



Kubernetes


Workshop with Java















Somkiat

Home









Somkiat Puisungnoen

Update Info
1


View Activity Log
10+

...


Timeline
About
Friends 3,138
Photos
More ▾



When did you work at Opendream?
×





...
22 Pending Items



Intro


Software Craftsmanship


Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556



Agile Practitioner and Technical at SPRINT3r


Post

Photo/Video

Live Video

Life Event



What's on your mind?


Public ▾

Post



Somkiat Puisungnoen

15 mins · Bangkok ·


Java and Bigdata

...

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Kubernetes workshop

3

Facebook interface for the page **somkiat.cc**. The top navigation bar includes the Facebook logo, the page name **somkiat.cc**, a search icon, and user profile icons for **Somkiat** and **Home**. Below the navigation bar, the main menu includes **Page**, **Messages**, **Notifications** (with a red badge showing 3), **Insights**, **Publishing Tools**, **Settings**, and **Help**.

The page content area features a large video player showing a man in a white Superman t-shirt with "SOMKIAT.CC" printed on it, posing against a white wall. A blue call-to-action button is overlaid on the video, reading "Help people take action on this Page." with a close icon. Below the video, there are buttons for **Liked**, **Following**, **Share**, and a three-dot menu icon. A blue button labeled **+ Add a Button** is also visible.

The left sidebar contains the page name **somkiat.cc**, the handle **@somkiat.cc**, and a list of navigation options: **Home**, **Posts**, **Videos**, and **Photos**.



Topics day 1

Kubernetes concepts

1. Pods
2. Replication Controllers
3. Replica Sets
4. Deployments
5. Services
6. Jobs
7. Volumes

Deploy the Java web app to Kubernetes



Topics day 2

Advance concepts

1. Persistent volume
2. Stateful Sets
3. Horizontal Pod Autoscaling (HPA)
4. Deamon Sets
5. Health of Pods
6. Namespaces
7. Rolling update



<https://github.com/up1/course-kubernetes-in-practice>



Kubernetes

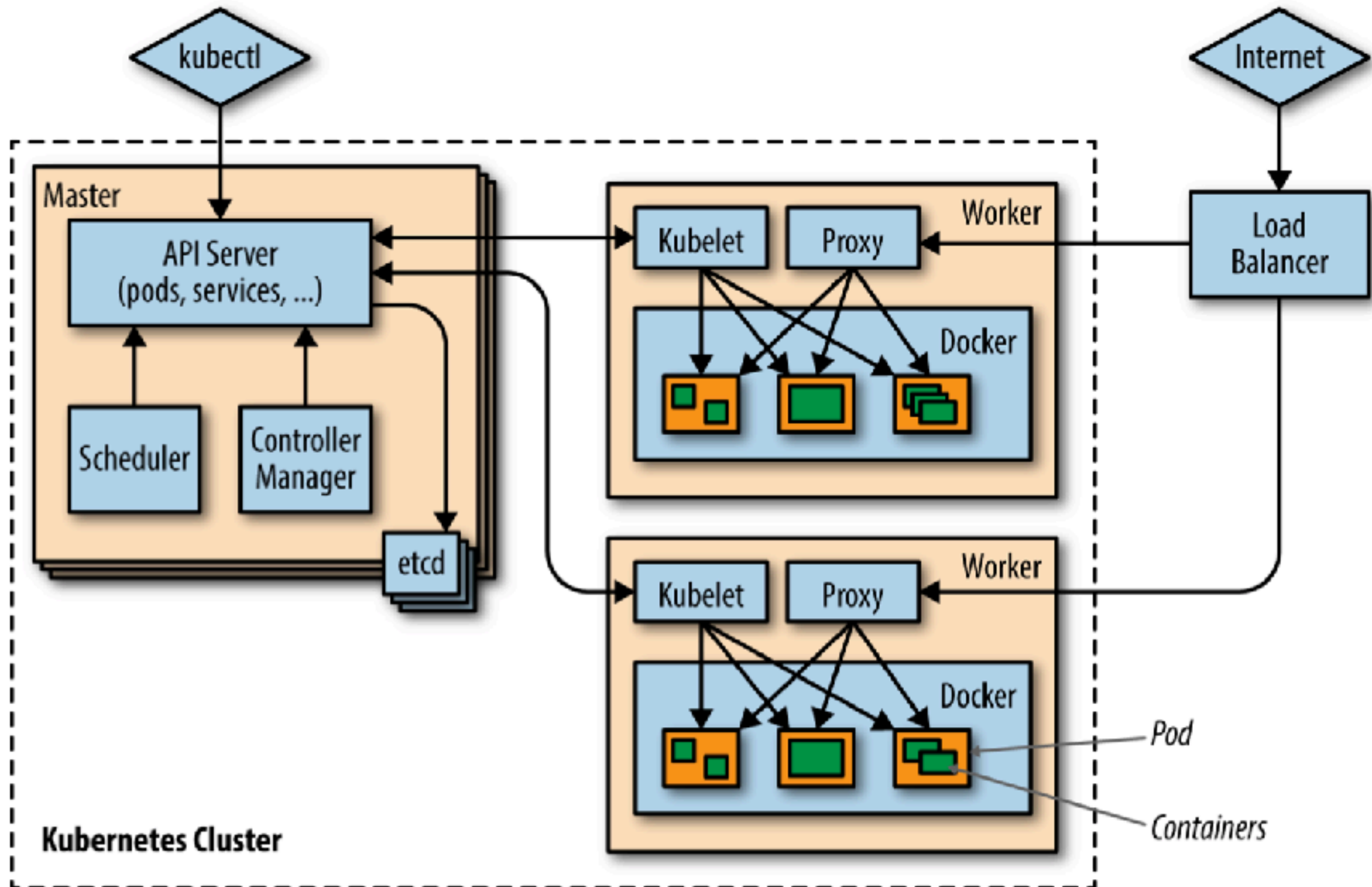
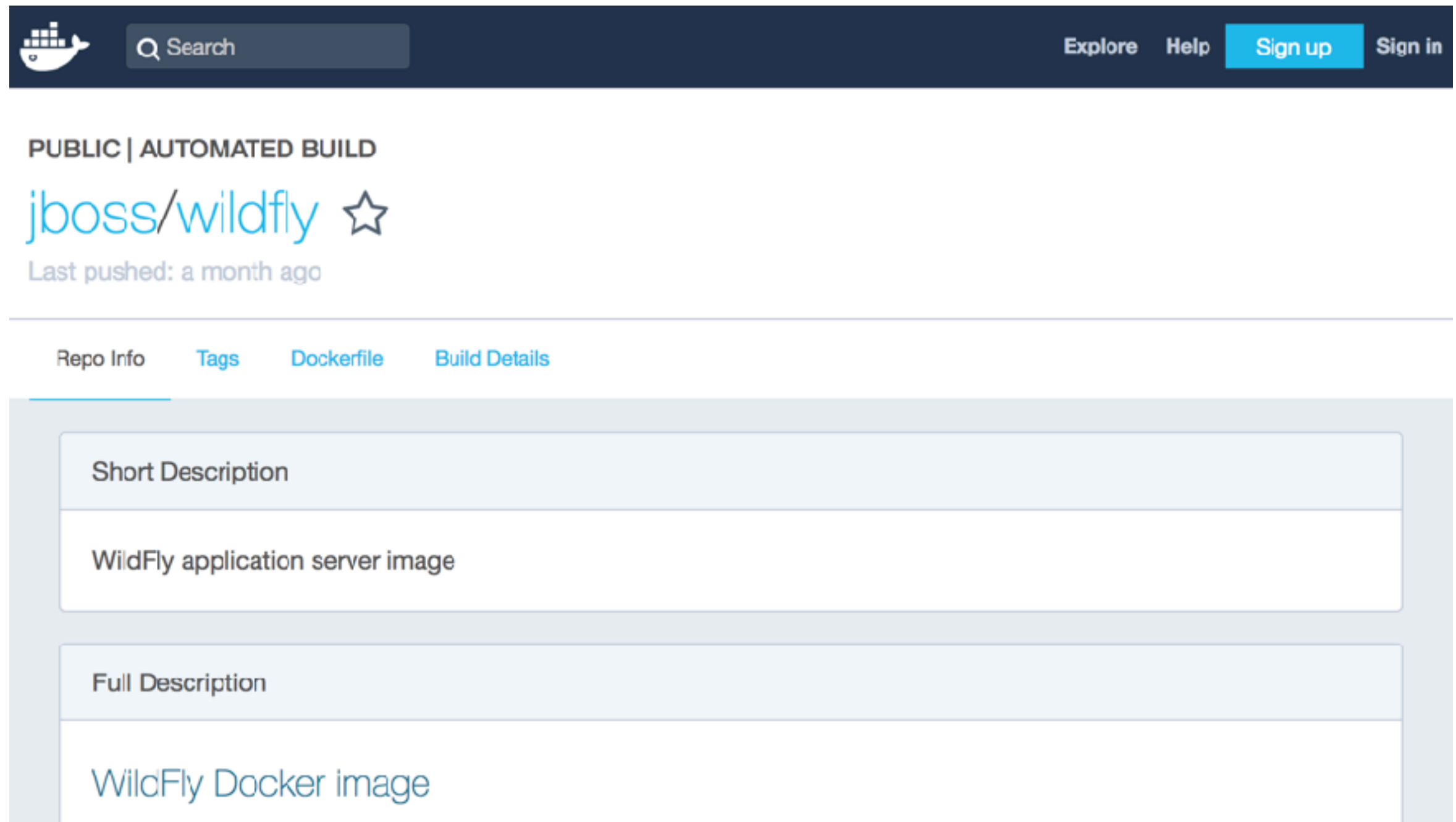


Image jboss/wildfly



The screenshot shows the Docker Hub interface for the `jboss/wildfly` image. At the top, there is a dark blue navigation bar with the Docker logo, a search bar, and links for 'Explore', 'Help', 'Sign up', and 'Sign in'. Below the navigation bar, the page indicates 'PUBLIC | AUTOMATED BUILD' and shows the repository name 'jboss/wildfly' with a star icon and the text 'Last pushed: a month ago'. A horizontal menu contains tabs for 'Repo Info', 'Tags', 'Dockerfile', and 'Build Details'. The 'Repo Info' tab is active, displaying a 'Short Description' section with the text 'WildFly application server image' and a 'Full Description' section with the text 'WildFly Docker image'.

<https://hub.docker.com/r/jboss/wildfly/>



Let's start



1. Pods

Smallest deployable unit

Logical collection of containers

Pods are created in namespaces(ns)

All containers in a pod share the ns, volume and networking



Configuration of Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: wildfly-pod
  labels:
    name: wildfly-pod
spec:
  containers:
  - name: wildfly
    image: jboss/wildfly:10.1.0.Final
    ports:
    - containerPort: 8080
```



2. Replication Controllers (RC)

Ensure that a specified # of pod replicas are running at any time

Automatically replaces if pods fail
Support scale-up and scale-down



Configuration of RC

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: wildfly-rc
spec:
  replicas: 2
  selector:
    app: wildfly-rc-pod
  template:
    metadata:
      labels:
        app: wildfly-rc-pod
    spec:
      containers:
        - name: wildfly
          image: jboss/wildfly:10.1.0.Final
          ports:
            - containerPort: 8080
```



3. Replica Sets

Next-generation of RC
Selector support



Configuration of Replica Sets

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: wildfly-rs
spec:
  replicas: 2
  selector:
    matchLabels:
      app: wildfly-rs-pod
    matchExpressions:
      - {key: tier, operator: In, values: ["backend"]}
      - {key: environment, operator: NotIn, values: ["prod"]}
  template:
    metadata:
      labels:
        app: wildfly-rs-pod
        tier: backend
        environment: dev
    spec:
      containers:
        - name: wildfly
          ...
```



4. Deployments

Provide declarative updates for Pods and
Replica Sets

Easy to start a RC or Replica set

Easy to check status of deployment

Update deployment to use a new image

Rollback deployment to previous version



Configuration of Deployments

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: wildfly-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: wildfly
    spec:
      containers:
      - name: wildfly
        image: jboss/wildfly:10.1.0.Final
        ports:
        - containerPort: 8080
```



5. Services

Abstraction to define a logical set of Pods
and policy to access them

IP of service doesn't change over time

Pods belong to a service are defined by a
label selector



Configuration of Services

```
apiVersion: v1
kind: Service
metadata:
  name: wildfly-service
spec:
  selector:
    app: wildfly-rc-pod
  ports:
    - name: web
      port: 8080
```



6. Volumes

Pods are ephemeral and work well with
stateless

Stateful containers require data to be
persisted outside

Directory that is accessible to the containers in
a Pod



Configuration of Volumes

spec:

containers:

- name: couchbase
- image: somkiat/couchbase:latest
- ports:
 - containerPort: 8091

Use Volume on Host Path

volumeMounts:

- mountPath: /var/couchbase/lib
- name: couchbase-data

volumes:

- name: couchbase-data
- hostPath:
 - path: /opt/data



Workshop#0

File /working-with-java/01-basic



Start Wildfly server

```
$kubectl run hello-wildfly --image=jboss/wildfly:12.0.0.Final --port=8080
```

```
$kubectl get pod, deployment
```

NAME	READY	STATUS	RESTARTS	AGE
hello-wildfly-7bd4cfc58-jw9xp	0/1	ContainerCreating	0	3m

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
hello-wildfly	1	1	1	0	3m

NAME	DESIRED	CURRENT	READY	AGE
hello-wildfly-7bd4cfc58	1	1	0	3m



Start Wildfly server

\$kubectl proxy



Try with configuration file

```
$kubectl create -f wildfly-pod.yaml
```

```
$kubectl create -f wildfly-rc.yaml
```

```
$kubectl create -f wildfly-service.yaml
```



Workshop #1

with Spring boot service

File /working-with-java/02-web-app



Structure of project



1. Build Spring Boot Service

Create executable JAR file

`./gradlew bootRepackage`

Create Docker image

`./gradlew buildDocker`



2. Push Docker image to Docker Hub

```
$docker login
```

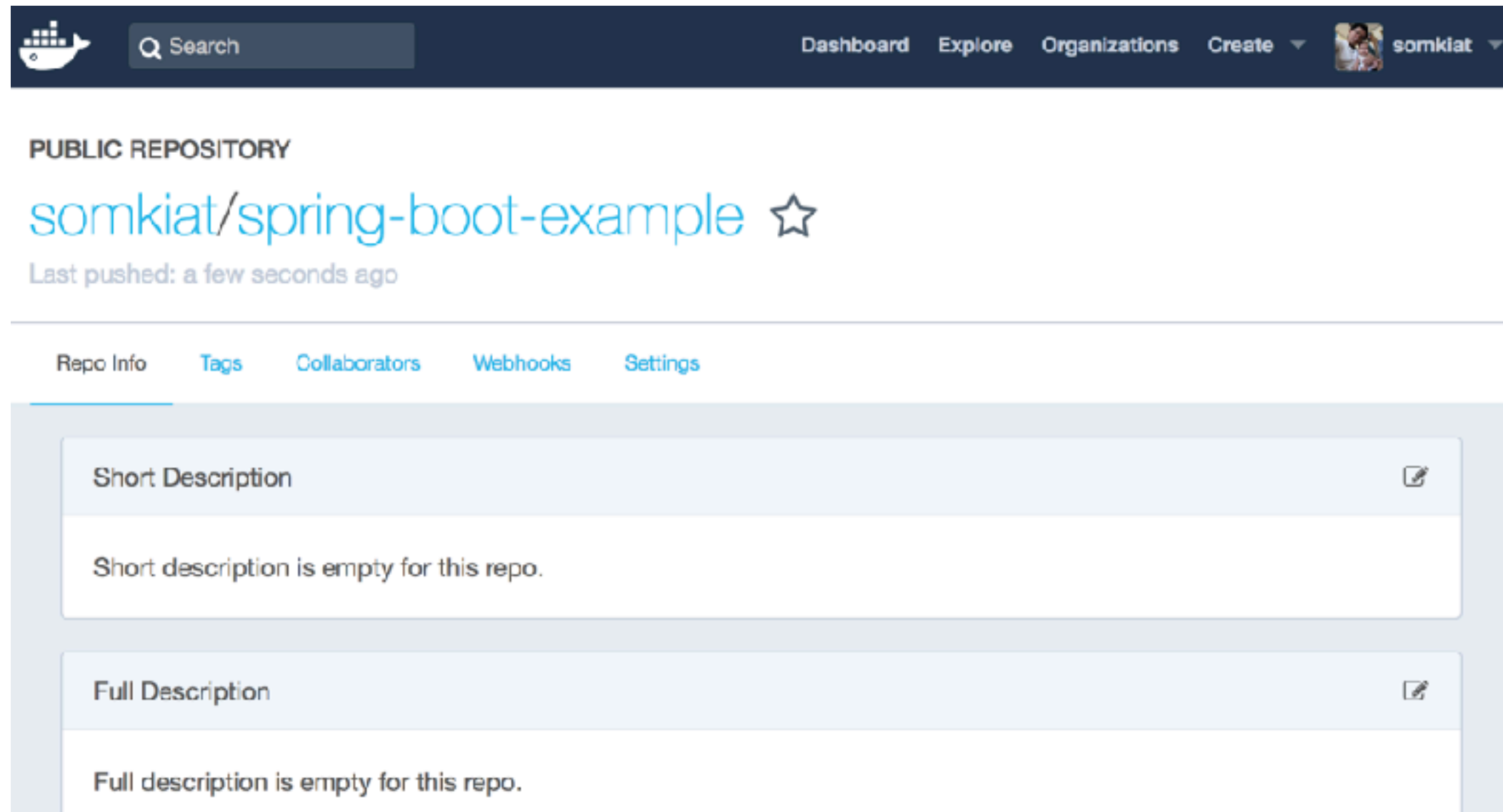
```
$docker push <username>/spring-boot-example
```

Required :: register at <https://hub.docker.com/>



2. Push Docker image to Docker Hub

Check your image



<https://hub.docker.com/r/somkiat/spring-boot-example/>



3. Deploy application

1. MongoDB

2. Spring boot service



3.1 Deploy MongoDB

```
$kubectl create -f mongo-controller.yaml
```

```
$kubectl create -f mongo-service.yaml
```



3.1 Mongo-controller.yaml

containers:

- image: mongo
name: mongo
ports:
 - name: mongo
containerPort: 27017
hostPort: 27017

volumeMounts:

- name: mongo-persistent-storage
mountPath: /data/db

Use Volume on Host Path

volumes:

- name: mongo-persistent-storage
hostPath:
 - path: /data/storage/mongodb



3.1 Mongo-controller.yaml

containers:

- image: mongo

name: mongo

ports:

- name: mongo

containerPort: 27017

hostPort: 27017

Mount volume from container

volumeMounts:

- name: mongo-persistent-storage

mountPath: /data/db

volumes:

- name: mongo-persistent-storage

hostPath:

path: /data/storage/mongodb



3.2 Deploy Spring boot service

```
$kubectl create -f boot-deployment.yaml
```

```
$kubectl create -f boot-service.yaml
```



3.2 Boot-deployment.yaml

spec:

containers:

- name: spring-boot-service
image: somkiat/spring-boot-example
ports:
- containerPort: 8080

Use environment variables

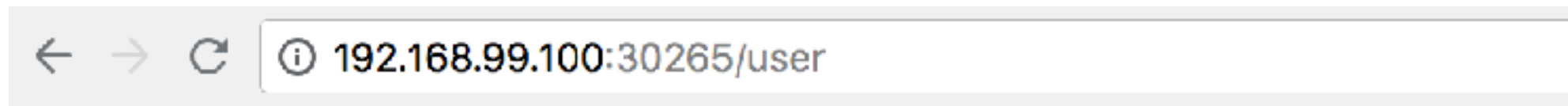
env:

- name: MONGO_URI
value: mongo-service



4. Testing

Try to add new data and see result in browser



```
[  
  - {  
    id: "5b043054cff47e0005b9f94e",  
    firstName: "Somkiat",  
    lastName: "Puisungnoen",  
    email: "xxx@gmail.com",  
    address: null  
  }  
]
```



Workshop #2

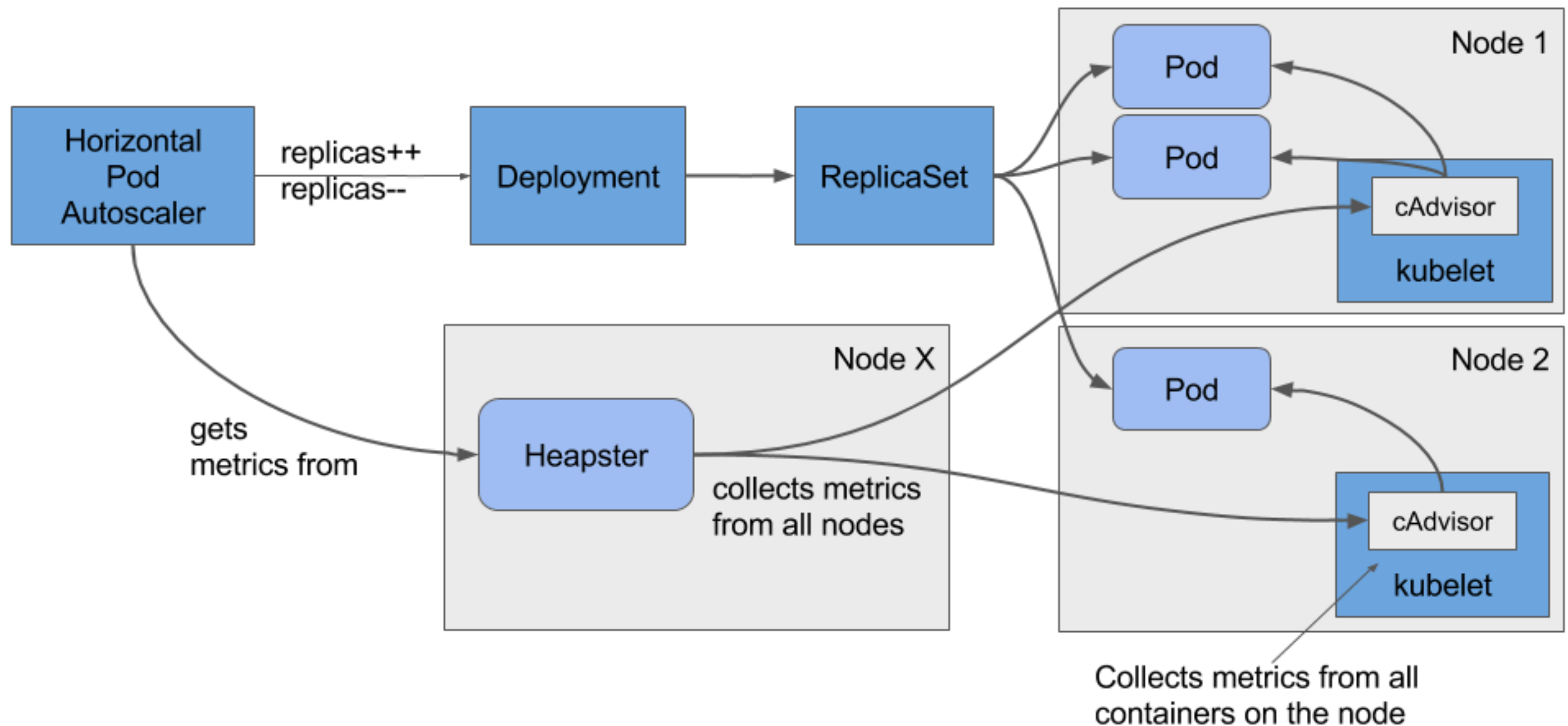
Horizontal Pods Autoscale

(HPA)

File /working-with-java/03-hpa



Components in Autoscaling



<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>



1. Enable hipster in minikube

\$minikube addons enable heapster

\$minikube addons start heapster

\$minikube addons open heapster



See result in Grafana

http://192.168.99.100:30002

The screenshot shows the Grafana web interface in a browser window. The address bar displays the URL `192.168.99.100:30002/?orgId=1`. The interface has a dark theme. At the top, there's a navigation bar with the Grafana logo, a 'Home' button, and a settings gear. On the right of the navigation bar, there are controls for 'Zoom Out', 'Last 6 hours' time range, and a refresh icon. The main content area is titled 'Home Dashboard'. Below this, there's a 'Getting Started with Grafana' section with a progress bar. The progress bar shows five steps: 'Install Grafana' (checked), 'Create your first data source' (checked), 'Create your first dashboard' (checked), 'Add Users' (highlighted with a green button), and 'Install apps & plugins' (with a gear icon). Below the progress bar, there are two columns. The left column has 'Starred dashboards' and 'Recently viewed dashboards'. Under 'Recently viewed dashboards', there are two items: 'Pods' and 'Cluster', each with a star icon. The right column has three sections: 'Installed Apps', 'Installed Panels', and 'Installed Datasources'. Each of these sections contains a message: 'None installed. Browse Grafana.net'.



2. Create metric server

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

<https://github.com/kubernetes-incubator/metrics-server>



3. Deploy Springboot service

```
$kubectl create -f boot-deployment.yaml  
$kubectl create -f boot-service.yaml
```



4. Create HPA in command line

```
$kubectl autoscale  
deployment/spring-boot-service-deployment  
--min=1 --max=5 --cpu-percent=5
```

min = minimum of replica

max = maximum of replica

cpu-percent = average of % of CPU usage



4. Create HPA in command line

\$kubectl get hpa

REFERENCE	TARGETS	MINPODS	MAXPODS
Deployment/spring-boot-service-deployment	0%/5%	1	5



4. Create HPA in command line

Scale-up can only happen if there was no rescaling within the last **3** minutes

Scale-down will wait for **5** minutes from the last rescaling

<https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/horizontal-pod-autoscaler.md#autoscaling-algorithm>



5. Load testing with command line



5. See result

\$kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
load-test-6f86656986-9qvdl	0/1	Error	0	4m
mongo-controller-xdh28	1/1	Running	0	1h
spring-boot-service-deployment-86c7b47b7b-9l9sn	1/1	Running	0	15m
spring-boot-service-deployment-86c7b47b7b-9srkg	1/1	Running	0	1m
spring-boot-service-deployment-86c7b47b7b-kbtrl	1/1	Running	0	1m
spring-boot-service-deployment-86c7b47b7b-pqbsm	1/1	Running	0	1m



5. See result

\$kubectl describe hpa


```
Metrics: ( current / target )
  resource cpu on pods (as a percentage of request): 134% (269m) / 5%
Min replicas: 1
Max replicas: 5
Conditions:
  Type           Status  Reason                Message
  ----           -
  AbleToScale    False   BackoffBoth           the time since the previous scale is still within both the downscale and upscale forbidden windows
  ScalingActive   True    ValidMetricFound      the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  True    TooManyReplicas       the desired replica count is more than the maximum replica count


















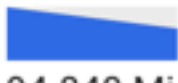








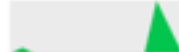








Events:
  Type           Reason                Age   From                Message
  ----           -
  Normal         SuccessfulRescale     1m    horizontal-pod-autoscaler  New size: 4; reason: cpu resource utilization (percentage of request) above target
  Warning        FailedGetResourceMetric 19s   horizontal-pod-autoscaler  unable to get metrics for resource cpu: unable to fetch metrics from API: the server could not find the requested resource (get pods.metrics.k8s.io)
  Warning        FailedUpdateStatus     19s   horizontal-pod-autoscaler  Operation cannot be fulfilled on horizontalpodautoscalers.autoscaling "spring-boot-service-deployment": the object has been modified; please apply your changes to the latest version and try again
  Warning        FailedComputeMetricsReplicas 19s   horizontal-pod-autoscaler  failed to get cpu utilization: unable to get metrics for resource cpu: unable to fetch metrics from API: the server could not find the requested resource (get pods.metrics.k8s.io)
```



5. See result

In dashboard of Pods

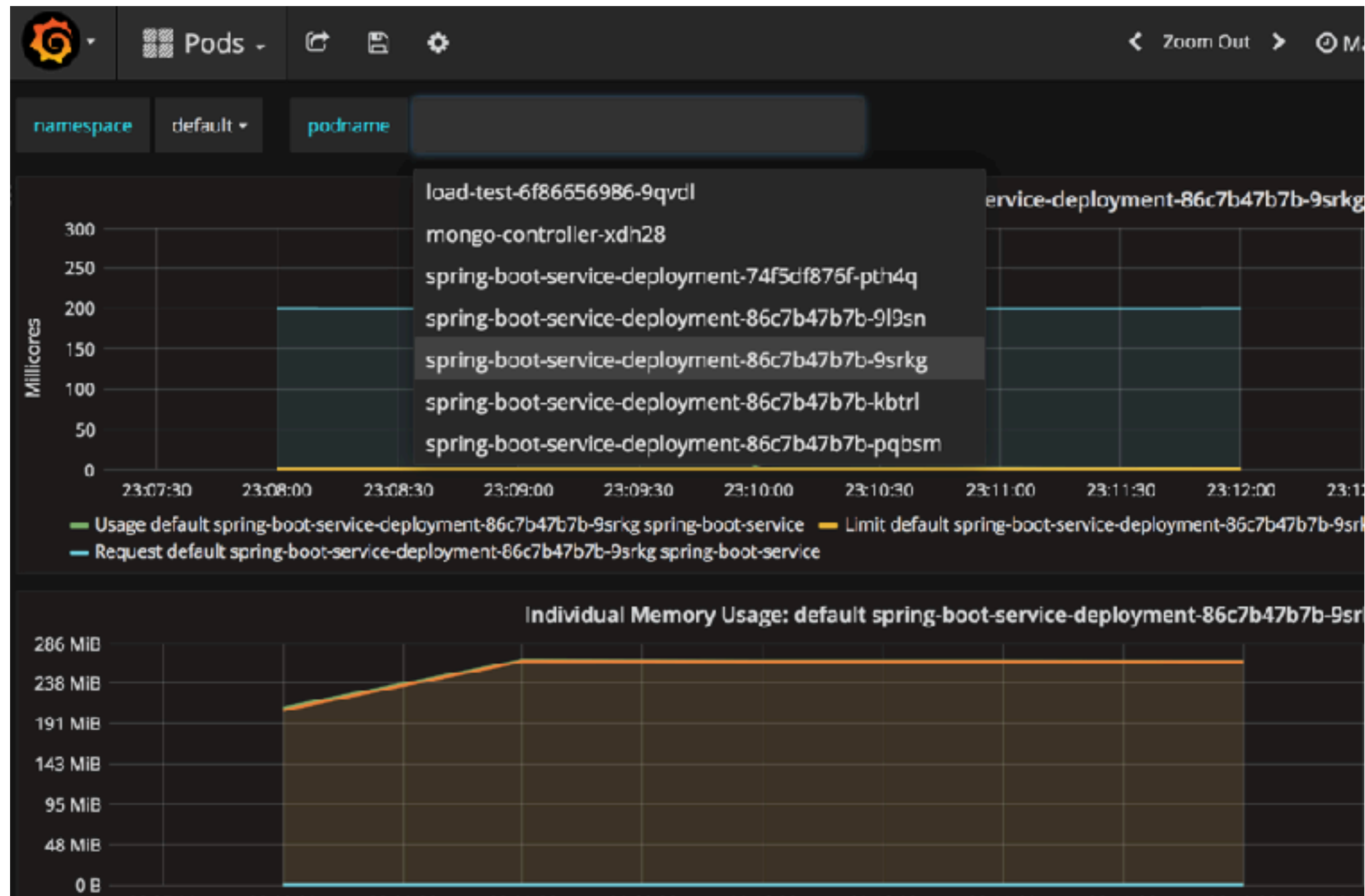
Pods 

Name 	Node	Status 	Restarts	Age 	CPU (cores)	Memory (bytes)		
 spring-boot-service-de	minikube	Running	0	2 minutes	 0.249	 265.926 Mi		
 spring-boot-service-de	minikube	Running	0	2 minutes	 0.03	 254.156 Mi		
 spring-boot-service-de	minikube	Running	0	2 minutes	 0.001	 94.840 Mi		
 load-test-6f86656986-	minikube	Running	1	5 minutes	 0	 688 Ki		
 spring-boot-service-de	minikube	Running	0	16 minutes	 0.018	 33.211 Mi		
 mongo-controller-xdh2	minikube	Running	0	an hour	 0.004	 24.453 Mi		



5. See result

In Grafana dashboard



Workshop #3

Secret and ConfigMap

File /working-with-java/04-secret-configmap



Secret

Good to working with secret data and config
Each container need some config and data

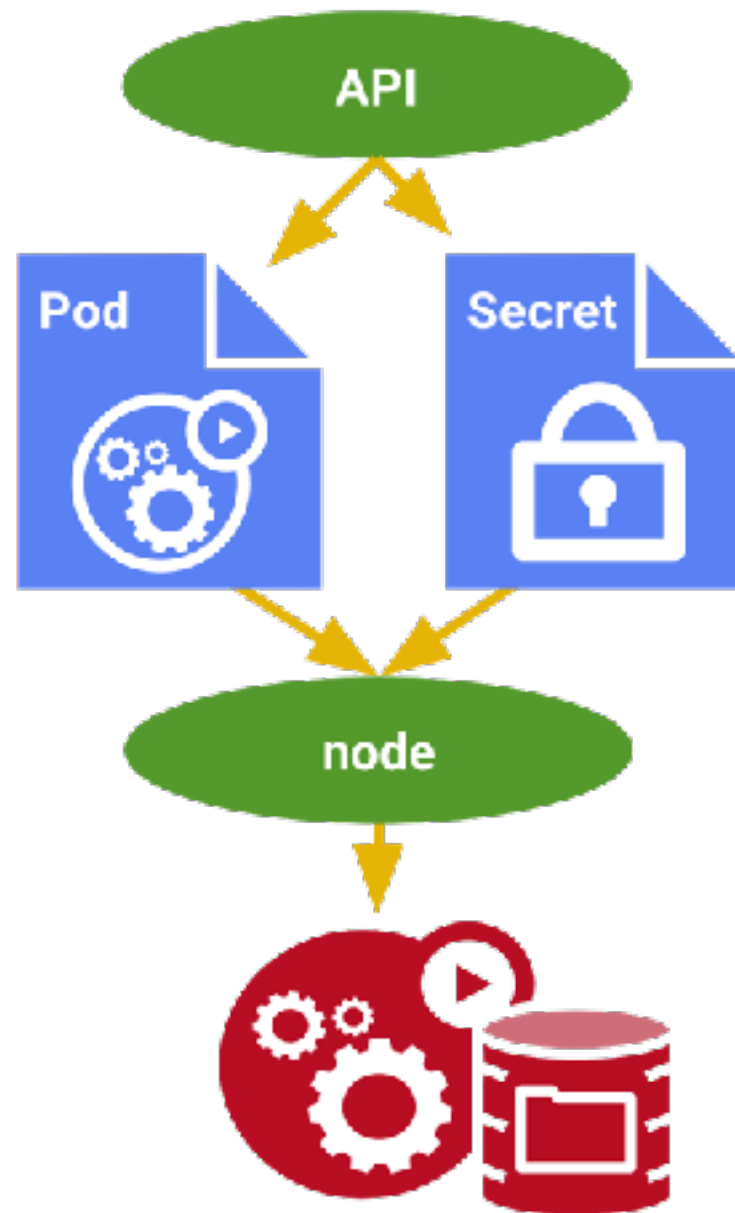
Store config/data in Pod/Container/Deployment ?

- Root password of database
- Environment variables
- Path to mount volume data



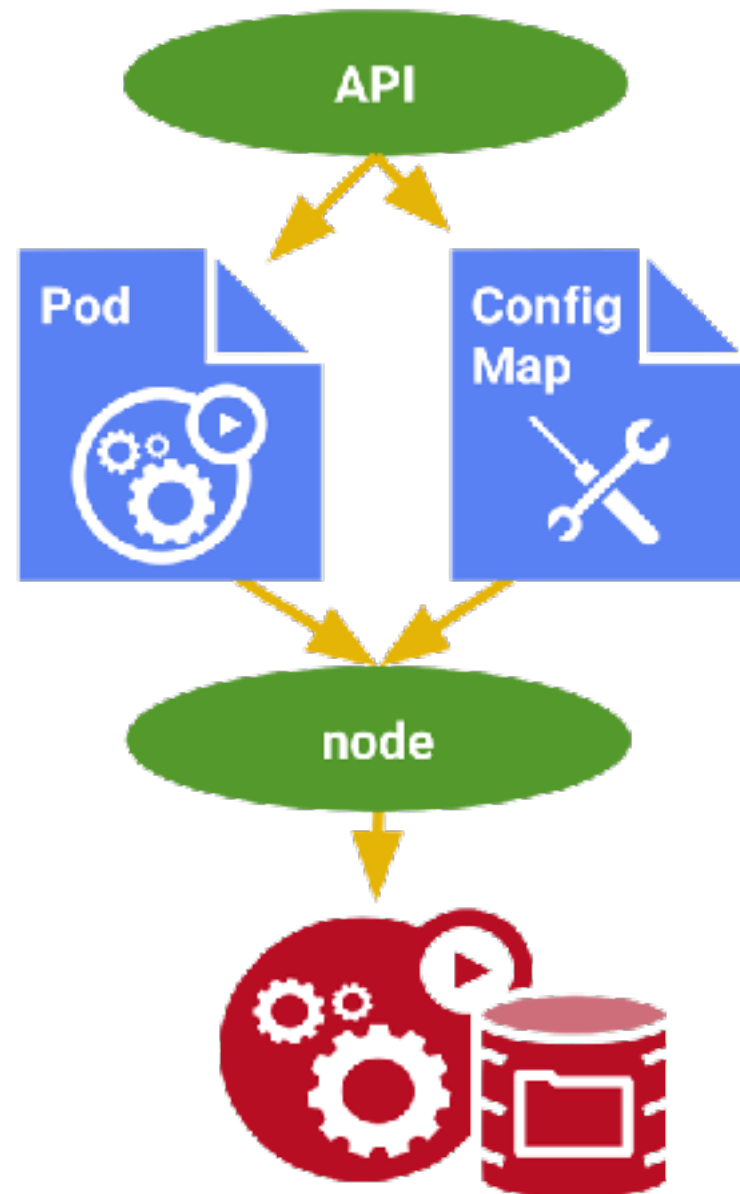
Secret

Secret encode sensitive data for keep secret



ConfigMap

ConfigMap provide centralize of configuration



Create ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: boot-configmap
  namespace: myservice
  labels:
    name: boot-configmap
data:
  MONGO_URI: mongo-service
```



Use Configmap from container

containers:

- name: spring-boot-service
image: somkiat/spring-boot-example
ports:
 - containerPort: 8080

env:

- name: MONGO_URI

valueFrom:

configMapKeyRef:

name: boot-configmap

key: MONGO_URI



Create Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: mongo-secret
  namespace: myservice
  labels:
    name: mongo-secret
type: Opaque
data:
  username: cm9vdA==
  password: cGFzc3dvcmQ=
```



Use Secret from container

spec:

containers:

- image: mongo
name: mongo
ports:
 - name: mongo
containerPort: 27017
hostPort: 27017

env:

- name: MONGO_USERNAME
valueFrom:
 - secretKeyRef:
 - name: mongo-secret
key: username
- name: MONGO_PASSWORD
valueFrom:
 - secretKeyRef:
 - name: mongo-secret
key: password



Workshop #4

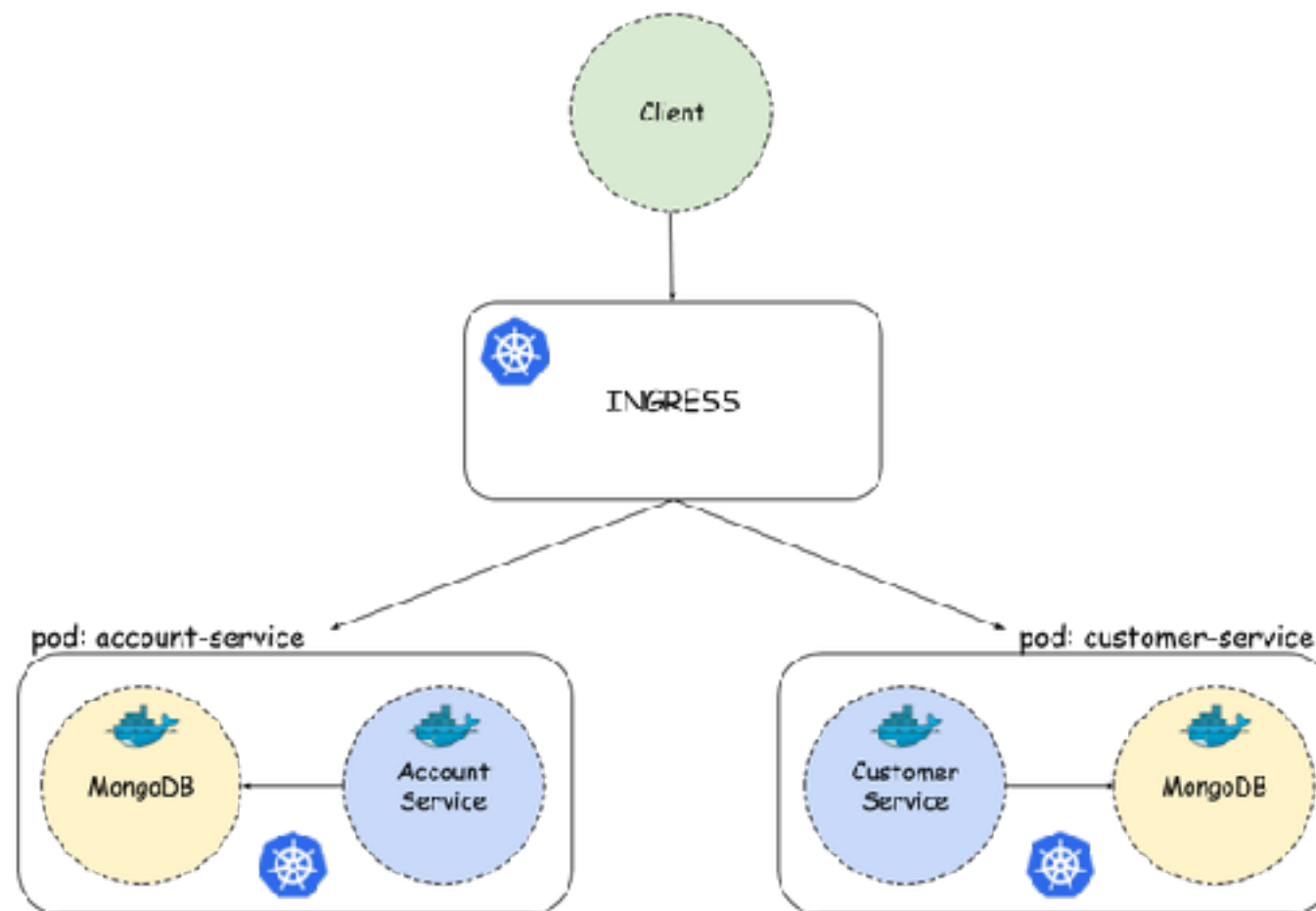
Ingress Network

File `/working-with-java/05-ingress`



Ingress Network

How to handle multiple services in same port ?
How to limit protocol to access ?



<https://kubernetes.io/docs/concepts/services-networking/ingress/>



Workshop #5

StatefulSet

File /working-with-java/06-stateful-set

