



Introduction to Kubernetes

Antonio G3mez D3az, PhD (he/him/his)
Ibone Gonz3lez Mauraza (she/her/hers)

Software Engineers @ SUSE

November 22th, 2024

 **SUSE Academic**
Training Program

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.

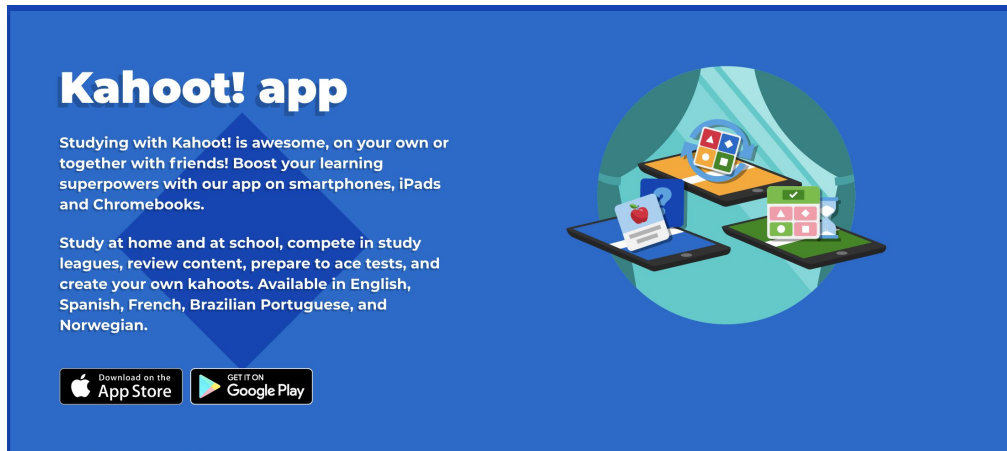
→ **CKAD**-certified by the CNCF.

→ 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



Kahoot! app

Studying with Kahoot! is awesome, on your own or together with friends! Boost your learning superpowers with our app on smartphones, iPads and Chromebooks.

Study at home and at school, compete in study leagues, review content, prepare to ace tests, and create your own kahoots. Available in English, Spanish, French, Brazilian Portuguese, and Norwegian.

Download on the App Store | GET IT ON Google Play

The banner features a blue background with a central illustration of three mobile devices (a tablet and two smartphones) displaying the Kahoot! app interface. The devices are arranged in a cluster, with the tablet at the top and the smartphones below it. The background of the banner has a subtle pattern of colorful shapes.



Agenda

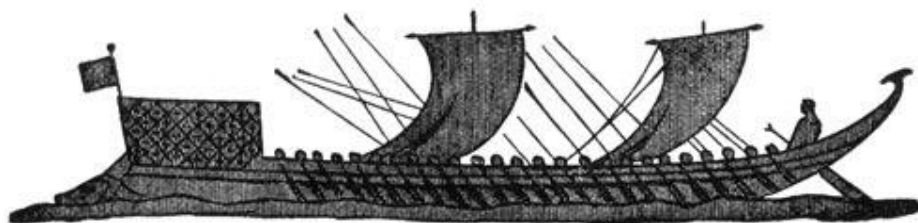
For today's session

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

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)

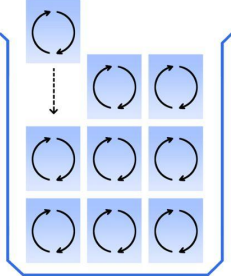


kubernetes

Introduction

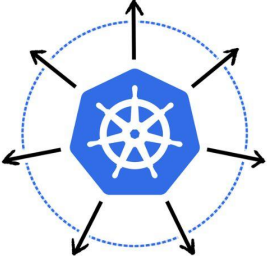
Why do I need Kubernetes?

TO DISTRIBUTE
CONTAINERS IN A **LOGICAL**
AND **EFFICIENT** WAY.



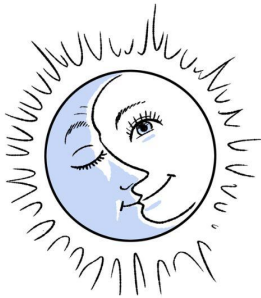
TRANSLATION:
MAXIMIZE CAPACITY

TO **SCALE UP** (OR DOWN)
FAST WITH THE OPS YOU
ALREADY HAVE.



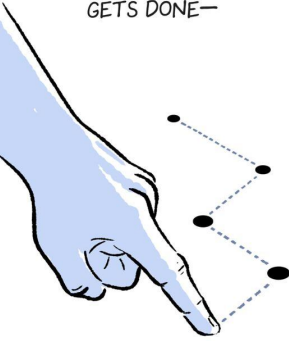
TRANSLATION:
ADAPT TO DEMAND

TO KEEP PROCESSES
CONTINUOUSLY **RUNNING**
AND **HEALTHY**.

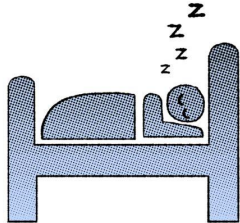


TRANSLATION:
DON'T GO DARK

—TO GIVE YOU
POWER OVER
WHAT
GETS DONE—



—WITHOUT
FORCING YOU TO
MICRO-MANAGE
HOW.



TRANSLATION:
OMG WEEKENDS

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.

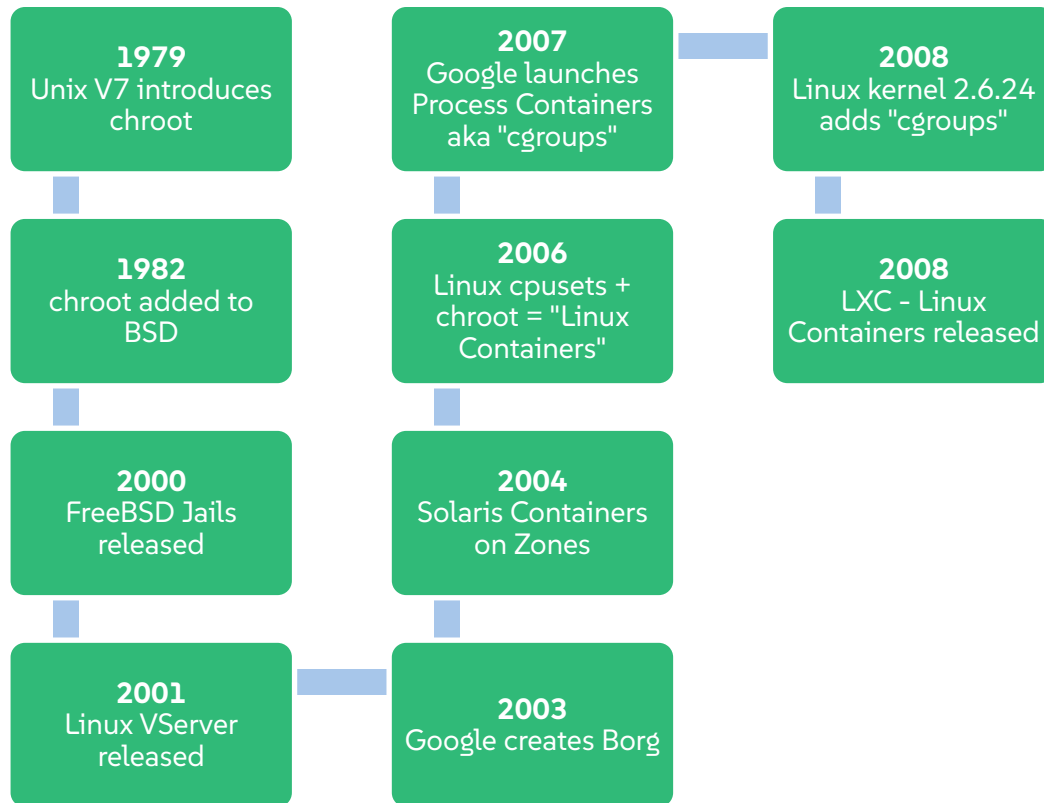
Introduction

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](#)).

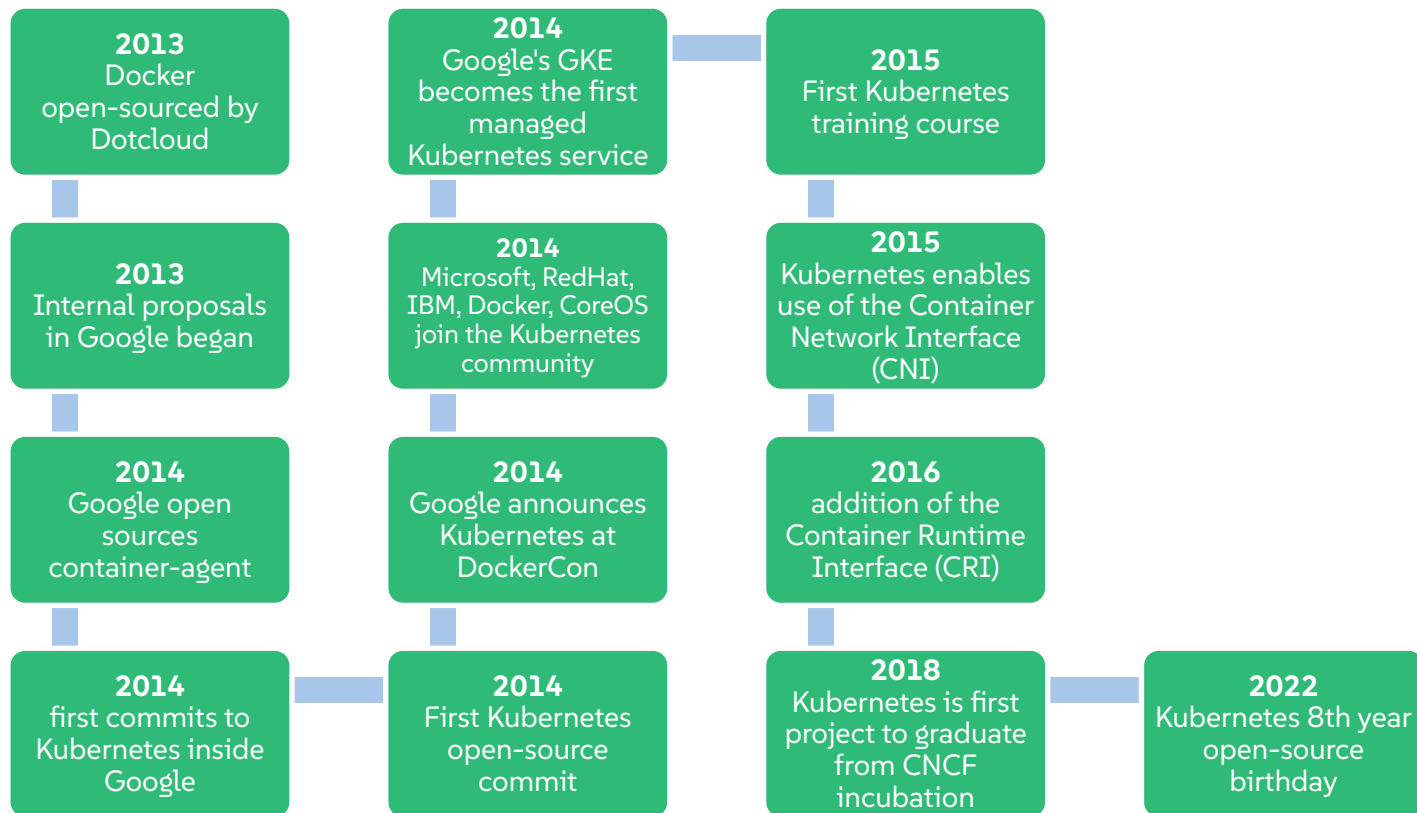
Introduction

The Kubernetes project



Introduction

The Kubernetes project



Introduction

Used in several projects



HUAWEI



PHILIPS

The
New York
Times



YAHOO!
JAPAN

Bla Bla Car

and more...



All product names, logos, and brands are property of their respective owners.

Introduction

Should I learn Kubernetes?

The screenshot shows the SUSE Jobs website interface. At the top, there is a navigation bar with the SUSE logo and links for Company, Life At SUSE, About Us, and Blog. A 'Saved Jobs (0)' indicator is also present. The main content area is titled 'Showing Search results for "kubernetes"'. On the left, there is a 'Refine your search' sidebar with filters for Category, Country, State, and City. Below this is a 'Create Job Alert' section with a form for an email address and a frequency selector set to 'Weekly'. The search results list four job titles: 'Kubernetes Engineer' (25 Jobs), 'Technology Consultant (Kubernetes)' (25 Jobs), 'Technical Consultant - Kubernetes' (5 Jobs), and 'Staff Software Engineer - Observability Agent' (6 Jobs). Each job listing includes the company name and location. A green box highlights the '25 Jobs' count for the first two listings.

The screenshot shows the LinkedIn search interface. The search bar contains 'Kubernetes' and the location is set to 'Unión Europea'. Below the search bar, there are filters for 'Empleos', 'Fecha de publicación', and 'Nivel de experiencia'. A blue banner displays 'Kubernetes en Unión Europea' with '33.861 resultados' and a 'Crear alerta' button. A green box highlights the number '33.861'.

157 ofertas de kubernetes en España

968 ofertas de Kubernetes en España

jobs.suse.com



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.

The screenshot shows the Google Research website interface. At the top, there is a navigation bar with links for 'Who we are', 'Research areas', 'Our work', 'Programs & events', 'Careers', and 'Blog'. Below the navigation bar, the page title is 'Large-scale cluster management at {Google} with {Borg}'. The authors listed are 'Abhishek Verma · Luis Pedrosa · Madhukar R. Korupolu · David Oppenheimer · Eric Tune · John Wilkes'. The publication venue is 'Proceedings of the European Conference on Computer Systems (EuroSys), ACM, Bordeaux, France (2015)'. There are two buttons: 'Google Scholar' and 'Copy Bibtext'. Below the title and authors, the word 'Abstract' is displayed. The abstract text describes the Borg system as a cluster manager that runs hundreds of thousands of jobs across many clusters, each with up to tens of thousands of machines. It highlights features like admission control, task-packing, over-commitment, machine sharing, process-level performance isolation, high-availability applications, and declarative job specification language. The abstract concludes by mentioning 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



CoreDNS



Prometheus



envoy



fluentd



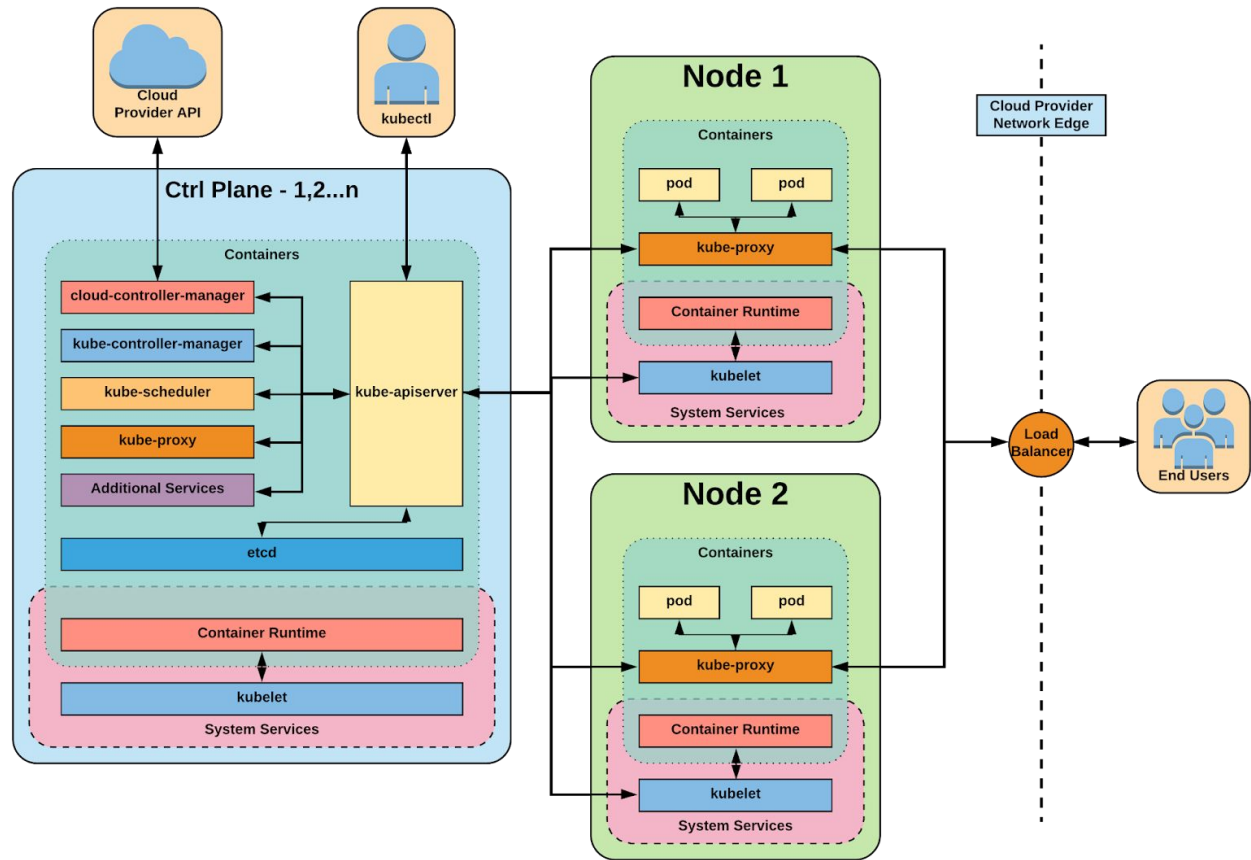
OPENTRACING

landscape.cncf.io



Introduction

Architectural overview

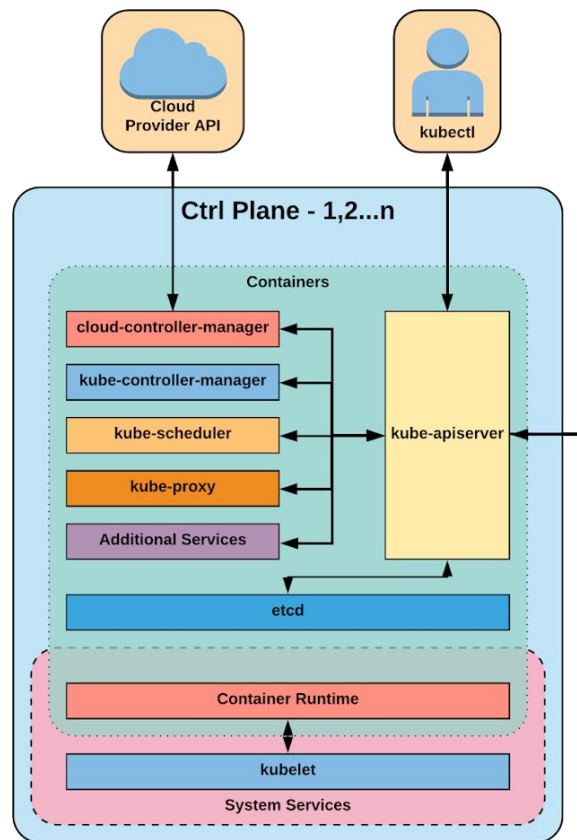


Picture from [Introduction to Kubernetes](#)

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.

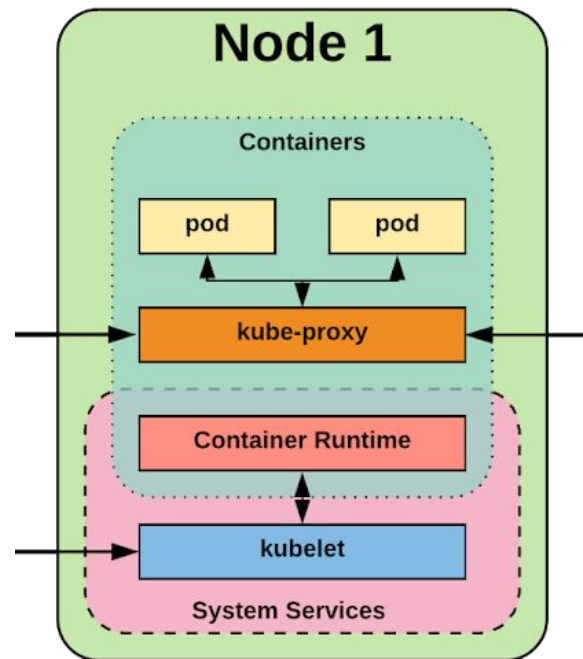


Picture from [Introduction to Kubernetes](#)

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...



Introduction

A tour of web resources

- [Kubernetes Documentation.](#)
- [Cloud Native Computing Foundation.](#)
- [Kubernetes · GitHub.](#)
- [Rancher Academy.](#)



2. Kubectl and the K8s API

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.

`apiVersion: v1`

`kind: Pod`

`metadata`

`name: pod-example`

`namespace: default`

`uid: ...`

`...`

YAML files.

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.

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**:

- [Killercodea](#)
- [Kind](#)
- [Minikube](#) genial para intros y necesidades flexibles
- [Kubeadm](#)
- [Microk8s](#)
- [K3s](#) va genial para testing
- [k3d](#) monta kubernetes en un contenedor docker

- **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](#)  **RANCHER**
BY SUSE

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
```

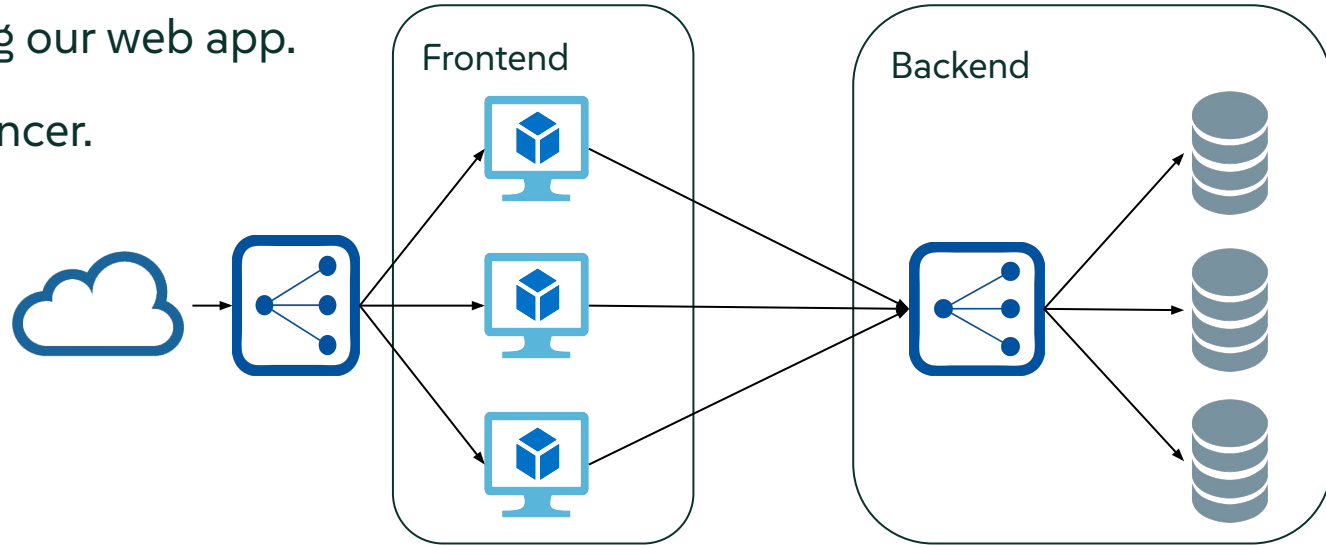
NAME	STATUS	ROLES	AGE	VERSION
k3d-k3s-cl-agent-0	Ready	<none>	9h	v1.30.4
k3d-k3s-cl-server-0	Ready	control-plane,master	9h	v1.30.4

3. Deploying apps on K8s: Pods and Deployments

Application Deployment

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

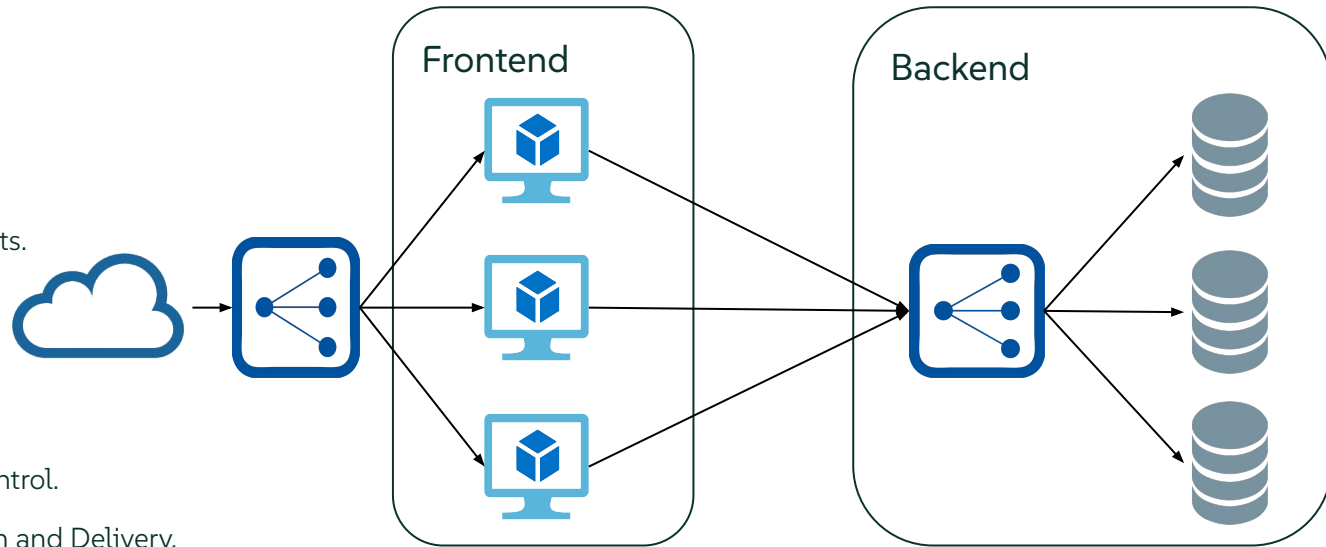
- External load balancer.
- Set of VMs running our web app.
- Internal load balancer.
- Set of databases.



Application Deployment

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

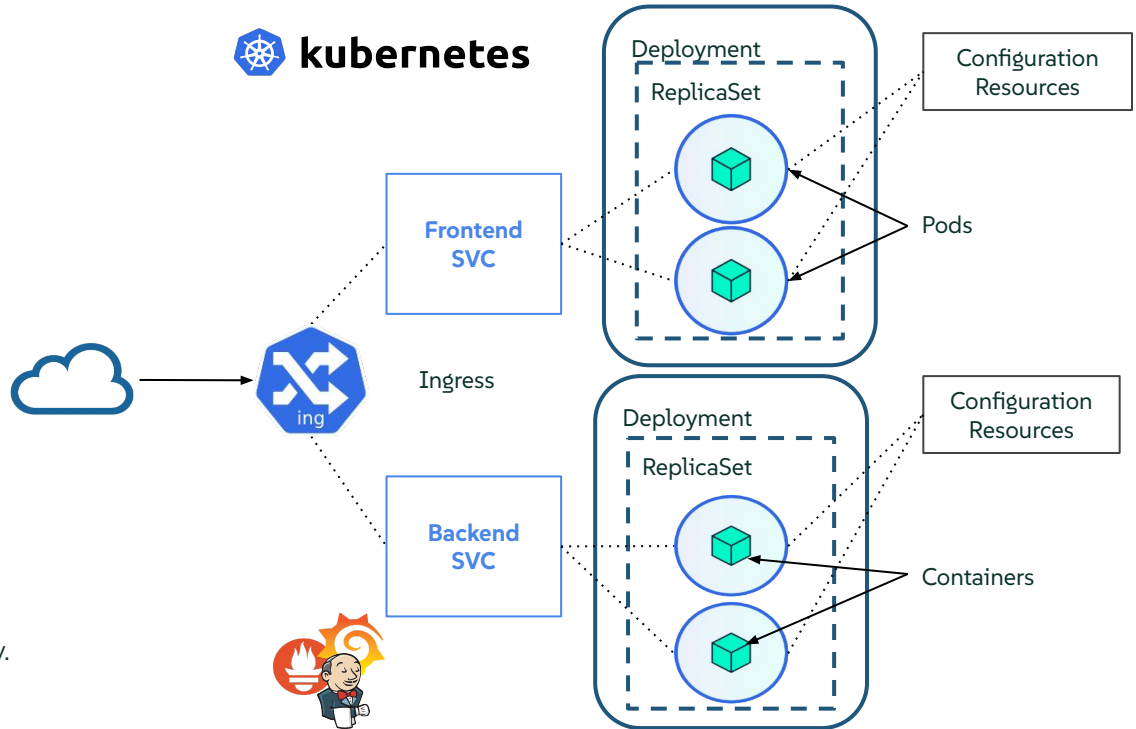
- However, this is not enough, we want:
 - Zero downtime.
 - Failover mechanisms.
 - Dynamic scalability.
 - Rolling updates.
 - Automatic deployments.
 - Load Balancing.
 - Easy migration.
 - Monitoring.
 - Role-Based Access Control.
 - Continuous Integration and Delivery.



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.

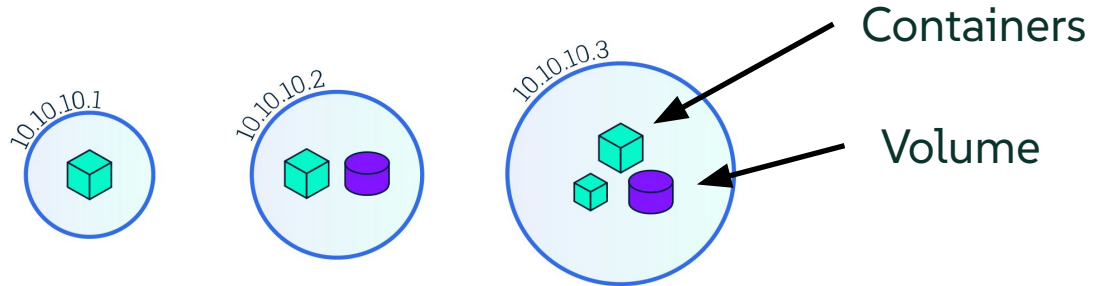




Basic Objects: Pod

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**.



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.

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**.

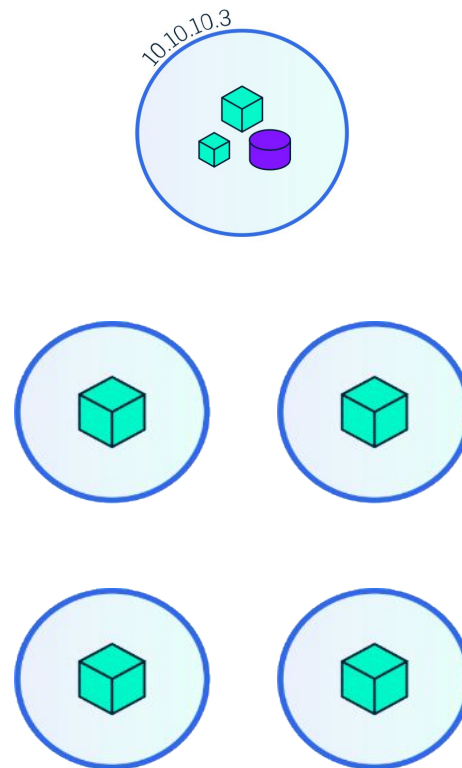
información contextual

Hands-on!



Question: multi-container or multiple pods?

- **NGINX** and its **PHP-FPM** module.
- **Wordpress** and its **MariaDB** database.
- MongoDB **primary** and **secondaries** nodes.





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
```

Describing K8s objects

los objetos pueden ser tres: pods, deployment y servicios



How does a K8s object look like? - Labels

```
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
```


Hands-on!



Creating a pod

mongo-pod.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

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
```

las etiquetas son importantes para gestionar después los servicios

- **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).

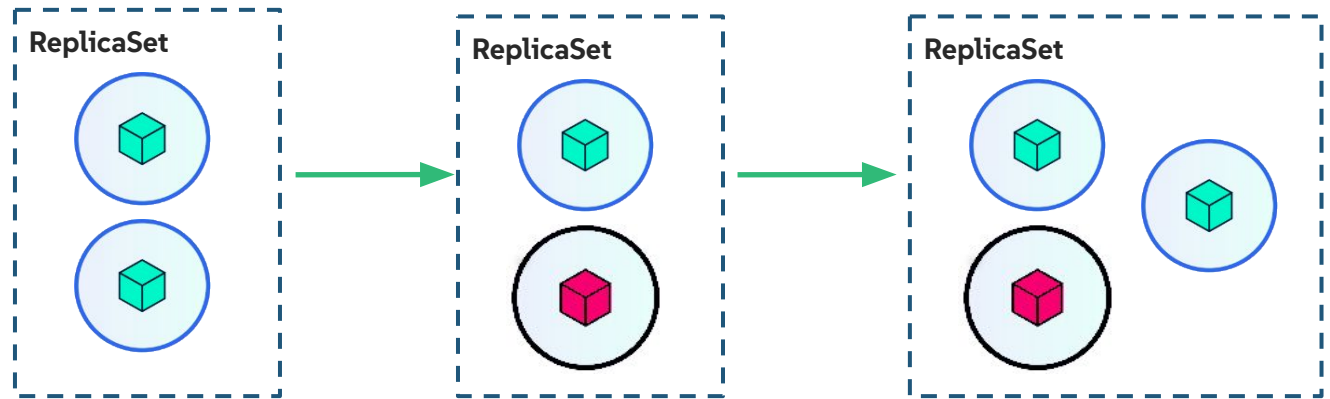
Basic Objects: ReplicaSet



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.



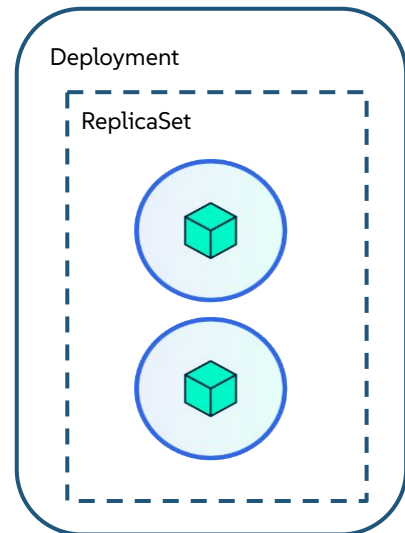
kubernetes.io/docs/concepts/workloads/controllers/replicaset

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. esto es común a RS y StS
 - 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.



es igual que el deployment con extras: ordenación, es único, red y persistencia



Hands-on!

Creating our first deployment: a simple web server with nginx

```
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
```

nginx-deploy.yaml

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.

Hands-on!



Creating our first deployment: a simple web server with nginx

nginx-deploy.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

```
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
```

para automatizar el reescalado en base a los recursos de las maquinas que tenemos, se usa hpa

```
# 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

```
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
```

vuelta atrás a una versión previa

```
# 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 replicaset
> kubectl get replicaset
```

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

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-vs4l	1/1	Running	0	49s

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
```

Kubectl Tips and Tricks

Mastering kubectl

- A few things to remember about **kubectl**.

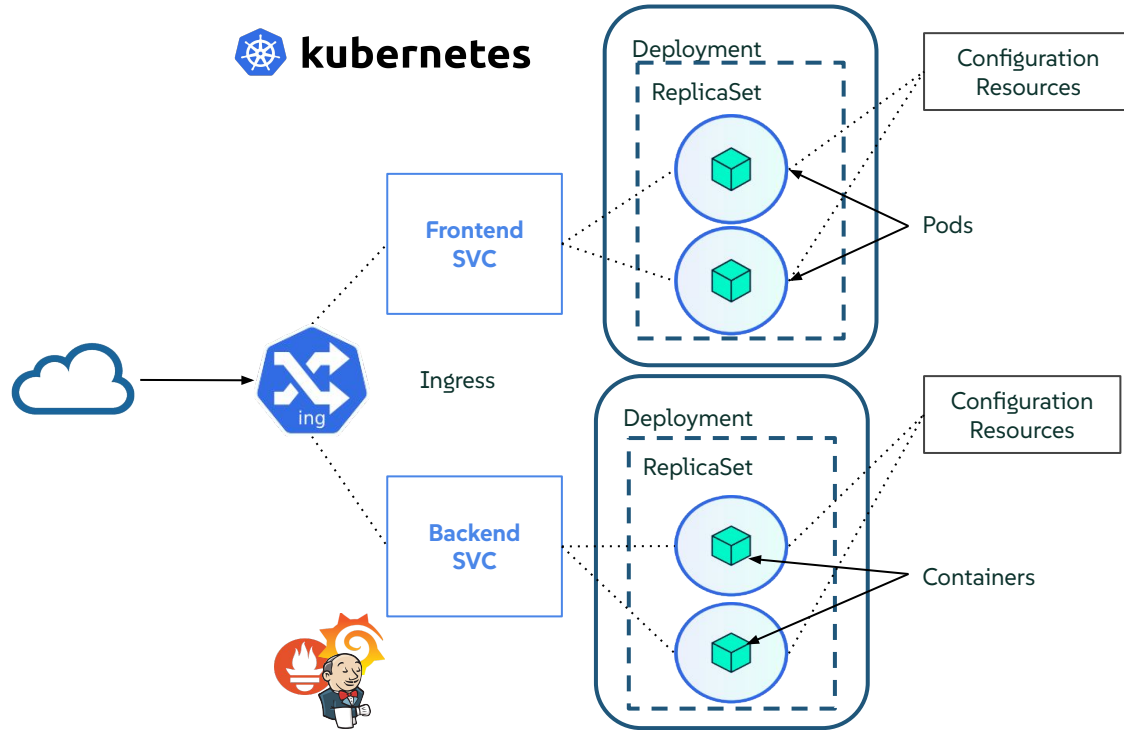
- And if you don't, check the [cheat sheet](#).

-help bien documentado

```
> 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?

4. Accessing to our apps: Services

Basic Objects: Services

dada la volatilidad de los los pods, necesitamos un endpoint estable para apuntar al servicio.
Por esto existen los servicios.



What it is?

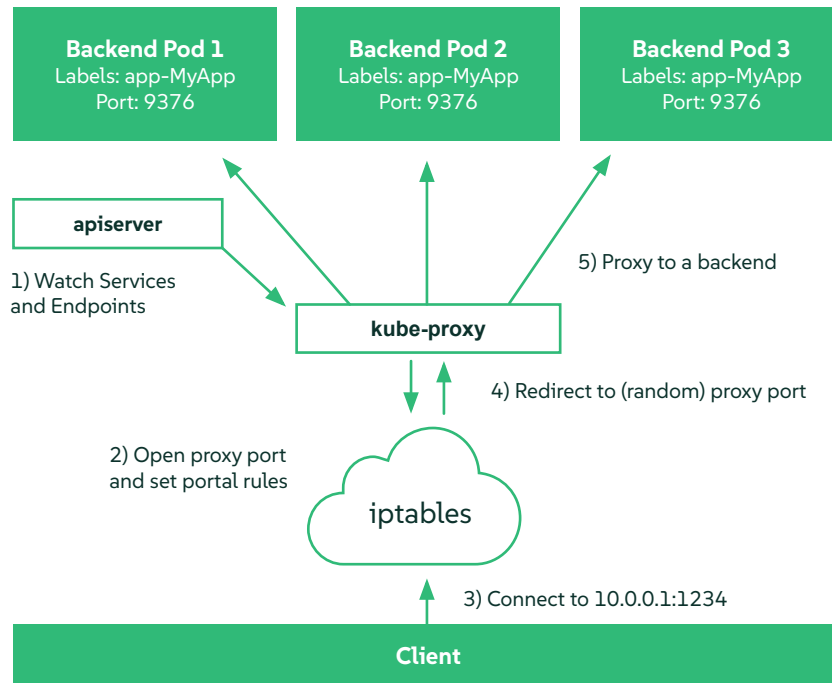
- The key question is: how do you **access your applications**?
- The answer is **Services**, yet another Kubernetes object.
 - An abstract way to **expose an application** running on a set of Pods as a network service.
 - They provide a **stable virtual endpoint** for **ephemeral Pods** in the cluster.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

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.



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

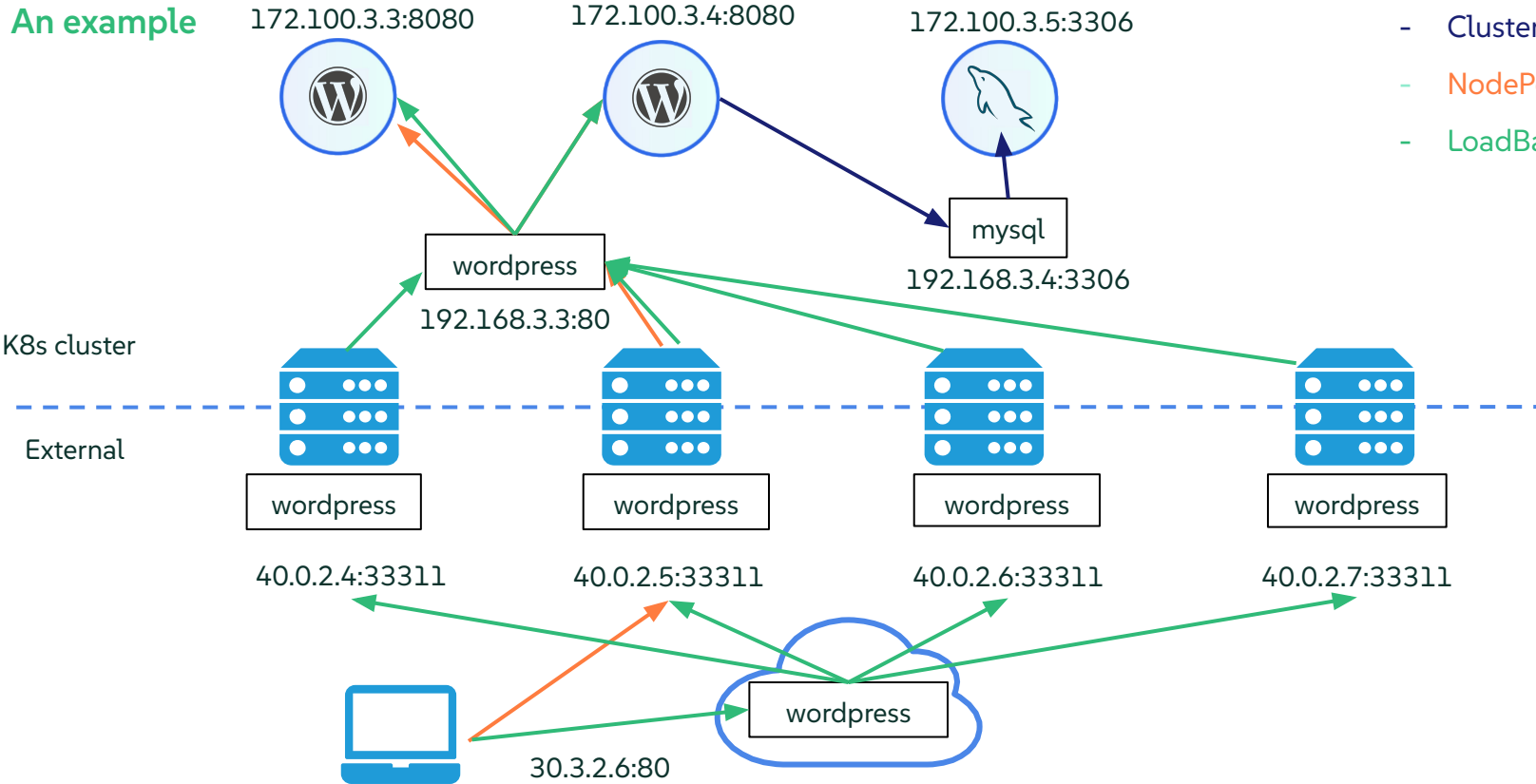
- Exposes the Service externally using a **cloud provider's load balancer** (like GKE, AKS, AWS, ...).
 - Usually add extra charges for its usage.
- Private clouds may also implement it with a **Cloud Provider Plugin**.
 - e.g., [Kind + Metalib](#) or [k3s's klipper-lb](#).

Basic Objects: Services



Types:

- ClusterIP
- NodePort
- LoadBalancer



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>

Hands-on!

Creating a service



nginx-svc.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).

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
```

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.

SUSE Academic Training Program



What it is?

- **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:

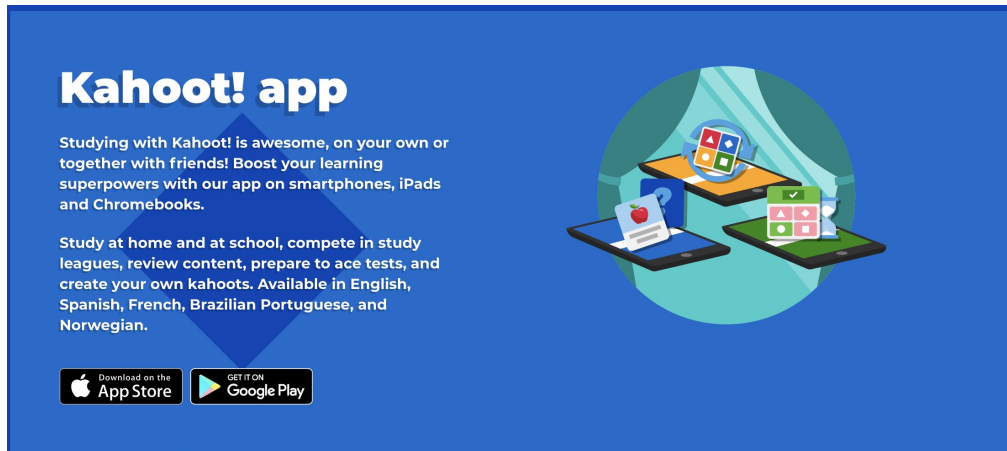
- Certification Exams.
- eLearning Subscriptions (Silver and Gold).

More information at suse.com/c/125963

Quiz time!

We might have some prizes for you :)

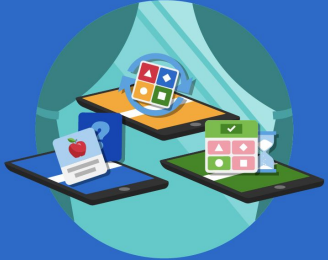
- Join using the Kahoot app or kahoot.it



Kahoot! app

Studying with Kahoot! is awesome, on your own or together with friends! Boost your learning superpowers with our app on smartphones, iPads and Chromebooks.

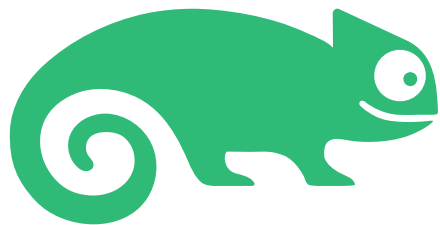
Study at home and at school, compete in study leagues, review content, prepare to ace tests, and create your own kahoots. Available in English, Spanish, French, Brazilian Portuguese, and Norwegian.



Download on the App Store

GET IT ON Google Play





SUSE



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](https://www.suse.com/training)

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