

ローカルホストでの Kubernetes 環境構築手順

2017年12月

SCSK株式会社

R&Dセンター 技術戦略部

※本文中の会社名、商品名は、各社の商標及び登録商標です。

はじめに

コンテナ管理では今や kubernetes がデファクトになっていますが、ここでは kubernetes 環境を簡単に構築する手順を紹介します。

kubernetes、docker の基本事項の説明は特に行いません。

kubernetes は「船の操舵手」を意味するギリシャ語と言う事ですが単語が長いので以下の文章中ではk8s (kとsの間に8文字ある為)と言う略称を使います。

はじめに

本書作成時点の使用環境

Ubuntu: 16.04

Kubernetes: 1.8.4

Docker: 1.12.1

他の環境、バージョンでは動きが異なるかもしれません。

K8sサービスの利用

代表的なk8sサービス

➤ Google GKE(Google Kubernetes Engine)

0~6ノード: 無料

6ノード~: \$0.15/h

別途ノード用の VM として GCE料金は必要

(GCE は Google Compute Engine)

➤ Azure AKS(Azure Container Service)

AKS 料金無料。別途 VM 料金は必要。

➤ AWS EKS(Amazon Elastic Container Service for Kubernetes)

2017/11/30から開始。現在 preview 中。

ローカルで構築する意味

- 使用料が要らない

と言うのがやはり一番の理由でしょう。

他にも以下の様な点があるかも知れません。

- 事前に試したい
- 安全性
- デバッグがし易い?

等々...

ローカル環境での構築

主なツール

- Minikube

k8s 提供のツール。K8s がインストールされた VM イメージを使用。

- Kubeadm

k8s 提供のツール。VM ではなくホスト上に環境構築。

- OpenShift

RedHat 提供のツール。ansible を使ってセットアップ。

- kubernetes-vagrant-coreos-cluster

Pires という人の OSS。Vagrant を使ってk8s環境を展開。

- Kops

k8s 提供のツール。AWS 上に k8s 環境を展開。

ローカル環境での構築

以降では下記の事項について説明します。

(但しまだ実用には堪えない段階のようです。)

- Minikube
 - 1ノード構成
 - ノードを VM イメージで展開。
 - 実環境への影響が少ない。
- Kubeadm
 - 複数ノード構成
 - 実環境に展開
 - ノードの追加も簡単
- プライベートレジストリ
 - dockerイメージのローカル管理

Minikubeでの構築

リポジトリは[こちら](#)

下記のVM環境に対応

VirtualBox

VMware

KVM

Hyperkit

xhyve

HyperV

VM 毎にドライバーが用意されています。

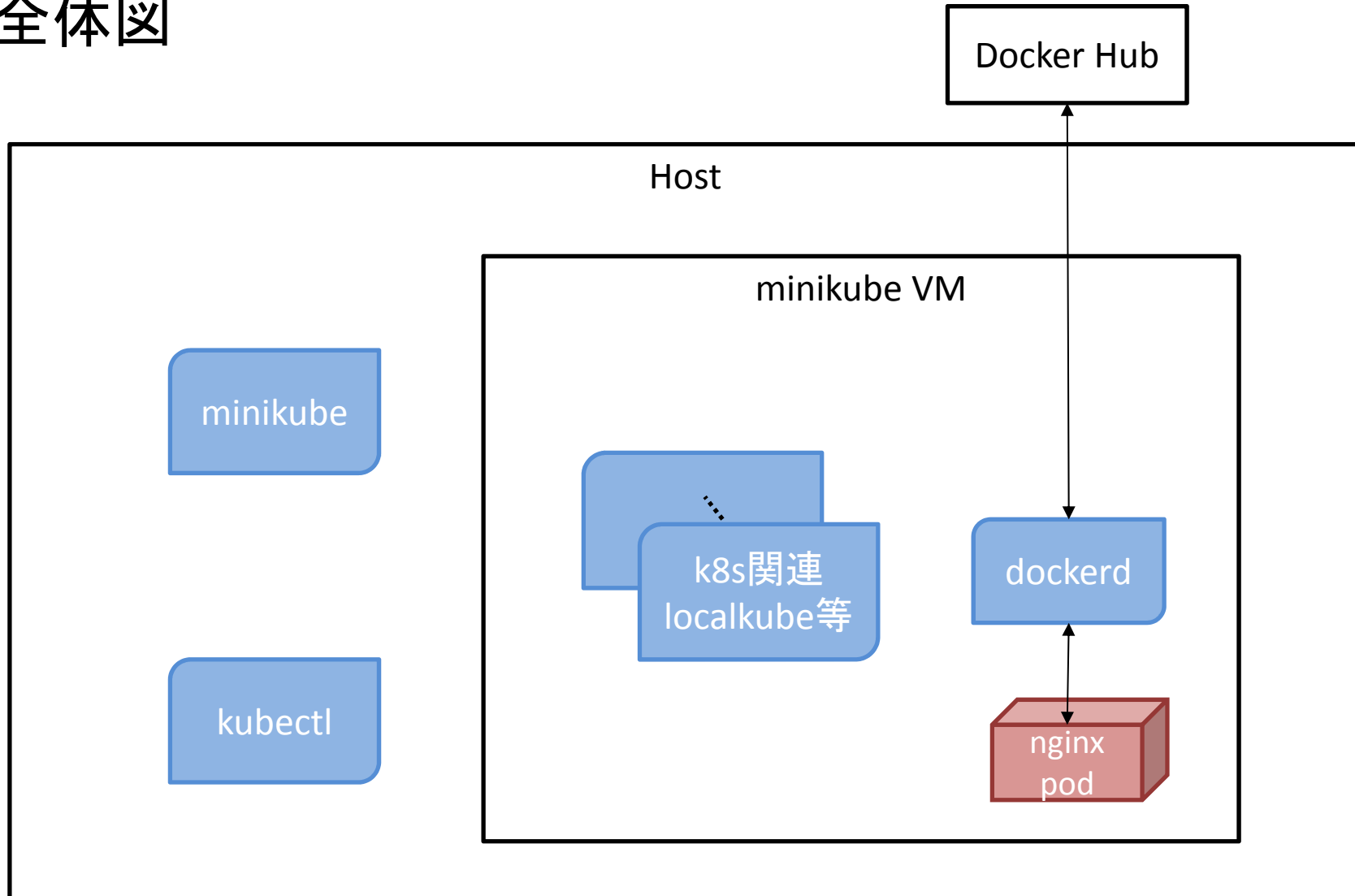
(但し VirtualBox, Vmware のドライバーは組み込み済みなので本体だけで使用可能)

ここでは linux でお馴染みの kvm を使用します。

次に概略図を示します。

Minikubeでの構築

全体図



Minikubeでの構築

➤ 準備作業

✓ 必要パッケージのインストール

```
# apt update
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
# apt update
# apt install libvirt-bin qemu-kvm kubectl docker.io
```

✓ Minikube 本体とドライバー

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64 && chmod +x minikube && sudo cp minikube /usr/local/bin/
$ curl -LO https://storage.googleapis.com/minikube/releases/latest/docker-machine-driver-
kvm2 && chmod +x docker-machine-driver-kvm2 && sudo cp docker-machine-driver-kvm2
/usr/local/bin/
$
```

Minikubeでの構築

➤ 起動

```
$ minikube status
minikube:
cluster:
kubect!:
$ minikube start --vm-driver kvm2
Starting local Kubernetes v1.8.0 cluster...
Starting VM...
Downloading Minikube ISO

140.01 MB / 140.01 MB [=====] 100.00% 0s
Getting VM IP address...
Moving files into cluster...
Downloading localkube binary

65 B / 65 B [=====] 100.00% 0sSetting up
certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubect! is now configured to use the cluster.
Loading cached images from config file.
$
```


Minikubeでの構築

➤ Docker コマンドを有効に

```
$ sudo usermod -a -G docker $(whoami)
$ newgrp docker
$ eval $(minikube docker-env)
$ docker ps
WARN[0000] Unable to use system certificate pool: requires building with go 1.7 or later
CONTAINER ID   IMAGE                                COMMAND
CREATED        STATUS      PORTS          NAMES
b67bbb9129b8   fed89e8b4248                                "/sidecar
--v=2 --..." 13 seconds ago   Up 8 seconds   k8s_sidecar_kube-dns-86f6f55dd5-
nk4lq_kube-system_57e59ca6-d8e8-11e7-b1c3-5c99775fd503_0
...
$ minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.39.140:2376"
export DOCKER_CERT_PATH="/home/ubuntu/.minikube/certs"
export DOCKER_API_VERSION="1.23"
# Run this command to configure your shell:
# eval $(minikube docker-env)
$
```

Minikubeでの構築

➤ Pod 作成

```
$ cat sv-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    run: nginx1
spec:
  containers:
  - name: nginx-container
    image: nginx
    ports:
    - containerPort: 80
$ kubectl apply -f sv-nginx.yaml
pod "nginx" created
$ kubectl get pods --all-namespaces
NAMESPACE  NAME                                READY  STATUS             RESTARTS  AGE
default    nginx                                0/1    ContainerCreating  0         13s
kube-system kube-addon-manager-minikube         1/1    Running            0         4m
kube-system kube-dns-86f6f55dd5-djt9z           3/3    Running            0         4m
kube-system kubernetes-dashboard-7bkwb        1/1    Running            2         4m
kube-system storage-provisioner      1/1    Running            0         4m
$
```

Minikubeでの構築

Pod へのアクセス

```
$ kubectl expose pod nginx --type=NodePort
$ kubectl get services
NAME      TYPE      CLUSTER-IP  EXTERNAL-IP  PORT(S)    AGE
kubernetes ClusterIP  10.96.0.1    <none>       443/TCP    15m
nginx     NodePort  10.98.54.159 <none>       80:31625/TCP 13s
$ curl $(minikube service nginx --url)
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

Minikubeでの構築

Pod へのアクセス

```
<p>For online documentation and support please refer to  
<a href="http://nginx.org/">nginx.org</a>.<br/>  
Commercial support is available at  
<a href="http://nginx.com/">nginx.com</a>.</p>
```

```
<p><em>Thank you for using nginx.</em></p>  
</body>  
</html>  
$ minikube service nginx --url  
http://192.168.39.140:31625  
$
```


Minikubeでの構築

➤ kubernetes dashboard の表示

```
$ minikube dashboard  
Opening kubernetes dashboard in default browser...  
$
```

上記で表示されない場合は直接 URL を指定

```
$ firefox $(minikube dashboard -url)  
$ minikube dashboard --url  
http://192.168.39.139:30000  
$
```

Kubernetes dashboard

The screenshot shows the Kubernetes Dashboard Overview page. The browser window title is "Overview - Kubernetes Dashboard - Mozilla Firefox (ホスト: minikube1)". The address bar shows the URL "192.168.39.140:30000/#/overview?namespace=default". The dashboard header includes the Kubernetes logo, a search bar, and a "+ CREATE" button. The left sidebar contains a navigation menu with categories: Cluster, Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Namespace (set to "default"), Overview (selected), Workloads, Discovery and Load Balancing, Config and Storage, and Settings. The main content area is titled "Workloads" and contains several sections: "Workloads Statuses" with a green circular gauge showing "100.00%" for "Pods"; "Pods" table with one entry: "nginx" on node "minikube" in "Running" status; "Discovery and Load Balancing" section with a "Services" table containing "nginx" and "kubernetes"; and "Config and Storage" section with a "Secrets" table containing "default-token-hz5fk".

Cluster

- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace

default

Overview

Workloads

- Cluster Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Discovery and Load Balancing

- Ingresses
- Services

Config and Storage

- Config Maps
- Persistent Volume Claims
- Secrets

Settings

About

Workloads

Workloads Statuses

100.00%

Pods

Name	Node	Status	Restarts	Age
nginx	minikube	Running	0	an hour

Discovery and Load Balancing

Services

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
nginx	run: nginx1	10.98.54.159	nginx:80 TCP nginx:31625 TCP	-	50 minutes
kubernetes	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	an hour

Config and Storage

Secrets

Name	Type	Age
default-token-hz5fk	kubernetes.io/service-account-token	an hour

Minikubeでの構築

➤ VM へのlogin

必要なら docker / tcuser で login可能

```
$ ssh docker@192.168.39.140
Warning: Permanently added '192.168.39.140' (ECDSA) to the list of known hosts.
docker@192.168.39.140's password:

      _ _ _ _ _      _ _ _ _ _      _ _ _ _ _      _ _ _ _ _
     /' _ _ _ _ _ `¥| |/' _ _ _ _ _ `¥| | | , < ( ) ( ) |' _ _ _ _ _ `¥ /' _ _ _ _ _
    | ( ) ( ) | | | | ( ) | | | | ¥`¥ | ( ) | | | ) ( _ _ /
   ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) `¥ _ _ /' ( , _ _ / `¥ _ _ )

$ ps -e | grep kube
2589 ?      00:29:38 localkube
3165 ?      00:00:08 kube-addons.sh
3326 ?      00:00:13 kube-dns
$
```

通常の k8s バイナリの kubelet ではなく localkube が動いているのが分かります。

Minikubeでの構築

➤ Minikube 停止(環境保持)

```
$ minikube stop  
Stopping local Kubernetes cluster...  
Machine stopped.
```

```
$ virsh list
```

```
Id   Name                               State  
-----
```

```
$ minikube status  
minikube: Stopped  
cluster:  
kubectl:  
$
```

➤ Minikube 停止(環境破棄)

```
$ minikube delete  
Deleting local Kubernetes cluster...  
Machine deleted.
```

```
$ minikube status  
minikube:  
cluster:  
kubectl:  
$
```

Minikubeでの構築

実は直接ホスト上でも動作する

➤ 準備作業

✓ 必要パッケージのインストール

```
# apt update
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
# apt update
# apt install kubectl docker.io
$ sudo usermod -a -G docker $(whoami)
$ newgrp docker
```

✓ Minikube

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64 && chmod +x minikube && sudo cp minikube /usr/local/bin/
$
```

Minikubeでの構築

➤ 起動

```
$ export CHANGE_MINIKUBE_NONE_USER=true
$ sudo -E minikube start --vm-driver=none
Starting local Kubernetes v1.8.0 cluster...
Starting VM...
Getting VM IP address...
Moving files into cluster...
Downloading localkube binary

65 B / 65 B [=====] 100.00% 0s
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubectl is now configured to use the cluster.
=====
WARNING: IT IS RECOMMENDED NOT TO RUN THE NONE DRIVER ON PERSONAL WORKSTATIONS
The 'none' driver will run an insecure kubernetes apiserver as root that may leave the host vulnerable to
CSRF attacks

Loading cached images from config file.
$ minikube status
minikube: Running
cluster: Running
kubectl: Correctly Configured: pointing to minikube-vm at 127.0.0.1
$
```

Minikubeでの構築

環境変数

```
export CHANGE_MINIKUBE_NONE_USER=true
```

を指定する事で Minikube がホスト上で k8s の操作を行う為の設定を自動的に行います。

これは下記と同等の設定

```
$ sudo mv /root/.kube $HOME/.kube
$ sudo chown -R $USER $HOME/.kube
$ sudo chgrp -R $USER $HOME/.kube

$ sudo mv /root/.minikube $HOME/.minikube
$ sudo chown -R $USER $HOME/.minikube
$ sudo chgrp -R $USER $HOME/.minikube
```

他の操作は VM 使用の場合と同じ為省略します。

Minikubeでの構築

➤ K8s のバージョン指定

Minikube では複数の k8s バージョンをサポートします。
使用可能なバージョンは下記で確認できます。

```
$ minikube get-k8s-versions
```

```
The following Kubernetes versions are available when using the localkube bootstrapper:
```

- v1.8.0
- v1.7.5
- v1.7.4
- v1.7.3
- v1.7.2
- v1.7.0

```
...
```


Minikubeでの構築

--kubernetes-version オプションにより指定します。

```
$ sudo -E minikube start --kubernetes-version v1.7.3 --vm-driver=none
Starting local Kubernetes v1.7.3 cluster...
Starting VM...
Getting VM IP address...
Moving files into cluster...
Downloading localkube binary
 138.65 MB / 138.65 MB [=====] 100.00% 0s
  65 B / 65 B [=====] 100.00% 0s
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubectl is now configured to use the cluster.
=====
WARNING: IT IS RECOMMENDED NOT TO RUN THE NONE DRIVER ON PERSONAL
WORKSTATIONS
    The 'none' driver will run an insecure kubernetes apiserver as root that may leave the host
vulnerable to CSRF attacks

Loading cached images from config file.
```

Minikubeでの構築

➤ docker 以外のコンテナ実装

-container-runtime オプションで指定する事でdocker
以外のランタイムも使用可能

今の所 rkt ぐらい

```
$ minikube start --container-runtime=rkt --iso-  
url=https://github.com/coreos/minikube-  
iso/releases/download/v0.0.5/minikube-v0.0.5.iso
```

Kubeadmでの構築

リポジトリは[こちら](#)

本ツールは現状ベータ版で、本番では使わない様にと
言う但し書きがヘルプや起動時に表示されます。

Docker は 1.12 推奨。

一応1.11, 1.13, 17.03 でも動作。17.06以降は未確認。

```
$ kubeadm --help
kubeadm: easily bootstrap a secure Kubernetes cluster.
```

```
| KUBEDM IS BETA, DO NOT USE IT FOR PRODUCTION CLUSTERS! |
|
| But, please try it out! Give us feedback at:           |
| https://github.com/kubernetes/kubeadm/issues          |
| and at-mention @kubernetes/sig-cluster-lifecycle-bugs |
| or @kubernetes/sig-cluster-lifecycle-feature-requests |
```

...

Kubeadmでの構築

メモ:

Docker のリリース番号は2017/3から変わりました。

1.13.1 の次は 17.03.0-ceになります。形式は下記。

yy.mm.revision-ce|ee

ce: Community Edition

ee: Enterprise Edition

edgeリリースとしてsecurity fix、bug fixされたものが毎月リリース。また四半期毎にstableリリース。詳細は[こちら](#)を参照。

Kubeadmでの構築

ここでは master と worker の2ノード構成のケースを示します。

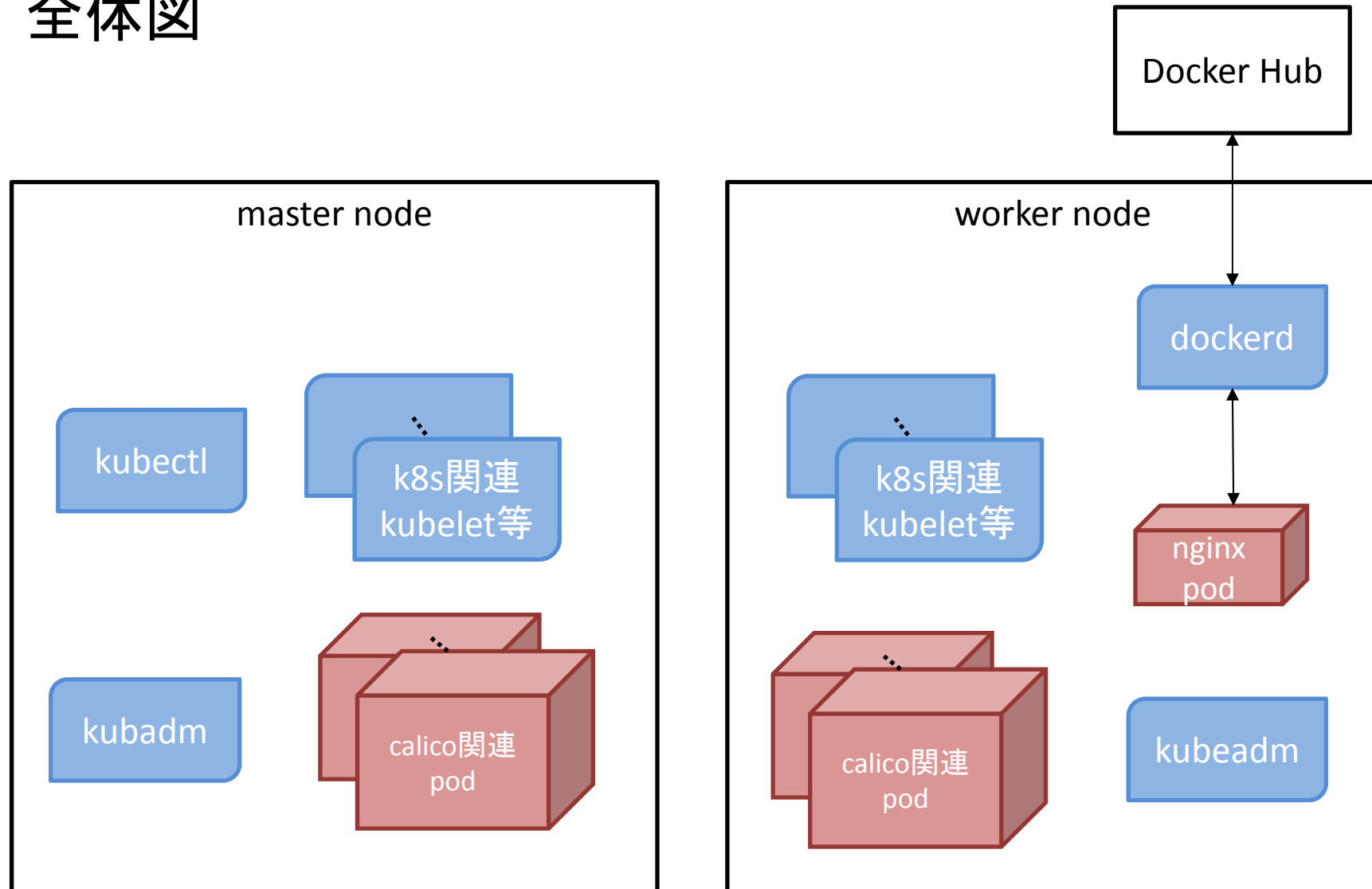
その前に

k8s は 1.8以降 swap が有効な場合動作しない。
swap 無しのホストを利用しても良いが、ここでは swap が有効のまま起動する手順を示します。

次に概略図を示します。

Kubeadmでの構築

全体図



Kubeadmでの構築

master ノードの構築

➤ 準備作業

必要パッケージのインストール。

```
# apt update
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
# apt update
# apt install kubeadm kubectl kubelet docker.io
$ sudo usermod -a -G docker $(whoami)
$ newgrp docker
```

Kubeadmでの構築

kubelet に `--fail-swap-on=false` オプション指定

ubuntuではsystemdの起動ファイル修正。

`/etc/systemd/system/kubelet.service.d/10-kubeadm.conf`

```
$ sudo diff -u 10-kubeadm.conf /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
--- 10-kubeadm.conf    2017-11-21 04:19:27.000000000 +0900
+++ /etc/systemd/system/kubelet.service.d/10-kubeadm.conf    2017-11-29 10:02:24.476308855
+0900
@@ -1,5 +1,5 @@
 [Service]
-Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-
kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
+Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-
kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --fail-swap-on=false"
 Environment="KUBELET_SYSTEM_PODS_ARGS=--pod-manifest-path=/etc/kubernetes/manifests --
allow-privileged=true"
 Environment="KUBELET_NETWORK_ARGS=--network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-
bin-dir=/opt/cni/bin"
 Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster.local"
$
```


Kubeadmでの構築

➤ 起動

起動オプションで `--skip-preflight-checks` 指定

```
$ sudo kubeadm init --skip-preflight-checks
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
[init] Using Kubernetes version: v1.8.4
[init] Using Authorization modes: [Node RBAC]
[preflight] Skipping pre-flight checks
[kubeadm] WARNING: starting in 1.8, tokens expire after 24 hours by default (if you require a
non-expiring token use --token-ttl 0)
[certificates] Using the existing ca certificate and key.
[certificates] Using the existing apiserver certificate and key.
[certificates] Using the existing apiserver-kubelet-client certificate and key.
[certificates] Using the existing sa key.
[certificates] Using the existing front-proxy-ca certificate and key.
[certificates] Using the existing front-proxy-client certificate and key.
...
```

Kubeadmでの構築

起動続き

...

[addons] Applied essential addon: kube-proxy

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 6ba6cd.616ef860c5d6d400 172.17.62.208:6443 --discovery-token-ca-cert-hash sha256:42cc9b39bfada3308d27005eb3eb3ef4f607472631c647143137b9b622bd4b3a
```

\$

Kubeadmでの構築

1.7 以前ではパッケージインストール後そのまま

```
$ sudo kubeadm init
```

だけで OK。

続いてガイドメッセージの通り実行します。

(ガイドメッセージはk8sのバージョンにより多少異なります)

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
$ ps -e |grep kube
 933 ?    00:11:38 kubelet
1581 ?    00:01:46 kube-scheduler
1590 ?    00:05:20 kube-controller
1668 ?    00:07:00 kube-apiserver
1928 ?    00:00:29 kube-controller
2065 ?    00:00:47 kube-proxy
$
```

Kubeadmでの構築

➤ ネットワークのインストール

ここでは calico を使用。

```
$ kubectl apply -f https://docs.projectcalico.org/v2.6/getting-started/kubernetes/installation/hosted/kubeadm/1.6/calico.yaml
$ kubectl get -o wide nodes
NAME     STATUS  ROLES    AGE   VERSION
k8s1    Ready   master   25m   v1.8.4
$ kubectl get pods --all-namespaces -o wide
NAMESPACE  NAME                                     READY  STATUS  RESTARTS  AGE
kube-system  calico-etcd-5pzch                       1/1    Running  0          1m
kube-system  calico-kube-controllers-685f8bc7fb-6rbkl 1/1    Running  0          1m
kube-system  calico-node-lpqqc                       2/2    Running  0          1m
kube-system  etcd-k8s1                               1/1    Running  0          24m
kube-system  kube-apiserver-k8s1                     1/1    Running  0          24m
kube-system  kube-controller-manager-k8s1           1/1    Running  0          24m
kube-system  kube-dns-545bc4bfd4-pz86d              0/3    Pending  0          24m
kube-system  kube-proxy-cvlhv                        1/1    Running  0          24m
kube-system  kube-scheduler-k8s1                    1/1    Running  0          24m
$
```

Kubeadmでの構築

calico 以外のネットワークについては下記参照。

<https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#pod-network>

k8s ネットワーキングについては弊社技術サイトに記事がありますのでそちらも参照ください。

<https://www.valinux.co.jp/technologylibrary/document/orchestration/kubernetes/>

Kubeadmでの構築

workerノードの構築

➤ 準備作業

必要パッケージのインストール。

```
# apt update
# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
# apt update
# apt install kubeadm docker.io
```

本来必要ないが、ここでは確認用に docker もインストールしている。

```
$ sudo usermod -a -G docker $(whoami)
$ newgrp docker
```

kubelet オプションの変更は master と同様。

Kubeadmでの構築

➤ ノードの参加

こちらは master インストール時のガイドメッセージで示された kubeadm join コマンドを各ホスト上で実行するだけ。

但し swap 有効の為 `--skip-preflight-checks` を指定します。

```
$ sudo kubeadm join --skip-preflight-checks --token 6ba6cd.616ef860c5d6d400 172.17.62.208:6443
--discovery-token-ca-cert-hash
sha256:42cc9b39bfada3308d27005eb3eb3ef4f607472631c647143137b9b622bd4b3a
...
[kubeadm] WARNING: kubeadm is in beta, please do not use it for production clusters.
[preflight] Skipping pre-flight checks
...
[bootstrap] Detected server version: v1.8.4
[bootstrap] The server supports the Certificates API (certificates.k8s.io/v1beta1)

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.
$
```

Kubeadmでの構築

➤ ノード状態の確認

```
$ kubectl get -o wide nodes
NAME     STATUS   ROLES    AGE     VERSION
k8s1    Ready   master   47m    v1.8.4
k8s2    Ready   <none>   5m     v1.8.4
$
```

追加したノード(k8s2)が Ready に。
これで終了です。

3台目以降もホスト上で“kubeadm join ...”を実行する
だけです。

Kubeadmでの構築

Minikube の時と同様 nginx を起動してアクセスしてみます。

```
$ kubectl apply -f sv-nginx.yaml
$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx-sv 1/1     Running   1           14h
$ kubectl expose pod nginx-sv --type=NodePort
$ kubectl get services
NAME     TYPE       CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes ClusterIP  10.96.0.1    <none>       443/TCP    15h
nginx-sv  NodePort   10.100.94.176 <none>       80:31494/TCP 14h
$ curl http://127.0.0.1:31494
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
```

Kubeadmでの構築

```
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>I updated this message!!!</p>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
$
```

Kubeadmでの構築

起動ノードを確認します。

```
$ kubectl describe pod nginx-sv | grep Node
Node:      k8s2/172.17.62.209
Node-Selectors: <none>
$
```

コンテナはk8s2で動いているようです。

```
$ hostname
k8s2
$ docker ps | grep nginx
6b7fe10938d1
nginx@sha256:b81f317384d7388708a498555c28a7cce778a8f291d90021208b3eba3fe74887
"nginx -g 'daemon off'" 6 hours ago      Up 6 hours          k8s_nginx-container_nginx-
sv_default_b9586736-db51-11e7-8ae7-525400a45781_1
677f23fcfa31    gcr.io/google_containers/pause-amd64:3.0
"/pause"        6 hours ago      Up 6 hours          k8s_POD_nginx-
sv_default_b9586736-db51-11e7-8ae7-525400a45781_1
$
```

確かにk8s2で動いている事が分かります。

Kubeadmでの構築

tokenを見失った場合は下記で確認できます。

```
$ sudo kubeadm token list
TOKEN          TTL    EXPIRES          USAGES          DESCRIPTION
EXTRA GROUPS
6ba6cd.616ef860c5d6d400 13h    2017-12-08T21:33:47+09:00 authentication,signing The
default bootstrap token generated by 'kubeadm init'. system:bootstrappers:kubeadm:default-
node-token
$
```

Kubeadmでの構築

1.8以降では token には期限があります。
起動メッセージでは24hと表示されています。
token が期限切れになった後にノードを追加する場合は再度tokenを生成します。

```
$ sudo kubeadm token list
TOKEN      TTL      EXPIRES  USAGES  DESCRIPTION  EXTRA GROUPS
$ sudo kubeadm token create
[kubeadm] WARNING: starting in 1.8, tokens expire after 24 hours by default (if you require a
non-expiring token use --ttl 0)
b93683.86b5bb815b885244
$ sudo kubeadm token list
TOKEN      TTL      EXPIRES  USAGES  DESCRIPTION  EXTRA
GROUPS
b93683.86b5bb815b885244  23h      2017-12-12T20:07:42+09:00  authentication,signing
<none>      system:bootstrappers:kubeadm:default-node-token
$
```

Kubeadmでの構築

また1.8以降では master の成り済まし等を防ぐための discovery用の token が追加されています。
これは master ノードの ca から生成されたものです。
(前述の --discovery-token-ca-cert-hash 指定のtoken)
忘れた場合は下記により生成できます。

```
$ openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der  
2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.* //'  
42cc9b39bfada3308d27005eb3eb3ef4f607472631c647143137b9b622bd4b3a  
$
```

プライベートレジストリの利用

コンテナイメージはレジストリで管理されています。
デフォルトでは docker コマンドは Docker Hub を参照
します。

有名なレジストリ

Docker 社 [Docker Hub](#)

CoreOS [Quay.io](#)

Amazon [ECR](#)

VMWare [Harbor](#)

Docker Hub には自作のイメージを登録する事もでき
ます。

プライベートレジストリの利用

しかし事情によっては外部イメージを利用したくない場合も

- ネット状況によってはイメージ取得に時間が掛かる
- 外部のイメージの安全性
- 公開したくない

Docker Hubにもプライベートリポジトリのサービスはありますが当然有料です。

<https://hub.docker.com/billing-plans/>

プライベートレジストリの利用

ここでは自由に登録できるプライベートレジストリを試します。

これはそれ程難しくありません。レジストリ用のイメージが用意されている為それが使えます。

レジストリに関するドキュメントは下記を参照。

<https://docs.docker.com/registry/>

プライベートレジストリの利用

レジストリ用のホストを用意しイメージを格納するディレクトリを決めます。

registry イメージをそのディレクトリにマウントして起動します。

```
$ mkdir -p /var/local/vplume
$ docker run -d -p 5000:5000 --name registry --hostname registry --restart on-failure:10 -v
/var/local/volume:/var/lib/registry registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
...
f45f535a83d2: Pull complete
Digest: sha256:0f8fe61fa337b8ef02217702ba979b47a7d68717d4628f31592ebff85915f3ba
Status: Downloaded newer image for registry:2
d61bab8de589f11fdce775e4b62a636640d2ed449573cd21a971984d7527a146
$
```

プライベートレジストリの利用

ついでにリポジトリを参照するフロントエンドも用意しましょう。docker-registry-frontend が人気の様なのでこれを使います。

<https://github.com/kwk/docker-registry-frontend>

```
$ docker run -d --name registry-fe --hostname registry-fe --restart on-failure:10 -e
ENV_DOCKER_REGISTRY_HOST=172.17.61.12 -e ENV_DOCKER_REGISTRY_PORT=5000 -p
8080:80 konradkleine/docker-registry-frontend:v2
Unable to find image 'konradkleine/docker-registry-frontend:v2' locally
v2: Pulling from konradkleine/docker-registry-frontend
...
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                                CREATED        STATUS
PORTS         NAMES
adf3cced635b   konradkleine/docker-registry-frontend:v2  "/bin/sh -c $START_SC"  8 weeks
ago          Up 2 hours      443/tcp, 0.0.0.0:8080->80/tcp  registry-fe
288a49f07d31   registry:2                             "/entrypoint.sh /etc/"  8 weeks ago   Up 2
hours        0.0.0.0:5000->5000/tcp  registry
$
```

プライベートレジストリの利用

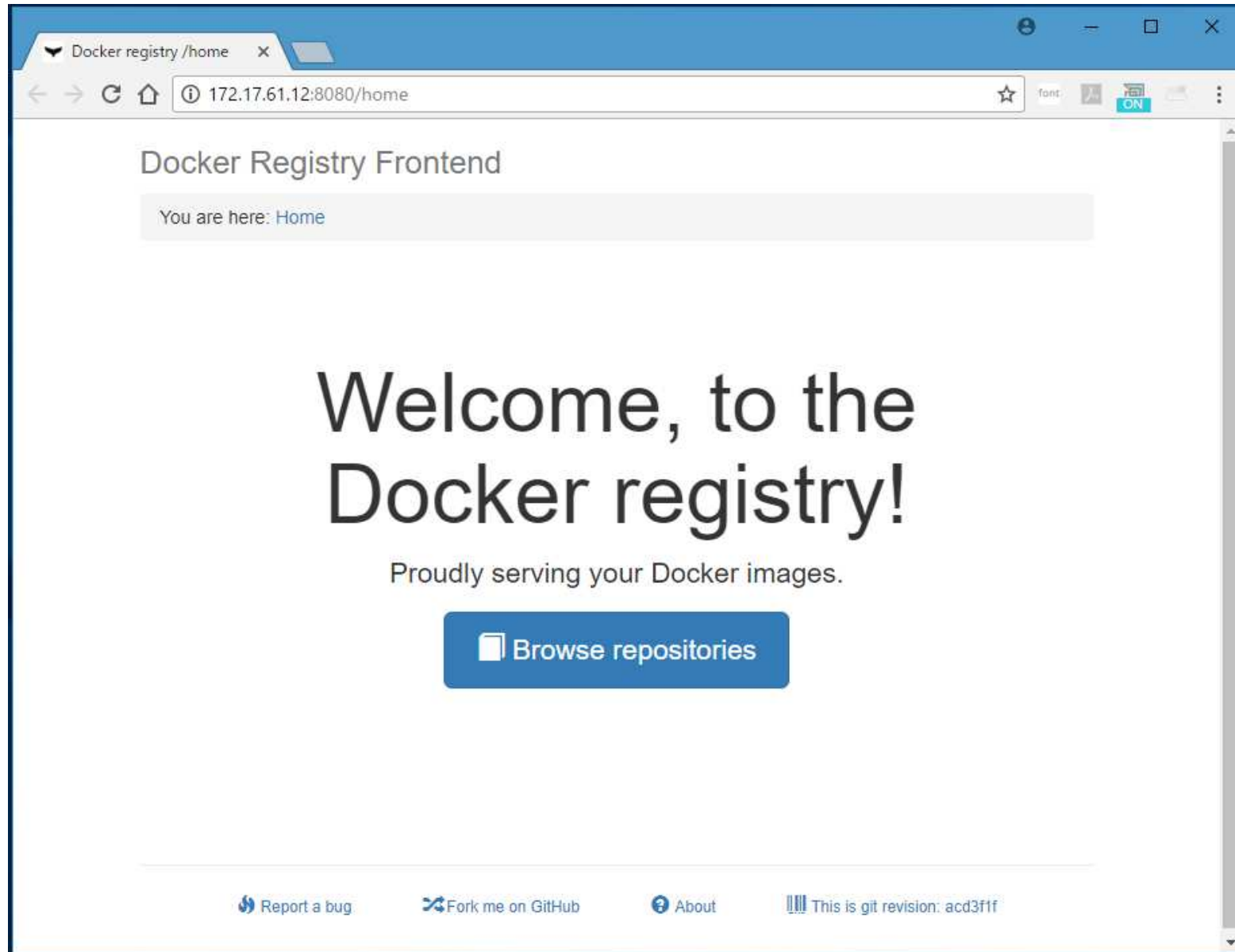
それでは先程使った nginx のイメージを登録します。
docker はTLSでアクセスしますが、証明書の登録等が面倒なのでここでは本レジストリをTLS対象外にします。

```
$ diff -u docker /etc/default/docker
--- docker    2017-12-10 01:11:45.217591145 +0900
+++ /etc/default/docker 2017-12-08 14:55:41.479762675 +0900
@@ -12,6 +12,7 @@

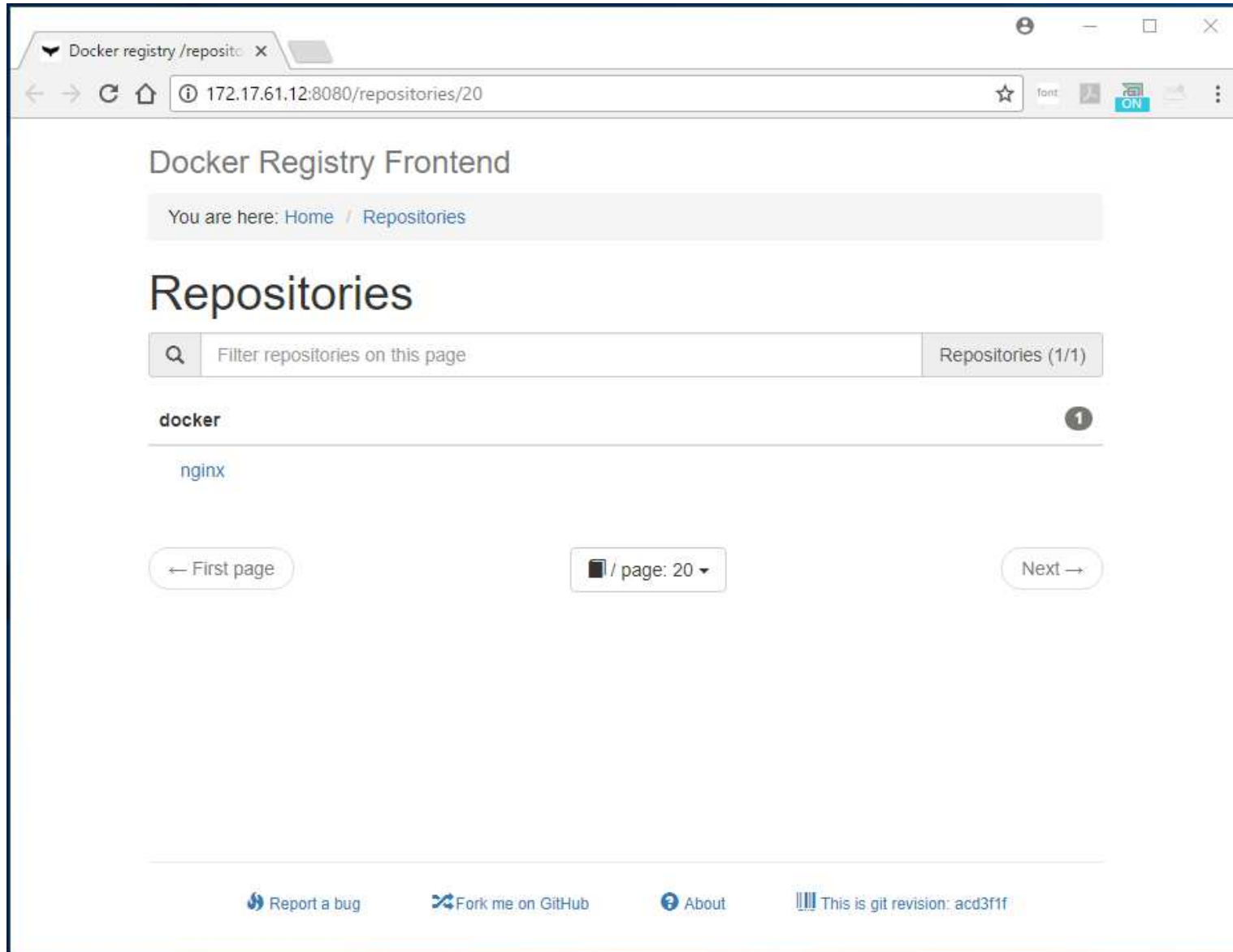
# Use DOCKER_OPTS to modify the daemon startup options.
#DOCKER_OPTS="--dns 8.8.8.8 --dns 8.8.4.4"
+DOCKER_OPTS="--insecure-registry 172.17.61.12:5000"

# If you need Docker to use an HTTP proxy, it can also be specified here.
#export http_proxy=http://127.0.0.1:3128/
$ sudo service docker restart
$ docker tag nginx:latest 172.17.61.12:5000/docker/nginx:latest
$ docker push 172.17.61.12:5000/docker/nginx
The push refers to a repository [172.17.61.12:5000/docker/nginx]
...
$
```

プライベートレジストリ情報の表示



プライベートレジストリ情報の表示



The screenshot shows a web browser window with the address bar displaying "172.17.61.12:8080/repositories/20". The page title is "Docker Registry Frontend". Below the title, a breadcrumb trail indicates "You are here: Home / Repositories". The main heading is "Repositories". A search bar contains the text "Filter repositories on this page" and a button labeled "Repositories (1/1)". Below the search bar, a list of repositories is shown, with "docker" highlighted in bold and a notification badge "1" next to it. Below the list, there are navigation buttons: "← First page", "page: 20", and "Next →". At the bottom of the page, there are links for "Report a bug", "Fork me on GitHub", "About", and "This is git revision: acd3f1f".

Docker Registry Frontend

You are here: Home / Repositories

Repositories

Filter repositories on this page Repositories (1/1)

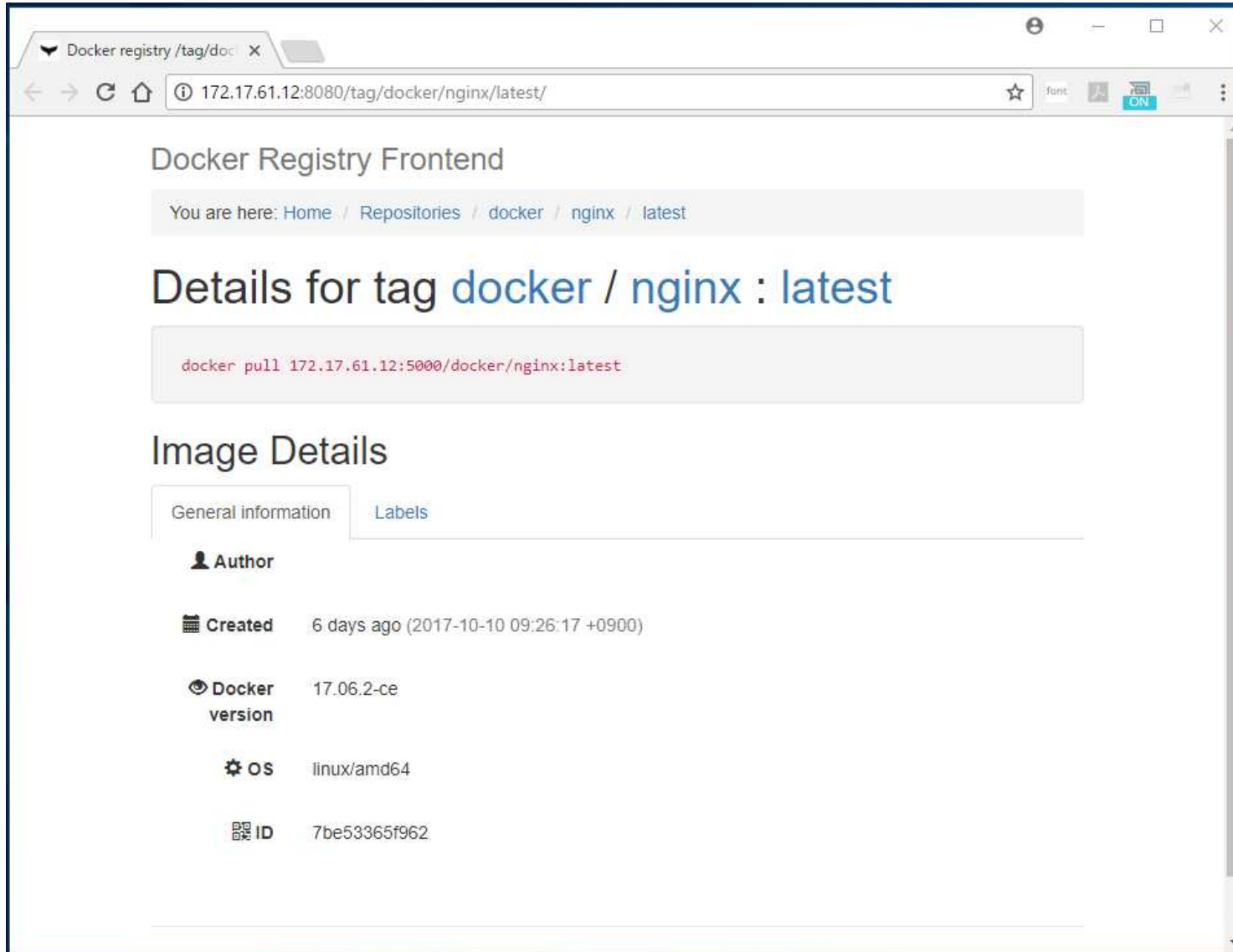
docker 1

nginx

← First page page: 20 Next →

[Report a bug](#) [Fork me on GitHub](#) [About](#) [This is git revision: acd3f1f](#)

プライベートレジストリ情報の表示



The screenshot shows a web browser window displaying the Docker Registry Frontend interface. The browser's address bar shows the URL `172.17.61.12:8080/tag/docker/nginx/latest/`. The page title is "Docker Registry Frontend". Below the title, a breadcrumb trail indicates the current location: "You are here: Home / Repositories / docker / nginx / latest". The main heading is "Details for tag docker / nginx : latest". Below this heading, a code block shows the command `docker pull 172.17.61.12:5000/docker/nginx:latest`. The "Image Details" section is active, with two tabs: "General information" (selected) and "Labels". Under "General information", the following details are listed:

- Author**: (represented by a person icon)
- Created**: 6 days ago (2017-10-10 09:26:17 +0900)
- Docker version**: 17.06.2-ce
- OS**: linux/amd64
- ID**: 7be53365f962

プライベートレジストリの利用

nginx をこのレジストリを使って起動します。

```
$ docker tag nginx:latest 172.17.61.12:5000/docker/nginx:latest
$ docker push 172.17.61.12:5000/docker/nginx
The push refers to a repository [172.17.61.12:5000/docker/nginx]
...
$ cat pr-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pr
  labels:
    run: nginx-pr1
spec:
  containers:
  - name: nginx-container
    image: 172.17.61.12:5000/docker/nginx
    ports:
    - containerPort: 80
$ kubectl apply -f pr-nginx.yaml
$ kubectl describe pod nginx-pr | grep Image:
  Image:      172.17.61.12:5000/docker/nginx
$
```


プライベートレジストリの利用

コンテナを置き換えるにはレジストリのイメージを更新しpodを再作成するだけです。

新しいイメージを作成しpushします。

```
$ docker push 172.17.61.12:5000/docker/nginx
The push refers to a repository [172.17.61.12:5000/docker/nginx]
...
$
```

podを作り直しても良いがここでは k8s2 をrebootします。

```
$ kubectl get serviceexpose pod nginx-pr --type=NodePort
$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	17h
nginx-pr	NodePort	10.100.251.184	<none>	80:31885/TCP	3s
nginx-sv	NodePort	10.99.38.40	<none>	80:30009/TCP	43s

```
$
```

プライベートレジストリの利用

```
$ curl http://127.0.0.1:31885
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>I updated this message!!!</p>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
$
```

← 追加した行

参考資料

ドキュメント

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

<https://kubernetes.io/docs/setup/independent/install-kubeadm/>

<https://docs.docker.com/engine/installation/>

<https://docs.projectcalico.org/v2.6/getting-started/kubernetes/installation/hosted/kubeadm/>

Contact

<http://slack.k8s.io/>

<https://groups.google.com/forum/#!forum/kubernetes-users>