Kubernetes in Practice

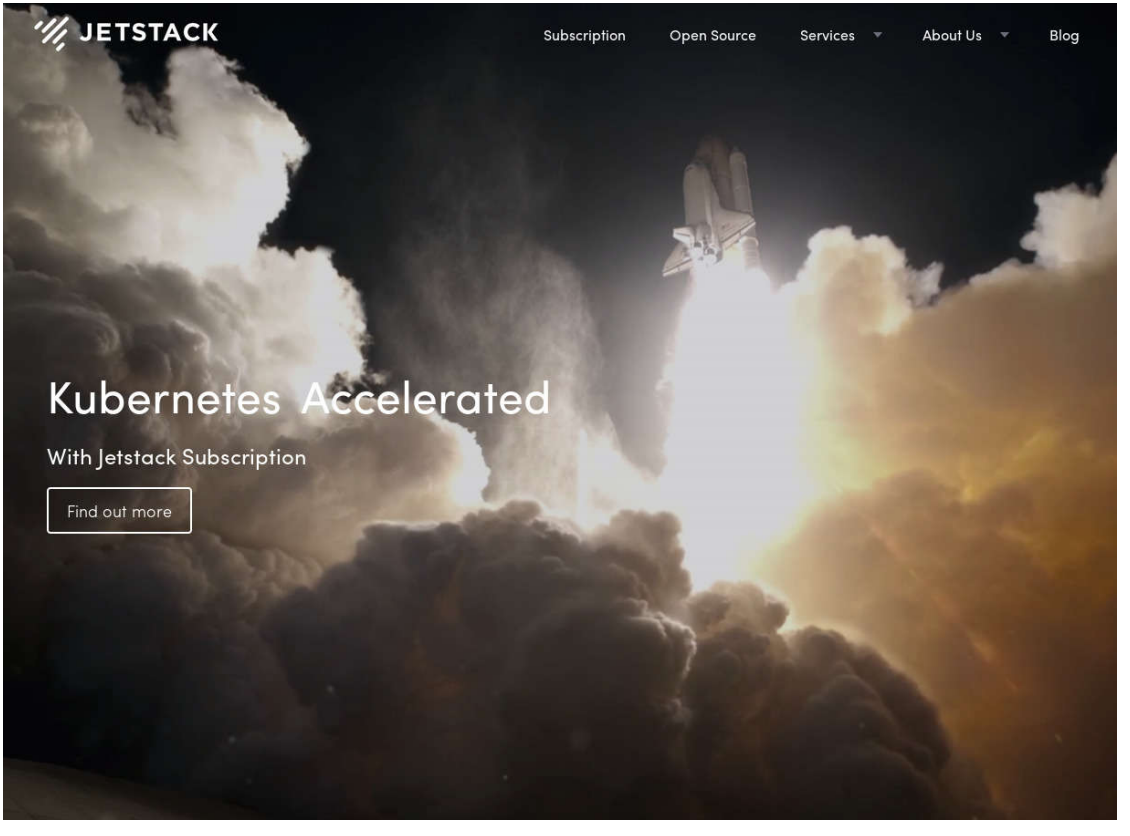
Intermediate

Schedule

Action packed and fast moving

- Morning
  - Kubernetes recap
  - Setup
  - Control-plane deep dive
  - Auto-scaling (pods and cluster)
- Afternoon
  - Advanced Scheduling
  - Ingress & TLS
  - StatefulSet
  - CI/CD with GitLab
  - Monitoring

Welcome


JETSTACK
Subscription
Open Source
Services
About Us
Blog
Kubernetes Accelerated
With Jetstack Subscription
Find out more

Welcome

Each person say:

- Hello & where they are from
- How much k8s experience they have
- The main thing they want to learn
- Favourite something computer related
- Favourite something not computer related

Workshop

1a. Install. Get everything installed and setup on your laptop

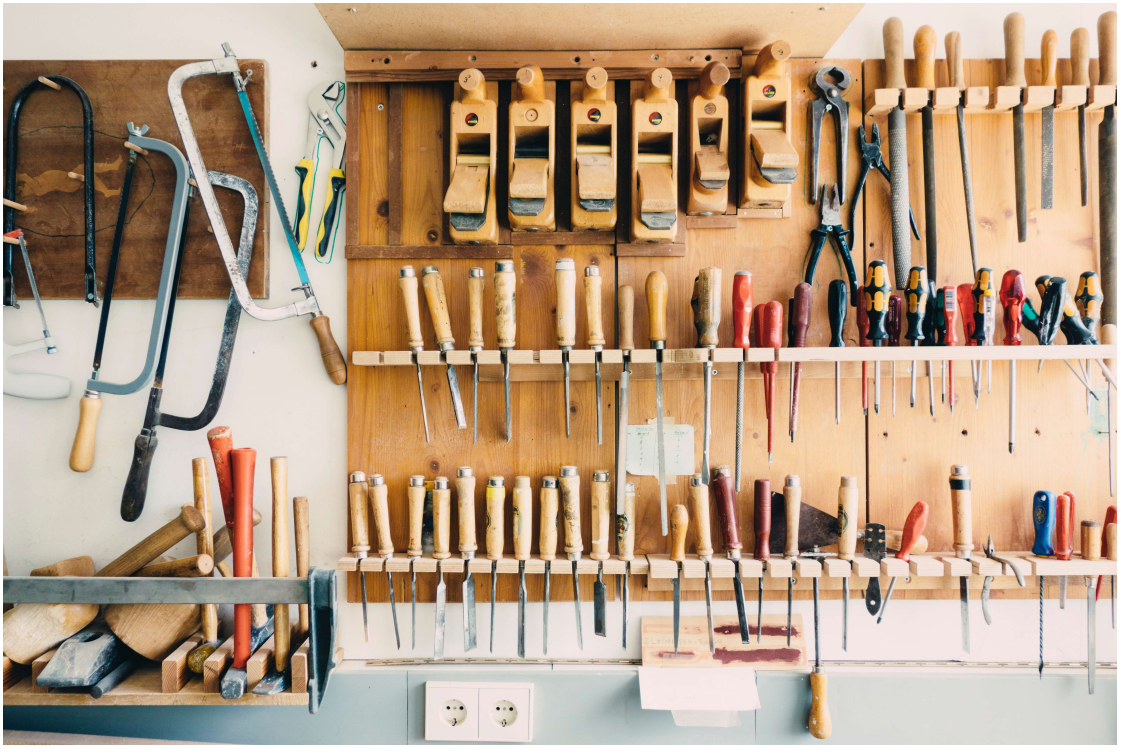


Photo by Barn Images on Unsplash

Workshop

- Browse to the workshop site
- URL: **https://intermediate.k8s.school**
- Username: **intermediate**
- Password: **k8sworkshop**
- Download workshop materials (top right)

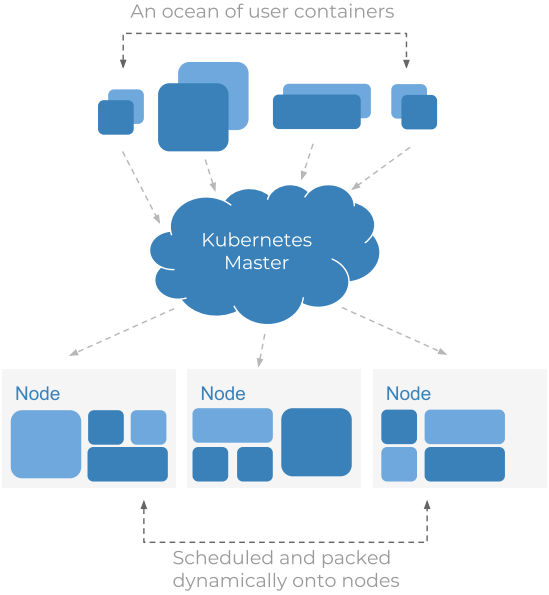
Recap

How much can you remember?

Recap

Overview

- Kubernetes handles server ‘Cattle’ to pick and choose resources
- Can be installed on many different types of infrastructure
- Abstracts away the servers so developers can concentrate on code
- Pro-actively monitors, scales, auto-heals and updates



Pod

The basic unit of Kubernetes

- A pod is a collection of containers
- Some namespaces are shared (eg. network, from 1.8 PID too)
- All containers in a pod share fate

ReplicaSet

Ensure N instances of a pod with a template exist

- Often used as a building block for other resources
- Does not provide rolling upgrade functionality
- Define a pod ‘template’ and it will ensure N replicas exist

Deployment

Ideal for ‘cattle’ services

- Deployments manage the lifecycle of ReplicaSets
- Can be used to perform rolling upgrades of a service
- Two replicas don’t have any differentiation. They are truly ‘replicas’
- Very common resource type

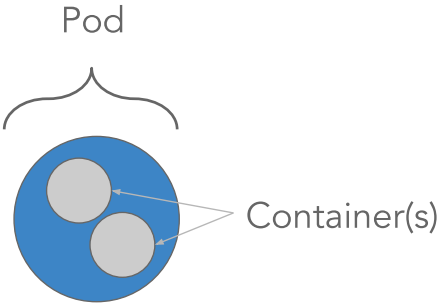
Services

Used to route L4 around your cluster

- Provides a basic primitive for load balancing
- Can be used to expose services either internally or externally
- Automatically provision TCP/UDP load balancers in your cloud

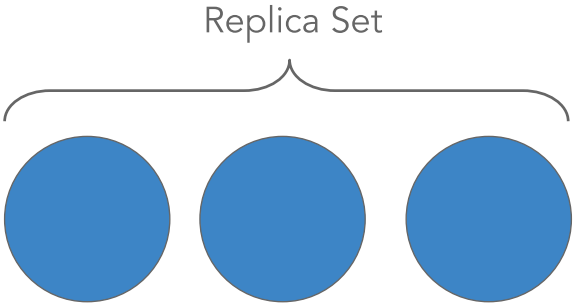
Workloads

Pods and containers



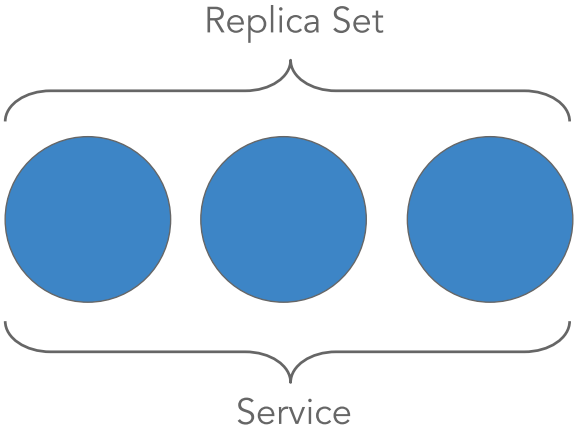
Workloads

Replica Set



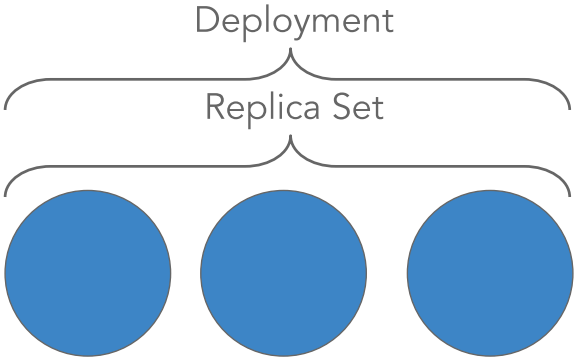
Workloads

Services



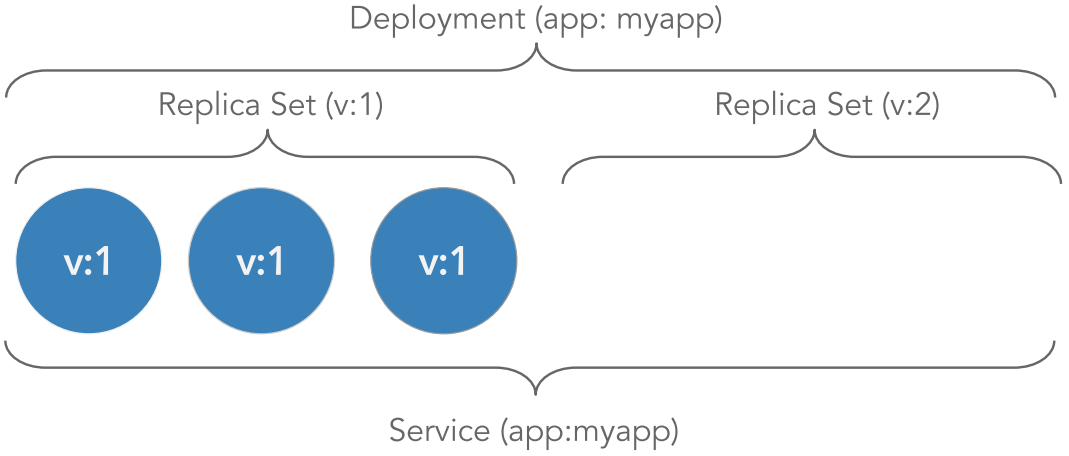
Workloads

Deployment



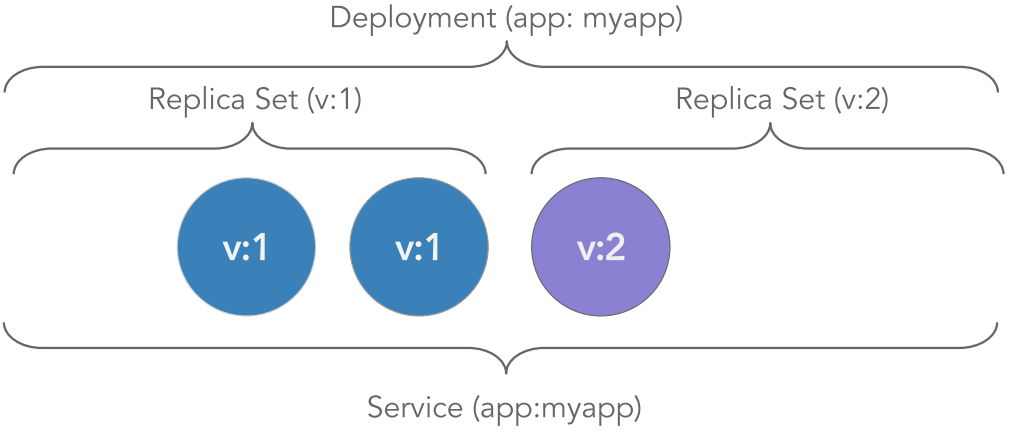
Workloads

Rolling Upgrade



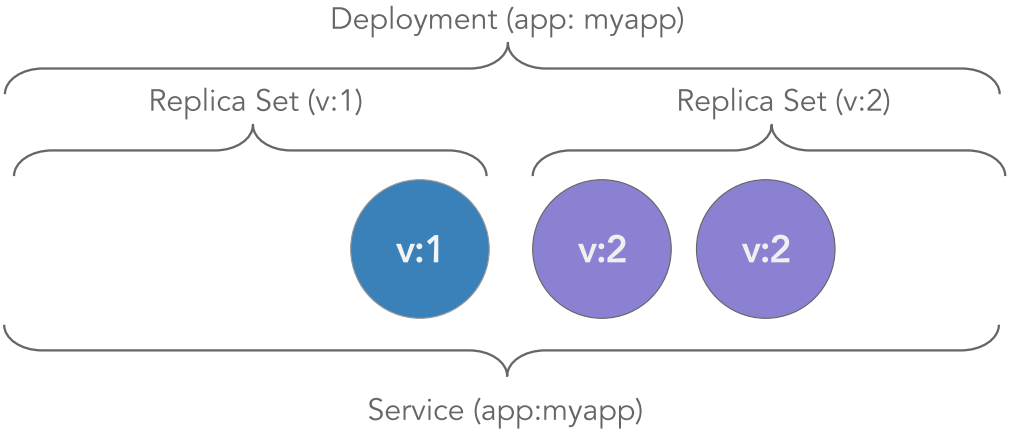
Workloads

Rolling Upgrade



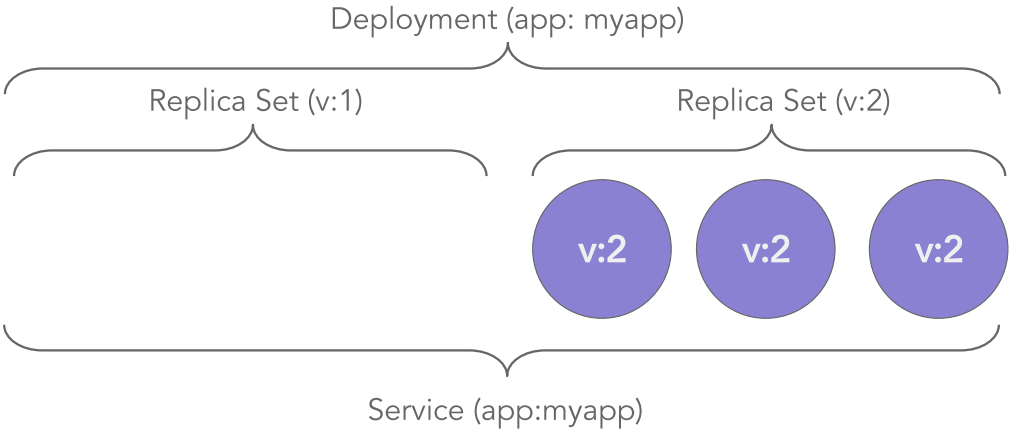
Workloads

Rolling Upgrade



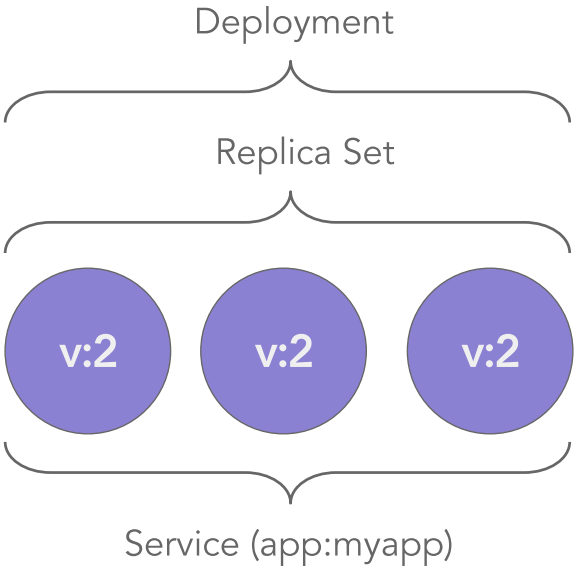
Workloads

Rolling Upgrade



Workloads

Rolling Upgrade



StatefulSet

Ideal for ‘pet’ services

- Ensure N replicas of a service exist
- Each pod has ‘identity’ (ordinal), eg. pod-0, pod-1, pod-2
- Stable hostname
- Persistent storage per pod
- Ideal for distributed systems/clustered databases

DaemonSet

Run an instance of a pod on each node

- Run one, and only one instance of a pod on each node
- Can use nodeSelector to target specific types of nodes
- Great for running system services (eg. kube-proxy)

Ingress

Expose L7 HTTP to the outside world

- HTTP-aware routing of traffic
- Can be used to enable TLS on services
- Path-based routing
- Host-based routing
- Implemented out of tree (eg. nginx-ingress, gclb-controller)

PersistentVolumeClaim

Represent a desire to consume persistent storage

- Provides an abstraction between persistent disks and applications
- Instead of specifying the type of volume, just ‘claim’ some space
- Claims are bound to particular PVs, or dynamically provisioned

ConfigMap

Decouple application configuration from your images

- Can be mounted into a container as a volume
- Or used as environment variables
- Easy way to mount configuration files into containers

Secret

Store sensitive data

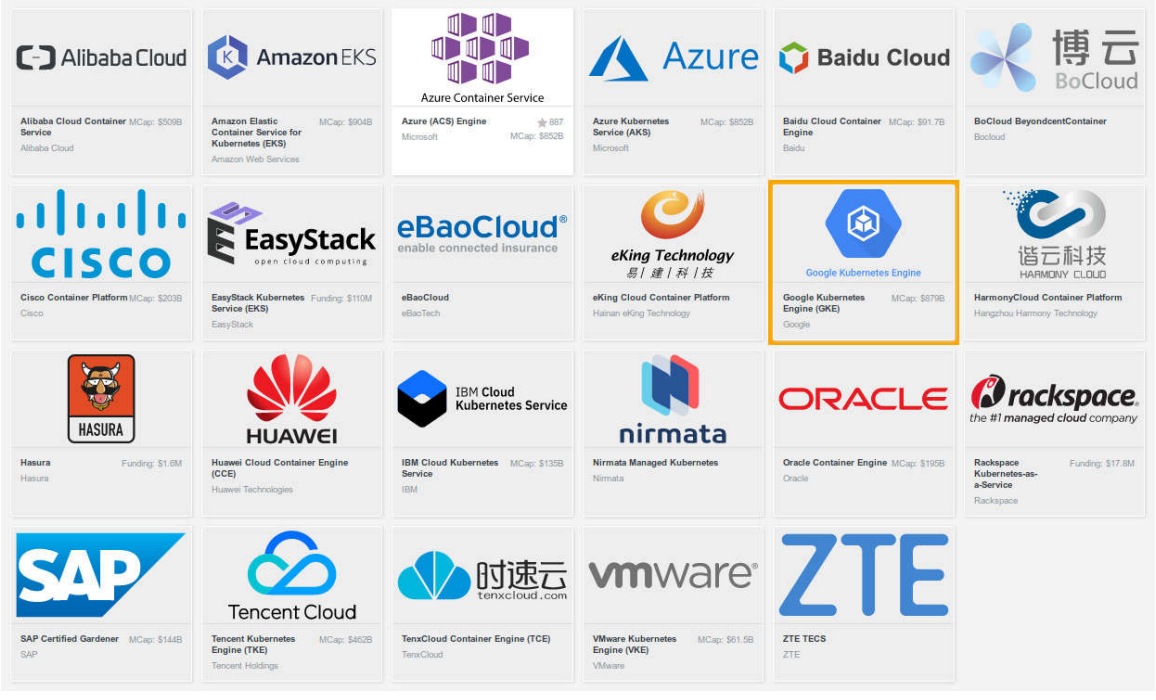
- Can be mounted in as a volume
- Will be stored on a ramdisk if used as a volume
- Or used as environment variables

Infrastructure

I need a cluster

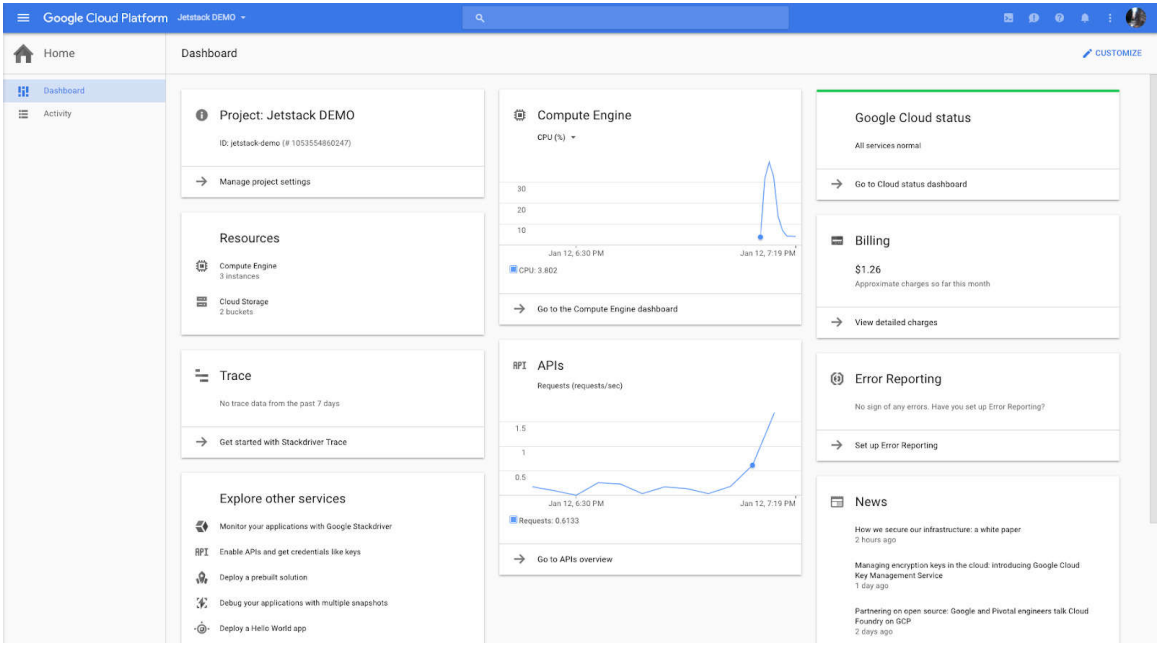
Infrastructure

Kubernetes is designed to be infrastructure agnostic



Infrastructure

We're going to be using GKE



Workshop

1b. Get a three node Kubernetes cluster and kubectl it

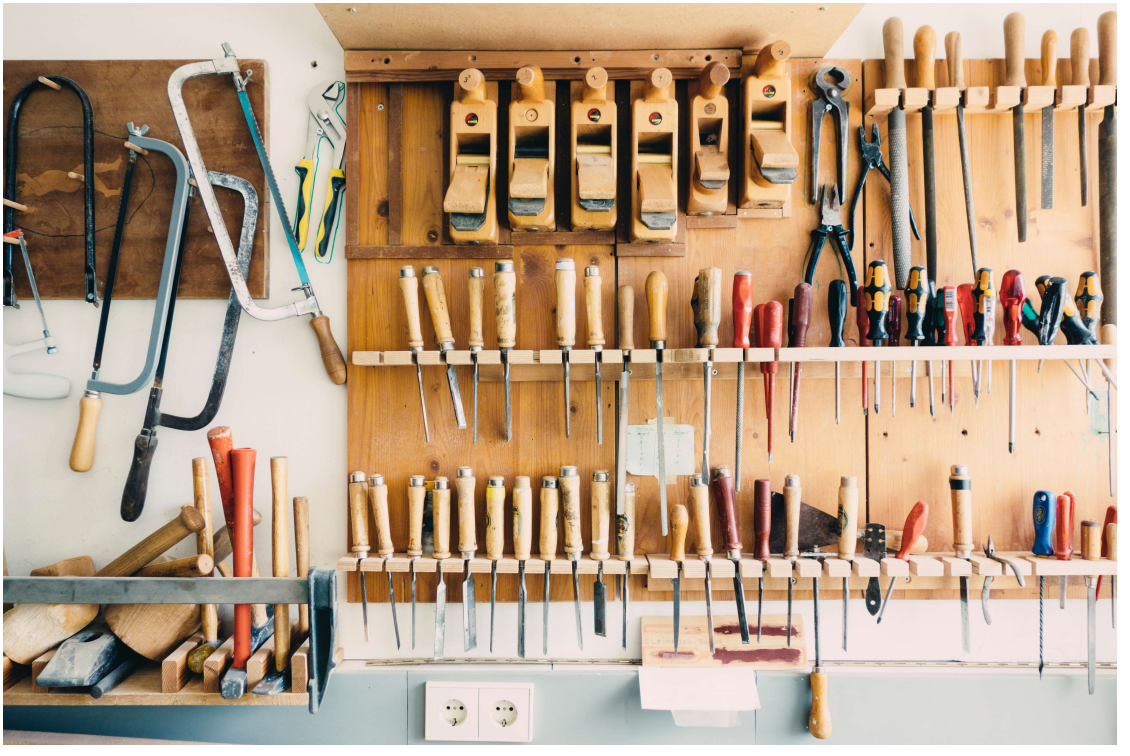


Photo by Barn Images on Unsplash

What makes a Cluster

Let's dig into what we just created

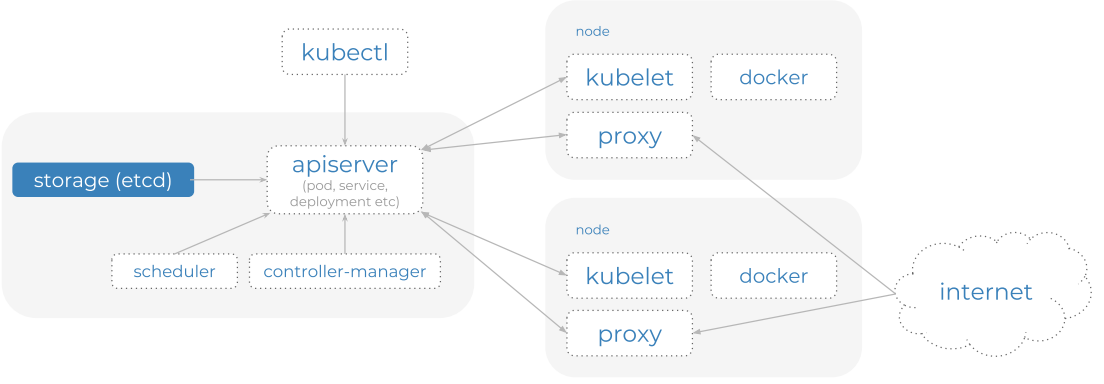
What makes a Cluster

What did we just create?

Client	
· kubectl	CLI client for the api server
Control plane	central cluster orchestration processes
· apiserver	REST API for kubectl
· scheduler	bind unscheduled pods to nodes
· controller-manager	perform cluster level operations
· etcd	keeps cluster state
Nodes	processes that run on each node in the cluster
· kubelet	agent that talks to the apiserver
· proxy	enforces services by editing iptables
· docker	container runtime

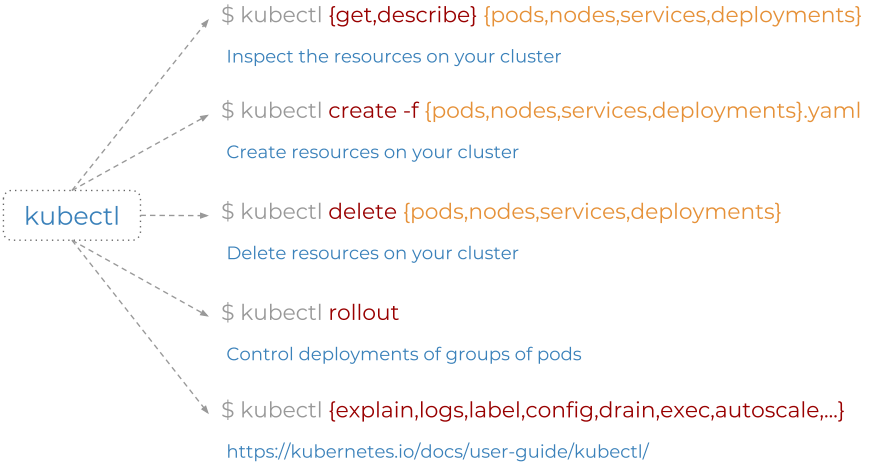
What makes a Cluster

Overview



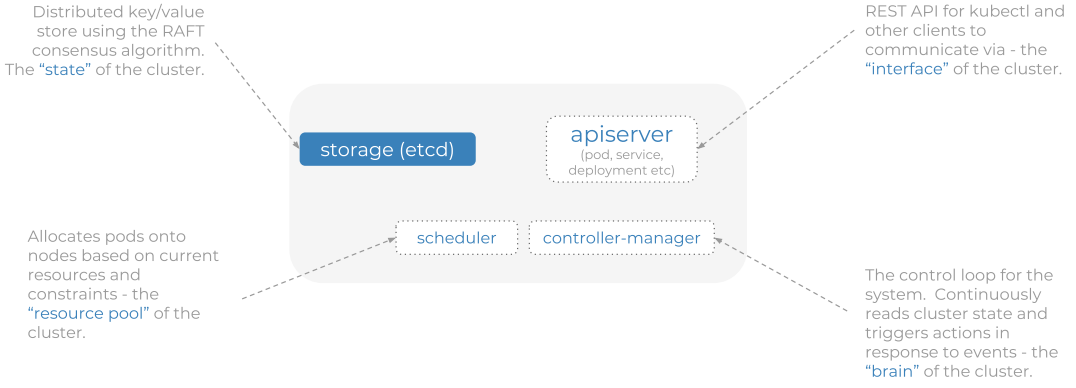
What makes a Cluster

kubectl



What makes a Cluster

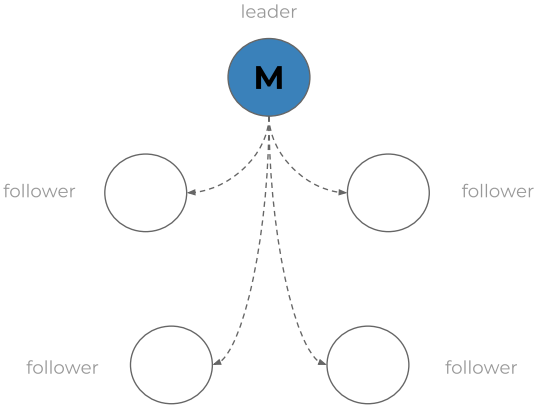
Control plane



What makes a Cluster

Control plane: etcd

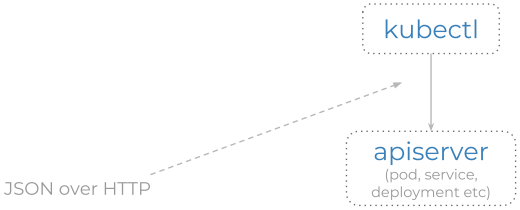
- Distributed HA configuration database
- CoreOS project
- **C**onsistent + **P**artition Tolerant
- Based on Raft consensus algorithm



What makes a Cluster

Control plane: apiserver

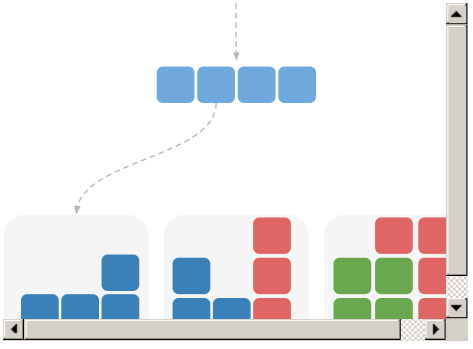
- Kubectl is a REST api client for the apiserver
- Programmatically control the k8s API from a language of your choice
- Admission controllers
- {GET,POST,PUT,DELETE} /api/v1/{pods,services,deployments,...}
- {GET,POST} /api/v1/namespaces
- {GET,POST,PUT,DELETE} /api/v1/namespaces/{namespace}/{pods,services}
- https://kubernetes.io/docs/api-reference/v1.5



What makes a Cluster

Control plane: scheduler

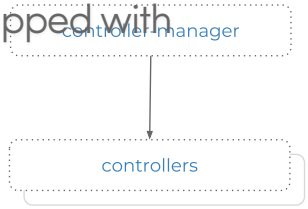
- Watch for pods with an empty Pod.Spec.NodeName
- Apply “predicates” to filter out inappropriate nodes
- Apply “priority functions” to rank the nodes
- The node with the highest priority is chosen



What makes a Cluster

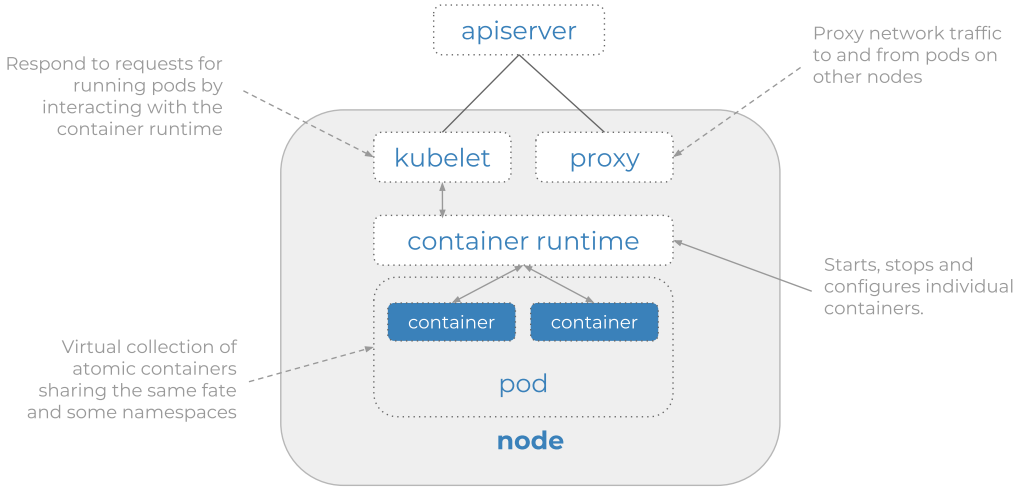
Control plane: controller-manager

- Daemon that embeds the core control loops shipped with Kubernetes
- Example controllers:
  - replication
  - endpoints
  - namespace
  - serviceaccounts
  - node
- For example, nodes are discovered, managed, and monitored by the node controller

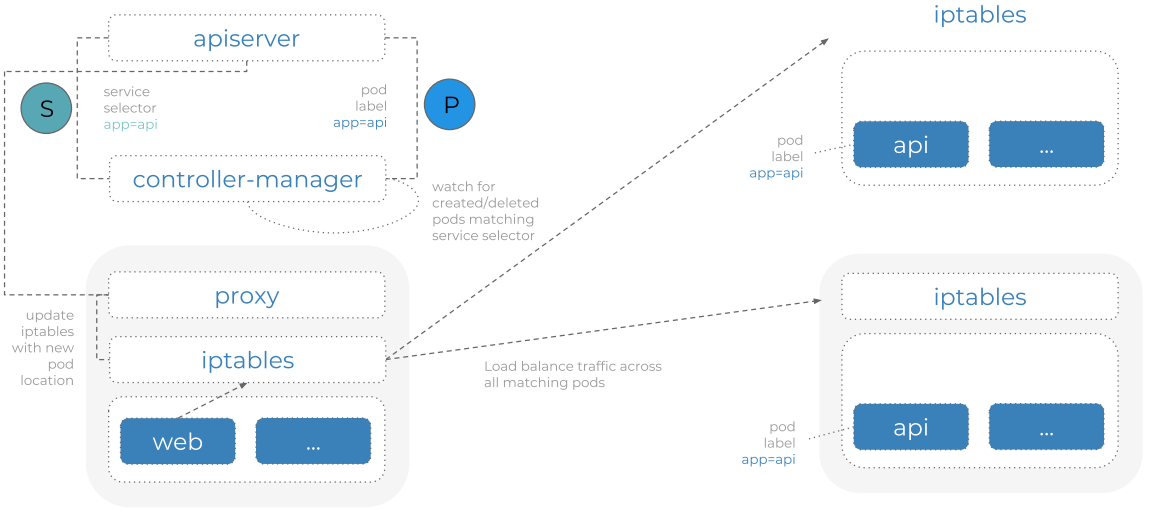


What makes a Cluster

Node



What makes a Cluster



Workshop

2. Deploy the entire stack as declarative deployments

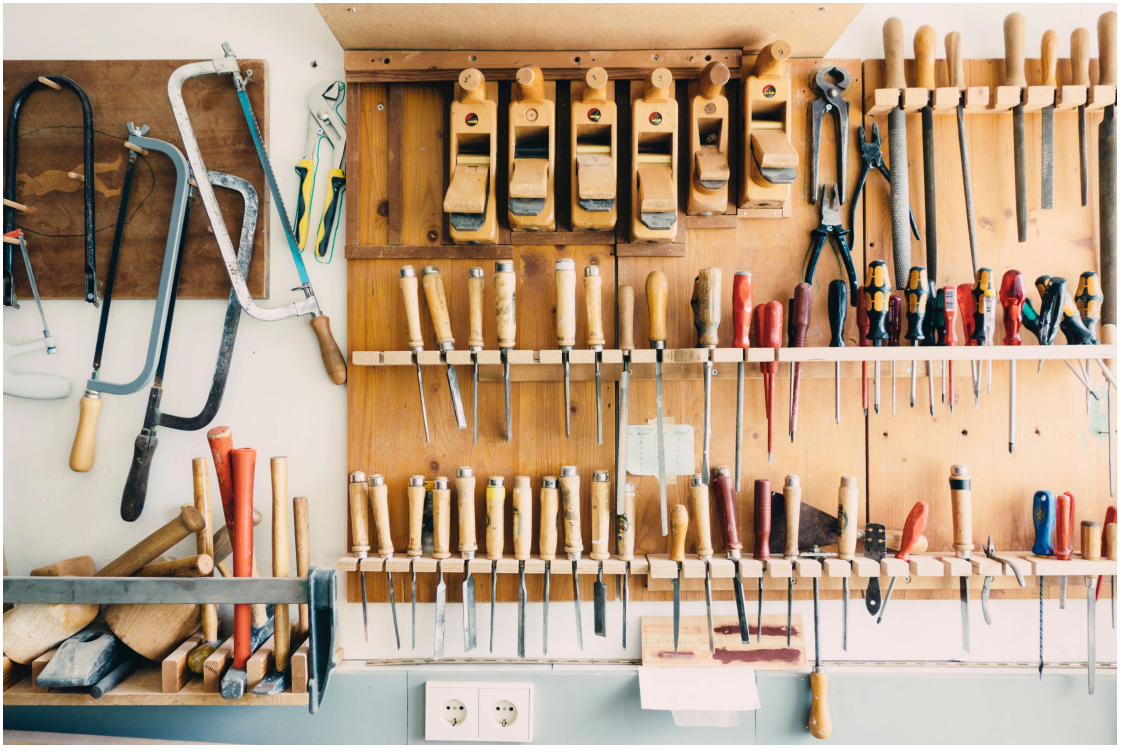


Photo by Barn Images on Unsplash

App Autoscaling

How to scale wordloads in-cluster to meet demand

App Autoscaling

Allow your **applications** to scale!

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: my-autoscaler
spec:
  minReplicas: 2
  maxReplicas: 5
  targetCPUUtilizationPercentage: 60
  scaleTargetRef:
    kind: Deployment
    name: my-deployment
    apiVersion: apps/v1
```

App Autoscaling

- HorizontalPodAutoscaler resource
- Avoids re-implementing scaling logic
- Separate resource that can scale many types of things
- v1 supports CPU based scaling
  - Relies on Heapster for metrics
  - Initial Heapster deprecation with Kubernetes 1.11
  - Removal of Heapster with Kubernetes 1.13
- v2beta1 supports much more

App Autoscaling

App Autoscaling

v2beta1 custom metrics

- Custom metrics allow us to define application specific metrics to scale our application
- Could be based on **anything**

  Implemented via aggregated APIs

  - metrics.k8s.io metrics-server, Prometheus
  - custom.metrics.k8s.io Prometheus, Stackdriver, ...
  - external.metrics.k8s.io Stackdriver
- 'Teach' k8s about your application
- Proposals: http://github.com/kubernetes/community/blob/master/contributors/des proposals/instrumentation

App Autoscaling

```yaml
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: my-autoscaler
spec:
  minReplicas: 2
  maxReplicas: 5
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 50
  - type: Pods
    pods:
      metric:
        name: requests-per-second
      targetAverageValue: 2k
```

Workshop

3a. Create an HPA resource for the front-end deployment

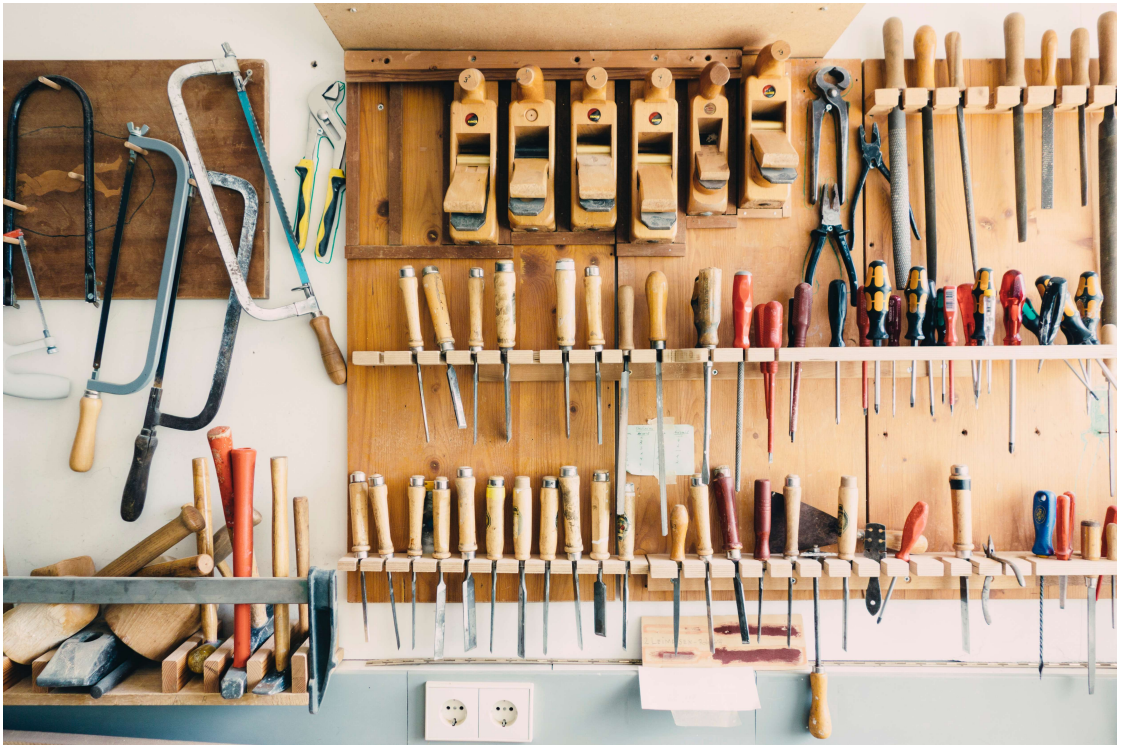


Photo by Barn Images on Unsplash

Cluster Autoscaling

Provision more compute in response to scaling pods

Cluster Autoscaling

Allow your **cluster** to scale!

- 'cluster-autoscaler' project
- Watches cluster for unschedulable pods, triggers scale up or down in response
- Won't scale on utilisation, only schedulability
- Supports GCP, GKE, AWS and Azure right now
- https://github.com/kubernetes/autoscaler

Cluster Autoscaling

Allow your **cluster** to scale!

- Not immediately responsive - it can take up to 5 mins to scale up, and even longer to scale down
- Won't scale down within 10 mins of a scale up event
- Cannot vertically scale (see: vertical pod autoscaler, in early alpha)
- CPU-based autoscalers are not effective for k8s. Properly set requests, HPAs and scheduling-based scaling.

Workshop

3b. Create a cluster autoscaler

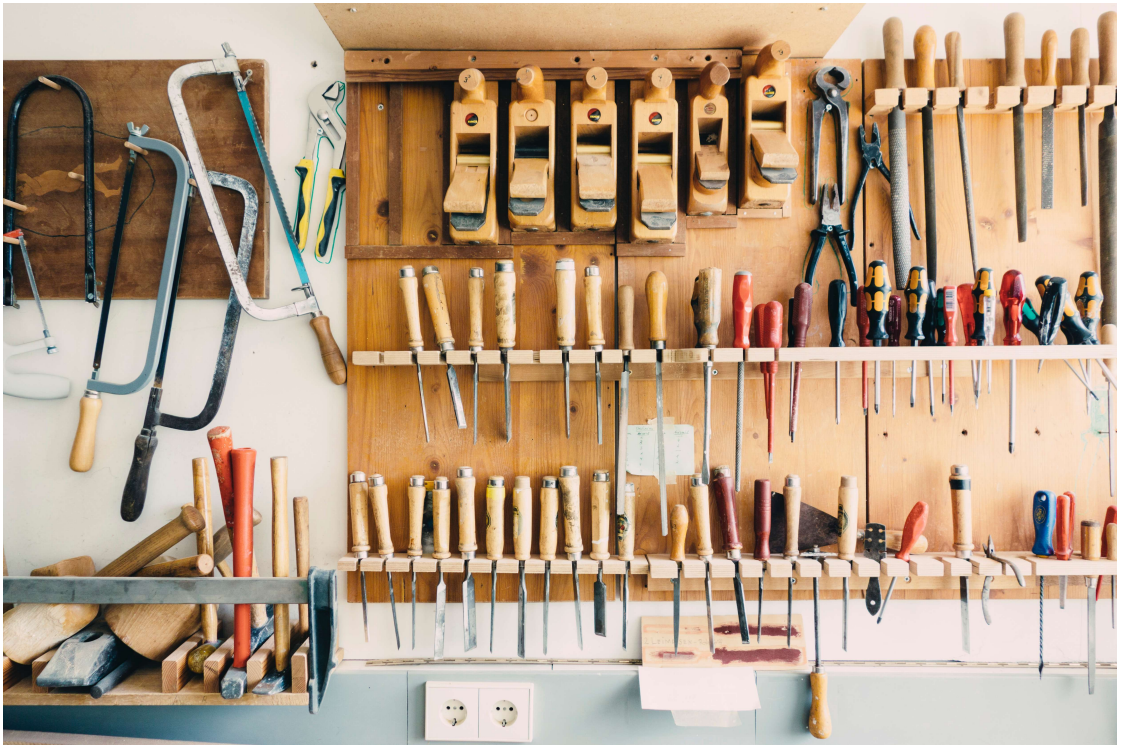


Photo by Barn Images on Unsplash

Scheduling

Learn and control how workloads are allocated to nodes

Scheduling

Influence the default scheduler's behaviour

- There are a number of ways to influence the behaviour of the scheduler
- This can be used to create multi-tenant, or multi-node-type clusters
- Run your cluster with mixed node types and use them efficiently
- Inter pod (anti)-affinity, node affinity, taints/tolerations

Scheduling

Pod [anti-]affinity

- Set 'rules' for how to schedule your pods based on labels of **other** pods
  - requiredDuringSchedulingIgnoredDuringExecution – 'hard'
  - preferredDuringSchedulingIgnoredDuringExecution – 'soft'
  - requiredDuringExecution equivalents coming soon

Scheduling

podAffinity

```
kind: Pod
...
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: tier
            operator: In
            values:
            - database
        topologyKey: failure-domain.beta.kubernetes.io/zone
```

Scheduling

podAntiAffinity

```
kind: Pod
...
spec:
  affinity:
    podAffinity:
    ...
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - my-app
          topologyKey: kubernetes.io/hostname
```

Scheduling

```
kind: Pod
...
spec:
 affinity:
  podAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
      matchExpressions:
      - key: tier
        operator: In
        values:
        - database
     topologyKey: failure-domain.beta.kubernetes.io/zone
  podAntiAffinity:
   preferredDuringSchedulingIgnoredDuringExecution:
   - weight: 100
     podAffinityTerm:
      labelSelector:
       matchExpressions:
       - key: app
         operator: In
         values:
         - my-app
       topologyKey: kubernetes.io/hostname
```

Scheduling

Pod [anti-]affinity

- The pod affinity rule says that the pod can schedule onto a node only if that node is in the same zone as at least one already-running pod that has a label with key “tier” and value “database”.
- The pod anti-affinity rule says that the pod prefers to not schedule onto a node if that node is already running a pod with label having key “app” and value “my-app”.

Scheduling

Node affinity

- Old **nodeSelector** mechanism, now deprecated for node affinity
- Set 'rules' for how to schedule your pods based on nodes
  - **requiredDuringSchedulingIgnoredDuringExecution** – 'hard'
  - **preferredDuringSchedulingIgnoredDuringExecution** – 'soft'
  - **requiredDuringExecution** equivalents coming soon

Scheduling

Node affinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/zone
            operator: In
            values:
            - availability-zone-1
            - availability-zone-2
```

Scheduling

Node affinity

```
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      ...
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: cloud.google.com/gke-nodepool
            operator: In
            values:
            - my-node-pool
```

Scheduling

Node affinity

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/zone
            operator: In
            values:
            - availability-zone-1
            - availability-zone-2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: cloud.google.com/gke-nodepool
            operator: In
            values:
            - my-node-pool
```

Scheduling

Taints and tolerations

- Kubernetes nodes can be ‘tainted’ with particular keys
- This is the opposite of affinity
- ‘Repel’ pods from a particular node
- Pods have to ‘tolerate’ these taints in order to be scheduled there

```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

Workshop

4. Try out the advanced scheduling features

Photo by Barn Images on Unsplash

Sock Shop

It's back. More socks to sell!

Sock Shop

Exposed to the world with no encryption

A blog post on reddit has exposed that we are collecting payment information without using SSL and Twitter has exploded with hate-mail.

Paying security consultants to handle things only resulted in a 896 page document but no SSL.

Meanwhile, the sales are down 85% and we need to move fast to fix it.

Ingress

Control and route traffic

Ingress

Routing traffic from the Internet to your services

- More advanced than services
- Not implemented in-tree
- Different implementations
  - e.g. NGINX, GCE
- L4 – L7
- TLS support

Ingress

Routing traffic from the Internet to your services

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: foo
          servicePort: 80
      - path: /bar
        backend:
          serviceName: bar
          servicePort: 80
```

Workshop

5. Configure nginx ingress with cert-manager for automated TLS

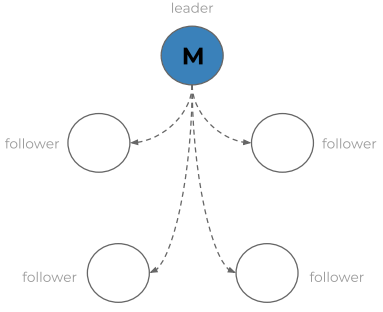
Photo by Barn Images on Unsplash

StatefulSet

What can I use for my stateful workloads?

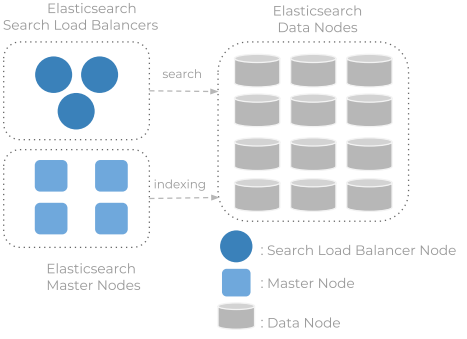
StatefulSet

All distributed systems were not created equal



Leader elected quorum

(e.g. etcd, ZK, MongoDB)

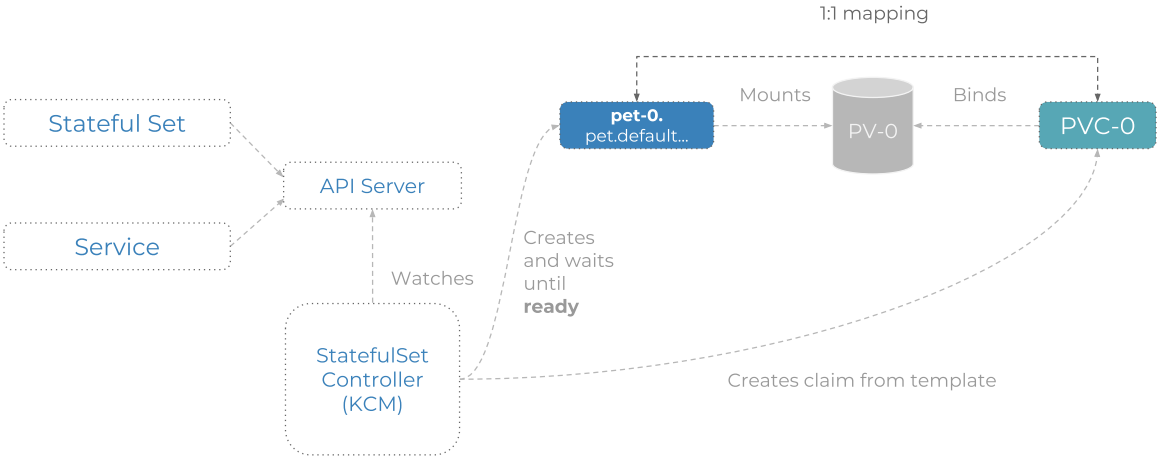


Active-active / multi-master

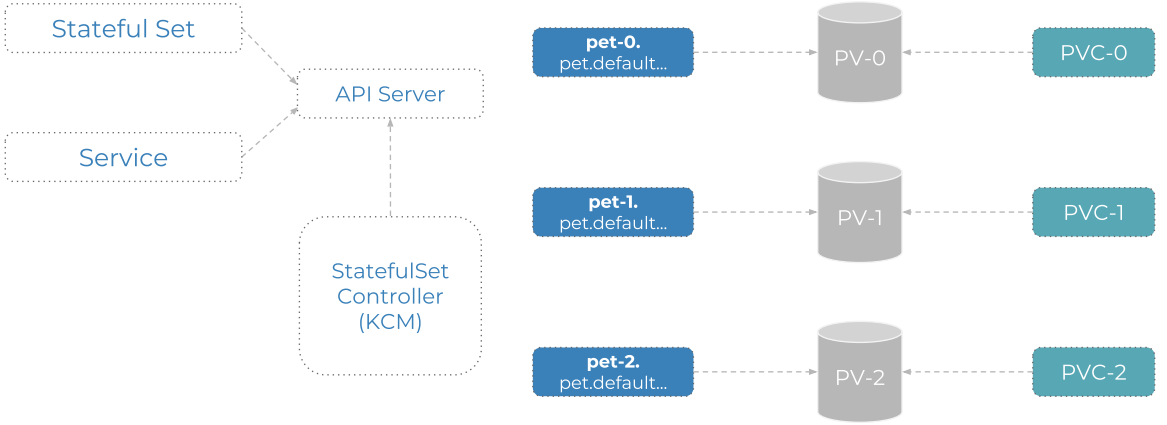
(e.g. MySQL Galera, Elasticsearch)

etc..

StatefulSet



StatefulSet



Workshop

6. Use Helm to deploy a stateful mongodb replica set

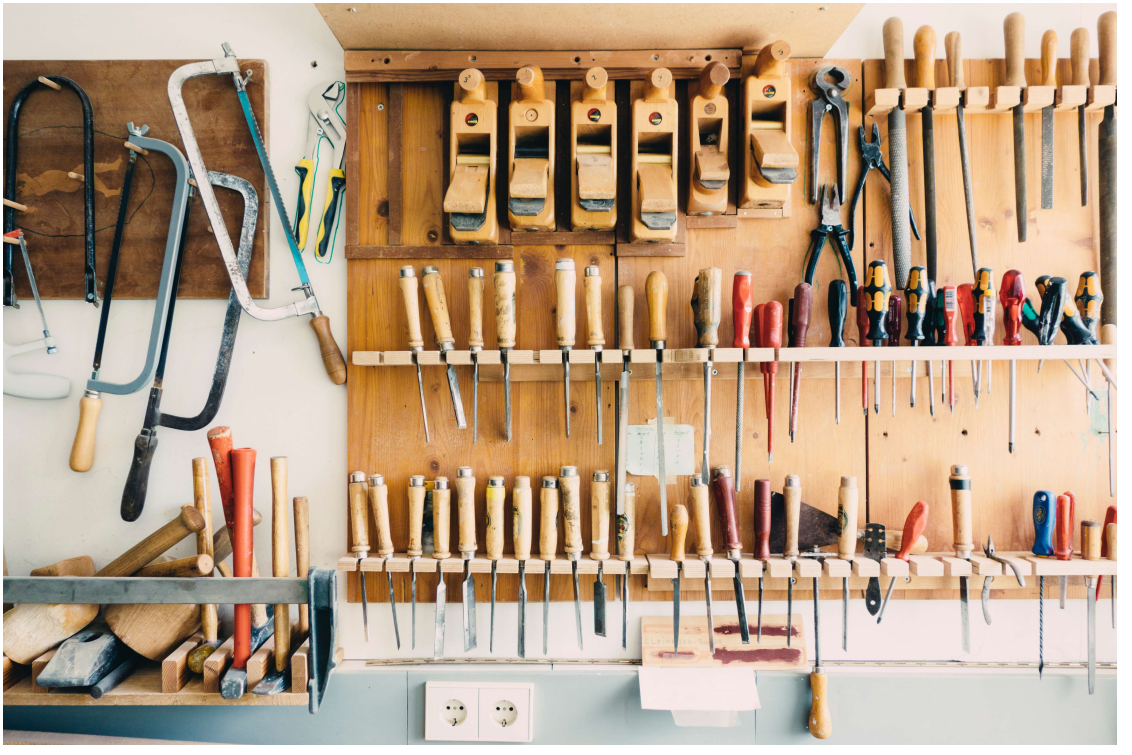


Photo by Barn Images on Unsplash

CI/CD Pipelines

Continuous Integration & Deployment in Kubernetes

CI/CD Pipelines

Integrate CI pipelines into the development workflow

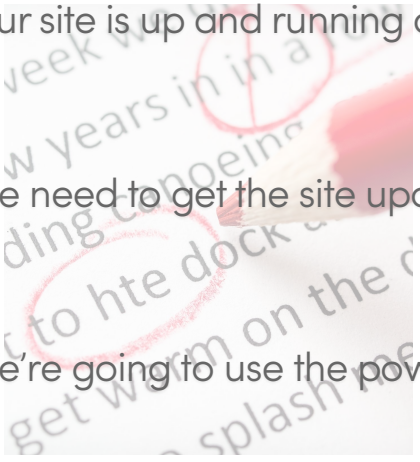
- If your project has automated tests, they can be run within your CI pipeline
- If the tests fail, the CI system won't allow you to deploy your code
- Allows you to confidently push changes regularly
- Write automated tests for anything that slips through
- But sometimes, simple tests aren't enough. A test can't tell if two colours don't 'go', or detect a spelling mistake...

CI/CD Pipelines

Discussion

- Number of different CI tools:
  - Jenkins
  - Circle CI
  - Travis
  - GitLab
  - Drone
  - Wercker
  - GoCD
- Some of these are hosted services, others can run directly on Kubernetes

Sock Shop



© Calkins/shutterstock.com

Uh oh, there's a mistake in our sock shop!

Our site is up and running and serving production traffic, but we've noticed a spelling mistake that wasn't picked up in review!

We need to get the site updated ASAP, but we also need to make sure we don't accidentally deploy anything incorrectly.

We're going to use the power of GitLab CI to push a proposed change, review that change, and finally deploy it to production. All using the practices we've learnt today.

Workshop

7. Write some new code, review, and deploy it.

Photo by Barn Images on Unsplash

Monitoring

How healthy is your cluster?

Workshop

8. Collect & display metrics with Prometheus & Grafana

Photo by Barn Images on Unsplash

Security

What underpins Kubernetes security?

Security

Ensuring security in a multi-tenanted Kubernetes cluster

- Linux security
- Namespace isolation and resource quota
- API server authentication/authorisation
- Admission control
- PodSecurityPolicy
- NetworkPolicy
- Audit (WIP as of 1.7)

Security

Many Linux security tools in the box

- SELinux/AppArmor
- Linux capabilities
- Seccomp
- Workshop – ci/cd

Authentication

Authenticating users with authentication plugins

- Users and processes (service accounts)
- Plugins
  - X509 client certificates
  - Service account tokens
  - OIDC (OpenID Connect)
  - HTTP basic auth (static token / password files)
  - Authenticating proxy
  - Webhook tokens

Authorisation

Who

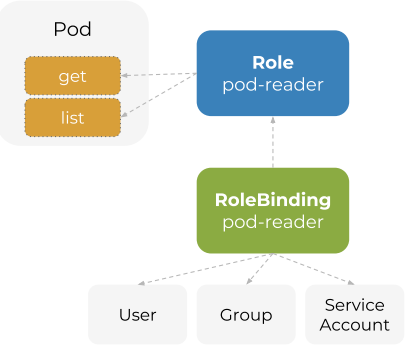
- ABAC
- RBAC
- Webhook

  Custom
-

Authorisation with RBAC

Who

- Role Based Access Control
- Beta in 1.6 and preferred over ABAC
- Cluster-wide and namespace-specific
- Role/ClusterRole
- RoleBinding/ClusterRoleBInding



Admission control

Intercepting API requests after authentication and authorisation

- Many admission controllers, but notably:
  - AlwaysPullImages
  - ImagePolicyWebhook
  - PodSecurityPolicy

PodSecurityPolicy

Protecting the cluster from pods and containers

- Cluster-level resource to control pod actions and access
- Conditions that must be met before being 'accepted' – e.g:
  - Capabilities (add/drop/allowed)
  - User ID
  - SELinux context
  - Read-only root filesystem
  - Host ports / PID / IPC
  - Volume plugins
- Enforced with PodSecurityPolicy admission controller

The End

Photo by Zoltan Tasi on Unsplash

Demo: App autoscaling

- Create a Deployment
- Create a HPA with minimum replicas

```
kubectl autoscale deployment/test --min=2 --max=5 --cpu-percent=60
```

Recap: services

- Intra-cluster
- Service VIPs

Recap: services