openprovider

People and software that help you grow

KUBERNETES
Дирижирование контейнерами Docker

Введение в архитектуру и демонстрация развертывания кластера.

Игорь Должиков, Openprovider

Все начинается с одного контейнера

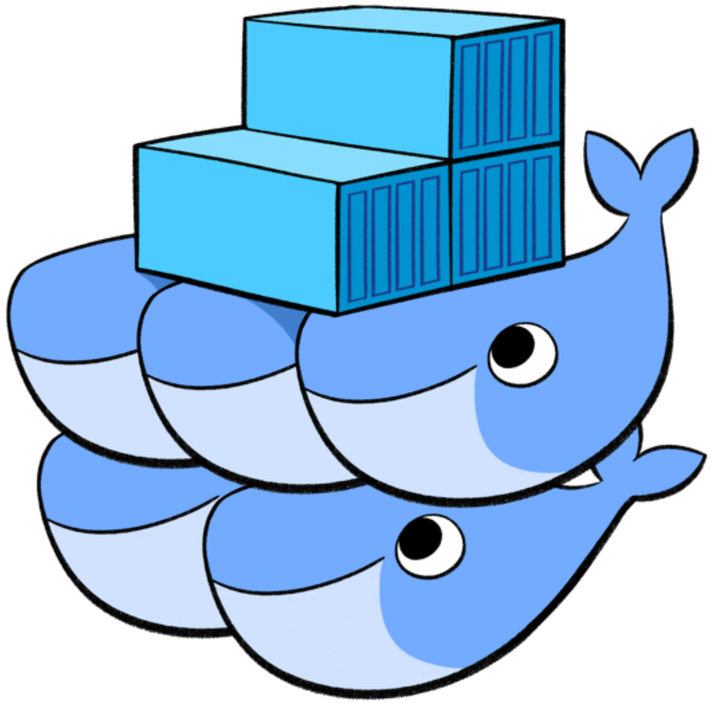
Несколько контейнеров, управляемых в ручную

Множество контейнеров без дирижера

Управляемый и сбалансированный кластер

Средства упраления контейнерами

- Docker Swarm - встроеная в Docker система управления.



Средства упраления контейнерами

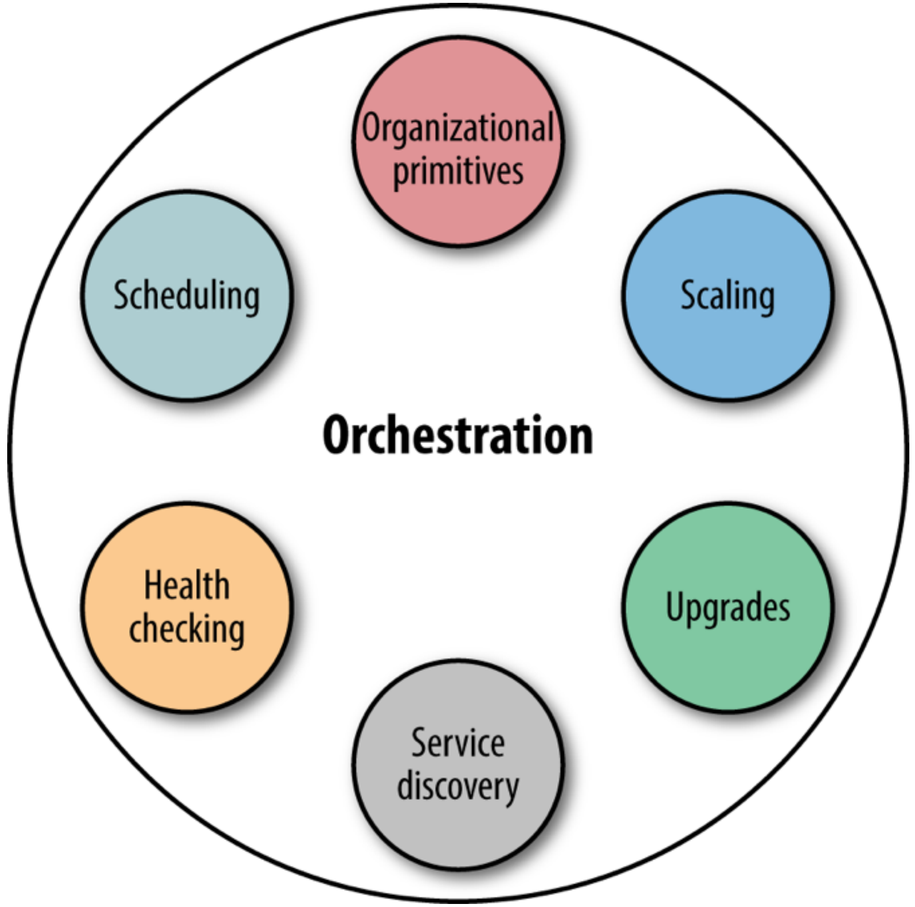
- Apache Mesos - мощная корпоративная система (применяется Твитер и Apple поддерживает более чем 10 000 контейнеров).

Средства упраления контейнерами

- Kubernetes - простое и гибкое решение от Google, которое благодаря сообществу активно развивается последние 2-3 года.

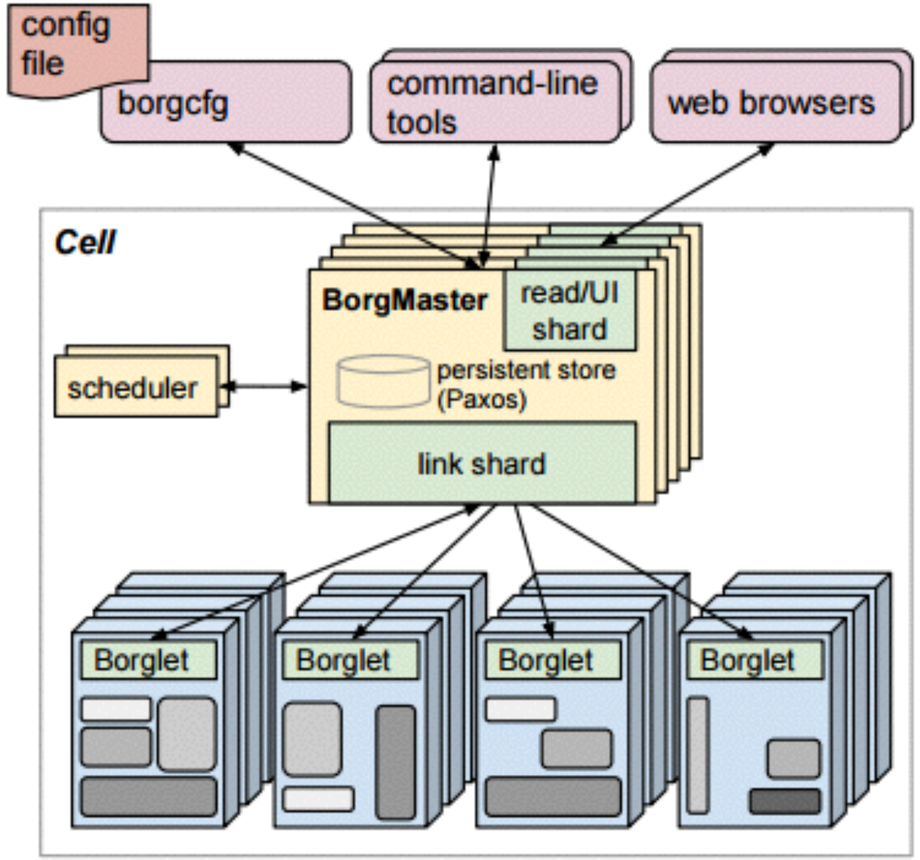
Средства упраления контейнерами

- И многие другие …



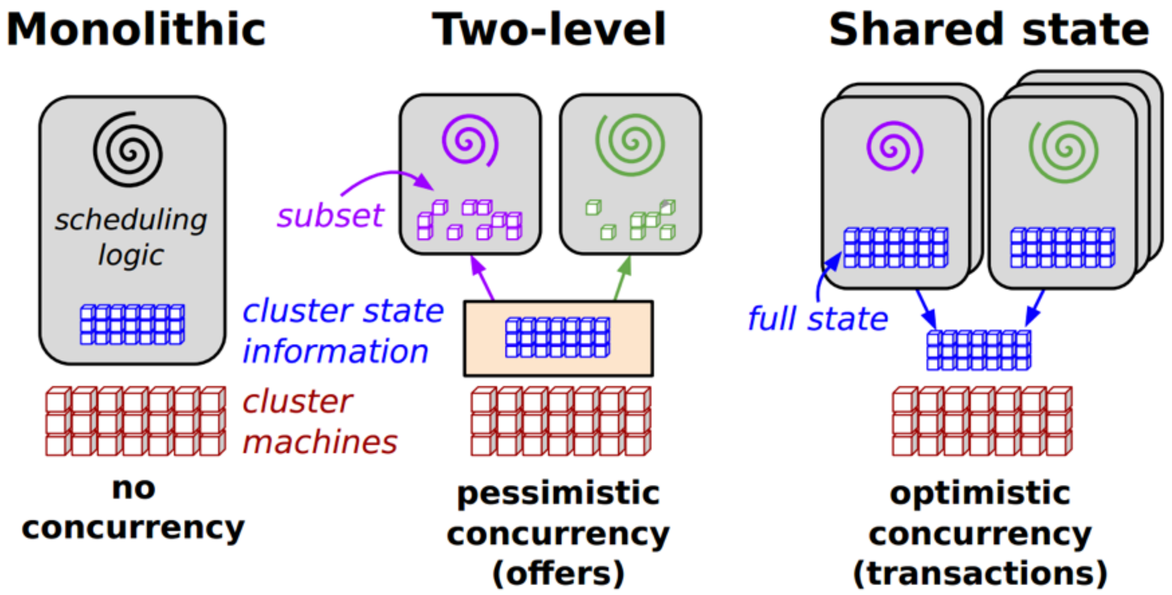
Происхождение Kubernetes

- Borg - Система управления контейнерами в Google.



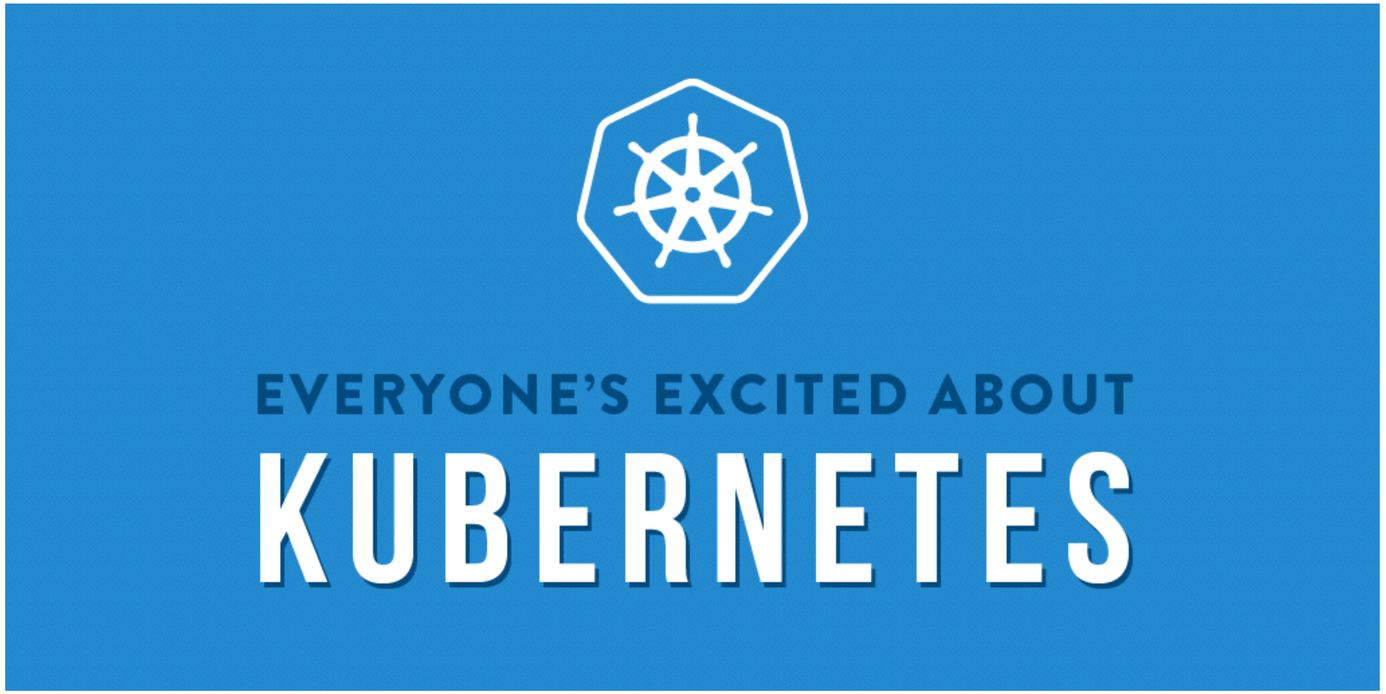
Происхождение Kubernetes

- Omega - Новая система оркестрации и управления сервисами в Google.

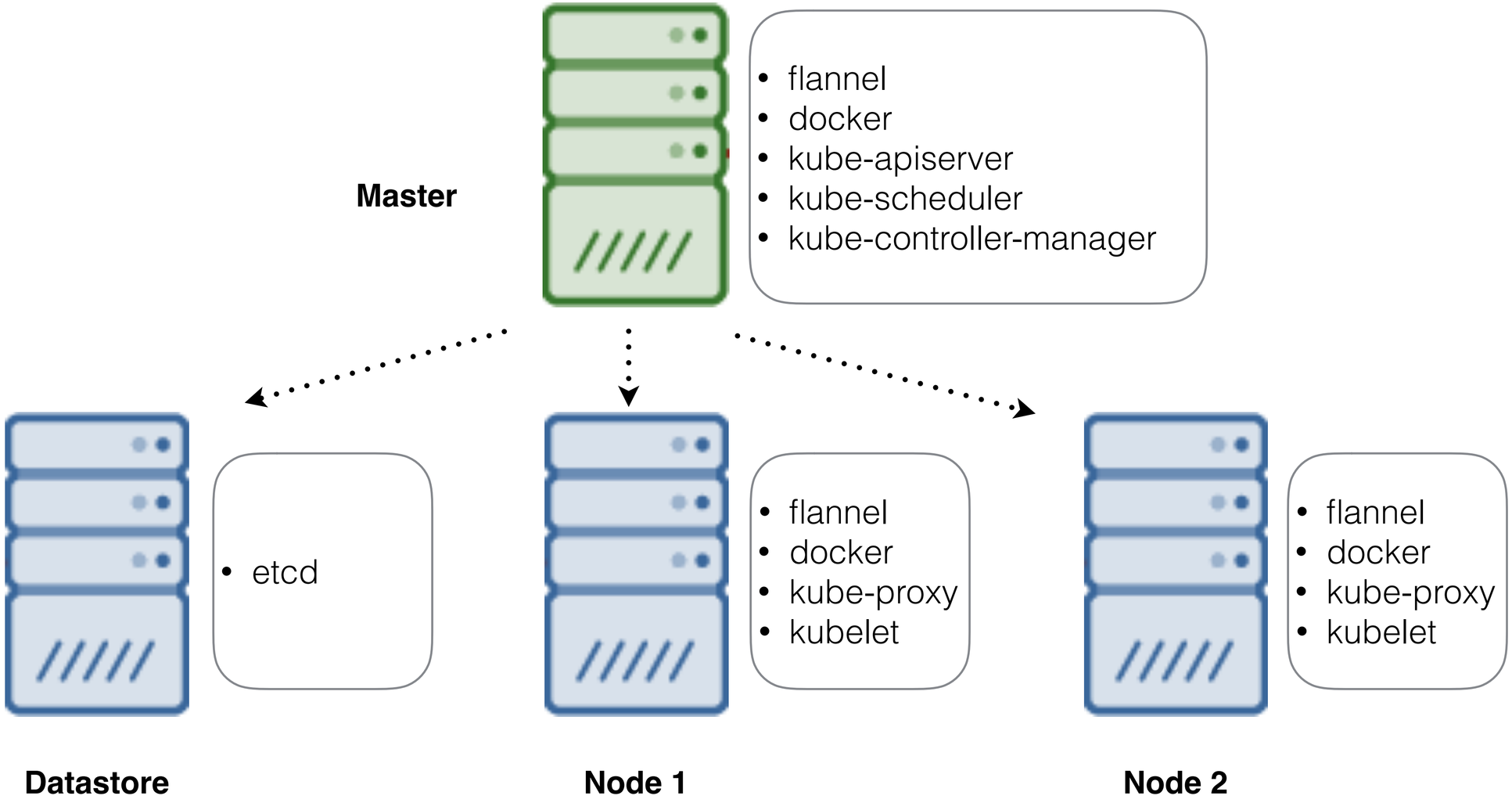


Происхождение Kubernetes

- Kubernetes - Opensource фреймворк, созданный на базе многолетнего опыта работы с контейнерами в Google.



Как устроен Kubernetes



Как устроен Kubernetes

- Kubernetes Master - централизованный сервис, который управляет кластером (посредством API, контроллера и планировщика).
- Cluster Datastore - хранилище конфигурации кластера (etcd).

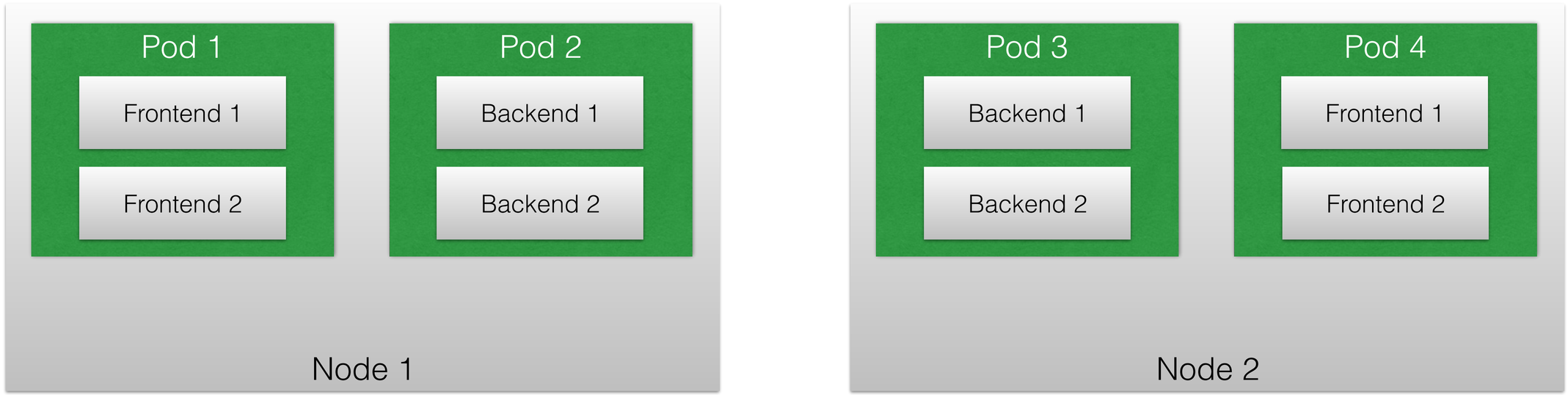
- kube-apiserver
- kube-scheduler
- kube-controller-manager

Master

- etcd

Datastore

Как устроен Kubernetes



- Nodes - VM в которых размещаются Pods и контейнеры.
- Pods - группа контейнеров, которые должны быть размещены на одной node.

Как устроен Kubernetes

- Kubectl - консоль управления кластером.
- Kubelet - агент, управляющий ресурсами и контейнерами на node.
- Kube proxy - предоставляют контейнеру связь с группой pods.

- kubectl
- kube-proxy
- kubelet

Master

- kubectl
- kube-proxy
- kubelet

Node

Как устроен Kubernetes

- Labels - метки, для классификации pods по каким-либо признакам.
- Replication Controllers - агенты, для горизонтального масштабирования сервисов.

- name = webapp
- version = 0.1
- type = production

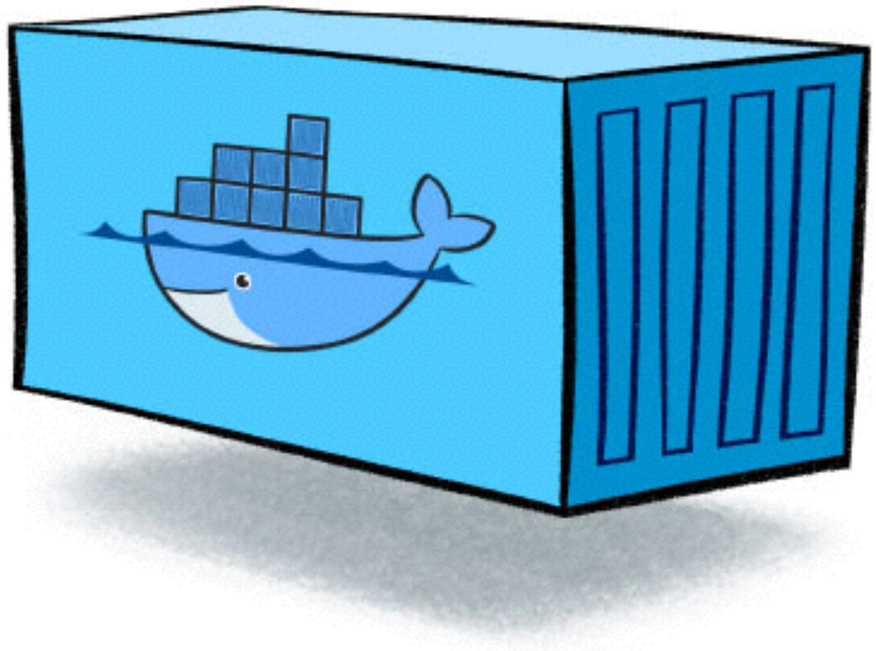
Node 1

- name = webapp
- version = 0.2
- type = stage

Node 2

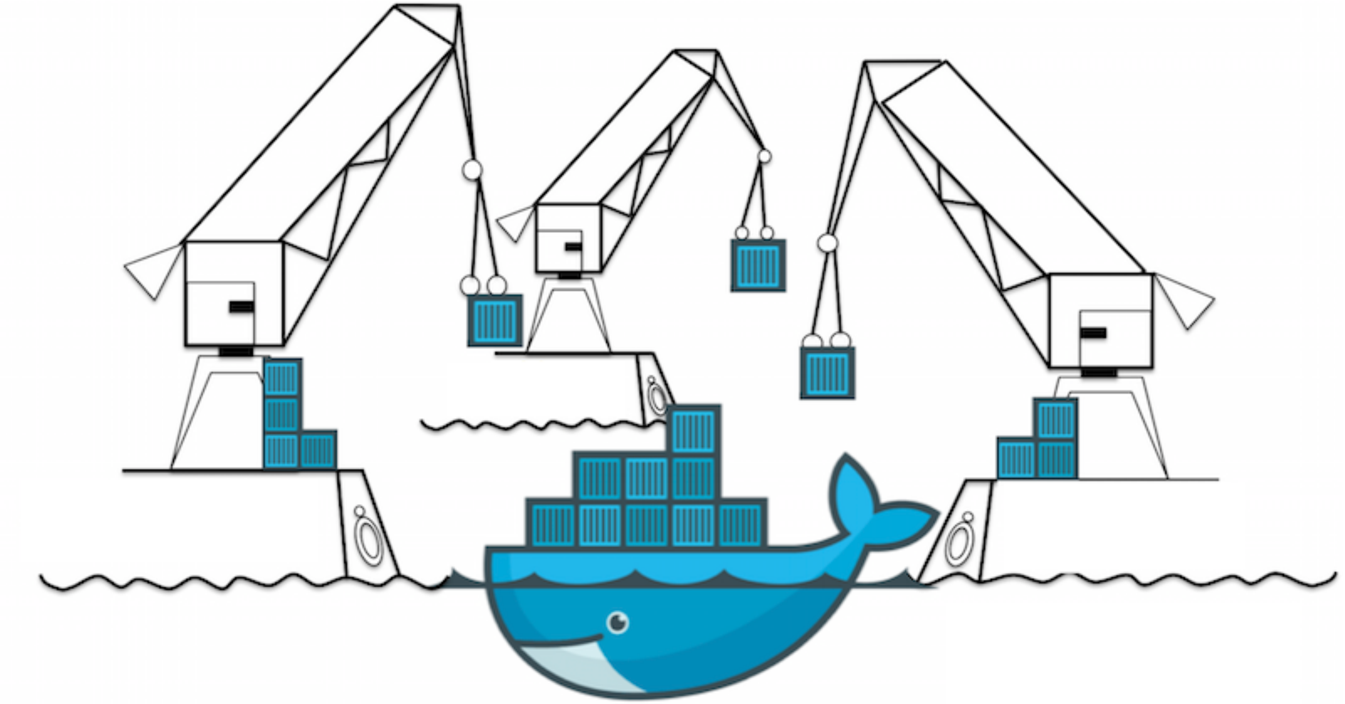
Особенности Kubernetes Cluster

- Автоматический мониторинг и рестарт контейнеров.



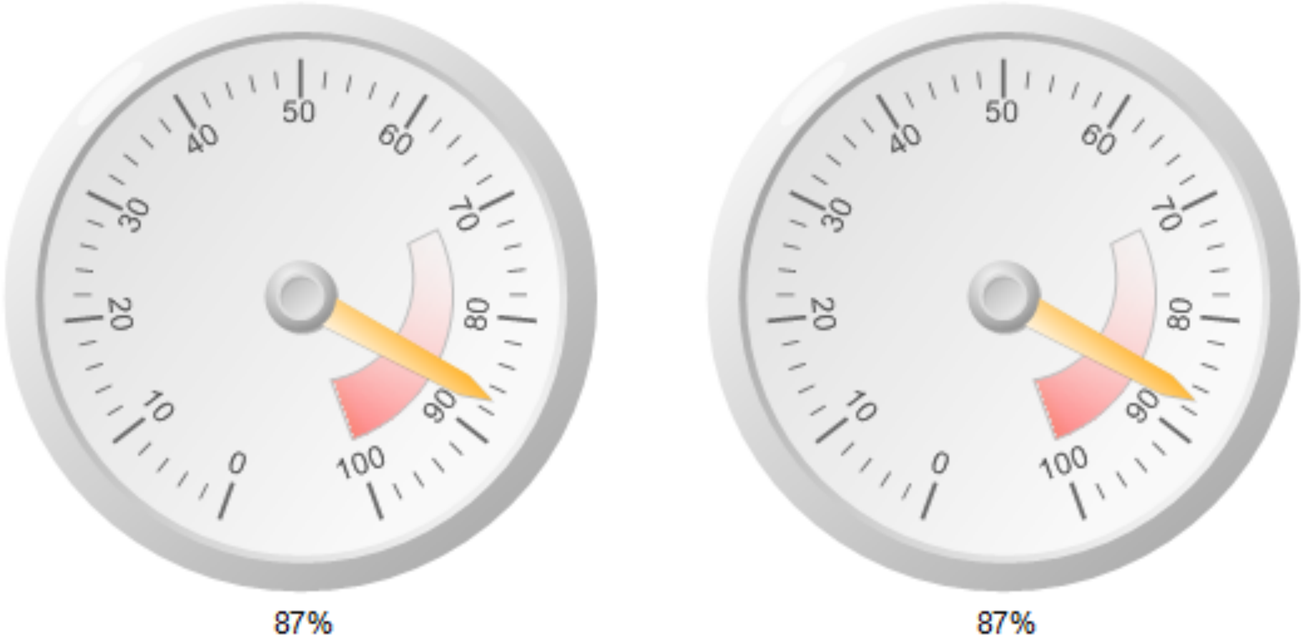
Особенности Kubernetes Cluster

- Автоматическое перемещение контейнеров с проблемной node.



Особенности Kubernetes Cluster

- Равномерная утилизация на каждой зарегистрированной node.

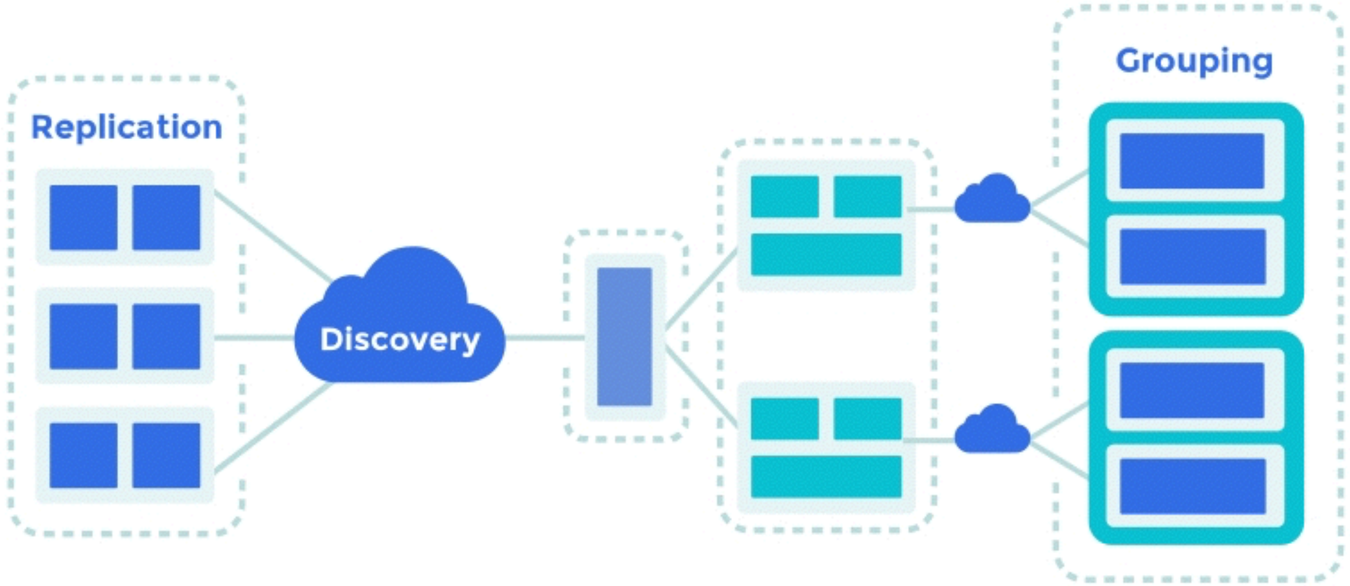


Особенности Kubernetes Cluster

- Rolling Update. Возможность проводить релиз без downtime и быстро вернуться на предыдущий релиз в случае неудачи.

Особенности Kubernetes Cluster

- Горизонтальное масштабирование и репликация.
- Группирование по меткам.
- Автоматическое обнаружение сервиса (discovery).

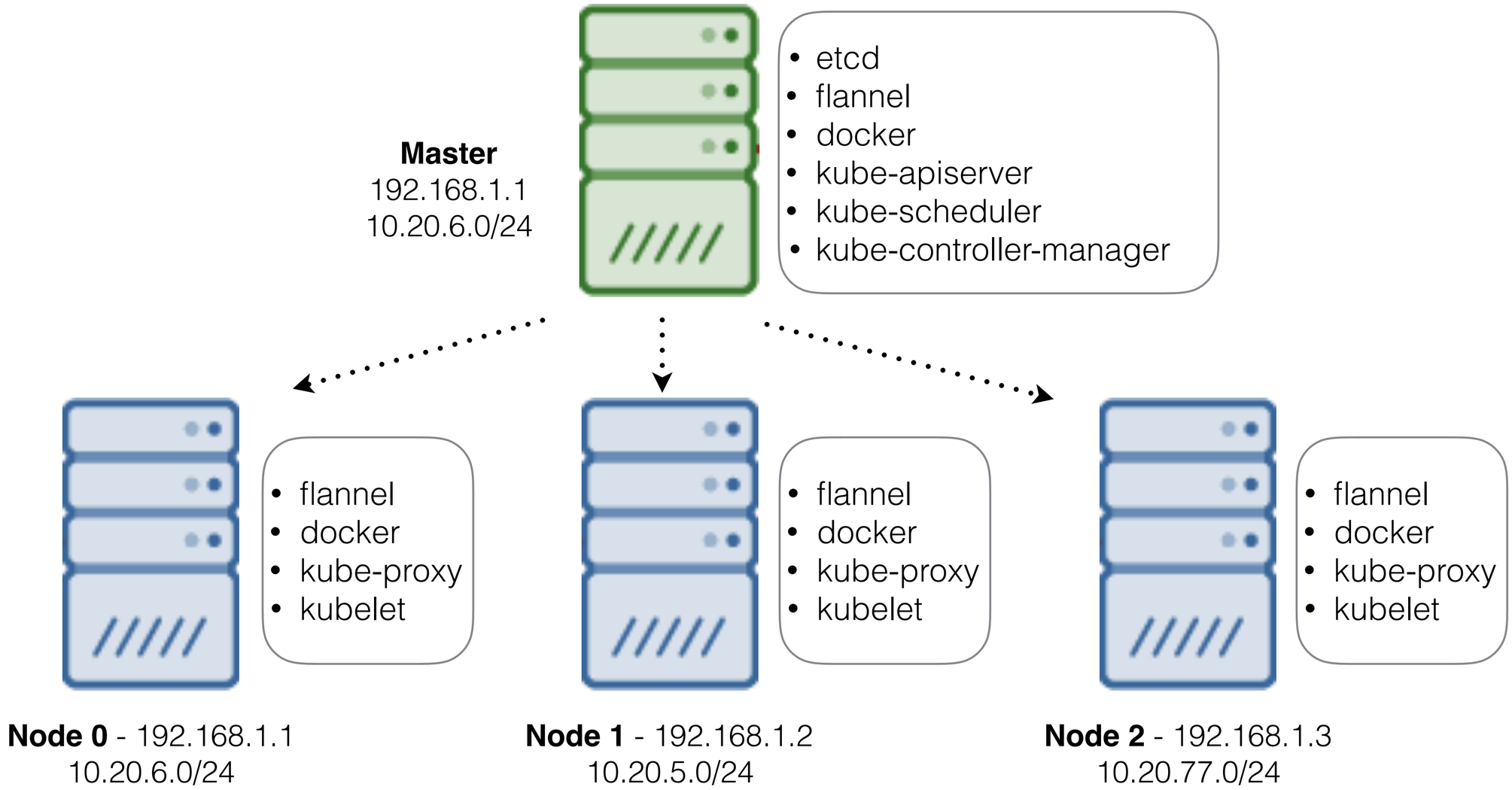


Setup Kubernetes Cluster

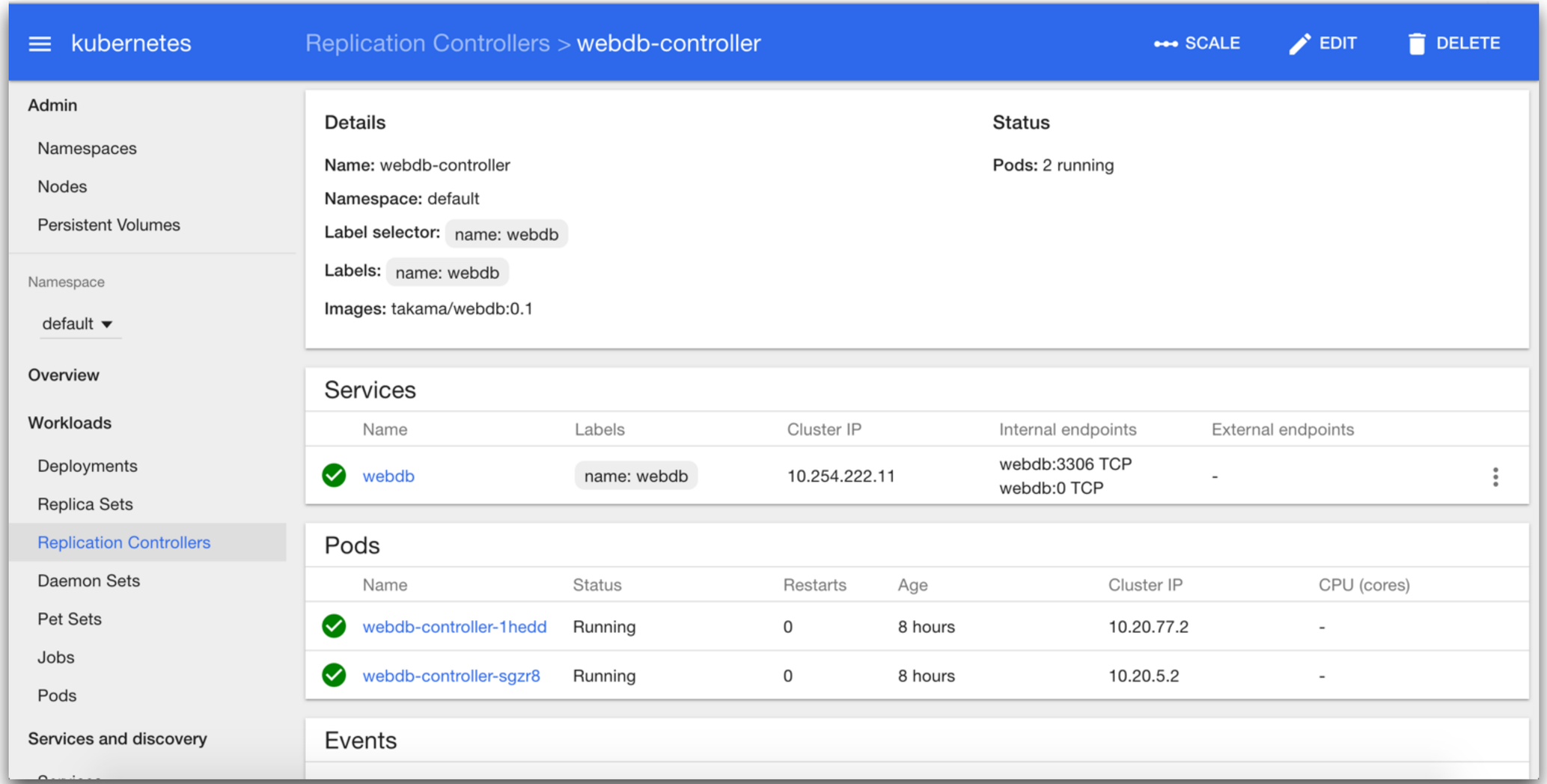
```
# Clone k8sdemo repository
$ git clone https://github.com/takama/k8sdemo.git

# Setup ansible (DevOps automation/deployment tools)
$ yum install ansible

# Go to working directory
$ cd k8sdemo/ansible

# Setup environment
$ vim inventory/cluster
[master]
k8s-master.your-domain

[node]
k8s-node-01.your-domain
k8s-node-02.your-domain

[datastore:children]
master
```

```
# Run ansible playbook
$ ansible-playbook playbooks/cluster/setup.yml

PLAY [datastore] ***************************************************************

TASK [setup] *******************************************************************
ok: [k8s-master.openprovider.nl]

TASK [ping] ********************************************************************
ok: [k8s-master.openprovider.nl]

PLAY [master] ******************************************************************

TASK [setup] *******************************************************************
ok: [k8s-master.openprovider.nl]

TASK [ping] ********************************************************************
ok: [k8s-master.openprovider.nl]

PLAY [node] ********************************************************************

TASK [setup] *******************************************************************
ok: [k8s-node-01.openprovider.nl]
ok: [k8s-node-02.openprovider.nl]

TASK [ping] ********************************************************************
ok: [k8s-node-01.openprovider.nl]
ok: [k8s-node-02.openprovider.nl]
```

Kubernetes Cluster

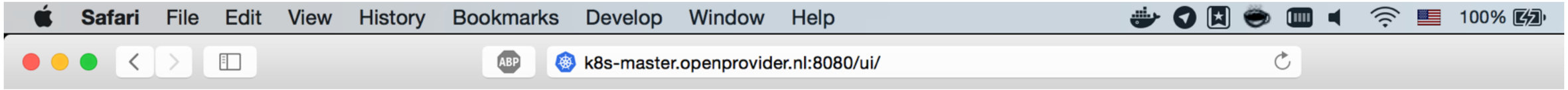


Kubernetes Dashboard

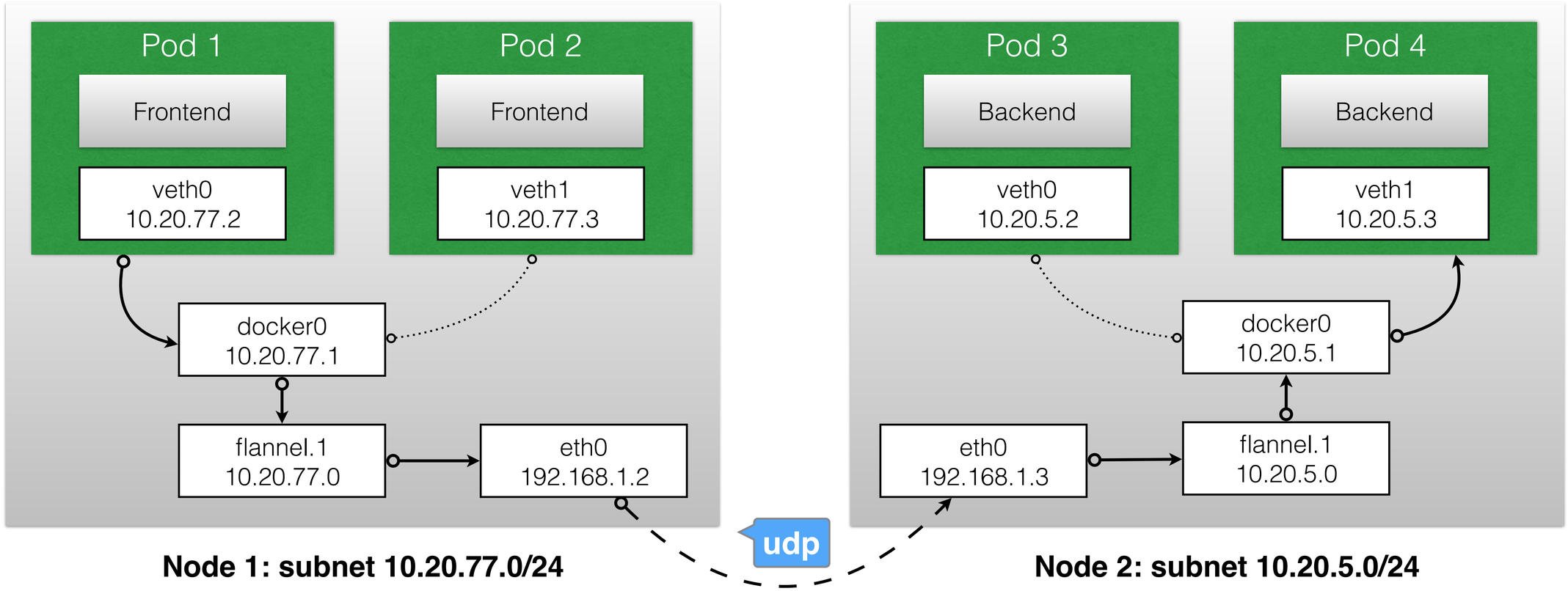


Setup Kubernetes Dashboard

```
# Register kybe-system namespace
$ kubectl create -f /etc/kubernetes/addons/kube-system-namespace.json

# Setup dashboard
$ kubectl create -f /etc/kubernetes/addons/dashboard.yaml

# Check installed dashboard
$ kubectl get svc --namespace=kube-system

NAME                   CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE
kube-dns               10.254.0.10      <none>        53/UDP,53/TCP   2d
kubernetes-dashboard   10.254.154.254   nodes         80/TCP          9d

# Open from browser
# http://master-host-name:8080/ui/
```



Overlay Network



Ping from Pod 1 to Pod 4: 10.2.77.2 -> 10.20.5.3

Setup Demo Manifests

```
# Run ansible playbook
$ ansible-playbook playbooks/cluster/demo.yml

PLAY [demo] ********************************************************************

TASK [setup] *******************************************************************
ok: [k8s-master.openprovider.nl]

TASK [ping] ********************************************************************
ok: [k8s-master.openprovider.nl]

PLAY [demo] ********************************************************************

TASK [setup] *******************************************************************
ok: [k8s-master.openprovider.nl]

TASK [set_fact] ****************************************************************
ok: [k8s-master.openprovider.nl]

TASK [demo : Check demo directory] *********************************************
ok: [k8s-master.openprovider.nl]

TASK [demo : Setup demo service] ***********************************************
changed: [k8s-master.openprovider.nl] => (item=backend-svc.yaml)
changed: [k8s-master.openprovider.nl] => (item=frontend-svc.yaml)
changed: [k8s-master.openprovider.nl] => (item=backend-rc.yaml)
changed: [k8s-master.openprovider.nl] => (item=frontend-rc.yaml)
changed: [k8s-master.openprovider.nl] => (item=busybox.yaml)

PLAY RECAP *********************************************************************
k8s-master.openprovider.nl : ok=6    changed=1    unreachable=0    failed=0
```

It copies manifests files into ~/services/

```
-rw-r--r--.  1 user  161 Sep  8 06:54 backend-svc.yaml
-rw-r--r--.  1 user  194 Sep  8 06:54 frontend-svc.yaml
-rw-r--r--.  1 user  342 Sep  8 06:54 backend-rc.yaml
-rw-r--r--.  1 user  356 Sep  8 06:55 frontend-rc.yaml
-rw-r--r--.  1 user  229 Sep  8 06:57 busybox.yaml
```

Overlay Network

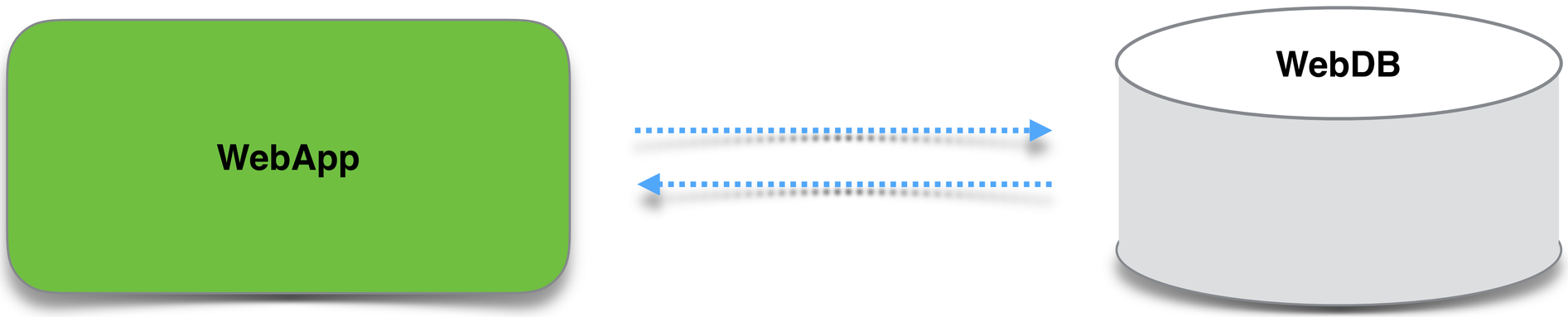
```
# Prepare Pod manifest (optional if demo scripts installed)
$ cat > ~/services/busybox.yaml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: default
spec:
  containers:
  - image: busybox
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
    name: busybox
  restartPolicy: Always
EOF

# Create Pod
$ kubectl create -f ~/services/busybox.yaml
```

```
# ping Pod4 from Pod1 ( check corresponded IPs)
$ kubectl exec busybox -- ping 10.20.5.3 -s 1424
PING 10.20.5.3 (10.20.5.3): 1424 data bytes
1432 bytes from 10.20.5.3: seq=0 ttl=62 time=1.882 ms
1432 bytes from 10.20.5.3: seq=1 ttl=62 time=0.690 ms
1432 bytes from 10.20.5.3: seq=2 ttl=62 time=0.665 ms
1432 bytes from 10.20.5.3: seq=3 ttl=62 time=0.802 ms
1432 bytes from 10.20.5.3: seq=4 ttl=62 time=0.884 ms

# Show packets
$ tcpdump -i flannel.1 -nnA src 10.20.77.0
tcpdump: verbose output suppressed, use -v or -vv for
full protocol decode
listening on flannel.1, link-type EN10MB (Ethernet),
capture size 65535 bytes
10:53:52.184108 IP 10.20.77.0 > 10.20.5.3: ICMP echo
request, id 1536, seq 47, length 1424
E..... .?.&m
.M.
.........../M..g.....................................
```

Demo Services

Frontend

Backend



Go service
based on official Alpine image

MariaDB
based on official image

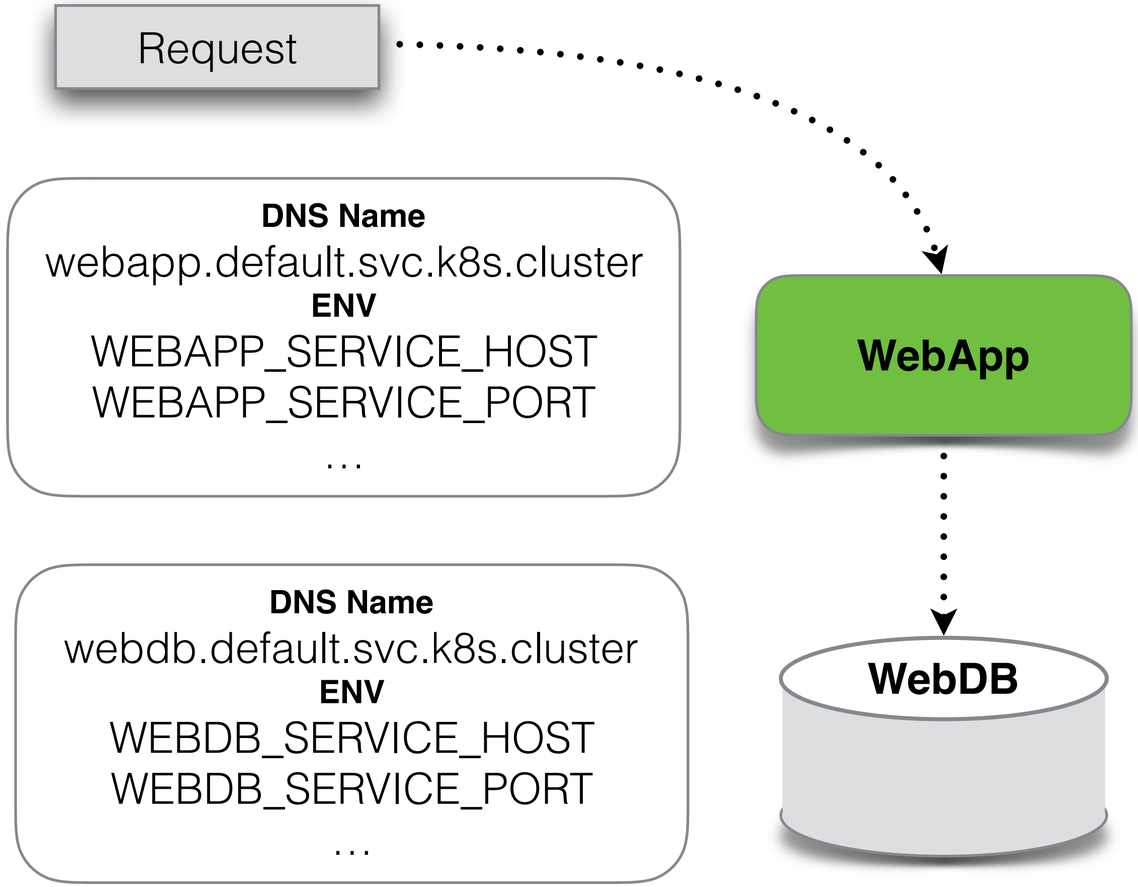
Prepare Demo Services

```
# Check component statuses
$ kubectl get cs

NAME                 STATUS    MESSAGE              ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health": "true"}

# Register backend service
$ kubectl create -f ~/services/backend-svc.yaml

# Register frontend service
$ kubectl create -f ~/services/frontend-svc.yaml

# Register backend Replication Controller
$ kubectl create -f ~/services/backend-rc.yaml

# Register frontend Replication Controller
$ kubectl create -f ~/services/frontend-rc.yaml
```

```
# Check services
$ kubectl get svc
NAME         CLUSTER-IP       EXTERNAL-IP     PORT(S)    AGE
kubernetes   10.254.0.1       <none>          443/TCP    11d
webapp       10.254.101.185   192.168.1.1     3000/TCP   1m
webdb        10.254.105.176   <none>          3306/TCP   1m

# Check replication controllers
$ kubectl get rc
NAME               DESIRED   CURRENT   AGE
webapp-controller  2         2         1m
webdb-controller   2         2         1m


# Check pods
$ kubectl get pods
NAME                                                READY     STATUS    RESTARTS   AGE
kube-apiserver-k8s-master.openprovider.nl           1/1       Running   0          20h
kube-controller-manager-k8s-master.openprovider.nl  1/1       Running   0          20h
kube-scheduler-k8s-master.openprovider.nl           1/1       Running   0          20h
webapp-controller-isule                             1/1       Running   0          1m
webapp-controller-j0emk                             1/1       Running   0          1m
webdb-controller-hic12                              1/1       Running   0          1m
webdb-controller-o33zd                              1/1       Running   0          1m
```

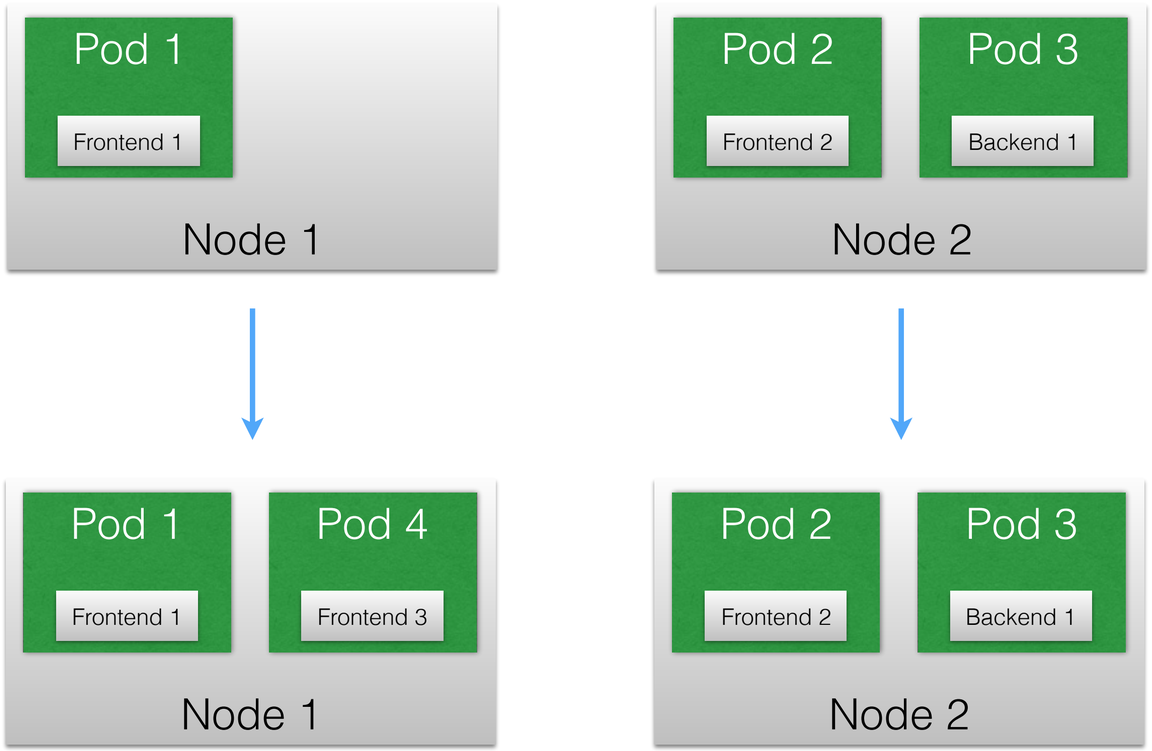
Use case 1: Relations

```
# Using Kube DNS for names resolving/discovering
#
# Template:
# < service name >< namespace >.svc.< cluster name >
#
# Lookup frontend service name from kibedns
$ kubectl exec busybox -- nslookup webapp

Server:    10.254.0.10
Address 1: 10.254.0.10 kube-dns.kube-system.svc.k8s.cluster

Name:      webapp
Address 1: 10.254.159.24 webapp.default.svc.k8s.cluster


# Get backend container environments
$ docker inspect 2195619e1fdc | jq '.[] | .Config.Env'
[
    “WEBAPP_SERVICE_HOST=10.254.159.24",
    "WEBDB_SERVICE_HOST=10.254.59.131",
    "WEBDB_SERVICE_PORT=3306",
    “WEBAPP_SERVICE_PORT=3000",
    . . .
    "GOSU_VERSION=1.7"
]
```



Use case 2: Scale

```
# Scale replicas from 2 to 3 for specified controller
$ kubectl scale --replicas=3 rc webapp-controller

replicationcontroller "webapp-controller" scaled

# Scale replicas from 3 to 2 for specified controller
$ kubectl scale --replicas=2 rc webapp-controller

replicationcontroller "webapp-controller" scaled

# Scale replicas from 2 to 1 for specified controller
$ kubectl scale --replicas=1 rc webapp-controller

replicationcontroller "webapp-controller" scaled


# Scale replicas from 1 to 2 for specified controller
$ kubectl scale --replicas=2 rc webapp-controller

replicationcontroller "webapp-controller" scaled
```



Scale Frontend: 2 -> 3

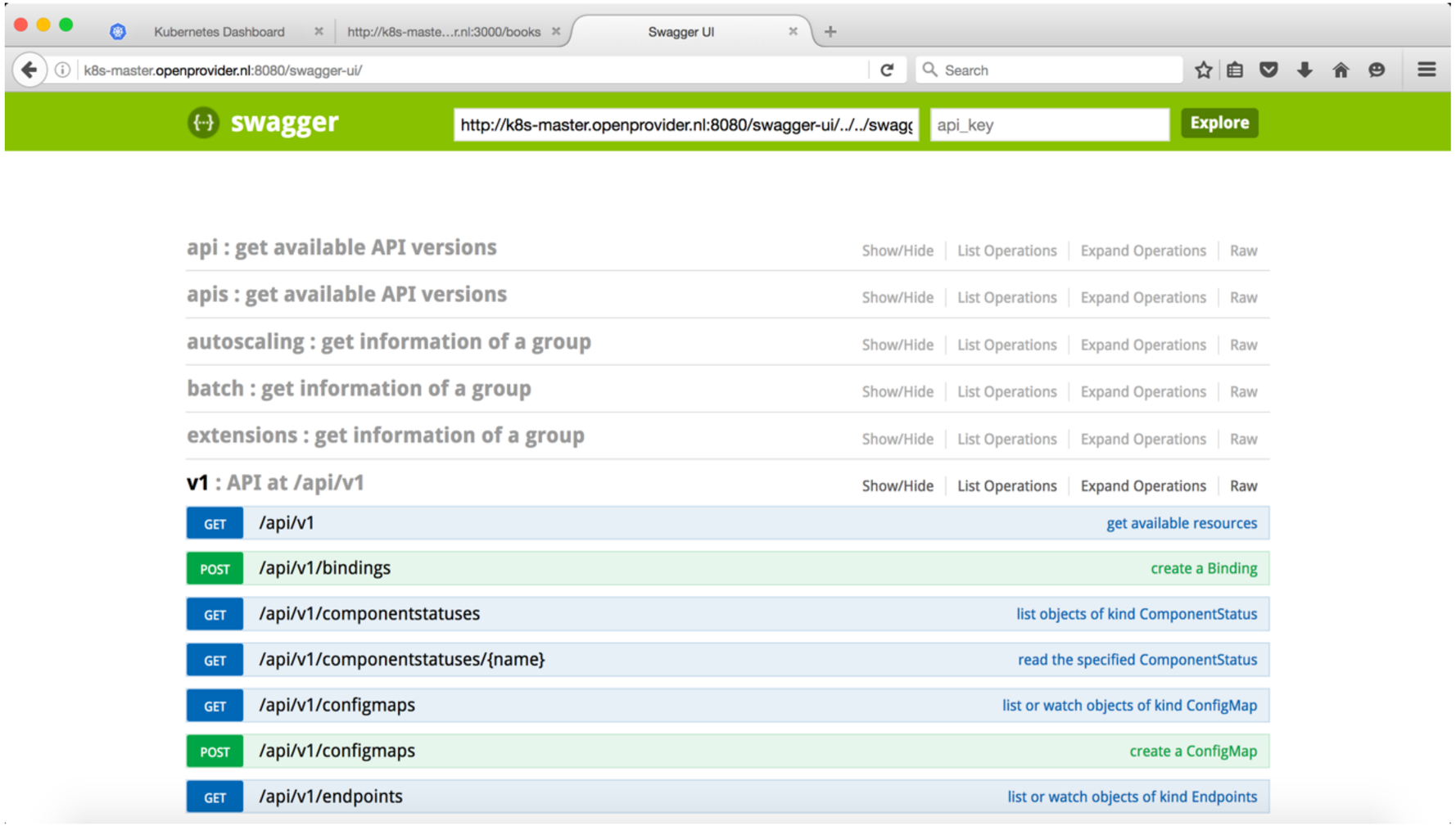
Use case 3: Rolling Updates

```
# Rolling update to new image version
$ kubectl rolling-update webapp-controller –image=takama/webapp:0.2

Created webapp-controller-ed8873daf1dc43420f537d3585fdf337
Scaling up webapp-controller-ed8873daf1dc43420f537d3585fdf337 from 0 to 2, scaling down webapp-controller from 2 to 0 (keep 2 pods available, don't exceed 3 pods)
Scaling webapp-controller-ed8873daf1dc43420f537d3585fdf337 up to 1
Scaling webapp-controller down to 1
Scaling webapp-controller-ed8873daf1dc43420f537d3585fdf337 up to 2
Scaling webapp-controller down to 0
Update succeeded. Deleting old controller: webapp-controller
Renaming webapp-controller-ed8873daf1dc43420f537d3585fdf337 to webapp-controller
replicationcontroller "webapp-controller" rolling updated

# Rolling update to old image version
$ kubectl rolling-update webapp-controller –image=takama/webapp:0.1

Created webapp-controller-544345bd9584544cb36bc884bedb5829
Scaling up webapp-controller-544345bd9584544cb36bc884bedb5829 from 0 to 2, scaling down webapp-controller from 2 to 0 (keep 2 pods available, don't exceed 3 pods)
Scaling webapp-controller-544345bd9584544cb36bc884bedb5829 up to 1
Scaling webapp-controller down to 1
Scaling webapp-controller-544345bd9584544cb36bc884bedb5829 up to 2
Scaling webapp-controller down to 0
Update succeeded. Deleting old controller: webapp-controller
Renaming webapp-controller-544345bd9584544cb36bc884bedb5829 to webapp-controller
replicationcontroller "webapp-controller" rolling updated
```

Use case 3: Rolling Updates

```
# Rolling update to new image version with changing controller name
$ kubectl rolling-update webapp-controller webapp-controller-v2 –image=takama/webapp:0.2

# Rolling update to old image version with changing controller name
$ kubectl rolling-update webapp-controller-v2 webapp-controller –image=takama/webapp:0.1

Created webapp-controller
Scaling up webapp-controller from 0 to 2, scaling down webapp-controller-v2 from 2 to 0 (keep 2 pods available, don't exceed 3 pods)
Scaling webapp-controller up to 1
Scaling webapp-controller-v2 down to 1
^C

# Rollback to previous state if something get wrong
$ kubectl rolling-update webapp-controller-v2 webapp-controller --rollback

Setting "webapp-controller-v2" replicas to 2
Continuing update with existing controller webapp-controller-v2.
Scaling up webapp-controller-v2 from 1 to 2, scaling down webapp-controller from 1 to 0 (keep 2 pods available, don't exceed 3 pods)
Scaling webapp-controller-v2 up to 2
Scaling webapp-controller down to 0
Update succeeded. Deleting webapp-controller

# Repeat rolling update to old image version with changing controller name
$ kubectl rolling-update webapp-controller-v2 webapp-controller –image=takama/webapp:0.1
```

http://<KUBE_MASTER_IP>:8080/swagger-ui/



Thank you

- Sources: https://github.com/takama/k8sdemo
- Docs: https://access.redhat.com/documentation/en/red-hat-enterprise-linux-atomic-host/version-7/getting-started-with-containers/