

Manu Jose

# First Experiences with Kubernetes at UBFFM

**AG Technische Infrastruktur, 19.11.2020**

# Why containers?

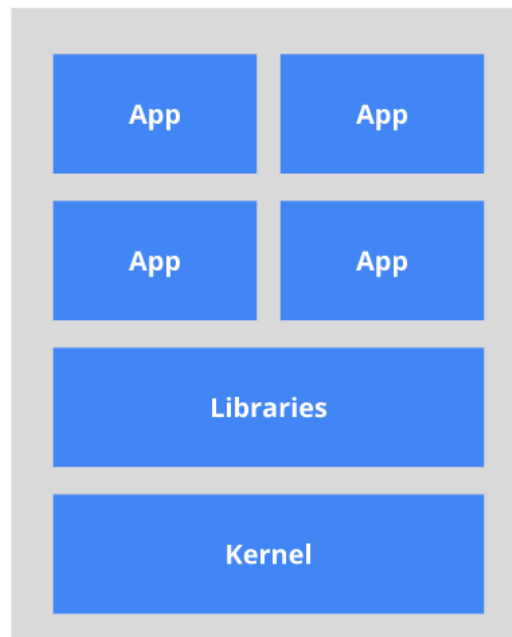
## Applications on a host

In this way applications on a host with the operating system package manager. This has disadvantage VM's are heavy and non-portable and executables, configurations, libraries, and application lifecycles are interconnected with host operating system.

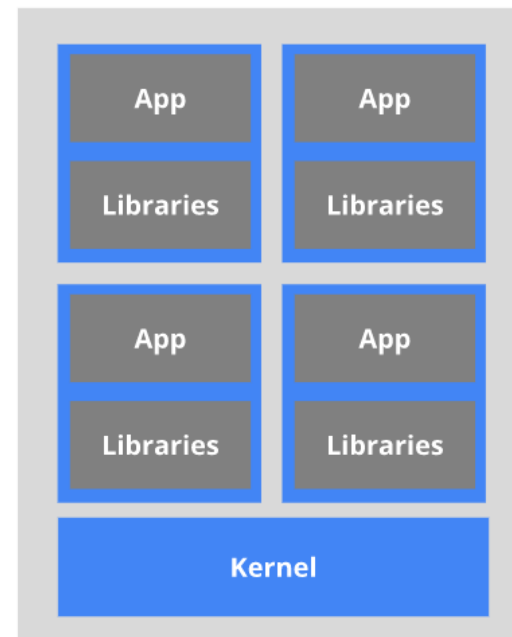
## Applications on a container

containers are small, lightweight, isolated from each other and from the host: they have their own file systems, they cannot see each other's processes, and their resource consumption can be limited.

The old way: Applications on host



The new way: Deploy containers



## Different container solutions

- 1) Docker
- 2) CoreOS rkt
- 3) Mesos Containerizer
- 4) LXC Linux Containers
- 5) OpenVZ

### Docker

#### Features

- Integrated & Automated container Security Policy.
- Runs trusted images only.
- No Lock-in: Supports almost any type of application, OS, infrastructure, and orchestrator.
- Unified and automated agile operations.
- Portable containers across the cloud.
- Automated governance.

## Docker Pros/Cons

### Pros

- Fits very well with CI/CD.
- Saves storage space.
- Plenty of docker images.
- Saves hours in patching and downtime when compared to virtualization.
- While working in a team, you need not worry about the different members having different versions of programming language, libraries, etc.
- Open source.
- A lot of plugins are available to enhance its features.

### Cons

- Quite tough to set up.
- Takes a fair amount of time to learn this tool.
- Creating persistent storage requires a lot of effort.
- Does not have a GUI.
- Does not have built-in support for Mac.

## Container-centric management environment - Kubernetes

- Kubernetes is a portable, extensible open source platform for managing containerized workloads and services.
- Google made the Kubernetes project available as an open source project in 2014
- Applications can be deployed in Kubernetes using as microservices, deployments, and pods.
- Kubernetes acts like more of an all-in-one framework when working with distributed systems.
- includes built-in tools for managing Logging and Monitoring
- Kubernetes has detailed dashboards to allow even non-technical users to control the clusters effectively.

# UB Deployment Of Kubernetes.



UNIVERSITÄTS  
BIBLIOTHEK  
FRANKFURT AM MAIN

Following are the servers/vm's used for bare-metal cluster build

## Haproxy Nodes (Proxy and Load Balancer for Master Nodes)

No.	Name	RAM	VCPU	OS-Distribution
1	Node 1	4	1	Centos -8
2	Node 2	4	1	Centos -8

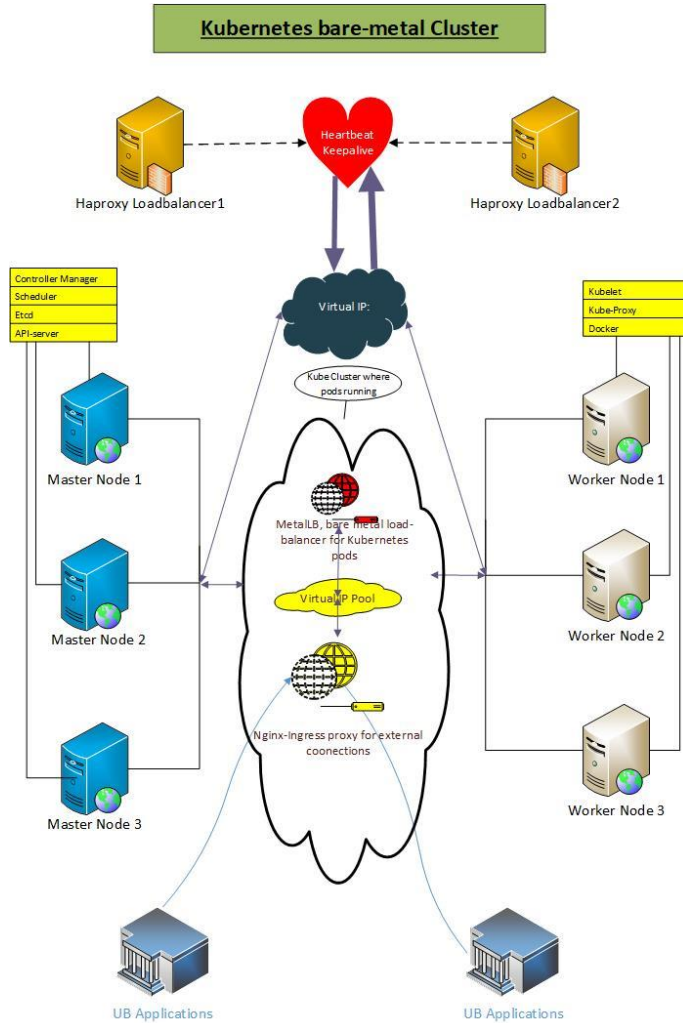
## Master Nodes

No.	Name	RAM	VCPU	OS-Distribution
1	Node 1	8	2	Centos -7
2	Node 2	8	2	Centos -7
3	Node 3	8	2	Centos -7

## Worker Nodes

No.	Name	RAM	VCPU	OS-Distribution
1	Node 1	32	6	Centos -7
2	Node 2	32	6	Centos -7
3	Node 3	32	6	Centos -7

## Following Diagram Shows our Cluster Structure.



## Components of Kubernetes Master Machines (Master)

### **etcd**

It stores the configuration information which can be used by each of the nodes in the cluster. It is a high availability key value store that can be distributed among multiple nodes. It is accessible only by Kubernetes API server as it may have some sensitive information. It is a distributed key value Store which is accessible to all.

### **API Server**

Kubernetes is an API server which provides all the operation on cluster using the API. API server implements an interface, which means different tools and libraries can readily communicate with it. Kubeconfig is a package along with the server-side tools that can be used for communication. It exposes Kubernetes API.

### **Controller Manager**

This component is responsible for most of the controllers that regulates the state of cluster and performs a task. In general, it can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server. It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state. The key controllers are replication controller, endpoint controller, namespace controller, and service account controller. The controller manager runs different kind of controllers to handle nodes, endpoints, etc.

### **Scheduler**

This is one of the key components of Kubernetes master. It is a service in master responsible for distributing the workload. It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload. In other words, this is the mechanism responsible for allocating pods to available nodes. The scheduler is responsible for workload utilization and allocating pod to new node.



## Components of Kubernetes worker Machines (worker)

### Docker

The first requirement of each node is Docker which helps in running the encapsulated application containers in a relatively isolated but lightweight operating environment.

### Kubelet Service

This is a small service in each node responsible for relaying information to and from control plane service. It interacts with etcd store to read configuration details and write values. This communicates with the master component to receive commands and work. The kubelet process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

### Kubernetes Proxy Service

This is a proxy service which runs on each node and helps in making services available to the external host. It helps in forwarding the request to correct containers and is capable of performing primitive load balancing. It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well. It manages pods on node, volumes, secrets, creating new containers' health check-up, etc.

## Configuration Information

### 1) Cluster Networking - Container Network Interface(CNI)-Calico v3.16.1

Calico is an open source networking and network security solution for containers, virtual machines, and native host-based workloads. Calico supports a broad range of platforms including Kubernetes, OpenShift, Docker EE, OpenStack, and bare metal services.

In our Kubernetes cluster uses Calico v3.16.1 with Self-managed on-premises.

Calico supports both Kubernetes API datastore (kdd) and etcd datastores. The Kubernetes API datastore is recommended for on-premises deployments, and supports only Kubernetes workloads; etcd is the best datastore for hybrid deployments. An example of a hybrid deployment is running Calico as the network plugin for both Kubernetes and OpenStack.

So we use Kubernetes API datastore (kdd).

<https://www.projectcalico.org/>

## Configuration Information

### 2) Load Balancer for Kubernetes cluster - MetalLB v0.9.3

MetalLB hooks into your Kubernetes cluster, and provides a network load-balancer (Layer 4) implementation. In short, it allows you to create Kubernetes services of type “Load Balancer” in clusters.

It has two modes:

- 1) Layer 2 mode (ARP/NDP)
- 2) BGP

We use Layer 2 mode.

<https://metallb.universe.tf/>

## Configuration Information

### 3) Nginx-Ingress Controller -3.8.0 – Proxy and Loadbalancer (Layer 7).

Ingress is a resource that enables traffic to come into your Kubernetes cluster from the outside. This is usually to HTTP or HTTPS service (Layer 7). There are ways to map non-HTTP services to an ingress, but that is less common.

### 4) Package manager for Kubernetes - HELM v3.3.0

Helm helps you manage Kubernetes applications — Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.

<https://helm.sh/>

### 5) NFS Server Provisioner (Persistent Volumes)v1.2.9

NFS Server Provisioner is an out-of-tree dynamic provisioner for Kubernetes. You can use it to quickly & easily deploy shared storage that works almost anywhere.

<https://github.com/helm/charts/tree/master/stable/nfs-server-provisioner>

## Configuration Information

### 6) Kubernetes Metrics Server

Metrics Server is a scalable, efficient source of container resource metrics for Kubernetes built-in autoscaling pipelines.

Metrics Server collects resource metrics from Kubelets and exposes them in Kubernetes apiserver through Metrics API for use by Horizontal Pod Autoscaler and Vertical Pod Autoscaler. Metrics API can also be accessed by `kubectl top`, making it easier to debug autoscaling pipeline.

<https://github.com/kubernetes-sigs/metrics-server>

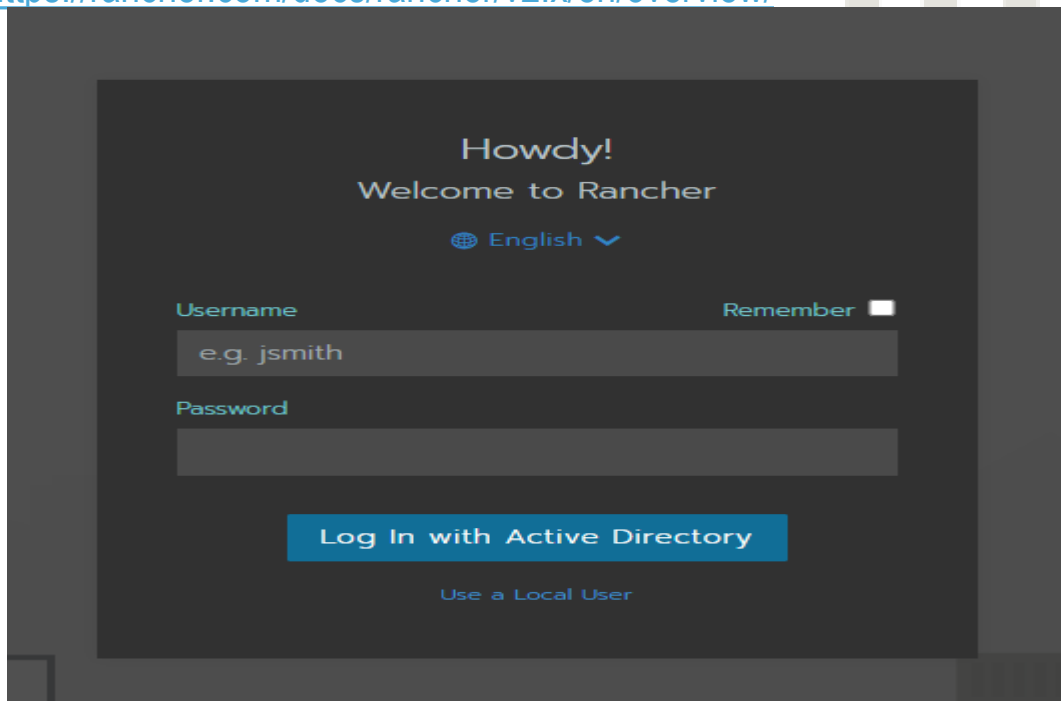
```
[root@ganymede ~]# kubectl top nodes
NAME          CPU (cores)   CPU%   MEMORY (bytes)   MEMORY%
amalthea      71m           1%     748Mi            2%
callisto      259m          12%    2118Mi           27%
dione         71m           1%     796Mi            2%
enceladus     70m           1%     704Mi            2%
ganymede      407m          20%    1653Mi           21%
chemisto      243m          12%    2088Mi           27%
[root@ganymede ~]#
```

## Configuration Information

### 7) Rancher-2 for Web bases monitoring and configuration tool.

Rancher is a container management platform built for organizations that deploy containers in production.  
Rancher makes it easy to run Kubernetes everywhere.

<https://rancher.com/docs/rancher/v2.x/en/overview/>



The screenshot shows the Rancher login page. At the top, it says "Howdy! Welcome to Rancher" with a language selector set to "English". Below this are two input fields: "Username" (with the example "e.g. jsmith") and "Password". To the right of the username field is a "Remember" checkbox. A prominent blue button labeled "Log In with Active Directory" is centered below the password field. At the bottom, there is a link that says "Use a Local User".

