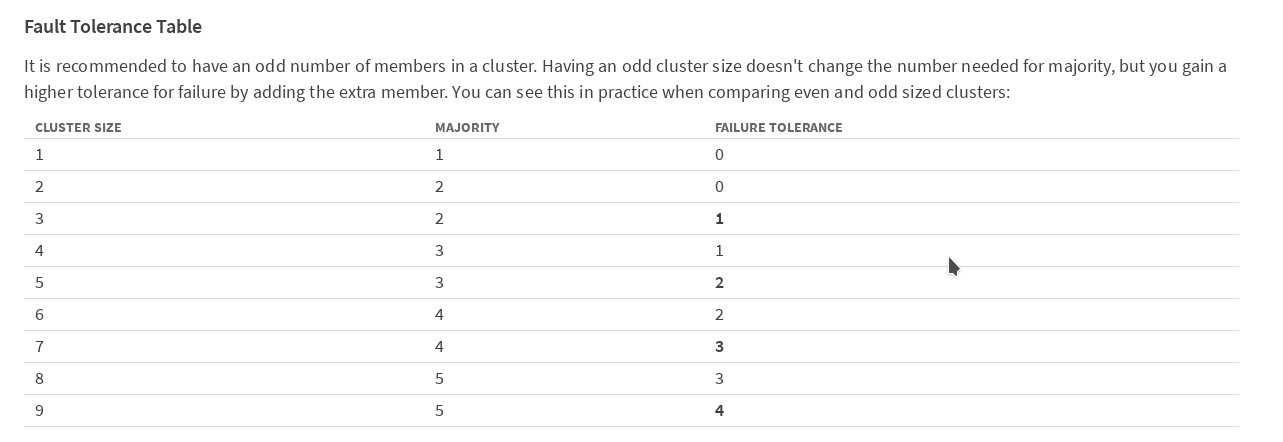
Docker swarm кластер c 3 мастерами

1. [Список используемого софта](https://habr.com/post/358264/#po)
2. [Список и назначение хостов](https://habr.com/post/358264/#hostslist)
3. [Принцип работы и развертывания](https://habr.com/post/358264/#deployshema)
4. [Подготовка ОС к развертыванию. Установка docker, kubeadm, kubelet и kubectl](https://habr.com/post/358264/#beforebegin)
5. [Подготовка конфигурационного скрипта](https://habr.com/post/358264/#startscript)
6. [Создание etcd кластера](https://habr.com/post/358264/#etcd)
7. [Инициализация мастера с помощью kubeadm](https://habr.com/post/358264/#firstmaster)
8. [Настройка CIDR](https://habr.com/post/358264/#cidr)
9. [Инициализация остальных мастернод](https://habr.com/post/358264/#othermaster)
10. [Настройка keepalived и виртуального IP](https://habr.com/post/358264/#balancer)
11. [Добавление рабочих нод в кластер](https://habr.com/post/358264/#worknode)
12. [Установка ingress-nginx](https://habr.com/post/358264/#ingress)
13. Дополнительно
    * [Dashboard](https://habr.com/post/358264/#dashboard)
    * [Heapster](https://habr.com/post/358264/#heapster)

Задача - поместить проект в отказоустойчивый кластер при помощи swarm.   
  
Для повышенной отказоустойчивости кластер было решено разворачивать с тремя мастер нодами.  
  
Согласно таблице в документации etcd на [сайте CoreOS](https://coreos.com/etcd/docs/latest/v2/admin_guide.html),   
  
  
  
рекомендуется иметь нечетное число членов в кластере. Для того, чтобы кластер продолжал работать после выхода из строя одного члена (в нашем случае мастера), нужно минимум 3 машины.

**Список используемого софта**

* ОС - CentOS 7.5-1804
* контейнеры:  
  docker version 18.06.1-ce, build e68fc7a, docker-compose version 1.22.0
* Балансировщик: nginx
* Виртуальный IP: keepalived

Список хостов

|  |  |  |  |
| --- | --- | --- | --- |
| **IP адрес** | **Имена хостов** | **Описание** | **Компоненты** |
| 10.90.2.41 | hb-master01 | master nodes \* 3 | docker |
| 10.90.2.51 | hb-master02 |
| 10.90.2.61 | hb-master03 |
| 10.90.2.42 | hb-node01 | worker nodes \* 3 | docker |
| 10.90.2.52 | hb-node02 |
| 10.90.2.62 | hb-node03 |
| 10.90.2.111 | N\A | keepalived virtual IP | N\A |

Подготовка ОС к развертыванию. Установка docker, pre & post install steps on all hosts

Для подготовки каждого хоста запускаем скрипт hosts-prepare.sh

ssh root@hb-master01 "bash -s" < ./master-hosts-prepare.sh > mhp01.log

ssh root@hb-master02 "bash -s" < ./master-hosts-prepare.sh > mhp02.log

ssh root@hb-master03 "bash -s" < ./master-hosts-prepare.sh > mhp03.log

ssh root@hb-node01 "bash -s" < ./worker-hosts-prepare.sh > whp01.log

ssh root@hb-node02 "bash -s" < ./worker-hosts-prepare.sh > whp02.log

ssh root@hb-node03 "bash -s" < ./worker-hosts-prepare.sh > whp03.log

на мастер лидере (hb-master01) выполняем

docker swarm init

docker swarm join-token manager | awk '/docker/' > join-master.sh

docker swarm join-token worker | awk '/docker/' > join-worker.sh

На каждой мастер ноде запускаем скрипт join-master.sh  
ssh root@hb-master02 "bash -s" < ./join-master.sh > join-mh02.log

ssh root@hb-master03 "bash -s" < ./join-master.sh > join-mh03.log

На каждой worker ноде запускаем скрипт join-worker.sh  
ssh root@hb-node01 "bash -s" < ./join-worker.sh > join-wh01.log

ssh root@hb-node02 "bash -s" < ./join-worker.sh > join-wh02.log

ssh root@hb-node03 "bash -s" < ./join-worker.sh > join-wh03.log

Создание etcd кластера

Настройка keepalived и виртуального IP

На каждом хосте стартуем keepalived

На hb-master01

./m1.sh

ssh root@hb-master02 "bash -s" < ./m2.sh

ssh root@hb-master03 "bash -s" < ./m3.sh

ssh root@hb-node01 "bash -s" < ./w1.sh

ssh root@hb-node02 "bash -s" < ./w2.sh

ssh root@hb-node03 "bash -s" < ./w3.sh

Проверяем

docker logs keepalived

в логах должны увидеть

I'm the MASTER! Whup whup.

Wed Sep 26 11:47:52 2018: Sending gratuitous ARP on bond0 for 10.90.2.111

Тепреь у нас есть общая точка входа 10.90.2.111

Ей и будем пользоваться при создании сервисов

Установка web менеджера portainer

Создание общего дискового пространства Ceph

Все операци проделываем на каждом хосте.

Во время установки CentOS7 диск был размечен по умолчанию.

[root@hb-node01 ~]# lvmdiskscan

/dev/centos/root [ 50.00 GiB]

/dev/sda1 [ 200.00 MiB]

/dev/centos/swap [ 7.62 GiB]

/dev/sda2 [ 1.00 GiB]

/dev/centos/home [ <220.05 GiB]

/dev/sda3 [ <277.68 GiB] LVM physical volume

3 disks

2 partitions

0 LVM physical volume whole disks

1 LVM physical volume

[root@hb-node01 ~]# lvdisplay

--- Logical volume ---

LV Path /dev/centos/swap

LV Name swap

VG Name centos

LV UUID 75T4bO-FvzF-CTSH-KGrR-4U1u-8bSP-kC0zNc

LV Write Access read/write

LV Creation host, time host0, 2018-09-11 03:53:15 -0400

LV Status available

# open 0

LV Size 7.62 GiB

Current LE 1952

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 253:1

--- Logical volume ---

LV Path /dev/centos/home

LV Name home

VG Name centos

LV UUID n17s19-eM7h-2aw7-7ptI-11yJ-ie4D-IddoMK

LV Write Access read/write

LV Creation host, time host0, 2018-09-11 03:53:15 -0400

LV Status available

# open 1

LV Size <220.05 GiB

Current LE 56332

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 253:2

--- Logical volume ---

LV Path /dev/centos/root

LV Name root

VG Name centos

LV UUID 1r9U5K-1Hpu-IjfY-chWr-hSNm-PaOs-KFsjxM

LV Write Access read/write

LV Creation host, time host0, 2018-09-11 03:53:19 -0400

LV Status available

# open 1

LV Size 50.00 GiB

Current LE 12800

Segments 1

Allocation inherit

Read ahead sectors auto

- currently set to 256

Block device 253:0

Для создания кластера Ceph нам требуется отдельный логический диск.

Отключаем раздел /dev/mapper/centos-home

[root@hb-node01 ~]# umount /home

проверяем

[root@hb-node01 ~]# df -H

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/centos-root 54G 1.6G 53G 3% /

devtmpfs 4.0G 0 4.0G 0% /dev

tmpfs 4.1G 0 4.1G 0% /dev/shm

tmpfs 4.1G 44M 4.0G 2% /run

tmpfs 4.1G 0 4.1G 0% /sys/fs/cgroup

/dev/sda2 1.1G 185M 879M 18% /boot

/dev/sda1 210M 12M 198M 6% /boot/efi

tmpfs 801M 0 801M 0% /run/user/0

Т.к. раздел в ФС XFS уменьшить не возможно, то удаляем и логический раздел lvm

[root@hb-node01 ~]# lvremove /dev/centos/home -y

Logical volume "home" successfully removed

Создаем логический раздел lvm для Ceph

[root@hb-node01 ~]# lvcreate -L200G -n ceph centos -y

Logical volume "ceph" created.

Создаем логический раздел lvm для /home

[root@hb-node01 ~]# lvcreate -l100%FREE centos -n home

Logical volume "home" created.

файловая система xfs на раздел home

[root@hb-node01 ~]# mkfs.xfs /dev/centos/home

meta-data=/dev/centos/home isize=512 agcount=4, agsize=1314048 blks

= sectsz=4096 attr=2, projid32bit=1

= crc=1 finobt=0, sparse=0

data = bsize=4096 blocks=5256192, imaxpct=25

= sunit=0 swidth=0 blks

naming =version 2 bsize=4096 ascii-ci=0 ftype=1

log =internal log bsize=4096 blocks=2566, version=2

= sectsz=4096 sunit=1 blks, lazy-count=1

realtime =none extsz=4096 blocks=0, rtextents=0

монтируем /home на новый раздел

[root@hb-node01 ~]# mount /dev/centos/home /home

Подготовка мониторов

На первой мастер ноде выполним

docker run -d --net=host \

--restart always \

-v /etc/ceph:/etc/ceph \

-v /var/lib/ceph/:/var/lib/ceph/ \

-e MON\_IP=10.90.2.41 \

-e CEPH\_PUBLIC\_NETWORK=10.90.2.0/24 \

--name="ceph-mon" \

ceph/daemon mon

теперь скопируем /etc/ceph на остальные ноды

scp -r /etc/ceph root@hb-master02:/etc/

scp -r /etc/ceph root@hb-master03:/etc/

scp -r /etc/ceph root@hb-node01:/etc/

scp -r /etc/ceph root@hb-node02:/etc/

scp -r /etc/ceph root@hb-node03:/etc/

На второй мастер ноде выполним

docker run -d --net=host \

--restart always \

-v /etc/ceph:/etc/ceph \

-v /var/lib/ceph/:/var/lib/ceph/ \

-e MON\_IP=10.90.2.51 \

-e CEPH\_PUBLIC\_NETWORK=10.90.2.0/24 \

--name="ceph-mon" \

ceph/daemon mon

На третьей мастер ноде выполним

docker run -d --net=host \

--restart always \

-v /etc/ceph:/etc/ceph \

-v /var/lib/ceph/:/var/lib/ceph/ \

-e MON\_IP=10.90.2.61 \

-e CEPH\_PUBLIC\_NETWORK=10.90.2.0/24 \

--name="ceph-mon" \

ceph/daemon mon

Подготовка менеджеров

На каждом хосте кластера выполним

docker run -d --net=host \

--privileged=true \

--pid=host \

-v /etc/ceph:/etc/ceph \

-v /var/lib/ceph/:/var/lib/ceph/ \

--name="ceph-mgr" \

--restart=always \

ceph/daemon mgr

Подготовка OSD

Поскольку в настоящее время у нас есть кластер только с мониторами - без OSD, подготовимся к созданию OSD, сбросив учетные данные для OSD в соответствующее место на базовой ОС:

Запустим на каждой ноде:

docker exec -it ceph-mgr ceph auth get client.bootstrap-osd -o \

/var/lib/ceph/bootstrap-osd/ceph.keyring

docker run -d --net=host \

--privileged=true \

--pid=host \

-v /etc/ceph:/etc/ceph \

-v /var/lib/ceph/:/var/lib/ceph/ \

-v /dev/:/dev/ \

-e OSD\_FORCE\_ZAP=1 \

-e OSD\_DEVICE=/dev/centos/ceph \

-e OSD\_TYPE=disk \

--name="ceph-osd" \

--restart=always \

ceph/daemon osd\_ceph\_disk

Добавление рабочих нод в кластер

На каждой рабочей ноде устанавливаем docke, kubernetes и kubeadm, по аналогии с мастерами. worker-hosts-prepare.sh  
worker-hosts-prepare.sh

reboot

Добавляем ноду в кластер, используя токены сгенерированные при инициализации hb-master01

kubeadm join 10.90.2.41:6443 --token yf0t14.7ekahohaetbdphqg --discovery-token-ca-cert-hash sha256:5c04e29c493d3290483e4f900c30b08527770394304eb4b974149061af276c1a

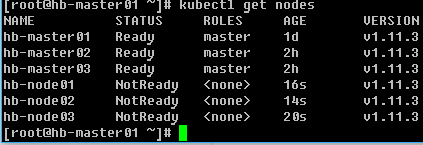
modprobe ip\_vs\_rr

modprobe ip\_vs\_wrr

modprobe ip\_vs\_sh

modprobe ip\_vs  
Проверяем, что все рабочие ноды вошли в кластер и они доступны. 

kubectl get nodes



**Только на рабочих нодах** в файлах /etc/kubernetes/bootstrap-kubelet.conf и /etc/kubernetes/kubelet.conf меняем  
значение переменной server на наш виртуальный IP

vi /etc/kubernetes/bootstrap-kubelet.conf

server: https://10.90.2.111:16443

vi /etc/kubernetes/kubelet.conf

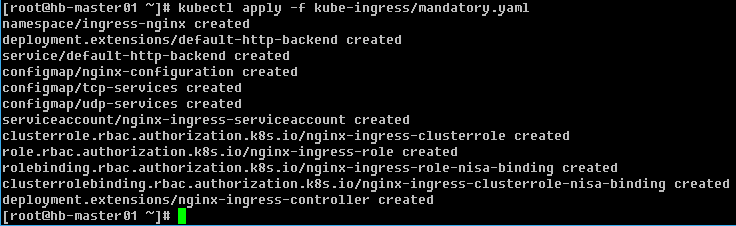
server: https://10.90.2.111:16443

systemctl restart docker kubelet

Далее можно увеличивать производительность кластера добавляя новые рабочие ноды, по мере надобности.

**Установка ingress-nginx**  
В документации kubernetes про Ingress написано следующее:  
*Объект API, который управляет внешним доступом к службам в кластере, обычно HTTP.  
Ingress может обеспечивать балансировку нагрузки, завершение SSL и виртуальный хостинг на основе имен.*  
  
В общем то более подробно я вряд ли смогу описать. Настройка ingress это материал для отдельной статьи, в контексте установки кластера я лишь опишу его установку. 

kubectl apply -f kube-ingress/mandatory.yaml

 kubectl apply -f kube-ingress/service-nodeport.yaml

  
Проверяем что ingress поднялся:

kubectl get all -n ingress-nginx



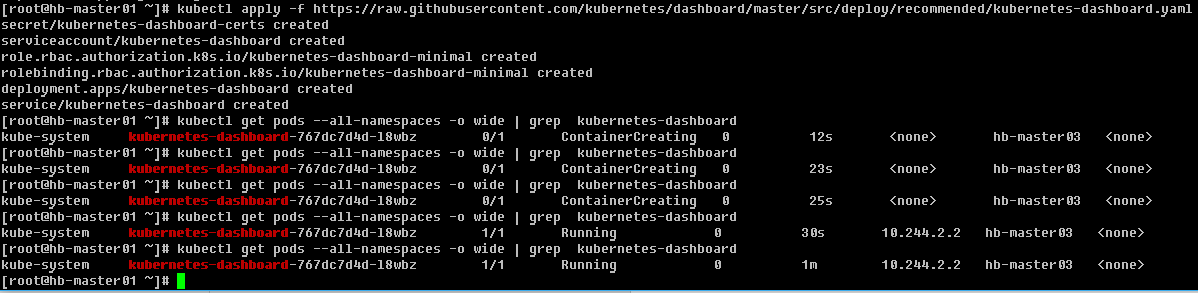
На этом шаге настройка кластера закончена. Получили отказоустойчивый, рабочий кластер Kubernetes c отказоустойчивой точкой входа и балансировщиком на виртуальном адресе. 

Дополнительно

|  
**Установка панели управления Kubernetes Dashboard**  
У Kubernetes кроме cli, имеется не плохая панель инструментов. Устанавливается она очень просто, инструкция и документация есть на [GitHub](https://github.com/kubernetes/dashboard)  
  
Команды можно выполнять на любом из 3 мастеров. Я работаю с hb-master01

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml

Проверяем:

kubectl get pods --all-namespaces -o wide | grep kubernetes-dashboar   
Панель теперь доступна по адресу:

<http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>

Но чтобы на нее попасть, нужно пробрасывать proxy c локальной машины с помощью команды

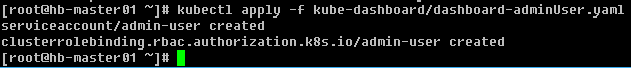
kubectl proxy

Это не удобно, поэтому используем [NodePort](https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport)и размещаем панель по адресу [https://10.90.2.111:30000](https://10.90.2.111:30000 )на первый доступный порт из диапазона, выделенного для NodePort.

kubectl -n kube-system edit service kubernetes-dashboard

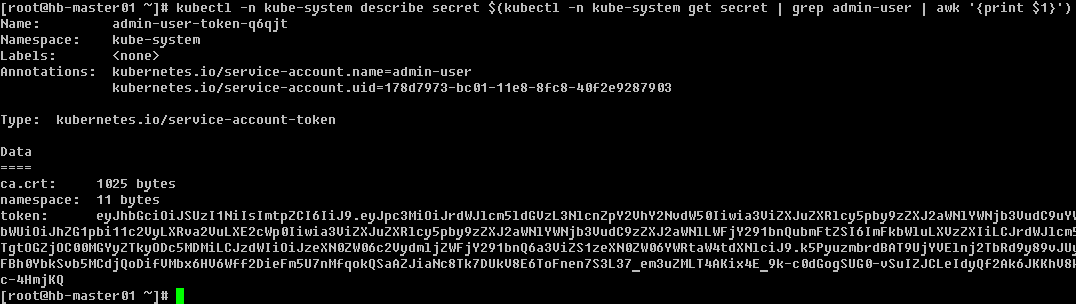
Заменяем значение type: ClusterIP на type: NodePort и в секцию ports: добавляем значение nodePort: 30000  
  
  
  
Далее создадим пользователя с именем admin-user и полномочиями администратора кластера.

kubectl apply -f kube-dashboard/dashboard-adminUser.yaml



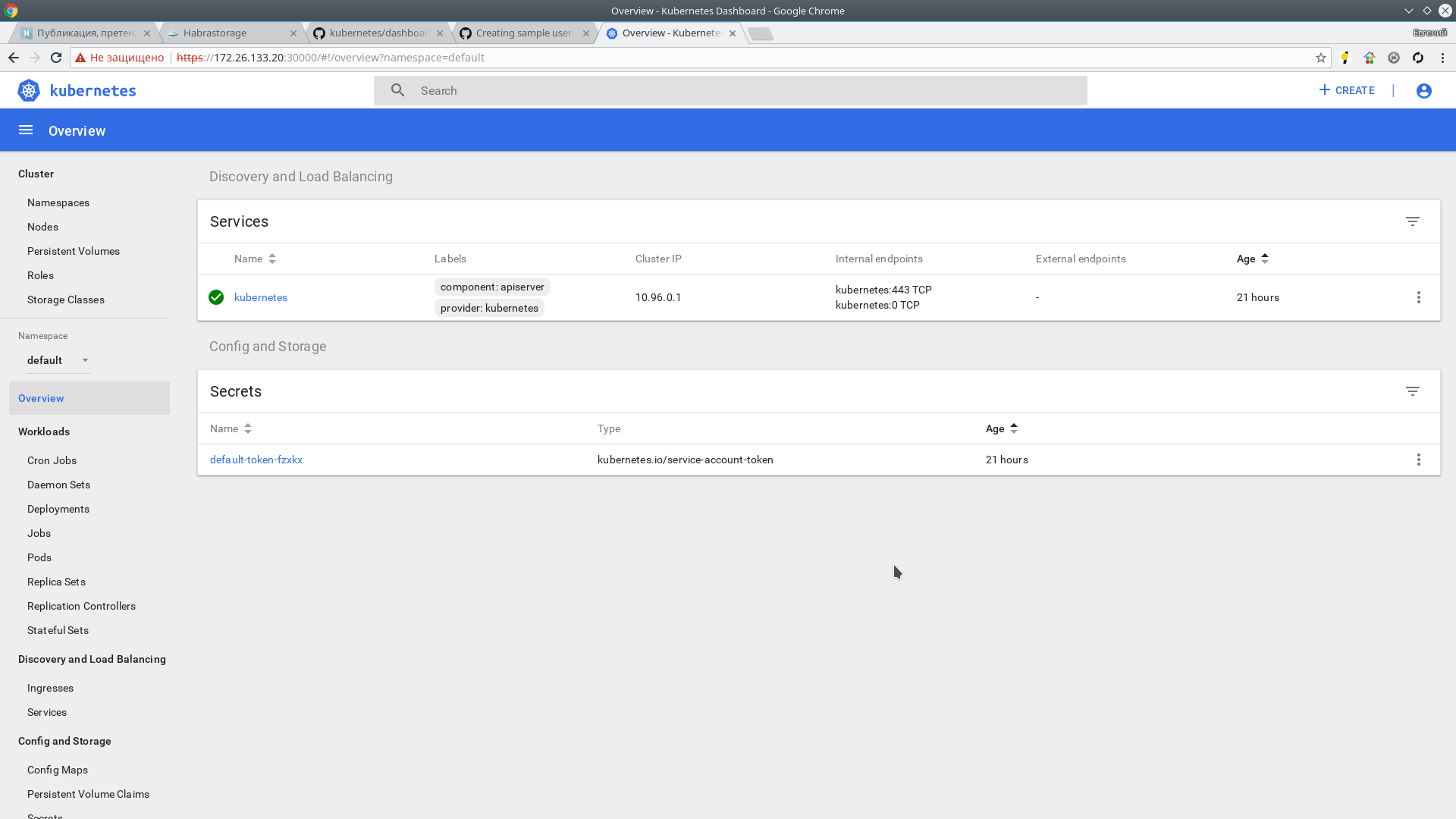
Получаем токен для пользователя admin-user

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')



Копируем token и переходим по адресу:

<https://172.26.133.20:30000/>

Теперь нам доступна панель управления Kubernetes кластера c полномочиями админа.   
  


Heapster

Далее установим Heapster. Это инструмент для мониторинга ресурсов всех составляющих кластера.[Страничка проекта на GitHub](https://github.com/kubernetes/heapster)   
  
Установка:

git clone https://github.com/kubernetes/heapster.git

cd heapster

kubectl create -f deploy/kube-config/influxdb/

deployment "monitoring-grafana" created

service "monitoring-grafana" created

serviceaccount "heapster" created

deployment "heapster" created

service "heapster" created

deployment "monitoring-influxdb" created

service "monitoring-influxdb" created

kubectl create -f deploy/kube-config/rbac/heapster-rbac.yaml

clusterrolebinding "heapster" created

Через пару минут должна пойти информация. Проверяем:

kubectl top nodes

NAME CPU(cores) CPU% MEMORY(bytes) MEMORY%

hb-master01 166m 4% 1216Mi 31%

hb-master02 135m 3% 1130Mi 29%

hb-master03 142m 3% 1091Mi 28%

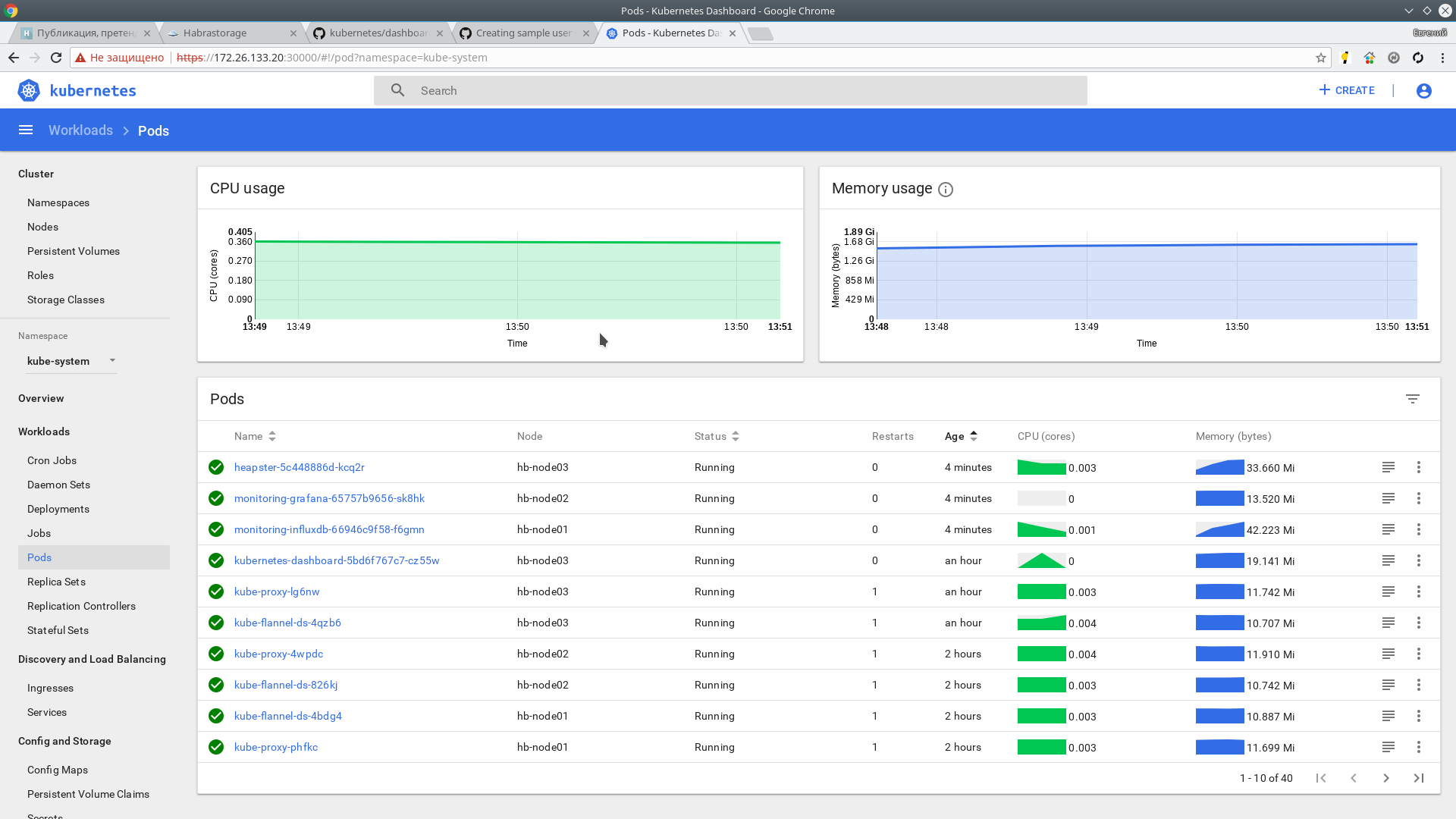
hb-master04 193m 4% 1149Mi 29%

hb-master05 144m 3% 1056Mi 27%

hb-node01 41m 1% 518Mi 3%

hb-node02 38m 0% 444Mi 2%

hb-node03 45m 1% 478Mi 2%

Также метрики доступны в web интерфейсе  
  
  
  
**Использование материалы:**

* [kubeadm-highavailiability — kubernetes high availiability deployment based on kubeadm, for Kubernetes version 1.9.x/1.7.x/1.6.x](https://github.com/cookeem/kubeadm-ha)
* [Kubernetes Documentation](https://kubernetes.io/docs/home/?path=users&persona=app-%20developer&level=foundational)
* [Основы Kubernetes](https://habrahabr.ru/post/258443/)
* [Accessing Kubernetes Pods from Outside of the Cluster](http://alesnosek.com/blog/2017/02/14/accessing-kubernetes-pods-from-outside-of-the-cluster/)

**Метки:**

* [k8s](https://habr.com/search/?q=%5Bk8s%5D&target_type=posts)
* [kubernetes](https://habr.com/search/?q=%5Bkubernetes%5D&target_type=posts)
* [HA](https://habr.com/search/?q=%5BHA%5D&target_type=posts)
* [кластер](https://habr.com/search/?q=%5B%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%5D&target_type=posts)