Introduction to HA Kubernetes

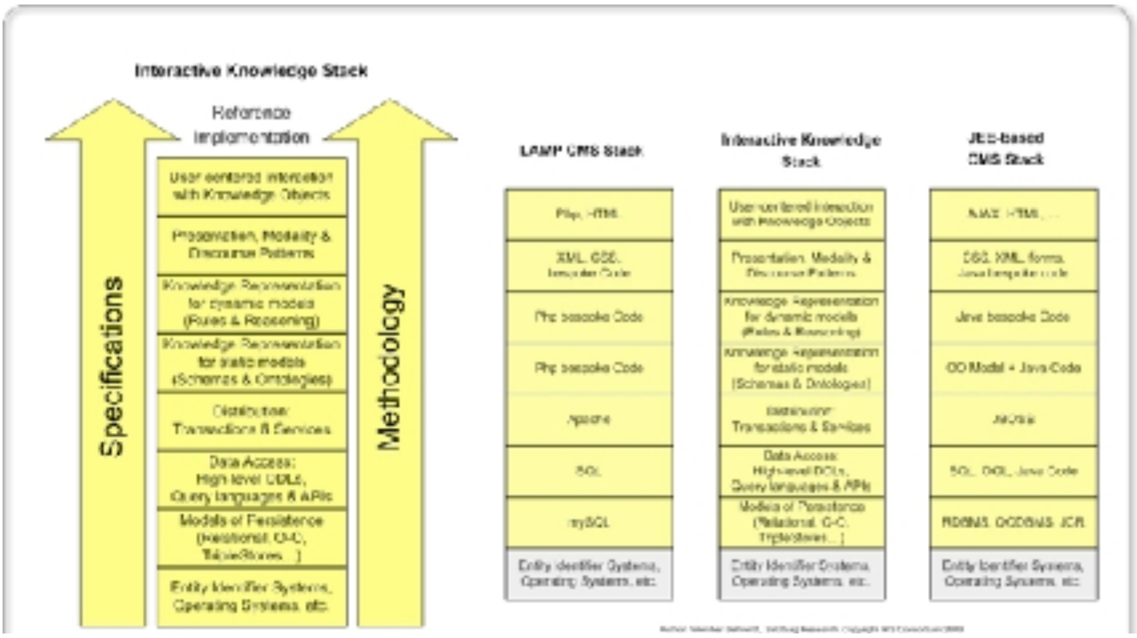
& related DevOps stuff

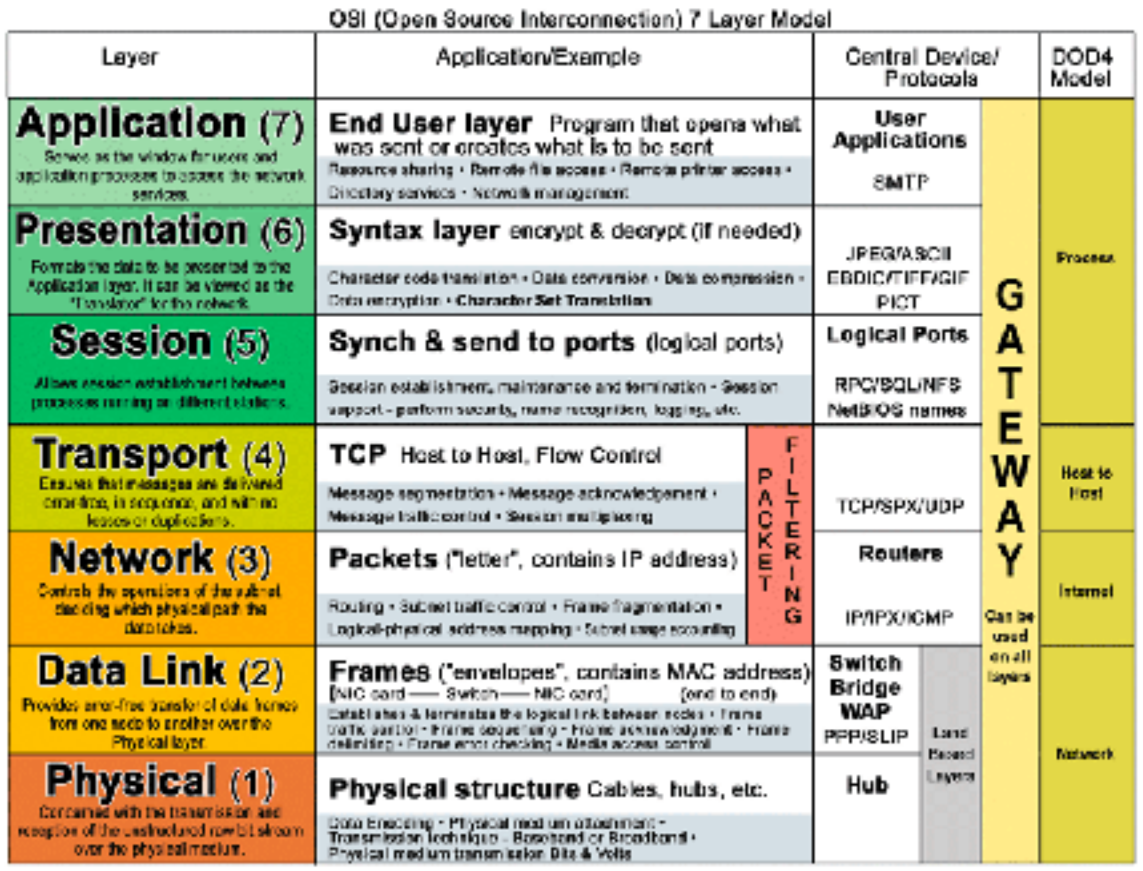
Takeaways

- How to promote the concepts - technology is an enabler but it’s not the hardest part
- Trigger curiosity and encourage you to experiment with the technology
- The technology stack can’t be thrusted, so what do we do?

About me

Technology stack 1


Interactive Knowledge Stack
Reference implementation
Specifications
Methodology
User centered interaction with Knowledge Objects
Presentation, Modality & Discourse Patterns
Knowledge Representation for dynamic models (Rules & Reasoning)
Knowledge Representation for static models (Schemas & Ontologies)
Distribution: Transactions & Services
Data Access: High level DDLs, Query languages & APIs
Models of Persistence (Relational, O-O, TripleStores, ...)
Entity Identifier Systems, Operating Systems, etc.
LAMP CMS Stack
Php, HTML
XML, CSS, template Code
Php bespoke Code
Php bespoke Code
Apache
SQL
mySQL
Entity Identifier Systems, Operating Systems, etc.
Interactive Knowledge Stack
JEE-based CMS Stack
AJAX, HTML, ...
CSS, XML forms, Java template code
Java bespoke Code
OO Model + Java Code
JavaEE
SQL, OQL, Java Code
RDBMS, OODBMS, JCR
Entity Identifier Systems, Operating Systems, etc.



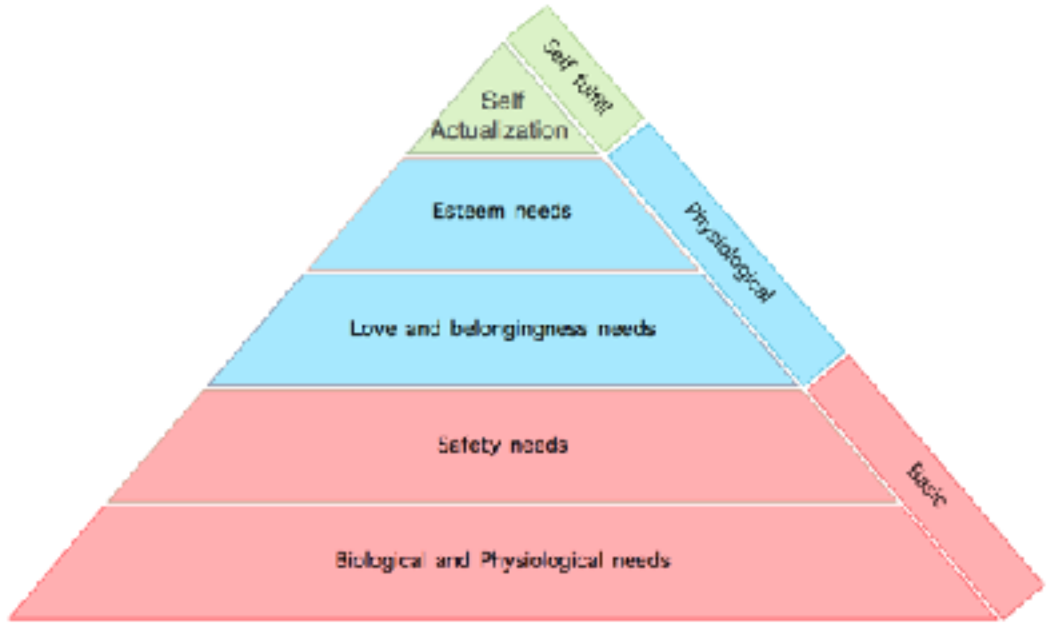
OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols			DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP		GATEWAY Can be used on all layers	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • Character Set Translation		JPEG/ASCII EBDIC/TIFF/GIF PICT			
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names			
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP			Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP			Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch** **Bridge** **WAP** PPP/SLIP	Land Based Layers		Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**			

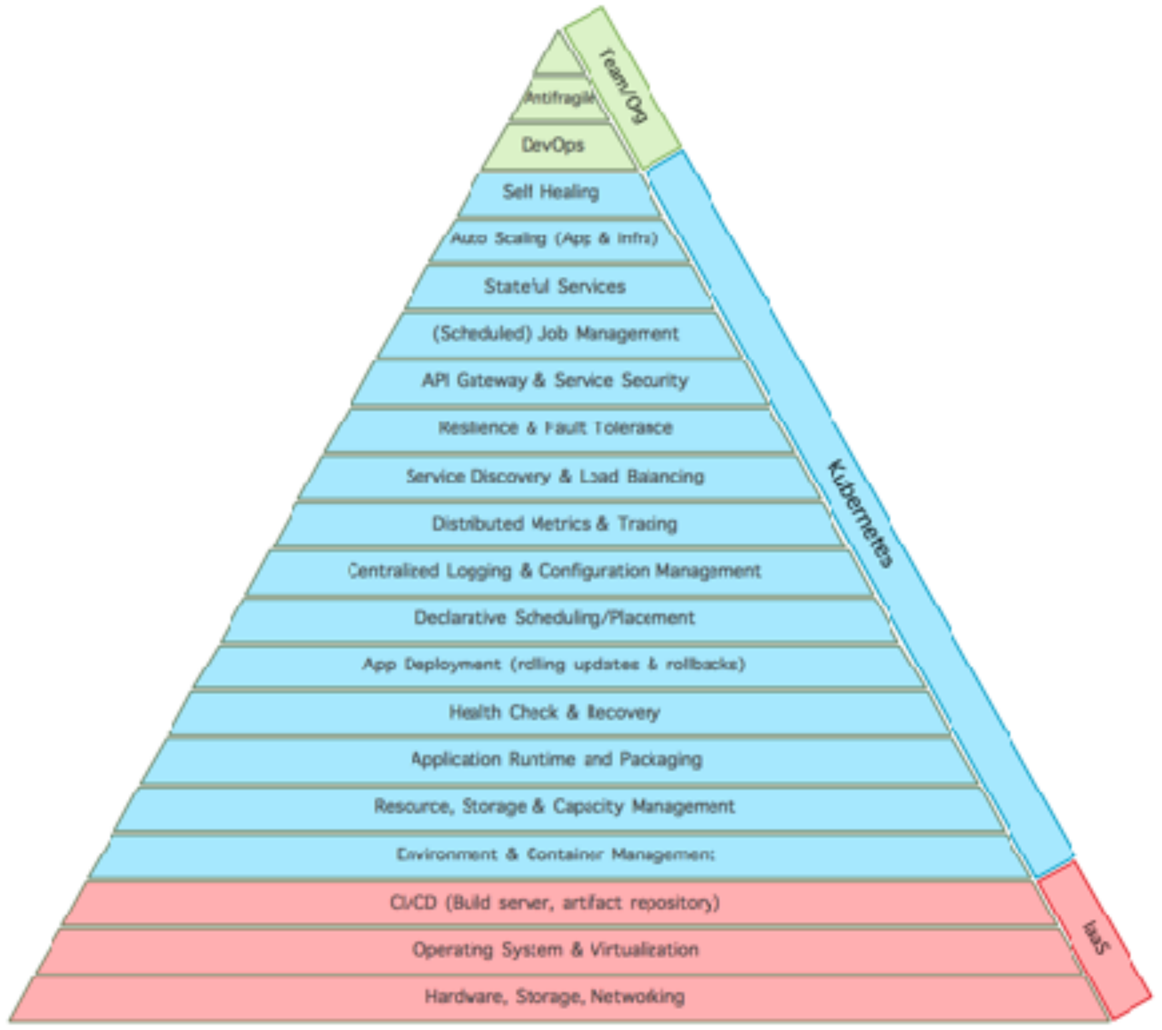
Technology stack 2

- Challenges
    - Communication
    - Unknown state Technology stack state
    - Both hardware and software are unreliable
- Solutions
    - Change the organisational culture (the hardest part)
    - Automation/Infrastructure as code
    - Immutable infrastructure

Maslow for Microservices



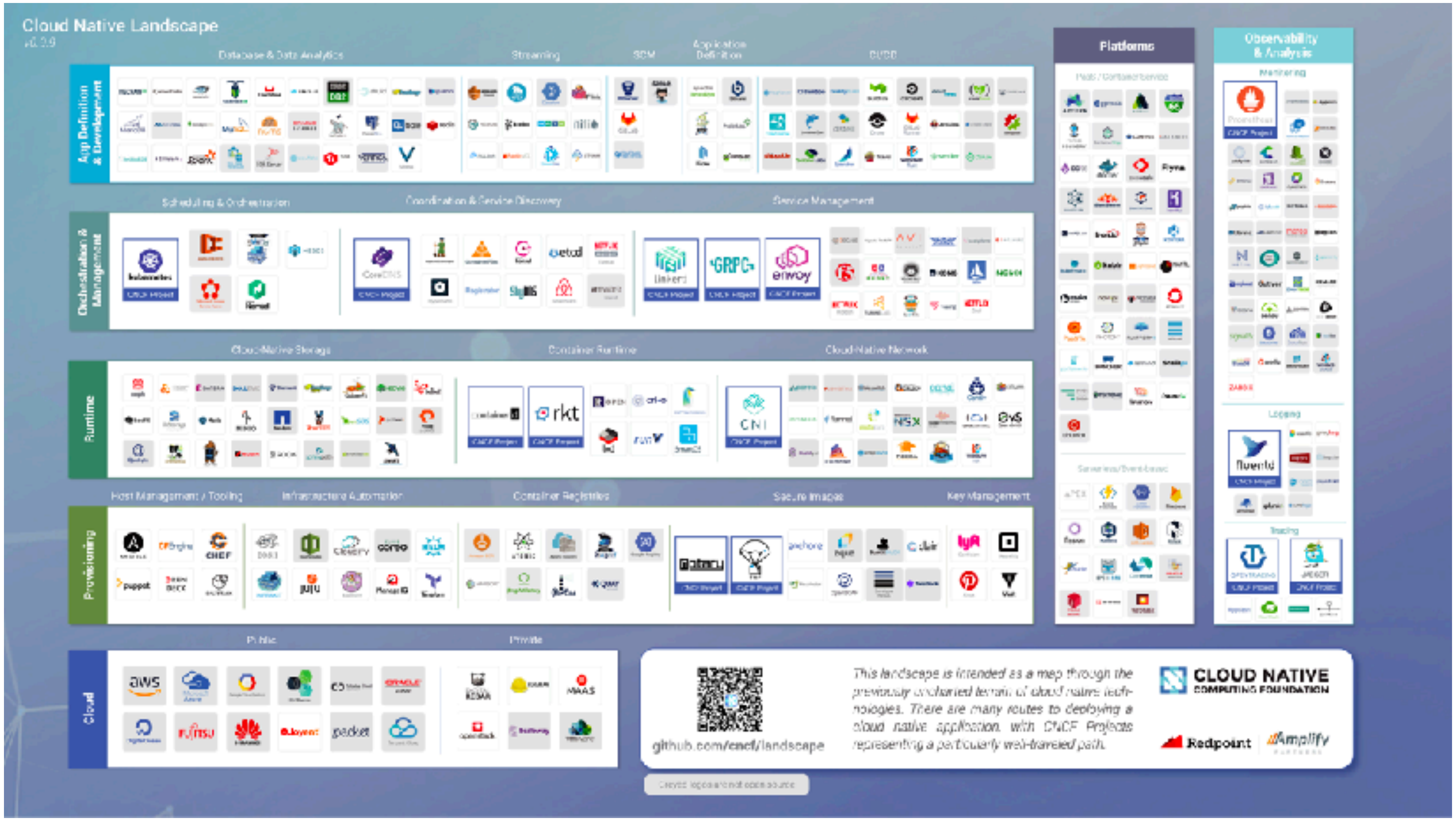
The Maslow Hierarchy of Needs


Team/Org
Antifragile
DevOps
Self Healing
Auto Scaling (Apps & Infra)
Stateful Services
(Scheduled) Job Management
API Gateway & Service Security
Resilience & Fault Tolerance
Service Discovery & Load Balancing
Distributed Metrics & Tracing
Kubernetes
Centralized Logging & Configuration Management
Declarative Scheduling/Placement
App Deployment (rolling updates & rollbacks)
Health Check & Recovery
Application Runtime and Packaging
Resource, Storage & Capacity Management
Environment & Container Management
CI/CD (Build server, artifact repository)
IaaS
Operating System & Virtualization
Hardware, Storage, Networking

The Microservices Hierarchy of Needs

https://thenewstack.io/introducing-microservices-hierarchy-needs/

CNCF technology landscape



This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

Why care as Developers

- Our software are no better than the underlying technology stack
- Flow/Productivity/Innovation

DevOps

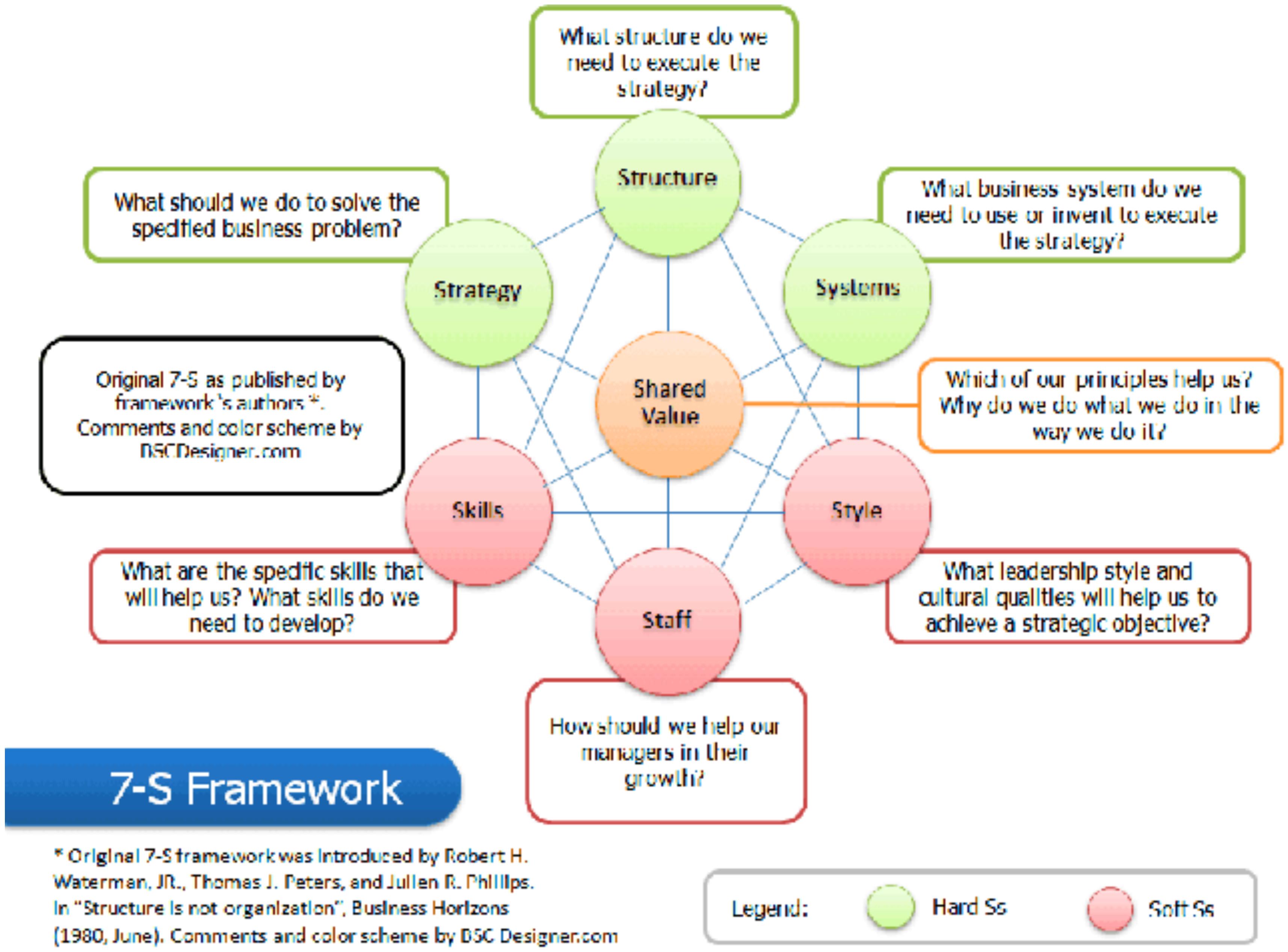
- DevOps is not a role
- It’s more a cultural movement about change and communication
- Westrum typology of organisations
- In addition to bing experts on technology some knowledge about culture and organisation is also required

Using the Westrum typology to measure culture

Pathological organisations are characterised by large amounts of fear and threat. People often hoard information or withhold it for political reasons, or distort it to make themselves look better.
Bureaucratic organisations protect departments. Those in the department want to maintain their "turf," insist on their own rules, and generally do things by the book — their book.
Generative organisations focus on the mission. How do we accomplish our goal? Everything is subordinated to good performance, to doing what we are supposed to do.
The Study of Information Flow: A Personal Journey; Westrum
The different organisational cultures have varied attributes along six axes:

Pathological	Bureaucratic	Generative
Low co-operation	Modest co-operation	High co-operation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

McKinsey 7-S Framework


What structure do we need to execute the strategy?
Structure
What should we do to solve the specified business problem?
Strategy
What business system do we need to use or invent to execute the strategy?
Systems
Original 7-S as published by framework's authors *. Comments and color scheme by BSCDesigner.com
Shared Value
Which of our principles help us? Why do we do what we do in the way we do it?
Skills
Style
What are the specific skills that will help us? What skills do we need to develop?
Staff
What leadership style and cultural qualities will help us to achieve a strategic objective?
How should we help our managers in their growth?
7-S Framework
* Original 7-S framework was introduced by Robert H. Waterman, JR., Thomas J. Peters, and Julien R. Phillips. In "Structure is not organization", Business Horizons (1980, June). Comments and color scheme by BSC Designer.com
Legend:
Hard Ss
Soft Ss

"Hard" elements are easier to define or identify and management can directly influence them: These are strategy statements; organisation charts and reporting lines; and formal processes and IT systems.

"Soft" elements, on the other hand, can be more difficult to describe, and are less tangible and more influenced by culture. However, these soft elements are as important as the hard elements if the organisation is going to be successful.

Provisioning tools

- Idempotency
- Tools
  - Ansible
  - Chef
  - Puppet

Introduction to Vagrant

Vagrant is a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past.

If you are already familiar with the basics of Vagrant, the documentation provides a better reference build for all available features and internals.

Vagrant network topology

```
                                 NAT network
       +---------------------------------------------+
       |               |              |              |
+---------+    +---------+    +---------+    +---------+
|         |    |         |    |         |    |         |
| Host    |    | K8s-01  |    | K8s-02  |    | K8s-03  |
|         |    |         |    |         |    |         |
+---------+    +---------+    +---------+    +---------+
      |               |              |               |
      +----------------------------------------------+
                                Host only NW
```

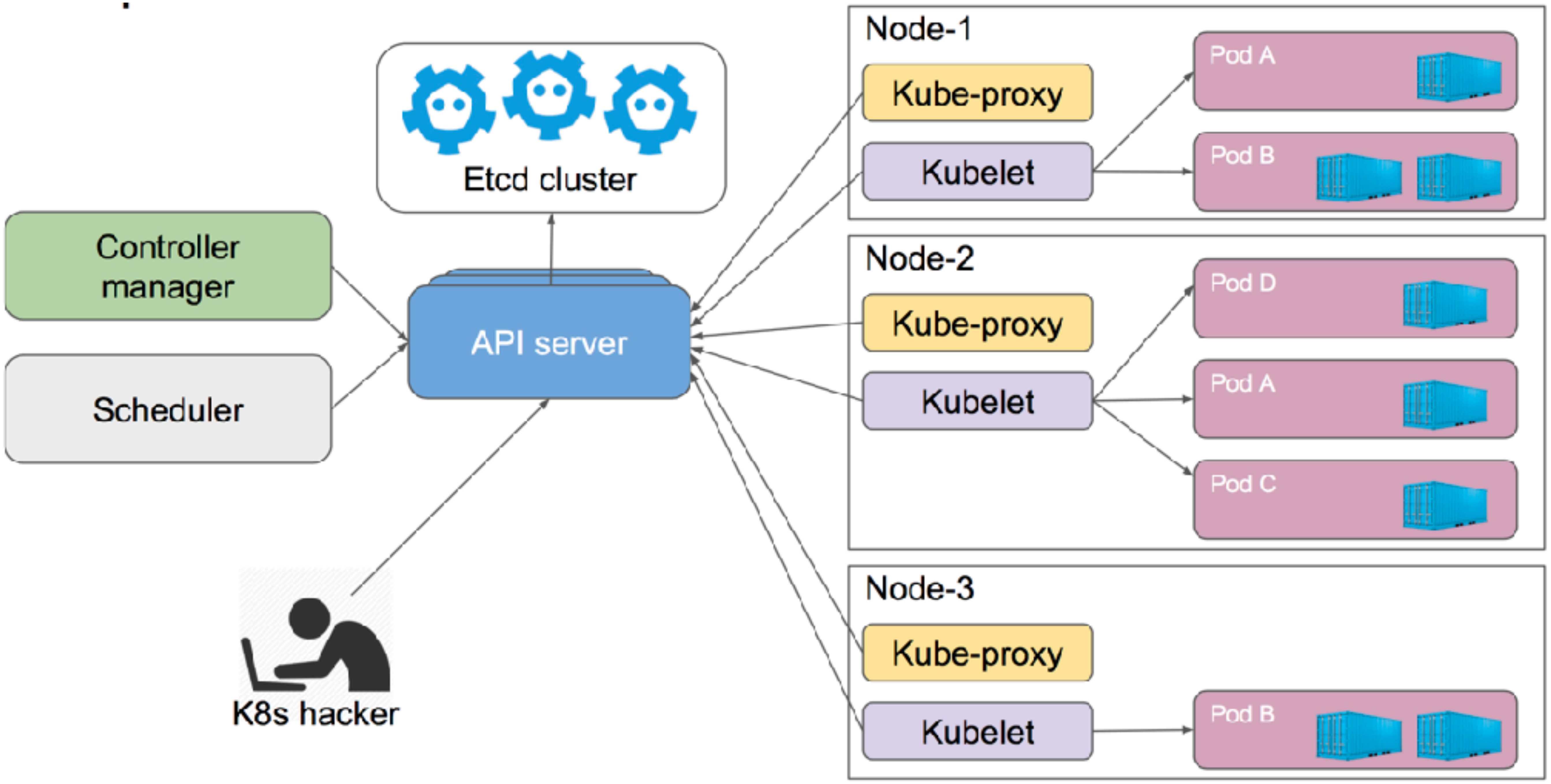
Why Kubernetes

- Seems like a nice abstraction for developers and a great platform for ops
- Infrastructure as code
- Embrase immutability
- Kubernetes isn’t the silver bullet. You still need to manage the entire cluster lifecycle. There is some nice handles for doing that.

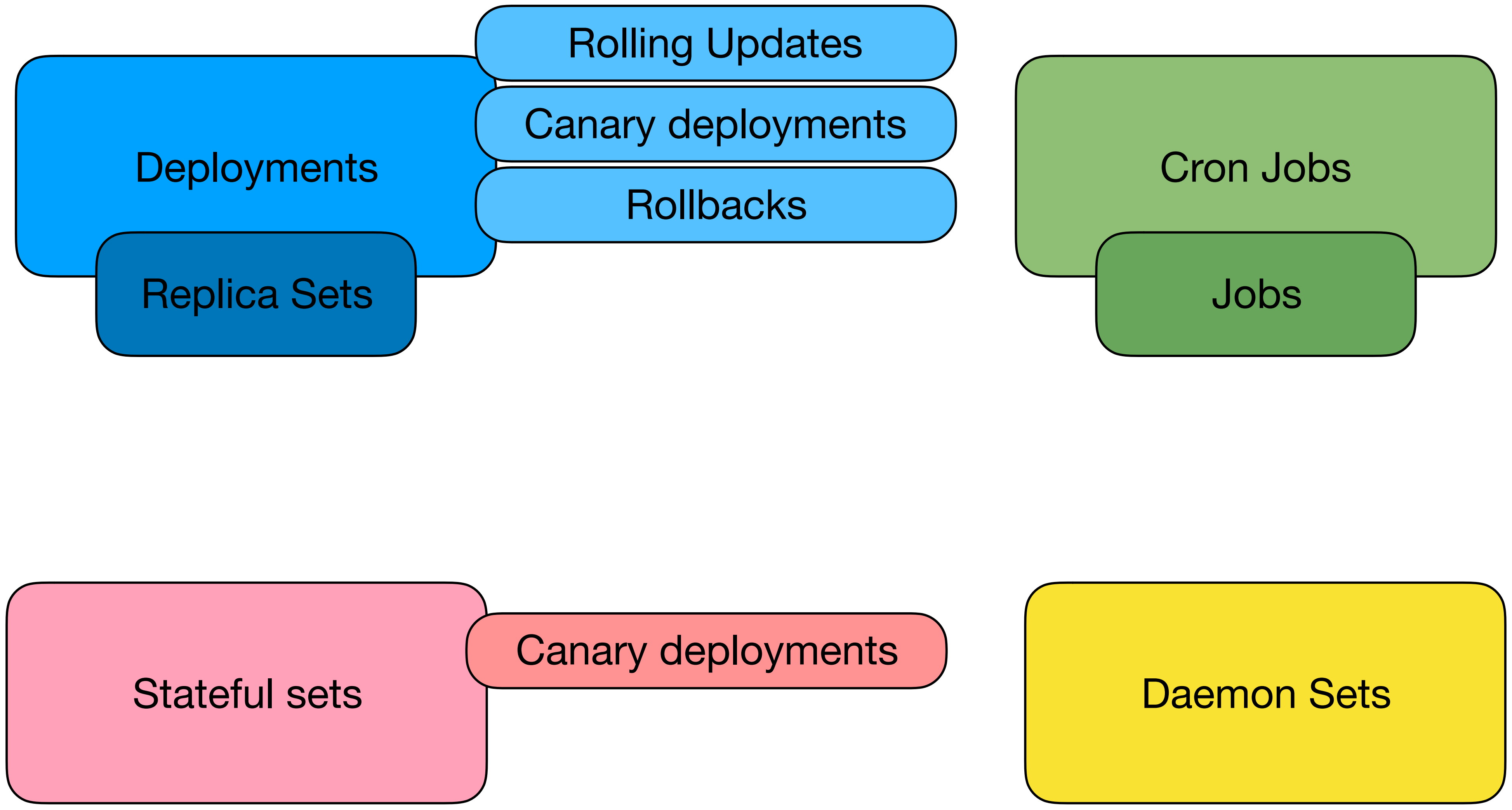
A history of Kubernetes

- Founded by Joe Beda, Brendan Burns and Craig McLuckie in 2014.
- It’s development and design are heavily influenced
  by Google's Borg system.
- It schedules and launches approximately 7,000 containers a second on any given day.
- Kubernetes v1.0 was released on July 21, 2015.

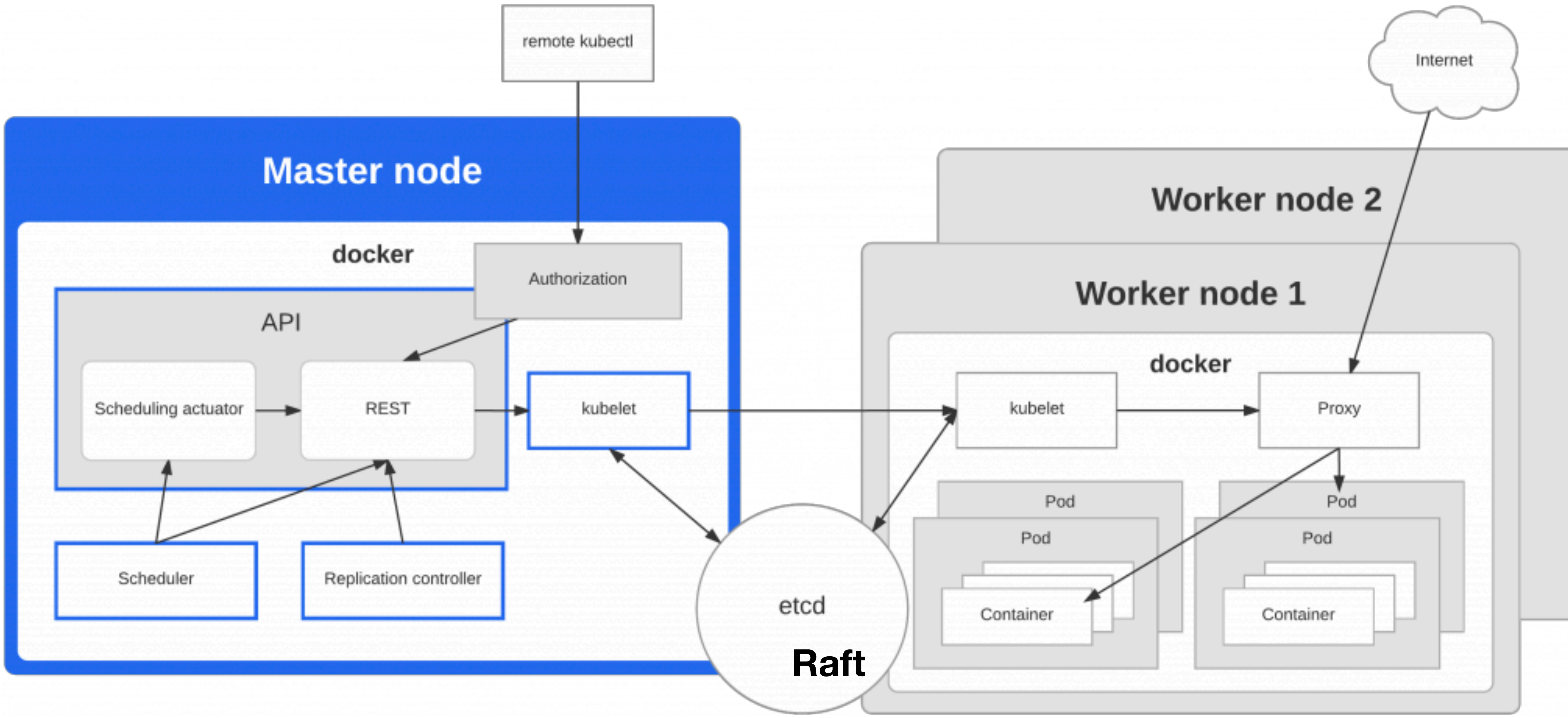
Components of Kubernetes



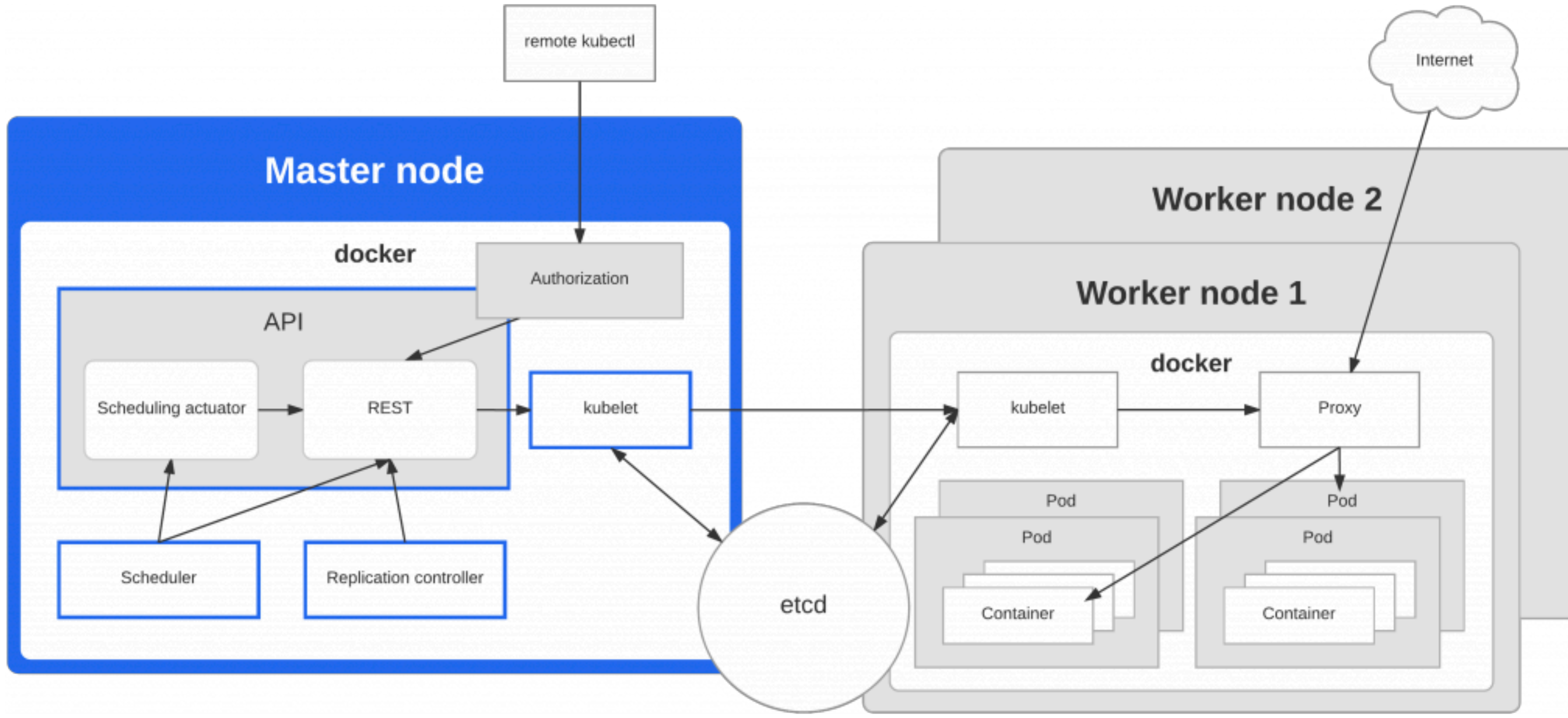
Controllers in Kubernetes



etcd

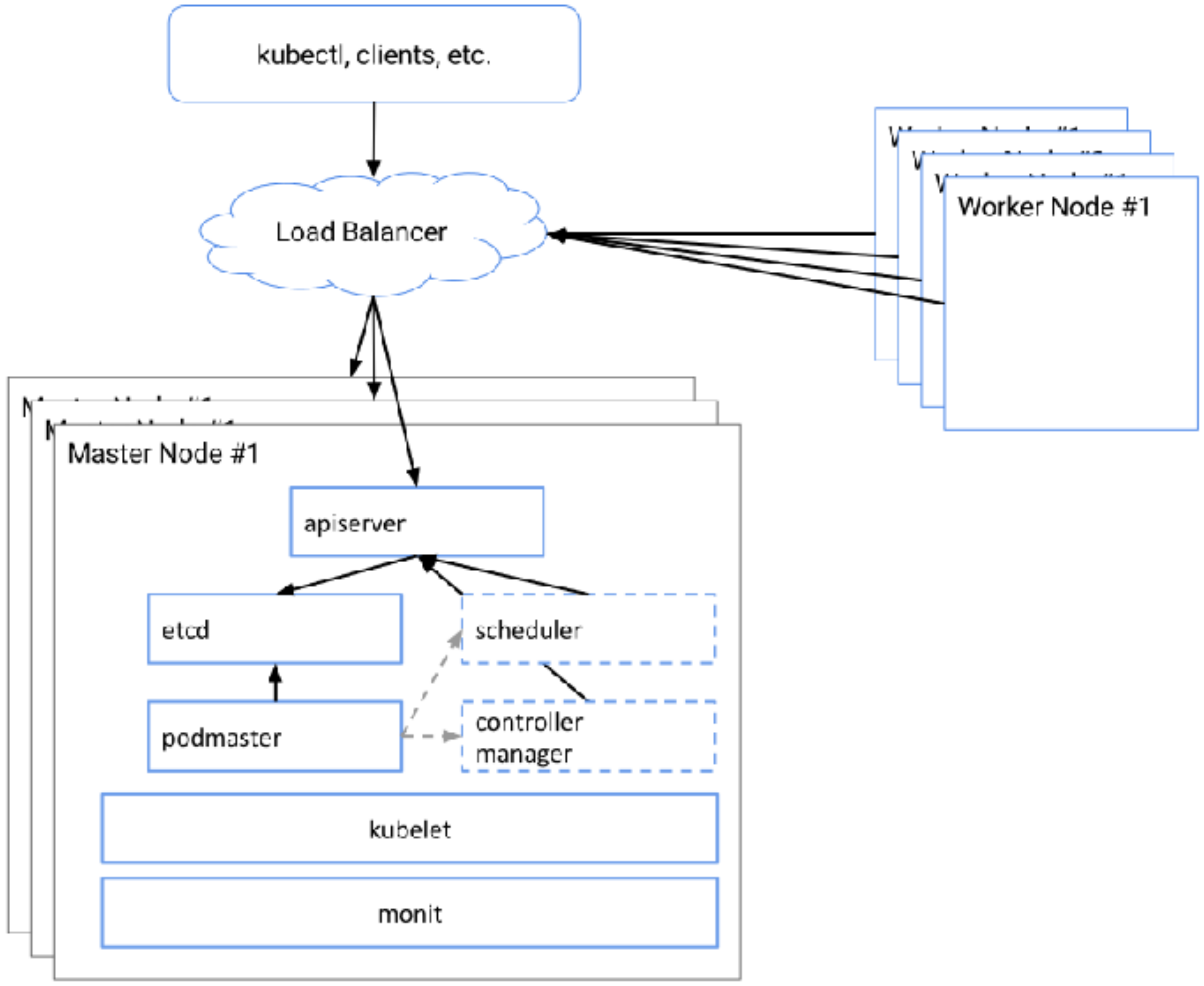

remote kubectl
Internet
Master node
docker
Authorization
API
Scheduling actuator
REST
kubelet
Scheduler
Replication controller
Worker node 2
Worker node 1
docker
kubelet
Proxy
Pod
Pod
Container
Pod
Pod
Container
etcd
Raft

PKI


remote kubectl
Internet
Master node
Worker node 2
Worker node 1
docker
Authorization
API
Scheduling actuator
REST
kubelet
Scheduler
Replication controller
etcd
docker
kubelet
Proxy
Pod
Pod
Container
Pod
Pod
Container

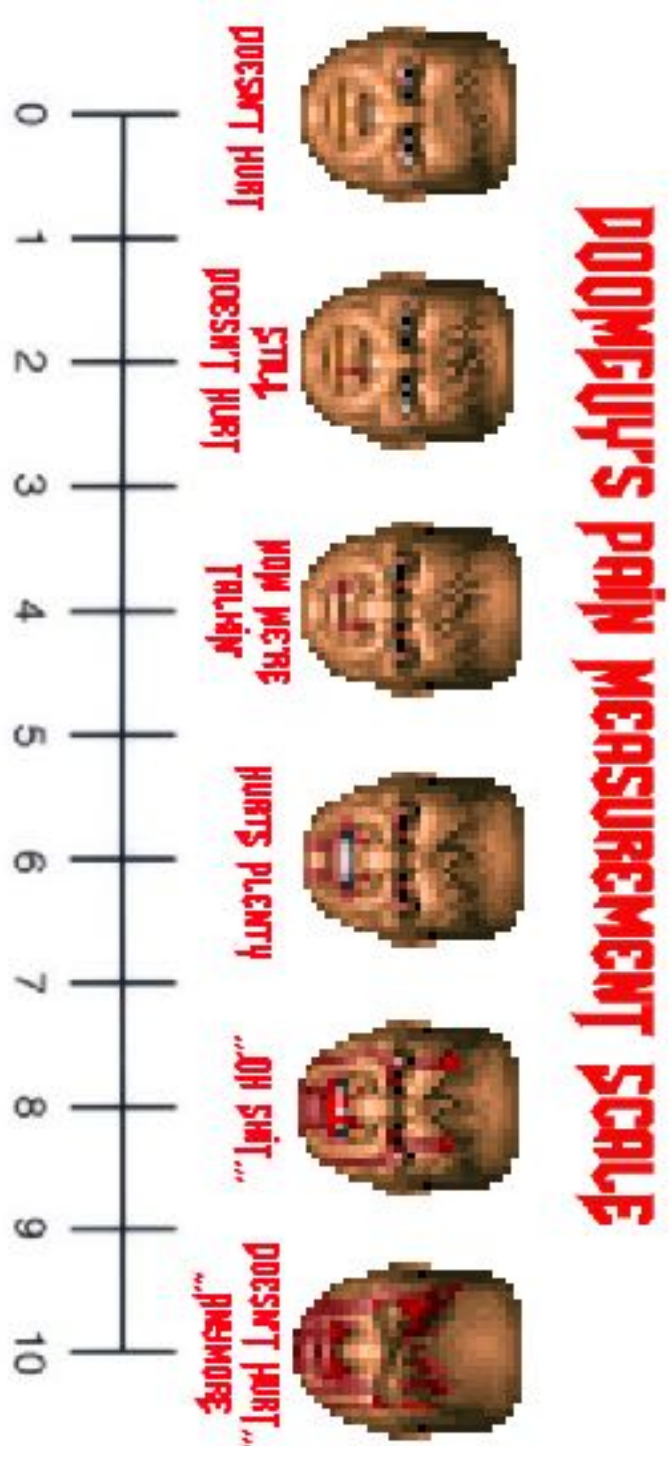
Demo

HA Kubernetes

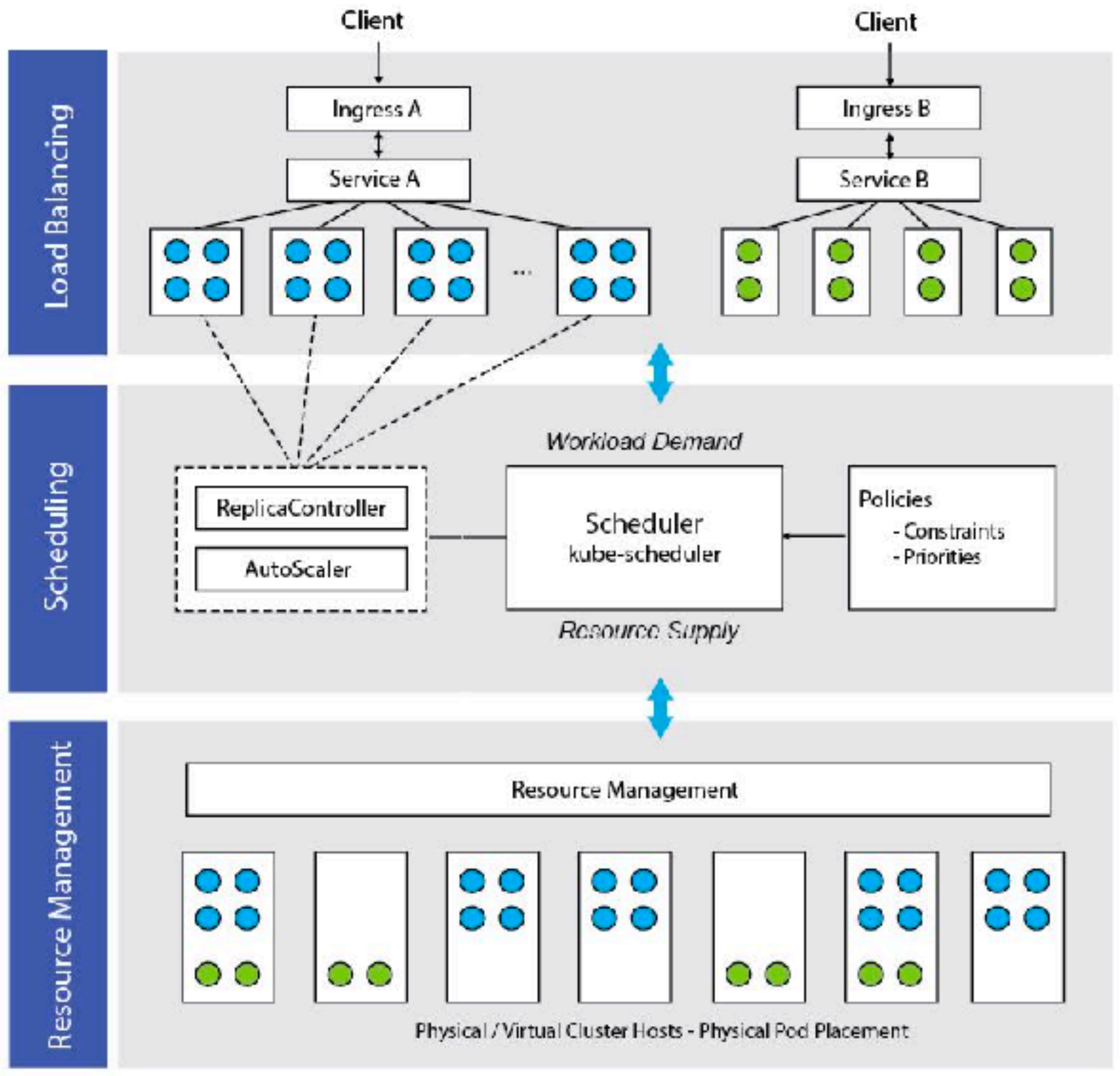


K8s provisioning

- Minikube
- Kubeadm (Lucas Käldström)
- Kubespray (On prem) vs Kops (Cloud)
- Kubernetes The Hard Way



The Three Pillars of Kubernetes Container Orchestration



application deployment and namespaces

```
kubectl create namespace mytest

kubectl -n mytest apply -f deployment.yaml

kubectl -n describe deployment nginx-
deployment
```

Working with Kubernetes

Command	Description
kubectl cluster-info	Get cluster status
kubectl get componentstatus	Get status of the cluster components
kubectl get serviceaccounts/default -o yaml --namespace=<NAMESPACE>	Get token for default serviceaccount in namespace
kubectl get pod [--all-namespaces]	Get information about all running pods
kubectl describe pod <pod>	Describe one pod
kubectl expose pod <pod> --port=4444 --name=frontend	Expose the port of a pod (creates a new service)
kubectl port-forward <pod> 8080	Port forward the exposed pod port to your local machine
kubectl attach <podname> -i	Attach the pod
kubectl exec <pod> — command	Executes a command on the pod
kubectl labels pods <pod> mylabel=awesome	Add a new label to a pod
kubectl run -i --tty busybox --image=busybox --restart=Never — sh	Run a shell in a pod - very useful for debugging
kubectl get deployments	Get information on current deployments
kubectl get rs	Get information about the replica sets
kubectl get pods --show-labels	Get pods, and also show labels attached to those pods
kubectl create -f <deployment yml file> [--record <CHANGE-CAUSE>]	Create a deployment, it's underlying replica sets and optionally a change cause description
kubectl rollout status deployment/<deployment name>	Get deployment status
kubectl set image deployment/<deployment name> <app name>=<docker repository image path>:<docker repository tag>	Run <app name> with the image label version <docker repository tag>
kubectl edit deployment/<deployment name>	Edit the deployment object
kubectl rollout status deployment/<deployment name>	Get status of the latest rollout
kubectl rollout history deployment/<deployment name>	Get the rollout history
kubectl rollout undo deployment/<deployment name>	Rollback to previous version
kubectl rollout undo deployment/<deployment name> --to-revision=n	Rollback to any version

Gitlab

- Demo Gitlab running in Docker
- Demo hostfile mangling or DNS server running in docker
- Demo exec into the docker container

Gitlab create registry cert

```
root@k8s-03:/etc/gitlab# tail -f /var/log/
gitlab/nginx/current

2017-10-11_11:07:12.09962 nginx: [emerg]
BIO_new_file("/etc/gitlab/ssl/
gitlab.example.com.crt") failed (SSL: error:
02001002:system library:fopen:No such file or
directory:fopen('/etc/gitlab/ssl/
gitlab.example.com.crt','r') error:
2006D080:BIO routines:BIO_new_file:no such
file)
```

Generate self-signed cert

```
root@k8s-03:/etc/gitlab/ssl#

openssl req -x509 -nodes -days 3650 -newkey
rsa:2048 -subj "/C=DK/ST=NA/L=Copenhagen/
O=Example Corp./OU=IT Department/
CN=gitlab.example.com" -keyout ./
gitlab.example.com.key -out ./
gitlab.example.com.crt
```

Let docker trust self signed cert

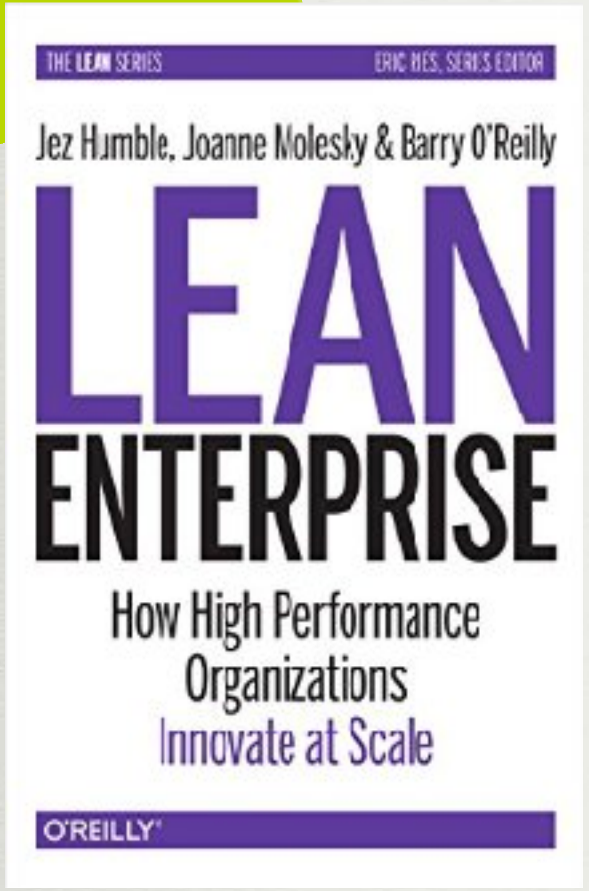
```
root@k8s-03:~# cp /srv/gitlab/config/ssl/gitlab.example.com.crt /usr/local/share/
ca-certificates/

update-ca-certificates



mkdir -p /etc/docker/certs.d/gitlab.example.com.crt:5005/

cp /srv/gitlab/config/ssl/gitlab.example.com.crt /etc/docker/certs.d/
gitlab.example.com.crt:5005/ca.crt



root@k8s-03:/etc/docker/certs.d/gitlab.example.com.crt:5005# systemctl daemon-
reload

root@k8s-03:/etc/docker/certs.d/gitlab.example.com.crt:5005# sudo systemctl restart
docker.service
```

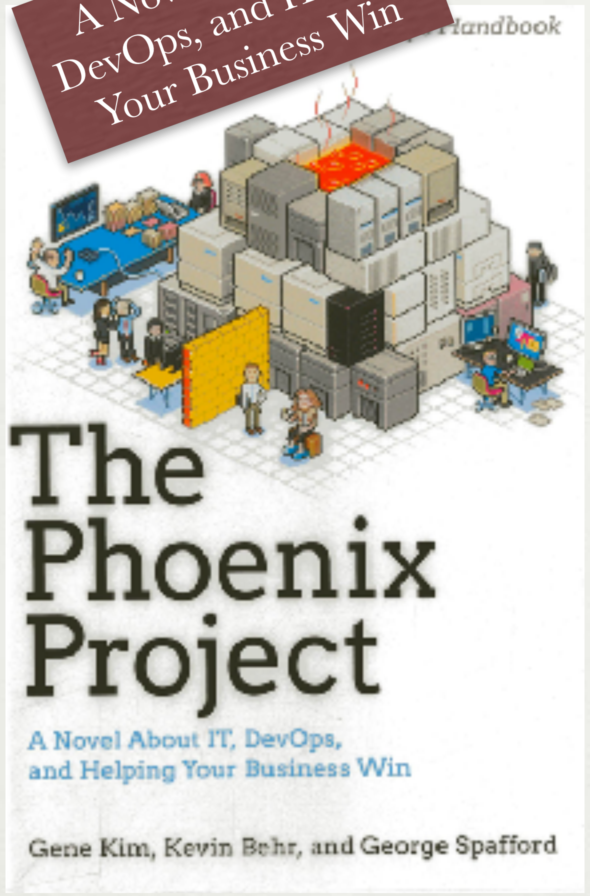
Resources

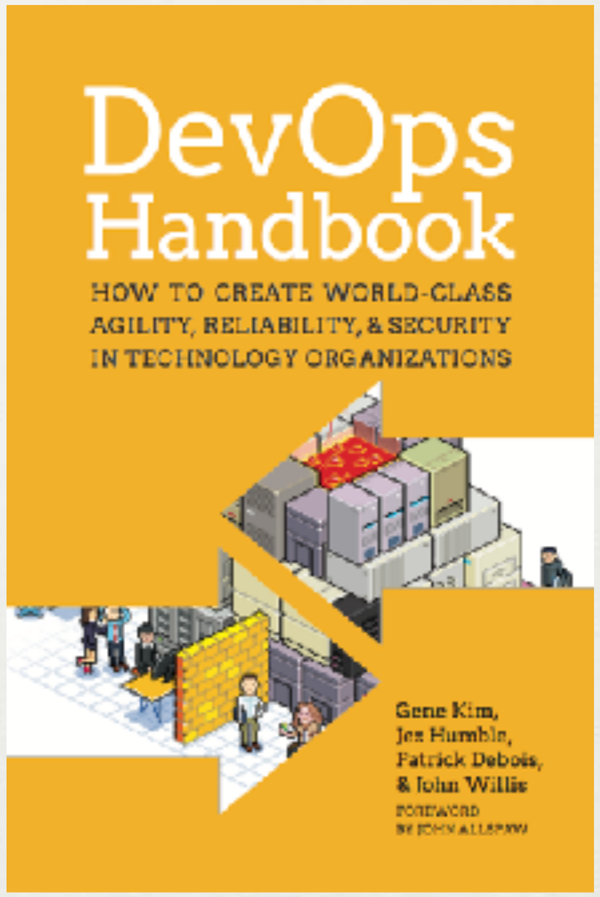
Win-win transactions

DORITH principle:
Do the RIght THing.
- Lars Kruse

A Novel about IT, DevOps, and Helping Your Business Win


THE LEAN SERIES
ERIC RIES, SERIES EDITOR
Jez Humble, Joanne Molesky & Barry O'Reilly
LEAN ENTERPRISE
How High Performance Organizations Innovate at Scale
O'REILLY


The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford


DevOps Handbook
HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS
Gene Kim, Jez Humble, Patrick Debois, & John Willis

Organisational

Hands-On

https://github.com/htesgaard/