheet | Kubernetes

Kubectl Reference Docs (kubernetes.io)

kubernetes

Kubectl: How to identify?

- `kubeconfig` is a file used to organize access to several cluster usually stored at `~/.kube/config`
- Needs to be kept on a secure place since have complete information about authn/authz of a user to a cluster
- This file should never be included on a repo or used for CI/CD process due to security reasons
- Uses the concept of contexts. A context makes a match between a cluster (API Server URL) and a user

kubernetes

Kubectl: Examples

- Get all clusters configuration available

```
kubectl config view
```

- Get actual context

```
kubectl config current-context
```

- Set another context

```
kubectl config use-context my-cluster-name
```

kubernetes

Krew: Plugin Manager for Kubectl

- Allow to search, install and update additional tools to use with kubectl
- 200+ plugins available: https://krew.sigs.k8s.io/plugins/
- Maintenance done by a Kubernetes SIG (Special Interest Group)
- List installed plugins

```
kubectl krew list
```

- Install a plugin

```
kubectl krew install <package>
```

kubernetes

Krew: Plugin Manager for Kubectl

- cert-manager: Manage cert-manager resources inside your cluster
- ctx: Switch between contexts in your kubeconfig
- ingress-nginx: Interact with ingress-nginx
- kubesec-scan: Scan Kubernetes resources with kubesec.io.
- ns: Switch between Kubernetes namespaces
- resource-capacity: Provides an overview of resource requests, limits, and utilization

kubernetes

Pods

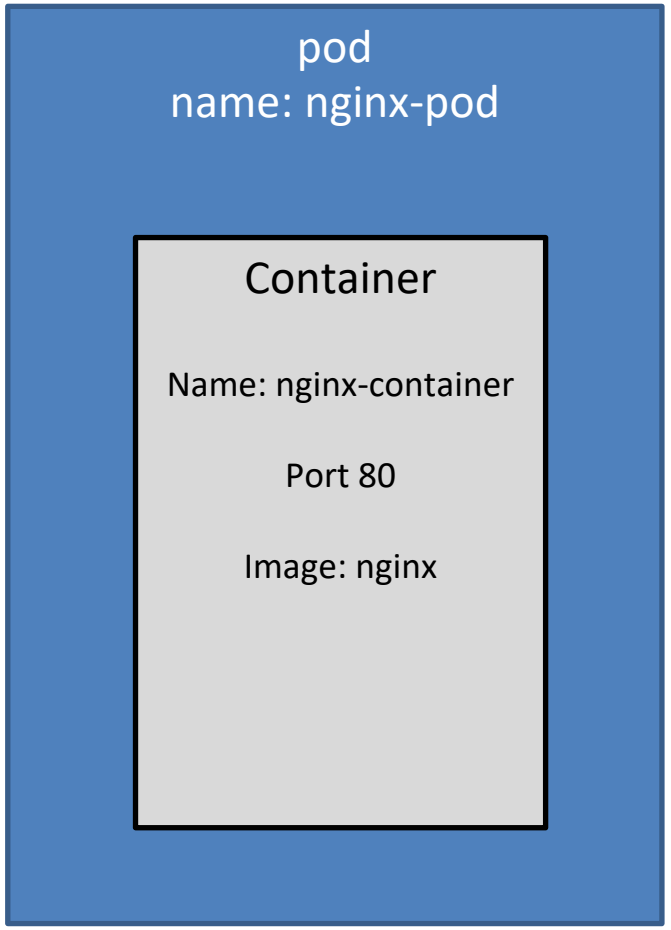
What is a Pod?

- Pods are the smallest deployable compute units you can create and manage in Kubernetes
- A Pod can manage one or more containers, with shared storage (volumes), environment variables, network resources, and a specification for how to run the containers
- Containers running in a Pod share the same IP and ports and communicate using native inter-process communication channels or localhost.
- Pods are immutable - if any change is made to the Pod specification (spec), a new Pod is created and then the old Pod is deleted

kubernetes

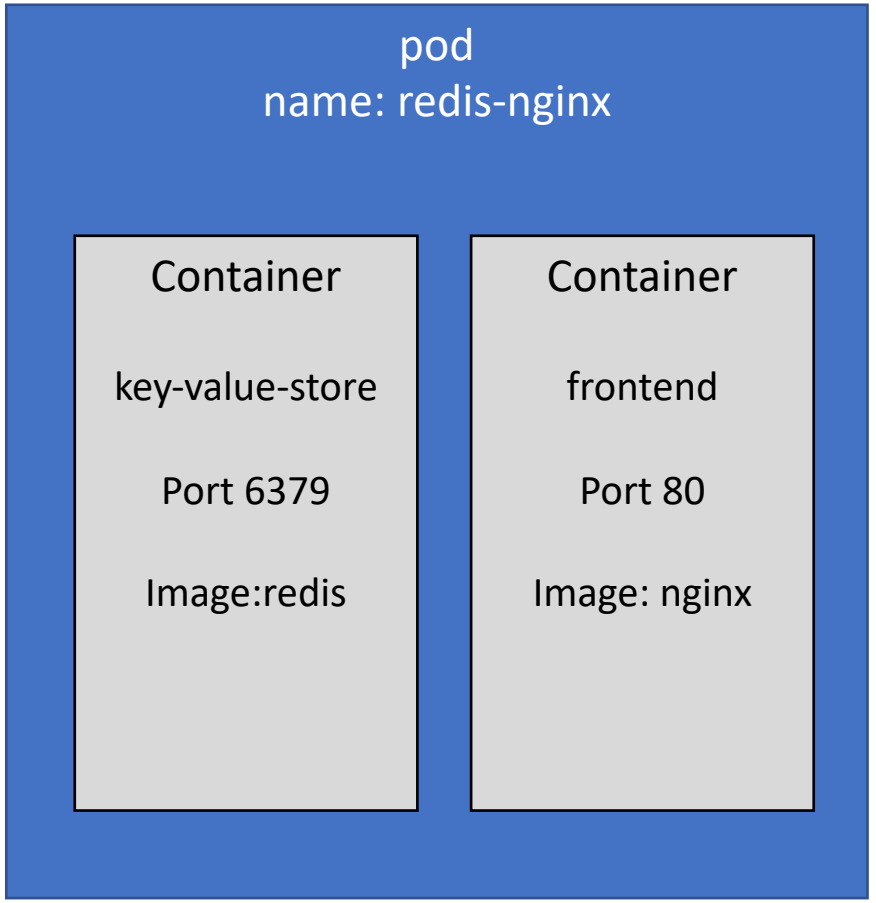
Pod with one container

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - image: nginx
    name: nginx-container
    resources: {}
    ports:
    - containerPort: 80
```



Pod with two containers

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-nginx
  labels:
    app: web
spec:
  containers:
    - name: key-value-store
      image: redis
      ports:
        - containerPort: 6379
    - name: frontend
      image: nginx
      ports:
        - containerPort: 80
```



Kubectl: Interact with pods

- Get access to **nginx-pod** pod

```
kubectl -it exec nginx-pod -- sh
```

- Get access to container frontend on **redis-nginx** pod

```
kubectl exec -it redis-nginx -c frontend -- bash
```

- Port forwarding to port 80 on **nginx-pod** pod

```
kubectl port-forward nginx-pod 8080:80
```

- Port forwarding to port 80 on **redis-nginx** pod

```
kubectl port-forward redis-nginx 8080:80
```

kubernetes

Pods: Handle Resources

- A good practice when deploying pods on Kubernetes is to define the resources that will be used by it
- Kubernetes uses 2 concepts: Requests and Limits
- Requests: Amount of resources used by scheduler to define which node could best fit. This amount is always reserved for the pod
- Limits: Maximum amount of resources a pod can use.
- You can define request and limits for CPUs, memory and GPUs

kubernetes

Pods: Self Healing

- A Pod don’t have any ability of self-healing
- Pods can be used directly but usually a controller is used to automatically manage your pods
- Each controller have specific way to control and manage their pods
  - ReplicaSets: Controls pods number of replicas
  - DaemonSets: Controls if one pods runs on each worker node
  - StatfulSets: Controls link between pod and persistent storage to handle pod state

kubernetes

Namespaces

Namespaces

- Kubernetes supports multiple virtual clusters backed by the same physical cluster
- These virtual clusters are called namespaces and creates logical partitions inside your cluster
- Namespaces provide a scope for names
- Names of resources need to be unique within a namespace, but not across namespaces.
- Namespaces can be used to
  - Allowing/denying network communications
  - Define RBAC permissions
  - Define resource quotas

kubernetes

Namespaces are mandatory?

- Usually, your resources are created in the context of a namespace
- "`default`" namespace is created on a empty cluster
- Your `kubectl` commands uses the namespace you set on your `kubeconfig` to be the default
- You can use `kubectl ns` plugin to easily change it
- When you want to not change the default, you need to use the flag `-n` in each command

kubernetes

Namespaces are mandatory?

- When a resource is not namespaced, means that is a cluster resource

- Get list of resources that aren't namespaced scope

```
kubectl api-resources --namespaced=false
```

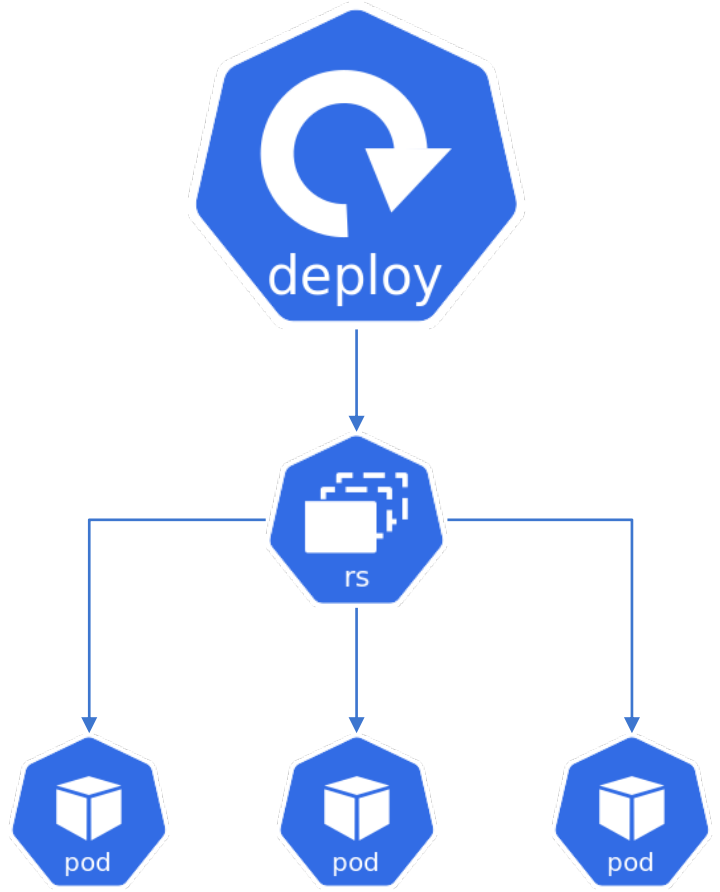
NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
componentstatuses	cs	v1	false	ComponentStatus
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumes	pv	v1	false	PersistentVolume
mutatingwebhookconfigurations		admissionregistration.k8s.io/v1	false	MutatingWebhookConfiguration
validatingwebhookconfigurations		admissionregistration.k8s.io/v1	false	ValidatingWebhookConfiguration
customresourcedefinitions	crd,crds	apiextensions.k8s.io/v1	false	CustomResourceDefinition

kubernetes

Deployments

Deployments

- A Deployment is a higher-level controller that manages ReplicaSets
- Provides declarative updates to Pods along with a lot of other useful features.



kubernetes

Deploymeny Manifest

Deployment properties

`replica` set the number of pods

`selector` finds the template

`template` defines PodSpec

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: workload-1-dep
spec:
  replicas: 6
  selector:
    matchLabels:
      app: workload-1
  template:
    metadata:
      labels:
        app: workload-1
        color: lime
    spec:
      containers:
        - name: workload
          image: nginx:1.18
          ports:
            - containerPort: 80
```

Deployment

- Describe a desired state in a Deployment
- Deployment Controller changes the actual state to the desired state at a controlled rate
- Deployments provide fine-grained control over how and when a new pod version is rolled out as well as rolled back to a previous state

kubernetes

Deployment Strategy

- A deployment strategy is a way to change or upgrade an application
- Rolling Update - Consists of slowly rolling out a new version of an application by replacing instances one after the old version until all the instances are rolled out. This is the default strategy for Kubernetes Deployments when none is specified.
- Recreate - Consists of shutting down all instances of the current version, then deploying the new version. This technique implies downtime of the service that depends on both shutdown and boot duration of the application.

kubernetes

Service

Services: Motivation

- Kubernetes Pods are created and destroyed to match the state of your cluster making them ephemeral
- Each Pod gets its own IP address, however in a Deployment, the set of Pods running in one moment in time could be different from the set of Pods running that application a moment later
- If some set of Pods provides functionality to other Pods inside your cluster, how do they find out and keep track of which IP address to connect to?

kubernetes

What is a Service?

- An abstraction that defines a logical set of loosely-coupled pods and a policy by which to access them as a network service.
- Use selectors to define which pods to include.
- Every type of service (unless ExternalName) load balances traffic to Pods (Layer 4)
- Preferrable way to expose Pods to other Pods within the cluster
- Maps an external and global port to target (container) ports

kubernetes

What is a Service?

- Kubernetes automatically updates which Pods are available to which service by creating Endpoints objects
- Exists 3 types of Services
  - ClusterIP
  - NodePort
  - LoadBalancer
- Exists one additional named ExternalName that to create an internal service to redirect to another service
  - Mostly used to create a proxy server to an external service that may be changes between environments

kubernetes

Load Balancing

- Depends on kube-proxy configuration mode
- User space proxy mode
  - Uses round-robin algorithm to select pods
- Iptables proxy mode (default)
  - Uses random selection of pods
- IPVS proxy mode
  - rr: round-robin
  - lc: least connection (smallest number of open connections)
  - dh: destination hashing
  - sh: source hashing

kubernetes

Service Discovery

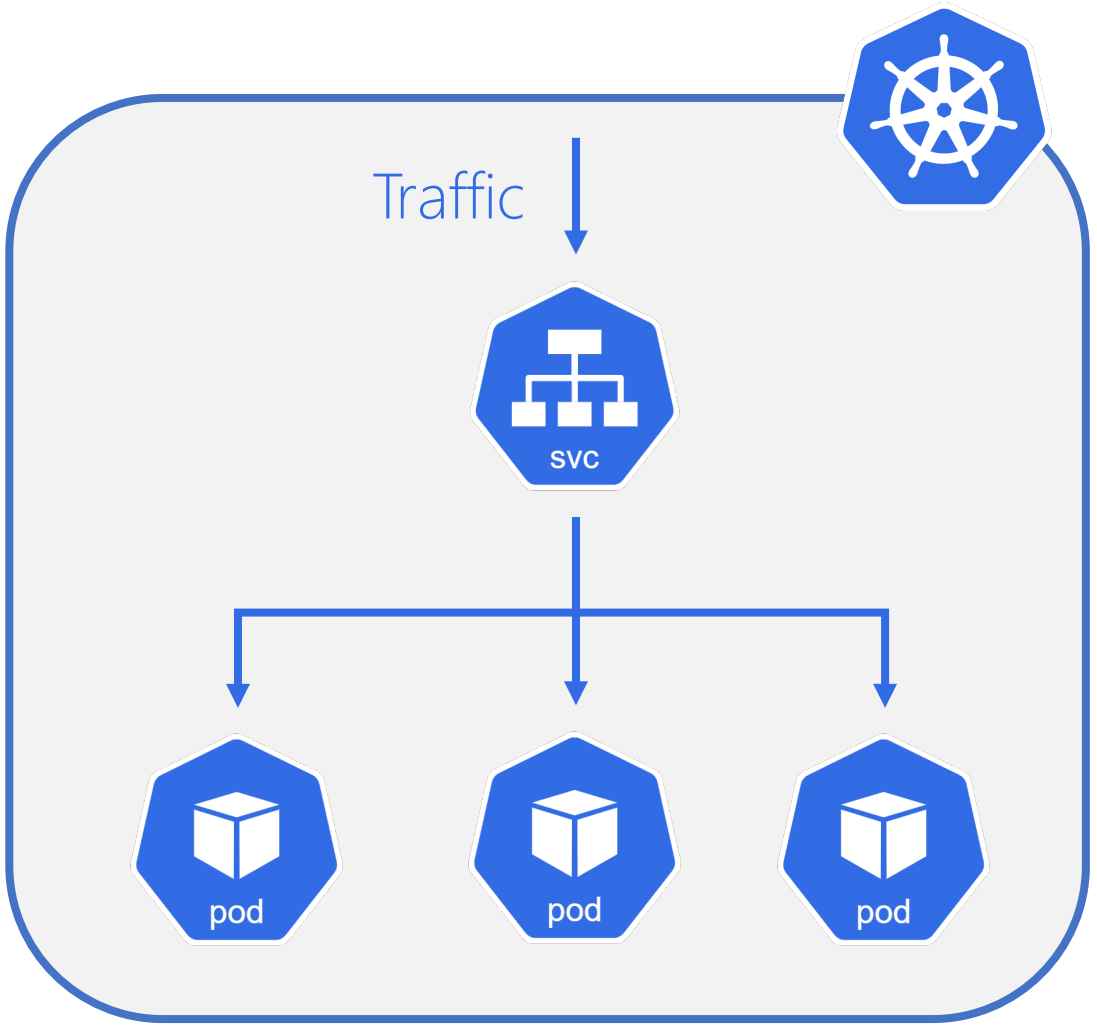
- Any cluster have an internal service to allow service discovery
- This service is implemented using a plugin strategy
- Most used is CoreDNS, an opensource DNS server
- Every cluster have a global ClusterIP service named `kube-dns` with endpoints to plugin pods
- Every pod is configured with `kube-dns` service IP as nameserver to name resolution

kubernetes

ClusterIP

ClusterIP

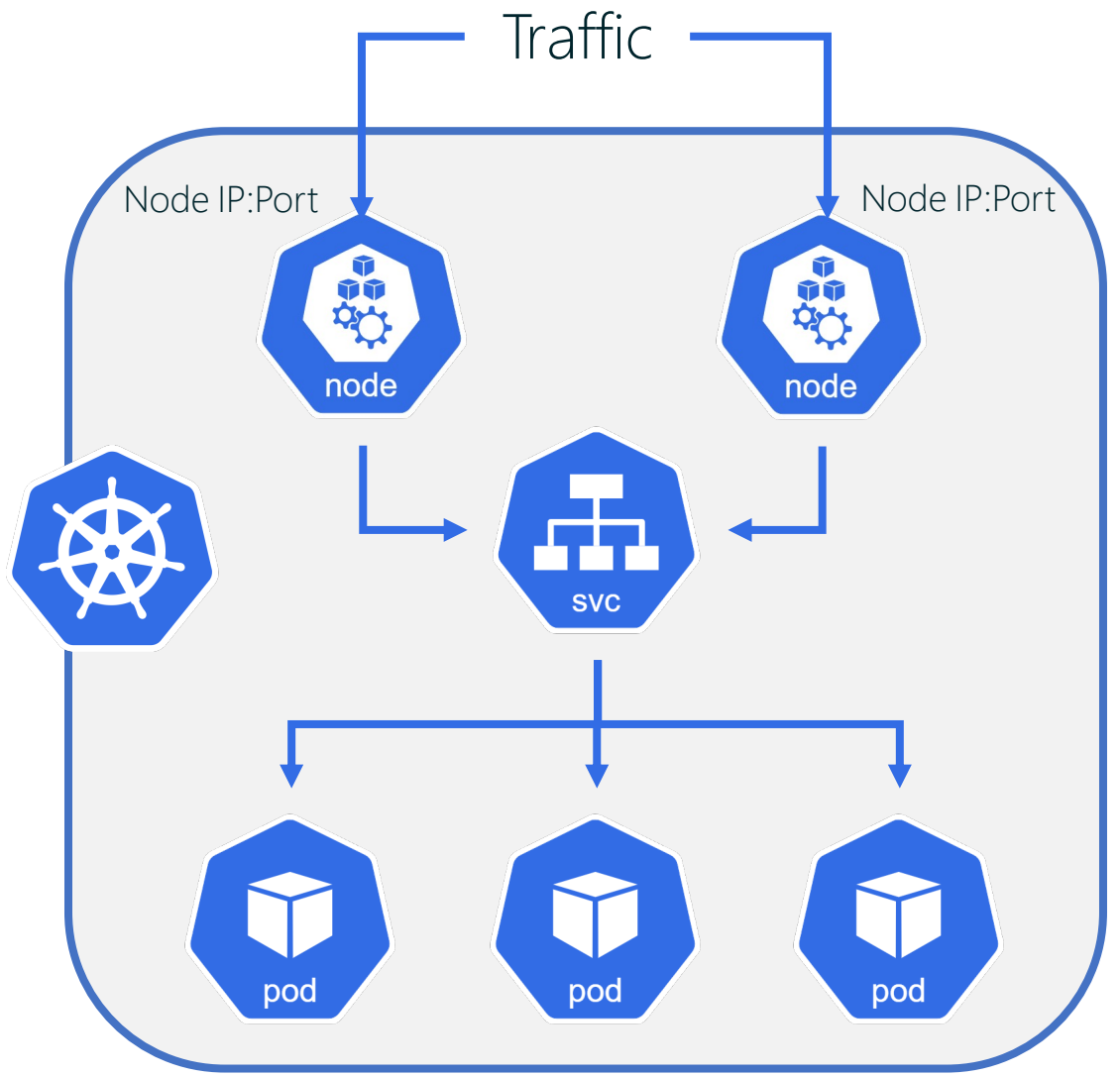
- Exposes the Service on a cluster-internal IP
- Choosing this value makes the Service only reachable from within the cluster
- Default service type
- Service name can be used as DNS
  - http://svc-name
  - http://svc-name.ns.svc.cluster.local



NodePort

NodePort

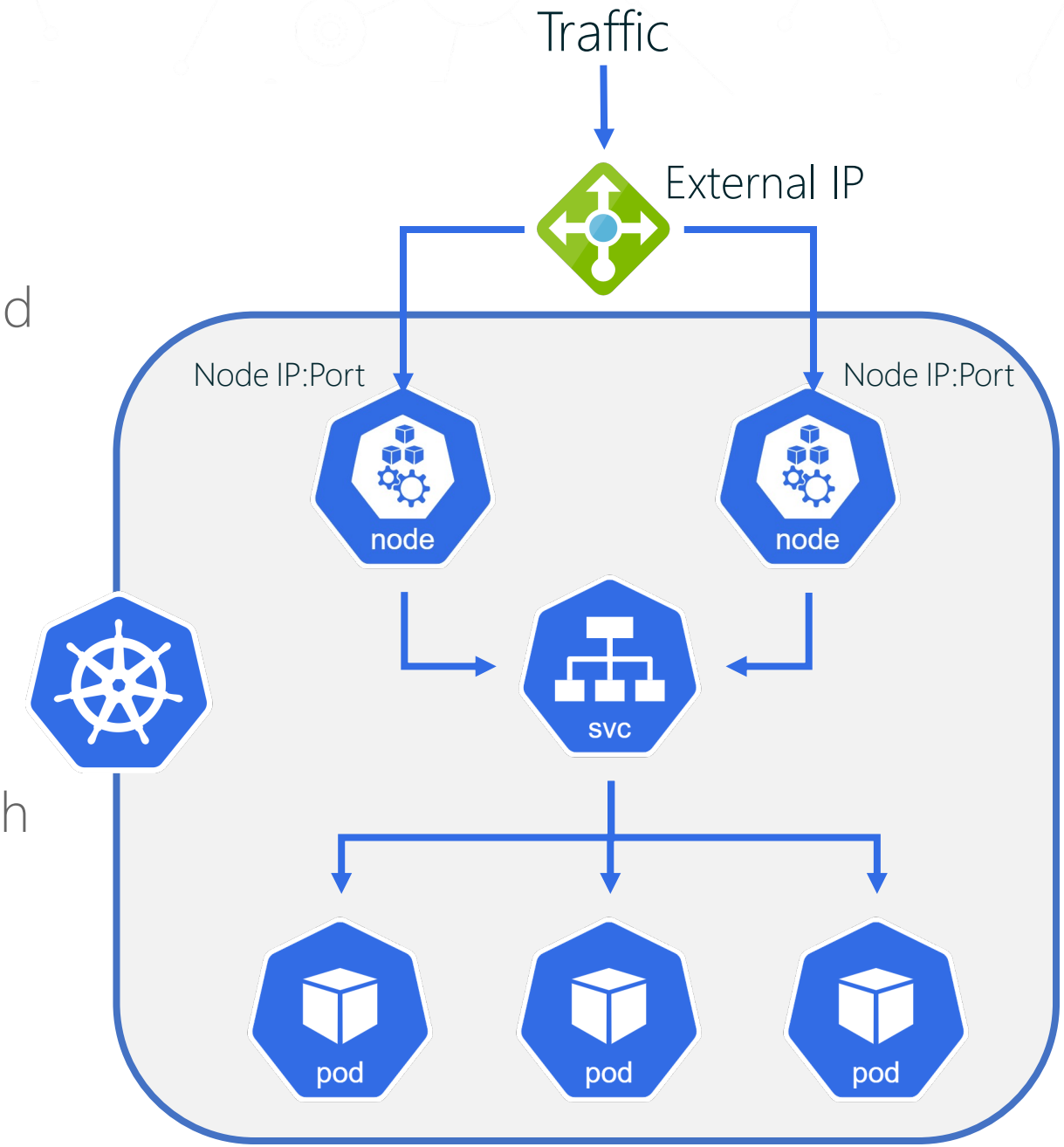
- Exposes the Service on each Node's IP at a static port (the NodePort)
- A ClusterIP Service, to which the NodePort Service routes, is automatically created.
- You'll be able to contact the NodePort Service, from outside the cluster, by requesting `<NodeIP>:<NodePort>`
- If `nodePort` property is not specified, is automatically set a port from the range 30000-32767



LoadBalancer

LoadBalancer

- Exposes the Service externally using a cloud provider's load balancer
- NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created
- On-prem needs manual configuration
- ExternalIP means an IP External related with cluster. No needs to be an Internet public IP



Ingress

Motivation

- When need to expose HTTP/HTTPS endpoint outside of the cluster, you can use LoadBalancer services
- That may overload the use of public (external IPs) and need several additional components to implement this service
- Using services, you are only exposing on Layer 4 and when using HTTP/HTTPS is more functional to work on Layer 7
- Security and configuration wise is important to have only one place to access externally the cluster (DNS configuration, etc.)

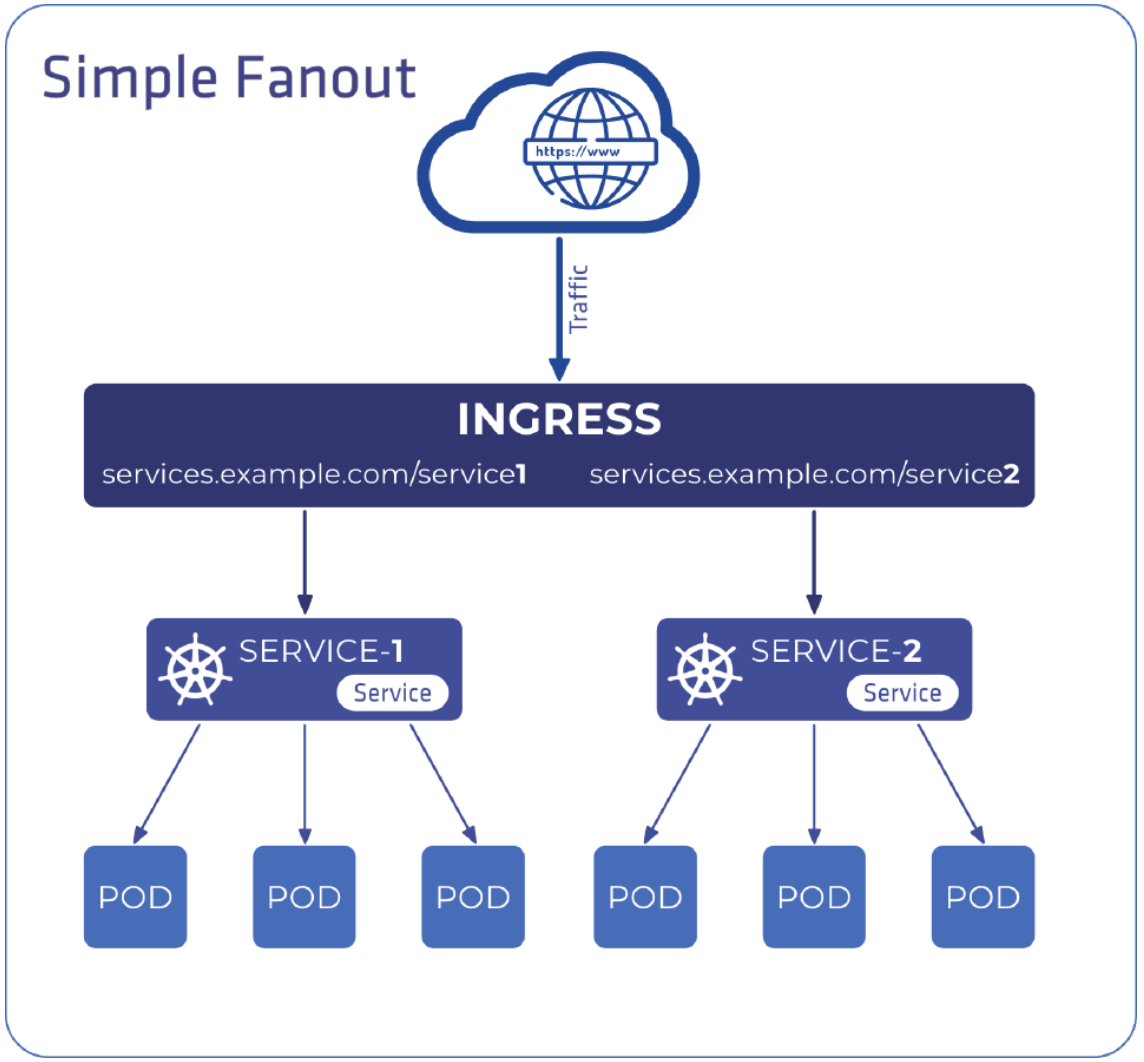
kubernetes

What is Ingress?

- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic routing is controlled by rules defined on the Ingress resource
- An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting
- An Ingress does not expose arbitrary ports or protocols. To expose services other than HTTP and HTTPS to the internet you need to use NodePort or LoadBalancer services

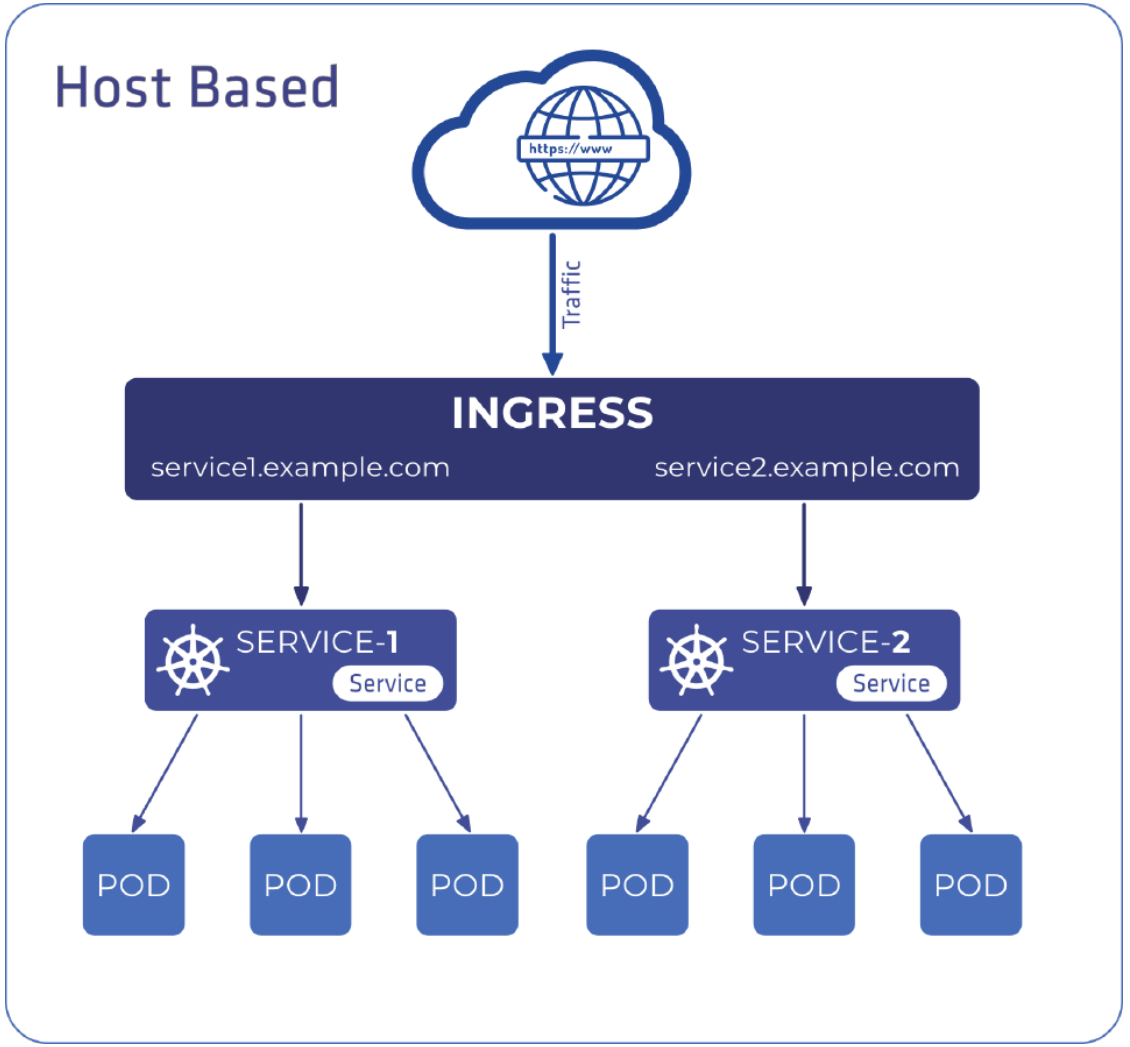
kubernetes

Ingress rule using URI



kubernetes

Ingress rule based on Host



kubernetes

Ingress Controller

- An Ingress is just a resource defining rules. This resource needs a controller to implement those rules
- Kubernetes don’t have a native Ingress Controller and must be installed by cluster administrator
- An ingress controller is mandatory to handle Ingress resource. Without it, Ingress resource doesn’t do anything

kubernetes

Ingress Controller

- Most used ingress controller
  - Nginx ingress controller
  - Contour ingress controller
  - HAProxy ingress controller
  - Traefik ingress controller
  - Kong Ingress Controller
  - ingress-gce (Google Cloud only)
  - aws-load-balancer-controller (AWS only)
  - application-gateway-kubernetes-ingress (Azure only)

kubernetes

Questions?

kubernetes

Lab #01: Cluster Ready

kubernetes