Kubernetes 101: Production-Workload Orchestration

By: Praparn L (eva10409@gmail.com)

kubernetes by Google

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 Meterial
- 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

Explore Help labdocker Sign up Log In

Repositories (7)

All

labdocker/alpineweb public	0 STARS	31 PULLS	DETAILS
labdocker/nginx public	0 STARS	16 PULLS	DETAILS
labdocker/alpine public	0 STARS	7 PULLS	DETAILS

Lab Resource

- Software in lab

Lab Resource

- Download on Google Drive
  - https://drive.google.com/open?id=0ByjcBsedtQQiMzVCdVp5emxpeDg
- Download on GitHub
  - git@github.com:praparn/kubernetesLab.git

Senior Infrastructure / System Engineer

Compute, Network & Storage System (CNS)

Infrastructure Enterprise Equipment

Virtual Machine for Open System in Prd

Core Application on IBM AS/400 System

Instructor / Consult for Docker's Solution

Kubernetes: Production Workload Orchestration

Landscape of the world now

Landscape of the world now

Workshop 1.1: Install Kubenetes

Container Principle

Container Principle

Existing Technology

- <u>Production Environment (Best design)</u>
- Day 1: Application 1: Implement

- Apache 2.20 Web Server
- PHP 5.5 Engine
- Larevel 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

- <u>Production Environment</u>
- Day 1: Application 1: Implement

- Apache 2.20 Web Server
- PHP 5.5 Engine
- Larevel 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[a]	No	No	No	No	No	No	Yes	No	No
Docker	Linux[7]	Apache License 2.0	2013	Yes	Yes	Not directly	Not directly	Yes	Yes	Yes	Yes	No	No
Linux-VServer (security context)	Linux	GNU GPLv2	2001	Yes	Yes	Yes	Yes[b]	Yes	Yes	Partial[c]	?	No	Partial[d]
lmctfy	Linux	Apache License 2.0	2013	Yes	Yes	Yes	Yes[b]	Yes	Yes	Partial[c]	?	No	Partial[d]
LXC	Linux	GNU GPLv2	2008	Yes[g]	Yes	Partial[e]	Partial[f]	Yes	Yes	Yes	Yes	No	Yes[g]
LXD	Linux	Apache License 2.0	2015	Yes	Yes	Partial(see LXC)	Partial(see LXC)	Yes	Yes	Yes	Yes	Partial[g]	Yes
OpenVZ	Linux	GNU GPLv2	2005	Yes	No	Yes	Yes[h]	Yes	Yes	Yes[i]	Partial[j]	Yes	Yes[k]
Virtuozzo	Linux, Windows	Proprietary	2000[14]	Yes	Yes	Yes	Yes[l]	Yes	Yes	Yes[i]	Partial[m]	Yes	Yes
Solaris Containers (Zones)	illumos (OpenSolaris), Solaris	CDDL, Proprietary	2004	Yes	Yes (ZFS)	Yes	Partial[n]	Yes	Yes	Yes[o][17][18]	Partial[p]	Partial[q][r]	Yes[s]
FreeBSD jail	FreeBSD	BSD License	2000[20]	Yes	Yes (ZFS)	Yes[t]	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[u]	No	Yes[25]	?
HP-UX Containers (SRP)[26]	HPUX	Proprietary	2007	Yes	No	Partial[v]	Yes	Yes	Yes	Yes	?	Yes	?
iCore Virtual Accounts	Windows XP	Proprietary/Freeware	2008	Yes	No	Yes	No	No	No	No	?	No	?
Sandboxie	Windows	Proprietary/Shareware	2004	Yes	Yes	Partial	No	No	No	Partial	No	No	Yes
Spoon	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

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

Container Principle

Kubernetes: Production Workload Orchestration

kubernetes by Google

Container Principle

rkt

A security-minded, standards-based container engine

Overview | Documentation | GitHub Project

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 widel

THE LATEST ON RKT

- CoreOS's rkt anc to CNCF
- What Kubernete container engin

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

```
praparns-MacBook-Pro:localkube praparn$ kubectl get svc
NAME         CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes   10.0.0.1     <none>        443/TCP        13m
webtest      10.0.0.83    <nodes>       80:30891/TCP   8s
praparns-MacBook-Pro:localkube praparn$ kubectl describe svc webtest
Name:                     webtest
Namespace:                default
Labels:                   run=webtest
Annotations:              <none>
Selector:                 run=webtest
Type:                     NodePort
IP:                       10.0.0.83
Port:                     <unset> 80/TCP
NodePort:                 <unset> 30891/TCP
Endpoints:                172.17.0.4:80
Session Affinity:         None
Events:                   <none>
praparns-MacBook-Pro:localkube praparn$
```

Introduction to Kubernetes

- What is "Kubernetes"
  - "Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications"
  - Greek word mean "Pilot"
  - Short word "K8s" (K+8 character +s)
  - Design and Reference from Google's Borg system
  - Donate to "**CLOUD NATIVE** COMPUTING FOUNDATION"
- Ref: Current release version
  - 1.6.0 (Exist Stable Version)
  - 1.7.0 (Stable Version: 1/7/2017)
  - 1.8.0 (Stable Version: 27/9/2017)

Ref:https://en.wikipedia.org/wiki/Kubernetes, https://en.wikipedia.org/wiki/Kubernetes

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/nodesminikube
    uid: f03de16d-58bb-11e7-aae1-080027559511
  spec:
    externalID: minikube
```

```
    nodeInfo:
      architecture: amd64
      bootID: cdf78a28-072d-4707-8c7d-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

CLOUD NATIVE COMPUTING FOUNDATION

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

kubernetes 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)

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

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

Introduction to Kubernetes

Kubernetes: Production Workload Orchestration

kubernetes by Google

Introduction to Kubernetes

Kubernetes: Production Workload Orchestration

kubernetes by Google

Introduction to Kubernetes

- 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 LimitsMemory 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, Azue 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

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

System Architecture

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

System Architecture

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

System Architecture

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

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

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

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/

Fundamental of Kubernetes

Kubernetes: Production Workload Orchestration

kubernetes by Google

Pods, Container and Services

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 kubenetes 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>

```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
```

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:

Pods, Container and Services

- Imperative object configuration

kubectl create –f <Filename>

- Create Pods (YAML)

```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)

```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

Code Beautify

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

YAML Converter

Save & Share

YAML Input sample

```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
  nodeSelector:
    kubernetes.io/hostname: kubernetes-1
```

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

```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>

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://04b9cdfdd6451a4e78c873f704fb4d35de7bba082343d0773c7ea0ebbbd3f03a
    Image:              labdocker/cluster:webservicelite
    Image ID:           docker://sha256:837c8f41c918ede06f05f9b554b6120fdb654cc2a8eb7eec74bc383b09865f2b
    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                                       Normal          Scheduled       Successfully assigned webtest to minikube
  14m           14m             1       kubelet, minikube       spec.containers{webtest}        Normal          Pulled          Container image "labdocker/cluster:webservicelite" alread
y present on machine
  14m           14m             1       kubelet, minikube       spec.containers{webtest}        Normal          Created         Created container with id 04b9cdfdd6451a4e78c873f704fb4d3
5de7bba082343d0773c7ea0ebbbd3f03a
  14m           14m             1       kubelet, minikube       spec.containers{webtest}        Normal          Started         Started container with id 04b9cdfdd6451a4e78c873f704fb4d3
5de7bba082343d0773c7ea0ebbbd3f03a
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

kubernetes by Google

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),
        DESCRIPE 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, DESCRIPE) VALUES (%s,%s,%s)", (req_json['uid'], req_json['user'], req_json['descripe']))
    #curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "descripe":"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
456afbce89a4        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", "descripe":"Slave"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Somchai Sunsukwan", "descripe":"Security Guard"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "3", "user":"Sanyachan Panrudee", "descripe":"House Keeping"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "4", "user":"Sakkan Yanyicharoen", "descripe":"Messenger"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "5", "user":"Chatchai Moungang", "descripe":"Programmer"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "6", "user":"Anusit Kannaphat", "descripe":"DevOps Manager"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "7", "user":"Meelarp Maisanuk", "descripe":"System Engineer"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "8", "user":"Pansa Bunsong", "descripe":"Secuirty Guard"}' http://$Server_IP:$Server_Port/users/insertuser
curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "9", "user":"Wiphanee Wongsaisawan", "descripe":"Administrator"}' http://$Server_IP:$Server_Port/users/insertus

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$
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", "descripe":"Slave"}' http://$Server_IP:$Serv
er_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$
praparns-MacBook-Pro:~ praparn$ curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Somchai Sunsukwan", "descripe":"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:~ praparn$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlappraparns-MacBook-Pro:~ praparn$
praparns-MacBook-Pro:~ praparn$ curl http://$Server_IP:$Server_Port/users/1
Praparn Luangphoonlap(Database Cache)praparns-MacBook-Pro:~ praparn$
```

  - Delete database

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

Pods, Container and Service

kubernetes by Google

Pods, Container and Service

Kubernetes: Production Workload Orchestration

kubernetes by Google

Pods, Container and Service

kubernetes by Google

Pods, Container and Service

- Pods: maindb(YML)

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: maindb
  labels:
    name: "maindb"
    owner: "Praparn_L"
    version: "1.0"
    module: "maindb"
    environment: "development"
spec:
  containers:
    - name: maindb
      image: labdocker/mysql:latest
      ports:
        - containerPort: 3306
          protocol: TCP
      env:
        -
          name: "MYSQL_ROOT_PASSWORD"
          value: "password"
```

Service: maindb(YML)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: maindb
  labels:
    name: "maindb"
    owner: "Praparn_L"
    version: "1.0"
    module: "maindb"
    environment: "development"
spec:
  ports:
  - port: 3306
    targetPort: 3306
  selector:
    name: "maindb"
    owner: "Praparn_L"
    version: "1.0"
    module: "maindb"
    environment: "development"
```

Pods, Container and Service

- Pods: web(YML)

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: web
  labels:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "web"
    environment: "development"
spec:
  containers:
    - name: cachedb
      image: labdocker/redis:latest
      ports:
        - containerPort: 6379
          protocol: TCP
    - name: webservice
      image: labdocker/cluster:webservice
      env:
        - name: "REDIS_HOST"
          value: "localhost"
      ports:
        - containerPort: 5000
          protocol: TCP
    - name: webcache
      image: labdocker/cluster:webcache_kubernetes
      ports:
        - containerPort: 80
          protocol: TCP
```

Service: web (YML)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web
  labels:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "Web"
    environment: "development"
spec:
  selector:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "web"
    environment: "development"
  type: NodePort
  ports:
   - port: 5000
     name: webservice
     targetPort: 5000
     protocol: TCP
     nodePort: 32500
   - port: 80
     name: webcache
     targetPort: 80
     protocol: TCP
     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$
```

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", "descripe":"
Slave"}' http://$Server_IP:$Server_Port/users/insertuser
HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Sun, 02 Jul 2017 13:18:45 GMT
Content-Type: application/json
Content-Length: 41
Connection: keep-alive

########### Record was added ###########
praparns-MacBook-Pro:multicontainer praparn$
```

Pods, Container and Service

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

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

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

```
praparns-MacBook-Pro:multicontainer praparn$ curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"Praparn Luangphoonlap", "descripe":"
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 is can create / destroy / restart every time (manual/automatic)
    - Mean "Pods" always change their nodes/ip/location everytime
    - Service don't care how Pods is <u>being</u>, But they still can <u>map the Pods</u>

  - Service is abstract of Pods (<u>1 – N Pods</u>)
    - 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”

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: web
  labels:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "web"
    environment: "development"
spec:
  containers:
    - name: cachedb
      image: labdocker/redis:latest
      ports:
        - containerPort: 6379
          protocol: TCP
    - name: webservice
      image: labdocker/cluster:webservice
      env:
        - name: "REDIS_HOST"
          value: "localhost"
      ports:
        - containerPort: 5000
          protocol: TCP
    - name: webcache
      image: labdocker/cluster:webcache_kubernetes
      ports:
        - containerPort: 80
          protocol: TCP
```

Pods “web2”

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: web2
  labels:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "web"
    environment: "development"
spec:
  containers:
    - name: cachedb
      image: labdocker/redis:latest
      ports:
        - containerPort: 6379
          protocol: TCP
    - name: webservice
      image: labdocker/cluster:webservice2
      env:
        - name: "REDIS_HOST"
          value: "localhost"
      ports:
        - containerPort: 5000
          protocol: TCP
    - name: webcache
      image: labdocker/cluster:webcache_kubernetes
      ports:
        - containerPort: 80
          protocol: TCP
```

Service “web”

```yaml
apiVersion: v1
kind: Service
metadata:
  name: web
  labels:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "Web"
    environment: "development"
spec:
  selector:
    name: "web"
    owner: "Praparn_L"
    version: "1.0"
    module: "web"
    environment: "development"
  type: NodePort
  ports:
   - port: 5000
     name: webservice
     targetPort: 5000
     protocol: TCP
     nodePort: 32500
   - port: 80
     name: webcache
     targetPort: 80
     protocol: TCP
     nodePort: 30500
```

Pods, Container and Services

- Services
  - Existing Pods "web"

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

```
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
```

```
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 addess 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”

```yaml
  type: NodePort
  ports:
   - port: 5000
     name: webservice
     targetPort: 5000
     protocol: TCP
     nodePort: 32500
   - port: 80
     name: webcache
     targetPort: 80
     protocol: TCP
     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:
    - External LoadBalance (Usually from cloud provider): service will allow loadbalance send traffic through Pods directly

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
      nodePort: 30061
  clusterIP: 10.0.171.239
  loadBalancerIP: 78.11.24.19
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 146.148.47.155
```

Pods, Container and Services

- Services
  - Publish Service Type:
    - External Ips: Specific external IP need to set-up from network equipment (L3 Switch/Firewall/F5 etc) to forward traffic to kubernetes's nodes (Not IP Address of node)

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
  externalIPs:
    - 80.11.12.10
```

Workshop 1.2: Pods & Service

- Part 2: Deploy web pods with multi container

kubernetes by Google

Pods, Container and Services

- Other Command

Secure https://kubernetes.io/docs/user-guide/kubectl/v1.7/#-strong-getting-started-strong-

Apps | NMac Ked - Mac O... | Medium | jerkins | vagrant | Mesos | Vue | docker | NGINX | Taiwan | Kubernetes | PWA_Progressive_... | MYSQL_Cluster | Angular | OpenStack | มาลอง ทำเว็บไซต์แบ... | Other Bookmarks

GETTING STARTED
run
expose

APP MANAGEMENT
annotate
autoscale
convert
create
delete
edit
get
label
patch
replace
rolling-update
rollout
scale
set

DECLARATIVE APP MANAGEMENT

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...]
```

example

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 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)

Daemon Set and RC

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
  version: "1.0"
  module: WebServer
  environment: development
```

Replication Controller (RC)

- Example
  - Pods “webtest”

```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
```

RC “webtest”

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  replicas: 3
  template:
    metadata:
      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
```

SVC “webtest”

```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
     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$
```

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$
```

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:webservicelite
    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-thhxf   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-thhxf
  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-jbqdq
  2m            2m              1       replication-controller                  Normal          SuccessfulDelete        Deleted pod: webtest-lgprd
  2m            2m              1       replication-controller                  Normal          SuccessfulDelete        Deleted pod: webtest-thhxf
  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-jbqdq
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$
```

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-thhxf
  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-jbqdq
  2m            2m            1       replication-controller                  Normal      SuccessfulDelete    Deleted pod: webtest-lgprd
  2m            2m            1       replication-controller                  Normal      SuccessfulDelete    Deleted pod: webtest-thhxf
  2m            2m            1       replication-controller                  Normal      SuccessfulDelete    Deleted pod: webtest-2hgdq
  2m            2m            1       replication-                                                                sqsjk
  2m            2m            1       replication-                                                                jbqdq
praparns-MacBook-Pro:ReplicationController praparn$
```

192.168.99.100:32500

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

Welcome Page from Container Python Lab

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

Deployment/RS and Update

Deployment/RS and Update

Deployment/RS and Update

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

Deployment/RS and Update

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

```
selector:
  name: web
  owner: Praparn_L
  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:webservicelite_v1

  - labdocker/cluster:webservicelite_v1.51rc

  - labdocker/cluster:webservicelite_v1.8ga

Deployment/RS and Update

- Example
  - Deployment “webtest”

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  replicas: 3
  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
```

SVC “webtest”

```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
     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

```yaml
spec:
  replicas: 3
  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
```

  - Step for produce update (rollout) with <u>Each Change</u>
    - 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$
```

Workshop 1.4: Deployment/RS

- Deploy replication controller

```
Conditions:
  Type          Status  Reason
  ----          ------  ------
  Available     True    MinimumReplicasAvailable
OldReplicaSets: <none>
NewReplicaSet:  webtest-3067359944 (3/3 replicas created)
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath   Type            Reason                  Message
  ---------     --------        -----   ----                    -------------   --------        ------                  -------
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled up replica set webtest-4
261491039 to 3
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled up replica set webtest-3
067359944 to 1
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled down replica set webtest
-4261491039 to 2
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled up replica set webtest-3
067359944 to 2
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled down replica set webtest
-4261491039 to 1
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled up replica set webtest-3
067359944 to 3
  1m            1m              1       deployment-controller                   Normal          ScalingReplicaSet       Scaled down replica set webtest
-4261491039 to 0
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 18:00:00 2017
praparns-MacBook-Pro:WorkShop_1.4_Deployment praparn$ kube
```

Volume

Volume

- By default all container in same Pods will share storage (emptyDir)
  - This storage will keep all read/write for all container in Pods
  - Persistent storage for container as long as Pods still avaliable
  - 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/distrube
- 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

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:

n.txt Workshop_2.2_Job_CronJob | instruction.txt WorkShop_1.7_Volume | cadvisor_deploy.yml

```yaml
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: cadvisor
        owner: Praparn_L
        version: "1.0"
        module: Monitor
        environment: development
    spec:
      containers:
        - name: cadvisor
          volumeMounts:
          - mountPath: /var/run
            name: volrun
          - mountPath: /sys
            name: volsys
          - mountPath: /var/lib/docker/
            name: voldocker
          image: labdocker/cadvisor:latest
          ports:
          - containerPort: 8080
            protocol: TCP
      volumes:
      - name: volrun
        hostPath:
          path: /var/run
      - name: volsys
        hostPath:
          path: /sys
      - name: voldocker
        hostPath:
          path: /var/lib/docker/
```

Volume

- hostPath:
- Container create file

Host check file

```
praparnlueangphoonlap — Terminal MAC Pro — kubectl exec -it cadviso
~ — Terminal MAC Pro — ssh ◂ minikube ssh
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

```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_v1
      ports:
        - containerPort: 5000
          protocol: TCP
      volumeMounts:
      - mountPath: /temp
        name: voldocker
  volumes:
    - name: voldocker
      hostPath:
        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

n.txt Workshop_2.2_Job_CronJob | instruction.txt WorkShop_1.7_Volume | cadvisor_deploy.yml

```yaml
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: cadvisor
        owner: Praparn_L
        version: "1.0"
        module: Monitor
        environment: development
    spec:
      containers:
        - name: cadvisor
          volumeMounts:
          - mountPath: /var/run
            name: volrun
          - mountPath: /sys
            name: volsys
          - mountPath: /var/lib/docker/
            name: voldocker
          image: labdocker/cadvisor:latest
          ports:
          - containerPort: 8080
            protocol: TCP
      volumes:
      - name: volrun
        hostPath:
          path: /var/run
      - name: volsys
        hostPath:
          path: /sys
      - name: voldocker
        hostPath:
          path: /var/lib/docker/
```

Volume

Host 1
DB
Host 2
2. Dev "Claims" PV
mysql-claim
Storage Pool
pv0001
Network Storage
iSCSI
NFS
EBS
RBD
1. Admin creates PV

Liveness and Readiness Probe

Liveness and Readiness Probe

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

Liveness and Readiness Probe

- Pods are you still alright ?
  - When Pods receive status from container. It will set Pods's status to inform kubelet for operate with Pods (Remain / Terminated / Restart)

  - Pods 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/

Liveness and Readiness Probe

- ExecAction:

```yaml
kind: Deployment
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
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
          readinessProbe:
            exec:
              command:
                - cat
                - /usr/src/app/main.py
            initialDelaySeconds: 15
            periodSeconds: 5
          livenessProbe:
            exec:
              command:
                - cat
                - /usr/src/app/main.py
            initialDelaySeconds: 15
            periodSeconds: 15
```

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

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/

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","na
me":"webtest-336910061","uid":"4992f987-9af0-11e7-bc1a-080027fb95be",...
Status:         Running
```

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

Liveness and Readiness Probe

- ExecAction:

```
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath                   Type            Reason                  Message
  ---------     --------        -----   ----                    -------------                   --------        ------                  -------
  7m            7m              1       default-scheduler                                       Normal          Scheduled               Successfull
y assigned webtest-336910061-26pdb to minikube
  7m            7m              1       kubelet, minikube                                       Normal          SuccessfulMountVolume   MountVolume
.SetUp succeeded for volume "default-token-pcw4h"
  2m            2m              8       kubelet, minikube       spec.containers{webtest}        Warning         Unhealthy               Readiness p
robe 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/

Liveness and Readiness Probe

- TCPSocketAction:

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
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:
          - name: webservice
            containerPort: 5000
            protocol: TCP
          readinessProbe:
            tcpSocket:
              port: 5000
            initialDelaySeconds: 15
            periodSeconds: 5
          livenessProbe:
            tcpSocket:
              port: 5000
            initialDelaySeconds: 15
            periodSeconds: 15
```

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

Liveness and Readiness Probe

- TCPSocketAction:

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

```
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath                   Type          Reason                  Message
  ---------     --------        -----   ----                    -------------                   --------      ------                  -------
  2m            2m              1       default-scheduler                                       Normal        Scheduled               Successfull
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 "labdocker/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/

Liveness and Readiness Probe

- TCPSocketAction:

instruction.txt | webtest_deploy_liveness_readiness_port.yml

```yaml
    spec:
      containers:
        - name: webtest
          image: labdocker/cluster:webservicelite_v1
          ports:
          - name: webservice
            containerPort: 5000
            protocol: TCP
          readinessProbe:
            tcpSocket:
              port: 3000
            initialDelaySeconds: 15
            periodSeconds: 5
          livenessProbe:
            tcpSocket:
              port: 3000
            initialDelaySeconds: 15
            periodSeconds: 15
```

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

Liveness and Readiness Probe

- TCPSocketAction:

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

Liveness and Readiness Probe

- HTTPGetAction:

Liveness and Readiness Probe

- HTTPGetAction:

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: web
        owner: Praparn_L
        version: "1.0"
        module: WebServer
        environment: development
```

```yaml
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: HTTP
          path: /
          port: webservice
          httpHeaders:
            - name: server
              value: "Werkzeug/0.12.2 Python/2.7.13"
        initialDelaySeconds: 15
        periodSeconds: 5
        timeoutSeconds: 1
        successThreshold: 1
        failureThreshold: 1
      livenessProbe:
        httpGet:
          # Optional "host: my-host" for set specific ip of Pods
          # Optional "scheme: HTTPS"
          #schema: HTTP
          path: /
          port: webservice
          httpHeaders:
            - name: server
              value: "Werkzeug/0.12.2 Python/2.7.13"
        initialDelaySeconds: 15
        periodSeconds: 5
        timeoutSeconds: 1
        successThreshold: 1
        failureThreshold: 1
```

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
-template-hash=1098172621,version=1.0
```

```
Events:
  FirstSeen     LastSeen        Count   From                    SubObjectPath                   Type            Reason                  Message
  ---------     --------        -----   ----                    -------------                   --------        ------                  -------
  9m            9m              1       default-scheduler                                       Normal          Scheduled               Successfull
y assigned webtest-1098172621-zvd22 to minikube
  9m            9m              1       kubelet, minikube                                       Normal          SuccessfulMountVolume   MountVolume
.SetUp succeeded for volume "default-token-pcw4h"
  9m            9m              1       kubelet, minikube       spec.containers{webtest}        Normal          Pulled                  Container i
mage "labdocker/cluster:webservicelite_v1" already present on machine
  9m            9m              1       kubelet, minikube       spec.containers{webtest}        Normal          Created                 Created con
tainer
  9m            9m              1       kubelet, minikube       spec.containers{webtest}        Normal          Started                 Started con
tainer
```

Liveness and Readiness Probe

- HTTPGetAction:

```yaml
          readinessProbe:
            httpGet:
              # Optional "host: my-host" for set specific ip of Pods
              # Optional "scheme: HTTPS"
              #schema: HTTPS
              path: /init
              port: webservice
              httpHeaders:
                - name: server
                  value: "Werkzeug/0.12.2 Python/2.7.13"
            initialDelaySeconds: 15
            periodSeconds: 5
            timeoutSeconds: 5
            successThreshold: 1
            failureThreshold: 3
          livenessProbe:
            httpGet:
              # Optional "host: my-host" for set specific ip of Pods
              # Optional "scheme: HTTPS"
              #schema: HTTPS
              path: /init
              port: webservice
              httpHeaders:
                - name: server
                  value: "Werkzeug/0.12.2 Python/2.7.13"
            initialDelaySeconds: 15
            periodSeconds: 5
            timeoutSeconds: 5
            successThreshold: 1
            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               Successfull
y assigned webtest-692878837-fxpw4 to minikube
  1m            1m              1       kubelet, minikube                                       Normal          SuccessfulMountVolume   MountVolume
.SetUp succeeded for volume "default-token-pcw4h"
  1m            13s             5       kubelet, minikube       spec.containers{webtest}        Normal          Pulled                  Container i
mage "labdocker/cluster:webservicelite_v1" already present on machine
  1m            13s             9       kubelet, minikube       spec.containers{webtest}        Warning         Unhealthy               Readiness p
robe failed: HTTP probe failed with statuscode: 404
  1m            13s             9       kubelet, minikube       spec.containers{webtest}        Warning         Unhealthy               Liveness pr
obe failed: HTTP probe failed with statuscode: 404
  1m            13s             4       kubelet, minikube       spec.containers{webtest}        Normal          Killing                 Killing con
tainer with id docker://webtest:pod "webtest-692878837-fxpw4_default(4f6f205f-9b00-11e7-bc1a-080027fb95be)" container "webtest" is unhealthy, it wi
ll be killed and re-created.
  1m            12s             5       kubelet, minikube       spec.containers{webtest}        Normal          Created                 Created con
tainer
  1m            12s             5       kubelet, minikube       spec.containers{webtest}        Normal          Started                 Started con
tainer
praparns-MacBook-Pro%
```

Workshop 1.6: Liveness and Readiness

Resource Management and HPA

Resource Management and HPA

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

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 Meterial)
      - 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

Resource Management and HPA

- Container level configuration
  - Pods "webtest"

```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_v1
      resources:
        requests:
          memory: "16Mi"
          cpu: "100m"
        limits:
          memory: "32Mi"
          cpu: "1"
      ports:
        - containerPort: 5000
          protocol: TCP
```

System status

```
...ckage/kubernetes — Terminal MAC Pro — zsh    docker@iZj6chay173lo9ivaj1drnZ: ~ — -bash    Terminal MAC Pro — ssh ◂ minikube ssh

top - 07:39:14 up  4:10,  1 user,  load average: 0.06, 0.29, 0.34
Tasks: 233 total,   1 running, 232 sleeping,   0 stopped,   0 zombie
%Cpu0  :   0.0/0.0     0[                                                    ]
%Cpu1  :   0.0/0.0     0[                                                    ]
%Cpu2  :   0.7/0.7     1[                                                    ]
%Cpu3  :   0.0/0.0     0[                                                    ]
%Cpu4  :   0.0/0.0     0[                                                    ]
%Cpu5  :   0.0/0.0     0[                                                    ]
%Cpu6  :   0.7/0.7     1[                                                    ]
%Cpu7  :   0.0/1.3     1[|                                                   ]
%Cpu8  :   0.0/0.0     0[                                                    ]
%Cpu9  :   0.7/0.7     1[                                                    ]
GiB Mem : 30.2/1.953    [                                                    ]
GiB Swap:  0.0/0.977    [                                                    ]

  PID USER      PR  NI    VIRT    RES  %CPU %MEM     TIME+ S COMMAND
    1 root      20   0   37.8m   5.9m   0.0  0.3   0:14.97 S systemd
 1440 root      20   0   65.9m  20.9m   0.0  1.0   0:06.73 S  `- systemd-journal
 2311 root      20   0   25.6m   3.2m   0.0  0.2   0:01.81 S  `- systemd-udevd
 2608 root      20   0    8.9m   0.4m   0.0  0.0   0:00.46 S  `- getty
 2617 root      20   0   24.2m   2.0m   0.0  0.1   0:00.02 S  `- rpcbind
 2638 root      20   0   25.5m   2.8m   0.0  0.1   0:00.17 S  `- systemd-logind
```

Resource Management and HPA

- Container level configuration
  - Allocate status on node

```
  Namespace                  Name                                CPU Requests    CPU Limits      Memory Requests Memory Limits
  ---------                  ----                                ------------    ----------      --------------- -------------
  default                    webtest                             100m (1%)       1 (10%)         16Mi (0%)       32Mi (1%)
  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%)
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  CPU Requests  CPU Limits      Memory Requests Memory Limits
  ------------  ----------      --------------- -------------
  365m (3%)     1 (10%)         176Mi (9%)      202Mi (10%)
```

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

```
top - 07:46:38 up  4:17,  1 user,  load average: 0.61, 0.47, 0.38
Tasks: 236 total,   3 running, 233 sleeping,   0 stopped,   0 zombie
%Cpu0  :   0.0/0.0     0[                                                  ]
%Cpu1  :   3.4/0.7     4[||                                                ]
%Cpu2  :   2.0/0.0     2[|                                                 ]
%Cpu3  :   6.8/43.2   50[||||||||||||||||||||||||||||||||||                ]
%Cpu4  :   2.0/0.7     3[|                                                 ]
%Cpu5  :   2.0/1.3     3[||                                                ]
%Cpu6  :   0.0/0.0     0[                                                  ]
%Cpu7  :   6.0/43.6   50[||||||||||||||||||||||||||||||||||                ]
%Cpu8  :   3.5/0.7     4[|||                                               ]
%Cpu9  :   2.1/1.4     3[|||                                               ]
GiB Mem : 31.2/1.953    [                                                  ]
GiB Swap:  0.0/0.977    [                                                  ]
```

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[||||||||||||||||||||||||||||||||||||||||||||||||||||]
%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

- 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
    - PersistenctVolumeClaims
    - Request.Storage
  - Counting Object
    - Pods
    - Service
    - Replication Controller
    - ConfigMap
    - Secrets

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: webtest-quota
  labels:
    name: webtest_quota
    owner: Praparn_L
    version: "1.0"
    module: Quota
    environment: development
spec:
  hard:
    pods: "4"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "4"
    limits.memory: 4Gi
```

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

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

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: webtest_limit
  labels:
    name: webtest_limit
    owner: Praparn_L
    version: "1.0"
    module: LimitRange
    environment: development
spec:
  limits:
  - max:
      cpu: "1"
      memory: 1Gi
    min:
      cpu: 200m
      memory: 6Mi
    type: Pod
  - default:
      cpu: 300m
      memory: 200Mi
    defaultRequest:
      cpu: 200m
      memory: 100Mi
    max:
      cpu: "1"
      memory: 1Gi
    min:
      cpu: 100m
      memory: 3Mi
    type: Container
```

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

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

Resource Management and HPA

- Deployment

```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
     nodePort: 32500
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
```

```yaml
    module: WebServer
    environment: development
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 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
```

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

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="1",memor
y=1Gi --record --namespace=webtest-namespace
deployment "webtest" resource requirements updated
praparns-MacBook-Pro% kubectl rollout status deployment/webtest --namespace=webtest-namespace
deployment "webtest" successfully rolled out
```

```
Containers:
  webtest:
    Container ID:       docker://e30e6233d0288a4b8ae47c12ac286e0fead22a53a2fee83739
    Image:              labdocker/cluster:webservicelite_v1
    Image ID:           docker-pullable://labdocker/cluster@sha256:f0f261307a9a0ce8
705ce67504326940
    Port:               5000/TCP
    State:              Running
      Started:          Sat, 08 Jul 2017 21:43:10 +0700
    Ready:              True
    Restart Count:      0
    Limits:
      cpu:      1
      memory:   1Gi
    Requests:
      cpu:              1
      memory:           1Gi
    Environment:        <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-cww0k (ro)
```

Workshop 1.7: Resource MNG/HPA

- Part 2: Namespace level configuration

192.168.99.100:32500

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

Page 1

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

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

Kubernetes 101: Production-Workload Orchestration

Day 2

By: Praparn L (eva10409@gmail.com)

kubernetes by Google

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
  - PetSets and StatefulSets

ConfigMap and Secret

ConfigMap and Secret

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

ConfigMap and Secret

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

kubectl create configmap <name> <option>

- Ex: "kubectl create configmap webmodule_configmap --from-literal=REDIS_HOST=localhost"
- Ex: "kubectl create –f webmodule_configmap.yml"

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: webmodule_configmap
  namespace: webmicroservice
  labels:
    name: "webmodule_configmap"
    owner: "Praparn_L"
    version: "1.0"
    module: "ConfigMap"
    environment: "development"
data:
  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$
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: databasemodule_secret
  namespace: webmicroservice
  labels:
    name: "databasemodule_secret"
    owner: "Praparn_L"
    version: "1.0"
    module: "Secret"
    environment: "development"
type: Opaque
data:
  username: cm9vdA==
  password: cGFzc3dvcmQ=
```

ConfigMap and Secret

ConfigMap and Secret

Pods.yml: databasemodule_pod.yml

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

Sourcecode: main.py

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

Pods.yml: webmodule_pod.yml

```yaml
      - name: webservice
        image: labdocker/cluster:webservice
        env:
          - name: "REDIS_HOST"
            value: "localhost"
```

ConfigMap and Secret

Secret.yml: databasemodule_secret.yml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: databasemodule-secret
  namespace: webmicroservice
  labels:
    name: "databasemodule-secret"
    owner: "Praparn_L"
    version: "1.0"
    module: "Secret"
    environment: "development"
type: Opaque
data:
  username: cm9vdA==
  password: cGFzc3dvcmQ=
```

Deploy.yml: databasemodule_deploy_config.yml

```yaml
apiVersion: "v1"
kind: Deployment
metadata:
  name: maindb
  labels:
    name: "maindb"
    owner: "Praparn_L"
    version: "1.0"
    module: "maindb"
    environment: "development"
spec:
  replicas: 1
  template:
    metadata:
      labels:
  spec:
    containers:
    - name: maindb
      image: labdocker/mysql:latest
      ports:
        - containerPort: 3306
          protocol: TCP
      env:
        - name: username
          valueFrom:
            secretKeyRef:
              name: databasemodule-secret
              key: username
        - name: password
          valueFrom:
            secretKeyRef:
              name: databasemodule-secret
              key: password
```

ConfigMap and Secret

Secret.yml:
databasemodule_secret.yml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: databasemodule-secret
  namespace: webmicroservice
  labels:
    name: "databasemodule-secret"
    owner: "Praparn_L"
    version: "1.0"
    module: "Secret"
    environment: "development"
type: Opaque
data:
  username: cm9vdA==
  password: cGFzc3dvcmQ=
```

Deploy.yml:
webmodule_deploy_config.yml

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

ConfigMap.yml:
webmodule_configmap.yml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: webmodule-configmap
  namespace: webmicroservice
  labels:
    name: "webmodule-configmap"
    owner: "Praparn_L"
    version: "1.0"
    module: "ConfigMap"
    environment: "development"
data:
  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>
```

```
praparns-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
praparns-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

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

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: linenotify
  labels:
    name: linenotify
    owner: Praparn_L
    version: "1.0"
    module: Job
    environment: development
spec:
  template:
    metadata:
      name: linenotify
    spec:
      containers:
      - name: linenotify
        image: labdocker/linenotify
        env:
        - name: TITLE
          value: "BOT:Praparn L."
        - name: INTERVAL
          value: "10000"
        - name: HEAP_HIGH
          value: "40"
        - name: MEM_HIGH
          value: "20"
        - name: SH_OS
          value: "Y"
        - name: TOKEN
          value: "EIeqyillzkzpaCo1ROrebZTcGbIyzDZHpOjcdd0t6CX"
      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

Workshop 2.2: Job and CronJob

Workshop 2.2: Job and CronJob

- Task1: Create Jobs for Monitor System via LINE

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: linenotify
  labels:
    name: linenotify
    owner: Praparn_L
    version: "1.0"
    module: Job
    environment: development
spec:
  template:
    metadata:
      name: linenotify
    spec:
      containers:
      - name: linenotify
        image: labdocker/linenotify
        env:
        - name: TITLE
          value: "BOT:Praparn L."
        - name: INTERVAL
          value: "10000"
        - name: HEAP_HIGH
          value: "20"
        - name: MEM_HIGH
          value: "20"
        - name: SH_OS
          value: "N"
        - name: TOKEN
          value: "EIeqyillzkzpaCo1ROrebZTcGbIyzDZHpOjcdd0t6CX"
      restartPolicy: Never
```

LINE

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

kubernetes by Google

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)
```

kubernetes

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.

Debug Log and Monitoring

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

Workshop 2.3: Log and Monitoring

Kubernetes: Production Workload Orchestration

kubernetes by Google

Ingress Network

Ingress Network

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

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

  - Service quite limit and non flexible for operate
    - How to handle for multiple service on same port ?
    - How to limit protocol for access ? (HTTP/HTTPS/FTP etc)
    - How to blinding 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:

Request:
http://192.168.99.101:30500
http://192.168.99.101:32500

Ingress Network

- Ingress:

Ingress Network

SVC: webtest1

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webtest1
  labels:
    name: webtest1
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  selector:
    name: webtest1
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development

  ports:
   - port: 80
     name: http
     targetPort: 5000
     protocol: TCP
---
```

Ingress: ingresswebtest

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingresswebtest
spec:
  rules:
  - host: webtest1.kubernetes labthailand.com
    http:
      paths:
      - backend:
          serviceName: webtest1
          servicePort: 80
  - host: webtest2.kubernetes labthailand.com
    http:
      paths:
      - backend:
          serviceName: webtest2
          servicePort: 80
```

SVC: webtest2

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webtest2
  labels:
    name: webtest2
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  selector:
    name: webtest2
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development

  ports:
   - port: 80
     name: http
     targetPort: 5000
     protocol: TCP
```

Ingress Network

- Ingress (TLS):

Ingress Network

www.selfsignedcertificate.com

NMac Ked - Mac O... | Medium | jerkins | 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

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webtest1
  labels:
    name: webtest1
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  selector:
    name: webtest1
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development

  ports:
   - port: 80
     name: http
     targetPort: 5000
     protocol: TCP
---
```

Ingress: ingresswebtest

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingresswebtesttls
spec:
  tls:
  - hosts:
    - webtest1.kubernetes labthailand.com
    secretName: webtest1secret
  - hosts:
    - webtest2.kubernetes labthailand.com
    secretName: webtest2secret
  rules:
  - host: webtest1.kubernetes labthailand.com
    http:
      paths:
      - backend:
          serviceName: webtest1
          servicePort: 80
  - host: webtest2.kubernetes labthailand.com
    http:
      paths:
      - backend:
          serviceName: webtest2
          servicePort: 80
```

Secret: webtest1

```yaml
apiVersion: v1
data:
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURMVENDQWhXZ0F3SUJBZ0lK
  tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUUVB
kind: Secret
metadata:
  name: webtest1secret
  namespace: default
type: Opaque
```

SVC: webtest2

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webtest2
  labels:
    name: webtest2
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  selector:
    name: webtest2
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development

  ports:
   - port: 80
     name: http
     targetPort: 5000
     protocol: TCP
```

Secret: webtest2

```yaml
apiVersion: v1
data:
  tls.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURMVENDQWhXZ0F3SU
  tls.key: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS
kind: Secret
metadata:
  name: webtest2secret
  namespace: default
type: Opaque
```

Kubernetes: Production Workload Orchestration

kubernetes by Google

Workshop 2.4: Ingress Network

Kubernetes in real world

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

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 100Mbits/s 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.

kubernetes

Documentation Blog Partners Community Case Stud

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 by Google

Kubernetes in real world

- Issue on Kubernetes 1.7.1

kubernetes / kubeadm

Watch 38 | Star 158 | Fork 20

Code | Issues 89 | Pull requests 1 | Projects 0 | Wiki | Insights

Kubeadm join fail #344

New issue

Closed PLoic opened this issue 11 days ago · 8 comments

PLoic commented 11 days ago

What happened?

Hello, I have an issue when I try to do a `kubeadm join` it appear to be succeed because I see :

```
[preflight] Starting the kubelet service
[discovery] Trying to connect to API Server "X.X.X.X:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://X.X.X.X:6443"
[discovery] Cluster info signature and contents are valid, will use API Server
"https://X.X.X.X:6443"
[discovery] Successfully established connection with API Server "X.X.X.X: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.

But on master node when I do a `kubectl get nodes` I only see my master node.

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications because you were mentioned.

luxas commented 5 days ago — Member

@praparn See: #347
It will be fixed in v1.7.2
Meanwhile, you can just set `--skip-preflight-checks`

praparn commented 3 days ago

@luxas Note with thanks krab

https://github.com/kubernetes/kubeadm/issues/344#issuecomment-316109138

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

```
#Install Base docker-engine
sudo apt-get remove docker docker-engine
sudo apt-get -y install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get -y install docker-ce
sudo groupadd docker
sudo usermod -aG docker $USER
sudo systemctl enable docker

#Install Kubernetes Base
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.7.0/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
sudo apt-get update && apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo touch /etc/apt/sources.list.d/kubernetes.list
#sudo bash -c 'echo "deb http://apt.kubernetes.io/ kubernetes-xenial-1.7 main" >
/etc/apt/sources.list.d/kubernetes.list'
sudo bash -c 'echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >
/etc/apt/sources.list.d/kubernetes.list'
sudo apt-get update
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 --toke
n 8c2350.f55343444a6ffc46
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
[init] Using Kubernetes version: v1.7.0
[init] Using Authorization modes: [Node RBAC]
[preflight] Running pre-flight checks
[preflight] WARNING: docker version is greater than the most recently validated version. Docker versio
n: 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.de
fault 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 need to run (as a regular user):

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

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

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

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

root@kubernetes-ms:~#
```

Kubernetes in real world

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

```
root@kubeserve-ms: ~ — zsh   praparn@kubernetes-ms: ~ — s...   praparn@kubernetes-1: ~ — ssh...   praparn@kubernetes-2: ~ — ssh...
praparn@kubernetes-ms:~$ kubectl apply -n kube-system -f "https://cloud.weave.works/k8s/net?k8s-versio
n=$(kubectl version | base64 | tr -d '\n')"
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
daemonset "weave-net" created
praparn@kubernetes-ms:~$ 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 versio
n: 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 s
igning 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

Machine: Kubernetes_MS

kubernetes

Kubectl

Role: Node
IP Address: 192.168.99.200
vCPU: 4
Memory: 2 GB

Machine: Kubernetes_1

kubernetes

Role: Node
IP Address: 192.168.99.201
vCPU: 4
Memory: 2 GB

Machine: Kubernetes_2

kubernetes

Role: Node
IP Address: 192.168.99.202
vCPU: 4
Memory: 2 GB

Command/Monitor

Gcloud

Google Cloud Platform

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:
Labels:                 beta.kubernetes.io/arch=amd64
                        beta.kubernetes.io/os=linux
                        kubernetes.io/hostname=kubernetes-1
                        storage=SSD
```

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

```
Name:                   kubernetes-ms
Role:
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:

```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
  nodeSelector:
    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 R
eason                   Message
  ---------     --------        -----   ----                    -------------                   ------
--      ------                  -------
  12m           12m             1       default-scheduler                                       NormaS
cheduled                Successfully assigned webtest to kubernetes-ms
  12m           12m             1       kubelet, kubernetes-ms                                  NormaS
uccessfulMountVolume    MountVolume.SetUp succeeded for volume "default-token-1ltqv"
  12m           12m             1       kubelet, kubernetes-ms  spec.containers{webtest}        NormaP
ulling                  pulling image "labdocker/cluster:webservicelite"
  11m           11m             1       kubelet, kubernetes-ms  spec.containers{webtest}        NormaP
ulled                   Successfully pulled image "labdocker/cluster:webservicelite"
  11m           11m             1       kubelet, kubernetes-ms  spec.containers{webtest}        NormaC
reated                  Created container
  11m           11m             1       kubelet, kubernetes-ms  spec.containers{webtest}        NormaS
tarted                  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:
Labels:                 beta.kubernetes.io/arch=amd64
                        beta.kubernetes.io/os=linux
                        kubernetes.io/hostname=kubernetes-1
                        storage=SSD
```

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

```
Name:                   kubernetes-ms
Role:
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:

```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
  nodeSelector:
    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:
    Image:              labdocker/cluster:webservicelite
    Image ID:
    Port:               5000/TCP
    State:              Waiting
      Reason:           ContainerCreating
    Ready:              False
    Restart Count:      0
    Environment:        <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-1ltqv (ro)
Conditions:
  Type          Status
  Initialized   True
  Ready         False
  PodScheduled  True
Volumes:
  default-token-1ltqv:
    Type:       Secret (a volume populated by a Secret)
    SecretName: default-token-1ltqv
    Optional:   false
QoS Class:      BestEffort
Node-Selectors: kubernetes.io/hostname=kubernetes-1
```

Affinity

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

Node Affinity

Pods YAML File:

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: beta.kubernetes.io/os
            operator: In #In, NotIn, Exists, DoesNotExist, Gt, Lt
            values:
            - linux
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: storage
            operator: In
            values:
            - SSD
  containers:
    - name: webtest
      image: labdocker/cluster:webservicelite
      ports:
        - containerPort: 5000
          protocol: TCP
```

Inter-pod Affinity and Anti-affinity

Pods YAML File:

```yaml
apiVersion: "v1"
kind: Pod
metadata:
  name: webtest2
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: environment
            operator: In
            values:
            - development
        topologyKey: kubernetes.io/hostname
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: module
              operator: In
              values:
              - DBServer
          topologyKey: storage
  containers:
    - name: webtest2
      image: labdocker/cluster:webservicelite
      ports:
        - 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:

```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
  tolerations:
  - key: "dedicated"
    operator: "Equal"
    value: "admin"
    effect: "NoSchedule"
  nodeSelector:
     kubernetes.io/hostname: kubernetes-1
```

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

Workshop 2.6: Orchestrator Assignment

Machine: Kubernetes_MS

kubernetes

Kubectl

Role: Node
IP Address: 192.168.99.200
vCPU: 4
Memory: 2 GB

Label: storage = M2

Machine: Kubernetes_1

kubernetes

Role: Node
IP Address: 192.168.99.201
vCPU: 4
Memory: 2 GB

Label: storage = SSD

Machine: Kubernetes_2

kubernetes

Role: Node
IP Address: 192.168.99.202
vCPU: 4
Memory: 2 GB

Label: storage = SAS

Command/Monitor

Gcloud

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

Stateful Application Deployment

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

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

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-share-pv
  labels:
    name: nfs-share-pv
    owner: Praparn_L
    version: "1.0"
    module: PV
    environment: development
spec:
  capacity:
    storage: 1Gi
  storageClassName: ""
  accessModes:
    - ReadWriteMany
  nfs:
    server: 192.168.99.200
    path: "/var/nfsshare"
```

Persistent Volume Claims

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-share-pvc
  labels:
    name: nfs-share-pvc
    owner: Praparn_L
    version: "1.0"
    module: PVC
    environment: development
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 500Mi
  selector:
    matchLabels:
      name: nfs-share-pv
      owner: Praparn_L
      version: "1.0"
      module: PV
      environment: development
```

Persistent Volume

- Deployment reference

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: webtest
  labels:
    name: web
    owner: Praparn_L
    version: "1.0"
    module: WebServer
    environment: development
spec:
  replicas: 3
  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
          volumeMounts:
            - name: nfs-share-pvc
              mountPath: "/usr/src/app"
      volumes:
      - name: nfs-share-pvc
        persistentVolumeClaim:
          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
```

Persistent Volume

- Create PVC

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

Persistent Volume

- Create Deployment

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

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:~$
```

10.38.14.201:32500

Apps NMac Ked - Mac OS... Medium jerkins vagrant Mesos Vue docker NGINX Taiwan Kubernetes PWA_Progressive_W... MY

Welcome Page from Container Python Lab

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

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.strft
ime("%c") +'\n'

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
docker@kubernetes-ms:~$
```

10.38.14.201:32500

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:~$
```

10.38.14.201:32500

Welcome Page from Container Python Lab

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

StatefulSet

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

```yaml
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: TEST
spec:
  serviceName: "TEST"
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: labdocker/nginx:latest
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: DB
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: DB
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: TEST
  labels:
    app: TEST
spec:
  ports:
  - port: 80
    name: TEST
  clusterIP: None
  selector:
    app: nginx
```

StatefulSet

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
dokuwiki
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
```

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