5. Проверка работы кластера

Набор файлов для создания тестового деплоя лежит здесь: https://gitlab.vsk.ru/infra/kubernetes/tree/master/sample

Создание образа приложения

Приложение простейшее, скрипт под nginx для файлборда (взят из обучающего курса Slurm):

Образ	Скрипт-приложение

Dockerfile
`FROM nexus.vsk.ru:8182/nginx:1.12`
`RUN set -ex && \ apt-get update && \ apt-get install -y procps && \ apt-get clean`
`COPY docker-entrypoint.sh /`
`ENTRYPOINT ["/docker-entrypoint.sh"]`
Основа - уже готовый официальный образ nginx, который проксируется на Nexus.
После чего запускаем сборку **build.sh**
	docker-entrypoint.sh
	`#!/usr/bin/env bash`
	`DIR="etc/nginx/"`
	`RED='\033[0;31m' GREEN='\033[0;32m' NC='\033[0m'`
	`log() { case $1 in error) LOG_LEVEL="error" COLOR=$RED ;; notice) LOG_LEVEL="notice" COLOR=$GREEN ;; esac timestamp="$(date +"%Y/%m/%d %H:%M:%S")" echo -e "$timestamp [$LOG_LEVEL] $0: ${COLOR}$2${NC}" }`
	`getmd5() { tar --strip-components=2 -C / -cf - $DIR | md5sum | awk '{print $1}' }`

```
#!/usr/bin/env bash
set +e
IMAGE_VERSION="0.1"
NAME="sample-fileboard"

docker build -t
nexus.vsk.ru:8183/vsk/$NAME:$
IMAGE_VERSION ./
docker tag
nexus.vsk.ru:8183/vsk/$NAME:$
IMAGE_VERSION
nexus.vsk.ru:8183/vsk/$NAME:l
atest
docker push
nexus.vsk.ru:8183/vsk/$NAME:$
IMAGE_VERSION
docker push
nexus.vsk.ru:8183/vsk/$NAME:l
atest
```

Листинг работы:

```
$ sudo ./build.sh
Sending build context to
Docker daemon 11.78 kB
Step 1/4 : FROM
nexus.vsk.ru:8182/nginx:1.12
Trying to pull repository
nexus.vsk.ru:8182/nginx ...
1.12: Pulling from
nexus.vsk.ru:8182/nginx
f2aa67a397c4: Pull complete
e3eaf3d87fe0: Pull complete
38cb13c1e4c9: Pull complete
Digest:
sha256:09e210fe1e7f54647344d2
78a8d0dee8a4f59f275b72280e8b5
a7c18c560057f
Status: Downloaded newer
image for
nexus.vsk.ru:8182/nginx:1.12
 ---> 4037a5562b03
Step 2/4 : RUN set -ex &&
apt-get update &&     apt-get
install -y procps &&
apt-get clean
 ---> Running in ef8474b0c291

+ apt-get update
Get:1
http://security.debian.org/de
```

```
if [ ! -d $DIR ]; then
    log error "/$DIR not
found"
    exit 1
fi

if ! [ -x "$(command -v
nginx)" ]; then
  log error "Nginx is not
installed"
  exit 1
fi

log notice "starting Nginx
process..."
nginx

log notice "watching /$DIR
for changes..."
checksum_initial=$(getmd5)

trap "exit 0" SIGINT SIGTERM
while true; do
    ps aux | grep 'master
process nginx' | grep -q -v
grep
    NGINX_STATUS=$?
    if [ $NGINX_STATUS -ne 0
]; then
        log error "Nginx
exited. Stopping entrypoint
script..."
        exit 1
    fi

checksum_current=$(getmd5)
    if [ "$checksum_initial"
!= "$checksum_current" ];
then

checksum_initial=$checksum_cu
rrent

        nginx -tq
        NGINX_CONF_STATUS=$?
        if [
$NGINX_CONF_STATUS -ne 0 ];
then
            log error
"couldn't reload Nginx due to
an error in the config file"
            continue
        fi
```

```
bian-security stretch/updates
InRelease [94.3 kB]
Get:2
http://nginx.org/packages/deb
ian stretch InRelease [2854
B]
Ign:3
http://cdn-fastly.deb.debian.
org/debian stretch InRelease
Get:4
http://cdn-fastly.deb.debian.
org/debian stretch-updates
InRelease [91.0 kB]
Get:5
http://cdn-fastly.deb.debian.
org/debian stretch Release
[118 kB]
Get:6
http://cdn-fastly.deb.debian.
org/debian stretch
Release.gpg [2434 B]
Get:7
http://nginx.org/packages/deb
ian stretch/nginx amd64
Packages [10.4 kB]
Get:8
http://security.debian.org/de
bian-security
stretch/updates/main amd64
Packages [489 kB]
Get:9
http://cdn-fastly.deb.debian.
org/debian
stretch-updates/main amd64
Packages [5476 B]
Get:10
http://cdn-fastly.deb.debian.
org/debian stretch/main amd64
Packages [9500 kB]
Fetched 10.3 MB in 13s (750
kB/s)
Reading package lists...
+ apt-get install -y procps
Reading package lists...
Building dependency tree...
Reading state information...
The following additional
packages will be installed:
  libprocps6 psmisc
The following NEW packages
will be installed:
  libprocps6 procps psmisc
0 upgraded, 3 newly
```

```
        nginx -s reload
        log notice "reloaded
Nginx config"
```

```
installed, 0 to remove and 13
not upgraded.
Need to get 431 kB of
archives.
After this operation, 1434 kB
of additional disk space will
be used.
Get:1
http://cdn-fastly.deb.debian.
org/debian stretch/main amd64
libprocps6 amd64
2:3.3.12-3+deb9u1 [58.5 kB]
Get:2
http://cdn-fastly.deb.debian.
org/debian stretch/main amd64
procps amd64
2:3.3.12-3+deb9u1 [250 kB]
Get:3
http://cdn-fastly.deb.debian.
org/debian stretch/main amd64
psmisc amd64 22.21-2.1+b2
[123 kB]
debconf: delaying package
configuration, since
apt-utils is not installed
Fetched 431 kB in 0s (491
kB/s)
Selecting previously
unselected package
libprocps6:amd64.
(Reading database ... 7027
files and directories
currently installed.)
Preparing to unpack
.../libprocps6_2%3a3.3.12-3+d
eb9u1_amd64.deb ...
Unpacking libprocps6:amd64
(2:3.3.12-3+deb9u1) ...
Selecting previously
unselected package procps.
Preparing to unpack
.../procps_2%3a3.3.12-3+deb9u
1_amd64.deb ...
Unpacking procps
(2:3.3.12-3+deb9u1) ...
Selecting previously
unselected package psmisc.
Preparing to unpack
.../psmisc_22.21-2.1+b2_amd64
.deb ...
Unpacking psmisc
(22.21-2.1+b2) ...
Setting up psmisc
```

```
    fi
    sleep 5
done
```

Собственно, все, что этот скрипт делает - перезапускает nginx

```
(22.21-2.1+b2) ...
Setting up libprocps6:amd64
(2:3.3.12-3+deb9u1) ...
Setting up procps
(2:3.3.12-3+deb9u1) ...
update-alternatives: using
/usr/bin/w.procps to provide
/usr/bin/w (w) in auto mode
update-alternatives: warning:
skip creation of
/usr/share/man/man1/w.1.gz
because associated file
/usr/share/man/man1/w.procps.
1.gz (of link group w)
doesn't exist
Processing triggers for
libc-bin (2.24-11+deb9u3) ...
+ apt-get clean
 ---> 2b823a47eb0d
Removing intermediate
container ef8474b0c291
Step 3/4 : COPY
docker-entrypoint.sh /
 ---> d9580d3198fc
Removing intermediate
container 05c5be1d6b0d
Step 4/4 : ENTRYPOINT
/docker-entrypoint.sh
 ---> Running in 06459fc02e02
 ---> baa162667810
Removing intermediate
container 06459fc02e02
Successfully built
baa162667810
The push refers to a
repository
[nexus.vsk.ru:8183/vsk/sample
-fileboard]
4dbb063a6048: Pushed
9efbf8b3ce98: Pushed
4258832b2570: Layer already
exists
683a28d1d7fd: Layer already
exists
d626a8ad97a1: Layer already
exists
0.1: digest:
sha256:df887a41fbbdef7d68b54d
e5cca97c80151e6f857675bdc67fd
87f48c7f3ded5 size: 1367
The push refers to a
repository
[nexus.vsk.ru:8183/vsk/sample
```

```
-fileboard]
4dbb063a6048: Layer already
exists
9efbf8b3ce98: Layer already
exists
4258832b2570: Layer already
exists
683a28d1d7fd: Layer already
exists
d626a8ad97a1: Layer already
exists
latest: digest:
sha256:df887a41fbbdef7d68b54d
e5cca97c80151e6f857675bdc67fd
87f48c7f3ded5 size: 1367


$ sudo docker images
REPOSITORY
TAG                 IMAGE ID
CREATED             SIZE
nexus.vsk.ru:8183/vsk/sample-
fileboard     0.1
baa162667810        10
minutes ago      126 MB
nexus.vsk.ru:8183/vsk/sample-
```

```
fileboard     latest
baa162667810        10
minutes ago      126 MB
```

Деплой приложения

> Не делайте это до заведения worker node - не запустится, не на чем

Kubernetes репозиторий имеет много преднастроенных POD которые удобно использовать для тестирования корректности работы кластера, например:

https://kubernetes.io/docs/tasks/access-application-cluster/service-access-application-cluster/ описан простейший пример установки готового приложения из репозитория.

Создание и запуск сервиса

Создать POD-файл с использованием image из docker репозитория:

nginx.yaml

```
cat <<EOF > swar-test.yaml
apiVersion: v1
kind: ReplicationController
metadata:
   name: nginx
spec:
   replicas: 2
   selector:
      app: nginx
   template:
      metadata:
         name: nginx
         labels:
            app: nginx
      spec:
         containers:
         - name: nginx
           image: nginx
           ports:
           - containerPort: 80

EOF
```

выполнить команду ***kubectl apply -f nginx.yaml***

посмотреть состояние развернутых приложений **kubectl get pods**

Запуск сервиса в кластере, структура POD

- kubectl apply -f nginx.yaml

результата можно посмотреть уже известными командами ***kubectl get pods, kubectl describe $NAME***

Простой пример развертывания сервиса в кластере от исходника до публикации сервиса:

Предполагается, что на сервере docker установлен и работает

установить openjdk, git, gradle

```
sudo yum install -y java-1.8.0-openjdk-devel git gradle
```

скачать исходники примерчика

```
git clone https://github.com/mayatnikov/rest-example.git
cd rest-example
```

собрать uber-jar

```
gradle build
```

проверить работоспособность модуля

```
 cp build/libs/app1-0.1.1.jar app.jar
 java -jar  app.jar --server.port=9993
```

в другом окне проверить REST запросы:

```
$ curl -XPOST localhost:9993/inf
OK. tik=1 cc-test/192.168.96.39:9993[mayatnikov@cc-test ~]$ curl -XPOST
localhost:9993/inf
OK. tik=2 cc-test/192.168.96.39:9993[mayatnikov@cc-test ~]$ curl -XPOST
localhost:9993/inf
OK. tik=3 cc-test/192.168.96.39:9993[mayatnikov@cc-test ~]$ curl -XPOST
localhost:9993/inf
OK. tik=4 cc-test/192.168.96.39:9993[mayatnikov@cc-test ~]$ curl -XPOST
localhost:9993/inf
OK. tik=5 cc-test/192.168.96.39:9993[mayatnikov@cc-test ~]$
```

собрать docker image

```
$ cd docker-files
$ cp ../build/libs/app1-0.1.1.jar app1.jar
$ docker build -t app1:latest .
$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
app1 latest 9d4a90edd4d1 8 minutes ago 118MB
```

на сервере k8s-master (мастер нода кластера) собрать POD файл

```
cat <<EOF > swarm-test.yaml
apiVersion: extensions/v1
kind: Deployment                                    # 1
metadata:
  name: swarm-test
spec:
  replicas: 2                                       # 2
  minReadySeconds: 15
  strategy:
    type: RollingUpdate                             # 3
    rollingUpdate:
      maxUnavailable: 1                             # 4
      maxSurge: 1                                   # 5
  template:
    metadata:
      labels:
        app: swarm-test                           #
    spec:
      containers:
        - image: mayatnikov/swarm-test              #     docker (
docker-hub)
          imagePullPolicy: Always                   # 7
          name: swarm-test
          ports:
            - containerPort: 8888
EOF
kubectl apply -f swarm-test.yaml
```

опубликовать сервис для внешнего доступа (kubernates-proxy service):

```
cat <<EOF > swarm-test-lb.yaml
apiVersion: v1
kind: Service
metadata:
  name: swarm-test-lb
spec:
  type: LoadBalancer
  externalTrafficPolicy: Local
  ports:
  - port: 7777
    protocol: TCP
    targetPort: 8888
  externalIPs:
  - 192.168.17.48  #
  - 192.168.17.44
  - 192.168.17.45
  - 192.168.17.130  #       (    )
  selector:
    app: swarm-test
EOF
kubectl apply -f swarm-test-lb.yaml
```

тестируем сервис:

```
curl -XPOST  192.168.17.48:7777/inf
curl -XPOST  192.168.17.44:7777/inf
curl -XPOST  192.168.17.45:7777/inf
curl -XPOST  192.168.17.48:7777/kill  #   ,  kubernates       replicas:
```

Все указанные действия легко скриптуются, а следовательно хорошо ложатся на CI/CD

В примере при сборке использовалась работа через github, для работы с nexus.vsk.ru нужны небольшие коррективы

Полезные ссылки:

Сайт продукта

Нужен ли Kubernetes бизнесу https://habr.com/company/eastbanctech/blog/419353/?utm_source=twitter.com&utm_medium=social&utm_campaign=perehod-na-kubernetes-biznesu-na-zametk

Ссылка на трансляцию Слёрма (интенсива по Kubernetes) aSkobin 3 августа в 09:13 1,9k

Трансляция второго дня Слёрма: интенсива по Кубернетес aSkobin 4 августа в 08:43 1k

Трансляция третьего дня Слёрма: интенсива по Кубернетес aSkobin 5 августа в 08:44 1,1k

Сборник примеров по базовым возможностям kubernates

Fabric8 - быстрый старт и примеры кнему