

WORKSHOP ADVANCED DOCKER



สอนการ deploy Dockers ด้วย Kubernetes
จากประสบการณ์ใช้งานจริงบน Production ของ
application ระดับประเทศ



วิทยากร : คุณ PRAPARN LUNGPOONLARP
INFRASTRUCTURE ENGINEER, NETWORK ENGINEER,
SYSTEM ENGINEER



kubernetes



Outline Day 1

- Container concept (Recap)
- Introduction to Kubernetes
- System Architecture
- Fundamental of Kubernetes
 - Pods, Container and Services
 - Daemon Sets and Replication Controller (RC)
 - Deployment/Replica-Set (RS) and Rolling update
 - Resource Management and Horizontal Pods Autoscaling (HPA)
 - ConfigMap Secret



Outline Day 2

- Fundamental of Kubernetes
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- Security on Kubernetes
 - Network Policy
 - Volume Policy
 - Resource Usage Policy
 - Resource Consumption Policy
 - Access Control Policy
 - Security Policy
- Kubernetes in real world
 - Cluster Setup for Bare Metal
 - Orchestrator Assignment
 - nodeSelector
 - Interlude
 - Affinity
 - Taints/Tolerations
- Stateful application deployment
 - Consideration and Awareness
 - Persistent Volumes
 - StatefulSets



Pre-require

- Windows 10 (64 bit) / Mac OSX (64 Bit) with memory 8 GB
- Intermediate understand for docker/compose and container concept
- Cloud / Bare Material /Play with K8S
- LINE Account
- Basic understand for network/load balance concept
- Tool for editor (atom etc)
- Tool for shell (putty / terminal etc)
- Tool for transfer file (winscp / scp)
- Internet for download / upload image



LabSheet

- <https://tinyurl.com/y8yazrxy>

LabSheet

ไฟล์ แก้ไข ดู แทรก รูปแบบ ข้อมูล เครื่องมือ ស่วนเสริม ความช่วยเหลือ แก้ไขล่าสุด 1 นาทีที่ผ่านมา

Group No

	A	B	C	D	E	F	G	H	I	J
1	Group No	No	Machine Name	Private IP Address	Public IP Address	Role	Username	ssh connect string		Remark
2		1	Training_AdvanceDockerwithK8S_StudentG1_1	10.0.1.109	13.229.92.115	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.92.115		
3		2	Training_AdvanceDockerwithK8S_StudentG1_2	10.0.1.139	13.250.11.169	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 13.250.11.169		
4	1	3	Training_AdvanceDockerwithK8S_StudentG1_3	10.0.1.47	52.221.249.56	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 52.221.249.56		
5		4	Training_AdvanceDockerwithK8S_StudentG2_1	10.0.1.50	13.229.215.190	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.215.190		
6		5	Training_AdvanceDockerwithK8S_StudentG2_2	10.0.1.196	13.229.72.95	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.72.95		
7	2	6	Training_AdvanceDockerwithK8S_StudentG2_3	10.0.1.77	13.229.251.109	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.251.109		
8		7	Training_AdvanceDockerwithK8S_StudentG3_1	10.0.1.174	54.169.75.239	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 54.169.75.239		
9		8	Training_AdvanceDockerwithK8S_StudentG3_2	10.0.1.252	13.229.247.143	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.247.143		
10	3	9	Training_AdvanceDockerwithK8S_StudentG3_3	10.0.1.75	13.229.65.47	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.65.47		
11		10	Training_AdvanceDockerwithK8S_StudentG4_1	10.0.1.85	13.229.206.76	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.206.76		
12		11	Training_AdvanceDockerwithK8S_StudentG4_2	10.0.1.208	13.229.70.154	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 13.229.70.154		
13	4	12	Training_AdvanceDockerwithK8S_StudentG4_3	10.0.1.86	3.0.147.248	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.147.248		
14		13	Training_AdvanceDockerwithK8S_StudentG5_1	10.0.1.234	3.0.146.98	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.146.98		
15		14	Training_AdvanceDockerwithK8S_StudentG5_2	10.0.1.209	54.255.172.4	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 54.255.172.4		
16	5	15	Training_AdvanceDockerwithK8S_StudentG5_3	10.0.1.207	54.255.136.224	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 54.255.136.224		
17		16	Training_AdvanceDockerwithK8S_StudentG6_1	10.0.1.134	54.169.204.141	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 54.169.204.141		
18		17	Training_AdvanceDockerwithK8S_StudentG6_2	10.0.1.62	3.0.183.181	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.183.181		
19	6	18	Training_AdvanceDockerwithK8S_StudentG6_3	10.0.1.95	18.136.199.90	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 18.136.199.90		
20		19	Training_AdvanceDockerwithK8S_StudentG7_1	10.0.1.202	3.0.176.230	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.176.230		
21		20	Training_AdvanceDockerwithK8S_StudentG7_2	10.0.1.41	3.0.94.126	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.94.126		
22	7	21	Training_AdvanceDockerwithK8S_StudentG7_3	10.0.1.81	3.0.54.130	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.54.130		
23		22	Training_AdvanceDockerwithK8S_StudentG8_1	10.0.1.65	3.0.98.215	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.98.215		
24		23	Training_AdvanceDockerwithK8S_StudentG8_2	10.0.1.93	18.136.210.250	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 18.136.210.250		
25	8	24	Training_AdvanceDockerwithK8S_StudentG8_3	10.0.1.146	54.254.214.135	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 54.254.214.135		
26		25	Training_AdvanceDockerwithK8S_StudentG9_1	10.0.1.154	54.255.138.111	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 54.255.138.111		
27		26	Training_AdvanceDockerwithK8S_StudentG9_2	10.0.1.223	3.0.92.216	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.92.216		
28	9	27	Training_AdvanceDockerwithK8S_StudentG9_3	10.0.1.224	3.0.17.138	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.17.138		
29		28	Training_AdvanceDockerwithK8S_StudentG10_1	10.0.1.55	13.250.38.8	Master	ubuntu	ssh -i "docker_lab" ubuntu@ 13.250.38.8		
30		29	Training_AdvanceDockerwithK8S_StudentG10_2	10.0.1.149	52.221.184.17	Worker1	ubuntu	ssh -i "docker_lab" ubuntu@ 52.221.184.17		
31	10	30	Training_AdvanceDockerwithK8S_StudentG10_3	10.0.1.17	3.0.50.103	Worker2	ubuntu	ssh -i "docker_lab" ubuntu@ 3.0.50.103		

Present by: Praparn L. (eva10409@gmail.com)



kubernetes
by Google

Lab Resource

- Repository for lab

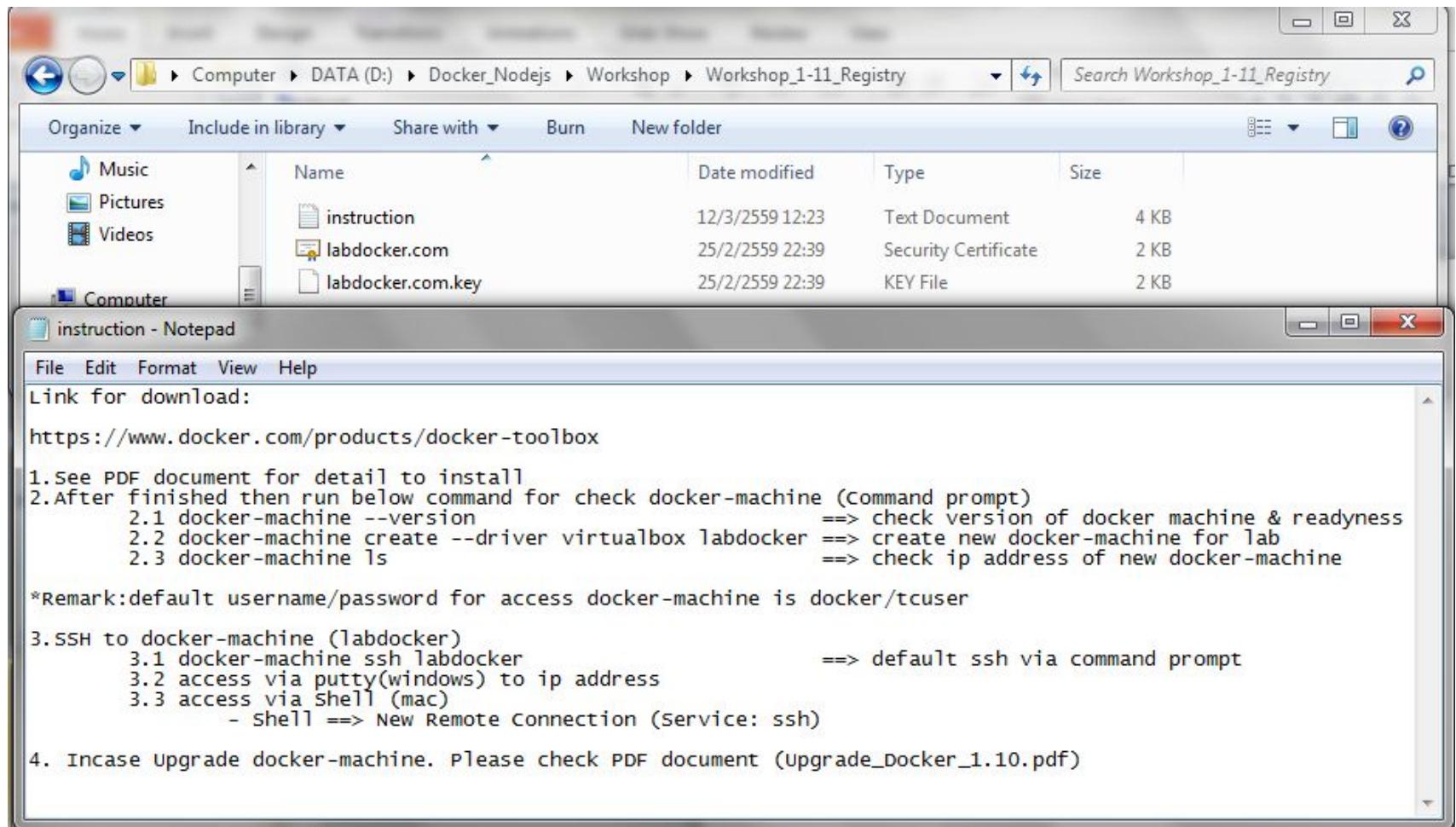
The screenshot shows a search results page for 'labdocker' on a platform that uses a Docker icon in its logo. The interface includes a search bar with the query 'labdocker', a 'Sign up' button, and a 'Log In' button. Below the search bar, the text 'Repositories (7)' is displayed. A dropdown menu labeled 'All' is visible. The main content area lists seven repositories:

Repository	Type	Stars	Pulls	Details
labdocker/alpineweb	public	0	31	DETAILS
labdocker/nginx	public	0	16	DETAILS
labdocker/alpine	public	0	7	DETAILS
labdocker/centos	public	0	5	DETAILS
labdocker/debian	public	0	4	DETAILS
labdocker/fedora	public	0	3	DETAILS
labdocker/ubuntu	public	0	2	DETAILS



Lab Resource

- Software in lab



Lab Resource

- Download on Google Drive
 - <http://tinyurl.com/yxoe8aj6>
- Download on GitHub
 - Git clone https://github.com/praparn/kubernetes_201907

No description, website, or topics provided.

Branch: master ▾ New pull request

Clone with SSH Use HTTPS
git@github.com:praparn/kuberneteslab.git

File	Created	Last Commit
WorkShop_1.1_Install_Kubernetes	2017/11/02 14:05:55	2017/11/02 14:05:55
WorkShop_1.2_Pods_Service_Deployment	2017/11/02 14:05:55	2017/11/02 14:05:55
WorkShop_1.3_Replication_Controller	2017/11/02 14:05:55	2017/11/02 14:05:55
WorkShop_1.4_Deployment	2017/11/02 14:05:55	2017/11/02 14:05:55
WorkShop_1.5_Volume	2017/11/02 14:05:55	an hour ago
WorkShop_1.6_Liveness_Readiness_Probe	2017/11/02 14:05:55	an hour ago
WorkShop_1.7_Resource_Management_and_HPA	2017/11/02 14:05:55	an hour ago
WorkShop_2.1_ConfigMap_Secret	2017/11/02 14:05:55	an hour ago
WorkShop_2.3_Log_and_Monitoring	2017/11/02 14:05:55	an hour ago
WorkShop_2.4_Ingress_Network	2017/11/02 14:05:55	an hour ago
WorkShop_2.5_Kubernetes_RealWorld	2017/11/02 14:05:55	an hour ago
WorkShop_2.6_Orchestrator_Assignment	2017/11/08 23:56:28	2 days ago
WorkShop_2.7_Persistent_Storage	2017/11/10	23 hours ago
Workshop_2.2_Job_CronJob	2017/11/02 14:05:55	an hour ago
.DS_Store	2017/11/00 03:43	22 hours ago
Kubernetes_Training_master.pdf	2017/11/00 03:43	22 hours ago
Untitled-1	2017/11/02 14:05:55	an hour ago



Workshop: Install Kubernetes



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop: Install Kubernetes

- LAB Sheet
- <https://tinyurl.com/y8yazrxy>

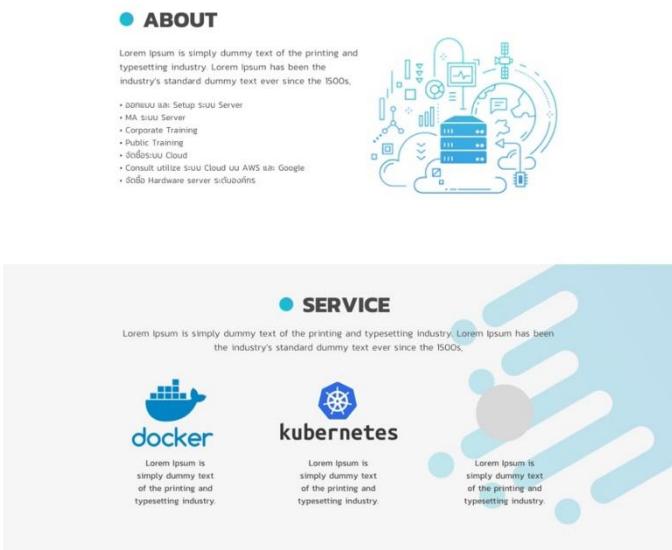
Table No	Group No	No	Public IP Address	Private IP Address	Role
1	1	1	54.169.12.33	10.0.1.25	Master
		2	54.255.243.139	10.0.1.102	Worker
		3	52.221.203.26	10.0.1.254	Worker
	2	1	13.250.121.106	10.0.1.100	Master
		2	54.255.243.173	10.0.1.202	Worker
		3	52.221.232.245	10.0.1.12	Worker
	3	1	54.169.39.94	10.0.1.169	Master
		2	13.229.72.232	10.0.1.107	Worker
		3	13.250.109.203	10.0.1.135	Worker
2	4	1	13.250.103.188	10.0.1.110	Master
		2	13.229.128.105	10.0.1.199	Worker
		3	54.169.134.39	10.0.1.162	Worker
	5	1	54.255.145.5	10.0.1.32	Master
		2	54.254.218.192	10.0.1.59	Worker
		3	52.221.216.253	10.0.1.166	Worker
3	6	1	13.250.112.82	10.0.1.45	Master
		2	54.254.166.86	10.0.1.50	Worker
		3	52.221.181.188	10.0.1.218	Worker
	7	1	52.221.208.50	10.0.1.182	Master
		2	52.77.249.187	10.0.1.4	Worker
		3	13.250.104.158	10.0.1.173	Worker
4	8	1	54.169.32.99	10.0.1.222	Master
		2	52.221.191.74	10.0.1.147	Worker
		3	13.229.80.170	10.0.1.167	Worker



Who are we ? (Opcellent)



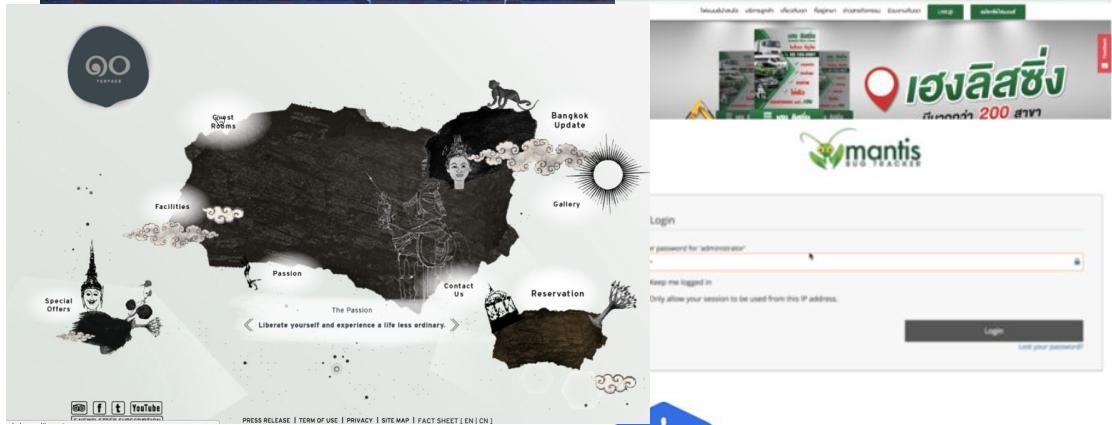
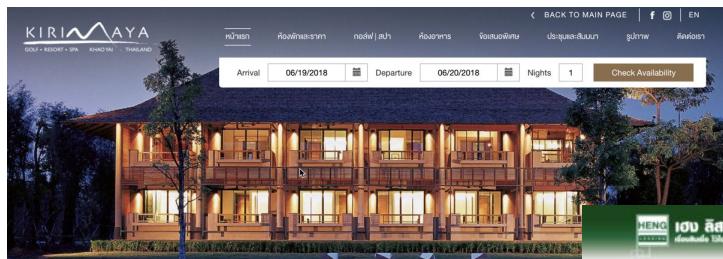
The Opcellent website features a light blue header with the company logo and navigation links for About, Service, Training, and Contact. Below the header is a large white section with a blue abstract graphic and the text "Modern Server Technology Implementer". A "GET DETAILS" button is located in the bottom left corner of this section.



The "SERVICE" page on the Opcellent website includes sections for Docker and Kubernetes. Each section contains a brief description of the service and icons for Docker and Kubernetes. The background features a large blue abstract graphic.



A screenshot of an event listing for "Advanced Docker and Kubernetes 101" presented by GDG Chiang Mai and Artisan Digital. The event is on Saturday, 12 May 2018, at Punspace Tha Pae Gate. It includes a brief description, a photo of the speaker, and registration details. Buttons for "Interested" and "Going" are shown.



A composite image showing the Heng Leasing website and a login screen. The website features a map of Thailand with various landmarks and text like "Guest R80ms", "Facilities", "Passion", "The Passion", "Contact Us", "Reservation", and "Bangkok Update". The login screen shows a "Login" button and a "Forgot your password?" link.

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes 1.20



[Documentation](#) [Kubernetes Blog](#)

Search

Home
Getting started
Release notes and version skew
v1.20 Release Notes
Kubernetes version and version skew support policy
Learning environment
Production environment
Best practices
Concepts
Tasks
Tutorials
Reference
Contribute

What's New (Major Themes)

Dockershim deprecation

Docker as an underlying runtime is being deprecated. Docker-produced images will continue to work in your cluster with all runtimes, as they always have. The Kubernetes community [has written a blog post about this in detail](#) with a dedicated FAQ page for it.

External credential provider for client-go

The client-go credential plugins can now be passed in the current cluster information via the `KUBERNETES_EXEC_INFO` environment variable. Learn more about this on [client-go credential plugins documentation](#).

CronJob controller v2 is available through feature gate

An alternative implementation of `CronJob` controller is now available as an alpha feature in this release, which has experimental performance improvement by using informers instead of polling. While this will be the default behavior in the future, you can [try them in this release through a feature gate](#).

PID Limits graduates to General Availability

PID Limits features are now generally available on both `SupportNodePidsLimit` (node-to-pod PID isolation) and `SupportPodPidsLimit` (ability to limit PIDs per pod), after being enabled-by-default in beta stage for a year.

API Priority and Fairness graduates to Beta

Initially introduced in 1.18, Kubernetes 1.20 now enables API Priority and Fairness (APF) by default. This allows `kube-apiserver` to [categorize incoming requests by priority levels](#).

IPv4/IPv6 run

IPv4/IPv6 dual-stack has been reimplemented for 1.20 to support dual-stack Services, based on user and community feedback. If your cluster has dual-stack enabled, you can create Services which can use IPv4, IPv6, or both, and you can change this setting for existing Services. Details are available in updated [IPv4/IPv6 dual-stack docs](#), which cover the nuanced array of options.

We expect this implementation to progress from alpha to beta and GA in coming releases, so we're eager to have you comment about your dual-stack experiences in [#k8s-dual-stack](#) or in [enhancements #563](#).

go1.15.5

go1.15.5 has been integrated to Kubernetes project as of this release, including other infrastructure related updates on this effort.

Ref: <https://kubernetes.io/docs/setup/release/notes/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes 1.20



[Documentation](#) [Kubernetes Blog](#)

2021

2020

A Custom Kubernetes Scheduler to Orchestrate Highly Available Applications

Kubernetes 1.20: Pod Impersonation and Short-lived Volumes in CSI Drivers

Third Party Device Metrics Reaches GA

Kubernetes 1.20: Granular Control of Volume Permission Changes

Kubernetes 1.20: Kubernetes Volume Snapshot Moves to GA

Kubernetes 1.20: The Raddest Release

GSOD 2020: Improving the API Reference Experience

Don't Panic: Kubernetes and Docker

Dockershim Deprecation FAQ

Cloud native security for your clusters

[Show More Posts...](#)

2019

2018

2017

2016

2015

Dockershim Deprecation FAQ

Wednesday, December 02, 2020

This document goes over some frequently asked questions regarding the Dockershim deprecation announced as a part of the Kubernetes v1.20 release. For more detail on the deprecation of Docker as a container runtime for Kubernetes kubelets, and what that means, check out the blog post [Don't Panic: Kubernetes and Docker](#).

Why is dockershim being deprecated?

Maintaining dockershim has become a heavy burden on the Kubernetes maintainers. The CRI standard was created to reduce this burden and allow smooth interoperability of different container runtimes. Docker itself doesn't currently implement CRI, thus the problem.

Dockershim was always intended to be a temporary solution (hence the name: shim). You can read more about the community discussion and planning in the [Dockershim Removal Kubernetes Enhancement Proposal](#).

Additionally, features that were largely incompatible with the dockershim, such as cgroups v2 and user namespaces are being implemented in these newer CRI runtimes. Removing support for the dockershim will allow further development in those areas.

Can I still use Docker in Kubernetes 1.20?

Yes, the only thing changing in 1.20 is a single warning log printed at `kubelet` startup if using Docker as the runtime.

When will dockershim be removed?

Given the impact of this change, we are using an extended deprecation timeline. It will not be removed before Kubernetes 1.22, meaning the earliest release without dockershim would be 1.23 in late 2021. We will be working closely with vendors and other ecosystem groups to ensure a smooth transition and will evaluate things as the situation evolves.

Will my existing Docker images still work? [↳](#)

Yes, the images produced from `docker build` will work with all CRI implementations. All your existing images will still work exactly the same.

What about private images?

Yes. All CRI runtimes support the same pull secrets configuration used in Kubernetes, either via the PodSpec or ServiceAccount.

Are Docker and containers the same thing?

Docker popularized the Linux containers pattern and has been instrumental in developing the underlying technology, however containers in Linux have existed for a long time. The container ecosystem has grown to be much broader than just Docker. Standards like OCI and CRI have helped many tools grow and thrive in our ecosystem, some replacing aspects of Docker while others enhance existing functionality.

Ref: <https://kubernetes.io/blog/2020/12/02/dockershim-faq/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes 1.20



Products Use Cases Delivery Approach

[BOOK A DEMO](#)

[CONTACT US](#)



Mirantis to take over support of Kubernetes dockershim

Adam Parco - December 4, 2020 - [kubernetes](#)



For many people this has sparked a moment of panic, but take a deep breath, everything is going to be OK.

Even better news, however, is that Mirantis and Docker have agreed to partner to maintain the shim code standalone outside Kubernetes, as a conformant CRI interface for the Docker Engine API. For Mirantis customers, that means that Docker Engine's commercially supported version, Mirantis Container Runtime (MCR), will be CRI compliant. We will start with the great initial prototype from [Dims](#) at <https://github.com/dims/cri-dockerd> and continue to make it available as an open source project, <https://github.com/Mirantis/cri-dockerd>. This means that you can continue to build Kubernetes based on the Docker Engine as before, just switching from the built in dockershim to the external one. We will work together on making sure it continues to work as well as before and that it passes all the conformance tests and continues to work just like the built in version did. Mirantis will be using this in Mirantis Kubernetes Engine, and Docker will continue to ship this shim in Docker Desktop.

Get blogs and other content delivered straight to your inbox.

Email

Country

[SUBSCRIBE](#)

LIVE WEBINAR

Want to learn OpenStack? Join us for OpenStack Basics

Thursday March 25,
8:00 PDT

[SAVE YOUR SEAT](#)

FREE VIRTUAL TRAINING

Creating K8s objects with YAML

Ref: <https://www.mirantis.com/blog/mirantis-to-take-over-support-of-kubernetes-dockershim-2/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes 1.20



Documentation Kubernetes Blog Training Partners Community Case Studies Versions English



Home

Getting started

Release notes and version skew

Learning environment

Production environment

Container runtimes

Installing Kubernetes with deployment tools

Turnkey Cloud Solutions

Windows in Kubernetes

Kubernetes Documentation / Getting started / Production environment / Container runtimes

Container runtimes

You need to install a container runtime into each node in the cluster so that Pods can run there. This page outlines what is involved and describes related tasks for setting up nodes.

This page lists details for using several common container runtimes with Kubernetes, on Linux:

- [containerd](#)
- [CRI-O](#)
- [Docker](#)

Note: For other operating systems, look for documentation specific to your platform.

[Edit this page](#)

[Create child page](#)

[Create an issue](#)

[Print entire section](#)

Cgroup drivers

Container runtimes

containerd

CRI-O

Docker

containerd

This section contains the necessary steps to use containerd as CRI runtime.

Use the following commands to install Containerd on your system:

Install and configure prerequisites:

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# Setup required sysctl params, these persist across reboots.
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

Ref: <https://kubernetes.io/docs/setup/production-environment/container-runtimes/#containerd>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

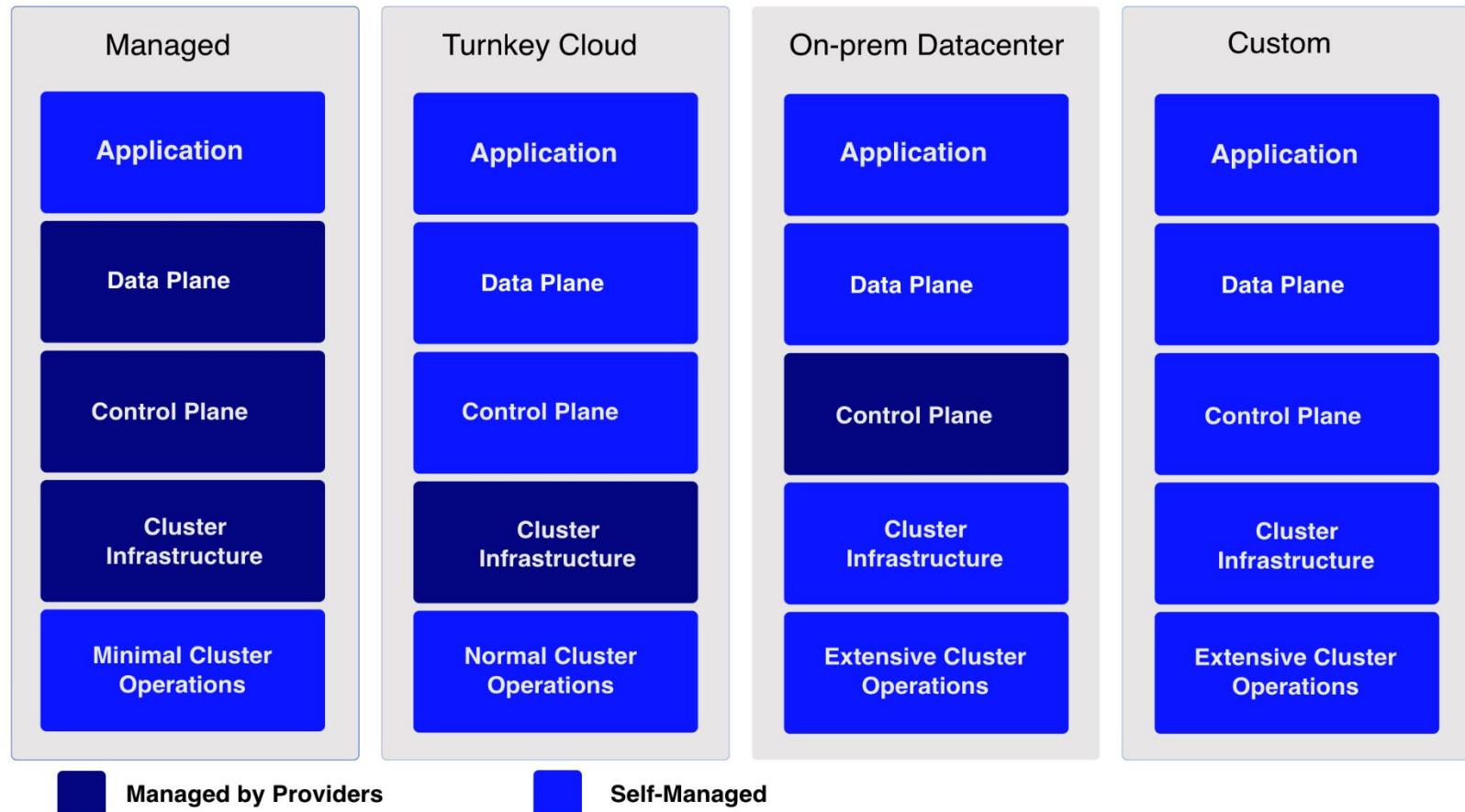
Kubernetes Environment

Production environment

When evaluating a solution for a production environment, consider which aspects of operating a Kubernetes cluster (or *abstractions*) you want to manage yourself or offload to a provider.

Some possible abstractions of a Kubernetes cluster are applications, data plane, control plane, cluster infrastructure, and cluster operations.

The following diagram lists the possible abstractions of a Kubernetes cluster and whether an abstraction is self-managed or managed by a provider.



Production environment solutions

Ref:<https://kubernetes.io/docs/setup/release/notes/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes Environment

Providers	Managed	Turnkey cloud	On-prem datacenter	Custom (cloud)	Custom (On-premises VMs)	Custom (Bare Metal)
Agile Stacks		✓	✓			
Alibaba Cloud		✓				
Amazon	Amazon EKS	Amazon EC2				
AppsCode	✓					
APPUiO	✓	✓	✓			
Banzai Cloud Pipeline Kubernetes Engine (PKE)		✓		✓	✓	✓
CenturyLink Cloud		✓				
Cisco Container Platform			✓			
Cloud Foundry Container Runtime (CFCR)				✓	✓	
CloudStack					✓	
Canonical	✓	✓	✓	✓	✓	✓
Containership	✓	✓				
Digital Rebar						✓
DigitalOcean	✓					
Docker Enterprise		✓	✓			✓
Fedora (Multi Node)					✓	✓
Fedora (Single Node)						✓
Gardener	✓	✓	✓ (via OpenStack)	✓		
Giant Swarm	✓	✓	✓			
Google	Google Kubernetes Engine (GKE)	Google Compute Engine (GCE)	GKE On-Prem			
IBM	IBM Cloud Kubernetes Service		IBM Cloud Private			
Ionos	Ionos Managed Kubernetes	Ionos Enterprise Cloud				
Kontena Pharos		✓	✓			
Kubermatic	✓	✓	✓			
KubeSail	✓					
Kubespray				✓	✓	✓
Kulbr	✓	✓	✓	✓	✓	✓
Microsoft Azure	Azure Kubernetes Service (AKS)					
Mirantis Cloud Platform			✓			
Nirmata		✓	✓			
Nutanix	Nutanix Karbon	Nutanix Karbon			Nutanix AHV	
OpenShift	OpenShift Dedicated and OpenShift Online		OpenShift Container Platform		OpenShift Container Platform	OpenShift Container Platform



Kubernetes Environment

Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE)	✓	✓				
oVirt					✓	
Pivotal		Enterprise Pivotal Container Service (PKS)	Enterprise Pivotal Container Service (PKS)			
Platform9	✓	✓	✓		✓	✓
Rancher		Rancher 2.x		Rancher Kubernetes Engine (RKE)		k3s
StackPoint	✓	✓				
Supergiant		✓				
SUSE		✓				
SysEleven	✓					
Tencent Cloud	Tencent Kubernetes Engine	✓	✓			✓
VEXXHOST	✓	✓				
VMware	VMware Cloud PKS	VMware Enterprise PKS	VMware Enterprise PKS	VMware Essential PKS		VMware Essential PKS
Z.A.R.V.I.S.	✓					



kubernetes
by Google

Introduction to Kubernetes



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop: Install Kubernetes



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop : Install Kubernetes

- Alternate Solution:

The screenshot shows the Katacoda Kubernetes Playground interface. At the top, there's a navigation bar with links like Apps, NMac Ked - Mac OS..., Medium, Jenkins, vagrant, Mesos, Vue, docker, NGINX, Taiwan, Kubernetes, PWA_Progressive_W..., MYSQL_Cluster, and GeneralKB. Below the bar, the Katacoda logo is displayed. The main area has a blue header with the text "Kubernetes Playground". On the left, there are two sections: "Launch Cluster" and "Health Check". Under "Launch Cluster", there's a "launch.sh" button and a note: "This will create a two node Kubernetes cluster using WeaveNet for networking.". Under "Health Check", there's a "kubectl cluster-info" button and a note: "Interested in writing your own Kubernetes scenarios and demos? Visit www.katacoda.com/teach". In the center, there's an orange "SUMMARY" button. To the right, there are two terminal windows labeled "Terminal Host 1" and "Terminal Host 2". Terminal Host 1 shows the command "master \$ launch.sh" followed by output: "Waiting for Kubernetes to start...", "Kubernetes started", and a "kubectl get nodes" command output:

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	23m	v1.8.0
node01	Ready	<none>	23m	v1.8.5

Terminal Host 2 shows the command "node01 \$ launch.sh" followed by output: "launch.sh: command not found".



Introduction to Kubernetes

- Check kubernetes version

```
kubectl get nodes -o yaml
```

```
[praparn-MacBook-Pro:~ praparn$ kubectl get nodes -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Node
  metadata:
    annotations:
      node.alpha.kubernetes.io/ttl: "0"
      volumes.kubernetes.io/controller-managed-attach-detach: "true"
    creationTimestamp: 2017-06-24T09:03:07Z
    labels:
      beta.kubernetes.io/arch: amd64
      beta.kubernetes.io/os: linux
      kubernetes.io/hostname: minikube
    name: minikube
    namespace: ""
    resourceVersion: "15676"
    selfLink: /api/v1/nodes/minikube
    uid: f03de16d-58bb-11e7-aae1-080027559511
  spec:
    externalID: minikube
          nodeInfo:
            architecture: amd64
            bootID: cdf78a28-072d-4707-8c1d-dbaa0b95fff7
            containerRuntimeVersion: docker://1.11.1
            kernelVersion: 4.9.13
            kubeProxyVersion: v1.6.0
            kubeletVersion: v1.6.0
            machineID: ef85c4ce9a23440e83266debd5cc9856
            operatingSystem: linux
            osImage: Buildroot 2017.02
            systemUUID: 801D91BA-D487-47D2-9C5D-C12F278B49C5
            kind: List
            metadata: {}
            resourceVersion: ""
            selfLink: "
```



Introduction to Kubernetes



About ▾ Projects ▾ Certification ▾ People ▾ Community ▾ Newsroom ▾

JOIN NOW



What is CNCF?

CNCF is an open source software foundation dedicated to making cloud native computing universal and sustainable. Cloud native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud native technologies enable software developers to build great products faster.

JOIN



Projects

We host and nurture components of cloud native software stacks, including Kubernetes, Prometheus and Envoy. Kubernetes and other CNCF projects are some of the **highest velocity projects** in the history of open source. We are regularly adding new projects to better support a full stack cloud native environment.



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes

- What is Orchestration (Computing)?
 - Align business request with Application/Data/Infrastructure
 - Centralize management for:
 - Resource Pool
 - Automated Workflow
 - Provisioning
 - Scale Up/Down
 - Monitoring
 - Billing
 - etc



Ref: [https://en.wikipedia.org/wiki/Orchestration_\(computing\)](https://en.wikipedia.org/wiki/Orchestration_(computing))

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes

- But why we need Container Orchestration ?
 - Production environment is cluster system
 - Microservice maintain connect was required
 - Application state-full will run on stateless architecture
 - Application scale up/Down was required
 - Many native command/shell for maintain container in production environment
 - etc



Introduction to Kubernetes



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes

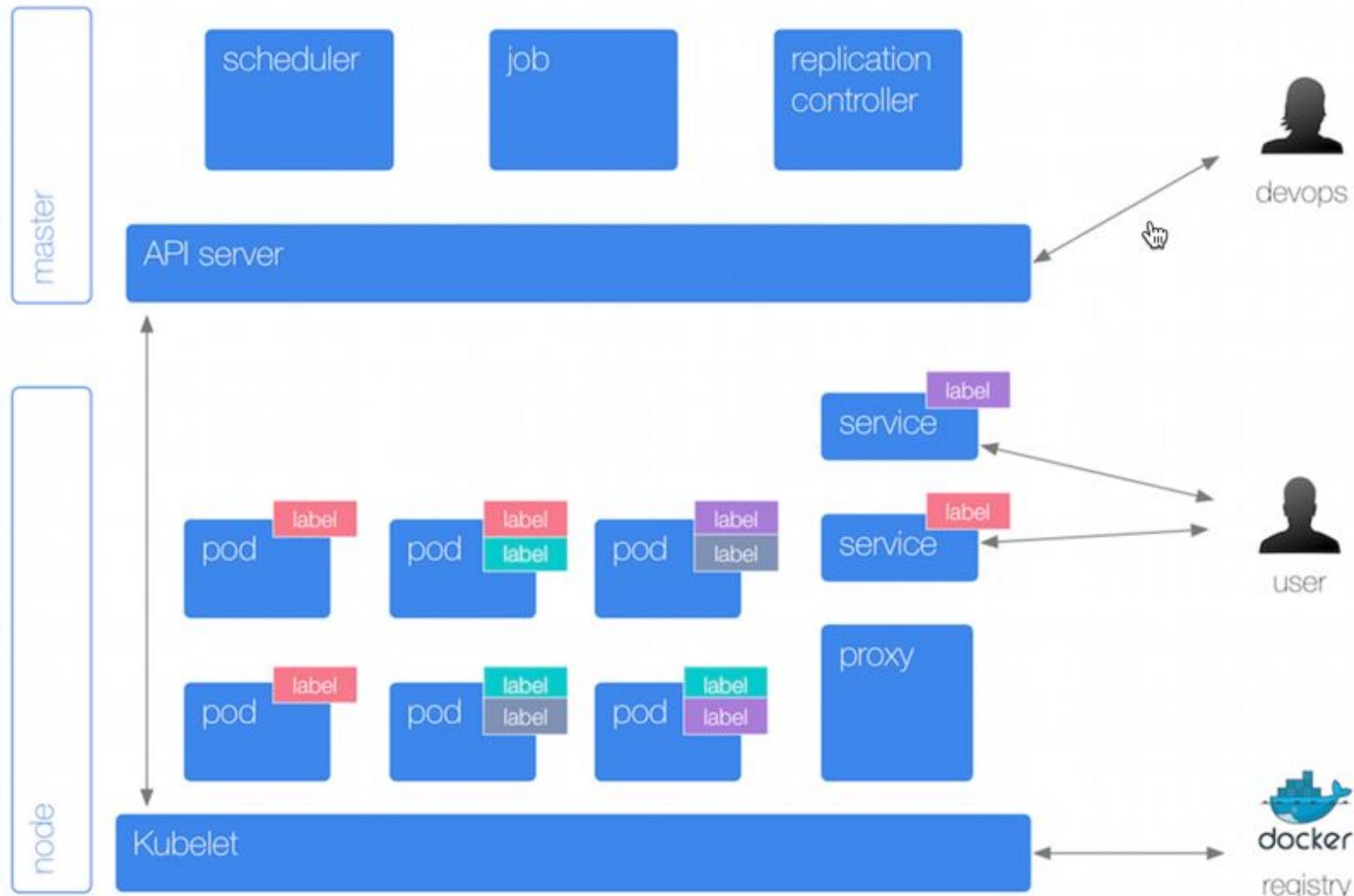


Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes



Introduction to Kubernetes

- Key Feature
 - Automatic binpacking
 - Horizontal Pod Autoscaling (HPA)
 - Automated rollouts and rollbacks
 - Storage orchestration
 - Self-healing
 - Service discovery and load balancing
 - Secret and configuration management
 - Batch execution



Introduction to Kubernetes

- Automatic binpacking
 - CPU/Memory's utilization can define on Pods (Smallest Unit of Kubernetes)
 - Schedule will select the node by ensure all resource is enough for running Pods as required
 - If reach memory limit
 - Current Pods will be terminate (Kill)
 - If restart flag was set. Kubenetes will try to restart Pods on other node
 - If reach cpu limit
 - Schedule will not kill Pods and waiting for it back to normal state

Check available node resource by command: `kubectl describe`

Non-terminated Pods:		(3 in total)			
Namespace	Name	CPU Requests	CPU Limits	Memory Requests	Memory Limits
kube-system	kube-addon-manager-minikube	5m (0%)	0 (0%)	50Mi (2%)	0 (0%)
kube-system	kube-dns-268032401-mx7s3	260m (13%)	0 (0%)	110Mi (5%)	170Mi (8%)
kube-system	kubernetes-dashboard-hdhrz	0 (0%)	0 (0%)	0 (0%)	0 (0%)

Allocated resources:

(Total limits may be over 100 percent, i.e., overcommitted.)

CPU Requests	CPU Limits	Memory Requests	Memory Limits
265m (13%)	0 (0%)	160Mi (8%)	170Mi (8%)

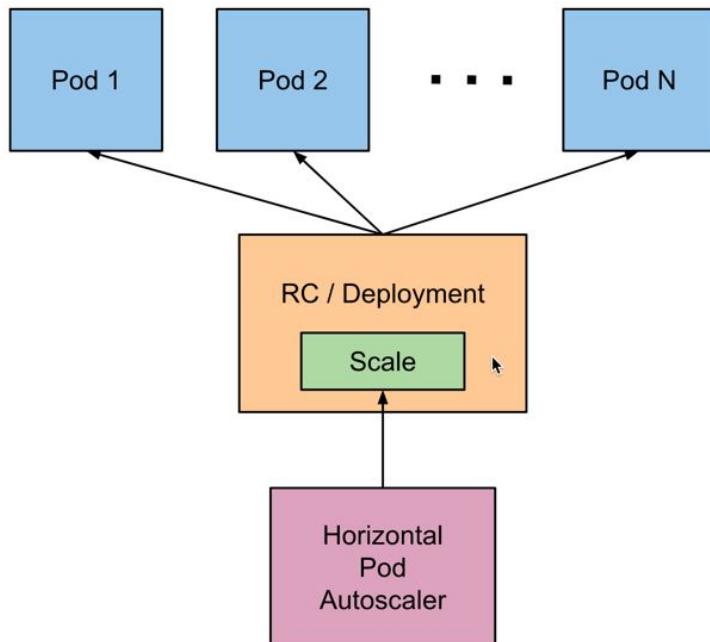
Events:

<none>



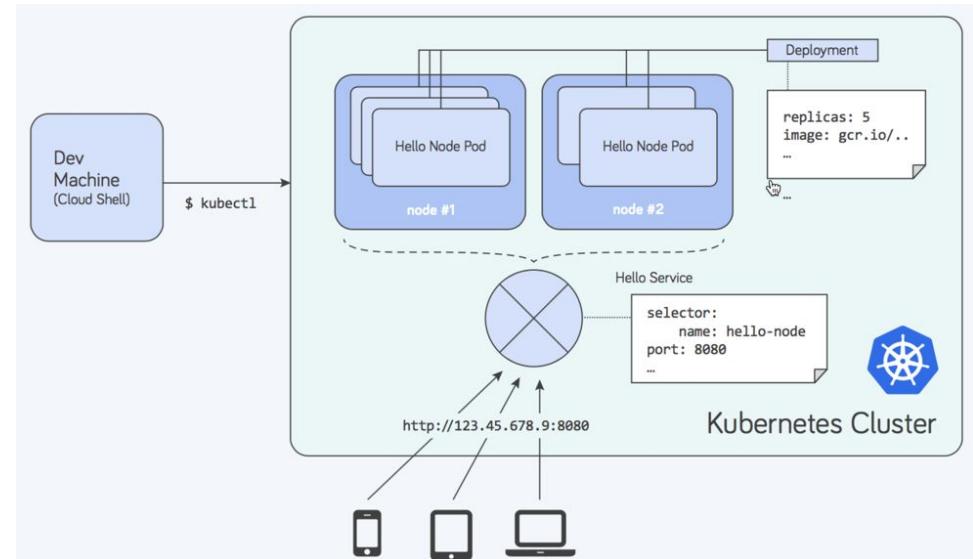
Introduction to Kubernetes

- Horizon Pods Autoscaling (hpa/vpa)
 - Talk with RC (Replication Controller) for complete task
 - Automatic scaling Pods's number base cpu's utilization
 - Support multiple criteria (metric) for scale (Alpha feature)
 - Support customize criteria (metric) for scale (Alpha feature)
 - Looping check resource's utilization every 30 seconds (default)



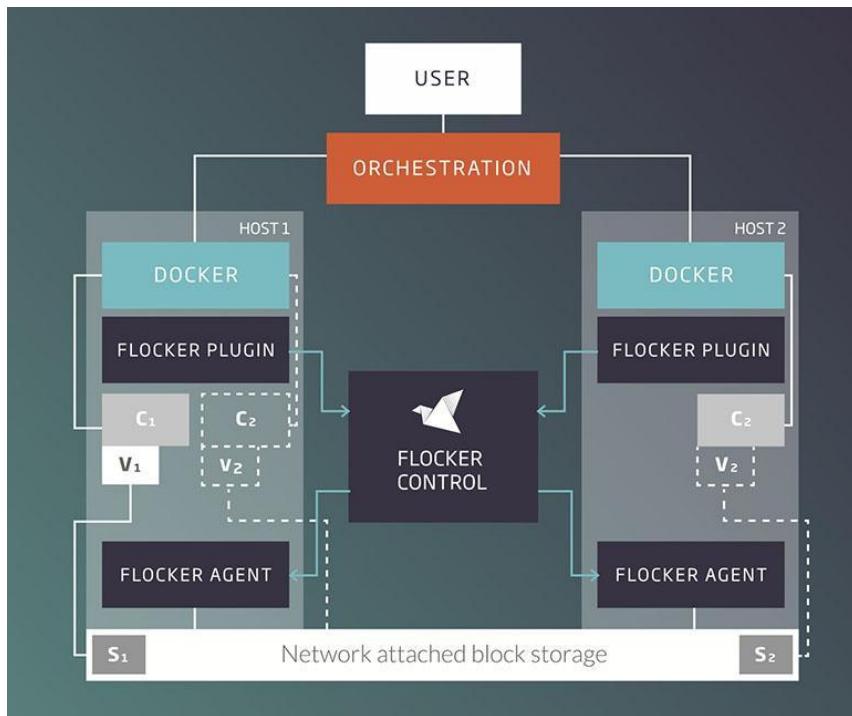
Introduction to Kubernetes

- Automated Rollout and Rollbacks
 - Update will base on Pod's template
 - Rollout existing deployment with all remain replicas number as desired
 - If rollout is success. New version will be provision until success
 - Rollback will possible with single command
 - Rollout process can pause/resume as need



Introduction to Kubernetes

- Storage Orchestrator
 - Support several storage type:
 - Local Storage
 - Network Storage (NFS, iScsi, Gluster, Ceph, Cinder, Flocker)
 - Cloud Storage (AWS, GCE, Azure Disk etc)



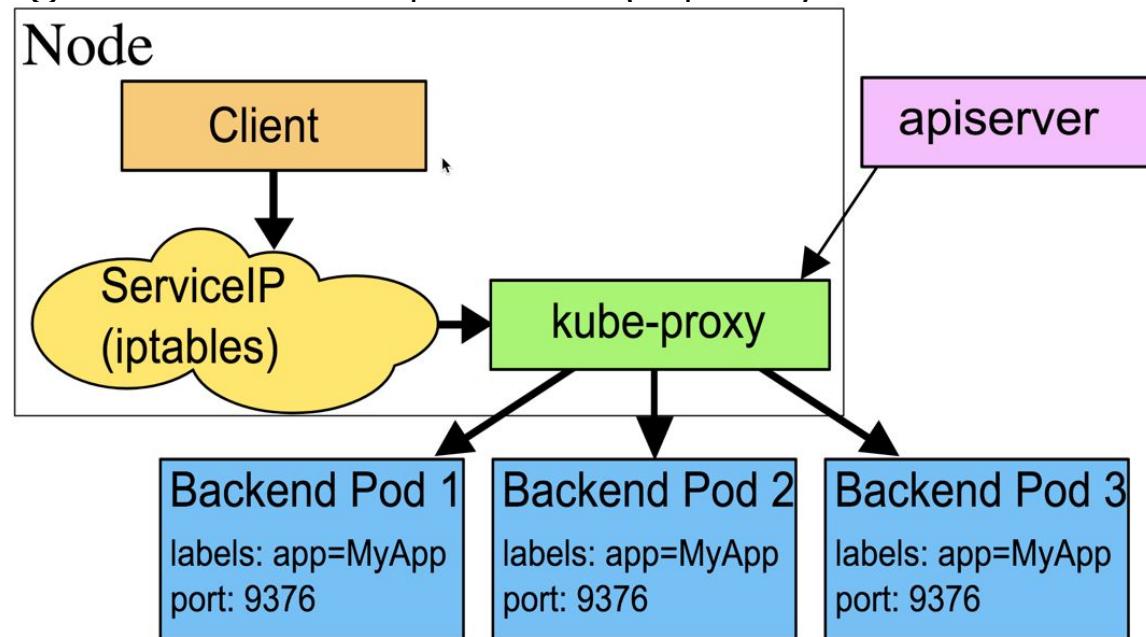
Introduction to Kubernetes

- Self-healing
 - Replication Controller (RC) will maintain unit of Pods as design (not to much (kill) and not to few (create))
 - Full-fail Pods on every failure case with automatic by system



Introduction to Kubernetes

- Service Discovery and Load Balancing
 - Service will act like “connector” for client need to connect with Pods
 - Discovery will use for service to look “Pods” by environment variable or dns service
 - Support load balancing between multiple Pods (replica)



Introduction to Kubernetes

- Secret and Configuration Management
 - Kubernetes can keep confidential data (such as username/password) for running application on encrypt format.
 - Can reference on Pods instead plan text configuration.

```
$ kubectl get secrets
NAME          TYPE        DATA  AGE
db-user-pass  Opaque      2     51s

$ kubectl describe secrets/db-user-pass
Name:         db-user-pass
Namespace:    default
Labels:       <none>
Annotations: <none>

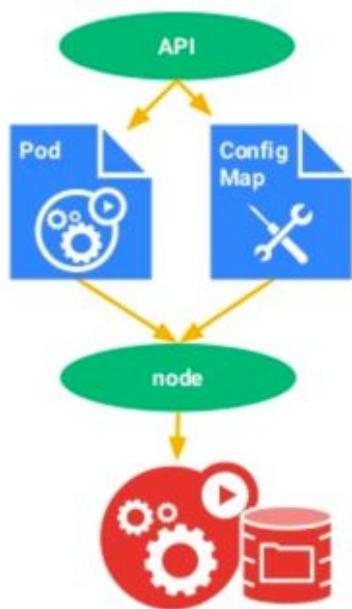
Type:         Opaque

Data
====
password.txt: 12 bytes
username.txt: 5 bytes
```



Introduction to Kubernetes

- Secret and Configuration Management
 - Sometimes we have many configuration that need to specify on each application, But it should be change every time need.
 - Idea is creating configuration file (ConfigMap) for define all configuration that reference on Pods instead



ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: dragon-config
  labels:
    environment: non-prod
data:
  dragon.how.much: very
  dragon.type: fast
```

```
apiVersion: v1
kind: Pod
metadata:
  name: dragon-pod
spec:
  containers:
    - name: dragon-container
      image: dragon-image
      env:
        - name: DRAGON_LEVEL
          valueFrom:
            configMapKeyRef:
              name: dragon-config
              key: dragon.how.much
        - name: DRAGON_TYPE
          valueFrom:
            configMapKeyRef:
              name: dragon-config
              key: dragon.type
```



Introduction to Kubernetes

- Batch Execution
 - A job will response some kind of batch execute by create special Pods (Terminate when batch complete)
 - Normally kubernetes will use job for maintain many background process for cluster system such as Replication Controller (RC) will submit job for start new Pods when existing is fail or delete.
 - Type of Job
 - Non-parallel jobs
 - Parallel jobs with fix-completion
 - Parallel jobs with work queue



System Architecture

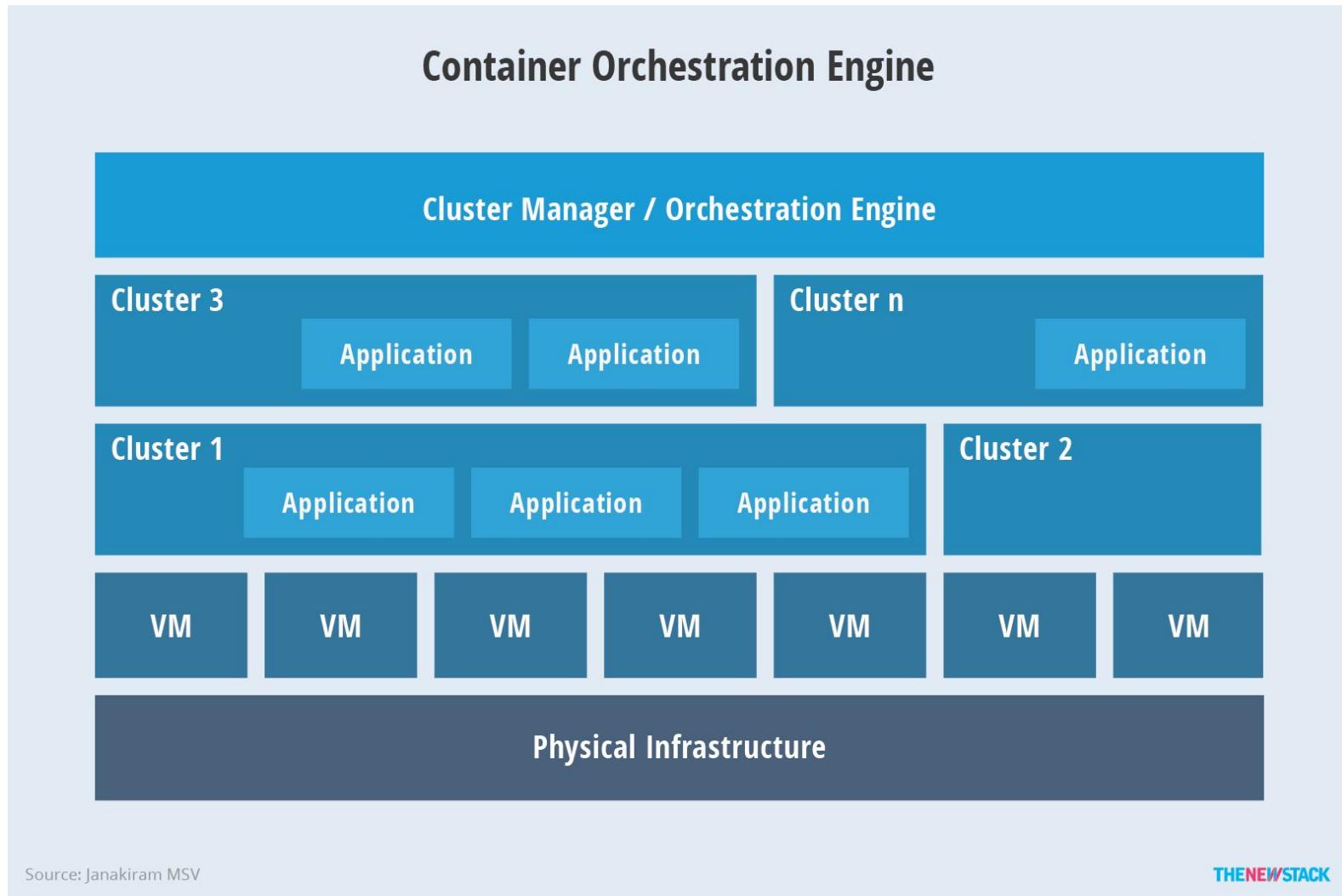


Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



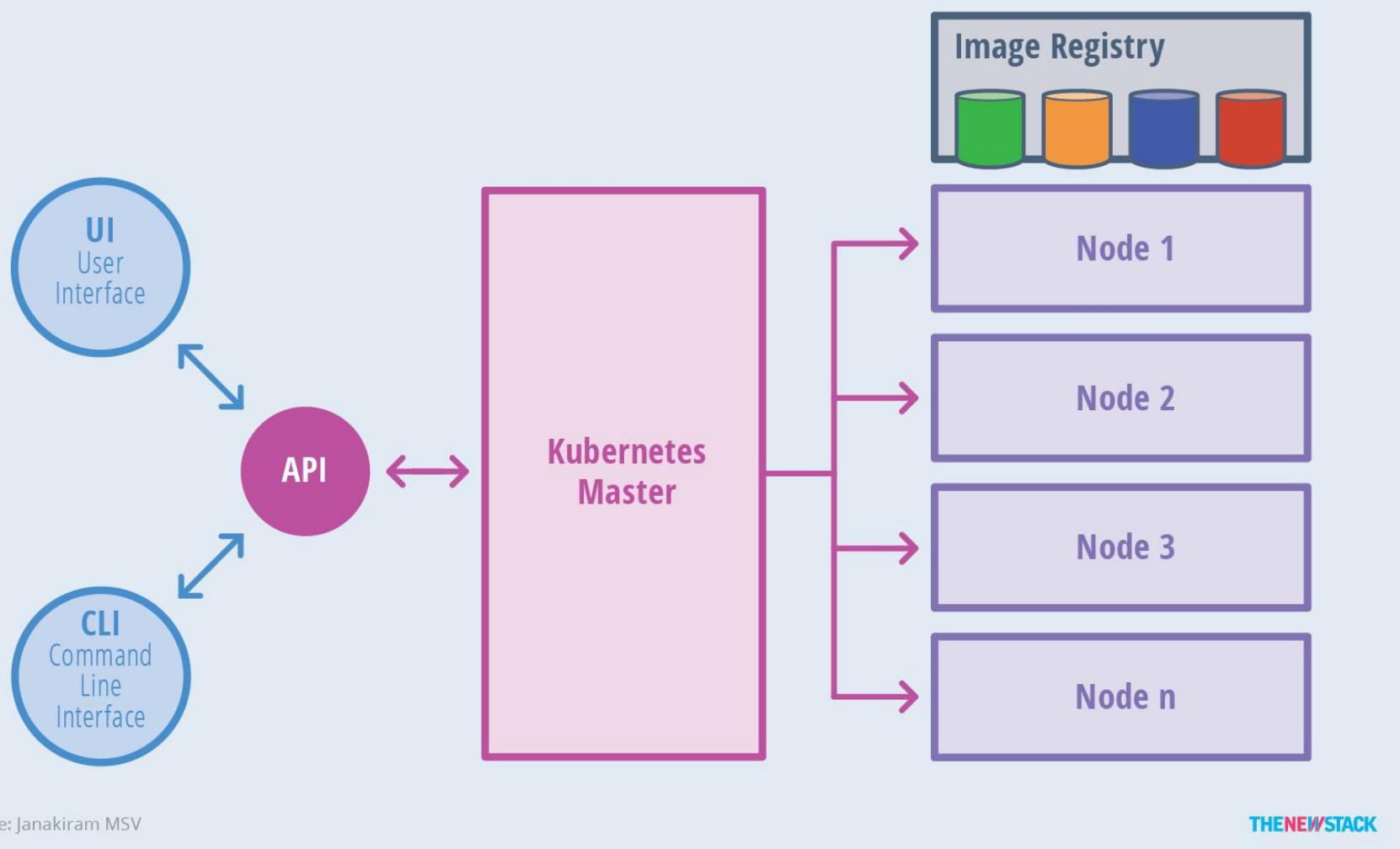
Ref: https://cdn.thenewstack.io/media/2016/11/Chart_02_Kubernetes-Architecture.png

Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



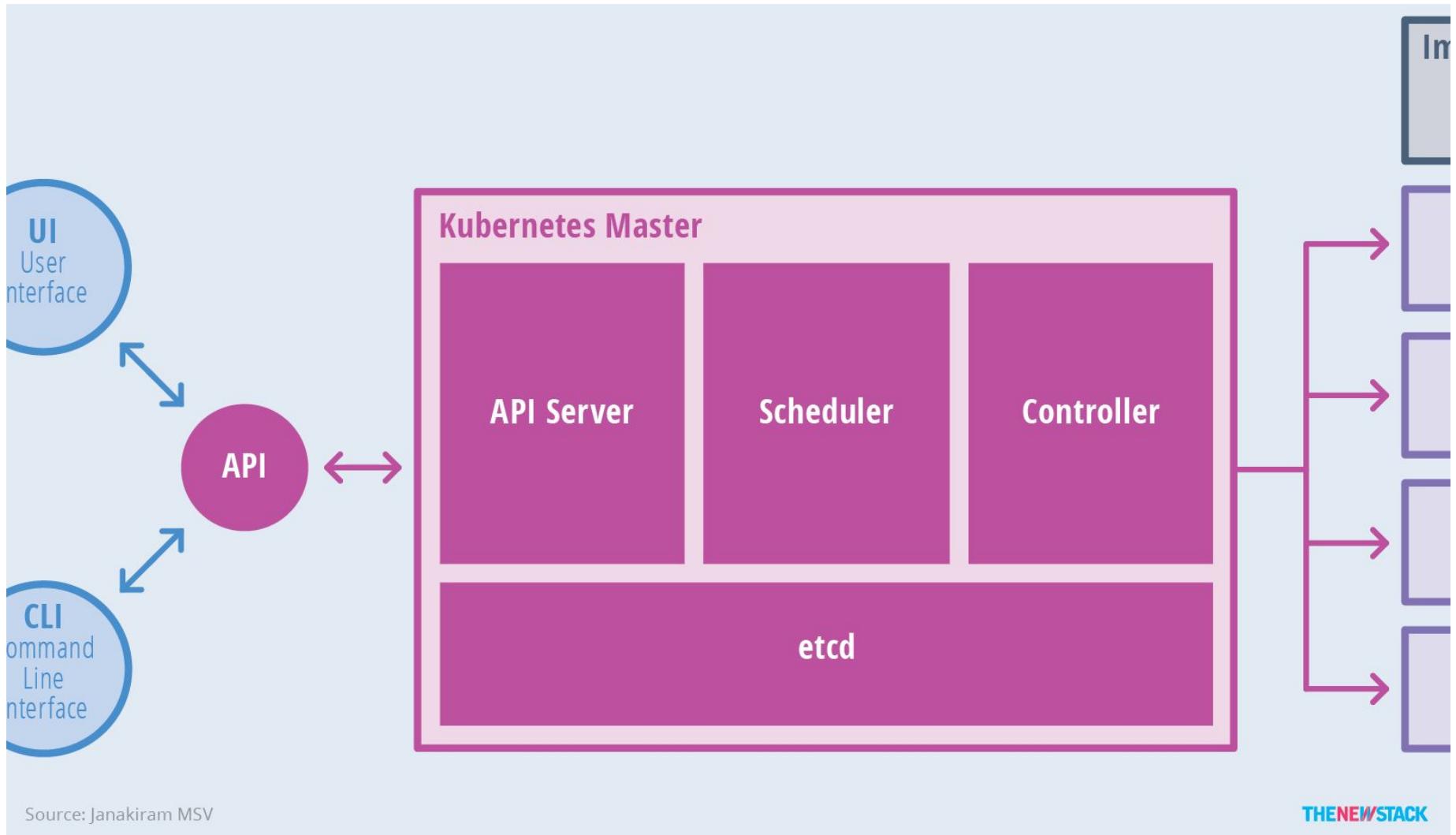
Ref: https://cdn.thenewstack.io/media/2016/11/Chart_02_Kubernetes-Architecture.png

Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



Ref: https://cdn.thenewstack.io/media/2016/11/Chart_02_Kubernetes-Architecture.png

Kubernetes: Production Workload Orchestration



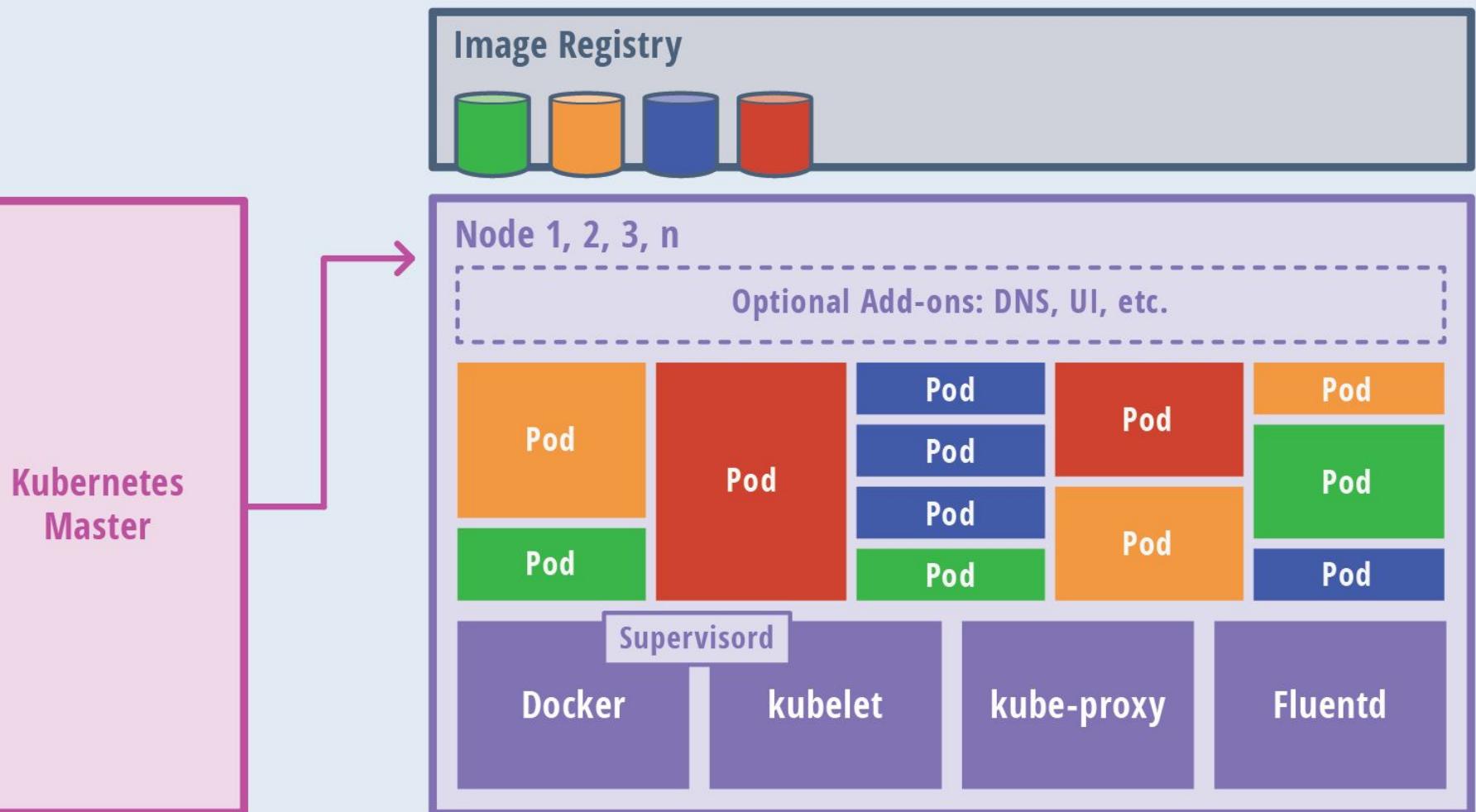
kubernetes
by Google

System Architecture

- Kubernetes Master
 - API Server: Interface for all UI to command that interact with kubernetes cluster
 - Scheduler (Job Dispatcher): Schedule all nodes's resource and dispatching Pods to nodes with match criteria
 - CPU ?
 - Memory ?
 - Affinity / Constraing ?
 - Controller and Replication Controller (RC) : Control/Coordinate all nodes for maintain server as configure on cluster system
 - Etcd (Open-source): Key-value database for keep state of nodes/Pods/Container
 - Secret: Encrypt confidential data
 - HPA (Horizon Pods Auto scaling): Scale pods with CPU criteria (major)
 - Event: Keep log and event on cluster
 - NameSpace: Limit resource quota via define namespace (cpu, memory, pods etc)



System Architecture



Source: Janakiram MSV

THE NEW STACK

Ref: https://cdn.thenewstack.io/media/2016/11/Chart_02_Kubernetes-Architecture.png

Kubernetes: Production Workload Orchestration



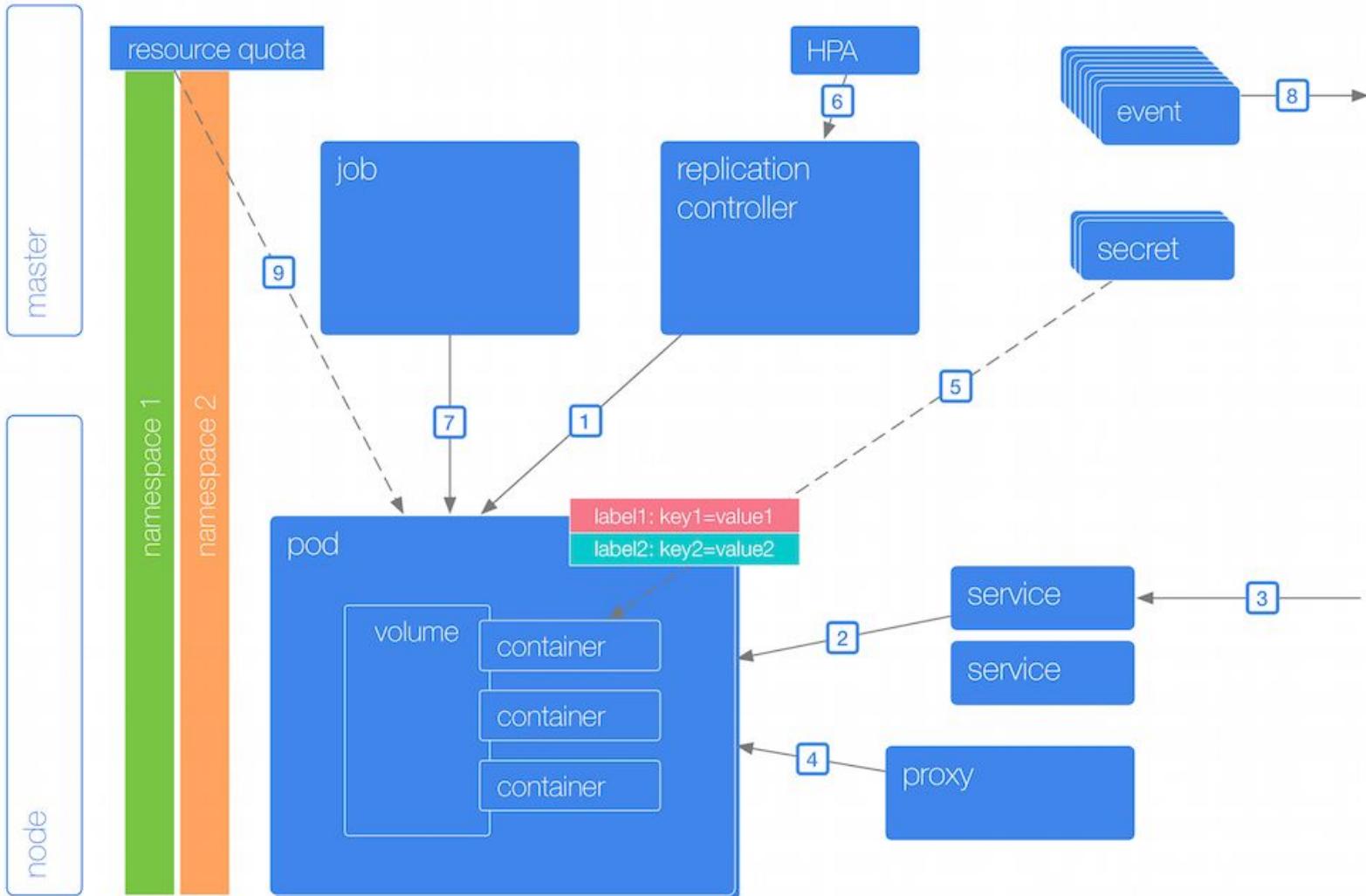
kubernetes
by Google

System Architecture

- Kubernetes Node
 - Pods: Pods is smallest deployable units of computing that can be created and manage in kubernetes (Pods != Container).
 - Docker: Docker engine for run container. Kubernetes support both docker and rkt (rocker)
 - Kubelet: Agent on node to talk with kubernetes master and send status/health of node
 - Supervisord (Docker+Kubelet): Process monitoring
 - Fluentd: Daemon for collect logs sent to kubernetes master
 - Kube-Proxy: Manage all network interface on node (Core network component)



System Architecture



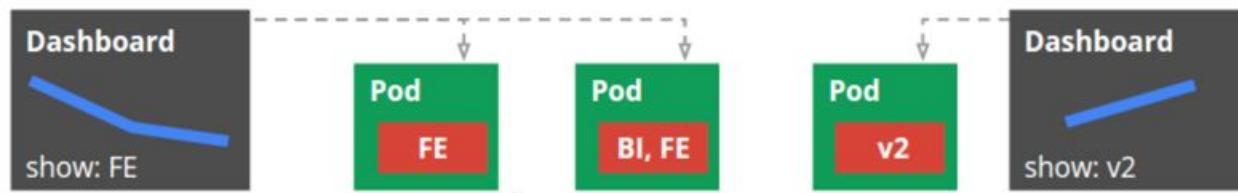
Ref: <http://k8s.info/cs.html>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



```
"labels": {  
  "key1" : "value1",  
  "key2" : "value2"  
}
```

```
"release" : "stable", "release" : "canary"
```

```
"environment" : "dev", "environment" : "qa", "environment" : "production"
```

```
"tier" : "frontend", "tier" : "backend", "tier" : "cache"
```

```
"partition" : "customerA", "partition" : "customerB"
```

```
"track" : "daily", "track" : "weekly"
```

System Architecture

GETTING STARTED

run

expose

APP MANAGEMENT

annotate

autoscale

convert

create

delete

edit

get

label

patch

replace

rolling-update

rollout

scale

set

DECLARATIVE APP MANAGEMENT

apply

WORKING WITH APPS

GETTING STARTED

This section contains the most basic commands for getting a workload running on your cluster.

- `run` will start running 1 or more instances of a container image on your cluster.
- `expose` will load balance traffic across the running instances, and can create a HA proxy for accessing the containers from outside the cluster.

Once your workloads are running, you can use the commands in the [WORKING WITH APPS](#) section to inspect them.



run

Create and run a particular image, possibly replicated.

Creates a deployment or job to manage the created container(s).

Usage

```
$ run NAME --image=image [--env="key=value"] [--port=port] [--replicas=replicas] [--dry-run=bool] [--overrides=inline-json] [--command] -- [COMMAND] [args...]
```

Flags

Start a single instance of nginx.

```
kubectl run nginx --image=nginx
```

Start a single instance of hazelcast and let the container expose port 5701 .

```
kubectl run hazelcast --image=hazelcast --port=5701
```

Ref: <https://kubernetes.io/docs/user-guide/kubectl/v1.7/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

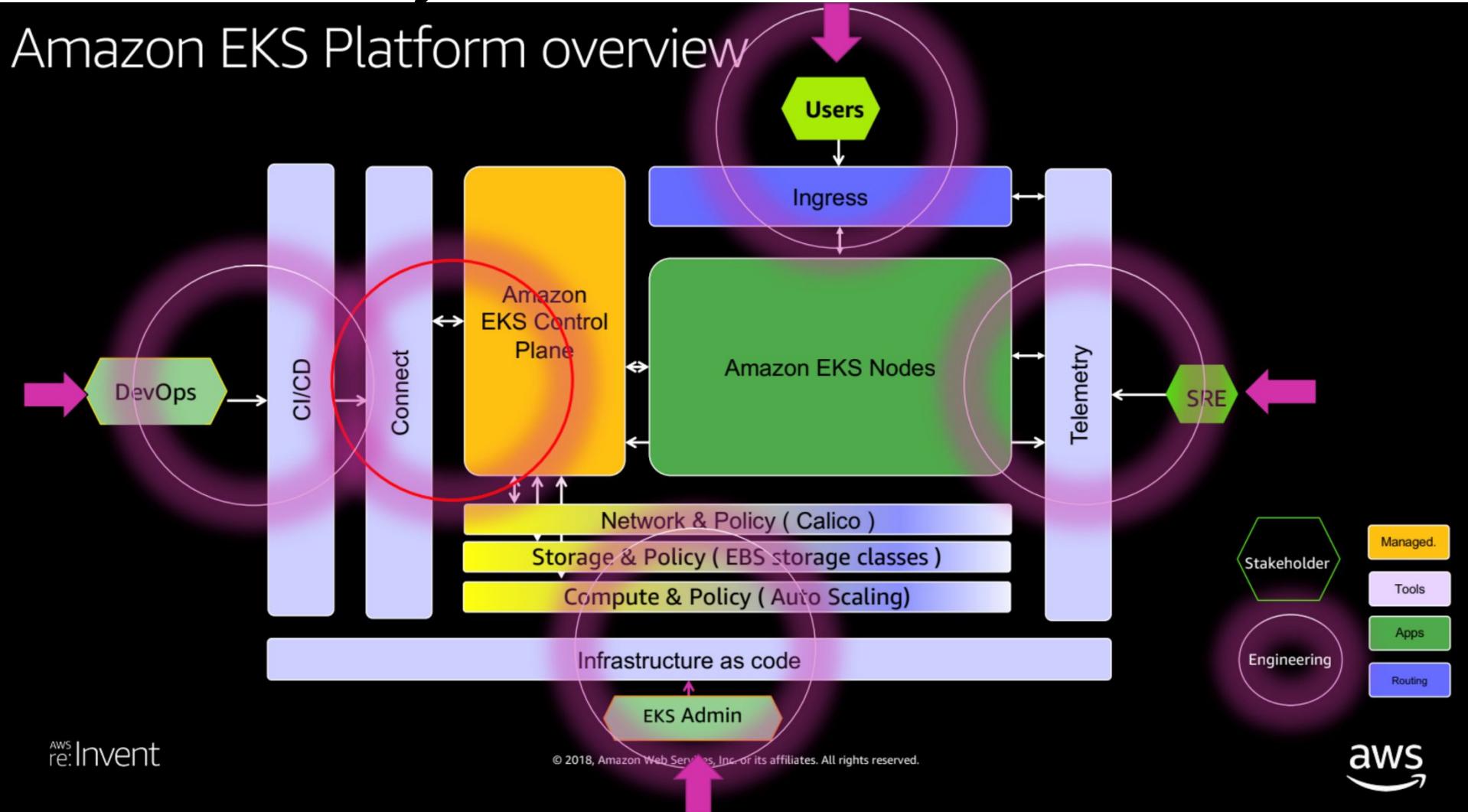
System Architecture

Topic	K8S	Docker/Swarm
Architecture	Open-system (Base on cluster manager "Borg" for support complex workload)	Swarm: Proprietary of Docker product, "Easy to use", "Extend capability of Docker in cluster"
Operation command	Almost operate by "YAML" file (Declarative Command)	Almost operate by "command" (Imperative Command)
Unit of Work	Pods (Pods >= Container)	Container
How to Identify Work	"Label operation"	Docker: By container name Swarm: By service/stack name
Level of workload management	Service Level: (Simple) Replication Level: (Auto healing) Deployment Level: (Auto healing + Roll Update)	Docker: N/A Swarm: Service Level (Snag with service/stack)
Auto scaling	HPA (Horizontal Pods Scaling) base on CPU	No
Health check	Liveness & Readiness (Multi option to check application health)	Service health only



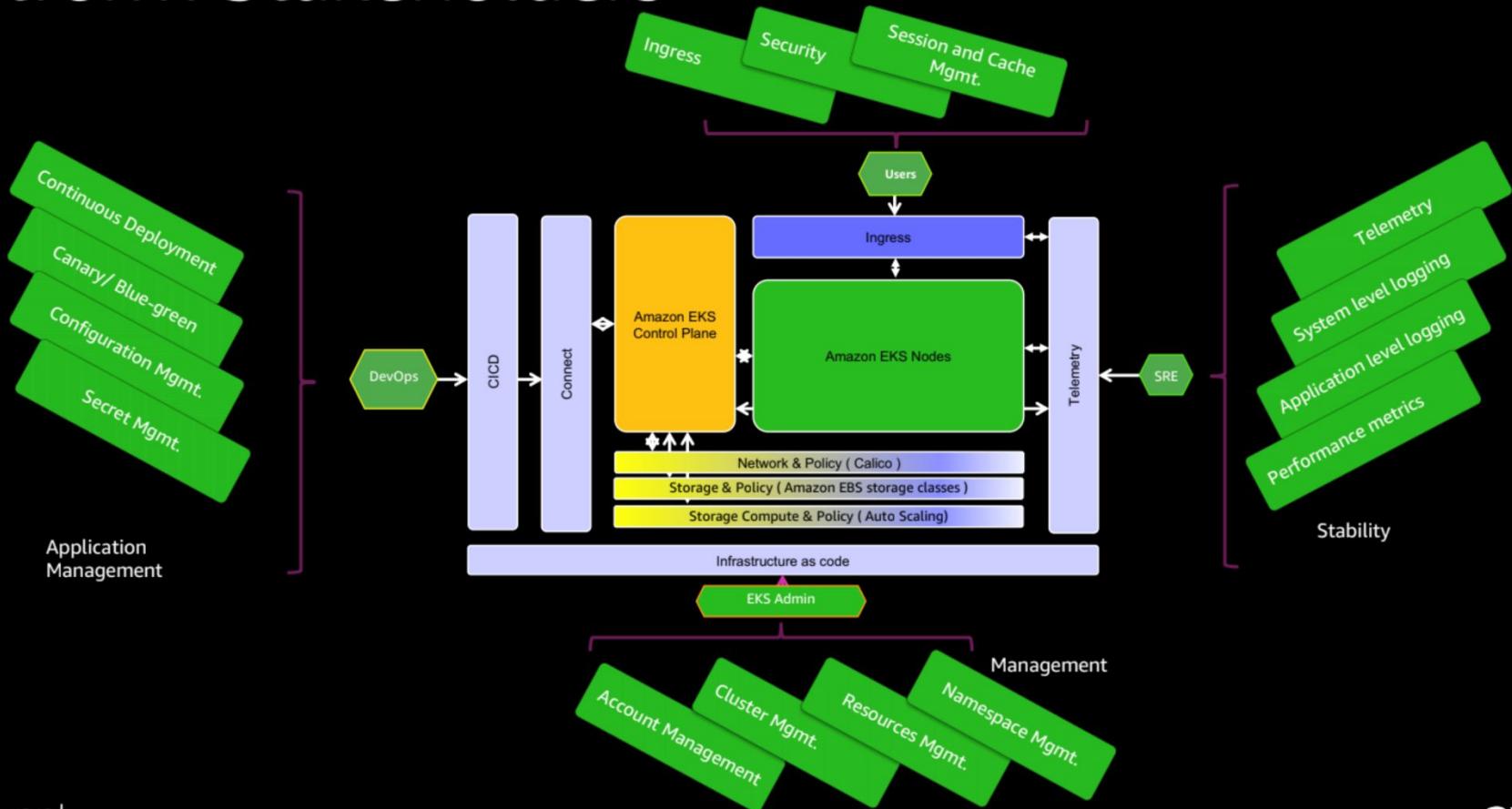
System Architecture

Amazon EKS Platform overview



System Architecture

Platform Stakeholders



© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws re:Invent



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Fundamental of Kubernetes



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Pods, Container and Services



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Pods, Container and Services

- Pods vs Container
 - Docker's view point:
 - 1 Container: 1 Application, 1 Component of Microservice
 - So for micro service we need multi container
 - Cache component
 - Web component
 - Database component
 - Etc
 - KuberneTEST's view point:
 - 1 Pods = 1 Container
 - 1 Pods = N Container (Container on the same context, Work closely)
 - So we can have 1 Pods for container more than 1 container

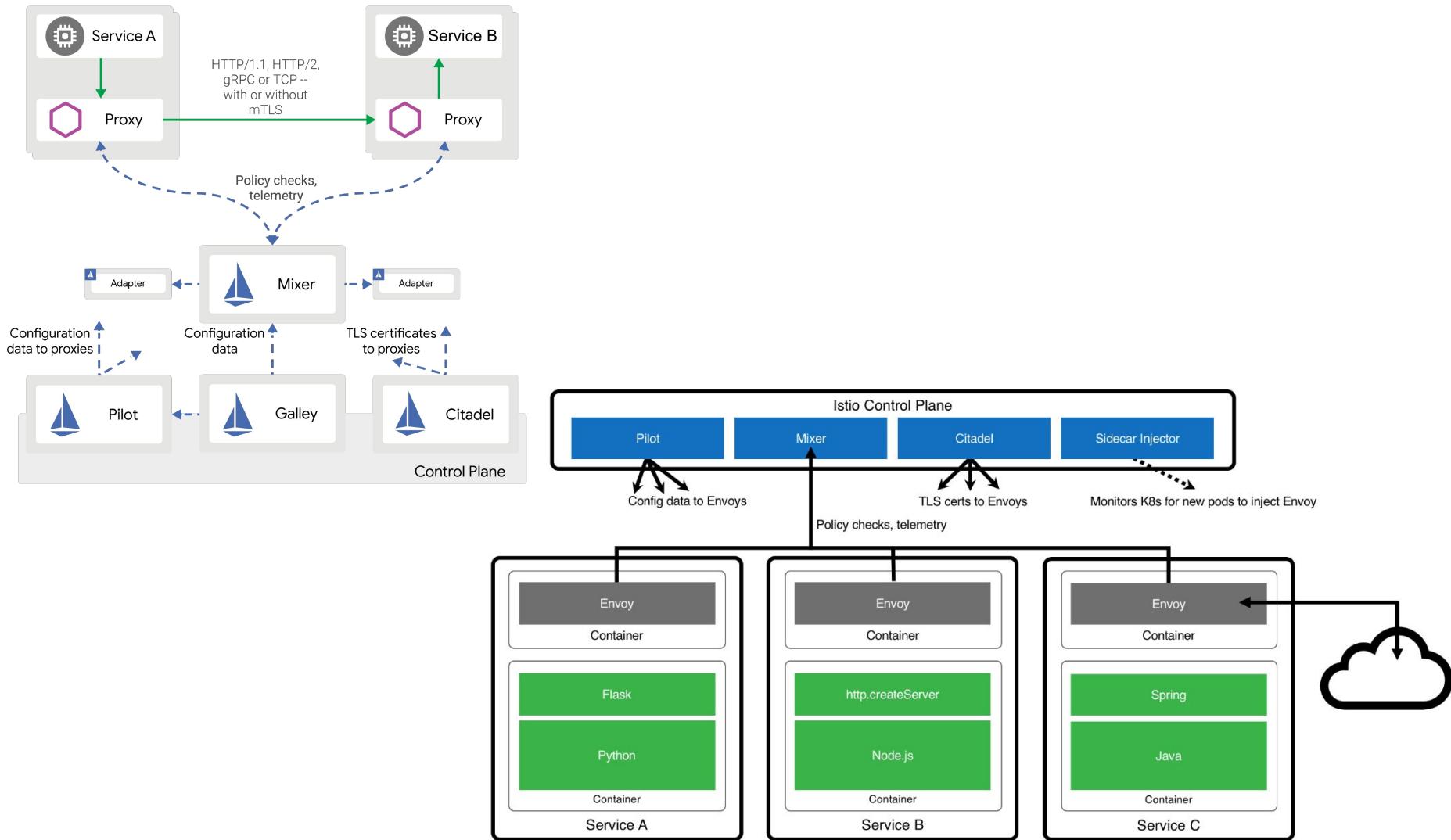


Pods, Container and Services

- Pods vs Container
 - All container on same Pods will share:
 - Process ID (PID)
 - Network access (Communicate to each other via “localhost”)
 - Internal Process Command (IPC)
 - Unix Time-Sharing (UTS)
 - Hostname
 - IP Address/Ports
 - Use Case for Multiple Pods:
 - Apache (1 Container) +Tomcat (1 Container)
 - Apache(1 Container) + PHP (1 Container)
 - Nginx (Cache: 1 Container) + Apache/PHP (1 Container)
 - Web Server (1 Container) + Data Volume(Cache: 1 Container)
 - Service Mesh (Istio / Kong etc)
 - Pods will can create replicas of 1000+ set on cluster system

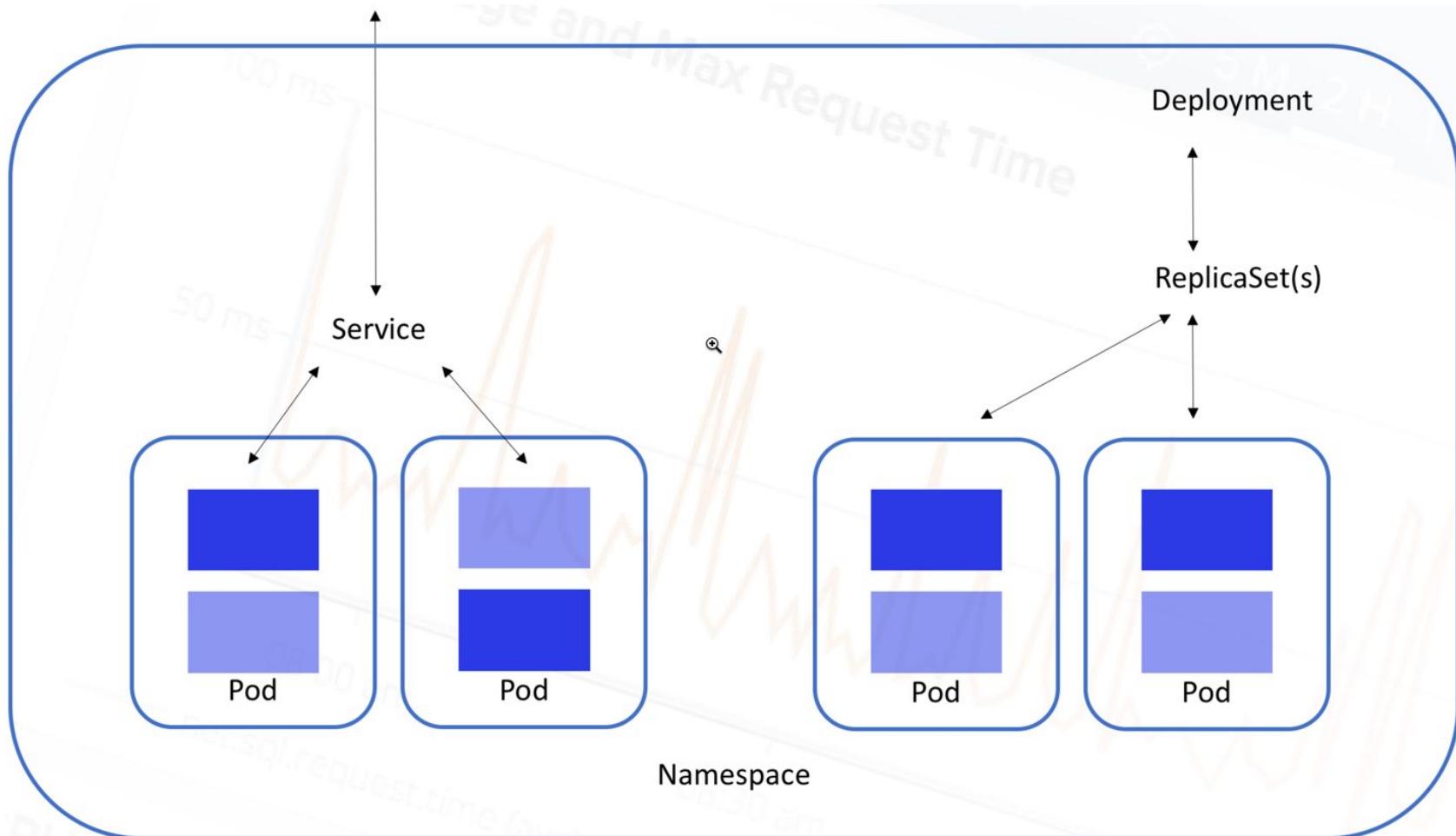


Pods, Container and Services



Pods, Container and Services

- Container/ Pods / Service / Deployment/ RS



Pods, Container and Services

- Way to deploy object in kubernetes by kubectl

Management technique	Operates on	Recommended environment	Supported writers	Learning curve
Imperative commands	Live objects	Development projects	1+	Lowest
Imperative object configuration	Individual files	Production projects	1	Moderate
Declarative object configuration	Directories of files	Production projects	1+	Highest

- Imperative commands:
 - Ex: kubectl <action> <type/name> <option>
 - Ex: kubectl run webtest --image labdocker/nginx:latest
- Imperative object configuration
 - Ex: kubectl <action> -f <YAML file>
 - Ex: kubectl create -f nginx.yml
 - Ex: kubectl replace -f nginx_update.yml
- Declarative object configuration
 - Ex: kubectl apply -f <directory>

```
apiVersion: "v1"
kind: Pod
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  containers:
    - name: webtest
      image: labdocker/cluster:webservicelite
      ports:
        - containerPort: 5000
          protocol: TCP
```



Pods, Container and Services

- Imperative commands:
 - “kubectl run” (Pods + Deployment+ RC (Replicas))

```
kubectl run -image=<image name> <option>
```

- Option:
 - --env="key=value"
 - --port=port
 - --replicas=<number of Pods replicas>
 - --overrides=<json>
 - --labels=<label>
 - Etc

- Example:

```
[praparns-MacBook-Pro:~ praparn$ kubectl run webtest --image=labdocker/cluster:webservicelite --port=5000  
deployment "webtest" created
```

- **DEPRECATED !!!**



Pods, Container and Services

- Imperative commands:

- kubectl expose (Service)

```
kubectl expose deployment <name> <option>
```

- Option:

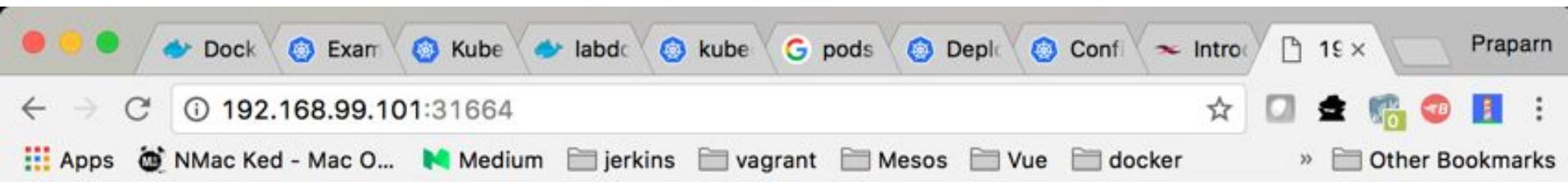
- --name=name
- --port=port for service
- --target-port=port of deployment set
- --type=<NodePort/ClusterIP/LoadBalance>
- --protocol=<TCP/UAT etc>
- Etc

```
[praparns-MacBook-Pro:~ praparn$ kubectl expose deployment webtest --target-port=5000 --type=NodePort
service "webtest" exposed
[praparns-MacBook-Pro:~ praparn$ kubectl get svc webtest
NAME      CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
webtest   10.0.0.96    <nodes>        5000:31664/TCP  8s
```



Pods, Container and Services

- Imperative commands:



Welcome Page from Container Python Lab

Checkpoint Date/Time: Sun Jul 2 04:20:03 2017

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Pods, Container and Services

- Imperative object configuration

```
kubectl create -f <Filename>
```

- Create Pods (YAML)

```
apiVersion: "v1"
kind: Pod
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  containers:
    - name: webtest
      image: labdocker/cluster:webservicelite
      ports:
        - containerPort: 5000
          protocol: TCP
```

- Create Service (YAML)

```
apiVersion: v1
kind: Service
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  selector:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
  type: NodePort
  ports:
    - port: 5000
      name: http
      targetPort: 5000
      protocol: TCP
```



Pods, Container and Services



JSON Formatter | My Ip | Search | Recent Links | Sample | More ▾ | Sign in | (?)

YAML Converter

Save & Share

YAML Input

```
apiVersion: "v1"
kind: Pod
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  containers:
    - name: webtest
      image: labdocker/cluster:webservicelite
      ports:
        - containerPort: 5000
          protocol: TCP
  nodeSelector:
    kubernetes.io/hostname: kubernetes-1
```

Result : YAML TO JSON

```
{ "apiVersion": "v1", "kind": "Pod", "metadata": { "name": "webtest", "labels": { "name": "web", "owner": "Praparn_L", "version": "1.0", "module": "WebServer", "environment": "development" } }, "spec": { "containers": [ { "name": "webtest", "image": "labdocker/cluster:webservicelite", "ports": [ { "containerPort": 5000, "protocol": "TCP" } ] }, "nodeSelector": { "kubernetes.io/hostname": "kubernetes-1" } ] }
```

Conversion Options

- Load Url
- Browse
- YAML TO JSON**
- YAML TO XML
- YAML TO CSV
- Validate

Feedback

Seen this ad multiple times Not interested in this ad ➤

Download



Pods, Container and Services

https://kubeyaml.com

Apps NMac Ked - Mac... M Medium F Infra NOVA jerkins vagrant Mesos Vue docker Taiwan NGINX Kubernetes PWA_Progressive... MYSQL_Cluster GeneralKB Nodejs mongodb Other Bookmarks

Kube YAML

Validating kubernetes objects since 2018

How to use this webpage

Paste your Kubernetes YAML in the text area below and see which versions it is valid for.

⚠ Please only enter one YAML document at a time, this will not split YAML documents with `---` correctly.

Example YAML	Validate	1.14 Errors	1.13 Errors	1.12 Errors	1.11 Errors
<pre>apiVersion: "v1" kind: Pod metadata: name: webtest labels: name: web owner: Praparn_L version: "1.0" module: WebServer environment: development spec: containers: - name: webtest image: labdocker/cluster:webservicelite ports: - containerPort: 5000 protocol: TCP</pre>	<button>Validate</button>	✓ No errors			



Pods, Container and Services

- Imperative object configuration

```
kubectl create -f <Filename>
```

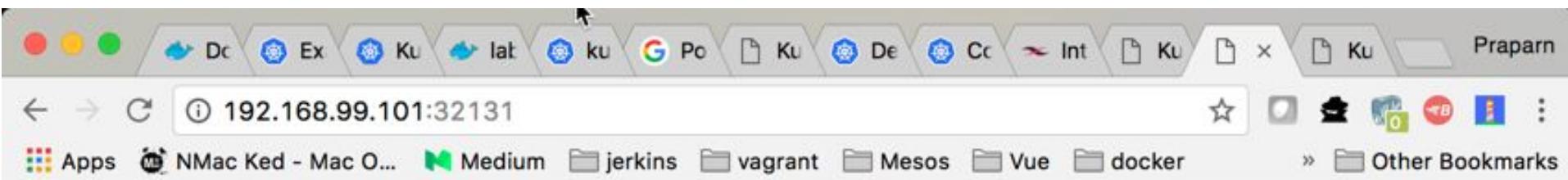
```
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ ls -lh
total 24
-rw-r--r--@ 1 praparn  staff  550B Jul  2 09:29 instruction.txt
-rw-r--r--@ 1 praparn  staff  321B Jul  2 12:52 webtest_pod.yml
-rw-r--r--@ 1 praparn  staff  393B Jul  2 12:52 webtest_svc.yml
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ kubectl create -f webtest_pod.yml
pod "webtest" created
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ kubectl get pods
NAME      READY     STATUS    RESTARTS   AGE
webtest   1/1      Running   0          4m
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ kubectl create -f webtest_svc.yml
service "webtest" created
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ kubectl get svc
NAME      CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes  10.0.0.1    <none>       443/TCP       4d
webtest    10.0.0.87    <nodes>      5000:32131/TCP  4m
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ curl http://192.168.99.101:32131
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Jul  2 06:04:08 2017
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$
```



Pods, Container and Services

- Imperative object configuration

```
kubectl create -f <Filename>
```



Pods, Container and Services

- Check log on container:

```
kubectl logs <Pods name> -c <container name>
```

```
praparns-MacBook-Pro:singlecontainer praparn$ kubectl logs webtest -c webtest
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 985-709-866
```

- Shell inside container:

```
kubectl exec -it <Pods name> -c <container name> sh
```

```
praparns-MacBook-Pro:singlecontainer praparn$ kubectl exec -it webtest -c webtest sh
/usr/src/app # ls -lh
total 36
-rw-r--r-- 1 root root 482 Jul 1 03:42 docker-compose.yml
-rw-r--r-- 1 root root 345 Jul 1 02:10 dockerfile_nginx
-rw-r--r-- 1 root root 80 Jun 30 17:32 dockerfile_python
-rw-r--r-- 1 root root 88 Jul 2 03:29 dockerfile_python_lite
-rw-r--r-- 1 root root 2.7K Jul 2 03:01 main.py
-rw-r--r-- 1 root root 291 Jul 2 03:46 mainlite.py
-rw-r--r-- 1 root root 1.2K Jul 1 03:51 nginx.conf
-rw-r--r-- 1 root root 24 Jun 18 04:33 requirements.txt
-rw-r--r-- 1 root root 6 Jul 2 03:32 requirementslite.txt
/usr/src/app # hostname
webtest
/usr/src/app #
```



Pods, Container and Services

- Check detail property of Pods/Service

```
kubectl describe <Pods/SVC/etc> <Name>
```

```
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ kubectl describe pods webtest
Name:           webtest
Namespace:      default
Node:          minikube/192.168.99.101
Start Time:    Sun, 02 Jul 2017 12:56:50 +0700
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
Annotations:   <none>
Status:        Running
IP:            172.17.0.4
Controllers:   <none>
Containers:
  webtest:
    Container ID:    docker://04b9cdfdd6451a4e78c873f704fb4d35de7bba082343d0773c7ea0ebbb3f03a
    Image:          labdockerc/cluster:webservicelite
    Image ID:       docker://sha256:837c8f41c918ede06705f9b554b6120fdb654cc2a8eb7eec74bc383b09865f2b
    Port:          5000/TCP
    State:         Running
    Started:      Sun, 02 Jul 2017 12:56:51 +0700
    Ready:         True
    Restart Count: 0
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-vxm58 (ro)
Conditions:
  Type      Status
  Initialized  True
  Ready      True
  PodScheduled  True
Volumes:
  default-token-vxm58:
    Type:  Secret (a volume populated by a Secret)
    SecretName: default-token-vxm58
    Optional:  false
QoS Class:  BestEffort
Node-Selectors: <none>
Tolerations: <none>
Events:
FirstSeen  LastSeen  Count  From             SubObjectPath          Type   Reason   Message
-----  -----  -----  -----  -----  -----
14m        14m       1  default-scheduler   spec.containers{webtest}  Normal  Scheduled  Successfully assigned webtest to minikube
14m        14m       1  kubelet, minikube   spec.containers{webtest}  Normal  Pulled    Container image "labdockerc/cluster:webservicelite" already
y present on machine
14m        14m       1  kubelet, minikube   spec.containers{webtest}  Normal  Created   Created container with id 04b9cdfdd6451a4e78c873f704fb4d3
5de7bba082343d0773c7ea0ebbb3f03a
14m        14m       1  kubelet, minikube   spec.containers{webtest}  Normal  Started   Started container with id 04b9cdfdd6451a4e78c873f704fb4d3
5de7bba082343d0773c7ea0ebbb3f03a
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$
```



Pods, Container and Services

- Check detail property of Pods/Service

```
kubectl describe <Pods/SVC/etc> <Name>
```

```
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ kubectl describe svc webtest
Name:           webtest
Namespace:      default
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
Annotations:    <none>
Selector:       environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
Type:           NodePort
IP:             10.0.0.87
Port:           http    5000/TCP
NodePort:       http    32131/TCP
Endpoints:     172.17.0.4:5000
Session Affinity: None
Events:         <none>
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ █
```

Pods, Container and Services

- Check overall of Pods/Service

```
kubectl get <Pods/SVC/etc>
```

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME      READY    STATUS    RESTARTS   AGE
maindb    1/1      Running   0          18m
web       3/3      Running   0          16m
praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes  10.0.0.1    <none>        443/TCP       4d
maindb     10.0.0.134   <none>        3306/TCP      17m
web        10.0.0.69    <nodes>       5000:30661/TCP,80:30500/TCP  16m
praparns-MacBook-Pro:multicontainer praparn$ █
```

- Remove Pod/Service

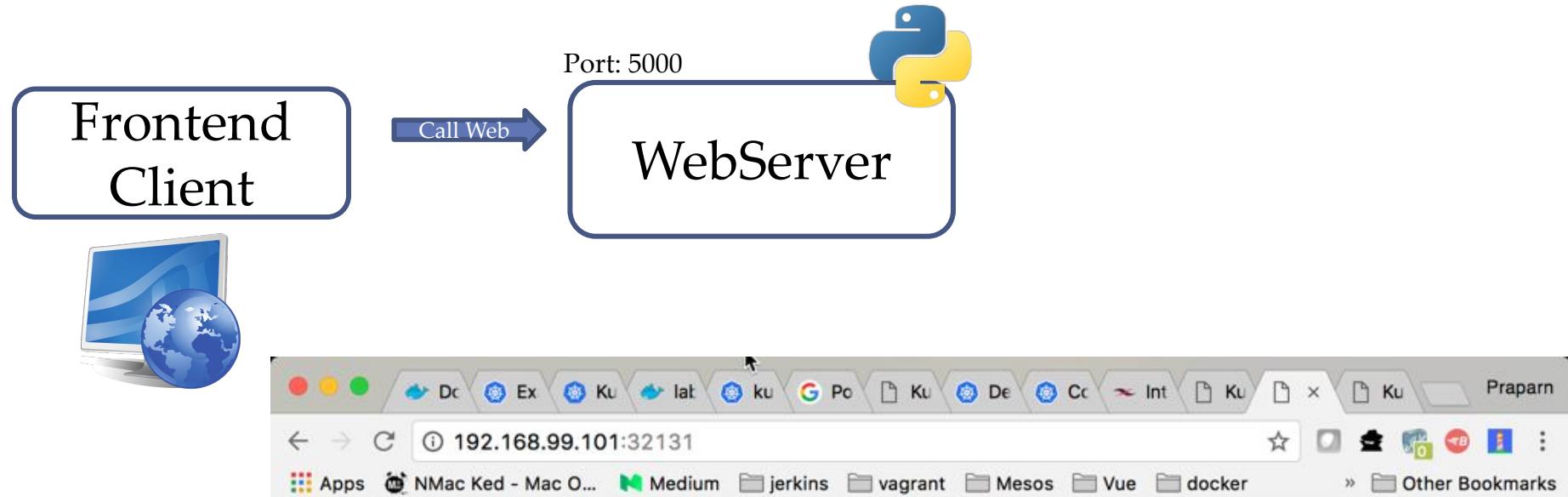
```
kubectl delete -f <Filename>
```

```
praparns-MacBook-Pro:singlecontainer praparn$ kubectl delete -f webtest_svc.yml
service "webtest" deleted █
praparns-MacBook-Pro:singlecontainer praparn$ kubectl delete -f webtest_pod.yml
pod "webtest" deleted
```



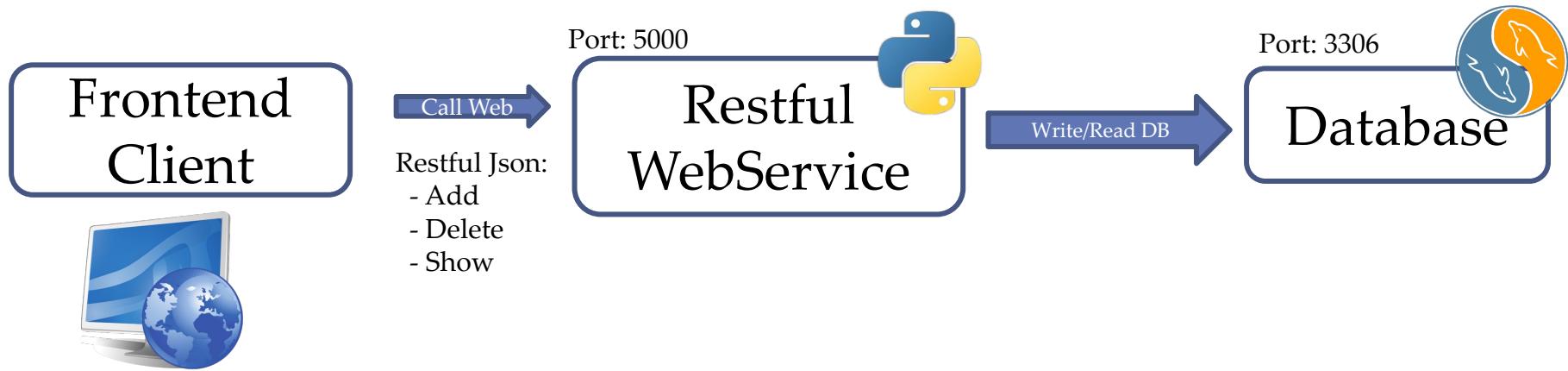
Workshop: Pods & Service

- Part 1: Deploy simple web pods with single container



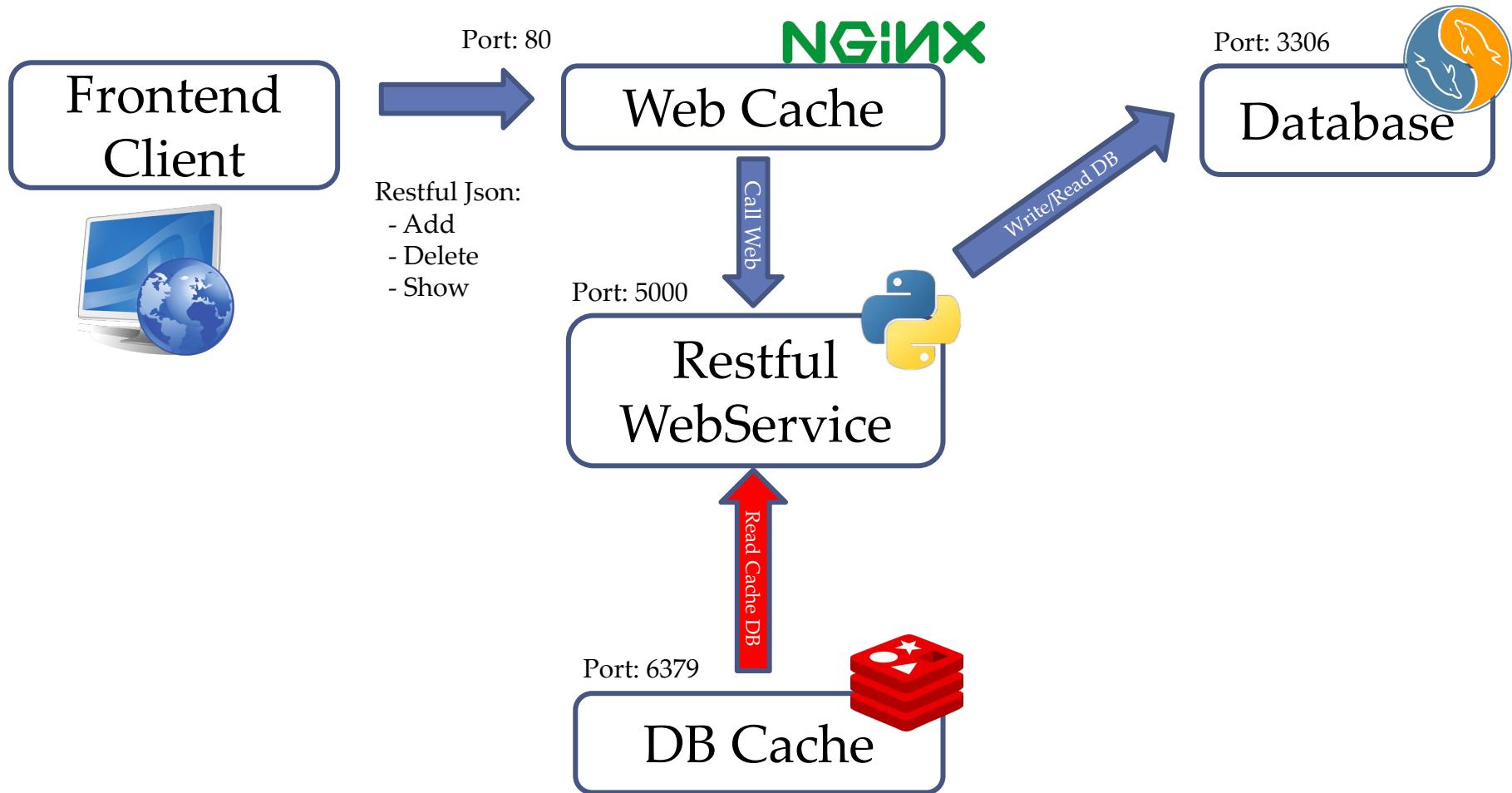
Pods, Container and Service

- Example Case: Basic Restful API



Pods, Container and Service

- Example Case: High I/O Restful API



Pods, Container and Service

- Restful WebService
 - /init

```
@app.route('/init')
def init():
    MAIN_DB.execute("DROP DATABASE IF EXISTS ACCTABLE")
    MAIN_DB.execute("CREATE DATABASE ACCTABLE")
    MAIN_DB.execute("USE ACCTABLE")
    sql = """CREATE TABLE users (
        ID int,
        USER char(30),
        DESRIPE char(250)
    )"""
    MAIN_DB.execute(sql)
    db.commit()
    return "##### Database Create New Account Table Done #####"
```

- /insertuser

```
@app.route("/users/insertuser", methods=['POST'])
def add_users():
    req_json = request.get_json()
    MAIN_DB.execute("INSERT INTO ACCTABLE.users (ID, USER, DESRIPE) VALUES (%s,%s,%s)", (req_json['uid'], req_json['user'], req_json['desripe']))
    #curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "desripe":"System Engineer"}' http://<IP>
    db.commit()
    return Response("##### Record was added #####", status=200, mimetype='application/json')
```



Pods, Container and Service

- Restful WebService
 - /removeuser/<uid>

```
@app.route('/users/removeuser/<uid>')
def remove_users(uid):
    hash = hashlib.sha224(str(uid)).hexdigest()
    key = "sql_cache:" + hash
    MAIN_DB.execute("DELETE FROM ACCTABLE.users WHERE ID =" + str(uid))
    db.commit()
    #curl http://<IP Host>:<Port>/users/removeuser/<uid>
    if (CACHE_DB.get(key)):
        CACHE_DB.delete(key)
        return Response("##### Record was deleted (Both Database Cache) #####", status=200, mimetype='application/json')
    else:
        return Response("##### Record was deleted #####", status=200, mimetype='application/json')
```



Pods, Container and Service

- Restful WebService
 - /users/<uid>

```
@app.route('/users/<uid>')
def get_users(uid):
    hash = hashlib.sha224(str(uid)).hexdigest()
    key = "sql_cache:" + hash
    #curl http://<IP Host>:<Port>/users/<uid>
    if (CACHE_DB.get(key)):
        return CACHE_DB.get(key) + "(Database Cache)"
    else:
        MAIN_DB.execute("select USER from ACCTABLE.users where ID=" + str(uid))
        data = MAIN_DB.fetchone()
        if data:
            CACHE_DB.set(key,data[0])
            CACHE_DB.expire(key, 36);
            return CACHE_DB.get(key)
        else:
            return "##### Record not found #####"
```

Pods, Container and Service

- Web Cache

```
http {
    client_max_body_size 500M;
    client_body_timeout 3000s;
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
    '$status $body_bytes_sent "'.$http_referer"
    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    tcp_nopush on;

    keepalive_timeout 65;
    [REDACTED]
    gzip on;

    include /etc/nginx/conf.d/*.conf;
server {
    listen 80;
    client_body_buffer_size 50M;
    index index.html      index.htm;
    location / {
        proxy_pass http://webservice:5000;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;
        proxy_redirect off;
        proxy_buffering off;
        proxy_set_header      Host          $host;
        proxy_set_header      X-Real-IP     $remote_addr;
        proxy_set_header      X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```



Pods, Container and Service

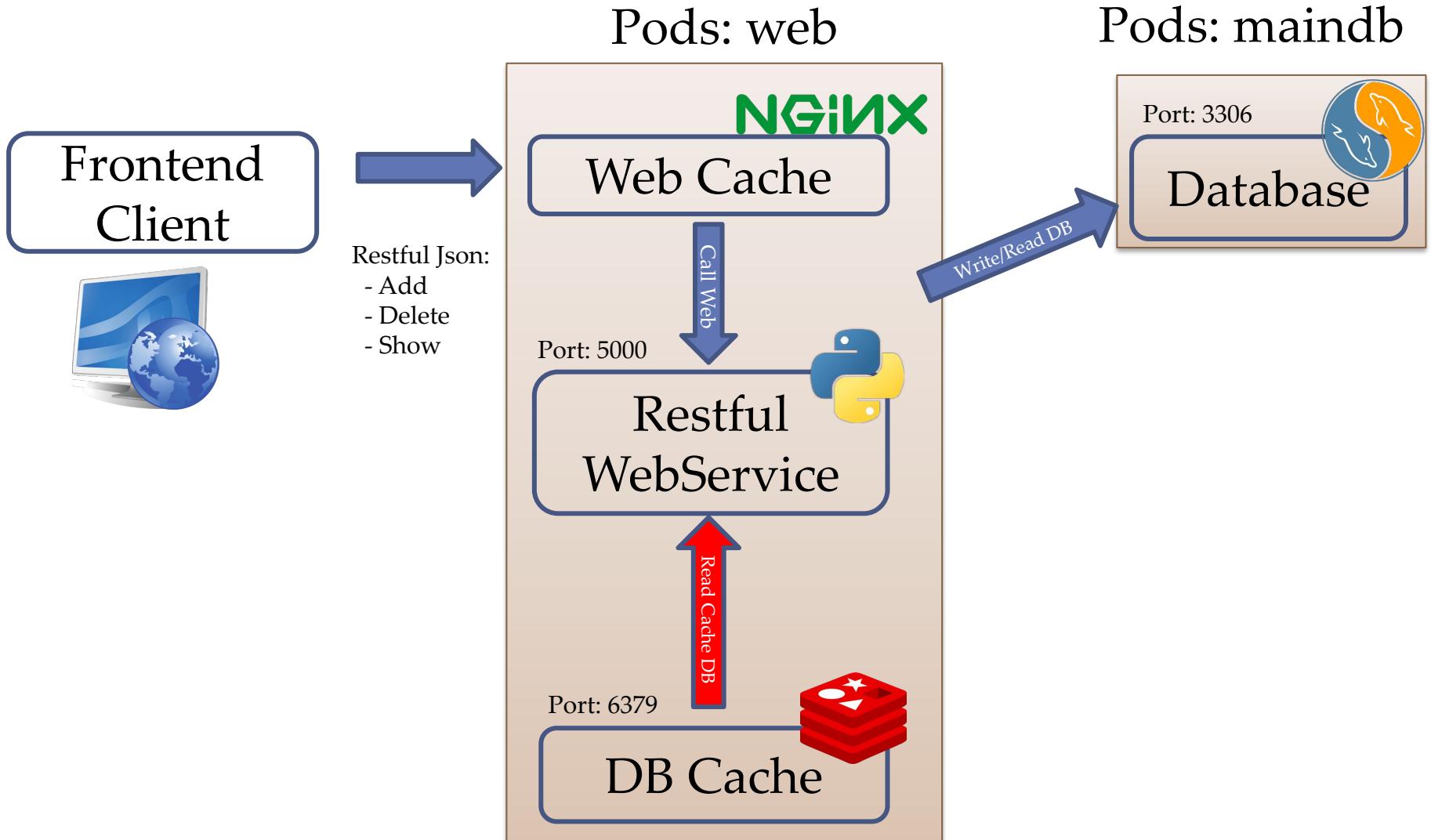
- Database and Database Cache

```
maindb:  
  image: labdocker/mysql:latest  
  container_name: maindb  
  environment:  
    MYSQL_ROOT_PASSWORD: password  
  
cachedb:  
  image: labdocker/redis:latest  
  container_name: cachedb  
  
webservice:  
  build: .  
  dockerfile: dockerfile_python  
  container_name: webservice  
  ports:  
    - "5000:5000"  
  links:  
    - cachedb:cachedb  
    - maindb:maindb  
  
webcache:  
  build: .  
  dockerfile: dockerfile_nginx  
  container_name: webcache  
  ports:  
    - "80:80"  
  links:  
    - webservice:webservice
```

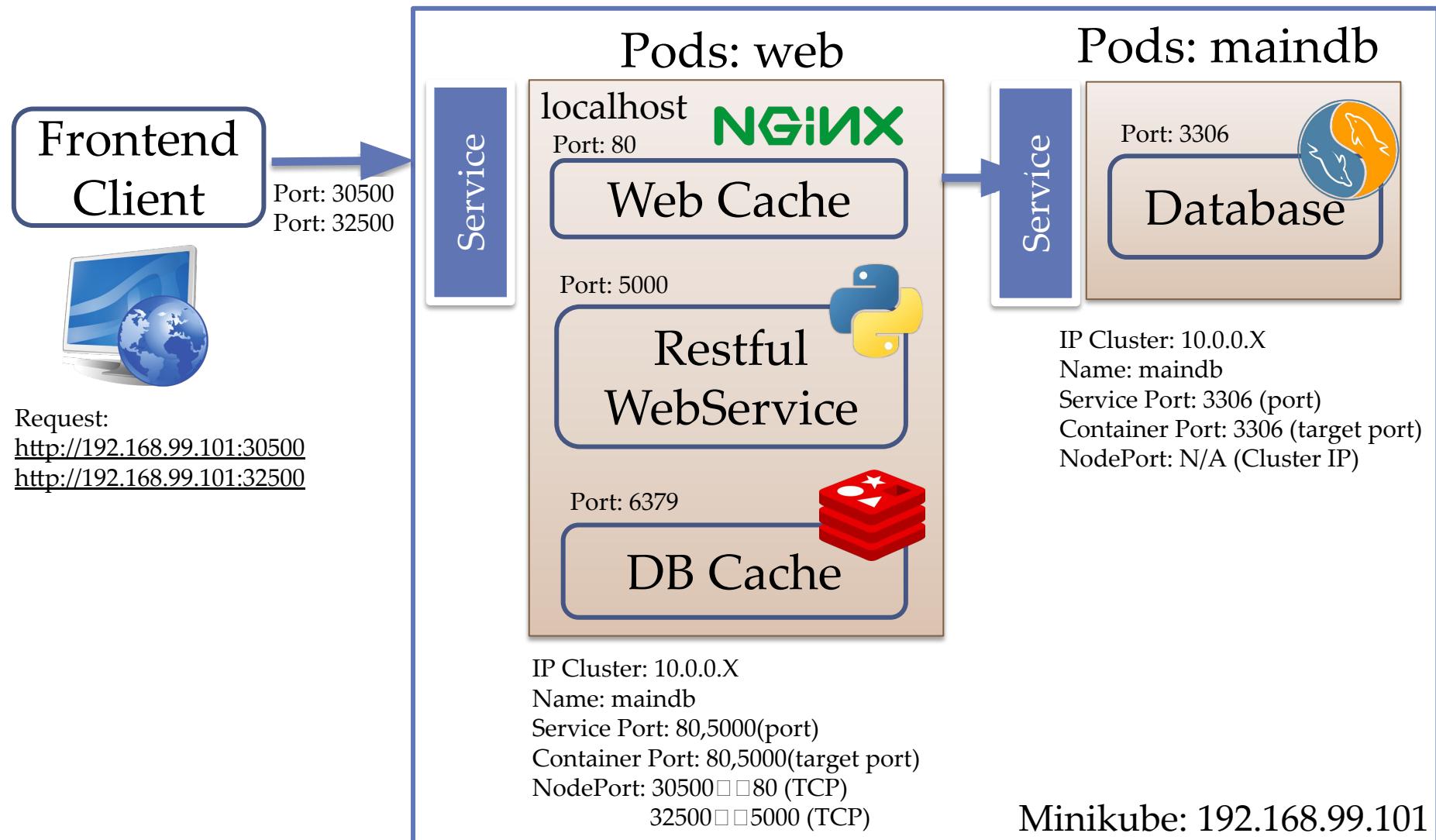
 Mysql Database

 Redis Key-Value Database

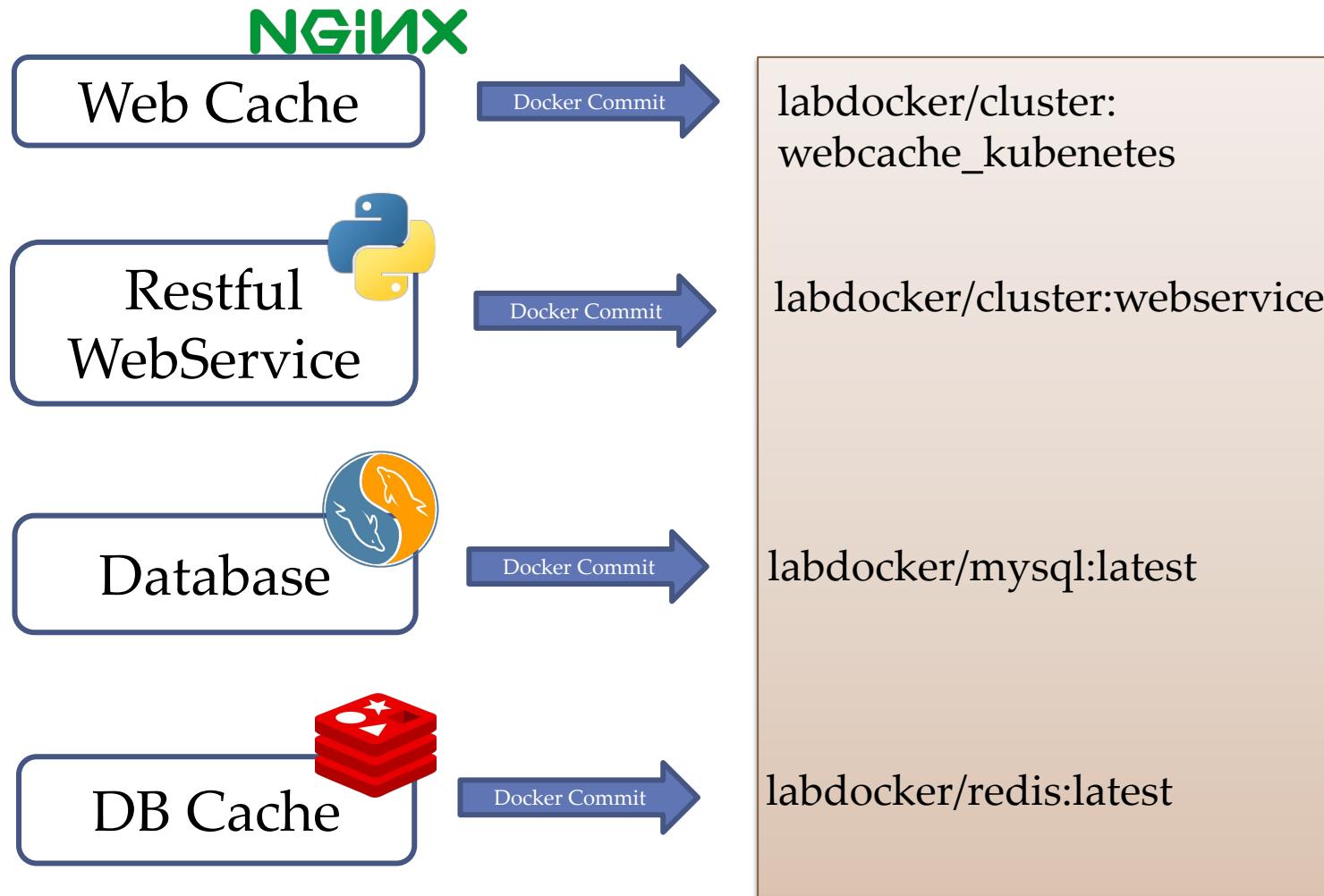
Pods, Container and Service



Pods, Container and Service



Pods, Container and Service



Pods, Container and Service

- Pods: maindb(YML)

```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: maindb
5    labels:
6      name: "maindb"
7      owner: "Praparn_L"
8      version: "1.0"
9      module: "maindb"
10     environment: "development"
11 spec:
12   containers:
13     - name: maindb
14       image: labdocker/mariadb:latest
15       ports:
16         - containerPort: 3306
17           protocol: TCP
18       env:
19         -
20           name: "MYSQL_ROOT_PASSWORD"
21           value: "password"
22
```

- Service: maindb(YML)

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: maindb
5    labels:
6      name: "maindb"
7      owner: "Praparn_L"
8      version: "1.0"
9      module: "maindb"
10     environment: "development"
11 spec:
12   ports:
13     - port: 3306
14       targetPort: 3306
15   selector:
16     name: "maindb"
17     owner: "Praparn_L"
18     version: "1.0"
19     module: "maindb"
20     environment: "development"
21
```



Pods, Container and Service

- Pods: web(YML)

```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: web
5    labels:
6      name: "web"
7      owner: "Praparn_L"
8      version: "1.0"
9      module: "web"
10     environment: "development"
11
12   spec:
13     containers:
14       - name: cachedb
15         image: labdocker/redis:latest
16         ports:
17           - containerPort: 6379
18             protocol: TCP
19       - name: webservice
20         image: labdocker/cluster:webservicev2
21         env:
22           - name: "REDIS_HOST"
23             value: "localhost"
24         ports:
25           - containerPort: 5000
26             protocol: TCP
27       - name: webcache
28         image: labdocker/cluster:webcache_kubernetes
29         ports:
30           - containerPort: 80
31             protocol: TCP
```

- Service: web (YML)

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: web
5    labels:
6      name: "web"
7      owner: "Praparn_L"
8      version: "1.0"
9      module: "Web"
10     environment: "development"
11
12   spec:
13     selector:
14       name: "web"
15       owner: "Praparn_L"
16       version: "1.0"
17       module: "web"
18       environment: "development"
19     type: NodePort
20     ports:
21       - port: 5000
22         name: webservice
23         targetPort: 5000
24         protocol: TCP
25         nodePort: 32500
26       - port: 80
27         name: webcache
28         targetPort: 80
29         protocol: TCP
30         nodePort: 30500
```



Pods, Container and Service

- Create "maindb" Pods and service

```
[praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
No resources found.
[praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
kubernetes  10.0.0.1    <none>        443/TCP  4d
[praparns-MacBook-Pro:multicontainer praparn$ kubectl create -f databasemodule_pod.yml
pod "maindb" created
[praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME      READY      STATUS      RESTARTS      AGE
maindb   1/1       Running     0            9s
[praparns-MacBook-Pro:multicontainer praparn$ kubectl create -f databasemodule_svc.yml
service "maindb" created
[praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
kubernetes  10.0.0.1    <none>        443/TCP  4d
maindb     10.0.0.134  <none>        3306/TCP  7s
praparns-MacBook-Pro:multicontainer praparn$ ]
```



Pods, Container and Service

- Create “web” Pods and service

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl create -f webmodule_pod.yml
pod "web" created
praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME      READY     STATUS    RESTARTS   AGE
maindb    1/1      Running   0          2m
web       3/3      Running   0          29s
praparns-MacBook-Pro:multicontainer praparn$ kubectl create -f webmodule_svc.yml
service "web" created
praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP    EXTERNAL-IP   PORT(S)           AGE
kubernetes  10.0.0.1    <none>        443/TCP         4d
maindb     10.0.0.134   <none>        3306/TCP        1m
web        10.0.0.69    <nodes>       5000:30661/TCP,80:30500/TCP   4s
praparns-MacBook-Pro:multicontainer praparn$
```



Welcome Page from Container Python Lab

Checkpoint Date/Time: Sun Jul 2 13:08:00 2017

Pods, Container and Service

- Initial database and insert data

```
praparns-MacBook-Pro:multicontainer praparn$ export Server_IP=192.168.99.100
praparns-MacBook-Pro:multicontainer praparn$ export Server_Port=30500
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port
curl: (7) Failed to connect to 192.168.99.100 port 30500: Connection refused
praparns-MacBook-Pro:multicontainer praparn$ clear
```

```
praparns-MacBook-Pro:multicontainer praparn$ export Server_IP=192.168.99.101
praparns-MacBook-Pro:multicontainer praparn$ export Server_Port=30500
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Jul  2 13:18:09 2017
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/init
##### Database Create New Account Table Done #####
```

```
praparns-MacBook-Pro:multicontainer praparn$ curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "desribe":"Slave"}' http://$Server_IP:$Server_Port/users/insertuser
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 02 Jul 2017 13:18:45 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive
#####
Record was added #####
praparns-MacBook-Pro:multicontainer praparn$
```

Pods, Container and Service

- Initial database, insert and get data (Direct/Cache)

```
praparns-MacBook-Pro:multicontainer praparn$ export Server_IP=192.168.99.100
praparns-MacBook-Pro:multicontainer praparn$ export Server_Port=30500
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port
curl: (7) Failed to connect to 192.168.99.100 port 30500: Connection refused
praparns-MacBook-Pro:multicontainer praparn$ clear
```

```
praparns-MacBook-Pro:multicontainer praparn$ export Server_IP=192.168.99.101
praparns-MacBook-Pro:multicontainer praparn$ export Server_Port=30500
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Jul  2 13:18:09 2017
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/init
##### Database Create New Account Table Done #####
```

```
praparns-MacBook-Pro:multicontainer praparn$ curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "desribe":"Slave"}' http://$Server_IP:$Server_Port/users/insertuser
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 02 Jul 2017 13:18:45 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive
##### Record was added #####
praparns-MacBook-Pro:multicontainer praparn$
```

```
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Direct)
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Cache)
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/4
Sakkan Yanyicharoen(Database Direct)
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/4
Sakkan Yanyicharoen(Database Cache)
```



Pods, Container and Service

- Delete data (Both cache and database)

```
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/removeuser/1
#####
Record was deleted (Both Database Cache)
#####
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/removeuser/2
#####
Record was deleted #####
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/removeuser/3
#####
Record was deleted #####
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port/users/removeuser/4
#####
Record was deleted (Both Database Cache)
#####
praparns-MacBook-Pro:multicontainer praparn$ █
```

- Remove

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl delete svc maindb web
service "maindb" deleted
service "web" deleted
praparns-MacBook-Pro:multicontainer praparn$ kubectl delete pods maindb web
pod "maindb" deleted
pod "web" deleted
praparns-MacBook-Pro:multicontainer praparn$ █
```

Pods, Container and Services

- Check how to operate with component on Kubernets via “Kubectl explain xxx”

```
kubectl explain <Pods/SVC/etc>
```

```
kubectl api-versions
```

```
ubuntu@ip-10-0-1-239:~$ kubectl explain pods
KIND:     Pod
VERSION:  v1

DESCRIPTION:
  Pod is a collection of containers that can run on a host. This resource is
  created by clients and scheduled onto hosts.

FIELDS:
  apiVersion  <string>
    APIVersion defines the versioned schema of this representation of an
    object. Servers should convert recognized schemas to the latest internal
    value, and may reject unrecognized values. More info:
    https://git.k8s.io/community/contributors/devel/api-conventions.md#resources

  kind <string>
    Kind is a string value representing the REST resource this object
    represents. Servers may infer this from the endpoint the client submits
    requests to. Cannot be updated. In CamelCase. More info:
    https://git.k8s.io/community/contributors/devel/api-conventions.md#types-kinds

  metadata    <Object>
    Standard object's metadata. More info:
    https://git.k8s.io/community/contributors/devel/api-conventions.md#metadata

  spec <Object>
    Specification of the desired behavior of the pod. More info:
    https://git.k8s.io/community/contributors/devel/api-conventions.md#spec-and-status

  status      <Object>
    Most recently observed status of the pod. This data may not be up to date.
    Populated by the system. Read-only. More info:
    https://git.k8s.io/community/contributors/devel/api-conventions.md#spec-and-status

ubuntu@ip-10-0-1-239:~$
```

```
ubuntu@ip-10-0-1-239:~$ kubectl api-versions
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
apps/v1beta1
apps/v1beta2
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1
coordination.k8s.io/v1beta1
crd.projectcalico.org/v1
events.k8s.io/v1beta1
extensions/v1beta1
networking.k8s.io/v1
networking.k8s.io/v1beta1
node.k8s.io/v1beta1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1
scheduling.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
ubuntu@ip-10-0-1-239:~$
```



Pods, Container and Services

- For check all component on K8S version

```
kubectl api-resources --o wide
```

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND	VERBS
bindings			true	Binding	[create] [get list]
componentstatuses	cs		false	ComponentStatus	[create delete deletecollection get list patch update watch]
configmaps	cm		true	ConfigMap	[create delete deletecollection get list patch update watch]
endpoints	ep		true	Endpoints	[create delete deletecollection get list patch update watch]
events	ev		true	Event	[create delete deletecollection get list patch update watch]
limitranges	limits		true	LimitRange	[create delete deletecollection get list patch update watch]
namespaces	ns		false	Namespace	[create delete get list patch update watch]
nodes	no		false	Node	[create delete deletecollection get list patch update watch]
persistentvolumeclaims	pvc		true	PersistentVolumeClaim	[create delete deletecollection get list patch update watch]
persistentvolumes	pv		false	PersistentVolume	[create delete deletecollection get list patch update watch]
pods	po		true	Pod	[create delete deletecollection get list patch update watch]
podtemplates			true	PodTemplate	[create delete deletecollection get list patch update watch]
replicationcontrollers	rc		true	ReplicationController	[create delete deletecollection get list patch update watch]
resourcequotas	quota		true	ResourceQuota	[create delete deletecollection get list patch update watch]
secrets			true	Secret	[create delete deletecollection get list patch update watch]
serviceaccounts	sa		true	ServiceAccount	[create delete deletecollection get list patch update watch]
services	svc		true	Service	[create delete get list patch update watch]
mutatingwebhookconfigurations		admissionregistration.k8s.io	false	MutatingWebhookConfiguration	[create delete deletecollection get list patch update watch]
validatingwebhookconfigurations		admissionregistration.k8s.io	false	ValidatingWebhookConfiguration	[create delete deletecollection get list patch update watch]
customresourcedefinitions	crd,crds	apiextensions.k8s.io	false	CustomResourceDefinition	[create delete deletecollection get list patch update watch]
apiservices		apiregistration.k8s.io	false	APIService	[create delete deletecollection get list patch update watch]
controllerrevisions		apps	true	ControllerRevision	[create delete deletecollection get list patch update watch]
daemonsets	ds	apps	true	DaemonSet	[create delete deletecollection get list patch update watch]
deployments	deploy	apps	true	Deployment	[create delete deletecollection get list patch update watch]
replicasets	rs	apps	true	ReplicaSet	[create delete deletecollection get list patch update watch]
statefulsets	sts	apps	true	StatefulSet	[create delete deletecollection get list patch update watch]
tokenreviews		authentication.k8s.io	false	TokenReview	[create]
localsubjectaccessreviews		authorization.k8s.io	true	LocalSubjectAccessReview	[create]
selfsubjectaccesreviews		authorization.k8s.io	false	SelfSubjectAccessReview	[create]
selfsubjectrulesreviews		authorization.k8s.io	false	SelfSubjectRulesReview	[create]
subjectaccesreviews		authorization.k8s.io	false	SubjectAccessReview	[create]
horizontalpodautoscalers	hpa	autoscaling	true	HorizontalPodAutoscaler	[create delete deletecollection get list patch update watch]
cronjobs	cj	batch	true	CronJob	[create delete deletecollection get list patch update watch]
jobs		batch	true	Job	[create delete deletecollection get list patch update watch]
certificatingsigningrequests	csr	certificates.k8s.io	false	CertificateSigningRequest	[create delete deletecollection get list patch update watch]
leases		coordination.k8s.io	true	Lease	[create delete deletecollection get list patch update watch]
bgpconfigurations		crd.projectcalico.org	false	BGPConfiguration	[delete deletecollection get list patch create update watch]
bgppeers		crd.projectcalico.org	false	BGPPeer	[delete deletecollection get list patch create update watch]
blockaffinities		crd.projectcalico.org	false	BlockAffinity	[delete deletecollection get list patch create update watch]
clusterinformations		crd.projectcalico.org	false	ClusterInformation	[delete deletecollection get list patch create update watch]
felixconfigurations		crd.projectcalico.org	false	FelixConfiguration	[delete deletecollection get list patch create update watch]
globalnetworkpolicies		crd.projectcalico.org	false	GlobalNetworkPolicy	[delete deletecollection get list patch create update watch]
globalnetworksets		crd.projectcalico.org	false	GlobalNetworkSet	[delete deletecollection get list patch create update watch]
hostendpoints		crd.projectcalico.org	false	HostEndpoint	[delete deletecollection get list patch create update watch]
ipamblocks		crd.projectcalico.org	false	IPAMBlock	[delete deletecollection get list patch create update watch]
ipamconfigs		crd.projectcalico.org	false	IPAMConfig	[delete deletecollection get list patch create update watch]
ipamhandles		crd.projectcalico.org	false	IPAMHandle	[delete deletecollection get list patch create update watch]
ippools		crd.projectcalico.org	false	IPPool	[delete deletecollection get list patch create update watch]
networkpolicies		crd.projectcalico.org	true	NetworkPolicy	[delete deletecollection get list patch create update watch]
networksets		crd.projectcalico.org	true	NetworkSet	[delete deletecollection get list patch create update watch]
events	ev	events.k8s.io	true	Event	[create delete deletecollection get list patch update watch]
daemonsets	ds	extensions	true	DaemonSet	[create delete deletecollection get list patch update watch]
deployments	deploy	extensions	true	Deployment	[create delete deletecollection get list patch update watch]
ingresses	ing	extensions	true	Ingress	[create delete deletecollection get list patch update watch]
networkpolicies	netpol	extensions	true	NetworkPolicy	[create delete deletecollection get list patch update watch]
podsecuritypolicies	psp	extensions	false	PodSecurityPolicy	[create delete deletecollection get list patch update watch]
replicasets	rs	extensions	true	ReplicaSet	[create delete deletecollection get list patch update watch]
ingresses	ing	networking.k8s.io	true	Ingress	[create delete deletecollection get list patch update watch]



Pods, Container and Services

Overview

WORKLOADS APIs

Container v1 core

CronJob v1beta1 batch

DaemonSet v1 apps

Deployment v1 apps

Job v1 batch

Pod v1 core

ReplicaSet v1 apps

ReplicationController v1 core

StatefulSet v1 apps

SERVICE APIs

Endpoints v1 core

Ingress v1beta1 networking.k8s.io

Service v1 core

CONFIG AND STORAGE APIs

ConfigMap v1 core

CSI Driver v1beta1 storage.k8s.io

CSINode v1beta1 storage.k8s.io

Secret v1 core

PersistentVolumeClaim v1 core

StorageClass v1 storage.k8s.io

Volume v1 core

VolumeAttachment v1 storage.k8s.io

METADATA APIs

ControllerRevision v1 apps

CustomResourceDefinition v1beta1 apiextensions.k8s.io

Event v1 core

LimitRange v1 core

HorizontalPodAutoscaler v1 autoscaling

API OVERVIEW

Welcome to the Kubernetes API. You can use the Kubernetes API to read and write Kubernetes resource objects via a Kubernetes API endpoint.

Resource Categories

This is a high-level overview of the basic types of resources provided by the Kubernetes API and their primary functions.

Workloads are objects you use to manage and run your containers on the cluster.

Discovery & LB resources are objects you use to "stitch" your workloads together into an externally accessible, load-balanced Service.

Config & Storage resources are objects you use to inject initialization data into your applications, and to persist data that is external to your container.

Cluster resources define how the cluster itself is configured; these are typically used only by cluster operators.

Metadata resources are objects you use to configure the behavior of other resources within the cluster, such as `HorizontalPodAutoscaler` for scaling workloads.

Resource Objects

Resource objects typically have 3 components:

- **Resource ObjectMeta:** This is metadata about the resource, such as its name, type, API version, annotations, and labels. This contains fields that may be updated both by the end user and the system (e.g. annotations).
- **ResourceSpec:** This is defined by the user and describes the desired state of the system. Fill this in when creating or updating an object.
- **ResourceStatus:** This is filled in by the server and reports the current state of the system. In most cases, users don't need to change this.

Ref: <https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.15/#-strong-api-overview-strong->

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Pods, Container and Services

- Services

- Service is independent from Pods
 - Pods can create / destroy / restart every time (manual/automatic)
 - Mean “Pods” always change their nodes/ip/location everytime
 - Service don't care how Pods are being, But they still can map the Pods
- Service is abstract of Pods (1 – N Pods)
 - Usually defined port by “Label”
 - Expose access Pods/Load Balance with service (kube-proxy: iptables)
 - “port” that service open for access
 - “target port” that map with Pods for access
 - “type” TCP/UDP
 - Discovery service option:
 - ENVIRONMENT:
 - {SVC_NAME_SERVICE_HOST}
 - {SVC_NAME_SERVICE_PORT}
 - DNS (cluster-addon): Name same service name



Pods, Container and Services

- Services
 - Pods “web”

```
1 apiVersion: "v1"
2 kind: Pod
3 metadata:
4   name: web
5   labels:
6     name: "web"
7     owner: "Praparn_L"
8     version: "1.0"
9     module: "web"
10    environment: "development"
11  spec:
12    containers:
13      - name: cachedb
14        image: labdocker/redis:latest
15        ports:
16          - containerPort: 6379
17            protocol: TCP
18        - name: webservice
19          image: labdocker/cluster:webservicev2
20          env:
21            - name: "REDIS_HOST"
22              value: "localhost"
23          ports:
24            - containerPort: 5000
25              protocol: TCP
26        - name: webcache
27          image: labdocker/cluster:webcache_kubernetes
28          ports:
29            - containerPort: 80
30              protocol: TCP
```

Pods “web2”

```
1 apiVersion: "v1"
2 kind: Pod
3 metadata:
4   name: web2
5   labels:
6     name: "web"
7     owner: "Praparn_L"
8     version: "1.0"
9     module: "web"
10    environment: "development"
11  spec:
12    containers:
13      - name: cachedb
14        image: labdocker/redis:latest
15        ports:
16          - containerPort: 6379
17            protocol: TCP
18        - name: webservice
19          image: labdocker/cluster:webservicev2set2
20          env:
21            - name: "REDIS_HOST"
22              value: "localhost"
23          ports:
24            - containerPort: 5000
25              protocol: TCP
26        - name: webcache
27          image: labdocker/cluster:webcache_kubernetes
28          ports:
29            - containerPort: 80
30              protocol: TCP
```

Service “web”

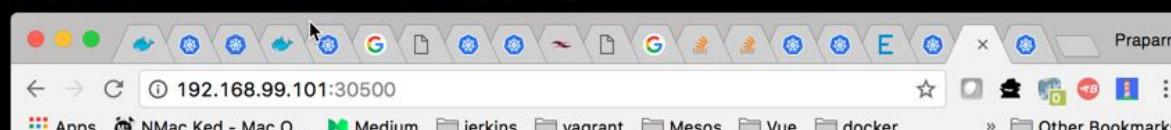
```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: web
5   labels:
6     name: "web"
7     owner: "Praparn_L"
8     version: "1.0"
9     module: "Web"
10    environment: "development"
11  spec:
12    selector:
13      name: "web"
14      owner: "Praparn_L"
15      version: "1.0"
16      module: "web"
17      environment: "development"
18    type: NodePort
19    ports:
20      - port: 5000
21        name: webservice
22        targetPort: 5000
23        protocol: TCP
24        nodePort: 32500
25      - port: 80
26        name: webcache
27        targetPort: 80
28        protocol: TCP
29        nodePort: 30500
```



Pods, Container and Services

- Services
 - Existing Pods “web”

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME          READY   STATUS      RESTARTS   AGE
curl-1580724602-gjjhq   0/1     CrashLoopBackOff   27        1h
maindb         1/1     Running    0          56m
web            3/3     Running    0          7m
praparns-MacBook-Pro:multicontainer praparn$ curl http://$Server_IP:$Server_Port
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Sun Jul  2 15:28:13 2017
praparns-MacBook-Pro:multicontainer praparn$ 
```



The screenshot shows a Mac OS X desktop environment. A browser window titled 'Praparn' is open, displaying a 'Welcome Page from Container Python Lab'. The URL in the address bar is '192.168.99.101:30500'. Below the address bar, the status bar shows 'Checkpoint Date/Time: Sun Jul 2 15:28:20 2017'. The browser's toolbar and menu bar are visible at the top.

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP   EXTERNAL-IP   PORT(S)           AGE
kubernetes  10.0.0.1    <none>        443/TCP          5d
maindb     10.0.0.236   <none>        3306/TCP         1m
web        10.0.0.212   <nodes>       5000:32500/TCP,80:30500/TCP   1m
praparns-MacBook-Pro:multicontainer praparn$ 
```

Pods, Container and Services

- Services
 - Replace new Pods “web2”

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl create -f webmodule_pod2.yml
pod "web2" created
praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
curl-1580724602-gjjhq  0/1     CrashLoopBackOff  27          1h
maindb      1/1     Running   0           58m
web         3/3     Running   0           10m
web2        3/3     Running   0           1m
praparns-MacBook-Pro:multicontainer praparn$ kubectl delete pods web
pod "web" deleted
praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
curl-1580724602-gjjhq  0/1     CrashLoopBackOff  27          1h
maindb      1/1     Running   0           59m
web2        3/3     Running   0           2m


Welcome Page from Container Python Lab (Set 2)



Checkpoint Date/Time: Sun Jul 2 15:30:53 2017



```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes 10.0.0.1 <none> 443/TCP 5d
maindb 10.0.0.236 <none> 3306/TCP 1m
web 10.0.0.212 <nodes> 5000:32500/TCP,80:30500/TCP 1m
praparns-MacBook-Pro:multicontainer praparn$
```


```



Pods, Container and Services

- Services:
 - Publish Service Type:
 - CLUSTER-IP (Default): blind port with ip address of cluster (Accessible from internal cluster system)
 - NodePort: blind port with ip address of node. By default kubernetes will random port (30000-32757). If we need to specify set the option: “nodePort: XXXXX”

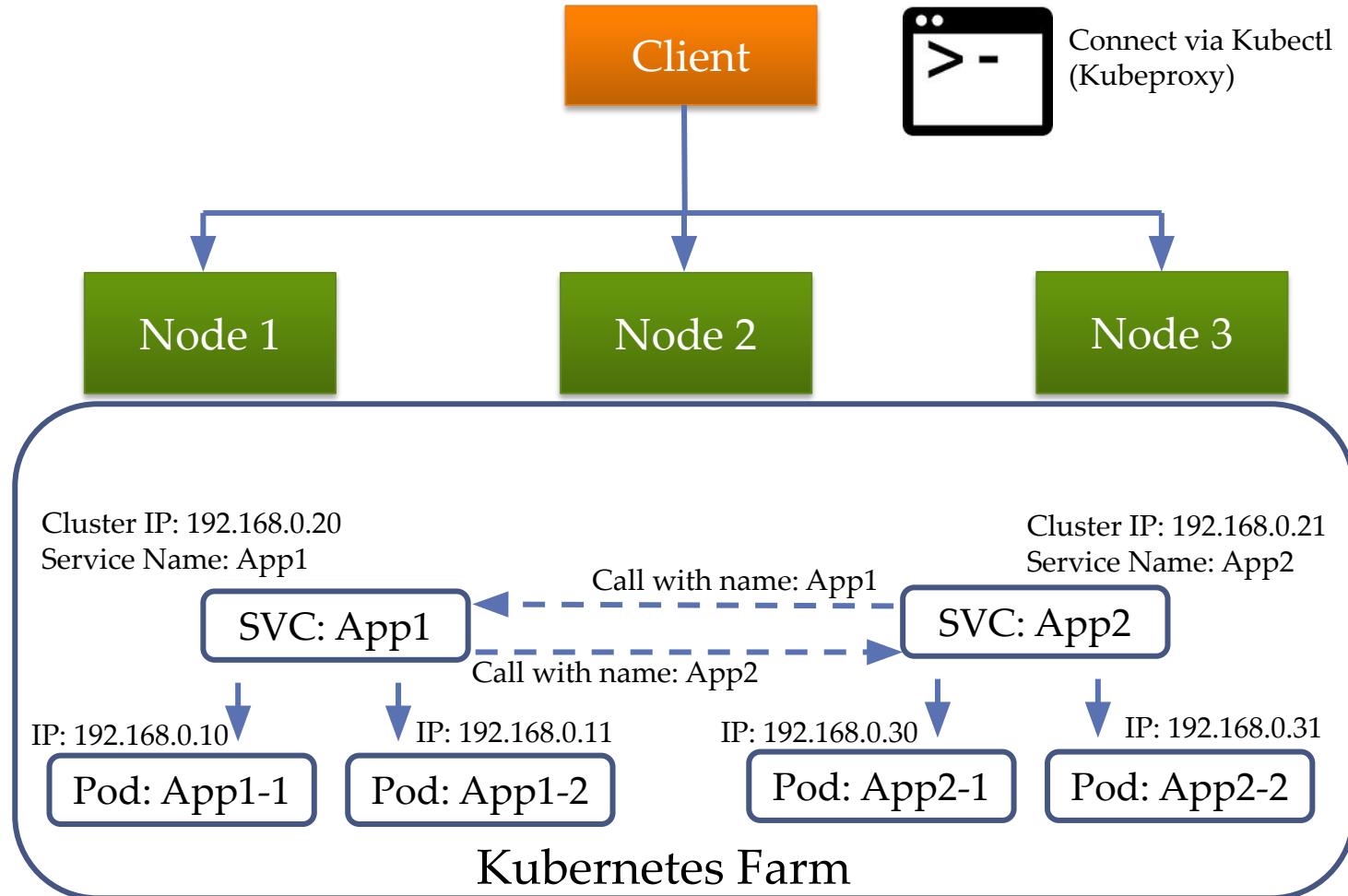
```
18 type: NodePort
19 ports:
20   - port: 5000
21     name: webservice
22     targetPort: 5000
23     protocol: TCP
24     nodePort: 32500
25   - port: 80
26     name: webcache
27     targetPort: 80
28     protocol: TCP
29     nodePort: 30500
```

```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP  EXTERNAL-IP  PORT(S)          AGE
kubernetes  10.0.0.1    <none>        443/TCP       5d
maindb     10.0.0.236   <none>        3306/TCP      1m
web        10.0.0.212   <nodes>       5000:32500/TCP,80:30500/TCP  1m
praparns-MacBook-Pro:multicontainer praparn$ █
```



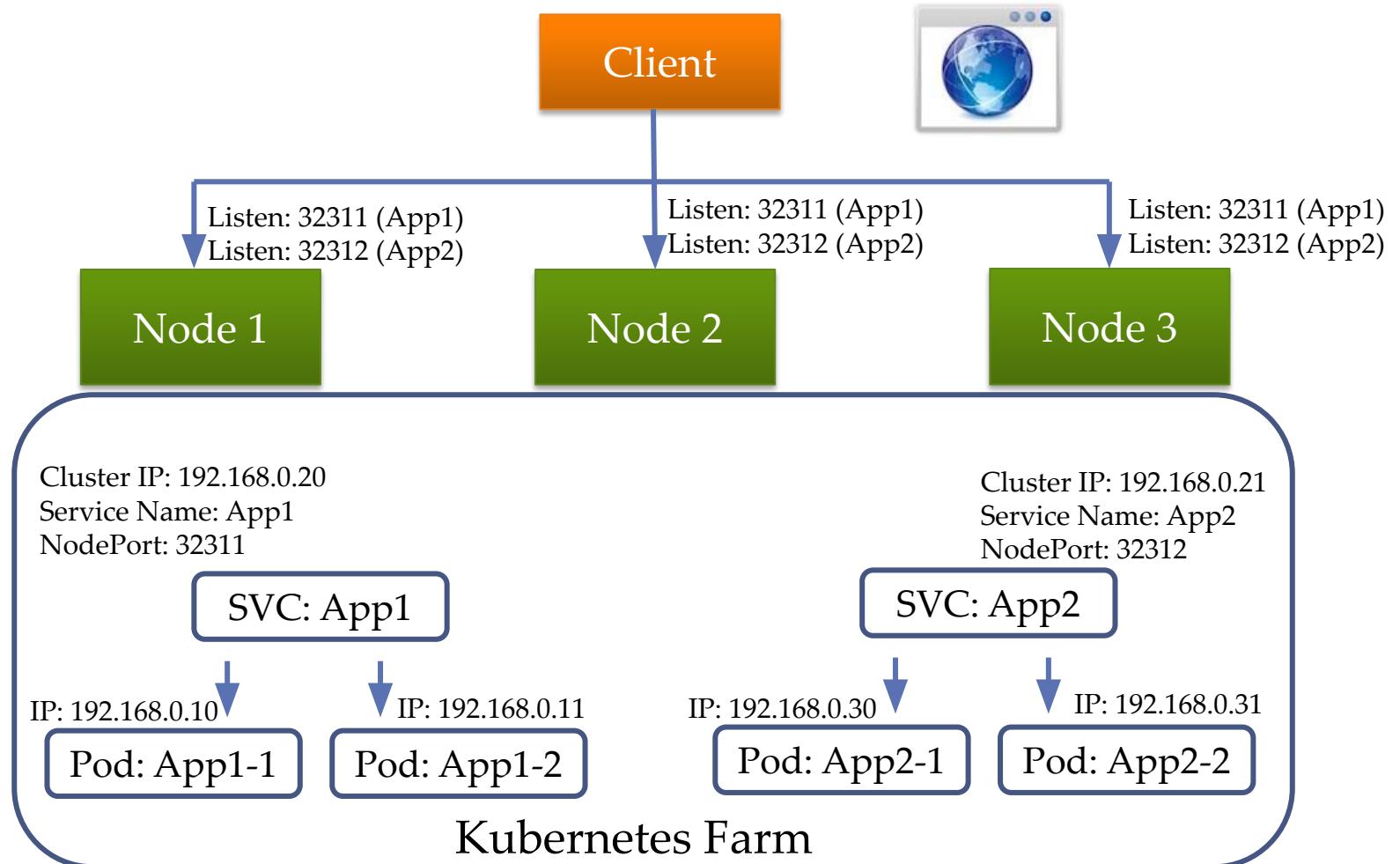
Pods, Container and Services

- Services: ClusterIP



Pods, Container and Services

- Services: NodePort



Pods, Container and Services

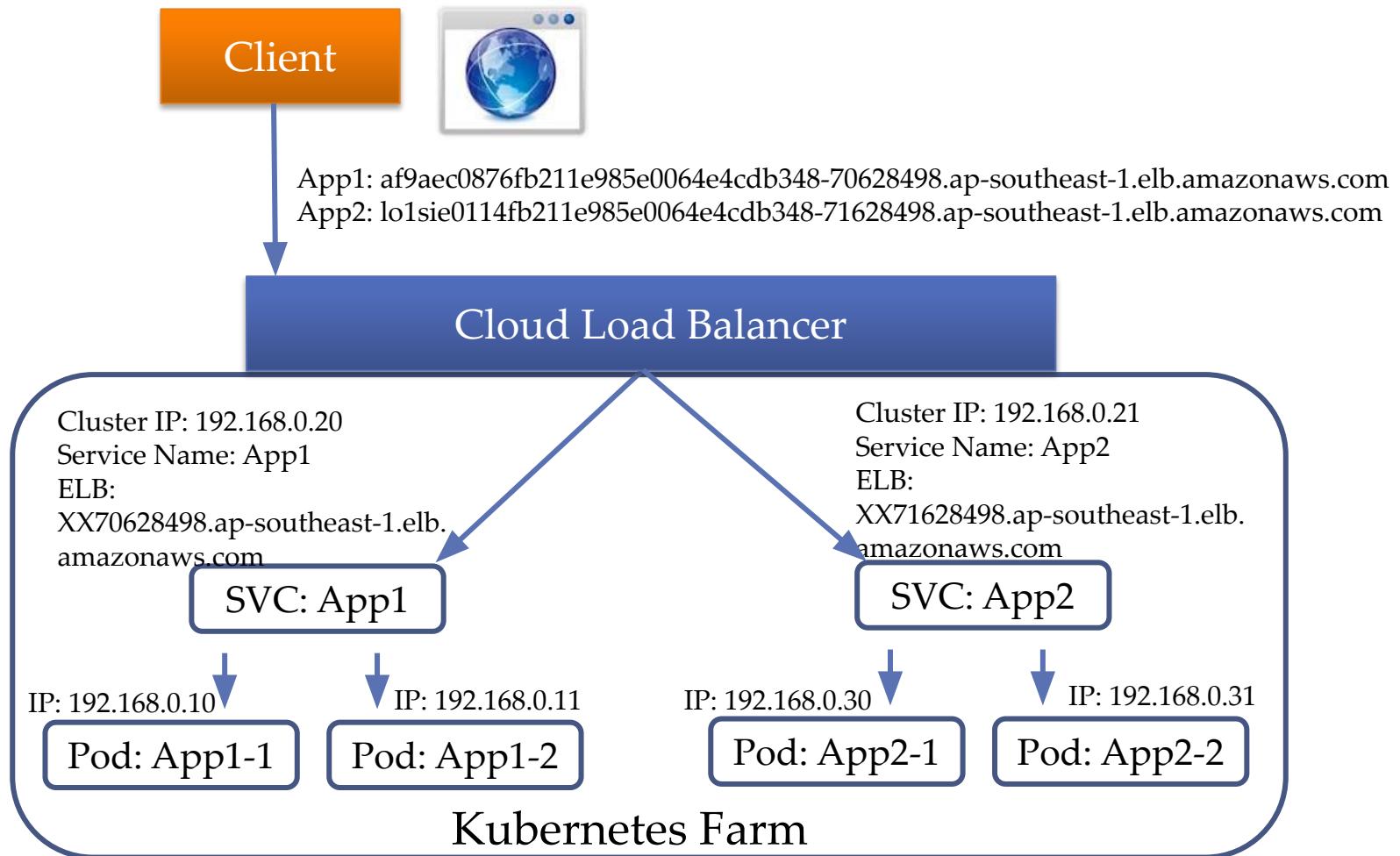
- Services
 - Publish Service Type:
 - LoadBalance:
 - Use load balancer from external cloud provider for intercept traffic
 - Manage by Kubernetes Itself
 - Flexible for use facilities on cloud



```
1 kind: Service
2 apiVersion: v1
3 metadata:
4   name: my-service
5 spec:
6   selector:
7     app: MyApp
8   ports:
9     - protocol: TCP
10    port: 80
11    targetPort: 9376
12    nodePort: 30061
13   clusterIP: 10.0.171.239
14   loadBalancerIP: 78.11.24.19
15   type: LoadBalancer
16 status:
17   loadBalancer:
18     ingress:
19       - ip: 146.148.47.155
20 |
```

Pods, Container and Services

- Services: Load Balancer



Pods, Container and Services

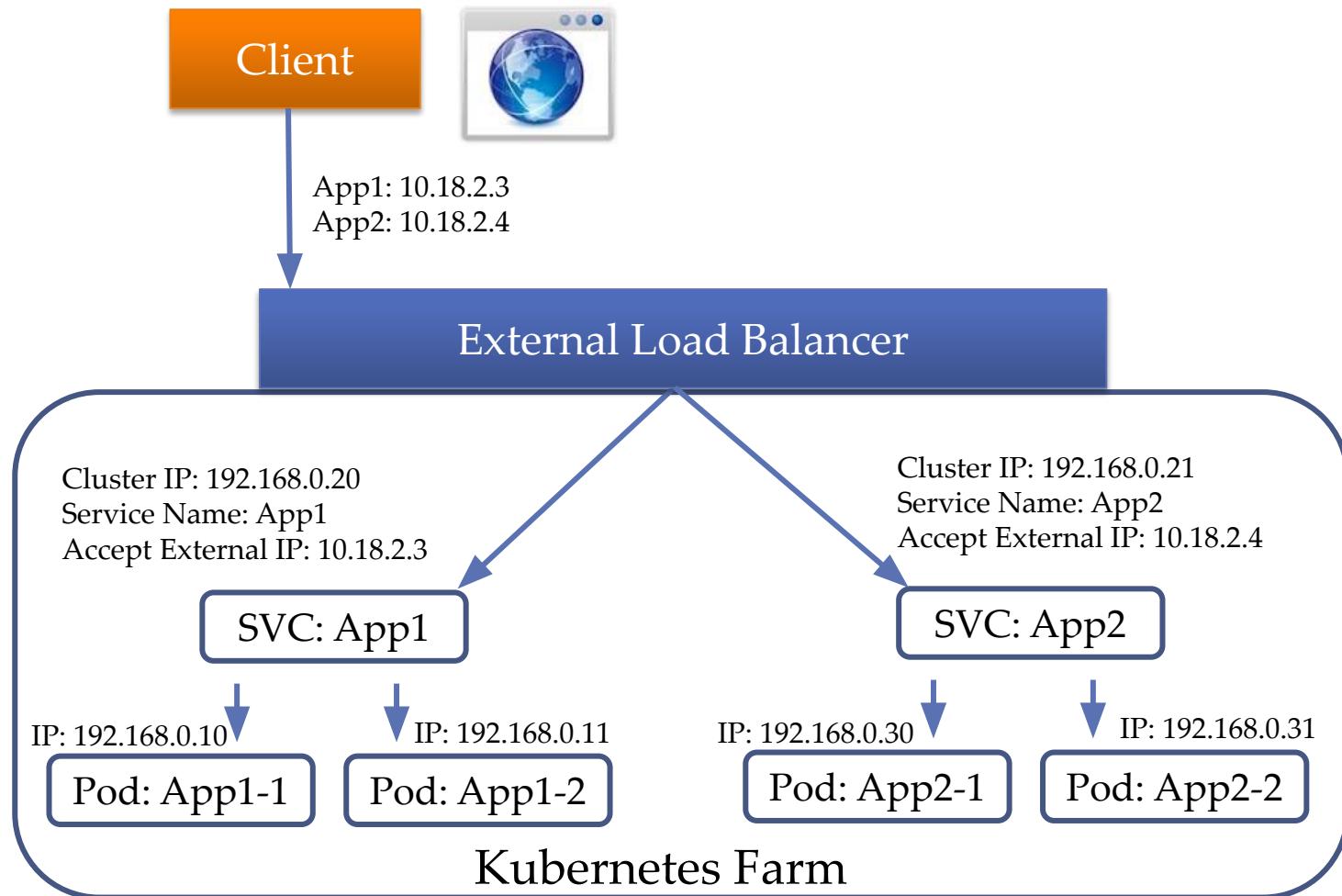
- Services:
 - Publish Service Type:
 - ExternalIP:
 - Similar with Load Balancer
 - Reference to external load balance by IP Address
 - Not manage by kubernetes itself

```
1 kind: Service
2 apiVersion: v1
3 metadata:
4   name: my-service
5 spec:
6   selector:
7     app: MyApp
8   ports:
9     - name: http
10    protocol: TCP
11    port: 80
12    targetPort: 9376
13   externalIPs:
14     - 80.11.12.10
```



Pods, Container and Services

- Services: External IP



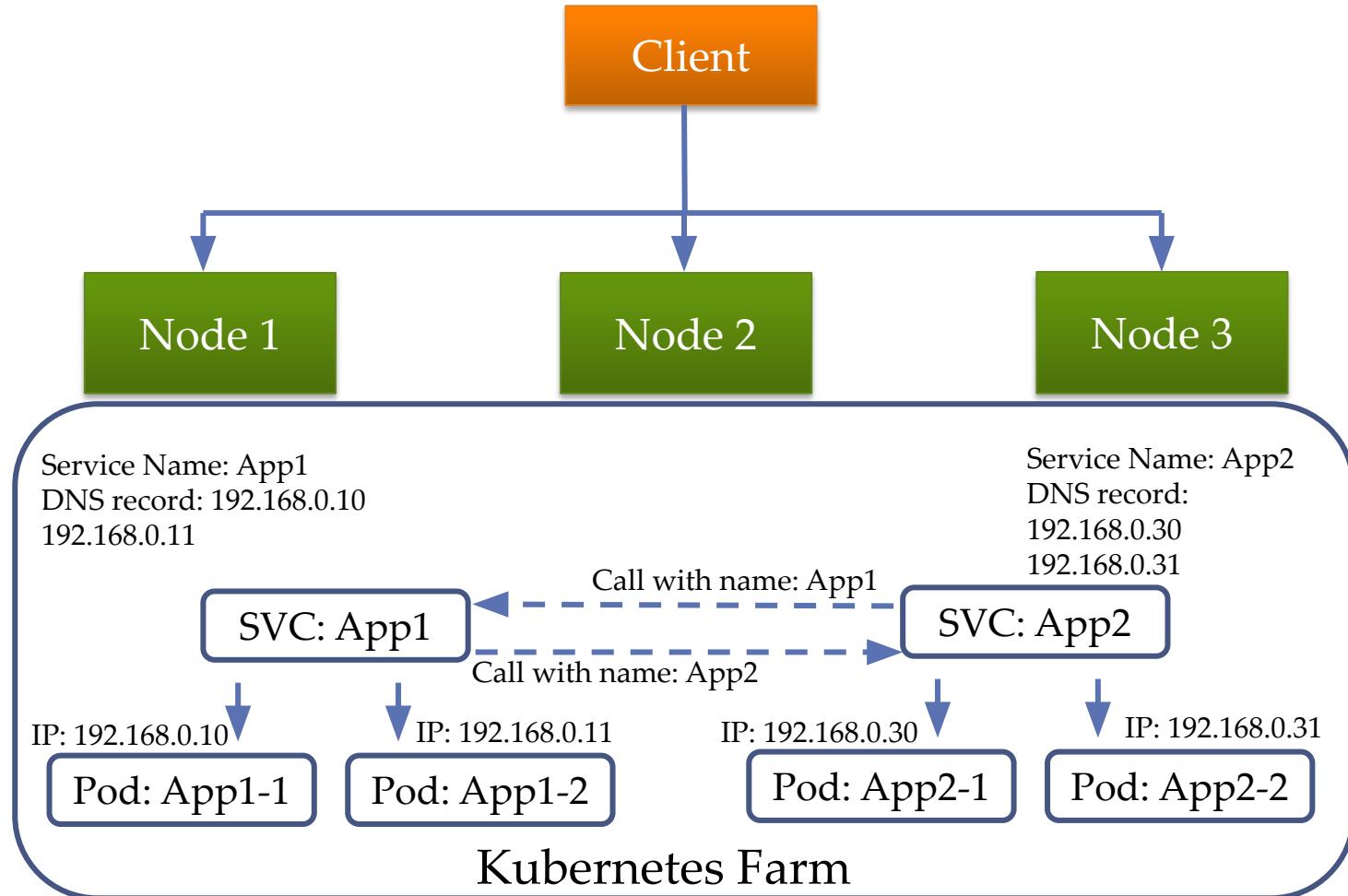
Pods, Container and Services

- Services
 - Publish Service Type:
 - Headless Service (StatefulSet): service will just dns round-robin for all statefulset pods's ip address

```
! 20dns.yml ✘
1 # A headless service to create DNS records
2 ---
3 apiVersion: v1
4 kind: Service
5 metadata:
6   name: kafka
7   namespace: sansiri-unityapi
8 spec:
9   ports:
10    - port: 9092
11      # [podname].broker.kafka.svc.cluster.local
12      clusterIP: None
13   selector:
14     app: kafka
15
```

Pods, Container and Services

- Services: Headless



Pods, Container and Services

- Services
 - Publish Service Type:
 - ExternalName:
 - No selector / No define pods / No endpoints
 - Use for return CNAME record for DNS-Load Balance
 - Work with kube-dns (Kubernetes V1.7 or higher)

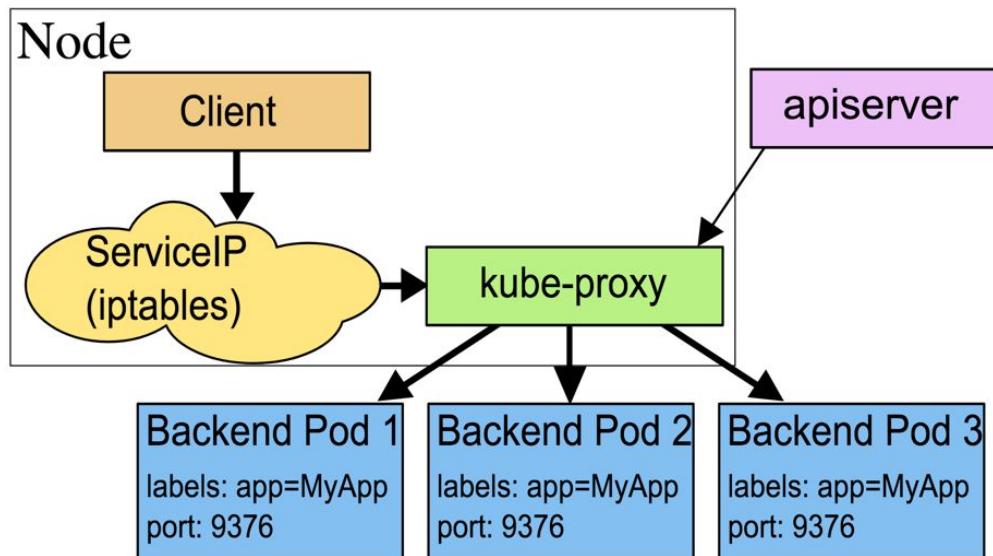
```
1 kind: Service
2 apiVersion: v1
3 metadata:
4   name: my-service
5   namespace: prod
6 spec:
7   type: ExternalName
8   externalName: my.database.example.com
```

- nslookup “my-service.prod.svc.CLUSTER” □ return CNAME: “my.database.example.com”



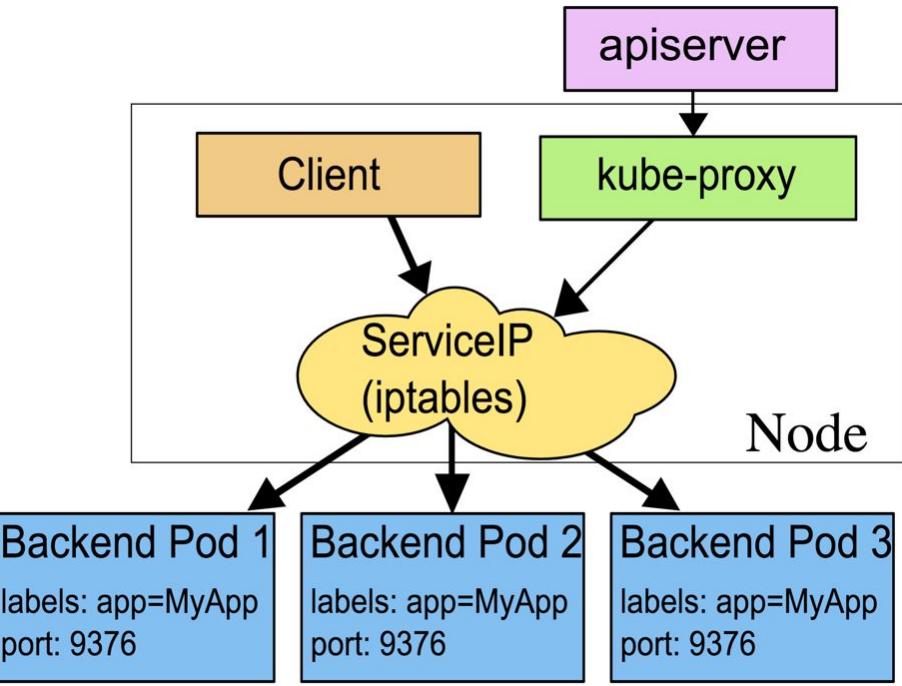
Pods, Container and Services

- kube-proxy (proxy-mode)
 - Every node on K8S will use kube-proxy for response to service
 - Initial Virtual IP for service (except External Name)
 - Vary “proxy-mode” for support in kube-proxy
 - Proxy-mode: userspace (default on kubernetes v1.0 – 1.7)
 - Open port on local node (random) and proxy to backend pods
 - Userspace (binary) will terminate and establish new connect to pods
 - Round robin traffic / retry on case pods fail



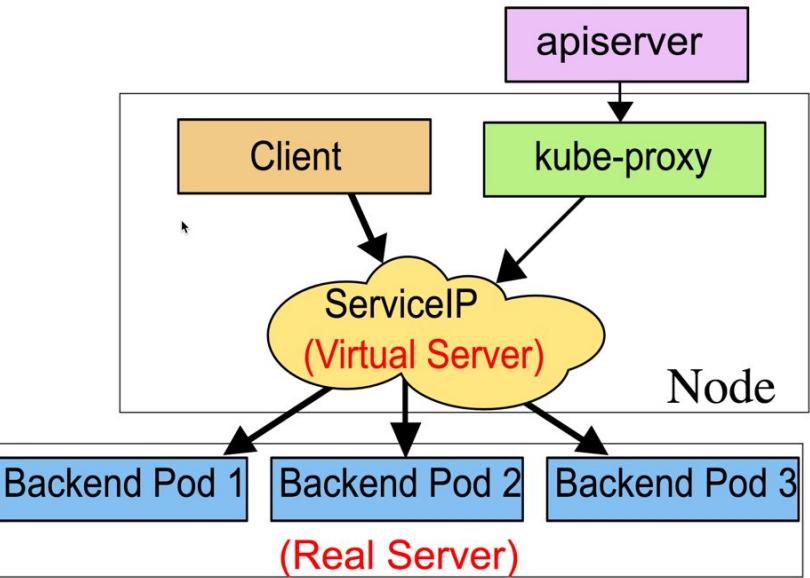
Pods, Container and Services

- kube-proxy (proxy-mode)
 - Proxy-mode: iptables (default on kubernetes v1.8 - 1.20)
 - Install iptable rule on ServiceIP (VIP) for capture traffic and port
 - Redirect traffic to local node for load balance traffic to pods
 - User kernelspace that faster than userspace (No switch between kernelspace and user space)
 - Not support in case pods fail



Pods, Container and Services

- kube-proxy (proxy-mode)
 - Proxy-mode: ipvs (GA on kubernetes v1.11)
 - L4 LB for TCP/UDP with Support Health Check and Retries
 - Create “netlink” interface for establish connection via IPVS
 - Establish IPVS (IP Virtual Server) rule and sync (periodic for consistency before establish)
 - Pure 100% kernelspace and better performance
 - Multiple option for load balance
 - rr: round-robin
 - wrr: weight round-robin
 - sh: source hash



Pods, Container and Services

- Configure kube-proxy (proxy-mode)

Run kube-proxy in IPVS mode

Currently, local-up scripts, GCE scripts and kubeadm support switching IPVS proxy mode via exporting environment variables or specifying flags.

Prerequisite

Ensure IPVS required kernel modules (**Notes:** use `nf_conntrack` instead of `nf_conntrack_ipv4` for Linux kernel 4.19 and later)

```
ip_vs
ip_vs_rr
ip_vs_wrr
ip_vs_sh
nf_conntrack_ipv4
```

Cluster Created by Kubeadm

If you are using kubeadm with a [configuration file](#), you have to add `mode: ipvs` and also add `SupportIPVSPProxyMode: true` below the `kubeProxy` field as part of the kubeadm configuration.

```
...
kubeProxy:
  config:
    featureGates:
      SupportIPVSPProxyMode: true
    mode: ipvs
...
```

Note that in Kubernetes 1.11 and later, `SupportIPVSPProxyMode` is set to `true` by default.

before running

```
kube init --config <path_to_configuration_file>
```

<https://github.com/kubernetes/kubernetes/tree/master/pkg/proxy/ipvs>

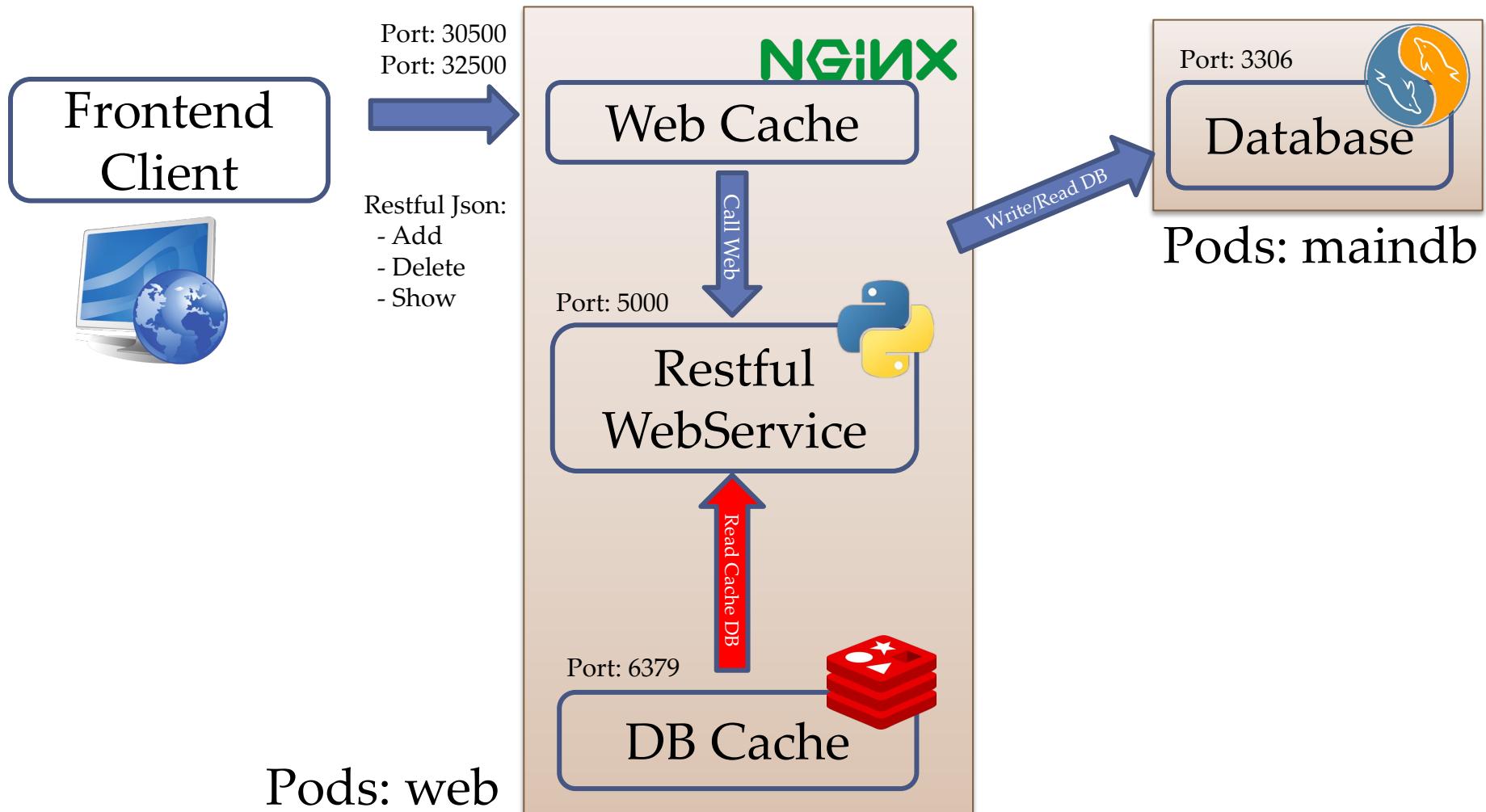
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop: Pods & Service

- Part 2: Deploy web pods with multi container



Pods, Container and Services

- Pods Priority and Preemption
 - Give the priority create for pods. When resource is insufficient kubernetes will consider to run higher priority pods first and evict the lower priority pods.
 - Priority will effective only when pods was “created”
 - Disable feature of “Preemption” is not recommend

```
1 apiVersion: componentconfig/v1alpha1
2 kind: KubeSchedulerConfiguration
3 algorithmSource:
4   provider: DefaultProvider
5
6 ...
7
8 disablePreemption: true
```



Pods, Container and Services

- Pods Priority and Preemption
 - Kubernetes will use value on object “PriorityClass” for consideration. Pods will reference this for define

```
1 apiVersion: scheduling.k8s.io/v1
2 kind: PriorityClass
3 metadata:
4   name: high-priority
5 value: 1000000
6 globalDefault: false
7 description: "This priority class for highest pods."
8
9 ---
10
11 apiVersion: scheduling.k8s.io/v1
12 kind: PriorityClass
13 metadata:
14   name: medium-priority
15 value: 1000
16 globalDefault: true
17 description: "This priority class for default pods."
18
19 ---
20
21 apiVersion: scheduling.k8s.io/v1
22 kind: PriorityClass
23 metadata:
24   name: low-priority
25 value: 100
26 globalDefault: false
27 description: "This priority class for lowest pods."
28
```

```
1 apiVersion: "v1"
2 kind: Pod
3 metadata:
4   name: webtest
5 labels:
6   name: web
7   owner: Praparn_L
8   version: "1.0"
9   module: WebServer
10  environment: development
11 spec:
12   containers:
13     - name: webtest
14       image: labdocker/cluster:webservicelite
15       ports:
16         - containerPort: 5000
17           protocol: TCP
18   priorityClassName: high-priority
```



Daemon Set and RC

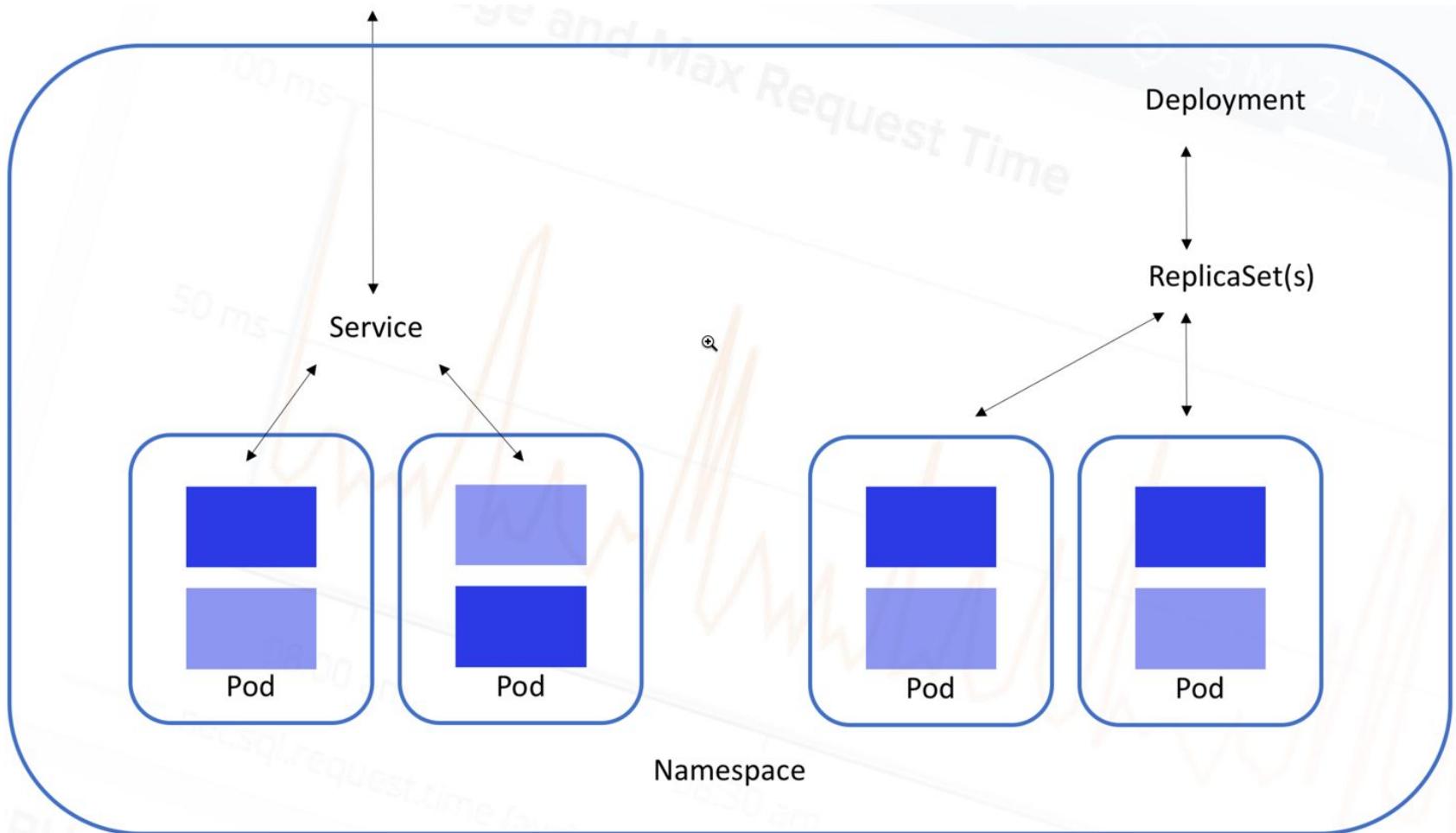


Kubernetes: Production Workload Orchestration



kubernetes
by Google

Daemon Set and RC



Daemon Set and RC

- What is Daemon Set
 - On basic pods and service can make microservice up and run!
 - But...
 - No maintain about available of Pods
 - Not response Pods scaling
 - Etc
 - Daemon Set will response:
 - Make sure that Pods run on all (or some) node in cluster
 - When node add to cluster. Pods will deploy automatic
 - When node remove, GC will automatic remove Pod
 - When remove Daemon Set, Pods will automatic remove
 - General Use Case
 - Storage Daemon (Ceph etc)
 - Node Monitor Daemon (Prometheus Agent etc)



Replication Controller (RC)

- What is RC
 - Daemon Set and Replication Controller work similarly.
 - RC (Replication Controller) will respond:
 - Create/Maintain Pods as "Replication Controller" (Pods Farm)
 - Keep copy of Pods (Replicas) as designed
 - Ensure that Pods are up and running with amount like design
 - If too much, it will kill some Pods
 - If too few, it will create another replicas of Pods
 - Auto healing if some Pods crash for any reason
 - Maintenance on cluster-level not node level



Replication Controller (RC)

- Replication Controller (RC)
 - RC is first version of module to maintain Pods available in cluster
 - Use Case...
 - Rescheduling
 - Scaling
 - Rolling updates (Complicate)
 - Map with service for manage release
 - RC is running base on label type: “Equality-based requirement”
 - Ex:

```
selector:  
  name: web  
  owner: Praparn_L      |  
  version: "1.0"  
  module: WebServer  
  environment: development
```

Replication Controller (RC)

- Example

- Pods “webtest”

```
1 apiVersion: "v1"
2 kind: Pod
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11  spec:
12    containers:
13      - name: webtest
14        image: labdocker/cluster:webservicelite
15        ports:
16          - containerPort: 5000
17            protocol: TCP
```

- RC “webtest”

```
1 apiVersion: v1
2 kind: ReplicationController
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11  spec:
12    replicas: 3
13    template:
14      metadata:
15        labels:
16          name: web
17          owner: Praparn_L
18          version: "1.0"
19          module: WebServer
20          environment: development
21    spec:
22      containers:
23        - name: webtest
24          image: labdocker/cluster:webservicelite
25          ports:
26            - containerPort: 5000
27              protocol: TCP
```

- SVC “webtest”

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11  spec:
12    selector:
13      name: web
14      owner: Praparn_L
15      version: "1.0"
16      module: WebServer
17      environment: development
18
19    type: NodePort
20    ports:
21      - port: 5000
22        name: http
23        targetPort: 5000
24        protocol: TCP
25        nodePort: 32500
```



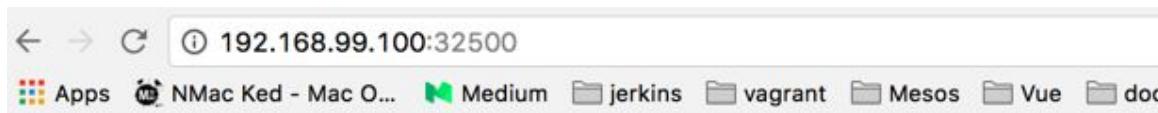
Replication Controller (RC)

- Create rc “webtest”

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl create -f webtest_rc.yml
replicationcontroller "webtest" created
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get rc
NAME      DESIRED   CURRENT   READY    AGE
webtest   3          3          3        2m
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
webtest-9m8tx  1/1    Running   0          3m
webtest-9xgt3  1/1    Running   0          3m
webtest-cns49  1/1    Running   0          3m
```

- Create svc “webtest”

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl create -f webtest_svc.yml
service "webtest" created
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get svc
NAME      CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes  10.0.0.1    <none>        443/TCP       8d
webtest     10.0.0.6    <nodes>       5000:32500/TCP  6s
praparns-MacBook-Pro:ReplicationController praparn$
```



Welcome Page from Container Python Lab

Checkpoint Date/Time: Wed Jul 5 15:55:31 2017



Replication Controller (RC)

- Test delete some Pods from command line

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
webtest-9m8tx  1/1     Running   0          11m
webtest-9xgt3  1/1     Running   0          11m
webtest-cns49  1/1     Running   0          11m
praparns-MacBook-Pro:ReplicationController praparn$ kubectl delete pods webtest-9m8tx
pod "webtest-9m8tx" deleted
```

- Recheck Pods unit and available

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
webtest-9xgt3  1/1     Running   0          21m
webtest-cns49  1/1     Running   0          21m
webtest-lmn54  1/1     Running   0          8m
praparns-MacBook-Pro:ReplicationController praparn$ curl http://192.168.99.100:32500
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Wed Jul  5 15:51:59 2017
praparns-MacBook-Pro:ReplicationController praparn$
```



Welcome Page from Container Python Lab

Checkpoint Date/Time: Wed Jul 5 15:55:31 2017



Replication Controller (RC)

- Check detail of create process on RC

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl describe rc webtest
Name:           webtest
Namespace:      default
Selector:       environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
Annotations:    <none>
Replicas:       3 current / 3 desired
Pods Status:   3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:        environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
  Containers:
    webtest:
      Image:        labdocker/cluster:webserviceelite
      Port:         5000/TCP
      Environment:  <none>
      Mounts:       <none>
      Volumes:      <none>
Events:
FirstSeen     LastSeen     Count  From            SubObjectPath  Type    Reason          Message
-----     -----     ----  ----            -----          ----  ----          -----
26m          26m          1     replication-controller      Normal  SuccessfulCreate  Created pod: webtest-9m8tx
26m          26m          1     replication-controller      Normal  SuccessfulCreate  Created pod: webtest-cns49
26m          26m          1     replication-controller      Normal  SuccessfulCreate  Created pod: webtest-9xgt3
13m          13m          1     replication-controller      Normal  SuccessfulCreate  Created pod: webtest-lmn54
praparns-MacBook-Pro:ReplicationController praparn$
```



Replication Controller (RC)

- Scale up replicas on RC

```
kubectl scale <option> --replicas <Type/Name>
```

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl scale --replicas=10 rc/webtest
replicationcontroller "webtest" scaled
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get rc
NAME      DESIRED   CURRENT   READY    AGE
webtest   10        10        5       32m
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
webtest-2hgdq 1/1     Running   0          18s
webtest-9xgt3 1/1     Running   0          32m
webtest-cns49 1/1     Running   0          32m
webtest-jbqdq 1/1     Running   0          18s
webtest-lgprd 1/1     Running   0          18s
webtest-lmn54 1/1     Running   0          19m
webtest-sqsjk 1/1     Running   0          18s
webtest-thhxsf 1/1    Running   0          18s
webtest-tx0px  1/1    Running   0          18s
webtest-v0n6s  1/1    Running   0          18s
```

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl scale --replicas=5 -f webtest_rc.yml
replicationcontroller "webtest" scaled
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get rc webtest
NAME      DESIRED   CURRENT   READY    AGE
webtest   5         5         5       40m
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
webtest-9xgt3 1/1     Running   0          40m
webtest-cns49 1/1     Running   0          40m
webtest-lmn54 1/1     Running   0          27m
webtest-tx0px  1/1    Running   0          7m
webtest-v0n6s  1/1    Running   0          7m
praparns-MacBook-Pro:ReplicationController praparn$
```



Replication Controller

- Check detail of create process on RC

```
Containers:
  webtest:
    Image:          labdocker/cluster:webservicelite
    Port:           5000/TCP
    Environment:   <none>
    Mounts:         <none>
    Volumes:        <none>

Events:
  FirstSeen  LastSeen  Count  From            SubObjectPath  Type    Reason               Message
  -----  -----  -----  -----  -----  -----  -----  -----
  42m       42m       1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-cns49
  42m       42m       1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-9xgt3
  42m       42m       1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-9m8tx
  29m       29m       1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-lmn54
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-lgprd
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-v0n6s
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-thhx
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-sqsjk
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-tx0px
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-2hgdq
  9m        9m        1      replication-controller  Normal   SuccessfulCreate  Created pod: webtest-jbqdd
  2m        2m        1      replication-controller  Normal   SuccessfulDelete  Deleted pod: webtest-lgprd
  2m        2m        1      replication-controller  Normal   SuccessfulDelete  Deleted pod: webtest-thhx
  2m        2m        1      replication-controller  Normal   SuccessfulDelete  Deleted pod: webtest-2hgdq
  2m        2m        1      replication-controller  Normal   SuccessfulDelete  Deleted pod: webtest-sqsjk
  2m        2m        1      replication-controller  Normal   SuccessfulDelete  Deleted pod: webtest-jbqdd
praparns-MacBook-Pro:ReplicationController praparn$ █
```



Replication Controller

- Test delete some Pods from command line

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
webtest-9m8tx  1/1     Running   0          11m
webtest-9xgt3  1/1     Running   0          11m
webtest-cns49  1/1     Running   0          11m
praparns-MacBook-Pro:ReplicationController praparn$ kubectl delete pods webtest-9m8tx
pod "webtest-9m8tx" deleted
```

- Recheck Pods unit and available

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
webtest-9xgt3  1/1     Running   0          21m
webtest-cns49  1/1     Running   0          21m
webtest-lmn54  1/1     Running   0          8m
praparns-MacBook-Pro:ReplicationController praparn$ curl http://192.168.99.100:32500
<H1> Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: Wed Jul  5 15:51:59 2017
praparns-MacBook-Pro:ReplicationController praparn$
```



Welcome Page from Container Python Lab

Checkpoint Date/Time: Wed Jul 5 15:55:31 2017



Replication Controller

- Cleanup Lab

```
praparns-MacBook-Pro:ReplicationController praparn$ kubectl delete -f webtest_svc.yml
service "webtest" deleted
praparns-MacBook-Pro:ReplicationController praparn$ kubectl delete -f webtest_rc.yml
replicationcontroller "webtest" deleted
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get rc
No resources found.
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get pods
No resources found.
praparns-MacBook-Pro:ReplicationController praparn$ kubectl get svc
NAME      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes  10.0.0.1    <none>        443/TCP   8d
praparns-MacBook-Pro:ReplicationController praparn$ █
```



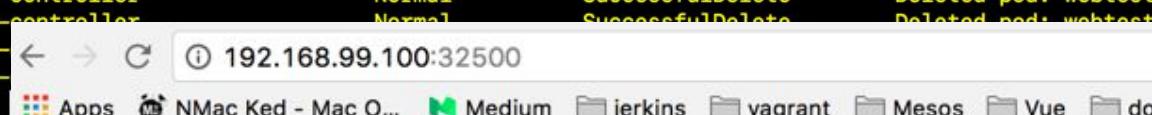
Workshop: RC

- Deploy replication controller

```
Containers:
  webtest:
    Image:          labdocker/cluster:webservicelite
    Port:           5000/TCP
    Environment:   <none>
    Mounts:        <none>
    Volumes:       <none>

Events:
  FirstSeen  LastSeen  Count  From            SubObjectPath  Type    Reason               Message
  -----  -----  -----  -----  -----  -----  -----  -----
  42m     42m      1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-cns49
  42m     42m      1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-9xgt3
  42m     42m      1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-9m8tx
  29m     29m      1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-lmn54
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-lgprd
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-v0n6s
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-thhx
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-sqsjk
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-tx0px
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-2hgdq
  9m      9m       1  replication-controller  Normal  SuccessfulCreate  Created pod: webtest-jbqdd
  2m      2m       1  replication-controller  Normal  SuccessfulDelete  Deleted pod: webtest-lgprd
  2m      2m       1  replication-controller  Normal  SuccessfulDelete  Deleted pod: webtest-thhx
  2m      2m       1  replication-controller  Normal  SuccessfulDelete  Deleted pod: webtest-2hgdq
  2m      2m       1  replication-          Normal  SuccessfulDelete  sqsjk
  2m      2m       1  replication-          Normal  SuccessfulDelete  jbqdd
praparns-MacBook-Pro:ReplicationController praparn$
```

192.168.99.100:32500



Apps NMac Ked - Mac O... Medium jenkins vagrant Mesos Vue doc

Welcome Page from Container Python Lab

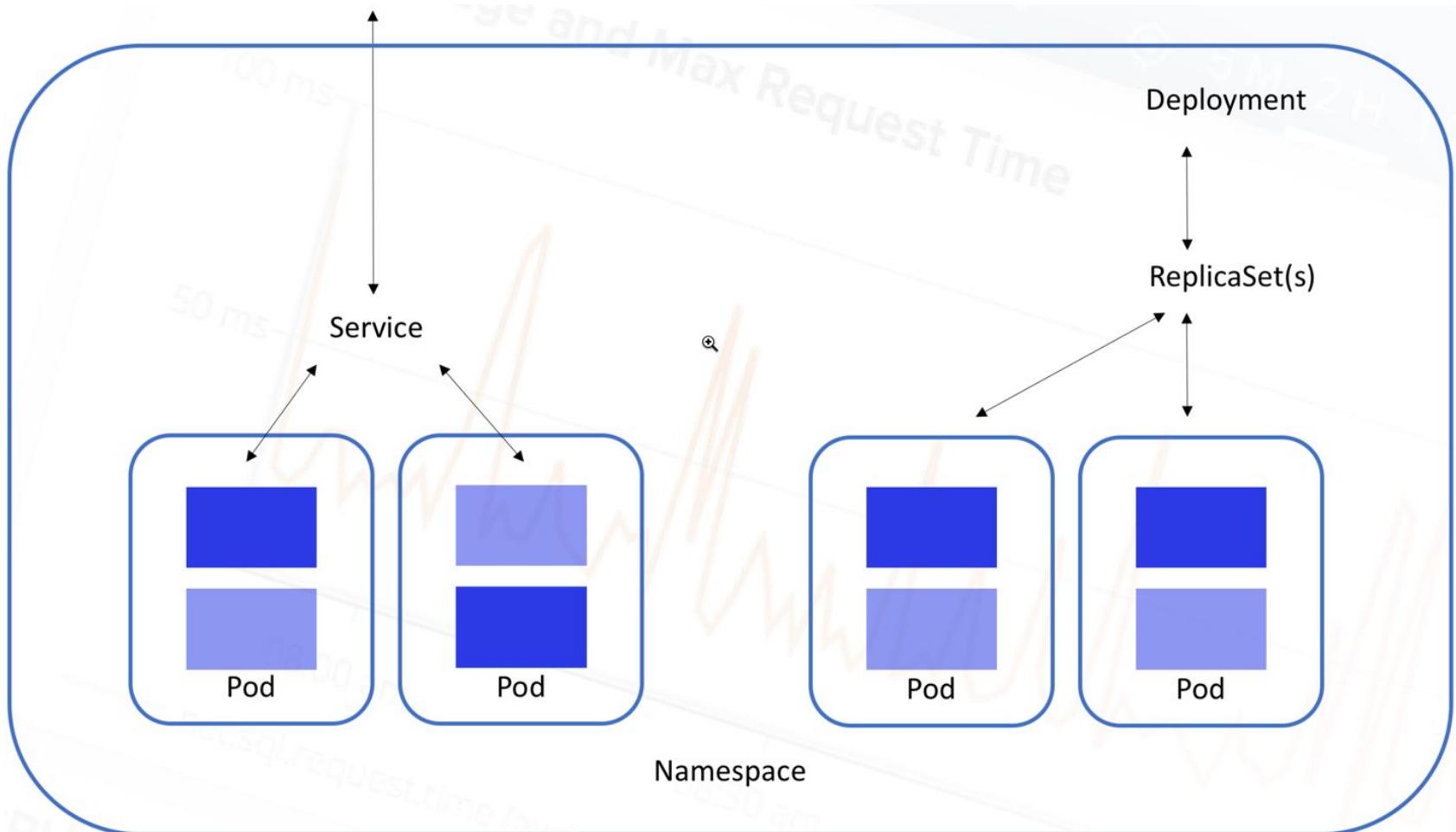
Checkpoint Date/Time: Wed Jul 5 15:55:31 2017



Deployment/RS and Update



Deployment/RS and Update



Deployment/RS and Update

- What is deployment/RS?
 - Deployment and RS (ReplicaSet) is set “next-generation of RC” by provide full function to maintain versioning of Pods in production (No downtime: On-the-fly)
 - Update new version (Rollout)
 - Revert old version (Rollback)
 - Pause/Resume process
 - Check status
 - By design deployment will order job to ReplicaSet(RS) for create Pods as design for up and run
 - When new version was rollout from deployment (Automatic)
 - Create new RS and start to scale as desired
 - Scale down existing RS to 0
 - Delete existing RS

Deployment/RS and Update

- ReplicaSet(RS) vs Replication Controller (RC)
 - RS is generation that evolution from RC with capability more dynamic
 - RC support label with method “Equality-based requirement”

```
selector:  
  name: web  
  owner: Praparn_L      I  
  version: "1.0"  
  module: WebServer  
  environment: development
```

- RS support label with method “Equality-based requirement” and

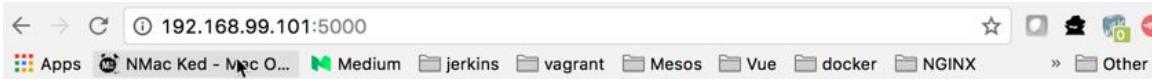
```
selector:  
  matchLabels:  
    environment: development  
  matchExpressions:  
    - {key: environment, operator: In, values: [development]}
```



Deployment/RS and Update

- Example

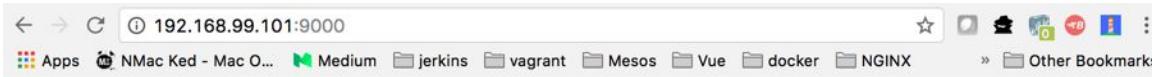
- labdocker/cluster:webservicelite_v1



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Thu Jul 6 15:19:27 2017

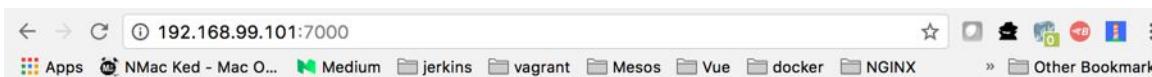
- labdocker/cluster:webservicelite_v1.51rc



Welcome Page from Container Python Lab Web Version 1.51 RC

Checkpoint Date/Time: Thu Jul 6 15:39:05 2017

- labdocker/cluster:webservicelite_v1.8ga



Welcome Page from Container Python Lab Web Version 1.80 GA

Checkpoint Date/Time: Thu Jul 6 15:36:54 2017



Deployment/RS and Update

- Example
 - Deployment “webtest”

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    replicas: 3
14    selector:
15      matchLabels:
16        name: web
17        owner: Praparn_L
18        version: "1.0"
19        module: WebServer
20        environment: development
21
22    template:
23      metadata:
24        labels:
25          name: web
26          owner: Praparn_L
27          version: "1.0"
28          module: WebServer
29          environment: development
30
31      spec:
32        containers:
33          - name: webtest
34            image: labdocker/cluster:webservicelite_v1
35            ports:
36              - containerPort: 5000
37                protocol: TCP
```

- SVC “webtest”

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    selector:
14      name: web
15      owner: Praparn_L
16      version: "1.0"
17      module: WebServer
18      environment: development
19
20    type: NodePort
21    ports:
22      - port: 5000
23        name: http
24        targetPort: 5000
25        protocol: TCP
26        nodePort: 32500
```



Deployment/RS and Update

- Create deployment “webtest” and ReplicaSet (RS)

```
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl create -f webtest_deploy.yml --record
deployment "webtest" created
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl get deployment webtest
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webtest   3          3          3           3           10s
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl get rs
NAME      DESIRED   CURRENT   READY      AGE
webtest-4261491039  3          3          3           16s
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
webtest-4261491039-b7gkv  1/1     Running   0          20s
webtest-4261491039-p231t  1/1     Running   0          20s
webtest-4261491039-rjzk4  1/1     Running   0          20s
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$
```

- Create svc “webtest”

```
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl create -f webtest_svc.yml --record
service "webtest" created
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl get svc
NAME      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
kubernetes  10.0.0.1    <none>        443/TCP     9d
webtest    10.0.0.97   <nodes>        5000:32500/TCP 4s
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$
```



Deployment/RS and Update

- Procedure for deployment operate
 - Check existing RS (0 when create new)
 - Create new RS
 - Scale RS to designed replicas

```
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl describe deployment/webtest
Name:           webtest
Namespace:      default
CreationTimestamp:   Thu, 06 Jul 2017 23:17:25 +0700
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
Annotations:    deployment.kubernetes.io/revision=1
Selector:       environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
Replicas:       3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:  RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 1 max unavailable, 1 max surge
Pod Template:
  Labels:      environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
```

```
Conditions:
  Type     Status  Reason
  ----
  Available  True    MinimumReplicasAvailable
OldReplicaSets: <none>
NewReplicaSet:  webtest-4261491039 (3/3 replicas created)
Events:
  FirstSeen  LastSeen  Count  From            SubObjectPath  Type        Reason          Message
  -----  -----  -----  -----  -----  -----  -----  -----
  10m       10m       1      deployment-controller      Normal      ScalingReplicaSet  Scaled up replica set webtest-4
261491039 to 3
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$
```



Deployment/RS and Update

- Deployment update strategy (RollingUpdate)
 - Update will start trigger when some of “spec:template” is changed

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12   spec:
13     replicas: 3
14     selector:
15       matchLabels:
16         name: web
17         owner: Praparn_L
18         version: "1.0"
19         module: WebServer
20         environment: development
21     template:
22       metadata:
23         labels:
24           name: web
25           owner: Praparn_L
26           version: "1.0"
27           module: WebServer
28           environment: development
29
30   
```

- Step for produce update (rollout) with Each Change (Default)
 - Create new RS for template change
 - Scale down existing RS to 1 (1 max unavailable)
 - Scale new RS as designed (No more than design +1: 1 max surge)
 - Delete existing RS
- Default deployment will allow 25% for lack (maxUnavailable) and 25% for over design (maxSurge)

Deployment/RS and Update

- Let's Talk about deployment's strategies !!!
 - On deployment we have many procedure for apply rollout strategy with deployment
 - Recreate (For Development): Destroy all pods and recreate it again.
 - RollingUpdate (Default): Create new version of pods and wait for it ready before terminate existing version
 - Blue/Green: Create new deployment for "green" version and reconfigure service for point to new deployment
 - Canary: Create new deployment with same "Label" but less replicas in begin and increase as need
 - A/B: Create new deployment and let's use service mesh for select some traffic to new deployment

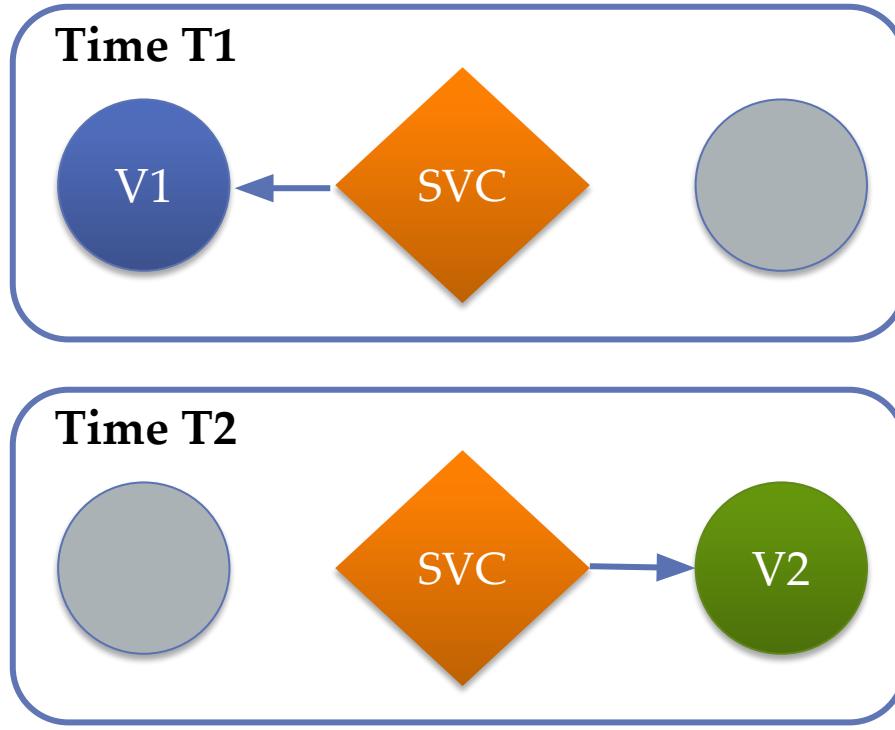
Ref:

<https://container-solutions.com/kubernetes-deployment-strategies/?fbclid=IwAR1T3mdkK46HmI7ckla6iEjtAuKKnijLWRcsrK3Jetzb5BOB21akiYBgm5Y>



Deployment/RS and Update

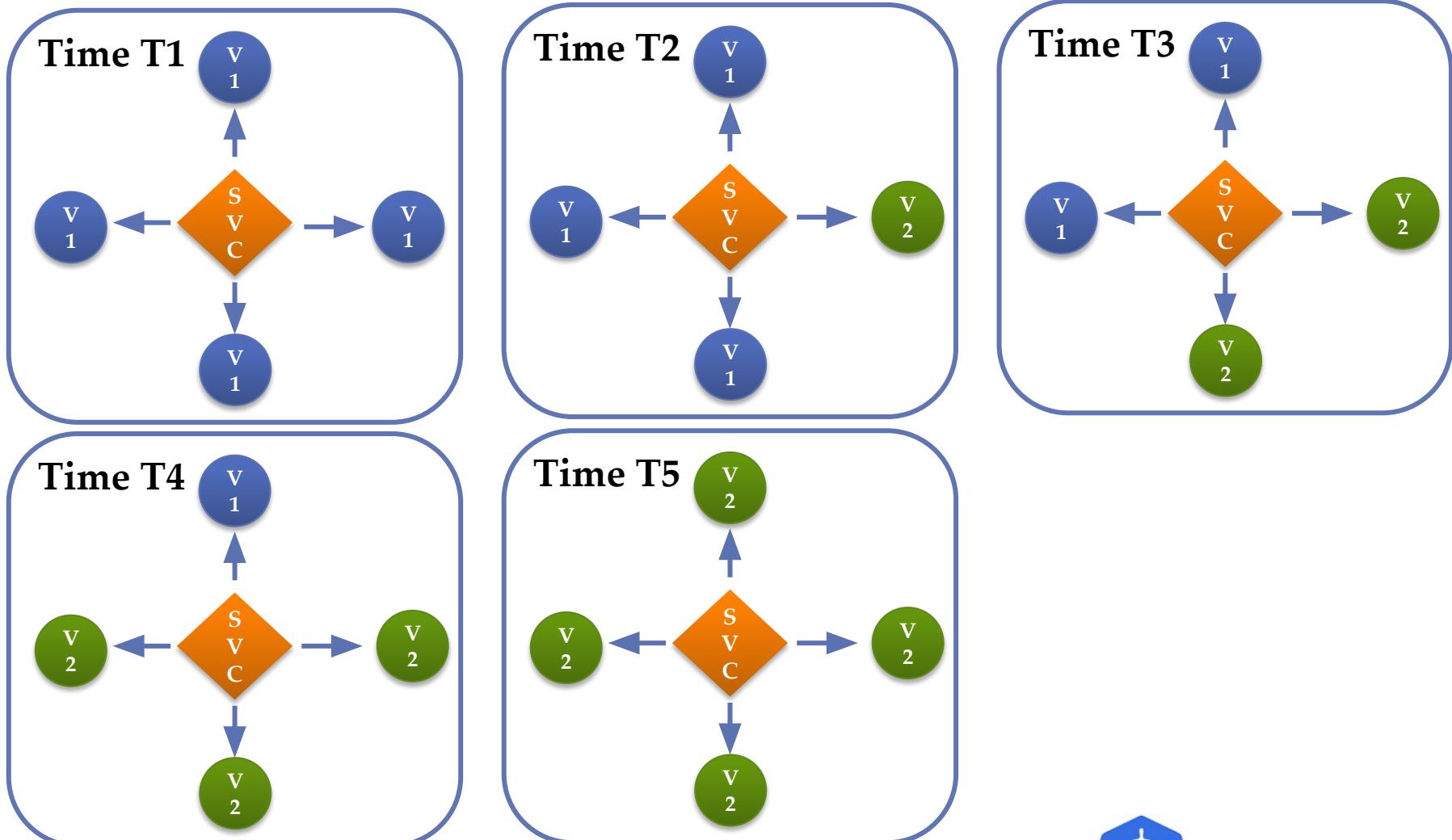
- Recreate (For Development)



```
! webtest_deploy.yml •  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: webtest  
5    labels:  
6      name: web  
7      owner: Praparn_L  
8    version: "1.0"  
9    module: WebServer  
10   environment: development  
11 spec:  
12   replicas: 1  
13   strategy:  
14     type: Recreate  
15   selector:  
16     matchLabels:  
17       name: web  
18       owner: Praparn_L
```

Deployment/RS and Update

- RollingUpdate (Default)



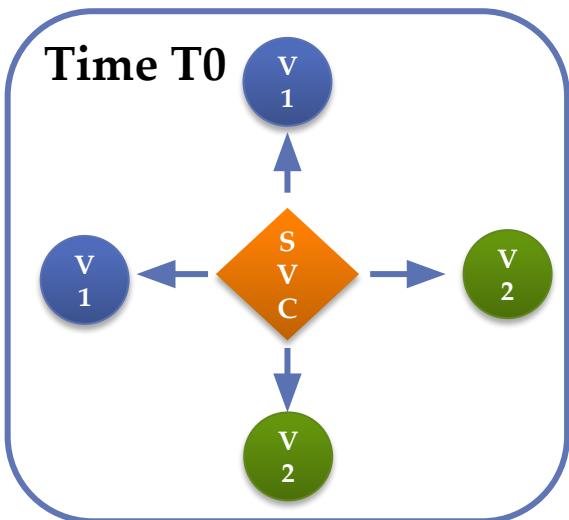
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Deployment/RS and Update

- RollingUpdate (Default)



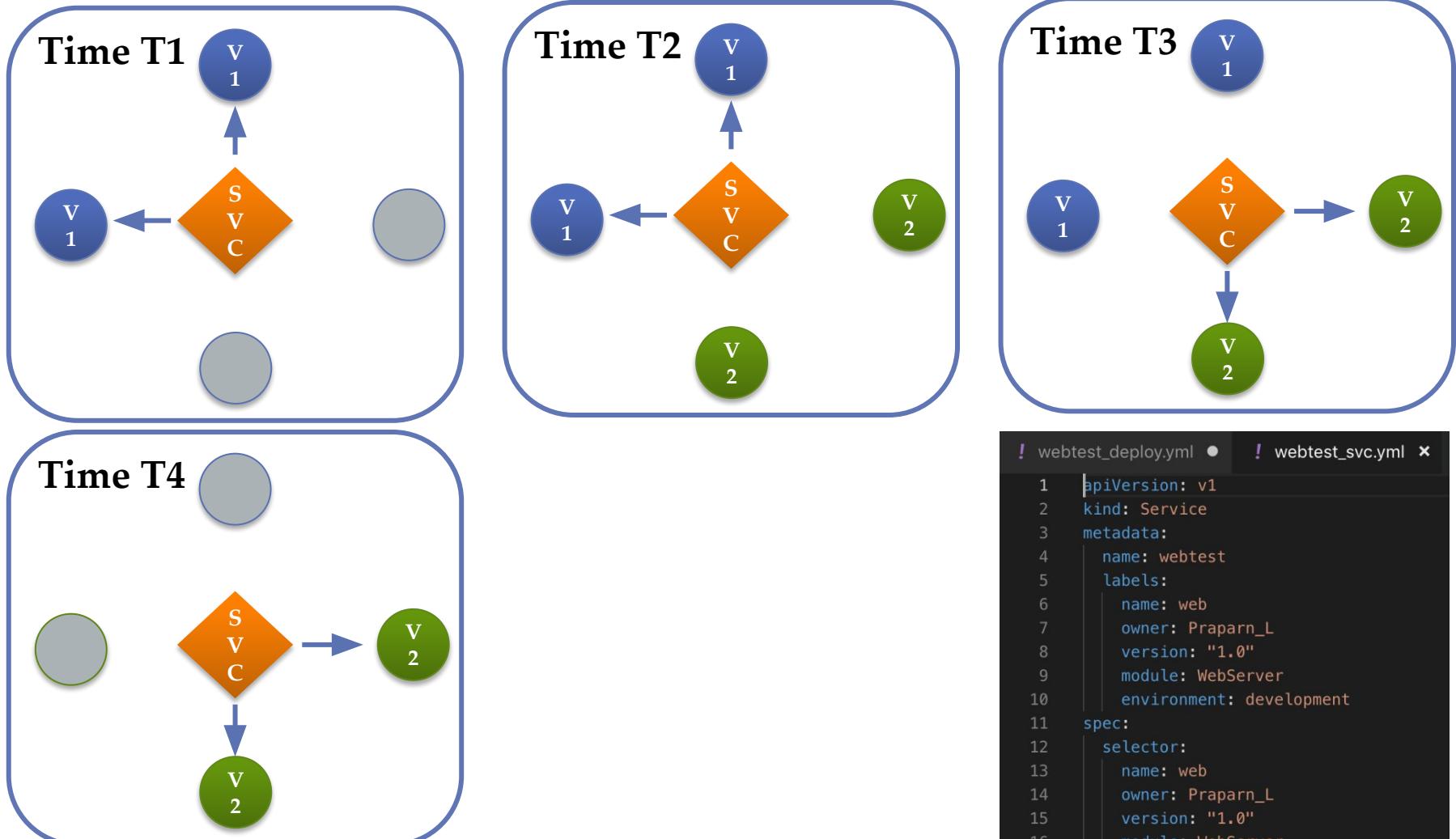
```
! webtest_deploy.yml ●  
1  apiVersion: apps/v1  
2  kind: Deployment  
3  metadata:  
4    name: webtest  
5    labels:  
6      name: web  
7      owner: Praparn_L  
8      version: "1.0"  
9      module: WebServer  
10     environment: development  
11 spec:  
12   replicas: 4  
13   strategy:  
14     type: RollingUpdate  
15     rollingUpdate:  
16       maxUnavailable: 1  
17       maxSurge: 1  
18   selector:  
19     matchLabels:  
20       name: web  
21       owner: Praparn_L
```



kubernetes
by Google

Deployment/RS and Update

- Blue/Green



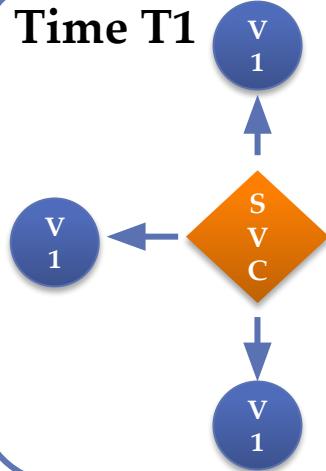
Kubernetes: Production Workload Orchestration

```
! webtest_deploy.yml • ! webtest_svc.yml x
1   apiVersion: v1
2   kind: Service
3   metadata:
4     name: webtest
5     labels:
6       name: web
7       owner: Praparn_L
8       version: "1.0"
9       module: WebServer
10      environment: development
11
12   spec:
13     selector:
14       name: web
15       owner: Praparn_L
16       version: "1.0"
17       module: WebServer
18       environment: development
```

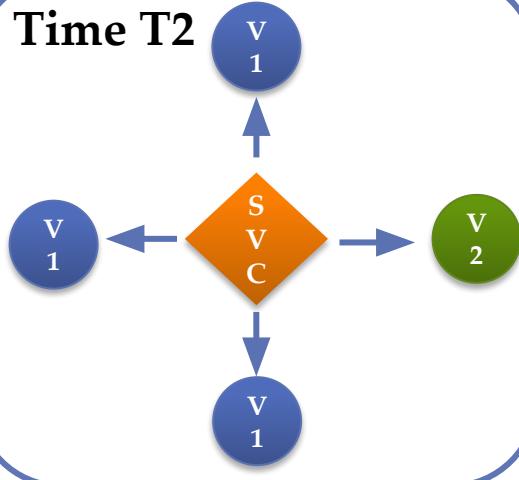
Deployment/RS and Update

- Canary: (2 Deployment same label)

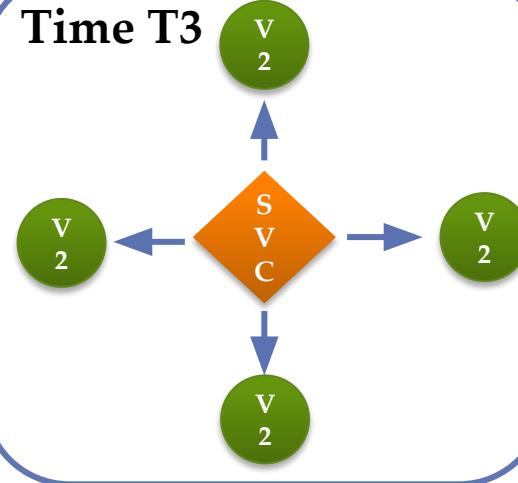
Time T1



Time T2



Time T3



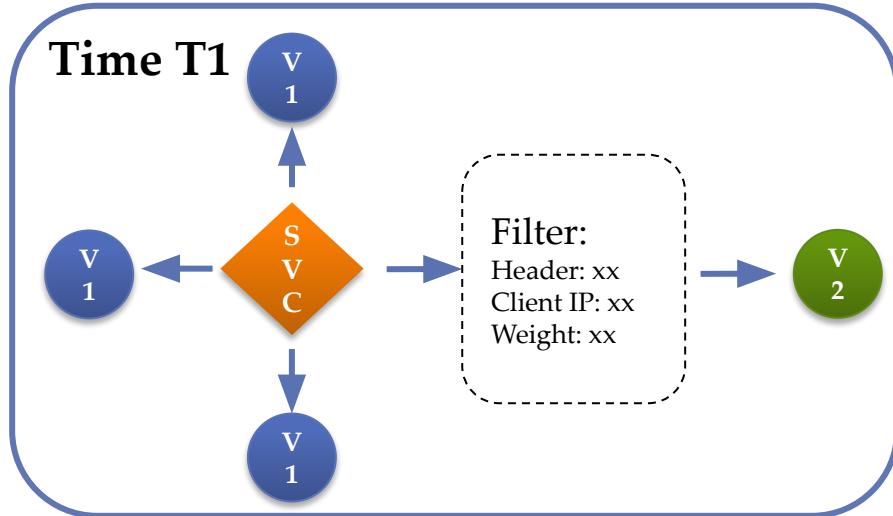
```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: webtest_V1
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11 spec:
12   replicas: 3
```

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: webtest_V2
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11 spec:
12   replicas: 1
```

```
! webtest_deploy.yml • ! webtest_svc.yml x
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11 spec:
12   selector:
13     name: web
14     owner: Praparn_L
15     version: "1.0"
16     module: WebServer
17     environment: development
```

Deployment/RS and Update

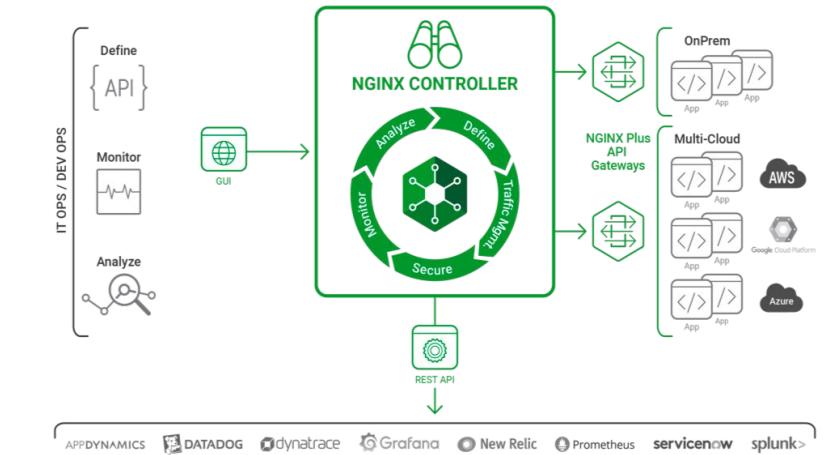
- A/B: (2 Deployment with Service Mesh)



Istio

Connect, secure, control, and observe services.

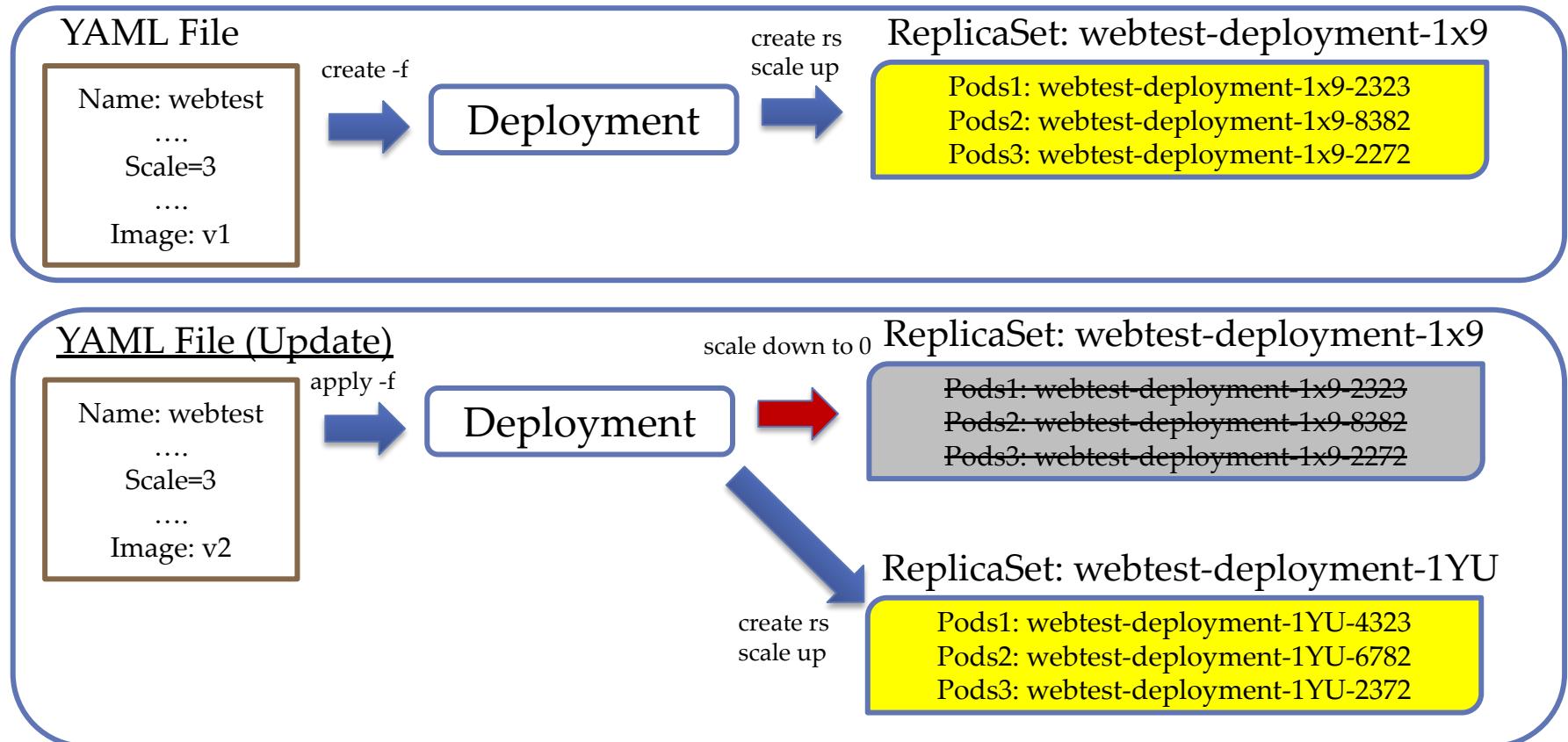
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Deployment/RS and Update

- Deployment update strategy (RollingUpdate)



Deployment/RS and Update

- How to be grateful rollout/rollback
 - Set the rollingupdate's strategy

replicas: 3

strategy:

 type: RollingUpdate

 rollingUpdate:

 maxUnavailable: 50%

 maxSurge: 100%

- Setup “terminationGracePeriodSeconds” (After SIGTERM)

spec:

 terminationGracePeriodSeconds: 30

- Setup “readiness” for indicate application is ready (Next chapter) and coding for set readiness fail when get “SIGTERM”
- Setup “readinessGates” for additional healthcheck for external hookback before start open pods

Deployment/RS and Update

- Deployment update strategy (Rollout)
 - Trigger Change on Deployment
 - **Method1: Set online**

```
kubectl set <Property> <Type/Name> Variable=Value
```

- Ex: kubectl set image deployment/webtest webtest:betaversion

- **Method2: Edit online**

```
kubectl edit <Type/Name>
```

- Ex: kubectl edit deployment/webtest

- **Method3: Edit YAML File and Apply**

```
Edit file <FileName>.YML
```

```
kubectl apply -f <FileName>
```

Deployment/RS and Update

- Check status of deployment's update (rollout)
 - Use “rollout” utility

```
kubectl rollout <Option> <Type/Name>
```

- Ex:
 - Check status of rollout process:
 - kubectl rollout status deployment/webtest
 - Check history of rollout process:
 - Kubectl rollout history deployment/webtest
 - Rollback rollout process:
 - kubectl rollout undo deployment/webtest --to-revision=2
 - Pause/Resume process:
 - kubectl rollout pause/resume deployment/webtest



Deployment/RS and Update

- Deployment update strategy
 - Set new image on deployment

```
ubuntu@ip-10-0-1-211:~$ kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.51rc --record=true
deployment.extensions/webtest image updated
ubuntu@ip-10-0-1-211:~$ kubectl rollout status deployment/webtest
Waiting for deployment "webtest" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "webtest" rollout to finish: 1 old replicas are pending termination...
deployment "webtest" successfully rolled out
ubuntu@ip-10-0-1-211:~$ kubectl get deployment
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webtest   3          3          3           3           98s
ubuntu@ip-10-0-1-211:~$ kubectl get rs
NAME      DESIRED   CURRENT   READY     AGE
webtest-594bdc7f5d   3          3          3           17s
webtest-7dc5bdccdd7 0          0          0           102s
ubuntu@ip-10-0-1-211:~$ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
webtest-594bdc7f5d-2pqwf  1/1     Running   0          22s
webtest-594bdc7f5d-fngzv  1/1     Running   0          19s
webtest-594bdc7f5d-wwkqj  1/1     Running   0          20s
ubuntu@ip-10-0-1-211:~$ kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.8ga --record=true
deployment.extensions/webtest image updated
ubuntu@ip-10-0-1-211:~$ kubectl rollout status deployment/webtest
deployment "webtest" successfully rolled out
ubuntu@ip-10-0-1-211:~$
```



Deployment/RS and Update

- Deployment update strategy
 - Rollout History

```
ubuntu@ip-10-0-1-211:~$ kubectl rollout history deployment/webtest
deployment.extensions/webtest
REVISION  CHANGE-CAUSE
1          kubectl create --filename=https://raw.githubusercontent.com/praparn/kubernetes_20180701/master/WorkShop_1.4_Deployment/webtest_deploy.yml --record=true
2          kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.51rc --record=true
3          kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.8ga --record=true

ubuntu@ip-10-0-1-211:~$ █
```



Deployment/RS and Update

- Deployment update strategy
 - Rollback process

```
[ubuntu@ip-10-0-1-211:~$ kubectl rollout undo deployment/webtest --to-revision=2
deployment.extensions/webtest
[ubuntu@ip-10-0-1-211:~$ kubectl rollout status deployment/webtest
deployment "webtest" successfully rolled out
[ubuntu@ip-10-0-1-211:~$ curl http://10.0.1.211:32500
<H1> Welcome Page from Container Python Lab Web Version 1.51 RC </H1>Checkpoint Date/Time: Mon Oct 29 08:10:17 2018
ubuntu@ip-10-0-1-211:~$ ]
```

- Rollout History

```
[ubuntu@ip-10-0-1-211:~$ kubectl rollout history deployment/webtest
deployment.extensions/webtest
REVISION  CHANGE-CAUSE
1          kubectl create --filename=https://raw.githubusercontent.com/praparn/kubernetes_20180701/master/WorkShop_1.4_Deployment/webtest_deploy.yml --record=true
3          kubectl set image deployment/webtest webtest=labdocker/cluster:webserviceelite_v1.8ga --record=true
4          kubectl set image deployment/webtest webtest=labdocker/cluster:webserviceelite_v1.51rc --record=true
ubuntu@ip-10-0-1-211:~$ ]
```



Volume



Volume

- By default all container in same Pods will share storage (emptyDir)
 - This storage will keep all read/write for all container in Pods
 - Persistent storage for container as long as Pods still available
 - Container crash not effect to this storage type
 - When Pods was delete from node. emptyDir was deleted forever
- Data in container/Pods is “ephemeral” that may loss when Pods was restart/distribute
- Kubernetes support many type of volume on system



Volume

- List of support storage
 - EmptyDir
 - **hostPath**
 - gcePersistent
 - awsElasticBlockStore
 - Nfs
 - Iscsi
 - fc(fibrechannel)
 - Flocker
 - Glusterfs
 - Rbd
 - Cephfs
 - gitRepo (deprecated)
 - Secret
 - **persistentVolumeClaim (PVC) (Talk Day2)**
 - downwardAPI
 - Projected
 - azureDisk
 - azureFile
 - vsphereVolume
 - Quobyte
 - PortworxVolume
 - ScaleIO
 - StorageOS
 - Local
 - ConfigMap
 - Secret
 - csi

Volume

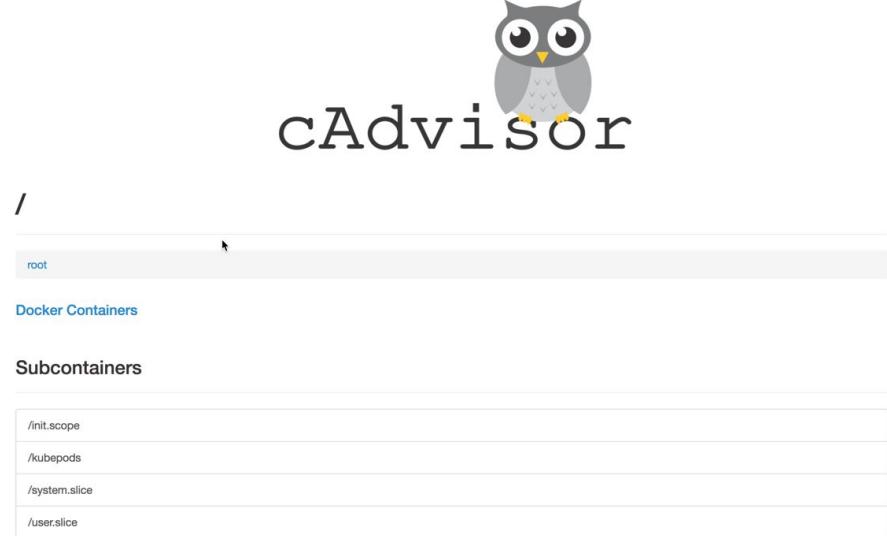
- hostPath:
 - Mount file/directory on host to Pods
 - Recommend for Pods that need read some data on host path (Such as cAdvisor etc)
 - Consideration
 - hostPath may effect to some of path/file not equal on different host
 - Kubernetes resource scheduling cannot check readiness of hostPath
 - Some hostPath may need privilege for access. This need to consideration.



Volume

- hostPath:

```
Users > paparnlueangphoonlap > Work > Kubernetes > Train
31   labels:
32     name: cadvisor
33     owner: Praparn_L
34     version: "1.0"
35     module: Monitor
36     environment: development
37 spec:
38   replicas: 1
39   selector:
40     matchLabels:
41       name: cadvisor
42       owner: Praparn_L
43       version: "1.0"
44       module: Monitor
45       environment: development
46 template:
47   metadata:
48     labels:
49       name: cadvisor
50       owner: Praparn_L
51       version: "1.0"
52       module: Monitor
53       environment: development
54 spec:
55   containers:
56     - name: cadvisor
57       volumeMounts:
58         - mountPath: /var/run
59           name: volrun
60         - mountPath: /sys
61           name: volsys
62         - mountPath: /var/lib/containerd/
63           name: voldocker
64           image: labdockerc/cadvisor:latest
65       ports:
66         - containerPort: 8080
67           protocol: TCP
68       volumes:
69         - name: volrun
70           hostPath:
71             path: /var/run
72         - name: volsys
73           hostPath:
74             path: /sys
75         - name: voldocker
76           hostPath:
77             path: /var/lib/containerd/
```



Volume

- hostPath:
- Container create file

Host check file

```
praparnlueangphoonlap — Terminal MAC Pro — kubectl exec -it cadvisor
~ — Terminal MAC Pro — ssh + minikube ssh
praparn-MacBook-Pro% kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
cAdvisor-1335144146-jgm3h   1/1     Running   0          15m
praparn-MacBook-Pro% kubectl exec -it cAdvisor-1335144146-jgm3h sh
[/ # touch /var/lib/docker/test_cAdvisor.txt
[/ # ls -lh /var/lib/docker
total 52
drwx----- 28 root      root      4.0K Sep 24 15:57 containers
drwx-----  3 root      root      4.0K Sep 24 04:02 image
drwxr-x---  3 root      root      4.0K Sep 24 04:02 network
drwx----- 155 root      root     20.0K Sep 24 15:57 overlay
drwx-----  2 root      root      4.0K Sep 24 04:02 swarm
-rw-r--r--  1 root      root      0 Sep 24 16:12 test_cAdvisor.txt
-rw-r--r--  1 root      root      0 Sep 24 16:00 testfile.txt
drwx-----  2 root      root      4.0K Sep 24 15:13 tmp
drwx-----  2 root      root      4.0K Sep 24 04:02 trust
drwx-----  2 root      root      4.0K Sep 24 09:27 volumes
/ #
```

```
praparn-MacBook-Pro% minikube ssh
$ sudo su -
# ls /var/lib/docker
containers image network overlay swarm test_cAdvisor.txt testfile.txt tmp trust volumes
#
```



Volume

```
Users > paparnlueangphoonlap > Work > Kubernetes > Training > k  
1  apiVersion: "v1"  
2  kind: Pod  
3  metadata:  
4    name: webtest  
5    labels:  
6      name: web  
7      owner: Paparn_L  
8      version: "1.0"  
9      module: WebServer  
10     environment: development  
11 spec:  
12   containers:  
13     - name: webtest  
14       image: labdocker/cluster:webservicelite_v1  
15       ports:  
16         - containerPort: 5000  
17         | protocol: TCP  
18       volumeMounts:  
19         - mountPath: /temp  
20         | name: voldocker  
21       volumes:  
22         - name: voldocker  
23           hostPath:  
24             path: /var/lib/containerd/
```

```
paparns-MacBook-Pro% kubectl create -f webtest_pod.yml  
pod "webtest" created  
paparns-MacBook-Pro% kubectl get pods  
NAME          READY   STATUS    RESTARTS   AGE  
cadvisor-1335144146-jgm3h   1/1    Running   0          21m  
webtest        1/1    Running   0          3s  
paparns-MacBook-Pro% kubectl exec -it webtest sh  
/usr/src/app # ls -lh /temp  
total 56  
drwx----- 32 root    root    4.0K Sep 24 16:18 containers  
drwx-----  3 root    root    4.0K Sep 24 04:02 image  
drwxr-x---  3 root    root    4.0K Sep 24 04:02 network  
drwx----- 163 root    root   24.0K Sep 24 16:18 overlay  
drwx-----  2 root    root    4.0K Sep 24 04:02 swarm  
-rw-r--r--  1 root    root     0 Sep 24 16:12 test_cAdvisor.txt  
-rw-r--r--  1 root    root     0 Sep 24 16:00 testfile.txt  
drwx-----  2 root    root    4.0K Sep 24 15:13 tmp  
drwx-----  2 root    root    4.0K Sep 24 04:02 trust  
drwx-----  2 root    root    4.0K Sep 24 09:27 volumes  
/usr/src/app # █
```



Workshop: Volume

```
Users > praparnlueangphoona > Work > Kubernetes > Train
31   labels:
32     name: cadvisor
33     owner: Praparn_L
34     version: "1.0"
35     module: Monitor
36     environment: development
37 spec:
38 replicas: 1
39 selector:
40   matchLabels:
41     name: cadvisor
42     owner: Praparn_L
43     version: "1.0"
44     module: Monitor
45     environment: development
46 template:
47 metadata:
48   labels:
49     name: cadvisor
50     owner: Praparn_L
51     version: "1.0"
52     module: Monitor
53     environment: development
54 spec:
55   containers:
56     - name: cadvisor
57       volumeMounts:
58         - mountPath: /var/run
59           name: volrun
60         - mountPath: /sys
61           name: volsys
62         - mountPath: /var/lib/containerd/
63           name: voldocker
64           image: labdocker/cadvisor:latest
65           ports:
66             - containerPort: 8080
67               protocol: TCP
68       volumes:
69         - name: volrun
70           hostPath:
71             path: /var/run
72         - name: volsys
73           hostPath:
74             path: /sys
75         - name: voldocker
76           hostPath:
77             path: /var/lib/containerd/
```



/

root

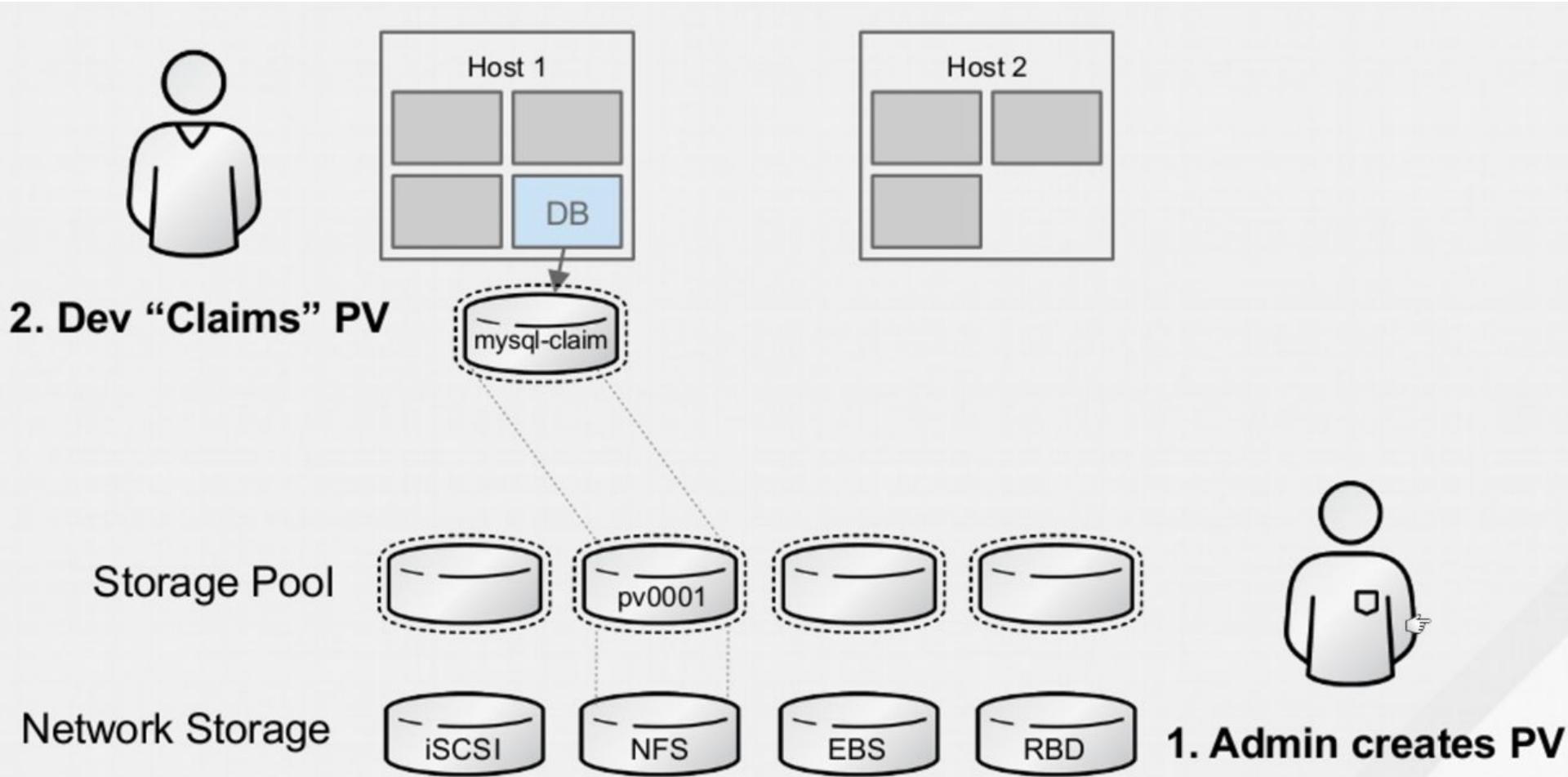
Docker Containers

Subcontainers

- /init.scope
- /kubepods
- /system.slice
- /user.slice



Volume



Liveness and Readiness Probe



Liveness and Readiness Probe

- Pods are you still alright ?
 - Normally kubernetes will check Pods status and process criteria for orchestrator following “restartPolicy”
 - Always (Default)
 - On-Failure
 - None
 - Pods status is depend on “ContainerState” in that Pods (1-M)
 - Waiting (ContainerStateWaiting)
 - Running (ContainerStateRunning)
 - Terminated (ContainerStateTerminated)



Liveness and Readiness Probe

- Pods are you still alright ?
 - When Pods receive status from container. It will set Pods's status to inform kubelet for operate with Pods (Remain / Terminated / Restart)
 - Pods Phase (Phase Value)
 - Pending: When initial Pods and loading images for container
 - Running: Some container running but not all
 - Successed: All container fullfill running
 - Failed: All container exit fail
 - Unknow: Can not monitor state
- Do we need more specific health-check?
 - Depend on container (application) fail condition.
 - If container fail condition will make process “crash” or effect to container down/unhealth. This no need to operate more check.
 - If container fail condition may occur inside application that possible process still online. This need more health check

Liveness and Readiness Probe

- Container probe process
 - Kubernetes have function for make properly make sure that container (inside Pods) still healthy by three options
 - ExecAction: Execute some command inside container (expect result: return 0)
 - TCPSocketAction: Start TCP communication on specific port of container (expect result: port open)
 - HTTPGetAction: Send http/https request to specific port and path (expect result: return 200 for OK)
 - Probe's result
 - Success
 - Failed
 - Unknown
 - livenessProbe and readinessProbe Difference
 - livenessProbe: check status of container is running properly or not ?
 - readinessProbe: check readiness for process service or not ?

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- Container Lifecycle Hooks
 - On kubernetes we can provide some process to hook on container Life cycle that kubernetes support in 2 event
 - PostStart:
 - Run some activity immediate when container is created (No guarantee to execute before “ENTRYPOINT” of container)
 - PreStop:
 - Run some activity immediate before container terminate (Delete, Liveness Fail) This is will blocking to do this task before terminate

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

```
41      - name: ES_JAVA_OPTS
42        value: -Xms5g -Xmx5g
43      - name: bootstrap.memory_lock
44        value: "false"
45      - name: cluster.name
46        value: pam-es-docker-cluster
47      - name: xpack.security.enabled
48        value: "false"
49    image: "3dsinteractive/elasticsearch-std:6.6"
50    lifecycle:
51      postStart:
52        exec:
53          command: ["/bin/sh", "-c", "chown -R 1000:1000 /usr/share/elasticsearch/data && chmod -R 777 /usr/share/elasticsearch/data"]
54
55  #       imagePullPolicy: Always
56  ports:
57    - name: "elastic9200"
58      containerPort: 9200
59    - name: "elastic9300"
60      containerPort: 9300
61
62  resources:
```

```
1  apiVersion: app/v1
2  kind: Deployment
3  metadata:
4    name: nginx
5  spec:
6    template:
7      metadata:
8        labels:
9          app: nginx
10     spec:
11       containers:
12         - name: nginx
13           image: nginx
14           ports:
15             - containerPort: 80
16           lifecycle:
17             preStop:
18               exec:
19                 command: ["nginx","-s","quit"]
```

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- ExecAction:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12    spec:
13      replicas: 1
14      selector:
15        matchLabels:
16          name: web
17          owner: Praparn_L
18          version: "1.0"
19          module: WebServer
20          environment: development
21      template:
22        metadata:
23          labels:
24            name: web
25            owner: Praparn_L
26            version: "1.0"
27            module: WebServer
28            environment: development
```

```
28
29   spec:
30     containers:
31       - name: webtest
32         image: labdocker/cluster:webservicelite_v1
33         ports:
34           - containerPort: 5000
35             protocol: TCP
36         readinessProbe:
37           exec:
38             command:
39               - cat
40               - /usr/src/app/main.py
41             initialDelaySeconds: 15
42             periodSeconds: 5
43         livenessProbe:
44           exec:
45             command:
46               - cat
47               - /usr/src/app/main.py
48             initialDelaySeconds: 15
49             periodSeconds: 15
```

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration

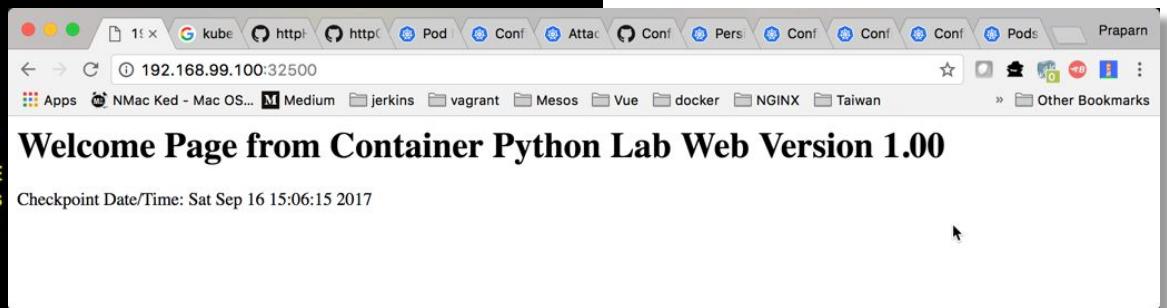


kubernetes
by Google

Liveness and Readiness Probe

- ExecAction:

```
praparns-MacBook-Pro% kubectl create -f webtest_deploy_liveness_readiness_exec.yml
    kubectl create -f webtest_svc.yml
deployment "webtest" created
service "webtest" created
praparns-MacBook-Pro% kubectl get svc/webtest
NAME      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
webtest   10.0.0.127   <nodes>       5000:32500/TCP  20s
praparns-MacBook-Pro% kubectl get deployment/webtest
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webtest   1         1         1           1           33s
praparns-MacBook-Pro% kubectl describe deployment/webtest
Name:           webtest
Namespace:      default
CreationTimestamp: Sat, 16 Sep 2017 22:04:07 +0700
Labels:          environment=development
                 module=WebServer
                 name=web
                 owner=Praparn_L
                 version=1.0
Annotations:    deployment.kubernetes.io/revision=1
Selector:        environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
Replicas:        1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 1 max unavailable, 1 max surge
Pod Template:
  Labels:      environment=development
               module=WebServer
               name=web
               owner=Praparn_L
               version=1.0
  Containers:
    webtest:
      Image:      labdocker/cluster:webservicelite_v1
      Port:       5000/TCP
      Liveness:   exec [cat /usr/src/app/main.py] delay=15s timeout=1s period=15s #success=1 #failure=3
      Readiness:  exec [cat /usr/src/app/main.py] delay=15s timeout=1s period=5s #success=1 #failure=3
      Environment: <none>
```



Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- ExecAction:

```
praparns-MacBook-Pro% kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
webtest-336910061-26pdb  1/1     Running   0          4m
praparns-MacBook-Pro% kubectl exec -it webtest-336910061-26pdb sh
/usr/src/app # ls
docker-compose.yml      dockerfile_python_lite  nginx.conf
dockerfile_nginx         main.py                  requirements.txt
dockerfile_python        mainlite.py             requirementslite.txt
/usr/src/app # rm main.py
/usr/src/app # exit
praparns-MacBook-Pro%
```

```
praparns-MacBook-Pro% kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
webtest-336910061-26pdb  1/1     Running   1          7m
praparns-MacBook-Pro% kubectl describe pods/webtest-336910061-26pdb
Name:           webtest-336910061-26pdb
Namespace:      default
Node:           minikube/192.168.99.100
Start Time:    Sat, 16 Sep 2017 22:04:07 +0700
Labels:         environment=development
               module=WebServer
               name=web
               owner=Praparn_L
               pod-template-hash=336910061
               version=1.0
Annotations:   kubernetes.io/created-by={"kind":"SerializedReference","apiVersion":"v1","reference":{"kind":"ReplicaSet","namespace":"default","name":"webtest-336910061","uid":"4992f987-9af0-11e7-bc1a-080027fb95be",...}
Status:        Running
```

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- ExecAction:

```
Events:
FirstSeen    LastSeen    Count  From          SubObjectPath             Type   Reason        Message
-----    -----    ----  -----          -----             -----   -----   -----
7m          7m         1      default-scheduler
y assigned webtest-336910061-26pdb to minikube
7m          7m         1      kubelet, minikube
 SetUp succeeded for volume "default-token-pcw4h"
2m          2m         8      kubelet, minikube  spec.containers{webtest}  Warning  Unhealthy
probe failed: cat: can't open '/usr/src/app/main.py': No such file or directory
2m          2m         3      kubelet, minikube  spec.containers{webtest}  Warning  Unhealthy
Liveness probe failed: cat: can't open '/us
r/src/app/main.py': No such file or directory
7m          2m         2      kubelet, minikube  spec.containers{webtest}  Normal  Pulled  Container image "labdocker/cluster:webservicelite_v
1" already present on machine
7m          2m         2      kubelet, minikube  spec.containers{webtest}  Normal  Created  Created container
7m          2m         2      kubelet, minikube  spec.containers{webtest}  Normal  Started  Started container
2m          2m         1      kubelet, minikube  spec.containers{webtest}  Normal  Killing  Killing container with id docker://webtest:pod "web
test-336910061-26pdb_default(499cef14-9af0-11e7-bc1a-080027fb95be)" container "webtest" is unhealthy, it will be killed and re-created.
praparns-MacBook-Pro% █
```

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- TCP Socket Action:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11 spec:
12   replicas: 1
13   selector:
14     matchLabels:
15       name: web
16       owner: Praparn_L
17       version: "1.0"
18       module: WebServer
19       environment: development
20   template:
21     metadata:
22       labels:
23         name: web
24         owner: Praparn_L
25         version: "1.0"
26         module: WebServer
27         environment: development
```

```
28   spec:
29     containers:
30       - name: webtest
31         image: labdocker/cluster:webservicelite_v1
32         ports:
33           - name: webservice
34             containerPort: 5000
35             protocol: TCP
36             readinessProbe:
37               tcpSocket:
38                 port: 5000
39               initialDelaySeconds: 15
40               periodSeconds: 5
41             livenessProbe:
42               tcpSocket:
43                 port: 5000
44               initialDelaySeconds: 15
45               periodSeconds: 15
```

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

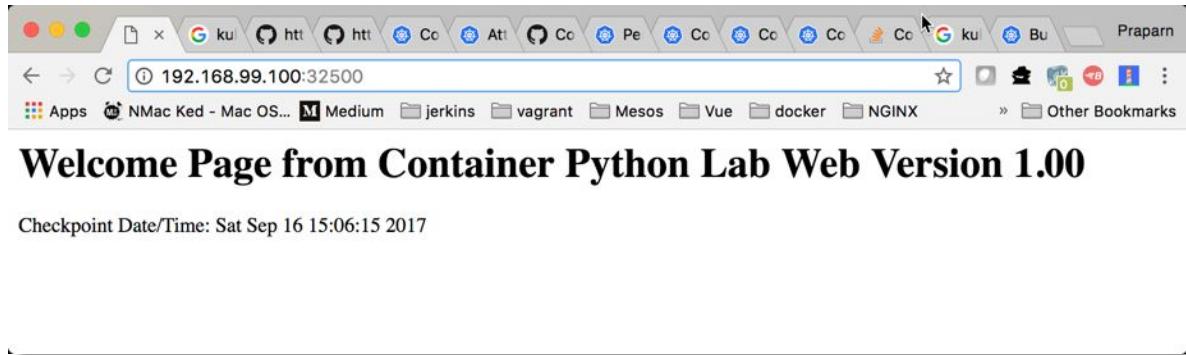
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- TCPSocketAction:



```
[praparns-MacBook-Pro% kubectl create -f webtest_deploy_liveness_readiness_port.yml
deployment "webtest" created
[praparns-MacBook-Pro% kubectl get deployment/webtest
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webtest   1          1          1           0           14s
[praparns-MacBook-Pro% kubectl get pods --show-labels
NAME                  READY   STATUS    RESTARTS   AGE   LABELS
webtest-3579274848-1w2mn   0/1     Running   0          18s   environment=development,modu
hash=3579274848,version=1.0
```

Events:							
FirstSeen	LastSeen	Count	From	SubObjectPath	Type	Reason	Message
2m	2m	1	default-scheduler		Normal	Scheduled	Successful
y	assigned webtest-3579274848-1w2mn to minikube				Normal	SuccessfulMountVolume	MountVolume
.	SetUp succeeded for volume "default-token-pcw4h"				Normal	Pulled	Container i
mage	"labdocke/cluster:webservicelite_v1" already present on machine			spec.containers{webtest}	Normal	Created	Created con
tainer		1	kubelet, minikube	spec.containers{webtest}	Normal	Started	Started con
tainer		1	kubelet, minikube	spec.containers{webtest}	Normal		

Ref: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- TCPSocketAction:

```
instruction.txt ! webtest_deploy_liveness_readiness_port.yml •  
21     spec:  
22       containers:  
23         - name: webtest  
24           image: labdocker/cluster:webserviceelite_v1  
25           ports:  
26             - name: webservice  
27               containerPort: 5000  
28               protocol: TCP  
29             readinessProbe:  
30               tcpSocket:  
31                 port: 3000  
32               initialDelaySeconds: 15  
33               periodSeconds: 5  
34             livenessProbe:  
35               tcpSocket:  
36                 port: 3000  
37               initialDelaySeconds: 15  
38               periodSeconds: 15
```

```
praparns-MacBook-Pro% kubectl apply -f webtest_deploy_liveness_readiness_port.yml  
deployment "webtest" configured  
praparns-MacBook-Pro% kubectl get pods  
NAME          READY     STATUS    RESTARTS   AGE  
webtest-1810247028-x5xdf   0/1      Running   0          13s  
praparns-MacBook-Pro% kubectl describe pods/webtest-1810247028-x5xdf  
Name:           webtest-1810247028-x5xdf  
Namespace:      default  
Node:           minikube/192.168.99.100  
Start Time:     Sat, 16 Sep 2017 22:32:18 +0700  
Labels:          environment=development  
                  module=WebServer  
                  name=web  
                  owner=Praparn_L  
                  pod-template-hash=1810247028  
                  version=1.0  
Annotations:    kubernetes.io/created-by={"kind":"SerializedReference","apiVersion":"v1","reference":{"kind":"ReplicaSet","namespace":"default","name":"webtest-1810247028","uid":"38ea7253-9af4-11e7-bc1a-080027fb95be"}...
```



Liveness and Readiness Probe

- TCPSocketAction:

Events:								
FirstSeen	LastSeen	Count	From	SubObjectPath	Type	Reason	Message	
4m	4m	1	default-scheduler		Normal	Scheduled	Successful	
y assigned webtest-1810247028-x5xdf to minikube								
4m	4m	1	kubelet, minikube		Normal	SuccessfulMountVolume	MountVolume	
.SetUp succeeded for volume "default-token-pcw4h"								
4m	38s	6	kubelet, minikube	spec.containers{webtest}	Normal	Pulled	Container i	
mage "labdocker/cluster:webservicelite_v1" already present on machine								
4m	38s	6	kubelet, minikube	spec.containers{webtest}	Normal	Created	Created con	
tainer								
4m	38s	6	kubelet, minikube	spec.containers{webtest}	Normal	Started	Started con	
tainer								
3m	38s	5	kubelet, minikube	spec.containers{webtest}	Normal	Killing	Killing con	
tainer with id docker://webtest:pod "webtest-1810247028-x5xdf_default(38f840f4-9af4-11e7-bc1a-080027fb95be)" container "webtest" is unhealthy, it w								
ill be killed and re-created.								
4m	9s	12	kubelet, minikube	spec.containers{webtest}	Warning	Unhealthy	Liveness pr	
obe failed: dial tcp 172.17.0.6:3000: getsockopt: connection refused								
4m	4s	37	kubelet, minikube	spec.containers{webtest}	Warning	Unhealthy	Readiness p	
robe failed: dial tcp 172.17.0.6:3000: getsockopt: connection refused								
www www MacBook Pro% █								

Liveness and Readiness Probe

- HTTPGetAction:



Liveness and Readiness Probe

- HTTPGetAction:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12    spec:
13      replicas: 1
14      selector:
15        matchLabels:
16          name: web
17          owner: Praparn_L
18          version: "1.0"
19          module: WebServer
20          environment: development
21      template:
22        metadata:
23          labels:
24            name: web
25            owner: Praparn_L
26            version: "1.0"
27            module: WebServer
28            environment: development
```

```
28
29   spec:
30     containers:
31       - name: webtest
32         image: labdocker/cluster:webservicelite_v1
33         ports:
34           - name: webservice
35             containerPort: 5000
36             protocol: TCP
37             readinessProbe:
38               httpGet:
39                 # Optional "host: my-host" for set specific ip of Pods
40                 # Optional "scheme: HTTPS"
41                 #schema: HTTPS
42                 path: /
43                 port: webservice
44                 initialDelaySeconds: 15
45                 periodSeconds: 5
46                 timeoutSeconds: 10
47                 successThreshold: 1
48                 failureThreshold: 3
49             livenessProbe:
50               httpGet:
51                 # Optional "host: my-host" for set specific ip of Pods
52                 # Optional "scheme: HTTPS"
53                 #schema: HTTPS
54                 path: /
55                 port: webservice
56                 initialDelaySeconds: 15
57                 periodSeconds: 10
58                 timeoutSeconds: 15
59                 successThreshold: 1
59                 failureThreshold: 3
```



Liveness and Readiness Probe

- HTTPGetAction:

```
praparns-MacBook-Pro% kubectl create -f webtest_deploy_liveness_readiness_http.yml
  kubectl get deployment/webtest
  kubectl get pods --show-labels
deployment "webtest" created
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webtest   1          0          0           0           0s
NAME              READY   STATUS        RESTARTS   AGE   LABELS
webtest-1098172621-zvd22  0/1   ContainerCreating   0          0s   environment=development,module=WebServer,name=web,owner=Praparn_L,pod
-ttemplate-hash=1098172621,version=1.0

```

Events:								
FirstSeen	LastSeen	Count	From	SubObjectPath	Type	Reason	Message	
9m	9m	1	default-scheduler		Normal	Scheduled	Successful	
y	assigned webtest-1098172621-zvd22 to minikube							
9m	9m	1	kubelet, minikube		Normal	SuccessfulMountVolume	MountVolume	
.SetUp succeeded for volume "default-token-pcw4h"								
9m	9m	1	kubelet, minikube	spec.containers{webtest}	Normal	Pulled	Container i	
mage	labdocker/cluster:webserviceelite_v1" already present on machine							
9m	9m	1	kubelet, minikube	spec.containers{webtest}	Normal	Created	Created con	
9m	9m	1	kubelet, minikube	spec.containers{webtest}	Normal	Started	Started con	
tainer								
tainer								



Liveness and Readiness Probe

- HTTPGetAction:

```
29      livenessProbe:  
30        httpGet:  
31          # Optional "host: my-host" for set specific ip of Pods  
32          # Optional "scheme: HTTPS"  
33          #schema: HTTPS  
34          path: /init  
35          port: webservice  
36          httpHeaders:  
37            - name: server  
38            value: "Werkzeug/0.12.2 Python/2.7.13"  
39          initialDelaySeconds: 15  
40          periodSeconds: 5  
41          timeoutSeconds: 5  
42          successThreshold: 1  
43          failureThreshold: 3  
44      livenessProbe:  
45        httpGet:  
46          # Optional "host: my-host" for set specific ip of Pods  
47          # Optional "scheme: HTTPS"  
48          #schema: HTTPS  
49          path: /init  
50          port: webservice  
51          httpHeaders:  
52            - name: server  
53            value: "Werkzeug/0.12.2 Python/2.7.13"  
54          initialDelaySeconds: 15  
55          periodSeconds: 5  
56          timeoutSeconds: 5  
57          successThreshold: 1  
58          failureThreshold: 3
```

```
praparns-MacBook-Pro% kubectl apply -f webtest_deploy_liveness_readiness_http.yml  
deployment "webtest" configured  
praparns-MacBook-Pro% kubectl get deployment/webtest  
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE  
webtest   1          1          1           0           14m  
praparns-MacBook-Pro% kubectl get deployment/webtest  
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE  
webtest   1          1          1           0           15m  
praparns-MacBook-Pro% kubectl get pods  
NAME                  READY   STATUS    RESTARTS   AGE  
webtest-692878837-fxpw4   0/1     Running   3          1m
```

Liveness and Readiness Probe

- HTTPGetAction:

Events:								
FirstSeen	LastSeen	Count	From	SubObjectPath	Type	Reason	Message	
1m	1m	1	default-scheduler		Normal	Scheduled	Successful	
y assigned webtest-692878837-fxpw4 to minikube								
1m	1m	1	kubelet, minikube		Normal	SuccessfulMountVolume	MountVolume	
.SetUp succeeded for volume "default-token-pcw4h"								
1m	13s	5	kubelet, minikube	spec.containers{webtest}	Normal	Pulled	Container i	
mage "labdocker/cluster:webservicelite_v1" already present on machine								
1m	13s	9	kubelet, minikube	spec.containers{webtest}	Warning	Unhealthy	Readiness p	
robe failed: HTTP probe failed with statuscode: 404								
1m	13s	9	kubelet, minikube	spec.containers{webtest}	Warning	Unhealthy	Liveness pr	
obe failed: HTTP probe failed with statuscode: 404								
1m	13s	4	kubelet, minikube	spec.containers{webtest}	Normal	Killing	Killing con	
tainer with id docker://webtest:pod "webtest-692878837-fxpw4_default(4f6f205f-9b00-11e7-bc1a-080027fb95be)" container "webtest" is unhealthy, it wi								
ll be killed and re-created.								
1m	12s	5	kubelet, minikube	spec.containers{webtest}	Normal	Created	Created con	
tainer								
1m	12s	5	kubelet, minikube	spec.containers{webtest}	Normal	Started	Started con	
tainer								
praparns-MacBook-Pro%								

Liveness and Readiness Probe

- ReadinessGate:

```
Users > praparnlueangphoonlap > Desktop > ! 1.yaml
1   kind: Pod
2   ...
3   spec:
4     readinessGates:
5       - conditionType: "www.example.com/feature-1"
6   status:
7     conditions:
8       - type: Ready                      # a built in PodCondition
9         status: "False"
10        lastProbeTime: null
11        lastTransitionTime: 2018-01-01T00:00:00Z
12       - type: "www.example.com/feature-1"      # an extra PodCondition
13         status: "False"
14         lastProbeTime: null
15         lastTransitionTime: 2018-01-01T00:00:00Z
16     containerStatuses:
17       - containerID: docker://abcd...
18         ready: true
19   ...
```

Workshop: Liveness and Readiness

The screenshot shows a web browser window with the URL `http://192.168.99.100:32500`. The page title is "Welcome Page from Container Python Lab Web Version 1.00". Below the title, it says "Checkpoint Date/Time: Sat Sep 16 15:54:10 2017". To the right of the browser, a Postman application window is open, showing a list of API requests and their details.

Postman Application Details:

- Builder Tab:** Shows a collection named "http://192.168.99.100" with one GET request to `http://192.168.99.100:32500`.
- Authorization Tab:** Set to "No Auth".
- Headers Tab:** Contains four headers:
 - `content-length` → 113
 - `content-type` → `text/html; charset=utf-8`
 - `date` → `Sat, 16 Sep 2017 15:56:21 GMT`
 - `server` → `Werkzeug/0.12.2 Python/2.7.13`
- Code Tab:** Shows status `200 OK` and time `46 ms`.

Postman History:

- Today:** One GET request to `http://192.168.99.100:32500`.
- September 8:** Four requests:
 - GET to `http://192.168.99.100/users/removeUser/99`
 - GET to `http://192.168.99.100/users/1`
 - POST to `http://192.168.99.100/users/insertUser`
 - GET to `http://192.168.99.100/init`
- August 27:** One GET request to `http://192.168.99.100/`.

Kubernetes: Production Workload Orchestration

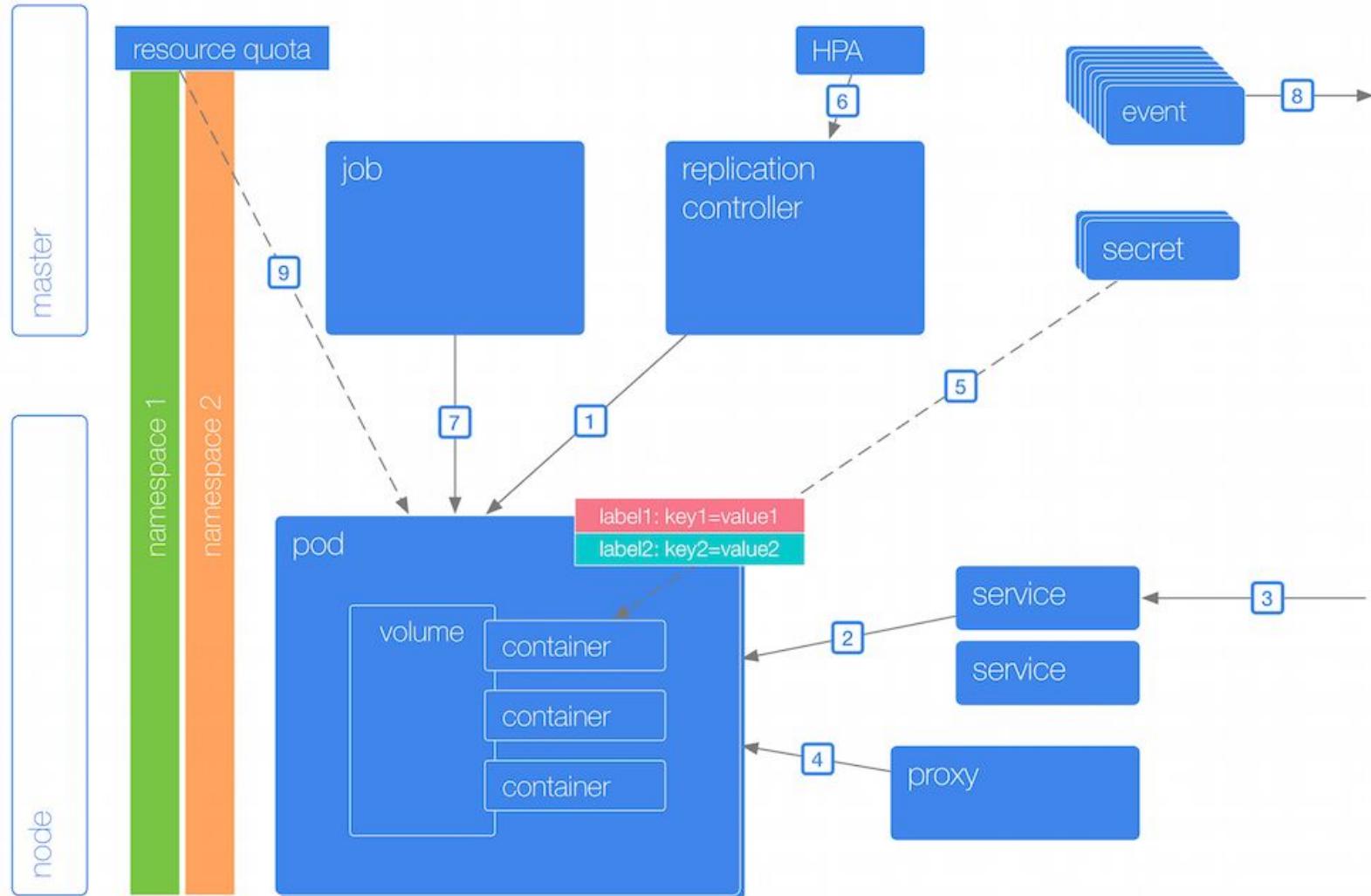


kubernetes
by Google

Resource Management and HPA



Resource Management and HPA



Ref: <http://k8s.info/cs.html>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Resource Management and HPA

- Kubernetes can manage resource in 2 ways
 - Container level configuration
 - Scope on Container unit independence
 - etc
 - Namespace level configuration
 - Scope on Pods and Container level
 - Create global limit and assign to namespace
 - Apply namespace to any Pods (Template Resource Limit)
- What happen when Container resource exceed ?
 - CPU
 - Container will not get CPU as it need and the free-slot of CPU was distributed to all container's request with ratio
 - Memory
 - Container was killed and restart it again



Resource Management and HPA

- Container level configuration
 - CPU
 - Request:
 - XXXm (Unit: millicores) (Ratio: 1024)
 - 0.1 = 100m
 - 1 CPU =
 - 1 vCPU (AWS)
 - 1 GCP (Google Cloud)
 - 1 Azure v Core
 - 1 Hyper thread (Bare Metal)
 - Consider as “--cpu-share”
 - Limit:
 - XXXm (Unit: millicores) (Ratio: 100000/1000: ~1000)
 - Consider as “--cpu-quota”
 - Memory
 - Request:
 - XXX (Unit: Ki, Mi, Gi, Pi, Ei)
 - Limit
 - XXX (Unit: Ki, Mi, Gi, Pi, Ei)

Ref: <https://github.com/kubernetes/community/blob/master/contributors/design-proposals/resource-qos.md>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Resource Management and HPA

- Container level configuration

- Pods “webtest”

```
1 apiVersion: "v1"
2 kind: Pod
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9   module: WebServer
10  environment: development
11 spec:
12   containers:
13     - name: webtest
14       image: labdocker/cluster:webservicelite_v1
15     resources:
16       requests:
17         memory: "16Mi"
18         cpu: "100m"
19       limits:
20         memory: "32Mi"
21         cpu: "1"
22     ports:
23       - containerPort: 5000
24         protocol: TCP
```

System status

...ckage/kubernetes — Terminal MAC Pro — zsh		docker@iZj6chay173lo9ivaj1drnZ: ~ — -bash		Terminal MAC Pro — ssh + minikube ssh			
top - 07:39:14 up 4:10, 1 user, load average: 0.06, 0.29, 0.34		Tasks: 233 total, 1 running, 232 sleeping, 0 stopped, 0 zombie					
%Cpu0 : 0.0/0.0 0[%Cpu1 : 0.0/0.0 0[]			
%Cpu2 : 0.7/0.7 1[%Cpu3 : 0.0/0.0 0[]			
%Cpu4 : 0.0/0.0 0[%Cpu5 : 0.0/0.0 0[]			
%Cpu6 : 0.7/0.7 1[%Cpu7 : 0.0/1.3 1[]			
%Cpu8 : 0.0/0.0 0[%Cpu9 : 0.7/0.7 1[]			
GiB Mem : 30.2/1.953 [███████████]		GiB Swap: 0.0/0.977 [
PID USER PR NI VIRT RES %CPU %MEM TIME+ S COMMAND							
1 root	20	0	37.8m	5.9m	0.0 0.3	0:14.97 S	systemd
1440 root	20	0	65.9m	20.9m	0.0 1.0	0:06.73 S	- systemd-journal
2311 root	20	0	25.6m	3.2m	0.0 0.2	0:01.81 S	- systemd-udevd
2608 root	20	0	8.9m	0.4m	0.0 0.0	0:00.46 S	- getty
2617 root	20	0	24.2m	2.0m	0.0 0.1	0:00.02 S	- rpcbind
2638 root	20	0	25.5m	2.8m	0.0 0.1	0:00.17 S	- systemd-logind



Resource Management and HPA

- Container level configuration
 - Allocate status on node

Namespace	Name	CPU Requests	CPU Limits	Memory Requests	Memory Limits
default	cadvisor-76c9899d7f-pbz29	0 (0%)	0 (0%)	0 (0%)	0 (0%)
default	webtest	100m (1%)	1 (10%)	16Mi (0%)	32Mi (1%)
kube-system	default-http-backend-xq22v	10m (0%)	10m (0%)	20Mi (1%)	20Mi (1%)
kube-system	heapster-4nfdm	0 (0%)	0 (0%)	0 (0%)	0 (0%)
kube-system	influxdb-grafana-kzmsf	0 (0%)	0 (0%)	0 (0%)	0 (0%)
kube-system	kube-addon-manager-minikube	5m (0%)	0 (0%)	50Mi (2%)	0 (0%)
kube-system	kube-dns-54cccfbd8-tqh51	260m (2%)	0 (0%)	110Mi (5%)	170Mi (8%)
kube-system	kubernetes-dashboard-77d8b98585-z4lxz	0 (0%)	0 (0%)	0 (0%)	0 (0%)
kube-system	nginx-ingress-controller-hjwt6	0 (0%)	0 (0%)	0 (0%)	0 (0%)
kube-system	storage-provisioner	0 (0%)	0 (0%)	0 (0%)	0 (0%)



Resource Management and HPA

- Container level configuration
 - Create Load on CPU (T1)

```
[praparns-MacBook-Pro% kubectl create -f webtest_pod.yml
pod "webtest" created
[praparns-MacBook-Pro% kubectl get pods
  NAME      READY     STATUS    RESTARTS   AGE
  webtest   1/1      Running   0          13m
[praparns-MacBook-Pro% kubectl exec webtest -c webtest md5sum /dev/urandom
^C
praparns-MacBook-Pro%
```

- System status

```
top - 07:45:00 up 4:16, 1 user, load average: 0.69, 0.35, 0.33
Tasks: 234 total, 2 running, 232 sleeping, 0 stopped, 0 zombie
%Cpu0 : 0.0/0.0  0[                ]
%Cpu1 : 0.0/1.3  1[|               ]
%Cpu2 : 0.0/0.0  0[                ]
%Cpu3 : 26.2/73.8 100[███████████]
%Cpu4 : 0.7/0.0  1[                ]
%Cpu5 : 0.7/0.0  1[                ]
%Cpu6 : 0.7/0.0  1[                ]
%Cpu7 : 0.7/0.7  1[                ]
%Cpu8 : 0.7/0.0  1[                ]
%Cpu9 : 0.7/0.7  1[                ]
GiB Mem : 31.2/1.953 [███████████]
GiB Swap: 0.0/0.977 [                ]
```

Resource Management and HPA

- Container level configuration
 - Create Load on CPU (T2)

```
praparns-MacBook-Pro% kubectl create -f webtest_pod.yml
pod "webtest" created
praparns-MacBook-Pro% kubectl get pods
NAME      READY     STATUS    RESTARTS   AGE
webtest   1/1      Running   0          13m
praparns-MacBook-Pro% kubectl exec webtest -c webtest md5sum /dev/urandom
^C
praparns-MacBook-Pro% kubectl exec webtest -c webtest md5sum /dev/urandom
^C
praparns-MacBook-Pro%
```

- ### ○ System status

Workshop: Resource MNG/HPA

- Part 1: Container level configuration

```
top - 07:45:00 up 4:16, 1 user, load average: 0.69, 0.35, 0.33
Tasks: 234 total, 2 running, 232 sleeping, 0 stopped, 0 zombie
%CPU0 : 0.0/0.0   0[ ]
%CPU1 : 0.0/1.3   1[ ]
%CPU2 : 0.0/0.0   0[ ]
%CPU3 : 26.2/73.8 100[███████████] 26.2%
%CPU4 : 0.7/0.0   1[ ]
%CPU5 : 0.7/0.0   1[ ]
%CPU6 : 0.7/0.0   1[ ]
%CPU7 : 0.7/0.7   1[ ]
%CPU8 : 0.7/0.0   1[ ]
%CPU9 : 0.7/0.7   1[ ]
GiB Mem : 31.2/1.953 [███████████] 1.6%
GiB Swap: 0.0/0.977 [ ] 0%
```



Resource Management and HPA

- Namespace level configuration
 - Name space provide collection of resource of cluster system that easy to defined and apply to object in cluster system
 - Name space can apply with
 - Pods
 - Services
 - Replication Controller
 - Deployment and ReplicaSets
 - Quota
 - LimitRange
 - Etc
 - For resource management resource in name space level. We will use Quota and LimitRange
 - Quota: Summary of resource control on name space
 - LimitRange: Resource admission control/Default resource define



Resource Management and HPA

- Quota
 - Compute resource
 - CPU
 - Limits
 - Request
 - Memory
 - Limits
 - Request
 - Disk I/O
 - PersistentVolumeClaims
 - Request.Storage
 - Counting Object
 - Pods
 - Service
 - Replication Controller
 - ConfigMap
 - Secrets

```
1  apiVersion: v1
2  kind: ResourceQuota
3  metadata:
4    name: webtest-quota
5    labels:
6      name: webtest_quota
7      owner: Praparn_L
8      version: "1.0"
9      module: Quota
10     environment: development
11   spec:
12     hard:
13       pods: "4"
14       requests.cpu: "1"
15       requests.memory: 1Gi
16       limits.cpu: "4"
17       limits.memory: 4Gi
```

Ref: <https://kubernetes.io/docs/concepts/policy/resource-quotas>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Resource Management and HPA

- LimitRange
 - Type: (Pods/Container)
 - Max
 - CPU:
 - Memory:
 - Min
 - CPU
 - Memory
 - Default (Limit Default for Container)
 - CPU
 - Memory
 - DefaultRequest (Default for Container)
 - CPU
 - Memory

```
1  apiVersion: v1
2  kind: LimitRange
3  metadata:
4    name: webtest_limit
5    labels:
6      name: webtest_limit
7      owner: Praparn_L
8      version: "1.0"
9    module: LimitRange
10   environment: development
11  spec:
12    limits:
13      - max:
14        cpu: "1"
15        memory: 1Gi
16      min:
17        cpu: 200m
18        memory: 6Mi
19      type: Pod
20    - default:
21      cpu: 300m
22      memory: 200Mi
23    defaultRequest:
24      cpu: 200m
25      memory: 100Mi
26    max:
27      cpu: "1"
28      memory: 1Gi
29    min:
30      cpu: 100m
31      memory: 3Mi
32    type: Container
```

Ref: <https://kubernetes.io/docs/tasks/administer-cluster/apply-resource-quota-limit/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Resource Management and HPA

- Create Namespace, assign quota with name space

```
kubectl create namespace <name>
```

```
kubectl create -f <Quota File> -- namespace <name>
```

```
praparns-MacBook-Pro% kubectl create namespace webtest-namespace
namespace "webtest-namespace" created
praparns-MacBook-Pro% kubectl create -f webtest_quota.yml --namespace=webtest-namespace
resourcequota "webtest-quota" created
praparns-MacBook-Pro% kubectl describe namespace webtest-namespace
Name:           webtest-namespace
Labels:         <none>
Annotations:    <none>
Status:         Active
                □

Resource Quotas
Name:           webtest-quota
Resource        Used   Hard
-----
limits.cpu      0      4
limits.memory   0      4Gi
pods            0      4
requests.cpu    0      1
requests.memory 0      1Gi

No resource limits.
praparns-MacBook-Pro%
```

Ref: <https://kubernetes.io/docs/concepts/policy/resource-quotas>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Resource Management and HPA

- Deployment

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Paparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12   spec:
13     selector:
14       name: web
15       owner: Paparn_L
16       version: "1.0"
17       module: WebServer
18       environment: development
19
20     type: NodePort
21     ports:
22       - port: 5000
23         name: http
24         targetPort: 5000
25         protocol: TCP
26         nodePort: 32500
27
28   apiVersion: apps/v1
29   kind: Deployment
30   metadata:
31     name: webtest
32     labels:
33       name: web
34       owner: Paparn_L
35       version: "1.0"
```

```
35     module: WebServer
36     environment: development
37
38   spec:
39     replicas: 1
40     template:
41       metadata:
42         labels:
43           name: web
44           owner: Paparn_L
45           version: "1.0"
46           module: WebServer
47           environment: development
48
49     spec:
50       containers:
51         - name: webtest
52           image: labdocker/cluster:webservicelite_v1
53           ports:
54             - containerPort: 5000
55               protocol: TCP
```



Resource Management and HPA

- Create Deployment and check result

```
kubectl create -f <Deployment File> --namespace <name>
```

```
praparns-MacBook-Pro% kubectl create -f webtest_deploy.yml --namespace=webtest-namespace
service "webtest" created
deployment "webtest" created
[praparns-MacBook-Pro% kubectl get deployment/webtest --namespace=webtest-namespace
NAME      DESIRED  CURRENT  UP-TO-DATE  AVAILABLE   AGE
webtest   1         0         0          0           5m
[praparns-MacBook-Pro% kubectl get rs --namespace=webtest-namespace
NAME      DESIRED  CURRENT  READY     AGE
webtest-4261491039  1         0         0           5m
[praparns-MacBook-Pro% kubectl get svc/webtest --namespace=webtest-namespace
NAME      CLUSTER-IP  EXTERNAL-IP  PORT(S)        AGE
webtest  10.0.0.48    <nodes>      5000:32500/TCP  5m
[praparns-MacBook-Pro% kubectl get pods --namespace=webtest-namespace
No resources found.
praparns-MacBook-Pro%
```



Resource Management and HPA

- Why it fail ?

```
kubectl describe rs --namespace <name>
```

```
Events:
FirstSeen      LastSeen      Count   From            SubObjectPath      Type    Reason          Message
-----      -----      ----   ----            -----      ----    ----          -----
6m           5m           15   replicaset-controller      Warning   FailedCreate  Error creating: pods "webtest-4261491039-" is
forbidden: failed quota: webtest-quota: must specify limits.cpu,limits.memory,requests.cpu,requests.memory
```

```
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: web
        owner: Praparn_L
        version: "1.0"
        module: WebServer
        environment: development
    spec:
      containers:
        - name: webtest
          image: labdocker/cluster:webservicelite_v1
          ports:
            - containerPort: 5000
              protocol: TCP
```



Resource Management and HPA

- Create LimitRange and check result

```
kubectl create -f <LimitRange File> --namespace <name>
```

```
praparns-MacBook-Pro% kubectl create -f webtest_limit.yml --namespace=webtest-namespace
limitrange "webtest-limit" created
praparns-MacBook-Pro% kubectl describe namespace/webtest-namespace
Name:           webtest-namespace
Labels:         <none>
Annotations:    <none>
Status:         Active

Resource Quotas
  Name:          webtest-quota
  Resource      Used   Hard
  ----          ---    ---
  limits.cpu    0       4
  limits.memory 0       4Gi
  pods          0       4
  requests.cpu  0       1
  requests.memory 0       1Gi

Resource Limits
  Type        Resource  Min   Max     Default Request Default Limit  Max Limit/Request Ratio
  ----        -----  ---   ---     -----      -----      -----      -----
  Pod         cpu      200m  1       -          -          -
  Pod         memory  6Mi   1Gi    -          -          -
  Container   cpu      100m  1       200m      300m      -
  Container   memory  3Mi   1Gi    100Mi    200Mi     -
```



Resource Management and HPA

- Recreate Deployment and check result

```
praparns-MacBook-Pro% kubectl delete -f webtest_deploy.yml --namespace=webtest-namespace
service "webtest" deleted
deployment "webtest" deleted
praparns-MacBook-Pro% kubectl create -f webtest_deploy.yml --namespace=webtest-namespace
service "webtest" created
deployment "webtest" created
praparns-MacBook-Pro% kubectl get deployment/webtest --namespace=webtest-namespace
NAME      DESIRED  CURRENT  UP-TO-DATE  AVAILABLE   AGE
webtest   1         1         1           1           5m
praparns-MacBook-Pro% kubectl get rs --namespace=webtest-namespace
NAME      DESIRED  CURRENT  READY     AGE
webtest-4261491039  1         1         1           5m
praparns-MacBook-Pro% kubectl get svc/webtest --namespace=webtest-namespace
NAME      CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
webtest  10.0.0.125  <nodes>        5000:32500/TCP  5m
praparns-MacBook-Pro% kubectl get pods --namespace=webtest-namespace
NAME          READY  STATUS    RESTARTS  AGE
webtest-4261491039-pft5m  1/1    Running   0          5m
praparns-MacBook-Pro% curl http://192.168.99.100:32500
<H1> Welcome Page from Container Python Lab Web Version 1.00 </H1>Checkpoint Date/Time: Sat Jul  8 13:44:12 2017
praparns-MacBook-Pro%
```

Resource Management and HPA

- Check describe of Pods

```
kubectl describe pods --namespace <name>
```

```
Status:          Running
IP:             172.17.0.2
Controllers:    ReplicaSet/webtest-4261491039
Containers:
  webtest:
    Container ID:      docker://0acf26d00b7b9f28b200fdf6
    Image:            labdocker/cluster:webservicelite_
    Image ID:          docker-pullable://labdocker/clust
    Port:             5000/TCP
    State:            Running
    Started:          Sat, 08 Jul 2017 20:43:27 +0700
    Ready:            True
    Restart Count:    0
    Limits:
      cpu:           300m
      memory:        200Mi
    Requests:
      cpu:           200m
      memory:        100Mi
    Environment:     <none>
    Mounts:
```



Resource Management and HPA

- Test burn cpu utilize and monitor

```
[praparns-MacBook-Pro: ~] kubectl exec -it webtest-4261491039-pft5m -c webtest md5sum /dev/urandom --namespace=webtest-namespace
```

```
top - 13:57:03 up 7:33, 1 user,  load average: 0.22, 0.35, 0.20
Tasks: 228 total,  2 running, 226 sleeping,  0 stopped,  0 zombie
%Cpu0 : 0.0/0.0   0[ ]
%Cpu1 : 0.7/0.7   1[|||]
%Cpu2 : 0.0/0.0   0[ ]
%Cpu3 : 0.0/0.0   0[ ]
%Cpu4 : 0.0/0.0   0[ ]
%Cpu5 : 0.0/0.7   1[|]
%Cpu6 : 0.0/0.0   0[ ]
%Cpu7 : 0.7/0.7   1[||]
%Cpu8 : 7.9/21.9  30[||||||||||||||||||||||||||||||||]
%Cpu9 : 0.0/0.0   0[ ]
GiB Mem : 32.4/1.953 [███████████] 100%
GiB Swap: 0.0/0.977 [ ]
```

Resource Management and HPA

- Edit deployment for update cpu/memory Limits,request

```
[praparn-MacBook-Pro% kubectl set resources deployment/webtest --limits=cpu="1",memory=1Gi --requests=cpu="0.8",memory=800Mi --record
--namespace=webtest-namespace
deployment "webtest" resource requirements updated

[praparn-MacBook-Pro% kubectl describe pods webtest-d6986bb64-bm5q9 --namespace=webtest-namespace
Name:           webtest-d6986bb64-bm5q9
Namespace:      webtest-namespace
Node:          minikube/192.168.99.100
Start Time:    Wed, 21 Mar 2018 00:46:27 +0700
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                pod-template-hash=825426620
                version=1.0
Annotations:   <none>
Status:        Running
IP:            172.17.0.11
Controlled By: ReplicaSet/webtest-d6986bb64
Containers:
  webtest:
    Container ID:  docker://7e1ae95ed7db8fb9c91b9bcf89b97aedef9e8592d129852930835e744a12b6b6
    Image:         labdocker/cluster:webserviceelite_v1
    Image ID:     docker-pullable://labdocker/cluster@sha256:f0f261307a9a0ce8acf317f533e22c3aa438b3a7c5d4c0b5705ce67504326940
    Port:          5000/TCP
    State:        Running
      Started:    Wed, 21 Mar 2018 00:46:28 +0700
    Ready:        True
    Restart Count: 0
    Limits:
      cpu: 1
      memory: 1Gi
    Requests:
      cpu: 800m
      memory: 800Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-n8h4m (ro)
```



Workshop: Resource MNG/HPA

- Part 2: Namespace level configuration



192.168.99.100:32500

Apps NMac Ked - Mac O... Medium jenkins vagrant Mesos Vue docker NGINX Taiwan Kuberr

Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Sat Jul 8 14:10:26 2017

System Info:	Value				
Machine ID:	88a2186a47ff442aa72465d9e09fc21f				
System UUID:	F631F911-A47C-493D-BFCB-872F1F5C2D30				
Boot ID:	99b9bb5a-67ba-43d0-bed1-e64fe04c367d				
Kernel Version:	4.9.13				
OS Image:	Buildroot 2017.02				
Operating System:	linux				
Architecture:	amd64				
Container Runtime Version:	docker://1.12.6				
Kubelet Version:	v1.7.0-alpha.2				
Kube-Proxy Version:	v1.7.0-alpha.2				
ExternalID:	minikube				
Non-terminated Pods:	(4 in total)				
Namespace	Name	CPU Requests	CPU Limits	Memory Requests	Memory Limits
kube-system	kube-addon-manager-minikube	5m (0%)	0 (0%)	50Mi (2%)	0 (0%)
kube-system	kube-dns-268032401-9h5s6	260m (2%)	0 (0%)	110Mi (5%)	170Mi (8%)
kube-system	kubernetes-dashboard-2vk98	0 (0%)	0 (0%)	0 (0%)	0 (0%)
webtest-namespace	webtest-4261491039-pft5m	200m (2%)	300m (3%)	100Mi (5%)	200Mi (10%)
Allocated resources:					
(Total limits may be over 100 percent, i.e., overcommitted.)					
CPU Requests	CPU Limits	Memory Requests	Memory Limits		
465m (4%)	300m (3%)	260Mi (13%)	370Mi (19%)		



Resource Management and HPA

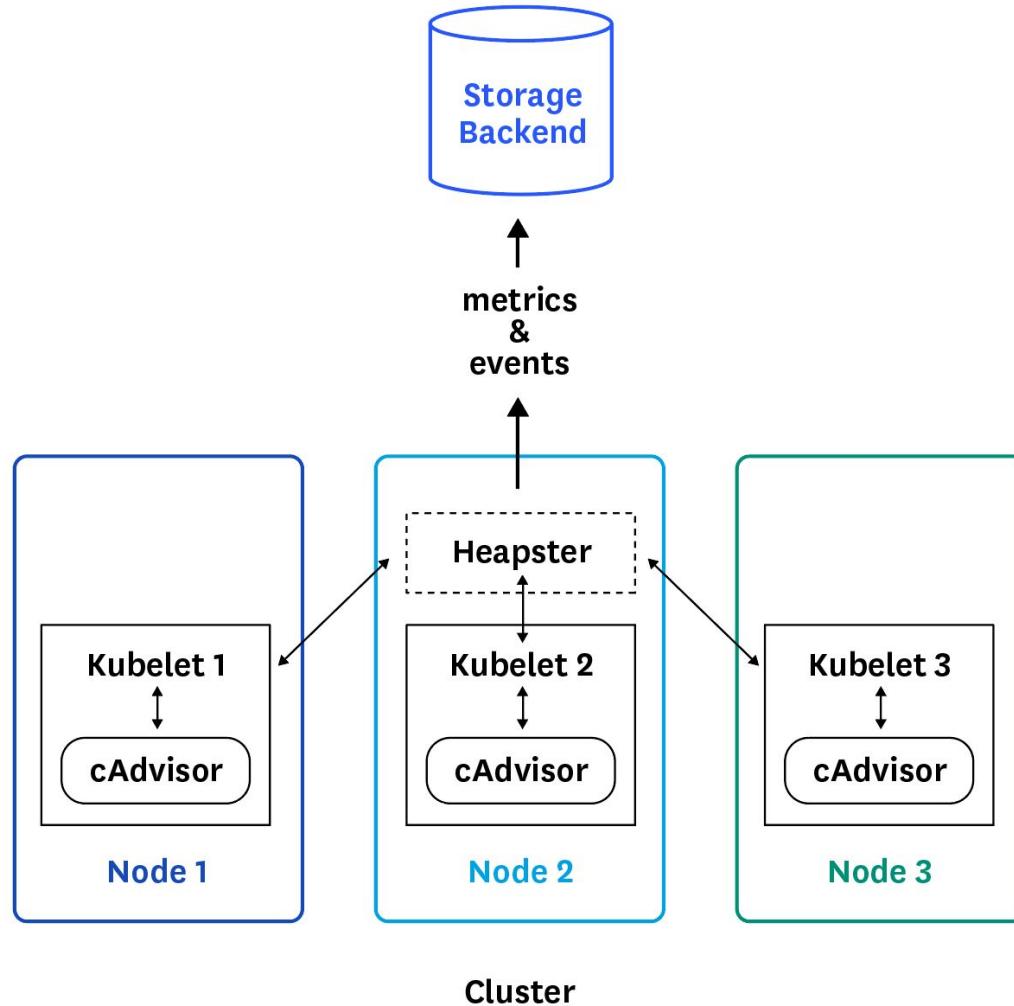
- Horizontal Pod Autoscaling (HPA)
 - The scale is easy to operate with "kubectl scale --replicas=XXX deployment/<name>" with single command
 - But..
 - How can you scale meet up/down actual required ?
 - HPA will response for monitor workload on Pods (Now base on CPU) and automatic trigger deployment to scale-up application

```
kubectl autoscale <type/name> <option:>
```

- Ex: kubectl autoscale deployment/webtest --min=1--max=10
--cpu-percent=80



Resource Management and HPA

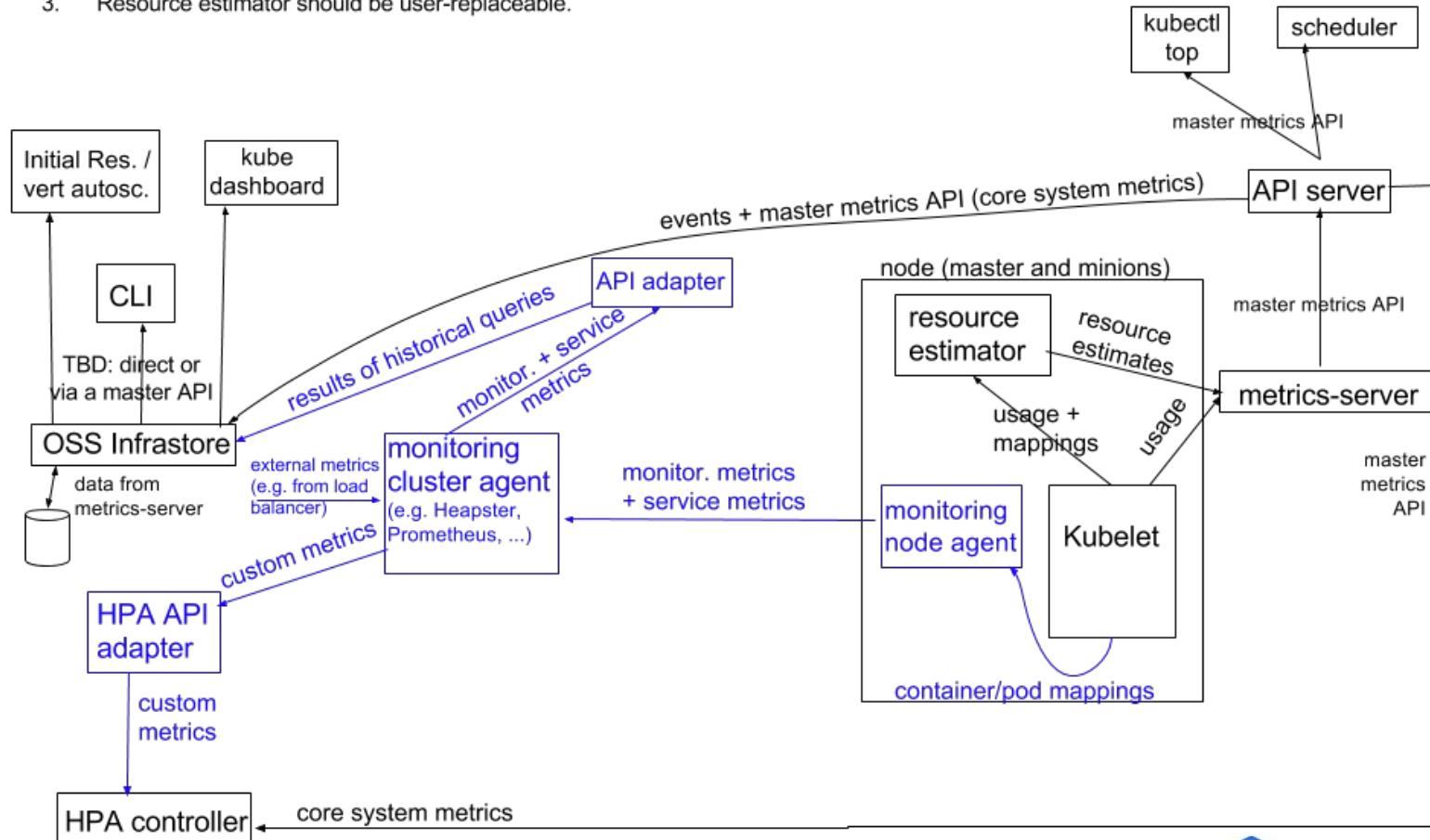


Resource Management and HPA

Monitoring architecture proposal: OSS
(arrows show direction of metrics flow)

Notes

1. Arrows show direction of metrics flow.
2. **Monitoring pipeline is in blue**. It is user-supplied and optional.
3. Resource estimator should be user-replaceable.



Resource Management and HPA

kubernetes / kubernetes

Watch 2,788 Star 43,351 Fork 15,077

Code Issues 2,196 Pull requests 942 Projects 12 Insights

Branch: master kubernetes / cluster / addons / metrics-server / Create new file Upload files Find file History

DirectXMan12 Bump metrics-server to v0.3.1 ... Latest commit 13d59fd on Sep 17

..

File	Description	Time Ago
OWNERS	Add kawych to Metrics Server owners	10 months ago
README.md	Add Troubleshooting sections to Heapster and Metrics Server addons do...	8 months ago
auth-delegator.yaml	Made metrics-server critical service managed by addon-manager	a year ago
auth-reader.yaml	Made metrics-server critical service managed by addon-manager	a year ago
metrics-apiservice.yaml	Bumped Metrics Server to v0.2.0	a year ago
metrics-server-deployment.yaml	Bump metrics-server to v0.3.1	2 months ago
metrics-server-service.yaml	Made metrics-server critical service managed by addon-manager	a year ago
resource-reader.yaml	Autoscaler metrics-server with pod-nanny	a year ago

README.md

Metrics Server

Metrics Server exposes core Kubernetes metrics via metrics API.

More details can be found in [Core metrics pipeline documentation](#).

Troubleshooting

Metrics Server supports up to 30 pods per cluster node. In clusters where there are more running pods, Metrics Server may be throttled or fail with OOM error. Starting with Kubernetes 1.9.2, Metrics Server resource requirements may be overwritten manually. [Learn more about Addon Resizer configuration](#)

Important notices

Decreasing resource requirements for cluster addons may cause system instability. The effects may include (but are not limited to):

- Horizontal Pod Autoscaler not working
- `kubectl top` not working (starting with Kubernetes 1.10)

Overwritten configuration persists through cluster updates, therefore may cause all effects above after a cluster update.



Resource Management and HPA

Algorithm Details

From the most basic perspective, the Horizontal Pod Autoscaler controller operates on the ratio between desired metric value and current metric value:

```
desiredReplicas = ceil[currentReplicas * ( currentMetricValue / desiredMetricValue )]
```

For example, if the current metric value is `200m`, and the desired value is `100m`, the number of replicas will be doubled, since `200.0 / 100.0 == 2.0`. If the current value is instead `50m`, we'll halve the number of replicas, since `50.0 / 100.0 == 0.5`. We'll skip scaling if the ratio is sufficiently close to 1.0 (within a globally-configurable tolerance, from the `--horizontal-pod-autoscaler-tolerance` flag, which defaults to 0.1).

--horizontal-pod-autoscaler-initial-readiness-delay

- Default 30 seconds

--horizontal-pod-autoscaler-cpu-initialization-period

- Default 300 seconds

--horizontal-pod-autoscaler-downscale-stabilization

- Default 300 seconds

Resource Management and HPA

- Example: Deployment for python

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11 spec:
12   selector:
13     name: web
14     owner: Praparn_L
15     version: "1.0"
16     module: WebServer
17     environment: development
18
19   type: NodePort
20   ports:
21     - port: 5000
22       name: http
23       targetPort: 5000
24       protocol: TCP
25       nodePort: 32500
```

```
27  apiVersion: apps/v1
28  kind: Deployment
29  metadata:
30    name: webtest
31    labels:
32      name: web
33      owner: Praparn_L
34      version: "1.0"
35      module: WebServer
36      environment: development
37  spec:
38    replicas: 1
39    selector:
40      matchLabels:
41        name: web
42        owner: Praparn_L
43        version: "1.0"
44        module: WebServer
45        environment: development
46    template:
47      metadata:
48        labels:
49          name: web
50          owner: Praparn_L
51          version: "1.0"
52          module: WebServer
53          environment: development
54  spec:
55    containers:
56      - name: webtest
57        image: labdockers/cluster:webservicelite_v1
58        resources:
59          requests:
60            cpu: "200m"
61        ports:
62          - containerPort: 5000
63            protocol: TCP
```



Resource Management and HPA

- Apply HPA for monitor and scale (TARGET CPU 10%)

```
docker@kubernetes-ms:~$ kubectl autoscale deployment/webtest --min=1 --max=10 --cpu-percent=10
deployment "webtest" autoscaled
docker@kubernetes-ms:~$ kubectl get hpa/webtest
NAME      REFERENCE      TARGETS      MINPODS      MAXPODS      REPLICAS      AGE
webtest   Deployment/webtest   <unknown> / 10%    1            10           0            10s
docker@kubernetes-ms:~$ kubectl get hpa/webtest
NAME      REFERENCE      TARGETS      MINPODS      MAXPODS      REPLICAS      AGE
webtest   Deployment/webtest   2% / 10%    1            10           1            59s
docker@kubernetes-ms:~$ kubectl describe hpa/webtest
Name:                 webtest
Namespace:            default
Labels:               <none>
Annotations:          <none>
CreationTimestamp:    Fri, 02 Feb 2018 10:29:38 -0600
Reference:            Deployment/webtest
Metrics:              resource cpu on pods  (as a percentage of request): 2% (5m) / 10%
Min replicas:         1
Max replicas:         10
Conditions:
  Type        Status  Reason
  ----        ----  -----
  AbleToScale  True    ReadyForNewScale
  ScalingActive  True    ValidMetricFound
  (age of request)
  ScalingLimited False   DesiredWithinRange
Events:               <none>
docker@kubernetes-ms:~$
```



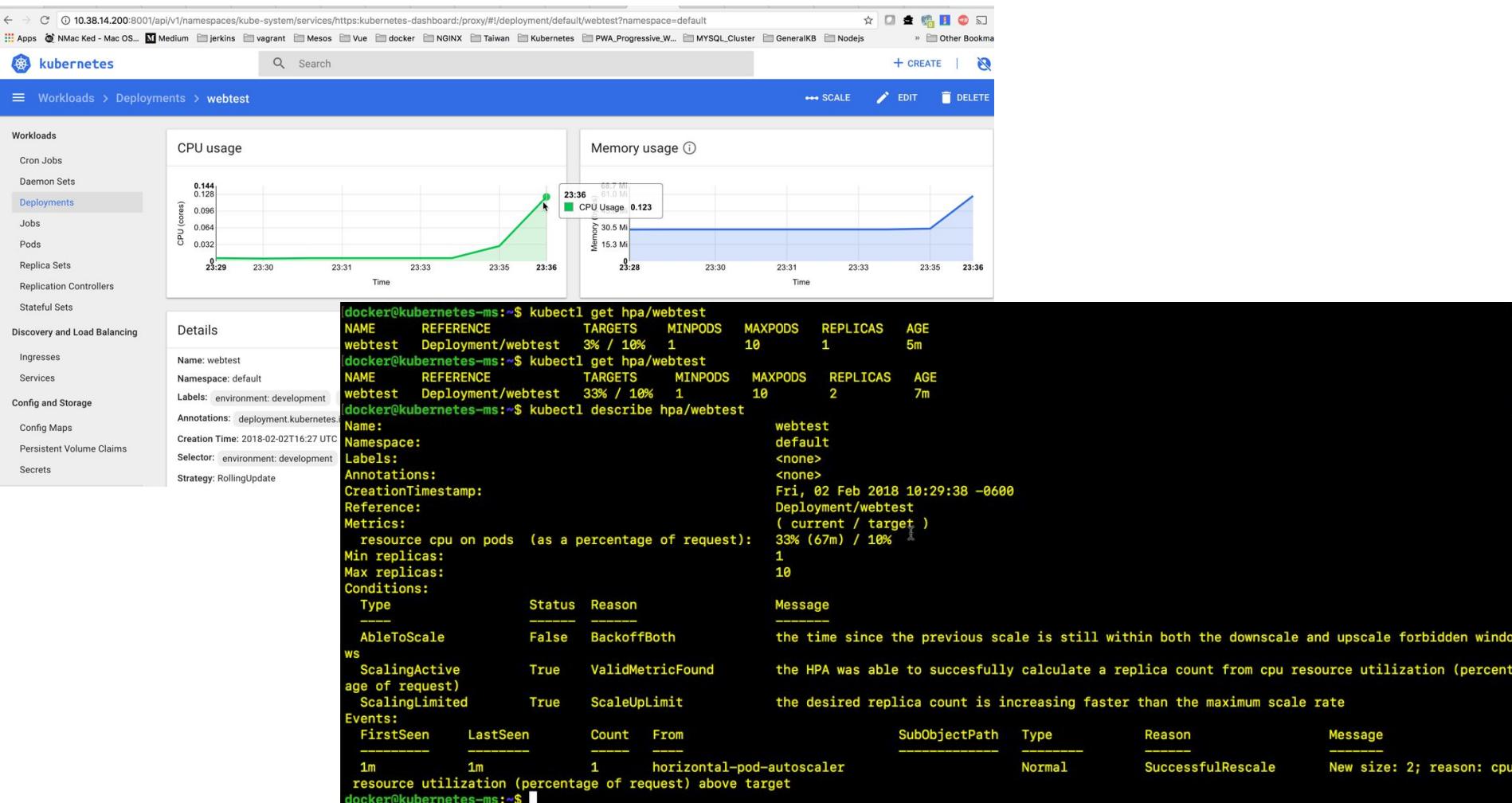
Resource Management and HPA

- Generate load by busybox (wget every 10 ms)

```
[docker@kubernetes-ms:~$ kubectl top nodes
NAME          CPU(cores)   CPU%    MEMORY(bytes)   MEMORY%
kubernetes-ms  283m        14%    1868Mi        48%
kubernetes-1   83m         4%    1434Mi        37%
kubernetes-2   114m        5%    1410Mi        36%
[docker@kubernetes-ms:~$ kubectl top pods
NAME          CPU(cores)   MEMORY(bytes)
webtest-7d89786977-6zktl  29m      29Mi
[docker@kubernetes-ms:~$ kubectl get hpa/webtest
```

Resource Management and HPA

- Load increase and HPA scale-out



Resource Management and HPA

- HPA Scale-Out until meet target (Interval every 5 min)

```
docker@kubernetes-ms:~$ kubectl top pods
NAME                               CPU(cores)   MEMORY(bytes)
webtest-7d89786977-6zktl          69m         29Mi
load-generator-5c4d59d5dd-psqm9   120m        7Mi
webtest-7d89786977-xsp6t          65m         29Mi
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  33% / 10%   1          10          4          10m
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  18% / 10%   1          10          4          11m
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  18% / 10%   1          10          8          14m
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  13% / 10%   1          10          8          15m
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  11% / 10%   1          10          8          16m
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  11% / 10%   1          10          8          17m
docker@kubernetes-ms:~$ kubectl get hpa
NAME      REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
webtest   Deployment/webtest  10% / 10%   1          10          9          19m
```

Resource Management and HPA

- Stop load and HPA Scale-down

Resource Management and HPA

- Stop load and HPA Scale-down

```
docker@kubernetes-ms:~$ kubectl top nodes
NAME          CPU(cores)   CPU%      MEMORY(bytes)  MEMORY%
kubernetes-2  374m        18%       1526Mi        39%
kubernetes-ms  390m        19%       1956Mi        50%
kubernetes-1  184m        9%        1537Mi        39%
docker@kubernetes-ms:~$ kubectl get hpa
NAME          REFERENCE    TARGETS      MINPODS  MAXPODS  REPLICAS  AGE
webtest       Deployment/webtest  5% / 10%   1         10        9          22m
docker@kubernetes-ms:~$ kubectl get hpa
NAME          REFERENCE    TARGETS      MINPODS  MAXPODS  REPLICAS  AGE
webtest       Deployment/webtest  3% / 10%   1         10        3          25m
docker@kubernetes-ms:~$ kubectl get hpa
NAME          REFERENCE    TARGETS      MINPODS  MAXPODS  REPLICAS  AGE
webtest       Deployment/webtest  3% / 10%   1         10        1          30m
docker@kubernetes-ms:~$ kubectl describe hpa/webtest
Name:                  webtest
Namespace:             default
Labels:                <none>
Annotations:           <none>
CreationTimestamp:     Fri, 02 Feb 2018 10:29:38 -0600
Reference:             Deployment/webtest
Metrics:               resource cpu on pods  (as a percentage of request): 3% (6m) / 10%
Min replicas:          1
Max replicas:          10
Conditions:
  Type        Status  Reason
  ----        ----  -----
  AbleToScale False   BackoffBoth
  ScalingActive True    ValidMetricFound
  ScalingLimited False   DesiredWithinRange
Events:
  FirstSeen  LastSeen  Count  From                    SubObjectPath  Type  Reason
  -----  -----  -----  -----                    -----  -----  -----
  24m        24m       1      horizontal-pod-autoscaler  Normal  SuccessfulRescale
  (percentage of request) above target
  21m        21m       1      horizontal-pod-autoscaler  Normal  SuccessfulRescale
  (percentage of request) above target
  17m        17m       1      horizontal-pod-autoscaler  Normal  SuccessfulRescale
  (percentage of request) above target
  13m        13m       1      horizontal-pod-autoscaler  Normal  SuccessfulRescale
  (percentage of request) above target
  7m         7m        1      horizontal-pod-autoscaler  Normal  SuccessfulRescale
  1m         1m        1      horizontal-pod-autoscaler  Normal  SuccessfulRescale
  New size: 2; reason: cpu resource utilization
  New size: 4; reason: cpu resource utilization
  New size: 8; reason: cpu resource utilization
  New size: 9; reason: cpu resource utilization
  New size: 3; reason: All metrics below target
  New size: 1; reason: All metrics below target
docker@kubernetes-ms:~$
```

ConfigMap and Secret



ConfigMap and Secret

- Make secret data and configuration great again !
- Many container need some configuration/potential data for make it work. But is it should store in image/configuration (Container, Pods, Deployment, RC etc)?
 - Root password of database
 - Environment variable
 - Custom variable
 - Path of mount volume data
 - Etc
- ConfigMap will provide central configuration for Pods operate
- Secret will encode sensitive data for keep secret



ConfigMap and Secret

- ConfigMap
 - ConfigMap belong to namespace scope
 - Option to create:
 - literal values
 - From file or folder
 - YAML file

```
kubectl create configmap <name> <option>
```

- Ex: “kubectl create configmap webmodule_configmap
--from-literal=REDIS_HOST=localhost”
- Ex: “kubectl create -f webmodule_configmap.yml”

```
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: webmodule_configmap
5    namespace: webmicroservice
6    labels:
7      name: "webmodule_configmap"
8      owner: "Praparn_L"
9      version: "1.0"
10     module: "ConfigMap"
11     environment: "development"
12   data:
13     REDIS_HOST: localhost
```



ConfigMap and Secret

- Secret
 - ConfigMap will encode64 algorithm
 - Option to create:
 - From file <store confidential value>
 - YAML file

```
kubectl create secret generic <name> <option>
```

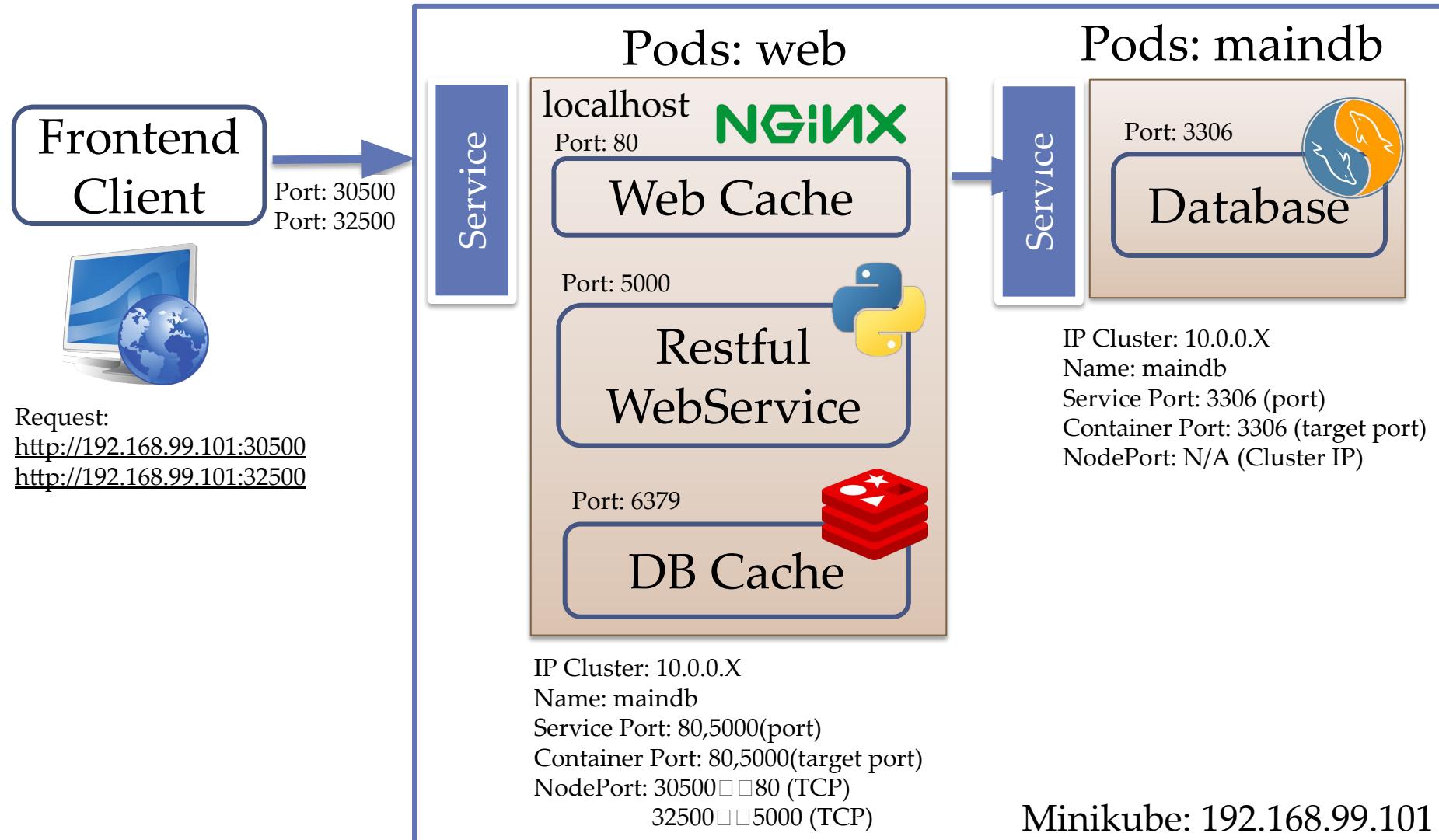
- Ex: “kubectl create secret generic databasemodule_secret --from-file=./username.txt --from-file=./password.txt”
- Ex: “kubectl create -f databasemodule_secret.yml”

```
|praparns-MBP:~ praparn$ echo -n "root"|base64  
cm9vdA==  
|praparns-MBP:~ praparn$ echo -n "password"|base64  
cGFzc3dvcmQ=  
praparns-MBP:~ praparn$ █
```

```
1  apiVersion: v1  
2  kind: Secret  
3  metadata:  
4    name: databasemodule_secret  
5    namespace: webmicroservice  
6    labels:  
7      name: "databasemodule_secret"  
8      owner: "Praparn_L"  
9      version: "1.0"  
10     module: "Secret"  
11     environment: "development"  
12   type: Opaque  
13   data:  
14     username: cm9vdA==  
15     password: cGFzc3dvcmQ=
```



ConfigMap and Secret



ConfigMap and Secret

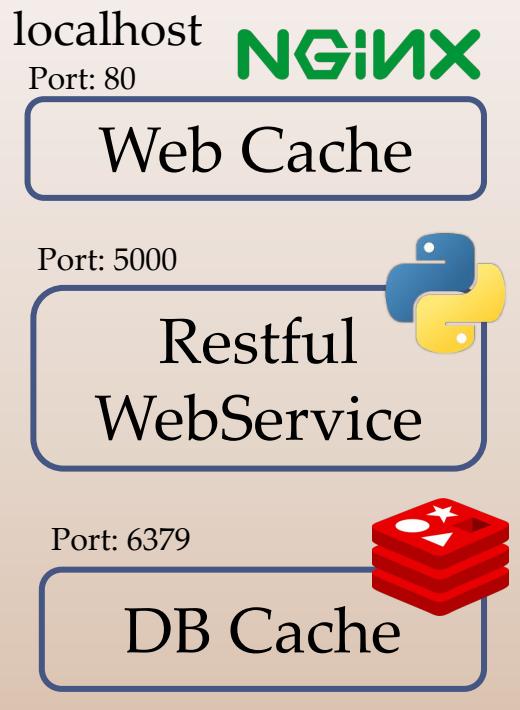
Service



Pods.yml: databasemodule_pod.yml

```
containers:
  - name: maindb
    image: labdocker/mysql:latest
    ports:
      - containerPort: 3306
        protocol: TCP
    env:
      - name: "MYSQL_ROOT_PASSWORD"
        value: "password"
```

Service



Sourcecode: main.py

```
6 CACHE_DB = redis.Redis(host=os.environ.get('REDIS_HOST', 'cachedb'), port=6379)
7 db = MySQLdb.connect("maindb","root","password")
8 MAIN_DB = db.cursor()
```

Pods.yml: webmodule_pod.yml

```
18 - name: webservice
19   image: labdocker/cluster:webservice
20   env:
21     - name: "REDIS_HOST"
22       value: "localhost"
```

Kubernetes: Production Workload Orchestration



kubernetes
by Google

ConfigMap and Secret

Service

Port: 3306

Database



Secret.yml: databasemodule_secret.yml

```
1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: databasemodule-secret
5   namespace: webmicroservice
6   labels:
7     name: "databasemodule-secret"
8     owner: "Praparn_L"
9     version: "1.0"
10    module: "Secret"
11    environment: "development"
12 type: Opaque
13 data:
14   username: cm9vdA==
15   password: cGFzc3dvcmQ=
```



Deploy.yml: databasemodule_deploy_config.yml

```
22 apiVersion: "v1"
23 kind: Deployment
24 metadata:
25   name: maindb
26   labels:
27     name: "maindb"
28     owner: "Praparn_L"
29     version: "1.0"
30   module: "maindb"
31   environment: "development"
32 spec:
33   replicas: 1
34   template:
35     metadata:
36       labels:
37     spec:
38       containers:
39         - name: maindb
40           image: labdocker/mysql:latest
41           ports:
42             - containerPort: 3306
43               protocol: TCP
44           env:
45             - name: username
46               valueFrom:
47                 secretKeyRef:
48                   name: databasemodule-secret
49                   key: username
50             - name: password
51               valueFrom:
52                 secretKeyRef:
53                   name: databasemodule-secret
54                   key: password
```

ConfigMap and Secret

Secret.yaml:

databasemodule_secret.yaml

```
1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: databasemodule-secret
5   namespace: webmicroservice
6   labels:
7     name: "databasemodule-secret"
8     owner: "Praparn_L"
9     version: "1.0"
10    module: "Secret"
11    environment: "development"
12   type: Opaque
13  data:
14    username: cm9vdA==
15    password: cGFzc3dvcmQ=
```

Deploy.yaml:

webmodule_deploy_config.yaml

```
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
```

```
spec:
  containers:
    - name: cachedb
      image: labdocker/redis:latest
      ports:
        - containerPort: 6379
          protocol: TCP
    - name: webservice
      image: labdocker/cluster:webservice
      env:
        - name: REDIS_HOST
          valueFrom:
            configMapKeyRef:
              name: webmodule_configmap
              key: REDIS_HOST
        - name: username
          valueFrom:
            secretKeyRef:
              name: databasemodule-secret
              key: username
        - name: password
          valueFrom:
            secretKeyRef:
              name: databasemodule-secret
              key: password
      ports:
        - containerPort: 5000
          protocol: TCP
```

ConfigMap.yaml:

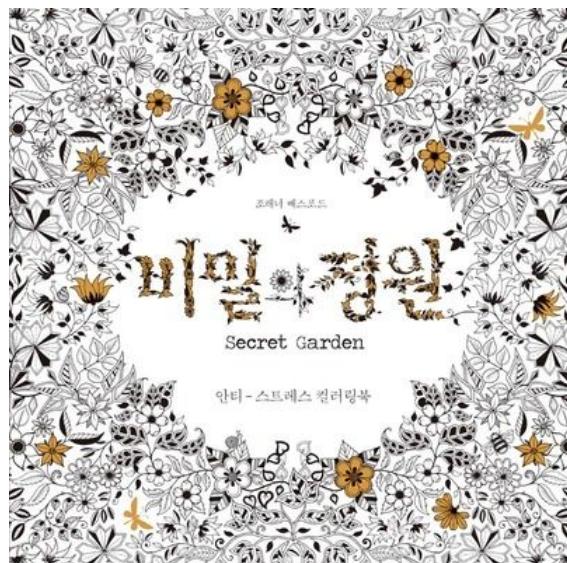
webmodule_configmap.yaml

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: webmodule-configmap
5   namespace: webmicroservice
6   labels:
7     name: "webmodule-configmap"
8     owner: "Praparn_L"
9     version: "1.0"
10    module: "ConfigMap"
11    environment: "development"
12   data:
13     REDIS_HOST: localhost
```



Workshop: ConfigMap and Secret

```
~ — Terminal MAC Pro — zsh          ~ — Terminal MAC Pro — WinSCP.exe TERM...          ~ — Terminal MAC Pro — ssh + docker-mach...          ~ — Terminal MAC Pro — -bash
Environment:
  username:           <set to the key 'username' in secret 'databasemodule-secret'>  Optional: false
  MYSQL_ROOT_PASSWORD: <set to the key 'password' in secret 'databasemodule-secret'>  Optional: false
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from default-token-ts1ql (ro)
Conditions:
  Type      Status
  Initialized  True
  Ready        True
  PodScheduled  True
Volumes:
  default-token-ts1ql:
    Type:     Secret (a volume populated by a Secret)
    SecretName: default-token-ts1ql
    Optional:  false
  QoS Class:  BestEffort
  Node-Selectors: <none>
  Tolerations:  <none>
```



```
[praparn-MacBook-Pro% kubectl describe configmap/webmodule-configmap --namespace webmicroservice
Name:            webmodule-configmap
Namespace:       webmicroservice
Labels:          environment=development
                  module=ConfigMap
                  name=webmodule-configmap
                  owner=Praparn_L
                  version=1.0
Annotations:     <none>

Data
=====
REDIS_HOST:
-----
localhost
[praparn-MacBook-Pro% kubectl describe secret/databasemodule-secret --namespace webmicroservice
Name:            databasemodule-secret
Namespace:       webmicroservice
Labels:          environment=development
                  module=Secret
                  name=databasemodule-secret
                  owner=Praparn_L
                  version=1.0
Annotations:     <none>

Type:  Opaque

Data
=====
password:        8 bytes
username:        4 bytes
```



Recap Day 1

- Container concept (Recap)
- Introduction to Kubernetes
- System Architecture
- Fundamental of Kubernetes
 - Pods, Container and Services
 - Replication Controller (RC)
 - Deployment/Replica-Set (RS) and Rolling update
 - Volume
 - Liveness and Readyness Probe
 - Resource Management and Horizontal Pods Autoscaling (HPA)
 - ConfigMap Secret



Outline Day 2

- Fundamental of Kubernetes
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- Security on Kubernetes
 - Network Policy
 - Volume Policy
 - Resource Usage Policy
 - Resource Consumption Policy
 - Access Control Policy
 - Security Policy
- Kubernetes in real world
 - Cluster Setup for Bare Metal
 - Orchestrator Assignment
 - nodeSelector
 - Interlude
 - Affinity
 - Taints/Tolerations
- Stateful application deployment
 - Consideration and Awareness
 - Persistent Volumes
 - StatefulSets



Question & Answer Section



By: Praparn L (eva10409@gmail.com)



kubernetes
by Google

WORKSHOP ADVANCED DOCKER



สอนการ deploy Dockers ด้วย Kubernetes
จากประสบการณ์ใช้งานจริงบน Production ของ
application ระดับประเทศ



วิทยากร : คุณ PRAPARN LUNGPOONLARP
INFRASTRUCTURE ENGINEER, NETWORK ENGINEER,
SYSTEM ENGINEER



kubernetes



Day 2



Outline Day 2

- Fundamental of Kubernetes
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- Security on Kubernetes
 - Network Policy
 - Volume Policy
 - Resource Usage Policy
 - Resource Consumption Policy
 - Access Control Policy
 - Security Policy
- Kubernetes in real world
 - Cluster Setup for Bare Metal
 - Orchestrator Assignment
 - nodeSelector
 - Interlude
 - Affinity
 - Taints/Tolerations
- Stateful application deployment
 - Consideration and Awareness
 - Persistent Volumes
 - StatefulSets



Job and Cron Jobs



Job and Cron Job

- Some task on kubernetes will batch process or non-interactive job
 - Update EOD Process
 - Monitor System Health
 - Calculate Balance
 - Run reindexing file/database
 - etc
- “Job” on kubernetes was design to operate special purpose
 - Job will track status of complete job
 - Job will autostart new Pods when it failed or deleted
 - Job will delete Pods when job was deleted

```
kubectl create -f <YAML File>
```

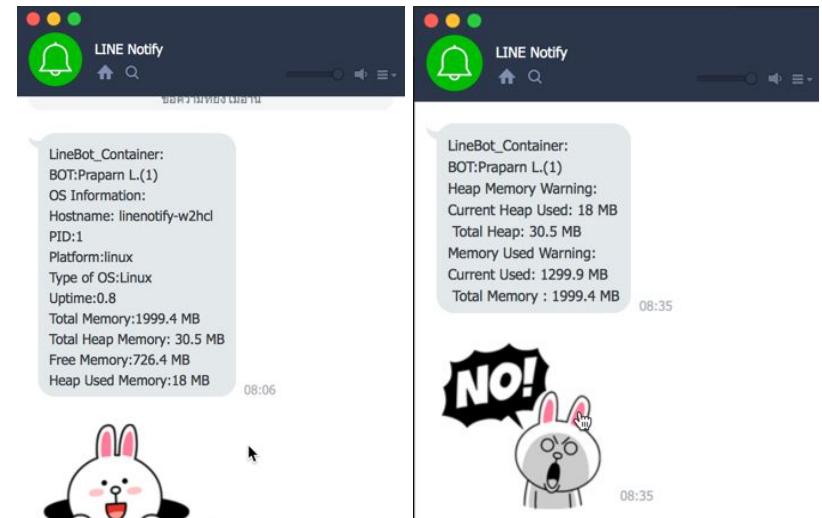


Job and Cron Job

Job.yml:

```
WorkShop_2.2_Job_CronJob › ! job.yml
 1  apiVersion: batch/v1
 2  kind: Job
 3  metadata:
 4    name: linenotify
 5    labels:
 6      name: linenotify
 7      owner: Paparn_L
 8      version: "1.0"
 9      module: Job
10      environment: development
11 spec:
12   ttlSecondsAfterFinished: 30
13   activeDeadlineSeconds: 60
14   template:
15     metadata:
16       name: linenotify
17     spec:
18       containers:
19         - name: linenotify
20           image: labdocker/linenotify:onetime
21           env:
22             - name: TITLE
23               value: "BOT:Paparn_L."
24             - name: INTERVAL
25               value: "10000"
26             - name: HEAP_HIGH
27               value: "20"
28             - name: MEM_HIGH
29               value: "20"
30             - name: SH_OS
31               value: "Y"
32             - name: TOKEN
33               value: "E1eqyillzkzpaCo1R0rebZTcGbIyzDZHp0jcdd0t6CX"
34   restartPolicy: Never
```

```
praparns-MacBook-Pro% kubectl create -f job.yml
job "linenotify" created
praparns-MacBook-Pro% kubectl get jobs
NAME      DESIRED  SUCCESSFUL  AGE
linenotify  1        0          1m
praparns-MacBook-Pro% kubectl get pods
NAME        READY  STATUS  RESTARTS  AGE
linenotify-w2hcl  1/1   Running  0   2m
praparns-MacBook-Pro% kubectl describe jobs/linenotify
Name:            linenotify
Namespace:       default
Selector:        controller-uid=e816fc3f-6e79-11e7-9aa8-08002763e747
Labels:          environment=development
                 module=Job
                 name=linenotify
                 owner=Paparn_L
                 version=1.0
Annotations:    <none>
```



Job and Cron Job



Reference

HOME GETTING STARTED CONCEPTS TASKS TUTORIALS REFERENCE CONTRIBUTE

Reference

Standardized Glossary

- ▶ Kubernetes Issues and Security
- ▶ Using the Kubernetes API
- ▶ Accessing the API
- ▶ API Reference
- ▼ Setup tools reference

Kubeadm

Overview of kubeadm

kubeadm init

kubeadm join

Options

```
--apiserver-advertise-address string      The IP address the API Server will advertise it's listening on. If not set the default netw
--apiserver-bind-port int32                Port for the API Server to bind to. (default 6443)
--apiserver-cert-extra-sans strings        Optional extra Subject Alternative Names (SANs) to use for the API Server serving certifica
--cert-dir string                          The path where to save and store the certificates. (default "/etc/kubernetes/pki")
--certificate-key string                  Key used to encrypt the control-plane certificates in the kubeadm-certs Secret.
--config string                           Path to a kubeadm configuration file.
--cri-socket string                      Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use
--dry-run                                 Don't apply any changes; just output what would be done.
--feature-gates string                   A set of key-value pairs that describe feature gates for various features. No feature gates
--help                                    help for init
--ignore-preflight-errors strings         A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'.
--image-repository string                 Choose a container registry to pull control plane images from (default "k8s.gcr.io")
--kubernetes-version string              Choose a specific Kubernetes version for the control plane. (default "stable-1")
--node-name string                        Specify the node name.
--pod-network-cidr string                Specify range of IP addresses for the pod network. If set, the control plane will automatic
--service-cidr string                     Use alternative range of IP address for service VIPs. (default "10.96.0.0/12")
--service-dns-domain string              Use alternative domain for services, e.g. "myorg.internal". (default "cluster.local")
--skip-certificate-key-print             Don't print the key used to encrypt the control-plane certificates.
--skip-phases strings                   List of phases to be skipped
--skip-token-print                       Skip printing of the default bootstrap token generated by 'kubeadm init'.
--token string                            The token to use for establishing bidirectional trust between nodes and control-plane nodes
--token-ttl duration                    The duration before the token is automatically deleted (e.g. 1s, 2m, 3h). If set to '0', th
--upload-certs                           Upload control-plane certificates to the kubeadm-certs Secret.
```

kubeadm init



savitaashture commented on Dec 13, 2018

Author + ⚡ ...

Thanks a lot.

Its working now.

Earlier i haven't set --feature-gates=TTLAfterFinished=true in controller yaml thats the reason it was not working.

3

TTL Mechanism for Finished Jobs

FEATURE STATE: Kubernetes v1.12 alpha

Another way to clean up finished Jobs (either `Complete` or `Failed`) automatically is to use a TTL mechanism provided by a `TTL` controller for finished resources, by specifying the `.spec.ttlSecondsAfterFinished` field of the Job.

When the TTL controller cleans up the Job, it will delete the Job cascadingly, i.e. delete its dependent objects, such as Pods, together with the Job. Note that when the Job is deleted, its lifecycle guarantees, such as finalizers, will be honored.

For example:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-with-ttl
spec:
  ttlSecondsAfterFinished: 100
  template:
    spec:
      containers:
        - name: pi
          image: perl
          command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
          restartPolicy: Never
```

The Job `pi-with-ttl` will be eligible to be automatically deleted, `100` seconds after it finishes.

<https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm-init/>

<https://github.com/kubernetes/kubernetes/issues/71993>

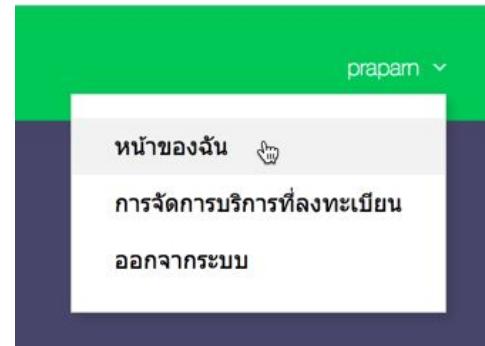
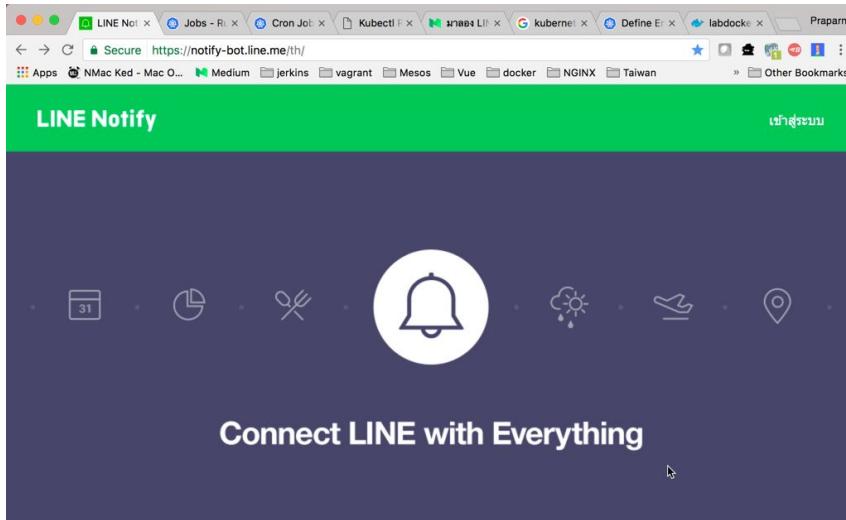
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop: Job and CronJob

- Task0: Generate LINE token
 - <https://notify-bot.line.me>



ออก Access Token (สำหรับผู้พัฒนา)

เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงทะเบียนกับเว็บเซอร์วิส



Workshop: Job and CronJob

ออก Token

โปรดใส่ชื่อ Token (จะแสดงเมื่อมีการแจ้งเตือน)

LINEBOT

โปรดเลือกห้องแขวงที่ต้องการส่งข้อความแจ้งเตือน

Search by group name

 รับการแจ้งเตือนแบบตัวต่อตัวจาก LINE Notify

Token ที่ออก

`zHOlcJCcpIIS8mEedn`

ถ้าออกจากหน้านี้ ระบบจะไม่แสดง Token ที่ออกใหม่ถ้าต่อไป โปรดลอก Token ก่อนออกจากหน้านี้



คัดลอก **ปิด**



2017.07.19 23:42

From: LineBot_Container

To: praparn



Workshop: Job and CronJob

- Task1: Create Jobs for Monitor System via LINE

```
1  apiVersion: batch/v1
2  kind: Job
3  metadata:
4    name: linenotify
5    labels:
6      name: linenotify
7      owner: Praparn_L
8      version: "1.0"
9      module: Job
10     environment: development
11
12    spec:
13      template:
14        metadata:
15          name: linenotify
16        spec:
17          containers:
18            - name: linenotify
19              image: labdocker/linenotify
20              env:
21                - name: TITLE
22                  value: "BOT:Praparn L."
23                - name: INTERVAL
24                  value: "10000"
25                - name: HEAP_HIGH
26                  value: "20"
27                - name: MEM_HIGH
28                  value: "20"
29                - name: SH_OS
30                  value: "N"
31                - name: TOKEN
32                  value: "EIEeqyillzkzpaCo1R0rebZTcGbIyzDZHp0jcd0t6CX"
33
34      restartPolicy: Never
```



TITLE: Input Name
INTERVAL : Input Check Interval (ms)
HEAP_HIGH: % of Heap Memory Warn
MEM_HIGH: % of Memory Used Warn
SH_OS: Mode
(Show information:Y, Show only Warning
Reach:N)
TOKEN: Input LINE TOKEN

Workshop: Job and CronJob

- Task2: Create Cronjob for Monitor System via LINE

```
WorkShop_1.9_Job_CronJob > ! job_notify_cron.yaml
```

```
1  apiVersion: batch/v1beta1
2  kind: CronJob
3  metadata:
4    name: linenotify
5    labels:
6      name: linenotify
7      owner: Praparn_L
8      version: "1.0"
9      module: Job
10     environment: development
11
12    spec:
13      schedule: "*/1 * * * *"
14      successfulJobsHistoryLimit: 5
15      failedJobsHistoryLimit: 1
16      jobTemplate:
17        spec:
18          template:
19            spec:
20              containers:
21                - name: linenotify
22                  image: labdocker/linenotify:onetime
23                  env:
24                    - name: TITLE
25                      value: "BOT NOTIFY:Praparn L."
26                    - name: INTERVAL
27                      value: "10000"
28                    - name: HEAP_HIGH
29                      value: "20"
30                    - name: MEM_HIGH
31                      value: "20"
32                    - name: SH_OS
33                      value: "N"
34                    - name: TOKEN
35                      value: "EIEeqyillzkzpaCo1R0rebZTcGbIyzDZHpojcdd0t6CX"
36
37      restartPolicy: Never
```



```
Every 2.0s: kubectl get jobs
```

NAME	COMPLETIONS	DURATION	AGE
linenotify-1570292460	1/1	14s	3m15s
linenotify-1570292520	1/1	14s	2m15s
linenotify-1570292580	1/1	15s	75s
linenotify-1570292640	1/1	15s	15s

Job and Cron Job

- “CronJob” is time base “Jobs” with
 - Run on specific point-in-time
 - Repeat point in time
 - etc

```
* * * * * command to be executed
- - - -
| | | |
| | | ----- Day of week (0 - 7) (Sunday=0 or 7)
| | | ----- Month (1 - 12)
| | ----- Day of month (1 - 31)
| ----- Hour (0 - 23)
----- Minute (0 - 59)
```

Cron Job Limitations

A cron job creates a job object *about* once per execution time of its schedule. We say “about” because there are certain circumstances where two jobs might be created, or no job might be created. We attempt to make these rare, but do not completely prevent them. Therefore, jobs should be *idempotent*.

If `startingDeadlineSeconds` is set to a large value or left unset (the default) and if `concurrencyPolicy` is set to `Allow`, the jobs will always run at least once.

For every CronJob, the CronJob Controller checks how many schedules it missed in the duration from its last scheduled time until now. If there are more than 100 missed schedules, then it does not start the job and logs the error

```
Cannot determine if job needs to be started. Too many missed start time (> 100). Set or decrease .spec.startingDeadlineSeconds or check clock skew.
```

It is important to note that if the `startingDeadlineSeconds` field is set (not `null`), the controller counts how many missed jobs occurred from the value of `startingDeadlineSeconds` until now rather than from the last scheduled time until now. For example, if `startingDeadlineSeconds` is `200`, the controller counts how many missed jobs occurred in the last 200 seconds.

A CronJob is counted as missed if it has failed to be created at its scheduled time. For example, If `concurrencyPolicy` is set to `Forbid` and a CronJob was attempted to be scheduled when there was a previous schedule still running, then it would count as missed.

For example, suppose a CronJob is set to schedule a new Job every one minute beginning at `08:30:00`, and its `startingDeadlineSeconds` field is not set. If the CronJob controller happens to be down from `08:29:00` to `10:21:00`, the job will not start as the number of missed jobs which missed their schedule is greater than 100.

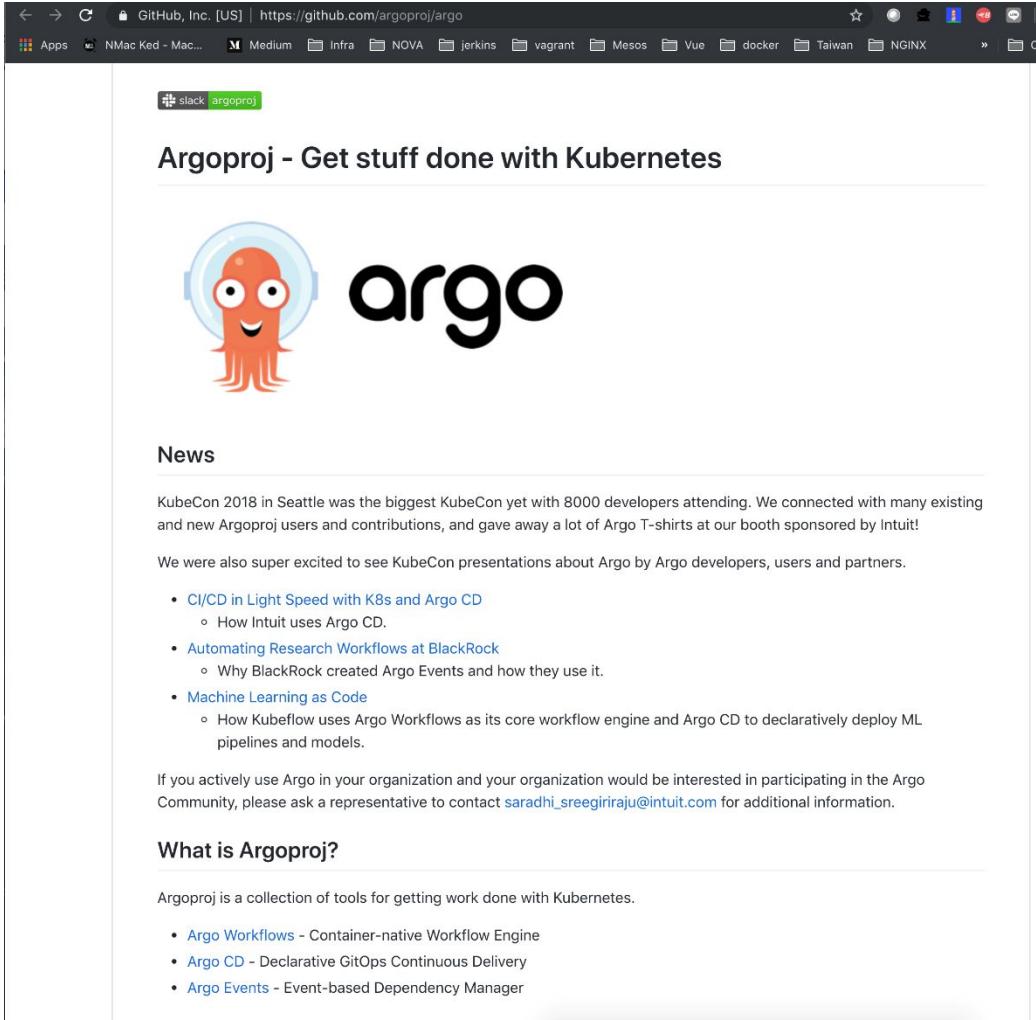
To illustrate this concept further, suppose a CronJob is set to schedule a new Job every one minute beginning at `08:30:00`, and its `startingDeadlineSeconds` is set to 200 seconds. If the CronJob controller happens to be down for the same period as the previous example (`08:29:00` to `10:21:00`), the Job will still start at 10:22:00. This happens as the controller now checks how many missed schedules happened in the last 200 seconds (ie, 3 missed schedules), rather than from the last scheduled time until now.

The CronJob is only responsible for creating Jobs that match its schedule, and the Job in turn is responsible for the management of the Pods it represents.



Job and Cron Job

- WorkFlow Job



The screenshot shows the GitHub repository page for <https://github.com/argoproj/argo>. The page features the Argoproj logo, which includes a cartoon octopus character inside a circular frame next to the word "argo". Below the logo is a section titled "News" containing a summary of KubeCon 2018 and a list of presentations. Further down, there is a section titled "What is Argoproj?" with a brief description and a list of tools: Argo Workflows, Argo CD, and Argo Events.

KubeCon 2018 in Seattle was the biggest KubeCon yet with 8000 developers attending. We connected with many existing and new Argoproj users and contributions, and gave away a lot of Argo T-shirts at our booth sponsored by Intuit!

We were also super excited to see KubeCon presentations about Argo by Argo developers, users and partners.

- [CI/CD in Light Speed with K8s and Argo CD](#)
 - How Intuit uses Argo CD.
- [Automating Research Workflows at BlackRock](#)
 - Why BlackRock created Argo Events and how they use it.
- [Machine Learning as Code](#)
 - How Kubeflow uses Argo Workflows as its core workflow engine and Argo CD to declaratively deploy ML pipelines and models.

If you actively use Argo in your organization and your organization would be interested in participating in the Argo Community, please ask a representative to contact saradhi_sreegiriraju@intuit.com for additional information.

What is Argoproj?

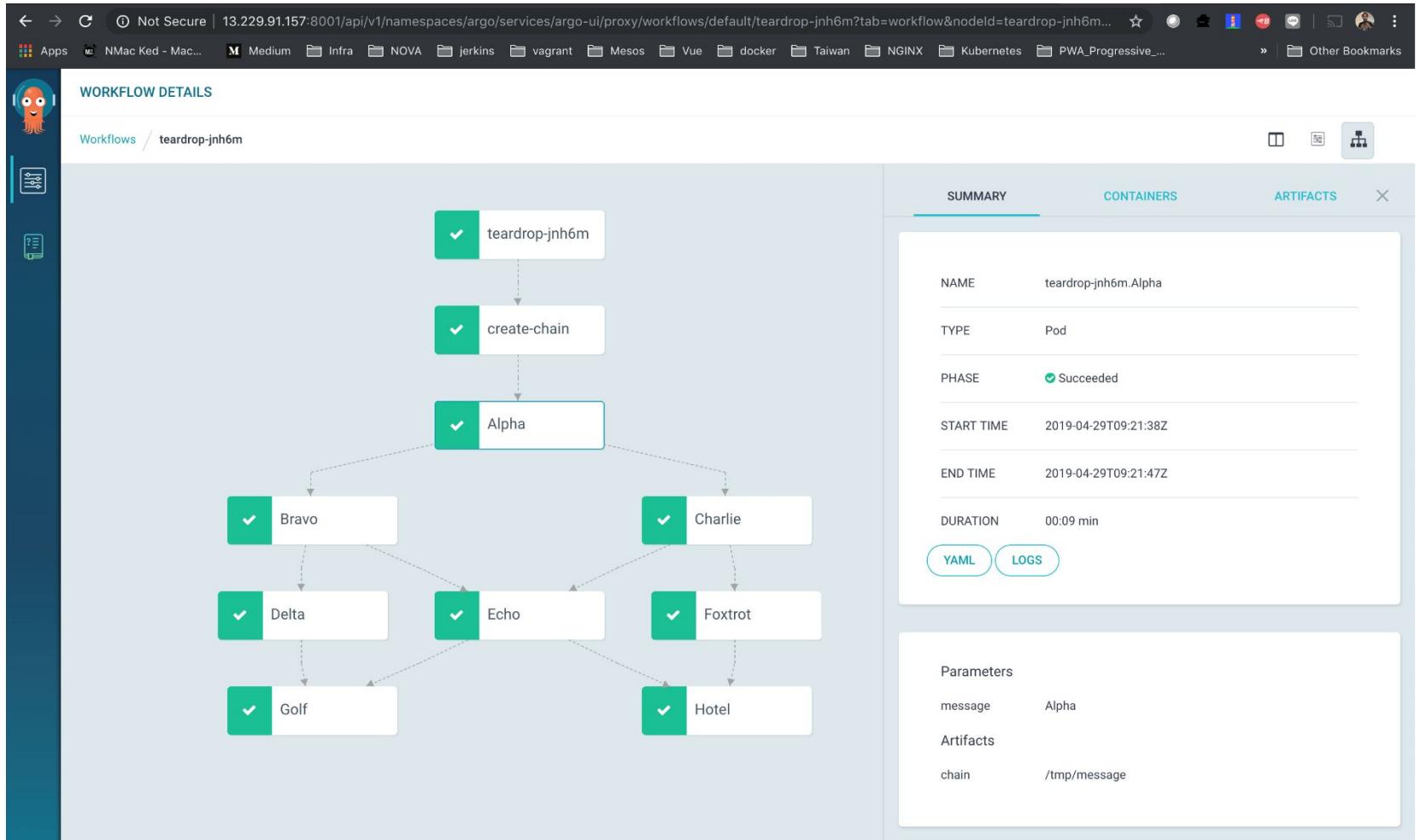
Argoproj is a collection of tools for getting work done with Kubernetes.

- [Argo Workflows](#) - Container-native Workflow Engine
- [Argo CD](#) - Declarative GitOps Continuous Delivery
- [Argo Events](#) - Event-based Dependency Manager

<https://github.com/argoproj/argo>

Job and Cron Job

- WorkFlow Job



<https://github.com/argoproj/argo>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Debug Log and Monitoring



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Debug Log and Monitoring

- Normally kubernetes provide tool for check log on Pods and container level with several option

```
kubectl logs pods/<pods name> -c <container name> <option>
```

- Option:
 - -p, --previous=false ==> Printout last container fail in Pods
 - -f, --follow=false ==> Follow stream log
 - -c, --container ==> Specific container for check log
 - -l, --selector ==> Select filter some label for check log
- Ex:
 - kubectl logs -f pods/maindb -c maindb

Debug Log and Monitoring

- Monitoring kubernetes system via dashboard

kubernetes dashboard

The screenshot shows the Kubernetes dashboard interface under the 'Workloads' tab. On the left is a sidebar with navigation links for Admin, Namespaces, Nodes, Persistent Volumes, Storage Classes, Namespace (default), Workloads (selected), Deployments, Replica Sets, Replication Controllers, Daemon Sets, Stateful Sets, Jobs, Pods, Services and discovery (Services, Ingresses, Storage), and Persistent Volume Claims.

Deployments

Name	Labels	Pods	Age	Images
maindb	environment: development module: maindb	2 / 1	48 seconds	labdocker/mysql:latest
web	environment: development module: web	2 / 1	41 seconds	labdocker/redis:latest labdocker/cluster/webservi... labdocker/cluster/webcach...

Replica Sets

Name	Labels	Pods	Age	Images
maindb-1334333237	environment: development module: maindb	1 / 1	48 seconds	labdocker/mysql:latest
	environment: development module: maindb

The screenshot shows the 'Edit a Service' dialog box. At the top is a search bar and a toolbar with icons for creating, deleting, and saving changes. Below is a tree-view configuration editor showing the service's structure:

```
Service {5}
  kind : Service
  apiVersion : v1
  metadata {7}
    name : kubernetes
    namespace : default
    selfLink : /api/v1/namespaces/default/services/kubernetes
    uid : a0367a8c-640c-11e7-92a2-08002763e747
    resourceVersion : 8
    creationTimestamp : 2017-07-08T18:38:24Z
  labels {2}
    component : apiserver
    provider : kubernetes
  spec {4}
    ports [1]
```

At the bottom are 'CANCEL' and 'UPDATE' buttons.



Debug Log and Monitoring

- Ephemeral Container:
 - Getting debug application inside "Pods"
 - Ideally is run container inside pods for operate as "debugger"
 - Not necessary for external component

```
Normal Started    78s kubelet, ip=10-0-1-228 Started container debugger
[ubuntu@ip-10-0-1-228:~$ more ~/kubernetes_202005/WorkShop_2.2_Log_and_Monitoring/databasemodule_ephemeralcontainers.json
{
  "apiVersion": "v1",
  "kind": "EphemeralContainers",
  "metadata": {
    "name": "maindb-f8dbf5bb7-z95zl",
    "labels": {
      "name": "maindb",
      "owner": "Praparn_L",
      "version": "1.0",
      "module": "maindb",
      "environment": "development"
    }
  },
  "ephemeralContainers": [
    {
      "command": [
        "sh"
      ],
      "image": "alpine:latest",
      "imagePullPolicy": "IfNotPresent",
      "name": "debugger",
      "stdin": true,
      "tty": true,
      "terminationMessagePolicy": "File"
    }
  ]
}
```



Workshop: Log and Monitoring



Ingress Network



Ingress Network

- Remember Service ?
 - Service will expose for other Pods / External to connect and access service on Pods inside

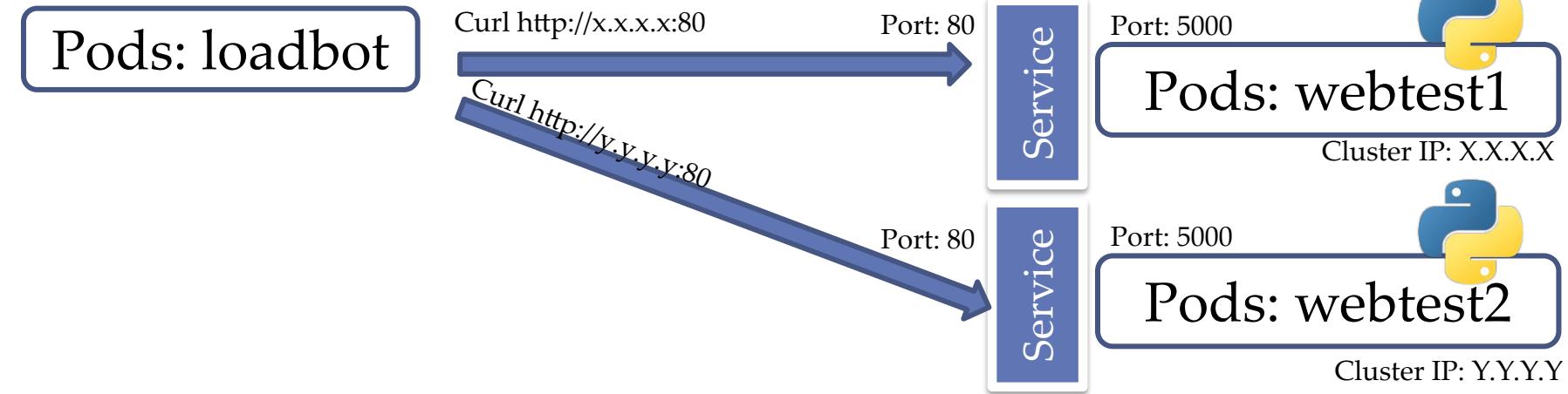
```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP   EXTERNAL-IP   PORT(S)           AGE
kubernetes  10.0.0.1    <none>        443/TCP          4d
maindb     10.0.0.134   <none>        3306/TCP         17m
web        10.0.0.69    <nodes>       5000:30661/TCP,80:30500/TCP  16m
praparns-MacBook-Pro:multicontainer praparn$
```

- Service quite limit and non flexible for operate
 - How to handle for multiple service on same port ?
 - How to limit protocol for access ? (HTTP/HTTPS/FTP etc)
 - How to binding with hostname ? (www.xxxx.yyy)
 - How to TLS connection ?
 - etc
- Ingress is tool will operate for that
 - Give some customize label for classify host and define some selector criteria for specific node run Pods
 - Ingress can be implemented by different controller and specify with ingress class (categories)



Ingress Network

- Service:



Request:
<http://192.168.99.101:30500>
<http://192.168.99.101:32500>

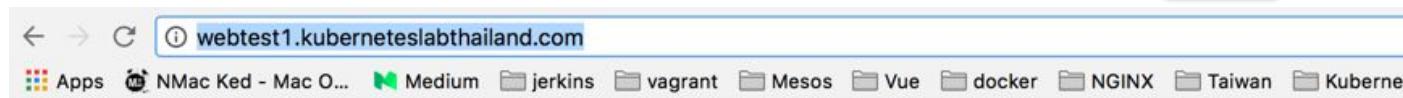
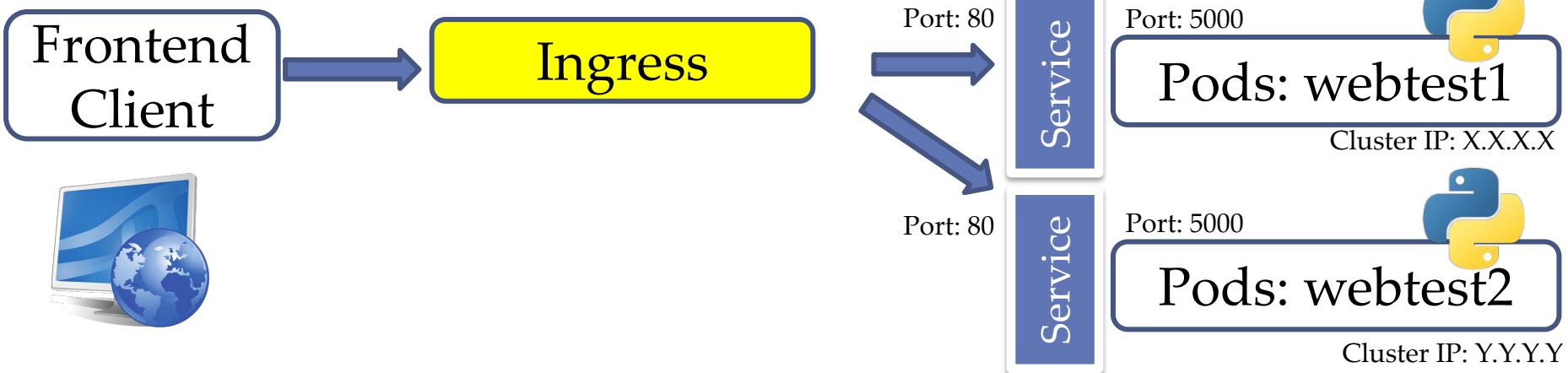
Kubernetes: Production Workload Orchestration



kubernetes
by Google

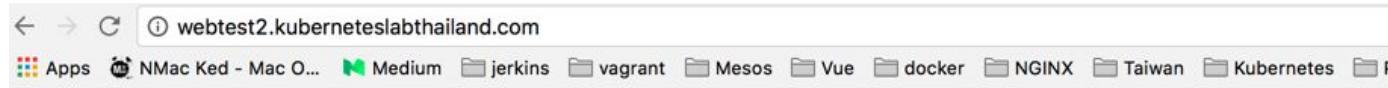
Ingress Network

- Ingress:



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Wed Jul 26 15:44:27 2017



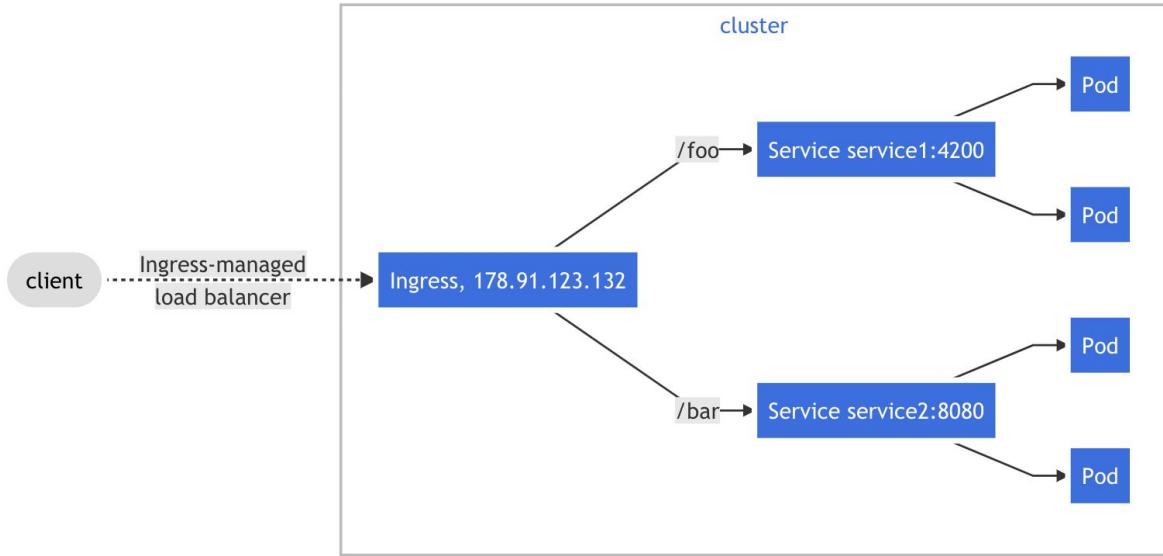
Welcome Page from Container Python Lab Web Version 1.51 RC

Checkpoint Date/Time: Wed Jul 26 15:49:23 2017



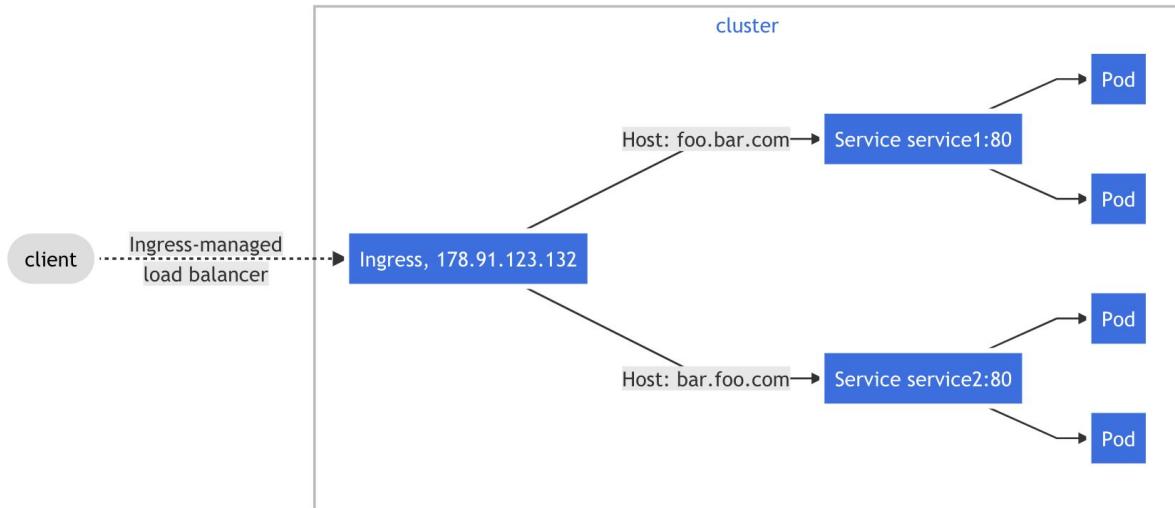
Ingress Network

```
Users > paparnlueangphoonlap > Desktop > !  
1  apiVersion: networking.k8s.io/v1  
2  kind: Ingress  
3  metadata:  
4    name: simple-fanout-example  
5  spec:  
6    rules:  
7      - host: foo.bar.com  
8        http:  
9          paths:  
10            - path: /foo  
11              pathType: Prefix  
12              backend:  
13                service:  
14                  name: service1  
15                  port:  
16                      number: 4200  
17            - path: /bar  
18              pathType: Prefix  
19              backend:  
20                service:  
21                  name: service2  
22                  port:  
23                      number: 8080|
```



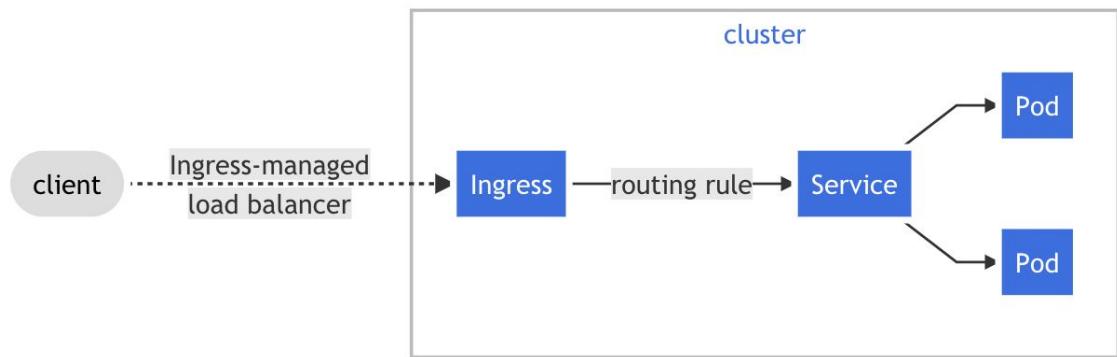
Ingress Network

```
Users > paparnlueangphoonlap > Desktop > !  
1  apiVersion: networking.k8s.io/v1  
2  kind: Ingress  
3  metadata:  
4    name: name-virtual-host-ingress  
5  spec:  
6    rules:  
7      - host: foo.bar.com  
8        http:  
9          paths:  
10            - pathType: Prefix  
11              path: "/"  
12              backend:  
13                service:  
14                  name: service1  
15                  port:  
16                      number: 80  
17      - host: bar.foo.com  
18        http:  
19          paths:  
20            - pathType: Prefix  
21              path: "/"  
22              backend:  
23                service:  
24                  name: service2  
25                  port:  
26                      number: 80
```



Ingress Network

```
Users > praparnlueangphoonlap > Desktop > ! 1.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: name-virtual-host-ingress-no-third-host
5  spec:
6    rules:
7      - host: first.bar.com
8        http:
9          paths:
10         - pathType: Prefix
11           path: "/"
12           backend:
13             service:
14               name: service1
15               port:
16                 number: 80
17      - host: second.bar.com
18        http:
19          paths:
20         - pathType: Prefix
21           path: "/"
22           backend:
23             service:
24               name: service2
25               port:
26                 number: 80
27      - http:
28        paths:
29         - pathType: Prefix
30           path: "/"
31           backend:
32             service:
33               name: service3
34               port:
35                 number: 80
```



Ingress Network

SVC: webtest1

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest1
5   labels:
6     name: webtest1
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    selector:
14      name: webtest1
15      owner: Praparn_L
16      version: "1.0"
17      module: WebServer
18      environment: development
19
20  ports:
21    - port: 80
22      name: http
23      targetPort: 5000
24      protocol: TCP
```



Ingress: ingresswebtest

```
1 apiVersion: networking.k8s.io/v1beta1
2 kind: Ingress
3 metadata:
4   name: ingresswebtest
5
6   spec:
7     rules:
8       - host: webtest1.kuberneteslabthailand.com
9         http:
10           paths:
11             - pathType: Prefix
12               path: "/"
13               backend:
14                 service:
15                   name: webtest1
16                   port:
17                     number: 80
18       - host: webtest2.kuberneteslabthailand.com
19         http:
20           paths:
21             - pathType: Prefix
22               path: "/"
23               backend:
24                 service:
25                   name: webtest2
26                   port:
27                     number: 80
```



SVC: webtest2

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest2
5   labels:
6     name: webtest2
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    selector:
14      name: webtest2
15      owner: Praparn_L
16      version: "1.0"
17      module: WebServer
18      environment: development
19
20  ports:
21    - port: 80
22      name: http
23      targetPort: 5000
24      protocol: TCP
```



Ingress Network

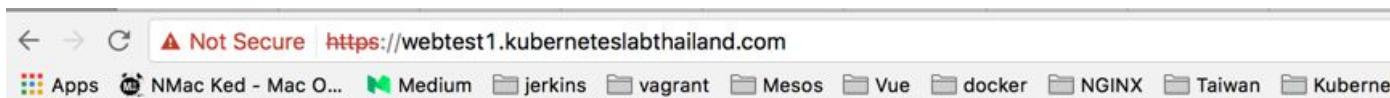
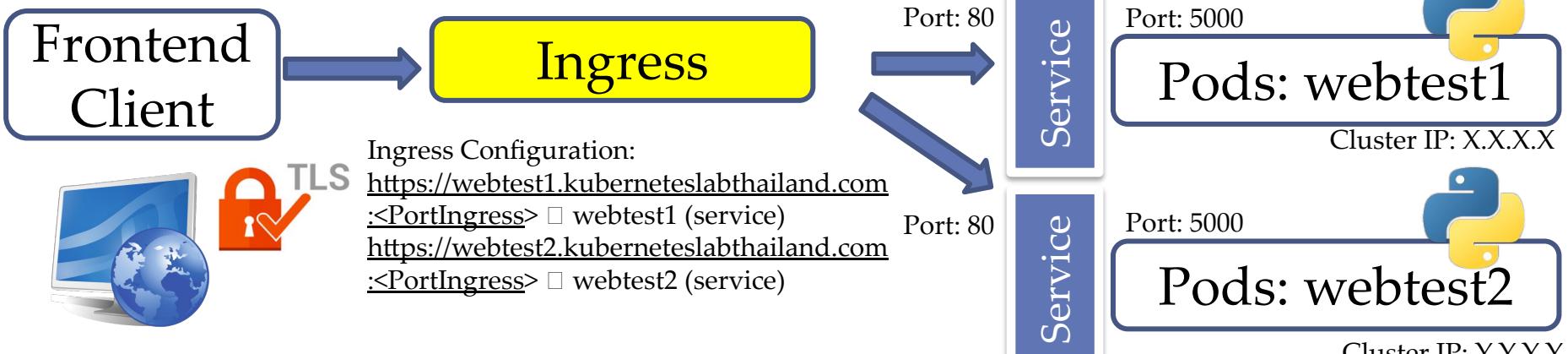
- Path Type:
 - Prefix: Match bases on URL path (Case sensitive)
 - Exact: Match path exactly and with case sensitivity
 - ImplementationSpecific: Use for ingress class
- Multiple match:
 - Longest match path (1)
 - Exact match path (2) (Exact > Prefix)

Ingress Network

Kind	Path(s)	Request path(s)	Matches?
Prefix	/	(all paths)	Yes
Exact	/foo	/foo	Yes
Exact	/foo	/bar	No
Exact	/foo	/foo/	No
Exact	/foo/	/foo	No
Prefix	/foo	/foo , /foo/	Yes
Prefix	/foo/	/foo , /foo/	Yes
Prefix	/aaa/bb	/aaa/bbb	No
Prefix	/aaa/bbb	/aaa/bbb	Yes
Prefix	/aaa/bbb/	/aaa/bbb	Yes, ignores trailing slash
Prefix	/aaa/bbb	/aaa/bbb/	Yes, matches trailing slash
Prefix	/aaa/bbb	/aaa/bbb/ccc	Yes, matches subpath
Prefix	/aaa/bbb	/aaa/bbbxyz	No, does not match string prefix
Prefix	/ , /aaa	/aaa/ccc	Yes, matches /aaa prefix
Prefix	/ , /aaa , /aaa/bbb	/aaa/bbb	Yes, matches /aaa/bbb prefix
Prefix	/ , /aaa , /aaa/bbb	/ccc	Yes, matches / prefix
Prefix	/aaa	/ccc	No, uses default backend
Mixed	/foo (Prefix), /foo (Exact)	/foo	Yes, prefers Exact

Ingress Network

- Ingress (TLS):



Ingress Network

① www.selfsignedcertificate.com

NMac Ked - Mac O... Medium jenkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes PWA_Progressive_... MYSQL_Cluster GeneralKB Nodejs

Self-Signed Certificate Generator

Development Tips About

Self-Signed Certificate Generator

Self-signed ssl certificates can be used to set up temporary ssl servers. You can use it for test and development servers where security is not a big concern. Use the form below to generate a self-signed ssl certificate and key.

Server name: **Generate »**

About SSL Certificates

SSL certificates are required in order to run web sites using the HTTPS protocol. For professional web sites, you usually buy such a certificate from Verisign, Thawte or any other ssl certificate vendor. SSL certificates use a chain of trust, where each certificate is signed (trusted) by a higher, more credible certificate. At the top of the chain of trust are the root certificates, owned by Verisign and others. These certificates are typically shipped with your operating system or web browser.

sslstore

AFFORDABLE
SSL CERTIFICATES

ISSUED IN
MINUTES*

FROM
\$10.99

SAVE 80%



Ingress Network

SVC: webtest1

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest1
5   labels:
6     name: webtest1
7     owner: Paparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    selector:
14      name: webtest1
15      owner: Paparn_L
16      version: "1.0"
17      module: WebServer
18      environment: development
19
20  ports:
21    - port: 80
22      name: http
23      targetPort: 5000
24      protocol: TCP
```

SVC: webtest2

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: webtest2
5   labels:
6     name: webtest2
7     owner: Paparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    selector:
14      name: webtest2
15      owner: Paparn_L
16      version: "1.0"
17      module: WebServer
18      environment: development
19
20  ports:
21    - port: 80
22      name: http
23      targetPort: 5000
24      protocol: TCP
```

Ingress: ingresswebtest

```
1 apiVersion: networking.k8s.io/v1beta1
2 kind: Ingress
3 metadata:
4   name: ingresswebtest
5   spec:
6     tls:
7       - hosts:
8         - webtest1.kuberneteslabthailand.com
9           secretName: webtest1secret
10        - hosts:
11          - webtest2.kuberneteslabthailand.com
12            secretName: webtest2secret
13        rules:
14          - host: webtest1.kuberneteslabthailand.com
15            http:
16              paths:
17                - pathType: Prefix
18                  path: "/"
19                  backend:
20                    service:
21                      name: webtest1
22                      port:
23                        number: 80
24        - host: webtest2.kuberneteslabthailand.com
25          http:
26            paths:
27              - pathType: Prefix
28                  path: "/"
29                  backend:
30                    service:
31                      name: webtest2
32                      port:
33                        number: 80
```

Secret: webtest1

```
1 apiVersion: v1
2 data:
3   tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURMVENDQWhXZ0F3SUJBZ01K
4   tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkJURSBLRVktLS0tLQpNSUlFb3dJQkFBSoNBUVB
5 kind: Secret
6 metadata:
7   name: webtest1secret
8   namespace: default
9 type: Opaque
```

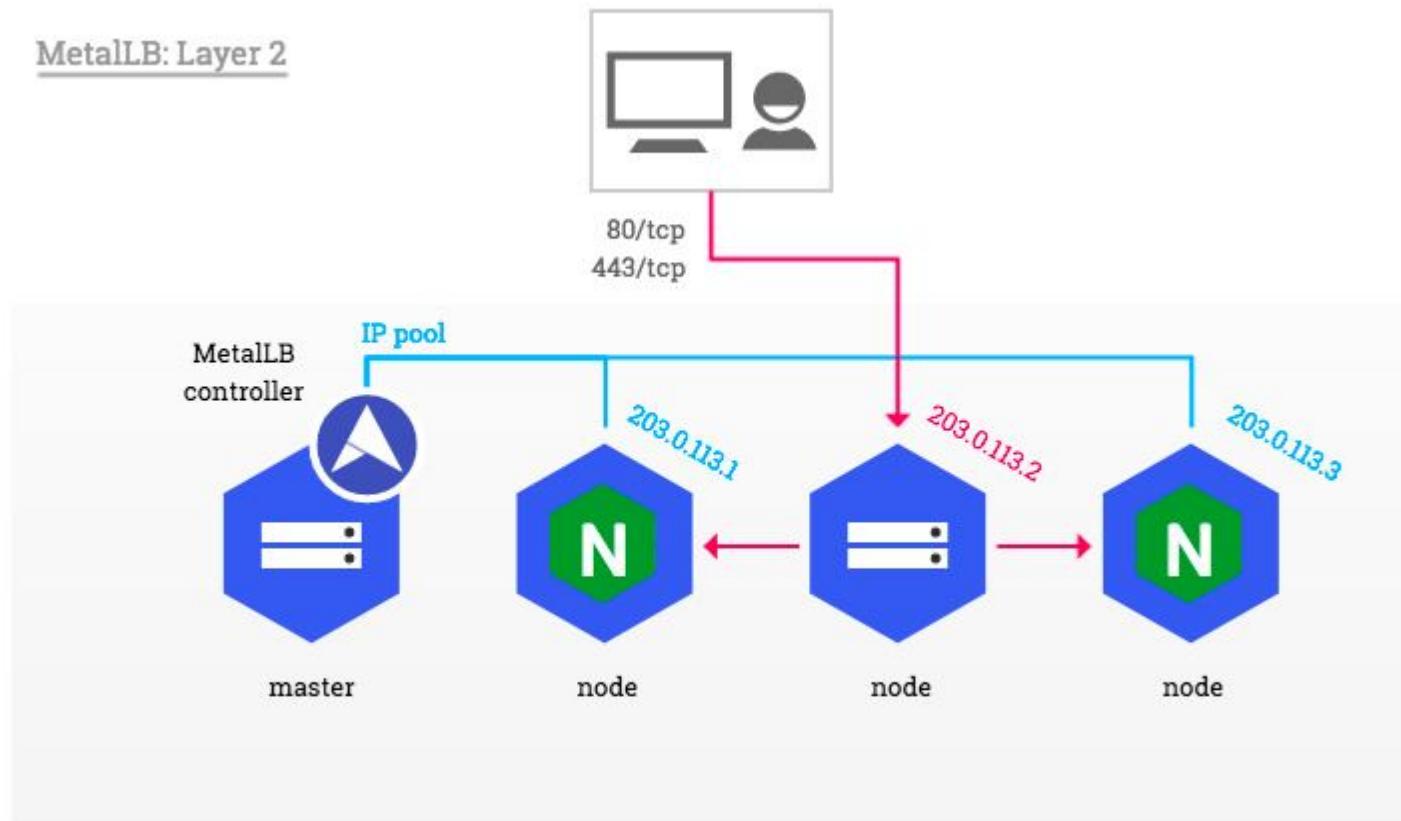
Secret: webtest2

```
1 apiVersion: v1
2 data:
3   tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURMVENDQWhXZ0F3SUJBZ01K
4   tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkJURSBLRVktLS0tLQpNSUlFb3dJQkFBSoNBUVB
5 kind: Secret
6 metadata:
7   name: webtest2secret
8   namespace: default
9 type: Opaque
```



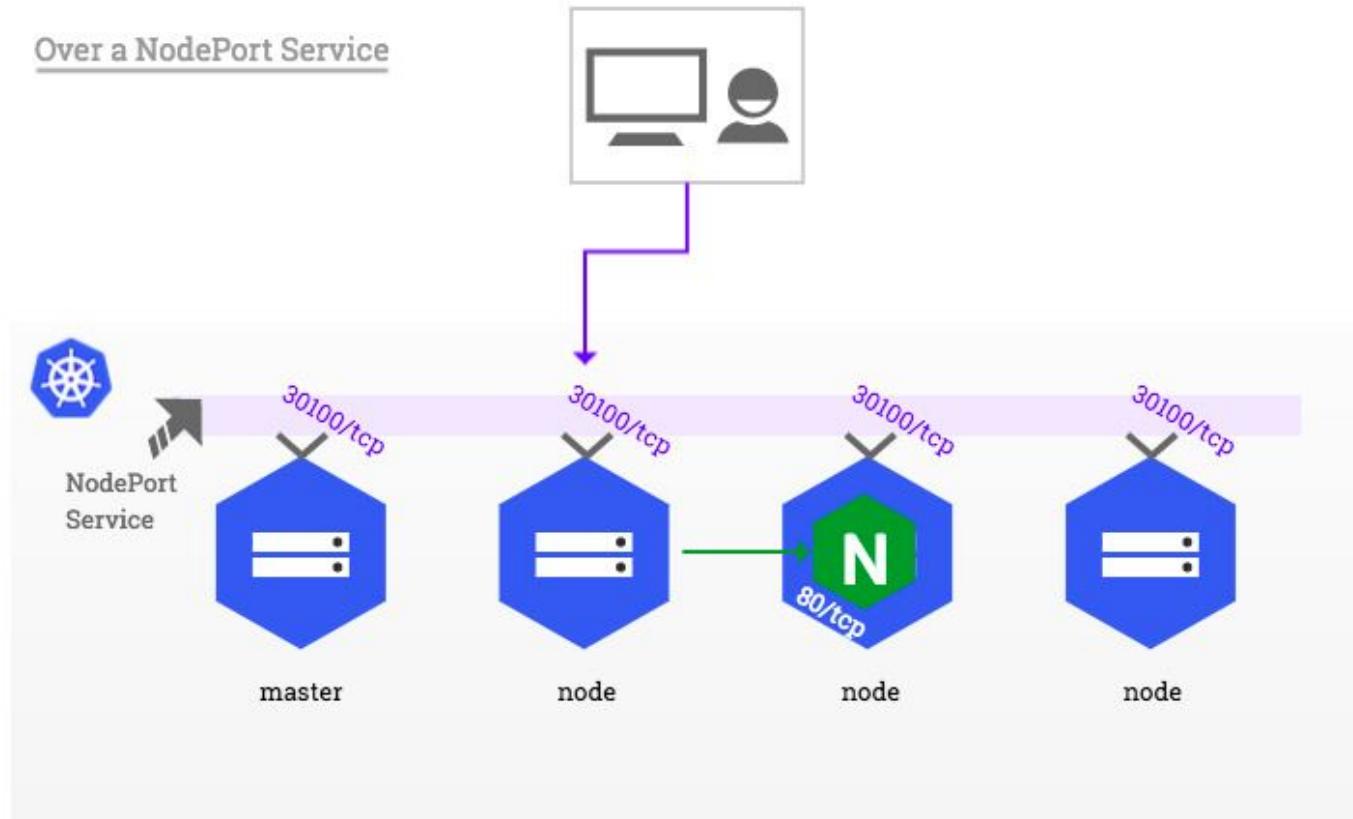
Ingress Network

- Ingress for On-prem strategy
 - MetalLB (L2 Load Balance)



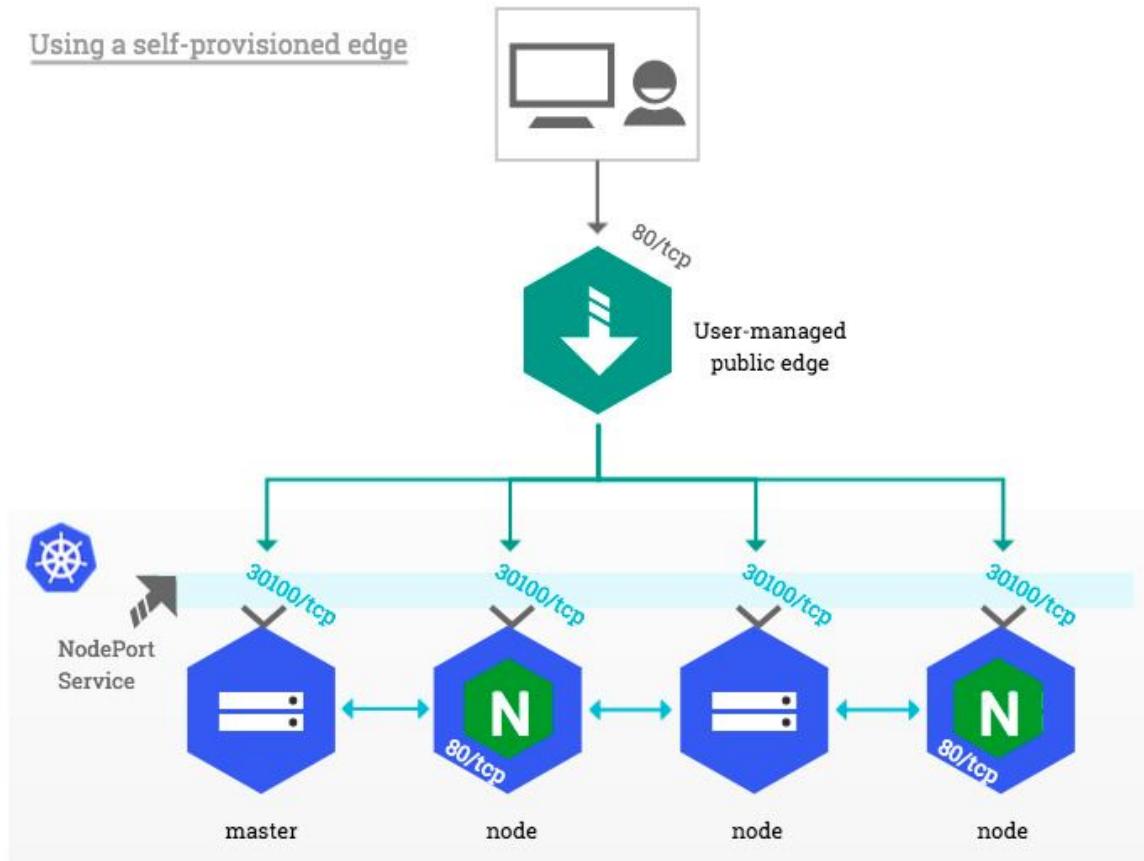
Ingress Network

- Ingress for On-prem strategy
 - NodePort (External Load Balance)



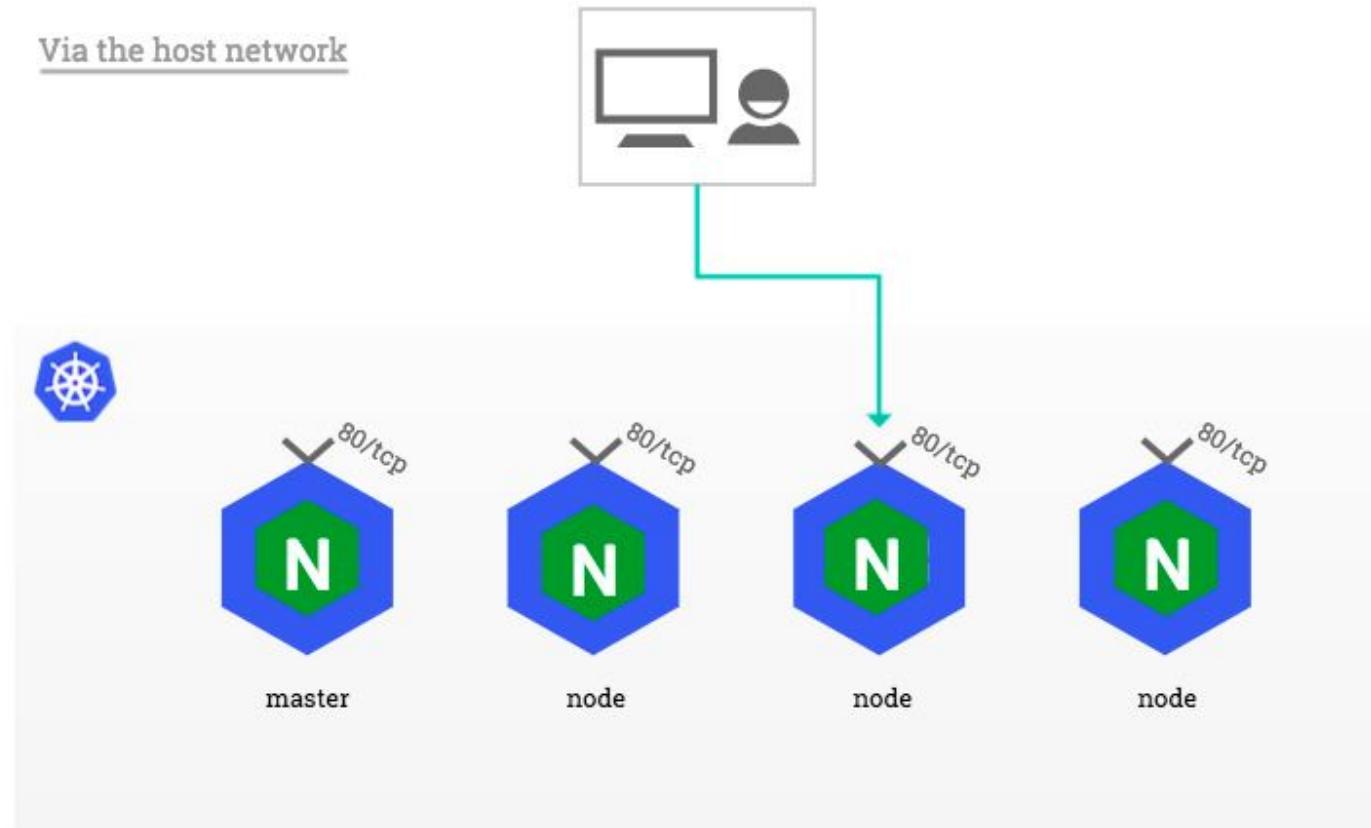
Ingress Network

- Ingress for On-prem strategy
 - NodePort (External Load Balance)



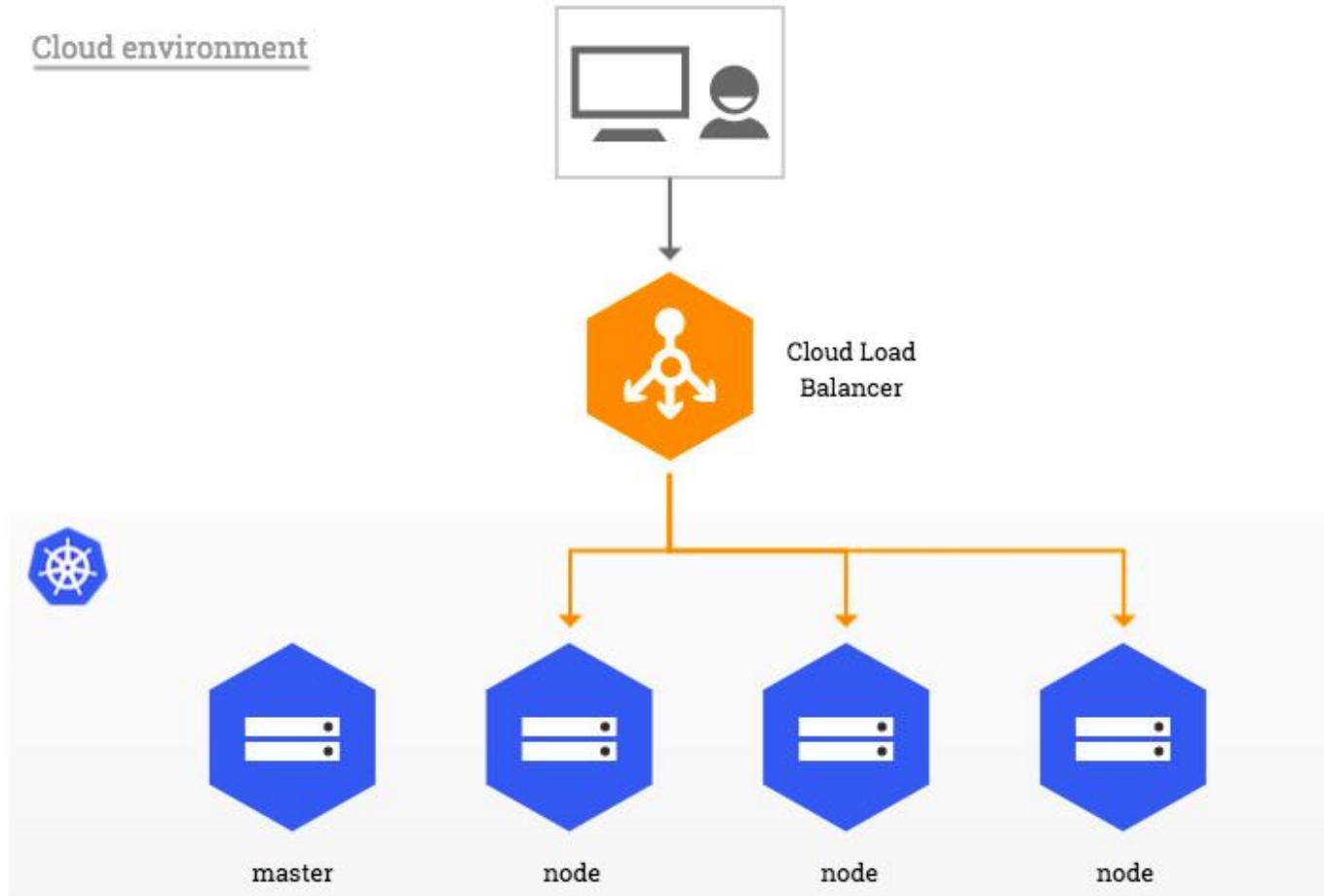
Ingress Network

- Ingress for On-prem strategy
 - Host network (Security concern, Not recommend)



Ingress Network

- Ingress for Cloud strategy



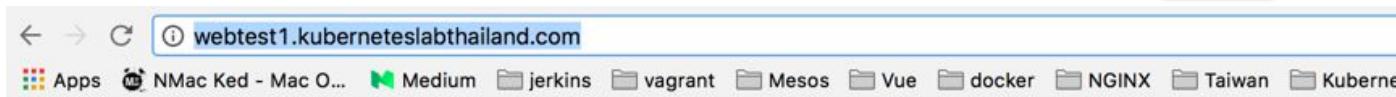
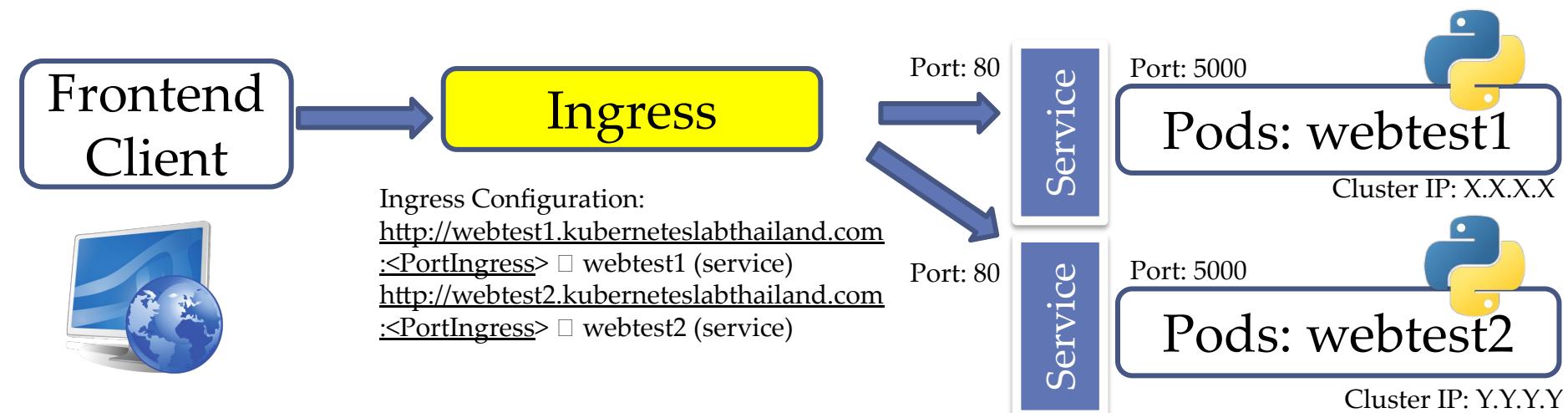
Ingress Network

- Ingress for Cloud strategy (Example: AWS)
 - Create Tag: kubernetes.io/cluster/<Cluster Name> for all resource
 - EC2 (VMWare Machine)
 - VPC
 - Subnet
 - Routing Table

Key	Value	
AZ	ap-southeast-1a	Show Column
Categories	traning	Show Column
Environment	uat	Show Column
Module	kubernetes farm	Show Column
Name	Training_DockerZerotoHero_StudentG5_1	Hide Column
Region	ap-southeast-1	Show Column
Zone	private	Show Column
kubernetes.io/cluster/TrainingAdvanceDockerwithK8SStudentG5	me	Show Column

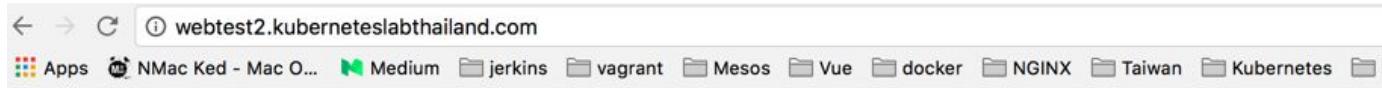


Workshop: Ingress Network



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Wed Jul 26 15:44:27 2017



Welcome Page from Container Python Lab Web Version 1.51 RC

Checkpoint Date/Time: Wed Jul 26 15:49:23 2017



Security



Kubernetes: Production Workload Orchestration



kubernetes
by Google

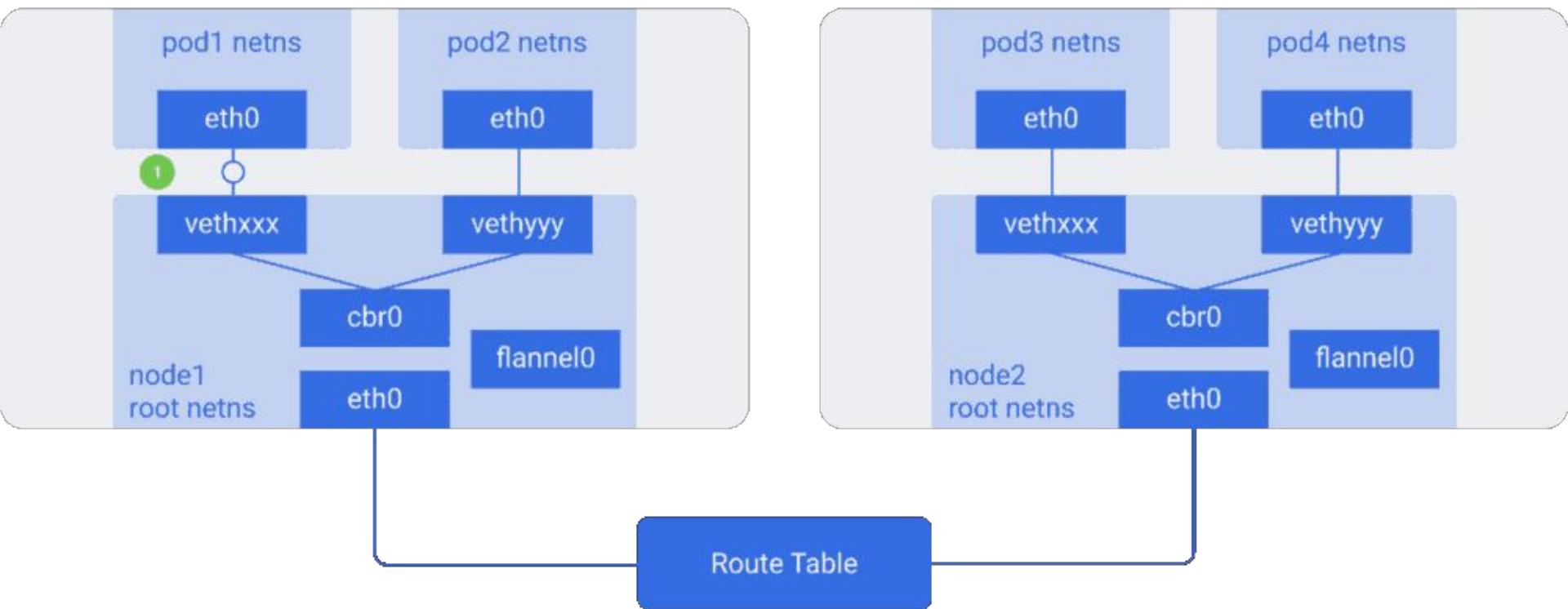
Security

- Kubernetes have several security enhancement for manage in farm
 - **Network Policy:** Control policy for allow/deny connection between endpoint
 - **Volume Policy:** Control/Limit total volume attached to node
 - **Resource Usage Policy:** Control lower/upper acceptable range in namespace (LimitRange)
 - **Resource Consumption Policy:** Limit maximum consumption policy in namespace (Quota)
 - **Access Control Policy:** Deny/Allow fine-grained permission by RBAC, Rule etc
 - **Security Policy(Since 1.15):** Control pods security (Apparmor etc)



Security

- Network Policy:
 - Within same namespace
 - Each service will call by <service> or <service>.<namespace>
 - Across service across namespaces will call <service>.<namespace>



Security

- Network Policy:
 - By default, pods are non-isolated; they accept traffic from any source.
 - Network policy will apply target by “podSelector”, “
 - We can configure network policy base on label
 - Ingress: Control traffic incoming to pods
 - Template:
 - ingress
 - from: <podSelector>/<namespaceSelector>
 - Egress
 - to: <podSelector>/<namespaceSelector>
 - Network Policy type
 - podSelector: Control policy of pods's in/out same namespace (Base on label)
 - namespaceSelector: Control policy of pods's in/out between namespace (Base on label)
 - ipBlock: Control policy by block ip address (Block ip range)

Security



[Documentation](#) [Kubernetes Blog](#)

Search

Home

Getting started

Concepts

Tasks

Install Tools

Administer a Cluster

Administration with kubeadm

Migrating from dockershim

Certificates

Manage Memory, CPU, and API Resources

Install a Network Policy Provider

Access Clusters Using the Kubernetes API

Access Services Running on Clusters

Advertise Extended Resources for a Node

Autoscale the DNS Service in a Cluster

Change the default StorageClass

Change the Reclaim Policy of a PersistentVolume

[Kubernetes Documentation](#) / [Tasks](#) / [Administer a Cluster](#) / [Declare Network Policy](#)

Declare Network Policy

This document helps you get started using the Kubernetes [NetworkPolicy API](#) to declare network policies that govern how pods communicate with each other.

Caution: This section links to third party projects that provide functionality required by Kubernetes. The Kubernetes project authors aren't responsible for these projects. This page follows [CNCF website guidelines](#) by listing projects alphabetically. To add a project to this list, read the [content guide](#) before submitting a change.

Before you begin

You need to have a Kubernetes cluster, and the `kubectl` command-line tool must be configured to communicate with your cluster. If you do not already have a cluster, you can create one by using [minikube](#) or you can use one of these Kubernetes playgrounds:

- [Katacoda](#)
- [Play with Kubernetes](#)

Your Kubernetes server must be at or later than version v1.8. To check the version, enter `kubectl version`.

Make sure you've configured a network provider with network policy support. There are a number of network providers that support [NetworkPolicy](#), including:

- [Calico](#)
- [Cilium](#)
- [Kube-router](#)
- [Romana](#)
- [Weave Net](#)



Security

- Network Policy:

```
WorkShop_2.2_Security > ! policy-default-allow-egress.yml
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: allow-all
5  spec:
6    podSelector: {}
7    egress:
8      - {}
9    policyTypes:
10   - Egress
```

```
WorkShop_2.2_Security > ! policy-default-allow-ingress.yml
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: allow-all
5  spec:
6    podSelector: {}
7    ingress:
8      - {}
9    policyTypes:
10   - Ingress
```

```
WorkShop_2.2_Security > ! policy-backend-policy.yml
1  kind: NetworkPolicy
2  apiVersion: networking.k8s.io/v1
3  metadata:
4    namespace: stars
5    name: backend-policy
6  spec:
7    podSelector:
8      matchLabels:
9        role: backend
10   ingress:
11     - from:
12       - podSelector:
13         matchLabels:
14           role: frontend
15       ports:
16         - protocol: TCP
17           port: 6379
```

```
WorkShop_2.2_Security > ! policy-frontend-policy.yml
1  kind: NetworkPolicy
2  apiVersion: networking.k8s.io/v1
3  metadata:
4    namespace: stars
5    name: frontend-policy
6  spec:
7    podSelector:
8      matchLabels:
9        role: frontend
10   ingress:
11     - from:
12       - namespaceSelector:
13         matchLabels:
14           role: client
15       ports:
16         - protocol: TCP
17           port: 80
```

```
WorkShop_2.2_Security > ! policy-template.yml
2  kind: NetworkPolicy
3  metadata:
4    name: policy-template
5    namespace: stars
6  spec:
7    podSelector:
8      matchLabels:
9        role: backend
10   policyTypes:
11     - Ingress
12     - Egress
13   ingress:
14     - from:
15       - ipBlock:
16         cidr: 172.17.0.0/16
17       except:
18         - 172.17.1.0/24
19     - namespaceSelector:
20       matchLabels:
21         role: management-ui
22   podSelector:
23     matchLabels:
24       role: frontend
25   ports:
26     - protocol: TCP
27       port: 6379
28   egress:
29     - to:
30       - ipBlock:
31         cidr: 10.0.0.0/24
32     ports:
33       - protocol: TCP
34         port: 5978
```

Security

- Network Policy:

Namespace: management-ui

<- Label: role=management-ui>

RC: management-ui

Pods: <label: role=management-ui>

manage-ui-xx

Service

Type: NodePort

Name: management-ui

Service Port: 80

Container Port: 9001

NodePort: 32500



Namespace: stars

<- Label: role=stars>

RC: frontend

Pods: <label: role=frontend>

frontend-xx

Type: Cluster-IP

Name: frontend

Service Port: 80

Container Port: 80

Url: http://frontend.stars

Service

RC: backend

Pods: <label: role=backend>

backend-xx

Type: Cluster-IP

Name: backend

Service Port: 6379

Container Port: 6379

Url: http://backend.stars:6379

Namespace: client

<- Label: role=client>

RC: client

Pods: <label: role=client>

client-xx

Type: Cluster-IP

Name: client

Service Port: 9000

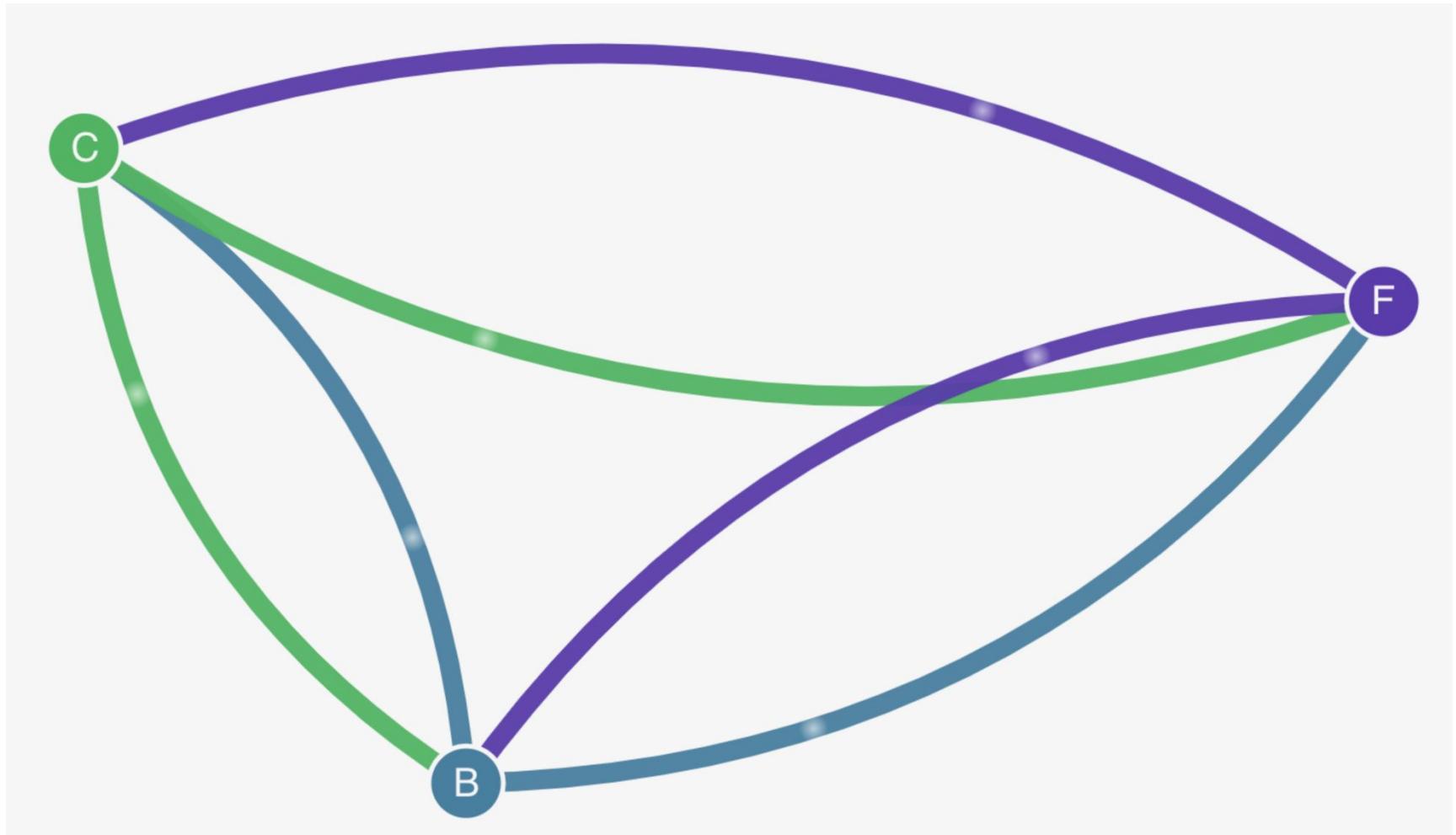
Container Port: 9000

Url: http://client.client:9000

Service

Security

- Network Policy:



Security

- Network Policy:

Namespace: management-ui

<- Label: role=management-ui>

RC: management-ui

Pods: <label: role=management-ui>

manage-ui-xx

Service

Type: NodePort

Name: management-ui

Service Port: 80

Container Port: 9001

NodePort: 32500



```
1 kind: NetworkPolicy
2 apiVersion: networking.k8s.io/v1
3 metadata:
4   name: default-deny
5   namespace: stars
6 spec:
7   podSelector:
8     matchLabels: {}
```

Namespace: stars

<- Label: role=stars>

RC: frontend

Pods: <label: role=frontend>

frontend-xx

Type: Cluster-IP

Name: frontend

Service Port: 80

Container Port: 80

Url: http://frontend.stars

X

Service

RC: backend

Pods: <label: role=backend>

backend-xx

Type: Cluster-IP

Name: backend

Service Port: 6379

Container Port: 6379

Url: http://backend.stars:6379

Namespace: client

<- Label: role=client>

RC: client

Pods: <label: role=client>

client-xx

Type: Cluster-IP

Name: client

Service Port: 9000

Container Port: 9000

Url: http://client.client:9000

X

```
1 kind: NetworkPolicy
2 apiVersion: networking.k8s.io/v1
3 metadata:
4   name: default-deny
5   namespace: client
6 spec:
7   podSelector:
8     matchLabels: {}
```



Security

- Network Policy:

[Toggle Unreachable](#)

[Toggle Markers](#)



Security

- Network Policy:

Namespace: management-ui

<- Label: role=management-ui>

RC: management-ui

Pods: <label: role=management-ui>

manage-ui-xx

Service

Type: NodePort

Name: management-ui

Service Port: 80

Container Port: 9001

NodePort: 32500



Kubernetes: I

Namespace: stars

<- Label: role=stars>

RC: frontend

Pods: <label: role=frontend>

frontend-xx

X

Type: Cluster-IP

Name: frontend

Service Port: 80

Container Port: 80

Url: http://frontend.stars

Y

RC: backend

Pods: <label: role=backend>

backend-xx

Type: Cluster-IP

Name: backend

Service Port: 6379

Container Port: 6379

Url: http://backend.stars:6379

Service

Namespace: client

<- Label: role=client>

RC: client

Pods: <label: role=client>

client-xx

X

Y

Type: Cluster-IP

Name: client

Service Port: 9000

Container Port: 9000

Url: http://client.client:9000

```
1 kind: NetworkPolicy
2 apiVersion: extensions/v1beta1
3 metadata:
4   namespace: stars
5   name: allow-ui
6 spec:
7   podSelector:
8     matchLabels: {}
9   ingress:
10    - from:
11      - namespaceSelector:
12        matchLabels:
13          role: management-ui
```

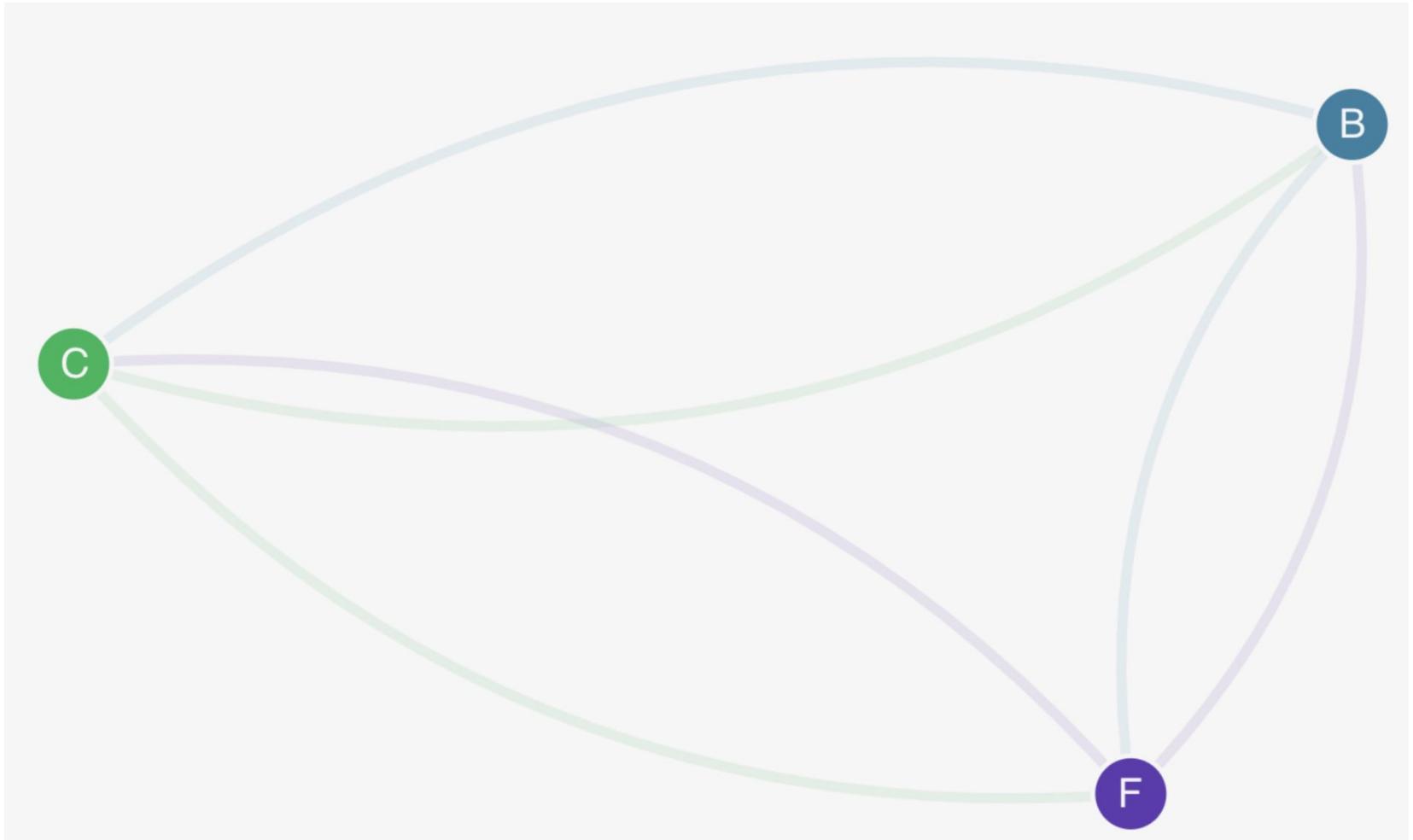
and Orchestration



kubernetes
by Google

Security

- Network Policy:



Security

- Network Policy:

Namespace: management-ui

<- Label: role=management-ui>

RC: management-ui

Pods: <label: role=management-ui>

manage-ui-xx

Service

Type: NodePort

Name: management-ui

Service Port: 80

Container Port: 9001

NodePort: 32500



Kubernetes: Pro

```
1 kind: NetworkPolicy
2 apiVersion: networking.k8s.io/v1
3 metadata:
4   namespace: stars
5   name: frontend-policy
6 spec:
7   podSelector:
8     matchLabels:
9       role: frontend
10    ingress:
11      - from:
12        - podSelector:
13          matchLabels:
14            role: backend
15        ports:
16          - protocol: TCP
17            port: 6379
```

Namespace: stars

<- Label: role=stars>

RC: frontend

Pods: <label: role=frontend>

frontend-xx

Service

Type: Cluster-IP

Name: frontend

Service Port: 80

Container Port: 80

Url: http://frontend.stars

Service

backend

Pods: <label: role=backend>

backend-xx

Type: Cluster-IP

Name: backend

Service Port: 6379

Container Port: 6379

Url: http://backend.stars:6379

Service



Namespace: client

<- Label: role=client>

RC: client

Pods: <label: role=client>

client-xx

Service



Type: Cluster-IP

Name: client

Service Port: 9000

Container Port: 9000

Url: http://client.client:9000

```
1 kind: NetworkPolicy
2 apiVersion: networking.k8s.io/v1
3 metadata:
4   namespace: stars
5   name: frontend-policy
6 spec:
7   podSelector:
8     matchLabels:
9       role: frontend
10    ingress:
11      - from:
12        - namespaceSelector:
13          matchLabels:
14            role: client
15        ports:
16          - protocol: TCP
17            port: 80
```

Orchestration



kubernetes

by Google

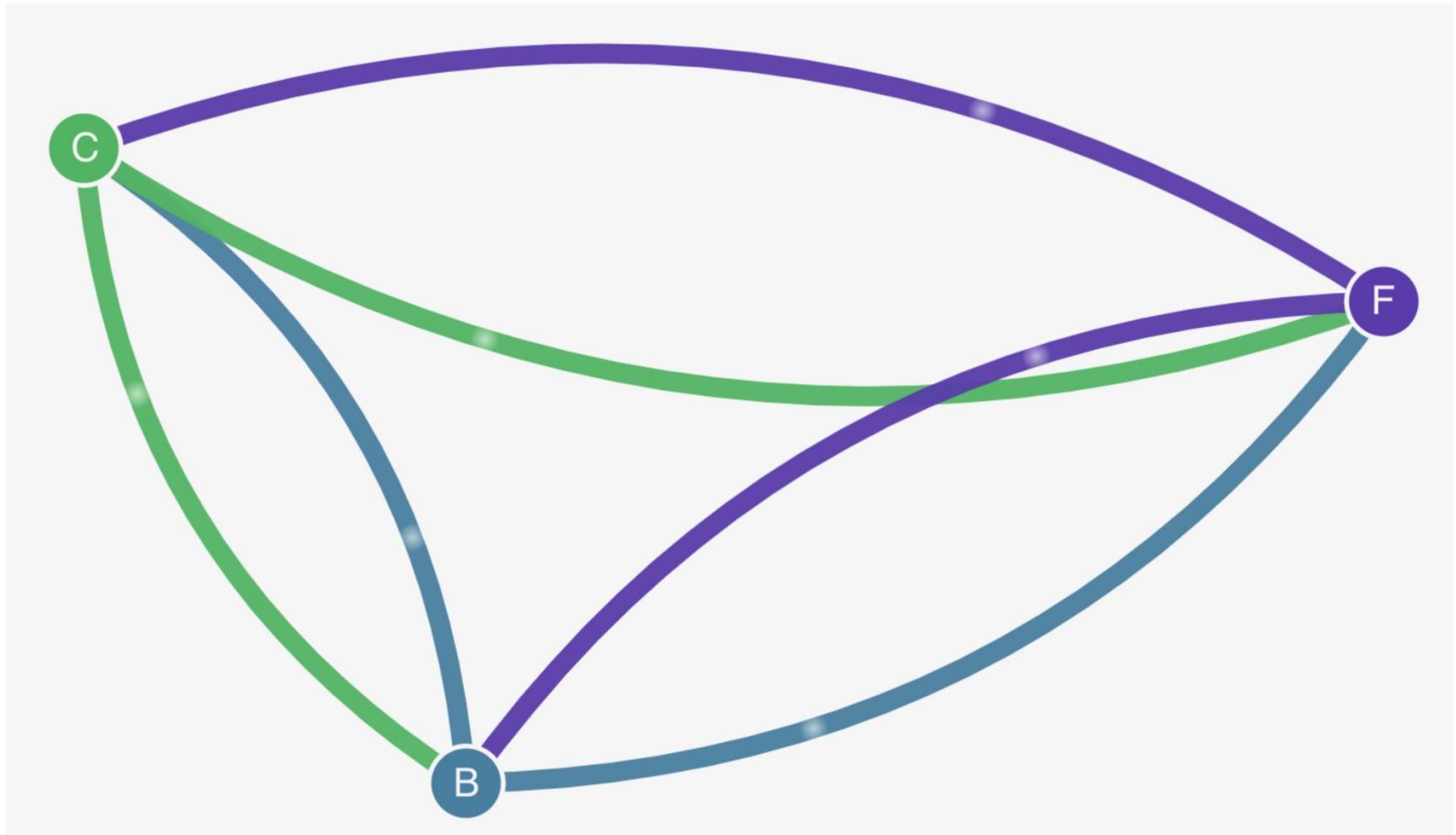
Security

- Network Policy:



Workshop: Security

- Part: Network Policy



Security

- Volume Policy:
 - Kubernetes have default limit volume attached per host
 - Custom this value by environment variable: KUBE_MAX_PD_VOLS



Node-specific Volume Limits

This page describes the maximum number of volumes that can be attached to a Node for various cloud providers.

Cloud providers like Google, Amazon, and Microsoft typically have a limit on how many volumes can be attached to a Node. It is important for Kubernetes to respect those limits. Otherwise, Pods scheduled on a Node could get stuck waiting for volumes to attach.

- [Kubernetes default limits](#)
- [Custom limits](#)
- [Dynamic volume limits](#)

Kubernetes default limits

The Kubernetes scheduler has default limits on the number of volumes that can be attached to a Node:

Cloud service	Maximum volumes per Node
Amazon Elastic Block Store (EBS)	39
Google Persistent Disk	16
Microsoft Azure Disk Storage	16

Ref: <https://kubernetes.io/docs/concepts/storage/storage-limits/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Security

- Resource Usage Policy:
 - Control by “ResourceQuota”

```
1 apiVersion: v1
2 kind: ResourceQuota
3 metadata:
4   name: webtest-quota
5   labels:
6     name: webtest_quota
7     owner: Praparn_L
8     version: "1.0"
9     module: Quota
10    environment: development
11 spec:
12   hard:
13     pods: "4"
14     requests.cpu: "1"
15     requests.memory: 1Gi
16     limits.cpu: "4"
17     limits.memory: 4Gi
```

- Resource Consume Policy:
 - Control by “LimitRange”

```
1 apiVersion: v1
2 kind: LimitRange
3 metadata:
4   name: webtest_limit
5   labels:
6     name: webtest_limit
7     owner: Praparn_L
8     version: "1.0"
9     module: LimitRange
10    environment: development
11 spec:
12   limits:
13     - max:
14       cpu: "1"
15       memory: 1Gi
16     min:
17       cpu: 200m
18       memory: 6Mi
19     type: Pod
20   default:
21     cpu: 300m
22     memory: 200Mi
23   defaultRequest:
24     cpu: 200m
25     memory: 100Mi
26   max:
27     cpu: "1"
28     memory: 1Gi
29   min:
30     cpu: 100m
31     memory: 3Mi
32   type: Container
```



Security

- Access Control Policy
 - Normally Kubernetes manage privilege via RBAC (Role-base access control) for flexibility to change and assign privilege in role
 - Account for access with cluster will define to “User Account/Service Account”
 - RBAC normally describe in 4 categories
 - Role and ClusterRole:
 - RoleBinding and ClusterRoleBinding:
 - Referring to Resources:
 - Aggregated ClusterRoles:

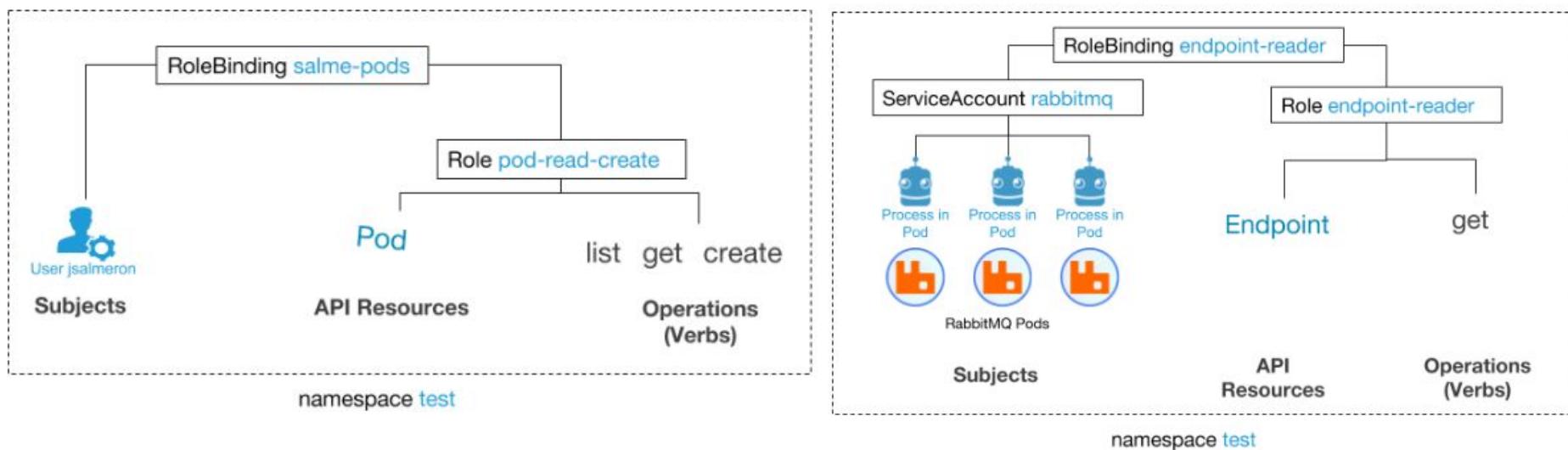


Security

- Access Control Policy

- Account Management

- User accounts are for humans. Service accounts are for processes, which run in pods.
 - User accounts are global scope. Names must be unique across all namespaces of a cluster.
 - Auditing considerations for humans and service accounts may differ.



Ref: <https://www.cncf.io/blog/2018/08/01/demystifying-rbac-in-kubernetes/>
Kubernetes: Production Workload Orchestration

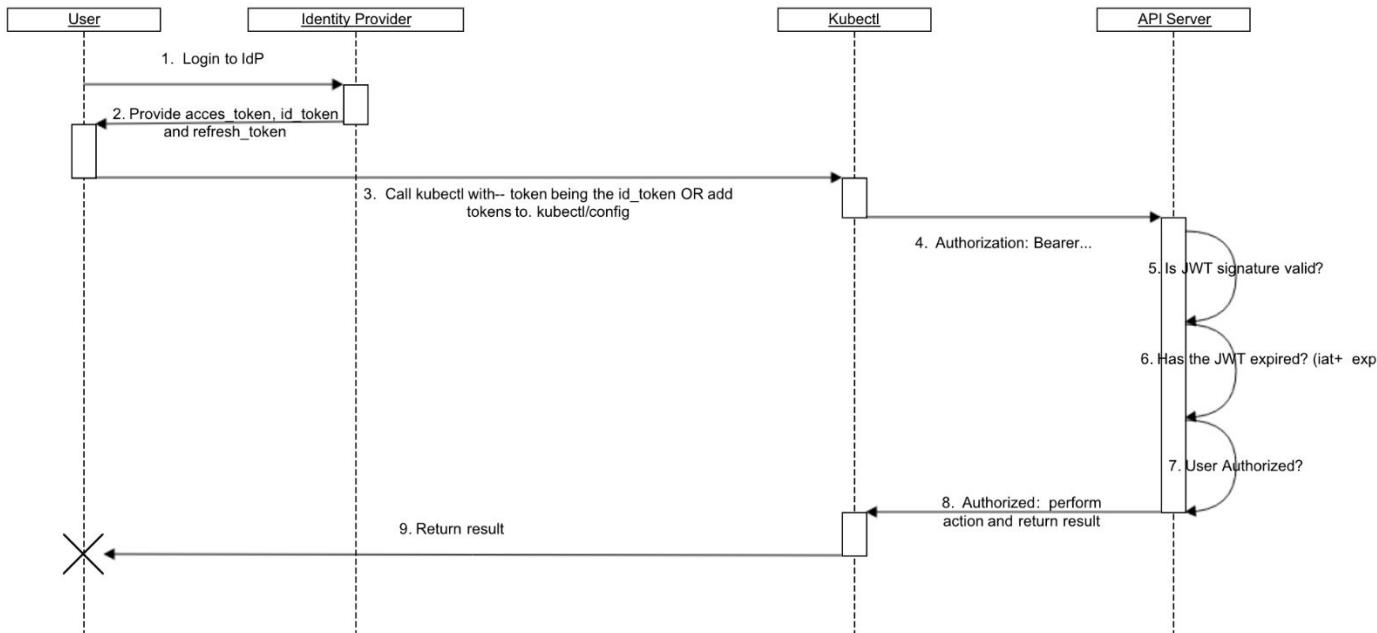
Security

- Access Control Policy
 - Account Management
 - Support OpenID.

OpenID Connect Tokens

OpenID Connect is a flavor of OAuth2 supported by some OAuth2 providers, notably Azure Active Directory, Salesforce, and Google. The protocol's main extension of OAuth2 is an additional field returned with the access token called an ID Token. This token is a JSON Web Token (JWT) with well known fields, such as a user's email, signed by the server.

To identify the user, the authenticator uses the `id_token` (not the `access_token`) from the OAuth2 [token response](#) as a bearer token. See [above](#) for how the token is included in a request.



Ref: <https://kubernetes.io/docs/reference/access-authn-authz/authentication/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Security

- Access Control Policy

- **Account Management**

- Service Accounts

```
1  apiVersion: v1
2  kind: ServiceAccount
3  metadata:
4    name: serviceaccount-readonly
```

- User Accounts

- No yaml file for create user account
 - Create via “openssl/kubectl” operate
 - private key for user and create csr (certificate sign request)(openssl)
 - generate final certificate from CA of Kubernetes CA (openssl)
 - create context for user via kubectl

Security

- Access Control Policy
 - **RBAC Operate**
 - Role and ClusterRole
 - Role: Describe rule for allow action on namespace level
 - ClusterRole: Describe rule for allow action on cluster level
 - Structure of syntax
 - rules
 - apiGroup[""] □ Core API Group
 - resource[""] □ King of resource to manage
 - verbs[""] □ Action to allow
 - Default ClusterRole can check on this
<https://github.com/kubernetes/kubernetes/blob/master/plugin/pkg/auth/authorizer/rbac/bootstrappolicy/testdata/cluster-roles.yaml>

Security

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: Role
3 metadata:
4   namespace: security
5   name: templaterole
6 rules:
7 - apiGroups: [""]
8   resources: ["pods"]
9   verbs: ["get", "watch", "list"]
10 - apiGroups: ["apps"]
11   resources: ["deployments"]
12   verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
13 - apiGroups: ["batch"]
14   resources: ["jobs"]
15   verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
16 - apiGroups: [""]
17   resources: ["configmaps"]
18   resourceNames: ["testconfigmap"]
19   verbs: ["get"]
20 - apiGroups: [""]
21   resources: ["nodes"]
22   verbs: ["get", "list", "watch"]
```

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: Role
3 metadata:
4   namespace: security
5   name: rolereadonly
6 rules:
7 - apiGroups: [ "", "apps" ]
8   resources: ["deployments", "replicasets", "pods"]
9   verbs: ["get", "list", "watch"]
```

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: webmodule-configmap
5   namespace: webmicroservice
6   labels:
7     name: webmodule-configmap
8     owner: "Praparn_L"
9     version: "1.0"
10    module: "ConfigMap"
11    environment: "development"
12 data:
13   REDIS_HOST: localhost
14
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webtest1
  labels:
    name: webtest1
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  replicas: 1
  selector:
    matchLabels:
      name: webtest1
```

WorkShop_1.9_Job_CronJob > ! job.yml

```
1 apiVersion: batch/v1
2 kind: Job
3 metadata:
4   name: linenotify
5   labels:
6     name: linenotify
7     owner: Praparn_L
8     version: "1.0"
9     module: Job
10    environment: development
11 spec:
12   ttlSecondsAfterFinished: 30
13   activeDeadlineSeconds: 60
14   template:
15     metadata:
16       name: linenotify
17     spec:
18       containers:
19         - name: linenotify
20           image: labdocker/linenotify:onetime
21
```



Security

```
1 kind: ClusterRole
2 metadata:
3   annotations:
4     | rbac.authorization.kubernetes.io/autoupdate: "true"
5   labels:
6     name: clusterrolereadonly
7   rules:
8     - apiGroups: [ "", "apps", "extensions" ]
9     resources: ["*"]
10    verbs: ["get", "list", "watch"]
```

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   name: templateclusterrole
5   labels:
6     | rbac.kubernetestthailand.com/security: "true"
7   rules:
8     - apiGroups: [""]
9       resources: ["pods"]
10      verbs: ["get", "watch", "list"]
11     - apiGroups: ["apps"]
12       resources: ["deployments"]
13       verbs: ["get", "list"]
14     - apiGroups: ["batch"]
15       resources: ["jobs"]
16       verbs: ["get", "list"]
17     - apiGroups: [""]
18       resources: ["nodes"]
19       verbs: ["get", "list", "watch"]
```



Security

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND	VERBS
bindings			true	Binding	[create] [get list]
componentstatuses	cs		false	ComponentStatus	[create delete deletecollection get list patch update watch]
configmaps	cm		true	ConfigMap	[create delete deletecollection get list patch update watch]
endpoints	ep		true	Endpoints	[create delete deletecollection get list patch update watch]
events	ev		true	Event	[create delete deletecollection get list patch update watch]
limitranges	limits		true	LimitRange	[create delete deletecollection get list patch update watch]
namespaces	ns		false	Namespace	[create delete get list patch update watch]
nodes	no		false	Node	[create delete deletecollection get list patch update watch]
persistentvolumeclaims	pvc		true	PersistentVolumeClaim	[create delete deletecollection get list patch update watch]
persistentvolumes	pv		false	PersistentVolume	[create delete deletecollection get list patch update watch]
pods	po		true	Pod	[create delete deletecollection get list patch update watch]
podtemplates			true	PodTemplate	[create delete deletecollection get list patch update watch]
replicationcontrollers	rc		true	ReplicationController	[create delete deletecollection get list patch update watch]
resourcequotas	quota		true	ResourceQuota	[create delete deletecollection get list patch update watch]
secrets			true	Secret	[create delete deletecollection get list patch update watch]
serviceaccounts	sa		true	ServiceAccount	[create delete deletecollection get list patch update watch]
services	svc		true	Service	[create delete get list patch update watch]
mutatingwebhookconfigurations		admissionregistration.k8s.io	false	MutatingWebhookConfiguration	[create delete deletecollection get list patch update watch]
validatingwebhookconfigurations		admissionregistration.k8s.io	false	ValidatingWebhookConfiguration	[create delete deletecollection get list patch update watch]
customresourcedefinitions	crd,crds	apiextensions.k8s.io	false	CustomResourceDefinition	[create delete deletecollection get list patch update watch]
apiservices		apiregistration.k8s.io	false	APIService	[create delete deletecollection get list patch update watch]
controllerrevisions		apps	true	ControllerRevision	[create delete deletecollection get list patch update watch]
daemonsets	ds	apps	true	DaemonSet	[create delete deletecollection get list patch update watch]
deployments	deploy	apps	true	Deployment	[create delete deletecollection get list patch update watch]
replicasesets	rs	apps	true	ReplicaSet	[create delete deletecollection get list patch update watch]
statefulsets	sts	apps	true	StatefulSet	[create delete deletecollection get list patch update watch]
tokenreviews		authentication.k8s.io	false	TokenReview	[create delete deletecollection get list patch update watch]
localsubjectaccessreviews		authorization.k8s.io	true	LocalSubjectAccessReview	[create]
selfsubjectaccesreviews		authorization.k8s.io	false	SelfSubjectAccessReview	[create]
selfsubjectrulesreviews		authorization.k8s.io	false	SelfSubjectRulesReview	[create]
subjectaccesreviews		authorization.k8s.io	false	SubjectAccessReview	[create]
horizontalpodautoscalers	hpa	autoscaling	true	HorizontalPodAutoscaler	[create delete deletecollection get list patch update watch]
cronjobs	cj	batch	true	CronJob	[create delete deletecollection get list patch update watch]
jobs		batch	true	Job	[create delete deletecollection get list patch update watch]
certificatesigningrequests	csr	certificates.k8s.io	false	CertificateSigningRequest	[create delete deletecollection get list patch update watch]
leases		coordination.k8s.io	true	Lease	[create delete deletecollection get list patch update watch]
bgpconfigurations		crd.projectcalico.org	false	BGPConfiguration	[delete deletecollection get list patch create update watch]
bgppeers		crd.projectcalico.org	false	BGPPeer	[delete deletecollection get list patch create update watch]
blockaffinities		crd.projectcalico.org	false	BlockAffinity	[delete deletecollection get list patch create update watch]
clusterinformations		crd.projectcalico.org	false	ClusterInformation	[delete deletecollection get list patch create update watch]
felixconfigurations		crd.projectcalico.org	false	FelixConfiguration	[delete deletecollection get list patch create update watch]
globalnetworkpolicies		crd.projectcalico.org	false	GlobalNetworkPolicy	[delete deletecollection get list patch create update watch]
globalnetworksets		crd.projectcalico.org	false	GlobalNetworkSet	[delete deletecollection get list patch create update watch]
hostendpoints		crd.projectcalico.org	false	HostEndpoint	[delete deletecollection get list patch create update watch]
ipamblocks		crd.projectcalico.org	false	IPAMBlock	[delete deletecollection get list patch create update watch]
ipamconfigs		crd.projectcalico.org	false	IPAMConfig	[delete deletecollection get list patch create update watch]
ipamhandles		crd.projectcalico.org	false	IPAMHandle	[delete deletecollection get list patch create update watch]
ippools		crd.projectcalico.org	false	IPPool	[delete deletecollection get list patch create update watch]
networkpolicies		crd.projectcalico.org	true	NetworkPolicy	[delete deletecollection get list patch create update watch]
networksets		crd.projectcalico.org	true	NetworkSet	[delete deletecollection get list patch create update watch]
events	ev	events.k8s.io	true	Event	[create delete deletecollection get list patch create update watch]
ingresses	ing	extensions	true	Ingress	[create delete deletecollection get list patch update watch]
ingresses	ing	networking.k8s.io	true	Ingress	[create delete deletecollection get list patch update watch]
networkpolicies	netpol	networking.k8s.io	true	NetworkPolicy	[create delete deletecollection get list patch update watch]
runtimeclasses		node.k8s.io	false	RuntimeClass	[create delete deletecollection get list patch update watch]
poddisruptionbudgets	pdb	policy	true	PodDisruptionBudget	[create delete deletecollection get list patch update watch]
podsecuritypolicies	psp	policy	false	PodSecurityPolicy	[create delete deletecollection get list patch update watch]
clusterrolebindings		rbac.authorization.k8s.io	false	ClusterRoleBinding	[create delete deletecollection get list patch update watch]
clusterroles		rbac.authorization.k8s.io	false	ClusterRole	[create delete deletecollection get list patch update watch]
rolebindings		rbac.authorization.k8s.io	true	RoleBinding	[create delete deletecollection get list patch update watch]
roles		rbac.authorization.k8s.io	true	Role	[create delete deletecollection get list patch update watch]
priorityclasses	pc	scheduling.k8s.io	false	PriorityClass	[create delete deletecollection get list patch update watch]
csidrivers		storage.k8s.io	false	CSI Driver	[create delete deletecollection get list patch update watch]
csinodes		storage.k8s.io	false	CSI Node	[create delete deletecollection get list patch update watch]
storageclasses	sc	storage.k8s.io	false	StorageClass	[create delete deletecollection get list patch update watch]
volumeattachments		storage.k8s.io	false	VolumeAttachment	[create delete deletecollection get list patch update watch]

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Security

API OVERVIEW

Overview

WORKLOADS APIs

Container v1 core

CronJob v1beta1 batch

DaemonSet v1 apps

Deployment v1 apps

Job v1 batch

Pod v1 core

ReplicaSet v1 apps

ReplicationController v1 core

StatefulSet v1 apps

SERVICE APIs

Endpoints v1 core

EndpointSlice v1alpha1 discovery.k8s.io

Ingress v1beta1 networking.k8s.io

Service v1 core

CONFIG AND STORAGE APIs

ConfigMap v1 core

CSI Driver v1beta1 storage.k8s.io

CSINode v1beta1 storage.k8s.io

Secret v1 core

PersistentVolumeClaim v1 core

StorageClass v1 storage.k8s.io

Volume v1 core

VolumeAttachment v1 storage.k8s.io

Resource Categories

This is a high-level overview of the basic types of resources provided by the Kubernetes API and their primary functions.

Workloads are objects you use to manage and run your containers on the cluster.

Discovery & LB resources are objects you use to "stitch" your workloads together into an externally accessible, load-balanced Service.

Config & Storage resources are objects you use to inject initialization data into your applications, and to persist data that is external to your container.

Cluster resources define how the cluster itself is configured; these are typically used only by cluster operators.

Metadata resources are objects you use to configure the behavior of other resources within the cluster, such as `HorizontalPodAutoscaler` for scaling workloads.

Resource Objects

Resource objects typically have 3 components:

- **Resource ObjectMeta:** This is metadata about the resource, such as its name, type, api version, annotations, and labels. This contains fields that may be updated both by the end user and the system (e.g. annotations).
- **ResourceSpec:** This is defined by the user and describes the desired state of the system. Fill this in when creating or updating an object.
- **ResourceStatus:** This is filled in by the server and reports the current state of the system. In most cases, users don't need to change this.

Ref: <https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.16/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Security

- Access Control Policy
 - **RBAC Operate**
 - RoleBinding and ClusterRoleBinding:
 - Binding user accounts/system accounts to role or clusterrole

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: RoleBinding
3 metadata:
4   name: rolebindinglabsecurity
5   namespace: security
6 subjects:
7 - kind: User
8   name: labreadonly
9   apiGroup: rbac.authorization.k8s.io
10 roleRef:
11   kind: Role
12   name: rolereadonly
13   apiGroup: rbac.authorization.k8s.io
```

```
1  apiVersion: v1
2  kind: ServiceAccount
3  metadata:
4    name: admin-user
5    namespace: kubernetes-dashboard
6
7  ---
8
9  apiVersion: rbac.authorization.k8s.io/v1
10 kind: ClusterRoleBinding
11 metadata:
12   name: admin-user
13 roleRef:
14   apiGroup: rbac.authorization.k8s.io
15   kind: ClusterRole
16   name: cluster-admin
17 subjects:
18 - kind: ServiceAccount
19   name: admin-user
20   namespace: kubernetes-dashboard
```



Security

- Access Control Policy
 - **RBAC Operate**
 - Referring to Resource
 - Identification "subresource" under main resource
 - Identification "specific resource" under main resource

```
WorkShop_2.2_Security > ! security-role-subresource.yml
1  apiVersion: rbac.authorization.k8s.io/v1
2  kind: Role
3  metadata:
4    namespace: security
5    name: templaterolesubresource
6  rules:
7    - apiGroups: []
8      resources: ["pods/log"]
9      verbs: ["get"]
10     - apiGroups: []
11       resources: ["configmaps"]
12       resourceName: ["testconfigmap"]
13       verbs: ["update", "get"]
```

Security

- Access Control Policy:
 - **RBAC Operate**
 - Aggregated ClusterRoles
 - Combine of multiple cluster roles
 - Match by label and rule will automatic filled

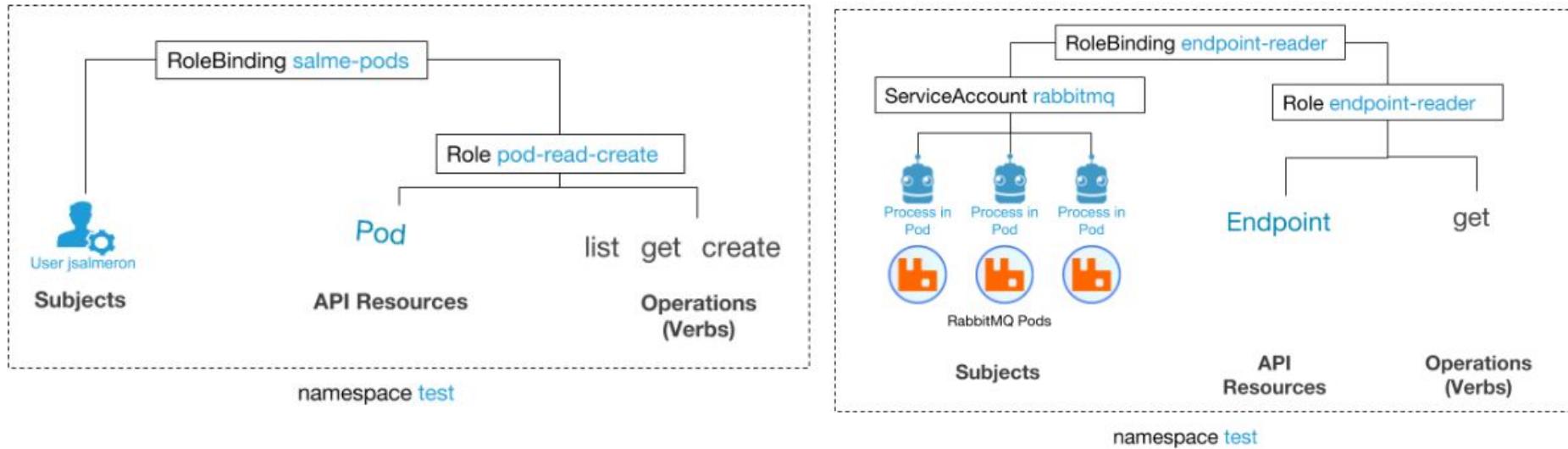
```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   name: templateaggregateclusterrole
5   namespace: security
6 aggregationRule:
7   clusterRoleSelectors:
8     - matchLabels:
9       rbac.kubernetestthailand.com/security: "true"
10  rules: [] # Rules are automatically filled in by the controller
```

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   name: templateclusterrole
5   labels:
6     |   rbac.kubernetestthailand.com/security: "true"
7   rules:
8     - apiGroups: [""] # "" indicates the core API group
9       resources: ["pods"]
10      verbs: ["get", "watch", "list"]
11     - apiGroups: ["apps"]
12       resources: ["deployments"]
13      verbs: ["get", "list"]
14     - apiGroups: ["batch"]
15       resources: ["jobs"]
16      verbs: ["get", "list"]
17     - apiGroups: [""]
18       resources: ["nodes"]
19       verbs: ["get", "list", "watch"]
```



Workshop: Security

- Part: Access Control Policy



```
[ubuntu@ip-10-0-1-78:~$ kubectl --context=labreadonly-context get pods
NAME      READY  STATUS    RESTARTS   AGE
webtest   1/1    Running   0          56m
[ubuntu@ip-10-0-1-78:~$ kubectl --context=labreadonly-context delete pods/webtest
Error from server (Forbidden): pods "webtest" is forbidden: User "labreadonly" cannot delete resource "pods" in API group "" in the namespace "security"
[ubuntu@ip-10-0-1-78:~$ kubectl --context=labreadonly-context run webtest --image=labdocker/nginx:http2 --port=443
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
Error from server (Forbidden): deployments.apps is forbidden: User "labreadonly" cannot create resource "deployments" in API group "apps" in the namespace "security"
[ubuntu@ip-10-0-1-78:~$ kubectl --context=labreadonly-context run webtest --image=labdocker/nginx:http2 --port=443 -n=security
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
Error from server (Forbidden): deployments.apps is forbidden: User "labreadonly" cannot create resource "deployments" in API group "apps" in the namespace "security"
[ubuntu@ip-10-0-1-78:~$ kubectl --context=labreadonly-context ls svc
```



Security

- Pod Security Policy:
 - Target for enhance security on pods
 - Privilege to run the pods
 - Privilege of pods to run in cluster
 - When apply “Pod Security Policy” on any namespace. Any unauthorized user cannot create pods on namespace
 - Kubernetes can operate to control pods running inside farm
 - Running of privileged containers
 - Usage of host namespaces
 - Usage of host networking and ports
 - Usage of volume types
 - Usage of the host filesystem
 - Restricting escalation to root privileges
 - The user and group IDs of the container
 - AppArmor or seccomp or sysctl profile used by containers



Security

- Pod Security Policy:

Enabling Pod Security Policies

Pod security policy control is implemented as an optional (but recommended) admission controller. PodSecurityPolicies are enforced by enabling the admission controller, but doing so without authorizing any policies will prevent any pods from being created in the cluster.

Since the pod security policy API (`policy/v1beta1/podsecuritypolicy`) is enabled independently of the admission controller, for existing clusters it is recommended that policies are added and authorized before enabling the admission controller.

APIServer flags

For details, see the reference documentation for `kube-apiserver`.

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: v1.16.0
apiServer:
  extraArgs:
    advertise-address: 192.168.0.103
    anonymous-auth: "false"
    enable-admission-plugins: AlwaysPullImages,DefaultStorageClass
    audit-log-path: /home/johndoe/audit.log
```



Security

- Security Policy:

Control Aspect	Field Names
Running of privileged containers	privileged
Usage of host namespaces	hostPID , hostIPC
Usage of host networking and ports	hostNetwork , hostPorts
Usage of volume types	volumes 
Usage of the host filesystem	allowedHostPaths <div style="border: 1px solid black; padding: 2px;">/var, /tmp, /home, /etc/docker etc..</div>
White list of FlexVolume drivers	allowedFlexVolumes
Allocating an FSGroup that owns the pod's volumes	fsGroup <div style="border: 1px solid black; padding: 2px;">MustRunAs, MustRunAsNonRoot, RunAsAny</div>
Requiring the use of a read only root file system	readOnlyRootFilesystem
The user and group IDs of the container	runAsUser , runAsGroup , supplementalGroups
Restricting escalation to root privileges	allowPrivilegeEscalation , defaultAllowPrivilegeEscalation
Linux capabilities	defaultAddCapabilities , requiredDropCapabilities , allowedCapabilities
The SELinux context of the container	selinux <div style="border: 1px solid black; padding: 2px;">MustRunAs, RunAsAny</div>
The Allowed Proc Mount types for the container	allowedProcMountTypes
The AppArmor profile used by containers	annotations
The seccomp profile used by containers	annotations
The sysctl profile used by containers	forbiddenSysctls , allowedUnsafeSysctls

The **recommended minimum set** of allowed volumes for new PSPs are:

- configMap
- downwardAPI
- emptyDir
- persistentVolumeClaim
- secret
- projected



Security

• Security Policy:

CAPABILITIES(7)

Linux Programmer's Manual

CAPABILITIES(7)

NAME top

capabilities - overview of Linux capabilities

DESCRIPTION top

For the purpose of performing permission checks, traditional UNIX implementations distinguish two categories of processes: *privileged* processes (whose effective user ID is 0, referred to as superuser or root), and *unprivileged* processes (whose effective UID is nonzero). Privileged processes bypass all kernel permission checks, while unprivileged processes are subject to full permission checking based on the process's credentials (usually: effective UID, effective GID, and supplementary group list).

Starting with kernel 2.2, Linux divides the privileges traditionally associated with superuser into distinct units, known as *capabilities*, which can be independently enabled and disabled. Capabilities are a per-thread attribute.

Capabilities list

The following list shows the capabilities implemented on Linux, and the operations or behaviors that each capability permits:

CAP_AUDIT_CONTROL (since Linux 2.6.11)

Enable and disable kernel auditing; change auditing filter rules; retrieve auditing status and filtering rules.

CAP_AUDIT_READ (since Linux 3.16)

Allow reading the audit log via a multicast netlink socket.

CAP_AUDIT_WRITE (since Linux 2.6.11)

Write records to kernel auditing log.

CAP_BLOCK_SUSPEND (since Linux 3.5)

Employ features that can block system suspend (`epoll(7)` `EPOLLWAKEUP`, `/proc/sys/wake_lock`).

CAP_CHOWN

Make arbitrary changes to file UIDs and GIDs (see `chown(2)`).

CAP_DAC_OVERRIDE

Bypass file read, write, and execute permission checks. (DAC is an abbreviation of "discretionary access control".)

CAP_DAC_READ_SEARCH

- * Bypass file read permission checks and directory read and execute permission checks;
- * invoke `open_by_handle_at(2)`;
- * use the `linkat(2)` `AT_EMPTY_PATH` flag to create a link to a file referred to by a file descriptor.

CAP_FOWNER

- * Bypass permission checks on operations that normally require the filesystem UID of the process to match the UID of the file (e.g., `chmod(2)`, `utime(2)`), excluding those operations covered by `CAP_DAC_OVERRIDE` and `CAP_DAC_READ_SEARCH`;
- * set inode flags (see `ioctl_iflags(2)`) on arbitrary files;
- * set Access Control Lists (ACLs) on arbitrary files;
- * ignore directory sticky bit on file deletion;
- * modify `user` extended attributes on sticky directory owned by any user;
- * specify `O_NOATIME` for arbitrary files in `open(2)` and `fcntl(2)`.

CAP_FSETID

- * Don't clear set-user-ID and set-group-ID mode bits when a file is modified;
- * set the set-group-ID bit for a file whose GID does not match the filesystem or any of the supplementary GIDs of the calling process.

CAP_IPC_LOCK

Lock memory (`mlock(2)`, `mlockall(2)`, `mmmap(2)`, `shmctl(2)`).

CAP_IPC_OWNER

Bypass permission checks for operations on System V IPC objects.

CAP_KILL

Bypass permission checks for sending signals (see `kill(2)`). This includes use of the `ioctl(2)` `KDSIGACCEPT` operation.

CAPLEASE (since Linux 2.4)

Establish leases on arbitrary files (see `fcntl(2)`).

CAP_LINUX_IMMUTABLE

Set the `FS_APPEND_FL` and `FS_IMMUTABLE_FL` inode flags (see `ioctl_iflags(2)`).

CAP_MAC_ADMIN (since Linux 2.6.25)

Allow MAC configuration or state changes. Implemented for the Smack Linux Security Module (LSM).

CAP_MAC_OVERRIDE (since Linux 2.6.25)

Override Mandatory Access Control (MAC). Implemented for the Smack LSM.

CAP_MKNOD (since Linux 2.4)

Create special files using `mknod(2)`.

Ref: <http://man7.org/linux/man-pages/man7/capabilities.7.html>

Kubernetes: Production Workload Orchestration



Security

- Security Policy:

```
1  apiVersion: policy/v1beta1
2  kind: PodSecurityPolicy
3  metadata:
4    name: restricted
5    namespace: security
6    annotations:
7      apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default,localhost/restrict-apparmor'
8      apparmor.security.beta.kubernetes.io/defaultProfileName: 'localhost/restrict-apparmor'
9  spec:
10   privileged: false
11   allowPrivilegeEscalation: false
12   requiredDropCapabilities:
13     - ALL
14   volumes:
15     - 'configMap'
16     - 'emptyDir'
17     - 'projected'
18     - 'secret'
19     - 'downwardAPI'
20     - 'persistentVolumeClaim'
21   hostNetwork: false
22   hostIPC: false
23   hostPID: false
24   runAsUser:
25     rule: 'MustRunAsNonRoot'
26   seLinux:
27     rule: 'RunAsAny'
28   supplementalGroups:
29     rule: 'MustRunAs'
30     ranges:
31       - min: 1
32         max: 65535
33   fsGroup:
34     rule: 'MustRunAs'
35     ranges:
36       - min: 1
37         max: 65535
38   hostPorts:
39     - min: 0
40       max: 0
41   readOnlyRootFilesystem: false
```



Security

- Security Policy:

```
ubuntu@ip-10-0-1-78:~/kubernetes_201911/WorkShop_2.2_Security$ kubectl apply -f ./security-policy-restrict.yml
podsecuritypolicy.policy/restricted created
ubuntu@ip-10-0-1-78:~/kubernetes_201911/WorkShop_2.2_Security$ kubectl get psp -n=security
NAME          PRIV   CAPS  SELINUX  RUNASUSER    FSGROUP     SUPGROUP    READONLYROOTFS   VOLUMES
restricted    false   RunAsAny  MustRunAsNonRoot  MustRunAs  MustRunAs  false        configMap,emptyDir,projected,secret,downwardAPI,persistentVolumeClaim
ubuntu@ip-10-0-1-78:~/kubernetes_201911/WorkShop_2.2_Security$ kubectl describe psp/restricted -n=security
Name:           restricted
Namespace:      <none>
Labels:         <none>
Annotations:   apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/default,localhost/restrict-apparmor
                apparmor.security.beta.kubernetes.io/defaultProfileName: localhost/restrict-apparmor
                kubctl.kubernetes.io/last-applied-configuration:
                  {"apiVersion":"policy/v1beta1","kind":"PodSecurityPolicy","metadata":{"annotations":{"apparmor.security.beta.kubernetes.io/allowedProfileN...
API Version:  policy/v1beta1
Kind:         PodSecurityPolicy
Metadata:
  Creation Timestamp: 2019-10-10T14:56:07Z
  Resource Version:  80452
  Self Link:        /apis/policy/v1beta1/podsecuritypolicies/restricted
  UID:              fe873603-6ca6-458c-be84-173a45025d8b
Spec:
  Allow Privilege Escalation:  false
  Fs Group:
    Ranges:
      Max: 65535
      Min: 1
    Rule: MustRunAs
  Host Ports:
    Max: 0
    Min: 0
  Required Drop Capabilities:
    ALL
  Run As User:
    Rule: MustRunAsNonRoot
  Se Linux:
    Rule: RunAsAny
  Supplemental Groups:
    Ranges:
      Max: 65535
      Min: 1
    Rule: MustRunAs
  Volumes:
    configMap
    emptyDir
    projected
    secret
    downwardAPI
    persistentVolumeClaim
  Events:  <none>
```



Security

- Security Policy:

```
ubuntu@ip-10-0-1-78:~/kubernetes_201911/WorkShop_2.2_Security$ kubectl apply -f ./security-policy-restrict.yml
podsecuritypolicy.policy/restricted created
ubuntu@ip-10-0-1-78:~/kubernetes_201911/WorkShop_2.2_Security$ kubectl get psp -n=security
NAME          PRIV   CAPS  SELINUX  RUNASUSER    FSGROUP  SUPGROUP  READONLYROOTFS  VOLUMES
restricted    false   RunAsAny  MustRunAsNonRoot  MustRunAs  MustRunAs  false      configMap,emptyDir,projected,secret,downwardAPI,persistentVolumeClaim
ubuntu@ip-10-0-1-78:~/kubernetes_201911/WorkShop_2.2_Security$ kubectl describe psp/restricted -n=security

example $ kubectl get pods
NAME          READY   STATUS
mymariadb-2389847989-8qrbs  0/1    STATUS
container has runAsNonRoot and image will run as root
                                         RESTARTS   AGE
0                                     33s
example $
```

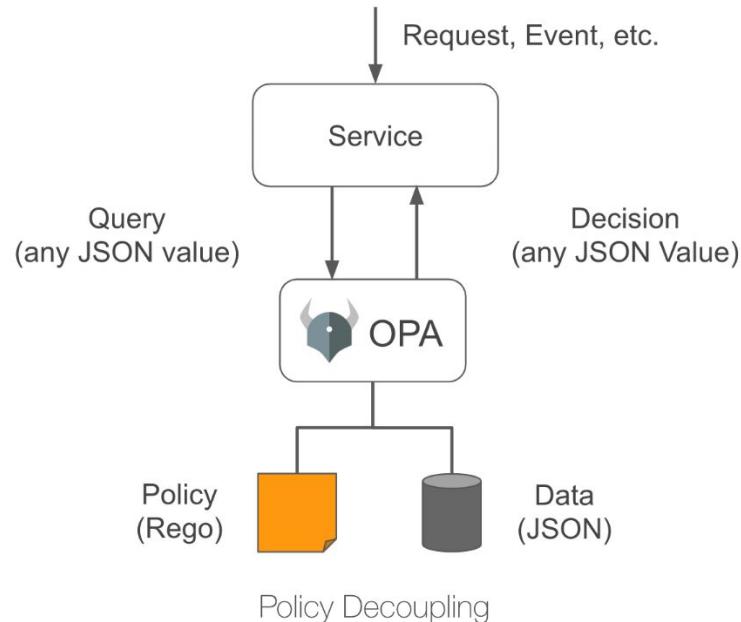


Security

The screenshot shows the official OPA website. At the top is a navigation bar with a logo of a bull's head, the text "Open Policy Agent", a version dropdown set to "v0.14.2 latest", and a search icon. Below the header is a sidebar with "Introduction" and "Overview" sections, followed by a list of topics: Philosophy, Policy Language, Policy Performance, Policy Testing, Policy Reference, Policy Cheatsheet, External Data, Integrating OPA, Extending OPA, and REST API.

Overview

OPA [decouples](#) policy decision-making from policy enforcement. When your software needs to make policy decisions it [queries](#) OPA and supplies structured data (e.g., JSON) as input. OPA accepts arbitrary structured data as input.



Kubernetes in real world



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Kubernetes have a lot of component need to install / configuration on real world
- Normally kubernetes need some 3rd tool for deployment
 - Ubuntu(MAAS, JUiu)
 - CoreOS
 - Fedora (Ansible)
- provide new solution for create cluster system with kubernetes orchestrator

kubeadm <init>

- Option:
 - --apiserver-advertise-address= x.x.x.x (Default: First Interface)
 - --apiserver-bind-port = xxx (Default: 6443)
 - --kubernetes-version = xxx (Default: latest)
 - --pod-network-cidr = x.x.x.x (CNI), Need for Flennel
 - --token = xxxx (Default: auto gen)
 - --skip-preflight-checks
 - etc



Kubernetes in real world

- Kubernetes have a lot of component need to install / configuration on real world
- Normally kubernetes need some 3rd tool for deployment
 - Ubuntu(MAAS, JUiu)
 - CoreOS
 - Fedora (Ansible)
- Since kubernetes 1.6 and later. Cluster system can make with command “kubeadm”
- Prerequisite
 - Docker Engine
 - Kubernetes Engine (Kubelet, Kubectl, Kubeadm)
- Role of the cluster system
 - Master:
 - Control all activity on cluster system (Pods, Services, Job, CronJob, RS, RC etc)
 - Node:
 - Worker for running as Master's order



Kubernetes in real world

- Create cluster from bare metal
- Step for setup
 - Phase 1: Install prerequisite component
 - Docker Engine
 - Kubelet Engine
 - Kubeadm Engine
 - Phase 2: Initialize Master node
 - `kubeadm init <option>`
 - Phase 3: Install Pods Network (3rd party) with CNI support
 - Flannel (Support Cross-Platform)
 - Weave Net (Support Cross-Platform)
 - Calico
 - Canal (Flannel + Calico)
 - Contiv
 - Romana
 - Kube-Router
 - Etc
 - Phase 4: Join node to cluster system
 - `kubeadm join <option>`



Kubernetes in real world

kubeadm maturity

kubeadm's overall feature state is **GA**. Some sub-features, like the configuration file API are still under active development. The implementation of creating the cluster may change slightly as the tool evolves, but the overall implementation should be pretty stable. Any commands under `kubeadm alpha` are by definition, supported on an alpha level.

Support timeframes

Kubernetes releases are generally supported for nine months, and during that period a patch release may be issued from the release branch if a severe bug or security issue is found. Here are the latest Kubernetes releases and the support timeframe; which also applies to `kubeadm`.

Kubernetes version	Release month	End-of-life-month
v1.13.x	December 2018	September 2019
v1.14.x	March 2019	December 2019
v1.15.x	June 2019	March 2020
v1.16.x	September 2019	June 2020

Ref: <https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

The image shows a web browser interface with two main pages. On the left, the Kubernetes landing page features a large blue hexagonal logo with a white steering wheel icon. The page has a "Welcome!" header, a "Before starting we need to verify you are a human" message, and a "Create session" button. The main title is "kubernetes by Google™". On the right, a session details page for a node named "d7498ce6_node1" is displayed. It includes a clock showing "03:52:40", an orange "CLOSE SESSION" button, and a "Instances" section with a "DELETE" button. The session details table shows IP: 10.0.12.3, Memory: 27.21% (1.087GiB / 3.996GiB), and CPU: 27.32%. Below this, a terminal window shows command-line output for Kubernetes commands like `kubectl get svc` and `kubectl get pods`.

host2.labs.play-with-k8s.com/p/d7498ce6-f563-4e73-b7e7-62503f1db77b#d7498ce6_node1

NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	10.96.0.1	<none>	443/TCP	4m

```
[node1 /]$ kubectl get pods
No resources found.
[node1 /]$ 
```

Kubernetes: Production Workload Orchestration

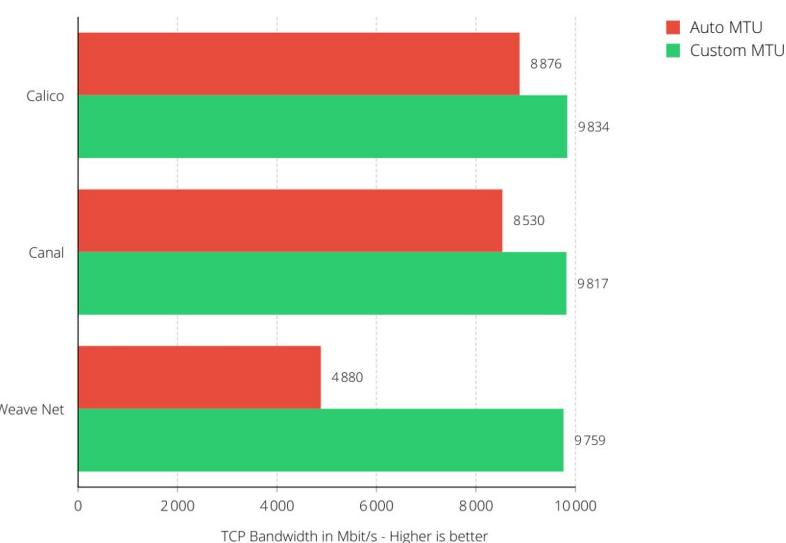


kubernetes
by Google

Kubernetes in real world

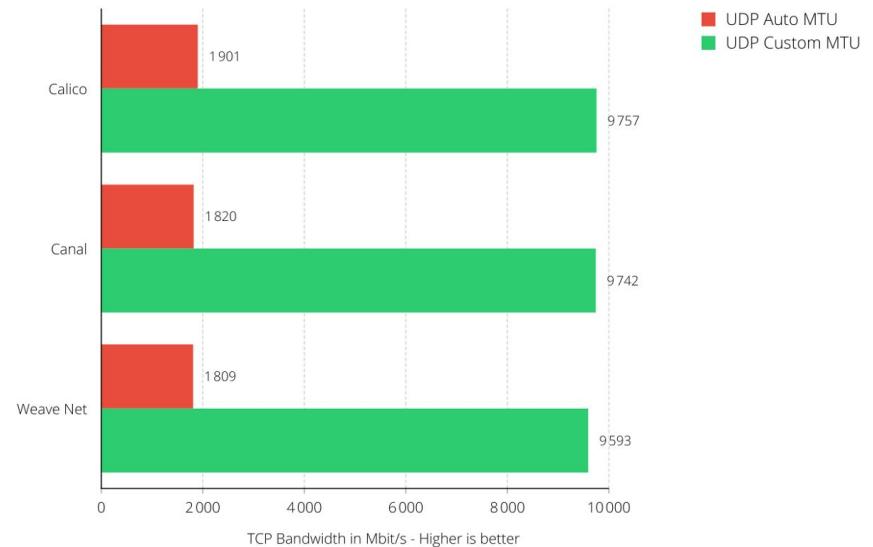
- Network Optional

K8S CNI Benchmark - MTU impact - TCP



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

K8S CNI Benchmark - MTU impact - UDP



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

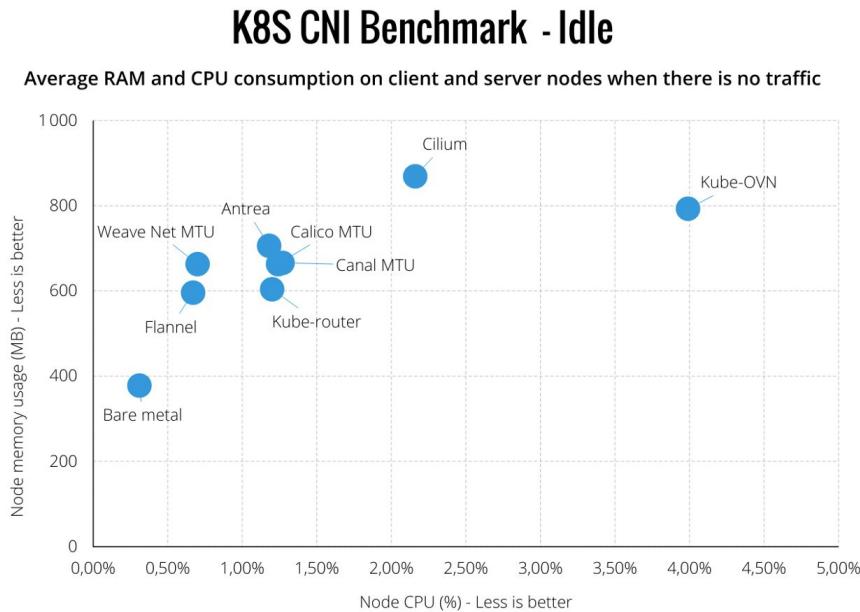
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Network Optional



2020-08-27 - Alexis Ducastel - infrabuilder - Benchmark tool : knb (<https://github.com/InfraBuilder/k8s-bench-suite>)



<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

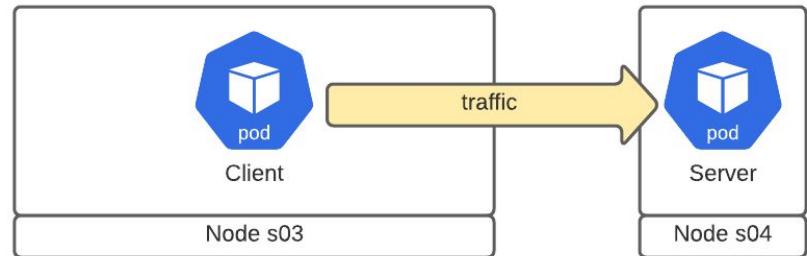
Kubernetes: Production Workload Orchestration



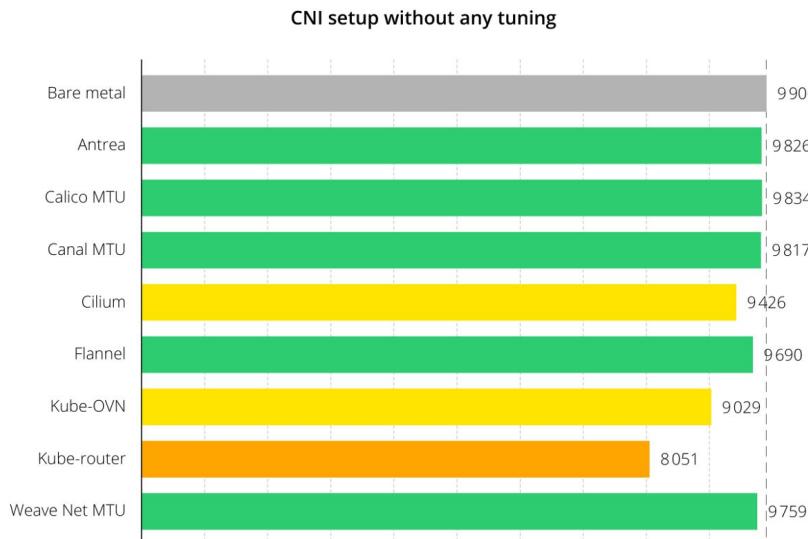
kubernetes
by Google

Kubernetes in real world

- Network Optional

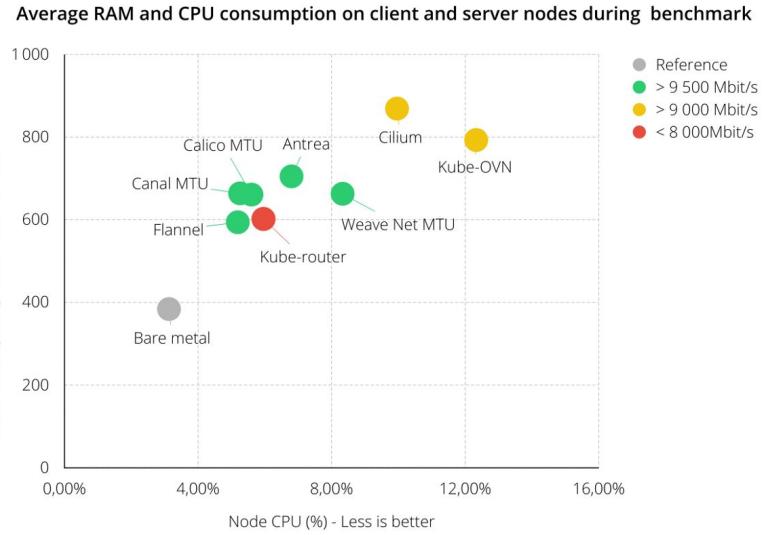


K8S CNI Benchmark - Pod to Pod - TCP - Bandwidth



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

K8S CNI Benchmark - Pod to Pod - TCP - Resources



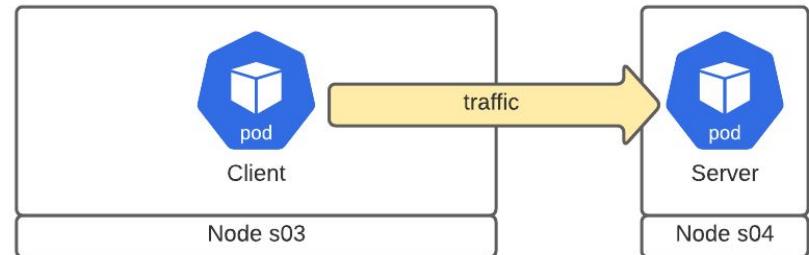
2020-08-27 - Alexis Ducastel - infrabuilder - Benchmark tool : knb (<https://github.com/InfraBuilder/k8s-bench-suite>)

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

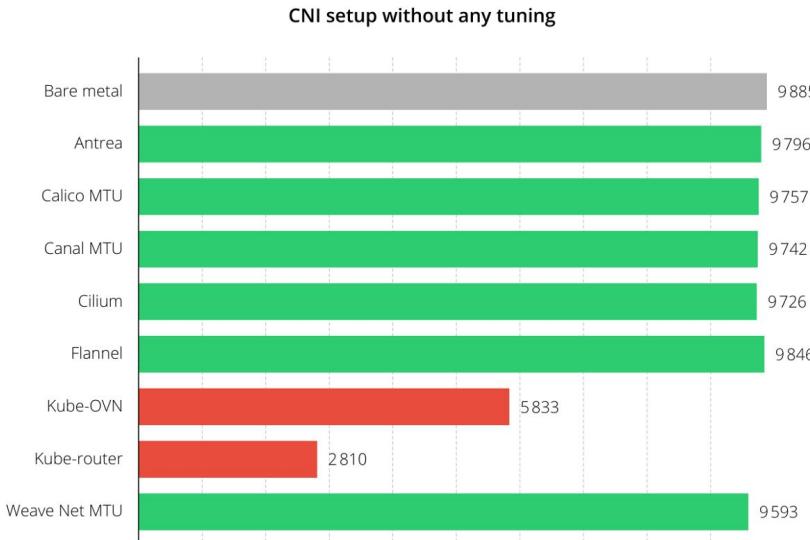


Kubernetes in real world

- Network Optional

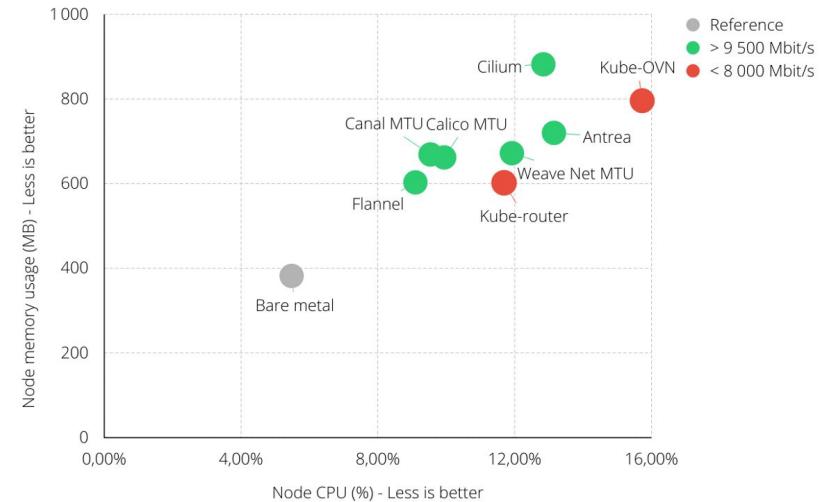


K8S CNI Benchmark - Pod to Pod - UDP - Bandwidth



K8S CNI Benchmark - Pod to Pod - UDP - Resources

Average RAM and CPU consumption on client and server nodes during benchmark



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

2020-08-27 - Alexis Ducastel - infrabuilder - Benchmark tool : knb (<https://github.com/InfraBuilder/k8s-bench-suite>)

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

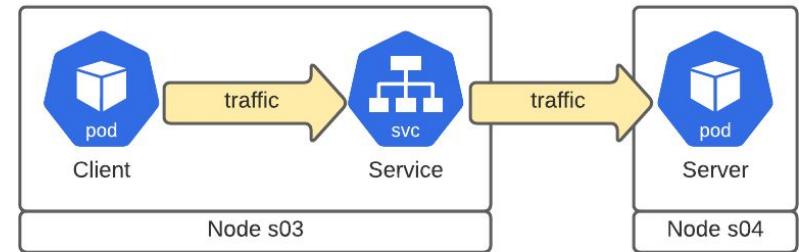
Kubernetes: Production Workload Orchestration



kubernetes
by Google

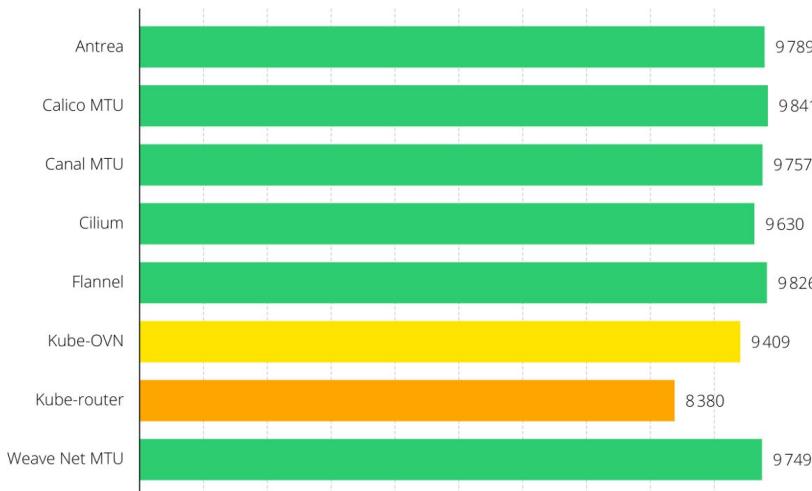
Kubernetes in real world

- Network Optional



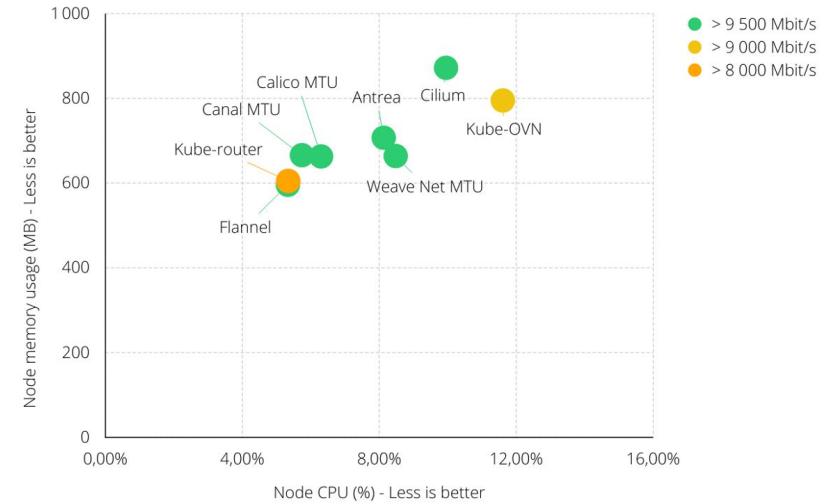
K8S CNI Benchmark - Pod to Service - TCP - Bandwidth

CNI setup without any tuning



K8S CNI Benchmark - Pod to Service - TCP - Resources

Average RAM and CPU consumption on client and server nodes during benchmark



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

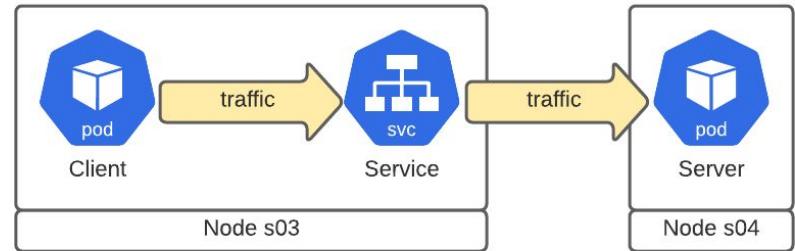
2020-08-27 - Alexis Ducastel - infrabuilder - Benchmark tool : knb (<https://github.com/InfraBuilder/k8s-bench-suite>)

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

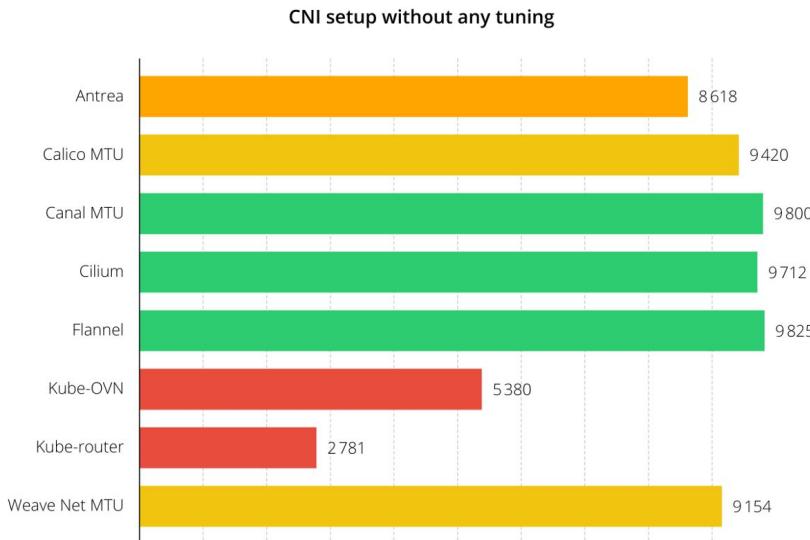


Kubernetes in real world

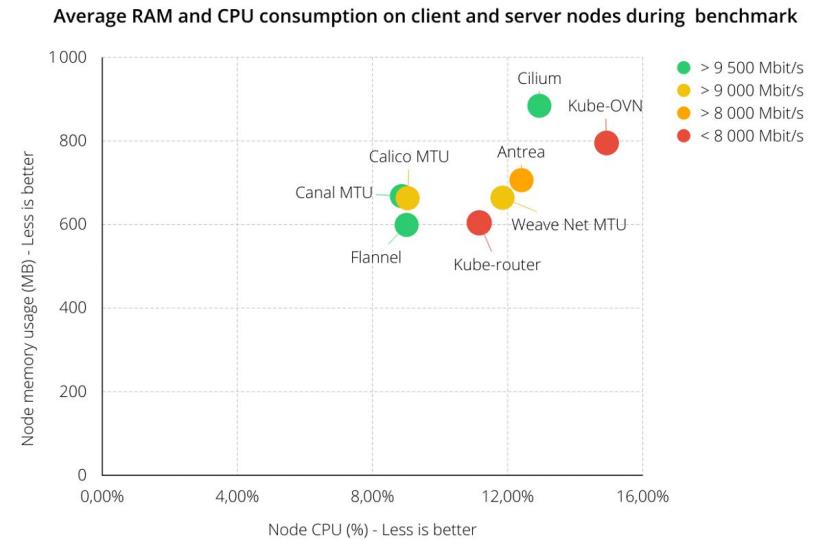
- Network Optional



K8S CNI Benchmark - Pod to Service - UDP - Bandwidth



K8S CNI Benchmark - Pod to Service - UDP - Resources



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

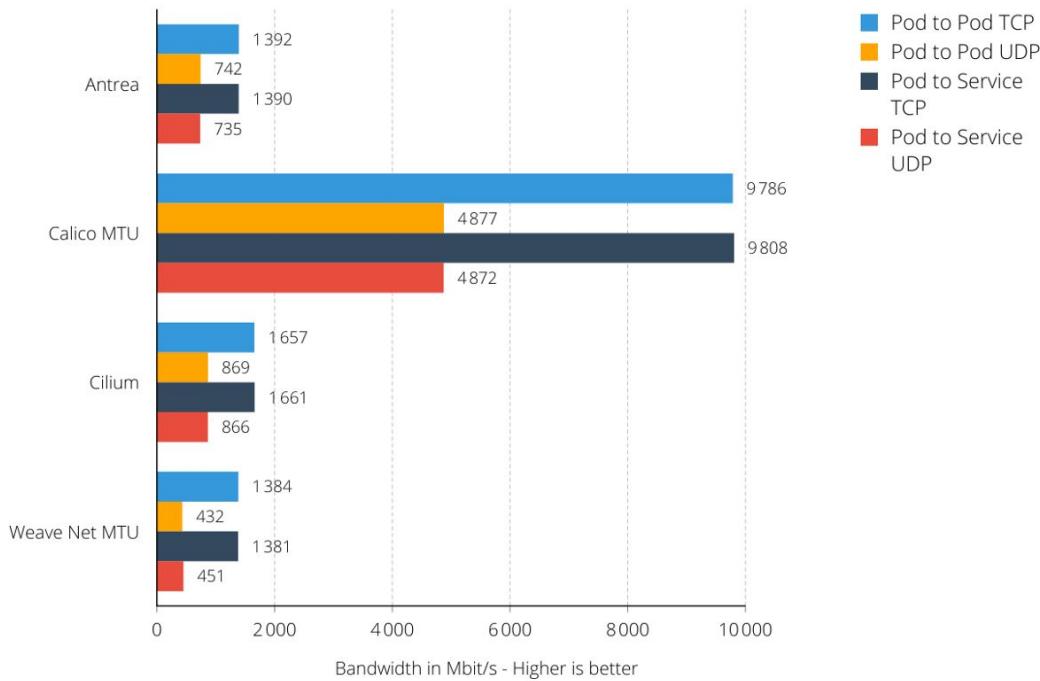
<https://1tnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-1ugbit-s-network-updated-august-2020-6e1b757b9e49>



Kubernetes in real world

- Network Optional

K8S CNI Benchmark - Encrypted CNIs



2020-08-27 - Alexis Ducastel - infrabuilder - Source : <https://github.com/InfraBuilder/benchmark-k8s-cni-2020-08>

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

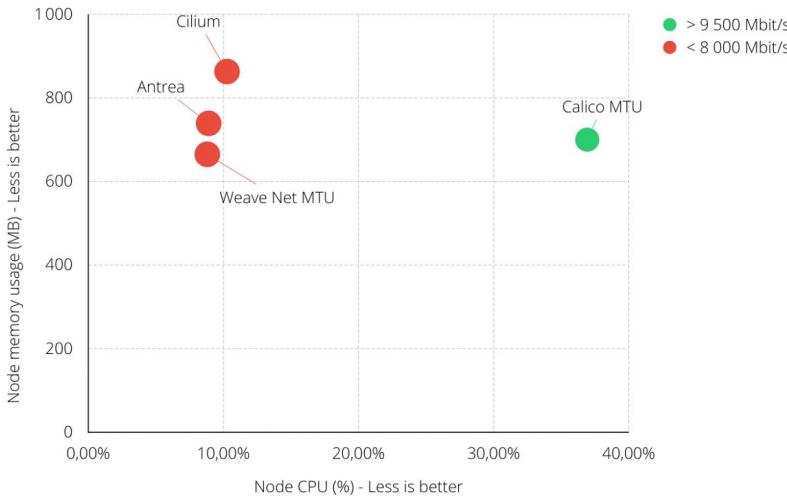


Kubernetes in real world

- Network Optional

K8S CNI Benchmark - Encrypted - P2P - TCP - Resources

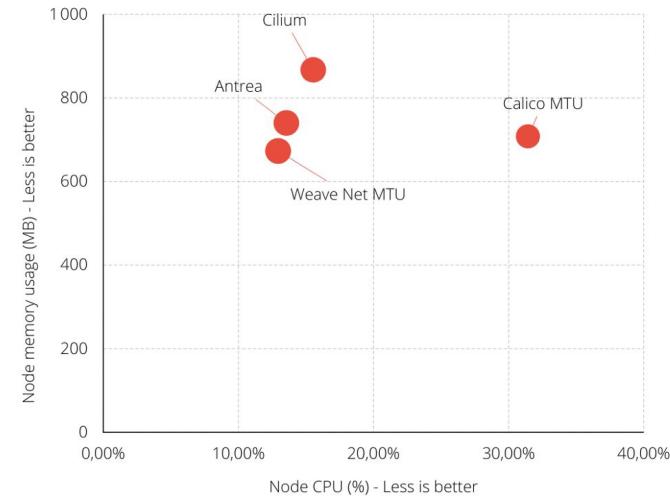
Average RAM and CPU consumption on client and server nodes during benchmark



2020-08-27 - Alexis Ducastel - infrabuilder - Benchmark tool : knb (<https://github.com/InfraBuilder/k8s-bench-suite>)

K8S CNI Benchmark - Encrypted - P2P - UDP - Resources

Average RAM and CPU consumption on client and server nodes during benchmark



2020-08-27 - Alexis Ducastel - infrabuilder - Benchmark tool : knb (<https://github.com/InfraBuilder/k8s-bench-suite>)

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Network Optional

CNI Benchmark August 2020 infraBuilder	Config	Performances (bandwidth)				Resources consumption (cpu/ram)					Security features					
		MTU	Pod to Pod		Pod to Service		Idle	Pod to Pod		Pod to Service		Network Policies	Encryption			
			setting	TCP	UDP	TCP	UDP	none	TCP	UDP	TCP	UDP	in	out	activation	Performance
Antrea	auto	Very fast	Very fast	Very fast	Slow	Low	Low	Low	Low	Low	Low	Low	yes	yes	at deploy time	Slow
Calico	manual	Very fast	Very fast	Very fast	Fast	Low	Very low	Very low	Very low	Very low	Very low	Very low	yes	yes	anytime	Very fast
Canal	manual	Very fast	Very fast	Very fast	Very fast	Low	Very low	Very low	Very low	Very low	Very low	Very low	yes	yes	no	n/a
Cilium	auto	Fast	Very fast	Very fast	Very fast	High	High	High	High	High	High	High	yes	yes	at deploy time	Slow
Flannel	auto	Very fast	Very fast	Very fast	Very fast	Very low	Very low	Very low	Very low	Very low	Very low	Very low	no	no	no	n/a
Kube-OVN	auto	Fast	Very slow	Fast	Very slow	High	High	High	High	High	High	High	yes	yes	no	n/a
Kube-router	none	Slow	Very slow	Slow	Very slow	Low	Very low	Low	Very low	Low	Very low	Low	yes	yes	no	n/a
Weave Net	manual	Very fast	Very fast	Very fast	Fast	Very low	Low	Low	Low	Low	Low	Low	yes	yes	at deploy time	Slow

<https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49>



Kubernetes in real world

- Initial Master role

```
kubeadm <init>
```

- Option:
 - --apiserver-advertise-address= x.x.x.x (Default: First Interface)
 - --apiserver-bind-port = xxx (Default: 6443)
 - --kubernetes-version = xxx (Default: latest)
 - --pod-network-cidr = x.x.x.x (CNI), Need for Flennel
 - --token = xxxx (Default: auto gen)
 - --skip-preflight-checks
 - Etc

- Join Node in Cluster

```
kubeadm <init>
```

- Option:
 - --apiserver-advertise-address= x.x.x.x (Default: First Interface)
 - --apiserver-bind-port = xxx (Default: 6443)
 - --kubernetes-version

Kubernetes in real world

- Initial Master role by config file

```
kubeadm init --config <configuration file>
```

```
Untitled-2 • instruction.txt ! kubeadm-init.yaml x
1 apiVersion: kubeadm.k8s.io/v1beta1
2 bootstrapTokens:
3   - groups:
4     - system:bootstrappers:kubeadm:default-node-token
5       ttl: 24h0m0s
6     usages:
7       - signing
8       - authentication
9   kind: InitConfiguration
10  localAPIEndpoint:
11    bindPort: 6443
12  nodeRegistration:
13    criSocket: /var/run/docker.sock
14    name: hostnamemaster
15    kubeletExtraArgs:
16      cloud-provider: aws
17    taints:
18      - effect: NoSchedule
19        key: node-role.kubernetes.io/master
20 ---
```

```
20 ---
21   apiServer:
22     certSANs:
23       - 1.1.1.1
24     extraArgs:
25       | authorization-mode: Node,RBAC
26       | timeoutForControlPlane: 4m0s
27
28   apiVersion: kubeadm.k8s.io/v1beta1
29   certificatesDir: /etc/kubernetes/pki
30   clusterName: Kubernetes
31   controlPlaneEndpoint: ""
32   controllerManager:
33     extraArgs:
34       | cloud-provider: aws
35       | configure-cloud-routes: "false"
36       | address: 0.0.0.0
37   dns:
38     type: CoreDNS
39   etcd:
40     local:
41       | dataDir: /var/lib/etcd
42   imageRepository: k8s.gcr.io
43   kind: ClusterConfiguration
44   kubernetesVersion: v1.13.0
45   networking:
46     dnsDomain: cluster.local
47     podSubnet: 192.168.0.0/16
48     serviceSubnet: 10.96.0.0/12
49   scheduler:
50     extraArgs:
51       | address: 0.0.0.0
```

Ref: <https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Join Worker role by config file

```
kubeadm join --config <configuration file>
```

```
Untitled-2 ● instruction.txt kubeadm-join.yaml x
1 apiVersion: kubeadm.k8s.io/v1beta1
2 caCertPath: /etc/kubernetes/pki/ca.crt
3 discovery:
4   bootstrapToken:
5     token: tokenid
6     apiServerEndpoint: hostnamemaster:6443
7     caCertHashes:
8       - "sha256:cahash"
9     timeout: 5m0s
10    kind: JoinConfiguration
11    nodeRegistration:
12      name: hostnameworker
13      kubeletExtraArgs:
14        cloud-provider: aws
```

Ref: <https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Other parameter

```
apiVersion: kubeadm.k8s.io/v1beta1
kind: InitConfiguration
bootstrapTokens:
- token: "9a08jv.c0izixklcxtnmze7"
  description: "Kubeadm bootstrap token"
  ttl: "24h"
- token: "783bde.3f89s0fje9f38fhf"
  description: "another bootstrap token"
  usages:
  - authentication
  - signing
  groups:
  - system:bootstrappers:kubeadm:default-node-token
nodeRegistration:
  name: "ec2-10-100-0-1"
  criSocket: "/var/run/dockershim.sock"
  taints:
  - key: "kubeadmNode"
    value: "master"
    effect: "NoSchedule"
  kubeletExtraArgs:
    cgroupDriver: "cgroupfs"
localAPIEndpoint:
  advertiseAddress: "10.100.0.1"
  bindPort: 6443
---
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
etcd:
# one of local or external
local:
  imageRepository: "k8s.gcr.io"
  imageTag: "3.2.24"
  dataDir: "/var/lib/etcd"
  extraArgs:
    listen-client-urls: "http://10.100.0.1:2379"
  serverCertSANs:
  - "ec2-10-100-0-1.compute-1.amazonaws.com"
  peerCertSANs:
  - "10.100.0.1"
# external:
# endpoints:
# - "10.100.0.1:2379"
# - "10.100.0.2:2379"
# caFile: "/etc/kubernetes/pki/etcd/etcd-ca.crt"
# certfile: "/etc/kubernetes/pki/etcd/etcd.crt"
# keyFile: "/etc/kubernetes/pki/etcd/etcd.key"
networking:
  serviceSubnet: "10.96.0.0/12"
  podSubnet: "10.100.0.1/24"
  dnsDomain: "cluster.local"
  kubernetesVersion: "v1.12.0"
  controlPlaneEndpoint: "10.100.0.1:6443"
apiServer:
  extraArgs:
    authorization-mode: "Node,RBAC"
  extraVolumes:
  - name: "some-volume"
    hostPath: "/etc/some-path"
    mountPath: "/etc/some-pod-path"
    readOnly: false
    pathType: File
  certSANs:
  - "10.100.1.1"
  - "ec2-10-100-0-1.compute-1.amazonaws.com"
  timeoutForControlPlane: 4m0s
controllerManager:
  extraArgs:
    "node-cidr-mask-size": "20"
  extraVolumes:
  - name: "some-volume"
    hostPath: "/etc/some-path"
    mountPath: "/etc/some-pod-path"
    readOnly: false
    pathType: File
scheduler:
  extraArgs:
    address: "10.100.0.1"
  extraVolumes:
  - name: "some-volume"
    hostPath: "/etc/some-path"
    mountPath: "/etc/some-pod-path"
    readOnly: false
    pathType: File
  certificatesDir: "/etc/kubernetes/pki"
  imageRepository: "k8s.gcr.io"
  useHyperKubeImage: false
  clusterName: "example-cluster"
---
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
# kubelet specific options here
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
# kube-proxy specific options here
```

Ref: <https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Kubeadm support multiple cloud provider
 - AWS
 - Azure
 - CloudStack
 - GCE
 - OpenStack
 - OVirt
 - Proton
 - Vsphere
 - IBM Cloud Kubernetes Service
 - Baidu Cloud Container Engine
- Get fully benefit for utilize capability from cloud provider.
Ex: Storage dynamic provision, Network load balancer etc

Ref: <https://kubernetes.io/docs/concepts/cluster-administration/cloud-providers/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Phase 1: Install prerequisite component

```
1 #Install Base docker-engine
2 sudo apt-get remove docker docker-engine
3 sudo apt-get -y install \
4     apt-transport-https \
5     ca-certificates \
6     curl \
7     software-properties-common
8 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
9 sudo apt-key fingerprint 0EBFCD88
10 sudo add-apt-repository \
11     "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
12     $(lsb_release -cs) \
13     stable"
14 sudo apt-get update
15 sudo apt-get -y install docker-ce
16 sudo groupadd docker
17 sudo usermod -aG docker $USER
18 sudo systemctl enable docker
19
20 #Install Kubernetes Base
21 curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.7.0/bin/linux/amd64/kubectl
22 chmod +x ./kubectl
23 sudo mv ./kubectl /usr/local/bin/kubectl
24 sudo apt-get update && apt-get install -y apt-transport-https
25 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
26 sudo touch /etc/apt/sources.list.d/kubernetes.list
27 #sudo bash -c 'echo "deb http://apt.kubernetes.io/ kubernetes-xenial-1.7 main" >
28 /etc/apt/sources.list.d/kubernetes.list'
29 sudo bash -c 'echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >
30 /etc/apt/sources.list.d/kubernetes.list'
31 sudo apt-get update
32 sudo apt-get install -y kubelet=1.7.0-00 kubeadm=1.7.0-00
```



Kubernetes in real world

- Phase 2: Initialize Master node

```
praparn@kubernetes-ms:~$ sudo su -
root@kubernetes-ms:~# kubeadm init --kubernetes-version=v1.7.0 --pod-network-cidr=10.244.0.0/16 --token 8c2350.f55343444a6ffc46
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
[init] Using Kubernetes version: v1.7.0
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks
[preflight] WARNING: docker version is greater than the most recently validated version. Docker version: 17.06.0-ce. Max validated version: 1.12
[certificates] Generated CA certificate and key.
[certificates] Generated API server certificate and key.
[certificates] API Server serving cert is signed for DNS names [kubernetes-ms kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.99.200]
[certificates] Generated API server kubelet client certificate and key.
[certificates] Generated service account token signing key and public key.
[certificates] Generated front-proxy CA certificate and key.
[certificates] Generated front-proxy client certificate and key.
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[apiclient] Created API client, waiting for the control plane to become ready
```

```
Your Kubernetes master has initialized successfully!

To start using your cluster, you'll need to run (as a regular user):

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  http://kubernetes.io/docs/admin/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join --token 8c2350.f55343444a6ffc46 192.168.99.200:6443

root@kubernetes-ms:~#
```



Kubernetes in real world

- Phase 3: Install Pods Network (3rd party) with CNI support

```
[praparn@kubernetes-ms:~]$ kubectl apply -n kube-system -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
daemonset "weave-net" created
[praparn@kubernetes-ms:~]$ kubectl get pods --all-namespaces
NAMESPACE      NAME          READY   STATUS    RESTARTS   AGE
kube-system    etcd-kubernetes-ms   1/1     Running   0          2m
kube-system    kube-apiserver-kubernetes-ms   1/1     Running   0          2m
kube-system    kube-controller-manager-kubernetes-ms   1/1     Running   0          2m
kube-system    kube-dns-2425271678-32cjf   0/3     Pending   0          2m
kube-system    kube-proxy-9jtxj    1/1     Running   0          2m
kube-system    kube-scheduler-kubernetes-ms   1/1     Running   0          2m
kube-system    weave-net-bg346    0/2     ContainerCreating   0          15s
[praparn@kubernetes-ms:~]$
```

```
[praparn@kubernetes-ms:~]$ kubectl get pods --all-namespaces
NAMESPACE      NAME          READY   STATUS    RESTARTS   AGE
kube-system    etcd-kubernetes-ms   1/1     Running   0          2m
kube-system    kube-apiserver-kubernetes-ms   1/1     Running   0          2m
kube-system    kube-controller-manager-kubernetes-ms   1/1     Running   0          2m
kube-system    kube-dns-2425271678-32cjf   3/3     Running   0          2m
kube-system    kube-proxy-9jtxj    1/1     Running   0          2m
kube-system    kube-scheduler-kubernetes-ms   1/1     Running   0          2m
kube-system    weave-net-bg346    2/2     Running   0          51s
[praparn@kubernetes-ms:~]$
```



Kubernetes in real world

- Phase 4: Join node to cluster system

```
praparn@kubernetes-1:~$ sudo su -
root@kubernetes-1:~# kubeadm --token 8c2350.f55343444a6ffc46 join 192.168.99.200:6443
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
[preflight] Running pre-flight checks
[preflight] WARNING: docker version is greater than the most recently validated version. Docker version: 17.06.0-ce. Max validated version: 1.12
[discovery] Trying to connect to API Server "192.168.99.200:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.99.200:6443"
[discovery] Cluster info signature and contents are valid, will use API Server "https://192.168.99.200:6443"
[discovery] Successfully established connection with API Server "192.168.99.200:6443"
[bootstrap] Detected server version: v1.7.0
[bootstrap] The server supports the Certificates API (certificates.k8s.io/v1beta1)
[csr] Created API client to obtain unique certificate for this node, generating keys and certificate signing request
[csr] Received signed certificate from the API server, generating KubeConfig...
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"

Node join complete:
* Certificate signing request sent to master and response received.
* Kubelet informed of new secure connection details.

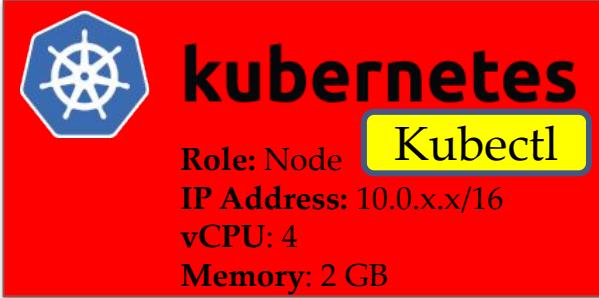
Run 'kubectl get nodes' on the master to see this machine join.
root@kubernetes-1:~#
```

```
praparn@kubernetes-ms:~$ kubectl get node
NAME      STATUS    AGE     VERSION
kubernetes-1  Ready    5m      v1.7.0
kubernetes-2  Ready    4m      v1.7.0
kubernetes-ms  Ready   11m      v1.7.0
praparn@kubernetes-ms:~$
```

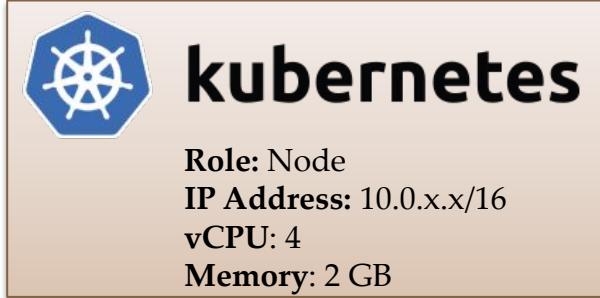


Workshop: Kubernetes Real World

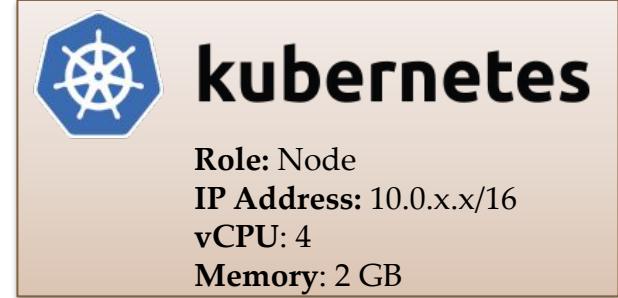
Machine: Kubernetes_MS



Machine: Kubernetes_1



Machine: Kubernetes_2



AWS CLI
Command/Monitor



Role: Client



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Multiple Master

The screenshot shows the official Kubernetes documentation website. The top navigation bar includes links for Documentation, Blog, Partners, Community, Case Studies, English, and v1.16. The main content area is titled "Creating Highly Available clusters with kubeadm". It explains two approaches: stacked control plane nodes and an external etcd cluster. A "Caution" box notes that neither approach works with Service objects of type LoadBalancer or dynamic PersistentVolumes in a cloud environment. Below the main content, a sidebar lists other "Getting started" topics like Release notes and version skew, Learning environment, Production environment (Container runtimes, Installing Kubernetes with deployment tools, Bootstrapping clusters with kubeadm), and more.

Creating Highly Available clusters with kubeadm

This page explains two different approaches to setting up a highly available Kubernetes cluster using kubeadm:

- With stacked control plane nodes. This approach requires less infrastructure. The etcd members and control plane nodes are co-located.
- With an external etcd cluster. This approach requires more infrastructure. The control plane nodes and etcd members are separated.

Before proceeding, you should carefully consider which approach best meets the needs of your applications and environment. [This comparison topic](#) outlines the advantages and disadvantages of each.

If you encounter issues with setting up the HA cluster, please provide us with feedback in the kubeadm [issue tracker](#).

See also [The upgrade documentation](#).

Caution: This page does not address running your cluster on a cloud provider. In a cloud environment, neither approach documented here works with Service objects of type LoadBalancer, or with dynamic PersistentVolumes.

[Before you begin](#)
[First steps for both methods](#)
[Stacked control plane and etcd nodes](#)
[External etcd nodes](#)
[Common tasks after bootstrapping control plane](#)
[Manual certificate distribution](#)

Ref: <https://kubernetes.io/docs/setup/independent/high-availability/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Orchestrator Assignment



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Orchestrator Assignment

- Kubernetes have several way for control/restrict pod to run on nodes
- nodeSelector
 - Give some customize label for classify host and define some selector criteria for specific node run Pods
- Interlude
 - Build-in labels with nodes
 - kubernetes.io/hostname
 - failure-domain.beta.kubernetes.io/zone
 - failure-domain.beta.kubernetes.io/region
 - beta.kubernetes.io/instance-type
 - beta.kubernetes.io/arch
- Affinity
 - Node Affinity
 - Inter-Pods Affinity and Anti-Affinity
- Taints and tolerations



nodeSelector

Nodes: kubernetes-ms, kubernetes-1, kubernetes-2

```
praparn@kubernetes-ms:~$ kubectl label nodes kubernetes-ms storage=M2
node "kubernetes-ms" labeled
praparn@kubernetes-ms:~$ kubectl label nodes kubernetes-1 storage=SSD
node "kubernetes-1" labeled
praparn@kubernetes-ms:~$ kubectl label nodes kubernetes-2 storage=SAS
node "kubernetes-2" labeled
praparn@kubernetes-ms:~$ kubectl describe nodes
Name:           kubernetes-1
Role:           kubernetes-1
Labels:          beta.kubernetes.io/arch=amd64
                  beta.kubernetes.io/os=linux
                  kubernetes.io/hostname=kubernetes-1
                  storage=SSD
Name:           kubernetes-2
Role:           kubernetes-2
Labels:          beta.kubernetes.io/arch=amd64
                  beta.kubernetes.io/os=linux
                  kubernetes.io/hostname=kubernetes-2
                  storage=SAS
Name:           kubernetes-ms
Role:           kubernetes-ms
Labels:          beta.kubernetes.io/arch=amd64
                  beta.kubernetes.io/os=linux
                  kubernetes.io/hostname=kubernetes-ms
                  node-role.kubernetes.io/master=
                  storage=M2
```

Pods YAML File:

```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12   spec:
13     containers:
14       - name: webtest
15         image: labdocker/cluster:webserviceelite
16         ports:
17           - containerPort: 5000
18             protocol: TCP
19     nodeSelector:
20       storage: M2
```



nodeSelector

```
[praparn@kubernetes-ms:~$ kubectl create -f webtest_pod_nodeselector.yml
pod "webtest" created
[praparn@kubernetes-ms:~$ kubectl describe pods/webtest
Name:           webtest
Namespace:      default
Node:          kubernetes-ms/192.168.99.200
Start Time:    Sun, 23 Jul 2017 13:20:34 +0000
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                version=1.0
Annotations:   <none>
Status:        Pending
```

```
Node-Selectors: storage=M2
Tolerations:  node.alpha.kubernetes.io/notReady:NoExecute for 300s
              node.alpha.kubernetes.io/unreachable:NoExecute for 300s
Events:
FirstSeen     LastSeen      Count  From           SubObjectPath      Type  Reason
---          ---          ---   ---           ---           ---           ---  ---
--          --          --   --           --           --           --  --
12m          12m          1    default-scheduler
cheduled      Successfully assigned webtest to kubernetes-ms
12m          12m          1    kubelet, kubernetes-ms
uccesfulMountVolume  MountVolume.SetUp succeeded for volume "default-token-1ltqv"
12m          12m          1    kubelet, kubernetes-ms spec.containers{webtest}
ulling       pulling image "labdocker/cluster:webservicelite"
11m          11m          1    kubelet, kubernetes-ms spec.containers{webtest}
ulled       Successfully pulled image "labdocker/cluster:webservicelite"
11m          11m          1    kubelet, kubernetes-ms spec.containers{webtest}
reated      Created container
11m          11m          1    kubelet, kubernetes-ms spec.containers{webtest}
arted       Started container
[praparn@kubernetes-ms:~$ kubectl get pods -o wide
NAME    READY  STATUS    RESTARTS   AGE     IP          NODE
webtest  1/1    Running   0          12m    10.32.0.3  kubernetes-ms
[praparn@kubernetes-ms:~$ ]
```



Interlude

Nodes: kubernetes-ms, kubernetes-1, kubernetes-2

```
praparn@kubernetes-ms:~$ kubectl label nodes kubernetes-ms storage=M2
node "kubernetes-ms" labeled
praparn@kubernetes-ms:~$ kubectl label nodes kubernetes-1 storage=SSD
node "kubernetes-1" labeled
praparn@kubernetes-ms:~$ kubectl label nodes kubernetes-2 storage=SAS
node "kubernetes-2" labeled
praparn@kubernetes-ms:~$ kubectl describe nodes
Name:           kubernetes-1
Role:           kubernetes-1
Labels:         beta.kubernetes.io/arch=amd64
                beta.kubernetes.io/os=linux
                kubernetes.io/hostname=kubernetes-1
                storage=SSD
```

```
Name:           kubernetes-2
Role:           kubernetes-2
Labels:         beta.kubernetes.io/arch=amd64
                beta.kubernetes.io/os=linux
                kubernetes.io/hostname=kubernetes-2
                storage=SAS
```

```
Name:           kubernetes-ms
Role:           kubernetes-ms
Labels:         beta.kubernetes.io/arch=amd64
                beta.kubernetes.io/os=linux
                kubernetes.io/hostname=kubernetes-ms
                node-role.kubernetes.io/master=
                storage=M2
```

Pods YAML File:

```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12   spec:
13     containers:
14       - name: webtest
15         image: labdocker/cluster:webservicelite
16         ports:
17           - containerPort: 5000
18             protocol: TCP
19             nodeSelector:
20               kubernetes.io/hostname: kubernetes-1
```



Interlude

```
praparn@kubernetes-ms:~$ kubectl create -f webtest_pod_interlude.yml
pod "webtest" created
praparn@kubernetes-ms:~$ kubectl get pods -o wide
NAME      READY   STATUS            RESTARTS   AGE       IP           NODE
webtest   0/1    ContainerCreating   0          6s      <none>     kubernetes-1
praparn@kubernetes-ms:~$ kubectl describe pods/webtest
Name:         webtest
Namespace:    default
Node:         kubernetes-1/192.168.99.201
Start Time:   Sun, 23 Jul 2017 14:11:36 +0000
Labels:       environment=development
              module=WebServer
              name=web
              owner=Praparn_L
              version=1.0
Annotations:  <none>
Status:       Pending
IP:
Containers:
  webtest:
    Container ID:        labdocker/cluster:webserviceelite
    Image:               labdocker/cluster:webserviceelite
    Image ID:             ...
    Port:                5000/TCP
    State:               Waiting
      Reason:             ContainerCreating
    Ready:               False
    Restart Count:       0
    Environment:         <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-1ltqv (ro)
Conditions:
  Type      Status
  Initialized  True
  Ready      False
  PodScheduled  True
Volumes:
  default-token-1ltqv:
    Type:      Secret (a volume populated by a Secret)
    SecretName: default-token-1ltqv
    Optional:   false
  QoS Class:  BestEffort
  Node-Selectors: kubernetes.io/hostname=kubernetes-1
```



Affinity

- Node Affinity
 - Similar to “nodeSelector” but more flexible for selection
 - requiredDuringSchedulingIgnoredDuringExecution (Hard)
 - preferredDuringSchedulingIgnoredDuringExecution (Soft)
 - If Pods was define both “nodeSelector” and “Affinity”. It need to satisfy both for operate
- Inter-pod Affinity and Anti-affinity
 - “based on labels on pods (X) that are already running on the node with criteria (Y)”
 - Add constrain/dependence from Pods
 - podAffinity (Run with existing pods)
 - podAntiAffinity (Not run with existing pods)
 - Criteria(Y) will consider as “topologyKey”
 - Ex:
 - Run pods (podAffinity) on any node with same hostname (topologyKey (Y)=hostname) with existing pods that label “environment=development” (X)

Node Affinity

Pods YAML File:

```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12    spec:
13      affinity:
14        nodeAffinity:
15          requiredDuringSchedulingIgnoredDuringExecution:
16            nodeSelectorTerms:
17              - matchExpressions:
18                - key: beta.kubernetes.io/os
19                  operator: In #In, NotIn, Exists, DoesNotExist, Gt, Lt
20                  values:
21                    - linux
22          preferredDuringSchedulingIgnoredDuringExecution:
23            - weight: 1
24              preference:
25                matchExpressions:
26                  - key: storage
27                      operator: In
28                      values:
29                        - SSD
30
31        containers:
32          - name: webtest
33            image: labdocker/cluster:webservicelite
34            ports:
35              - containerPort: 5000
36                protocol: TCP
```



Inter-pod Affinity and Anti-affinity

Pods YAML File:

```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: webtest2
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11
12   spec:
13     affinity:
14       podAffinity:
15         requiredDuringSchedulingIgnoredDuringExecution:
16           - labelSelector:
17             matchExpressions:
18               - key: environment
19                 operator: In
20                 values:
21                   - development
22             topologyKey: kubernetes.io/hostname
23
24     podAntiAffinity:
25       preferredDuringSchedulingIgnoredDuringExecution:
26         - weight: 1
27           podAffinityTerm:
28             labelSelector:
29               matchExpressions:
30                 - key: module
31                   operator: In
32                   values:
33                     - DBServer
34             topologyKey: storage
35
36   containers:
37     - name: webtest2
38       image: labdocker/cluster:webservicelite
39       ports:
40         - containerPort: 5000
```

“Run Pods on any node that have existing pods with key (environment=development) on node same hostname”

“Don’t run Pods on any node that have existing pods with key (module=DBServer) on node same storage type”



Taint and Tolerations

- Node side consideration
- Force/Repel Pods from Node
- Taint will apply to Node for protect “node” to run any pods without “tolerate” match taint
- Tolerations apply to Pods for suggest (Not required) Pods to schedule
- Use Case:
 - Dedicated Node / Maintenance Node
 - Special Hardware Node
- Value for Taint
 - PreferNoSchedule
 - NoSchedule
 - NoExecute



Taint and Tolerations

Taint on Node:

```
praparn@kubernetes-ms:~$ kubectl taint nodes kubernetes-1 dedicated=admin:NoSchedule
node "kubernetes-1" tainted
```

Pods YAML File:

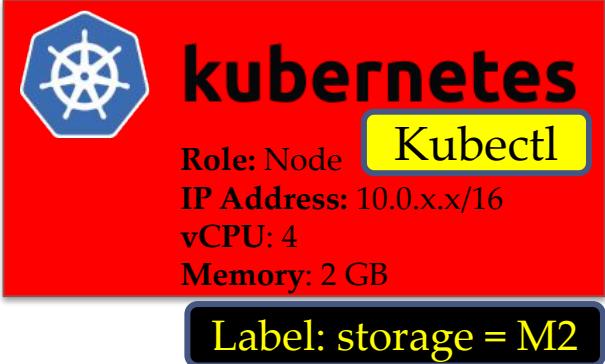
```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11 spec:
12   containers:
13     - name: webtest
14       image: labdocker/cluster:webservicelite
15       ports:
16         - containerPort: 5000
17           protocol: TCP
18   tolerations:
19     - key: "dedicated"
20       operator: "Equal"
21       value: "admin"
22       effect: "NoSchedule"
23   nodeSelector:
24     kubernetes.io/hostname: kubernetes-1
```

Tolerations:
Key="dedicated"
Operator="Equal"
Value="Admin"
Effect="NoSchedule"

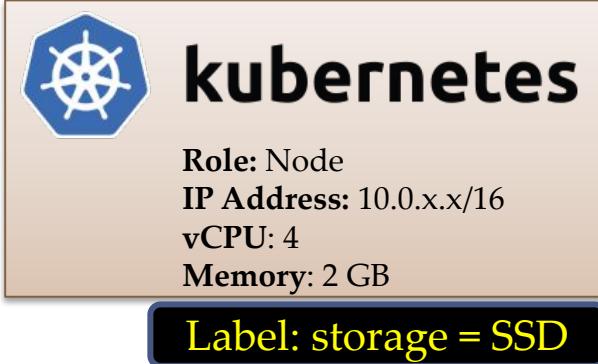


Workshop: Orchestrator Assignment

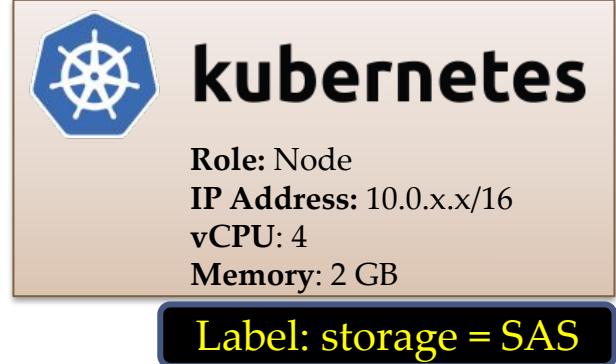
Machine: Kubernetes_MS



Machine: Kubernetes_1



Machine: Kubernetes_2



Role: Client

Lab Section:

- Part 1: nodeSelector
- Part 2: Interlude
- Part 3: Affinity (Node)
- Part 4: Inter-Pod Affinity and Anti-affinity
- Part 5: Taint and Tolerations

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Stateful Application Deployment



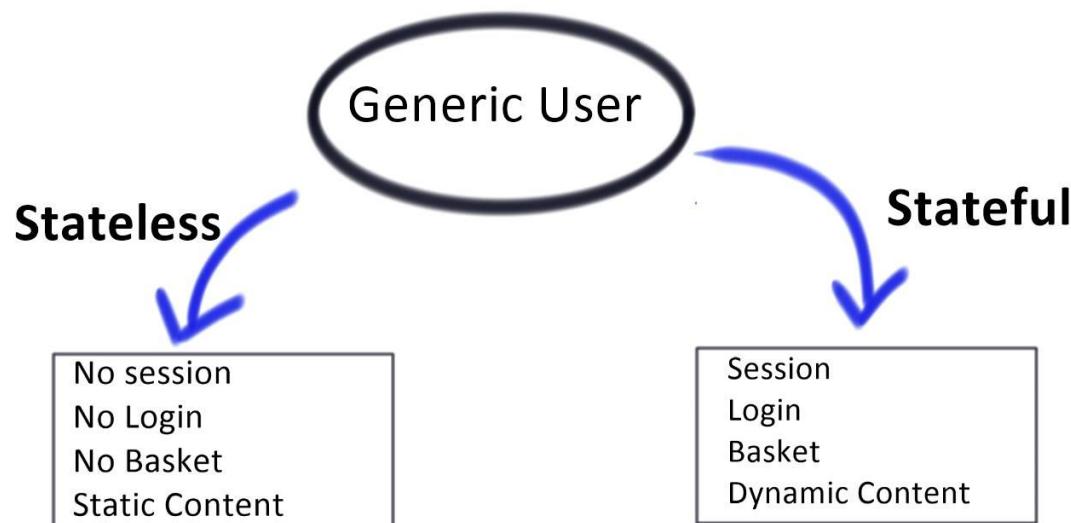
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Stateful Application Deployment

- Some application need “state” for keep application flow and store some information on “session” (Such as login’s session id) that store on web server or middle tier server module
- Ex: Joomla, Wordpress, Mantis (Bug Tracking), Normal etc

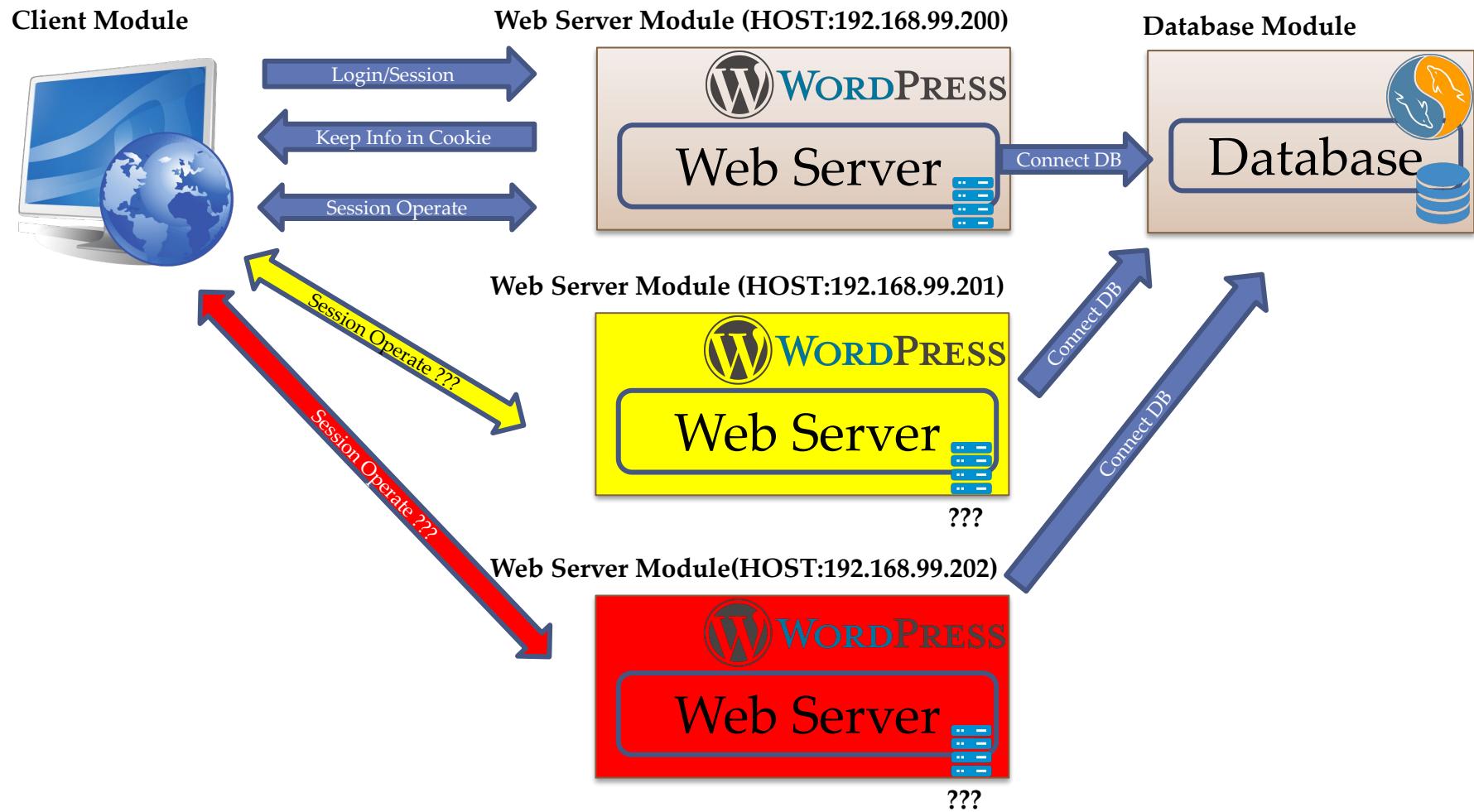


Stateful Application Deployment

- Considering
 - Original “HTTP” protocol is “stateless”
 - Stateful application need to keep session by web/app server and keep “cookies” on client for pass authentication
 - Work on memory for keep session (Fast/Easy but consume resource)
 - Many problem with native mobile app/Centralize Problem
 - Scale will effect for consideration traffic redirect to correct server (Keep state)
- Awareness
 - Container is naturally design for “stateless” application
 - All load-balance/dispatch job is not aware about “state” of application inside



Stateful Application Deployment

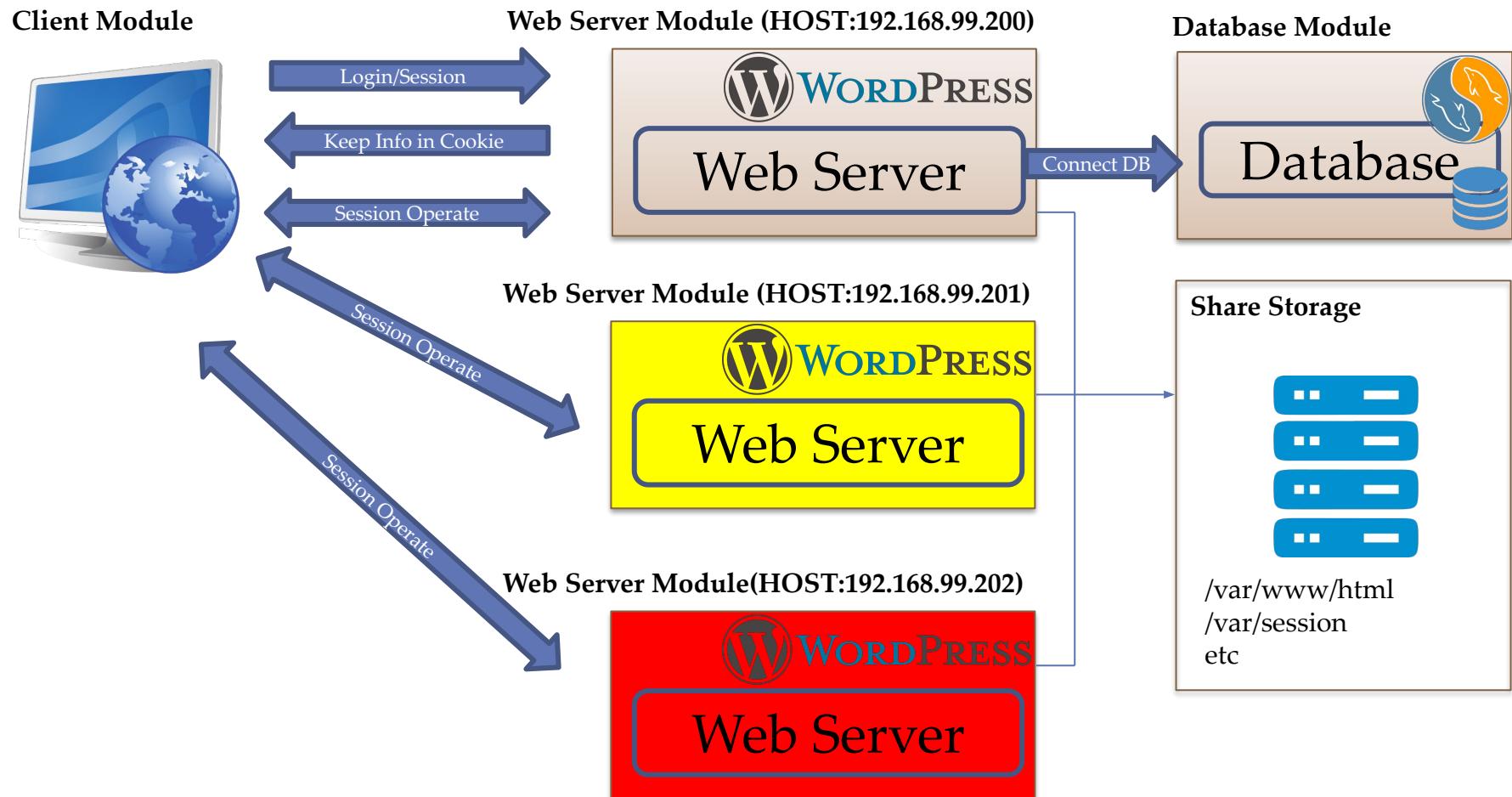


Stateful Application Deployment

- Solution ?

- SDS (Software-Defined Storage) for make centralize storage pool
- Share centralize storage pool for all node
- For Web/App Server
 - Keep application path / Session path on storage pool
 - Every server will read/write on same place
- For Database Server
 - Many option for operate (depend on type of database)
 - Active/Active
 - Active/Hot-Passive
 - Active/Cold-Passive
 - Postgres Kubedb Tool (Beta)
 - Idea also keep data on storage pool

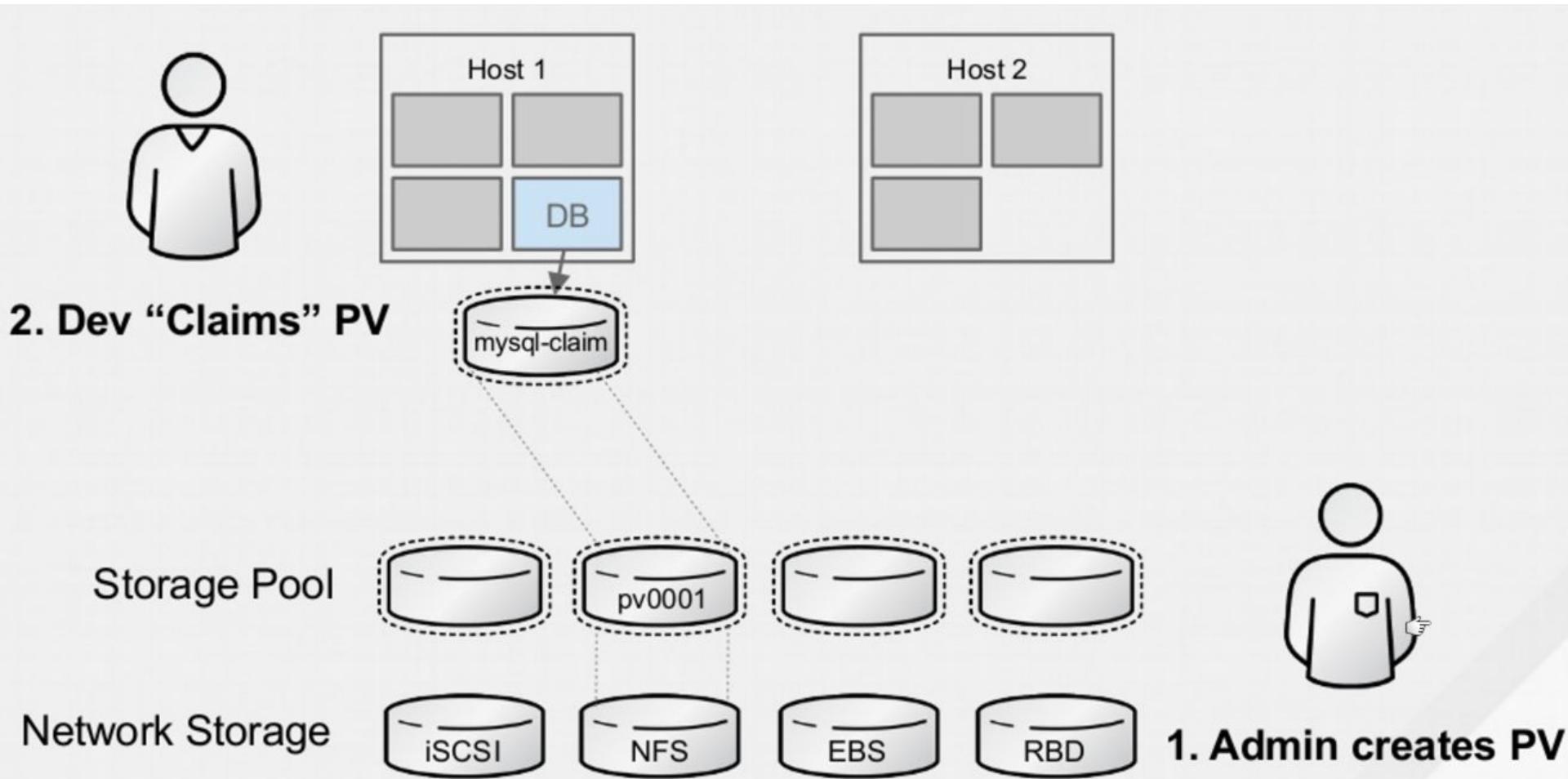
Stateful Application Deployment



Persistent Volume



Persistent Volume



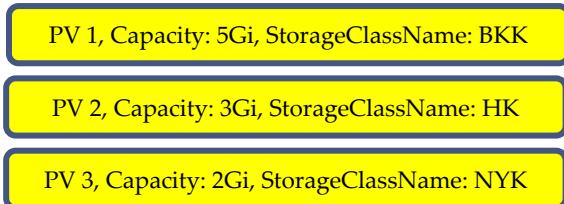
Persistent Volume

- Persistent Volume (PV)
 - Resource in cluster system
 - Act like pieces of storage (Independence from Pods)
 - Lifecycle was depended on Pods who used PV via PVC(Persistent Volume Claim)
 - Multiple type of PV as plugin support
- Persistent Volume Claim (PVC)
 - Similar Pod, That create for request storage from PV
 - PVC can specific
 - Size: Datasize for claim storage
 - Access Method:
 - ReadWriteOnce (RWO)
 - ReadOnlyMany (ROX)
 - ReadWriteMany (RWX)
- StorageClass
 - “Profiling” storage concept
 - Easy to classify storage (IOPS, Region etc)

Persistent Volume

- Volume Life Cycle

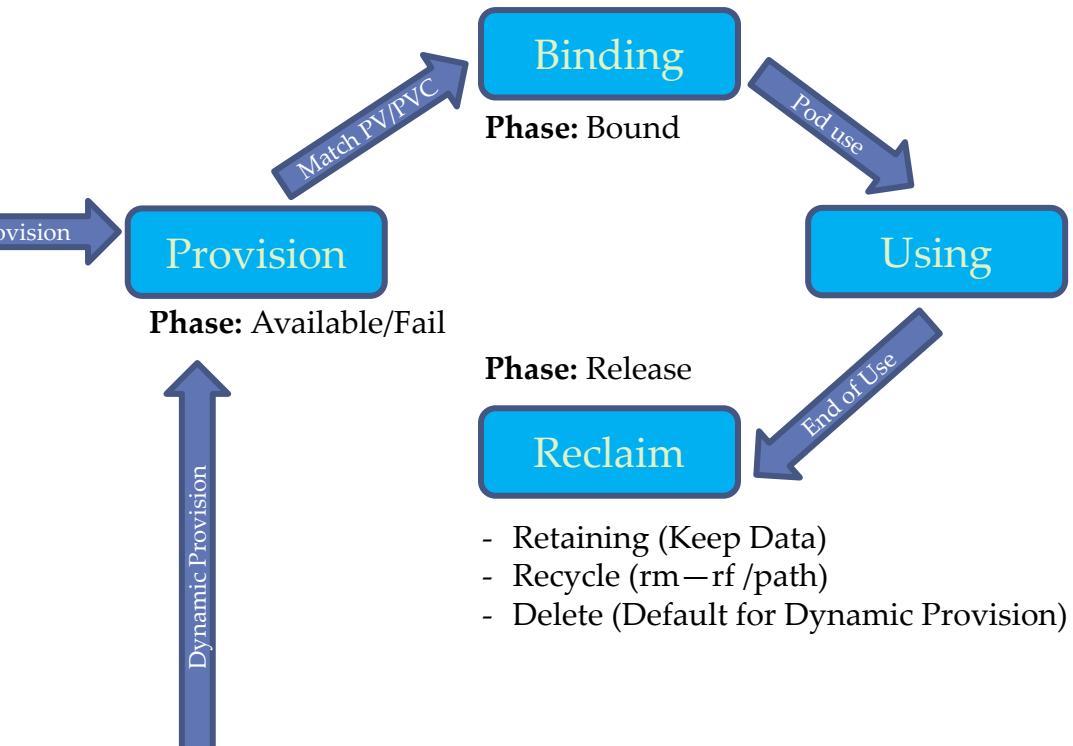
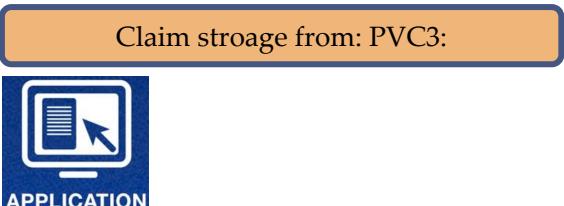
PV Pool



PVC Request

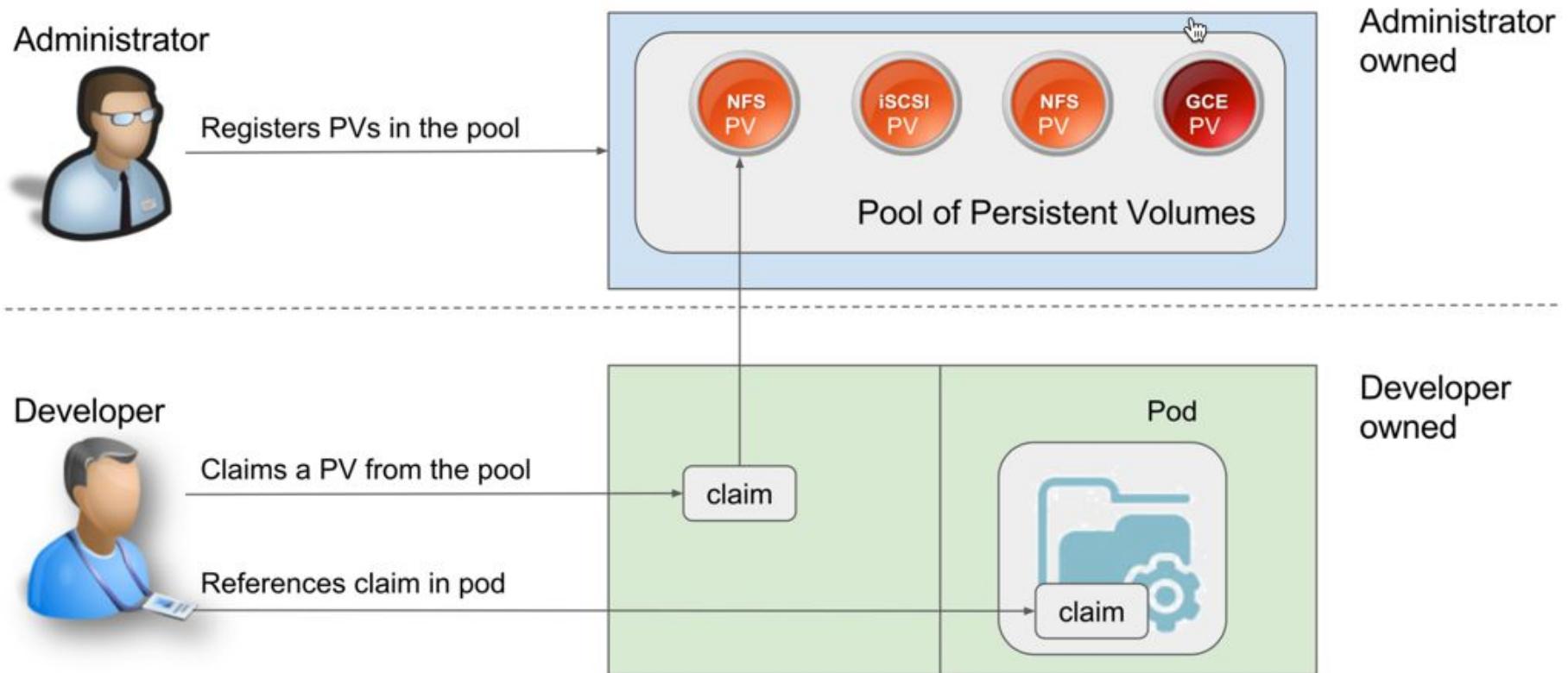


Pods for Application



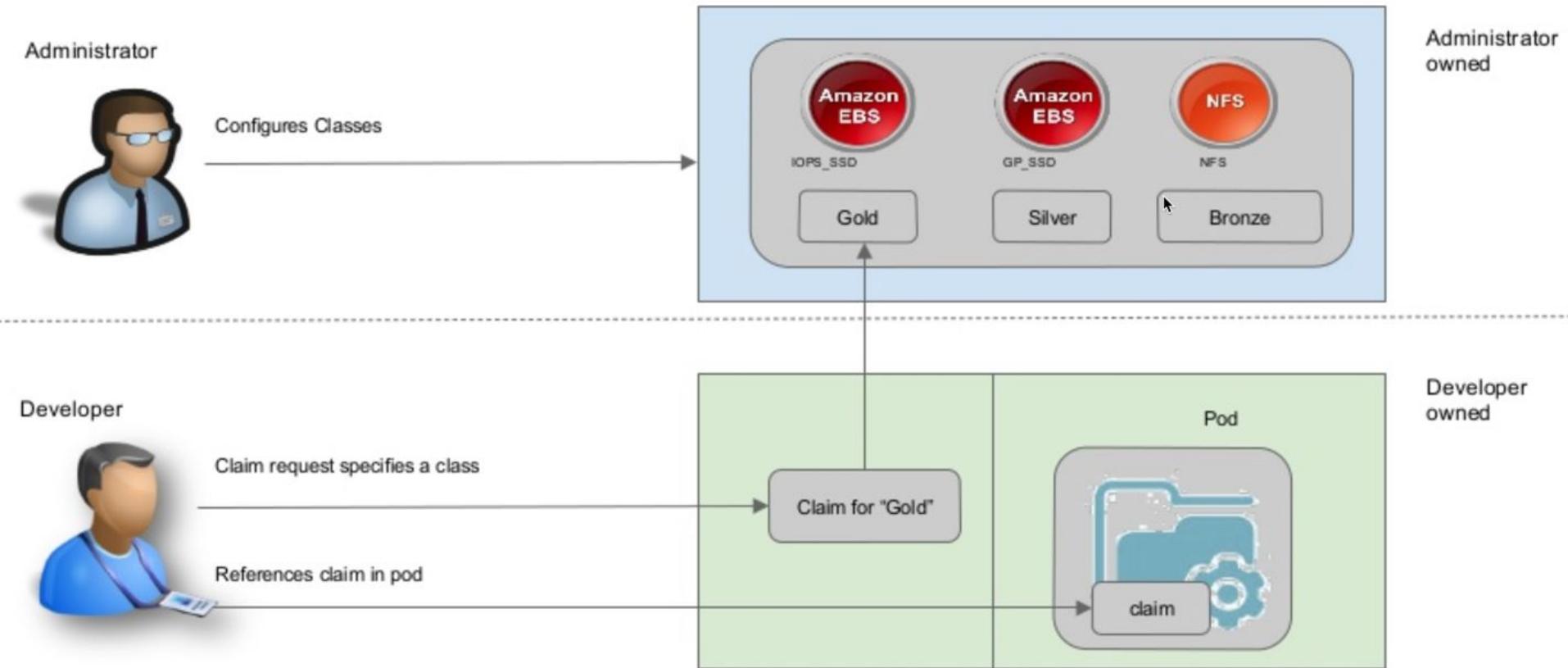
Persistent Volume

- Static Provision



Persistent Volume

- Dynamic Provision



Persistent Volume

- Type of Persistent Volume/Access Method

Volume Plugin	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
AWSElasticBlockStore	✓	-	-
AzureFile	✓	✓	✓
AzureDisk	✓	-	-
CephFS	✓	✓	✓
Cinder	✓	-	-
FC	✓	✓	-
FlexVolume	✓	✓	-
Flocker	✓	-	-
GCEPersistentDisk	✓	✓	-
Glusterfs	✓	✓	✓
HostPath	✓	-	-
iSCSI	✓	✓	-
PhotonPersistentDisk	✓	-	-
Quobyte	✓	✓	✓
NFS	✓	✓	✓
RBD	✓	✓	-
VsphereVolume	✓	-	-
PortworxVolume	✓	-	✓
ScaleIO	✓	✓	-
StorageOS	✓	-	-

Persistent Volume

- Static Provision
- Persistent Volume

```
1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    name: nfs-share-pv
5    labels:
6      name: nfs-share-pv
7      owner: Praparn_L
8      version: "1.0"
9    module: PV
10   environment: development
11
12  spec:
13    capacity:
14      storage: 1Gi
15    storageClassName: ""
16    accessModes:
17      - ReadWriteMany
18    nfs:
19      server: 192.168.99.200
20      path: "/var/nfsshare"
```

Persistent Volume Claims

```
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: nfs-share-pvc
5    labels:
6      name: nfs-share-pvc
7      owner: Praparn_L
8      version: "1.0"
9    module: PVC
10   environment: development
11
12  spec:
13    accessModes:
14      - ReadWriteMany
15    storageClassName: ""
16    resources:
17      requests:
18        storage: 500Mi
19    selector:
20      matchLabels:
21        name: nfs-share-pv
22        owner: Praparn_L
23        version: "1.0"
24        module: PV
25        environment: development
```

Persistent Volume

- Dynamic Provision
- Storage Class

```
1 kind: StorageClass
2 apiVersion: storage.k8s.io/v1
3 metadata:
4   name: aws-ebs
5   provisioner: kubernetes.io/aws-ebs
6   parameters:
7     type: gp2
8     fsType: ext4
```

Persistent Volume Claims

```
1   apiVersion: v1
2   kind: PersistentVolumeClaim
3   metadata:
4     name: aws-ebs-pvc
5     annotations:
6       volume.beta.kubernetes.io/storage-class: aws-ebs
7   spec:
8     accessModes:
9       - ReadWriteOnce
10    resources:
11      requests:
12        storage: 5Gi
```



Persistent Volume

- Dynamic Provision
 - Test create SC/PVC

```
[ubuntu@ip-10-0-1-229:~$ kubectl apply -f ~/kubernetes_201904/WorkShop_2.7_Persistent_Storage/aws_sc.yml
storageclass.storage.k8s.io/aws-ebs created
[ubuntu@ip-10-0-1-229:~$ kubectl describe -f ~/kubernetes_201904/WorkShop_2.7_Persistent_Storage/aws_sc.yml
Name:           aws-ebs
IsDefaultClass: No
Annotations:   kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":{"annotations":{},"name":"aws-ebs"},"parameters":{"fsType":"ext4","type":"gp2"},"provisioner":"kubernetes.io/aws-ebs"}
Provisioner:    kubernetes.io/aws-ebs
Parameters:    fsType=ext4,type=gp2
AllowVolumeExpansion: <unset>
MountOptions:  <none>
ReclaimPolicy: Delete
VolumeBindingMode: Immediate
Events:        <none>
[ubuntu@ip-10-0-1-229:~$ kubectl describe sc/aws-ebs
Name:           aws-ebs
IsDefaultClass: No
Annotations:   kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":{"annotations":{},"name":"aws-ebs"},"parameters":{"fsType":"ext4","type":"gp2"},"provisioner":"kubernetes.io/aws-ebs"}
Provisioner:    kubernetes.io/aws-ebs
Parameters:    fsType=ext4,type=gp2
AllowVolumeExpansion: <unset>
MountOptions:  <none>
ReclaimPolicy: Delete
VolumeBindingMode: Immediate
Events:        <none>
[ubuntu@ip-10-0-1-229:~$ kubectl apply -f ~/kubernetes_201904/WorkShop_2.7_Persistent_Storage/aws_pvc.yml
persistentvolumeclaim/aws-ebs-pvc created
[ubuntu@ip-10-0-1-229:~$ kubectl get pvc
NAME      STATUS    VOLUME    CAPACITY   ACCESS MODES  STORAGECLASS   AGE
aws-ebs-pvc  Pending          aws-ebs      4s
[ubuntu@ip-10-0-1-229:~$ kubectl get pvc
NAME      STATUS    VOLUME    CAPACITY   ACCESS MODES  STORAGECLASS   AGE
aws-ebs-pvc  Bound    pvc-0a14702e-4268-11e9-9e8e-02d2fffaea91a  5Gi        RWO          aws-ebs     18s
[ubuntu@ip-10-0-1-229:~$ kubectl describe pvc
NAME: aws-ebs-pvc
Status: Pending
```



Persistent Volume

- Dynamic Provision
 - Test create SC/PVC

```
[ubuntu@ip-10-0-1-229:~$ kubectl get pv
NAME           CAPACITY   ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM          STORAGECLASS  REASON  AGE
pvc-0a14702e-4268-11e9-9e8e-02d2ffaea91a  5Gi        RWO          Delete        Bound    default/aws-ebs-pvc  aws-ebs      26m
[ubuntu@ip-10-0-1-229:~$ kubectl get pvc
NAME      STATUS    VOLUME
aws-ebs-pvc  Bound    pvc-0a14702e-4268-11e9-9e8e-02d2ffaea91a
[ubuntu@ip-10-0-1-229:~$ ]
```

Name	Volume ID	Size	Type	IOPS	Snapshot	Created	Avg.
kubernetes-dynamic-pvc-0a14702e-4268-11e9-9e8e-02d2ffaea91a	vol-0be1ef23e8b209c0d	5 GiB	gp2	100		March 9, 2019 at 7:37:09 PM UTC+7	ap-southeast-1a
	vol-038c386...	30 GiB	gp2	100	snap-0f7083a4...	March 8, 2019 at 10:45:40 PM UTC+7	ap-southeast-1a



Persistent Volume

- Dynamic Provision

The screenshot shows the Kubernetes web interface with the following details:

Cluster sidebar: Namespaces, Nodes, Persistent Volumes, Roles, **Storage Classes** (selected), Namespace, default.

Storage Classes page:

- Details** section:
 - Name: aws-ebs
 - Annotations: `kubectl.kubernetes.io/last-applied-configuration`
 - Creation Time: 2019-03-09T12:34 UTC
 - Labels: -
 - Provisioner: kubernetes.io/aws-ebs
 - Parameters: `fsType: ext4`, `type: gp2`
- Persistent Volumes** table:

Name	Capacity	Access Modes	Reclaim Policy	Status	Claim	Storage Class	Reason	Age	⋮
pvc-0a14702e-4...	5Gi	ReadWriteOnce	Delete	Bound	default/aws-ebs-...	aws-ebs	-	29 minutes	⋮



Persistent Volume

- Dynamic Provision

The screenshot shows two views in the Kubernetes dashboard:

Persistent Volumes View:

Name	Capacity	Access Modes	Reclaim Policy	Status	Claim	Storage Class	Reason	Age
pvc-0a14702e-4...	5Gi	ReadWriteOnce	Delete	Bound	default/aws-ebs-...	aws-ebs	-	30 minutes

Persistent Volume Claims View:

Name	Status	Volume	Capacity	Access Modes	Storage Class	Age
aws-ebs-pvc	Bound	pvc-0a14702e-4268-11e9-9e8e-02d2ffaea91a	5Gi	ReadWriteOnce	-	30 minutes



Persistent Volume

- Dynamic Provision

```
[ubuntu@ip-10-0-1-229:~$ kubectl get pv
NAME          CAPACITY   ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM           STORAGECLASS  REASON  AGE
pvc-0a14702e-4268-11e9-9e8e-02d2ffa91a  5Gi        RWO          Delete        Bound    default/aws-ebs-pvc  aws-ebs      37m
[ubuntu@ip-10-0-1-229:~$ kubectl get pvc
NAME      STATUS    VOLUME          CAPACITY   ACCESS MODES  STORAGECLASS  AGE
aws-ebs-pvc  Bound    pvc-0a14702e-4268-11e9-9e8e-02d2ffa91a  5Gi        RWO          aws-ebs      37m
[ubuntu@ip-10-0-1-229:~$ kubectl delete -f ~/kubernetes_201904/WorkShop_2.7_Persistent_Storage/aws_pvc.yml
persistentvolumeclaim "aws-ebs-pvc" deleted
[ubuntu@ip-10-0-1-229:~$ kubectl get pv
No resources found.
[ubuntu@ip-10-0-1-229:~$ kubectl get pvc
No resources found.
[ubuntu@ip-10-0-1-229:~$ ]
```

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Launch Templates

Spot Requests

Create Volume

Actions

Name : kubernetes-dynamic-pvc-0a14702e-4268-11... Add filter

Name	Volume ID	Size	Volume Type	IOPS
No Volumes found				

Select a volume above



Persistent Volume

- Deployment reference

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: webtest
5   labels:
6     name: web
7     owner: Praparn_L
8     version: "1.0"
9     module: WebServer
10    environment: development
11
12  spec:
13    replicas: 1
14    selector:
15      matchLabels:
16        name: web
17        owner: "Praparn_L"
18        version: "1.0"
19        module: WebServer
20        environment: development
21
22    template:
23      metadata:
24        labels:
25          name: web
26          owner: Praparn_L
27          version: "1.0"
28          module: WebServer
29          environment: development
30
31      spec:
32        containers:
33          - name: webtest
34            image: labdocker/cluster:webservicelite_v1
35            ports:
36              - containerPort: 5000
37                protocol: TCP
38            volumeMounts:
39              - name: aws-share-pvc
40                mountPath: /usr/src/app
41            initContainers:
42              - name: download
43                image: labdocker/alpineweb:latest
44                volumeMounts:
45                  - name: aws-share-pvc
46                    mountPath: /source
47                command:
48                  - wget
49                  - "-O"
50                  - "/source/mainlite.py"
51                  - "https://raw.githubusercontent.com/praparn/kubernetes\_201904/master/WorkShop\_2.7\_Persistent\_Storage/mainlite.py"
52
53      volumes:
54        - name: aws-share-pvc
55          persistentVolumeClaim:
56            claimName: aws-ebs-pvc
```



Run iniContainers Process



Reference for mount disk volume from "PVC"



Persistent Volume

- Dynamic Provision on Deployment

```
[ubuntu@ip-10-0-1-229:~$ kubectl create -f https://raw.githubusercontent.com/praparn/kubernetes_201904/master/WorkShop_2.7_Persistent_Storage/aws_webtest_deployall.yaml
deployment.apps/webtest created
[ubuntu@ip-10-0-1-229:~$ kubectl create -f https://raw.githubusercontent.com/praparn/kubernetes_201904/master/WorkShop_2.7_Persistent_Storage/aws_webtest_svc.yaml
service/webtest created
[ubuntu@ip-10-0-1-229:~$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
webtest   1/1     1           1           13s
[ubuntu@ip-10-0-1-229:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
webtest-6f4bdd5f6b-kj4jn   1/1     Running   0          18s
[ubuntu@ip-10-0-1-229:~$ kubectl describe pods/webtest-6f4bdd5f6b-kj4jn
Name:           webtest-6f4bdd5f6b-kj4jn
Namespace:      default
Priority:       0
PriorityClassName: <none>
Node:          ip-10-0-1-165.ap-southeast-1.compute.internal/10.0.1.165
Start Time:    Sat, 09 Mar 2019 16:05:40 +0000
Labels:         environment=development
                module=WebServer
                name=web
                owner=Praparn_L
                pod-template-hash=6f4bdd5f6b
                version=1.0
Annotations:   cni.projectcalico.org/podIP: 192.168.1.40/32
Status:        Running
IP:           192.168.1.40
Controlled By: ReplicaSet/webtest-6f4bdd5f6b
```

Events:				
Type	Reason	Age	From	Message
Normal	Scheduled	28s	default-scheduler	Successfully assigned default/webtest-6f4bdd5f6b-kj4jn to ip-10-0-1-165.ap-southeast-1.compute.internal
Normal	SuccessfulAttachVolume	27s	attachdetach-controller	AttachVolume.Attach succeeded for volume "pvc-e7988ede-4284-11e9-9e8e-02d2ffaea91a"
Normal	Pulling	23s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	pulling image "labdocker/alpineweb:latest"
Normal	Pulled	19s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	Successfully pulled image "labdocker/alpineweb:latest"
Normal	Created	19s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	Created container
Normal	Started	19s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	Started container
Normal	Pulled	18s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	Container image "labdocker/cluster:webservicelite_v1" already present on machine
Normal	Created	18s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	Created container
Normal	Started	18s	kubelet, ip-10-0-1-165.ap-southeast-1.compute.internal	Started container

Persistent Volume

A screenshot of a web browser window. The title bar shows various icons and links: kube, Create, Linu, doc, EC2, prep, Sett, QUI, High, and M. The address bar displays a URL: Not Secure | 18.136.102.184:32500. Below the address bar, a message says "Click to go back, hold to see history". The main content area features a large, bold heading: "Welcome Page from Container Python Lab". Below the heading, a timestamp is shown: "Checkpoint Date/Time: Sat Mar 9 16:08:09 2019".

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Persistent Volume

```
...-0-1-77: ~ -- bash | ...orker-1: ~ -- bash | ...-219-46: ~ -- bash
[ubuntu@ip-10-0-1-229:~]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
webtest-6f4bdd5f6b-kj4jn   1/1     Running   0          2m51s
[ubuntu@ip-10-0-1-229:~]$ kubectl exec -it webtest-6f4bdd5f6b-kj4jn sh
[/usr/src/app # vi mainlite.py
/usr/src/app #
```

```
from flask import Flask
import os
import time
app = Flask(__name__)

@app.route('/')
def hello():
    return '<H1>TEST EDIT Welcome Page from Container Python Lab </H1>Checkpoint Date/Time: ' + time.strftime("%c") +'\n'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
~
```

← → ⌂ ⓘ Not Secure | 18.136.102.184:32500

apps NMac Ked - Mac ... M Medium _Infra_ NOVA _jenkins_ vagrant Mesos Vue do

TEST EDIT Welcome Page from Container Python Lab

Checkpoint Date/Time: Sat Mar 9 16:10:42 2019



Persistent Volume

- Create Deployment

The screenshot shows the Kubernetes UI for managing workloads. The top navigation bar includes a search bar and buttons for creating, executing, viewing logs, editing, and deleting resources. The main content area displays a pod named "webtest-6f4bdd5f6b-kj4jn" in the "default" namespace. The "Details" tab shows the pod's metadata, such as name, namespace, labels, annotations, creation time, status, and QoS class. The "Containers" tab lists the "webtest" and "download" containers. The "webtest" container uses the image "labdocker/cluster:webservicelite_v1". The "download" container uses the image "labdocker/alpineweb:latest" and has a command to download a file from a GitHub URL: `wget -O /source/mainlite.py https://raw.githubusercontent.com/praparn/kubernetes_201904/master/WorkShop_2.7_Persistent_Storage/mainlite.py`.

Name:	webtest-6f4bdd5f6b-kj4jn	Network
Namespace:	default	Node: ip-10-0-1-165.ap-southeast-1.compute.internal
Labels:	environment: development, module: WebServer, name: web, owner: Paparn_L, pod-template-hash: 6f4bdd5f6b	IP: 192.168.1.40
Annotations:	cni.projectcalico.org/podIP: 192.168.1.40/32	
Creation Time:	2019-03-09T16:05 UTC	
Status:	Running	
QoS Class:	BestEffort	

Container	Image	Environment variables	Commands	Args
webtest	labdocker/cluster:webservicelite_v1	-	-	-
download	labdocker/alpineweb:latest	-	wget -O /source/mainlite.py https://raw.githubusercontent.com/praparn/kubernetes_201904/master/WorkShop_2.7_Persistent_Storage/mainlite.py	-



Persistent Volume

- Create Deployment

Kubernetes dashboard interface showing the details of a Pod named `webtest-6f4bdd5f6b-kj4jn`.

Pod Details:

Field	Value
name	web
owner	Paparn_L
pod-template-hash	6f4bdd5f6b

Events:

Message	Source	Sub-object	Count	First seen	Last seen
Successfully assigned default/webtest-6f4bdd5f6b-kj4jn to ip-10-0-1-165.ap-southeast-1.compute.internal	default-scheduler	-	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
AttachVolume.Attach succeeded for volume "pvc-e7988ede-4284-11e9-9e8e-02d2ffaea91a"	attachdetach-controller	-	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
pulling image "labdocker/alpineweb:latest"	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers(download)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Successfully pulled image "labdocker/alpineweb:latest"	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers(download)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Created container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers(download)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Started container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers(download)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Container image "labdocker/cluster:webserviceelite_v1" already present on machine	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.containers(webtest)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Created container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.containers(webtest)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Started container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.containers(webtest)	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC

Persistent Volume Claims:

Name	Status	Volume	Capacity	Access Modes	Storage Class	Age
aws-ebs-pvc	Bound	pvc-e7988ede-4284-11e9-9e8e-02d2ffaea91a	5Gi	ReadWriteOnce	-	9 minutes



Workshop: Persistence Storage

 kubernetes

Search

+ CREATE | 

☰ Workloads > Pods > webtest-6f4bdd5f6b-kj4jn

Cluster Namespaces Nodes Persistent Volumes Roles Storage Classes

Namespace default

Overview Workloads Cron Jobs Daemon Sets Deployments Jobs Pods Replica Sets Replication Controllers Stateful Sets Discovery and Load Balancing Ingresses Services

Config and Storage Config Maps Persistent Volume Claims Secrets

replicaset name: web owner: Paparn_L 1 / 1 7 minutes labdocker/cluster:webserviceclit... :

Events

Message	Source	Sub-object	Count	First seen	Last seen
Successfully assigned default/webtest-6f4bdd5f6b-kj4jn to ip-10-0-1-165.ap-southeast-1.compute.internal	default-scheduler	-	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
AttachVolume.Attach succeeded for volume "pvc-e7988ede-4284-11e9-9e8e-02d2ffaea91a"	attachdetach-controller	-	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
pulling image "labdocker/alpineweb.latest"	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers{download}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Successfully pulled image "labdocker/alpineweb:latest"	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers{download}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Created container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers{download}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Started container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.initContainers{download}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Container image "labdocker/cluster:webserviceclite_v1" already present on machine	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.containers{webtest}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Created container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.containers{webtest}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC
Started container	kubelet ip-10-0-1-165.ap-southeast-1.compute.internal	spec.containers{webtest}	1	2019-03-09T16:05 UTC	2019-03-09T16:05 UTC

Persistent Volume Claims

Name	Status	Volume	Capacity	Access Modes	Storage Class	Age
aws-ebs-pvc	Bound	pvc-e7988ede-4284-11e9-9e8e-02d2ffaea91a	5Gi	ReadWriteOnce	-	9 minutes



StatefulSet



Kubernetes: Production Workload Orchestration

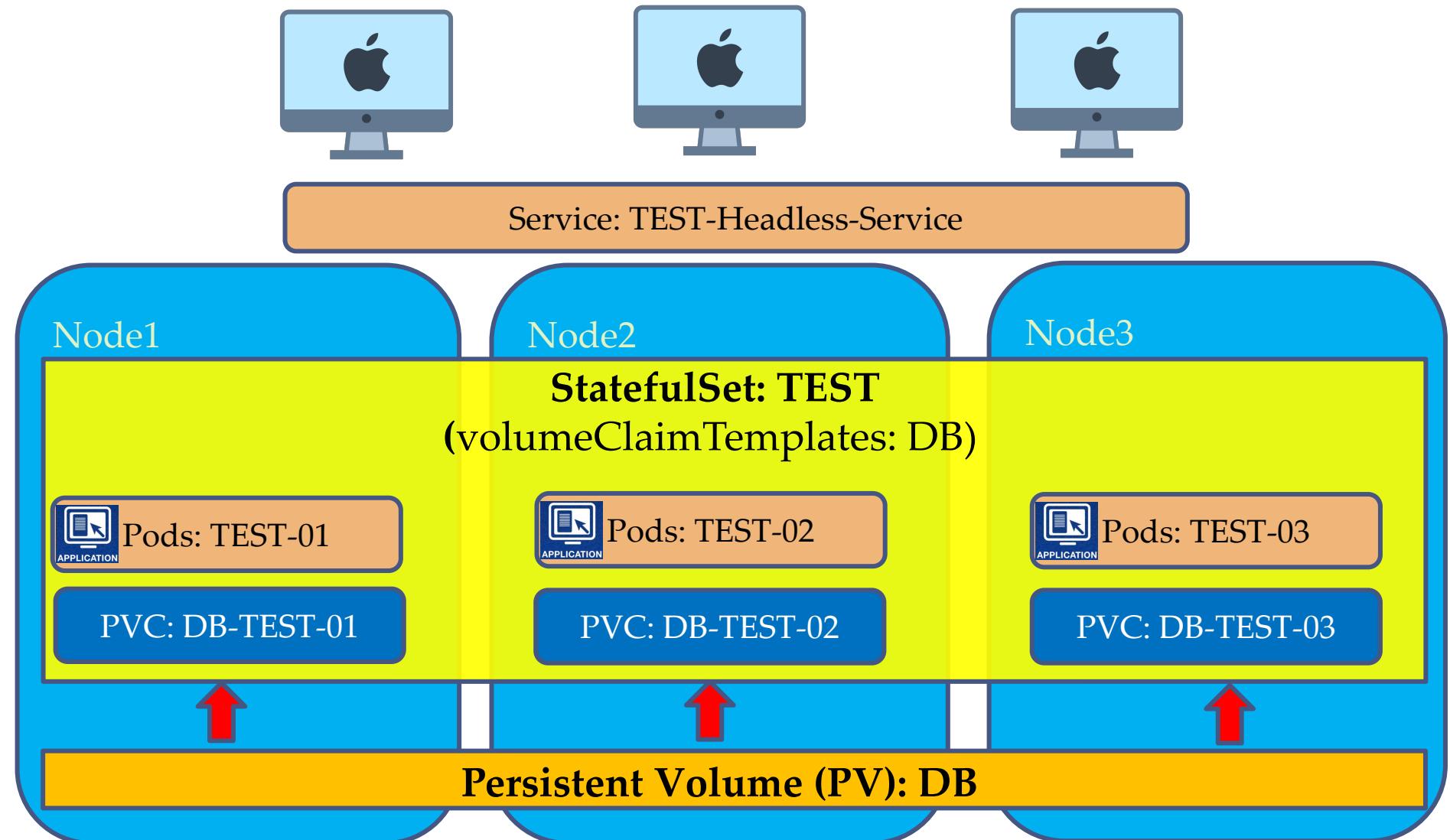


kubernetes
by Google

StatefulSet

- What is StatefulSet?
 - Do you remember Deployment ?
 - StatefulSet similar with Deployment
 - But StatefulSet is design for applicate that need
 - Persistent Storage
 - Stable Storage/Stable Network
 - Ordered for
 - Deployment (Create)
 - Scale
 - Roll Update
- Benefit from StatefulSet
 - Sequential create/scale/update Pods
 - Each Pods got unique resource
 - Name of Pods
 - Storage on Pods (Dedicate PVC)
 - Exist on Hosts/Network

StatefulSet



StatefulSet

- StatefulSet

```
16 apiVersion: apps/v1beta2
17 kind: StatefulSet
18 metadata:
19   name: TEST
20 spec:
21   serviceName: "TEST"
22   replicas: 3
23   selector:
24     matchLabels:
25       app: nginx
26   template:
27     metadata:
28       labels:
29         app: nginx
30     spec:
31       containers:
32         - name: nginx
33           image: labdocker/nginx:latest
34           ports:
35             - containerPort: 80
36               name: web
37               volumeMounts:
38                 - name: DB
39                   mountPath: /usr/share/nginx/html
40   volumeClaimTemplates:
41     - metadata:
42       name: DB
43       spec:
44         accessModes: [ "ReadWriteOnce" ]
45         resources:
46           requests:
47             storage: 1Gi
```

Service

```
2  apiVersion: v1
3  kind: Service
4  metadata:
5    name: TEST
6    labels:
7      app: TEST
8  spec:
9    ports:
10      - port: 80
11        name: TEST
12        clusterIP: None
13        selector:
14          app: nginx
```



Recapture Day 2

- Fundamental of Kubernetes
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- Security on Kubernetes
 - Network Policy
 - Volume Policy
 - Resource Usage Policy
 - Resource Consumption Policy
 - Access Control Policy
 - Security Policy
- Kubernetes in real world
 - Cluster Setup for Bare Metal
 - Orchestrator Assignment
 - nodeSelector
 - Interlude
 - Affinity
 - Taints/Tolerations
- Stateful application deployment
 - Consideration and Awareness
 - Persistent Volumes
 - StatefulSets



WORKSHOP ADVANCED DOCKER



สอนการ deploy Dockers ด้วย Kubernetes
จากประสบการณ์ใช้งานจริงบน Production ของ
application ระดับประเทศ



วิทยากร : คุณ PRAPARN LUNGPOONLARP
INFRASTRUCTURE ENGINEER, NETWORK ENGINEER,
SYSTEM ENGINEER



kubernetes

