

Introduction to containers and kubernetes



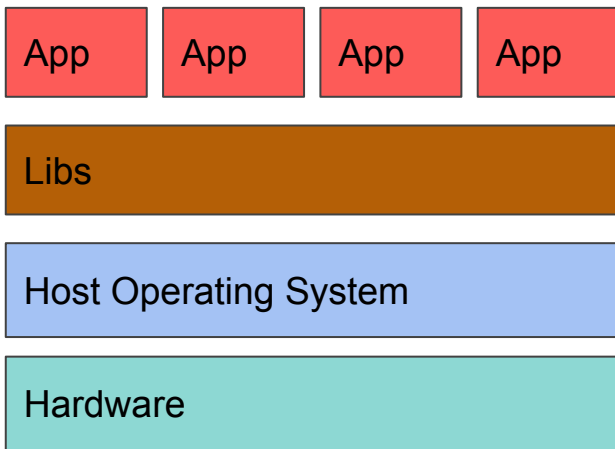


Agenda

- History of application development
- Container orchestration - the motivation
- Kubernetes

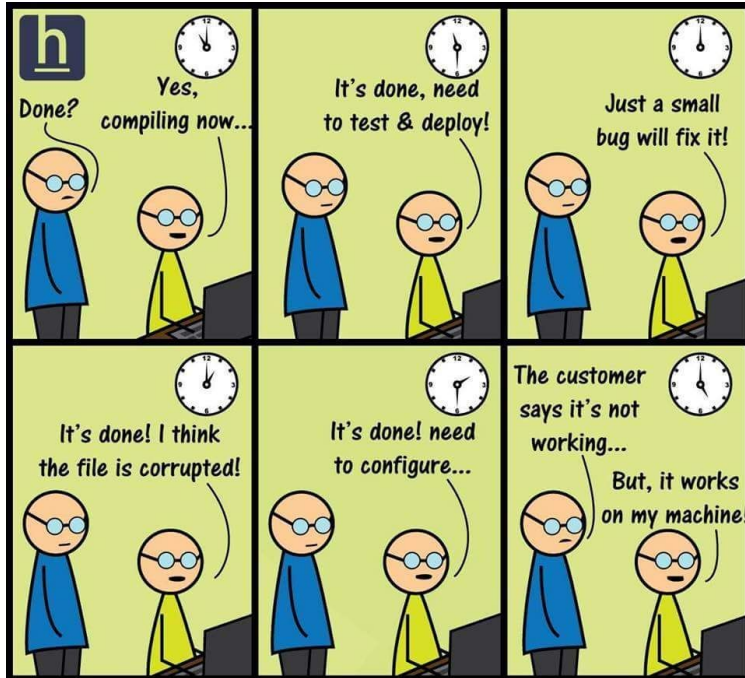


History of application deployment - Traditional Deployment



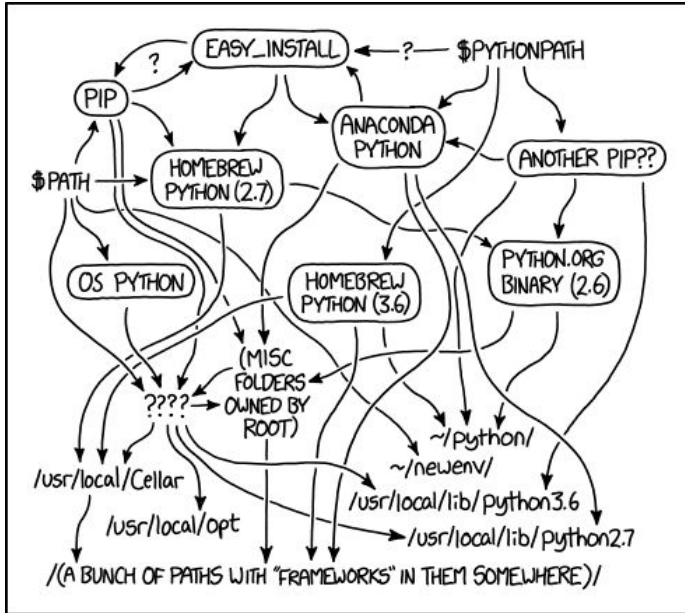
- Monolithic architecture
- Deployment was done using packaging tools (eg InstallShield)
- 3 different environments (development, testing, deployment) means you need to set it up 3 times
- Weeks, months wasted setting up the different environments
- Deployment environment may not be “clean” (dependency issues)

History of application deployment - It works for my machine



- Developer “forget” to include dependencies
- Bugs/defects are not reproducible

History of application development - dependency hell

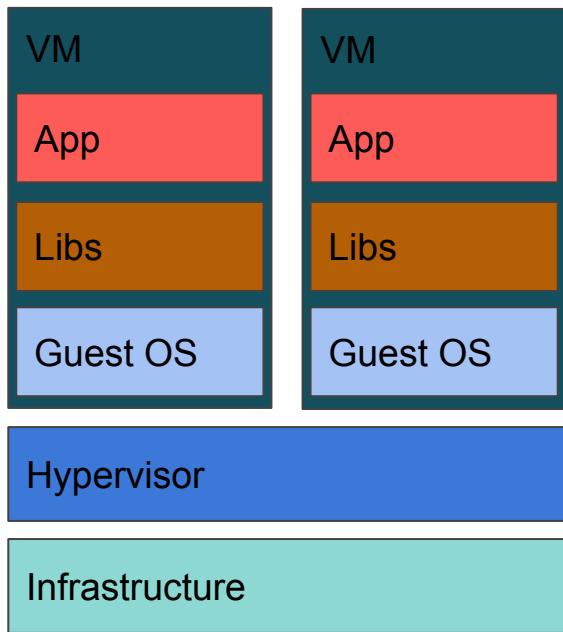


MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

- Different applications require different versions of libraries
- Sometimes these libraries clash!



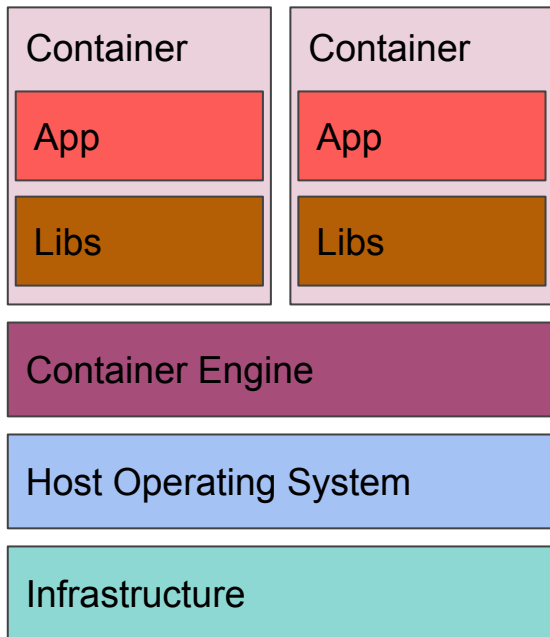
History of application deployment - Virtual Machines



- Virtualize the hardware
- Applications are now isolated from each other
- Time to deploy reduced
- High overhead (each VM is its own OS)



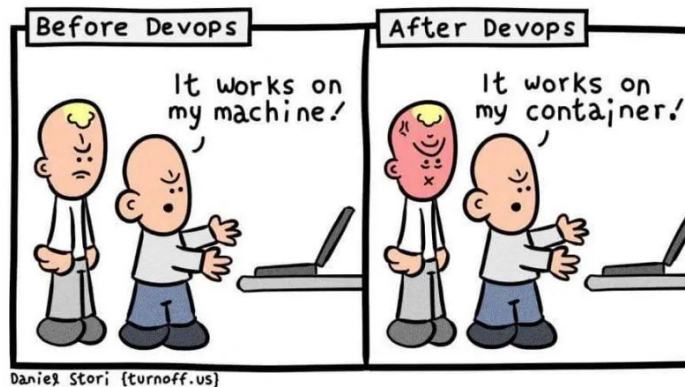
History of application deployment - Container



- Virtualize the OS
- Increased developer velocity
- Run more applications
- Much easier and faster to simulate the production environment
- Development environment and deployment environment are almost identical
- Microservices architecture

History of application deployment - Container

- Things are definitely much better, but applications are getting more complex
- Really much simpler for developers to create dependencies for development and testing





Container Orchestration motivations

- What happens when the container crashes/stops running?
- You want the different containers to talk to each other?
- When you want to scale across multiple hardware?



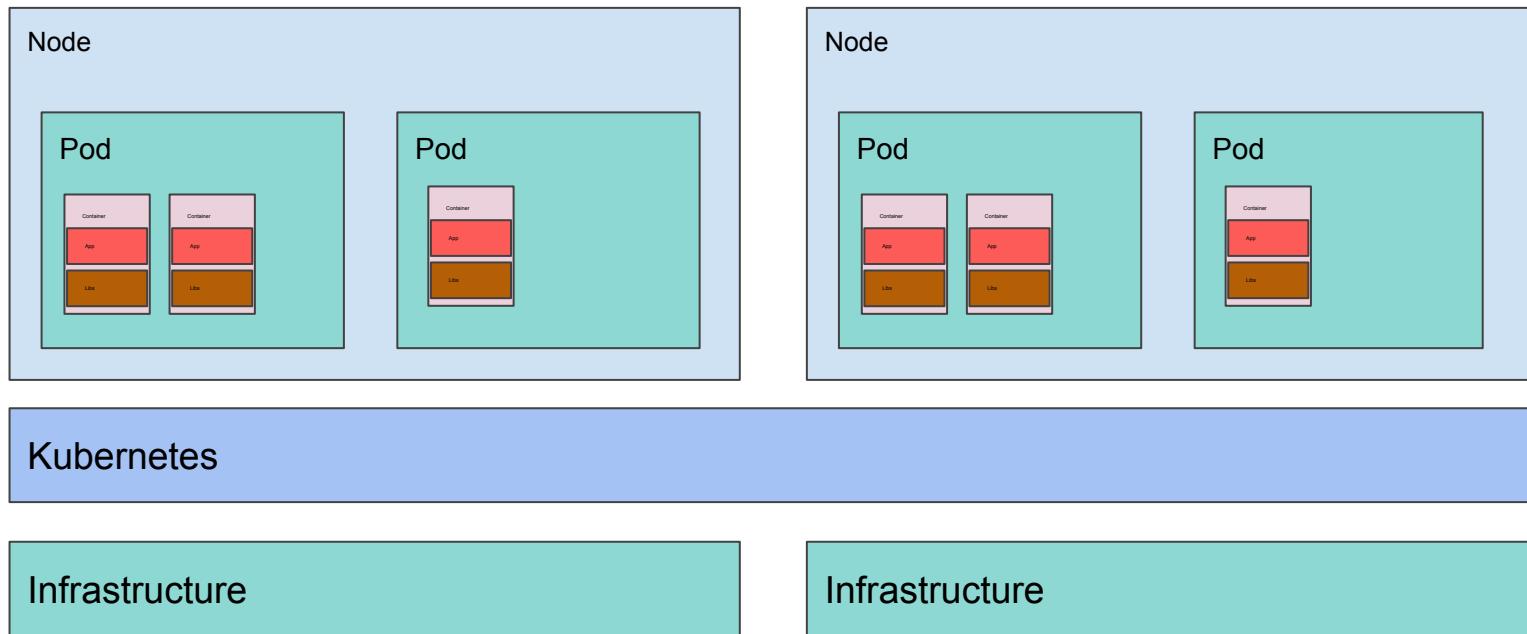


Kubernetes

- Container orchestration
- Experience accumulated by Google on running production workloads
- Automated configuring/coordinating/managing of your containers
- Runs across multiple hardware
- Manage storage
- Manage traffic



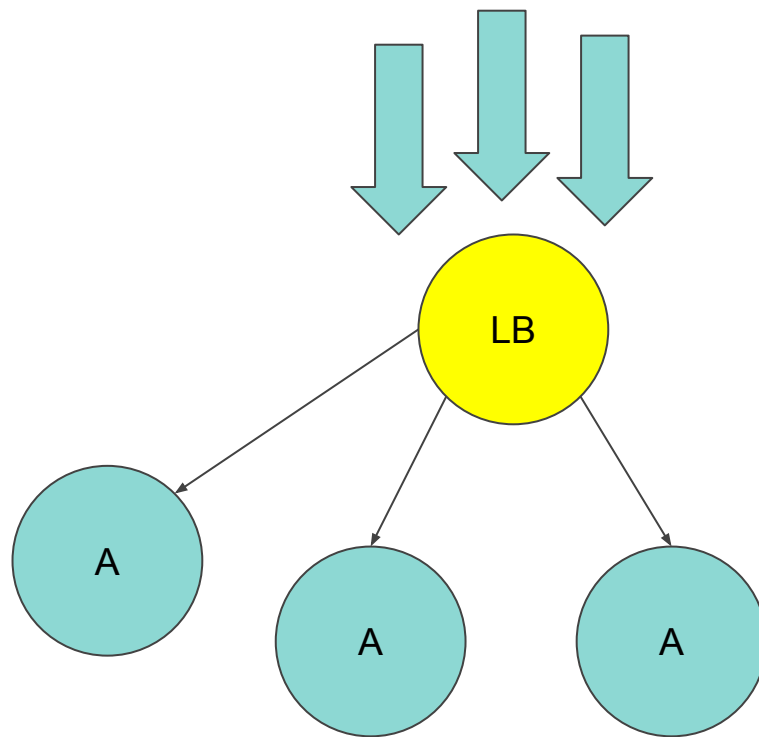
Kubernetes - nodes, pods, containers





Kubernetes

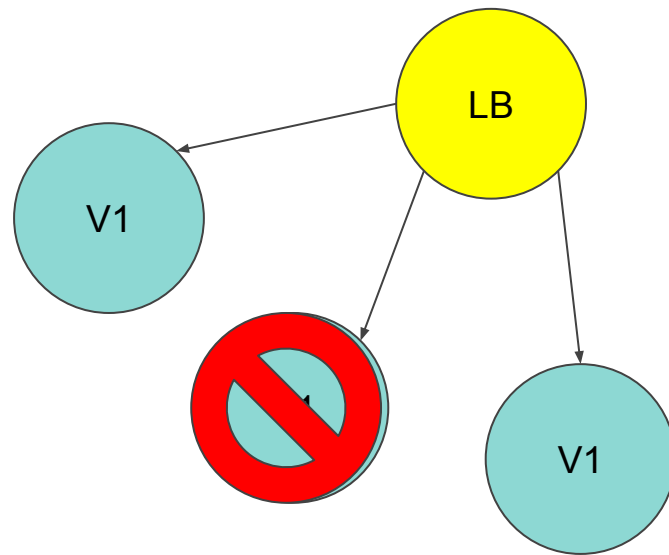
- Scale your applications





Kubernetes

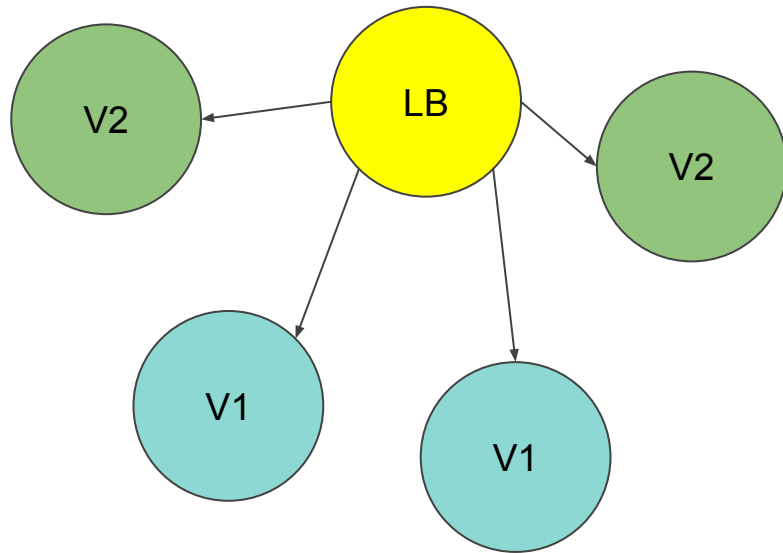
- Replace containers





Kubernetes

- Update your application





End of Presentation

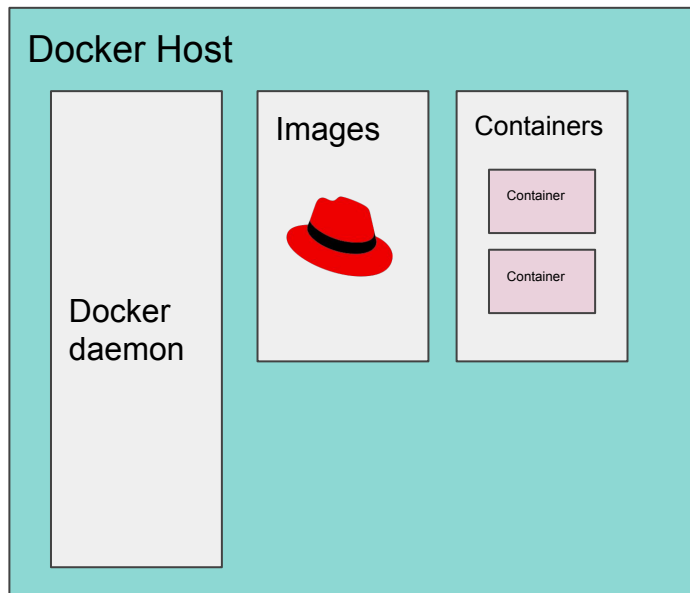
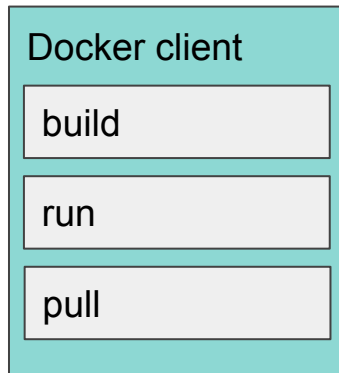


Additional Notes





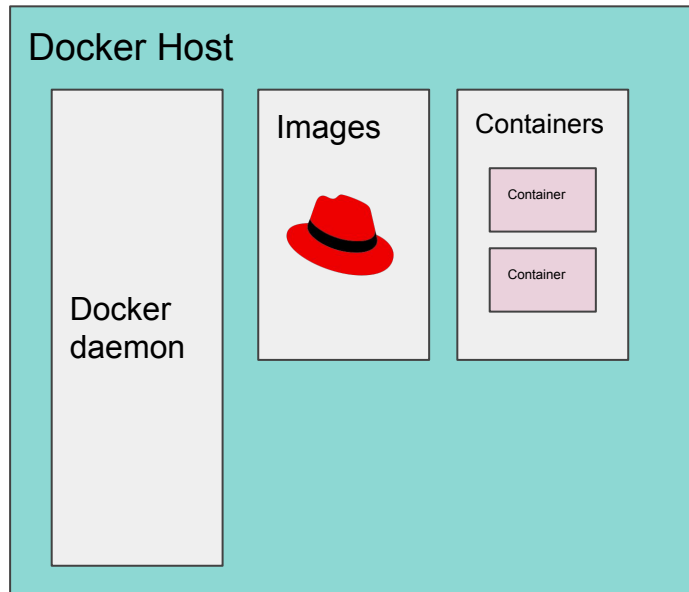
Container - A docker example





Container - A docker example

- Docker host
- Docker daemon - listens to API requests and manages images, containers, networks, volumes on the docker host
- Images - package of binaries and dependencies
- Containers - instance of images running





Container - A docker example

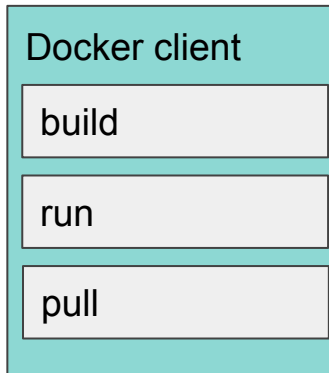
- Docker registry
- A repository of all images





Container - A docker example

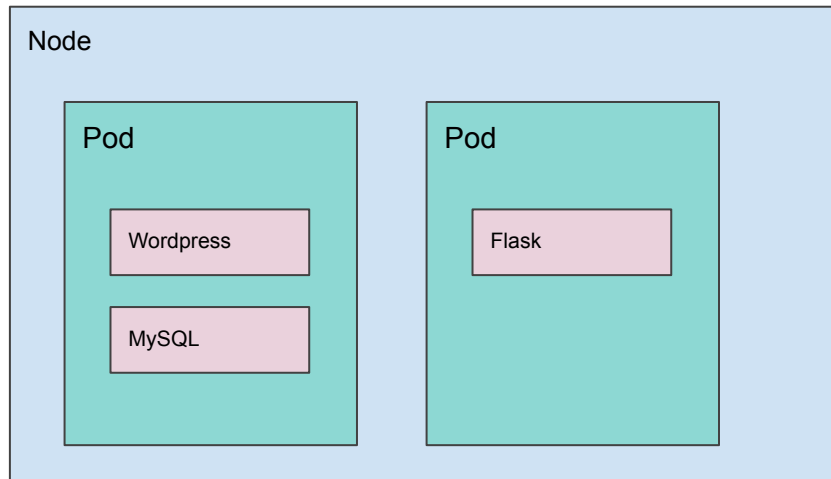
- Build - build/create a docker image
- Run - creates a running instance of a particular image
- Pull - pulls a docker image from the docker repository





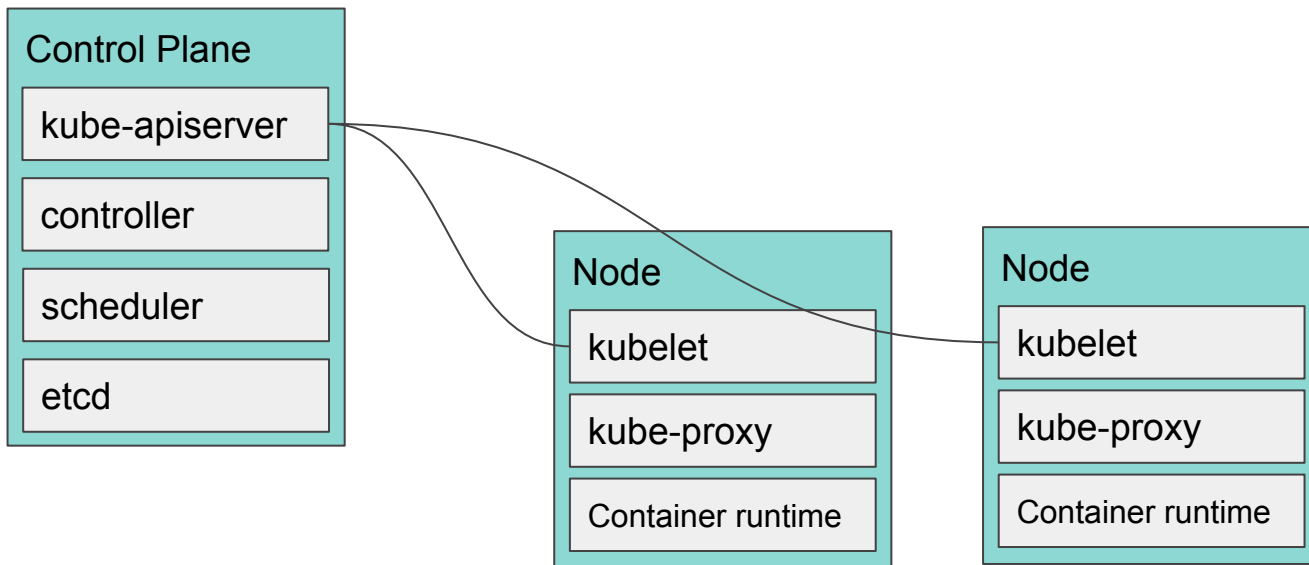
Kubernetes - nodes, pods, containers

- Node - one particular machine, contains one or more pods
- Pod - single instance of an app, containing 1 or more containers





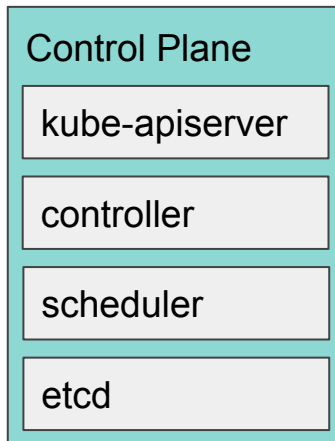
Kubernetes cluster





Kubernetes - control plane

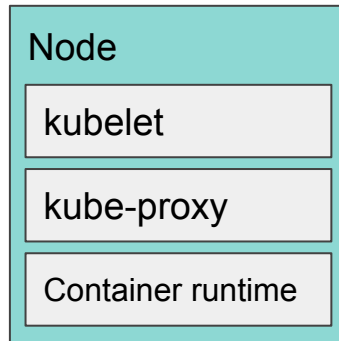
- Manages worker nodes and pods in the cluster
- Kube-apiserver exposes a rest API for clients to communicate commands
- Controller monitors the state of the objects in the cluster, and takes action to ensure the cluster stays in the wanted state
- Scheduler monitors and assigns containers to nodes





Kubernetes - node

- Kubelet - agent running on the node, that monitors request from the API server, and ensures containers are running
- Kube-proxy - network proxy, maintaining network rules on nodes
- Container runtime - software responsible for running containers





Sources

- Overview of kubernetes [<https://kubernetes.io/docs/concepts/overview/>]
- Introduction to docker [<https://learn.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/>]
- Introduction to containers [<https://www.digitalocean.com/community/conceptual-articles/introduction-to-containers>]
- Introduction to containers [<https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=concepts-introduction-containers>]
- Introduction to containers [<https://medium.com/geekculture/introduction-to-containers-basics-of-containerization-bb60503df931>]
- Introduction to containers [<https://www.sentiablog.com/introduction-to-containers>]
- Introduction to kubernetes [<https://learn.microsoft.com/en-us/training/modules/intro-to-kubernetes/>]
- Introduction to kubernetes [<https://www.geeksforgeeks.org/introduction-to-kubernetes-k8s/>]
- Introduction to kubernetes [<https://www.jeremyjordan.me/kubernetes/>]
- Introduction to kubernetes [<https://www.slideshare.net/rishabhindoria52/introduction-to-kubernetes-139878615>]