Kubernetes Essentials

Rey Lejano @ DeveloperWeek 2024

kubernetes

whoami

Rey Lejano

- Solutions Architect @ Red Hat
- Kubernetes SIG Docs co-chair
- Kubernetes SIG Security subproject lead
- Kubernetes v1.23 Release Lead
- CNCF Ambassador
- DevOps Institute Ambassador

@reylejano

kubernetes

Agenda

- Container basics
- Kubernetes history
- Kubernetes architecture
- Kubernetes resources to deploy an application and get outside traffic to the application
- Tutorial

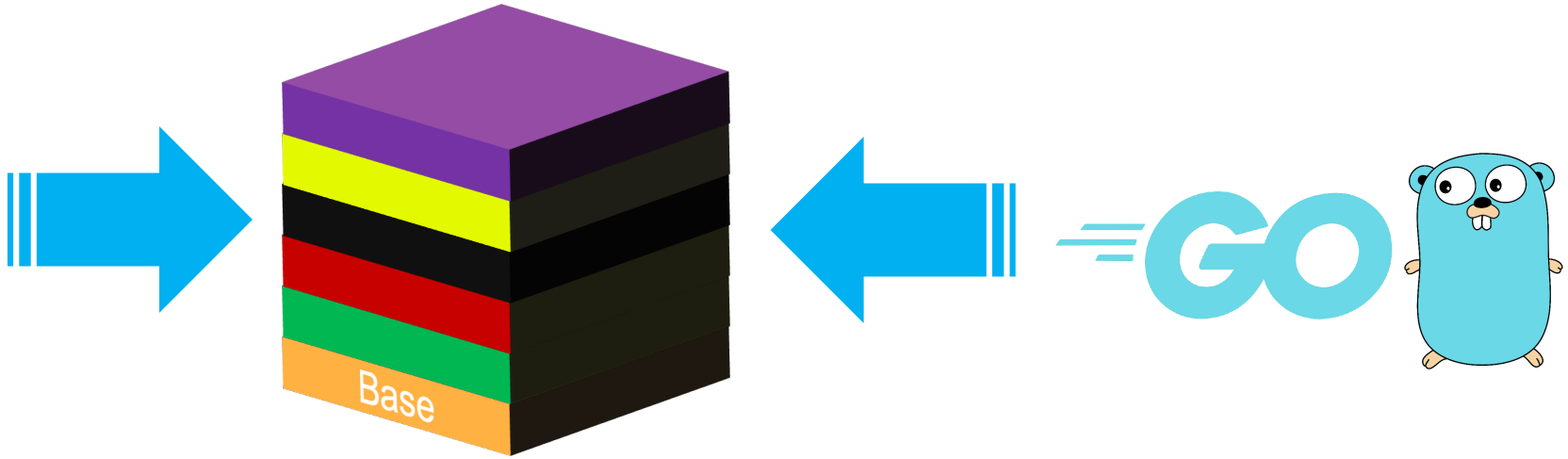
  https://github.com/reylejano/kubernetes-essentials-devweek

kubernetes

Container Basics

Containerization

```go
package main

import (
    "fmt"
    "net/http"
)

func hello(w http.ResponseWriter, req *http.Request) {
    fmt.Fprintf(w, "Hello, world!\n")
}

func headers(w http.ResponseWriter, req *http.Request) {
    for name, headers := range req.Header {
        for _, h := range headers {
            fmt.Fprintf(w, "%v: %v\n", name, h)
        }
    }
}

func main() {
    http.HandleFunc("/hello", hello)
    http.HandleFunc("/headers", headers)
    http.ListenAndServe(":80", nil)
}
```



Containers are the implementation of
cgroups & namespaces

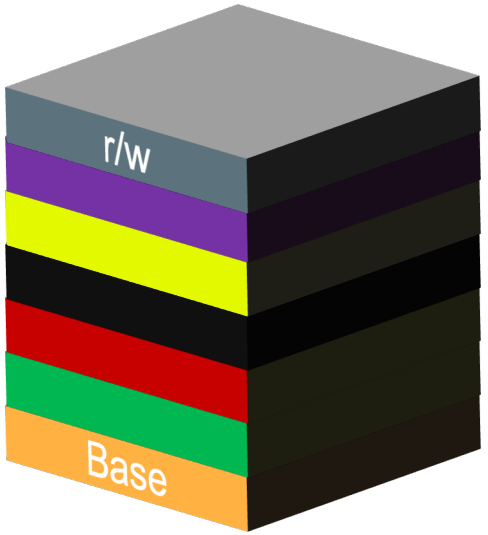
kubernetes

Container Runtime

docker

containerd

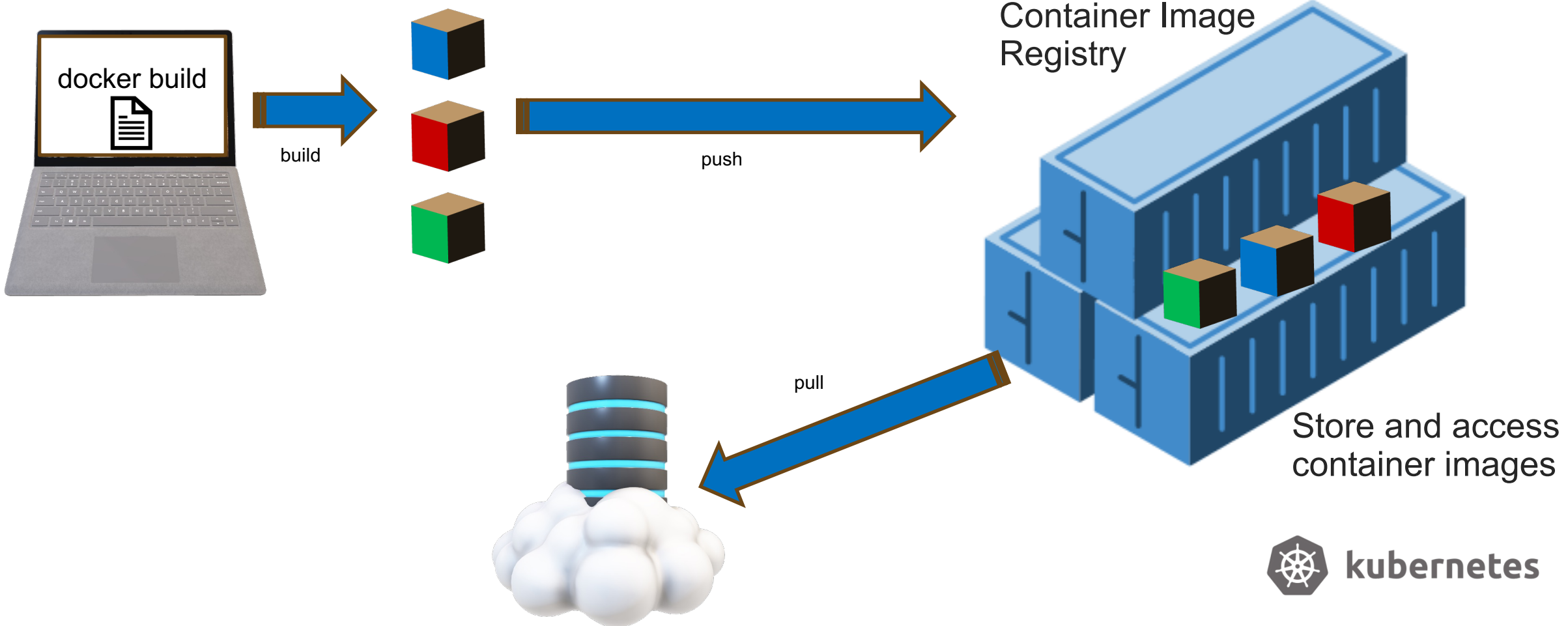
runc



Container runtime runs executes containers
based on container images

kubernetes

Container Registry



Cloud Native

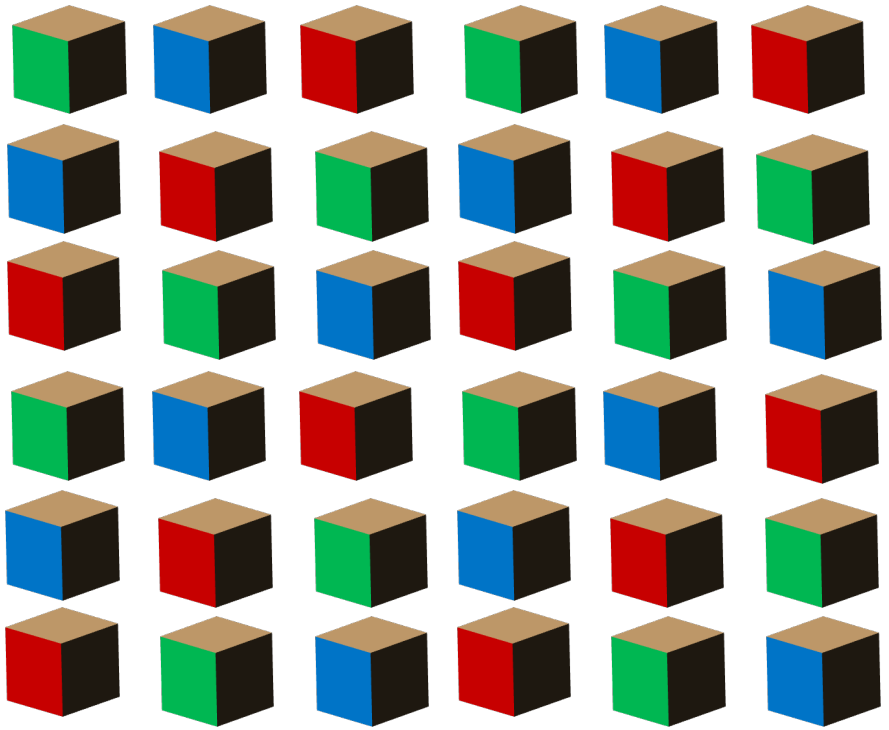
- Containers are the foundation of cloud native applications – take advantage of cloud computing: runs anywhere, takes advantage of distributed compute and scales

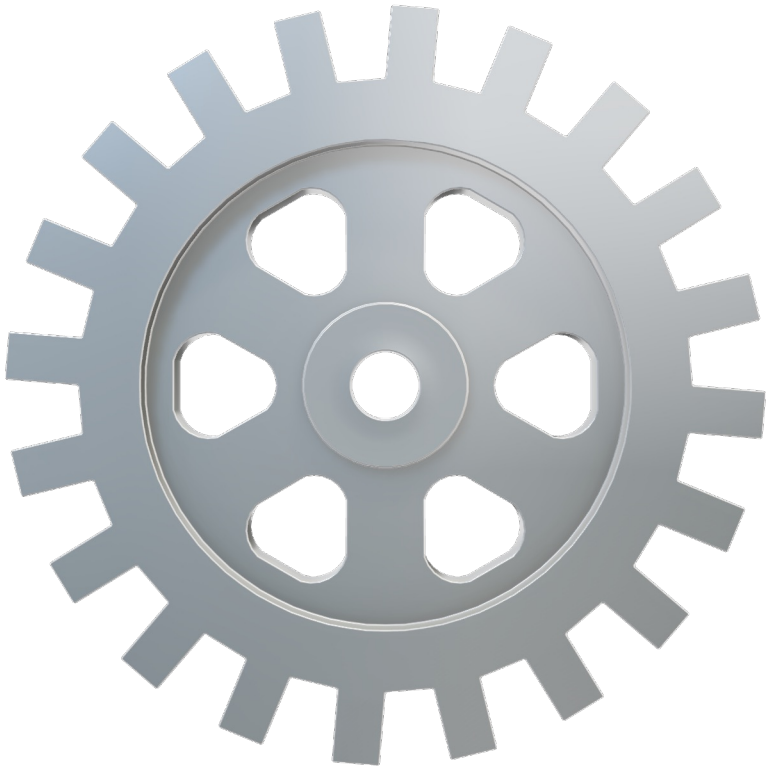
- Decouples apps to specific environments / microservices / 12 factor app
- Open source
- Runs anywhere
- Interoperates with other tools and common conventions

kubernetes

Container Orchestration

Container Orchestration

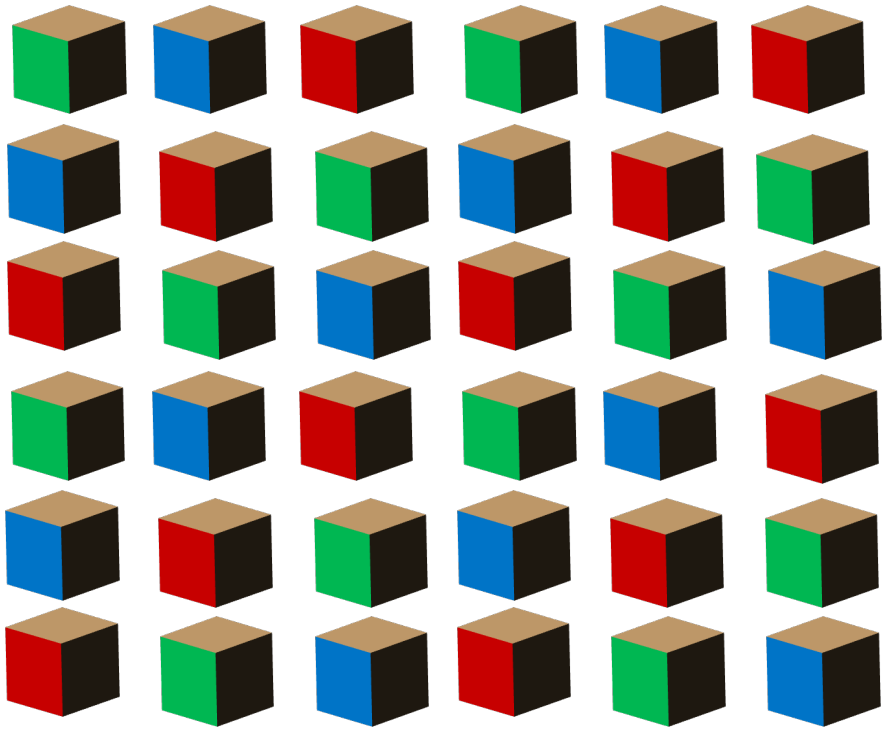




How do we solve…
- Automation?
- Orchestration?
- Security?
- Networking?

kubernetes

Container Orchestration



kubernetes

Kubernetes

- Container orchestrator that deploys, manages, scales containerized workloads on distributed compute at-scale.
- Greek for Helmsman or Pilot **κυβερνήτης**
- Utilizes the kernel of distributed systems so it can run ”anywhere” AKA “Linux kernel” of distributed systems
- Abstracts the underlying hardware of distributed compute and use the same API on every cloud and on-prem to manage workloads
- Works so that the actual state matches the desired state (self-healing)

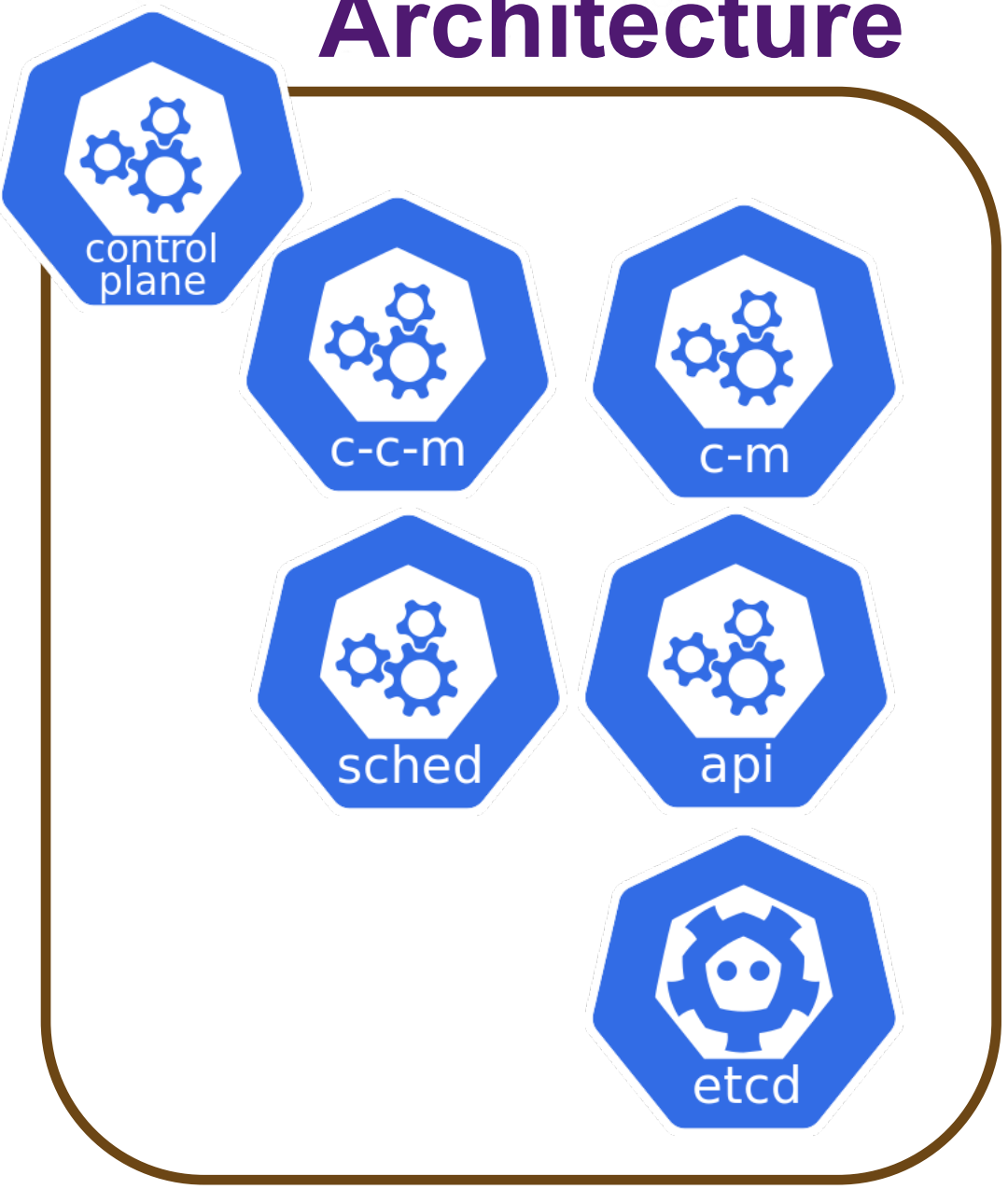
kubernetes

Kubernetes History

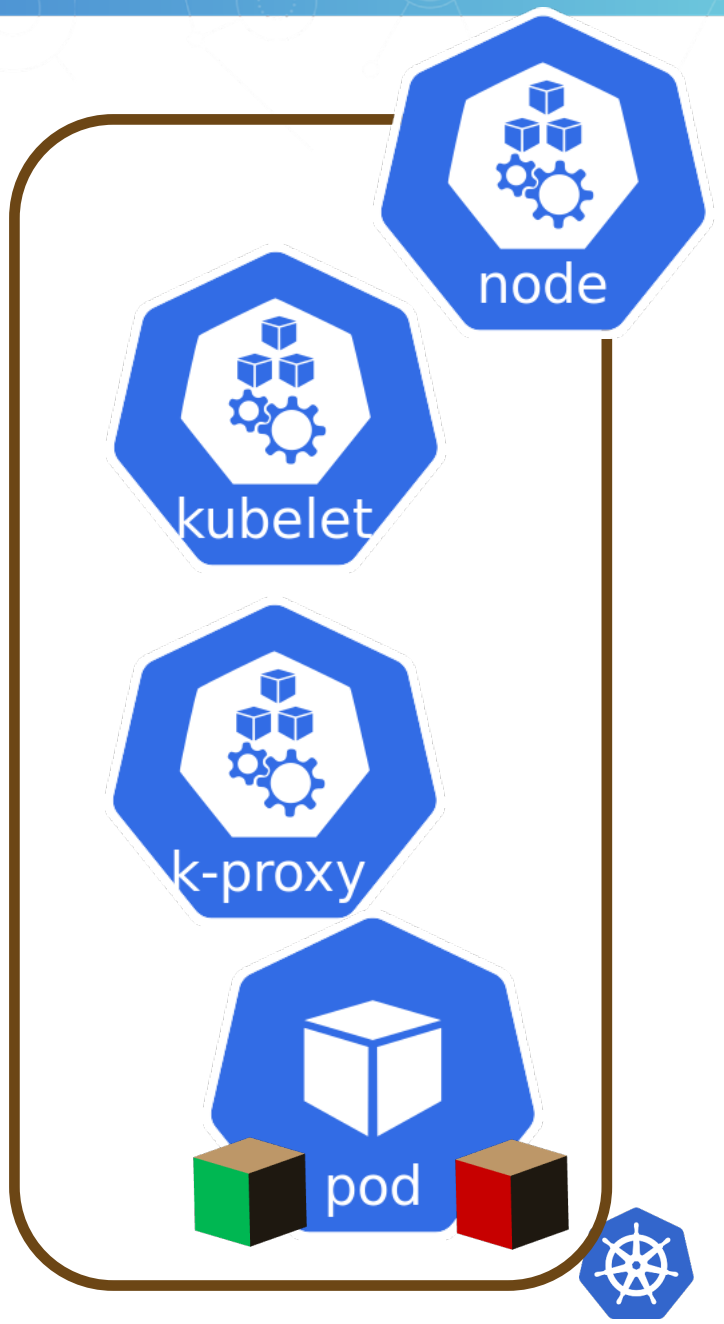
- Public release June 2014
- Created by Google built from lessons learned from Google's Borg & Omega container orchestrators. White paper titled Borg, Omega, and Kubernetes
- June 2015 Open Container Initiative (OCI) - standardization
- July 2015 Kubernetes v1.0. Google and the Linux Foundation helped form the Cloud Native Computing Foundation (CNCF) and donated Kubernetes to the CNCF as the seed project
- 2016 helm, kubeadm released. Notable case study on Pokémon Go on Kubernetes
- 2017 Notable talk at KubeCon + CloudNativeCon Europe Keynote from Open AI titled Building the Infrastructure that Powers the Future of AI
- 10 year birthday on June 6, 2024

kubernetes

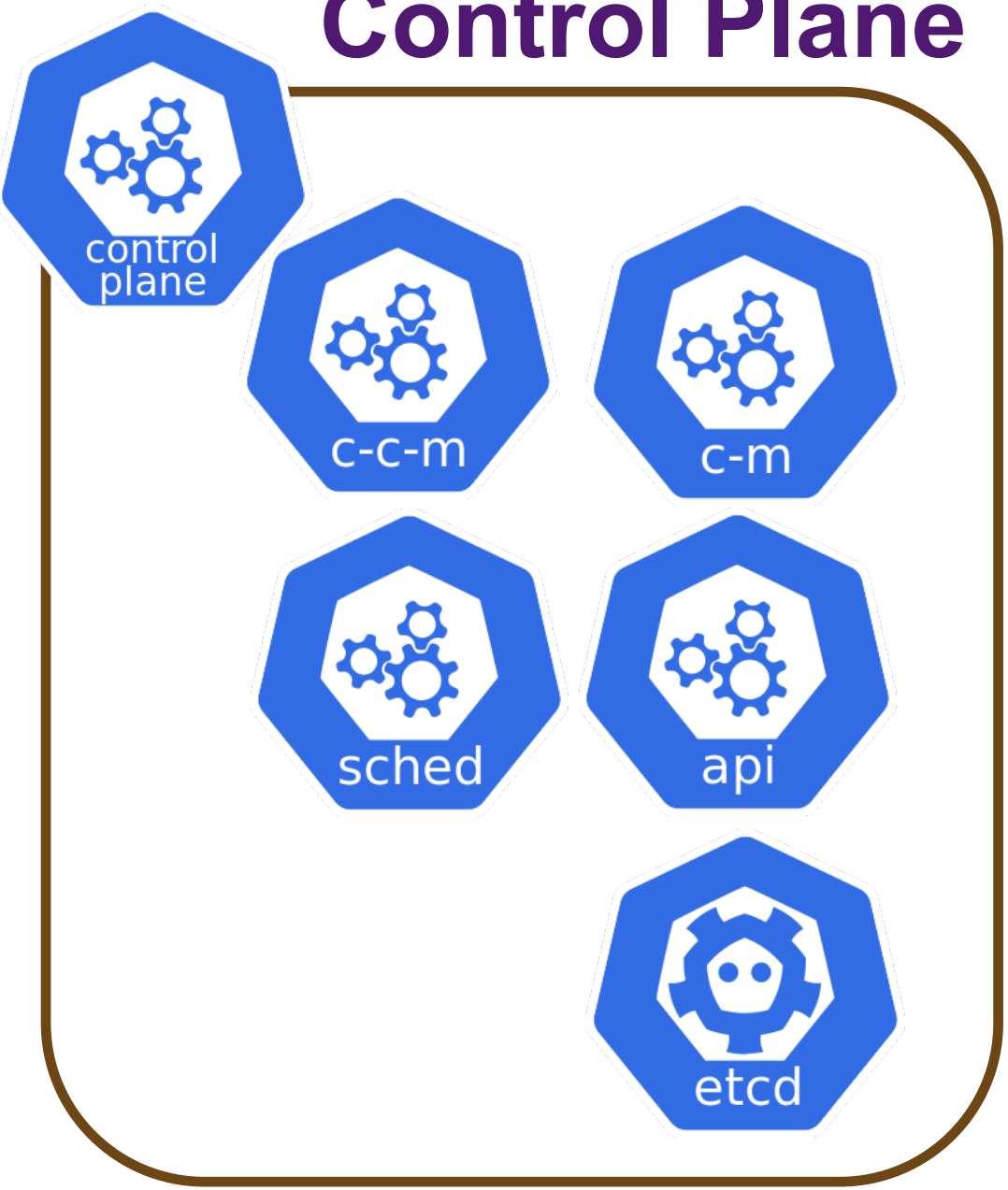
Architecture



kubectl



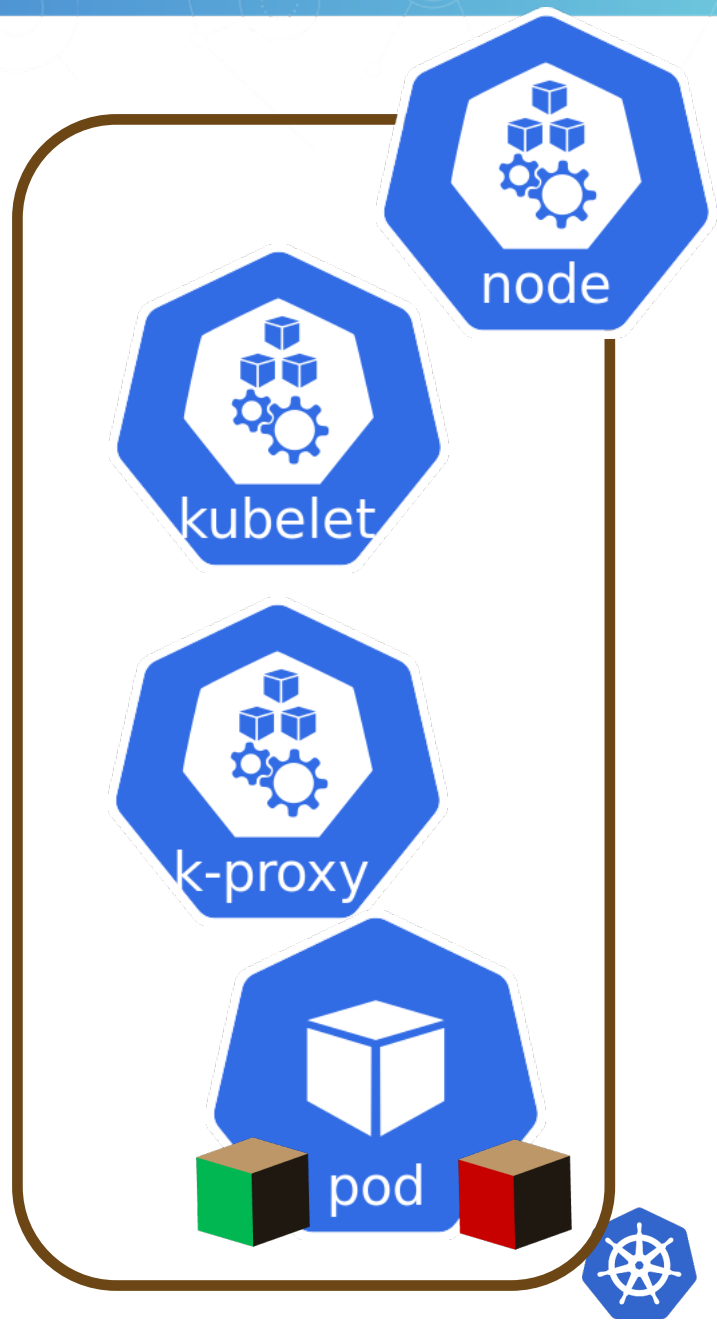
Control Plane



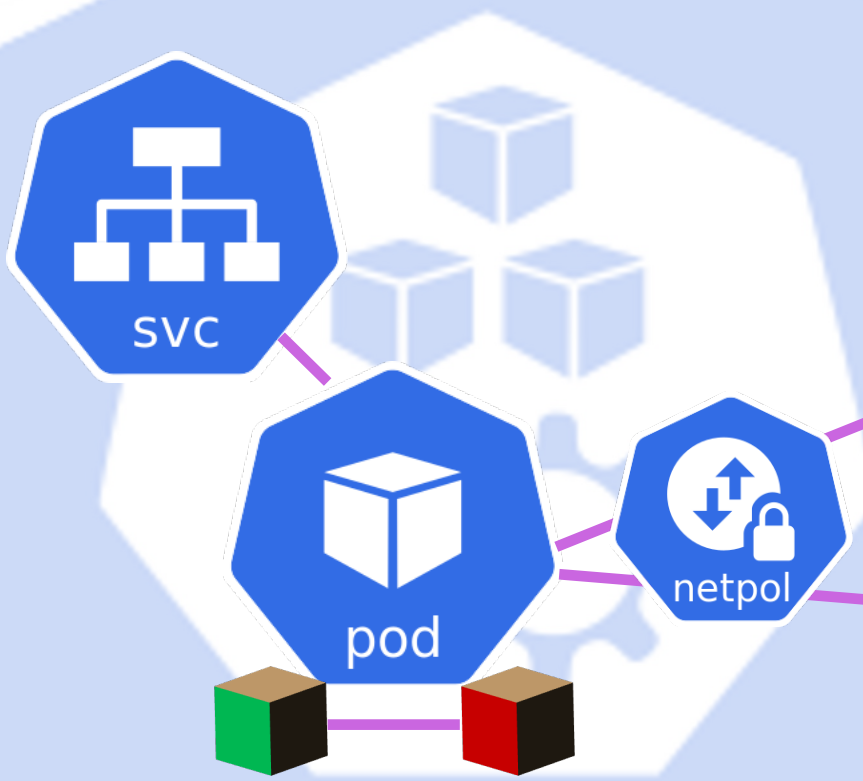
- kube-apiserver – REST interface, handles all interaction between users, nodes, etcd, 3rd party
- etcd – cluster data store (key-value)
- cloud controller manager – provides cloud-provider specific integration capability
- controller manager – daemon that manages all component control loops, monitors cluster state and steers the cluster towards desired state
- scheduler – binds (schedules) pods to nodes

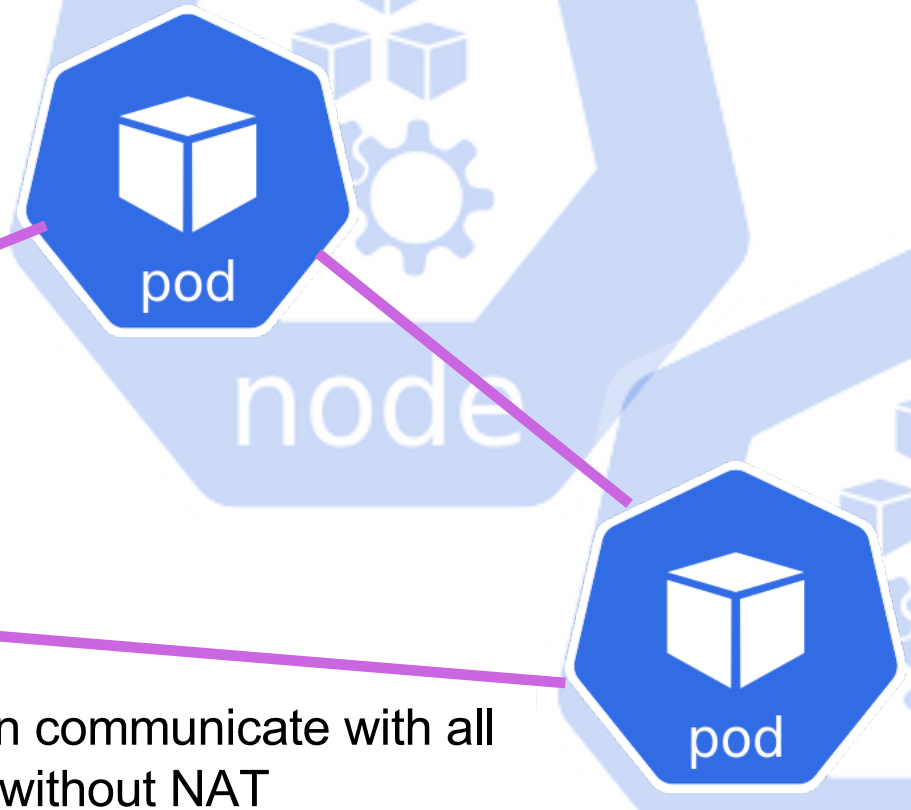
Worker Node

- kubelet – node agent, responsible for Pod lifeycyle
- kube-proxy – manages network rules on each node and performs forwarding and load balancing for Services
- container runtime



Kubernetes Networking





- All Pods can communicate with all other Pods without NAT
- All Nodes can communicate with all Pods without NAT

kubernetes

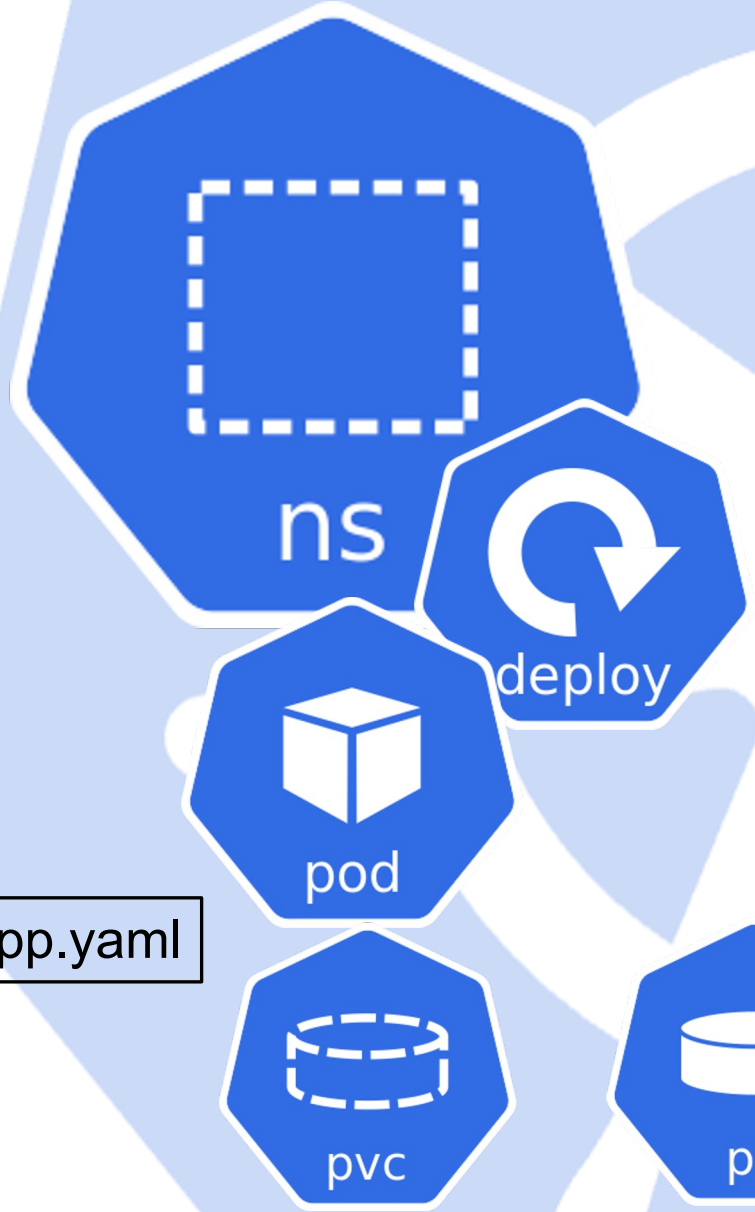
Kubernetes Resources

Deploying on Kubernetes

```go
package main

import (
    "fmt"
    "net/http"
    "os"
    "log"
)

func main() {
    logger := log.New(os.Stdout, "http: ", log.LstdFlags)
    logger.Printf("Server is starting...")
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}

func handler(w http.ResponseWriter, r *http.Request) {
    hostname, err := os.Hostname()
    if err != nil {
        fmt.Println(err)
        os.Exit(1)
    }
    fmt.Fprintf(w, "Hostname is " + hostname)
```

container image

app.yaml



Namespace

```
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespace
```

- Provides isolation of resources
- Divide cluster resources among users, applications, projects, teams
- Can have limits and quotas
- Well-known namespaces:
- - default
- - kube-system – created for the Kubernetes system

- Divide cluster resources with resource quotas
- Limit number of resources e.g. # of Pods, Deployments, Services, etc
- CPU & Memory requests and limits

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: myquota
  namespace: mynamespace
spec:
  hard:
    memory: 2Gi
    pods: “20”
```

kubernetes

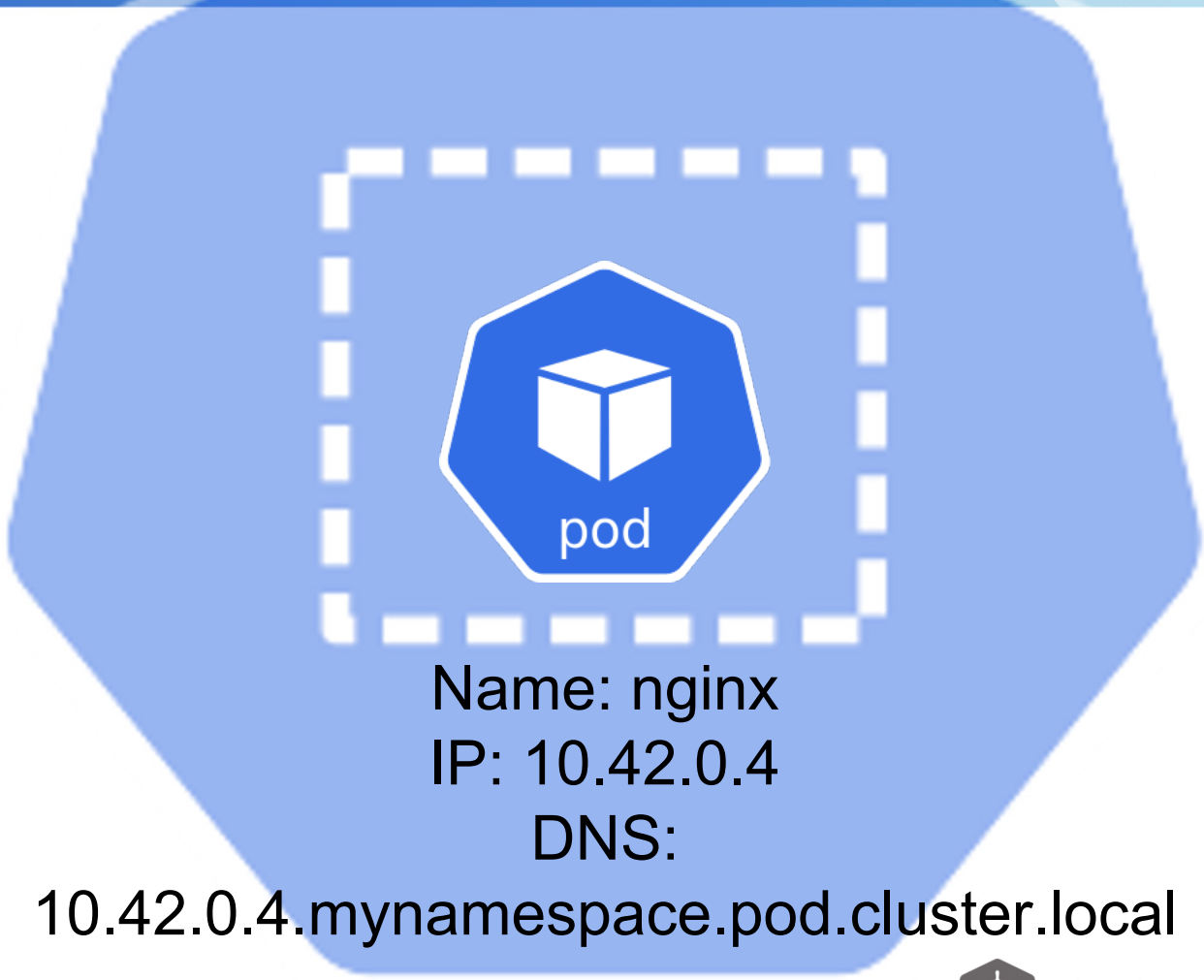
Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace:
mynamespace
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.25.4
    ports:
    - containerPort: 80
```

- Smallest deployable unit
- Composed of 1 or more containers with shared resources like networking, storage
- Pods get a virtual IP (10.42.0.0/16) and a DNS record with the following format: <pod-IPv4>.<namespace>.pod.cluster.local
- Containers of a Pod are co-located

Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace:
mynamespace
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.25.4
    ports:
    - containerPort: 80
```



kubernetes

Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace:
mynamespace
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.25.4
    ports:
    - containerPort: 80
```

Don’t manage Pods manually!

Name: nginx
IP: 10.42.0.4
DNS:
10.42.0.4.mynamespace.pod.cluster.local

kubernetes

Workload Resources

Deployment

Job

DaemonSet

StatefulSet

Cronjob

Deployment

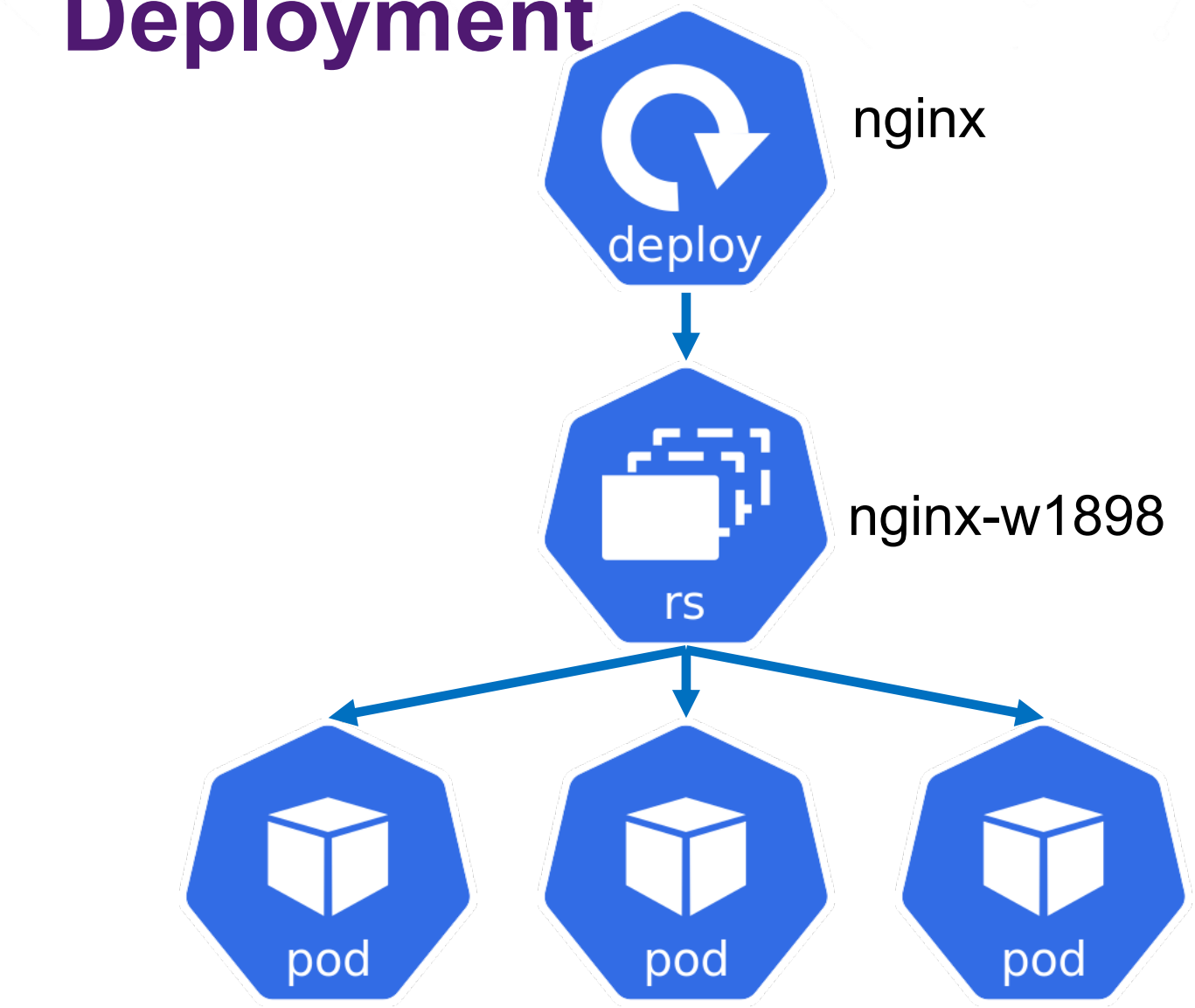
- Declarative updates to Pods (via ReplicaSets) like updates, rollouts, rollbacks
- Set the desired state in a Deployment
- Controller changes the actual state to the desired state at a controlled rate
  - (default) no less than 75% and no more than 125% of the desired number of Pods are up

ReplicaSet

Maintains a stable set of Pod replicas

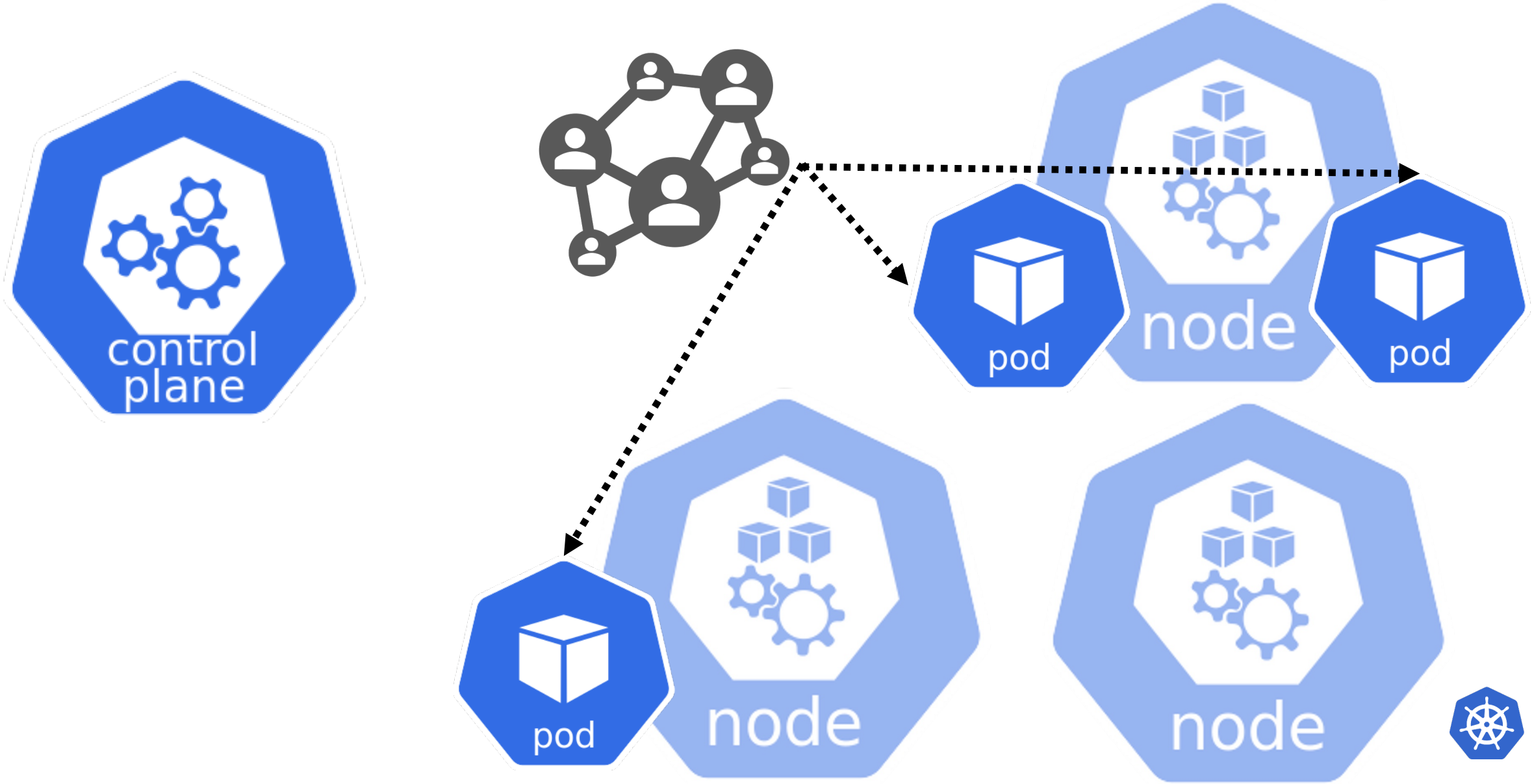
```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
  namespace: mynamespace
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
  matchLabels:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25.4
        ports:
        - containerPort: 80
```

Deployment




```yaml
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
  namespace: mynamespace
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
  matchLabels:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25.4
        ports:
        - containerPort: 80
```

Getting Traffic to Your Application / Microservice

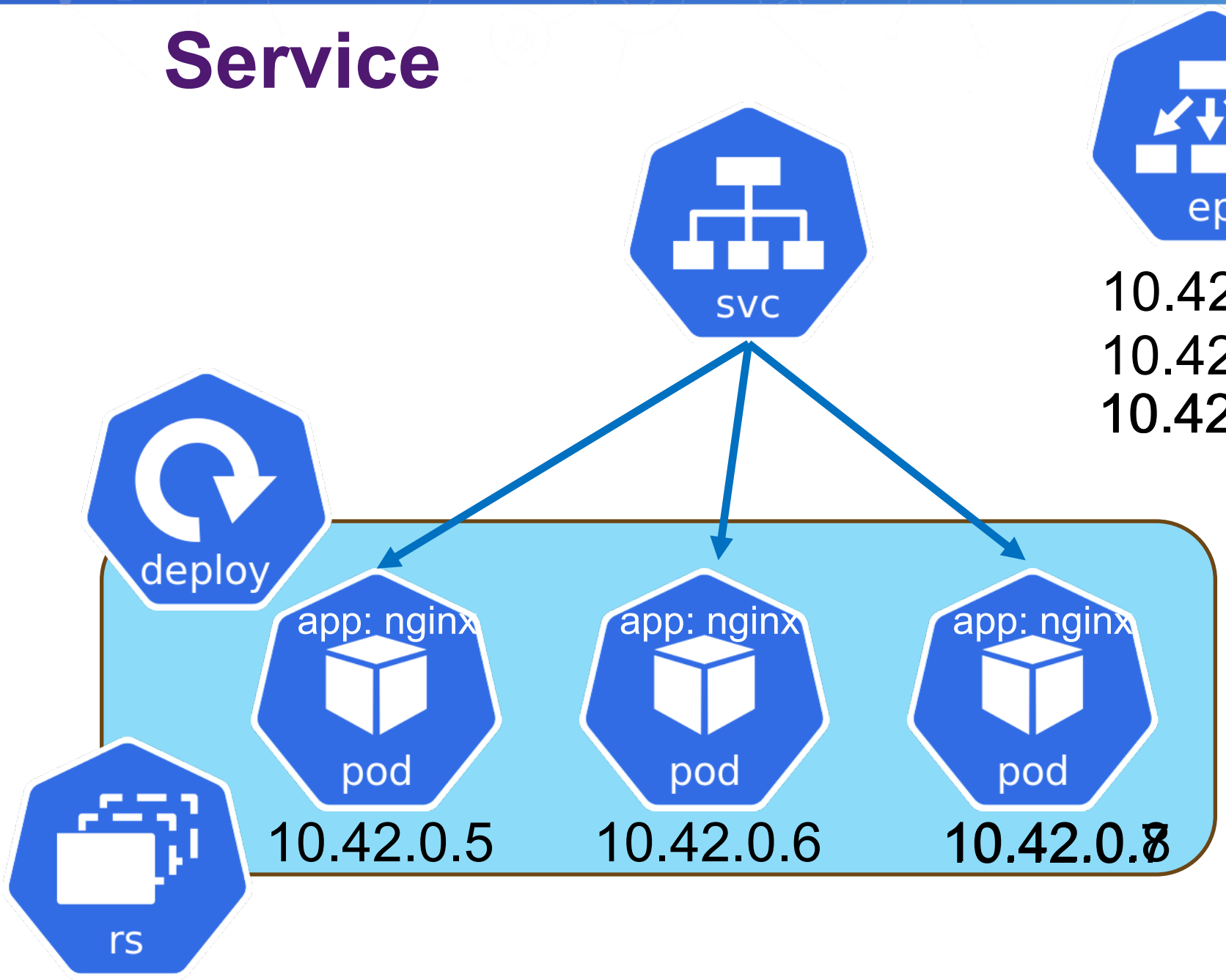


Service

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: mynamespace
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

- An abstraction to expose a group of Pods
- Service defines a logical set of Endpoints (Pods)
- Pods exposed by a Service are targeted with a label selector
- Types of Services:
  - ClusterIP (inter-cluster, CIDR 10.43.0.0/16)
  - NodePort (port 30000 to 32767)
  - LoadBalancer (cloud provider or external LB)

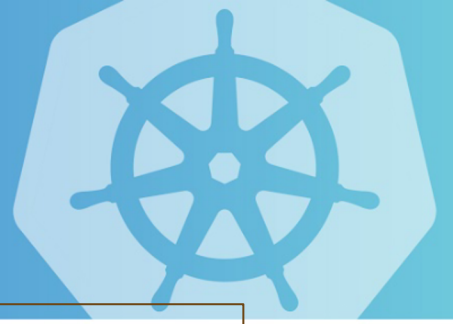
Service



10.42.0.5
10.42.0.6
10.42.0.8

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: mynamespace
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

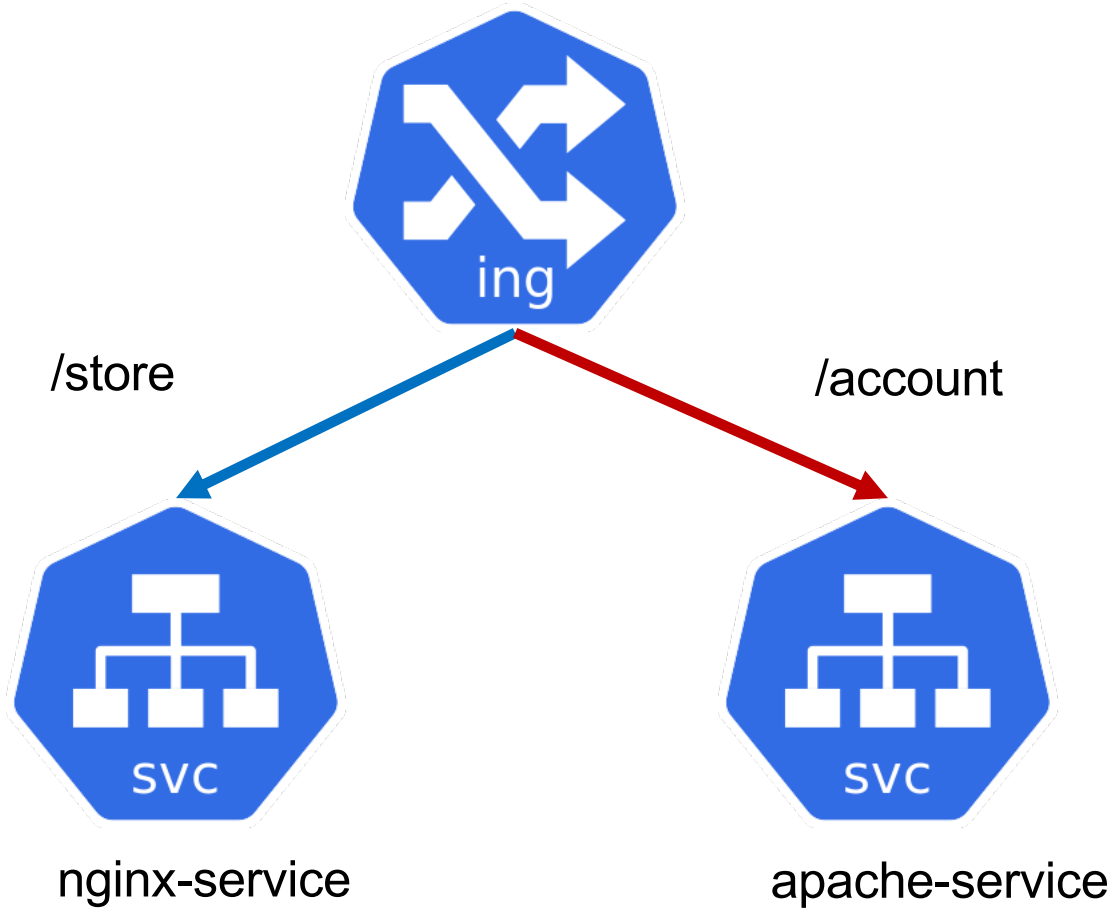
Ingress



- External access to Services over HTTP/HTTPS with an externally reachable URL
- Load balances traffic
- Terminates SSL/TLS

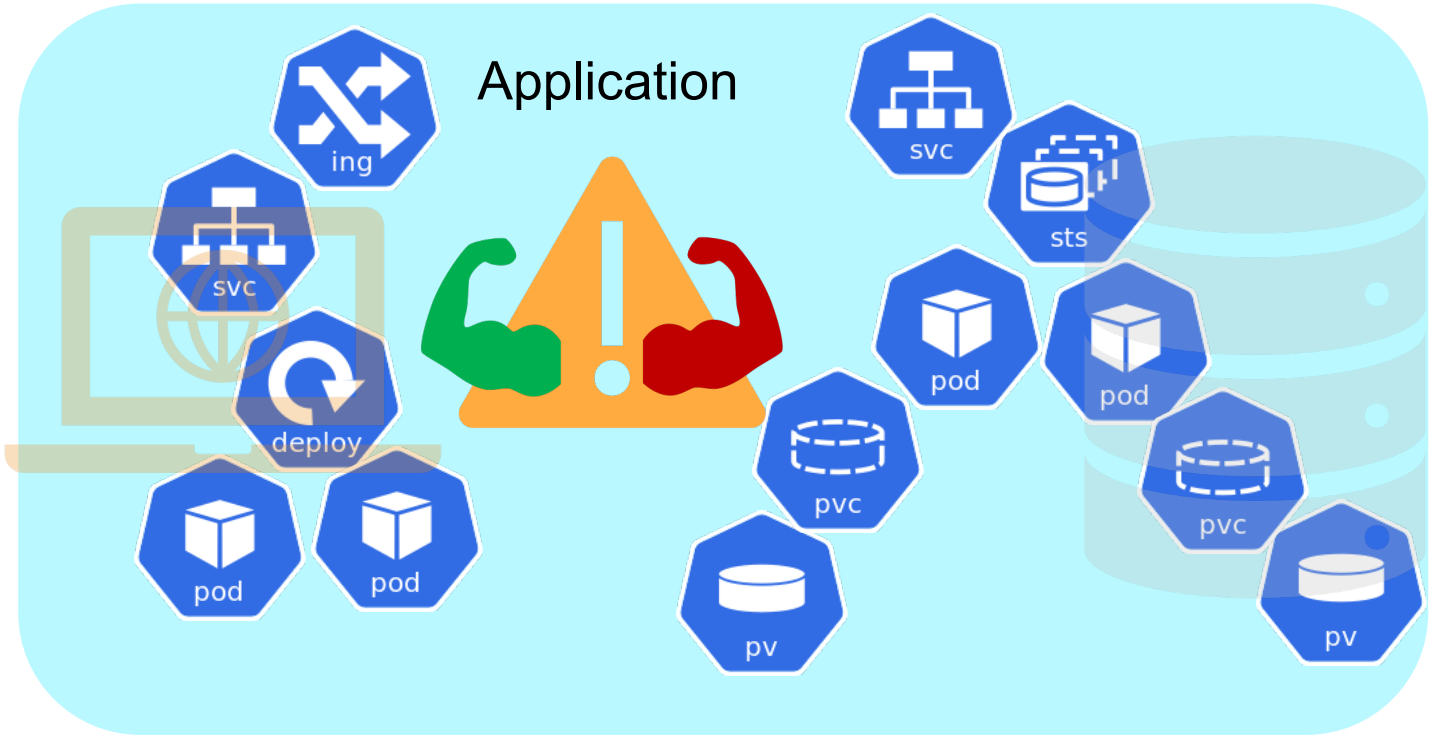
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myapp-ingress
  namespace: mynamespace
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /store
        backend:
          serviceName: nginx-service
          servicePort: 80
      - path: /account
        backend:
          serviceName: apache-service
          servicePort: 8080
```

Ingress




```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: myapp-ingress
  namespace: mynamespace
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /store
        backend:
          serviceName: nginx-service
          servicePort: 80
      - path: /account
        backend:
          serviceName: apache-service
          servicePort: 8080
```

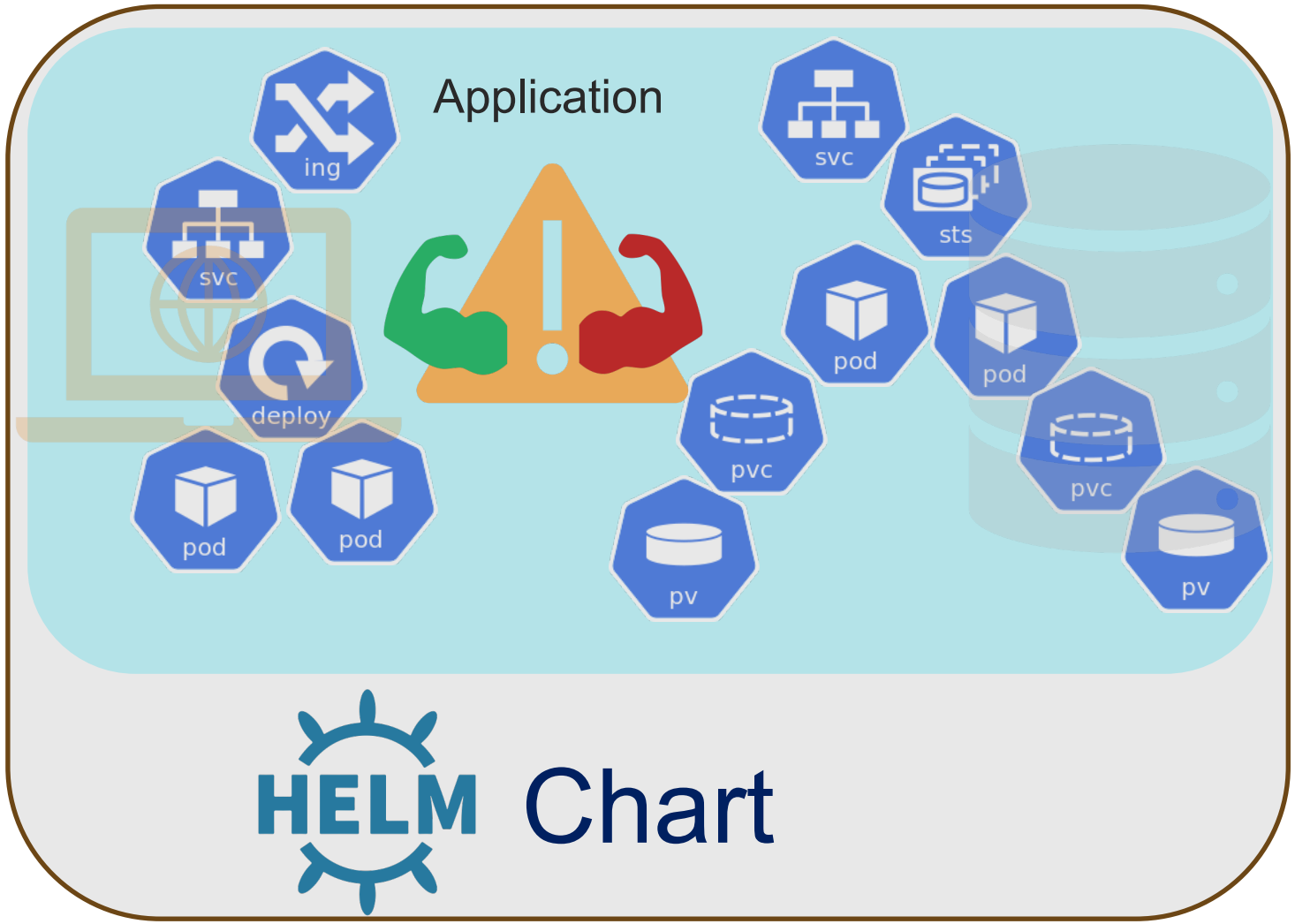
Application Definition



HELM

The “Kubernetes Package Manager”

Application Definition



HELM

The “Kubernetes Package Manager”

Test Environments

minikube

https://minikube.sigs.k8s.io/docs/start/

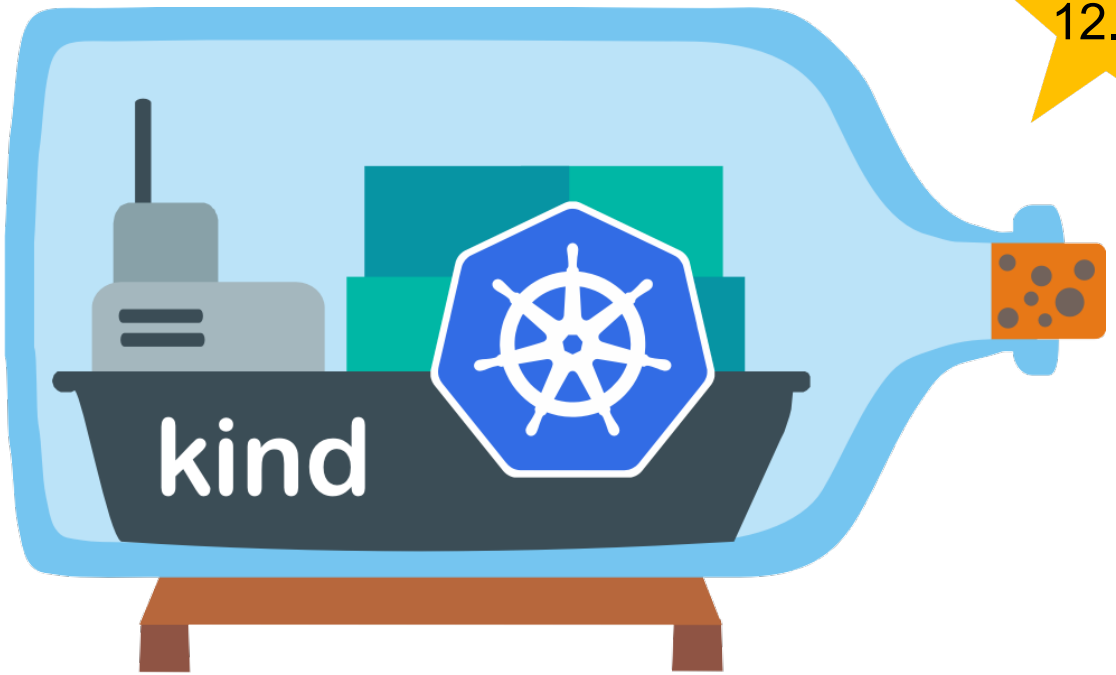
Implements a local cluster on Linux, Windows, macOS

Pre-reqs: container runtime or virtual machine manager

- Podman
- Docker
- QEMU
- Hyperkit
- Hyper-V
- KVM
- Parallels
- VirtualBox
- VMware Fusion/Workstation

kubernetes

kind



12.5k

- Run local Kubernetes clusters using Docker container “nodes” on MacOS, Windows, Linux
- Used by the Kubernetes project to test and run integration tests

Pre-reqs:

- Docker
- kubectl

https://kind.sigs.k8s.io/docs/user/quick-start

kubernetes

K3s

K3S

https://k3s.io/

- CNCF Project
- Great for resource-constrained environments
- Linux only
- Uses an etcd shim called kine (kine is not etcd)
- Can use etcd, PostgreSQL, MySQL, SQLite, MariaDB, NATS JetStream

kubernetes

K3s

- Wrapper around k3s to run in Docker

Pre-reqs:

- Docker
- kubectl

https://k3d.io/

kubernetes

Tutorial

Tutorial

https://github.com/reylejano/kubernetes-essentials-devweek

- Deploy a cluster
- Deploy containerized application in a Pod
- Deploy the containerized application in a Deployment
- Update the application
- Scale the Deployment up and down
- Get traffic to our application from outside the cluster

https://github.com/reylejano/kubernetes-essentials-devweek

kubernetes

Thank you