

WORKSHOP ADVANCED DOCKER



สอนการ deploy Dockers บน 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)
 - Map and Secret
 - Jobs and Cron Jobs
 - Debug Log and Monitoring



Outline Day 2

- Fundamental of Kubernetes
 - ConfigMap Secret
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- 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
- Google Cloud Account (Free for 300\$) or other Cloud / Bare Metal / 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



Lab Resource

- Repository for lab

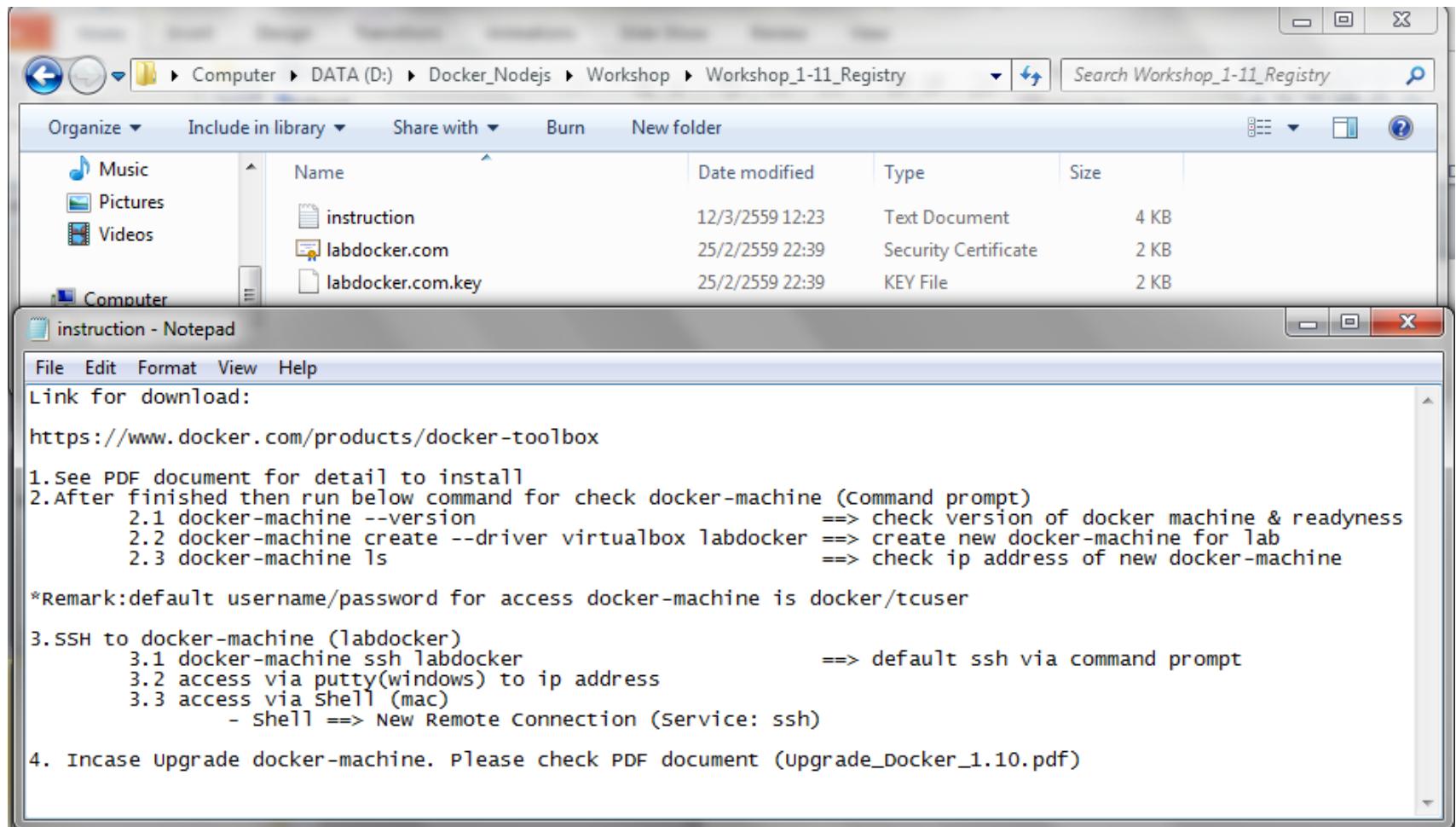
The screenshot shows a search results page for 'labdocker' on a platform that uses a Docker icon in its logo. The interface includes a search bar with the query 'labdocker', a 'Sign up' button, and a 'Log In' button. Below the search bar, the text 'Repositories (7)' is displayed. A dropdown menu shows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

Repository	Type	Stars	Pulls	Details
labdocker/alpineweb	public	0	31	DETAILS
labdocker/nginx	public	0	16	DETAILS
labdocker/alpine	public	0	7	DETAILS



Lab Resource

- Software in lab



Lab Resource

- Download on Google Drive
 - <https://goo.gl/oCUayz>
- Download on GitHub
 - git@github.com:praparn/kubernetes_20170128.git

No description, website, or topics provided.

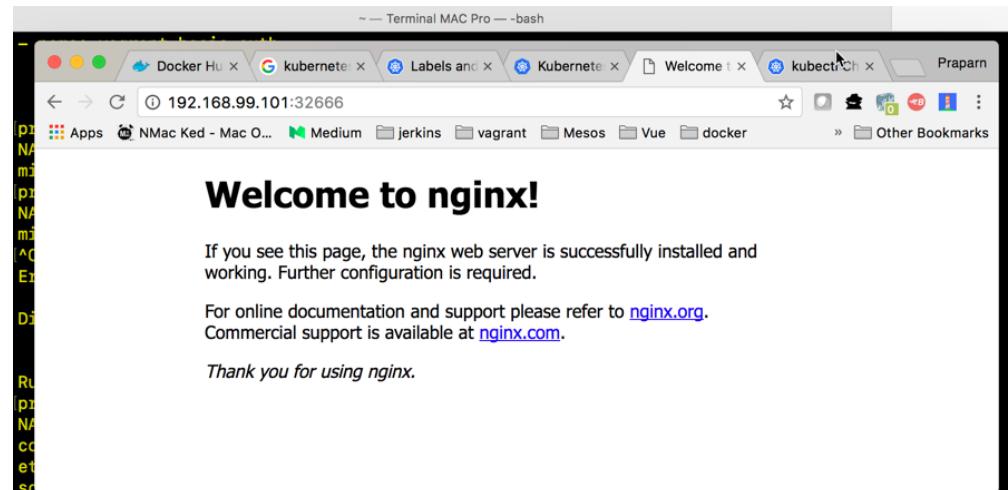
Branch: master ▾ New pull request

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

File	Commit	Date
.DS_Store	20171110003943	22 hours ago
Kubernetes_Training_master.pdf	20171110003943	22 hours ago
Untitled-1	2017110214055	an hour ago
WorkShop_1.1_Install_Kubernetes	2017110214055	an hour ago
WorkShop_1.2_Pods_Service_Deployment	2017110214055	an hour ago
WorkShop_1.3_Replication_Controller	2017110214055	an hour ago
WorkShop_1.4_Deployment	2017110214055	an hour ago
WorkShop_1.5_Volume	2017110214055	an hour ago
WorkShop_1.6_Liveness_Readiness_Probe	2017110214055	an hour ago
WorkShop_1.7_Resource_Management_and_HPA	2017110214055	an hour ago
WorkShop_2.1_ConfigMap_Secret	2017110214055	an hour ago
WorkShop_2.3_Log_and_Monitoring	2017110214055	an hour ago
WorkShop_2.4_Ingress_Network	2017110214055	an hour ago
WorkShop_2.5_Kubernetes_RealWorld	2017110214055	an hour ago
WorkShop_2.6_Orchestrator_Assignment	20171108235628	2 days ago
WorkShop_2.7_Persistent_Storage	20171110	23 hours ago
Workshop_2.2_Job_CronJob	20171110214055	an hour ago



Workshop 1.1: Install Kubernetes



A screenshot of a Mac browser window titled "Terminal MAC Pro - bash". The address bar shows "192.168.99.101:32666". The page content is "Welcome to nginx!". It says: "If you see this page, the nginx web server is successfully installed and working. Further configuration is required." Below that, it says: "For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com." At the bottom, it says "Thank you for using nginx.".

```
praparns-MacBook-Pro:~ praparn$ kubectl run webtest --image=labdocker/nginx:latest --port=80
deployment "webtest" created
praparns-MacBook-Pro:~ praparn$ clear

praparns-MacBook-Pro:~ praparn$ minikube ssh
$ exit
logout
praparns-MacBook-Pro:~ praparn$ kubectl expose deployment webtest --target-port=80 --type=NodePort
service "webtest" exposed
praparns-MacBook-Pro:~ praparn$ kubectl get svc
NAME      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes  10.0.0.1    <none>        443/TCP       4d
webtest    10.0.0.114   <nodes>       80:32666/TCP   6s
praparns-MacBook-Pro:~ praparn$ 
```



minikube

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop 1.1: Install Kubernetes

The screenshot shows the Kubernetes dashboard interface. The top navigation bar includes a back button, forward button, refresh button, and a search bar with the IP address 192.168.99.100. The title bar says "kubernetes Admin". On the left, a sidebar menu is open under the "Admin" tab, showing options like Namespaces, Nodes, Persistent Volumes, Storage Classes, Namespace, and Workloads (with sub-options: Deployments, Replica Sets, Replication Controllers, Daemon Sets, Stateful Sets, Jobs, Pods, and Services and discovery). The main content area is divided into three sections: "Namespaces", "Nodes", and "Storage Classes".

Namespaces

Name	Labels	Status	Age
default	-	Active	an hour
kube-public	-	Active	an hour
kube-system	-	Active	an hour

Nodes

Name	Labels	Ready	Age
minikube	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/hostname: minikube	True	an hour

Storage Classes

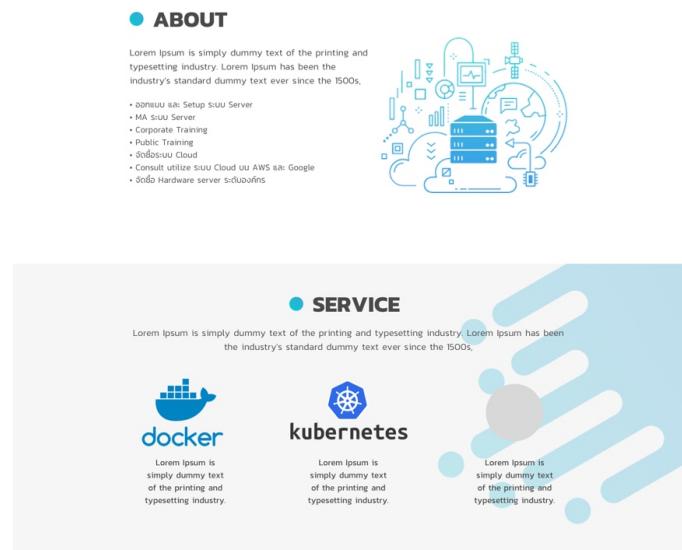
Name	Labels	Provisioner	Parameters	Age	⋮
standard	addonmanager.kuber...	k8s.io/minikube-hostpath	-	an hour	⋮



Who are we ? (Opcellent)



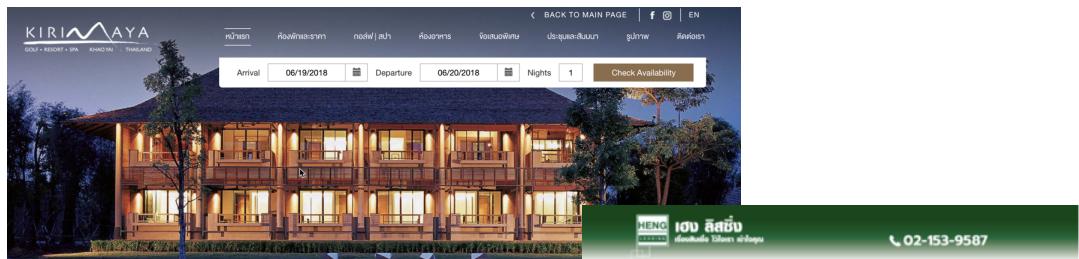
The homepage features a large blue abstract graphic at the top. Below it is a white header bar with the Opcellent logo and navigation links: About, Service, Training, Contact. A blue circular icon with a gear and arrows is on the left. To the right is a blue hand icon with lines radiating from it. Below the graphic, the text "Modern Server Technology Implementer" is centered. At the bottom is a "GET DETAILS" button.



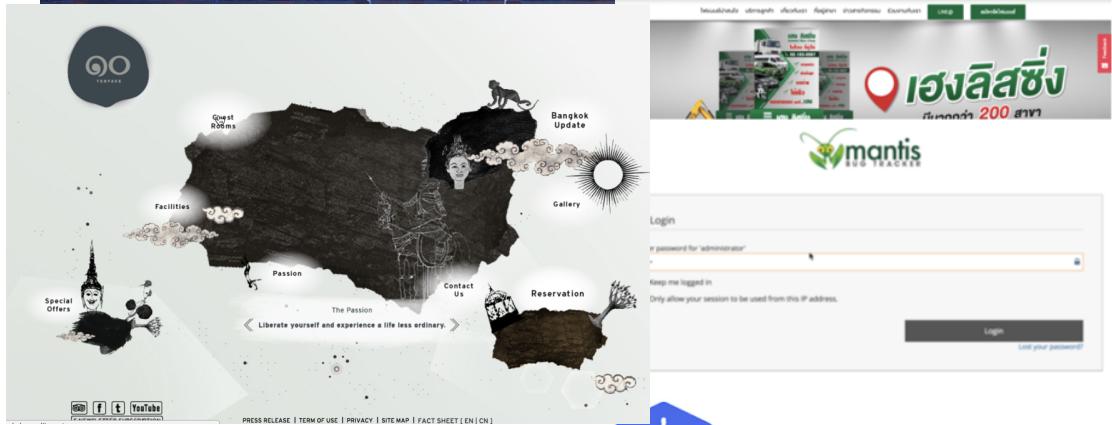
A service page section titled "SERVICE". It features three icons: Docker (blue ship), Kubernetes (blue steering wheel), and another blue hand icon. Each icon has a "Lorem Ipsum" placeholder text below it. The background is light blue with white abstract shapes.



A screenshot of an event listing. At the top are logos for HENG LEASING, GDG Chiang Mai, and ARTICAN DIGITAL. The event title is "Workshop : Advanced Docker and Kubernetes 101" by Praparn Lungpoonlap, System Engineer. It's scheduled for Saturday, 12 May 2018, at Punspace Tha Pae Gate. The schedule includes 6pm Register and Networking, 7pm Workshop, and 9pm Networking and Games. A note says "FREE FOOD AND BEER! - BUT PLEASE BRING YOUR OWN LAPTOP :)" and attendees should have basic knowledge or experience in Docker or container technology. Buttons for "Interested" and "Going" are shown.



A screenshot of a hotel booking interface for Kirinaya Golf & Resort Spa, Khao Yai, Thailand. The interface shows a night view of the resort buildings. It includes fields for Arrival (06/19/2018), Departure (06/20/2018), Nights (1), and a "Check Availability" button. There are also language and currency options at the top.



A screenshot of the Mantis Sod Tracker login interface. It features a map of Thailand with various landmarks labeled: Guest, R80ms, Bangkok Update, Gallery, Reservation, Passion, Contact Us, The Passion, Facilities, Special Offers, and Contact. To the right is a login form with fields for "username" (set to "administrator"), "password", and "Keep me logged in". Below the form is a note about session security. At the bottom are links for Press Release, Term of Use, Privacy, Site Map, and Fact Sheet (EN | CN).

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



Landscape of the world now

Cloud Native Landscape

v2.1

See the interactive landscape at landscape.cncf.io

Greyed logos are not open source

App Definition & Development



Database & Data Warehouse

Streaming

Source Code Management

Continuous Integration / Continuous Delivery (CI/CD)

Orchestration & Management



Scheduling & Orchestration



Coordination & Service Discovery



Service Management



Runtime



Cloud-Native Storage

Container Runtime

Cloud-Native Network

Provisioning



Host Management / Tooling



Infrastructure Automation

Container Registries

Secure Images

Key Management

Cloud



Public

Private



github.com/cncf/landscape

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



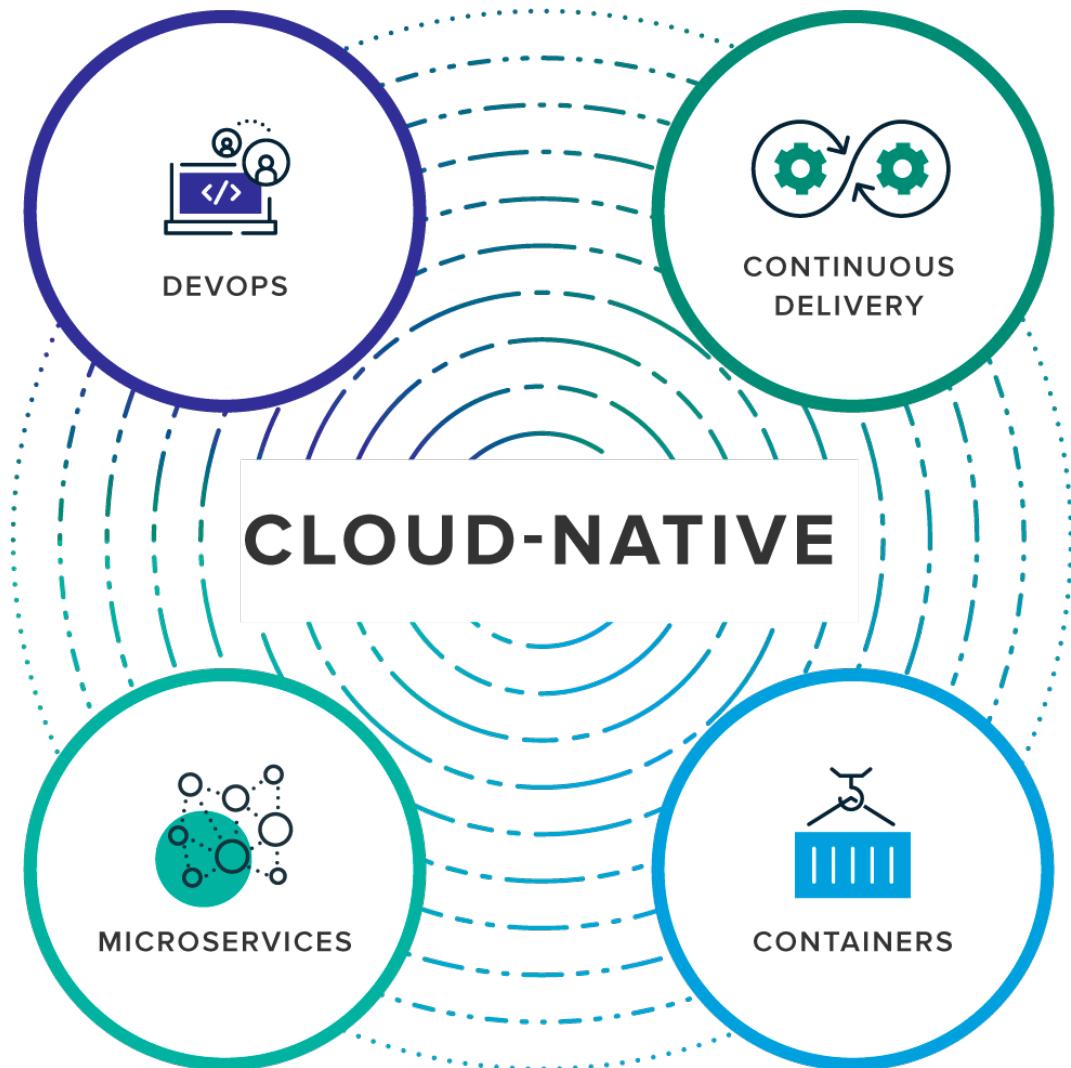
Special



kubernetes
by Google

Kubernetes: Production Workload Orchestration

Landscape of the world now



Highlight from DockerCon17 (EU)



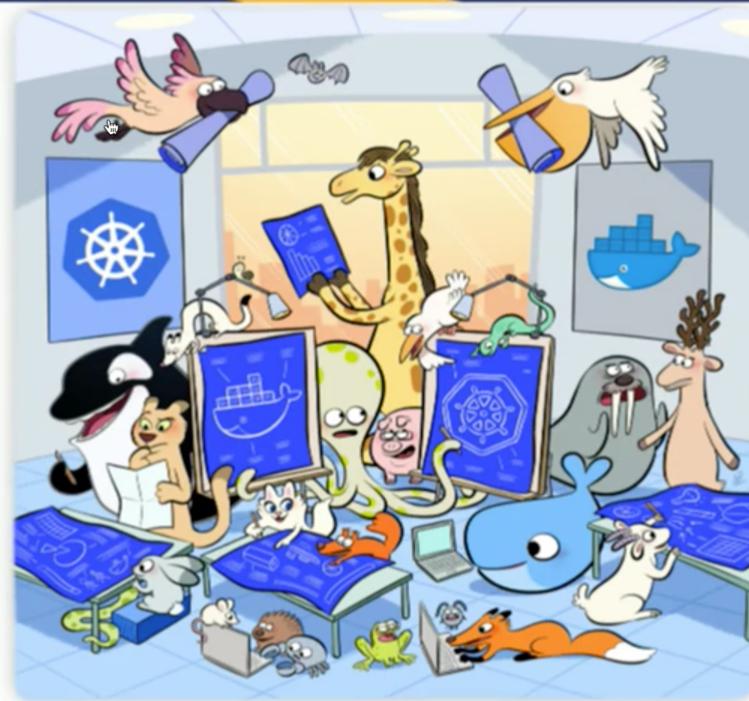
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now

WE ARE
ONE BIG
COMMUNITY



docker
con 17
EU

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now

Docker with Swarm and Kubernetes

1 →

The best enterprise container security and management

← 2

2

The best container development workflow

3 →

Native Kubernetes integration provides full ecosystem compatibility

← 4

4

Industry-standard container runtime



 dockercon¹⁷ EU

Kubernetes: Production Workload Orchestration



kubernetes
by Google

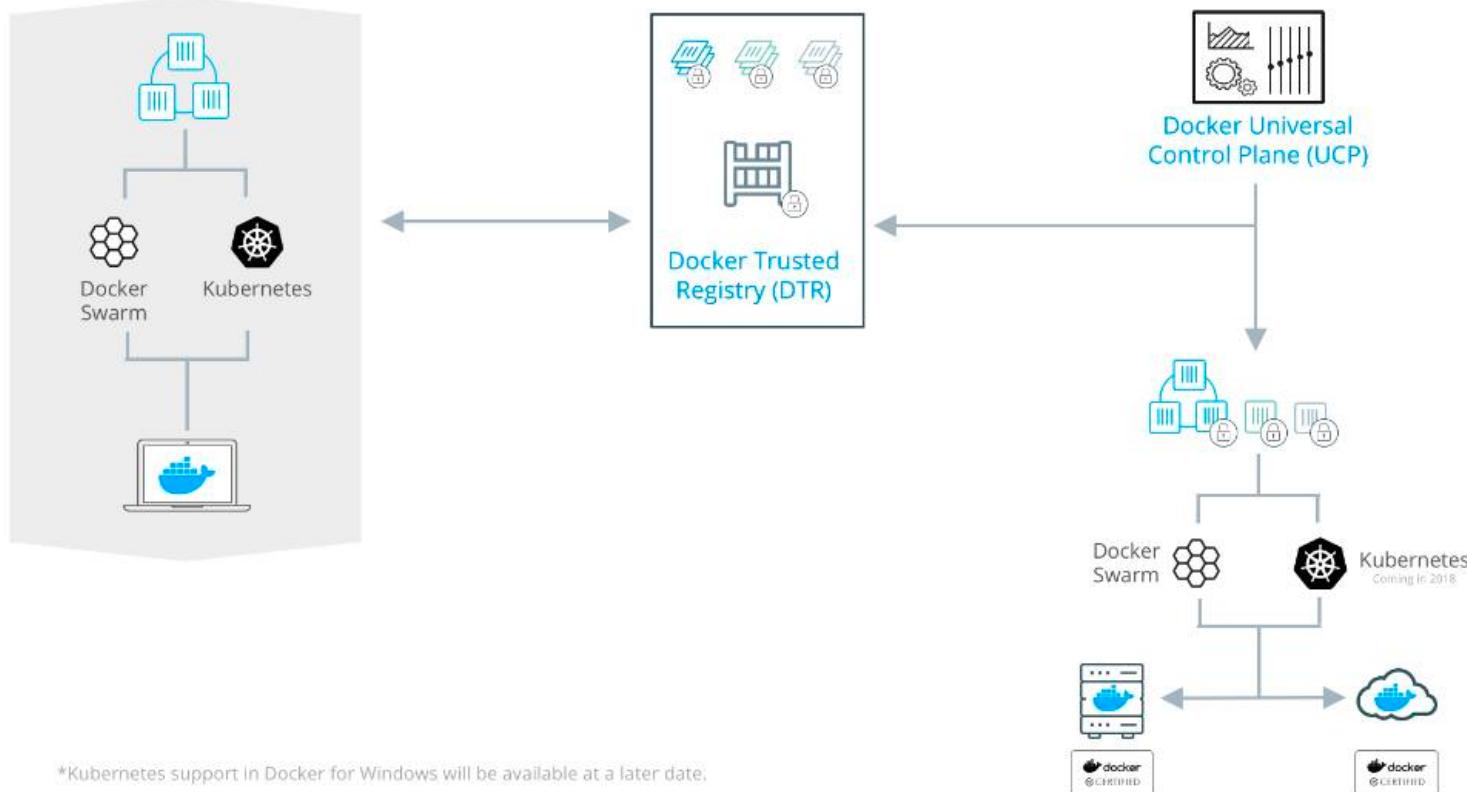
Landscape of the world now

Docker for Mac
Docker for Windows*

Build Test Ship

Docker
Enterprise Edition

Secure Deploy Manage



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now

Kubernetes Support in Docker
Enterprise Edition

Vivek Saraswat, Product Manager



<https://blog.docker.com/2018/01/docker-ee-kubernetes/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now

Slant ▾



Search or Ask a Question

Log In or Join Now

ASK QUESTION

Development

Backend Development

Sysadmin

Follow

...

What are the best Docker orchestration tools?

11

OPTIONS CONSIDERED

77

RECOMMENDATIONS

Mar 13, 2018

LAST UPDATED

THE BEST 1 OF 11 OPTIONS WHY?

11 Options Considered	Price	Last Updated
Kubernetes THE BEST		Mar 13, 2018
Docker Swarm		Dec 11, 2017
Docker Compose		Sep 23, 2017
Nomad		Sep 23, 2017
OpenShift		Jan 22, 2018

See Full List

RELATED QUESTIONS

[What are the best power user tools for macOS?](#)

[What are the best continuous integration tools?](#)

[What are the best developer tools for Mac OSX?](#)

[What are the best software tools for live streaming?](#)

[What are the best log aggregation & monitoring tools?](#)

[What are the best mockup and wireframing tools for websites?](#)

[What are the best diff tools for Git?](#)

[What are the best mind mapping tools?](#)

[What are the best 2D animation tools for game development?](#)

[What are the best power user tools for Windows?](#)

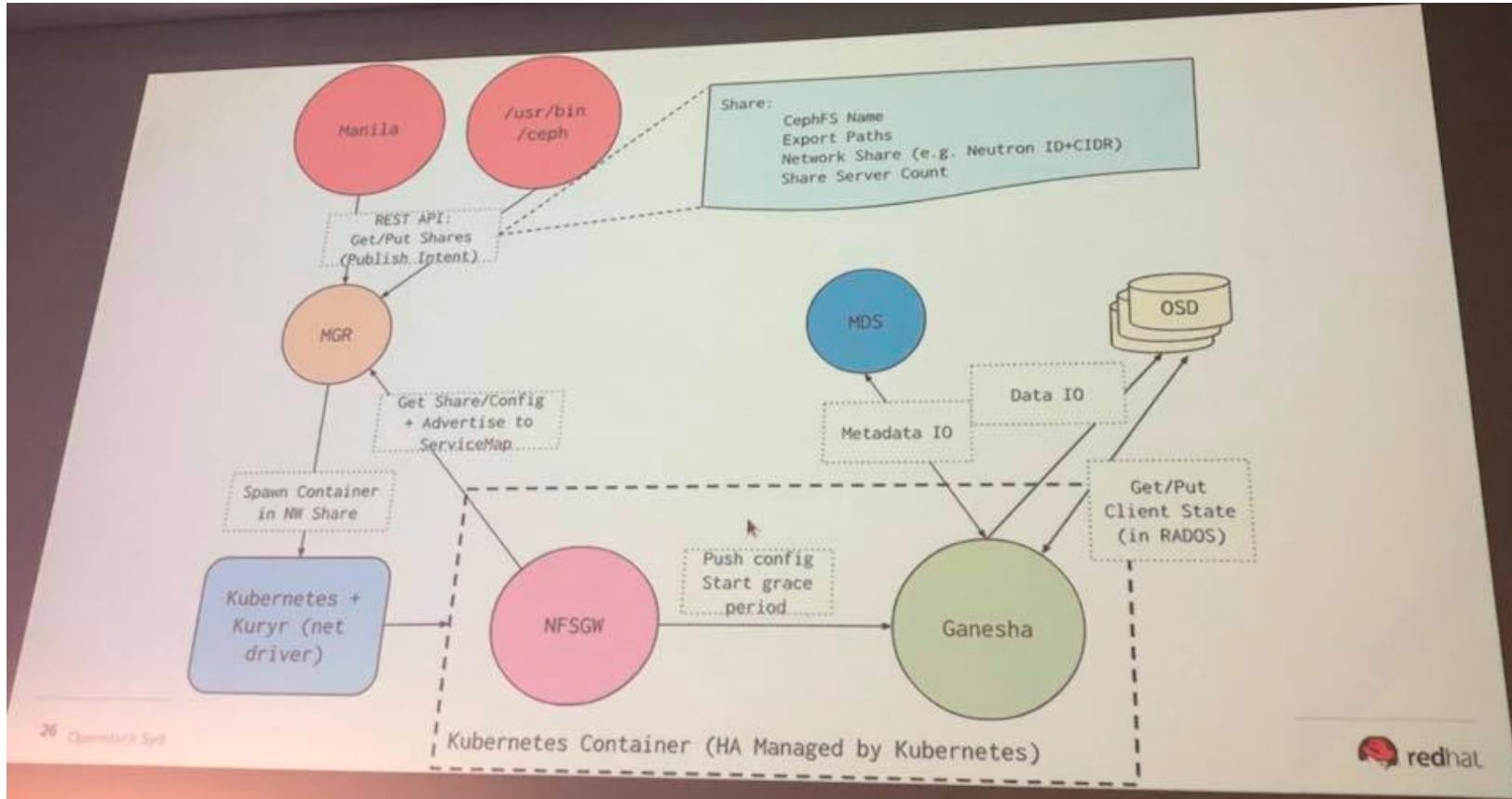
<https://www.slant.co/topics/3929/~docker-orchestration-tools>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now

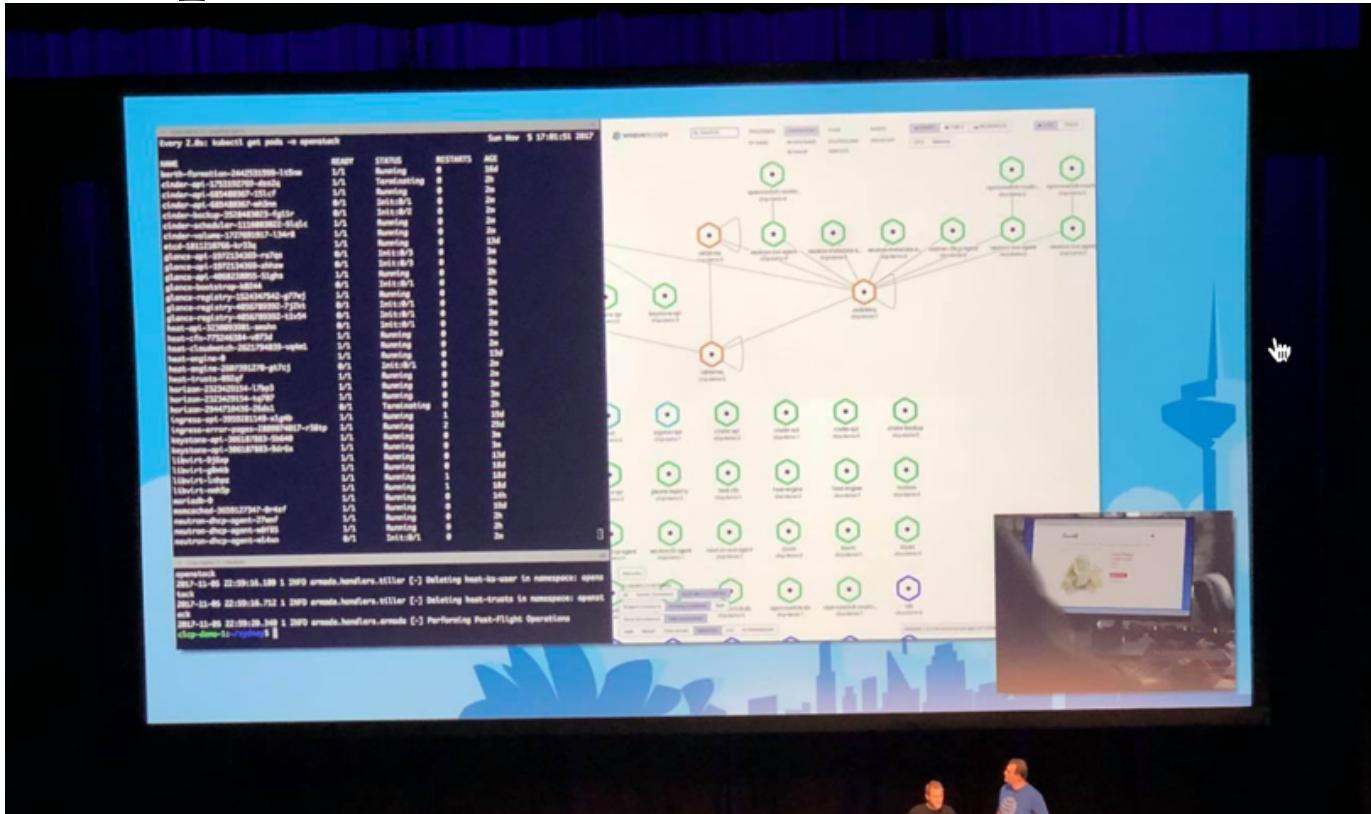


Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Landscape of the world now



The image shows a Red Hat press release page. At the top, there's a navigation bar with the Red Hat logo, followed by links for Technologies, Services & support, Success stories, and About Red Hat. Below the navigation is a section labeled "PRESS RELEASE". The main title of the article is "Red Hat Improves IT Flexibility and Reduces Complexity with Linux Containers in Latest Version of Production-Ready OpenStack Platform". A subtitle below the title reads "Red Hat OpenStack Platform 12 introduces containerized services and improves overall platform security". The background of the page features a dark, abstract graphic with red geometric patterns.



SYDNEY – OpenStack Summit Sydney 2017 – November 6, 2017 – Red Hat, Inc. (NYSE: RHT), the world's leading provider of open-source solutions, today announced Red Hat OpenStack Platform 12, the latest version of Red Hat's massively scalable and agile cloud Infrastructure-as-a-Service (IaaS). Based on the OpenStack "Pike" release, Red Hat OpenStack Platform 12 introduces containerized services, improving flexibility while decreasing complexity for faster application development. Red Hat OpenStack Platform 12 delivers many new enhancements, including upgraded DCI (distributed continuous integration) and improved security to help maintain data compliance and manage risk.

“To achieve the benefits offered by digital transformation, enterprises need to update their infrastructure to better support the next-generation of applications that take

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Container Principle

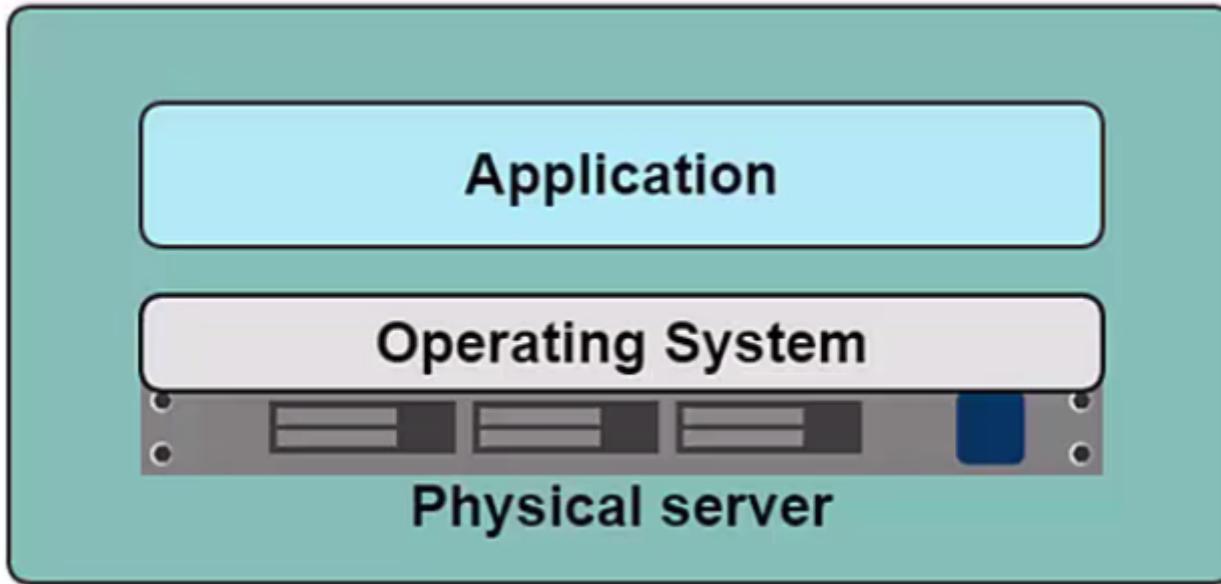


Kubernetes: Production Workload Orchestration



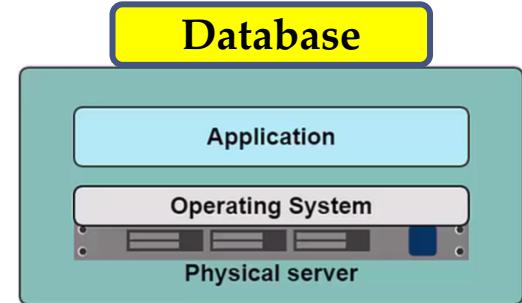
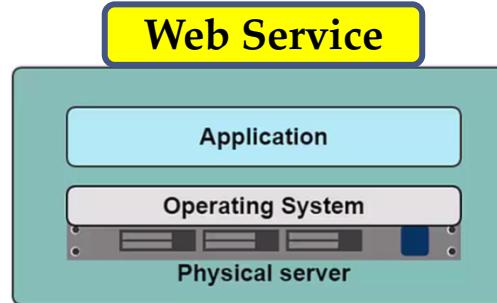
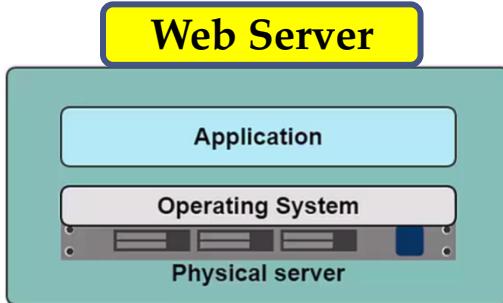
kubernetes
by Google

Container Principle



Existing Technology

- Production Environment (Best design)
- Day 1: Application 1: Implement



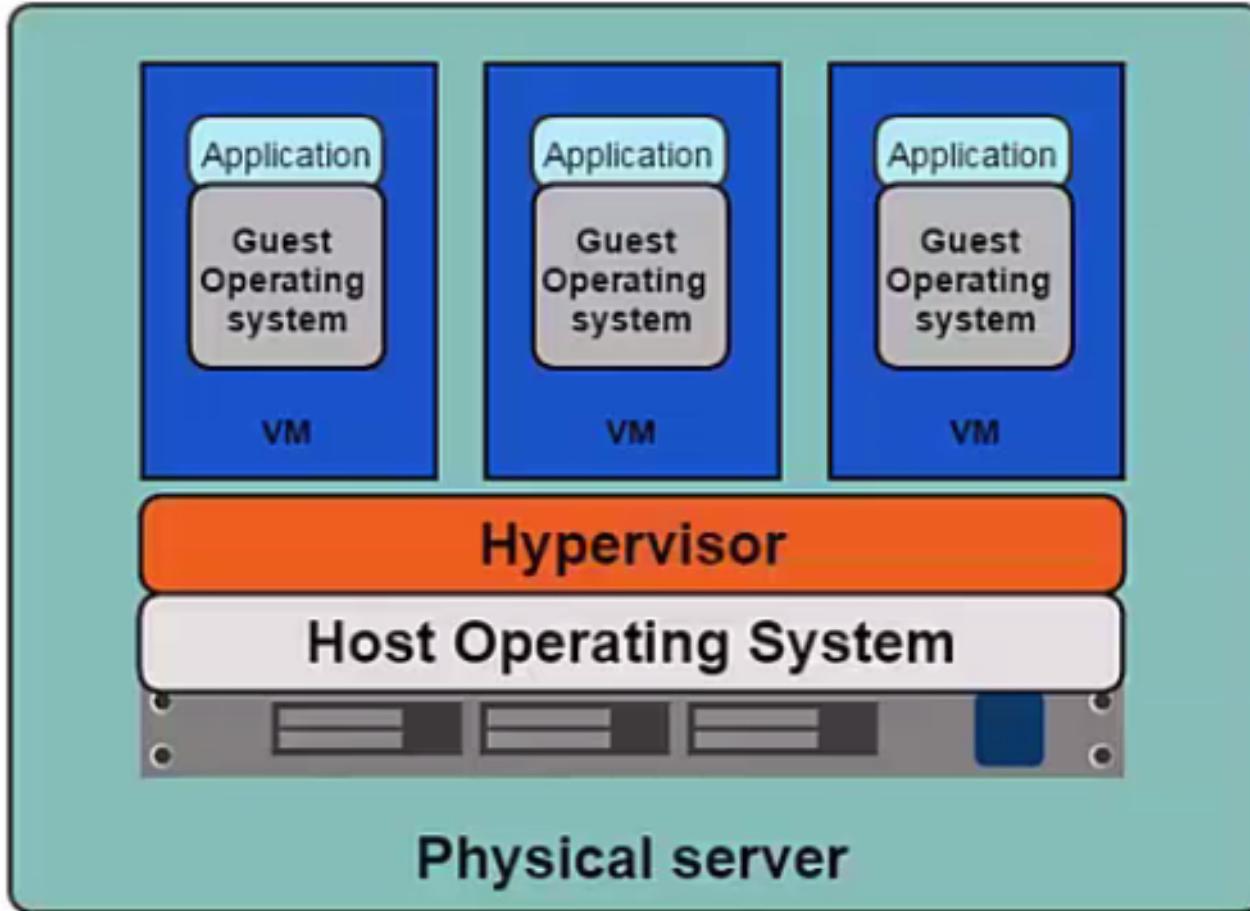
- Apache 2.20 Web Server
- PHP 5.5 Engine
- Laravel 4.1 Framework

- IIS 8
- .Net Framework 3.5

- MariaDB 5.1

- Day 2: Application 2: Need to implement
 - Need PHP 7.0 ?
 - MariaDB 10.1.14 (Need search feature on 10.1)
- Problem ?
 - Possible to upgrade PHP to 7.0 ? / How to test existing application ?
 - What effect to MariaDB upgrade ?

Container Principle



Container Principle

- Production Environment
- Day 1: Application 1: Implement



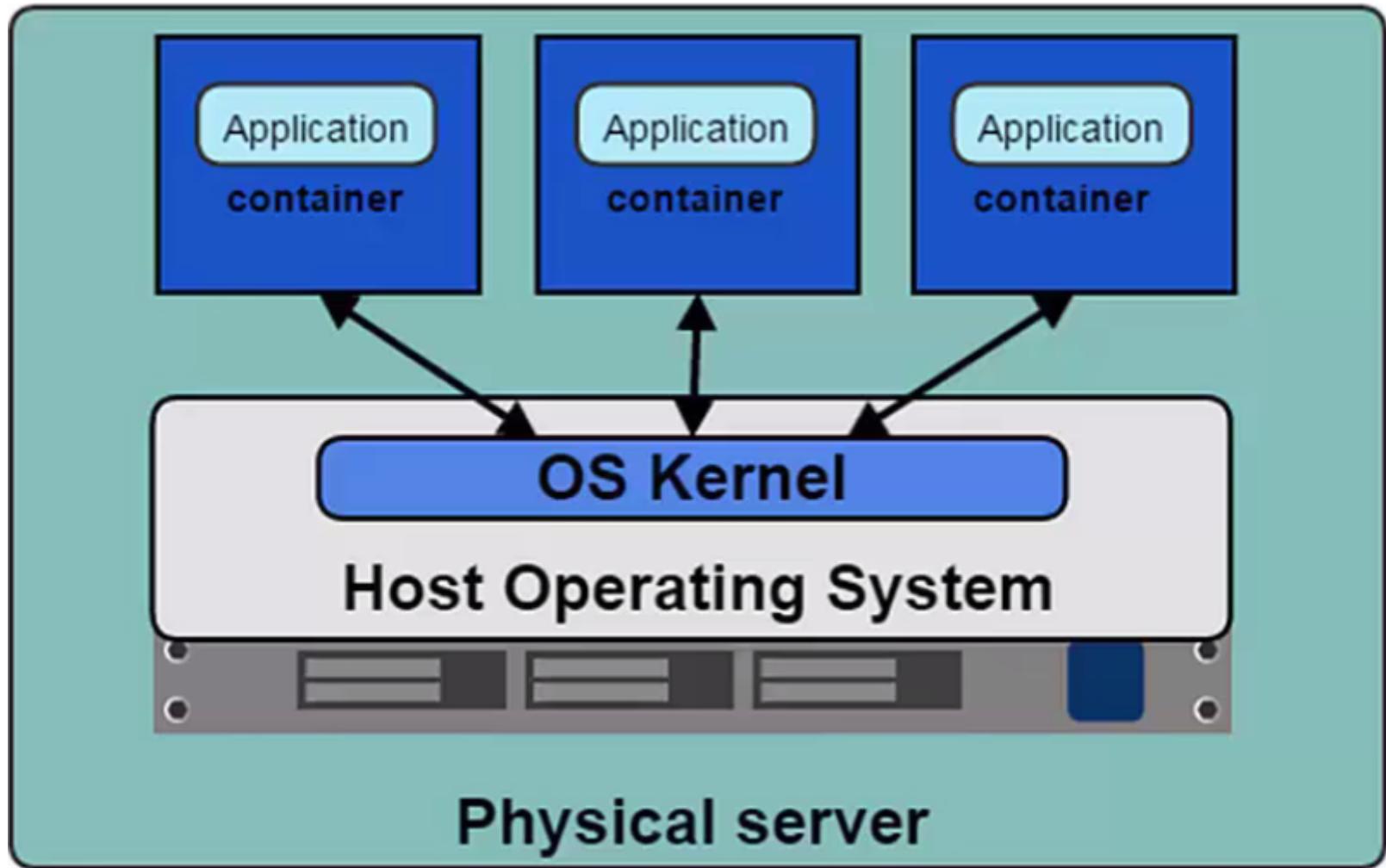
- Apache 2.20 Web Server
- PHP 5.5 Engine
- Laravel 4.1 Framework

- IIS 8
- .Net FrameWork 3.5

- MariaDB 5.1

- Day 2: Application 2: Need to implement
 - Need PHP 7.0 ?
 - MariaDB 10.1.14 (Need search feature on 10.1)
- So... The problem still exist.

Container Principle



Container Principle

Mechanism	Operating system	License	Available since/between	Features									
				File system isolation	Copy on Write	Disk quotas	I/O rate limiting	Memory limits	CPU quotas	Network isolation	Nested virtualization	Partition checkpointing and live migration	Root privilege isolation
chroot	most UNIX-like operating systems	varies by operating system	1982	Partial ^[3]	No	No	No	No	No	No	Yes	No	No
Docker Linux-VServer (security context)	Linux ^[7]	Apache License 2.0	2013	Yes	Yes	Not directly	Not directly	Yes	Yes	Yes	Yes	No	No
	Linux	GNU GPLv2	2001	Yes	Yes	Yes	Yes ^[8]	Yes	Yes	Partial ^[9]	?	No	Partial ^[9]
lxc LXC	Linux	Apache License 2.0	2013	Yes	Yes	Yes	Yes ^[9]	Yes	Yes	Partial ^[9]	?	No	Partial ^[9]
	Linux	GNU GPLv2	2008	Yes ^[9]	Yes	Partial ^[9]	Partial ^[9]	Yes	Yes	Yes	Yes	No	Yes ^[9]
LXD	Linux	Apache License 2.0	2015	Yes	Yes	Partial(see LXC)	Partial(see LXC)	Yes	Yes	Yes	Yes	Partial ^[9]	Yes
OpenVZ	Linux	GNU GPLv2	2005	Yes	No	Yes	Yes ^[10]	Yes	Yes	Yes ^[11]	Partial ^[10]	Yes	Yes ^[10]
Virtuozzo	Linux, Windows	Proprietary	2000 ^[14]	Yes	Yes	Yes	Yes ^[11]	Yes	Yes	Yes ^[11]	Partial ^[15]	Yes	Yes
Solaris Containers (Zones)	illumos (OpenSolaris), Solaris	CDDL, Proprietary	2004	Yes	Yes (ZFS)	Yes	Partial ^[16]	Yes	Yes	Yes ^{[17][18]}	Partial ^[16]	Partial ^[17]	Yes ^[8]
FreeBSD jail	FreeBSD	BSD License	2000 ^[20]	Yes	Yes (ZFS)	Yes ^[19]	No	Yes ^[21]	Yes	Yes ^[22]	Yes	No	Yes ^[23]
sysjail	OpenBSD, NetBSD	BSD License	2006–2009 (As of March 3, 2009, it is no longer supported)	Yes	No	No	No	No	No	Yes	No	No	?
WPARs	AIX	Proprietary	2007	Yes	No	Yes	Yes	Yes	Yes	Yes ^[24]	No	Yes ^[25]	?
HP-UX Containers (SRP) ^[9]	HPUX	Proprietary	2007	Yes	No	Partial ^[16]	Yes	Yes	Yes	Yes	?	Yes	?
iCore Virtual Accounts	Windows XP	Proprietary/Freeware	2008	Yes	No	Yes	No	No	No	No	?	No	?
Sandboxie Spoon	Windows	Proprietary/Shareware	2004	Yes	Yes	Partial	No	No	No	Partial	No	No	Yes
	Windows	Proprietary	2012	Yes	Yes	No	No	No	No	Yes	No	No	Yes
VMware ThinApp	Windows	Proprietary	2008	Yes	Yes	No	No	No	No	Yes	No	No	Yes

Reference: https://en.wikipedia.org/wiki/Operating-system-level_virtualization



Container Principle



Docker Enterprise Edition (EE) and Community Edition (CE)

Enterprise Edition (EE)

- CaaS enabled platform subscription (integrated container orchestration, management and security)
- Enterprise class support
- Quarterly releases, supported for one year each with backported patches and hotfixes.
- Certified Infrastructure, Plugins, Containers

Community Edition (CE)

- Free Docker platform for "do it yourself" dev and ops
- Monthly Edge release with latest features for developers
- Quarterly release with maintenance for ops

Lifecycle

Squaring the circle: Faster releases and better stability



Docker EE Availability

From Docker



OEM: Direct L2 / L2 Support Included



Cloud Marketplaces



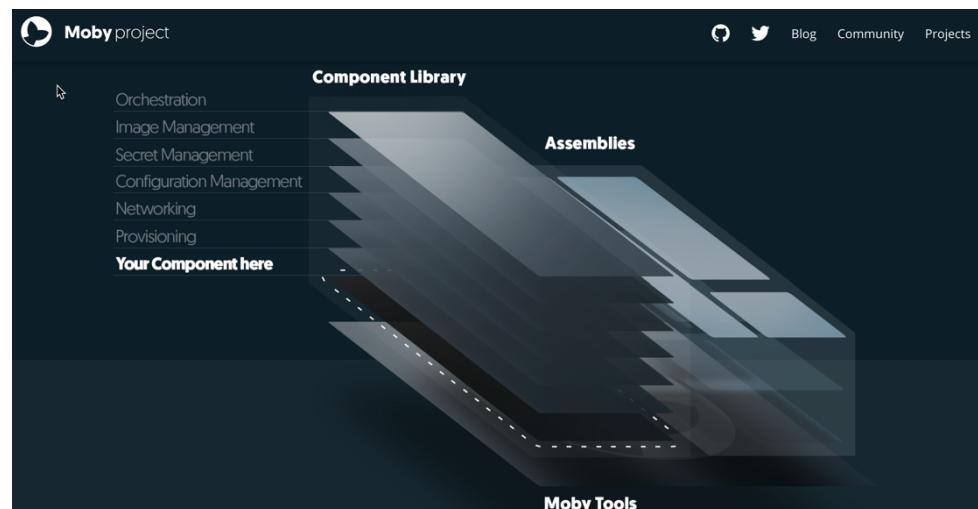
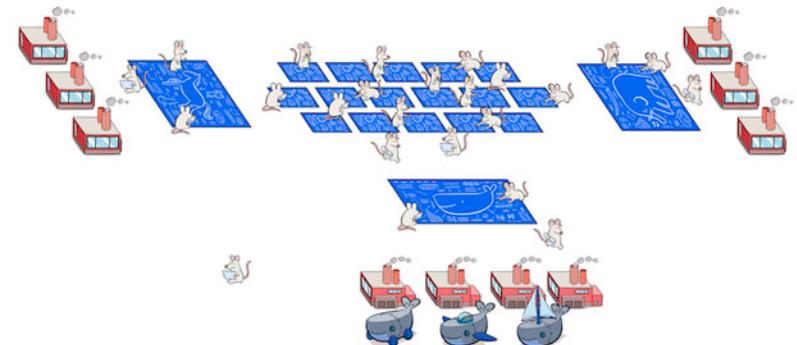
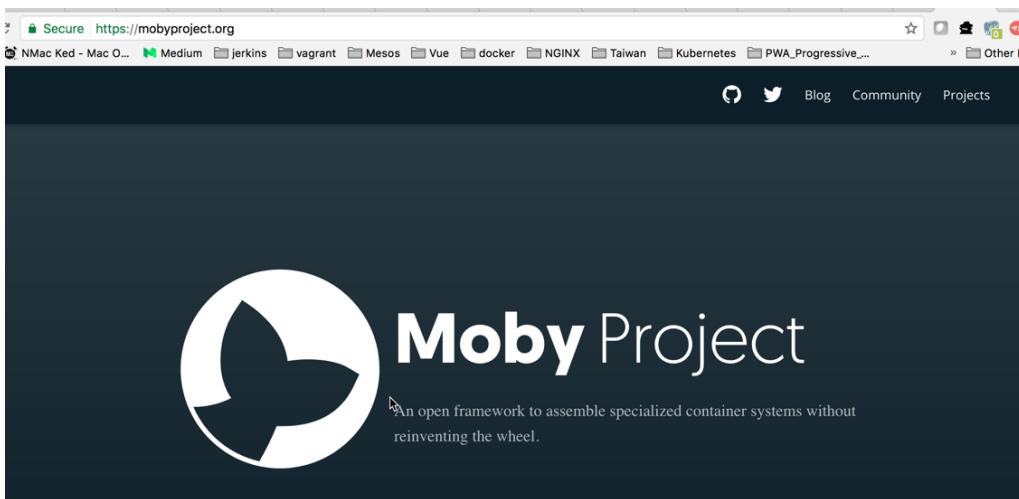
Ref: <https://blog.docker.com/2017/03/docker-online-meetup-recap-docker-enterprise-edition-ee-community-edition-ce/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Container Principle



Kubernetes: Production Workload Orchestration



kubernetes
by Google

Container Principle

The screenshot shows the official website for rkt. At the top is a dark blue header with the rkt logo and the text "A security-minded, standards-based container engine". Below the header is a navigation bar with three items: "Overview" (selected), "Documentation", and "GitHub Project". The main content area has a heading "Overview" followed by a detailed description of rkt's features and architecture.

Overview

rkt is an application container engine developed for modern production cloud-native environments. It features a pod-native approach, a pluggable execution environment, and a well-defined surface area that makes it ideal for integration with other systems.

The core execution unit of rkt is the *pod*, a collection of one or more applications executing in a shared context (rkt's pods are synonymous with [the concept in the Kubernetes orchestration system](#)). rkt allows users to apply different configurations (like isolation parameters) at both pod-level and at the more granular per-application level. rkt's architecture means that each pod executes directly in the classic Unix process model (i.e. there is no central daemon), in a self-contained, isolated environment. rkt implements a modern, open, standard container format, the App Container (appc) spec, but can also execute other container images, like those created with Docker.

Since its introduction by CoreOS in December 2014, the rkt project has greatly matured and is widely used in production environments.

THE LATEST ON RKT

- [CoreOS's rkt announcement to CNCF](#)
- [What Kubernetes container engine?](#)

MORE INFORMATION

[Download](#)



CoreOS teams with Intel to make
Rocket containers more secure

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes



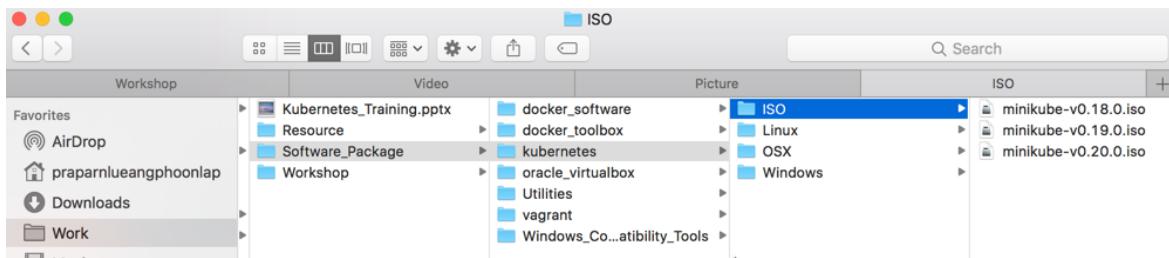
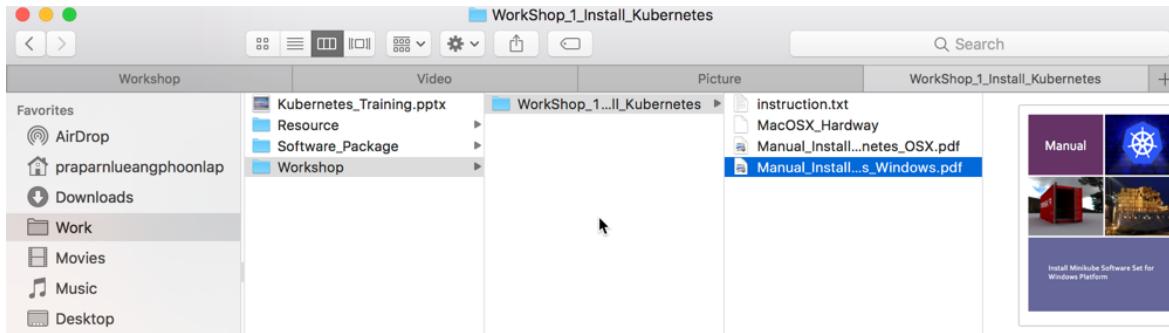
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Workshop Day1: Unit 1

- Install lab prerequisite and minikube



Workshop Day1: Unit 1

- Alternate Solution:

The screenshot shows the Play with Kubernetes (PWK) web interface. On the left, there's a landing page with a large blue Kubernetes logo and the text "Play with Kubernetes". It describes PWK as a simple, interactive and fun playground to learn Kubernetes. It includes a "Login" button and a note about the project's creators and sponsors.

On the right, a browser window displays a session dashboard. The URL is https://labs.play-with-k8s.com/p/b9moh1tduh0g00e4uge0#b9moh1td_b9moh35duh0g00e4ugeg. The dashboard shows a timer at 03:59:50, a "CLOSE SESSION" button, and sections for "Instances" and "Memory". An instance named "node1" is listed with IP 192.168.0.18 and memory usage 0.77% (30.64MiB / 3.906GiB). There are "DELETE" and "+ ADD NEW INSTANCE" buttons.

The main content area contains a terminal window with the following text:

```
WARNING!!!
This is a sandbox environment. Using personal credentials
is HIGHLY! discouraged. Any consequences of doing so, are
completely the user's responsibilites.

You can bootstrap a cluster as follows:

1. Initializes cluster master node:
   kubeadm init --apiserver-advertise-address=$(hostname -i)

2. Initialize cluster networking:
   kubectl apply -n kube-system -f \
     "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

3. (Optional) Create an nginx deployment:
   kubectl apply -f https://k8s.io/docs/user-guide//nginx-app.yaml

The PWK team.

[node1 ~] $
```



Workshop Day1: Unit 1

- 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 navigation bar is the Katacoda logo and the title "Kubernetes Playground".

Launch Cluster

A "launch.sh" button is highlighted. Below it, a message says: "This will create a two node Kubernetes cluster using WeaveNet for networking."

Health Check

A "kubectl cluster-info" button is highlighted. Below it, a message says: "Interested in writing your own Kubernetes scenarios and demos? Visit www.katacoda.com/teach".

SUMMARY

Terminal Host 1
Your Interactive Bash Terminal.

```
master $ launch.sh
Waiting for Kubernetes to start...
Kubernetes started
master $ kubectl get nodes
NAME      STATUS    ROLES      AGE       VERSION
master    Ready     master    23m      v1.8.0
node01   Ready     <none>   23m      v1.8.5
master $
```

Terminal Host 2
Your Interactive Bash Terminal.

```
node01 $ launch.sh
launch.sh: command not found
node01 $
```



Introduction to Kubernetes

- Check kubernetes version

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

```
[praparns-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

THE LINUX FOUNDATION PROJECTS



About ▾ Projects ▾ People ▾ Community ▾ Newsroom ▾



NOW AVAILABLE: SELF-PACED CLASS KUBERNETES FUNDAMENTALS

Learn how to deploy a containerized application and manipulate resources via the API.

REGISTER NOW

CURRENTLY HOSTED PROJECTS



Prometheus



OPENTRACING



fluentd



linkerd



gRPC



CoreDNS

containerd



rkt



CNI

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))

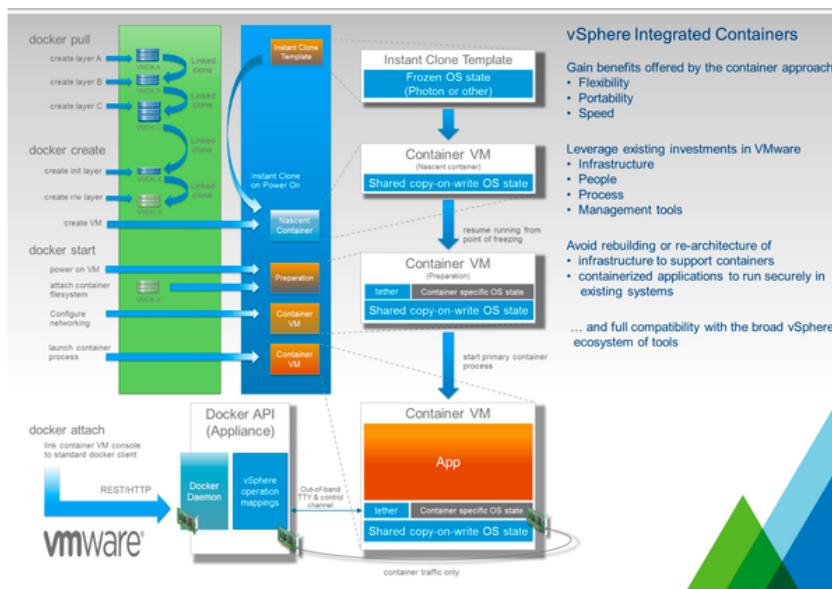
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes

- But why Container Orchestration ?
 - Container Platform is next generation that partner/beat existing virtualize technology
 - Easy to build-ship-run on dev and production
 - Zero Configure
 - Suitable for Microservice architecture
 - Good for application, Good for business (Win/Win !!!)



Introduction to Kubernetes

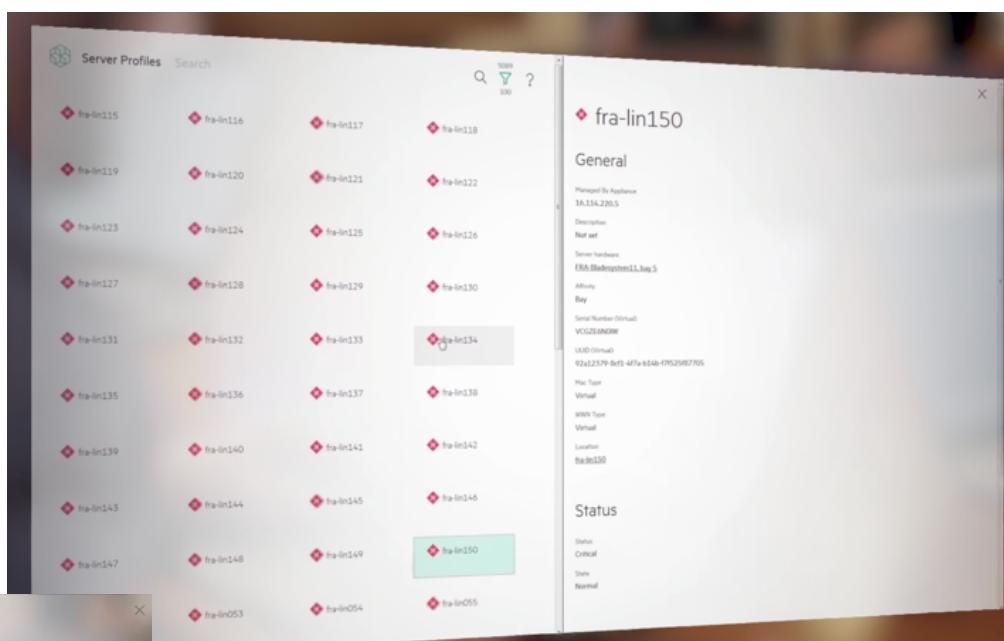
How HPE Synergy works

Explore the Engine of the Idea Economy

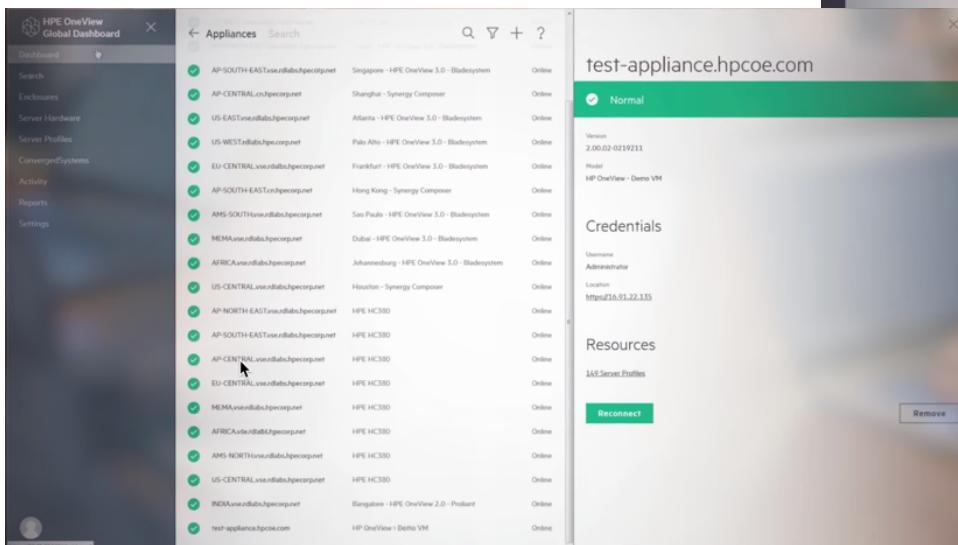
Imagine infrastructure so intelligent, it can adapt to every workload and deliver services at lightning speed.



Introduction to HPE Synergy Composable Infrastructure Software-defined Intelligence Pooling and Provisioning Simple Scalability Open Integration



The screenshot shows the HPE OneView interface. On the left, a list of server profiles is displayed, each with a red diamond icon and a name like 'fra-lin115' through 'fra-lin150'. On the right, a detailed view for 'fra-lin150' is shown under the 'General' tab, listing managed by Appliance ID, description, server hardware, affinity, bay, serial number, UUID, mac type, WWN type, and location. Below that is the 'Status' tab, which shows status, serial, and store information.



The screenshot shows the HPE OneView interface. On the left, a list of appliances is shown, each with a green checkmark icon and a name like 'AP-SOUTH-EASTus.rblabs.hpecorp.net'. On the right, a detailed view for 'test-appliance.hpcoc.com' is shown under the 'Normal' tab, displaying version 2.00.02-019211, model 'HP OneView - Demo VM', credentials (username: Administrator), and resources (149 Server Profiles). A 'Reconnect' button is visible at the bottom.

Your infrastructure as code

Gain efficiency and control with HPE Synergy, a powerful software-defined solution that lets you manage your infrastructure as code, deploying IT resources quickly and for any workload. Through a single interface, you can compose fluid pools of physical and virtual compute, storage, and fabric resources into any configuration for any application. With HPE Synergy, IT goes beyond the role of internal service broker to offer new applications that elevate and enhance the business.

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

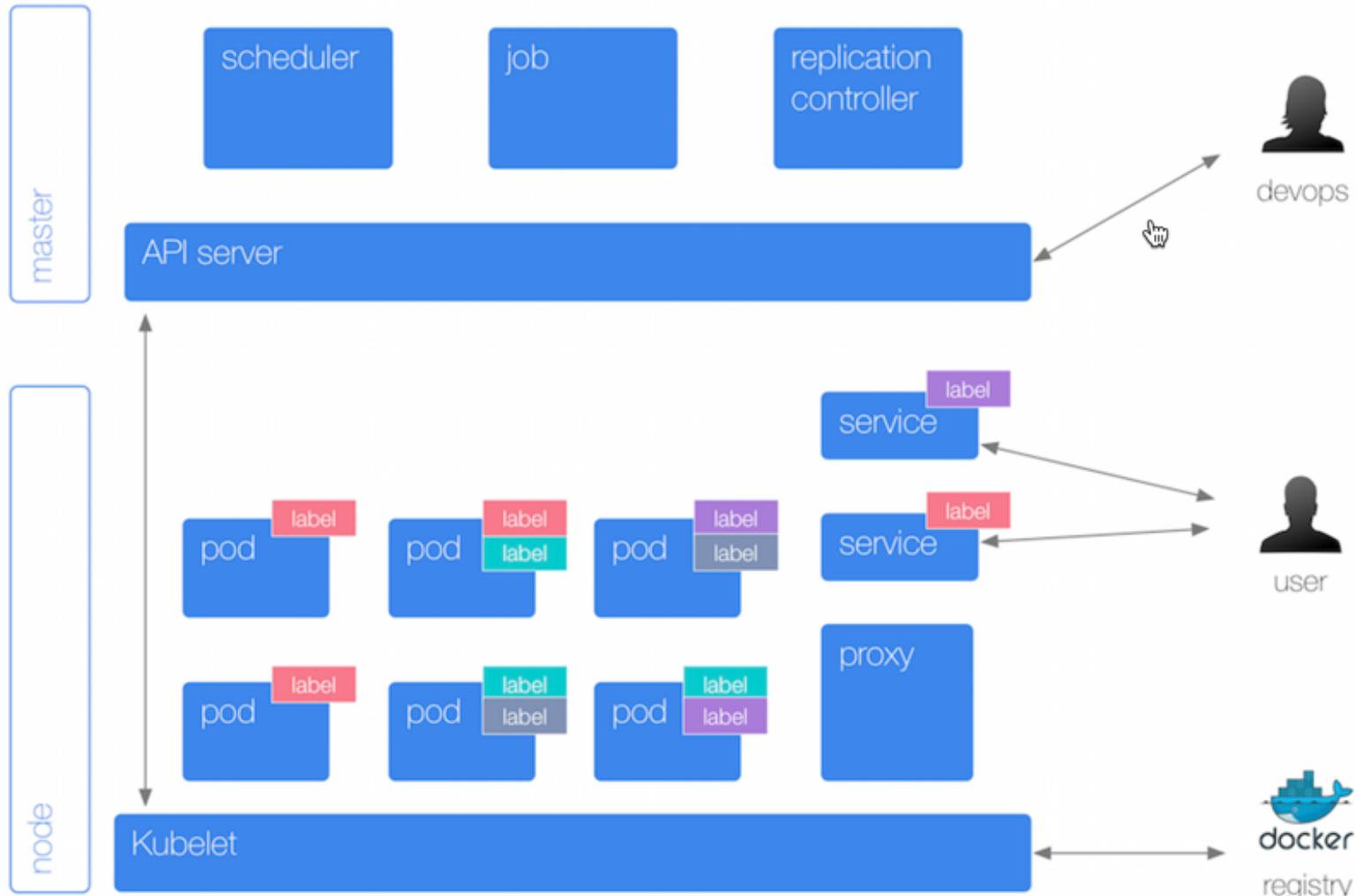


Kubernetes: Production Workload Orchestration



kubernetes
by Google

Introduction to Kubernetes



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 node

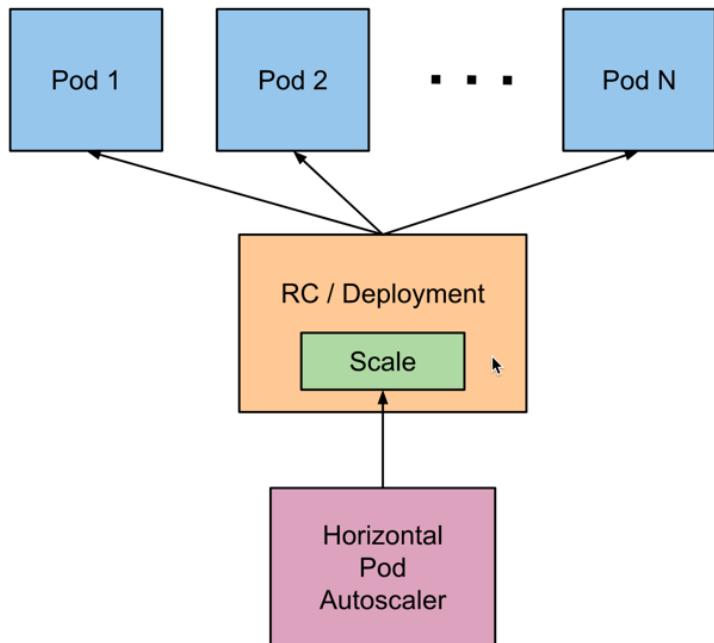
Non-terminated Pods: (3 in total)					
Namespace	Name	CPU Requests	CPU Limits	Memory Requests	Memory Limits
kube-system	kube-addon-manager-minikube	5m (0%)	0 (0%)	50Mi (2%)	0 (0%)
kube-system	kube-dns-268032401-mx7s3	260m (13%)	0 (0%)	110Mi (5%)	170Mi (8%)
kube-system	kubernetes-dashboard-hdhrz	0 (0%)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



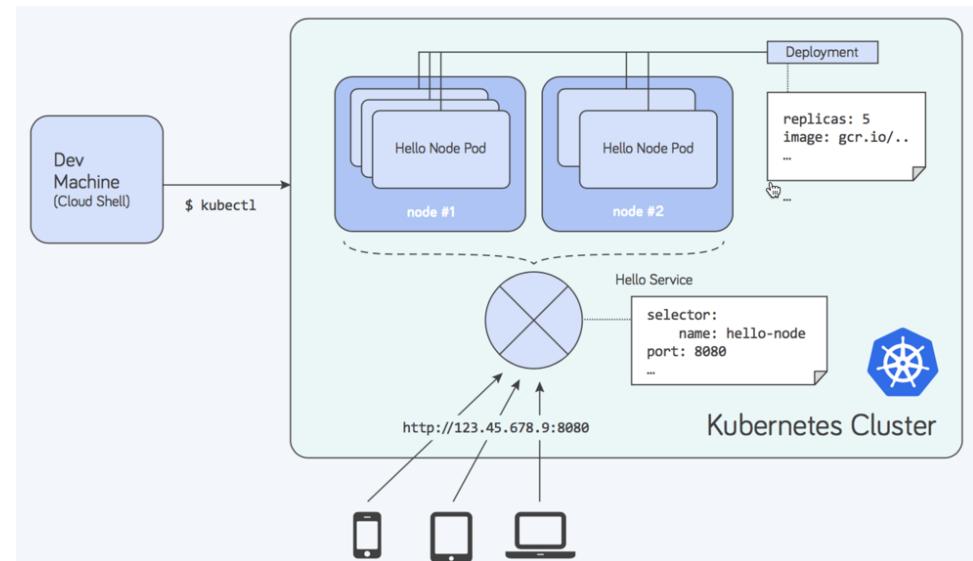
Introduction to Kubernetes

- Horizon Pods Autoscaling (hpa)
 - 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)



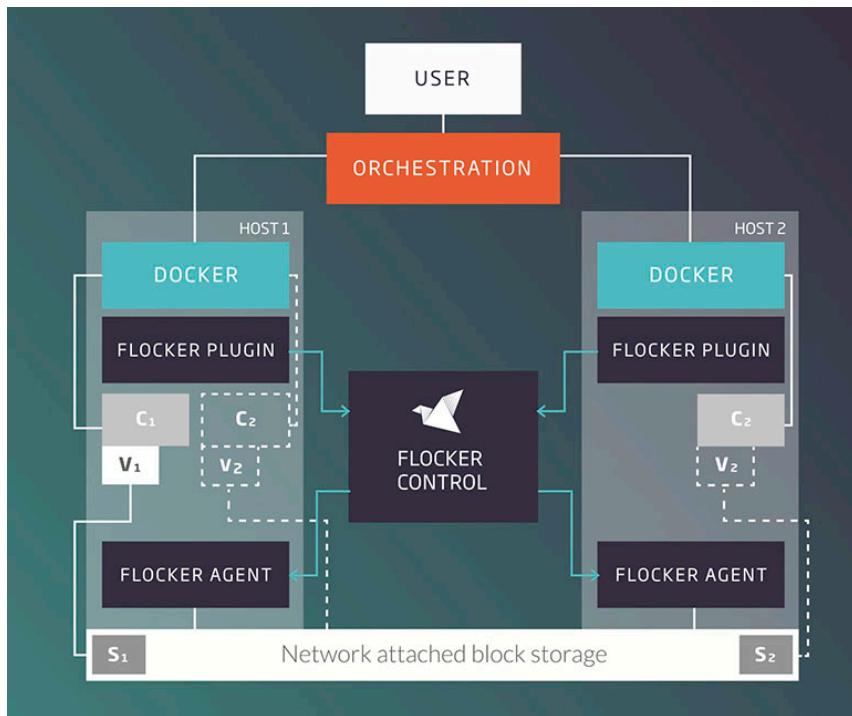
Introduction to Kubernetes

- 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)



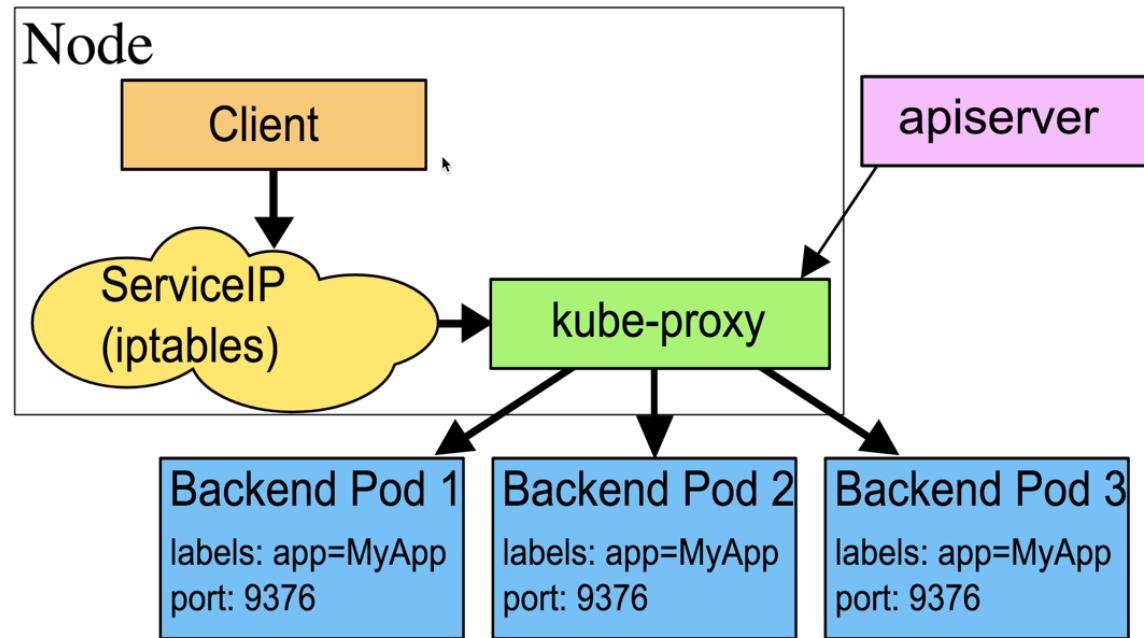
Introduction to Kubernetes

- Self-healing
 - Replication Controller (RC) will maintain unit of Pods as design (not to much (kill) and not to few (create))
 - Full-fill 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>

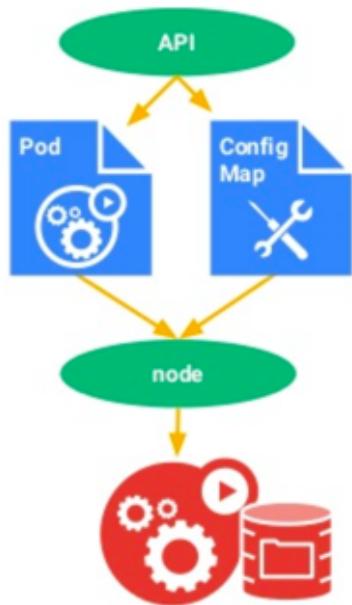
Type:         Opaque

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



Introduction to Kubernetes

- Secret and Configuration Management
 - Sometime we have a many configuration that need to specify on each application, But it should be change every time need.
 - Idea is create 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

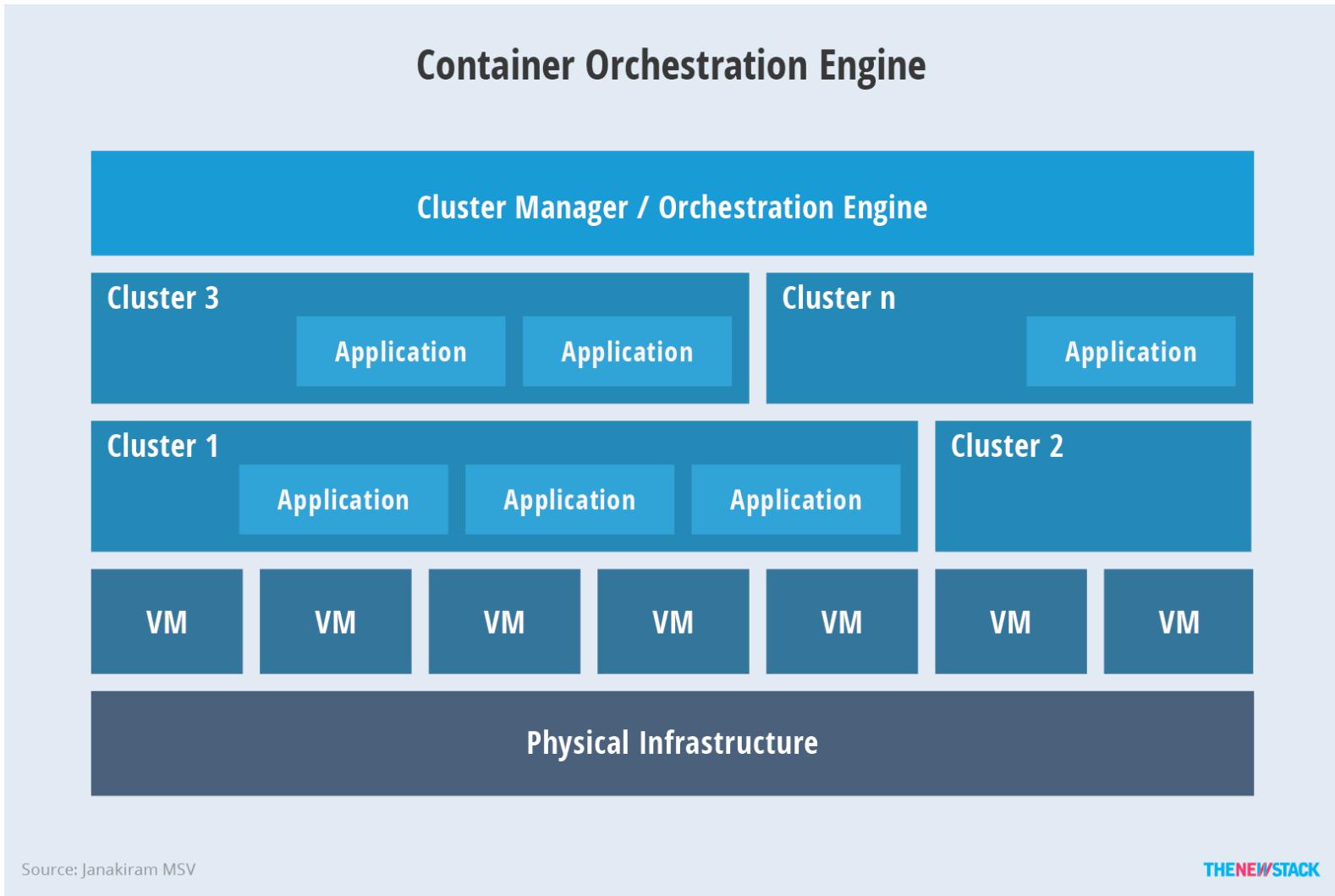


Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



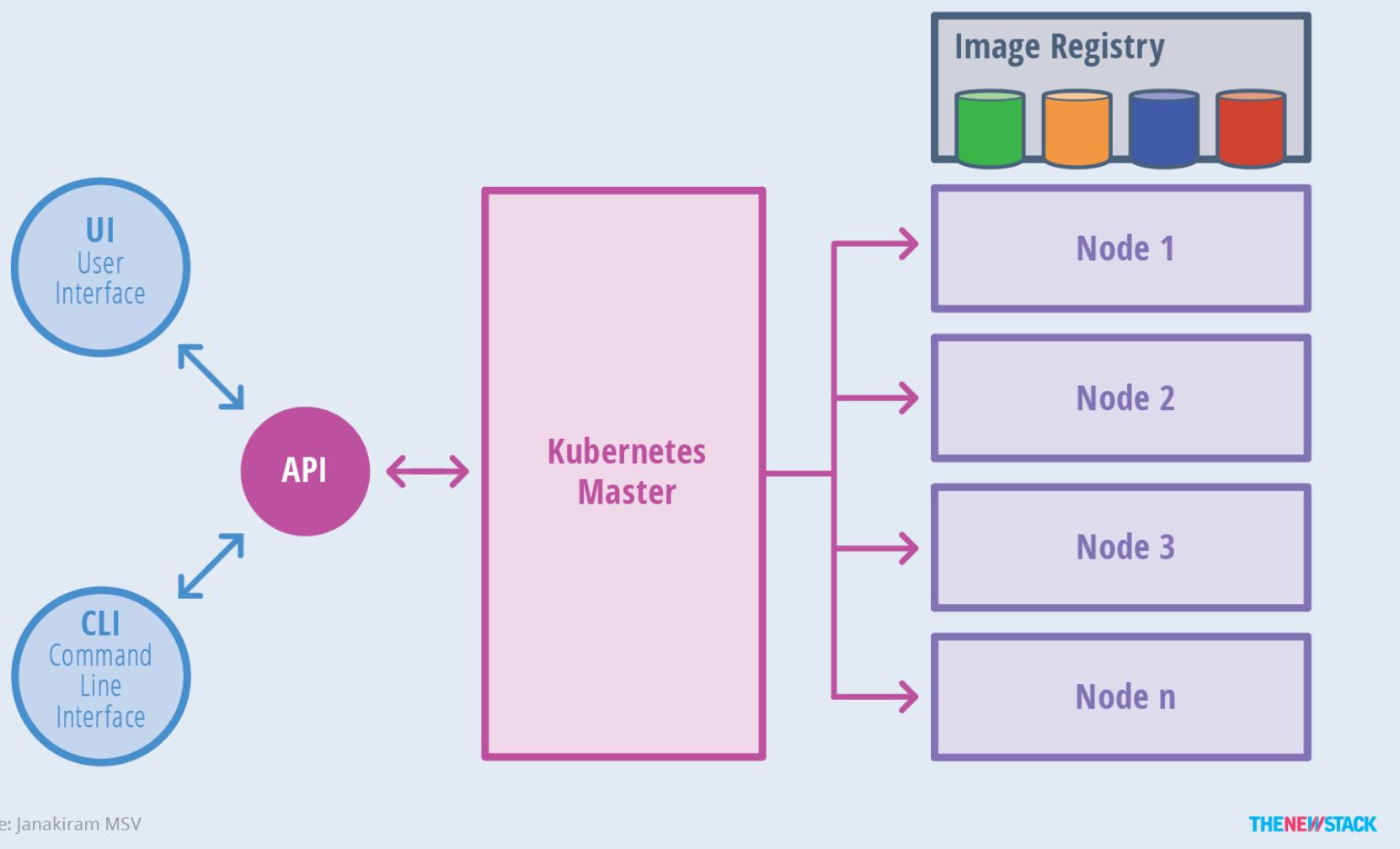
Ref: https://cdn.thenewstack.io/media/2016/11/Chart_02_Kubernetes-Architecture.png

Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



Source: Janakiram MSV

THE NEW STACK

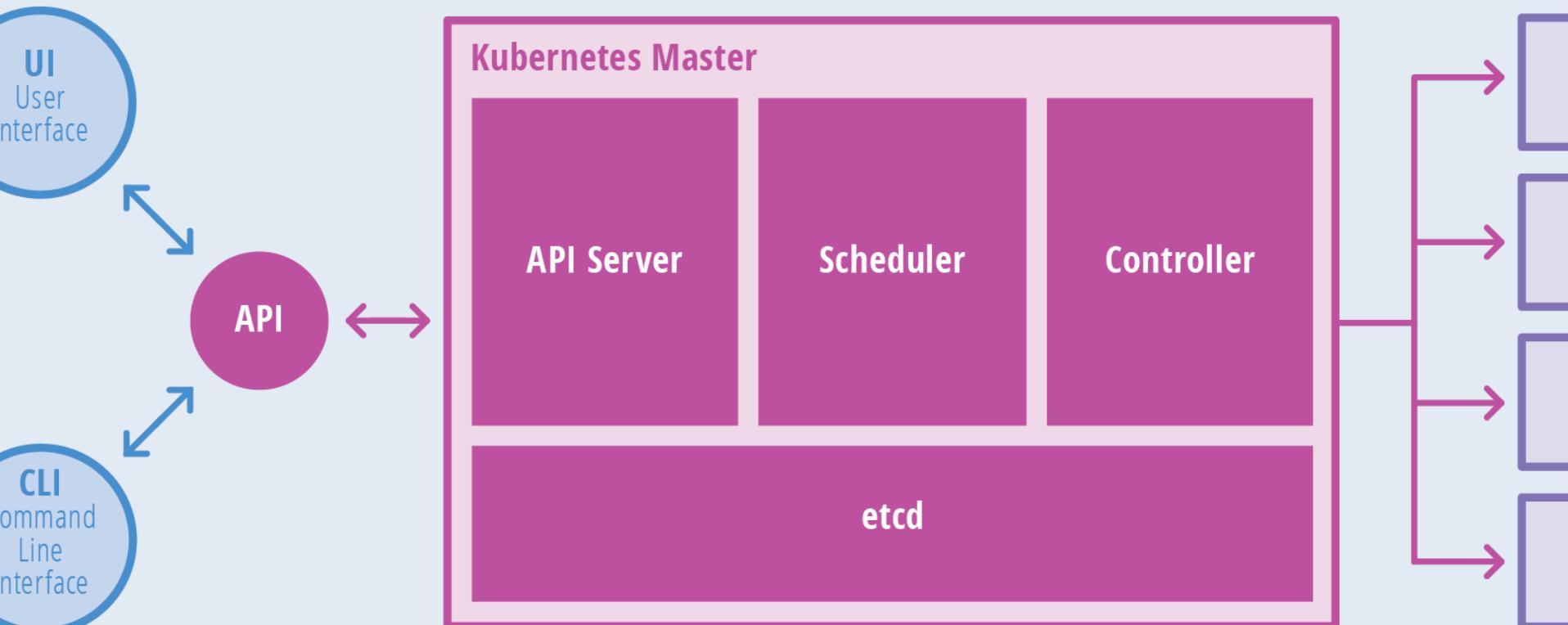
Ref: https://cdn.thenewstack.io/media/2016/11/Chart_02_Kubernetes-Architecture.png

Kubernetes: Production Workload Orchestration



kubernetes
by Google

System Architecture



Source: Janakiram MSV

THE NEW STACK

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

Kubernetes: Production Workload Orchestration



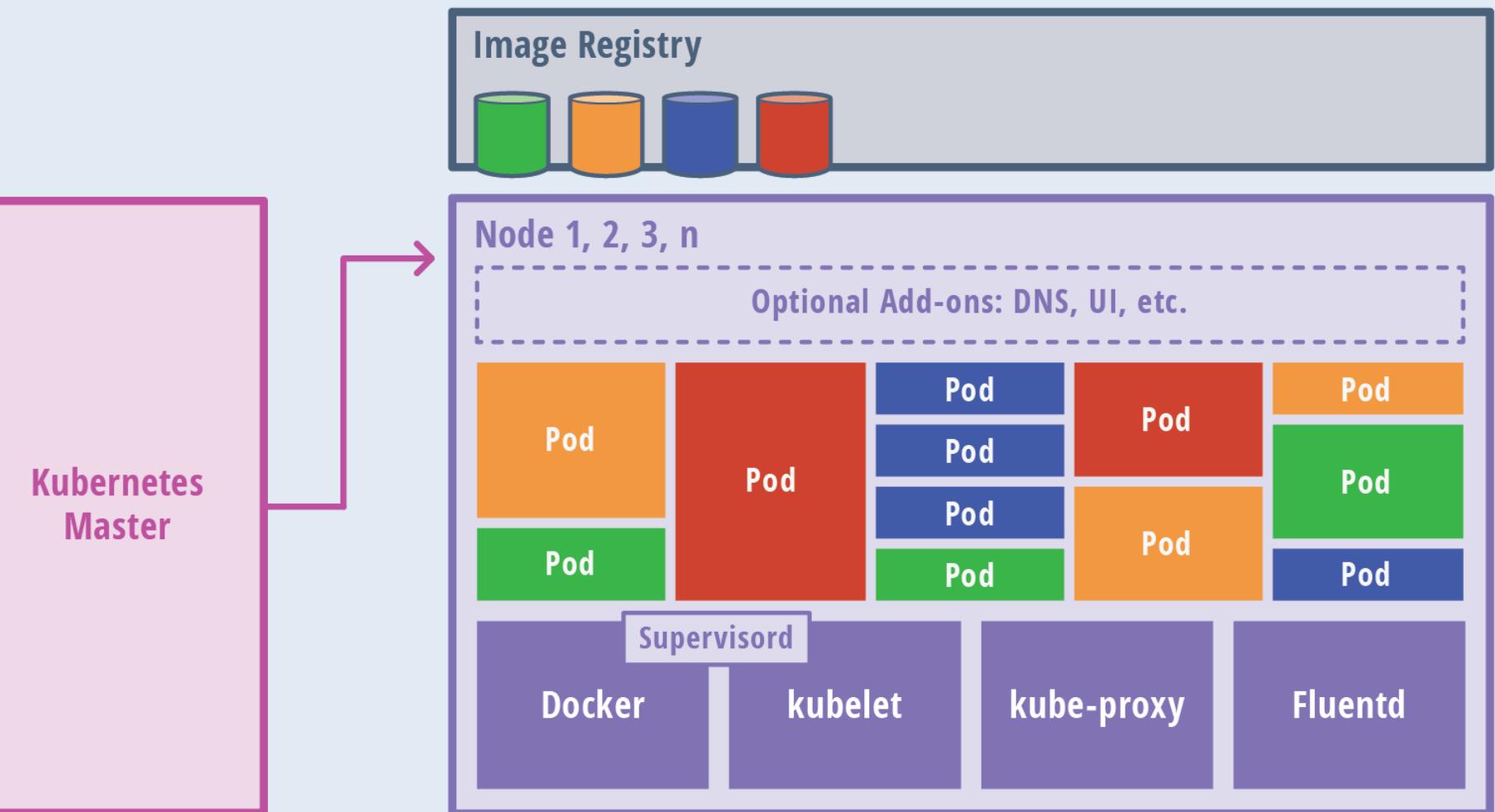
kubernetes
by Google

System Architecture

- Kubernetes 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

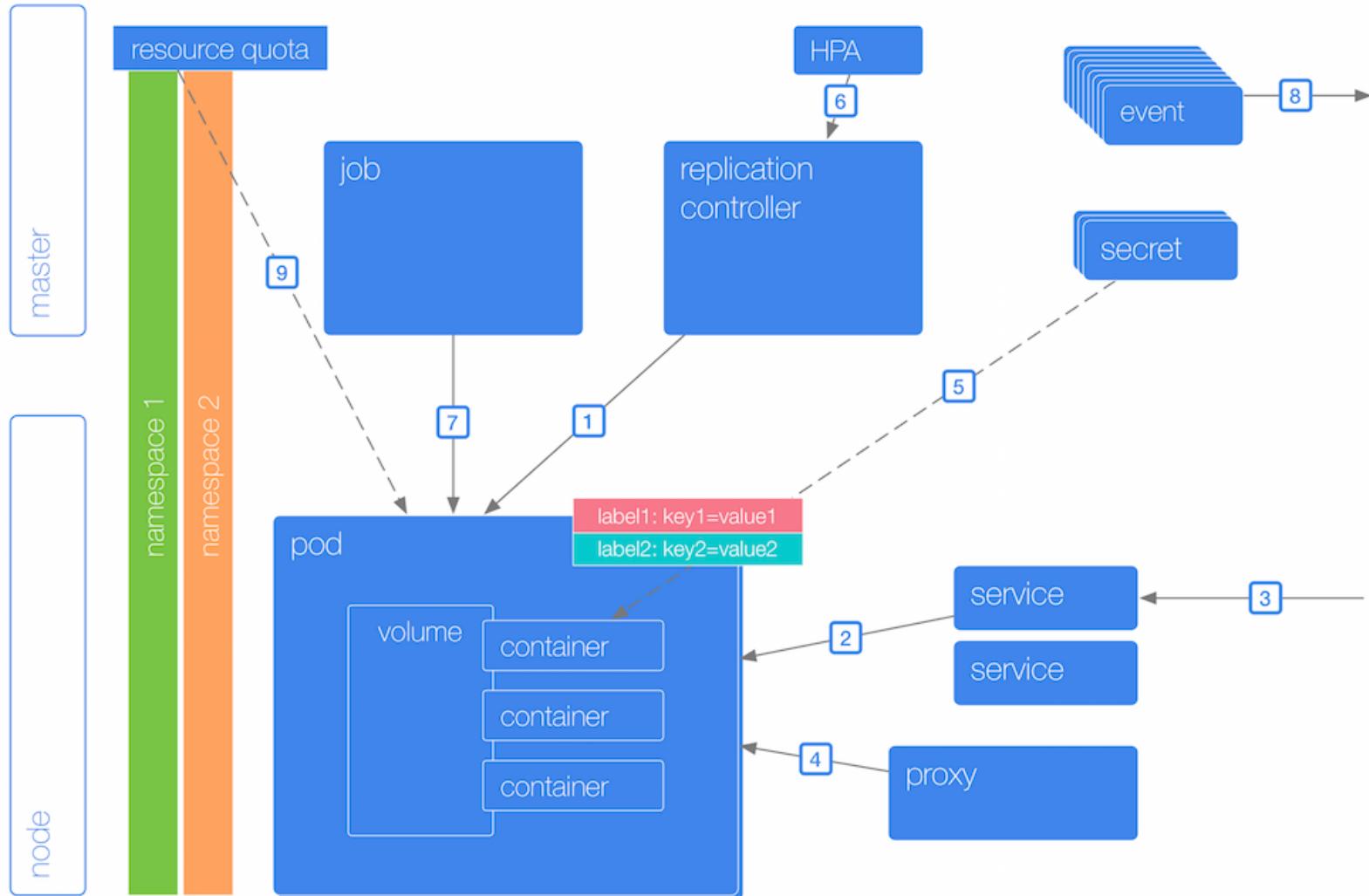
System Architecture

- 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

The screenshot shows the top navigation bar of the Kubernetes Blog. It includes the Kubernetes logo, a search icon, and links for Documentation, Blog (which is underlined), Partners, and Community.

Wednesday, June 27, 2018

Kubernetes 1.11: In-Cluster Load Balancing and CoreDNS Plugin Graduate to General Availability

Author: Kubernetes 1.11 Release Team

We're pleased to announce the delivery of Kubernetes 1.11, our second release of 2018!

Today's release continues to advance maturity, scalability, and flexibility of Kubernetes, marking significant progress on features that the team has been hard at work on over the last year. This newest version graduates key features in networking, opens up two major features from SIG-API Machinery and SIG-Node for beta testing, and continues to enhance storage features that have been a focal point of the past two releases. The features in this release make it increasingly possible to plug any infrastructure, cloud or on-premise, into the Kubernetes system.

Notable additions in this release include two highly-anticipated features graduating to general availability: IPVS-based In-Cluster Load Balancing and CoreDNS as a cluster DNS add-on option, which means increased scalability and flexibility for production applications.

Let's dive into the key features of this release:

IPVS-Based In-Cluster Service Load Balancing Graduates to General Availability

In this release, [IPVS-based in-cluster service load balancing](#) has moved to stable. IPVS (IP Virtual Server) provides high-performance in-kernel load balancing, with a simpler programming interface than iptables. This change delivers better network throughput, better programming latency, and higher scalability limits for the cluster-wide distributed load-balancer that comprises the Kubernetes Service model. IPVS is not yet the default but clusters can begin to use it for production traffic.

CoreDNS Promoted to General Availability

[CoreDNS](#) is now available as a [cluster DNS add-on option](#), and is the default when using kubeadm. CoreDNS is a flexible, extensible authoritative DNS server and directly integrates with the Kubernetes API. CoreDNS has fewer moving parts than the previous DNS server, since it's a single executable and a single process, and supports flexible use cases by creating custom DNS entries. It's also written in Go making it memory-safe. You can learn more about CoreDNS [here](#).

Dynamic Kubelet Configuration Moves to Beta

This feature makes it possible for new Kubelet configurations to be rolled out in a live cluster. Currently, Kubelets are configured via command-line flags, which makes it difficult to update Kubelet configurations in a running cluster. With this beta feature, [users can configure Kubelets in a live cluster](#) via the API server.

Custom Resource Definitions Can Now Define Multiple Versions

Custom Resource Definitions are no longer restricted to defining a single version of the custom resource, a restriction versions of the resource can be defined. In the future, this will be expanded to support some automatic conversions; for changes, e.g. v1beta1 to v1, and to create a migration path for resources which do have changes.

Custom Resource Definitions now also support "[status](#)" and "[scale](#)" [subresources](#), which integrate with monitoring and run cloud-native applications in production using Custom Resource Definitions.

Known Issues

- IPVS based kube-proxy doesn't support graceful close connections for terminating pod. This issue will be fixed in a future release. ([#57841](#), [@jsravn](#))
- kube-proxy needs to be configured to override hostname in some environments. ([#857](#), [@detiber](#))
- There's a known issue where the Vertical Pod Autoscaler will radically change implementation in 1.12, so users of VPA (alpha) in 1.11 are warned that they will not be able to automatically migrate their VPA configs from 1.11 to 1.12.

<https://kubernetes.io/docs/reference/workloads-18-19/>

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



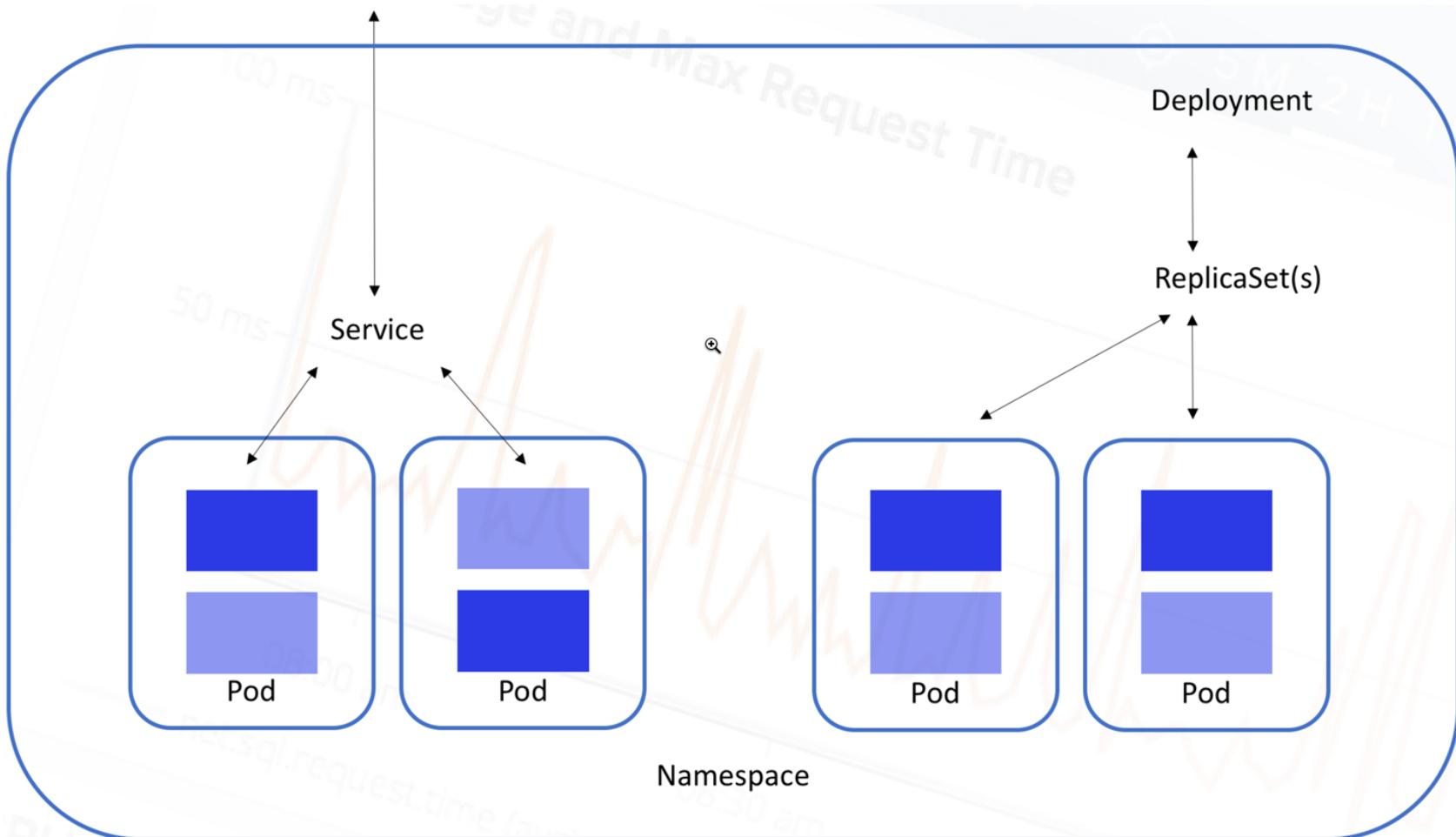
Pods, Container and Services

- 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)
 - Pods will can create replicas of 1000+ set on cluster system



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
```

Pods, Container and Services

- Imperative commands:

- kubectl expose (Service)

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

- Option:

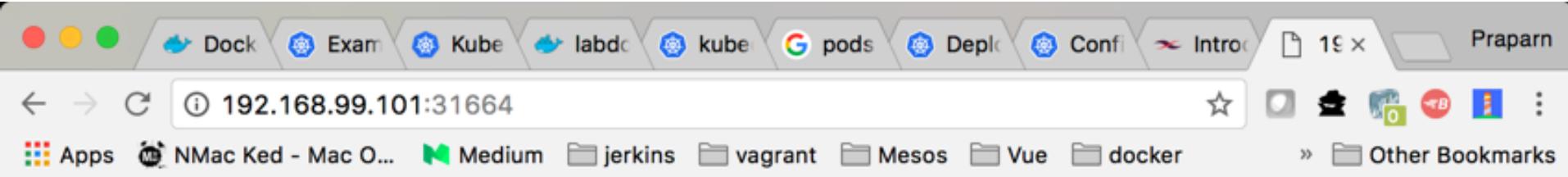
- --name=name
- --port=port for service
- --target-port=port of deployment set
- --type=<NodePort/ClusterIP/LoadBalance>
- --protocol=<TCP/UAT etc>
- Etc

```
[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
```

sample

Load Url

Browse

YAML TO JSON

YAML TO XML

YAML TO CSV

Validate

Seen this ad multiple times Not interested in this ad

Download

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"
    }
}
```



Pods, Container and Services

- Imperative object configuration

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

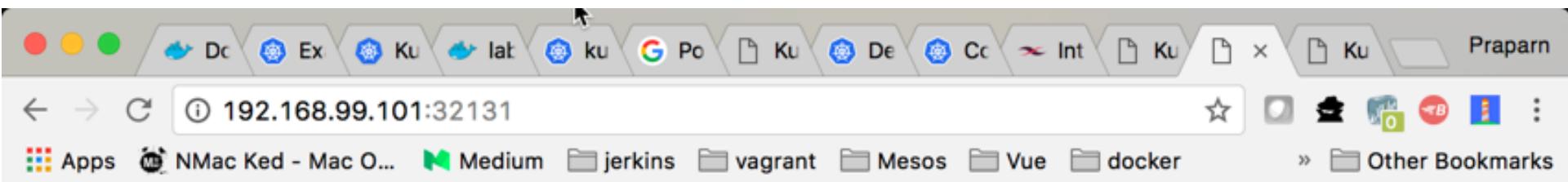
```
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment praparn$ ls -lh
total 24
-rw-r--r--@ 1 praparn  staff  550B Jul  2 09:29 instruction.txt
-rw-r--r--@ 1 praparn  staff  321B Jul  2 12:52 webtest_pod.yml
-rw-r--r--@ 1 praparn  staff  393B Jul  2 12:52 webtest_svc.yml
praparns-MacBook-Pro:WorkShop_1.2_Pods_Service_Deployment 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>
```



Welcome Page from Container Python Lab

Checkpoint Date/Time: Sun Jul 2 06:05:00 2017



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://sha256:837c8f41c918ede067f05f9b554b6120fdb654cc2a8eb7eec74bc383b09865f2b
    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
```



Pods, Container and Services

- Trade-offs
 - Imperative commands:
 - Advantage:
 - Simple & Easy
 - Single Step for All
 - Effective immediate real-time
 - Disadvantage:
 - Hard to review on complete deployment
 - Unable to track change of them/Store source of deployment
 - No template for reiteration
 - Very long syntax when facing with complex configuration



Pods, Container and Services

- Trade-offs
 - Imperative object configuration
 - Advantage:
 - Deployment source can keep and review process (svn, git etc)
 - Provide formal template for deployment
 - No skill set required deployment process
 - Disadvantage:
 - Need to understand of kubernetes before use
 - Writing YAML file before operate
 - Lost configuration update when it not appear on file (Next replacement will lost)
 - Design for manage via file not directory



Pods, Container and Services

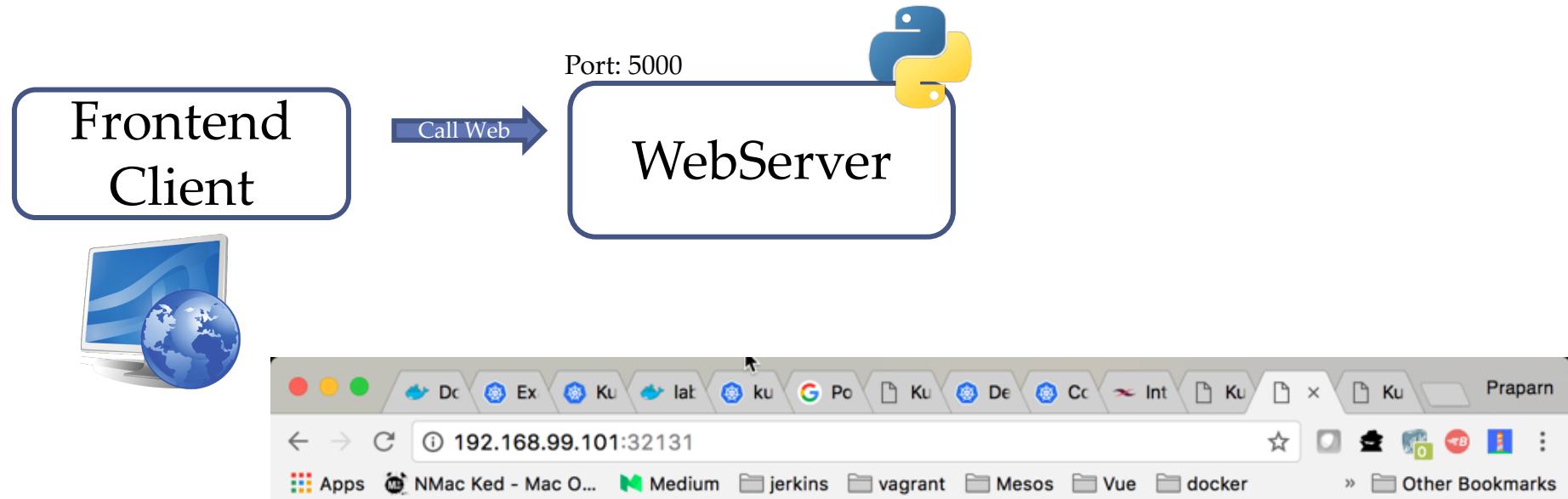
- Trade-offs
 - Declarative object configuration
 - Advantage:
 - Coverage all change in single command (Entire folder)
 - Fully support live-object configuration
 - Keep track/Merge all change in configuration file
 - Update default value in configuration file
 - Disadvantage:
 - More complicate on configuration file

Field in object configuration file	Field in live object configuration	Field in last-applied-configuration	Action
Yes	Yes	-	Set live to configuration file value.
Yes	No	-	Set live to local configuration.
No	-	Yes	Clear from live configuration.
No	-	No	Do nothing. Keep live value.



Workshop 1.2: Pods & Service

- Part 1: Deploy simple web pods with single container



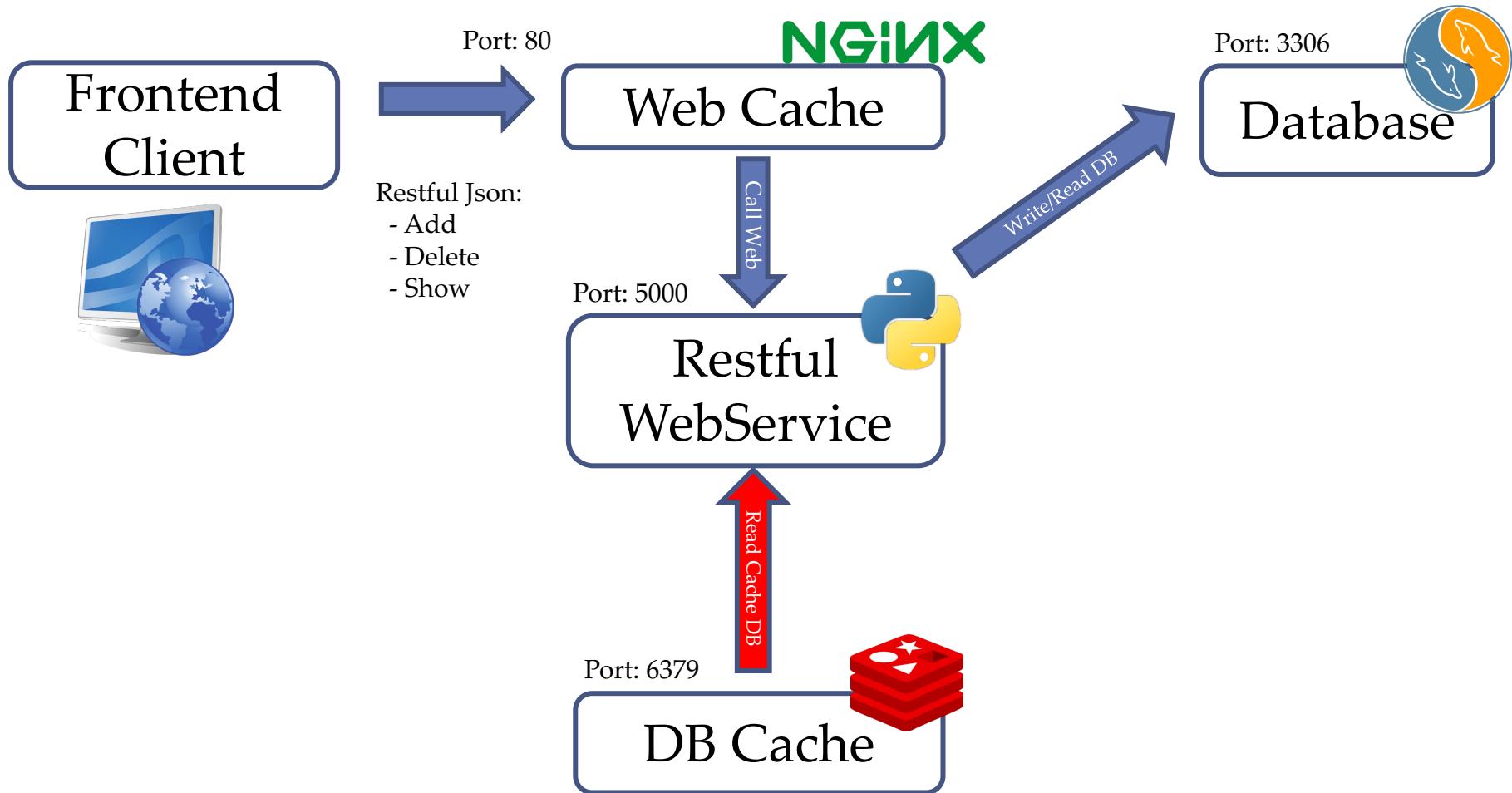
Pods, Container and Service

- 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;
}

gzip on;

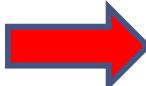
include /etc/nginx/conf.d/*.conf;
server {
    listen 80;
    client_body_buffer_size 50M;
    index index.html      index.htm;
    location / {
        proxy_pass http://webservice:5000;
        proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;
        proxy_redirect 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

- Setup container for test
 - docker-compose build --no-cache
 - docker-compose up -d
 - docker-compose start

```
docker@labdocker:~/PYTHON_REDIS_NGINX/Docker$ docker-compose build --no-cache
cachedb uses an image, skipping
maindb uses an image, skipping
Building webservice
Step 1/3 : FROM labdocker/alpinepython:2.7-onbuild
# Executing 3 build triggers...
Step 1/1 : COPY requirements.txt /usr/src/app/
Step 1/1 : RUN pip install --no-cache-dir -r requirements.txt
--> Running in a4a8824e872b
Collecting flask (from -r requirements.txt (line 1))
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting redis (from -r requirements.txt (line 2))
  Downloading redis-2.10.5-py2.py3-none-any.whl (60kB)
Collecting mysql-python (from -r requirements.txt (line 3))
  Downloading MySQL-python-1.2.5.zip (108kB)
Collecting itsdangerous>=0.21 (from flask->-r requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
```

```
docker@labdocker:~/PYTHON_REDIS_NGINX/Docker$ docker-compose up -d
Creating cachedb
Creating maindb
Creating webservice
Creating webcache
docker@labdocker:~/PYTHON_REDIS_NGINX/Docker$ docker-compose start
Starting webservice
```

```
docker@labdocker:~/PYTHON_REDIS_NGINX/Docker$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
0948b0369239        docker_webcache   "nginx -c /etc/ngi..."   3 hours ago        Up 3 hours         0.0.0.0:80->80/tcp   webcache
e873e02c3bcf        docker_webservice  "python main.py"      3 hours ago        Up 3 hours         0.0.0.0:5000->5000/tcp  webservice
456afbcce89a4       labdocker/mysql:latest "docker-entrypoint..."  3 hours ago        Up 3 hours         3306/tcp              maindb
fe5521999d1e        labdocker/redis:latest "docker-entrypoint..."  3 hours ago        Up 3 hours         6379/tcp              cachedb
docker@labdocker:~/PYTHON_REDIS_NGINX/Docker$
```



Pods, Container and Service

- How to test ?

```
Set environment:  
export Server_IP=<ip of docker host>  
export Server_Port=80 ==> if need to direct test change to 5000  
  
Initial Main Database:  
curl http://$Server_IP:$Server_Port/init  
  
Insert User on Database:  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "desribe":"Slave"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Somchai Sunsukan", "desribe":"Security Guard"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "3", "user":"Sanyachan Panrudee", "desribe":"House Keeping"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "4", "user":"Sakkhan Yanyicharoen", "desribe":"Messenger"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "5", "user":"Chatchai Moungang", "desribe":"Programmer"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "6", "user":"Anusit Kannaphat", "desribe":"DevOps Manager"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "7", "user":"Meelarp Maisanuk", "desribe":"System Engineer"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "8", "user":"Pansa Bunsong", "desribe":"Security Guard"}' http://$Server_IP:$Server_Port/users/insertuser  
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "9", "user":"Wiphanee Wongsaisawan", "desribe":"Administrator"}' http://$Server_IP:$Server_Port/users/insertuser  
  
Retrieve Data from Database:  
curl http://$Server_IP:$Server_Port/users/1  
curl http://$Server_IP:$Server_Port/users/1 ==> Expect from Cache  
curl http://$Server_IP:$Server_Port/users/4  
curl http://$Server_IP:$Server_Port/users/4 ==> Expect from Cache  
  
Delete Data from database:  
curl http://$Server_IP:$Server_Port/users/removeuser/1  
curl http://$Server_IP:$Server_Port/users/removeuser/2  
curl http://$Server_IP:$Server_Port/users/removeuser/3  
curl http://$Server_IP:$Server_Port/users/removeuser/4
```



Pods, Container and Service

- How to test ?
 - Initial database

```
praparns-MacBook-Pro:~ praparn$ export Server_IP=192.168.99.100
praparns-MacBook-Pro:~ praparn$ export Server_Port=80
praparns-MacBook-Pro:~ praparn$ curl http://$Server_IP:$Server_Port/init
##### Database Create New Account Table Done #####
praparns-MacBook-Pro:~ praparn$ █
```

- Insert database

```
praparns-MacBook-Pro:~ 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 00:48:00 GMT
Content-Type: application/json
Content-Length: 40
Connection: keep-alive
[...]
#####
Record was added #####
praparns-MacBook-Pro:~ praparn$ curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Somchai Sunsukwan", "desribe":"Security Guard"}' http://$Server_IP:$Server_Port/users/insertuser
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 02 Jul 2017 00:48:22 GMT
Content-Type: application/json
Content-Length: 40
Connection: keep-alive
[...]
#####
Record was added #####
praparns-MacBook-Pro:~ praparn$ █
```



Pods, Container and Service

- How to test ?
 - Retrieve database/cache

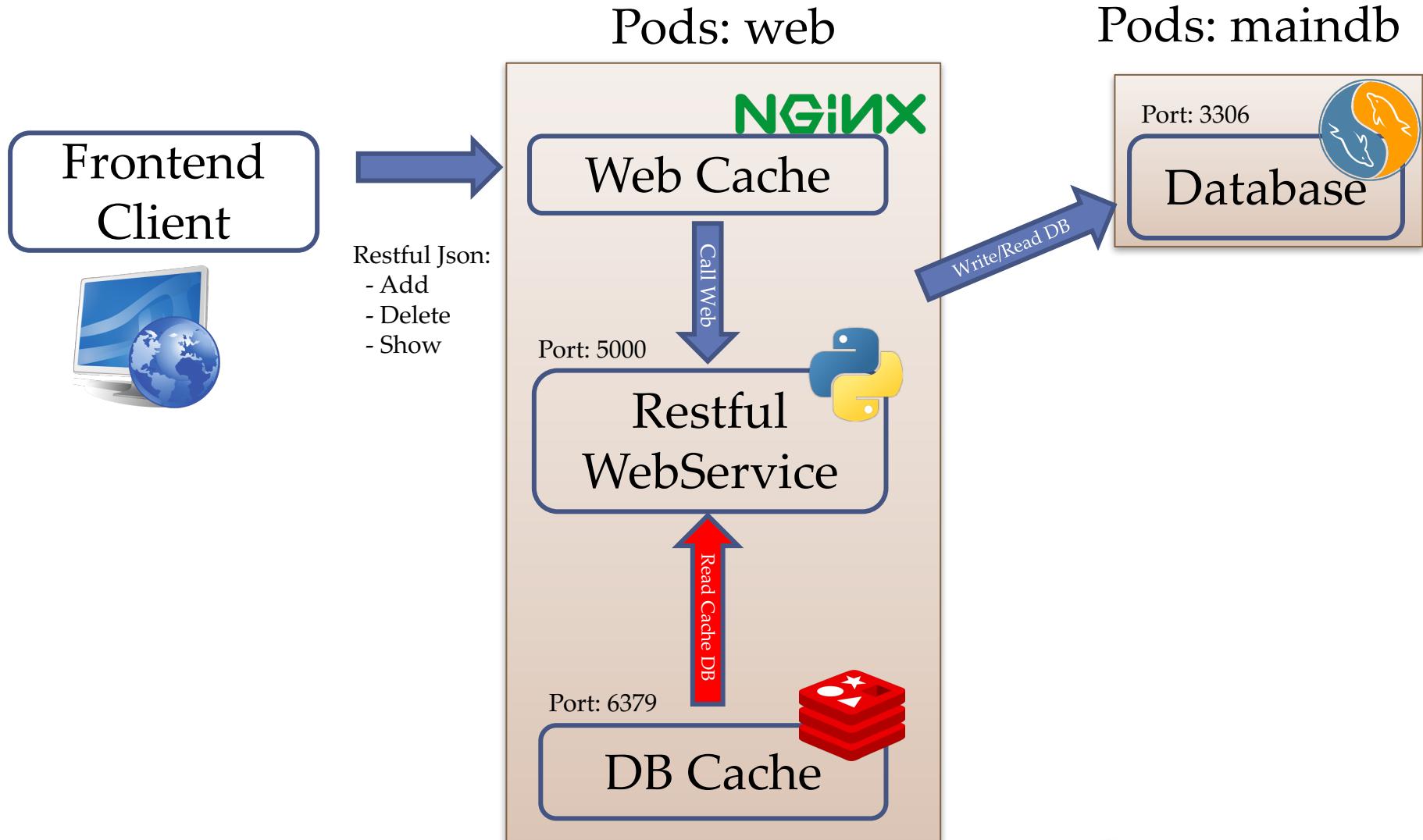
```
praparns-MacBook-Pro:~ paparn$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlappraparns-MacBook-Pro:~ paparn$
praparns-MacBook-Pro:~ paparn$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Cache)praparns-MacBook-Pro:~ paparn$
```

- Delete database

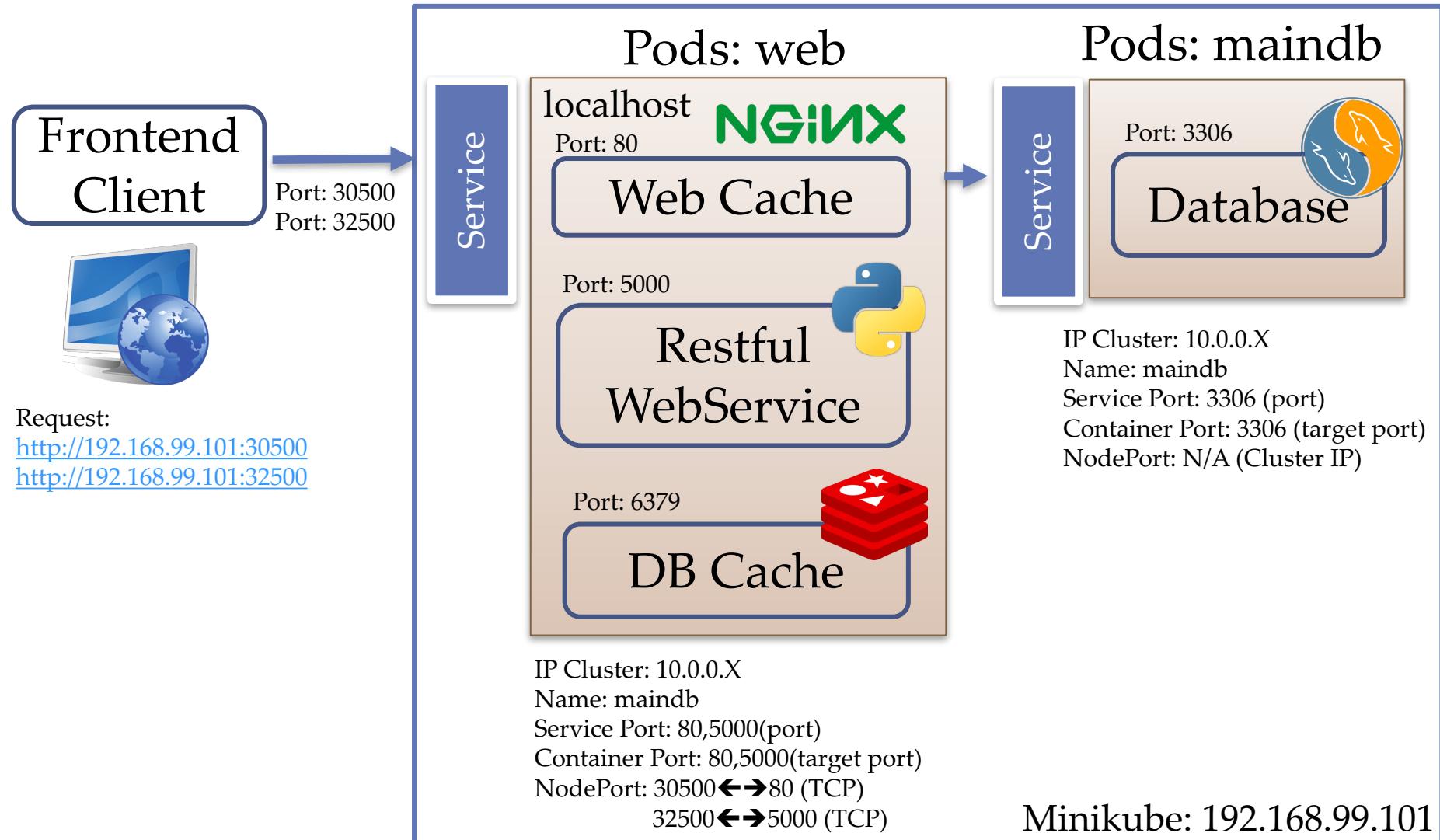
```
praparns-MacBook-Pro:~ paparn$ curl http://$Server_IP:$Server_Port/users/removeuser/1
##### Record was deleted (Both Database Cache) #####
praparns-MacBook-Pro:~ paparn$ curl http://$Server_IP:$Server_Port/users/removeuser/2
##### Record was deleted #####
praparns-MacBook-Pro:~ paparn$
```



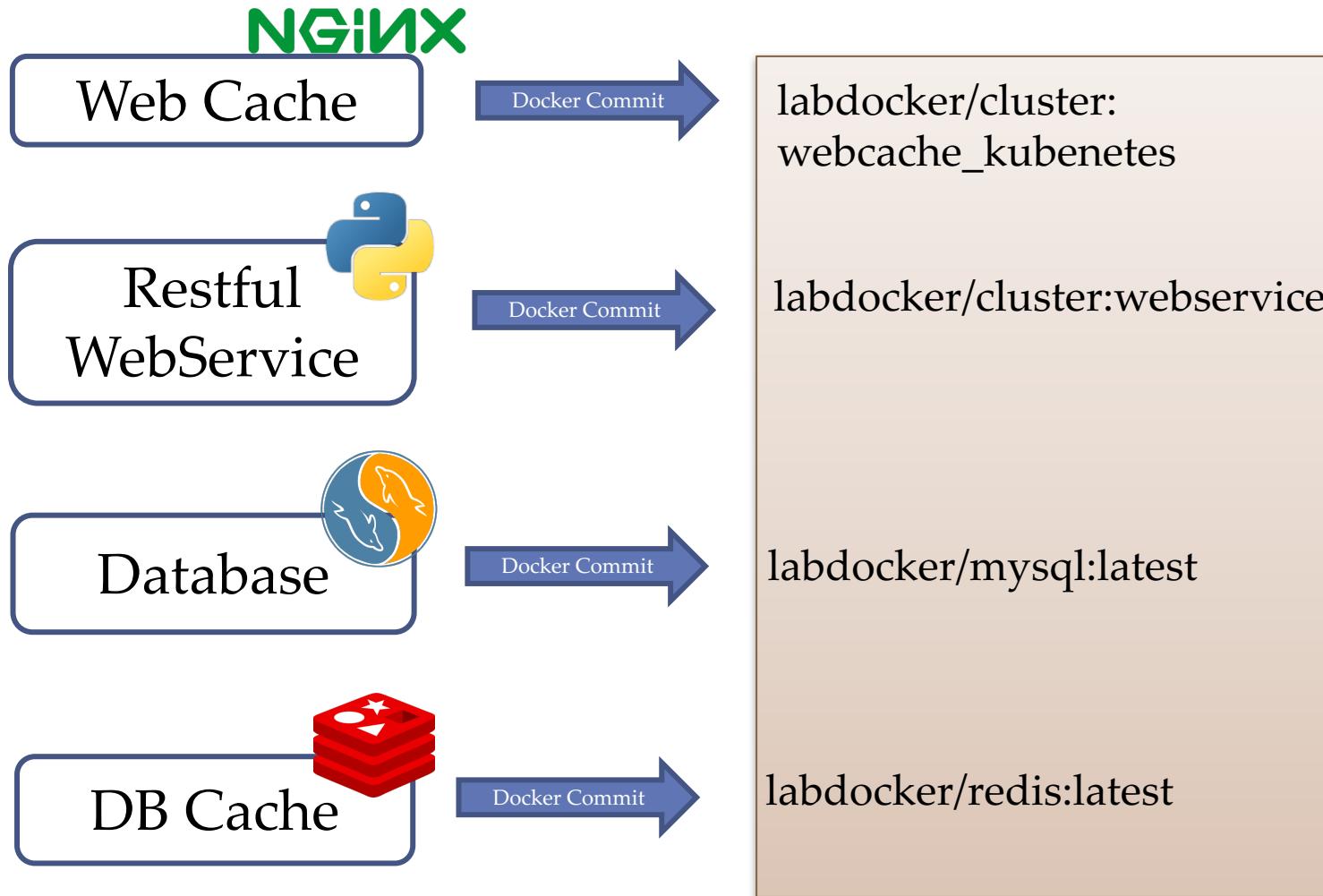
Pods, Container and Service



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/mysql:latest
15       ports:
16         - containerPort: 3306
17           protocol: TCP
18       env:
19         -
20           name: "MYSQL_ROOT_PASSWORD"
21           value: "password"
```

- 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"
```



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 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:webservice
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 (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 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 Service

- Create "maindb" Pods and service

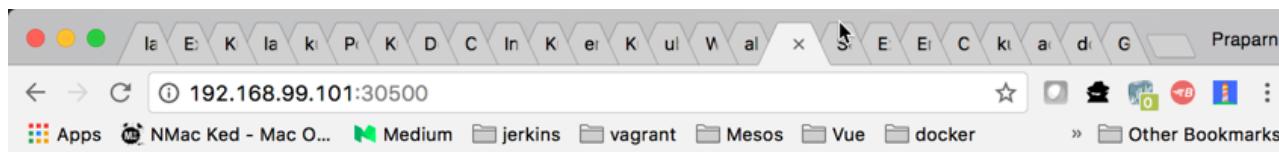
```
[praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
No resources found.
[praparns-MacBook-Pro:multicontainer praparn$ kubectl get svc
NAME      CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
kubernetes  10.0.0.1    <none>        443/TCP  4d
[praparns-MacBook-Pro:multicontainer praparn$ kubectl 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
I
#####
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

- 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:webservice
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:webservice2
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
```

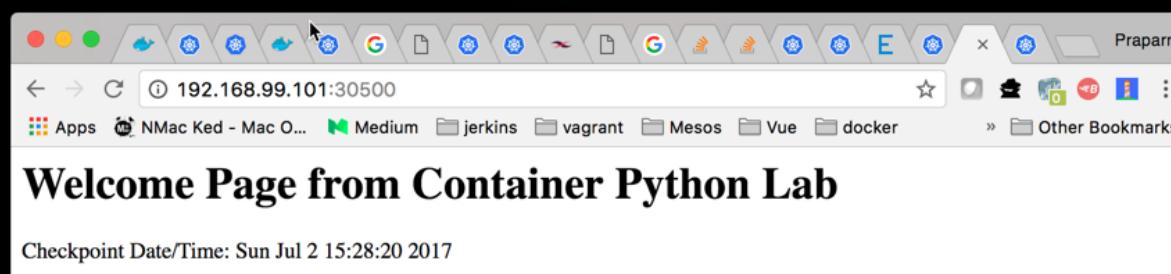


Pods, Container and Services

- Services

- Existing Pods “web”

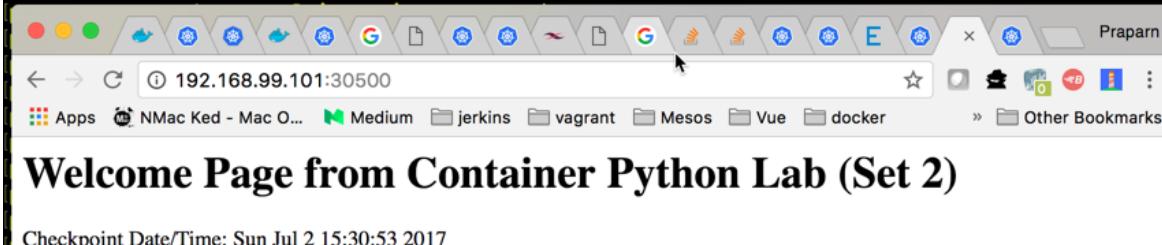
```
praparns-MacBook-Pro:multicontainer praparn$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
curl-1580724602-gjihq   0/1     CrashLoopBackOff  27      1h
maindb        1/1     Running   0          56m
web           3/3     Running   0          7m
praparns-MacBook-Pro:multicontainer 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$ 
```



```
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

- Publish Service Type:

- LoadBalance (Usually from cloud provider): service will allow loadbalance send traffic through Pods directly

```
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

- 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)

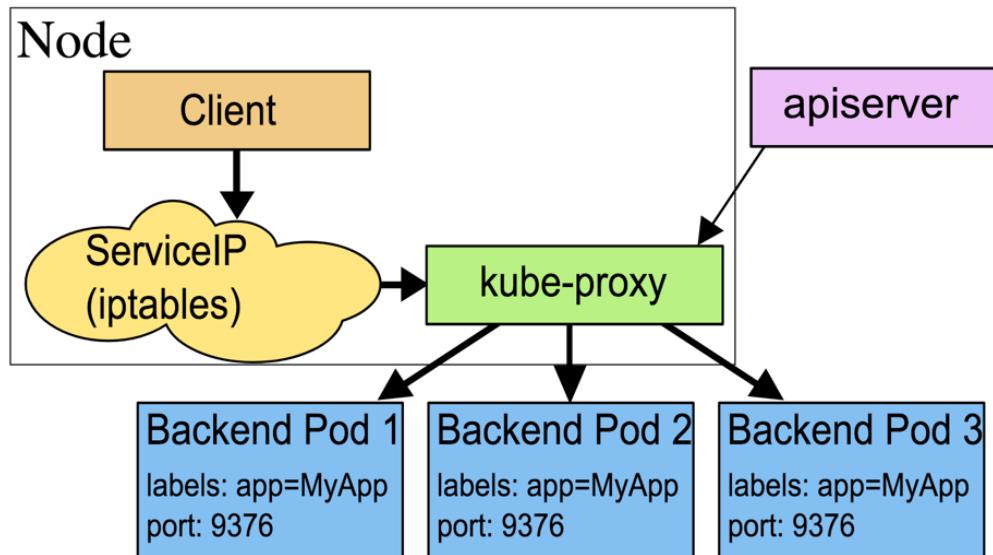
```
1 kind: Service
2 apiVersion: v1
3 metadata:
4   name: my-service
5   namespace: prod
6 spec:
7   type: ExternalName
8   externalName: my.database.example.com
```

- nslookup "my-service.prod.svc.CLUSTER" → return CNAME: "my.database.example.com"



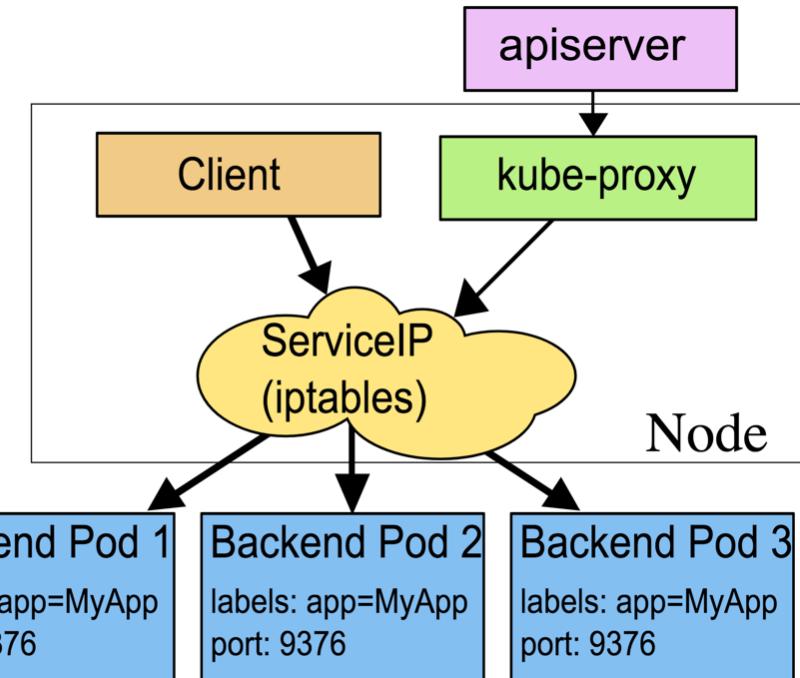
Pods, Container and Services

- 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



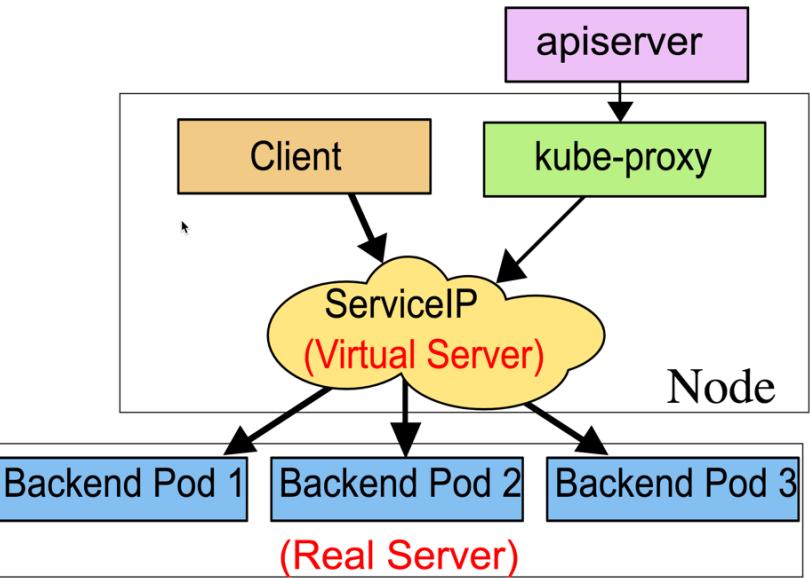
Pods, Container and Services

- kube-proxy (proxy-mode)
 - Proxy-mode: iptables (default on kubernetes v1.8 – current)
 - 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 (beta on kubernetes v1.9)
 - 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
 - lc: least connection
 - dh: destination hash
 - sh: source hash
 - sed: shortest expected delay
 - nq: never queue



Pods, Container and Services

- Configure kube-proxy (proxy-mode)

How to use IPVS

This document shows how to use kube-proxy ipvs mode.

What is IPVS

IPVS (IP Virtual Server) implements transport-layer load balancing, usually called Layer 4 LAN switching, as part of Linux kernel.

IPVS runs on a host and acts as a load balancer in front of a cluster of real servers. IPVS can direct requests for TCP and UDP-based services to the real servers, and make services of real servers appear as virtual services on a single IP address.

Run kube-proxy in ipvs mode

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

Local UP Cluster

Kube-proxy will run in iptables mode by default in a [local-up cluster](#).

Users should export the env `KUBE_PROXY_MODE=ipvs` to specify the ipvs mode before deploying the cluster if want to run kube-proxy in ipvs mode.

Cluster Created by Kubeadm

Kube-proxy will run in iptables mode by default in a cluster deployed by [kubeadm](#).

If you are using kubeadm with a [configuration file](#), you can specify the ipvs mode adding `SupportIPVSPProxyMode: true` below the `kubeProxy` field. Then the configuration file is similar to:

```
kind: MasterConfiguration
apiVersion: kubeadm.k8s.io/v1alpha1
...
kubeProxy:
  config:
    featureGates: SupportIPVSPProxyMode=true
    mode: ipvs
  ...
...
```

Note: ipvs mode assumes IPVS kernel modules are installed on the node before running kube-proxy. When kube-proxy starts with ipvs proxy mode, kube-proxy would validate if IPVS modules are installed on the node, if it's not installed kube-proxy will fall back to iptables proxy mode.

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

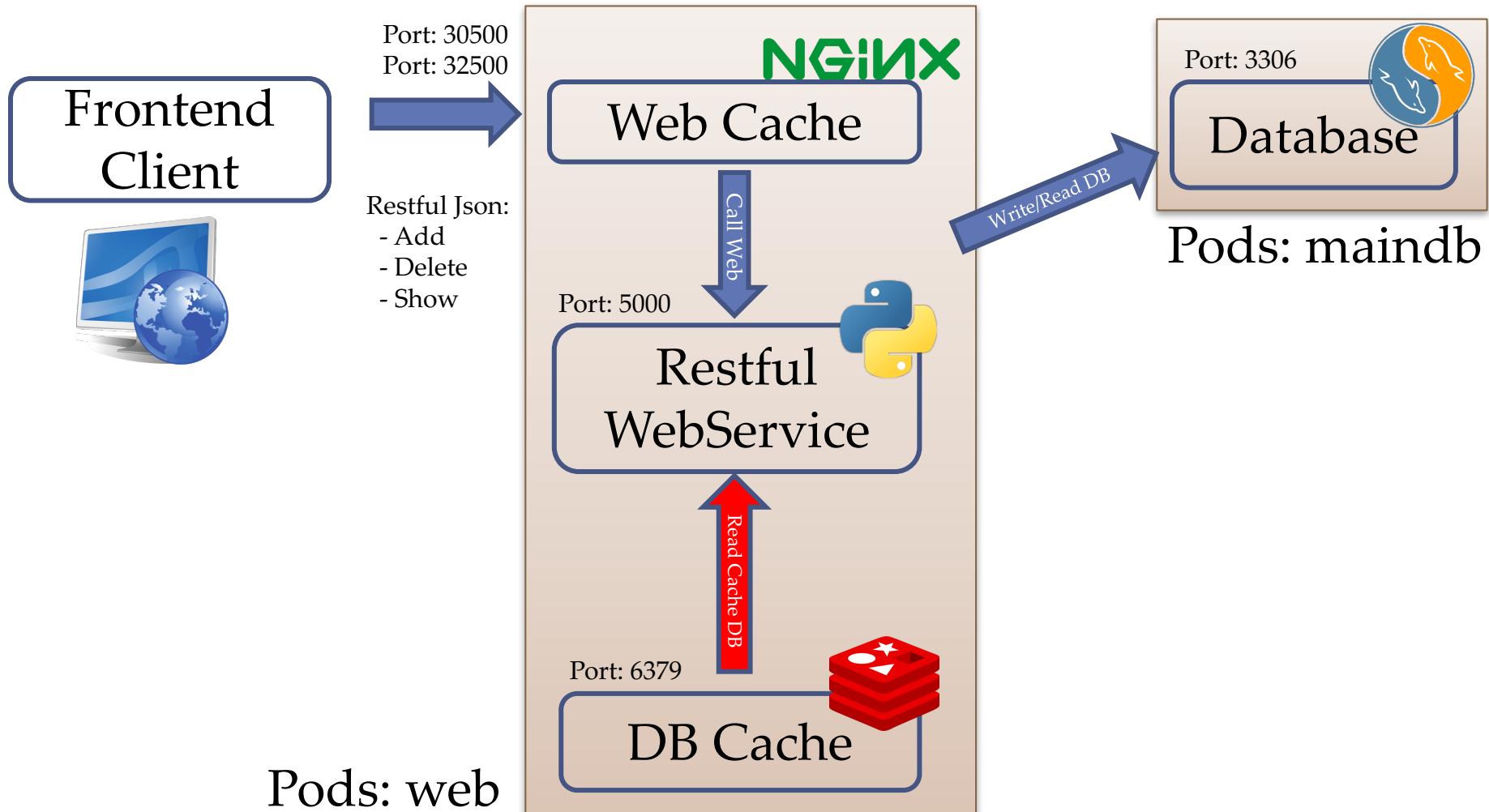
Kubernetes: Production Workload Orchestration



kubernetes
by Google

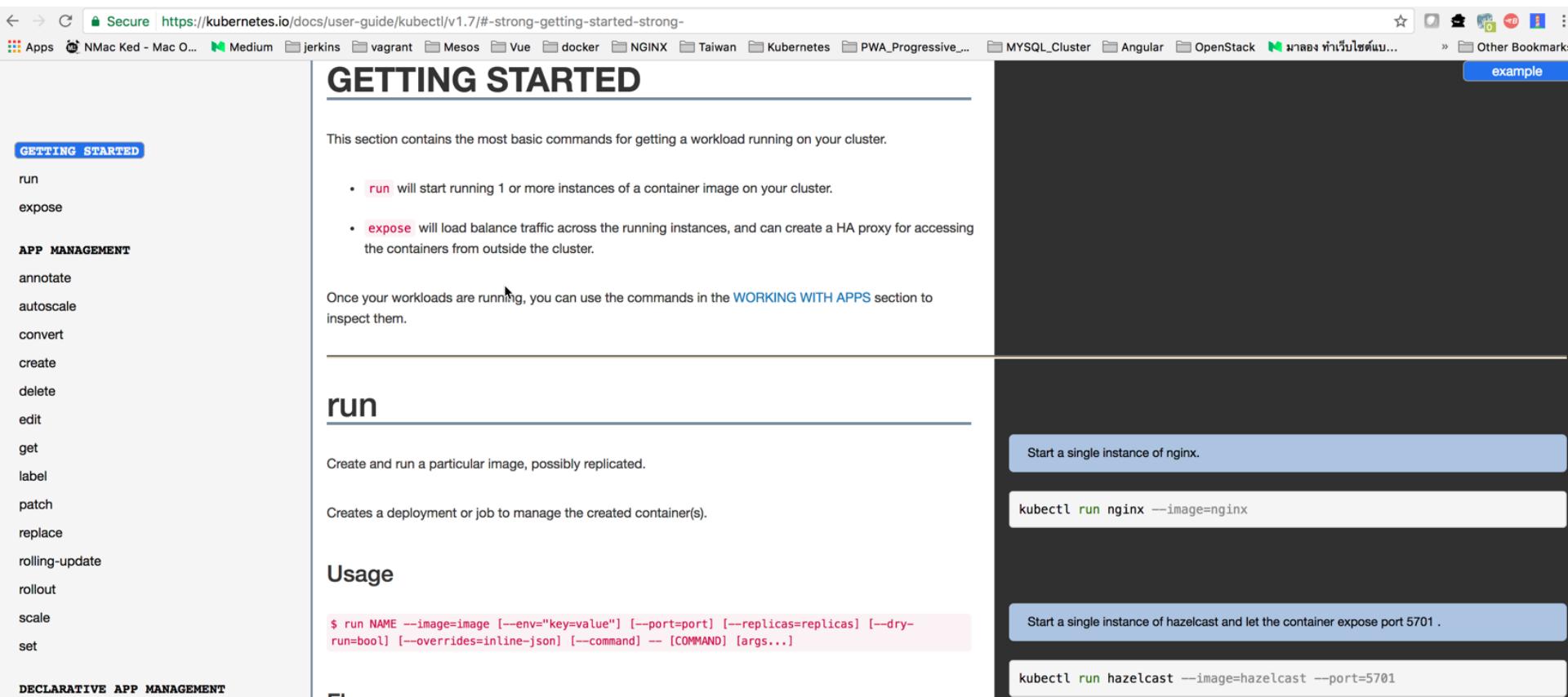
Workshop 1.2: Pods & Service

- Part 2: Deploy web pods with multi container



Pods, Container and Services

- Other Command



The screenshot shows a web browser displaying the Kubernetes documentation at <https://kubernetes.io/docs/user-guide/kubectl/v1.7/#-strong-getting-started-strong->. The page title is "GETTING STARTED". On the left sidebar, under "GETTING STARTED", there is a list of commands: run, expose, and others. Under "APP MANAGEMENT", there are more commands: annotate, autoscale, convert, create, delete, edit, get, label, patch, replace, rolling-update, rollout, scale, set, and DECLARATIVE APP MANAGEMENT. The main content area starts with a section about "run" which describes it as creating and running a container image, possibly replicated. It includes a usage example: `$ run NAME --image=image [--env="key=value"] [--port=port] [--replicas=replicas] [--dry-run=bool] [--overrides.inline.json] [--command] -- [COMMAND] [args...]`. To the right, there are two examples: "Start a single instance of nginx." with the command `kubectl run nginx --image=nginx`, and "Start a single instance of hazelcast and let the container expose port 5701." with the command `kubectl run hazelcast --image=hazelcast --port=5701`.

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

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Daemon Set and RC

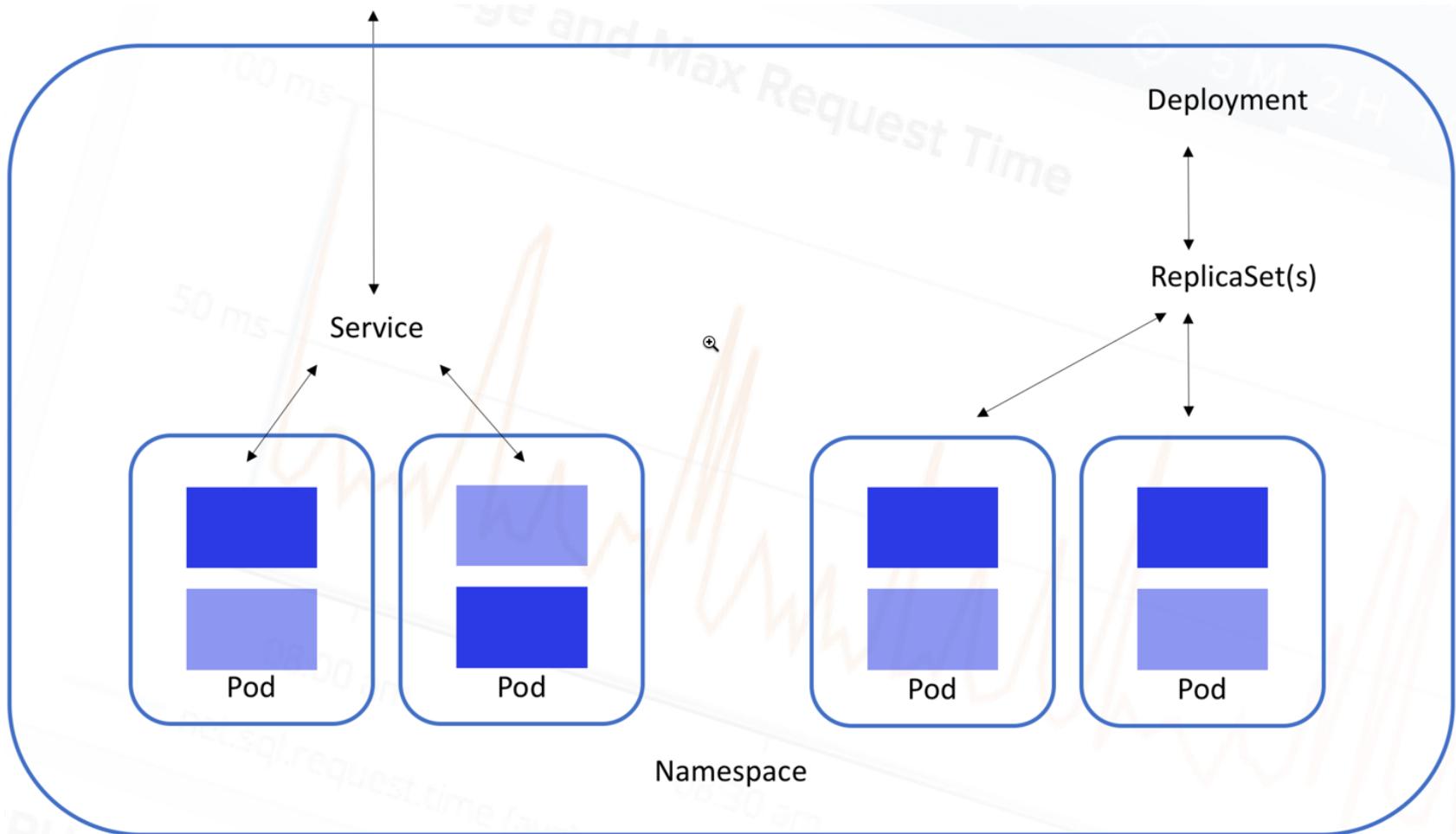


Kubernetes: Production Workload Orchestration



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 is work similar.
 - RC (Replication Controller) will response:
 - Create/Maintain Pods as "Replication Controller" (Pods Farm)
 - Keep copy of Pods (Replicas) as design
 - Ensure that Pods is up and run with amount like design
 - If too much, It will kill some Pods
 - If too kill, It will create another replicas of Pods
 - Auto healing if some Pods crash with 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      I  
  version: "1.0"  
  module: WebServer  
  environment: development
```

Replication Controller (RC)

- Example
 - Pods “webtest”
 - RC “webtest”
 - SVC “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
```

```
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
```

```
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
```



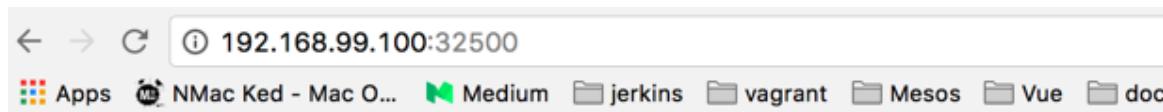
Replication Controller (RC)

- 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-jbqqd
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-jbqqd
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$ █
```



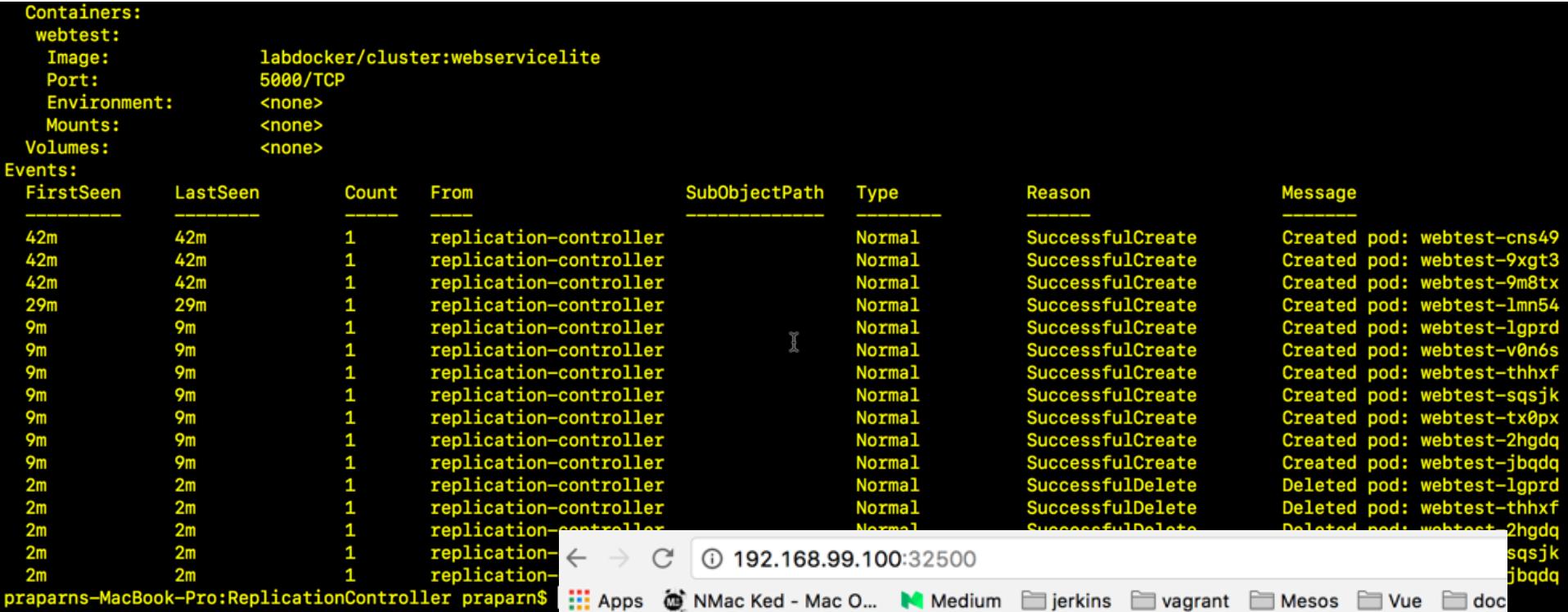
Workshop 1.3: RC

- 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-jbqqd
  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  jbqqd
praparns-MacBook-Pro:ReplicationController praparn$
```

192.168.99.100:32500



Welcome Page from Container Python Lab

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



Deployment/RS and Update

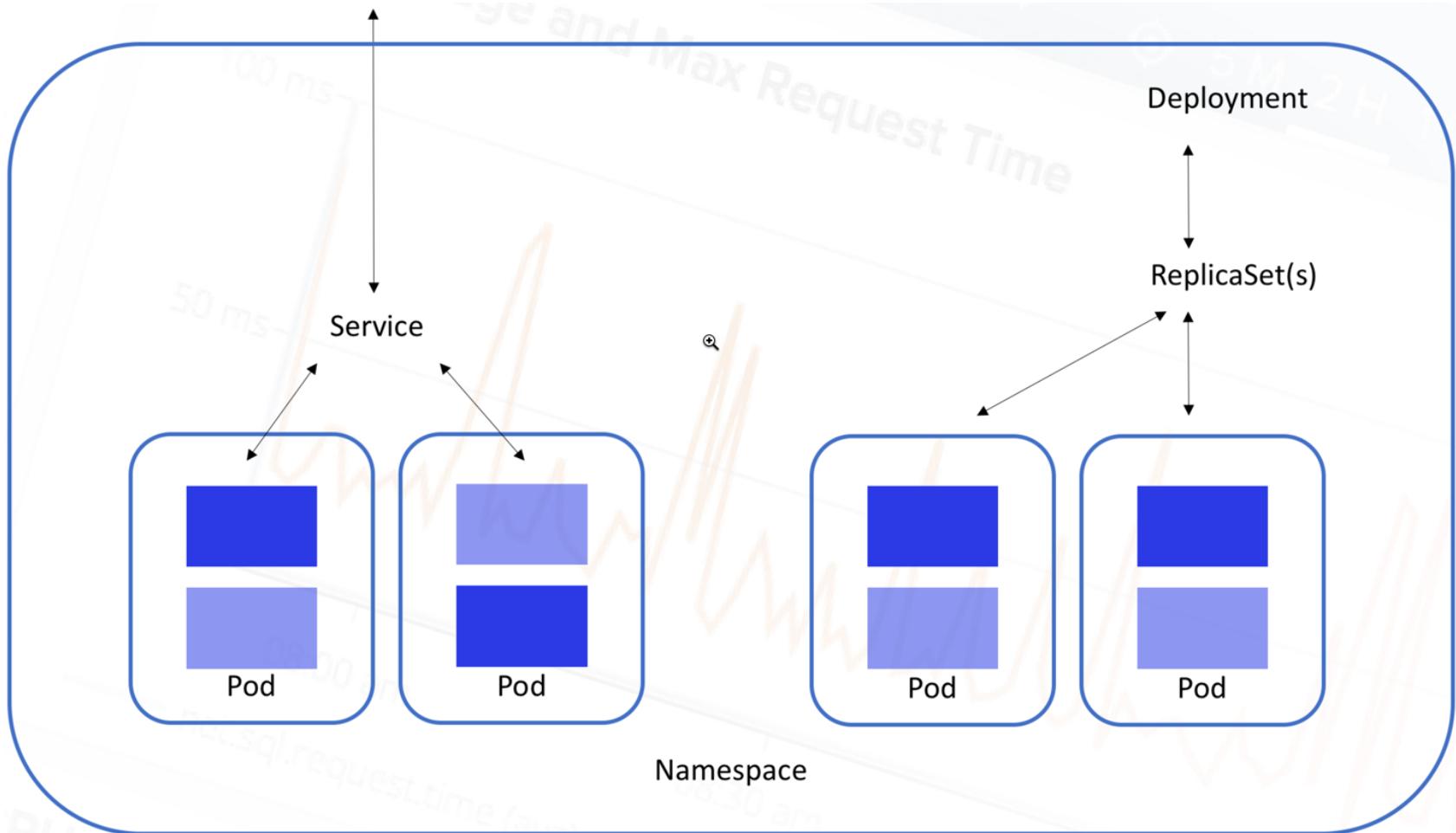


Kubernetes: Production Workload Orchestration



kubernetes
by Google

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
```

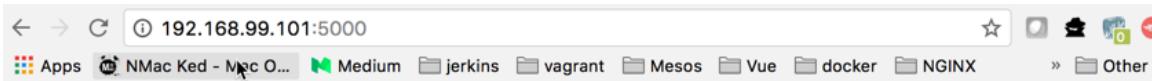
- RS support label with method “Equality-based requirement” and “Set-based requirement”

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

Deployment/RS and Update

- Example

- labdocker/cluster:webserviceelite_v1



Welcome Page from Container Python Lab Web Version 1.00

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

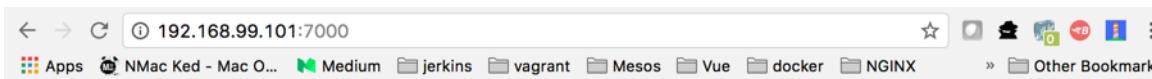
- labdocker/cluster:webserviceelite_v1.51rc



Welcome Page from Container Python Lab Web Version 1.51 RC

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

- labdocker/cluster:webserviceelite_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: Paparn_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: Paparn_L
18        version: "1.0"
19        module: WebServer
20        environment: development
21    template:
22      metadata:
23        labels:
24          name: web
25          owner: Paparn_L
26          version: "1.0"
27          module: WebServer
28          environment: development
29    spec:
30      containers:
31        - name: webtest
32          image: labdocker/cluster:webservicelite_v1
33          ports:
34            - containerPort: 5000
35              protocol: TCP
```

SVC “webtest”

```
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
```



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

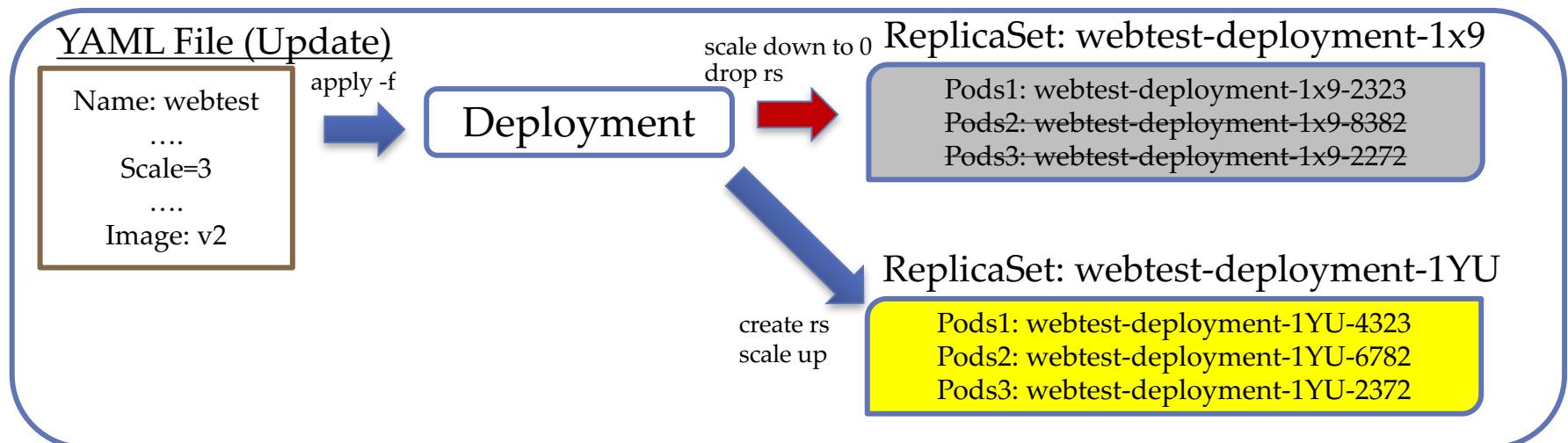
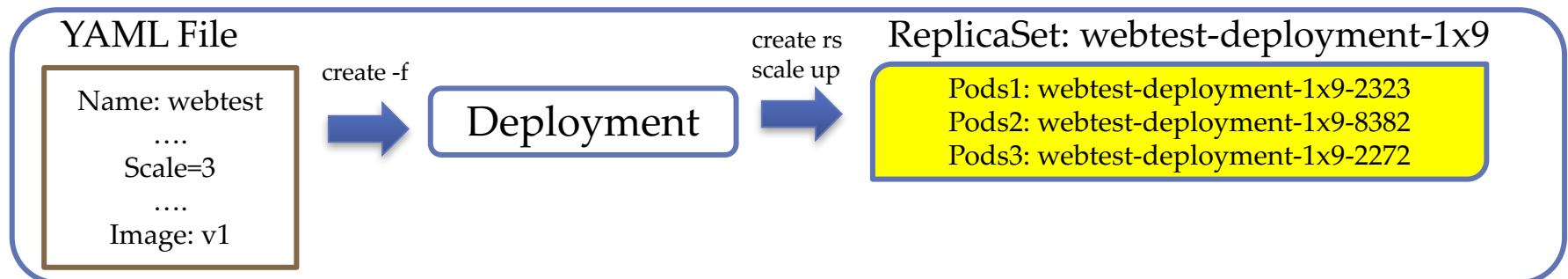
- Deployment update strategy (Rollout)
 - Update will start trigger when some of “spec:template” is changed

```
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   spec:
12     replicas: 3
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:
```

- Step for produce update (rollout) with Each Change
 - Scale down existing RS to 1 (1 max unavailable)
 - Create new RS for template change
 - Scale new RS as designed (No more than design +1: 1 max surge)
 - Delete existing RS

Deployment/RS and Update

- Deployment update strategy (Rollout)



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

```
[praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.51rc
deployment "webtest" image updated
[praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl rollout status deployment/webtest
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 old replicas are pending termination...
deployment "webtest" successfully rolled out
[praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.8ga
deployment "webtest" image updated
[praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl rollout status deployment/webtest
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for rollout to finish: 1 old replicas are pending termination...
deployment "webtest" successfully rolled out
```

- Rollout History

```
[praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl rollout history deployment/webtest
deployments "webtest"
REVISION      CHANGE-CAUSE
1             kubectl create --filename=webtest_deploy.yml --record=true
2             kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.51rc
3             kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.8ga
```



Deployment/RS and Update

- Deployment update strategy

- Rollback process

```
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl rollout undo deployment/webtest --to-revision=2
deployment "webtest" rolled back
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl rollout history deployment/webtest
deployments "webtest"
REVISION      CHANGE-CAUSE
1            kubectl create --filename=webtest_deploy.yml --record=true
3            kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.8ga
4            kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.51rc

praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ curl http://192.168.99.100:32500
<H1> Welcome Page from Container Python Lab Web Version 1.51 RC </H1>Checkpoint Date/Time: Thu Jul  6 18:35:53 2017
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ █
```

- Rollout History

```
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kubectl rollout history deployment/webtest
deployments "webtest"
REVISION      CHANGE-CAUSE
1            kubectl create --filename=webtest_deploy.yml --record=true
3            kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.8ga
4            kubectl set image deployment/webtest webtest=labdocker/cluster:webservicelite_v1.51rc

praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ █
```



Volume



Kubernetes: Production Workload Orchestration



kubernetes
by Google

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
 - DiskawsElasticBlockStore
 - Nfs
 - Iscsi
 - fc(fibrechannel)
 - Flocker
 - Glusterfs
 - Rbd
 - Cephfs
 - gitRepo
 - Secret
 - **persistentVolumeClaim (PVC) (Talk Day2)**
 - downwardAPI
 - Projected
 - azureFileVolume
 - azureDisk
 - 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:

```
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/docker/
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/docker/
```



The screenshot shows the cAdvisor interface. At the top, there's a navigation bar with a root icon and a Docker Containers link. Below that is a section titled 'Subcontainers' with a list of subcontainers: /init.scope, /kubepods, /system.slice, and /user.slice.



Volume

- hostPath:
- Container create file

Host check file

```
praparnlueangphoonlap — Terminal MAC Pro — kubectl exec -it cadvisor
~ — Terminal MAC Pro — ssh + minikube ssh
~ — Terminal

praparns-MacBook-Pro% kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
cadvisor-1335144146-jgm3h   1/1     Running   0          15m
praparns-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
/ #
```

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



Volume

- hostPath:
- Create another Pods and check file

```
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/docker/
```

```
praparns-MacBook-Pro% kubectl create -f webtest_pod.yml
pod "webtest" created
praparns-MacBook-Pro% kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
cadvisor-1335144146-jgm3h   1/1     Running   0          21m
webtest        1/1     Running   0          3s
praparns-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 1.5: Volume

```
32      name: cadvisor
33      owner: Paparn_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: Paparn_L
43        version: "1.0"
44        module: Monitor
45        environment: development
46  template:
47    metadata:
48      labels:
49        name: cadvisor
50        owner: Paparn_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/docker/
63              name: voldocker
64              image: labdockerc/cadvisor:latest
65          ports:
66            - containerPort: 8080
67              protocol: TCP
68  I  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/docker/
```



/

root

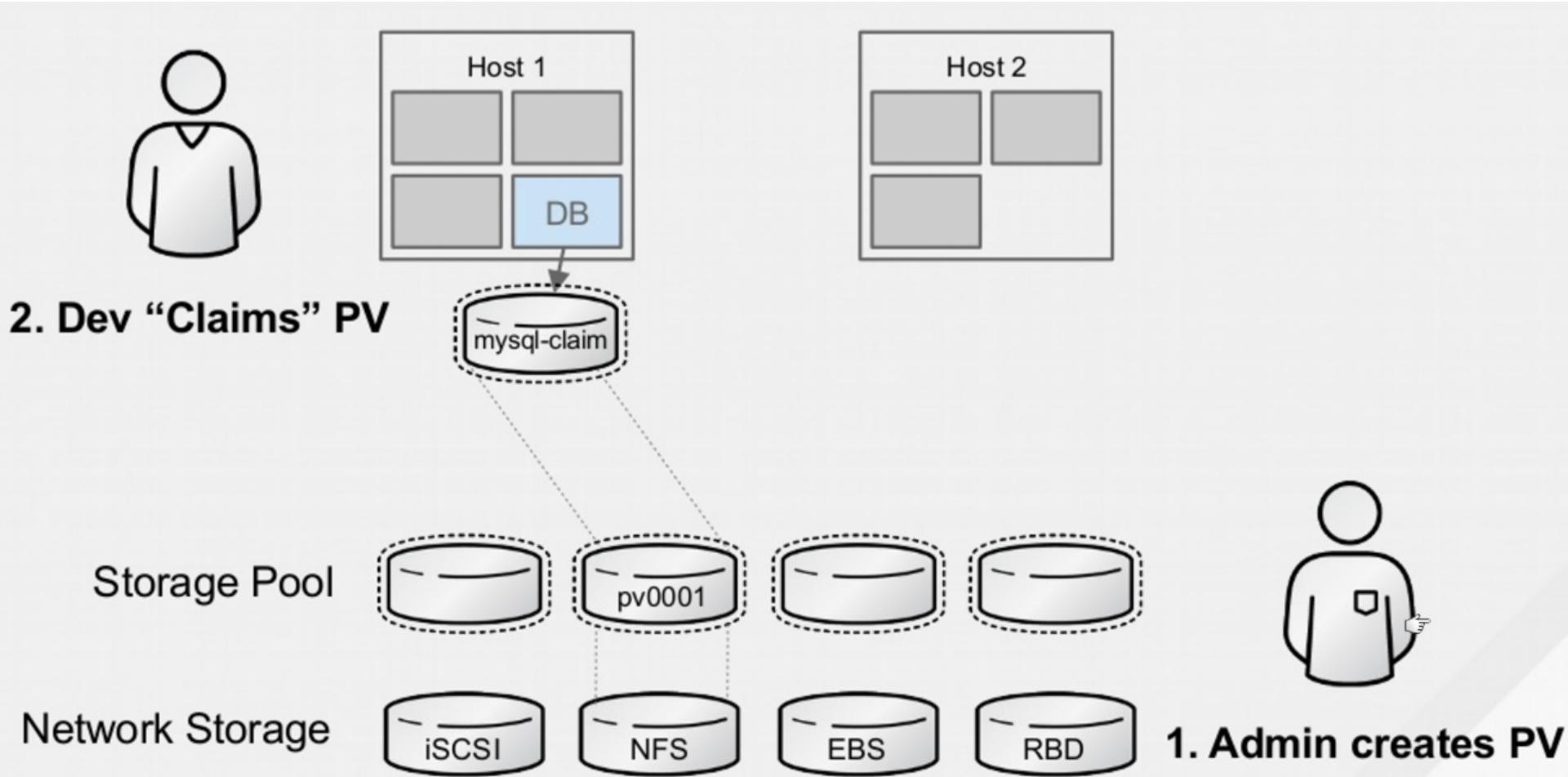
Docker Containers

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 Pod's status to inform kubelet for operate with Pods (Remain / Terminated / Restart)
 - Pods Status (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

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

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

Kubernetes: Production Workload Orchestration

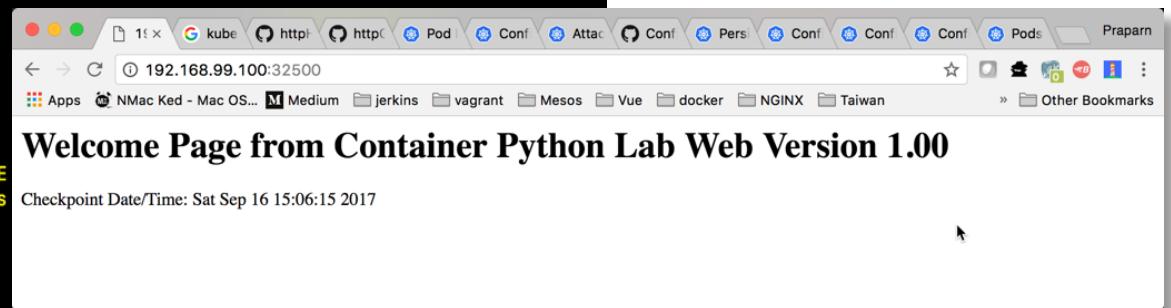


kubernetes
by Google

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

- TCPSocketAction:

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

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

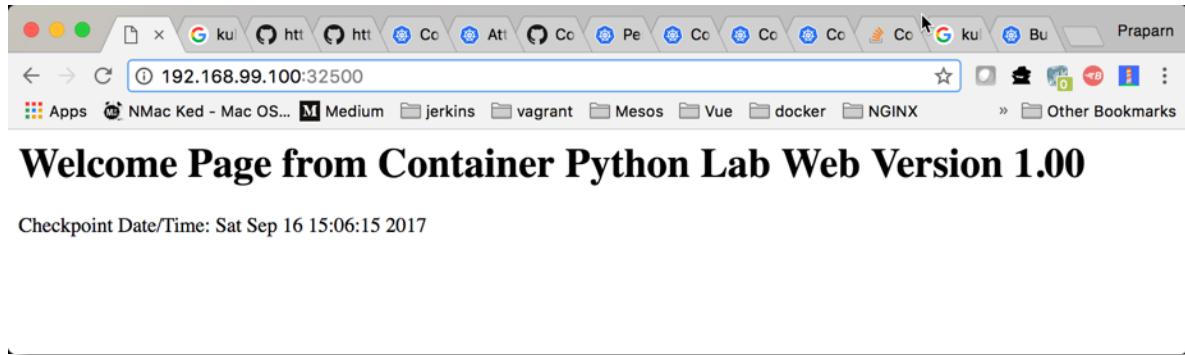
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Liveness and Readiness Probe

- TCPSocketAction:



```
[praparn-MacBook-Pro% kubectl create -f webtest_deploy_liveness_readiness_port.yml
deployment "webtest" created
[praparn-MacBook-Pro% kubectl get deployment/webtest
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
webtest   1          1          1           0           14s
[praparn-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						
2m	2m	1	kubelet, minikube		Normal	SuccessfulMountVolume	MountVolume
.SetUp succeeded for volume "default-token-pcw4h"							
2m	2m	1	kubelet, minikube	spec.containers{webtest}	Normal	Pulled	Container i
mage	image "labdocke/cluster:webservicelite_v1" already present on machine						
2m	2m	1	kubelet, minikube	spec.containers{webtest}	Normal	Created	Created con
tainer							
2m	2m	1	kubelet, minikube	spec.containers{webtest}	Normal	Started	Started con
tainer							

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

- TCP Socket Action:

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.								
obe failed: dial tcp 172.17.0.6:3000: getsockopt: connection refused								
4m	9s	12	kubelet, minikube	spec.containers{webtest}	Warning	Unhealthy	Liveness pr	
robe failed: dial tcp 172.17.0.6:3000: getsockopt: connection refused								
4m	4s	37	kubelet, minikube	spec.containers{webtest}	Warning	Unhealthy	Readiness p	
robes failed: dial tcp 172.17.0.6:3000: getsockopt: connection refused								
www. www. Macbook Pro% █								

Liveness and Readiness Probe

- HTTPGetAction:



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Sat Sep 16 15:54:10 2017

A screenshot of the Postman application. The left sidebar shows a history of requests: a GET request to "http://192.168.99.100:32500" (today), and several requests made on "September 8" (GET, POST, GET, GET, GET). The main panel shows a request configuration for "http://192.168.99.100:32500" using the "Builder" tab. The "Authorization" tab is selected, showing "No Auth". The "Headers" tab is also visible, listing "content-length", "content-type", "date", and "server". The status bar at the bottom indicates "Status: 200 OK" and "Time: 46 ms".



Liveness and Readiness Probe

- HTTPGetAction:

```
1  apiVersion: apps/v1
2  kind: Deployment
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      replicas: 1
14      selector:
15        matchLabels:
16          name: web
17          owner: Paparn_L
18          version: "1.0"
19          module: WebServer
20          environment: development
21      template:
22        metadata:
23          labels:
24            name: web
25            owner: Paparn_L
26            version: "1.0"
27            module: WebServer
28            environment: development
```

```
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
spec:
  containers:
    - name: webtest
      image: labdocker/cluster:webservicelite_v1
      ports:
        - name: webservice
          containerPort: 5000
          protocol: TCP
      readinessProbe:
        httpGet:
          # Optional "host: my-host" for set specific ip of Pods
          # Optional "scheme: HTTPS"
          #schema: HTTPS
          path: /
          port: webservice
          httpHeaders:
            - name: server
              value: "Werkzeug/0.12.2 Python/2.7.13"
      initialDelaySeconds: 15
      periodSeconds: 5
      timeoutSeconds: 10
      successThreshold: 1
      failureThreshold: 3
    livenessProbe:
      httpGet:
        # Optional "host: my-host" for set specific ip of Pods
        # Optional "scheme: HTTPS"
        #schema: HTTPS
        path: /
        port: webservice
        httpHeaders:
          - name: server
            value: "Werkzeug/0.12.2 Python/2.7.13"
      initialDelaySeconds: 15
      periodSeconds: 5
      timeoutSeconds: 10
      successThreshold: 1
      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
-tmpalte-hash=1098172621,version=1.0
Events:
FirstSeen   LastSeen   Count   From               SubObjectPath   Type   Reason   Message
-----   -----   ----   -----   -----   -----   -----   -----
9m         9m         1   default-scheduler
y assigned webtest-1098172621-zvd22 to minikube
  9m         9m         1   kubelet, minikube
 SetUp succeeded for volume "default-token-pcw4h"
  9m         9m         1   kubelet, minikube   spec.containers{webtest}
image "labdockey/cluster:webserviceelite_v1" already present on machine
  9m         9m         1   kubelet, minikube   spec.containers{webtest}
tainer
  9m         9m         1   kubelet, minikube   spec.containers{webtest}
tainer
  9m         9m         1   kubelet, minikube
```



Liveness and Readiness Probe

- HTTPGetAction:

```
29      readinessProbe:  
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-fpxw4 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-fpxw4_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	
praparns-MacBook-Pro%								

Workshop 1.6: 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". The browser's address bar also displays `192.168.99.100:32500`.

Below the browser window, a larger screenshot of the Postman application is shown. The Postman interface includes:

- Builder tab:** Active tab.
- Request URL:** `http://192.168.99.100:32500`
- Method:** GET
- Authorization:** Type: No Auth
- Headers:** (4) (highlighted)
- Body:** (empty)
- Cookies:** (empty)
- Tests:** (empty)
- Status:** 200 OK
- Time:** 46 ms

The left sidebar of Postman shows a history of requests:

- Today:**
 - GET `http://192.168.99.100:32500`
- September 8:**
 - GET `http://192.168.99.100/users/removeuser/99`
 - GET `http://192.168.99.100/users/1`
 - POST `http://192.168.99.100/users/insertuser`
 - GET `http://192.168.99.100/init`
 - GET `http://192.168.99.100/`
- August 27:** (empty)



Resource Management and HPA

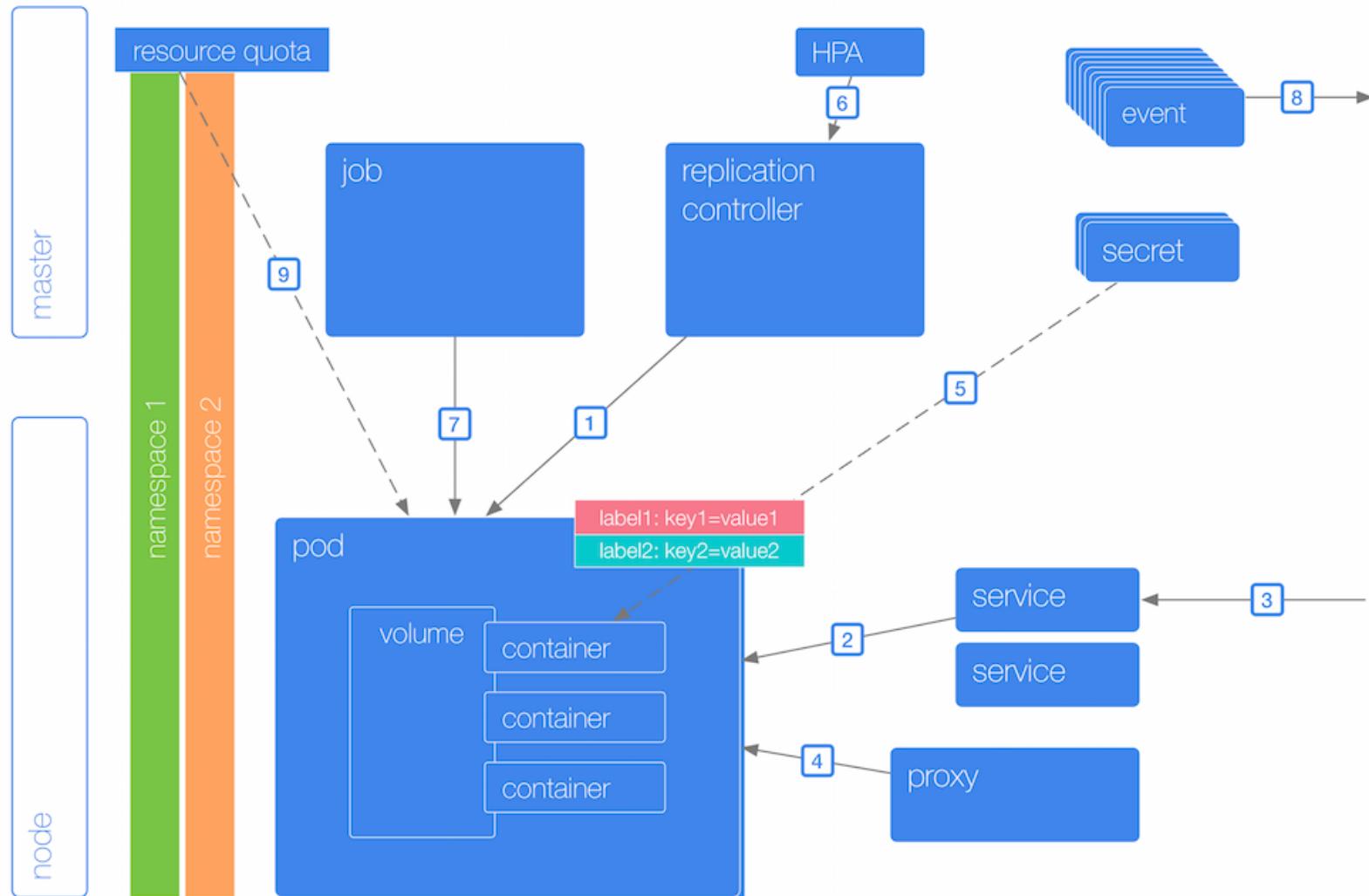


Kubernetes: Production Workload Orchestration



kubernetes
by Google

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
 - Use in case experiment/test purpose
 - 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”

System status

```
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
```

```
...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 [███████████] 0.0%
GiB Swap: 0.0/0.977 [ ] 0.0%
```

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 1.7: 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 [███████████] 1.58%
GiB Swap: 0.0/0.977 [ ] 0.0%
```



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



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 spec:
12   selector:
13     name: web
14     owner: Paparn_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
26
27   apiVersion: apps/v1
28   kind: Deployment
29   metadata:
30     name: webtest
31     labels:
32       name: web
33       owner: Paparn_L
34       version: "1.0"
```

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



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        -
praparns-MacBook-Pro%
```



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://0acf26d00b7b9f28b200fdfc
    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 [███████████]
GiB Swap: 0.0/0.977 [ ]
```

Resource Management and HPA

- Edit deployment for update cpu/memory Limits,request

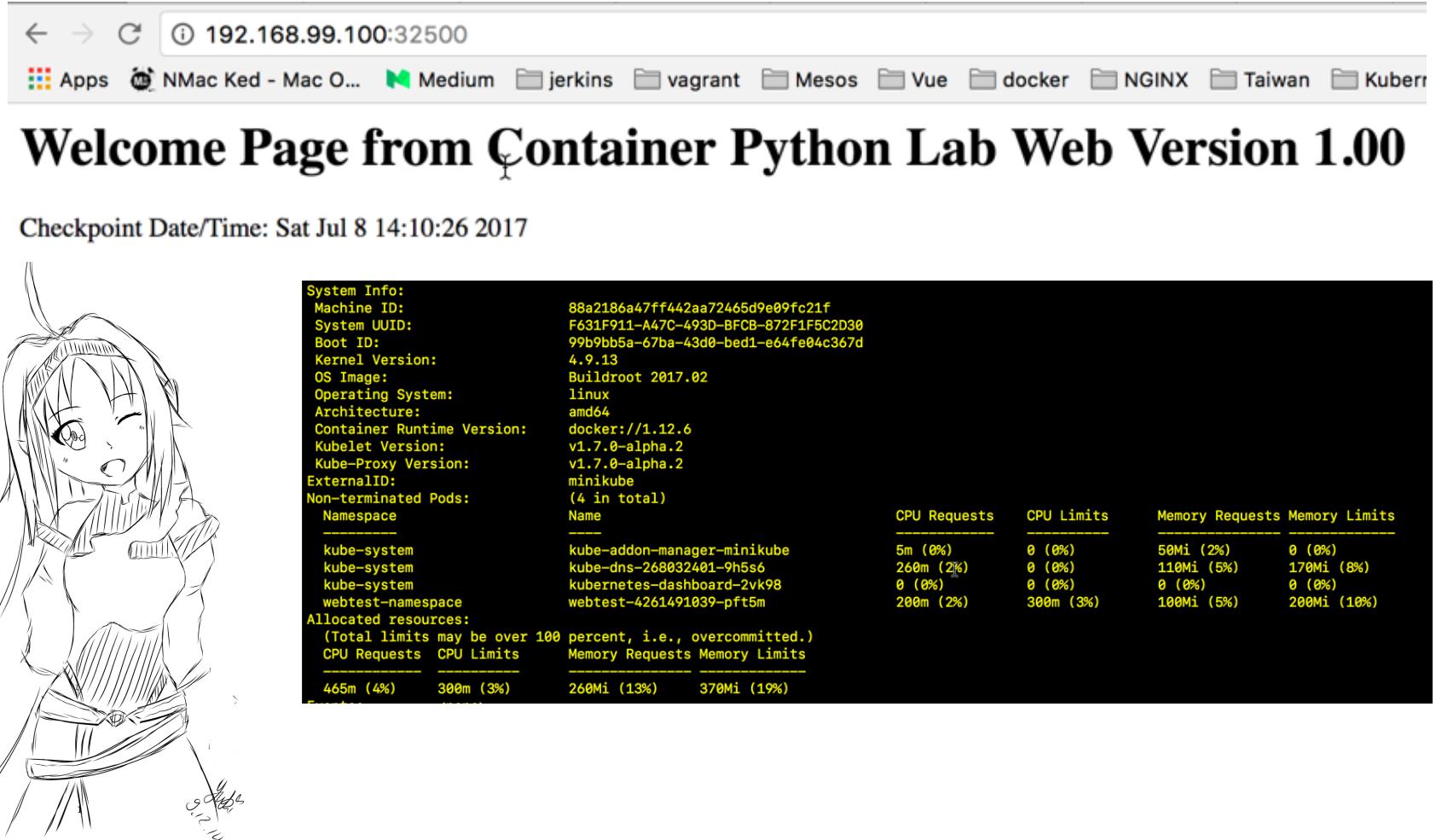
```
[praparns-MacBook-Pro% kubectl set resources deployment/webtest --limits=cpu="1",memory=1Gi --requests=cpu="0.8",memory=800Mi --record
--namespace=webtest-namespace
deployment "webtest" resource requirements updated

[praparns-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 1.7: 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-872F1F5C0D30				
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:					
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

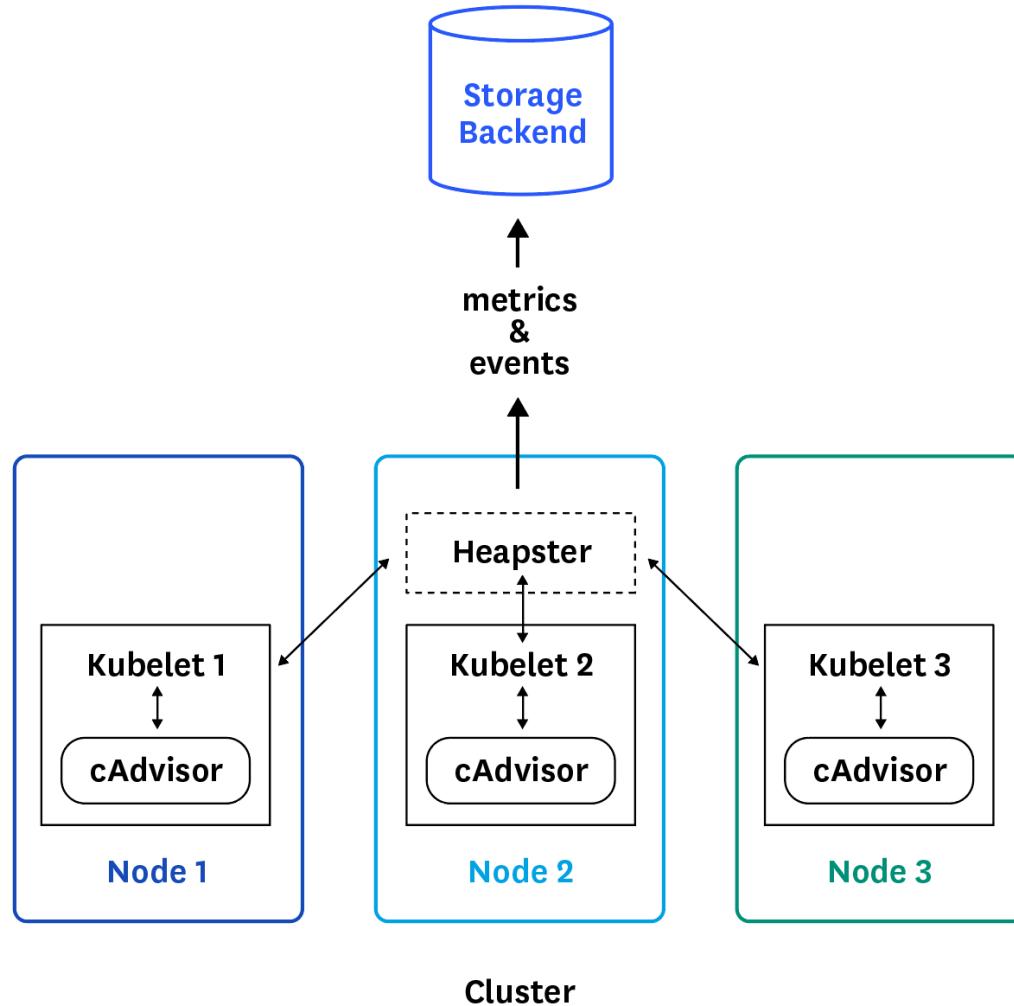
- Horizontal Pod Autoscaling (HPA)
 - The scale is easy to operate with "kubectl scale --replicas=XXX deployment/<name>" with single command
 - But..
 - How can you scale meet up/down actual required ?
 - HPA will response for monitor workload on Pods (Now base on CPU) and automatic trigger deployment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

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

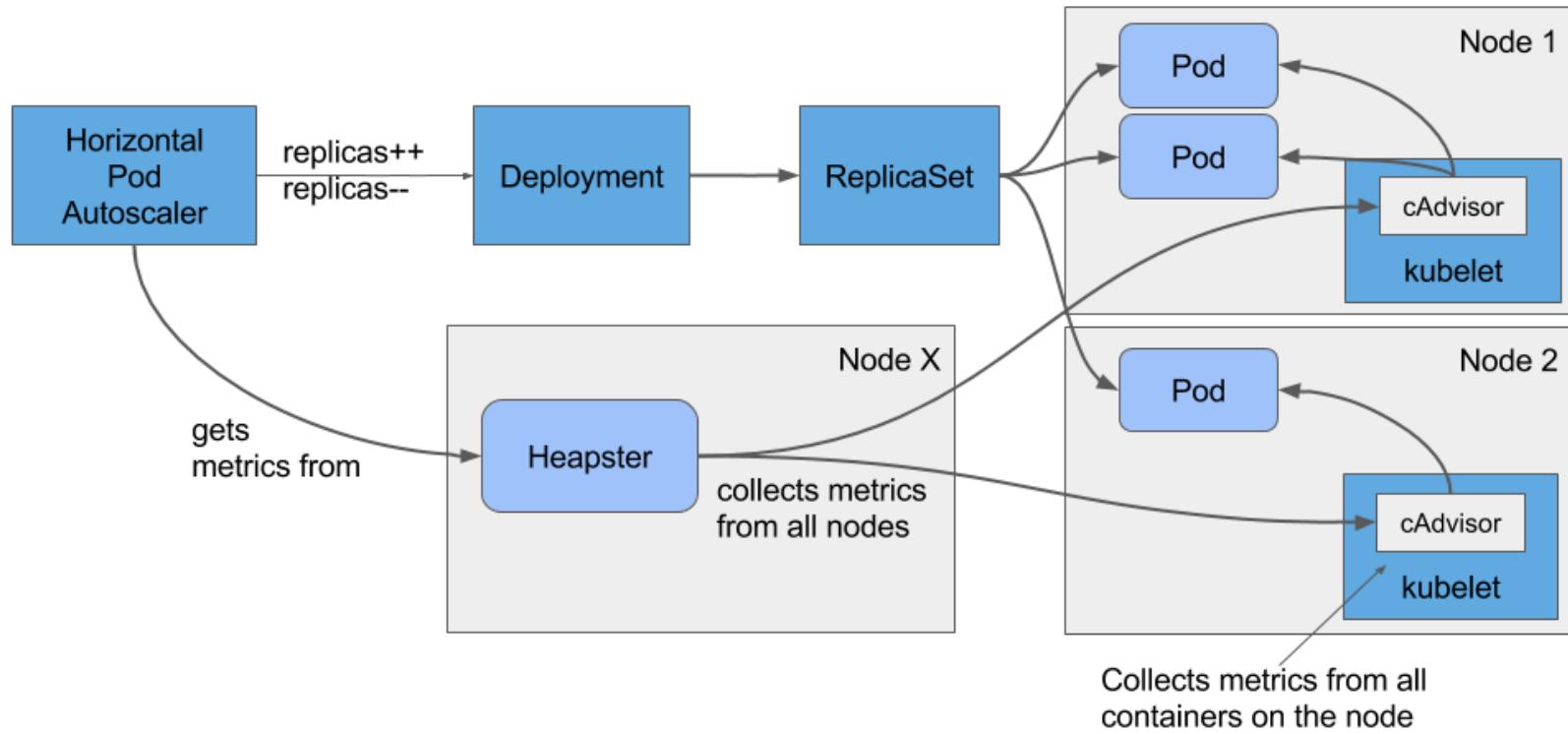
- Ex: kubectl autoscale deployment/webtest --min=1--max=10 --cpu-percent=80



Resource Management and HPA

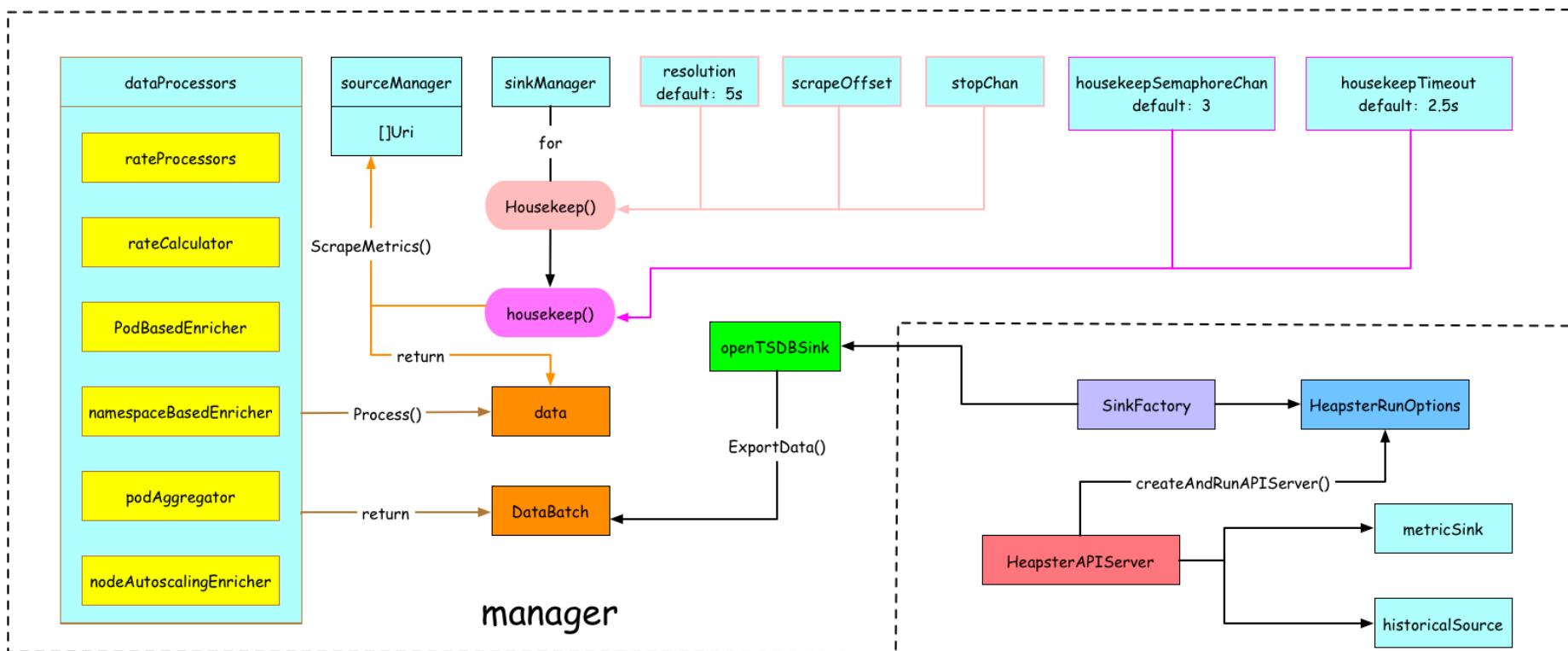


Resource Management and HPA



Resource Management and HPA

Heapster



© Jimmy Song <https://github.com/rootsongjc/kubernetes-handbook>



Resource Management and HPA

A screenshot of a GitHub issue thread. The URL is <https://github.com/kubernetes/kubernetes/issues/57673>. The thread discusses the transition from the cadvisor metrics pipeline to the Metrics API. A user named gjmzj comments on Dec 28, 2017, stating: "I found a solution in my case: kube-controller-manager's parameter --horizontal-pod-autoscaler-use-default value is true, while in k8s 1.8.x is false. change it to false and it works." Another user, MaciekPytel, responds with: "With --horizontal-pod-autoscaler-use-rest-clients=true HPA uses new instead of old way of getting metrics. Setting it to false as you did works, long term solution is to run metrics server as part of your cluster set up. Th <https://kubernetes.io/docs/tasks/debug-application-cluster/core-metrics-pipeline/>". There are 2 likes for the first comment and 1 like for the second.

A screenshot of a GitHub issue thread. The URL is <https://github.com/juju-solutions/bundle-canonical-kubernetes/issues/484>. The thread discusses the Autoscaler with CDK v1.9. A user named ktsakalozos opens the issue 2 days ago. They mention: "Starting with v1.9 HPA uses new resource metrics API that is not available in CDK out of the box. If you want to use autoscaler you should do a: tes-master controller-manager-extra-args="--horizontal-pod-autoscaler-use-rest-clients=false". Based on the discussion here [kubernetes/kubernetes#57673](https://github.com/kubernetes/kubernetes#57673) we can either set the --horizontal-pod-autoscaler-use-rest-clients=false or deploy this Metrics server: <https://kubernetes.io/docs/tasks/debug-application-cluster/core-metrics-pipeline/>". They also note: "Issue reported here as well: kubernetes/kubernetes#57996". A user named hyperbolic2346 comments 2 days ago: "The metrics server is deployed by default in kube-up. I would think we should deploy it as well." There is 1 like for the first comment.

Core metrics pipeline

Starting from Kubernetes 1.8, resource usage metrics, such as container CPU and memory usage, are available in Kubernetes through the Metrics API. These metrics can be either accessed directly by user, for example by using `kubectl top` command, or used by a controller in the cluster, e.g. Horizontal Pod Autoscaler, to make decisions.

The Metrics API

Through the Metrics API you can get the amount of resource currently used by a given node or a given pod. This API doesn't store the metric values, so it's not possible for example to get the amount of resources used by a given node 10 minutes ago.

The API is no different from any other API:

- it is discoverable through the same endpoint as the other Kubernetes APIs under `/apis/metrics.k8s.io/` path
- it offers the same security, scalability and reliability guarantees

The API is defined in k8s.io/metrics repository. You can find more information about the API there.

Note: The API requires metrics server to be deployed in the cluster. Otherwise it will be not available.

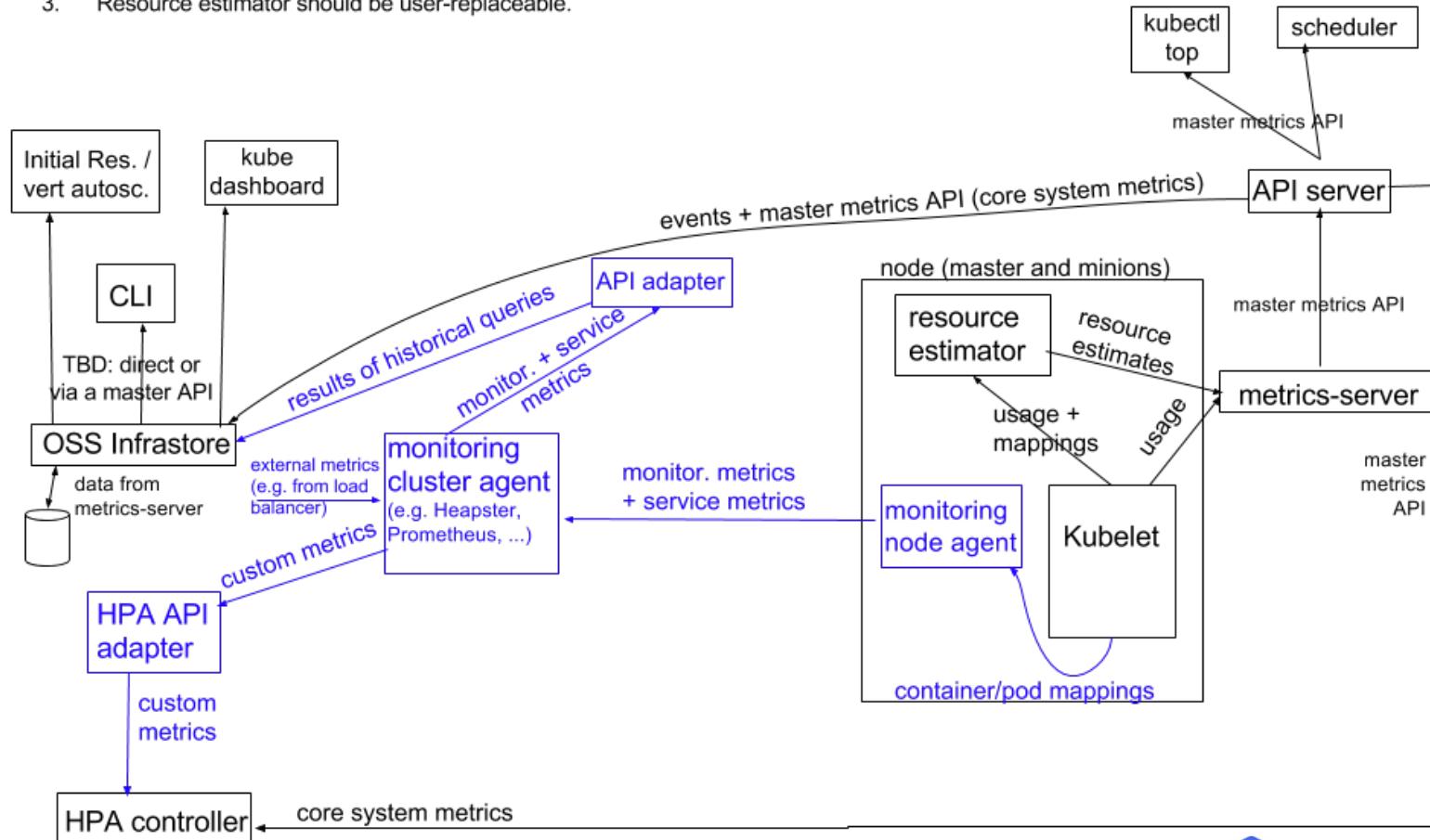


Resource Management and HPA

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

- 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
```

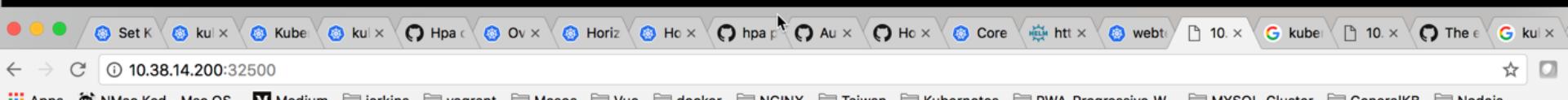
```
27  apiVersion: apps/v1
28  kind: Deployment
29  metadata:
30    name: webtest
31    labels:
32      name: web
33      owner: Praparn_L
34      version: "1.0"
35      module: WebServer
36      environment: development
37 spec:
38   replicas: 1
39   selector:
40     matchLabels:
41       name: web
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: labdocker/cluster:webservicelite_v1
58       resources:
59         requests:
60           cpu: "200m"
61       ports:
62         - containerPort: 5000
63           protocol: TCP
```



Resource Management and HPA

- Create deployment for python

```
docker@kubernetes-ms:~$ kubectl create -f https://raw.githubusercontent.com/praparn/kubernetes_20170128/master/WorkShop_1.7_Resource_Management_and_HPA/webtest_hpa.yml
service "webtest" created
deployment "webtest" created
docker@kubernetes-ms:~$ kubectl get pods/webtest -o wide
Error from server (NotFound): pods "webtest" not found
docker@kubernetes-ms:~$ kubectl get svc/webtest -o wide
NAME      CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE      SELECTOR
webtest   10.107.217.17    <nodes>        5000:32500/TCP  5s      environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
docker@kubernetes-ms:~$ kubectl get deployment/webtest -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE      CONTAINER(S)   IMAGE(S)        SELECTOR
webtest   1         1         1           1           18s     webtest       labdockerc/cluster:webserviceelite_v1   environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
Server, name=web, owner=Praparn_L, version=1.0
docker@kubernetes-ms:~$ kubectl get svc/webtest -o wide
NAME      CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE      SELECTOR
webtest   10.107.217.17    <nodes>        5000:32500/TCP  19s     environment=development,module=WebServer,name=web,owner=Praparn_L,version=1.0
docker@kubernetes-ms:~$ 
```



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Fri Feb 2 16:27:53 2018



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
  Events:      <none>
  ScalingLimited  False   DesiredWithinRange
  Message:    the last scale time was sufficiently old as to warrant a new scale
               the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  Message:    the desired count is within the acceptable range
Events:               <none>
docker@kubernetes-ms:~$
```



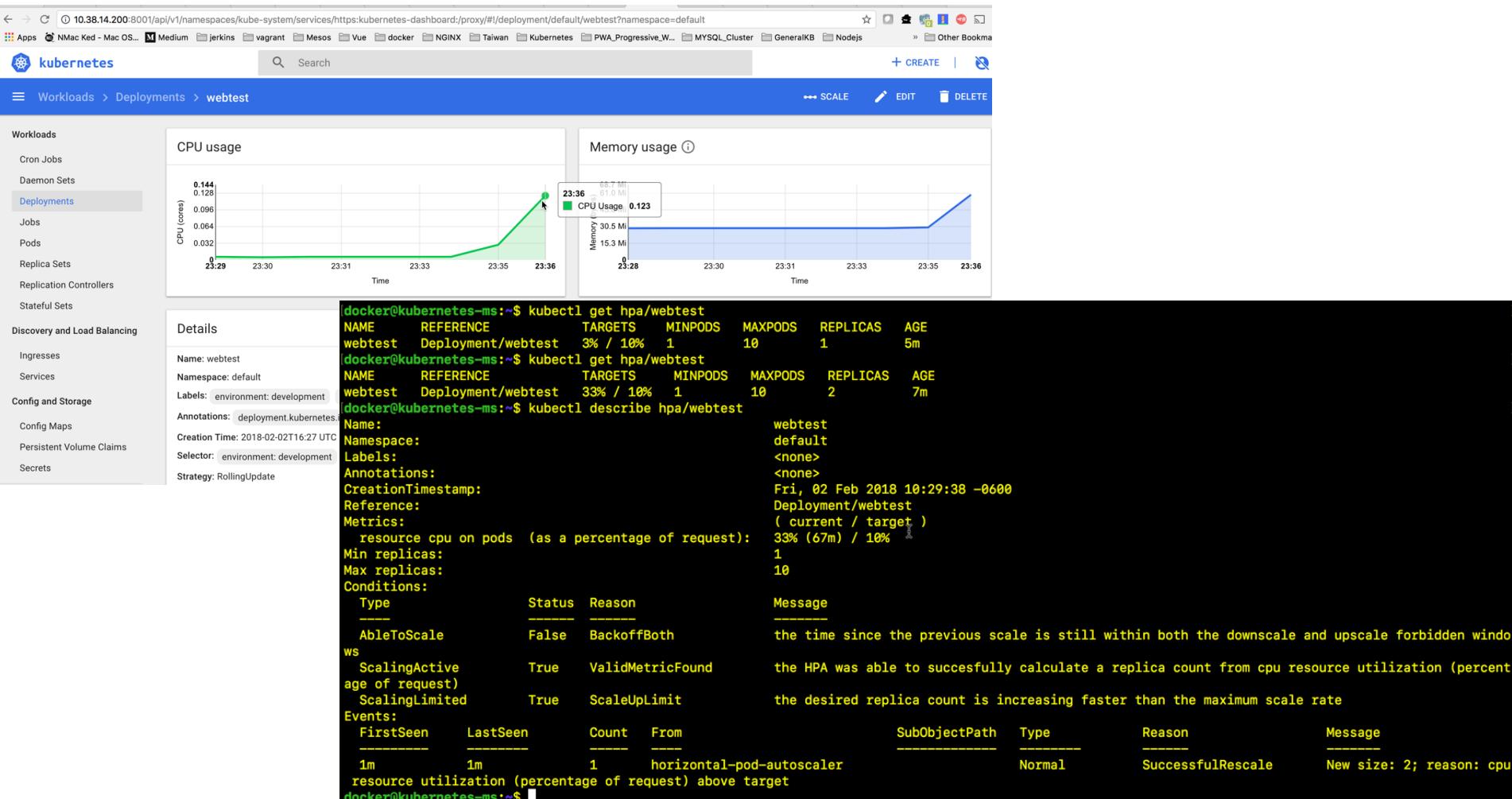
Resource Management and HPA

- 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        58%
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:~$
```

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)



Outline Day 2

- Fundamental of Kubernetes (Con't)
 - ConfigMap Secret
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- 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 บน 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
 - ConfigMap Secret
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- 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



Outline Day 2

- Fundamental of Kubernetes
 - ConfigMap Secret
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- 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



Outline Day 2

- LAB Sheet
- <https://tinyurl.com/y84sh9zv>

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



ConfigMap and Secret



Kubernetes: Production Workload Orchestration



kubernetes
by Google

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>
```

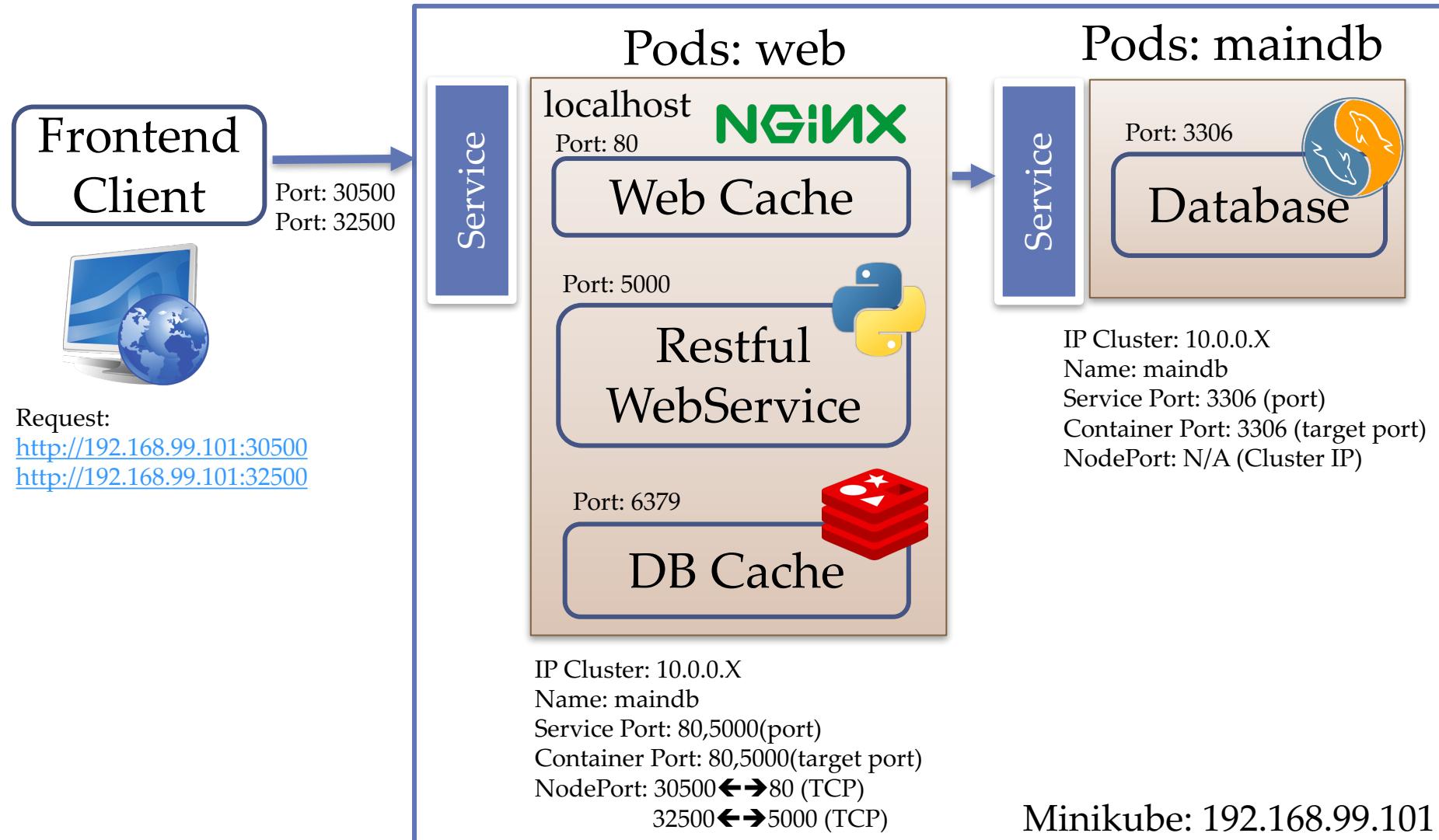
- Ex: “kubectl create secret generic databasemodule_secret --from-file=./username.txt --from-file=./password.txt”
- Ex: “kubectl create -f databasemodule_secret.yml”

```
|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

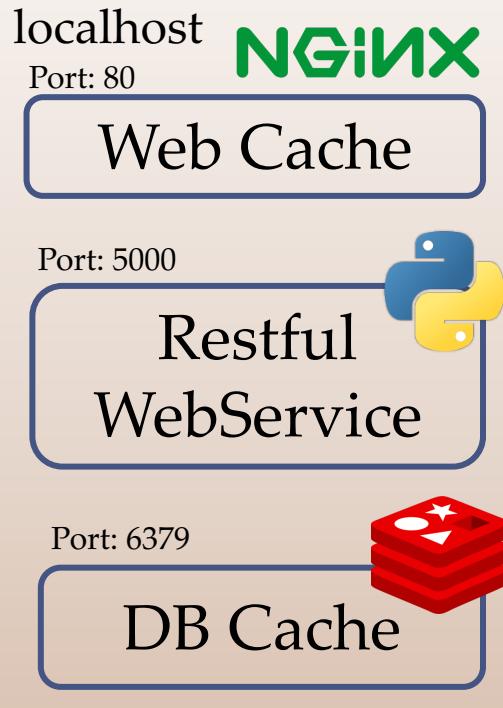
Service



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

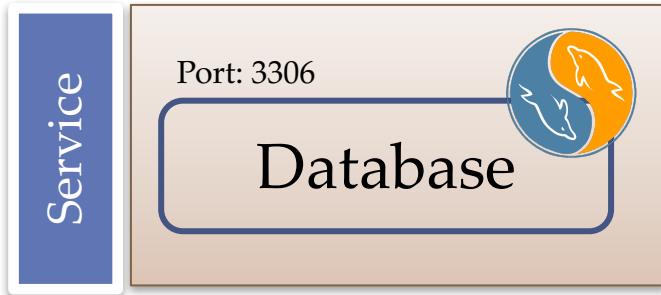
```
6 CACHE_DB = redis.Redis(host=os.environ.get('REDIS_HOST', 'cachedb'), port=6379)
7 db = MySQLdb.connect("maindb","root","password")
8 MAIN_DB = db.cursor()
```

Pods.yml: webmodule_pod.yml

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



ConfigMap and Secret



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: cGFzc3dvcnQ=
```

Deploy.yaml:

webmodule_deploy_config.yaml

```
46 spec:
47   containers:
48     - name: cachedb
49       image: labdocker/redis:latest
50       ports:
51         - containerPort: 6379
52           protocol: TCP
53     - name: webservice
54       image: labdocker/cluster:webservice
55       env:
56         - name: REDIS_HOST
57           valueFrom:
58             configMapKeyRef:
59               name: webmodule_configmap
60               key: REDIS_HOST
61         - name: username
62           valueFrom:
63             secretKeyRef:
64               name: databasemodule-secret
65               key: username
66         - name: password
67           valueFrom:
68             secretKeyRef:
69               name: databasemodule-secret
70               key: password
71       ports:
72         - containerPort: 5000
73           protocol: TCP
```

ConfigMap.yaml:

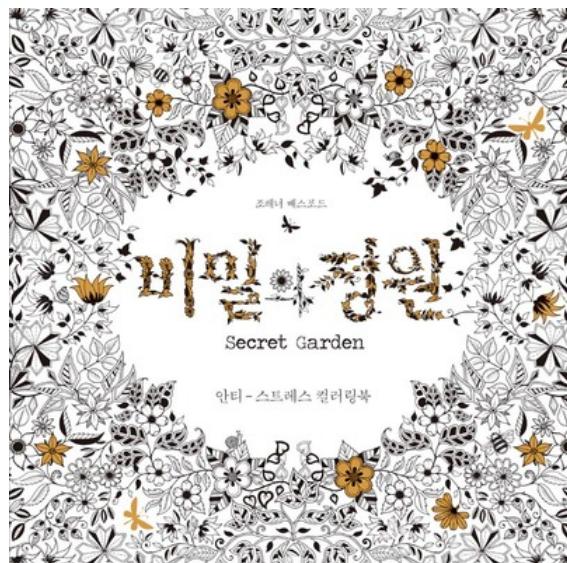
webmodule_configmap.yaml

```
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 2.1: 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
```



Job and Cron Jobs



Kubernetes: Production Workload Orchestration



kubernetes
by Google

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

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

- “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

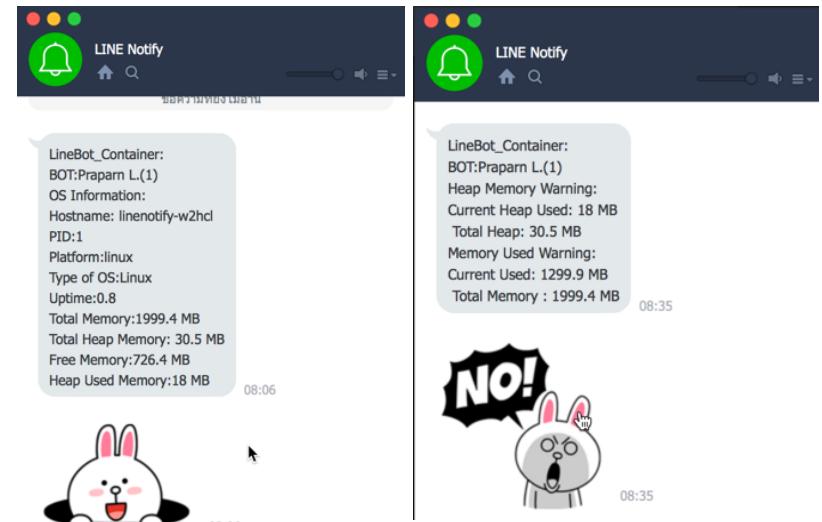


Job and Cron Job

Job.yml: job.yml

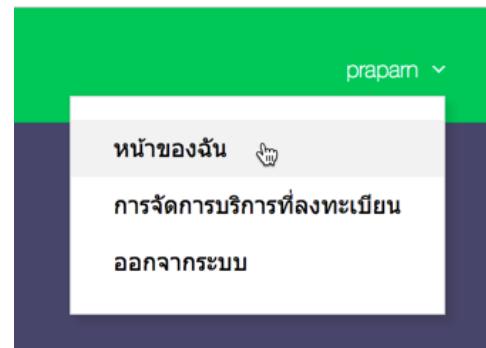
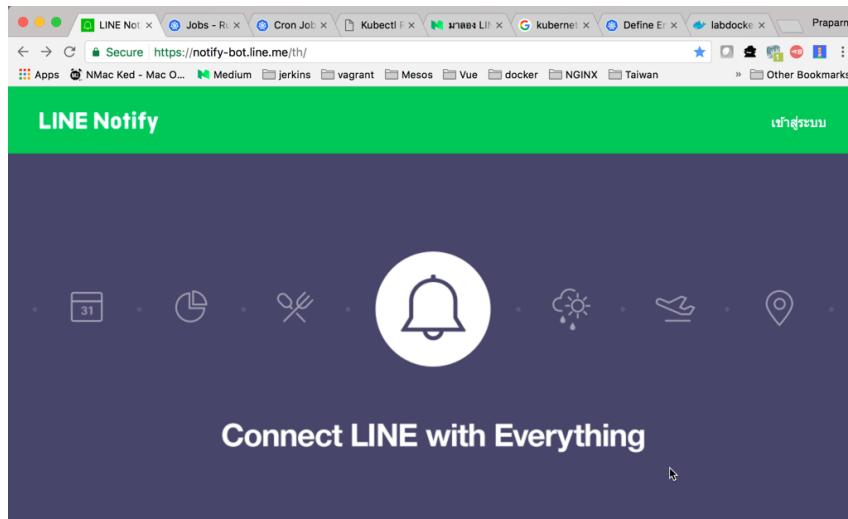
```
1  apiVersion: batch/v1
2  kind: Job
3  metadata:
4      name: linenotify
5      labels:
6          name: linenotify
7          owner: Praparn_L
8          version: "1.0"
9          module: Job
10         environment: development
11
12     spec:
13         template:
14             metadata:
15                 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: "40"
27                             - name: MEM_HIGH
28                             value: "20"
29                             - name: SH_OS
30                             value: "Y"
31                             - name: TOKEN
32                             value: "EIEeqyillzkpaCo1R0rebZTcGbIyzDZHpojcdd0t6CX"
33             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=Praparn_L
                 version=1.0
Annotations:    <none>
```



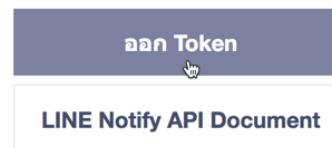
Workshop 2.2: Job and CronJob

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



ออก Access Token (สำหรับผู้พัฒนา)

เมื่อใช้ Access Token แบบบุคคล จะสามารถตั้งค่าการแจ้งเตือนได้โดยไม่ต้องลงทะเบียนกับเว็บเซอร์วิส



Workshop 2.2: Job and CronJob

ออก Token

โปรดใส่ชื่อ Token (จะแสดงเมื่อมีการแจ้งเตือน)

LINEBOT

โปรดเลือกห้องแขวงที่ต้องการส่งข้อความแจ้งเตือน

Search by group name

 รับการแจ้งเตือนแบบตัวต่อตัวจาก LINE Notify

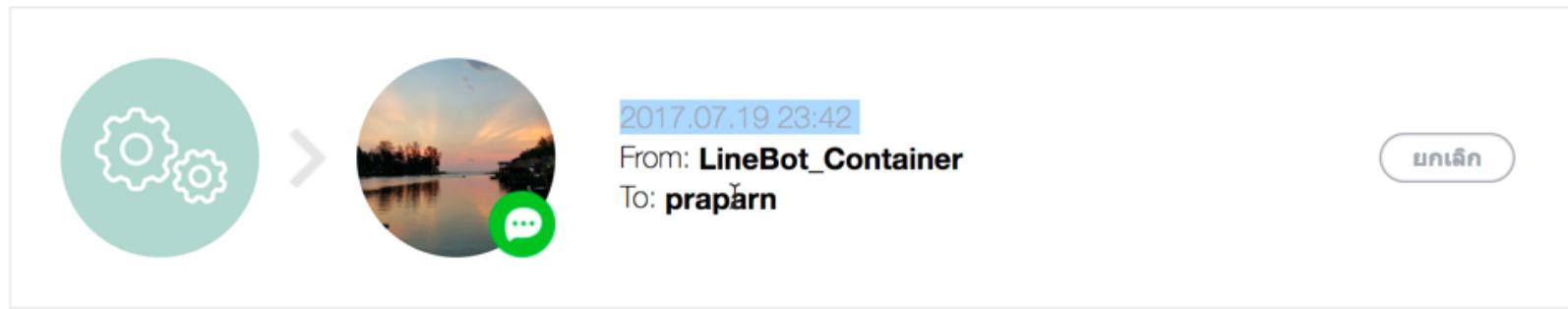
Token ที่ออก

zHOlcJCcpIIS8mEedn 

ถ้าออกจากหน้านี้ ระบบจะไม่แสดง Token ที่ออกใหม่ล่าสุดอีกต่อไป โปรดศึกษา Token ก่อนออกจากหน้านี้



คัดลอก **ปิด**



Workshop 2.2: 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: "EIEeqyillzkpaCo1R0rebZTcGbIyzDZHp0jcd0t6CX"
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

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)
```



[Documentation](#) [Blog](#) [Partners](#) [Community](#) [Case Studies](#)

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*.

The job is responsible for retrying pods, parallelism among pods it creates, and determining the success or failure of the set of pods. A cron job does not examine pods at all.



Job and Cron Job

- AutoCleanup Job

 This repository Search Pull requests Issues Marketplace Explore + ⚙

lwolf / kube-cleanup-operator

Code Issues 2 Pull requests 1 Projects 0 Wiki Insights

Kubernetes Operator to automatically delete completed Jobs and their Pods

kubernetes kubernetes-operator golang

18 commits 1 branch 2 releases 3 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

lwolf committed 26 days ago go fmt, update README, more flexible makefile	Latest commit 1183f6f 26 days ago
cmd go fmt, update README, more flexible makefile	26 days ago
deploy Fix rbac permissions, improve readme	5 months ago
pkg/controller go fmt, update README, more flexible makefile	26 days ago
.gitignore initial commit	5 months ago
.travis.yml Feautre/makefile (#10)	26 days ago
Dockerfile Feautre/makefile (#10)	26 days ago
Gopkg.lock initial commit	5 months ago
Gopkg.toml initial commit	5 months ago
LICENSE Initial commit	5 months ago
Makefile go fmt, update README, more flexible makefile	26 days ago
README.md go fmt, update README, more flexible makefile	26 days ago

README.md

Kubernetes cleanup operator

build passing

Experimental Kubernetes Operator to automatically delete completed Jobs and their Pods. Controller listens for changes in Pods created by Jobs and deletes it on Completion.

Some defaults:

- All Namespaces are monitored by default
- Only Pods created by Jobs are monitored

<https://github.com/lwolf/kube-cleanup-operator>



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

minikube dashboard

The screenshot shows the 'Workloads' section of the minikube dashboard. On the left, a sidebar lists various Kubernetes components like Namespaces, Nodes, and Storage Classes. Under 'Workloads', it shows Deployments and Replica Sets. The Deployments section lists 'maindb' and 'web'. The 'maindb' deployment has 2 pods, with one being green (running) and one grey (pending). The 'web' deployment also has 2 pods, both green. The Replica Sets section shows a single 'maindb-1334333237' entry with 1 pod running.

The screenshot shows the 'Edit a Service' dialog. It displays the JSON configuration for a service named 'kubernetes'. The configuration includes:

```
kind: Service
apiVersion: v1
metadata:
  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:
  component: apiserver
  provider: kubernetes
spec:
  ports: [1]
```

At the bottom right of the dialog are 'CANCEL' and 'UPDATE' buttons.



Workshop 2.3: Log and Monitoring



Ingress Network



Kubernetes: Production Workload Orchestration



kubernetes
by Google

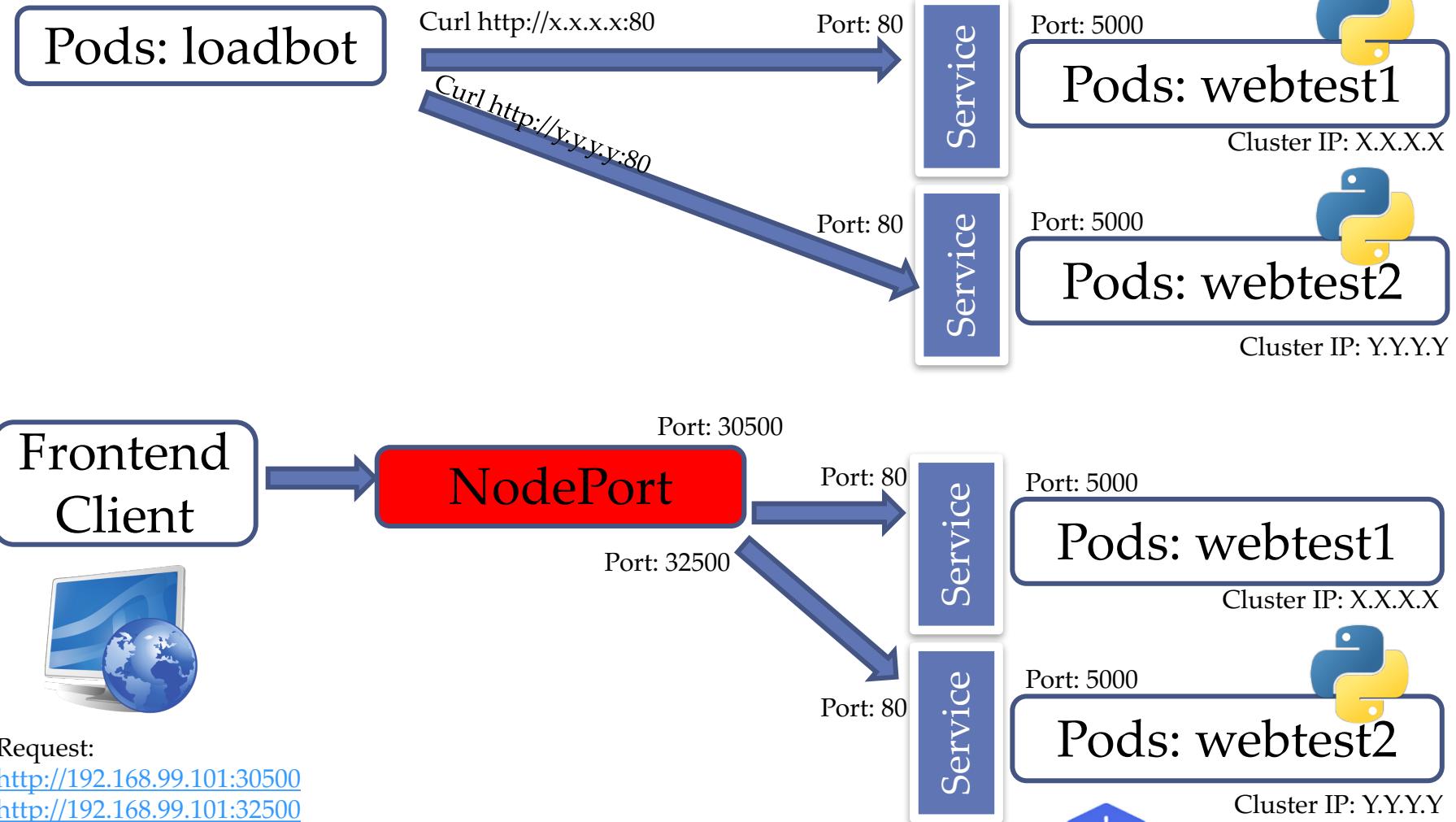
Ingress Network

- 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

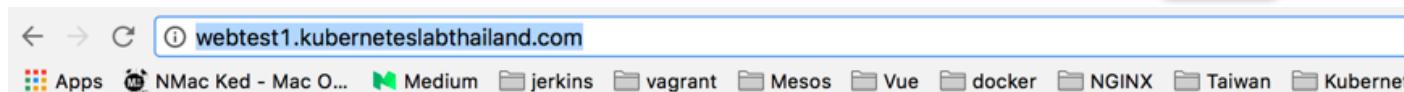
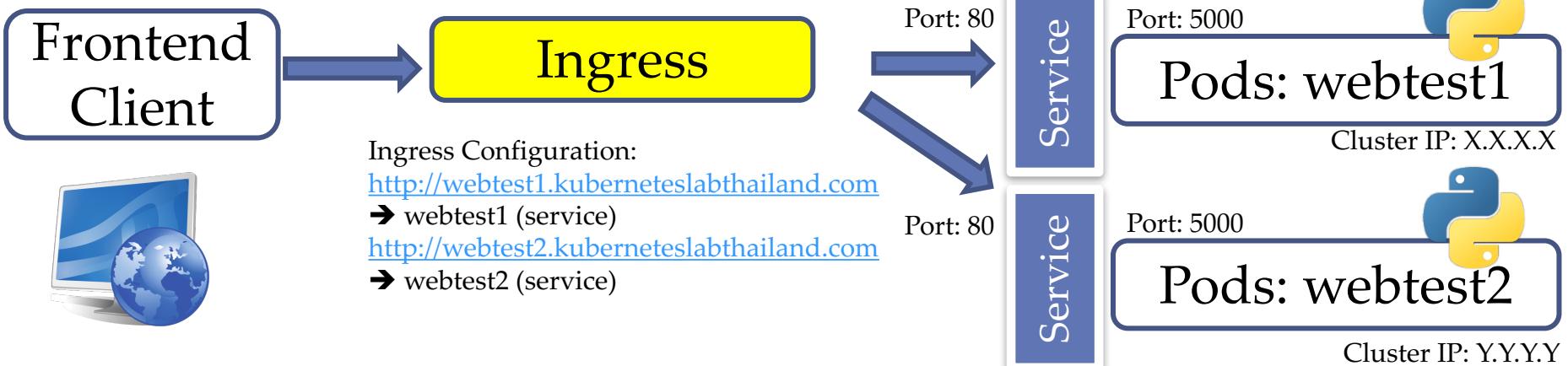
Ingress Network

- Service:



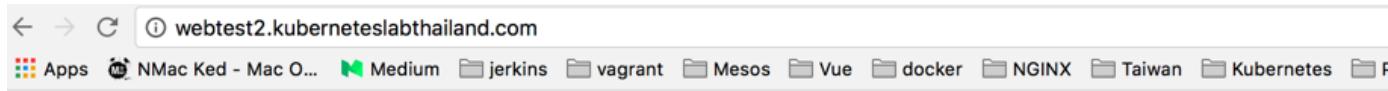
Ingress Network

- 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

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
```

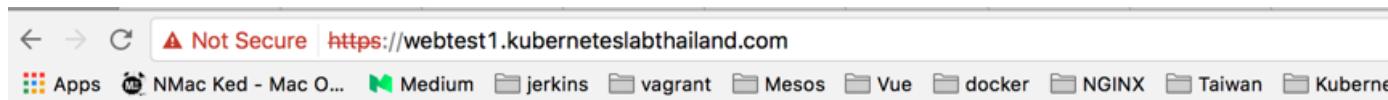
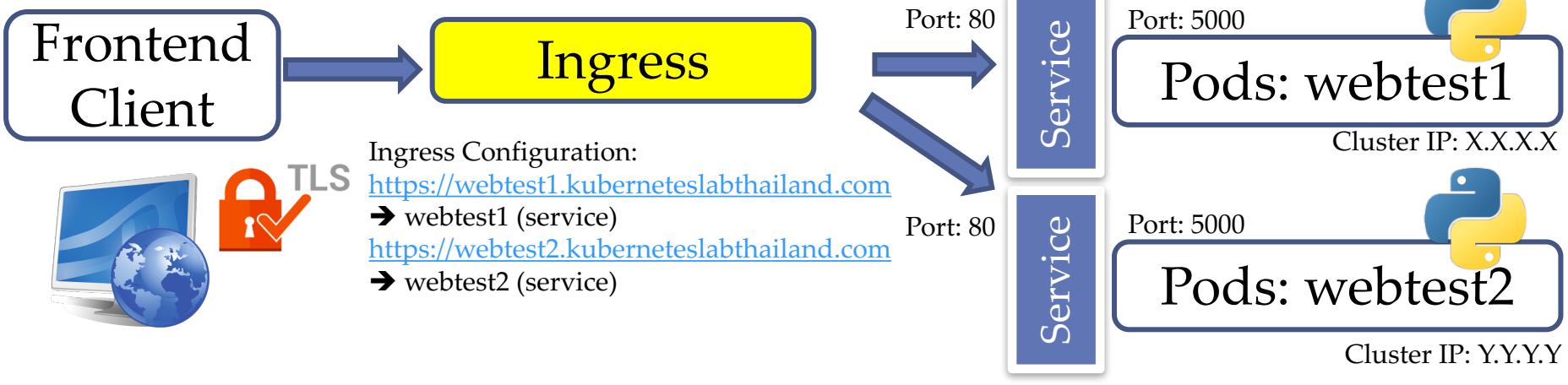
Ingress: ingresswebtest

```
1  apiVersion: extensions/v1beta1
2  kind: Ingress
3  metadata:
4    name: ingresswebtest
5  spec:
6    rules:
7      - host: webtest1.kuberneteslabthailand.com
8        http:
9          paths:
10            - backend:
11              serviceName: webtest1
12              servicePort: 80
13      - host: webtest2.kuberneteslabthailand.com
14        http:
15          paths:
16            - backend:
17              serviceName: webtest2
18              servicePort: 80
```



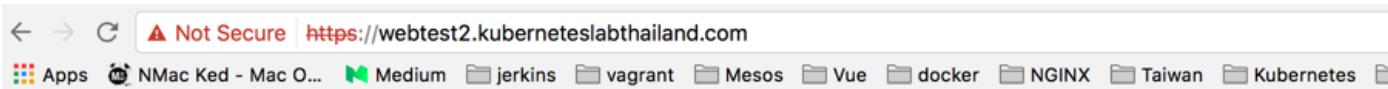
Ingress Network

- Ingress (TLS):



Welcome Page from Container Python Lab Web Version 1.00

Checkpoint Date/Time: Wed Jul 26 16:36:39 2017



Welcome Page from Container Python Lab Web Version 1.51 RC

Checkpoint Date/Time: Wed Jul 26 16:39:09 2017



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: Praparn_L
8     version: "1.0"
9   module: WebServer
10  environment: development
11 spec:
12   selector:
13     name: webtest1
14     owner: Praparn_L
15     version: "1.0"
16     module: WebServer
17     environment: development
18
19 ports:
20 - port: 80
21   name: http
22   targetPort: 5000
23   protocol: TCP
24
```

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 spec:
12   selector:
13     name: webtest2
14     owner: Praparn_L
15     version: "1.0"
16     module: WebServer
17     environment: development
18
19 ports:
20 - port: 80
21   name: http
22   targetPort: 5000
23   protocol: TCP
24
```

Ingress: ingresswebtest

```
1 apiVersion: extensions/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              - backend:
18                serviceName: webtest1
19                servicePort: 80
20        - host: webtest2.kuberneteslabthailand.com
21          http:
22            paths:
23              - backend:
24                serviceName: webtest2
25                servicePort: 80
```

Secret: webtest1

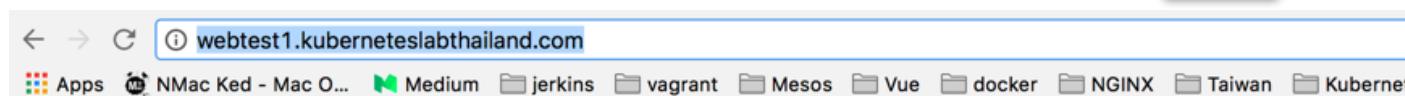
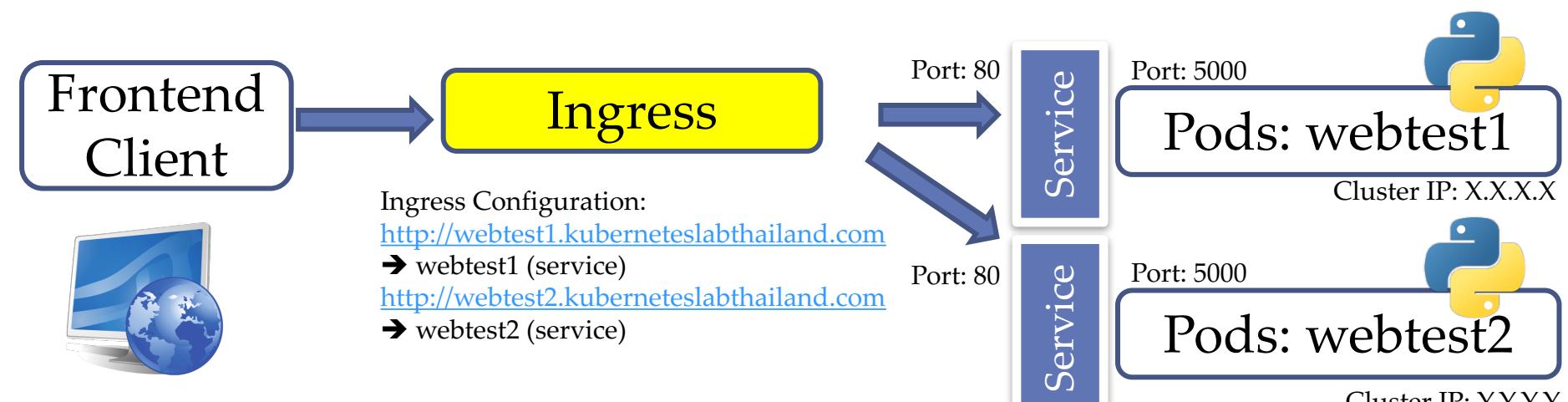
```
1 apiVersion: v1
2 data:
3   tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURMVENDQWhXZ0F3SUJBZ0lK
4   tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkfURSBLRVktLS0tLQpNSU1Fb3dJQkFB50NBUVB
5 kind: Secret
6 metadata:
7   name: webtest1secret
8   namespace: default
9 type: Opaque
```

Secret: webtest2

```
1 apiVersion: v1
2 data:
3   tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURMVENDQWhXZ0F3SUJBZ0lK
4   tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkfURSBLRVktLS0tLQpNSU1FcEFJQkFB50NBUVB
5 kind: Secret
6 metadata:
7   name: webtest2secret
8   namespace: default
9 type: Opaque
```

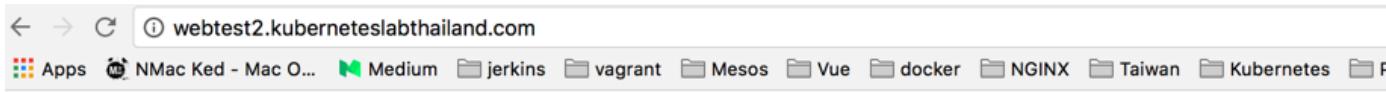


Workshop 2.4: Ingress Network



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



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
 - Etc
 - Phase 4: Join node to cluster system
 - `kubeadm join <option>`



Kubernetes in real world

The screenshot shows a web browser window with the following content:

- Header:** A navigation bar with a search field containing "labs.play-with-k8s.com". Below it is a bookmark bar with various links including "NMac Ked - Mac OS...", "Medium", "jenkins", "vagrant", "Mesos", "Vue", "docker", "NGINX", "Taiwan", "Kubernetes", "PWA_Progressive_W...", "MYSQL_Cluster", "GeneralKB", "Nodejs", "mongodb", "POSTFIX", and "Other Bookmarks".
- Landing Page:** A "Welcome!" message followed by a CAPTCHA challenge: "Before starting we need to verify you are a human". A "Create session" button is present.
- Kubernetes Logo:** A large blue hexagon with a white steering wheel icon.
- Main Logo:** The text "kubernetes" in large bold letters, followed by "by Google™".
- Session Management:** A sidebar for session "d7498ce6-f563-4e73-b7e7-62503f1db77b#d7498ce6_node1". It shows the timestamp "03:52:40", a "CLOSE SESSION" button, and an "Instances" section with a "+ ADD NEW INSTANCE" button. A user icon and the name "d7498ce6_node1" are listed.
- Node Details:** A main panel for node "d7498ce6_node1" showing IP "10.0.12.3", Memory usage "27.21% (1.087GiB / 3.996GiB)", and CPU usage "27.32%". It includes a "DELETE" button.
- Terminal:** A terminal window showing command-line output:

```
[node1 /]$ kubectl get svc
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 /]$
```

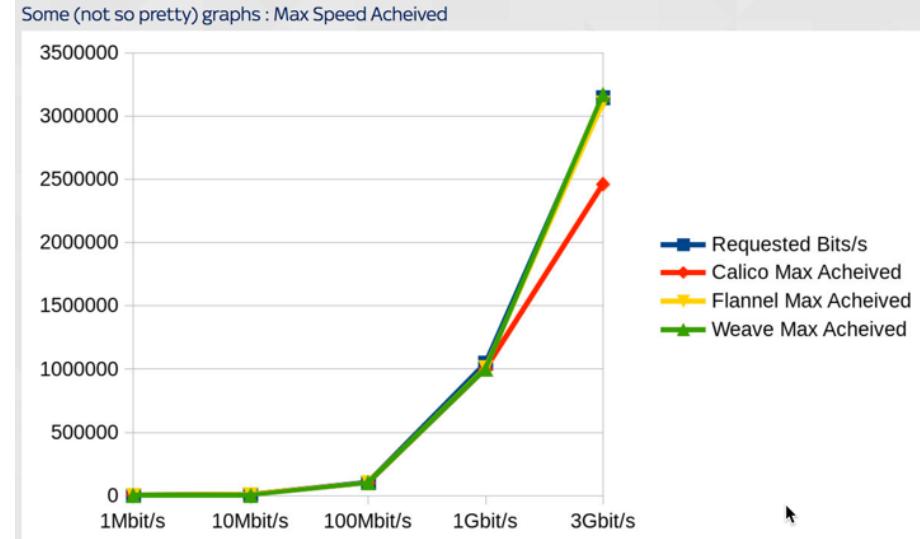
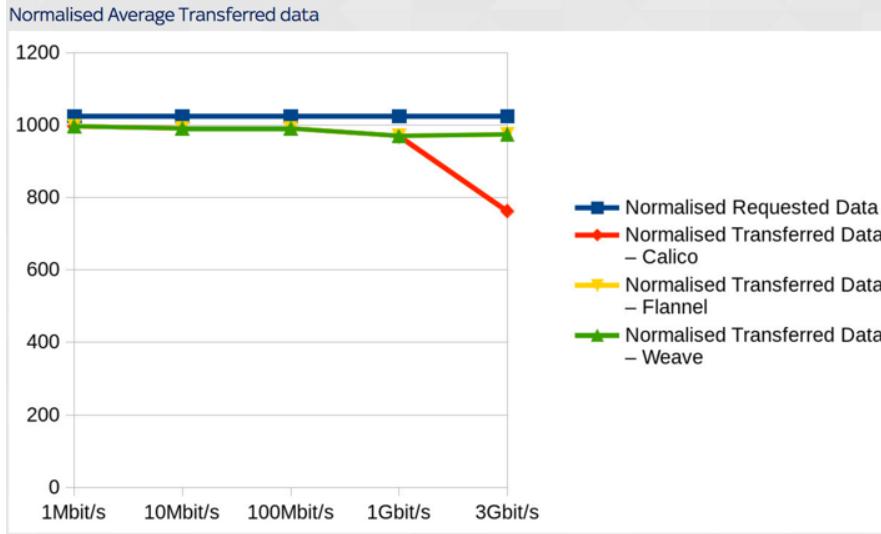
Kubernetes: Production Workload Orchestration



kubernetes
by Google

Kubernetes in real world

- Network Optional



And Finally... Interesting lessons learned

- Be sure to check your hosts have the "Check Source/Dest" disabled if you're running an overlay network on AWS or you may have issues. Unfortunately there's no way to set this on an ASG/Launch config level, so if you want to auto-scale, you'll need to give the hosts permission to set this on boot.
- Calico did happen to go non-responsive on one host machine multiple times during my tests, but I didn't really have time to investigate why, and I'd got the performance data I needed, but it's another reason I wouldn't feel comfortable choosing that tech.
- Keep your eye on your network as well as CPU utilisation on your container clusters if running on AWS – consider upgrading to the more expensive hosts if you see latency issues.
- Flannel can be a pain to get running, especially in AWS, I struggled to install it in the same way as the others to keep the data equivalent but fell back to running it in the "CoreOS" way.
- Once you hit 100Mbps of UDP traffic, packet loss hits you, hard. This was borne out in testing not only on AWS but also on VMWare, so it isn't something "new" about overlay networks but is interesting.



[Documentation](#) [Blog](#) [Partners](#) [Community](#) [Case Studies](#)

If you are on another architecture than amd64, you should use the flannel or Weave Net overlay networks as described in [the multi-platform section](#)

Once a pod network has been installed, you can confirm that it is working by checking that the kube-dns pod is Running in the output of `kubectl get pods --all-namespaces`. And once the kube-dns pod is up and running, you can continue by joining your nodes.

If your network is not working or kube-dns is not in the Running state, check out the [troubleshooting section](#) below.

<http://engineering.skybettingandgaming.com/2017/02/03/overlay-network-performance-testing/>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

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

- 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.f5534344a6ffc46
[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.f5534344a6ffc46 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:~# ]
```

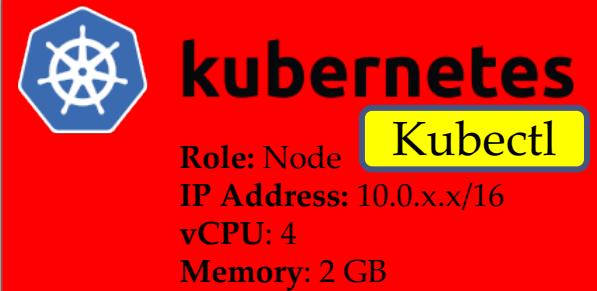
```
[praparn@kubernetes-ms:~$ kubectl get node
NAME        STATUS    AGE      VERSION
kubernetes-1  Ready    5m       v1.7.0
kubernetes-2  Ready    4m       v1.7.0
kubernetes-ms Ready   11m       v1.7.0
praparn@kubernetes-ms:~$ ]
```



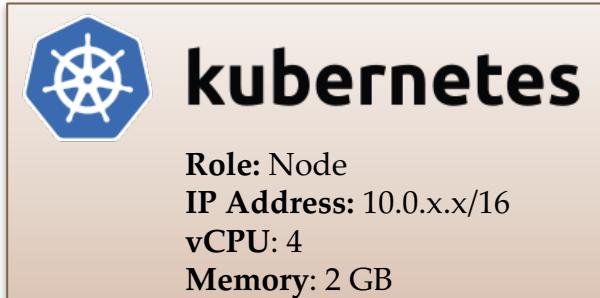
Workshop 2.5: Kubernetes Lab

Style 1: Google Cloud

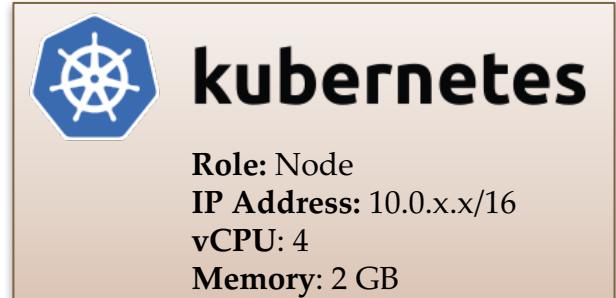
Machine: Kubernetes_MS



Machine: Kubernetes_1



Machine: Kubernetes_2



Role: Client



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   Normal  Scheduled
                Successfully assigned webtest to kubernetes-ms
12m        12m       1     kubelet, kubernetes-ms  Normal  SuccessfulMountVolume
                MountVolume.SetUp succeeded for volume "default-token-1ltqv"
12m        12m       1     kubelet, kubernetes-ms  Normal  Pulling
                pulling image "labdocker/cluster:webservicelite"
11m        11m       1     kubelet, kubernetes-ms  Normal  Pulled
                Successfully pulled image "labdocker/cluster:webservicelite"
11m        11m       1     kubelet, kubernetes-ms  Normal  Created
                Created container
11m        11m       1     kubelet, kubernetes-ms  Normal  Started
                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:webservicelite
    Image:                 labdocker/cluster:webservicelite
    Image ID:               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



Taint and Tolerations

Taint on Node:

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

Pods YAML File:

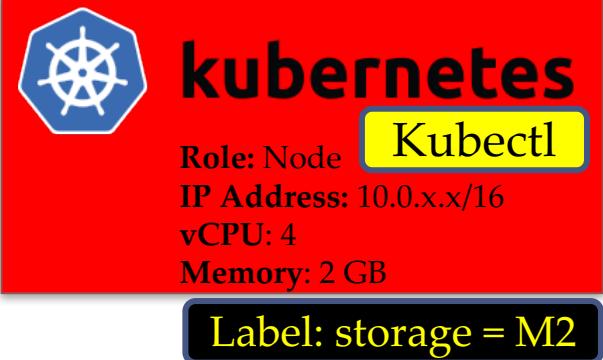
```
1  apiVersion: "v1"
2  kind: Pod
3  metadata:
4    name: webtest
5    labels:
6      name: web
7      owner: Praparn_L
8      version: "1.0"
9      module: WebServer
10     environment: development
11 spec:
12   containers:
13     - name: webtest
14       image: labdocker/cluster:webservicelite
15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

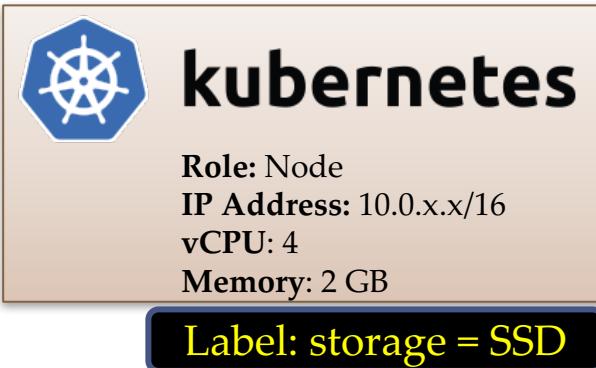


Workshop 2.6: Orchestrator Assignment

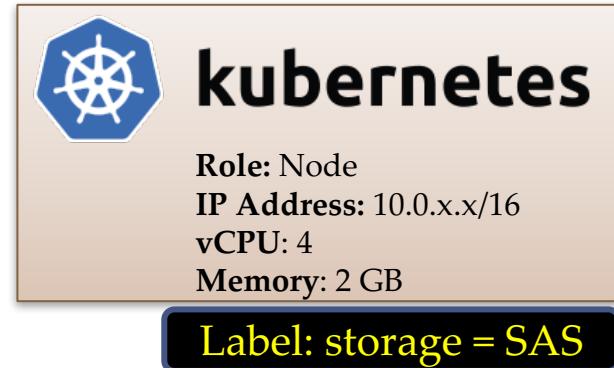
Machine: Kubernetes_MS



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



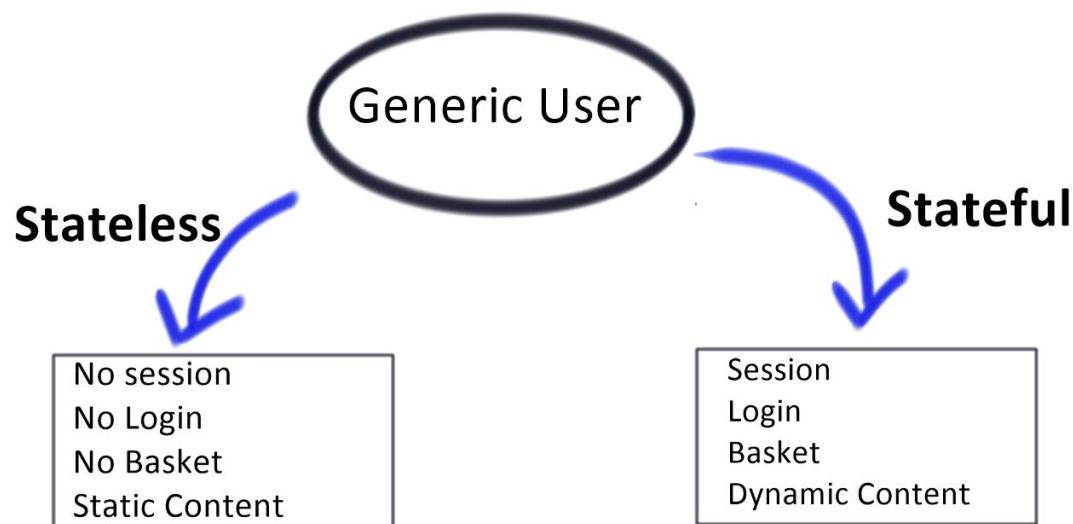
Kubernetes: Production Workload Orchestration



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

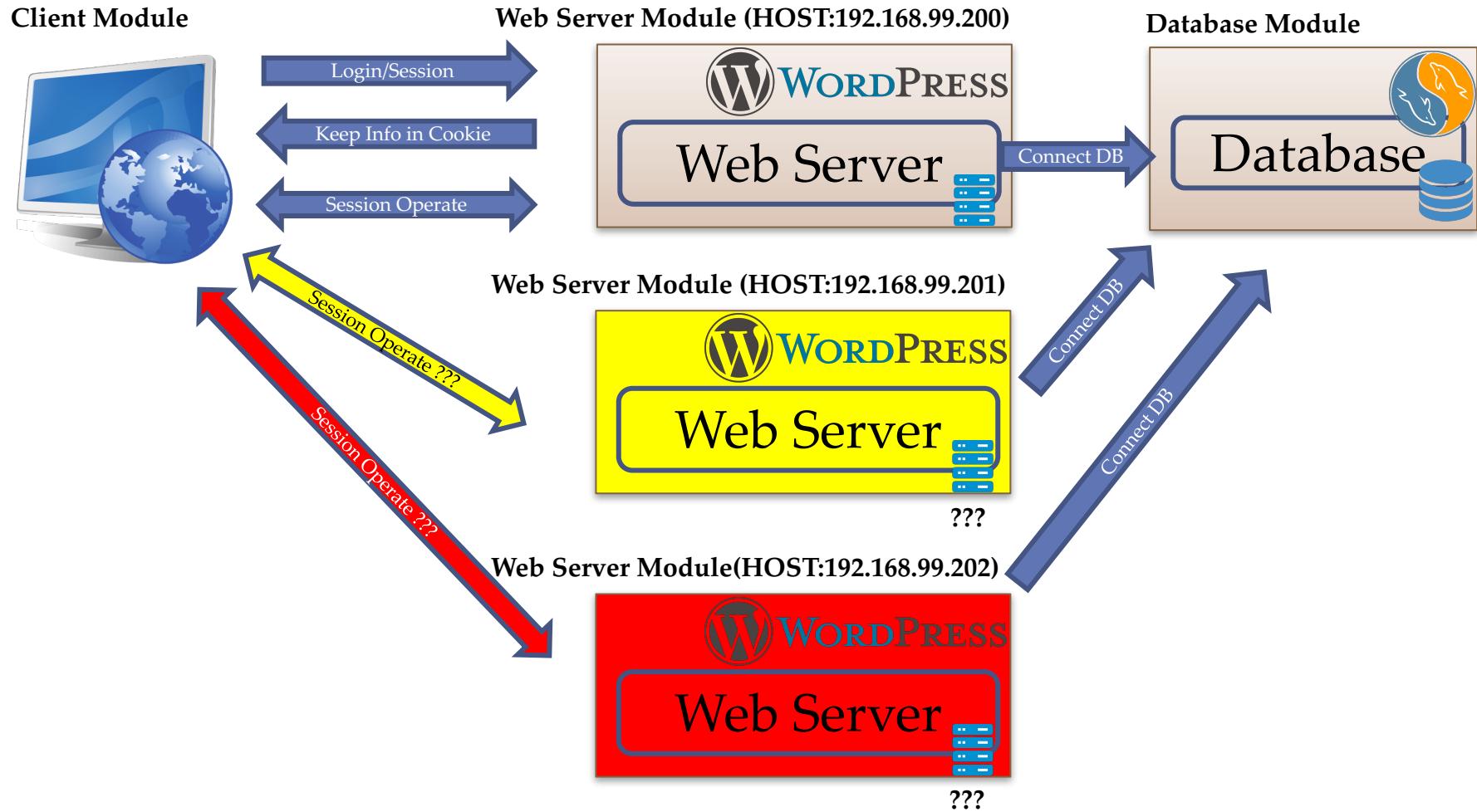


Stateful Application Deployment

- 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



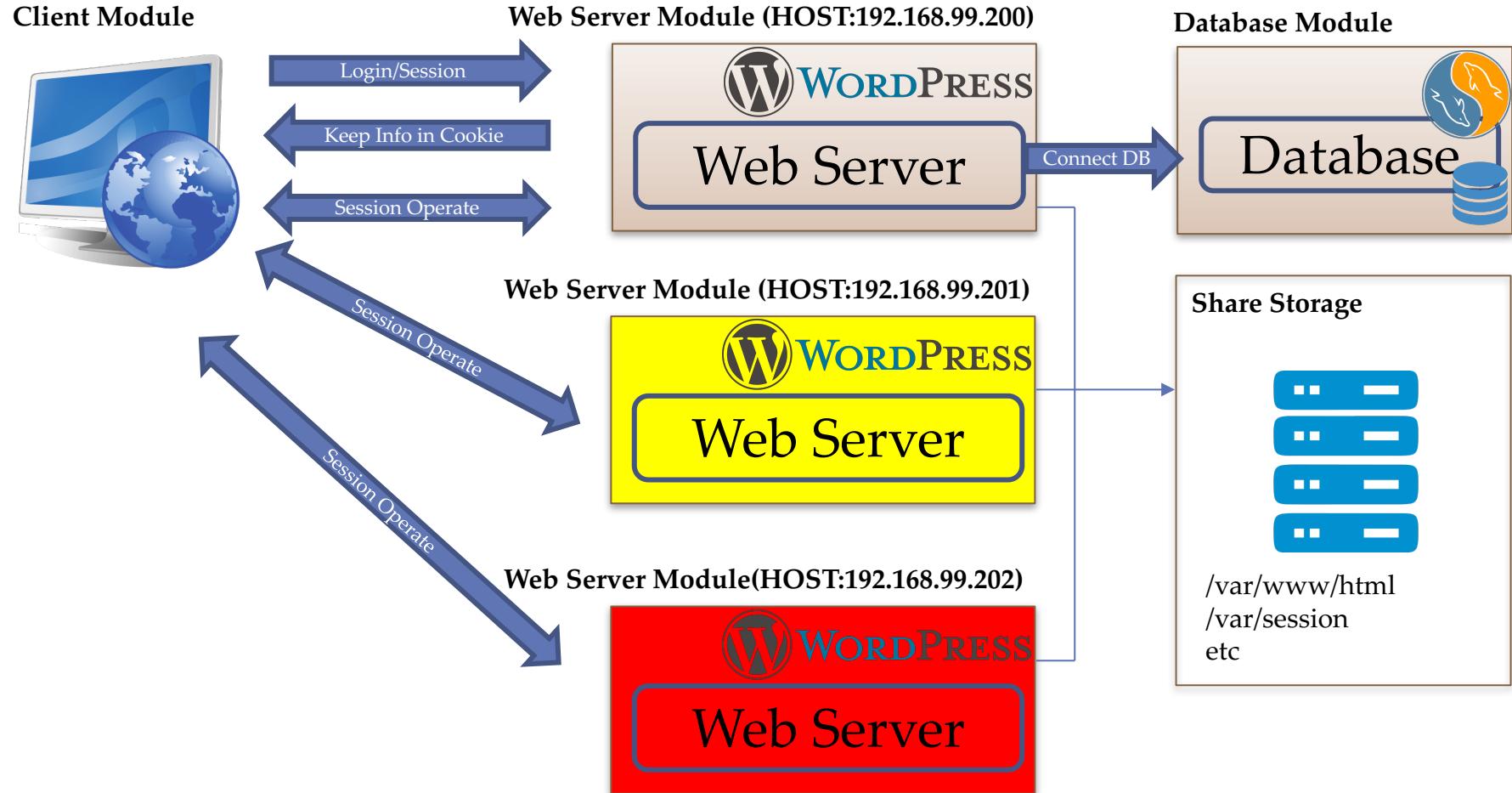
Stateful Application Deployment



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

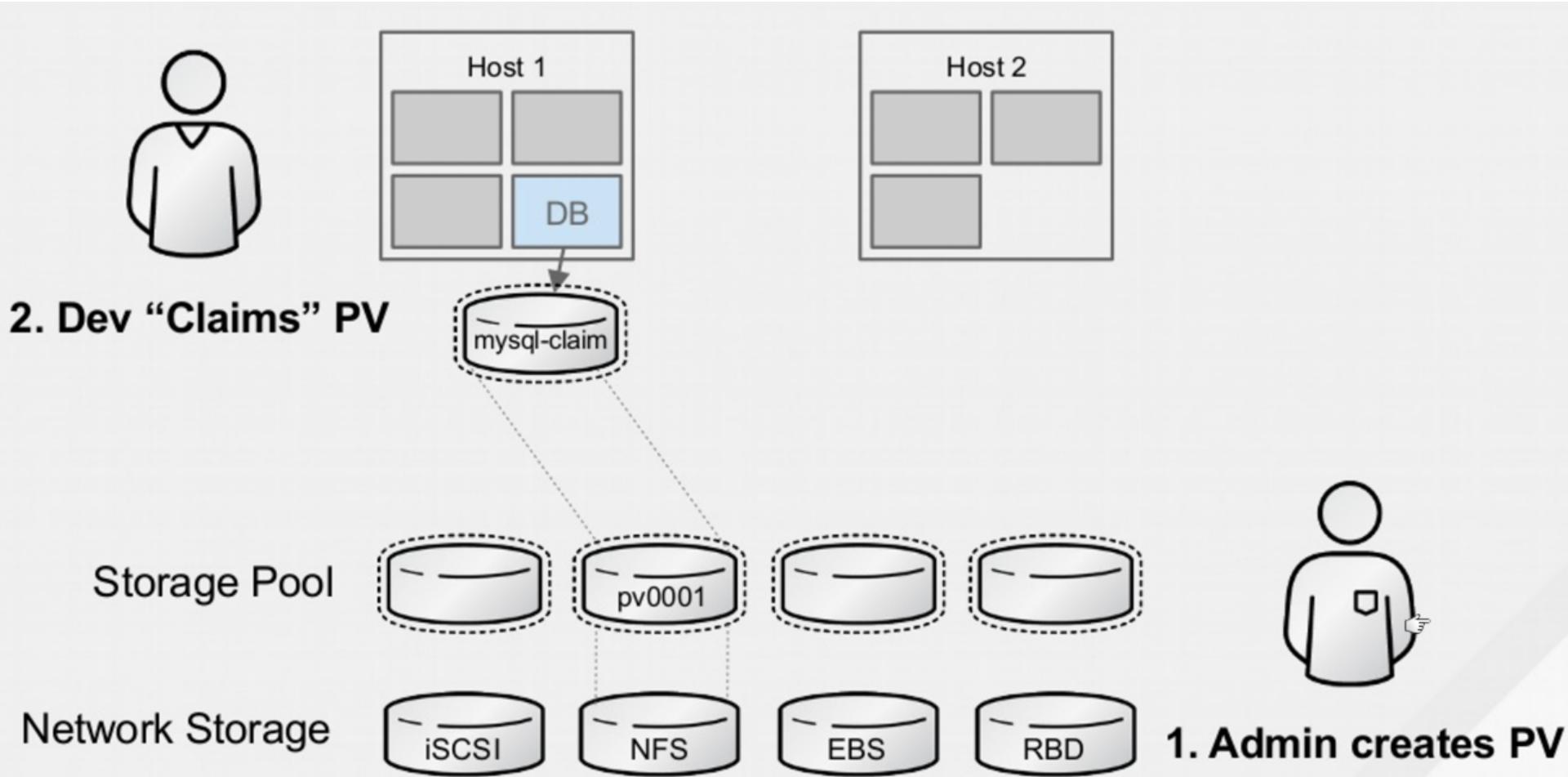
Stateful Application Deployment



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

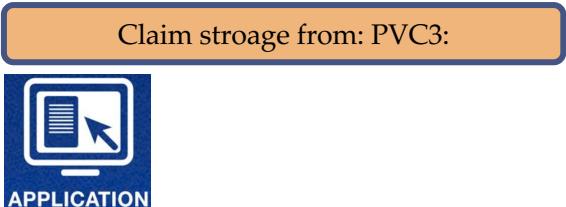
PV Pool



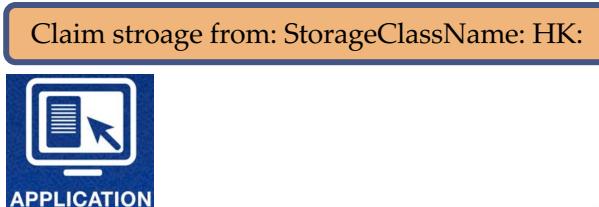
PVC Request



Pods for Application



Pods for Application



Static Provision

Provision

Phase: Available/Fail

Match PV/PVC

Binding

Phase: Bound

Pod use

Using

Dynamic Provision

Phase: Release

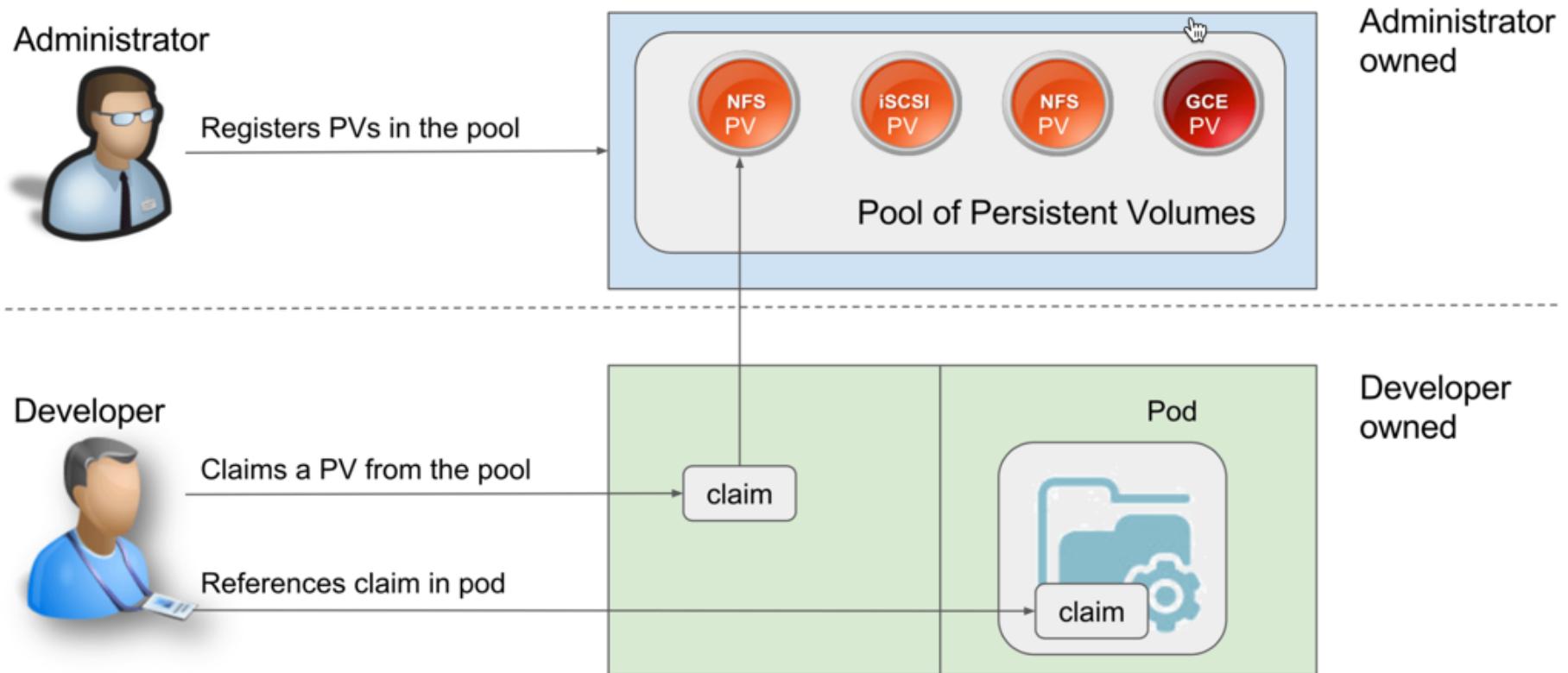
Reclaim

- Retaining (Keep Data)
- Recycle (rm -rf /path)
- Delete (Default for Dynamic Provision)



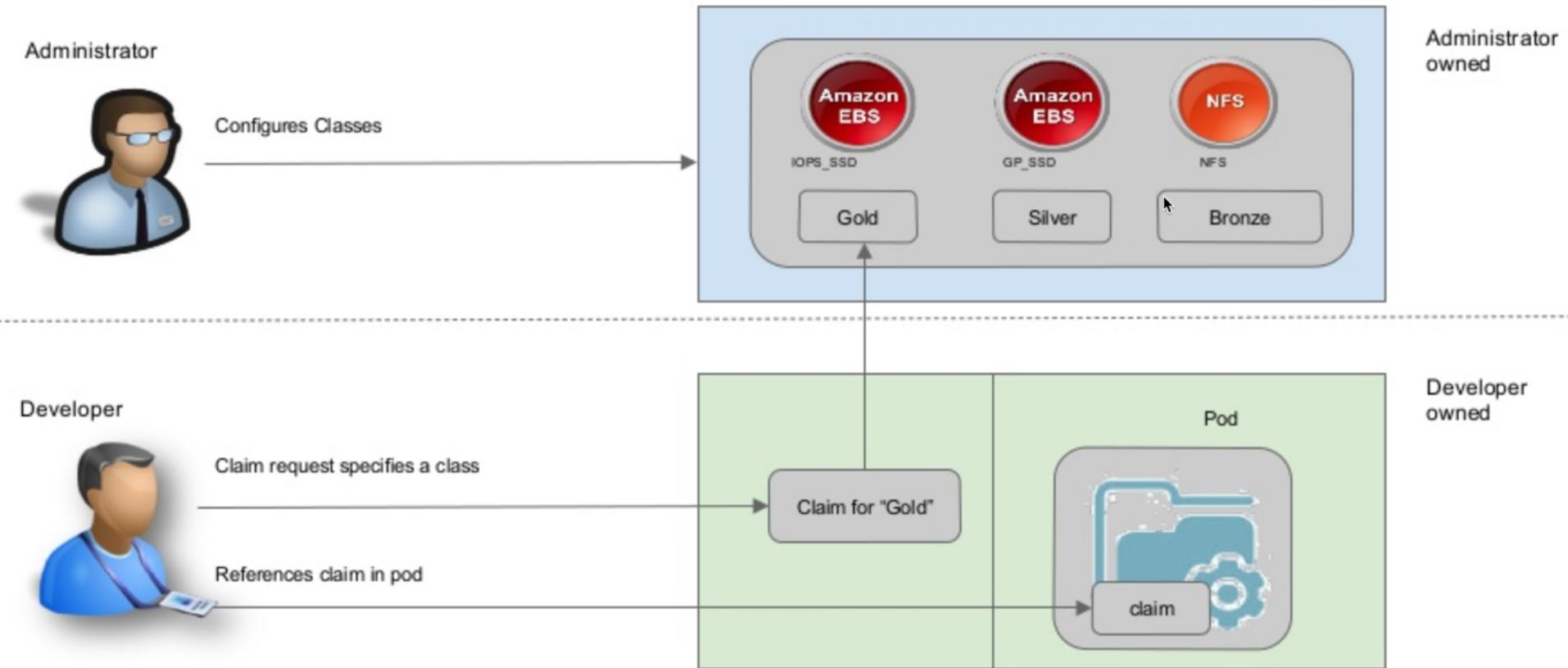
Persistent Volume

- 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

- 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

- 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   spec:
12     replicas: 3
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           - containerPort: 5000
34             protocol: TCP
35         volumeMounts:
36           - name: nfs-share-pvc
37             mountPath: "/usr/src/app"
38       volumes:
39       - name: nfs-share-pvc
40         persistentVolumeClaim:
41           claimName: nfs-share-pvc
```

Reference for mount disk volume from “PersistentVolumeClaim”



Persistent Volume

- Create PV

```
docker@kubernetes-ms:~$ kubectl create -f nfs_pv.yml
persistentvolume "nfs-share-pv" created
docker@kubernetes-ms:~$ kubectl create -f nfs_pvc.yml
persistentvolumeclaim "nfs-share-pvc" created
docker@kubernetes-ms:~$ kubectl get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS   CLAIM           STORAGECLASS   REASON
nfs-share-pv  1Gi        RWX          Retain         Bound    default/nfs-share-pvc
```

The screenshot shows the Kubernetes dashboard interface. The left sidebar has a navigation tree: Cluster > Persistent Volumes > nfs-share-pv. The 'Persistent Volumes' item is currently selected. The main content area has two tabs: 'Details' and 'Persistent volume source'. The 'Details' tab shows the volume's name, labels, annotations, creation time, status, claim, reclaim policy, access modes, storage class, capacity, reason, and message. The 'Persistent volume source' tab shows the NFS source details, including the server IP (10.38.14.200), path (/var/nfsshare), and read-only status.

NAME	CAPACITY	ACCESSMODES	RECLAIMPOLICY	STATUS	CLAIM	STORAGECLASS	REASON
nfs-share-pv	1Gi	RWX	Retain	Bound	default/nfs-share-pvc		

Details

Name: nfs-share-pv
Labels: environment: development, module: PV, name: nfs-share-pv, owner: Paparn_L, version: 1.0
Annotations: pv.kubernetes.io/bound-by-controller: yes
Creation time: 2017-10-03T15:08
Status: Bound
Claim: default/nfs-share-pvc
Reclaim policy: Retain
Access modes: ReadWriteMany
Storage class: -
Capacity: 1Gi
Reason: -
Message: -

Persistent volume source

NFS
Server: 10.38.14.200
Path: /var/nfsshare
Read only: -



Persistent Volume

- Create PVC

```
[docker@kubernetes-ms:~$ kubectl create -f nfs_pvc.yml
persistentvolumeclaim "nfs-share-pvc" created
```

The screenshot shows the Kubernetes UI for managing Persistent Volume Claims. The left sidebar lists various workloads and storage classes. The main area shows the details for a Persistent Volume Claim named 'nfs-share-pvc'. The 'Annotations' section is currently selected, showing two annotations: 'pv.kubernetes.io/bind-completed: yes' and 'pv.kubernetes.io/bound-by-controller: yes'. Other visible details include the Name, Namespace, Labels, Creation time, Status, Volume, Capacity, Access modes, and Storage class.

Details	
Name:	nfs-share-pvc
Namespace:	default
Labels:	environment: development, module: PVC, name: nfs-share-pvc, owner: Paparn_L, version: 1.0
Annotations:	pv.kubernetes.io/bind-completed: yes, pv.kubernetes.io/bound-by-controller: yes
Creation time:	2017-10-03T15:08
Status:	Bound
Volume:	nfs-share-pv
Capacity:	{"storage":"1Gi"}
Access modes:	ReadWriteMany
Storage class:	-



Persistent Volume

- Create Deployment

```
[docker@kubernetes-ms:~$ kubectl create -f nfs_webtest_deploy.yml
deployment "webtest" created
```

The screenshot shows the Kubernetes Web UI interface. On the left, there is a sidebar with navigation links: Storage Classes, Namespace (set to default), Overview, Workloads (selected), Daemon Sets, and Deployments. The main area is titled 'Deployments' and lists a single deployment named 'webtest'. The table columns are Name, Labels, Pods, Age, and Images. The 'webtest' row has the following details:

Name	Labels	Pods	Age	Images
webtest	environment: development module: WebServer name: web owner: Paparn_L version: 1.0	3 / 3	-	labdocker/cluster:webservicel

Persistent Volume

- Check source code via NFS and Test Web Page

```
docker@kubernetes-ms:~$ more /mnt/nfs_share/mainlite.py
from flask import Flask
import os
import time
app = Flask(__name__)

@app.route('/')
def hello():
    return '<H1> 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)
docker@kubernetes-ms:~$ 
```

A screenshot of a web browser window. The address bar shows the URL `10.38.14.201:32500`. The browser tabs include "Persister", "StatefulS", "Overview", "Services", "10.38.14", "Certificates", and "0". The main content area displays the text "Welcome Page from Container Python Lab" and "Checkpoint Date/Time: Tue Oct 3 15:15:25 2017". Below the browser window, a Mac OS X dock shows various application icons like Apps, NMac Ked - Mac OS..., Medium, Jenkins, vagrant, Mesos, Vue, docker, NGINX, Taiwan, Kubernetes, PWA_Progressive_W..., and MY.



Persistent Volume

- Test Edit source code via NFS

```
[docker@kubernetes-ms:~$ more /mnt/nfs_share/mainlite.py
from flask import Flask
import os
import time
app = Flask(__name__)

@app.route('/')
def hello():
    return '<H1> Welcome Page Edit from NFS Share ^^ 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)
docker@kubernetes-ms:~$ ]
```

Welcome Page Edit from NFS Share ^^ from Container Python Lab

Checkpoint Date/Time: Tue Oct 3 15:17:01 2017



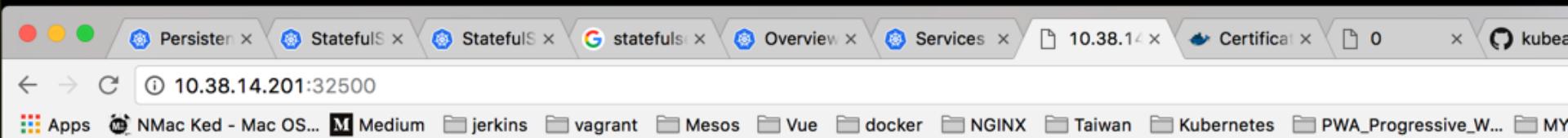
WorkShop 2.7: Persistency Storage

- Check source code via NFS and Test Web Page

```
docker@kubernetes-ms:~$ more /mnt/nfs_share/mainlite.py
from flask import Flask
import os
import time
app = Flask(__name__)

@app.route('/')
def hello():
    return '<H1> 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)
docker@kubernetes-ms:~$ 
```



Welcome Page from Container Python Lab

Checkpoint Date/Time: Tue Oct 3 15:15:25 2017



StatefulSet



Kubernetes: Production Workload Orchestration

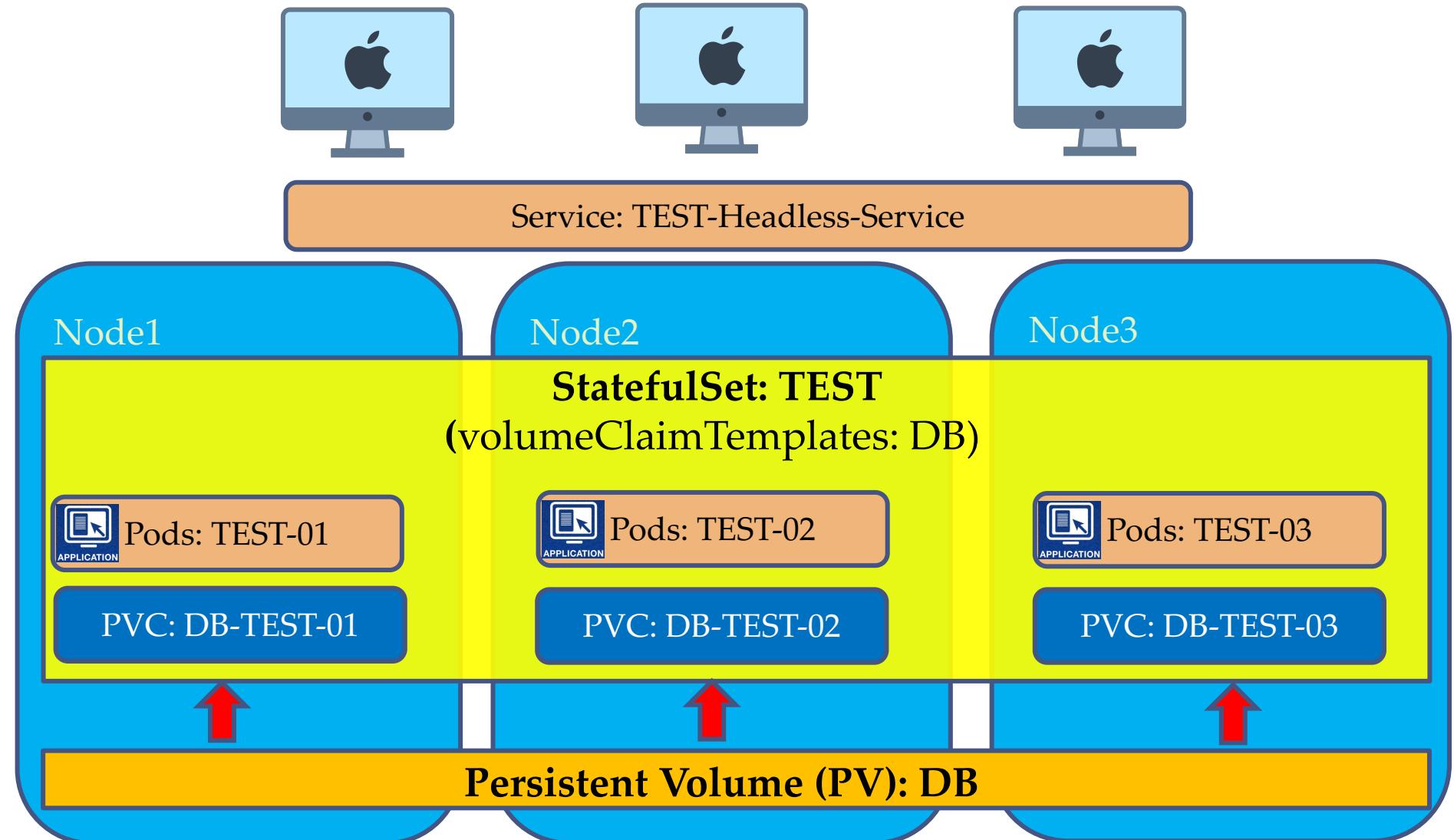


kubernetes
by Google

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



Kubernetes: Production Workload Orchestration



kubernetes
by Google

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
```



StatefulSet

[① Open](#) apeschel opened this issue on Aug 26 · 2 comments



apeschel commented on Aug 26 • edited

Contributor +

There seems to be a recurring bad practice among the charts in this repository: using a Deployment to manage pods using Persistent Volume Claims, rather than the proper StatefulSet.

To demonstrate just how pervasive the problem is, one can compare the list of charts using a StatefulSet vs a Deployment.

The list of stateful charts using a StatefulSet:

```
$ git grep -li 'kind: *StatefulSet' |  
  awk -F '/' '{print $1}'  
cockroachdb  
concourse  
consul  
ipfs  
memcached  
minio  
mongodb-replicaset  
rethinkdb
```

versus the stateful charts using a Deployment:

```
$ git grep -l -i 'kind: *Deployment' |  
  xargs grep -i PersistentVolumeClaim |  
  awk -F '/' '{print $1}' |  
  sort -u  
artifactory  
chronograf  
dokewiki  
drupal  
factorio  
ghost  
gitlab-ce  
gitlab-ee  
grafana  
influxdb  
jasperreports  
jenkins  
joomla  
kapacitor  
magento  
mariadb  
mediawiki  
minecraft  
minio  
mongodb  
moodle  
mysql  
odoo  
opencart  
openvpn  
orangehrm  
osclass  
owncloud  
percona  
phabricator  
phpbb  
postgresql  
prestashop  
prometheus  
rabbitmq  
redis  
redmine  
rocketchat  
sentry  
testlink  
traefik  
wordpress
```



StatefulSet

Kelsey Hightower  @kelseyhightower

Kubernetes has made huge improvements in the ability to run stateful workloads including databases and message queues, but I still prefer not to run them on Kubernetes.

06:04 - 13 ก.พ. 2561

295 รีทวีต 671 ชื่นชอบ

Replying to [@chanwit](#)

Kelsey Hightower  @kelseyhightower · 13 ก.พ.
Kubernetes can only meet stateful workloads half way and I lack the expertise to manage a production configuration of Kafka, RabbitMQ, or Postgres on static infrastructure, let alone a Kubernetes cluster.

2 รีทวีต 18 ชื่นชอบ

Kelsey Hightower  @kelseyhightower · 13 ก.พ.
Kubernetes makes it easier to deploy stateful services not manage them. Stateful services must meet Kubernetes half way and manage their own cluster membership, failover, and replication. CockroachDB and Consul are two great examples, but far from perfect.

5 รีทวีต 20 ชื่นชอบ

Kelsey Hightower  @kelseyhightower · 13 ก.พ.
Even when stateful services do the right things managing state is still hard. Mixing stateful and stateless applications on the same cluster elevates the complexity of the entire cluster. Cluster security and upgrades become much harder.

3 รีทวีต 12 ชื่นชอบ

Kelsey Hightower  @kelseyhightower · 13 ก.พ.
The solution I've been using: isolate stateful services to a dedicated set of machines or leverage a managed service. If I really need to run stateful services on a shared Kubernetes cluster, I isolate them to a dedicated node pool and disable dynamic scheduling.

8 รีทวีต 41 ชื่นชอบ

Chanwit Kaewkasi @chanwit · 36m
Don't get me wrong. I read the thread > 3 times before posting this. My perspective to complexity is:
"the already working things + Kubernetes"
I just have no reason to learn Kubernetes, as I already have some stateful workloads running outside orchestrator, just like you said.

1 รีทวีต 1 ใจถูกใจ

Kelsey Hightower  @kelseyhightower

Following

<https://sites.google.com/site/chanwit/blogs/kelseyhightower-not-use-k8s-for-stateful-workloads>

Kubernetes: Production Workload Orchestration



kubernetes
by Google

Recap Day 2

- Fundamental of Kubernetes
 - ConfigMap Secret
 - Job and CronJob
 - Log and Monitoring
- Ingress Networking
- Kubernetes in real world
 - Cluster Setup for Bare Metal
 - Orchestrator Assignment
 - nodeSelector
 - Interlude
 - Affinity
 - Taints/Tolerations
- Stateful application deployment
 - Consideration and Awareness
 - Persistent Volumes
 - StatefulSet



Question & Answer Section



By: Praparn L (eva10409@gmail.com)



kubernetes
by Google