Docker containers with Kubernetes : Introduction

Introduction – Kubernetes

Kubernetes is a tool used to manage clusters of **containerized applications**. In computing, this process is often referred to as **orchestration**.
Kubernetes coordinates lots of **microservices** that together form a useful application.

Understanding Kubernetes architecture is crucial for deploying and maintaining containerized applications.

Kubernetes continuously gaining importance as a runtime environment for the **development and operation of microservices**.

Kubernetes is an open-source project and under the Apache license. It is managed by the Linux foundation and was originally created at Google.
Minikube is a version of Kubernetes for installing a test and development system on a laptop.

There are more versions that can be installed on servers or used as cloud offerings:
- ✓ Amazon Elastic Container Service for Kubernetes (Amazon EKS) provides Kubernetes clusters on AWS.
- ✓ Google Cloud also supports Kubernetes with the Google Container Engine.
- ✓ Microsoft Azure provides the Azure Container Service
- ✓ IBM Bluemix provides Kubernetes with the IBM Bluemix Container Service.

kops is a tool which enables the installation of a Kubernetes cluster in different types of environments like AWS (Amazon Web Services), etc.

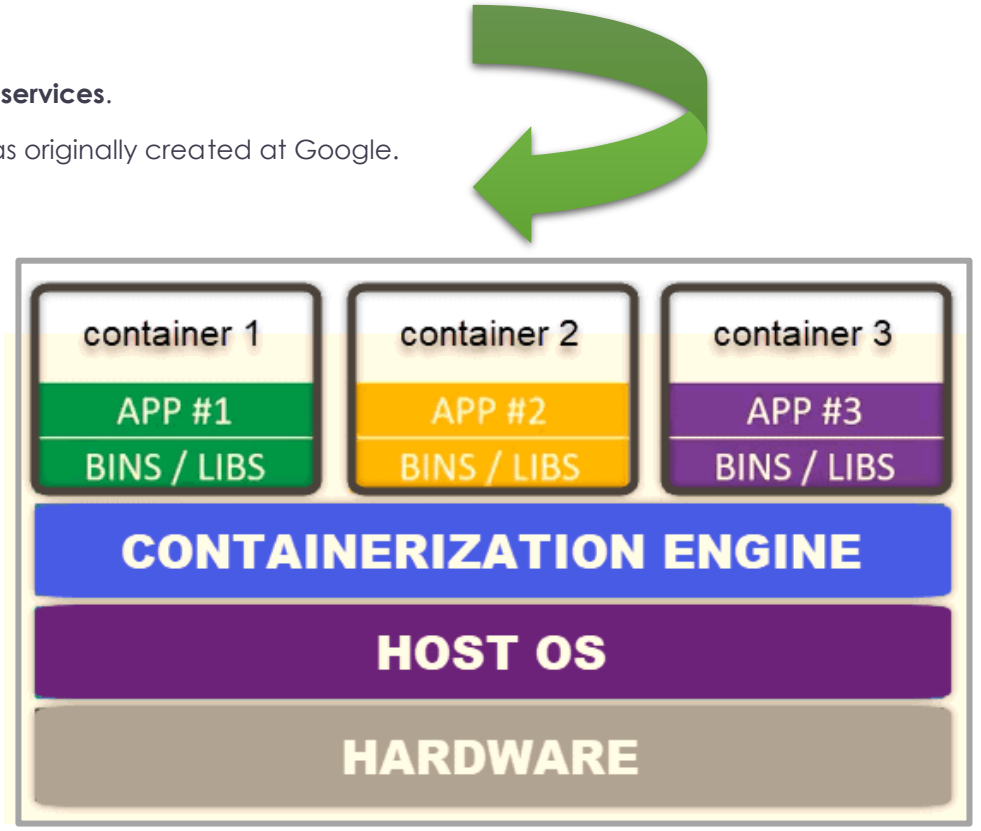
How to define Kubernetes, with Developer Perspective?

Kubernetes, or **k8s** for short, is a system for automating application deployment. Modern applications or distributed system are dispersed across clouds, virtual machines, and servers. Administering apps manually is no longer a viable option.

K8s transforms virtual and physical machines into a unified API surface. A developer can then use the Kubernetes API to deploy, scale, and manage containerized applications.

Kubernetes architecture supports a flexible framework for distributed systems. K8s automatically orchestrates scaling and failovers for your applications and provides deployment patterns.

It helps manage containers that run the applications and ensures there is no downtime in a production environment. For example, if a container goes down, another container automatically takes its place without the end-user ever noticing.



Orchestration - High Level View

Kubernetes : Orchestration working 1/5

What is Container Orchestration?

A container orchestration tool, such as Kubernetes, automates container management in a constantly shifting and chaotic environment.

To fully understand its role, we shall dive deep into the complexity of container environments.

Containers are small virtual environments with individual memory, system files, and processing space. This is lighter than traditional virtual machine.

Kubernetes Architecture and Components

Kubernetes has a decentralized architecture that does not handle tasks sequentially. It functions based on a declarative model and implements the concept of a '*desired state*.'

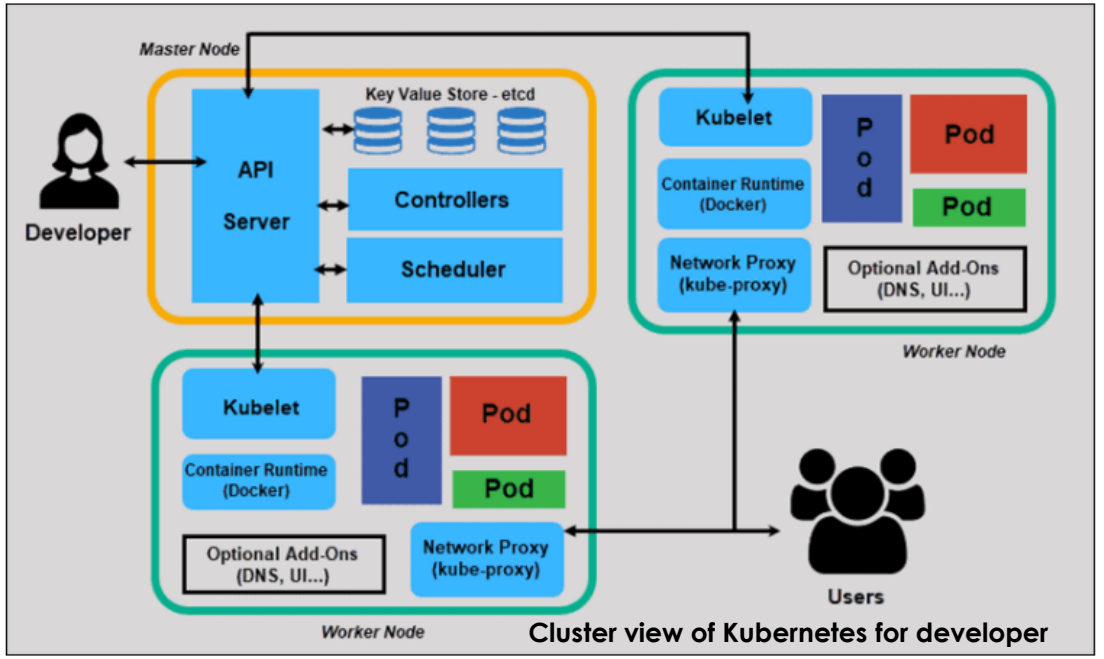
Steps to illustrate the basic Kubernetes process:

1. An administrator creates and places the desired state of an application into a **manifest file**.
2. The file is provided to the Kubernetes API Server using a CLI or UI. Kubernetes' default command-line tool is called ***kubectl***. In case you need a comprehensive list of kubectl commands, check out this https://phoenixnap.com/kb/kubectl-commands-cheat-sheet
3. Kubernetes stores the file (an application's desired state) in a database called the **Key-Value Store (etcd)**.
4. Kubernetes then implements the desired state on all the relevant applications within the cluster.
5. To make sure the current state of the application does not vary from the desired state. For monitoring the state and to understand the nitty gritty , please follow the link Monitoring Kubernetes With Prometheus: Made Simple (phoenixnap.com)

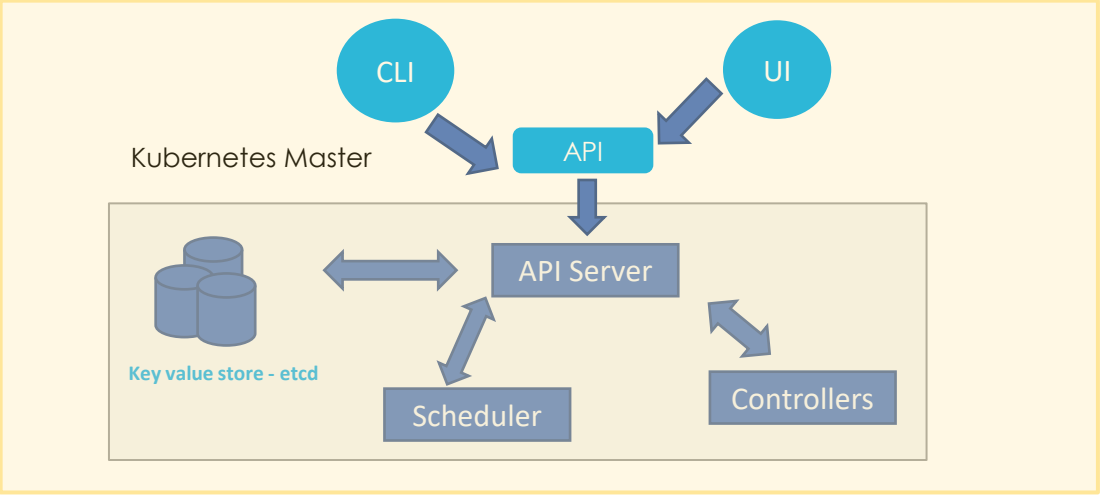
Master Node

The Kubernetes Master (Master Node) receives input from a CLI (Command-Line Interface) or UI (User Interface) via an API. These are the commands you provide to Kubernetes.

You define pods, replica sets, and services that you want Kubernetes to maintain. For example, which container image to use, which ports to expose, and how many pod replicas to run.



Cluster view of Kubernetes for developer



Kubernetes : Orchestration working 2/5

Master Node (contd. From previous page)

You also provide the parameters of the desired state for the application(s) running in that cluster. Which is used in manifest file and monitoring service will monitor if your application state differ from desired state.

Let's define the individual blocks inside Kubernetes Master Diagram.

API Server

The API Server is the front-end of the control plane and the only component in the control plane that we interact with directly. Internal system components, as well as external user components, all communicate via the same API Server.

Key Value Store (etcd)

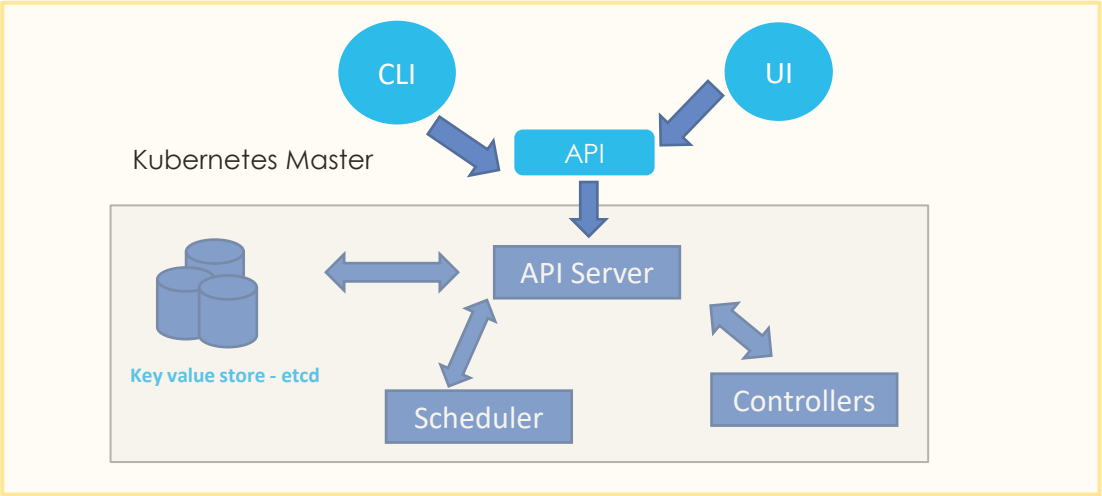
The Key-Value Store, also called etcd, is a database Kubernetes uses to back-up all cluster data. It stores the entire configuration and state of the cluster. The Master node queries etcd to retrieve parameters for the state of the nodes, pods, and containers.

Controller

The role of the Controller is to obtain the desired state from the API Server. It checks the current state of the nodes it is tasked to control, and determines if there are any differences, and resolves them, if any.

Scheduler
A Scheduler watches for new requests coming from the API Server and assigns them to healthy nodes.
It ranks the quality of the nodes and deploys pods to the best-suited node. If there are no suitable nodes, the pods are put in a pending state until such a node appears.

@Note: It is considered good Kubernetes practice not to run user applications on a Master node. This setup allows the Kubernetes Master to concentrate entirely on managing the cluster.



Kubernetes master - diagram

Worker Node
What is Worker Node in Kubernetes Architecture?

Worker nodes listen to the API Server for new work assignments. They execute the work assignments and then report the results back to the Kubernetes Master node.

Contd… to next page

Kubernetes : Orchestration working 3/5

Worker Node (contd. From previous page)

Kubelet
The Kubelet runs on every node of the cluster.
It is the principal Kubernetes agent.
Once you installed Kubelet on the node, CPU, RAM and storage became part of the broader cluster.

Another task node with Kubelet performing is to take the command from API server and execute. After completion return the result to API server.

Monitor the pod and return to the controller if any pod is not fully functional.

Container Runtime
The container runtime pulls images from a container image registry and starts and stops containers.
A **3rd party** software or plugin, **such as Docker**, usually performs this function.

Kube-proxy
The primary role of this component is to make sure that each node gets it's IP address, Implements local iptables and rules to handle routing and traffic load-balancing.

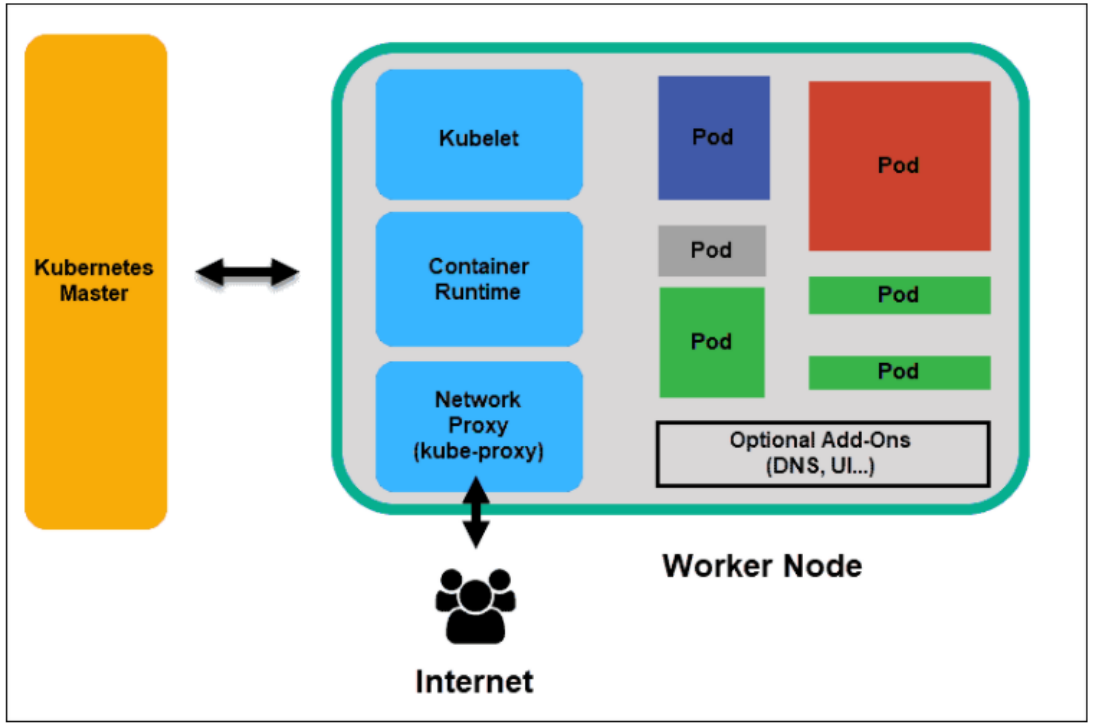
Pod
A pod is the smallest element of scheduling in Kubernetes.

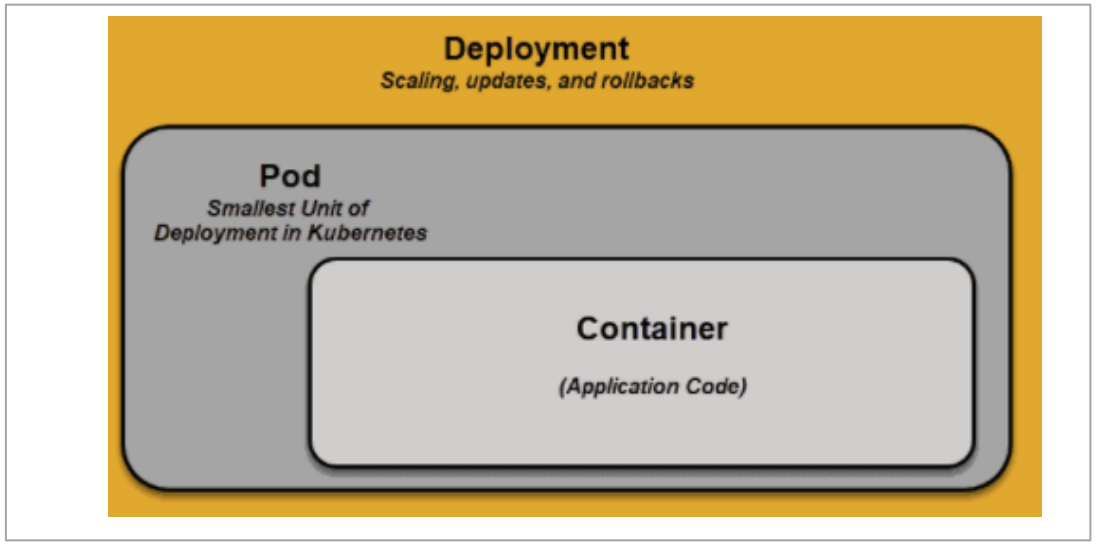
Also, it is mandatory for node to be part of cluster. If you need to scale your app, you can only do so by adding or removing pods.

The pod serves as a 'wrapper' for a single container with the application code.

Based on the availability of resources, the Master schedules the pod on a specific node and coordinates with the container runtime to launch the container.

Thumb Rule , @Note: Kubernetes in principle believing that fixing of a pod runtime condition, better to launch new pod and assign the task to be executed instead of fixing the pod issue.





Kubernetes : Orchestration working 4/5

Kubernetes services

Pods are not constant. One of the best features Kubernetes offers is that non-functioning pods get replaced by new ones automatically.

So, with addition of new pod, one working complexity introduced here. IP address mismatch

By controlling traffic coming and going to the pod, a Kubernetes service provides a stable networking endpoint(using **kube-proxy**) – a fixed IP, DNS, and port. Through a service, any pod can be added or removed without the fear that basic network information would change in any way.

How Do Kubernetes Services Work?
Pods are associated with services through key-value pairs called **labels** and **selectors**. A service automatically discovers a new pod with labels that match the selector.

This process seamlessly adds new pods to the service, and at the same time, removes terminated pods from the cluster.

As an example, if the desired state includes **three replicas of a pod** and a node running **one replica fails**, the current state is reduced to two pods.

Kubernetes observers that the desired state is three pods. It then **schedules one new replica** to take the place of the failed pod and assigns it to another node in the cluster.

The same would apply when updating or scaling the application by adding or removing pods. Once we update the desired state, Kubernetes notices the discrepancy and adds or removes pods to match the **manifest file**.

And for continuous monitoring, The Kubernetes control panel records, implements, and **runs background reconciliation loops** that continuously check to see if the environment matches user-defined requirements.

Kubernetes Deployment

What is Container Deployment?

To fully understand how and what Kubernetes orchestrates, we need to explore the concept of container deployment.

Traditional Deployment

Developers deployed applications on individual physical servers. This type of deployment posed several challenges.

The sharing of physical resources meant that one application could take up most of the processing power, limiting the performance of other applications on the same machine.

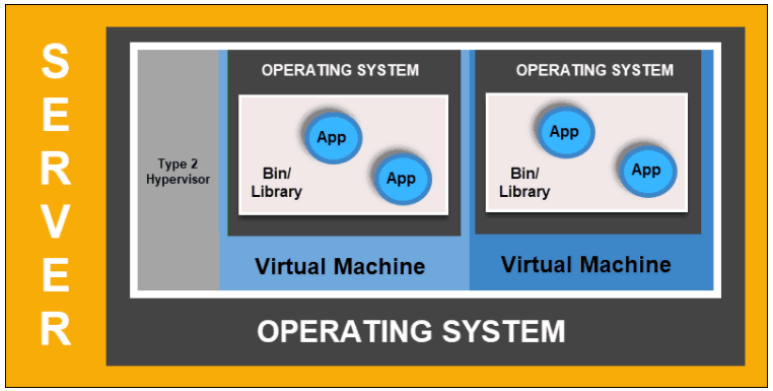
It happens because there is no logical boundary for using the resource on physical system.

So, this poses challenges to the industry and after the solution to this problem came with introduction of virtualization.

Virtualized deployment

Virtualized deployment allows you to create isolated virtual environments, Virtual Machines (VM), on a single physical server.

This solution isolates applications within a VM, limits the use of resources, and increases security.



Kubernetes : Orchestration working 5/5

Virtualized Deployment cont.

Virtualized deployments allow you to scale quickly and spread the resources of a single physical server, update at will, and keep hardware costs in check.

Each VM has its operating system and can run all necessary systems on top of the virtualized hardware.

Container Deployment

Container Deployment is the next step in the drive to create a more flexible and efficient model. Much like VMs, containers have individual memory, system files, and processing space.

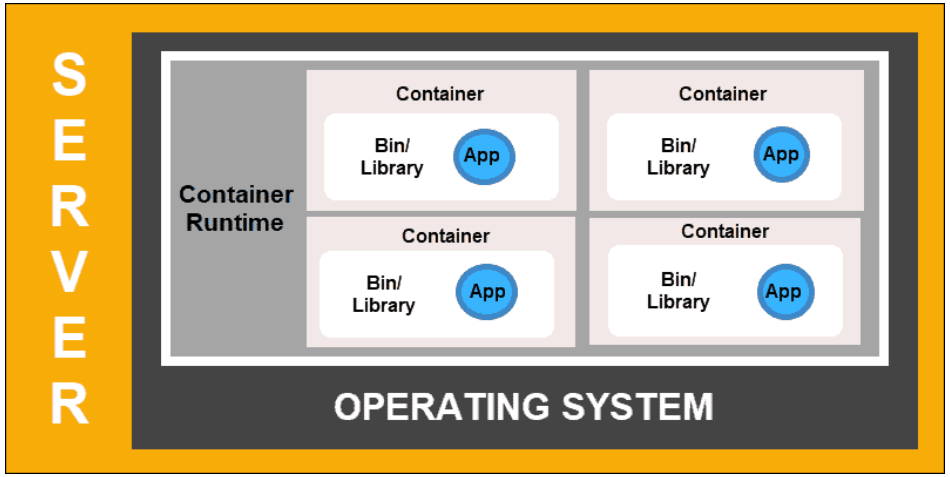
However, strict isolation is no longer a limiting factor.

Multiple applications can now share the same underlying operating system. This feature makes containers much more efficient than full-blown VMs.

Other benefits of this container structure is to manage the application and their dependencies as separate individual units who might be placed on different physical machines in distributed fashion. This is otherwise very difficult to manage.

So, Feature of Kubernetes are listed below

1. Better management of physical resource in cluster by Kubernetes service
2. With better pod management, Kubernetes provide better flexibility in auto management for complex fail-safe scenario of distributed system
3. Kubernetes architecture designed for load balancing and supporting the scalability.
4. Kubernetes have inbuilt service discovery for the microservices running into the container inside the Kubernetes cluster.
5. Kubernetes works on level of container, so it has no influence at the microservice level. It helps them to abstract the language dependency influence between micro-service & Kubernetes container.



An automation solution, such as Kubernetes, is required to effectively manage all the moving parts involved in this process.

I am hoping that now you have better understanding of Kubernetes.

Conclusion

Kubernetes operates using a very simple model. We input how we would like our system to function – Kubernetes compares the desired state to the current state within a cluster. Its service then works to align the two states and achieve and maintain the desired state.

Extend Netflix Example using Kubernetes

The Example with Kubernetes

The example for this chapter can be found at https://github.com/ewolff/microservice . It consists of three microservices:

- ✓ The catalog microservice manages the information about the items. It provides an HTML UI and a REST interface.
- ✓ The customer microservice stores the customer data and provides an HTML UI and a REST interface.
- ✓ The order microservice can receive new orders. It provides an HTML UI and uses the REST interfaces of the catalog and customer microservice.
- ✓ An Apache web server facilitates access to the individual microservices. It forwards the calls to the respective services.

The microservices are accessible from the outside via node ports. On each node in the cluster, a request to a specific port will be forwarded to the service. However, the port numbers are assigned by Kubernetes, so there is no port number in the figure.

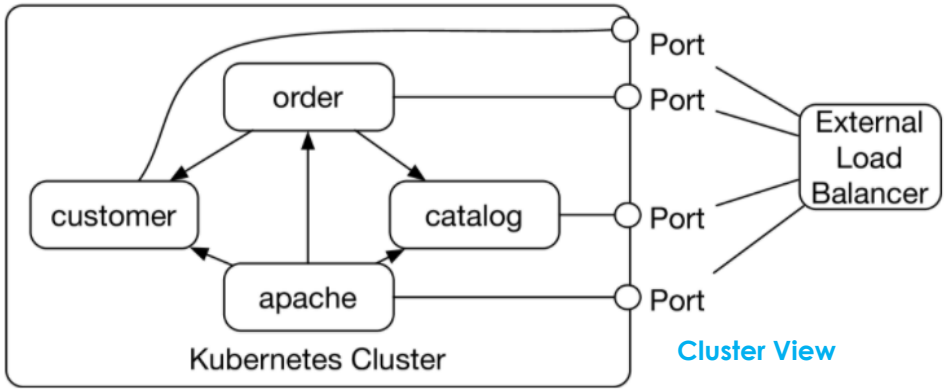
A load balancer is set up by the Kubernetes service to distribute the load across Kubernetes nodes.

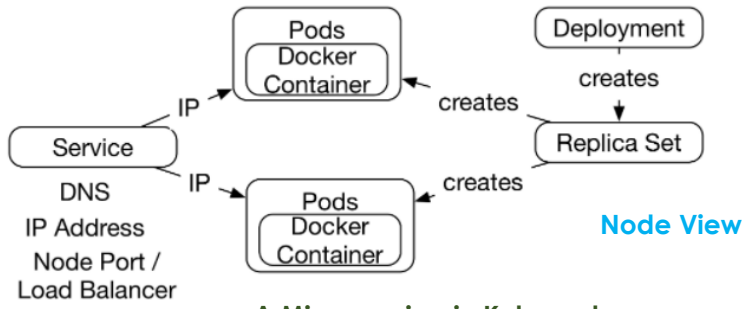
Implementing microservices with Kubernetes
The drawing above shows the interaction of the Kubernetes components for a microservice.
- ✓ A **deployment** creates a **replica set** with the help of Docker images.
- ✓ The **replica set** starts one or multiple pods.
- ✓ The **pods** in the example only comprise a single Docker container in which the microservice is running.

Let's discuss, how service discovery will be performed inside the Kubernetes?

A Service makes the replica set accessible:

- ✓ The service provides the pods with an IP address and a DNS record.
- ✓ Other pods communicate with the service by reading the IP address from the DNS record.
- ✓ Thereby, Kubernetes implements **service discovery with DNS**.





A Microservice in Kubernetes

Fail-safety
The microservices are so fail-safe because the replica set ensures that a certain number of pods is always running. **If a pod fails, a new one is started.**.

Reference Page : Links for further deep dive

https://kubernetes.io/docs/setup/
https://github.com/GoogleCloudPlatform/kubernetes-workshops
https://phoenixnap.com/kb/understanding-kubernetes-architecture-diagrams
https://phoenixnap.com/kb/kubernetes-best-practices