SUSE

Introduction to Kubernetes

Antonio Gámez Díaz, PhD (he/him/his)
Ibone González Mauraza (she/her/hers)

Software Engineers @ SUSE

SUSE Academic Training Program

November 22th, 2024

Disclaimer: *this material is **NOT** part of the official SUSE training offering, this is just part of an employee-led initiative. For official or certification-led training, please always refer to suse.com/training*

Speakers

Who is who

- **Antonio Gámez Díaz, PhD**
  - Sr. Software Engineer @ SUSE
  - antonio.gamez@suse.com

- Bringing the power of **Kubernetes** to **SAP solutions** in SUSE since 2024.
- **PhD** in **Software Engineering** by the Universidad de **Sevilla**.
- Loves **APIs** and **SLAs**.

Speakers

Who is who

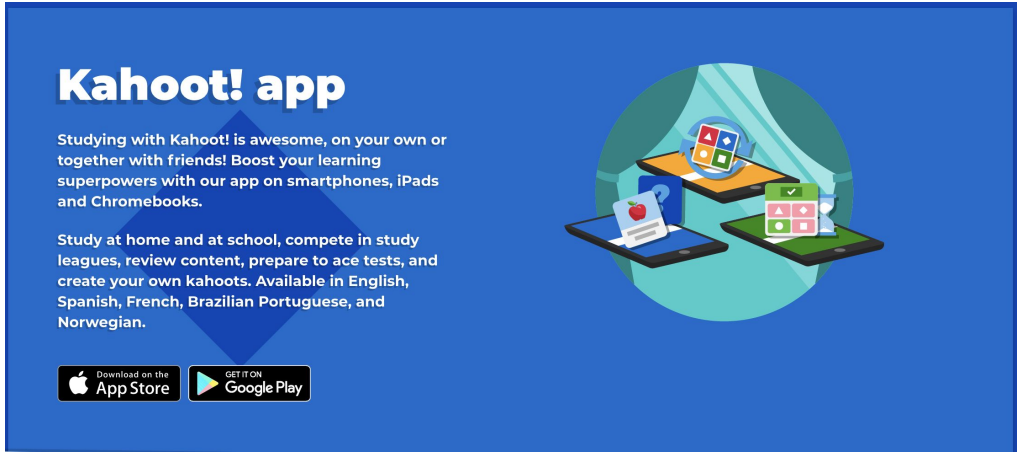
- **Ibone González Mauraza**
  - Software Engineer @ SUSE
  - ibone.gonzalez@suse.com

- Full-stack engineer at the **SUSE Customer Center** team since 2024.
- **CKAD**-certified by the CNCF.
- Passionate about **Kubernetes** and **lettering**.

Stay tuned for the Kahoot

We might have some prizes for you :)

- At the end of the session we will provide a **Kahoot PIN**
  - Join using the Kahoot app or kahoot.it



Agenda

For today's session

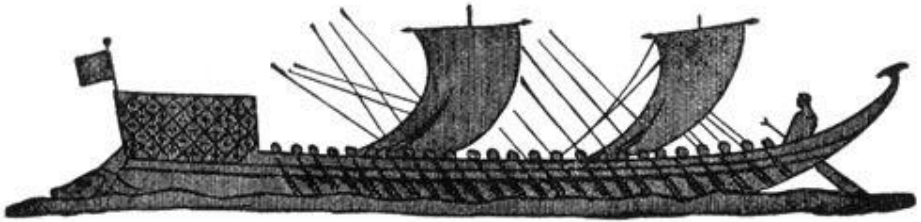
1. Introduction to Kubernetes.
2. Kubectl and the K8s API.
3. Deploying apps on K8s: Pods and Deployments.
4. Accessing to our apps: Services.

1. Introduction to Kubernetes

SUSE

κυβερνήτης

kube...what?



κυβερνήτης (kyvernítis)
m (*plural* **κυβερνήτες**)

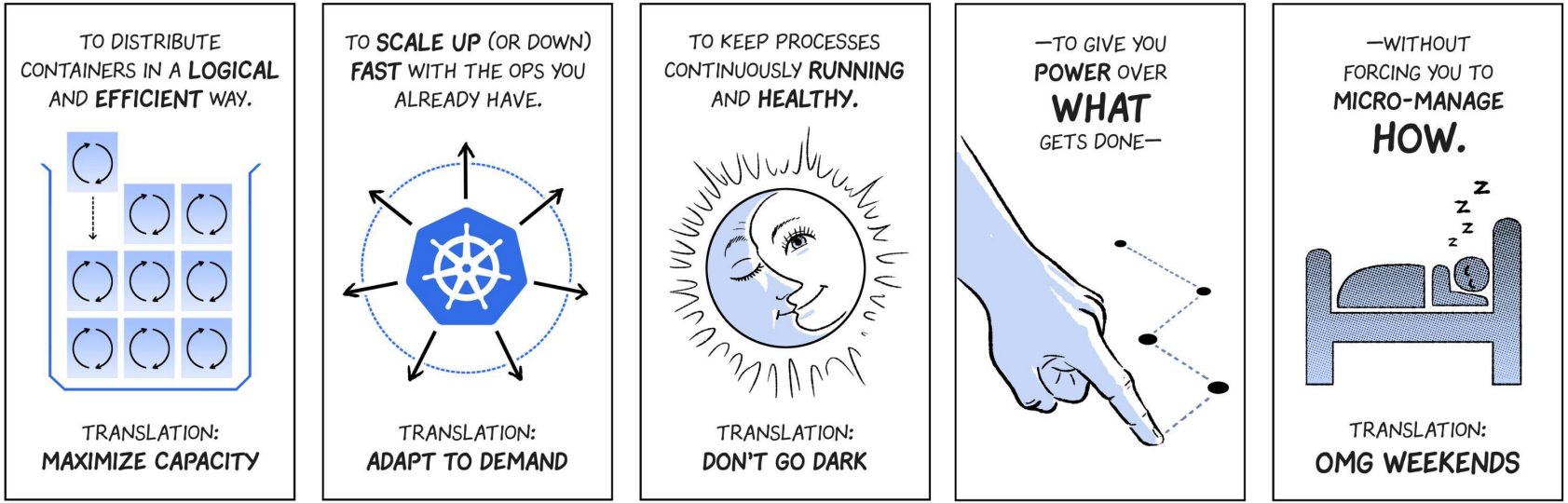
1. governor (leader of a region or state)
2. (*nautical*) captain, skipper
3. pilot (of an aircraft)

kubernetes

Picture from karenswhimsy.com/ancient-greek-ships

Introduction

Why do I need Kubernetes?



Picture from cloud.google.com/kubernetes-engine/kubernetes-comic

Introduction

What is Kubernetes

- Kubernetes is an open-source software for automating **deployment**, **scaling**, and **management** of **containerized** applications.
- Provides a powerful API to manage distributed applications.
- Built on 15 years of experience at **Google.**
- Apache Software License.
- Now governed by the CNCF (Cloud Native Computing Foundation) at the **Linux Foundation.**
    - landscape.cncf.io
- Several Special Interest Groups (SIG).
- Open to everyone.
- Weekly hangouts.

Introduction

The Kubernetes project

- Open-sourced in **June 2014 (10 years old).**
- +3.7K **contributors.**
- ~126K **commits.**
- Google and other companies are lead contributors
  - Check contributions by company.
  - **SUSE** has **+150 commits** in the project
- +203K **people** on Slack (kubernetes.slack.com).
- 1 major release every 3 months (currently 1.31).

Data extracted on Nov'24

Introduction

The Kubernetes project

1979
Unix V7 introduces chroot

1982
chroot added to BSD

2000
FreeBSD Jails released

2001
Linux VServer released

2003
Google creates Borg

2004
Solaris Containers on Zones

2006
Linux cpusets + chroot = "Linux Containers"

2007
Google launches Process Containers aka "cgroups"

2008
Linux kernel 2.6.24 adds "cgroups"

2008
LXC - Linux Containers released

Introduction

The Kubernetes project

2013
Docker open-sourced by Dotcloud

2013
Internal proposals in Google began

2014
Google open sources container-agent

2014
first commits to Kubernetes inside Google

2014
First Kubernetes open-source commit

2014
Google announces Kubernetes at DockerCon

2014
Microsoft, RedHat, IBM, Docker, CoreOS join the Kubernetes community

2014
Google's GKE becomes the first managed Kubernetes service

2015
First Kubernetes training course

2015
Kubernetes enables use of the Container Network Interface (CNI)

2016
addition of the Container Runtime Interface (CRI)

2018
Kubernetes is first project to graduate from CNCF incubation

2022
Kubernetes 8th year open-source birthday

Introduction

Used in several projects

and more...

Introduction

Should I learn Kubernetes?

SUSE Company Life At SUSE About Us Blog Saved Jobs (0)

Refine your search

Category +

Country +

State +

City +

Create Job Alert

NOTE: Use refine search filters above to get better job alerts

Email Address

Enter mail

You'll get emails Weekly

Create job alert

Showing Search results for "kubernetes"

Search from below list 25 Jobs Sort by Most relevant

Kubernetes Engineer
sala, Distrito Federal, Brazil | Engineering & IT

Technology Consultant (Kubernetes)
Professional & Consulting Services
Job available in 2 locations

Technical Consultant - Kubernetes
Professional & Consulting Services
Job available in 5 locations

Staff Software Engineer - Observability Agent
Engineering & IT
Job available in 6 locations

in Kubernetes Unión Europea

Empleos Fecha de publicación Nivel de experienc

Kubernetes en Unión Europea
33.861 resultados
Crear alerta

157 ofertas de kubernetes en España

968 ofertas de Kubernetes en España

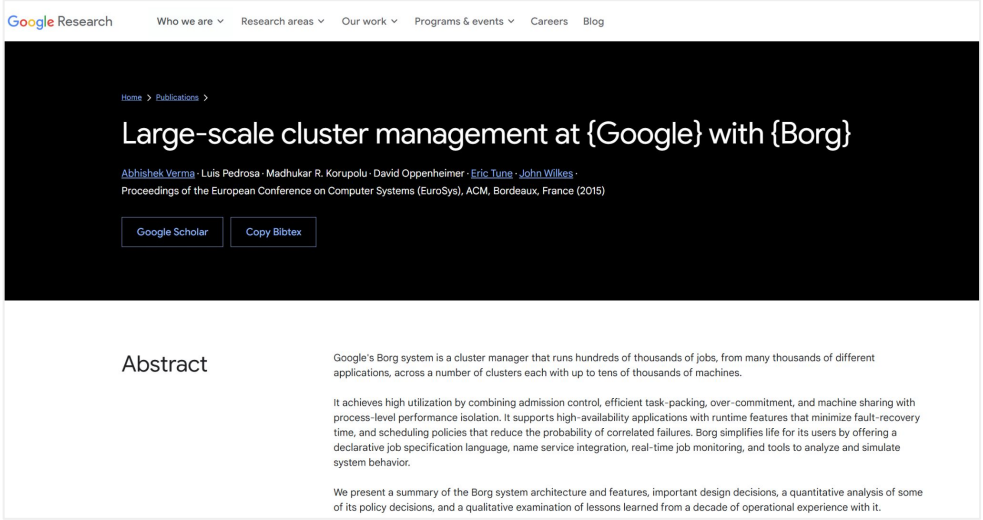
jobs.suse.com

Data extracted on Nov'24

Introduction

Origin of K8s: Borg

- **Borg** was a **Google** secret for a long time.
- Orchestration system to **manage all Google applications** at scale.
- Finally described publicly in **2015.**
- Paper explaining ideas behind Kubernetes.



Google Research Who we are Research areas Our work Programs & events Careers Blog

Home > Publications >

Large-scale cluster management at {Google} with {Borg}

Abhishek Verma · Luis Pedrosa · Madhukar R. Korupolu · David Oppenheimer · Eric Tune · John Wilkes ·
Proceedings of the European Conference on Computer Systems (EuroSys), ACM, Bordeaux, France (2015)

Google Scholar Copy Bibtex

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

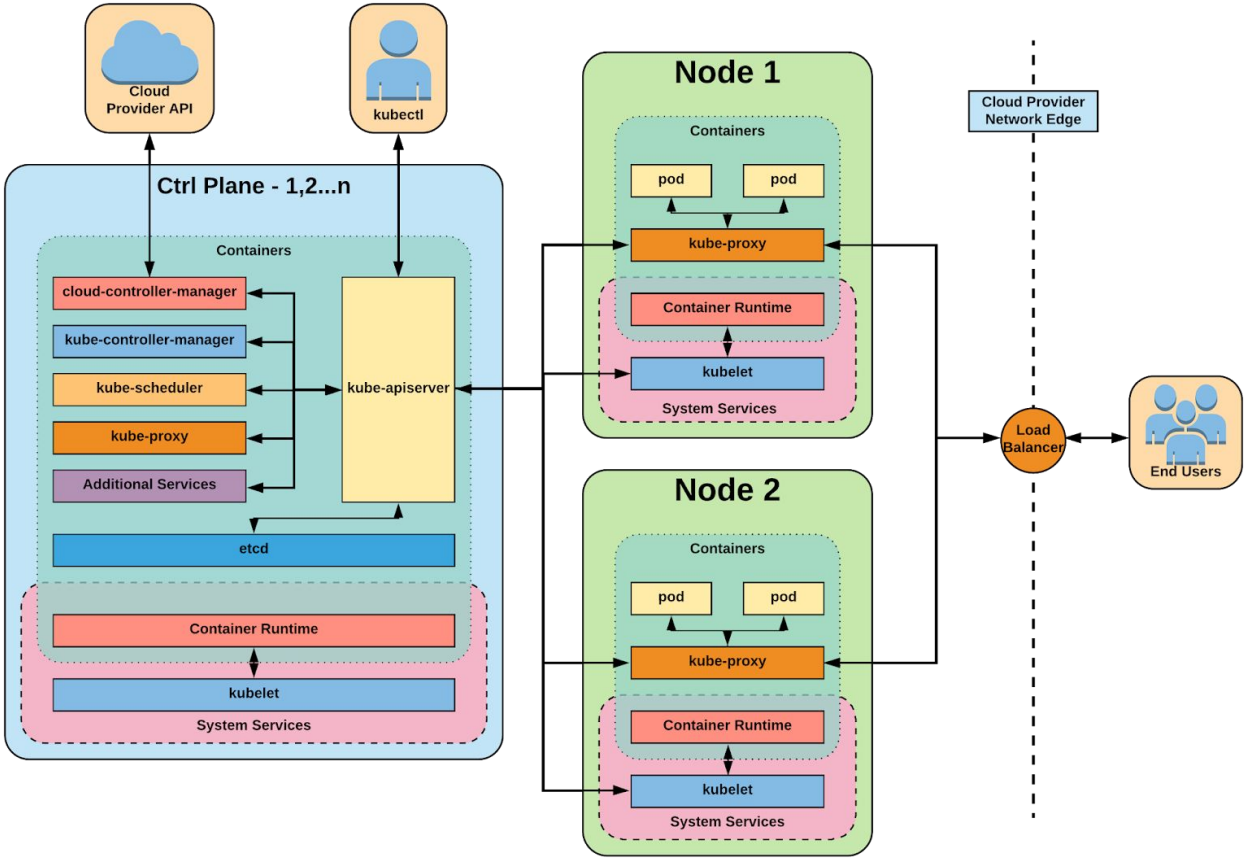
Introduction

Kubernetes lineage

landscape.cncf.io

Introduction

Architectural overview

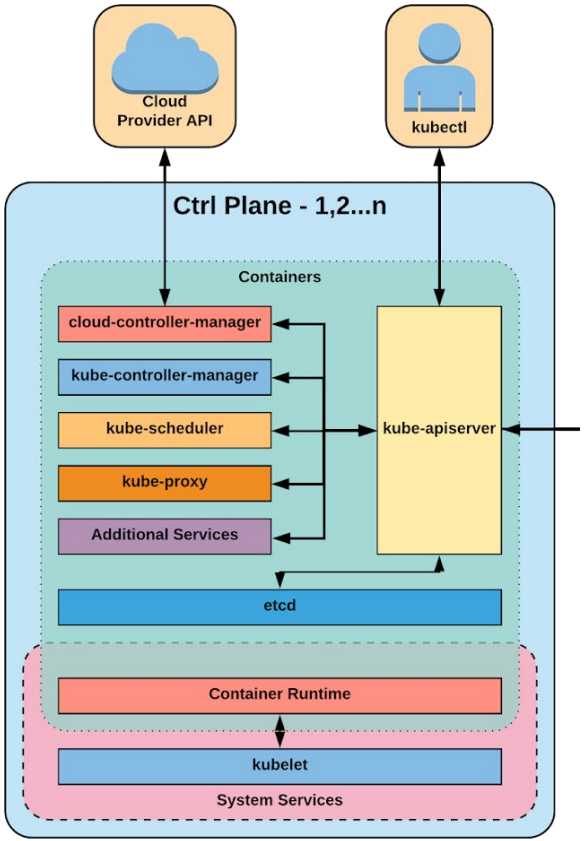


Picture from Introduction to Kubernetes

Introduction

Inside a control plane

- **kube-apiserver:**
  - It is where the cluster is **administered**, it implements a **REST API** (*kubectl* talks to this API).
- **etcd:**
  - Lightweight and distributed **key-value storage**.
- **kube-controller-manager:**
  - Monitors the **cluster state** and steers the cluster towards the **desired state**.
- **kube-scheduler:**
  - **Assigns workloads** to each node, selecting the best one.

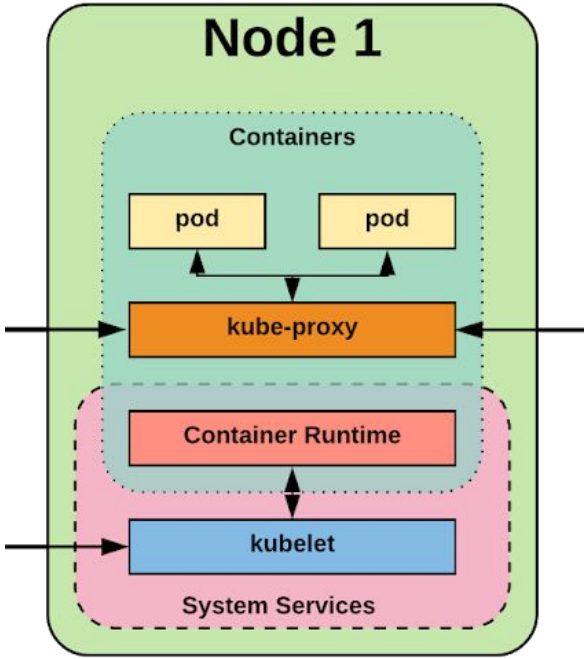


Picture from Introduction to Kubernetes

Introduction

Inside a node

- **kubelet**:
  - Interacts with the control plane and **etcd** and receives **workloads**.
- **kube-proxy**:
  - **Forward the workloads** to the container.
- **Container Runtime Engine**:
  - It is the **container runtime**, such as *Containerd* (~Docker), *Rkt*, *CRI-o*, *Kata*, *Virtlet*, etc...



Picture from Introduction to Kubernetes

Introduction

A tour of web resources

- Kubernetes Documentation.
- Cloud Native Computing Foundation.
- Kubernetes · GitHub.
- Rancher Academy.

2. Kubectl and the K8s API

SUSE

Kubectl and the K8s API

API overview: everything is an API object

```
Format:
/apis/<group>/<version>/<resource>

Examples:
/apis/apps/v1/deployments
/apis/batch/v1beta1/cronjobs
```

Everything in k8s is an API object.

```yaml
apiVersion: v1
kind: Pod

metadata
  name: pod-example
  namespace: default
  uid: ...
...
```

YAML files.

kubernetes.io/docs/reference

Kubectl and the K8s API

API overview: kubectl

- **kubectl** is the way to interact with the k8s API:

```
kubectl <command> <type> <name> <flags>>
```

  - `command` – operation to execute.
  - `type` – k8s API resource.
  - `name` – name of the resource.
  - `flags` – optional arguments.

kubernetes.io/docs/tasks/tools/#kubectl

Kubectl and the K8s API

Install kubectl

- Install the **kubectl** binary:

```
# Linux
> curl -Lo "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
> chmod +x ./kubectl
> sudo mv ./kubectl /usr/local/bin/kubectl

# MacOS
> curl -Lo "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl"
> chmod +x ./kubectl
> sudo mv ./kubectl /usr/local/bin/kubectl
```

Find detailed instructions to install it on Linux, MacOS or Windows on the Kubernetes documentation.

Kubectl and the K8s API

But… I need a k8s cluster!

- For **learning** and **developing**:
  - Killercoda
  - Kind
  - Minikube
  - Kubeadm
  - Microk8s
  - K3s
  - k3d

- **Production-grade** Kubernetes distributions:
  - On-premise k8s (~private cloud)
    - Bare-metal deployment
    - RKE2
  - Managed-clusters on public clouds
    - GKE, EKS, AKS, …
- Managing multiple Kubernetes clusters in a consolidated way:
  - Rancher

Kubectl and the K8s API

Bootstrapping a simple cluster: k3d

- k3d is a lightweight wrapper to **run a Kubernetes cluster** (k3s) in a **container**.
  - Can create single and multi node clusters.
- **Prerequisite**: install Docker.
  - Or any container management tool, like Podman (extra configuration required)

Kubectl and the K8s API

Using k3d: installing the binary and creating a cluster

- Install the **k3d binary** and create a **cluster**:

```
# Linux
> curl -s
"https://raw.githubusercontent.com/k3d-io/k3d/main/install.sh"| bash

# Create/delete a cluster
> k3d cluster create
> k3d cluster delete
```

Find detailed instructions to install it on Linux, MacOS or Windows on the k3d website.

Kubectl and the K8s API

Using k3d: installing the binary and creating a cluster (with custom config)

- If you want to use **NodePort** or **Ingress** services,
  - the k3d cluster must be created with:

```
# Exposing NodePort 30000 in the host system, port 30000
> k3d cluster create -p "30000:30000@agent:0" --agents 1

# Exposing Ingress controller in the host system, port 8080
> k3d cluster create -p "8080:80@loadbalancer" --agents 1
```

Find detailed instructions on how to expose services on the k3d documentation.

Kubectl and the K8s API

Inspect the cluster

- Check the Kubernetes cluster is **up and running**:

```
> kubectl cluster-info
Kubernetes control plane is running at https://0.0.0.0:65392
CoreDNS is running at
https://0.0.0.0:65392/api/v1/namespaces/kube-system/services/kube-d
ns:dns/proxy
Metrics-server is running at
https://0.0.0.0:65392/api/v1/namespaces/kube-system/services/https:
metrics-server:https/proxy

> kubectl get nodes
NAME                  STATUS   ROLES                  AGE   VERSION
k3d-k3s-cl-agent-0     Ready    <none>                  9h   v1.30.4
k3d-k3s-cl-server-0    Ready    control-plane,master    9h   v1.30.4
```

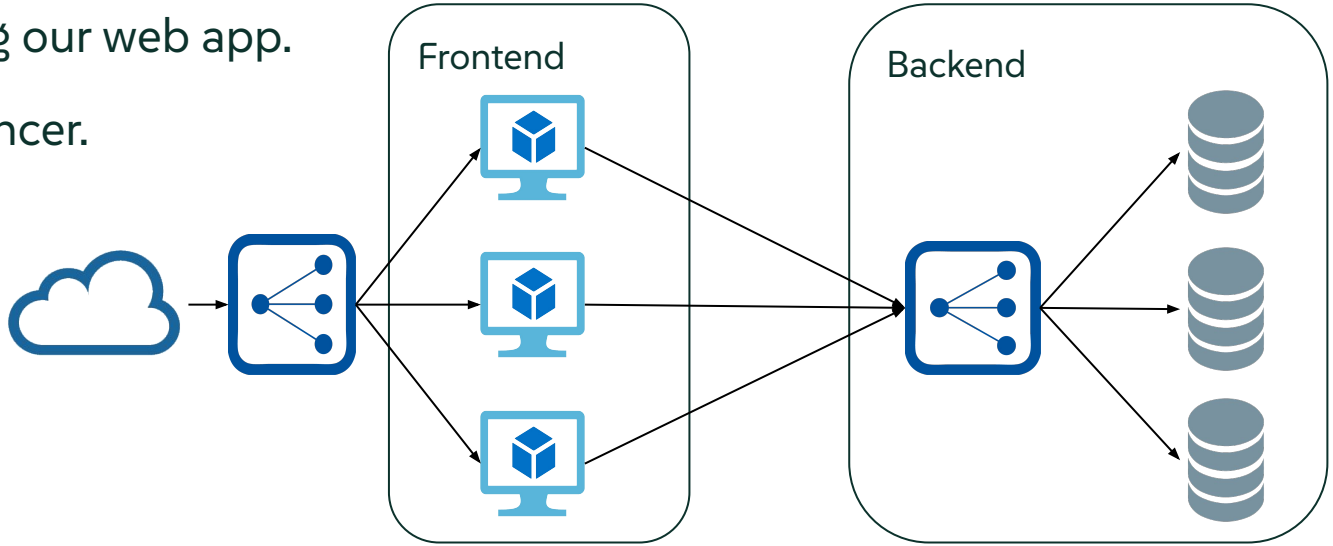
3. Deploying apps on K8s: Pods and Deployments

SUSE

Application Deployment

A common scenario: web application (frontend) using a database (backend)

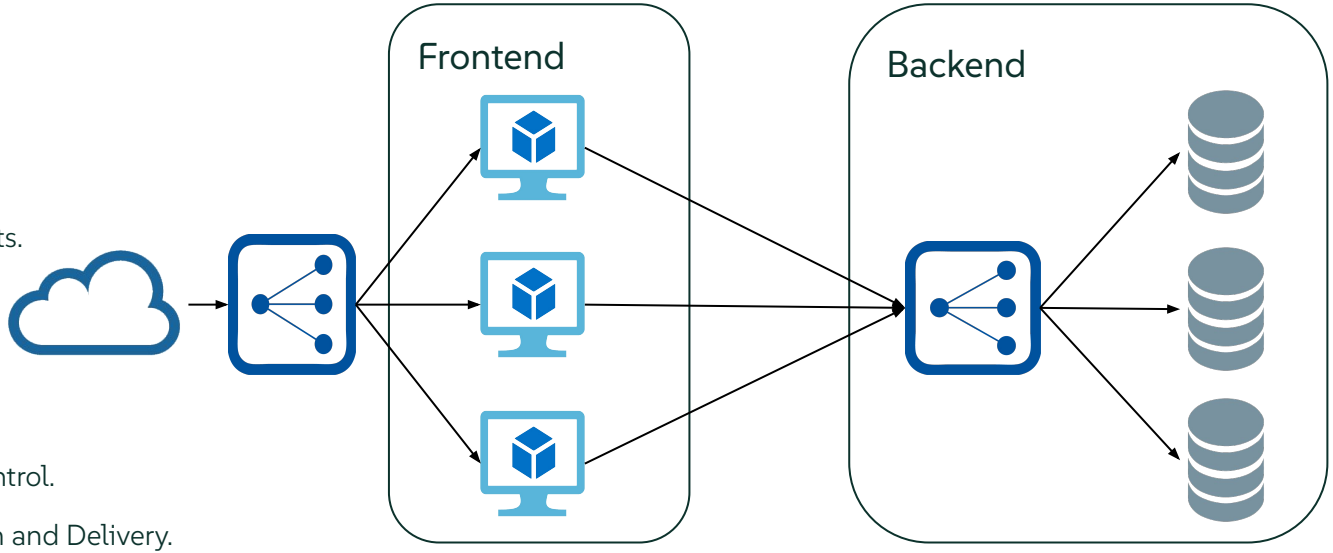
- External load balancer.
- Set of VMs running our web app.
- Internal load balancer.
- Set of databases.



Application Deployment

A common scenario: web application (frontend) using a database (backend)

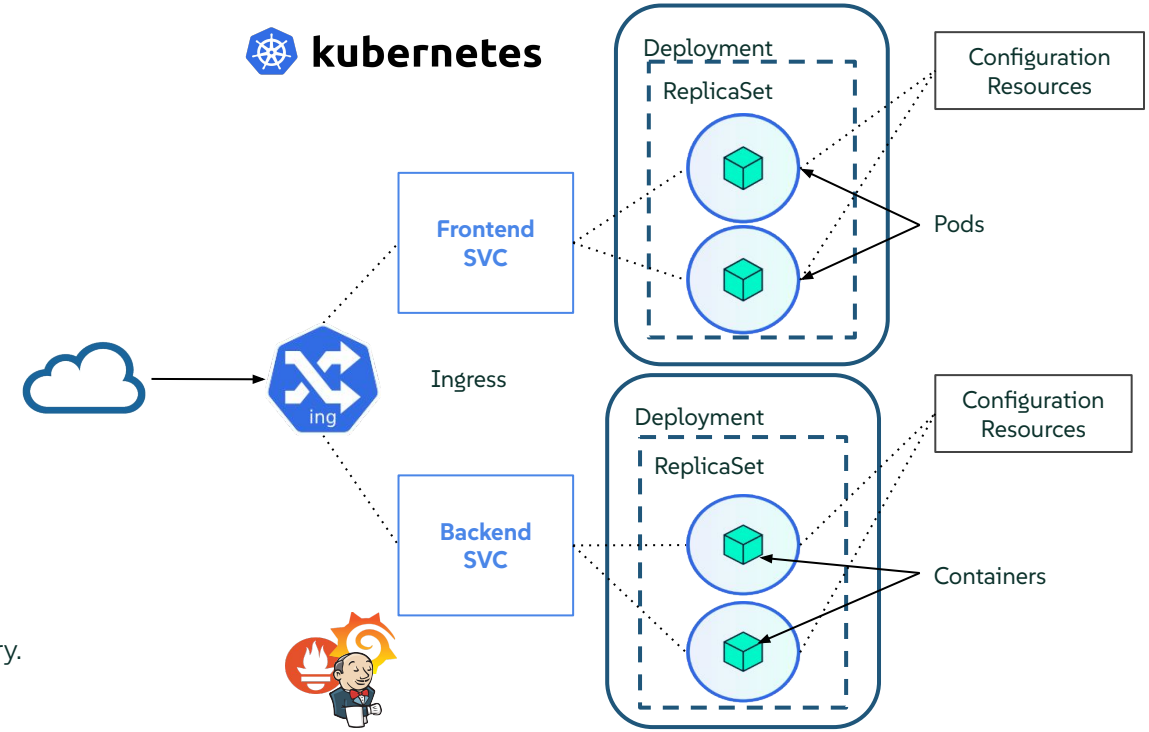
- However, this is not enough, we want:
  - Zero downtime.
  - Failover mechanisms.
  - Dynamic scalability.
  - Rolling updates.
  - Automatic deployments.
  - Load Balancing.
  - Easy migration.
  - Monitoring.
  - Role-Based Access Control.
  - Continuous Integration and Delivery.



Application Deployment

A common scenario: web application (frontend) using a database (backend)

- With Kubernetes we get:
  - Zero downtime.
  - Failover mechanisms.
  - Dynamic scalability.
  - Rolling updates.
  - Automatic deployments.
  - Load Balancing.
  - Easy migration.
  - Monitoring.
  - Role-Based Access Control.
  - Continuous Integration and Delivery.



Pets vs Cattle

A different approach for your servers

- **Pets:**
  - Treated as unique.
  - Typically, manually built managed and updated.
  - Indispensable, can't be down.

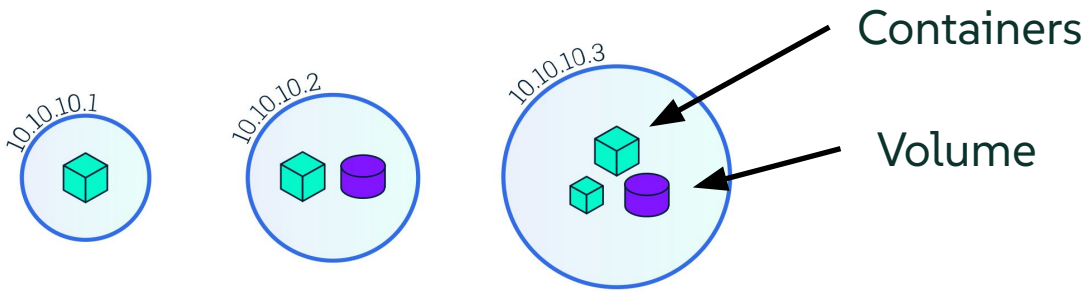
- **Cattle:**
  - Treated as "just one more".
  - Automatically built.
  - Designed for failure.

Read more at [The History of Pets vs Cattle and How to Use the Analogy Properly]

Basic Objects: Pod

What is a Pod?

- **Smallest** compute unit in Kubernetes.
  - Top level API object to run containers.
- Represents a **group** of collocated containers sharing **storage** resources and **IP**.
  - Pod's containers get restarted if they fail.
- Pods are **EPHEMERAL**.



kubernetes.io/docs/concepts/workloads/pod

Basic Objects: Pod

Why pods?

- K8s is supposed to **manage containers**, but pods are the basic building block...
  - "one process, one container" principle.
    - No more VMs with dozens of applications. Use **a container per process**.
    - But... I need more than one app/process cooperating to run my service:
      - more than one container **sharing storage** and **IP** ensuring efficient communication between them.

kubernetes.io/docs/concepts/workloads/pod

Basic Objects: Pod

Why pods?

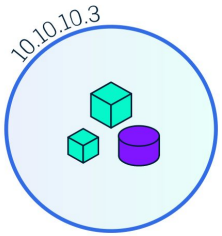
- Pods as a **new layer of abstraction**:
  - A **container** can not only be a Docker container, but also a *Rocket container* or a *VM managed by Virtlet*. Each solution has different requirements/specifications.
  - K8s needs **additional information** that a sole container doesn't have:
    - **Restart** policies.
    - Readiness/Liveness **probes**.

kubernetes.io/docs/concepts/workloads/pod

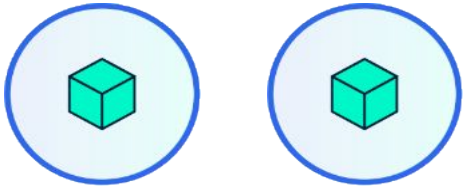
Hands-on!

Question: multi-container or multiple pods?

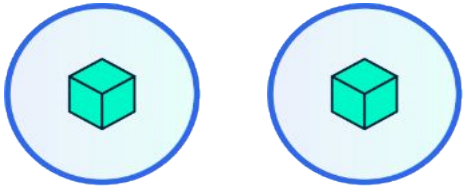
- **NGINX** and its **PHP-FPM** module.



- **Wordpress** and its **MariaDB** database.



- MongoDB **primary** and **secondaries** nodes.



kubernetes.io/docs/concepts/workloads/pod

Describing K8s objects

How does a K8s object look like? - Metadata and Spec

```
apiVersion: v1
# required field
kind: Pod
# required field
metadata:
  # required field
  name: my-pod
  # required field
  namespace: default
  labels:
    app: my-pod
  ...
spec:
  # required field
  containers:
    - image: myImage:latest
  ...
status:
  hostIP: X.X.X.X
  phase: Running
  ...
```

metadata:
Data that helps uniquely identify the K8s object.

spec:
Different on every K8s object
Describes the characteristics of the K8s object.

```
> kubectl get pod my-pod
```

kubernetes.io/docs/concepts/overview/working-with-objects

Describing K8s objects

How does a K8s object look like? - Labels

```yaml
apiVersion: v1
# required field
kind: Pod
# required field
metadata:
  # required field
  name: my-pod
  # required field
  namespace: default
  labels:
    app: my-pod
  ...
spec:
  # required field
  containers:
    - image: myImage:latest
  ...
status:
  hostIP: X.X.X.X
  phase: Running
  ...
```

labels:
You can define your labels in the object specifications.

Labels are **key/value pairs** that are attached to objects, such as pods.

```
> kubectl get pod my-pod
```

kubernetes.io/docs/concepts/overview/working-with-objects/labels

Hands-on!

Creating a pod

mongo-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mongo
spec:
  containers:
    - image: mongo
      name: mongo
```

Have a look at the Pod specification.

- Create your first Pod:

```
> kubectl create -f mongo-pod.yaml
pod/mongo created

> kubectl get pod mongo
NAME     READY   STATUS    RESTARTS   AGE
mongo    1/1     Running   0          9s
```

kubernetes.io/docs/concepts/workloads/pod

Hands-on!

Managing labels

```
# Create a new label on-the-fly
> kubectl label pods mongo my-label=my-value
pod/mongo-labels labeled

# Show labels in the output
> kubectl get pods --show-labels
NAME            READY   STATUS    RESTARTS   AGE   LABELS
mongo           1/1     Running   0          9m    my-label=my-value

# Find pods having label "my-label" equals to "my-value"
> kubectl get pods -l my-label=foo
NAME            READY   STATUS    RESTARTS   AGE
mongo           1/1     Running   0          9m

# List pods with a new column showing the label value "my-label"
> kubectl get pods -L my-label
NAME            READY   STATUS    RESTARTS   AGE   MY-LABEL
mongo           1/1     Running   0          9m    my-value
```

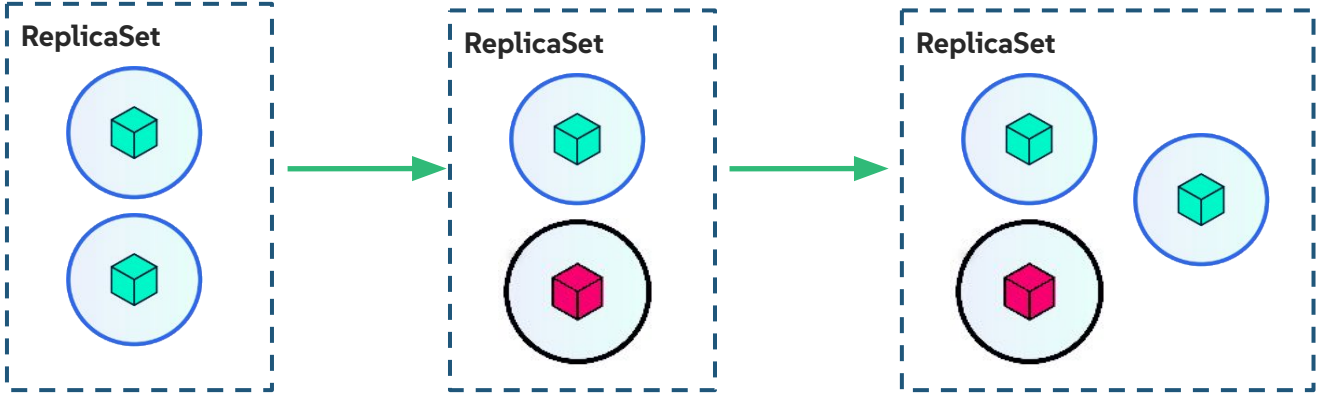
- **Why use labels?**
  - For **querying** and **selecting** resources
  - e.g., force the scheduling of a Pod on a specific Node (using nodeSelector in a Pod definition).

kubernetes.io/docs/concepts/overview/working-with-objects/labels

Basic Objects: ReplicaSet

What it is?

- A **ReplicaSet** ensures that a specified number of **pod "replicas"** are running at any one time.
  - The replication controller ensures that a pod(s) are always up and available.
- We usually don't interact with a ReplicaSet, but with a higher-level object: **Deployments**.

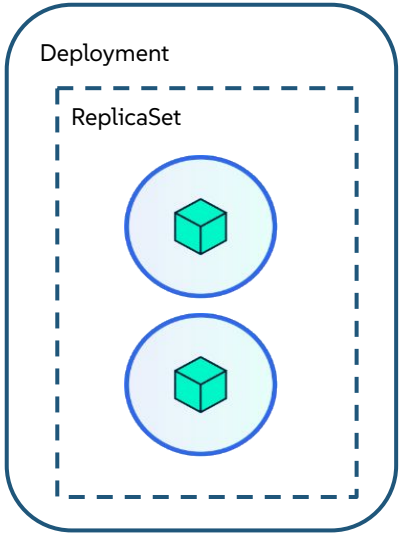


kubernetes.io/docs/concepts/workloads/controllers/replicaset

Basic Objects: Deployment

What it is?

- A **Deployment** is a higher-level concept that **manages ReplicaSets**.
- It allows several management operations like:
  - Replica management.
  - Pod scaling.
  - Rolling updates.
  - Rollback to a previous version.
  - Clean-up policies.
- Extra! - **StatefulSet**: like a Deployment, but...
  - provides guarantees about the **ordering** and **uniqueness** of the Pods.
  - offers **stable network identities** (even if the pod is rescheduled) - headless service required.
  - also offers **persistent storage** (PVC) for each Pod.



kubernetes.io/docs/concepts/workloads/controllers/deployment

Hands-on!

Creating our first deployment: a simple web server with nginx

nginx-deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginxdemos/hello
```

replicas:
number of desired instances of our apps.

matchLabels:
We are selecting pods matching "app=nginx".

label:
"app=nginx"

containers[]
array of containers that are part of our pod.

kubernetes.io/docs/concepts/workloads/controllers/deployment

Hands-on!

Creating our first deployment: a simple web server with nginx

nginx-deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginxdemos/hello
```

```
# Create a new deployment
> kubectl apply -f nginx-deploy.yaml
deployment.apps/nginx

# Show all the deployments
> kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx               1/1     1            1           30s
```

What if... we want more replicas?

kubernetes.io/docs/concepts/workloads/controllers/deployment

Hands-on!

Modifying the deployment replicas

nginx-deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginxdemos/hello
```

```
# Scale up our deployment
> kubectl scale deployment nginx --replicas=3
deployment.apps/nginx scaled

# Show all the deployments again
> kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx               3/3     3            3           30s
```

Now we have 3 replicas

What if... we want to change the image in all the replicas?

kubernetes.io/docs/concepts/workloads/controllers/deployment

Hands-on!

Changing the image used in our deployment

nginx-deploy.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.26-bookworm
```

```
# Replace the image used in the container "nginx"
> kubectl set image deployment nginx nginx=nginx:1.26-bookworm --all
deployment.apps/nginx image updated

# Describe the deployment
> kubectl describe deployment nginx
...
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:       nginx:1.26-bookworm

# Get all replicasets
> kubectl get replicasets
NAME                 DESIRED   CURRENT   READY   AGE
nginx-67c9d5bc66     3         3         3       30s
nginx-6c46465cc6     0         0         0       9m
```

The image has been changed

A new ReplicaSet is created

kubernetes.io/docs/concepts/workloads/controllers/deployment

Hands-on!

Rolling back a deployment

```
# Create a deployment without writing any YAML file :)
> kubectl create deployment bad-nginx --image=nginx
deployment.apps/bad-nginx created

# But.. everything is a YAML
> kubectl get deployment/bad-nginx -o yaml
...

# Replace the image with a non-existent one and record the changes in log
> kubectl set image deployment bad-nginx nginx=nginx:bad --all --record
deployment.apps/nginx image updated

# The pod will be in "ErrImagePull" since the image does not exist
> kubectl get pods -l app=bad-nginx
NAME                          READY   STATUS         RESTARTS   AGE
bad-nginx-69cbfbf986-4754v    0/1     ErrImagePull   0          9s
bad-nginx-8ff678449-vsd4l     1/1     Running        0          49s
```

kubernetes.io/docs/concepts/workloads/controllers/deployment

Hands-on!

Rolling back a deployment

```
# Get the deployment rollout history
# In revision 1 we created the deployment
> kubectl rollout history deployment/bad-nginx
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deployment bad-nginx nginx=nginx:bad --all=true --record=true

# Let's undo and come back to the previous revision
> kubectl rollout undo deployment/bad-nginx
deployment.apps/bad-nginx rolled back

# Get the pods again, now it is working again
> kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
bad-nginx-8ff678449-vsd4l   1/1     Running   0          6m22s
```

kubernetes.io/docs/concepts/workloads/controllers/deployment

Kubectl Tips and Tricks

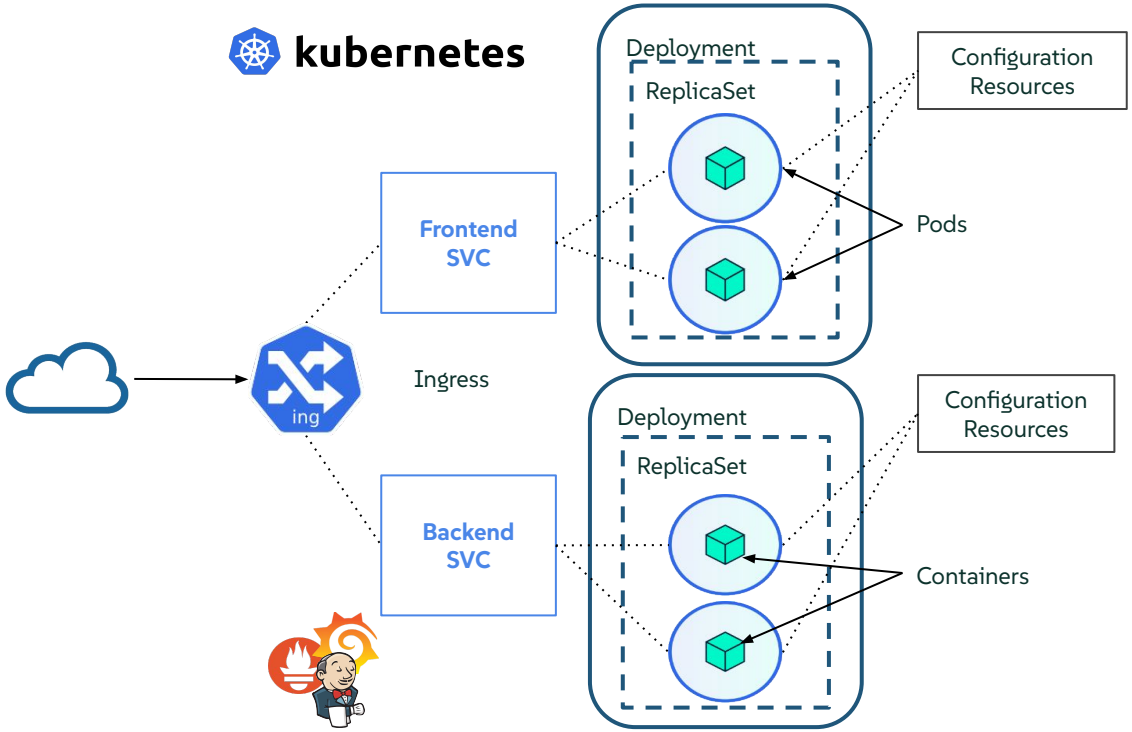
Mastering kubectl

- A few things to remember about **kubectl**.
  - And if you don't, check the cheat sheet.

```
> kubectl config view
> kubectl config use-context
> kubectl annotate
> kubectl label
> kubectl create -f ./<DIR>
> kubectl create -f <URL>
> kubectl edit ...
> kubectl proxy ...
> kubectl exec ...
> kubectl logs ...
> kubectl get pods, deployments, services
> kubectl --v=99 ...
> kubectl describe ...
```

Quick recap

Kubernetes – Kubectl – Pods – Deployments



Hands on!

Guestbook – Part I

1. Create the **frontend** deployment.
2. **Access** the frontend using "`kubectl port forward`". What do you see?
3. Create the **backend** deployment. What now?

See kubernetes.io/docs/tutorials/stateless-application/guestbook

4. Accessing to our apps: Services

SUSE

Basic Objects: Services

What it is?

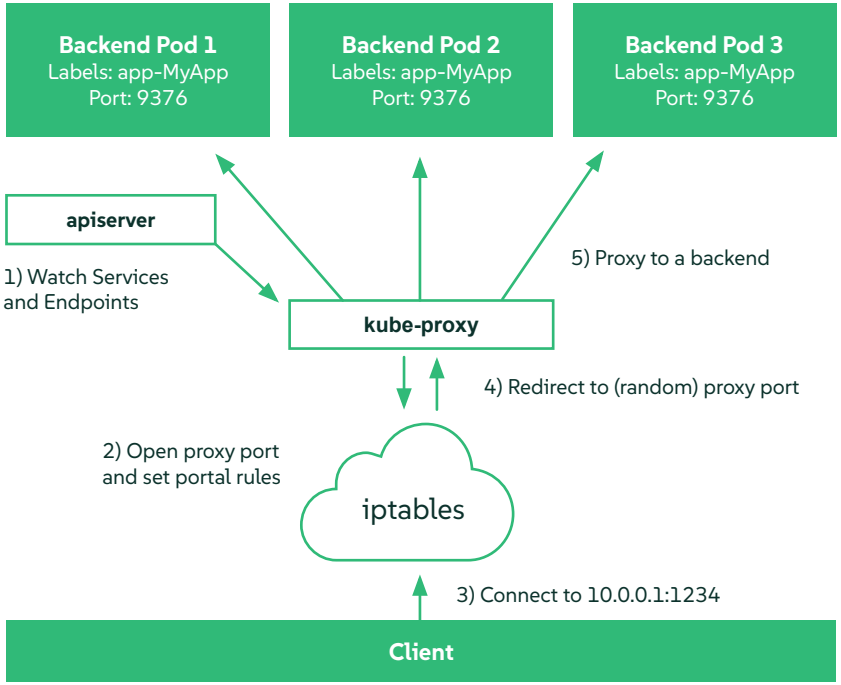
- The key question is: how do you **access your applications**?
- The answer is **Services**, yet another Kubernetes object.
  - An abstract way to **expose an application** running on a set of Pods as a network service.
  - They provide a **stable virtual endpoint** for **ephemeral Pods** in the cluster.
    - This way, other Services can target them and will be redirected to the endpoints matching the service Pod selection.

kubernetes.io/docs/concepts/services-networking/service

Basic Objects: Services

What it is?

- Implemented via **iptables**.
- **kube-proxy** watches K8s API for new Services and Endpoints being created.
- It opens **random ports** on Nodes listening on ClusterIP:Port.
  - Then forwards to a random* service endpoints.
    - * defaults to round-robin in userspace.



kubernetes.io/docs/concepts/services-networking/service

Basic Objects: Services

Different types of Services

ClusterIP

- Exposes the Service on a **cluster-internal IP**.
- It is the **default** type.
- Only provides access **internally**.
  - except if manually creating an external endpoint.
  - To access, run "kubectl proxy".
- Great for development.

NodePort

- Exposes the Service on each **Node's IP** at a **static port**.
  - Defaults ports: 30000-32767.
  - The port may have to be open in the firewall.
- Great for debugging.
- Used for manually creating load balancers.

LoadBalancer

- Exposes the Service externally using a **cloud provider's load balancer** (like GKE, AKS, AWS, ...).
  - Usually add extra charges for its usage.
- Private clouds may also implement it with a **Cloud Provider Plugin**.
  - e.g., Kind + Metalib or k3s's klipper-lb.

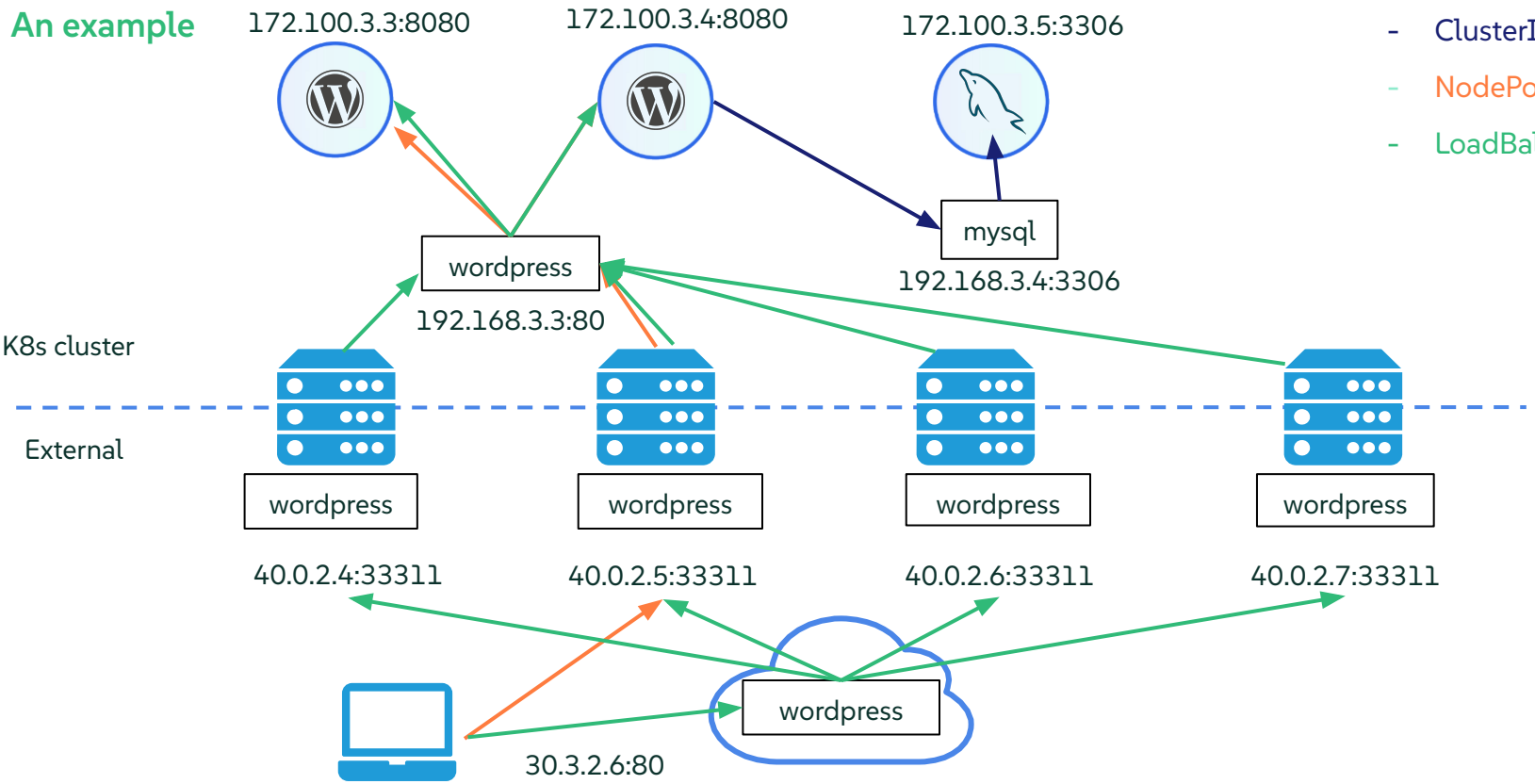
kubernetes.io/docs/concepts/services-networking/service

Basic Objects: Services

Types:

- ClusterIP
- NodePort
- LoadBalancer

An example



Hands-on!

Creating a service: ClusterIP + port-forwarding

```
# Create a new deployment (or use an existent one)
> kubectl create deployment nginx-exposed --image=nginxdemos/hello
deployment.apps/nginx-exposed created

# Create a service with a command
# The service listens on :8080, but the container (our app) does on :80
> kubectl expose deployment/nginx-exposed --name nginx-clusterip --port=8080 --target-port=80
--type=ClusterIP
service/nginx-clusterip exposed

# Local port forwarding (the service is still internal, though)
# The service listened on :8080, but we will port-forward it through the :7777
> kubectl port-forward service/nginx-clusterip 7777:8080
```

http://localhost:7777

kubernetes.io/docs/concepts/services-networking/service

Hands-on!

Creating a service

nginx-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
spec:
  selector:
      app: nginx-exposed
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      nodePort: 30000
```

port:
the port that the container is listening to.

nodePort:
if none, it will be auto-generated.

```
# Create a service
# The service listens on :30000, but the container does on :80
> kubectl apply -f nginx-svc.yaml
service/nginx-nodeport created

# Get all the services
> kubectl get services
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
nginx-clusterip   ClusterIP   10.96.45.45     <none>        8080/TCP
nginx-nodeport    NodePort    10.96.161.114   <none>    80:30000/TCP

# Find out which is the IP of our cluster
> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:57589
```

http://127.0.0.1:30000

kubernetes.io/docs/concepts/services-networking/service

Basic Objects: Services

Extra: DNS

- A **DNS service** is provided as a Kubernetes add-on in clusters.
  - On many distributions, this DNS service is provided **by default**.
- When a **Service** is created it **gets registered** in the DNS.
  - The DNS lookup will direct traffic to **one of the matching Pods** via the ClusterIP of the Service.
- Interesting read about **Headless** services.
  - Services **without a Cluster IP** will resolve to a **set of IPs** (round-robin).

kubernetes.io/docs/concepts/services-networking/service/#dns

Hands-on!

Accessing a ClusterIP service from inside!

```
# Get the service "nginx-clusterip", note that it listens on :8080
> kubectl get service nginx-clusterip
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
nginx-clusterip  ClusterIP   10.96.45.45     <none>        8080/TCP
...

# Run curl inside a Pod that will get deleted after running the command
> kubectl run -it --rm --restart=Never busybox --image=busybox wget http://nginx-clusterip:8080
Connecting to nginx-clusterip:8080 (10.96.45.45:8080)
saving to 'index.html'
index.html           100% |*******************************|  7237  0:00:00 ETA
'index.html' saved
pod "busybox" deleted
```

kubernetes.io/docs/concepts/services-networking/service/#dns

That is just the beginning...

Welcome to the cloud!

Much more to know about...

- Ingress.
- Persistence.
- Jobs, CronJobs and initContainers.
- Configuring your applications.
- Pod patterns.
- Packaging applications: Helm.
- Extra: multi-cluster management with Rancher.

Certifications:

- CKA, CKAD, CKS, KCNA, KCSA.

CKAD Curriculum

20% - Application Design and Build

- Define, build and modify container images
- Understand Jobs and CronJobs
- Understand multi-container Pod design patterns (e.g. sidecar, init and others)
- Utilize persistent and ephemeral volumes

20% - Application Deployment

- Use Kubernetes primitives to implement common deployment strategies (e.g. blue/green or canary)
- Understand Deployments and how to perform rolling updates
- Use the Helm package manager to deploy existing packages

15% - Application observability and maintenance

- Understand API deprecations
- Implement probes and health checks
- Use provided tools to monitor Kubernetes applications
- Utilize container logs
- Debugging in Kubernetes

25% - Application Environment, Configuration and Security

- Discover and use resources that extend Kubernetes (CRD)
- Understand authentication, authorization and admission control
- Understanding and defining resource requirements, limits and quotas
- Understand ConfigMaps
- Create & consume Secrets
- Understand ServiceAccounts
- Understand SecurityContexts

20% - Services & Networking

- Demonstrate basic understanding of NetworkPolicies
- Provide and troubleshoot access to applications via services
- Use Ingress rules to expose applications

Hands on!

Guestbook – Part II

1. Add a **Service** to the backend. What changed?
2. Add a **Service** for the frontend.

See kubernetes.io/docs/tutorials/stateless-application/guestbook

SUSE Academic Training Program

What it is?

SUSE Academic Training Program

- **Free Enterprise Open Source Training:**

Participating universities have the opportunity to incorporate our Enterprise Open Source Product Training into their curriculum. Universities can integrate these courses, including the technical labs, into their IT courses over the desired duration.

- SUSE Linux Enterprise Server 15 Administration.
- Kubernetes Administration.
- Rancher Manager.

- **Discounted Certification Exams and eLearning Subscriptions:**

In addition to free training, we're pleased to offer students a discount on:

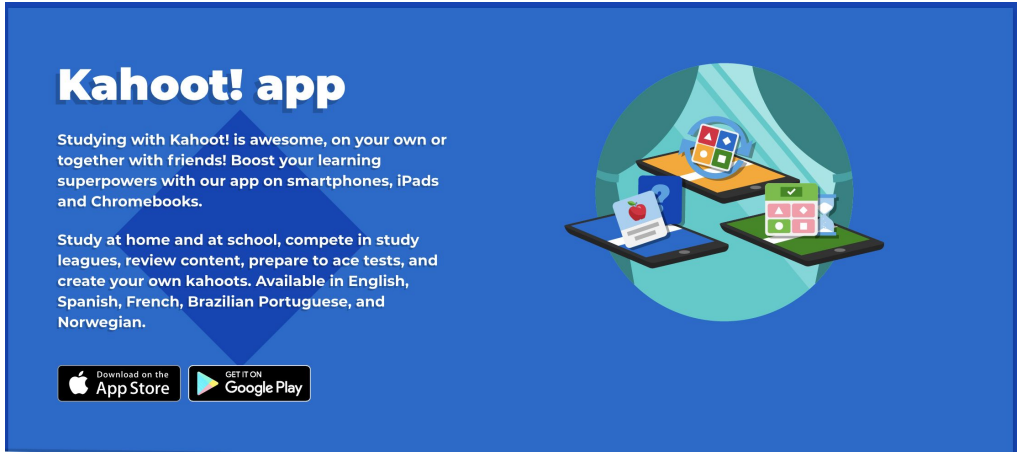
- Certification Exams.
- eLearning Subscriptions (Silver and Gold).

More information at suse.com/c/125963

Quiz time!

We might have some prizes for you :)

- Join using the Kahoot app or kahoot.it



SUSE

Thanks!

Contact us anytime:

- antonio.gamez@suse.com
- ibone.gonzalez@suse.com

Follow us on social media

- x.com/SUSE
- mastodon.social/@suseuniversities
- linkedin.com/company/suse

SUSE Academic
Training Program