194100, Санкт-Петербург,
Большой Сампсониевский проспект, 68
Тел./факс: +7 (812) 339-45-01
E-mail: info@lar.tech
Web: www.lar.tech

LPWAN.DLMS

ИНСТРУКЦИЯ ПО УСТАНОВКЕ

RU.ЛРВМ.425190-01 32 01-1

Листов 66

Аннотация

Настоящий документ распространяется на программное обеспечение – сервер LPWAN.DLMS, предназначенный для работы с приборами учёта, установленными в рамках инфраструктуры «интернета вещей» (Internet of Things).

Internet of Things (IoT) – это совокупность подключенных объектов со встроенной аппаратной и программной частями, датчиками и беспроводными протоколами связи, предназначенная для сбора и обмена данными с приложениями через беспроводную сеть с доступом в Интернет. IoT позволяет приложениям управлять подключенными объектами и получать от них данные, используя существующую сетевую инфраструктуру в сочетании со специально разработанными системами беспроводного доступа. В результате появляется возможность оцифровки данных полученных от физического мира. IoT генерирует большие объемы данных, позволяющие уменьшить потребление ресурсов и повысить эффективность подключенных систем.

Программное обеспечение «LPWAN.DLMS» является сервером бизнес-логики и обеспечивает категоризацию данных, приходящих от устройств по беспроводным протоколам LPWAN и последующую передачу категорированной информации в приложения пользователей по протоколам DLMS\СПОДЭС и Lartech Metering API.

© Лартех, 2020 г. LoRa Alliance Member

Оглавление

Аннотация 2
1. Общие требования 4
1.1. Процедура установки 4
1.2. Обеспечение установки 5
2. Установка кластера Kubernetes 6
2.1. Установить операционную систему 6
2.2. Настроить ОС 9
2.3. Установить Libvirt 13
2.4. Установить Ceph 19
2.5. Подключить Ceph к Libvirt 22
2.6. Создать виртуальные машины 24
2.7. Установить Kubespray 30
2.8. Настроить балансировщик для внешнего доступа 36
2.9. Установить на кластер дополнения 39
3. Установка LPWAN.DLMS 44
3.1. Установить базовые компоненты 44
3.2. Установить компоненты LPWAN.DLMS 61
3.3. Установить компоненты мониторинга LPWAN.DLMS 64
Лист изменений 66

© Лартех, 2020 г. LoRa Alliance Member

1. Общие требования

Настоящий документ определяет процедуру установки (развертывания) серверной платформы Sophrosyne в виде кластера Kubernetes на выделенном сервере.

Процедура установки кластера Kubernetes на выделенный сервер представляет собой последовательность действий, точное выполнение которых обеспечивает успешное завершение процедуры и достижение заданного результата – установленная на выделенный сервер серверная платформа Sophrosyne в виде кластера Kubernetes.

Документ выполнен в виде технологической инструкции и ориентирован на специалистов группы эксплуатации и технической поддержки.

1.1. Процедура установки

Процедура установки кластера Kubernetes и LPWAN.DLMS на выделенный сервер предполагает выполнение следующих операций:

- Установить операционную систему Ubuntu Server 18.04.1 (64-bit).
- Настроить ОС.
- Установить Libvirt.
- Установить Ceph.
- Подключить Ceph к Libvirt.
- Создать виртуальные машины.
- Установить Kubespray.
- Настроить балансировщик для внешнего доступа.
- Установить на кластер дополнения:
  - панель управления kubernetes (Dashboard);
  - утилиту для управления kubernetes (Kops);
  - пакетный менеджер (Helm);
  - дополнение для приема журналов регистрации системной информации (Elasticsearch);
  - дополнение для базового мониторинга кластера и интеграции метрик в инструментальную панель Kubernetes;
  - Ingress контроллер для NGINX.
- Установить базовые компоненты:
  - PostgresPro;
  - Redis;
  - Kafka.
- Установить компоненты LPWAN.DLMS:
  - metering-discovery;
  - metering—dlms-assistant;
  - metering-mbus-service;
  - metering-network-gateway;
  - metering-processing;
  - metering-public-api;
  - metering-registry;
  - metering-system-router;
  - metering-timeseries.
- Установить компоненты мониторинга LPWAN.DLMS:
  - Prometheus;
  - Grafana.

© Лартех, 2020 г. LoRa Alliance Member

1.2. Обеспечение установки

Для выполнения перечисленных операций необходимы следующие виды обеспечения:

1. Аппаратное обеспечение:
   - Компьютер (Intel Xeon E3-1230 3.4 ГГц, 32 Гб DDR4, 2*240 ГГб SSD, 2*2 Тб SATA) с выходом в Интернет.
2. Программное обеспечение:
   - Дистрибутив Ubuntu Server 18.04.1 (64-bit) (http://releases.ubuntu.com/18.04/ubuntu-18.04.1-live-server-amd64.iso).
3. Информационное обеспечение:
   - Настоящая инструкция.
4. Организационное обеспечение:

   Сотруднику, выполняющему установку кластера Kubernetes на выделенный сервер, должен быть предоставлен доступ к репозиторию распределённой система управления версиями, с правами, необходимыми для загрузки и установки перечисленных в инструкции компонентов.

© Лартех, 2020 г. LoRa Alliance Member

2. Установка кластера Kubernetes

2.1. Установить операционную систему

2.1.1. Скачать образ `Ubuntu Server 18.04.1 (64-bit)` (http://releases.ubuntu.com/18.04/ubuntu-18.04.1-live-server-amd64.iso):

2.1.2. На основе скачанного образа создать загрузочный носитель (flesh-drive, DVD).

2.1.3. Загрузиться с созданного носителя и следовать указаниям установщика.

Исключения: на шаге «Filesystem setup» необходимо разметить диск соответствующим образом; на шаге «Profile setup» необходимо дать серверу имя и создать учетную запись пользователя (имя, пароль).

2.1.4. Для использования LVM в меню разметки диска выбрать: «`Use An Entire Disk And Set Up LVM`».

Примечание: необходимо использовать `LVM` (Logical Volume Manager – система управления томами с данными для Linux).

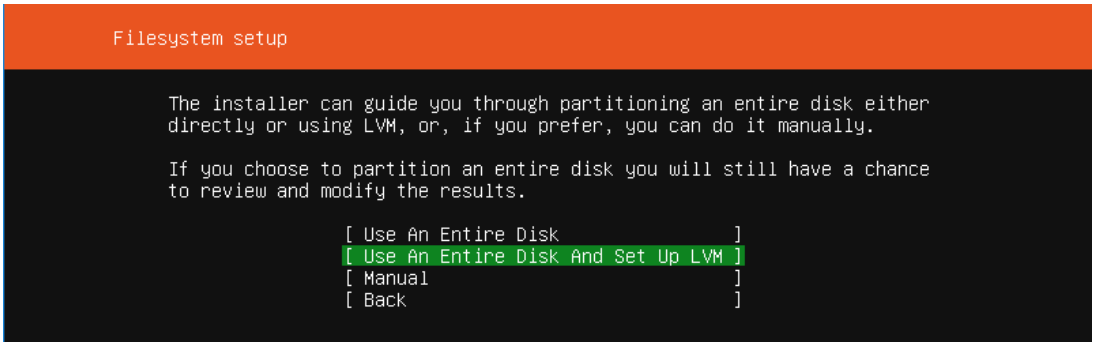


Рисунок 2.1 Режим разметки диска с использованием LVM

2.1.5. Разметить диск:

Примечание: в нашем случае диск был размечен в соответствии с приведенной разметкой: корневой раздел: 160Гб; ceph: 80Гб; ceph1 и ceph2: по 345Гб (все оставшееся свободное место пополам).

© Лартех, 2020 г. LoRa Alliance Member

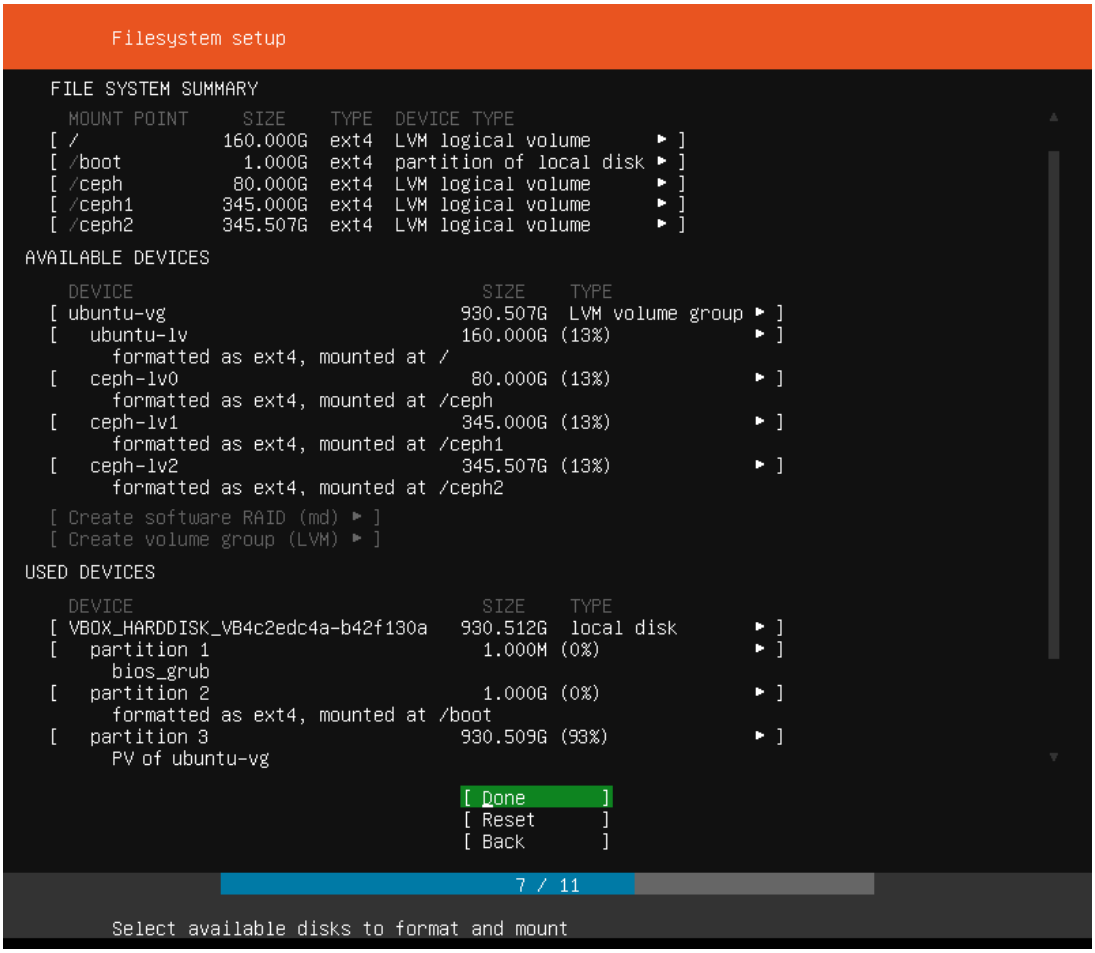


Рисунок 2.2 Диск после разметки

2.1.6. Заполнить профайл.

Примечание: в нашем случае после заполнения профайл имел следующий вид.

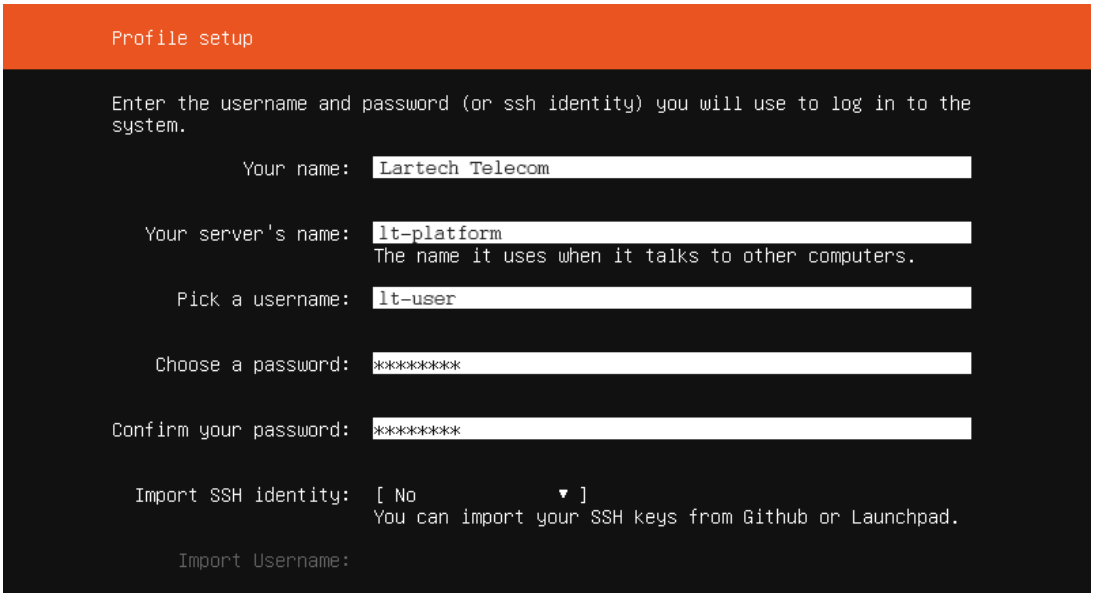


Рисунок 2.3 Заполненный профайл

Примечание: пароль для `lt-user`*:* `lt-user`*.*

2.1.7. Перегрузить систему по завершению сценария установки.

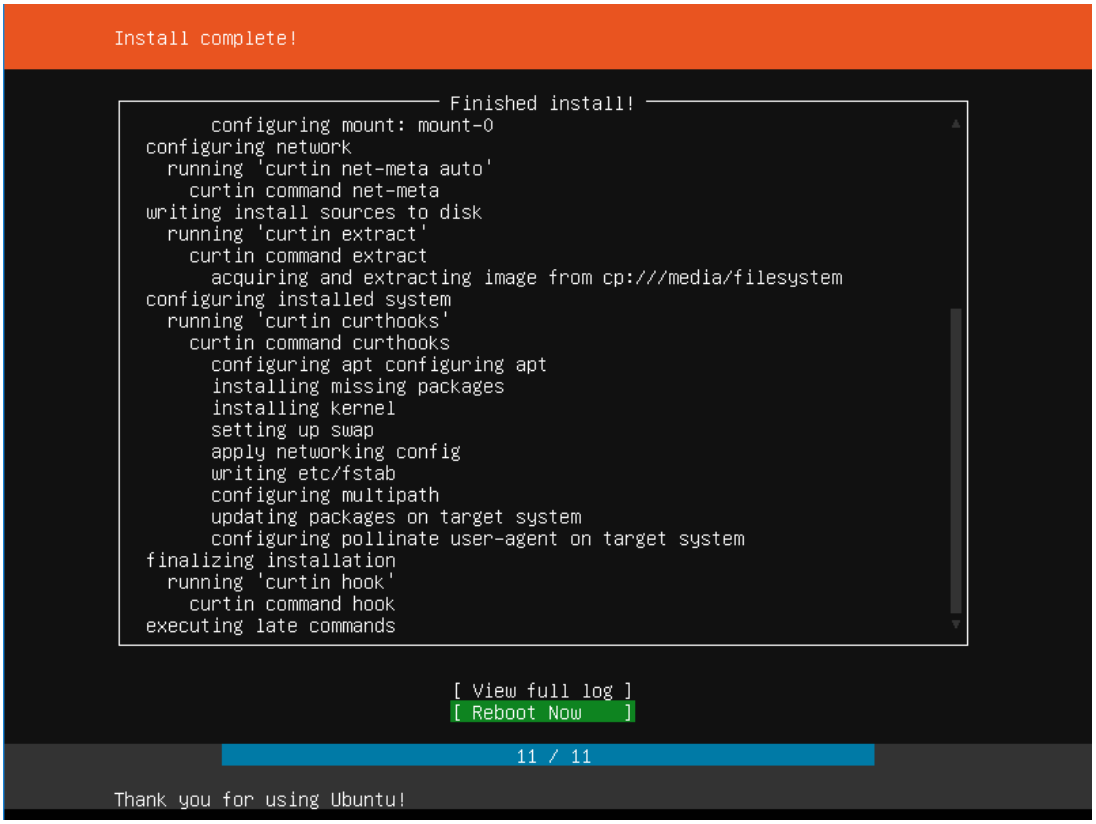


Рисунок 2.4 Перезагрузка системы в конце сценария установки

© Лартех, 2020 г. LoRa Alliance Member

2.2. Настроить ОС

2.2.1. Войти в систему под пользователем `lt-user`.



Рисунок 2.5 Приглашение на вход после перезагрузки системы

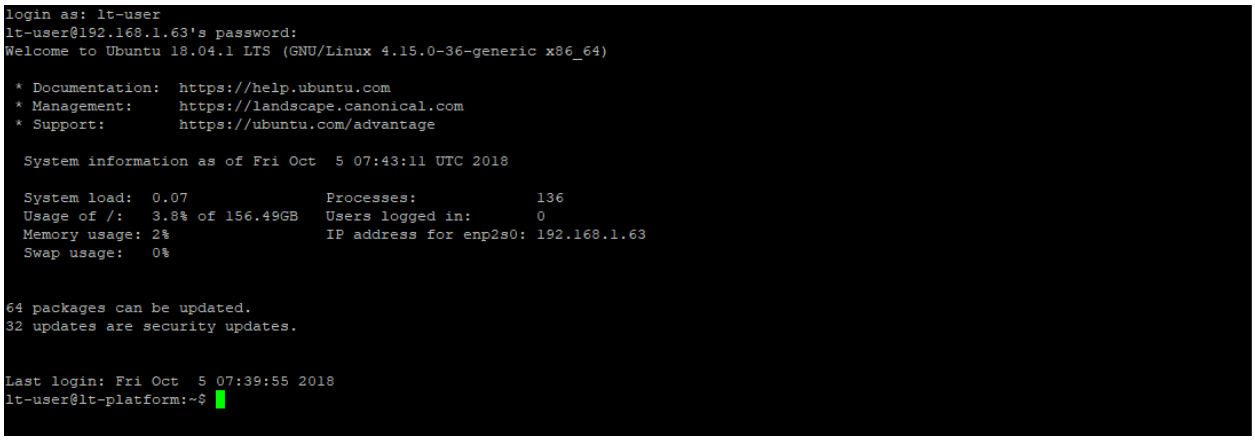


Рисунок 2.6 Состояние после успешного входа lt-user в систему

2.2.2. Перейти в режим суперпользователя.

```
sudo su -
```



Рисунок 2.7 Режим суперпользователься

2.2.3. Создать пользователя `ops`:

```
useradd -m -s /bin/bash -G sudo ops
echo "ops ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/ops
```

2.2.4. Установить пароль:

Примечание: в нашем случае был установлен пароль: `opspasswd`.

```
passwd ops
```

Примечание: для установки пароля потребуется дважды ввести новый пароль: `Enter new UNIX password` и `Retype new UNIX password`.

2.2.5. Отключить вход пользователю `root` по `ssh`:

```
sed -i 's/^#*PermitRootLogin\s\w.*$/PermitRootLogin no/'
/etc/ssh/sshd_config
```

2.2.6. Перезапустить `sshd`:

```
systemctl restart sshd
```

2.2.7. Завершить сеанс и войти под новым пользователем (`ops`).

Примечание: в дальнейшем все действия предполагается выполнять средствами `sudo` от имени этого пользователя.

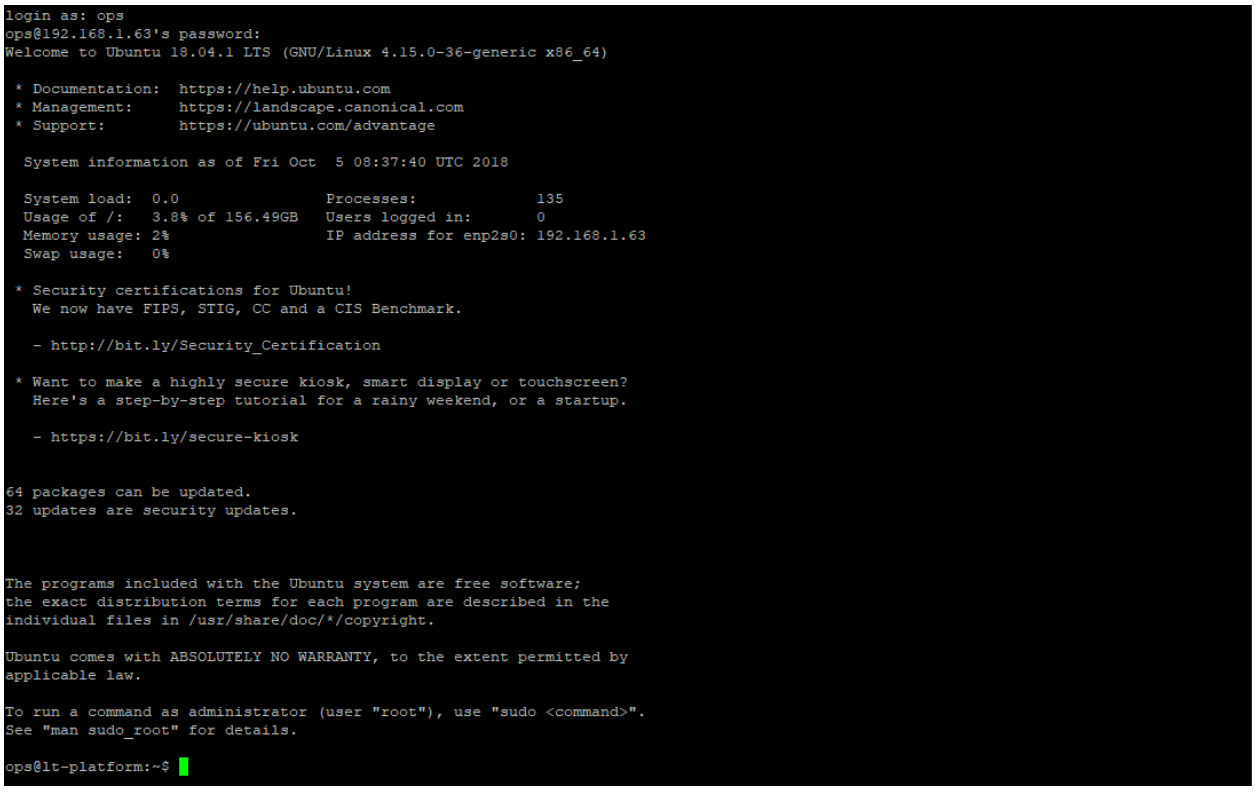


Рисунок 2.8 Состояние после успешного входа под пользователем ops

2.2.8. Сгенерировать `ssh`-ключ.

```
ssh-keygen -q -N "" -f $HOME/.ssh/id_rsa
```

```
cat $HOME/.ssh/id_rsa.pub > $HOME/.ssh/authorized_keys
```

2.2.9. Создать `$HOME/.ssh/config`.

```
cat <<EOF | tee $HOME/.ssh/config
AddKeysToAgent yes
ForwardAgent yes
User ops

Host kube-*
  Hostname %h
  StrictHostKeyChecking no

Host 192.168.100.1
  Hostname %h
  StrictHostKeyChecking no
EOF
```

2.2.10. Добавить следующие строки в файл репозиториев `/etc/apt/sources.list`.

```
deb http://ru.archive.ubuntu.com/ubuntu/ bionic universe
deb-src http://ru.archive.ubuntu.com/ubuntu/ bionic universe
deb http://ru.archive.ubuntu.com/ubuntu/ bionic-updates universe
deb-src http://ru.archive.ubuntu.com/ubuntu/ bionic-updates universe
```

```
sudo nano /etc/apt/sources.list
```

© Лартех, 2020 г. LoRa Alliance Member

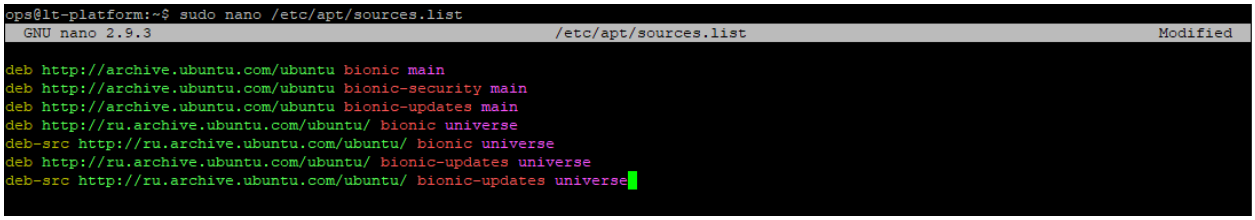


Рисунок 2.9 Содержимое файла /etc/apt/source.list после редактирования

2.2.11. Обновить локальный индекс пакетов до последних изменений в репозиториях.

```
sudo apt update
```

2.2.12. Установить утилиты.

```
sudo apt install --no-install-recommends -y \
  apt-transport-https \
  fail2ban \
  gnupg \
  software-properties-common \
  unzip
```

2.2.13. Отключить `IPv6`.

```
echo 'net.ipv6.conf.all.disable_ipv6 = 1' | sudo tee -a
/etc/sysctl.d/99-disable_ipv6.conf
```

2.2.14. Просмотреть установленные системные переменные.

```
sudo sysctl --system
```

© Лартех, 2020 г. LoRa Alliance Member

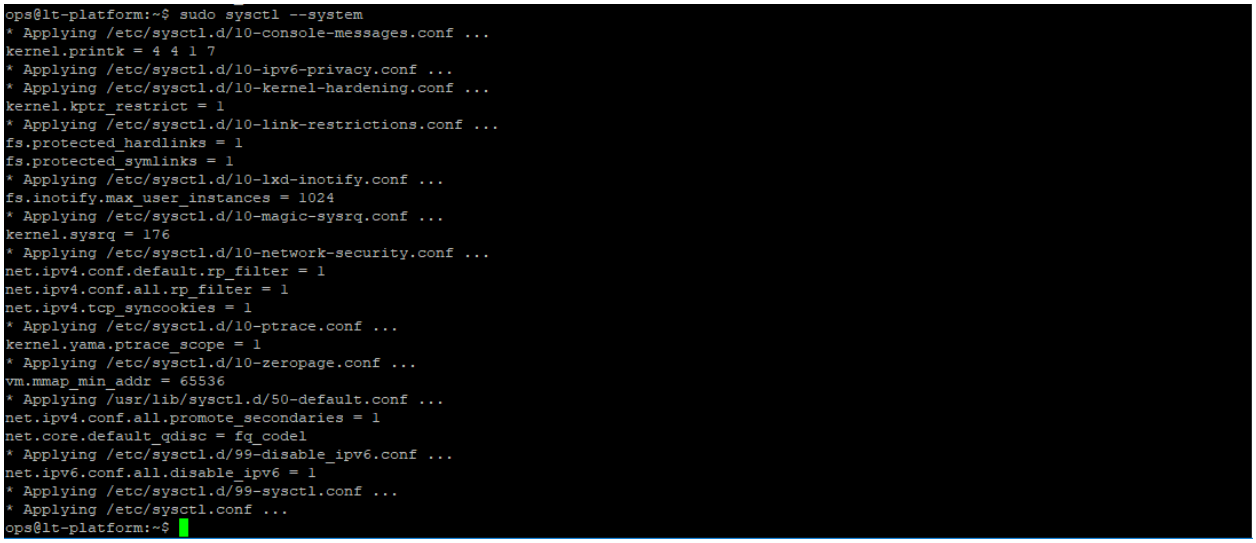


Рисунок 2.10 Установленные по факту настройки ОС переменные

2.3. Установить Libvirt

2.3.1. Установить необходимые пакеты:

```
sudo apt install -y \
  bridge-utils \
  dnsmasq \
  ebtables \
  genisoimage \
  libguestfs-tools \
  libosinfo-bin \
  libvirt-clients \
  libvirt-daemon-driver-storage-rbd \
  libvirt-daemon-system \
  python-guestfs \
  qemu-kvm \
  qemu-utils \
  virt-top \
  virtinst
```

Примечание: почему-то не не ставится пакет `nux-tools`, поэтому требуется его установить отдельной командой `sudo apt install nux-tools`.

2.3.2. Удалить дефолтную виртуальную сеть:

```
sudo virsh net-autostart --disable default
```

```
sudo virsh net-destroy default
```

```
sudo virsh net-undefine default
```

2.3.3. Создать `dummy`-интерфейс.

```
cat <<EOF | sudo tee /etc/systemd/network/10-dummy0.netdev
[NetDev]
Name=dummy0
Kind=dummy
EOF
```

```
cat <<EOF | sudo tee /etc/systemd/network/dummy0.network
[Match]
Name=dummy0

[Network]
Bridge=vbr0
EOF
```

2.3.4. Создать сетевой мост `vbr0`.

```
cat <<EOF | sudo tee /etc/systemd/network/vbr0.netdev
[NetDev]
Name=vbr0
Kind=bridge
EOF
```

```
cat <<EOF | sudo tee /etc/systemd/network/vbr0.network
[Match]
Name=vbr0

[Network]
Address=192.168.100.1/24
LinkLocalAddressing=no
IPv6AcceptRA=no
EOF
```

2.3.5. Перезапустить `systemd-networkd`.

```
sudo systemctl restart systemd-networkd
```

2.3.6. Включить пересылку пакетов между интерфейсами:

```
echo 'net.ipv4.ip_forward = 1' | sudo tee /etc/sysctl.d/99-ip_forwarding.conf
```

```
echo 'net.ipv4.conf.all.forwarding = 1' | sudo tee /etc/sysctl.d/99-ip_forwarding.conf
```

```
sudo sysctl --system
```

2.3.7. Добавить правила `iptables`.

```
sudo iptables -t filter -P INPUT ACCEPT
sudo iptables -t filter -P OUTPUT ACCEPT
sudo iptables -t filter -P FORWARD ACCEPT
sudo iptables -t filter -F
sudo iptables -t filter -A INPUT -i lo -j ACCEPT
sudo iptables -t filter -A INPUT -m conntrack --ctstate RE-
LATED,ESTABLISHED -j ACCEPT
sudo iptables -t filter -A INPUT -p ICMP -m conntrack --ctstate
NEW --icmp-type echo-request -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -m conntrack --ctstate
NEW --dport 22 -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -m conntrack --ctstate
NEW --dport 80 -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -m conntrack --ctstate
NEW --dport 443 -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -m conntrack --ctstate
NEW --dport 6443 -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -i vbr0 -m conntrack -
-ctstate NEW --dport 6789 -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -i vbr0 -m conntrack -
-ctstate NEW -m multiport --dports 6800:7300 -j ACCEPT
sudo iptables -t filter -A INPUT -p TCP -i vbr0 -m conntrack -
-ctstate NEW -m multiport --dports 53,67 -j ACCEPT
sudo iptables -t filter -A INPUT -p UDP -i vbr0 -m conntrack -
-ctstate NEW -m multiport --dports 53,67 -j ACCEPT
sudo iptables -t filter -P INPUT DROP
sudo iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -d
224.0.0.0/24 -j RETURN
sudo iptables -t nat -A POSTROUTING -s 192.168.100.0/24 -d
255.255.255.255/32 -j RETURN
sudo iptables -t nat -A POSTROUTING -s 192.168.100.0/24 ! -d
192.168.100.0/24 -j MASQUERADE
```

2.3.8. Установить пакет `iptables-persistent`.

```
sudo apt install -y iptables-persistent
```

Примечание: в ходе установки пакета потребуется дважды подтвердить необходимость сохранить: «`Save current IPv4 rules`» и «`Save current IPv6 rules`».

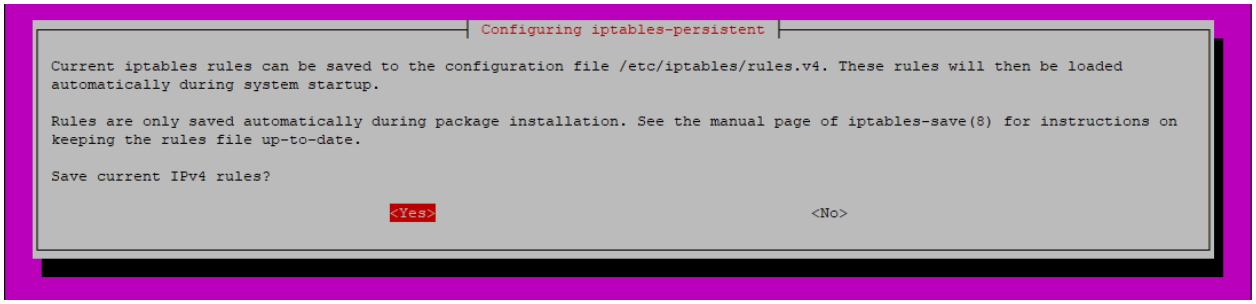


Рисунок 2.11 Запрос на подтверждение сохранение правил для IPv4

2.3.9. Добавить сервис `netfilter-persistent` в автозапуск и перезапустить его.

```
sudo systemctl enable netfilter-persistent
```

```
sudo systemctl restart netfilter-persistent
```

2.3.10. Сконфигурировать `dnsmasq`.

© Лартех, 2020 г. LoRa Alliance Member

```
cat <<EOF | sudo tee /etc/dnsmasq.conf
user=libvirt-dnsmasq
bind-interfaces
interface=vbr0
except-interface=lo
listen-address=127.0.0.1,192.168.100.1
server=1.1.1.1
server=1.0.0.1
dhcp-authoritative
dhcp-hostsfile=/var/lib/libvirt/dnsmasq/virt.hosts
dhcp-leasefile=/var/lib/libvirt/dnsmasq/virt.leases
dhcp-lease-max=1000
dhcp-no-override
dhcp-range=192.168.100.100,192.168.100.254
strict-order
addn-hosts=/var/lib/libvirt/dnsmasq/virt.addnhosts
EOF
```

```
sudo touch /var/lib/libvirt/dnsmasq/virt.addnhosts
sudo touch /var/lib/libvirt/dnsmasq/virt.hosts
sudo touch /var/lib/libvirt/dnsmasq/virt.leases
```

2.3.11. Добавить имя хоста `hostname` в `/etc/hosts`.

Примечание: `hostname` в нашем случае `lt-platform`.

```
sudo nano /etc/hosts
```

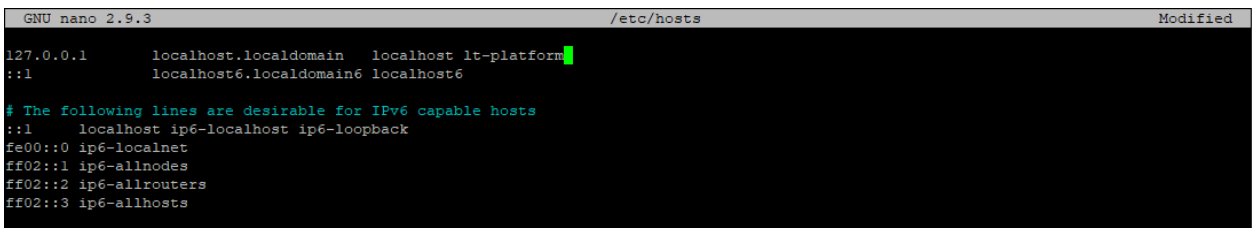


Рисунок 2.12 Содержимое файла /etc/hosts после редактирования

2.3.12. Отключить `systemd-resolved`.

```
sudo systemctl disable systemd-resolved
sudo systemctl stop systemd-resolved
sudo rm /etc/resolv.conf
```

2.3.13. Создать файл `/etc/resolv.conf`.

```
cat <<EOF | sudo tee /etc/resolv.conf
nameserver 127.0.0.1
nameserver 1.1.1.1
nameserver 1.0.0.1
EOF
```

2.3.14. Перезапустить `dnsmasq`.

```
sudo systemctl restart dnsmasq
```

2.4. Установить Ceph

2.4.1. Добавить ключ репозитория `Ceph`.

```
wget -q -O- "https://download.ceph.com/keys/release.asc" |
sudo apt-key add -
```

2.4.2. Добавить репозиторий `Ceph`.

```
echo "deb https://download.ceph.com/debian-mimic/ $(lsb_re-
lease -sc) main" | sudo tee /etc/apt/sources.list.d/ceph.list
```

2.4.3. Установить пакет `ceph-deploy`.

```
sudo apt update && sudo apt install -y ceph-deploy
```

2.4.4. Создать директорию, в которой будут храниться конфиги и ключи для `Ceph`.

```
mkdir $HOME/ceph-cluster && cd $HOME/ceph-cluster
```

2.4.5. Установить `Ceph`.

```
ceph-deploy install $HOSTNAME
```

2.4.6. Создать кластер:

```
ceph-deploy new $HOSTNAME:192.168.100.1
```

2.4.7. Добавить изменения в конфигурацию `Ceph`.

```
echo "public network = 192.168.100.0/24" >> ceph.conf
echo "osd crush chooseleaf type = 0" >> ceph.conf
echo "osd pool default size = 1" >> ceph.conf
```

2.4.8. Установить `Ceph Monitor`.

```
ceph-deploy mon create-initial
```

2.4.9. Запустить кластер:

```
ceph-deploy admin $HOSTNAME
```

2.4.10. Установить `Ceph Manager`.

```
ceph-deploy mgr create $HOSTNAME
```

2.4.11. Установить `Ceph OSD`.

```
ceph-deploy osd create --data /ceph $HOSTNAME
ceph-deploy osd create --data /dev/sdc $HOSTNAME
ceph-deploy osd create --data /dev/sdd $HOSTNAME
```

Примечание: в нашем случае необходимо изменить названия томов в соответствии с произведенной на этапе установки ОС разметке (раздел 2.1). Проверить называния томов можно командой `sudo lvs`.

```
ceph-deploy osd create --data ubuntu-vg/ceph-lv0 $HOSTNAME
ceph-deploy osd create --data ubuntu-vg/ceph-lv1 $HOSTNAME
ceph-deploy osd create --data ubuntu-vg/ceph-lv2 $HOSTNAME
```

2.4.12. Отредактировать `/etc/fstab`.

Примечание: необходимость этого шага связана с тем, что OSD тома Ceph имеют формат xfs и при загрузке ОС возникает ошибка при их монтировании.

```
sudo nano /etc/fstab
```

Примечание: необходимо закомментировать строки монтирования логических дисков `ceph`, `ceph1`, `ceph2`.

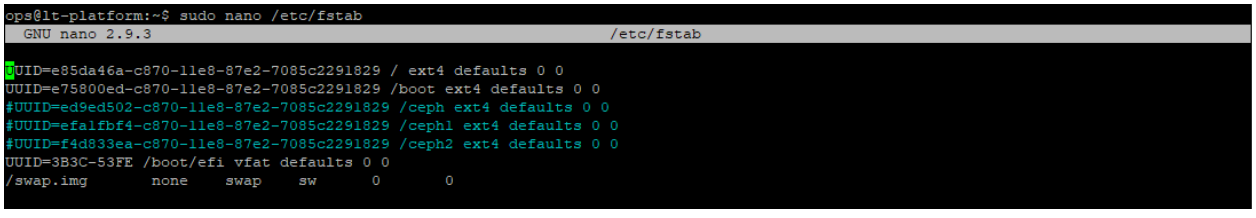


Рисунок 2.13 Содержимое файла /etc/fstab после редактирования

2.4.13. Перегрузить сервер `lt-platform`.

```
sudo shutdown -r now
```

2.4.14. Проверить статус `Ceph` кластера.

```
sudo ceph -s
```



Рисунок 2.14 Текущий старус Ceph кластера

2.5. Подключить Ceph к Libvirt

2.5.1. Создать пул для хранения образов виртуальных машин.

```
sudo ceph osd pool create libvirt 128 128
sudo ceph osd pool application enable libvirt libvirt
```

2.5.2. Создать `ceph`-пользователя.

```
sudo ceph auth get-or-create client.libvirt
```

Примечание: в результате будет создан ключ:

`[client.libvirt]`

`key = AQDdBr5bG3xgHRAAb+8FhAkgbcVQHG6clK9Q6g==`

```
sudo ceph auth caps client.libvirt \
  mon 'allow r' \
  osd 'allow class-read object_prefix rbd_children, allow rwx
  pool=libvirt'
```

2.5.3. Создать `libvirt`-секрет.

```
LIBVIRT_CEPH_AUTH_KEY=$(sudo ceph auth get-or-create cli-
ent.libvirt | grep -v '^\[' | awk '{print $3}')
LIBVIRT_CEPH_SECRET_UUID=$(uuidgen -r)
```

```
cat <<EOF | sudo tee /etc/libvirt/ceph-secret.xml
<secret ephemeral='no' private='no'>
  <uuid>${LIBVIRT_CEPH_SECRET_UUID}</uuid>
  <usage type='ceph'>
    <name>client.libvirt secret</name>
  </usage>
</secret>
EOF
```

```
sudo virsh secret-define --file /etc/libvirt/ceph-secret.xml
```

Примечание: в результате будет создан секрет:

```
Secret 1375363e-0982-491c-8e1b-5bda18710d70 created
```

```
sudo virsh secret-set-value "$LIBVIRT_CEPH_SECRET_UUID" "$LIB-
VIRT_CEPH_AUTH_KEY"
```

2.5.4. Создать `libvirt storage`-пул.

```
cat <<EOF | sudo tee /etc/libvirt/ceph-storage-pool.xml
<pool type="rbd">
  <name>ceph</name>
  <source>
    <name>libvirt</name>
    <host name='192.168.100.1' port='6789'/>
    <auth username='libvirt' type='ceph'>
      <secret uuid='${LIBVIRT_CEPH_SECRET_UUID}'/>
    </auth>
  </source>
</pool>
EOF
```

```
sudo virsh pool-define /etc/libvirt/ceph-storage-pool.xml
sudo virsh pool-autostart ceph
sudo virsh pool-start ceph
```

2.6. Создать виртуальные машины

2.6.1. Создать файл ответов `preseed.cfg` для автоустановки.

```
cat <<EOF | sudo tee /var/lib/libvirt/images/preseed.cfg
d-i debian-installer/language string en
d-i debian-installer/country string US
d-i debian-installer/locale string en_US.UTF-8
d-i localechooser/supported-locales multiselect ru_RU.UTF-8
d-i console-setup/ask_detect boolean false
d-i keyboard-configuration/xkb-keymap select us
d-i netcfg/choose_interface select auto
d-i netcfg/get_hostname string ubuntu
d-i netcfg/get_domain string localdomain
d-i mirror/country string manual
d-i mirror/http/hostname string mirror.yandex.ru
d-i mirror/http/directory string /ubuntu
d-i mirror/http/proxy string
d-i passwd/user-fullname string ops
d-i passwd/username string ops
d-i passwd/user-password password themostlystrongpassphraselol
d-i passwd/user-password-again password themostlystrongpass-
phraselol
d-i passwd/user-default-groups string audio cdrom video sudo
d-i user-setup/encrypt-home boolean false
d-i clock-setup/utc boolean true
d-i time/zone string Europe/Moscow
d-i clock-setup/ntp boolean true
d-i partman-auto/disk string /dev/vda
d-i partman-auto/method string lvm
d-i partman-lvm/device_remove_lvm boolean true
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true
d-i partman-auto-lvm/guided_size string max
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
d-i pkgsel/upgrade select full-upgrade
```

© Лартех, 2020 г. LoRa Alliance Member

```
d-i pkgsel/update-policy select unattended-upgrades
tasksel tasksel/first multiselect openssh-server
d-i pkgsel/include string acpid dbus dnsutils haveged htop net-
tools psmisc software-properties-common qemu-guest-agent
d-i grub-installer/only_debian boolean true
d-i finish-install/reboot_in_progress note
d-i debian-installer/exit/poweroff boolean true
tasksel tasksel/skip-tasks string standard
d-i pkgsel/install-language-support boolean false
d-i debian-installer/quiet boolean false
d-i debian-installer/splash boolean false
d-i grub-installer/timeout string 2
EOF
```

2.6.2. Создать виртуальные машины `kube-master-1/2/3`.

© Лартех, 2020 г. LoRa Alliance Member

```
for i in 1 2 3; do
sudo virt-install -n kube-master-${i} \
--vcpus 1 \
--ram 2048 \
--disk pool=ceph,size=16,bus=virtio \
--network bridge=vbr0,model=virtio \
--graphics vnc,listen=socket \
--console pty,target_type=serial \
--channel                        unix,mode=bind,target_type=vir-
tio,name=org.qemu.guest_agent.0 \
--location           https://mirror.yandex.ru/ubuntu/dists/bi-
onic/main/installer-amd64 \
--os-type=linux \
--os-variant=ubuntu17.10 \
--virt-type kvm \
--initrd-inject=/var/lib/libvirt/images/preseed.cfg \
--extra-args "file=file:/preseed.cfg" \
--extra-args "locale=en_US.UTF-8" \
--extra-args "keyboard-configuration/layoutcode=us" \
--extra-args "console-setup/ask_detect=false" \
--extra-args "console=ttyS0,115200n8 serial quiet" \
--connect qemu:///system \
--autostart \
--wait 0 \
--quiet
done
```

2.6.3. Создать виртуальные машины `kube-node-1/2/3`.

```
for i in 1 2 3; do
sudo virt-install -n kube-node-${i} \
--vcpus 1 \
--ram 6144 \
--disk pool=ceph,size=16,bus=virtio \
--network bridge=vbr0,model=virtio \
--graphics vnc,listen=socket \
--console pty,target_type=serial \
--channel                        unix,mode=bind,target_type=vir-
tio,name=org.qemu.guest_agent.0 \
--location           https://mirror.yandex.ru/ubuntu/dists/bi-
onic/main/installer-amd64 \
--os-type=linux \
--os-variant=ubuntu17.10 \
--virt-type kvm \
--initrd-inject=/var/lib/libvirt/images/preseed.cfg \
--extra-args "file=file:/preseed.cfg" \
--extra-args "locale=en_US.UTF-8" \
--extra-args "keyboard-configuration/layoutcode=us" \
--extra-args "console-setup/ask_detect=false" \
--extra-args "console=ttyS0,115200n8 serial quiet" \
--connect qemu:///system \
--autostart \
--wait 0 \
--quiet
done
```

Примечание: за недостатком памяти на машине `lt-platform`, изменил строку `--ram 6144` на `--ram 2048`.

2.6.4. Проверить состояние виртуальных машин.

```
sudo virsh list --all
```



Рисунок 2.15 Состояние виртуальных машин после успешной установки

Примечание: создание виртуальных машин (установка ОС) занимает от 20 до 60 минут. По окончанию установки ОС, виртуальные машины будут автоматически переведены в состоянии "`shut off`". Если ВМ находятся в состоянии running, то или еще ставится ОС или что-то пошло не так. Можно понаблюдать через монитор, что происходит на виртуалке (например, процесс установки ОС на ВМ): `sudo virsh console ИМЯ ВИРТУАЛКИ`.

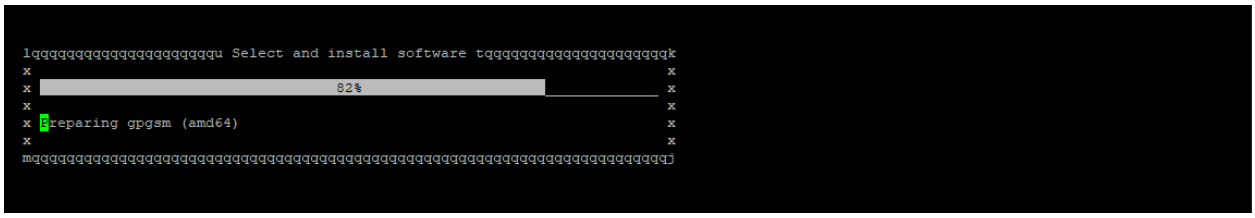


Рисунок 2.16 Отображение процесса установки ОС на ВМ через монитор

2.6.5. Пред-настроить все виртуальные машины.

```
for vm in $(sudo virsh list --state-shutoff --name); do
  sudo virt-customize -d ${vm} \
    --no-network \
    --hostname ${vm} \
    --timezone "Europe/Moscow" \
    --ssh-inject ops:file:$HOME/.ssh/id_rsa.pub \
    --write "/etc/sudoers.d/ops:ops ALL=(ALL) NOPASSWD: ALL"
done
```

2.6.6. Запустить все ВМ:

```
for vm in $(sudo virsh list --state-shutoff --name); do
sudo virsh start ${vm};
done
```

2.7. Установить Kubespray

2.7.1. Установить необходимые пакеты.

```
sudo apt install -y git python-pip sshpass
```

2.7.2. Загрузить репозиторий `kubespray`.

```
git clone https://github.com/kubernetes-incubator/kubespray.git $HOME/kubespray
```

```
cd $HOME/kubespray
```

```
git checkout -b mycluster
```

2.7.3. Установить зависимости.

```
pip install -r requirements.txt && source ~/.profile
```

2.7.4. Скопировать образец `inventory`-файла.

```
cp -rfp inventory/sample inventory/mycluster
```

2.7.5. Сгенерировать файл `hosts.ini`.

© Лартех, 2020 г. LoRa Alliance Member

```
echo "[all]" > inventory/mycluster/hosts.ini
```

```
for i in 1 2 3; do
echo "kube-master-${i} ansible_host=$(dig +short kube-master-${i})" >> inventory/mycluster/hosts.ini
done
```

```
for i in 1 2 3; do
echo "kube-node-${i} ansible_host=$(dig +short kube-node-${i})" >> inventory/mycluster/hosts.ini
done
```

```
cat <<EOF | tee -a inventory/mycluster/hosts.ini

[kube-master]
kube-master-1
kube-master-2
kube-master-3

[kube-node]
kube-node-1
kube-node-2
kube-node-3

[etcd]
kube-master-1
kube-master-2
kube-master-3

[k8s-cluster:children]
kube-node
kube-master
EOF
```

2.7.6. Отредактировать файл `inventory/mycluster/group_vars/all/all.yml`.

```
nano inventory/mycluster/group_vars/all/all.yml
```

Необходимо прописать следующие строки:

```
bootstrap_os: ubuntu
kubelet_load_modules: true
```

2.7.7. Отредактировать файл `inventory/mycluster/group_vars/all/docker.yml`:

```
nano inventory/mycluster/group_vars/all/docker.yml
```

Необходимо прописать следующие строки:

```
docker_storage_options: -s overlay2
```

2.7.8. Отредактировать файл `inventory/mycluster/group_vars/k8s-cluster/k8s-cluster.yml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/k8s-cluster.yml
```

Необходимо прописать следующие строки:

```
kube_api_anonymous_auth: false
kube_oidc_auth: false
kube_basic_auth: false
kube_network_plugin: flannel
dns_mode: coredns
kubeconfig_localhost: true
kubectl_localhost: true
supplementary_addresses_in_ssl_keys: [192.168.1.93]
```

© Лартех, 2020 г. LoRa Alliance Member

Примечание: `[ВНЕШНИЙ IP-АДРЕС СЕРВЕРА]` в настоящий момент `192.168.1.93`.

2.7.9. Отредактировать файл `inventory/mycluster/group_vars/k8s-cluster/k8s-net-flannel.yml`.

```
nano inventory/mycluster/group_vars/k8s-cluster/k8s-net-flannel.yml
```

Необходимо прописать следующие строки:

```
flannel_backend_type: "host-gw"
flannel_interface_regexp: '192\\.168\\.100\\.\\d{1,3}'
```

2.7.10. Отредактировать файл `inventory/mycluster/group_vars/k8s-cluster/addons.yml`.

```
nano inventory/mycluster/group_vars/k8s-cluster/addons.yml
```

Необходимо прописать следующие строки:

```
helm_enabled: true
ingress_nginx_enabled: true
ingress_nginx_host_network: true
ingress_nginx_configmap:
  server-tokens: "False"
  worker-shutdown-timeout: "180"
cert_manager_enabled: true
```

2.7.11. Запустить установку кластера:

```
ansible-playbook -i inventory/mycluster/hosts.ini cluster.yml -b
```

Примечание: установка кластера может занять от 20 до 60 минут. Надо дождаться, когда процесс установки завершится (появится подсказка в командной строке).

© Лартех, 2020 г. LoRa Alliance Member

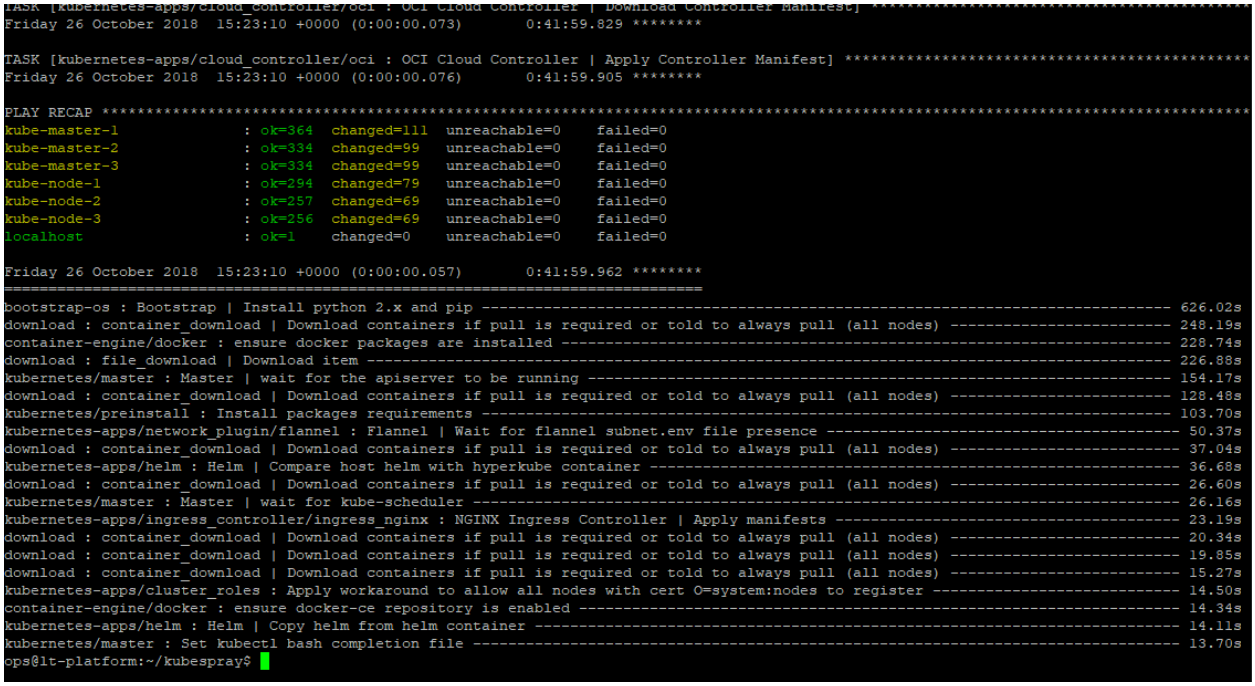


Рисунок 2.17 Процесс установки кластера завершился

2.7.12. Скопировать `kubeconfig` в домашнюю директорию.

```
mkdir -p $HOME/.kube && cp inventory/mycluster/artifacts/admin.conf $HOME/.kube/config
```

2.7.13. Скопировать `kubectl`.

```
sudo cp inventory/mycluster/artifacts/kubectl /usr/local/bin/
```

2.7.14. Проверить состояние `Kubernetes` кластера.

```
kubectl get nodes
```



Рисунок 2.18 Текущее состояние Kubernetes кластера

© Лартех, 2020 г. LoRa Alliance Member

2.7.15. Создать `rbd`-пул для кластера.

```
sudo ceph osd pool create kubernetes 128 128
sudo ceph osd pool application enable kubernetes kubernetes
```

2.7.16. Создать `ceph`-пользователя.

```
sudo ceph auth get-or-create client.kubernetes \
mon 'allow r, allow command "osd blacklist"' \
osd 'allow rwx pool=kubernetes'
```

Примечание: в результате будет создан ключ:

`[client.kubernetes]`

`key = AQDpHL9bDs+eAxAAj7eC5aJpmyYYwHCakKlKBw==`

2.7.17. Создать `kubernetes`-секреты для доступа к пулу.

```
cat <<EOF | kubectl create -f -
kind: Secret
apiVersion: v1
metadata:
 name: ceph-admin
 namespace: kube-system
type: kubernetes.io/rbd
data:
 key: "$(sudo ceph auth get-key client.admin | base64)"
EOF
```

Примечание: перед `name`, `namespace`, `key` должны быть пробелы (хотя бы один), иначе будет ошибка:

```
cat <<EOF | kubectl create -f -
kind: Secret
apiVersion: v1
metadata:
 name: ceph-kubernetes
 namespace: kube-system
type: kubernetes.io/rbd
data:
 key: "$(sudo ceph auth get-key client.kubernetes | base64)"
EOF
```

2.7.18. Создать `StorageClass`.

```
cat <<EOF | kubectl create -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: ceph
provisioner: kubernetes.io/rbd
parameters:
 monitors: 192.168.100.1:6789
 adminId: admin
 adminSecretName: ceph-admin
 adminSecretNamespace: kube-system
 pool: kubernetes
 userId: kubernetes
 userSecretName: ceph-kubernetes
 userSecretNamespace: kube-system
 imageFormat: "2"
 imageFeatures: layering
EOF
```

2.8. Настроить балансировщик для внешнего доступа

2.8.1. Установить `nginx` из официального репозитория.

```
wget -q -O- "http://nginx.org/keys/nginx_signing.key" | sudo
apt-key add -
```

```
echo "deb http://nginx.org/packages/ubuntu/ $(lsb_release -sc)
nginx" | sudo tee /etc/apt/sources.list.d/nginx.list
```

```
sudo apt update && sudo apt install -y nginx
```

2.8.2. Создать конфигурационный файл.

```
cat <<EOF | sudo tee /etc/nginx/nginx.conf
error_log stderr notice;

worker_processes auto;
events {
 multi_accept on;
 use epoll;
 worker_connections 1024;
}

stream {
 upstream kube_apiserver {
  least_conn;
  server $(dig @192.168.100.1 -t A +short kube-master-1):6443;
  server $(dig @192.168.100.1 -t A +short kube-master-2):6443;
  server $(dig @192.168.100.1 -t A +short kube-master-3):6443;
 }

 upstream ingress_http {
  least_conn;
  server $(dig @192.168.100.1 -t A +short kube-node-1):80;
  server $(dig @192.168.100.1 -t A +short kube-node-2):80;
  server $(dig @192.168.100.1 -t A +short kube-node-3):80;
 }

 upstream ingress_https {
  least_conn;
  server $(dig @192.168.100.1 -t A +short kube-node-1):443;
  server $(dig @192.168.100.1 -t A +short kube-node-2):443;
  server $(dig @192.168.100.1 -t A +short kube-node-3):443;
 }

 server {
  listen 6443;
```

© Лартех, 2020 г. LoRa Alliance Member

```
    proxy_pass kube_apiserver;
    proxy_timeout 10m;
    proxy_connect_timeout 1s;
   }

   server {
    listen 80;
    proxy_pass ingress_http;
    proxy_timeout 10m;
    proxy_connect_timeout 1s;
   }

   server {
    listen 443;
    proxy_pass ingress_https;
    proxy_timeout 10m;
    proxy_connect_timeout 1s;
   }
  }
  EOF
```

2.8.3. Перезапустить `nginx`.

```
sudo systemctl restart nginx
```

2.9. Установить на кластер дополнения

2.9.1. Для доступа к службе `dashboard` с правами администратора создать файл `dashboard-admin.yaml`.

```
nano inventory/mycluster/group_vars/k8s-cluster/dashboard-admin.yaml
```

Вставить в файл следующие строки:

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
  labels:
    k8s-app: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system
```

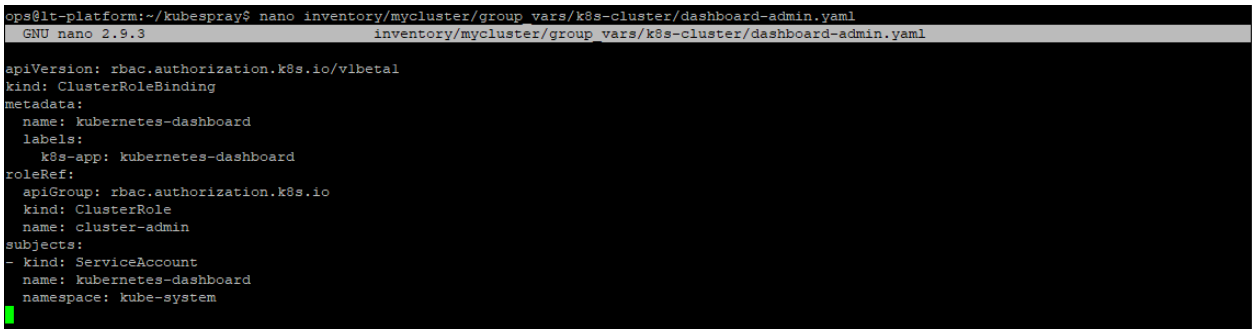


Рисунок 2.19 Содержимое файла dashboard-admin.yaml после редактирования

2.9.2. Применить права администратора к учетной записи службы `dashboard` из файла `dashboard-admin.yaml`.

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/dashboard-admin.yaml
```

2.9.3. Разрешить порт для `proxy`-сервер `kubernetes`.

```
sudo iptables -t filter -A INPUT -p TCP --dport 8001 -j ACCEPT
```

2.9.4. Запустить `proxy`-сервер `kubernetes` в фоновом режиме.

```
nohup kubectl proxy --address="192.168.1.93" -p 8001 --accept-hosts='^*$' &
```

© Лартех, 2020 г. LoRa Alliance Member

Примечание: в нашем случае адрес сервера, на котором развернута платформа: `192.168.1.93` и порт: `8001`.

```
ops@lt-platform:~/kubespray$ sudo kubectl proxy --address="192.168.1.93" -p 8001 --accept-hosts='^*$' &
[1] 56387
ops@lt-platform:~/kubespray$ Starting to serve on 192.168.1.93:8001

ops@lt-platform:~/kubespray$
```

Рисунок 2.20 Сообщение о запуске proxy-сервера Kubernetes

2.9.5. Проверить наличие доступа к панели управления `kubernetes`.

Примечание: в нашем случае необходимо в интернет-браузере набрать следующий URL: `http://192.168.1.93:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/`.

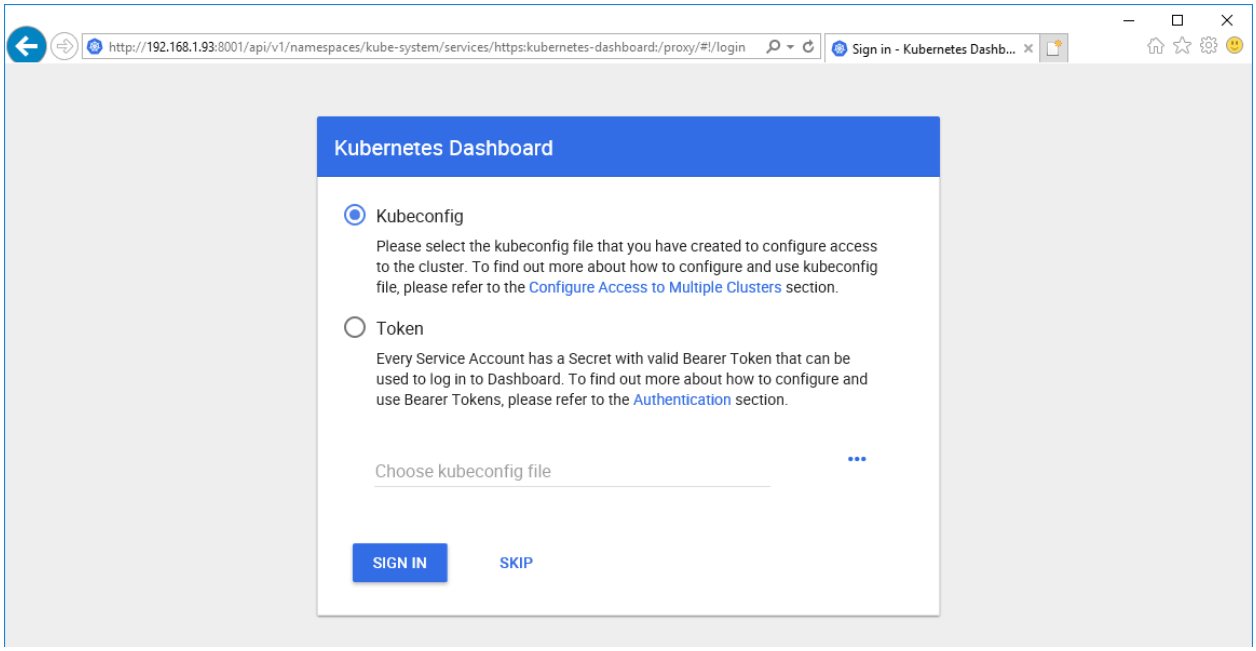


Рисунок 2.21 Панель управления Kubernetes (выбор типа авторизации)

2.9.6. Установить утилиту `kops` для управления `kubernetes`.

```
sudo snap install kops
```

2.9.7. Установить пакетный менеджер `helm`.

```
sudo snap install helm --classic
```

2.9.8. Установить дополнение для приема журналов регистрации системной информации `Elasticsearch` (Рисунок 2.19).

```
kubectl create -f https://raw.githubusercontent.com/kubernetes/kops/master/addons/logging-elasticsearch/v1.5.0.yaml
```

2.9.9. Установить дополнение для базового мониторинга кластера и интеграции метрик в инструментальную панель `Kubernetes` (Рисунок 2.19).

```
kubectl create -f https://raw.githubusercontent.com/kubernetes/kops/master/addons/monitoring-standalone/v1.6.0.yaml
```

2.9.10. Установить `Ingress` контроллер для `NGINX` (Рисунок 2.19).

```
kubectl create -f https://raw.githubusercontent.com/kubernetes/kops/master/addons/ingress-nginx/v1.6.0.yaml
```

2.9.11. Отобразить (показать) установленные на кластере сервисы (Рисунок 2.20):

```
kubectl -n kube-system get pods
```

© Лартех, 2020 г. LoRa Alliance Member

```
ops@lt-platform:~/kubespray$ kubectl -n kube-system get pods
NAME                                    READY   STATUS              RESTARTS   AGE
coredns-f88b9bbc6-sldnl                 1/1     Running             0          5d2h
coredns-f88b9bbc6-tfrg2                 1/1     Running             0          5d2h
elasticsearch-logging-0                 0/1     Pending             0          2m9s
fluentd-es-d8tfx                        0/1     ContainerCreating   0          2m12s
fluentd-es-dm7mq                        0/1     ContainerCreating   0          2m13s
fluentd-es-sg9k6                        0/1     ContainerCreating   0          2m12s
heapster-75bf65c585-cwhg7               0/2     ContainerCreating   0          53s
kibana-logging-7c544cbf6f-4r77j         0/1     ContainerCreating   0          2m1s
kube-apiserver-kube-master-1            1/1     Running             1          5d2h
kube-apiserver-kube-master-2            1/1     Running             1          5d2h
kube-apiserver-kube-master-3            1/1     Running             2          5d2h
kube-controller-manager-kube-master-1   1/1     Running             28         5d2h
kube-controller-manager-kube-master-2   1/1     Running             31         5d2h
kube-flannel-2k771                      2/2     Running             0          5d2h
kube-flannel-954qk                      2/2     Running             0          5d2h
kube-flannel-bnt9z                      2/2     Running             0          5d2h
kube-flannel-fcpkn                      2/2     Running             0          5d2h
kube-flannel-wc89c                      2/2     Running             0          5d2h
kube-flannel-zqmg7                      2/2     Running             0          5d2h
kube-proxy-kube-master-1                1/1     Running             0          5d2h
kube-proxy-kube-master-2                1/1     Running             1          5d2h
kube-proxy-kube-master-3                1/1     Running             1          5d2h
kube-proxy-kube-node-1                  1/1     Running             2          5d2h
kube-proxy-kube-node-2                  1/1     Running             1          5d2h
kube-proxy-kube-node-3                  1/1     Running             1          5d2h
kube-scheduler-kube-master-1            0/1     Error               28         5d2h
kube-scheduler-kube-master-2            1/1     Running             31         5d2h
kubernetes-dashboard-5db4d9f45f-hjqtp   1/1     Running             0          5d2h
nginx-proxy-kube-node-1                 1/1     Running             0          5d2h
nginx-proxy-kube-node-2                 1/1     Running             0          5d2h
nginx-proxy-kube-node-3                 1/1     Running             0          5d2h
tiller-deploy-694dc94c65-x76kd          1/1     Running             0          5d2h
ops@lt-platform:~/kubespray$
```

Рисунок 2.22 Установленные на кластере сервисы

2.9.12. Установить `Tiller` в `Kubernetes` кластере (Рисунок 2.21).

```
helm init --upgrade
```

Примечание: с установкой `Tiller` будут созданы все необходимые конфигурации.

2.9.13. Проверить корректность установки `Tiller`.

```
helm list
```

```
ops@lt-platform:~/kubespray$ helm list
Error: could not find a ready tiller pod
ops@lt-platform:~/kubespray$ helm list
ops@lt-platform:~/kubespray$
```

Рисунок 2.23 Результат проверки корректности установки Tiller

Примечание: если отображается `Error: could not find a ready tiller pod`, значит `tiller pod` создается, необходимо подождать и снова выполнить команду `helm list`.

3. Установка LPWAN.DLMS

3.1. Установить базовые компоненты

3.1.1. Установить `PostgresPro`.

3.1.1.1. Создать `postgres-configmap.yaml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/postgres-configmap.yaml
```

Вставить в файл следующие строки:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-config
  labels:
    app: postgres
data:
  POSTGRES_DB: postgresdb
  POSTGRES_USER: postgresadmin
  POSTGRES_PASSWORD: admin123
```

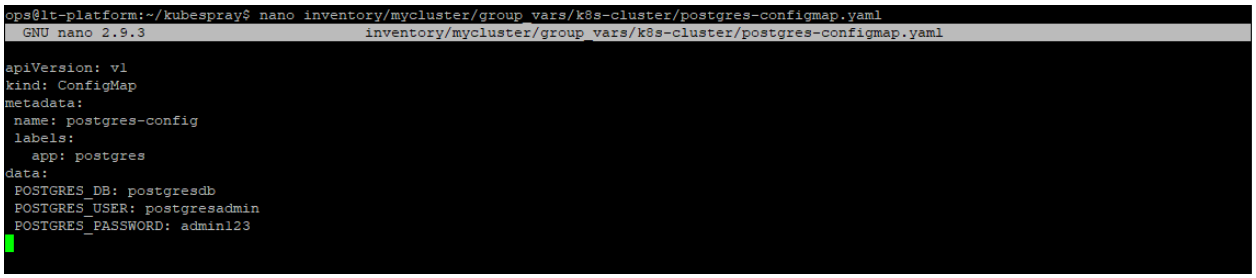


Рисунок 2.1 Содержимое файла postgres-configmap.yaml после редактирования

3.1.1.2. Создать `ConfigMap`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/postgres-configmap.yaml
```

3.1.1.3. Создать `postgres-storage.yaml`:

```
nano      inventory/mycluster/group_vars/k8s-cluster/postgres-
storage.yaml
```

Вставить в файл следующие строки:

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: postgres-pv-volume
  labels:
    type: local
    app: postgres
spec:
  storageClassName: ceph
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: postgres-pv-claim
  labels:
    app: postgres
spec:
  storageClassName: ceph
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

© Лартех, 2020 г. LoRa Alliance Member

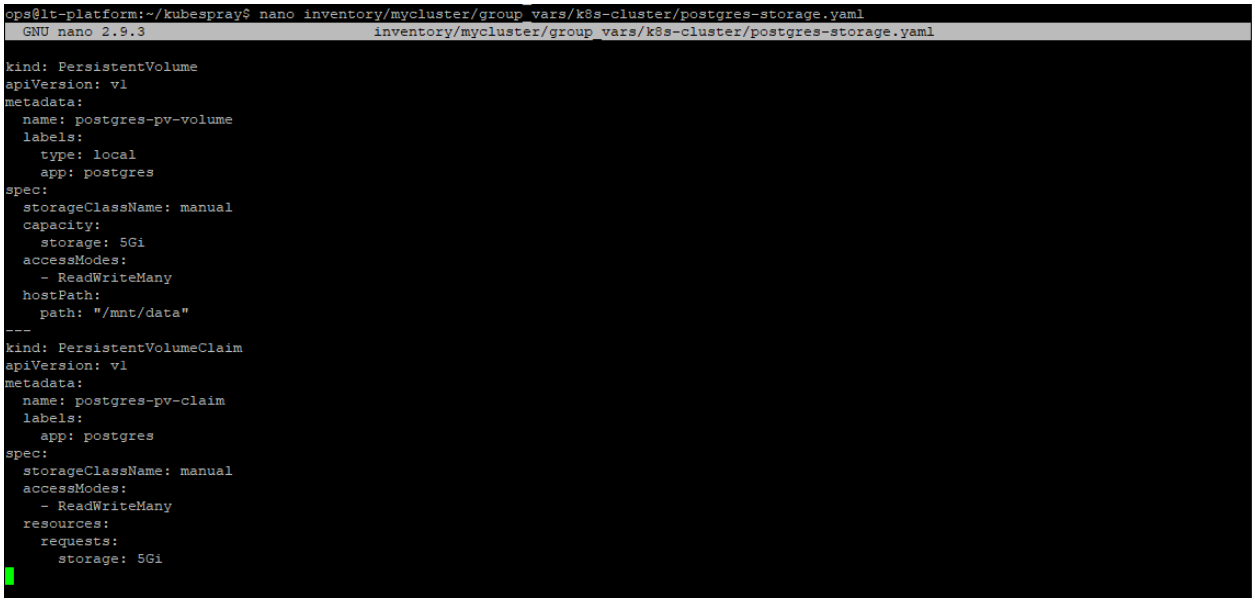


Рисунок 2.2 Содержимое файла postgres-storage.yaml после редактирования

3.1.1.4. Создать `PersistentVolume`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/postgres-storage.yaml
```

3.1.1.5. Создать `postgres-deployment.yaml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/postgres-deployment.yaml
```

Вставить в файл следующие строки:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:10.4
```

© Лартех, 2020 г. LoRa Alliance Member

```
          imagePullPolicy: "IfNotPresent"
          ports:
            - containerPort: 5432
          envFrom:
            - configMapRef:
                name: postgres-config
          volumeMounts:
            - mountPath: /var/lib/postgrespro/data
              name: postgredb
      volumes:
        - name: postgredb
          persistentVolumeClaim:
            claimName: postgres-pv-claim
```

```
ops@lt-platform:~/kubespray$ nano inventory/mycluster/group_vars/k8s-cluster/postgres-deployment.yaml
  GNU nano 2.9.3                    inventory/mycluster/group_vars/k8s-cluster/postgres-deployment.yaml

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgres
          image: postgres:10.4
          imagePullPolicy: "IfNotPresent"
          ports:
            - containerPort: 5432
          envFrom:
            - configMapRef:
                name: postgres-config
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: postgredb
      volumes:
        - name: postgredb
          persistentVolumeClaim:
            claimName: postgres-pv-claim
```

Рисунок 2.3 Содержимое файла postgres-deployment.yaml после редактирования

3.1.1.6. Создать `Deployment`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/postgres-deployment.yaml
```

3.1.1.7. Создать `postgres-service.yaml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/postgres-service.yaml
```

Вставить в файл следующие строки:

```
apiVersion: v1
```

```
kind: Service
metadata:
  name: postgres
  labels:
    app: postgres
spec:
  type: NodePort
  ports:
    - port: 5432
  selector:
    app: postgres
```

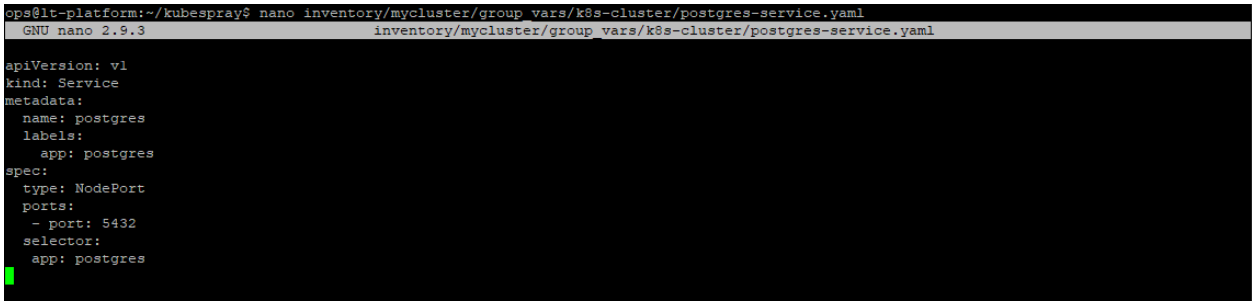


Рисунок 2.4 Содержимое файла postgres-service.yaml после редактирования

3.1.1.8. Создать `Service`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/postgres-service.yaml
```

3.1.1.9. Убедиться, что сервис `postgres` создан:

```
kubectl get services
```

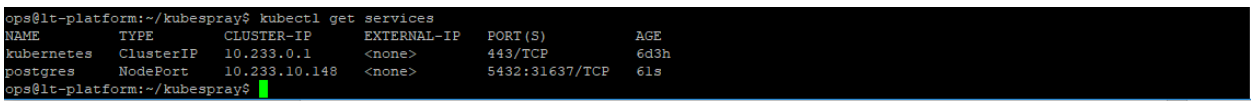


Рисунок 2.5 Результат проверки наличия сервиса PostgresPro

3.1.1.10. Установить `postgres` клиента:

```
sudo apt-get install postgrespro-client
```

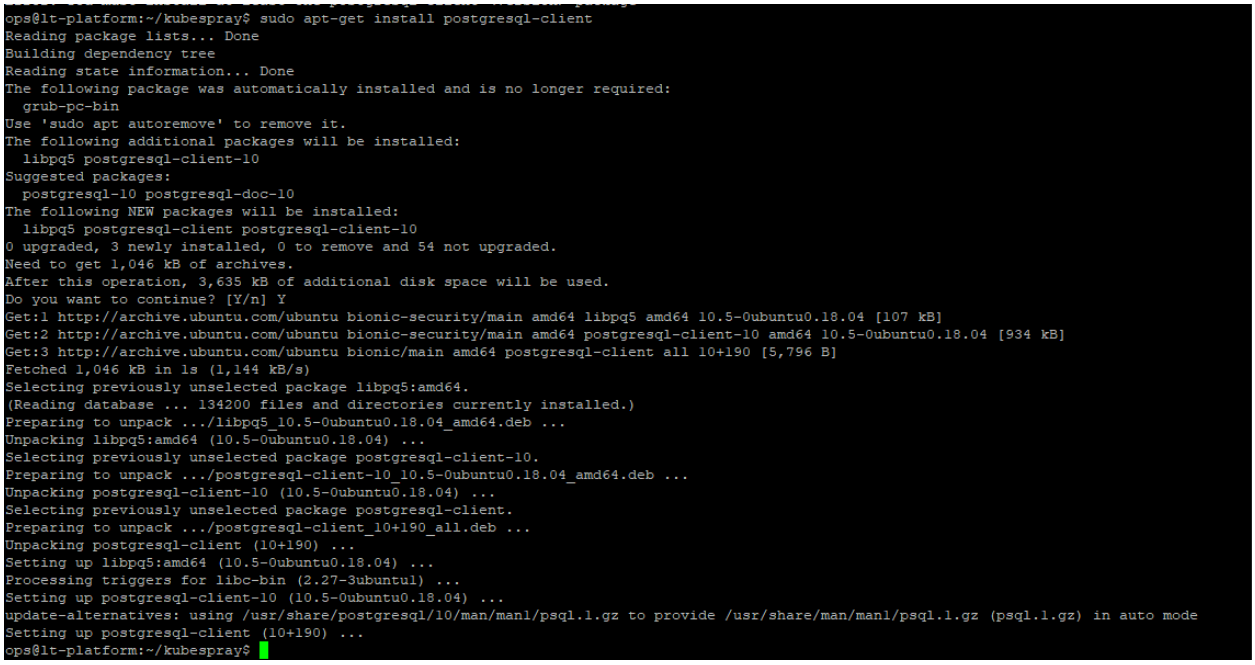


Рисунок 2.6 Результат установки клиента Postgres Pro

3.1.1.11. Посмотреть имя установленного контейнера `postgres`:

```
kubectl get pods --output=wide
```

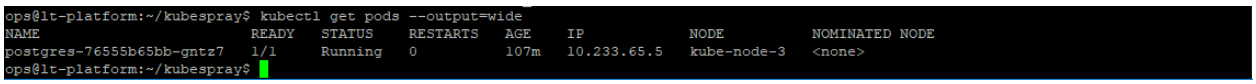


Рисунок 2.7 Просмотр сведений о запущенном контейнере Postgres Pro

Примечание: в нашем случае имя контейнера `postgres-76555b65bb-gntz7`.

3.1.1.12. Пробросить порт для доступа к контейнеру `postgres`:

```
kubectl port-forward postgres-76555b65bb-gntz7 5432:5432 &
```

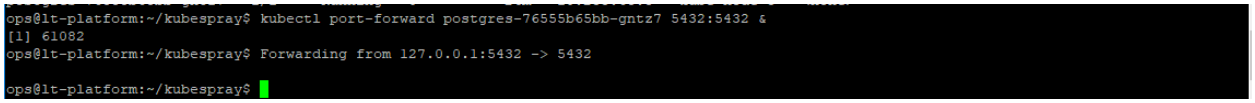


Рисунок 2.8 Сообщение об успешной переадресации запросов на порт Postgres Pro

© Лартех, 2020 г. LoRa Alliance Member

3.1.1.13. Проверить доступ к `postgrespro`:

```
psql -h localhost -p 5432 -U postgresadmin -d postgresdb
```

Примечание: в нашем случае имя базы данных `postgresdb` и имя пользователя `postgresadmin`.

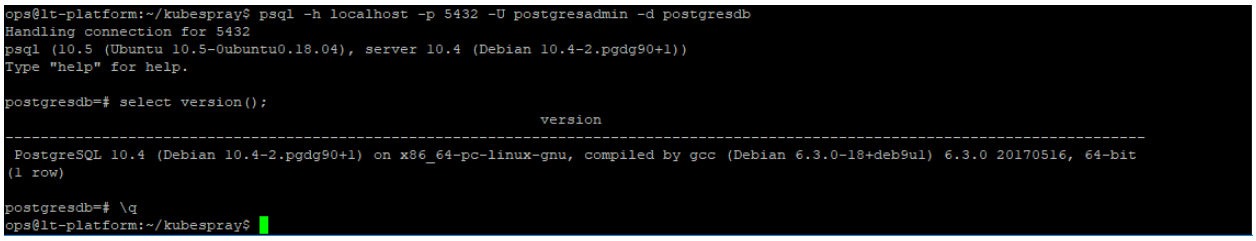


Рисунок 2.9 Результат проверки доступа клиента к Postgres Pro

Примечание: для проверки работоспособности `postgres` выполняем запрос установленной версии СУБД `select version();`.

3.1.2. Установить `Redis`.

3.1.2.1. Создать `redis-configmap.yaml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/redis-config-
map.yaml
```

Вставить в файл следующие строки:

```
apiVersion: v1
data:
  redis-config: |
    maxmemory 2mb
    maxmemory-policy allkeys-lru
kind: ConfigMap
metadata:
  creationTimestamp: 2016-03-30T18:14:41Z
  name: example-redis-config
  namespace: default
  resourceVersion: "24686"
  selfLink: /api/v1/namespaces/default/configmaps/example-redis-config
  uid: 460a2b6e-f6a3-11e5-8ae5-42010af00002
```



Рисунок 2.10 Содержимое файла redis-configmap.yaml после редактирования

3.1.2.2. Создать `redis-configmap`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/redis-configmap.yaml
```

3.1.2.3. Создать `redis-storage.yaml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/redis-storage.yaml
```

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: redis-pv-volume
  labels:
    type: local
    app: redis
spec:
  storageClassName: ceph
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-pv-claim
  labels:
    app: redis
```

```
spec:
  storageClassName: ceph
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

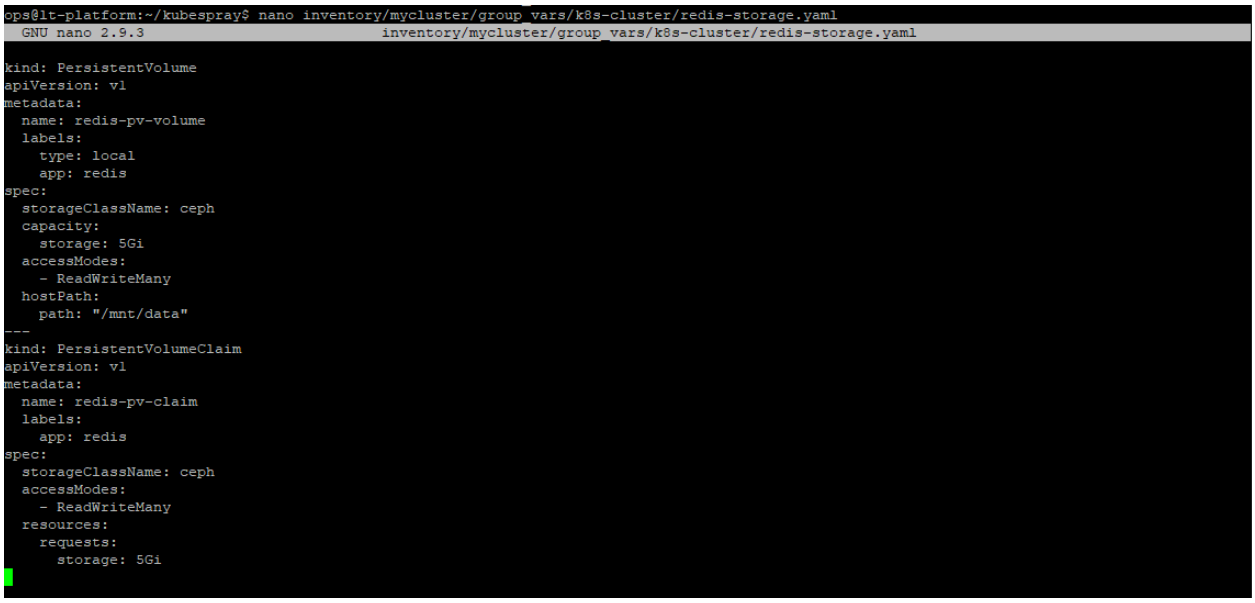


Рисунок 2.11 Содержимое файла redis-configmap.yaml после редактирования

3.1.2.4. Создать `redis-storage`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/redis-storage.yaml
```

3.1.2.5. Создать `redis-pod.yaml`:

```
nano inventory/mycluster/group_vars/k8s-cluster/redis-pod.yaml
```

Вставить в файл следующие строки:

```
apiVersion: v1
kind: Pod
metadata:
  name: redis
```

```
spec:
  containers:
  - name: redis
    image: kubernetes/redis:v1
    env:
    - name: MASTER
      value: "true"
    ports:
    - containerPort: 6379
    resources:
      limits:
        cpu: "0.1"
    volumeMounts:
    - mountPath: /redis-master-data
      name: data
    - mountPath: /redis-master
      name: config
  volumes:
    - name: data
      emptyDir: {}
    - name: config
      configMap:
        name: example-redis-config
        items:
        - key: redis-config
          path: redis.conf
```

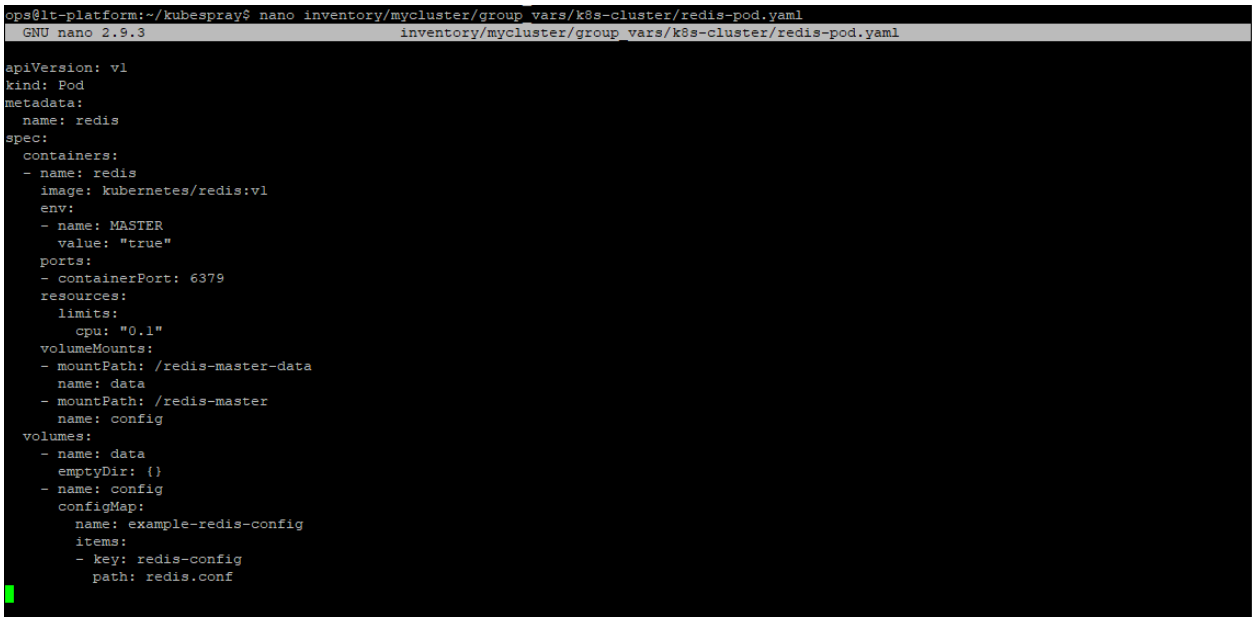


Рисунок 2.12 Содержимое файла redis-pod.yaml после редактирования

3.1.2.6. Создать `redis-pod`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/redis-pod.yaml
```

Примечание: надо подождать пока запустится `pod redis`, проверяем `kubectl get pods --output=wide`.

```
ops@lt-platform:~/kubespray$ kubectl get pods --output=wide
NAME                                    READY   STATUS    RESTARTS   AGE    IP             NODE          NOMINATED NODE
kafka-broker0-5566b4cfd5-t95dl          1/1     Running   0          3h5m   10.233.64.7    kube-node-2   <none>
postgres-76555b65bb-gntz7               1/1     Running   0          23h    10.233.65.5    kube-node-3   <none>
redis                                   1/1     Running   0          13m    10.233.64.8    kube-node-2   <none>
zookeeper-deployment-1-6d858cbdfd-qlgq8 1/1     Running   0          20h    10.233.65.6    kube-node-3   <none>
ops@lt-platform:~/kubespray$
```

Рисунок 2.13 Результат успешного запуска контейнера Redis

3.1.2.7. Для проверки работоспособности запустить клиента `Redis`:

```
kubectl exec -it redis redis-cli
```

```
ops@lt-platform:~/kubespray$ kubectl exec -it redis
error: expected 'exec POD_NAME COMMAND [ARG1] [ARG2] ... [ARGN]'.
POD_NAME and COMMAND are required arguments for the exec command
See 'kubectl exec -h' for help and examples.
ops@lt-platform:~/kubespray$ kubectl exec -it redis redis-cli
127.0.0.1:6379> CONFIG GET maxmemory
1) "maxmemory"
2) "2097152"
127.0.0.1:6379> CONFIG GET maxmemory-policy
1) "maxmemory-policy"
2) "allkeys-lru"
127.0.0.1:6379> quit
ops@lt-platform:~/kubespray$
```

Рисунок 2.14 Результат проверки доступа клиента к Redis

Примечание: после запуска клиента `redis` выполняем следующие запросы: `CONFIG GET maxmemory` и `CONFIG GET maxmemory-policy`.

3.1.3. Установить `Kafka`.

3.1.3.1. Создать файл с именем `kafka-namespace.yml` и добавить следующее содержимое:

```
nano inventory/mycluster/group_vars/k8s-cluster/kafka-namespace.yml
```

Вставить в файл следующие строки:

```
apiVersion: v1
kind: Namespace
metadata:
  name: kafka-project
```



Рисунок 2.15 Содержимое файла kafka-namespace.yml после редактирования

3.1.3.2. Создать пространство имен для установки `Kafka`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/kafka-namespace.yml
```

3.1.3.3. Создать файл с именем `zookeeper.yml` и добавьте это содержимое:

```
nano inventory/mycluster/group_vars/k8s-cluster/zookeeper.yml
```

Вставить в файл следующие строки:

```
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: zookeeper-deployment-1
  namespace: kafka-project
spec:
  template:
    metadata:
      labels:
        app: zookeeper-1
    spec:
      containers:
      - name: zoo1
        image: digitalwonderland/zookeeper
        ports:
        - containerPort: 2181
        env:
        - name: ZOOKEEPER_ID
```

© Лартех, 2020 г. LoRa Alliance Member

```
          value: "1"
        - name: ZOOKEEPER_SERVER_1
          value: zoo1
---
apiVersion: v1
kind: Service
metadata:
  name: zoo1
  namespace: kafka-project
  labels:
    app: zookeeper-1
spec:
  ports:
  - name: client
    port: 2181
    protocol: TCP
  - name: follower
    port: 2888
    protocol: TCP
  - name: leader
    port: 3888
    protocol: TCP
  selector:
    app: zookeeper-1
```

```
ops@lt-platform:~/kubespray$ nano inventory/mycluster/group_vars/k8s-cluster/zookeeper.yml
  GNU nano 2.9.3                    inventory/mycluster/group_vars/k8s-cluster/zookeeper.yml

---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: zookeeper-deployment-1
  namespace: kafka-project
spec:
  template:
    metadata:
      labels:
        app: zookeeper-1
    spec:
      containers:
      - name: zoo1
        image: digitalwonderland/zookeeper
        ports:
        - containerPort: 2181
        env:
        - name: ZOOKEEPER_ID
          value: "1"
        - name: ZOOKEEPER_SERVER_1
          value: zoo1
---
apiVersion: v1
kind: Service
metadata:
  name: zoo1
  namespace: kafka-project
  labels:
    app: zookeeper-1
spec:
  ports:
  - name: client
    port: 2181
    protocol: TCP
  - name: follower
    port: 2888
    protocol: TCP
  - name: leader
    port: 3888
    protocol: TCP
  selector:
    app: zookeeper-1
```

Рисунок 2.16 Содержимое файла zookeeper.yml после редактирования

© Лартех, 2020 г. LoRa Alliance Member

3.1.3.4. Установить `zookeeper`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/zookeeper.yml
```

3.1.3.5. Проверить результат установки `zookeeper`:

```
kubectl get deployments --namespace kafka-project
```

```
kubectl get svc --namespace kafka-project
```

```
ops@lt-platform:~/kubespray$ kubectl get deployments --namespace kafka-project
NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
zookeeper-deployment-1   1         1         1            1           2m
ops@lt-platform:~/kubespray$ kubectl get svc --namespace kafka-project
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
zoo1    ClusterIP   10.233.37.148   <none>        2181/TCP,2888/TCP,3888/TCP   2m8s
ops@lt-platform:~/kubespray$
```

Рисунок 2.17 Результат успешной установки Zookeeper

Примечание: в нашем случае имя контейнера Zookeeper `zoo1`.

3.1.3.6. Создать файл с именем `kafka-service.yml` и добавьте это содержимое:

```
nano inventory/mycluster/group_vars/k8s-cluster/kafka-service.yml
```

Вставить в файл следующие строки:

```
---
apiVersion: v1
kind: Service
metadata:
  name: kafka-service
  namespace: kafka-project
  labels:
    name: kafka
spec:
  ports:
  - port: 9092
```

```
    name: kafka-port
    protocol: TCP
  selector:
    app: kafka
    id: "0"
  type: LoadBalancer
```

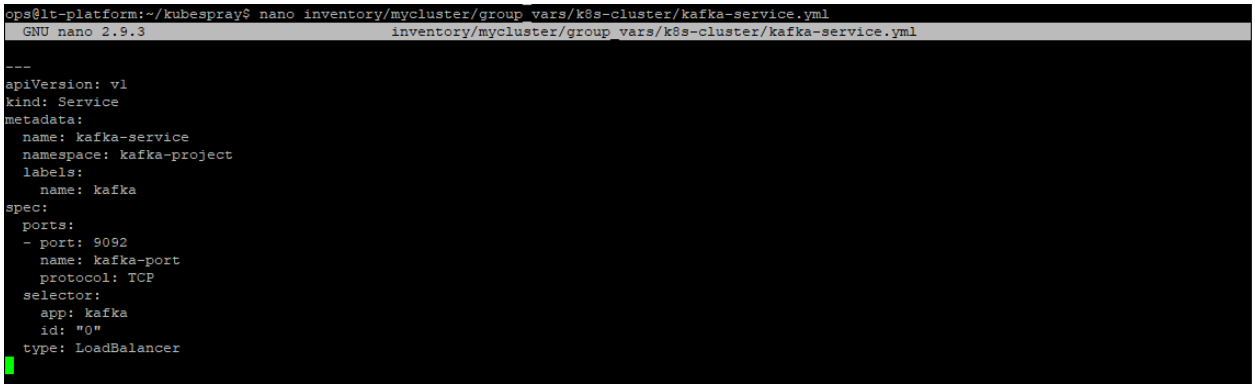


Рисунок 2.18 Содержимое файла kafka-service.yml после редактирования

3.1.3.7. Установить `kafka-service`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/kafka-service.yml
```

3.1.3.8. Проверить результат установки `kafka-service`:

```
kubectl get svc --namespace kafka-project
```

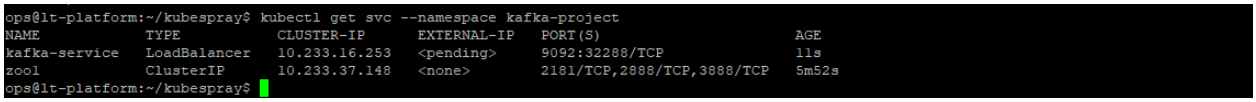


Рисунок 2.19 Результат успешной установки сервиса Kafka

3.1.3.9. Создать файл с именем `kafka-broker.yml` и добавьте это содержимое:

```
nano inventory/mycluster/group_vars/k8s-cluster/kafka-broker.yml
```

Вставить в файл следующие строки:

```
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: kafka-broker0
  namespace: kafka-project
spec:
  template:
    metadata:
      labels:
        app: kafka
        id: "0"
    spec:
      containers:
      - name: kafka
        image: wurstmeister/kafka
        ports:
        - containerPort: 9092
        env:
        - name: KAFKA_ADVERTISED_PORT
          value: "9092"
        - name: KAFKA_ADVERTISED_HOST_NAME
          value: 127.0.0.1
        - name: KAFKA_ZOOKEEPER_CONNECT
          value: zoo1:2181
        - name: KAFKA_BROKER_ID
          value: "0"
        - name: KAFKA_CREATE_TOPICS
          value: admintome-test:1:1
```

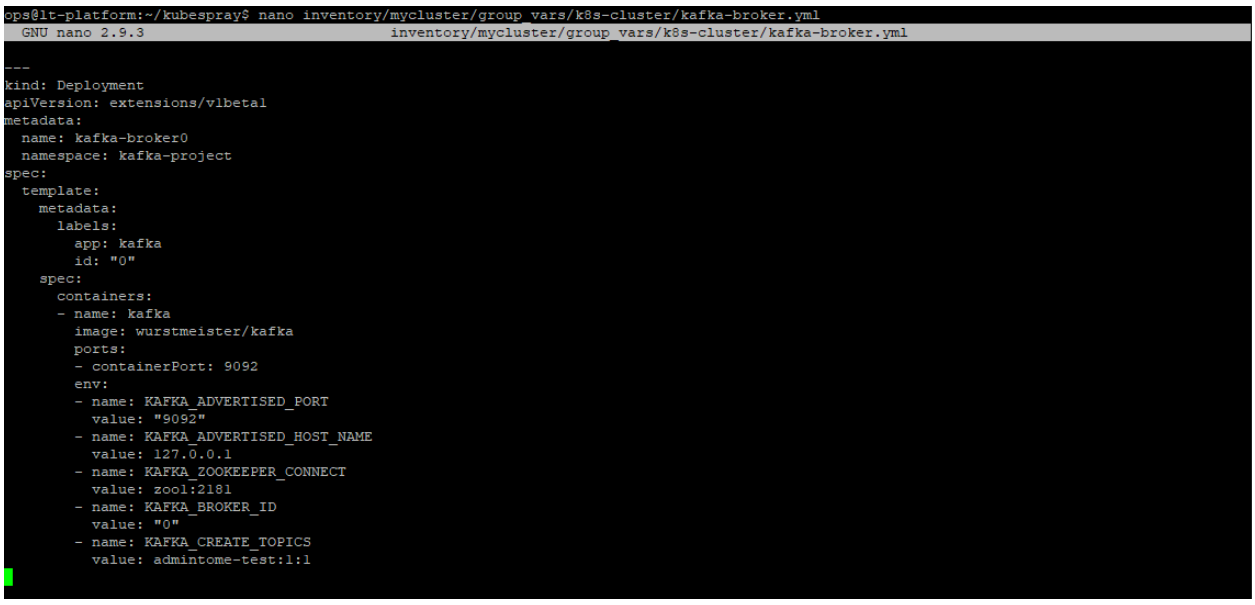


Рисунок 2.20 Содержимое файла kafka-broker.yml после редактирования

© Лартех, 2020 г. LoRa Alliance Member

3.1.3.10. Создать `kafka-broker`:

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/kafka-broker.yml
```

3.1.3.11. Проверить результат установки `kafka-broker`:

```
kubectl get deployments --namespace kafka-project
```

```
kubectl get pods --namespace kafka-project
```

```
ops@lt-platform:~/kubespray$ kubectl get deployments --namespace kafka-project
NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kafka-broker0           1         1         1            1           118m
zookeeper-deployment-1  1         1         1            1           141m
ops@lt-platform:~/kubespray$ kubectl get pods --namespace kafka-project
NAME                                    READY   STATUS    RESTARTS   AGE
kafka-broker0-5566b4cfd5-9f4xt          1/1     Running   1          118m
zookeeper-deployment-1-6d858cbdfd-p71vk 1/1     Running   0          141m
ops@lt-platform:~/kubespray$
```

Рисунок 2.21 Результат успешной установки брокера Kafka

Примечание: на следующем шаге нам потребуется наименование `pod`, соответствующего `kafka-broker`: `kafka-broker0-5566b4cfd5-9f4xt`.

3.1.3.12. Пробросить порт для `kafka-broker`:

```
kubectl port-forward kafka-broker0-5566b4cfd5-9f4xt --namespace kafka-project 9092 &
```

3.1.3.13. Проверить работу `kafka`:

Отправить сообщение `"Am I receiving this message?"` в тему `«admintome-test»`:

```
echo "Am I receiving this message?" | kafkacat -P -b 127.0.0.1:9092 -t admintome-test
```

Считать сообщение `"Am I receiving this message?"` из темы `«admintome-test»` и записать сообщение `«"Hello World! Hello IgorKaSan!"»`:

```
kafkacat -C -b 127.0.0.1:9092 -t admintome-test
```

Считать все сообщения из темы `«admintome-test»`:

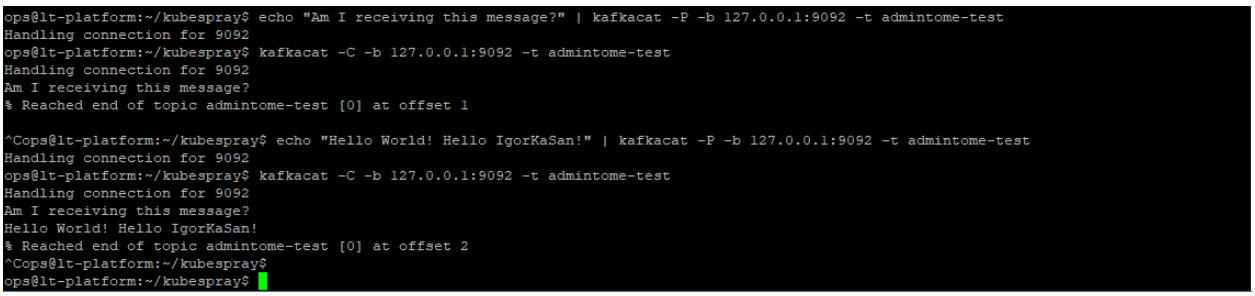


Рисунок 2.22 Результат проверки работоспособности Kafka

Примечание: если во время следующей сессии, при попытке обратиться к теме `Kafka`, вы получите ошибку: «`All broker connections are down: 1/1 brokers are down : terminating`», то удалите (перезапустите) контейнер `Zookeeper`: `kubectl delete pod zookeeper-deployment-1-6d858cbdfd-p7lvk --namespace kafka-project` и, при необходимости, повторно пробросить порт 9092 для брокера `kubectl port-forward kafka-broker0-5566b4cfd5-85s86 --namespace kafka-project 9092 &`. Мне помогло...

3.2. Установить компоненты LPWAN.DLMS

3.2.1. Создать файл с именем lpwan_dlms.yml и добавить следующее содержимое:

```
nano inventory/mycluster/group_vars/k8s-clus-
ter/lpwan_dlms.yml
```

Вставить в файл следующие строки:

```
apiVersion: v1
```

```
kind: Namespace
metadata:
  name: lpwan_dlms
```

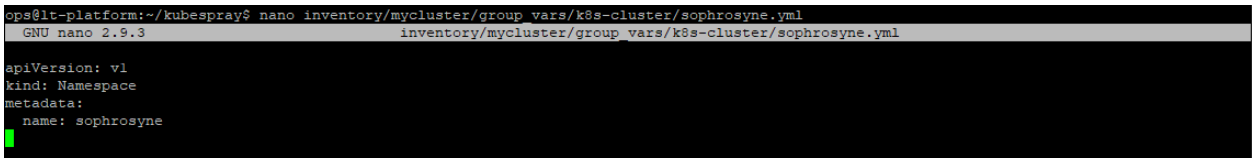


Рисунок 2.23 Содержимое файла lpwan_dlms.yml после редактирования

3.2.2. Создать пространство имен для установки `lpwan_dlms`.

```
kubectl create -f inventory/mycluster/group_vars/k8s-cluster/lpwan_dlms.yml
```

3.2.3. Проверить успешность создания пространства имен `lpwan_dlms`.

```
kubectl get namespaces
```

В списке отображаемых имён должно присутствовать “lpwan_dlms”

3.2.4. Установить компонент `metering-discovery`.

```
helm install --set service.jolokia.nodePort=30110,node.instancegroup=nodes --name metering-discovery --devel --debug sp-releases/metering-discovery --version latest --namespace sophrosyne
```

3.2.5. Установить компонент `metering—dlms-assistant`.

```
helm install --set service.metering-system-router.nodePort=30170 --name metering—dlms-assistant --devel --debug sp-releases/ metering—dlms-assistant --version latest --namespace sophrosyne
```

3.2.6. Установить компонент `metering-mbus-service`.

© Лартех, 2020 г. LoRa Alliance Member

```
helm install -set node.instancegroup=nodes --name metering-mbus-service --devel --debug sp-releases/metering-mbus-service --version latest --namespace sophrosyne
```

3.2.7. Установить компонент `metering-network-gateway`.

```
helm install --set service.metering-system-router.nodePort=30170 --name metering-network-gateway --devel --debug sp-releases/metering-network-gateway --version latest --namespace sophrosyne
```

3.2.8. Установить компонент `metering-processing`.

```
helm install --set service.metering-system-router.nodePort=30170 --name metering-processing --devel --debug sp-releases/metering-processing --version latest --namespace sophrosyne
```

3.2.9. Установить компонент `metering-public-api`.

```
helm install --set service.metering-system-router.nodePort=30170 --name metering-public-api --devel --debug sp-releases/metering-public-api --version latest --namespace sophrosyne
```

3.2.10. Установить компонент `metering-registry`.

```
helm install --set service.metering-system-router.nodePort=30170--name metering-registry --devel --debug sp-releases/metering-registry --version latest --namespace sophrosyne
```

3.2.11. Установить компонент `metering-system-router`.

```
helm install --set service.metering-system-router.node-
Port=30170 --name metering-system-router --devel --debug sp-
releases/metering-system-router --version latest --namespace
sophrosyne
```

3.2.12. Установить компонент `metering-timeseries`.

```
helm install --set service.metering-system-router.node-
Port=30170 --name metering-timeseries-adpt-core --devel --de-
bug sp-releases/metering-timeseries-adpt-core --version latest
--namespace sophrosyne
```

3.3. Установить компоненты мониторинга LPWAN.DLMS

3.3.1. Создать пространство имен kube-monitoring.

```
kubectl create namespace kube-monitoring
```

3.3.2. Установить компонент prometheus.

```
helm install stable/prometheus --debug --namespace kube-moni-
toring
```

3.3.3. Установить компонент grafana.

```
helm install stable/grafana --debug --namespace kube-monitor-
ing
```

3.3.4. Проверить успешность установки prometheus и grafana.

```
kubectl get pods --namespace kube-monitoring
```

© Лартех, 2020 г. LoRa Alliance Member

```
ops@lt-platform:~/kubespray$ kubectl get pods --namespace kube-monitoring
NAME                                                   READY   STATUS    RESTARTS   AGE
cranky-penguin-grafana-6bf6df9696-fsnhz                1/1     Running   0          7m29s
ranting-eagle-prometheus-alertmanager-7b58b4c659-vfmcl 0/2     Pending   0          28m
ranting-eagle-prometheus-kube-state-metrics-5c447f9f98-rm4sq   1/1     Running   0          28m
ranting-eagle-prometheus-node-exporter-chf9w           1/1     Running   0          28m
ranting-eagle-prometheus-node-exporter-cndq6           1/1     Running   0          28m
ranting-eagle-prometheus-node-exporter-jb4bx           1/1     Running   0          28m
ranting-eagle-prometheus-pushgateway-568d557d9c-tk9hd  1/1     Running   0          28m
ranting-eagle-prometheus-server-54cd6474f-d9pl8        0/2     Pending   0          28m
ops@lt-platform:~/kubespray$
```

Рисунок 2.24 Установленные компоненты мониторинга

3.3.5. Запросить секретное слово (пароль).

```
kubectl get secret --namespace kube-monitoring foiled-umbrellabird-grafana -o jsonpath="{.data.grafana-admin-password}" | base64 --decode ; echo
```

```
[vagrant@k8s-kops ~]$ kubectl get secret --namespace kube-monitoring foiled-umbrellabird-grafana -o jsonpath="{.data.grafana-admin-password}" | base64 --decode ; echo
GBCWcUHPd4
[vagrant@k8s-kops ~]$
```

Рисунок 2.25 Секретное слово для входа в панель мониторинга grafana

Примечание: Секретное слово потребуется для входа в панель мониторинга `grafana`.

© Лартех, 2020 г. LoRa Alliance Member

Лист изменений

Версия	Дата	Описание
1.0		Редакция первая