entigo

Kubernetes ABC

Martin Vool, Entigo

Tallinn 2020

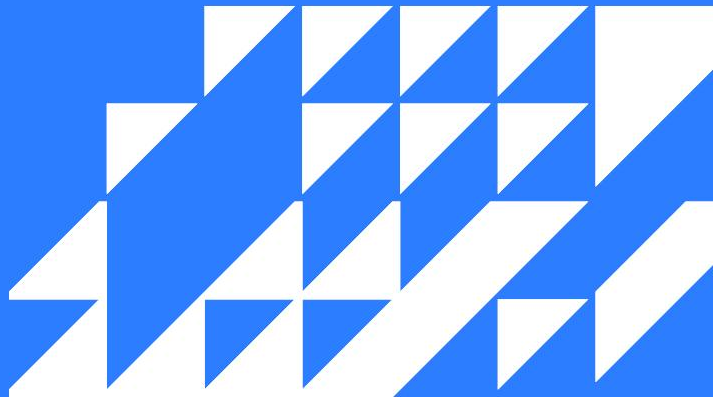


Table of contents

1. Introduction

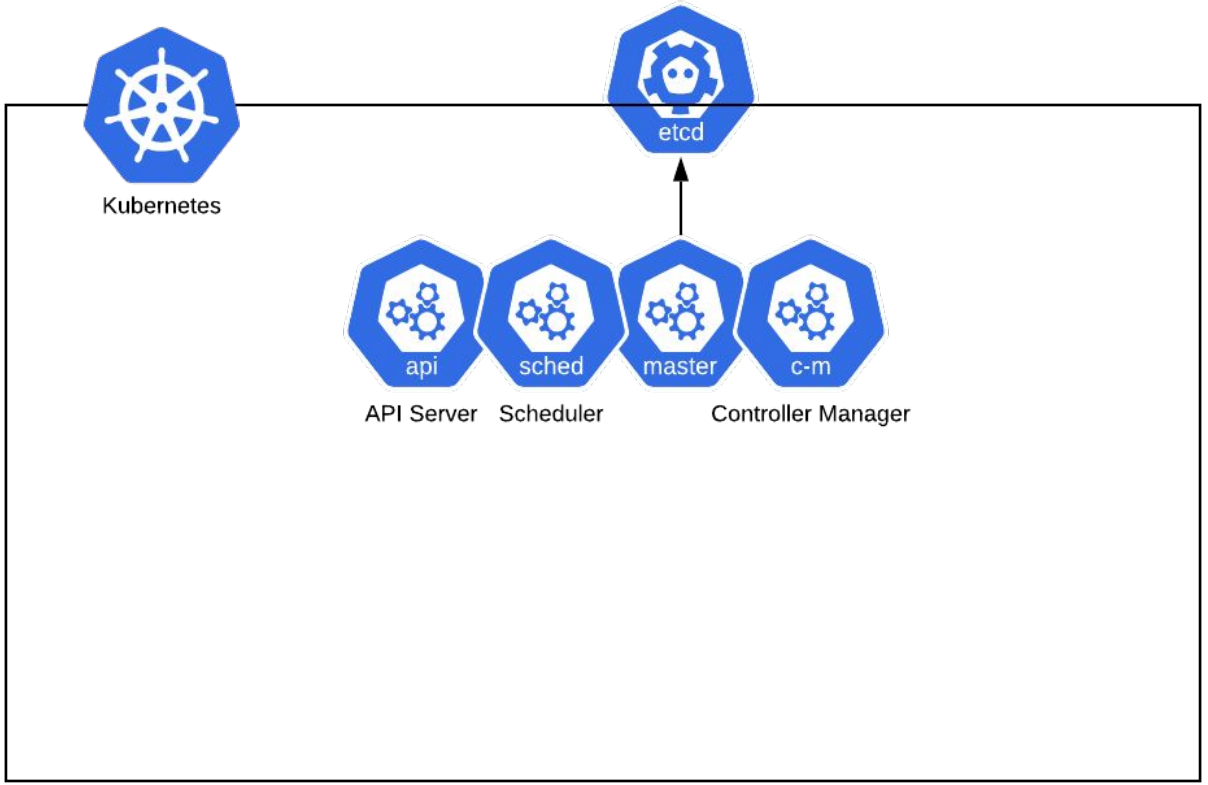
- Based on the concept of Google Borg(2003-2004)
- Kubernetes initial release 7 June 2014
- Kubernetes was designed for medium to big deployments
- Very large developer base and interest
- Releases every 3 months
- Wikipedia: Kubernetes (commonly stylized as k8s) is an **open-source container-orchestration** system for automating application deployment, scaling, and management *https://en.wikipedia.org/wiki/Kubernetes*

1. Introduction

A kubernetes master

- One or more servers, can be a cloud service too
- Instructions are sent to REST API
- Keeps state in an etcd database
- Components:
  - API server
  - Scheduler
  - Controller manager
- Controls Nodes
- Sometimes called a Master Node

1. Introduction



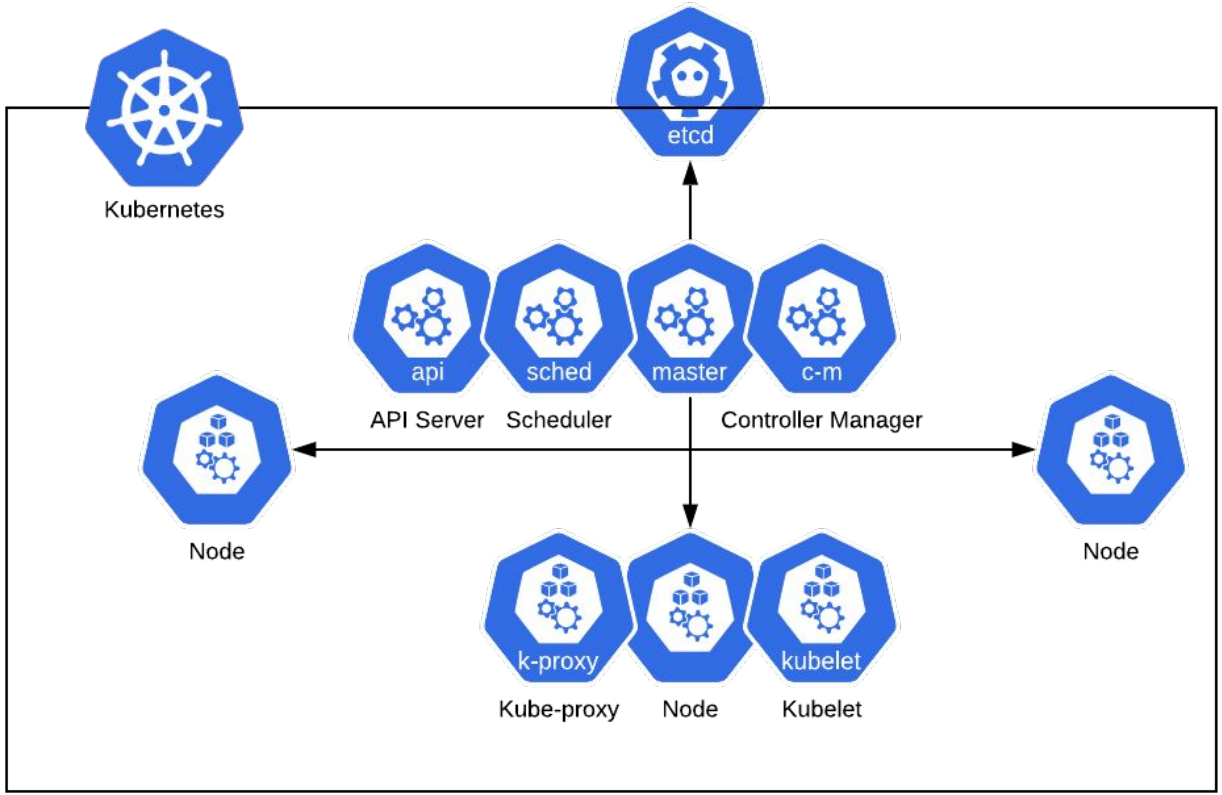
entigo

1. Introduction

A kubernetes Node

- Also called **minions** or **workers**
- Run **kubelet** and **kube proxy** daemons
- All of them run the docker engine, rkt or ...
- They run the applications/containers
- Masters Controller manager tells Nodes what to run, how to expose applications, how-to commit changes and so on
- A Master can also be a Worker Node itself – but it is not advised

1. Introduction



entigo

1. Introduction

- When the master dies, all the applications on nodes keep running, but changes will not happen
- Without the master pods will not be restarted
- A master is not required for application network communication
- Multiple masters may not be required
- Multiple clusters might be a better idea
- Having HA etcd might be a good idea

2. Kubernetes building blocks

- **Namespace** – virtual cluster on top of the “physical” kubernetes cluster. Used to separate teams or applications. Almost all kubernetes resources are in a namespace
- **Pods** - the lowest compute unit of Kubernetes. It can be a single container or multiple containers
- **Deployment** - allow updates to a set of pods at a specified rate. They are needed for rolling updates, rolling back, auto scaling. Manages ReplicaSets
- **ReplicaSet** - maintain a stable set of Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods

2. Kubernetes building blocks

- **Replication Controllers(RC)** - They provide a declarative definition of what a pod should be and how many you want running. **Deprecated in favor of Deployments**

2. Kubernetes building blocks

- **Service** – used to expose pods. Give multiple pods a DNS name to load-balance across them
- **Ingress** – manages external access to services inside the cluster. Often provides SSL offload and name based virtual hosting
- **Configmap** – contains configuration files or key-value pairs. Allow you to decouple configuration artifacts from image content to keep containerized applications portable
- **Secret** – Similar to Configmap. Contains passwords, secrets, certificates
- ...there is a lot more

3. Overview of LAB

- The labs are independent from each other
- If you have another kubernetes cluster to your disposal you can use it.
- Otherwise you should be able to log in to:

  https://k8s-test.riigipilv.ee/

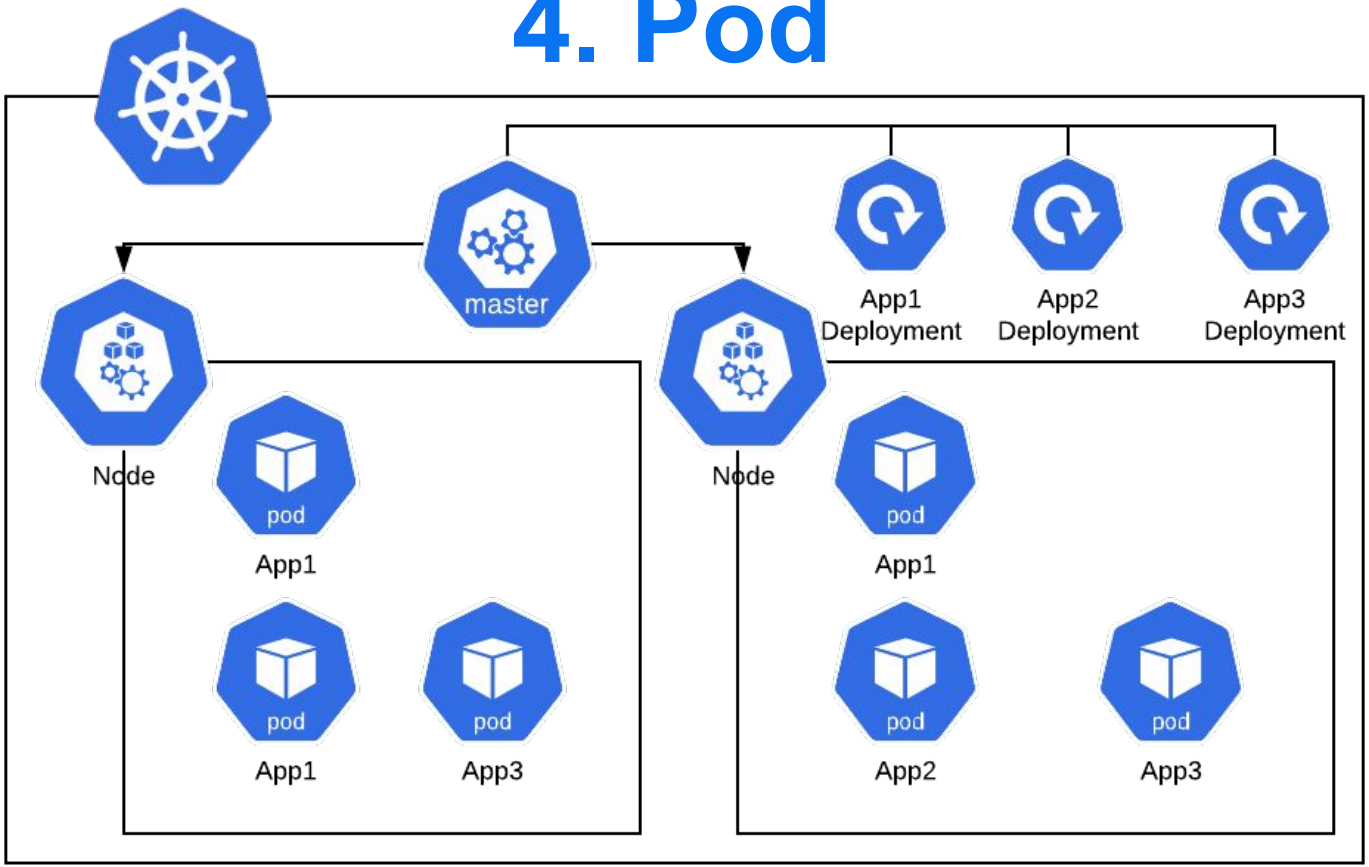
- Lab instructions are found at:

  https://abc.entigo.dev/

4. Pod

- Pods run on worker nodes
- When we do scaling, we scale Pods
- When we do a rolling update then we update Pod by Pod
- When we delete a Deployment, then the Pods of this Deployment get deleted
- When we create a Service, we provide access to pods
- One pod can only run on one worker node at a time
- Two pods of the same Deployment usually run on different worker nodes

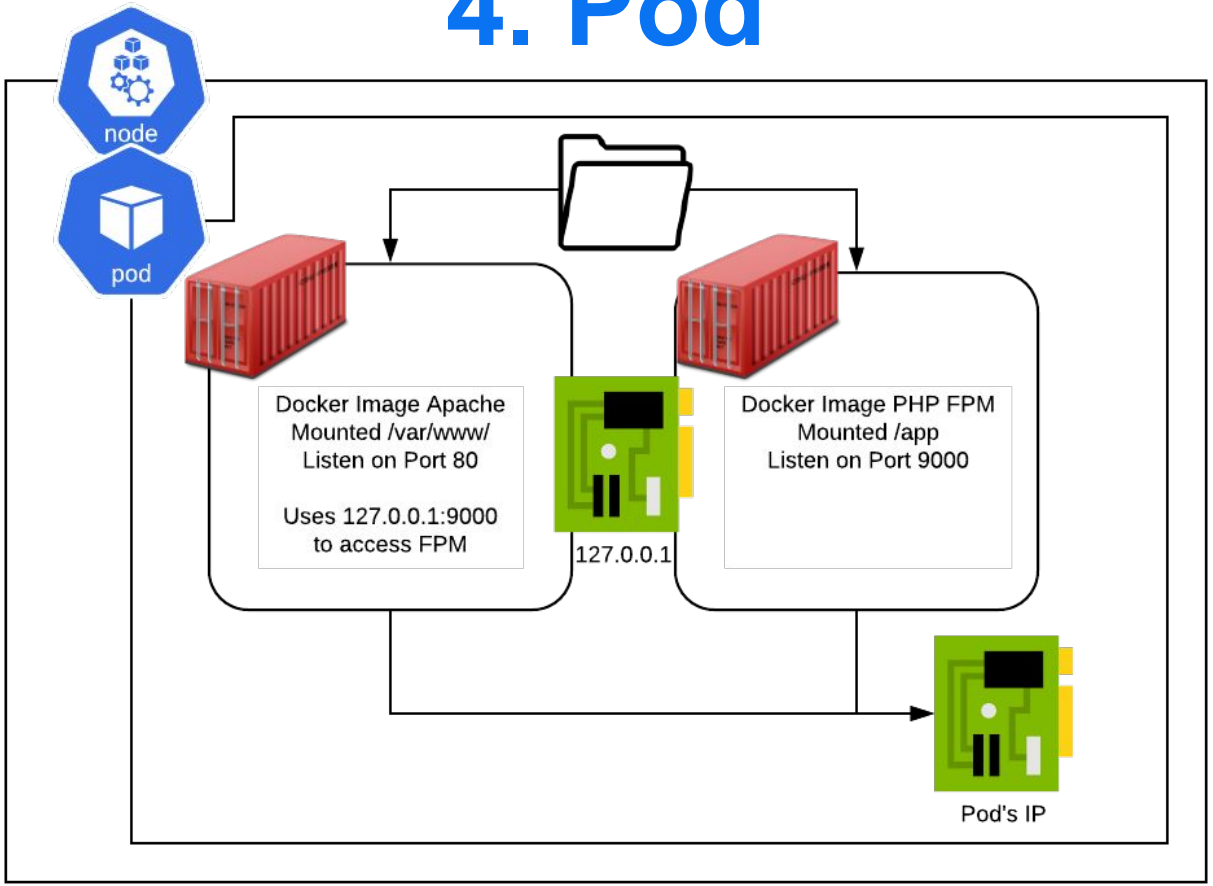
4. Pod


master
App1
Deployment
App2
Deployment
App3
Deployment
Node
pod
App1
pod
App1
pod
App3
Node
pod
App1
pod
App2
pod
App3

4. Pod

- The lowest compute unit of Kubernetes
- It can be a single container or multiple containers
- Has unique IP in kubernetes cluster
- Only tightly coupled containers should be in same pod
- Multiple containers in one pod:
  - Have their own process table
  - Don't see the other containers processes
  - Have their own filesystem
  - Can have common storage mounted
  - Share the network – IP and Localhost

4. Pod


node
pod
Docker Image Apache
Mounted /var/www/
Listen on Port 80
Uses 127.0.0.1:9000
to access FPM
127.0.0.1
Docker Image PHP FPM
Mounted /app
Listen on Port 9000
Pod's IP

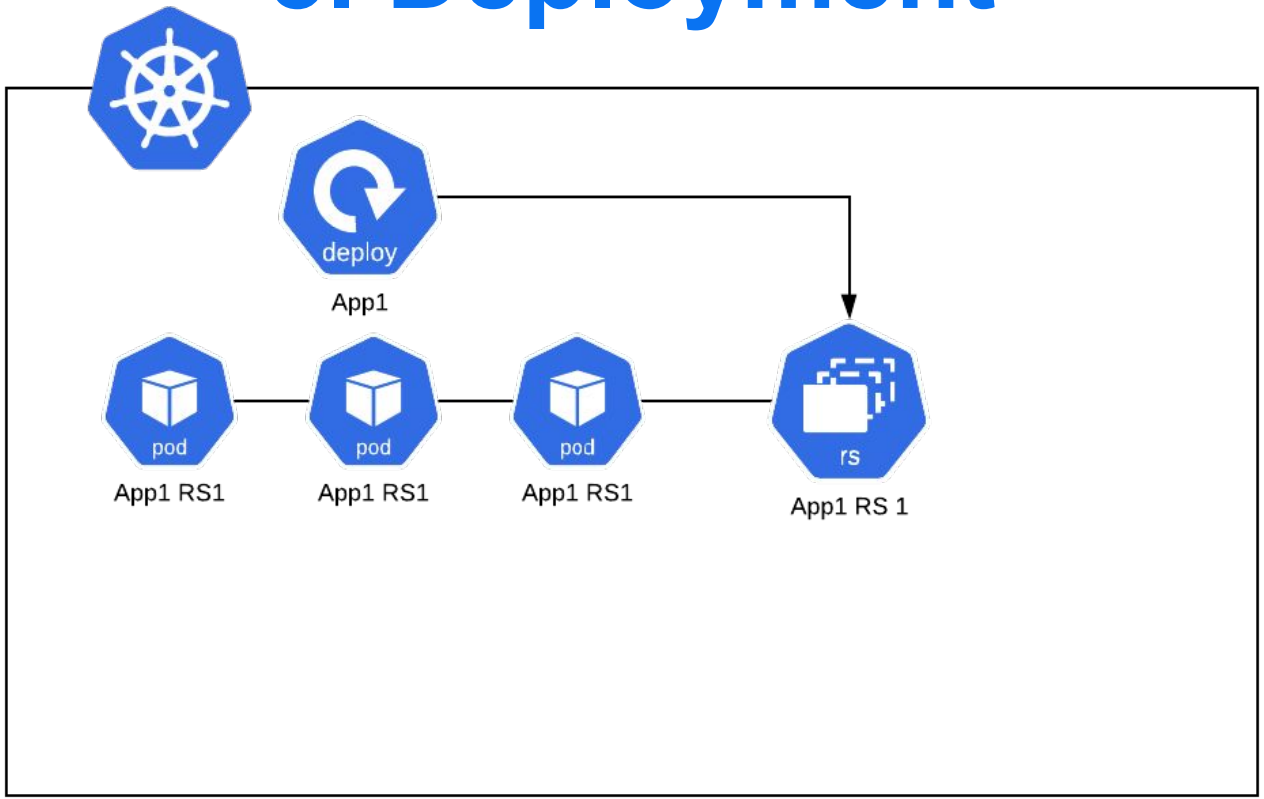
4. Pod

- You can start lab 4 now

5. Deployment

- Contains POD template
- Creates and manages Replica Sets
- Scale up and Scale down
- Roll out new versions of pods
- Able to roll back
- Best for stateless applications
- Different update strategies (Recreate, Rolling)

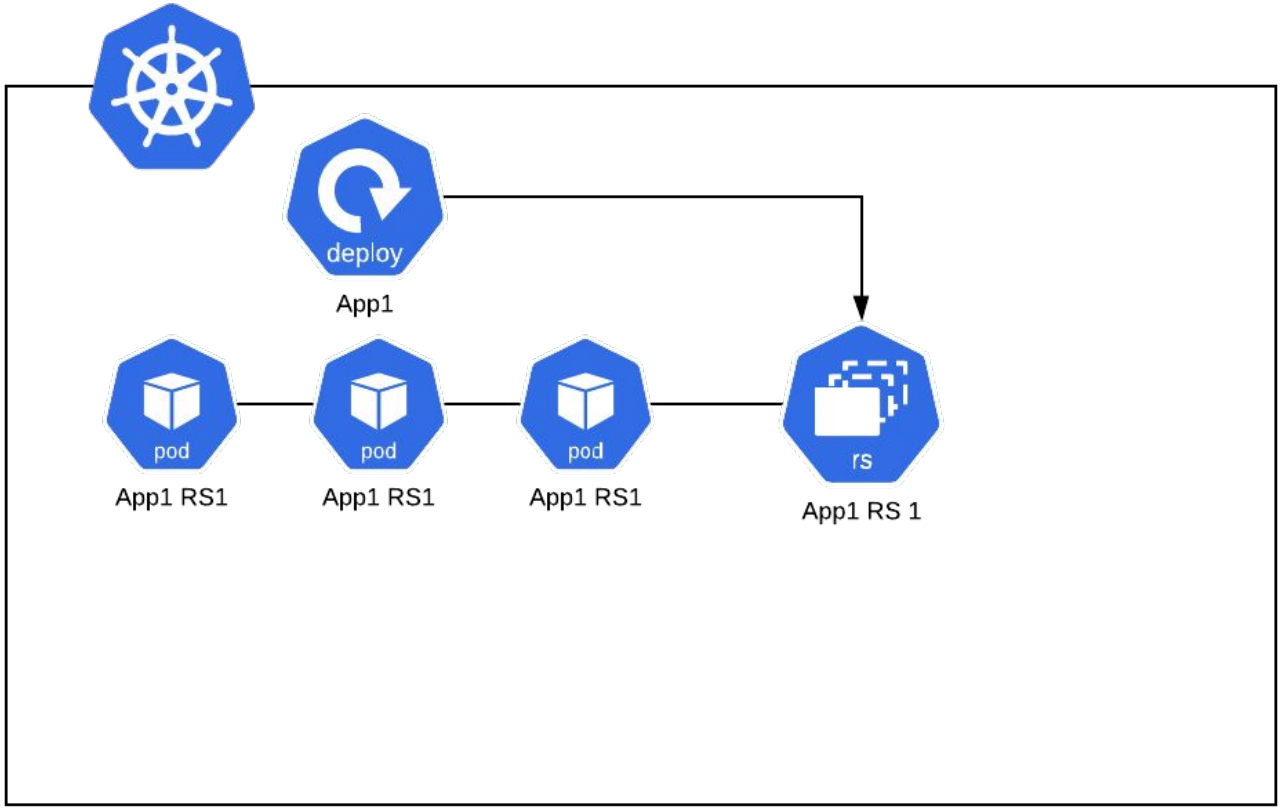
5. Deployment



5. Deployment

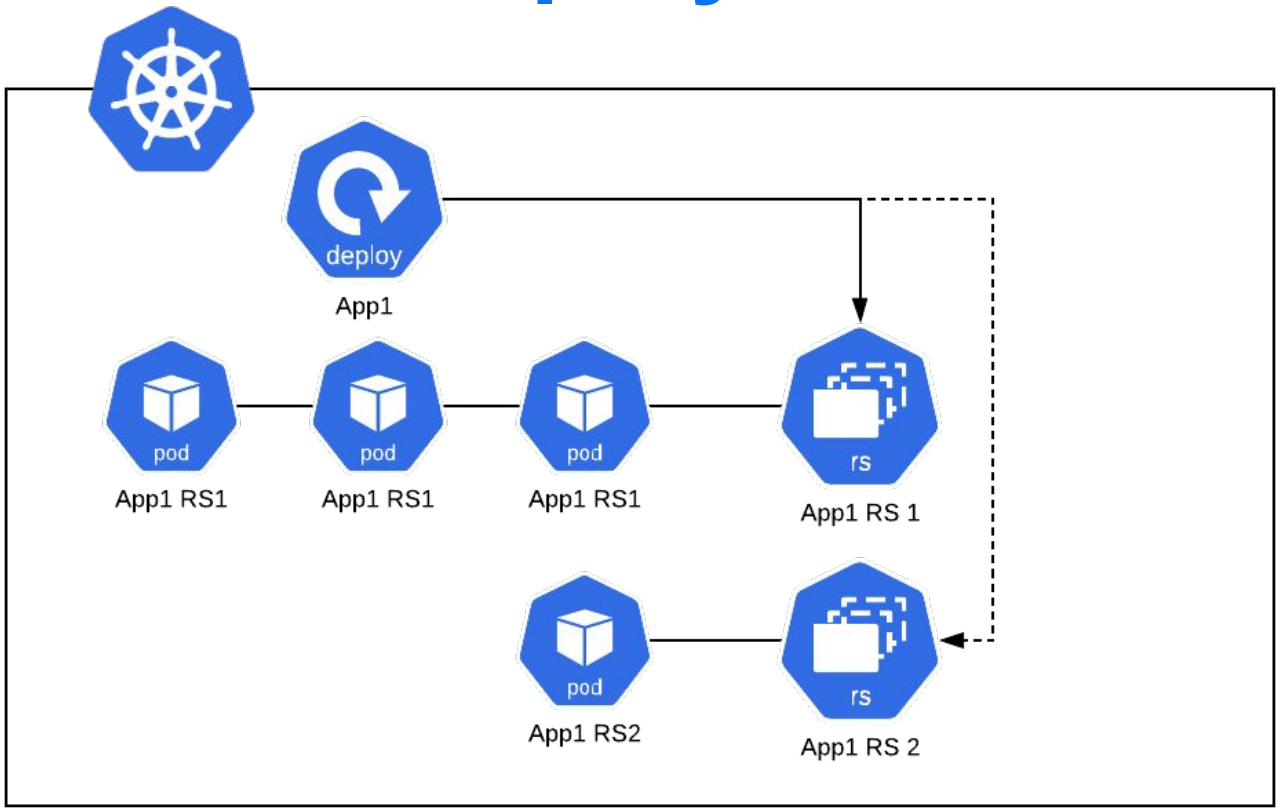
- Deployment update strategies are:
- Recreate
  - Scales the existing ReplicaSet to 0(removes Pods)
  - Creates a new ReplicaSet with all pods scaled up
- Rolling
  - Creates a new ReplicaSet with 0 pods
  - According to policy start scaling up the new RS and scale down the old RS until the old is 0 and new is the desired capacity of the deployment

5. Deployment



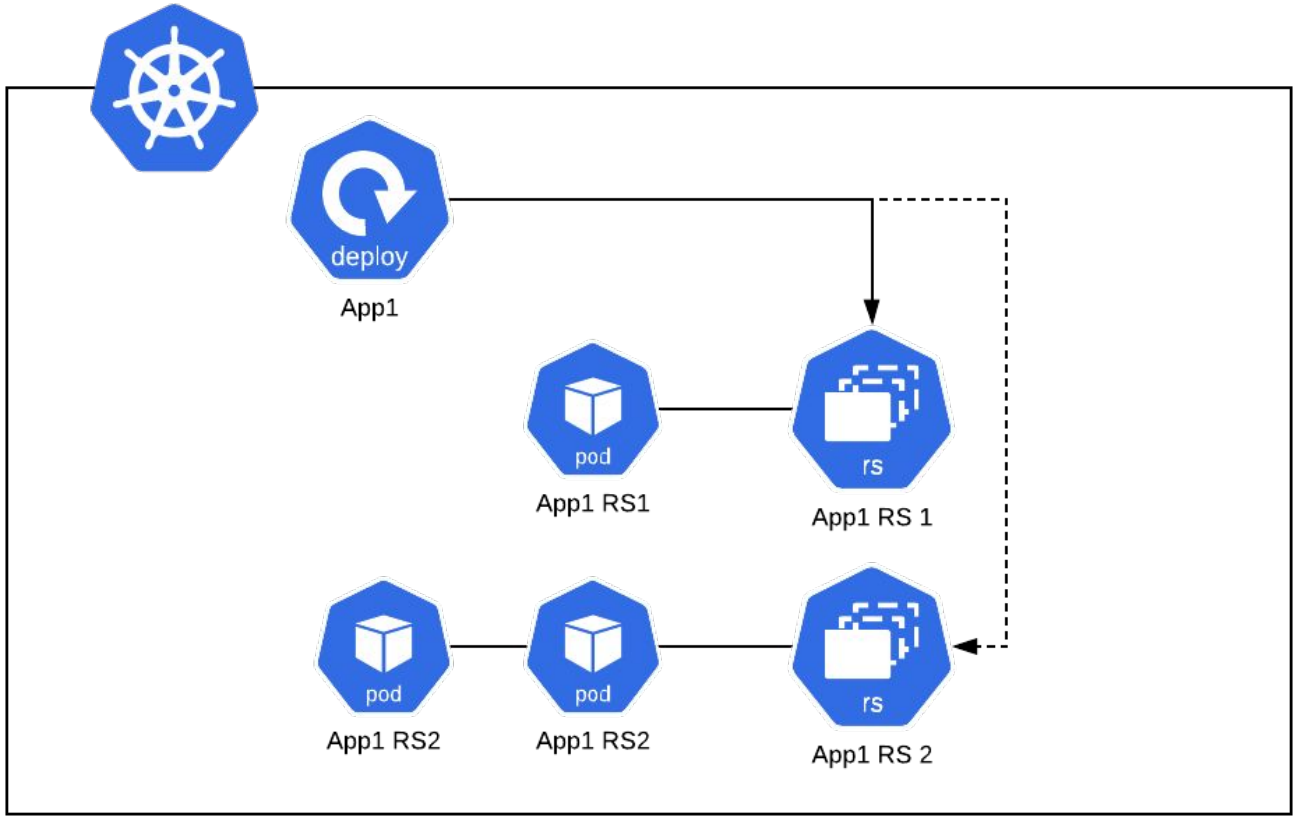
entigo

5. Deployment



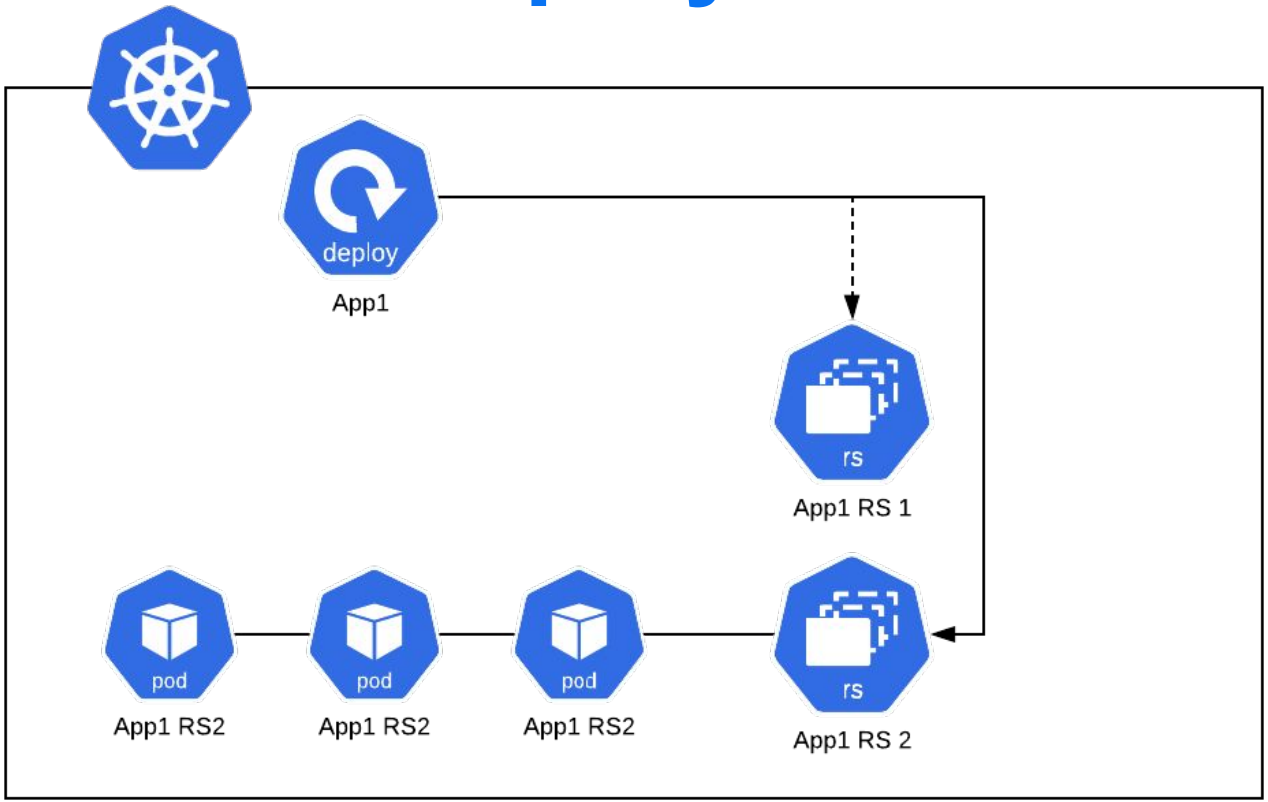
entigo

5. Deployment



entigo

5. Deployment



entigo

5. Deployment

- You can use the kubectl command to change image versions
- You can change the yaml and apply it again with kubectl
- In most cases some sort of CI system is used that communicates with kubernetes cluster

5. Deployment

- You can start lab 5 now

6. Services

- Services direct network traffic to a set of pods that provide a service
- Services provide a virtual endpoint for pods in your cluster. Other applications can target them and will be loadbalanced to the group of pods
- The kube-proxy agent running on all the Kubernetes nodes watches the Kubernetes API for new services and endpoints being created
- It opens ports on nodes to listen to the traffic and redirects to service endpoints
- Services can be of four types: **ClusterIP, NodePort, LoadBalancer, ExternalName**

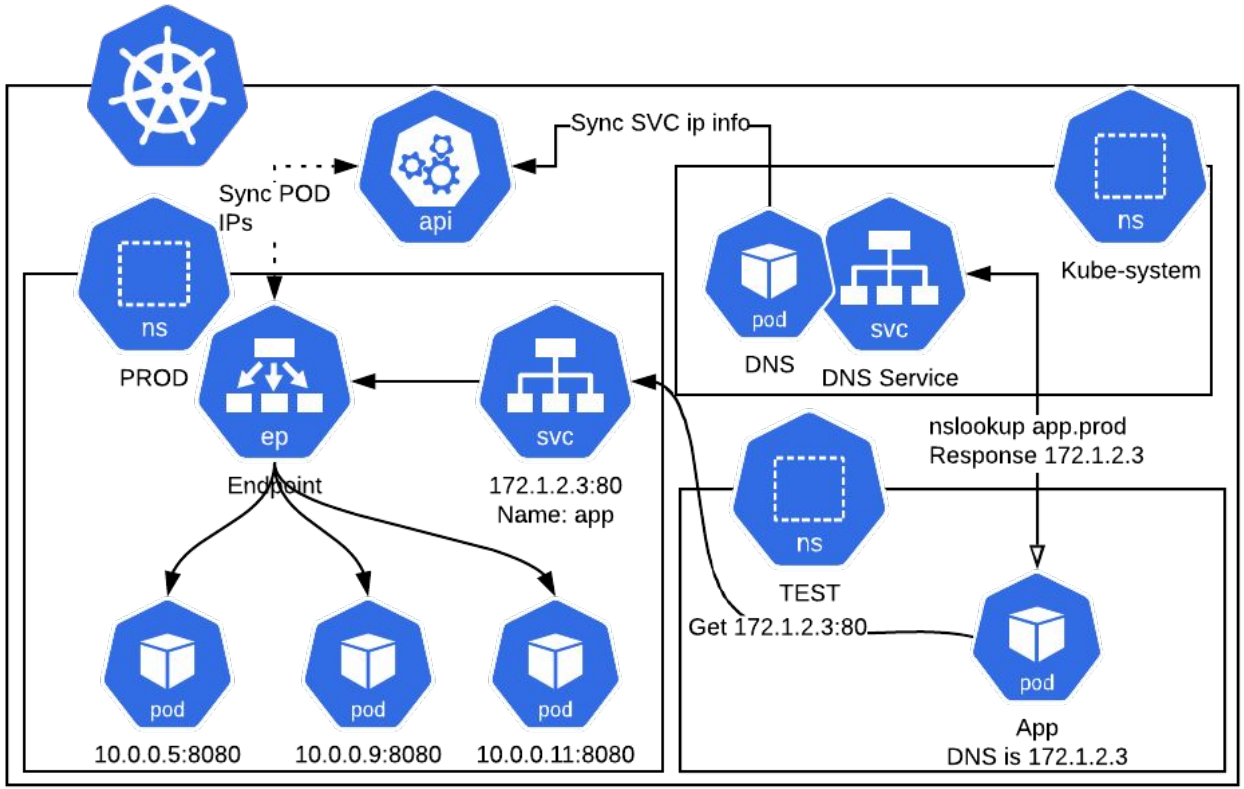
6. Services

- ClusterIP service type is the default and only provides access inside the cluster. The IP address range used is different from the pods
- NodePort will make the service listen on the worker node interface. It is not recommended for public access but can be used for it
- LoadBalancer services are currently only implemented on public cloud providers like GKE and AWS. Others have to rely on Ingress
- ExternalName Service to a DNS name. Instead of IP, kubernetes DNS will return the CNAME of the DNS name specified

6. Services

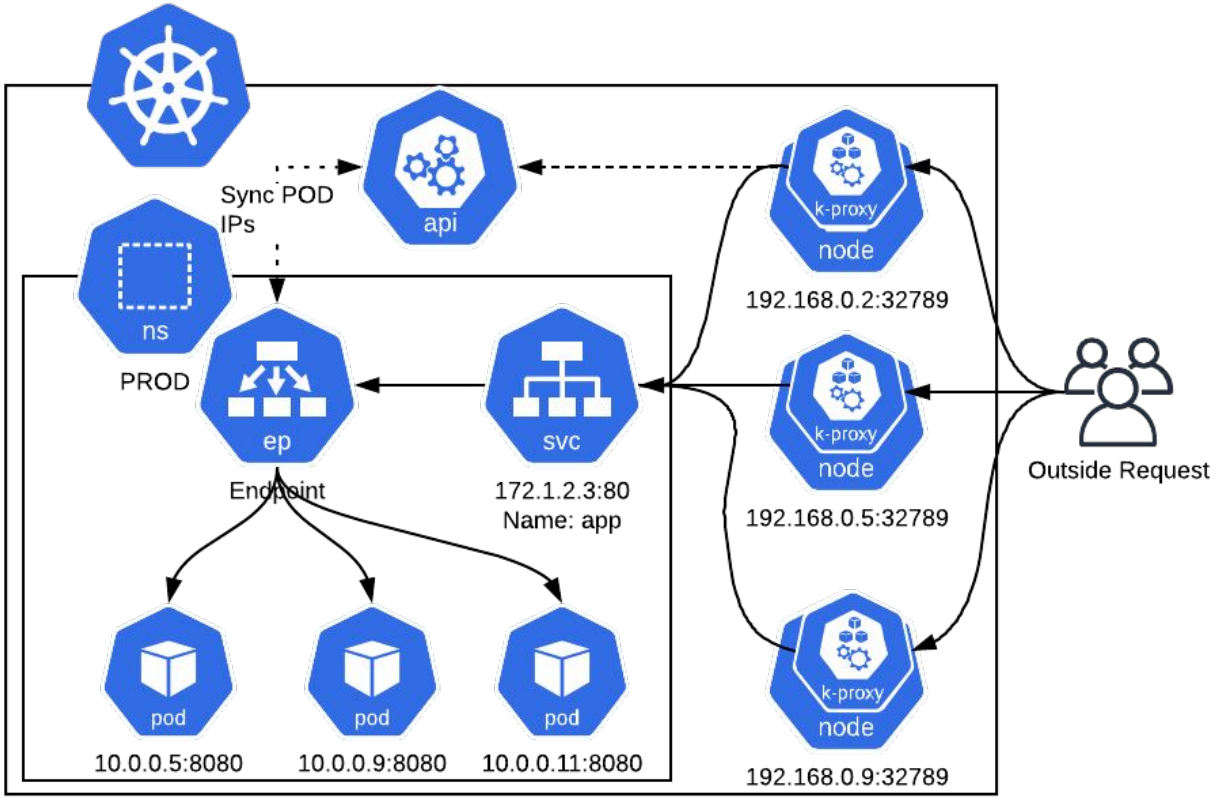
- Kubernetes schedules DNS Pods and Service on the cluster. Worker nodes to tell containers to use the DNS Service's IP to resolve DNS names
- Every Service defined in the cluster (including the DNS server itself) is assigned a DNS name. By default, a client Pod's DNS search list will include the Pod's own namespace and the cluster's default domain
- Imagine a Service named "app" in the Kubernetes namespace "prod". A Pod running in namespace "test" can look up this service by simply doing a DNS query for "app.prod". A Pod running in namespace "prod" can look up this service by doing a DNS query for "app"

6. Services



entigo

6. Services



entigo

6. Services

- You can start lab 6 now

7. Data and volumes basic

- Containers in a Pod can have data volumes mounted
- A pods defined volumes can be mounted in a specific location in each container
- The environment and purpose of the volume determines the type to use.
- Whether you need scratch space, share data between containers, persistent storage…

7. Data and volumes basic

- **EmptyDir** – temporary storage assigned from the worker node where the pod is running. Deleted when the pod is deleted. Survives container restarts
- **HostPath** - volumes from the worker node filesystem. Volume is not cleared when pod is deleted. It is a specific path on the worker node
- **PersistantVolumeClaim** – is an abstract storage resource. It is used to decouple the Pod from the underlying environment. Your Pod template should not need to change when it is deployed to a different infrastructure

7. Data and volumes basic

- You can start lab 7 now

8. Configmap and Secret

- **Configmap** – contains configuration files or key-value pairs
- Is used to make pod templates universal across environments
- As the name suggest they provide a container with configuration
- In prod and test you would have different configmap but the same pod templates

8. Configmap and Secret

- **Secret** – Similar to Configmap. Contains passwords, secrets, certificates, docker registry access
- Secret values are base64 encoded
- Used to separate delicate and non delicate information
- Also related to access rights
- Developer might have access to prod configmaps but not secrets
- Configmaps and Secrets can be environment variables or volumes that provide configuration files

8. Configmap and Secret

- kubectl create secret docker-registry <name> --docker-server=REGISTRY_SERVER --docker-username=USER --docker-password=PASSWORD –docker-email=EMAIL

```
kind: Pod

spec:

  containers:

    - name: foo

      image: REGISTRY_SERVER/myimage:latest

  imagePullSecrets:

    - name: <name>
```

8. Configmap and Secret

- You can start lab 8 now

9. Liveliness and Readiness

- **Liveliness** – is a health probe that determines if the container in the pod is working or not. If the probe fails then the container is restarted
- **Readiness** – is a health probe that determines if the container in the pod is ready to accept traffic
- A Pod is considered ready when all of its Containers are ready. When a Pod is not ready, it is not participating in a Service

9. Liveliness and Readiness

- **initialDelaySeconds**: Number of seconds after the container has started before probes are initiated
- **periodSeconds**: How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1
- **timeoutSeconds**: Number of seconds after which the probe times out. Defaults to 1 second. Minimum value is 1
- **successThreshold**: Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Only changed for Readiness
- **failureThreshold**: Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1

9. Liveliness and Readiness

- **Command probe**
- The kubelet of the worker node executes a command inside the Container
- If the command succeeds, it returns 0, and the probe is a success. If the command returns a non-zero value, the probe is considered failed
- If it is a liveliness probe – the kubelet might restart it when it is failing
- It is is a readiness probe – the service might include or exclude it – depending on the state

9. Liveliness and Readiness

- **HTTP probe**
- Kubelet of the worker node sends a HTTP GET request to the Container. Any code **greater than or equal to 200 and less than 400** indicates success. Any other code indicates failure

9. Liveliness and Readiness

- You can start lab 9 now

Thank you!

Martin Vool, Entigo

Tallinn 2020