BATCH : Batch 85

LESSON : **Kubernetes-1**

DATE : 07.11.2022

SUBJECT : **Kubernetes Intro and Installation**

techproeducation

techproeducation

techproeducation

techproeducation

techproedu

Kubernetes

Context

- What is k8s?
- Key Concepts
- Architecture
  - Control Plane components
  - Node components
- Installations

Monolith vs Microservices

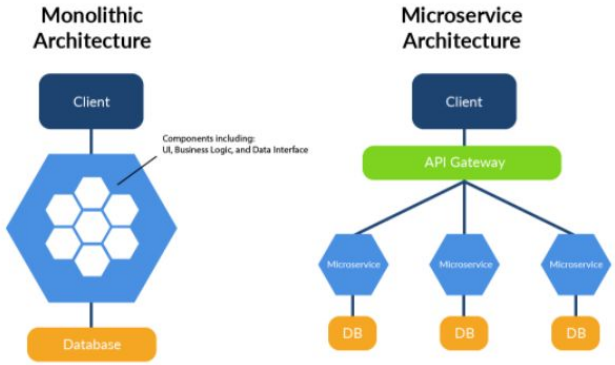
Monolith Architecture

Benefits of Monolithic Architecture

1. Simple to develop.
2. Simple to test.
3. Simple to deploy.
4. Simple to scale horizontally by running multiple copies behind a load balancer.

Drawbacks of Monolithic Architecture

1. This simple approach has a limitation in size and complexity.
2. Application is too large and complex to fully understand and made changes fast and correctly.
3. The size of the application can slow down the start-up time.
4. You must redeploy the entire application on each update.
5. Impact of a change is usually not very well understood which leads to do extensive manual testing.
Continuous deployment is difficult.
6. Monolithic applications can also be difficult to scale when different modules have conflicting resource requirements.
7. Another problem with monolithic applications is reliability. Bug in any module (e.g. memory leak) can potentially bring down the entire process. Moreover, since all instances of the application are identical, that bug will impact the availability of the entire application.
8. Monolithic applications has a barrier to adopting new technologies. Since changes in frameworks or languages will affect an entire application it is extremely expensive in both time and cost.



Microservices Architecture

Benefits of Microservices Architecture

1. It tackles the problem of complexity by decomposing application into a set of manageable services which are much faster to develop, and much easier to understand and maintain.
2. It enables each service to be developed independently by a team that is focused on that service.
3. It reduces barrier of adopting new technologies since the developers are free to choose whatever technologies make sense for their service and not bounded to the choices made at the start of the project.
4. Microservice architecture enables each microservice to be deployed independently. As a result, it makes continuous deployment possible for complex applications.
5. Microservice architecture enables each service to be scaled independently.

Drawbacks of Microservices Architecture

1. Microservices architecture adding a complexity.
2. Microservices has the partitioned database architecture.
3. Testing a microservices application is also much more complex then in case of monolithic web application.
4. It is more difficult to implement changes that span multiple services.
5.Deploying a microservices-based application is also more complex.

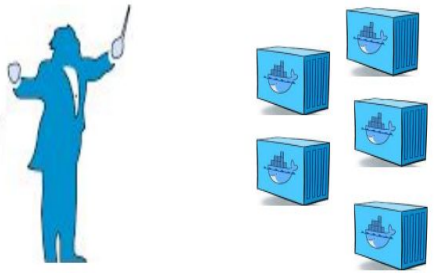
Orchestration

- Containers are great, but when you get lots of them running, at some point, you need them all working together in harmony to solve business problems.
- Tools to manage, scale, and maintain containerized applications are called orchestrators, and the most common example of this is Kubernetes

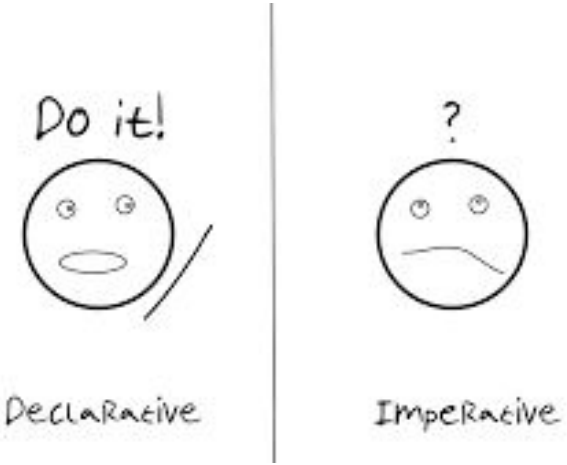
Orchestration

Container orchestration is used to automate the following tasks at scale:

- Provisioning and deployments of containers
- Availability of containers
- Load balancing, traffic routing and service discovery containers
- Health monitoring of containers
- Securing the interactions between containers.
- Configuring and scheduling of containers



Declarative vs Imperative



Imperative	Declarative
Explicit Instructions	Describe the Outcome
The system is stupid, you are smart	The system is smart, you don't care

What Does “Kubernetes” Mean?

Greek for “pilot” or
“Helmsman of a ship”

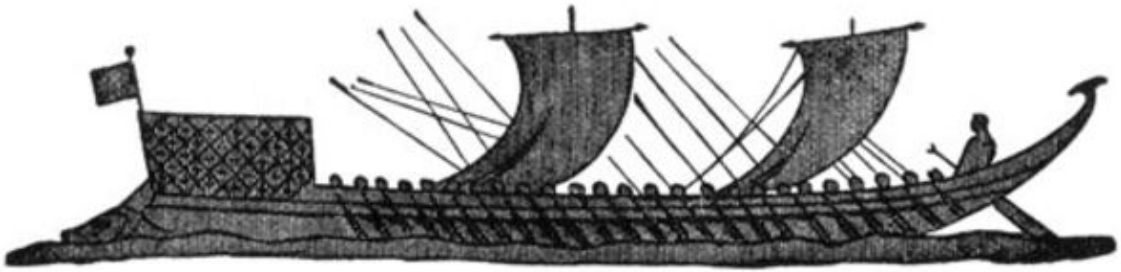


Image Source

What is Kubernetes?

- Kubernetes is Open Source Orchestration system for Containerized Applications.
- Kubernetes is a platform that eliminates the manual processes involved in deploying containerized applications.
- Kubernetes used to manage the State of Containers.
  - Start Containers on Specific Nodes.
  - Restart Containers when gets Killed.
  - Move containers from one Node to Another.

Why you need Kubernetes?

- Containers are a perfect way to get the applications packaged and run. In a production environment, you should manage the containers that run the applications and ensure no downtime.

- Kubernetes supplies you with:

  Service discovery and load balancing

  Storage orchestration

  Automated rollouts and rollbacks

  Automatic bin packing

  Self-healing

  Secret and configuration management

What can K8s really do?

- Autoscale Workloads
- Blue/Green deployments
- Fire off jobs and scheduled cronjobs
- Manage stateless and stateful applications
- Provide native methods of service discovery
- Easily integrate and support 3rd party apps

Alternatives

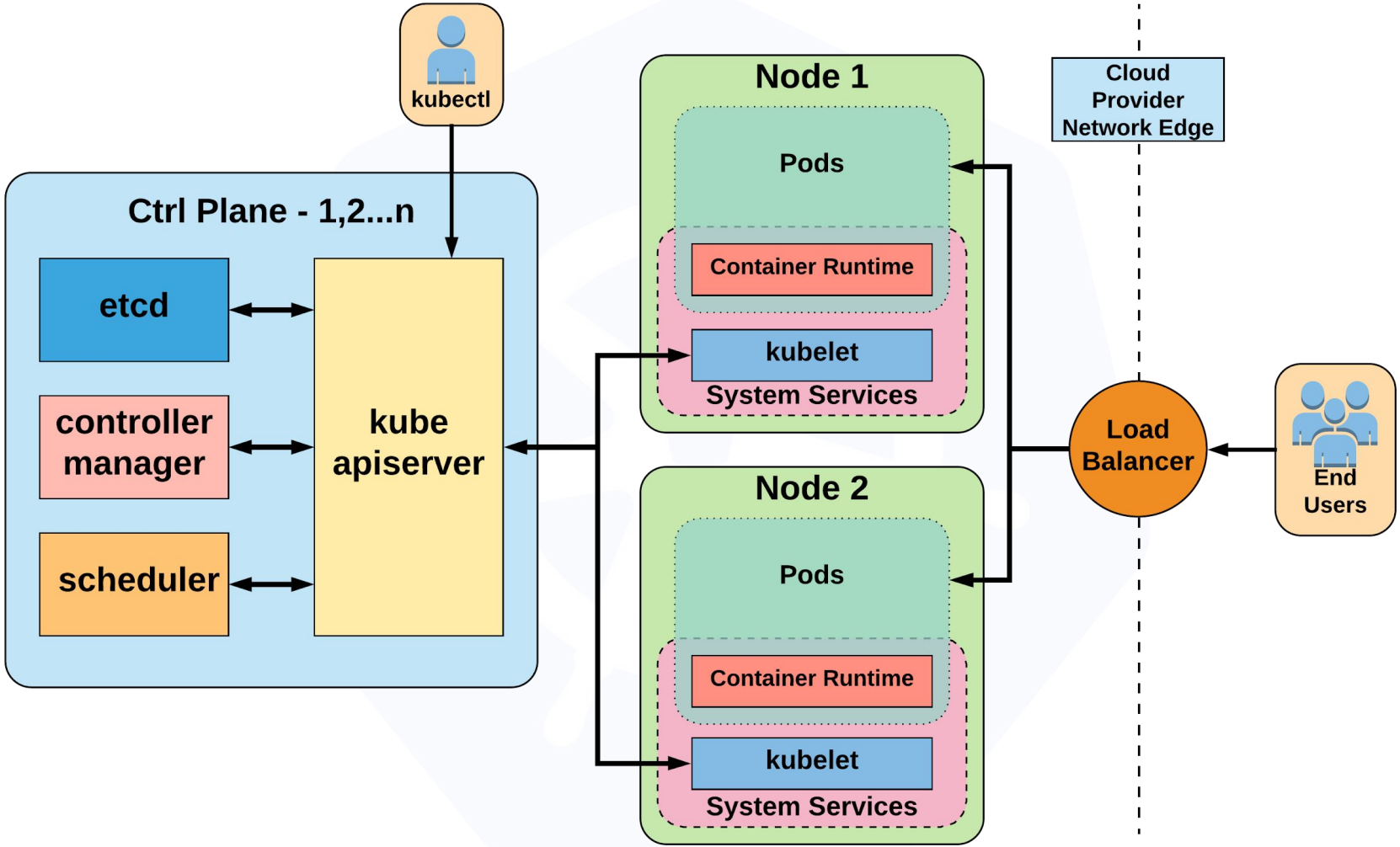
Kubernetes is the most used container orchestration tool, what about others?

Who manages Kubernetes?

CLOUD NATIVE
COMPUTING FOUNDATION

The CNCF is a child entity of the Linux Foundation and operates as a vendor neutral governance group.


Architecture
Overview

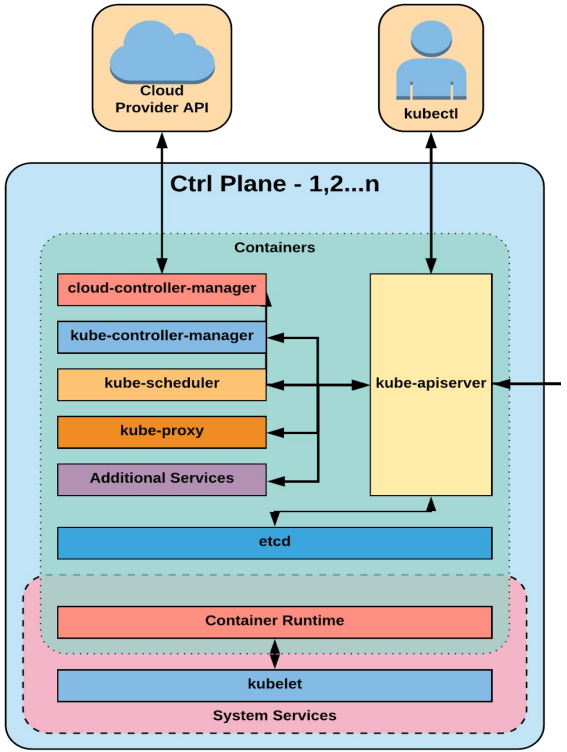

kubectl
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
Node 1
Pods
Container Runtime
kubelet
System Services
Node 2
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

Control Plane Components

Architecture Overview

Control Plane Components

- kube-apiserver
- etcd
- kube-controller-manager
- kube-scheduler



kube-apiserver

- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes strictly through the API Server.
- Acts as the gatekeeper to the cluster by handling authentication and authorization, request validation, mutation, and admission control in addition to being the front-end to the backing datastore.

etcd

- etcd acts as the cluster datastore.
- Purpose in relation to Kubernetes is to provide a strong, consistent and highly available key-value store for persisting cluster state.
- Stores objects and config information.

etcd

Uses *"Raft Consensus"* among a quorum of systems to create a fault-tolerant consistent *"view"* of the cluster.

https://raft.github.io/

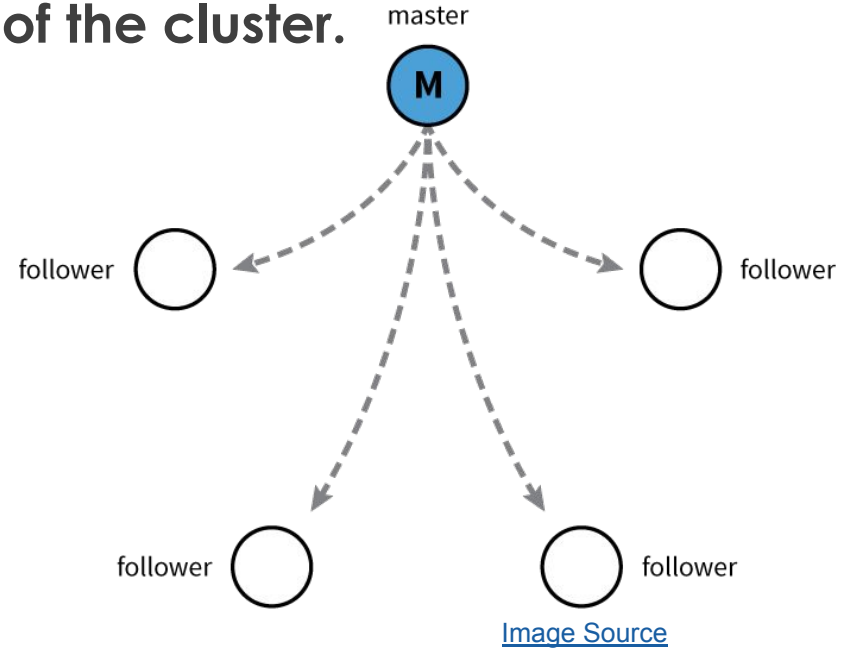


Image Source

kube-controller-manager

- Serves as the primary daemon that manages all core component control loops.
- Monitors the cluster state via the apiserver and steers the cluster towards the desired state.
- List of core controllers: https://github.com/kubernetes/kubernetes/blob/master/cmd/kube-controller-manager/app/controllermanager.go#L344

cloud-controller-manager

- Daemon that provides cloud-provider specific knowledge and integration capability into the core control loop of Kubernetes.
- The controllers include Node, Route, Service, and add an additional controller to handle things such as PersistentVolume Labels.

kube-scheduler

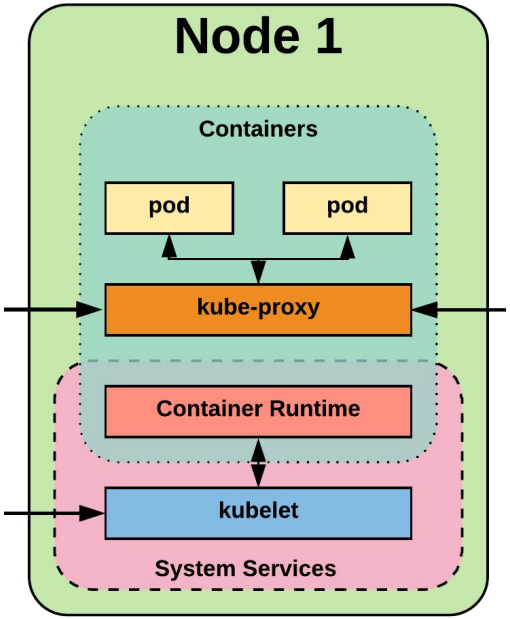
- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Default scheduler uses bin packing.
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.

Node Components

Architecture Overview

Node Components

- kubelet
- kube-proxy
- Container Runtime Engine



kubelet

- Acts as the node agent responsible for managing the lifecycle of every pod on its host.
- Kubelet understands YAML container manifests that it can read from several sources:
  - file path
  - HTTP Endpoint
  - etcd watch acting on any changes
  - HTTP Server mode accepting container manifests over a simple API.

kube-proxy

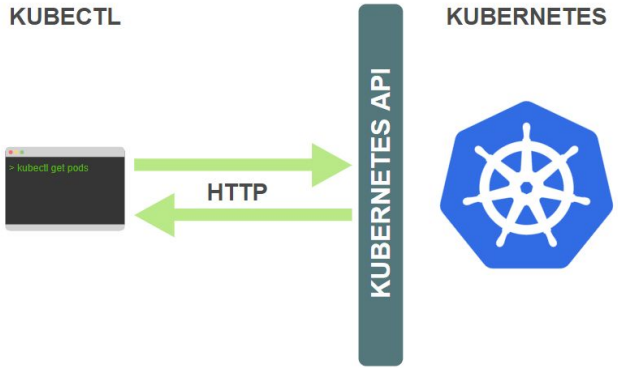
- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.
- Available Proxy Modes:
  - Userspace
  - iptables
  - ipvs (default if supported)

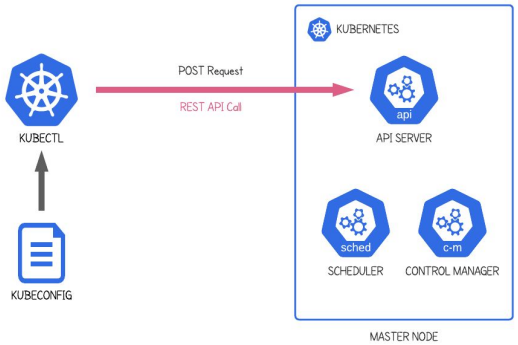
Container Runtime Engine

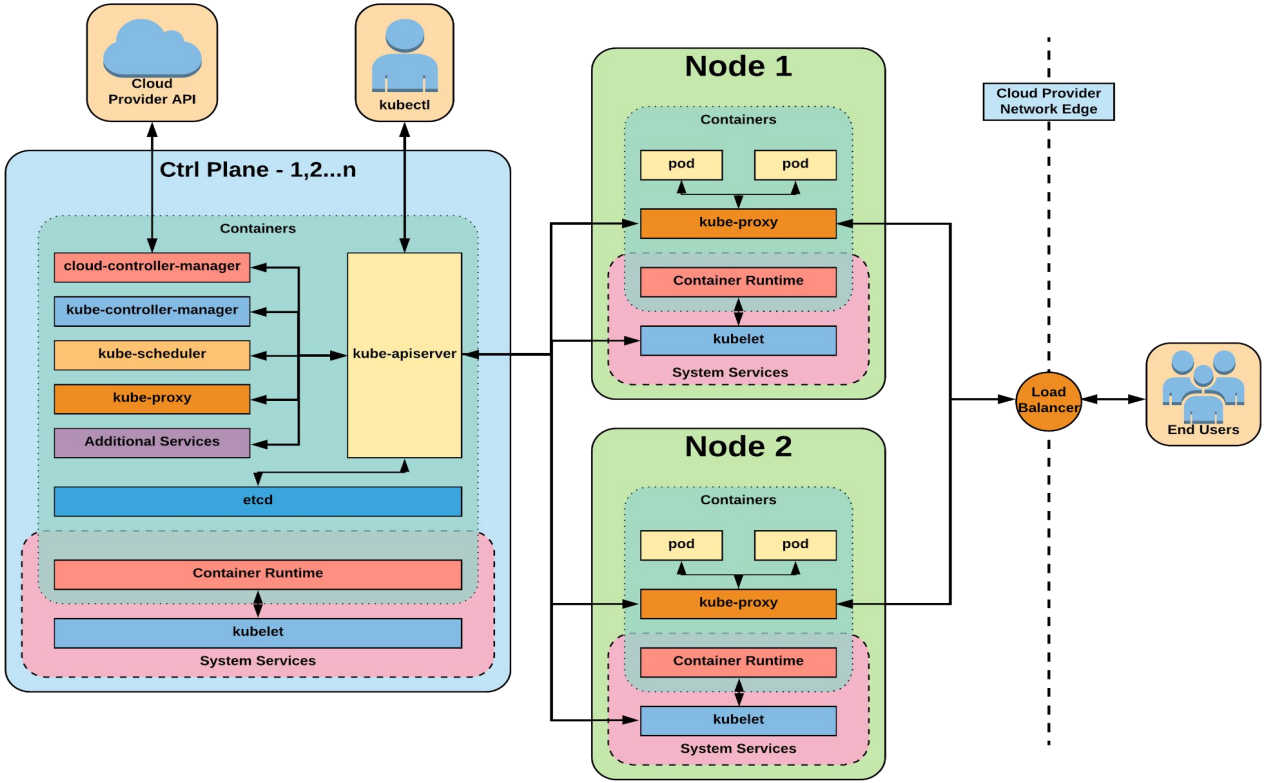
- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
  - Containerd (docker)
  - Cri-o
  - Rkt
  - Kata (formerly clear and hyper)
  - Virtlet (VM CRI compatible runtime)

kubectl

- Kubectl is (almost) the only tool we'll need to talk to Kubernetes
  - It is a rich CLI tool around the Kubernetes API
  - Everything you can do with kubectl, you can do directly with the API
  - [illegible] e C T L", "Cube c



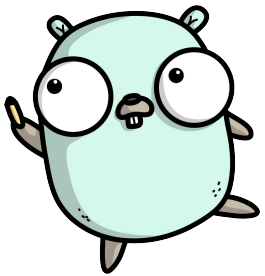


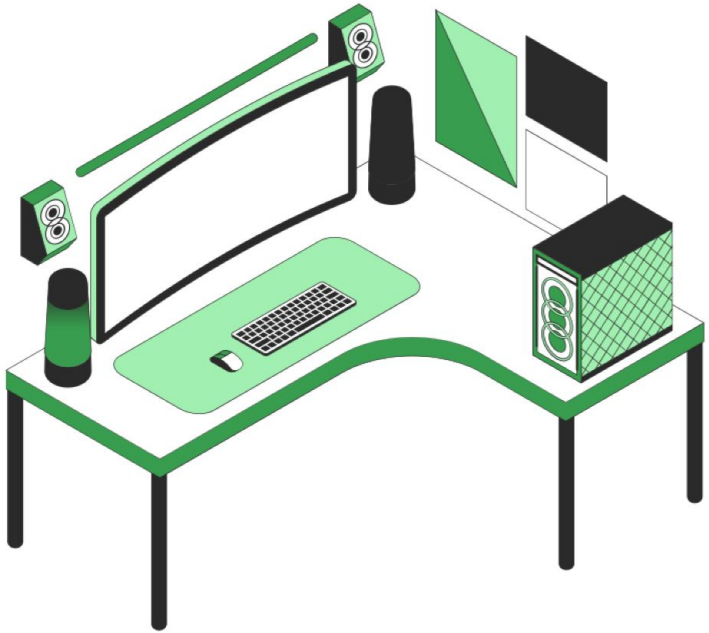

Cloud Provider API
kubectl
Ctrl Plane - 1,2...n
Containers
cloud-controller-manager
kube-controller-manager
kube-scheduler
kube-proxy
Additional Services
kube-apiserver
etcd
Container Runtime
kubelet
System Services
Node 1
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Node 2
Containers
pod
pod
kube-proxy
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

Installation

Requirements:

- Minikube:
  https://github.com/kubernetes/minikube
- Virtualbox*:
  https://www.virtualbox.org/wiki/Downloads
- kubectl:
  https://kubernetes.io/docs/tasks/tools/install-kubectl/
- k8s-intro-tutorials repo:
  https://github.com/mrbobbytables/k8s-intro-tutorials





Do you have any questions?

Send it to us! We hope you learned something new.