Kubernetes Introduction

Sandor Guba - Infrastructure Engineer at IBM Budapest Labs

$: whoami

- Infrastructure Engineer at IBM Budapest Labs
- Working with Linux servers more than 10 years
- Main topics are Virtualization and Containerization both in private and public cloud
- And a little bit of networking ;)

What is this presentation about?

- The beginning of containerization decades ago...
- Some basic concepts
- And Kubernetes of course
  - Architecture
  - Quick walkthrough
  - Some basic example
  - Useful links and tools

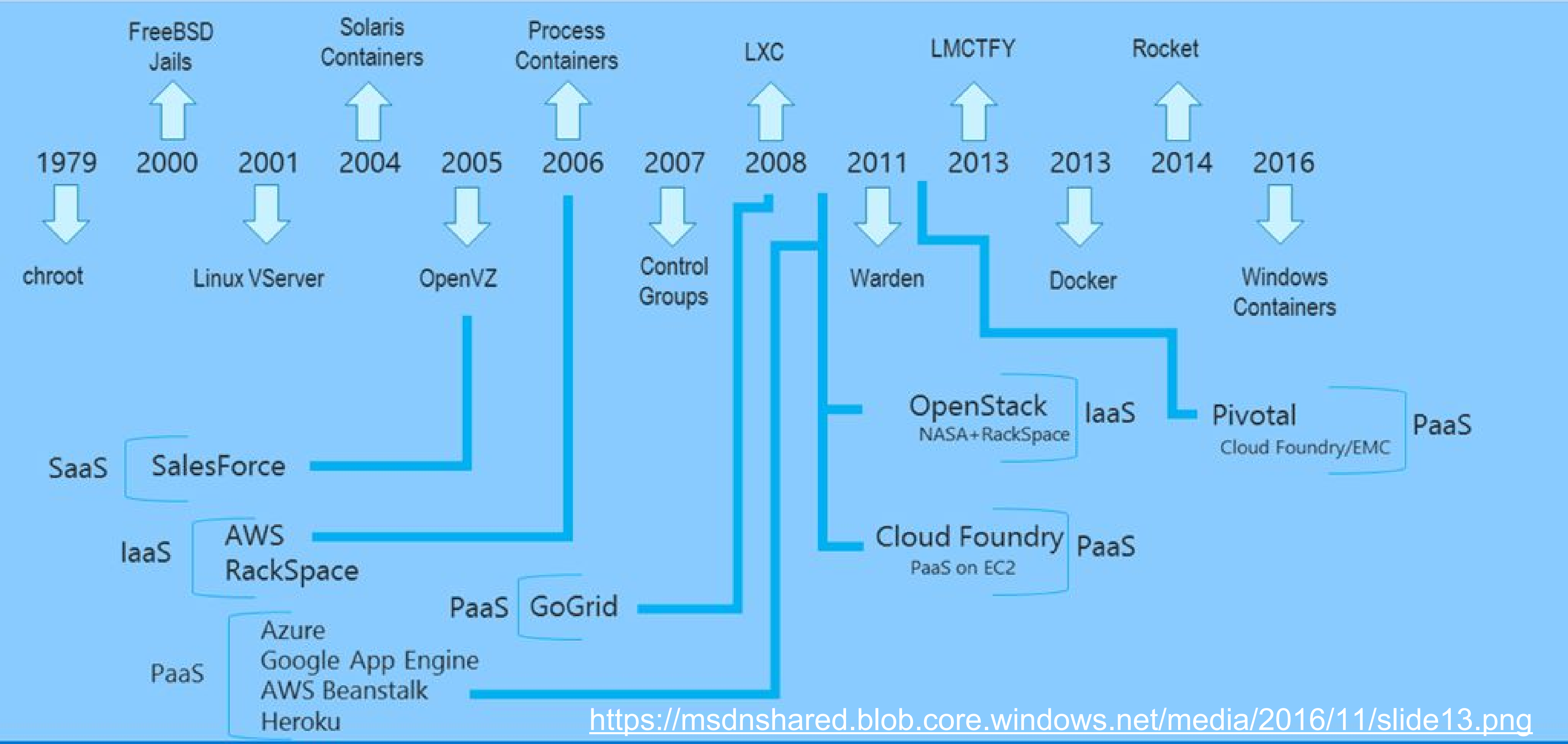
First there were Bare Metal

- At the beginning everything runs on physical server
  - It worked
  - Gets troublesome with the growing number of servers
  - Lots of different tools and unique solutions hard to maintain
- Later with the evolution of virtualization “everything” runs inside a VM
  - Optimized workloads
  - A bit more secure
  - Divide workload into smaller units
  - Operating System is a considerable overhead
- Next step?

Next Step is Containerization

- The goal is a virtual machine like separation
- Without the OS overhead
- Immutable deployment
  - Production like behaviour during development
- Lots of problem to solve (mainly in Linux kernel)

Surprising to most is that containers have been around for decades



1979 chroot

2000 FreeBSD Jails

2001 Linux VServer

2004 Solaris Containers

2005 OpenVZ

2006 Process Containers

2007 Control Groups

2008 LXC

2011 Warden

2013 LMCTFY

2013 Docker

2014 Rocket

2016 Windows Containers

SaaS SalesForce

IaaS AWS RackSpace

PaaS GoGrid

PaaS Azure Google App Engine AWS Beanstalk Heroku

OpenStack NASA+RackSpace IaaS

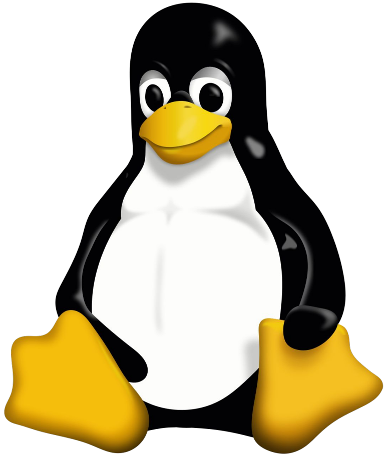
Cloud Foundry PaaS on EC2 PaaS

Pivotal Cloud Foundry/EMC PaaS

https://msdnshared.blob.core.windows.net/media/2016/11/slide13.png

The beginning of the road to Cloud Native

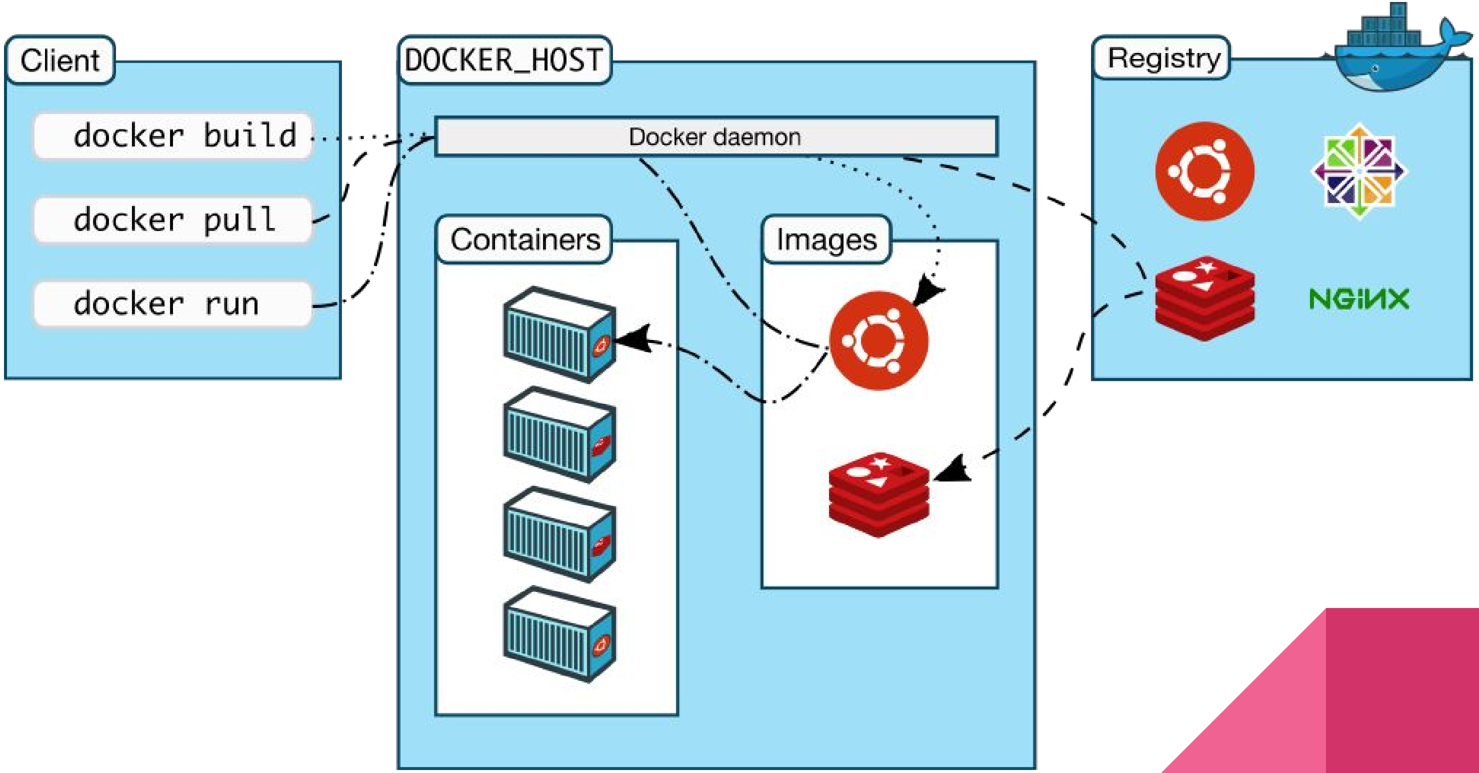
- Namespaces
  - Mnt, pid, net, ipc, hostname, user ids (isolate process environments)
- Cgroups
  - Cpu, memory, disk, i/o (limit resource usages)
- AppArmor, SELinux
  - security/access control
- Seccomp
  - Compute isolation
- Chroot
  - Filesystem isolation



Container runtimes

- LXC, LXD, CGManager, LXCFS - https://linuxcontainers.org/
- Docker - https://www.docker.com/
- Rkt - https://coreos.com/rkt/docs/latest/
- Open Container Initiative - https://www.opencontainers.org/
- Cri-o (OCI implementation) https://github.com/kubernetes-incubator/cri-o

Docker ecosystem



Here we go to Kubernetes

What is Kubernetes?

An Open Source software to automating

- Deployment
- Scaling
- Management

Of Containerized applications.

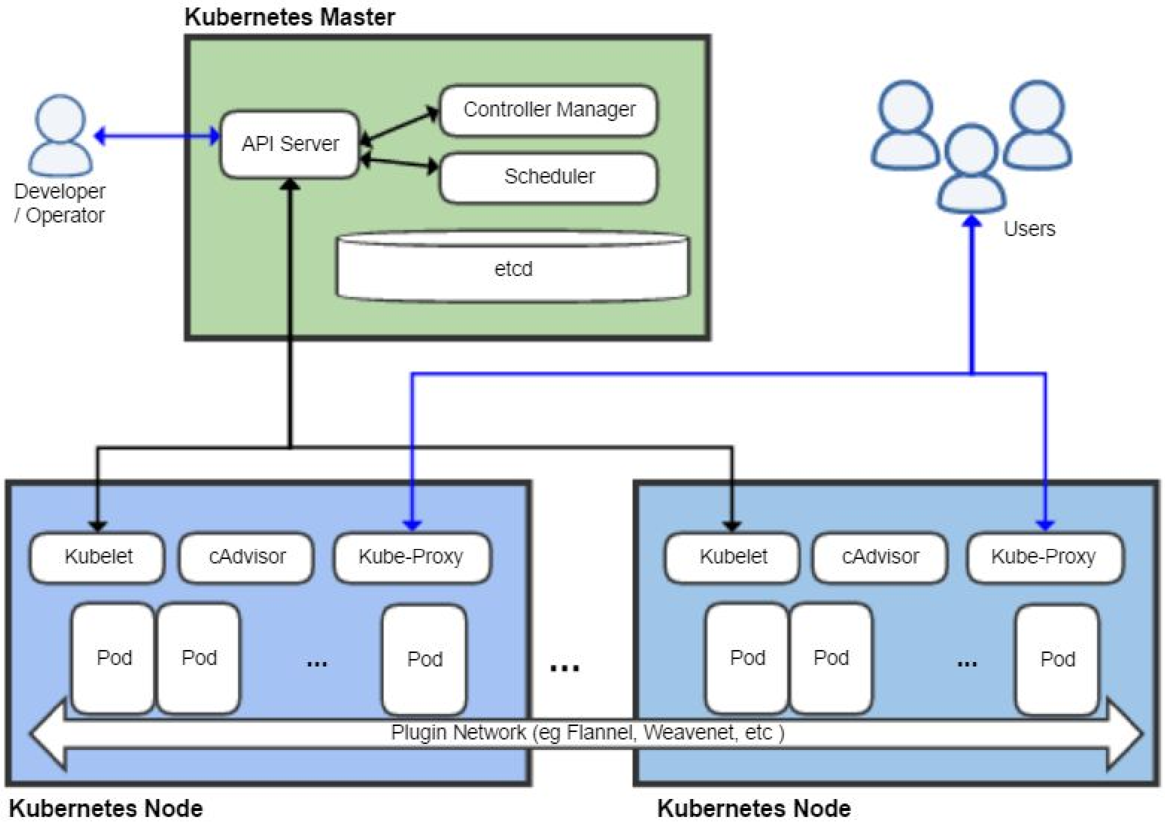
Long story short

- Kubernetes is based on Google Borg
- OpenSource
- Rapidly evolving
  - It was first announced by Google in mid-2014
  - Kubernetes v1.0 was released on July 21, 2015
  - As today (Nov 21, 2017) v1.9 is in progress
- Google partnered with the Linux Foundation to form the Cloud Native Computing Foundation(CNCF)[12] and offered Kubernetes as a seed technology.
- Big Community

Kubernetes community

- Really active open source project
- 29k stars, 1400+ contributors, ~60k commits
- Apache 2 licensed
- Written in Go
- Hosted by the Cloud Native Computing Foundation (CNCF)

Architecture



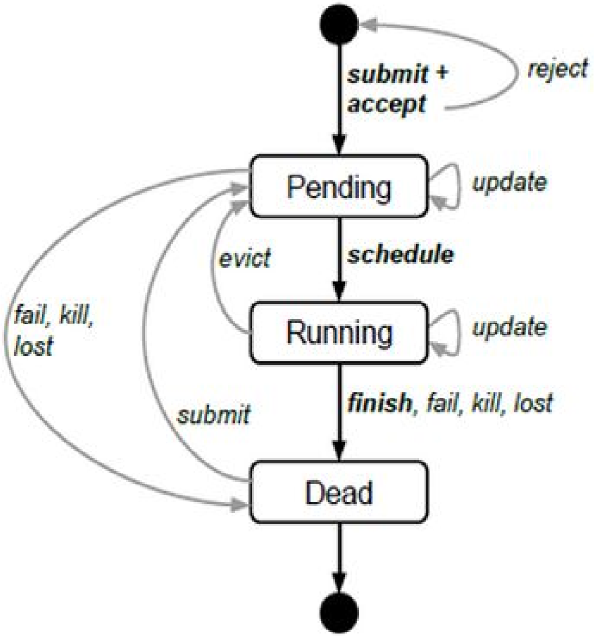
POD

- “Smallest deployable units of a computing”
- It consist 1 or more container
- Containers shares IP address, ports and namespace
    - Localhost
    - Shared memory
- Ephemeral

Lifecycle of a POD

- Create -> Running -> Delete
- Readiness and Liveness probe
  - Exec
  - TCP
  - HTTP
- RestartPolicy
  - Always
  - OnFailure
  - Never

```
livenessProbe:
      httpGet:
        # scheme: HTTPS
        path: /healthz
        port: 8080
        httpHeaders:
        - name: X-Custom-Header
          value: Awesome
      initialDelaySeconds: 15
      timeoutSeconds: 1
    name: liveness
```



Deployment and ReplicaSet

- Declarative representation of Pods and Replica Sets
- Rollout and Rollback different versions
- Manage scaling of Pods

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

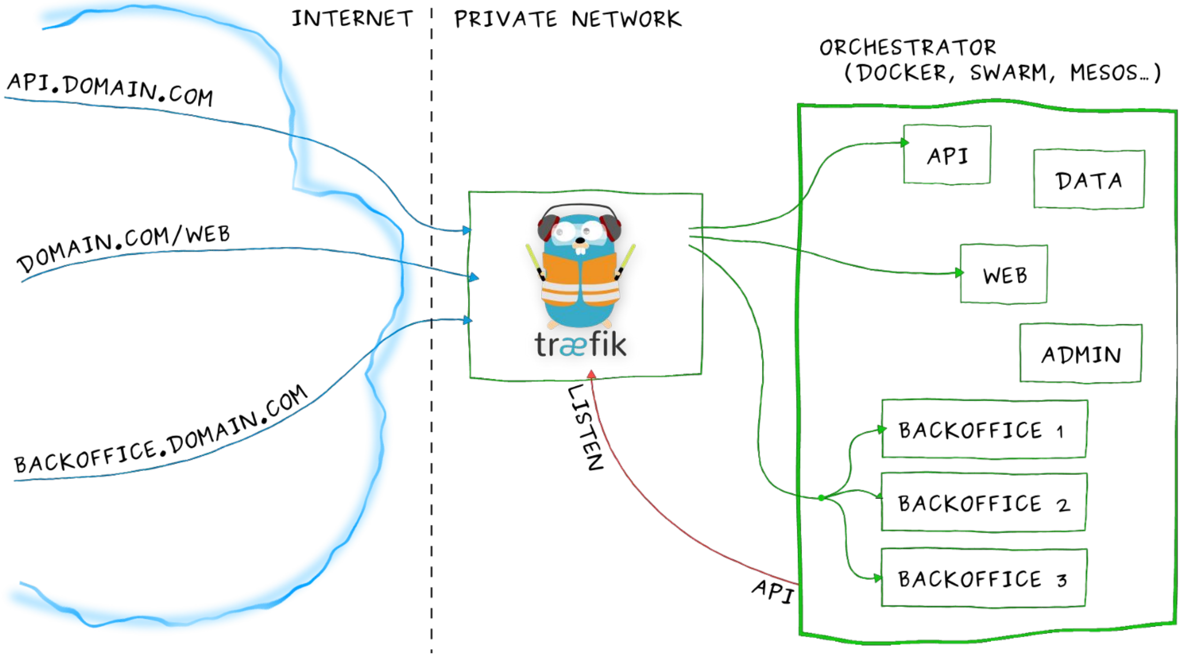
Service

- Abstraction layer that defines a set of Pods
- Service discovery (DNS based)
- Cloud support (AWS, Google, …)
- Multiple mode:
  - NodePort (HostPort)
  - Headless (Direkt POD addresses)
  - LoadBalancer (Internal or External with VIP)

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

Ingress

- L7 Load Balancer
- Routing based on
  - Host
  - Path
  - Custom rules
- Implementations
  - nginx
  - traefik



NGINX

Namespaces

- Virtual clusters called namespaces
- Resources can be separated by namespaces
- Quotas are applied to namespaces
  - Requests cpu/memory (for scheduling)
  - Limits cpu/memory (for hard usage limit SIGKILL)
- Service names separated by namespaces by design
  - `<service-name>.<namespace-name>.svc.cluster.local`

Jobs and Cron Jobs

- As the name tells these are Pods for a specific task
- Or periodic task
- Example
  - ElasticSearch Curator
  - Backup scripts

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    metadata:
      name: pi
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi",
"-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

StatefulSet

- Guaranties the order and uniqueness of a set of Pods
- Fix naming `$(statefulset name)-$(ordinal)`
- Example
  - MySQL cluster
  - Kafka
  - Cassandra

DaemonSet

- Ensure that Pod is running on all or a set of nodes
- Example
  - Fluentd
  - Kube-metrics

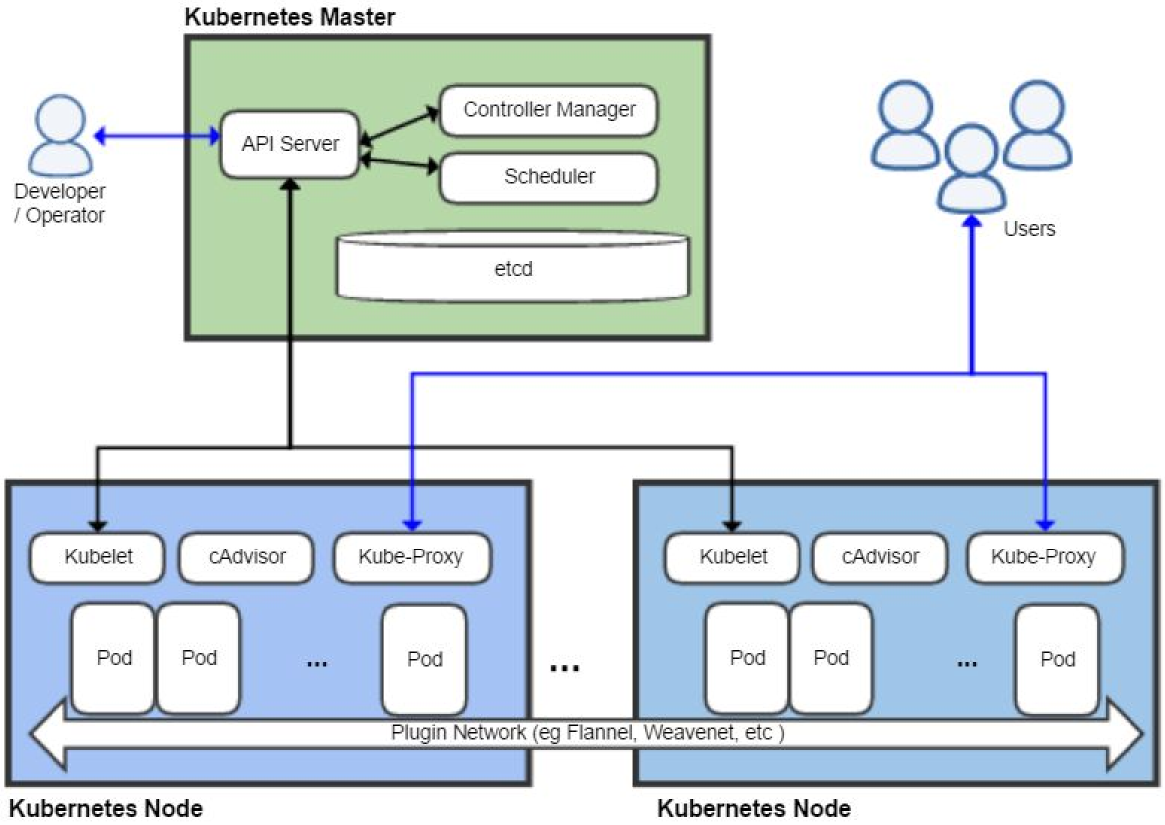
+1 Kubectl and Dashboard

- Your best friend communicating with kubernetes
- 1:1 mapping to the REST API
- Simple, deterministic interface
- Useful commands
- Basic proxy, port-forward functions

Examples

```
$: kubectl get nodes

$: kubectl get pod

$: kubectl get services

$: kubectl create deployment -f nginx.yaml

$: kubectl port-forward <pod_name> 8080:8080

$: kubectl proxy
```

Architecture



High Level Architecture

Kubernetes **Master** Components

- API Server
- Etcd
- Scheduler
- Controller Manager

Kubernetes **Node** Components

- Kubelet
- Kube-proxy

Kubernetes **Concepts**:

- POD
- Deployment
- Service
- Ingress

+1 Kubectl

API server

- REST operations for the resources managed by Kubernetes
- Stateless API backed by etcd

-
  ```
  curl --cert myuser.pem --key myuser-key.pem --cacert /path/to/ca.pem https://apiserver:6443/api/v1/pods
  ```
- API reference https://kubernetes.io/docs/api-reference/v1.8/
- Kubectl command line tool

etcd

- Distributed and reliable key-value store
- Secure with TLS
- Fast (https://coreos.com/blog/performance-of-etcd.html)
- written in Go and uses the Raft consensus algorithm

Scheduler

- Topology aware, workload specific functions
- Schedule based on
  - Requirements
  - Limits
  - Affinity
  - Policy
- Pluggable (custom schedulers)

Controller Manager

- Embeds the core control loops
- Watches the state of the cluster and makes changes
- Handling
    - replication controller
    - endpoints controller
    - namespace controller
    - serviceaccounts controller.
    - +1 cloud provider controller

Kubelet and Kube Proxy

Kubelet

- It is the primary node agent
- Handles POD spec and runs pods

Kube Proxy

- Network proxy that handles TCP, UDP forwarding across a set of backends
- It handles service IP as well (usually backed by DNS)

Networking

- CNI - Container Network Interface (pluggable)
- Calico (native L3)
- Flannel (tunnel)
- Weave (tunnel)
- And there are many many more...

And many many more...

- Role Based Access Control (RBAC)
  - Control what users have access to what objects
- Multiple Schedulers
- Flexible Scheduling Constraints
  - Affinity, anti-affinity, taints, tolerations
- Automatic Cluster Scaling

- K8s publishes signals that allow external services to scale the cluster automatically.
- Cloud Provider Integration
  - GCP, AWS, Azure, OpenStack, vSphere
- Network Policy
- Network ingress policy

Okay where should I start?

Minikube - https://github.com/kubernetes/minikube

Katacoda - https://www.katacoda.com/

Deploying K8s - https://github.com/kelseyhightower/kubernetes-the-hard-way

Introduction to K8s - https://www.edx.org/course/introduction-kubernetes-linuxfoundationx-lfs158x

Kubernetes - https://kubernetes.io/

Helm - https://helm.sh/

Questions?