|  |
| --- |
| **СП "LPWAN.SmartGrid"** |
| **Кластер Kubernetes на выделенном сервере** |
| **Инструкция по установке** |
| **ЛРВМ.501522.004.И2.02.1** |
|  |

# **Общие сведения**

Настоящий документ определяет процедуру установки (развертывания) серверной платформы LPWAN.SmartGrid 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 LPWAN.SmartGrid в виде кластера Kubernetes.

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

# **Процедура установки кластера Kubernetes на выделенный сервер**

Процедура установки кластера Kubernetes 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:
* PostgreSQL;
* Redis;
* Kafka.
* Установить компоненты платформы:
* mpu-srv-core;
* devreg-srv-core;
* auth-srv-core;
* nwk-srv-core;
* semtech-gtw-core.
* Установить компоненты мониторинга:
* Prometheus;
* Grafana.

Для выполнения перечисленных операций необходимы следующие виды обеспечения:

1. Аппаратное:

* Компьютер (Intel Xeon E3-1230 3.4 ГГц, 32 Гб DDR4, 2\*240 ГГб SSD, 2\*2 Тб SATA) с выходом в Интернет.

*Примечание: в нашим случае использовался Intel Core5 3.5 ГГц, 20 Гб DDR4, 1 Тб SATA.*

1. Программное обеспечение:

* Дистрибутив Ubuntu Server 18.04.1 (64-bit) (<http://releases.ubuntu.com/18.04/ubuntu-18.04.1-live-server-amd64.iso>).

1. Информационное:

* Инструкция по установке серверной платформы LPWAN.SmartGrid в виде кластера Kubernetes на выделенный сервер.

1. Организационное:

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.

## **Установить операционную систему**

* + 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. На основе скачанного образа создать загрузочный носитель (flesh-drive, DVD).
    3. Загрузиться с созданного носителя и следовать указаниям установщика.

*Примечание: все этапы сценария установки выполняются по умолчанию.*

*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).*

* + 1. 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).*

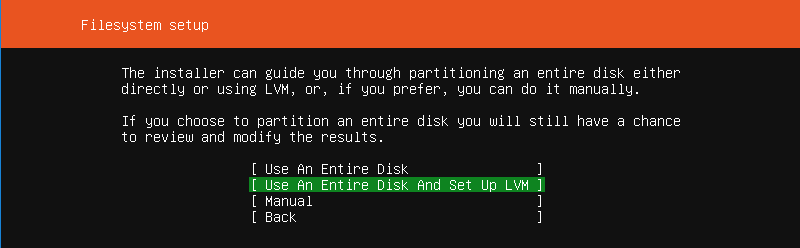


Рисунок 2.1 Режим разметки диска с использованием LVM

* + 1. 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).*

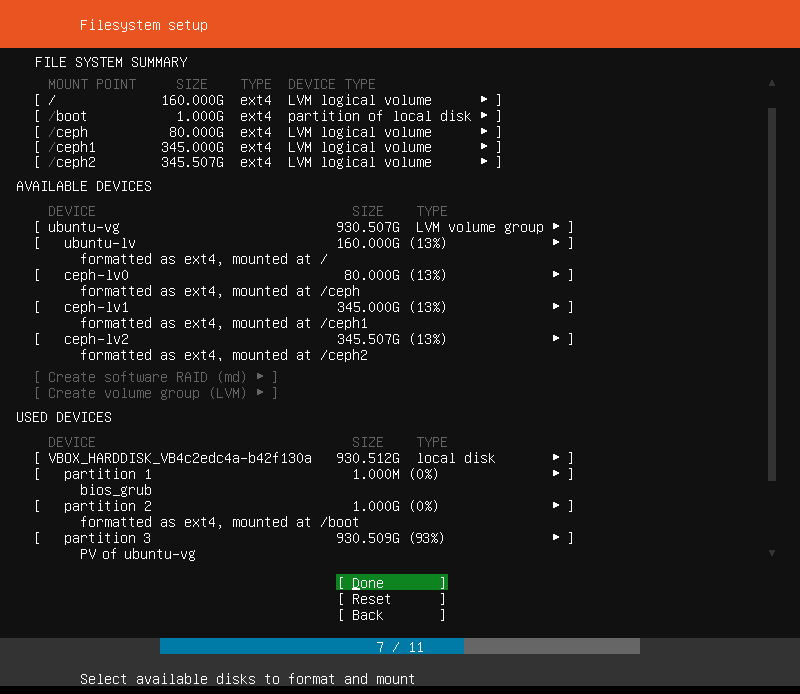


Рисунок 2.2 Диск после разметки

* + 1. Заполнить профайл.

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

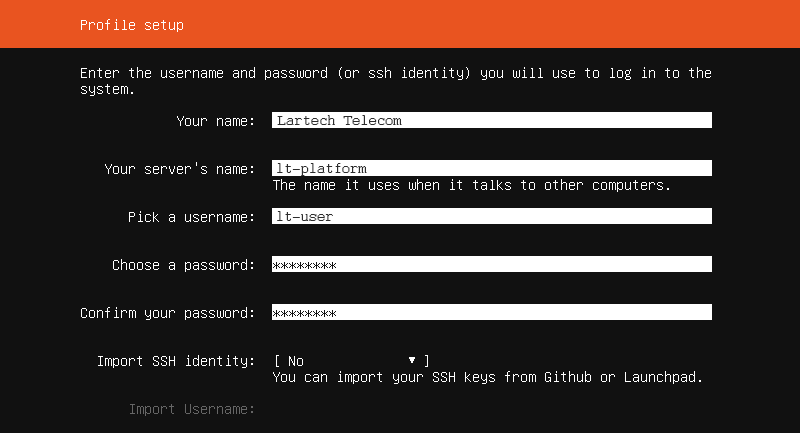


Рисунок 2.3 Заполненный профайл

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

* + 1. Перегрузить систему по завершению сценария установки.

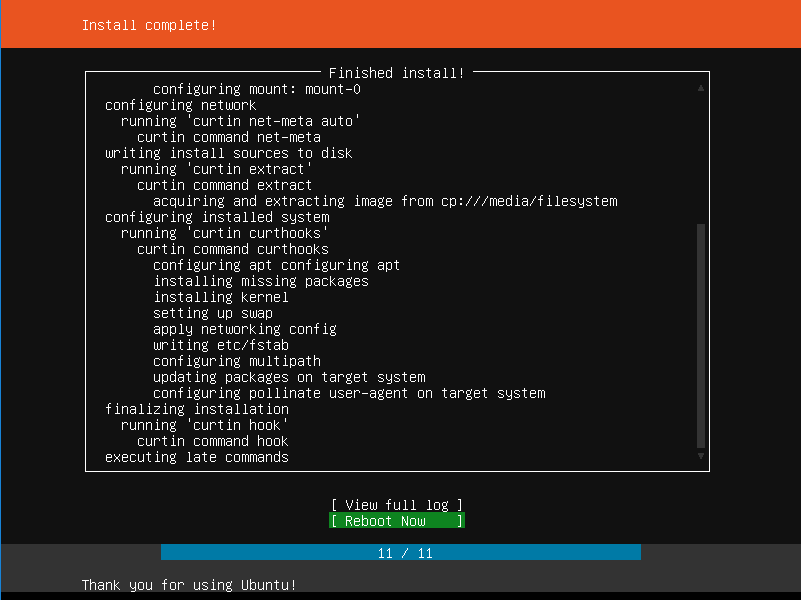


Рисунок 2.4 Перезагрузка системы в конце сценария установки

## **Настроить ОС**

* + 1. Войти в систему под пользователем lt-user.

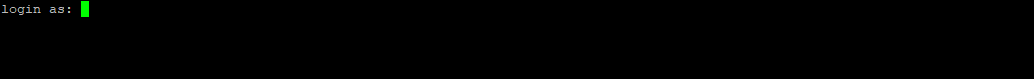


Рисунок 2.5 Приглашение на вход после перезагрузки системы

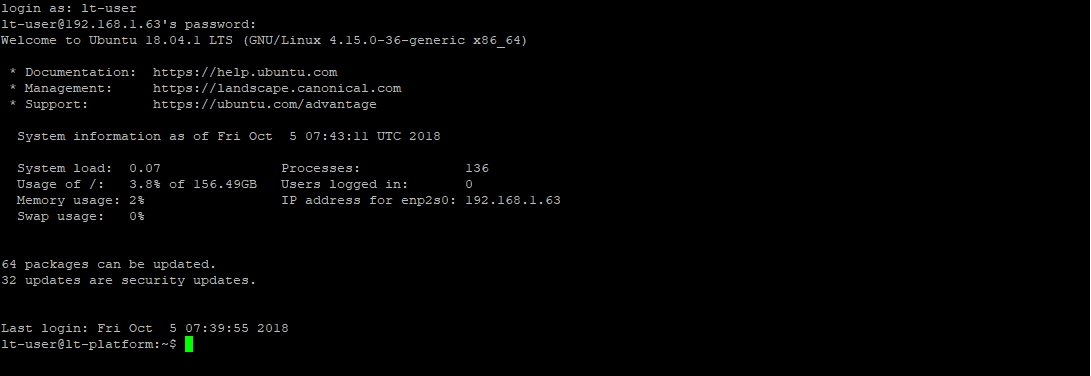


Рисунок 2.6 Состояние после успешного входа lt-user в систему

* + 1. Перейти в режим суперпользователя.

|  |  |
| --- | --- |
|  | sudo su - |

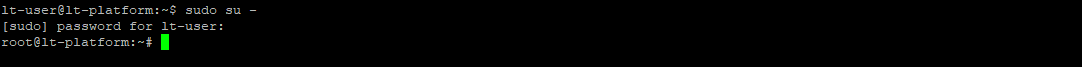


Рисунок 2.7 Режим суперпользователся

* + 1. Создать пользователя ops:

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

* + 1. 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.*

* + 1. Отключить вход пользователю root по ssh:

|  |  |
| --- | --- |
|  | sed -i 's/^#\*PermitRootLogin\s\w.\*$/PermitRootLogin no/' /etc/ssh/sshd\_config |

* + 1. Перезапустить sshd:

|  |  |
| --- | --- |
|  | systemctl restart sshd |

* + 1. Завершить сеанс и войти под новым пользователем (ops).

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

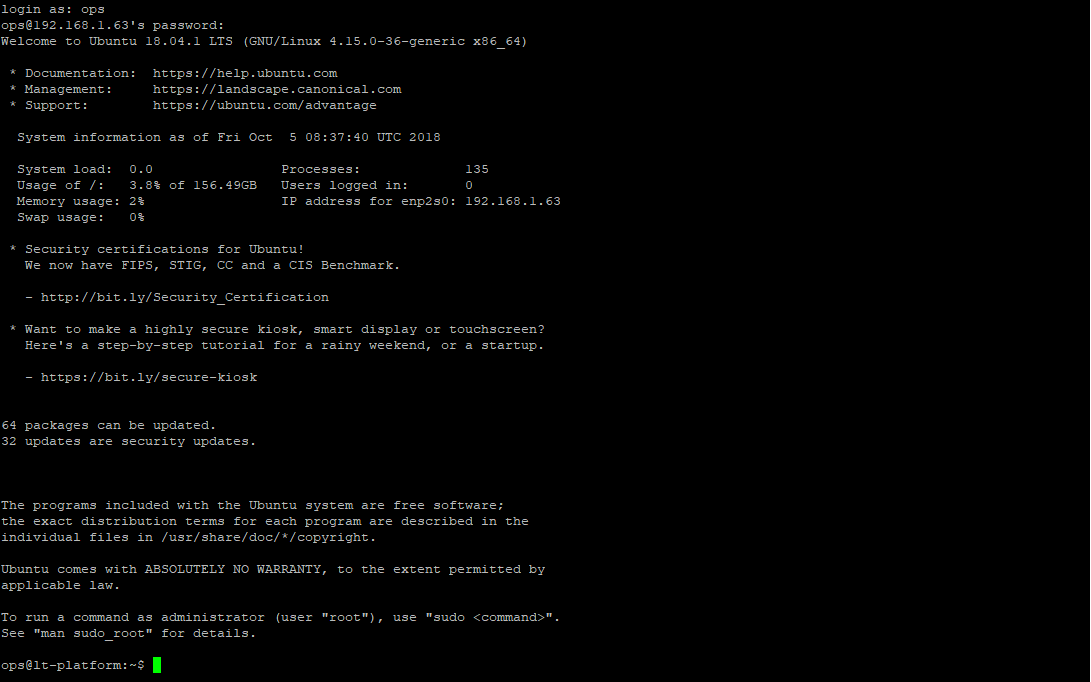


Рисунок 2.8 Состояние после успешного входа под пользователем ops

* + 1. Сгенерировать ssh-ключ.

|  |  |
| --- | --- |
|  | ssh-keygen -q -N "" -f $HOME/.ssh/id\_rsa |

|  |  |
| --- | --- |
|  | cat $HOME/.ssh/id\_rsa.pub > $HOME/.ssh/authorized\_keys |

* + 1. 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 |

* + 1. 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 |

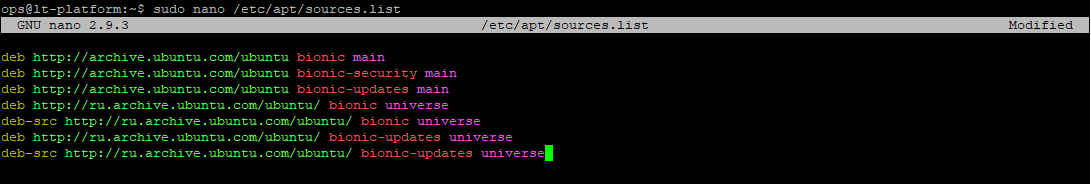


Рисунок 2.9 Содержимое файла /etc/apt/source.list после редактирования

* + 1. Обновить локальный индекс пакетов до последних изменений в репозиториях.

|  |  |
| --- | --- |
|  | sudo apt update |

* + 1. Установить утилиты.

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

* + 1. Отключить IPv6.

|  |  |
| --- | --- |
|  | echo 'net.ipv6.conf.all.disable\_ipv6 = 1' | sudo tee -a /etc/sysctl.d/99-disable\_ipv6.conf |

* + 1. Просмотреть установленные системные переменные.

|  |  |
| --- | --- |
|  | sudo sysctl --system |

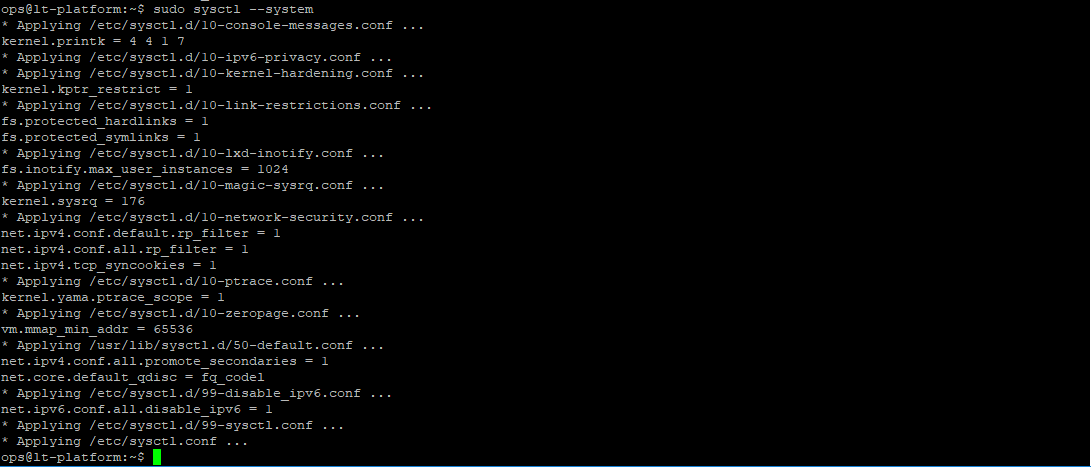


Рисунок 2.10 Установленные по факту настройки ОС переменные

## **Установить Libvirt**

* + 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.*

* + 1. Удалить дефолтную виртуальную сеть:

|  |  |
| --- | --- |
|  | sudo virsh net-autostart --disable default |

|  |  |
| --- | --- |
|  | sudo virsh net-destroy default |

|  |  |
| --- | --- |
|  | sudo virsh net-undefine default |

* + 1. 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 |

* + 1. 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 |

* + 1. Перезапустить systemd-networkd.

|  |  |
| --- | --- |
|  | sudo systemctl restart systemd-networkd |

* + 1. 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 |

* + 1. 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 RELATED,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 |

* + 1. 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».*

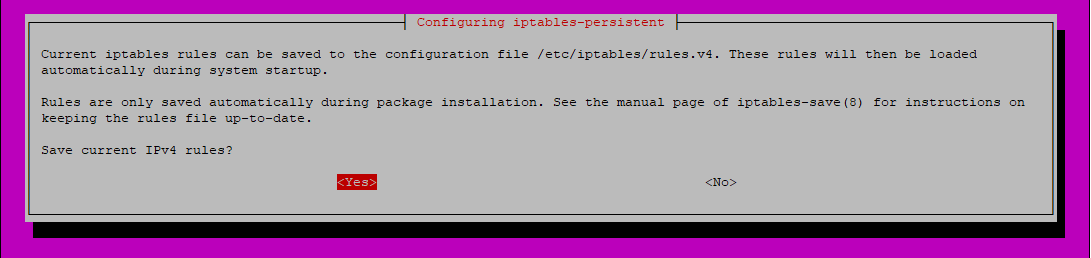


Рисунок 2.11 Запрос на подтверждение сохранение правил для IPv4

* + 1. Добавить сервис netfilter-persistent в автозапуск и перезапустить его.

|  |  |
| --- | --- |
|  | sudo systemctl enable netfilter-persistent |

|  |  |
| --- | --- |
|  | sudo systemctl restart netfilter-persistent |

* + 1. Сконфигурировать dnsmasq.

|  |  |
| --- | --- |
|  | 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 |

* + 1. Добавить имя хоста hostname в /etc/hosts.

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

|  |  |
| --- | --- |
|  | sudo nano /etc/hosts |

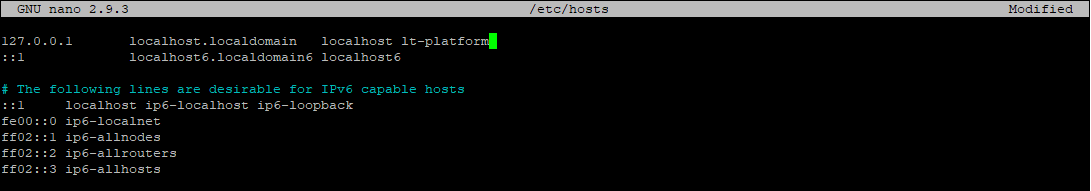


Рисунок 2.12 Содержимое файла /etc/hosts после редактирования

* + 1. Отключить systemd-resolved.

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

* + 1. 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 |

* + 1. Перезапустить dnsmasq.

|  |  |
| --- | --- |
|  | sudo systemctl restart dnsmasq |

## **Установить Ceph**

* + 1. Добавить ключ репозитория Ceph.

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

* + 1. Добавить репозиторий Ceph.

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

* + 1. Установить пакет ceph-deploy.

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

* + 1. Создать директорию, в которой будут храниться конфиги и ключи для Ceph.

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

* + 1. Установить Ceph.

|  |  |
| --- | --- |
|  | ceph-deploy install $HOSTNAME |

* + 1. Создать кластер:

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

* + 1. 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 |

* + 1. Установить Ceph Monitor.

|  |  |
| --- | --- |
|  | ceph-deploy mon create-initial |

* + 1. Запустить кластер:

|  |  |
| --- | --- |
|  | ceph-deploy admin $HOSTNAME |

* + 1. Установить Ceph Manager.

|  |  |
| --- | --- |
|  | ceph-deploy mgr create $HOSTNAME |

* + 1. 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 |

* + 1. 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.*

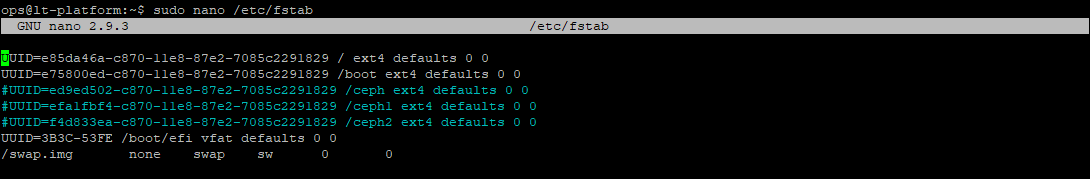


Рисунок 2.13 Содержимое файла /etc/fstab после редактирования

* + 1. Перегрузить сервер lt-platform.

|  |  |
| --- | --- |
|  | sudo shutdown -r now |

* + 1. Проверить статус Ceph кластера.

|  |  |
| --- | --- |
|  | sudo ceph -s |

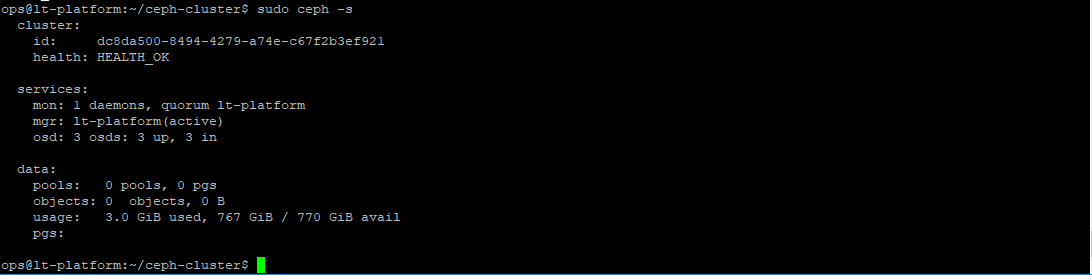


Рисунок 2.14 Текущий старус Ceph кластера

## **Подключить Ceph к Libvirt**

* + 1. Создать пул для хранения образов виртуальных машин.

|  |  |
| --- | --- |
|  | sudo ceph osd pool create libvirt 128 128  sudo ceph osd pool application enable libvirt libvirt |

* + 1. 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' |

* + 1. Создать libvirt-секрет.

|  |  |
| --- | --- |
|  | LIBVIRT\_CEPH\_AUTH\_KEY=$(sudo ceph auth get-or-create client.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 "$LIBVIRT\_CEPH\_SECRET\_UUID" "$LIBVIRT\_CEPH\_AUTH\_KEY" |

* + 1. 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 |

## **Создать виртуальные машины**

* + 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 themostlystrongpassphraselol  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  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 |

* + 1. Создать виртуальные машины kube-master-1/2/3.

|  |  |
| --- | --- |
|  | 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=virtio,name=org.qemu.guest\_agent.0 \  --location https://mirror.yandex.ru/ubuntu/dists/bionic/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 |

* + 1. 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=virtio,name=org.qemu.guest\_agent.0 \  --location https://mirror.yandex.ru/ubuntu/dists/bionic/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.*

* + 1. Проверить состояние виртуальных машин.

|  |  |
| --- | --- |
|  | sudo virsh list --all |

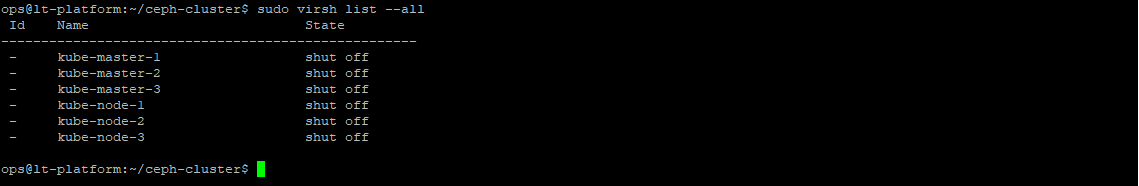


Рисунок 2.15 Состояние виртуальных машин после успешной установки

*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

* + 1. 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 |

* + 1. Запустить все ВМ:

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

## **Установить Kubespray**

* + 1. Установить необходимые пакеты.

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

* + 1. Загрузить репозиторий kubespray.

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

|  |  |
| --- | --- |
|  | cd $HOME/kubespray |

|  |  |
| --- | --- |
|  | git checkout -b mycluster |

* + 1. Установить зависимости.

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

* + 1. Скопировать образец inventory-файла.

|  |  |
| --- | --- |
|  | cp -rfp inventory/sample inventory/mycluster |

* + 1. Сгенерировать файл hosts.ini.

|  |  |
| --- | --- |
|  | 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 |

* + 1. 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 |

* + 1. 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 |

* + 1. 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] |

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

* + 1. 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}' |

* + 1. 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 |

* + 1. 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).*

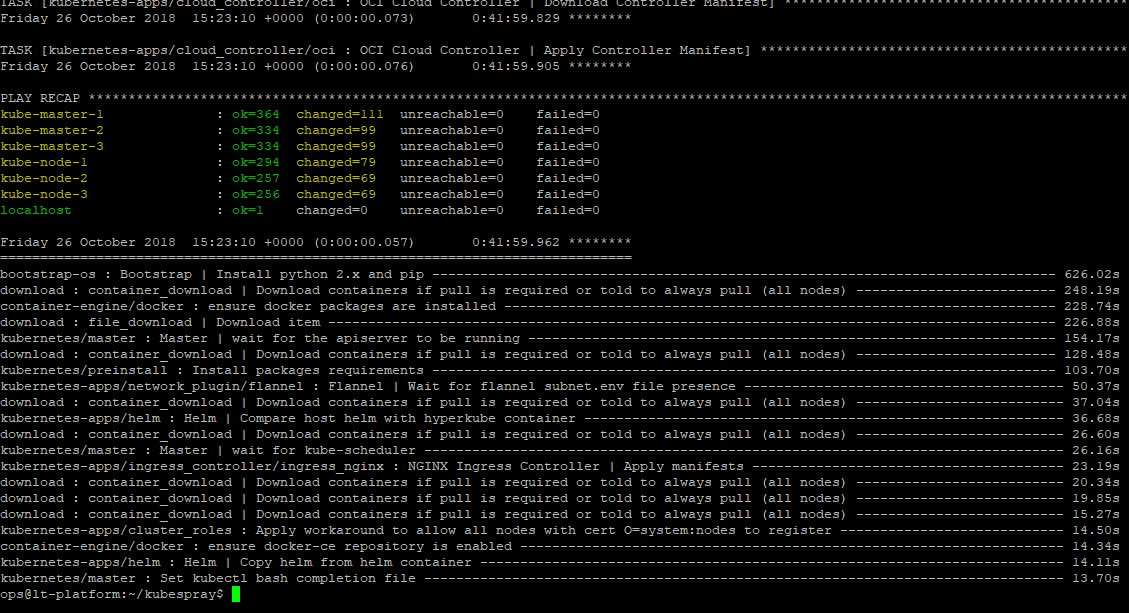


Рисунок 2.17 Процесс установки кластера завершился

* + 1. Скопировать kubeconfig в домашнюю директорию.

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

* + 1. Скопировать kubectl.

|  |  |
| --- | --- |
|  | sudo cp inventory/mycluster/artifacts/kubectl /usr/local/bin/ |

* + 1. Проверить состояние Kubernetesкластера.

|  |  |
| --- | --- |
|  | kubectl get nodes |

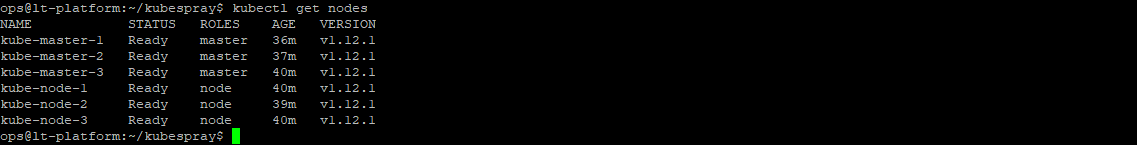


Рисунок 2.18 Текущее состояние Kubernetes кластера

* + 1. Создать rbd-пул для кластера.

|  |  |
| --- | --- |
|  | sudo ceph osd pool create kubernetes 128 128  sudo ceph osd pool application enable kubernetes kubernetes |

* + 1. 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==*

* + 1. 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 |

* + 1. 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 |

## **Настроить балансировщик для внешнего доступа**

* + 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 |

* + 1. 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;  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 |

* + 1. Перезапустить nginx.

|  |  |
| --- | --- |
|  | sudo systemctl restart nginx |

## **Установить на кластер дополнения**

* + 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*

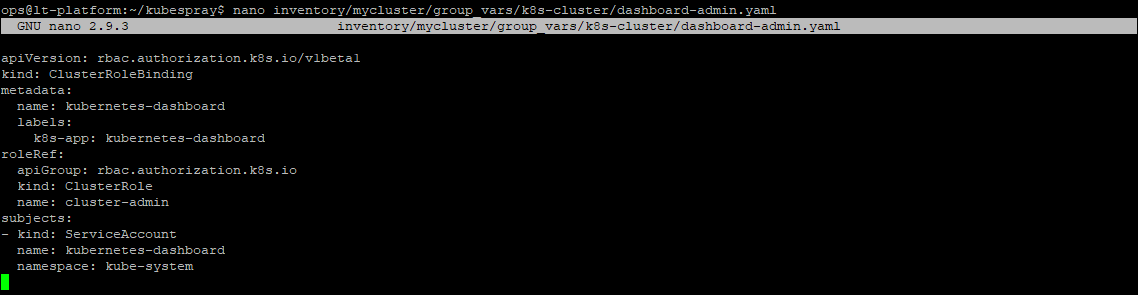


Рисунок 2.19 Содержимое файла dashboard-admin.yaml после редактирования

* + 1. 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 |

* + 1. Разрешить порт для proxy-сервер kubernetes.

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

* + 1. Запустить proxy-сервер kubernetes в фоновом режиме.

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

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

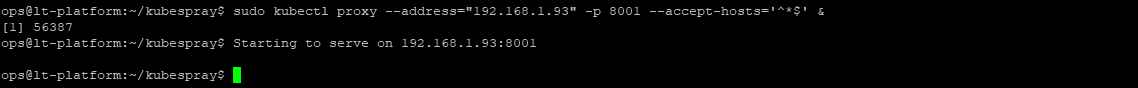


Рисунок 2.20 Сообщение о запуске proxy-сервера Kubernetes

* + 1. 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/.*

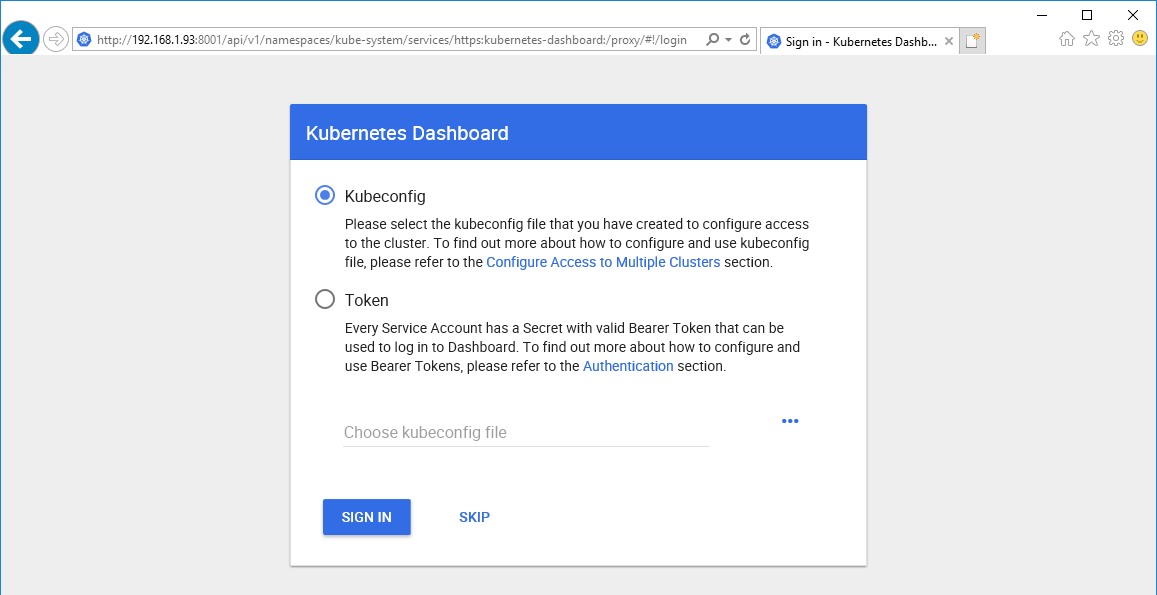


Рисунок 2.21 Панель управления Kubernetes (выбор типа авторизации)

* + 1. Установить утилиту *kops* для управления kubernetes.

|  |  |
| --- | --- |
|  | sudo snap install kops |

* + 1. Установить пакетный менеджер *helm*.

|  |  |
| --- | --- |
|  | sudo snap install helm --classic |

* + 1. 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 |

* + 1. 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 |

* + 1. 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 |

* + 1. Отобразить (показать) установленные на кластере сервисы (Рисунок 2.20):

|  |  |
| --- | --- |
|  | kubectl -n kube-system get pods |

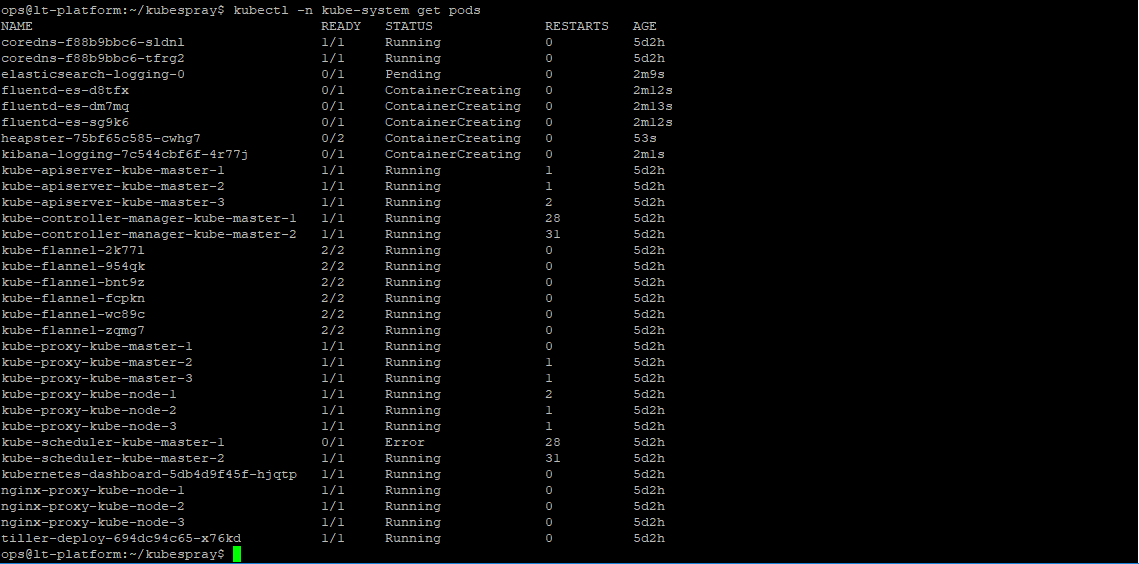


Рисунок 2.22 Установленные на кластере сервисы

* + 1. Установить Tiller в Kubernetes кластере (Рисунок 2.21).

|  |  |
| --- | --- |
|  | helm init --upgrade |

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

* + 1. Проверить корректность установки Tiller.

|  |  |
| --- | --- |
|  | helm list |



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.*

## **Установить базовые компоненты**

* + 1. Установить PostgreSQL.
       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*

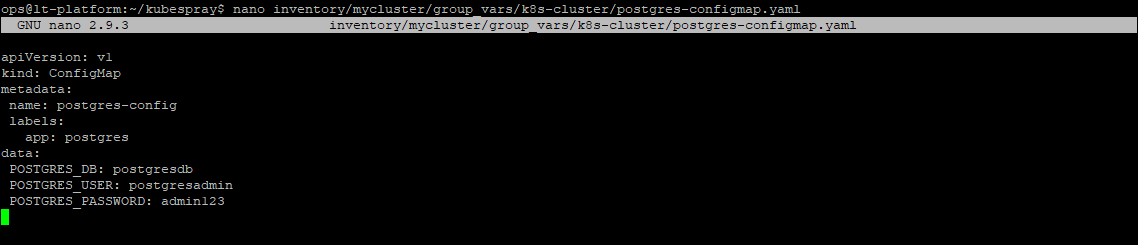


Рисунок 2.24 Содержимое файла postgres-configmap.yaml после редактирования

* + - 1. Создать ConfigMap:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/postgres-configmap.yaml |

* + - 1. 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*

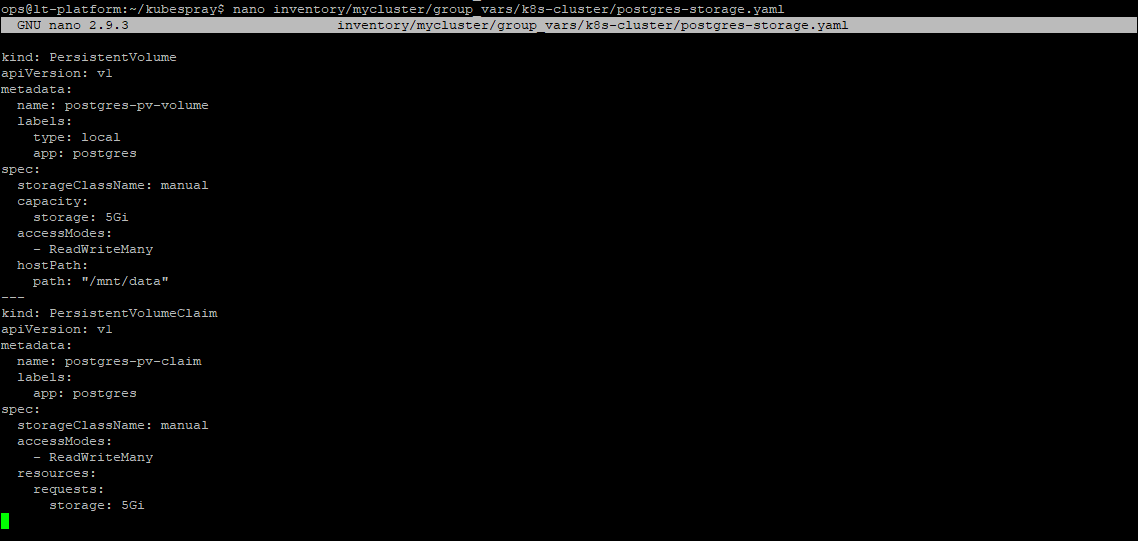


Рисунок 2.25 Содержимое файла postgres-storage.yaml после редактирования

* + - 1. Создать PersistentVolume:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/postgres-storage.yaml |

* + - 1. 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*

*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*

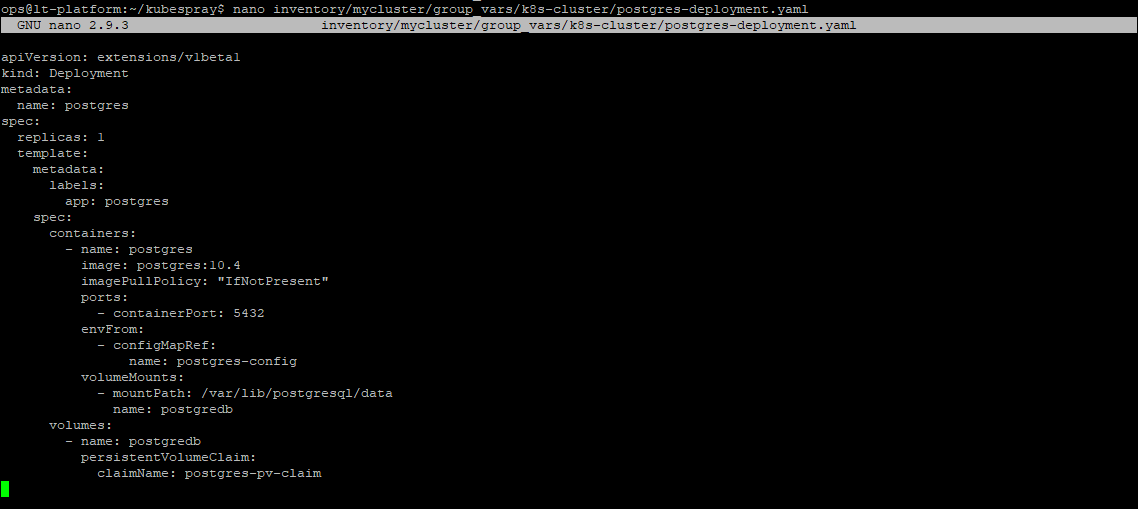


Рисунок 2.26 Содержимое файла postgres-deployment.yaml после редактирования

* + - 1. Создать Deployment:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/postgres-deployment.yaml |

* + - 1. 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*

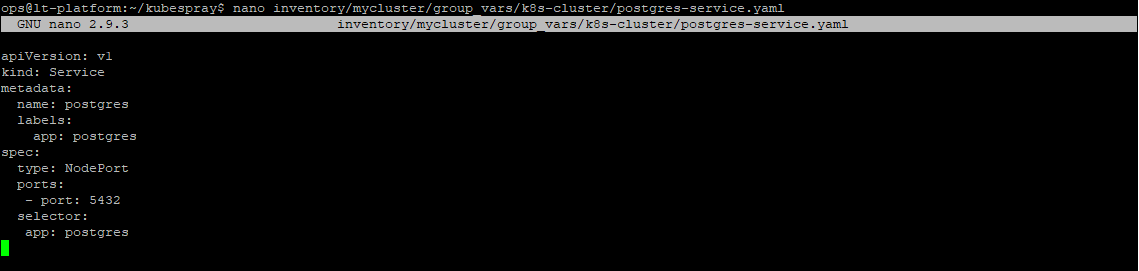


Рисунок 2.27 Содержимое файла postgres-service.yaml после редактирования

* + - 1. Создать Service:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/postgres-service.yaml |

* + - 1. Убедиться, что сервис postgres создан:

|  |  |
| --- | --- |
|  | kubectl get services |

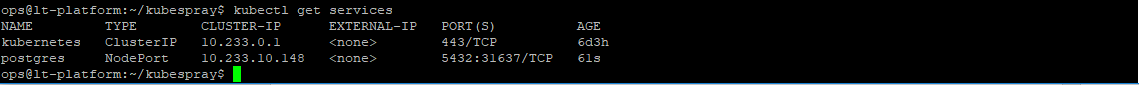


Рисунок 2.28 Результат проверки наличия сервиса PostgreSQL

* + - 1. Установить postgres клиента:

|  |  |
| --- | --- |
|  | sudo apt-get install postgresql-client |



Рисунок 2.29 Результат установки клиента PostgreSQL

* + - 1. Посмотреть имя установленного контейнера postgres:

|  |  |
| --- | --- |
|  | kubectl get pods --output=wide |

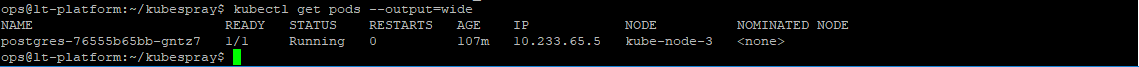


Рисунок 2.30 Просмотр сведений о запущенном контейнере PostgreSQL

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

* + - 1. Пробросить порт для доступа к контейнеру postgres:

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

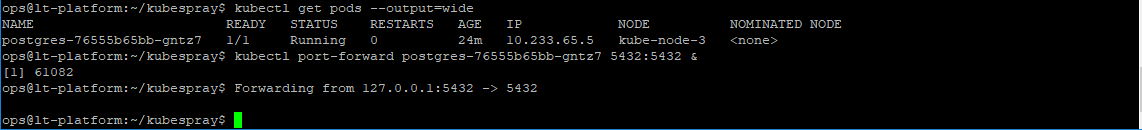


Рисунок 2.31 Сообщение об успешной переадресации запросов на порт PostgreSQL

* + - 1. Проверить доступ к postgres:

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

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

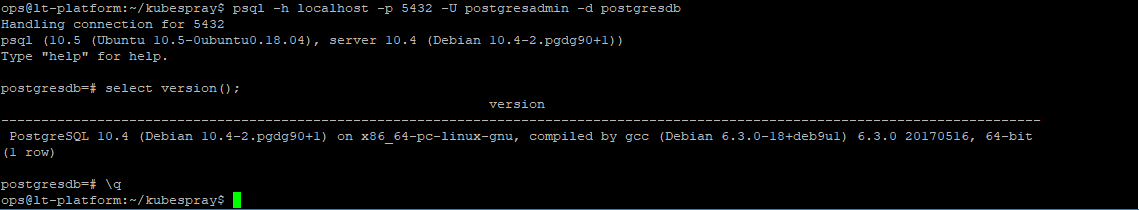


Рисунок 2.32 Результат проверки доступа клиента к PostgreSQL

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

* + 1. Установить Redis.
       1. Создать redis-configmap.yaml:

|  |  |
| --- | --- |
|  | nano inventory/mycluster/group\_vars/k8s-cluster/redis-configmap.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*

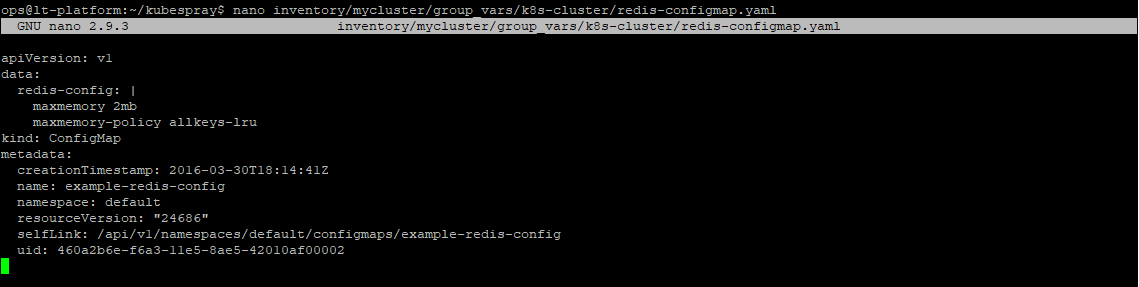


Рисунок 2.33 Содержимое файла redis-configmap.yaml после редактирования

* + - 1. Создать redis-configmap:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/redis-configmap.yaml |

* + - 1. 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*

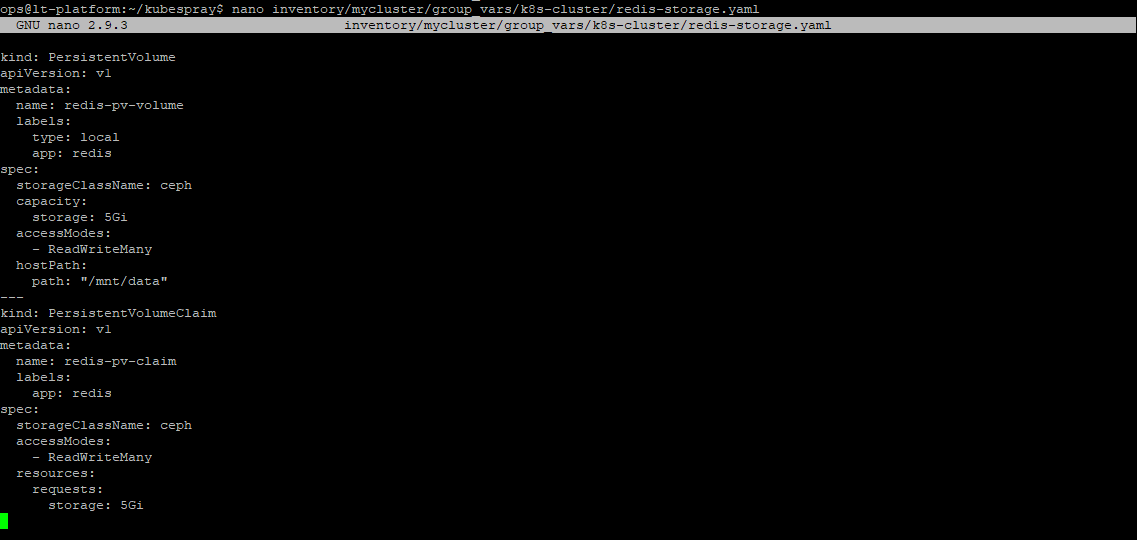


Рисунок 2.34 Содержимое файла redis-configmap.yaml после редактирования

* + - 1. Создать redis-storage:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/redis-storage.yaml |

* + - 1. 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 2.35 Содержимое файла redis-pod.yaml после редактирования

* + - 1. 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.*

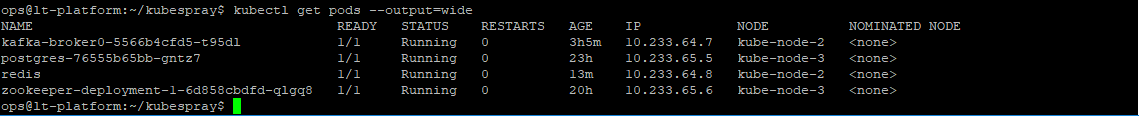


Рисунок 2.36 Результат успешного запуска контейнера Redis

* + - 1. Для проверки работоспособности запустить клиента Redis:

|  |  |
| --- | --- |
|  | kubectl exec -it redis redis-cli |

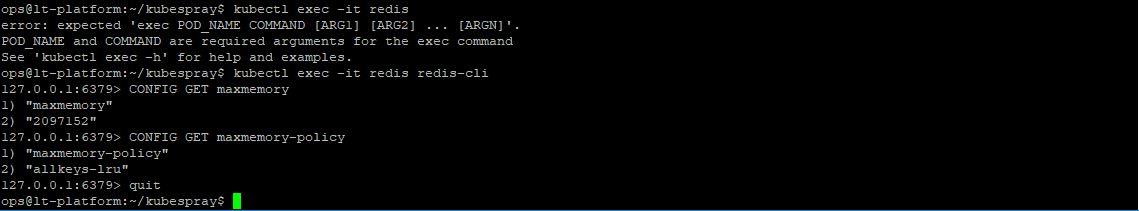


Рисунок 2.37 Результат проверки доступа клиента к Redis

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

* + 1. Установить Kafka.
       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*

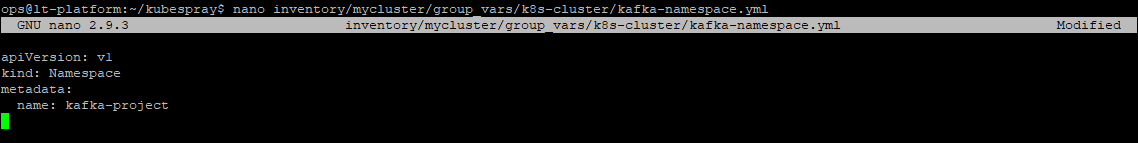


Рисунок 2.38 Содержимое файла kafka-namespace.yml после редактирования

* + - 1. Создать пространство имен для установки Kafka:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/kafka-namespace.yml |

* + - 1. 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*

*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*

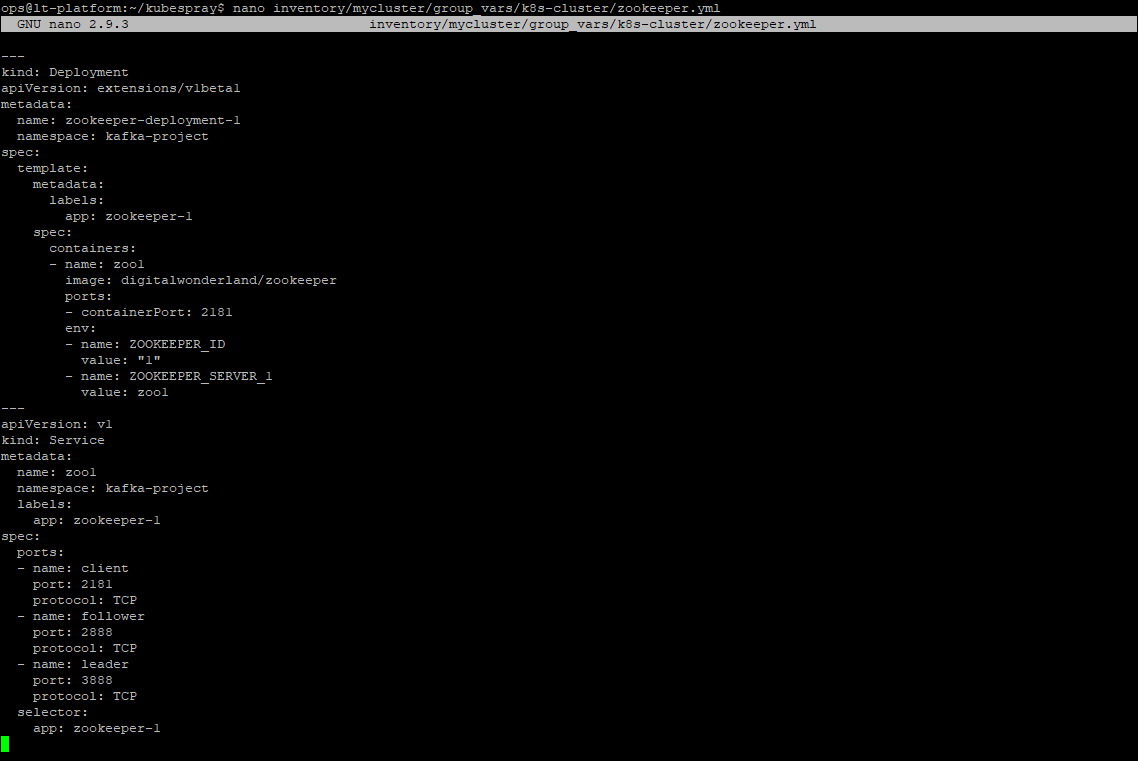


Рисунок 2.39 Содержимое файла zookeeper.yml после редактирования

* + - 1. Установить zookeeper:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/zookeeper.yml |

* + - 1. Проверить результат установки zookeeper:

|  |  |
| --- | --- |
|  | kubectl get deployments --namespace kafka-project |

|  |  |
| --- | --- |
|  | kubectl get svc --namespace kafka-project |

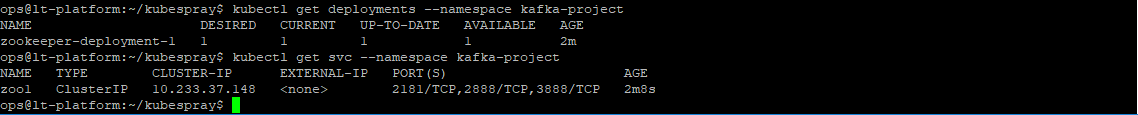


Рисунок 2.40 Результат успешной установки Zookeeper

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

* + - 1. 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*

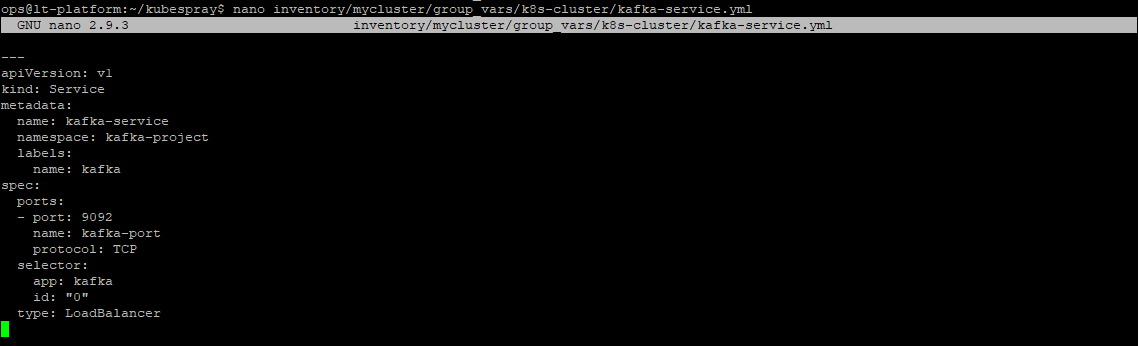


Рисунок 2.41 Содержимое файла kafka-service.yml после редактирования

* + - 1. Установить kafka-service:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/kafka-service.yml |

* + - 1. Проверить результат установки kafka-service:

|  |  |
| --- | --- |
|  | kubectl get svc --namespace kafka-project |

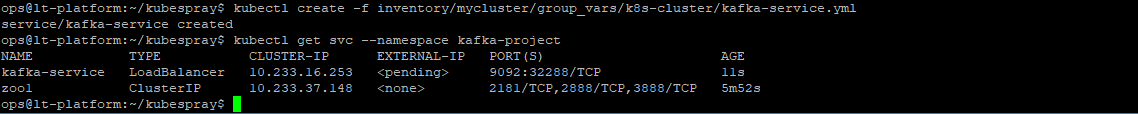


Рисунок 2.42 Результат успешной установки сервиса Kafka

* + - 1. 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*

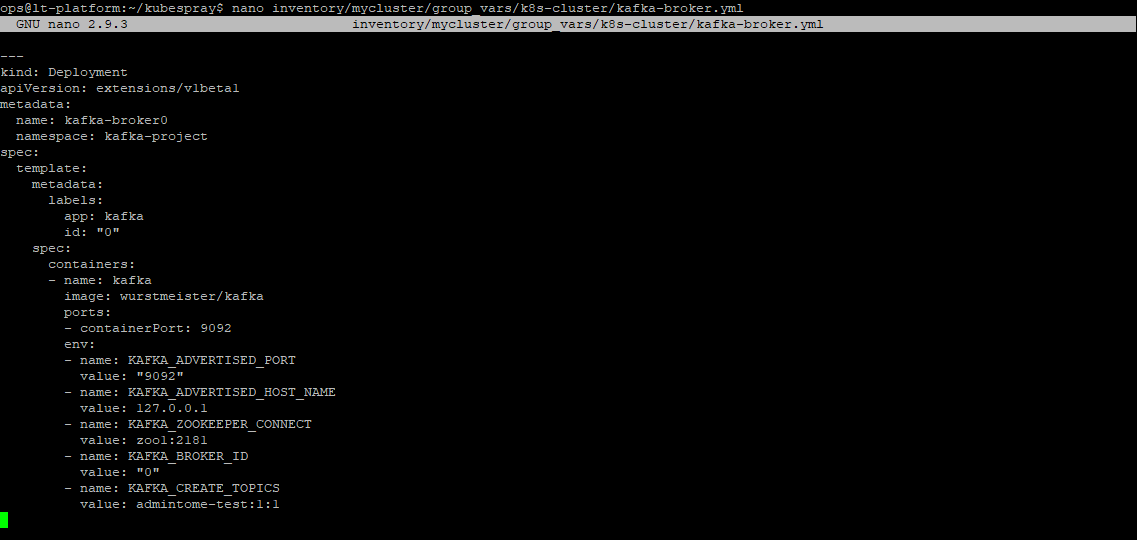


Рисунок 2.43 Содержимое файла kafka-broker.yml после редактирования

* + - 1. Создать kafka-broker:

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/kafka-broker.yml |

* + - 1. Проверить результат установки kafka-broker:

|  |  |
| --- | --- |
|  | kubectl get deployments --namespace kafka-project |

|  |  |
| --- | --- |
|  | kubectl get pods --namespace kafka-project |

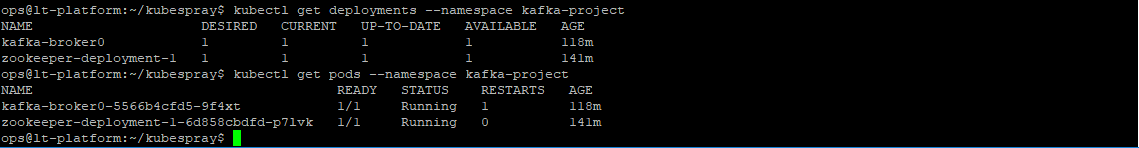


Рисунок 2.44 Результат успешной установки брокера Kafka

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

* + - 1. Пробросить порт для kafka-broker:

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

* + - 1. 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»:

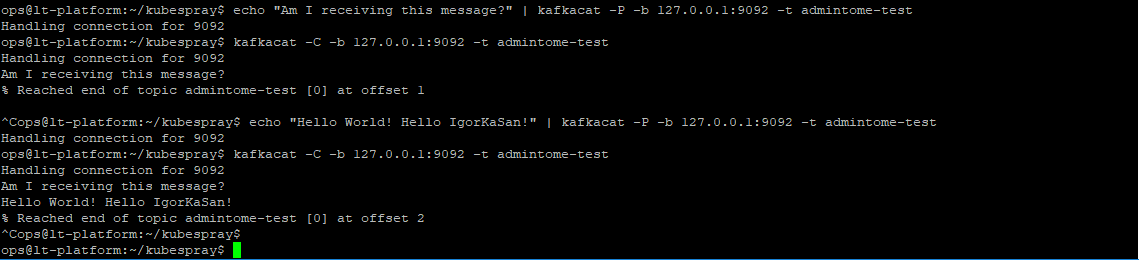


Рисунок 2.45 Результат проверки работоспособности Kafka

*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...*

## **Установить компоненты платформы**

* + 1. Создать файл с именем sophrosyne.yml и добавить следующее содержимое:

|  |  |
| --- | --- |
|  | nano inventory/mycluster/group\_vars/k8s-cluster/sophrosyne.yml |

Вставить в файл следующие строки:

*apiVersion: v1*

*kind: Namespace*

*metadata:*

*name: sophrosyne*

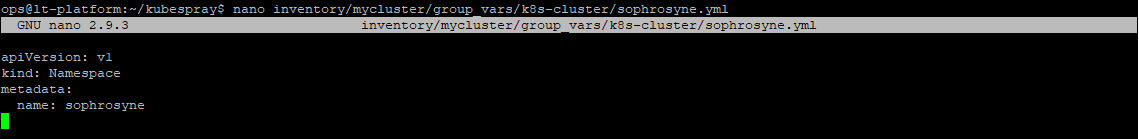


Рисунок 2.46 Содержимое файла sophrosyne.yml после редактирования

* + 1. Создать пространство имен для установки sophrosyne.

|  |  |
| --- | --- |
|  | kubectl create -f inventory/mycluster/group\_vars/k8s-cluster/sophrosyne.yml |

* + 1. Проверить успешность создания пространства имен sophrosyne.

|  |  |
| --- | --- |
|  | kubectl get namespaces |

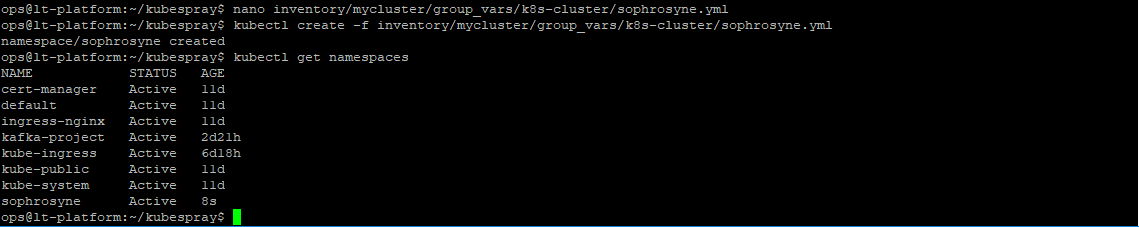


Рисунок 2.47 Результат проверки наличия пространства имен sophrosine

* + 1. Установить компонент mpu-srv-core.

|  |  |
| --- | --- |
|  | helm install --set service.jolokia.nodePort=30110,node.instancegroup=nodes --name mpu-srv-core --devel --debug sp-releases/mpu-srv-core --version 1.11.4 --namespace sophrosyne |

* + 1. Установить компонент devreg-srv-core.

|  |  |
| --- | --- |
|  | helm install --set service.devreg.nodePort=30100 --name devreg-srv-core --devel --debug sp-releases/devreg-srv-core --version 1.10.6 --namespace sophrosyne |

* + 1. Установить компонент auth-srv-core.

|  |  |
| --- | --- |
|  | helm install --set service.oauth.nodePort=30150,node.instancegroup=nodes --name auth-srv-core --devel --debug sp-releases/auth-srv-core --version 1.11.2 --namespace sophrosyne |

* + 1. Установить компонент nwk-srv-core.

|  |  |
| --- | --- |
|  | helm install --set service.shell.nodePort=30370 --name nwk-srv-core --devel --debug sp-releases/nwk-srv-core --version 1.14.0 --namespace sophrosyne |

* + 1. Установить компонент semtech-gtw-core.

|  |  |
| --- | --- |
|  | helm install --set service.gtw.nodePort=30000 --name semtech-gtw-core --devel --debug sp-releases/semtech-gtw-core --version 1.7.11 --namespace sophrosyne |

* + 1. Установить компонент userspace.

|  |  |
| --- | --- |
|  | helm install --set service.userspace.nodePort=30170 --name userspace --devel --debug sp-releases/userspace --version 2.4.0 --namespace sophrosyne |

* + 1. Установить компонент metering.

|  |  |
| --- | --- |
|  | helm install --set service.metering.nodePort=30210 --name metering-adpt-core --devel --debug sp-releases/metering-adpt-core --version 1.1.0 --namespace sophrosyne |

* + 1. Установить компонент routing.

|  |  |
| --- | --- |
|  | helm install --set service.routing.nodePort=30220 --name routing-srv-core --devel --debug sp-releases/routing-srv-core --version 1.3.12 --namespace sophrosyne |

## **Установить компоненты мониторинга**

* + 1. Создать пространство имен kube-monitoring.

|  |  |
| --- | --- |
|  | kubectl create namespace kube-monitoring |

* + 1. Установить компонент prometheus.

|  |  |
| --- | --- |
|  | helm install stable/prometheus --debug --namespace kube-monitoring |

* + 1. Установить компонент grafana.

|  |  |
| --- | --- |
|  | helm install stable/grafana --debug --namespace kube-monitoring |

* + 1. Проверить успешность установки prometheus и grafana.

|  |  |
| --- | --- |
|  | kubectl get pods --namespace kube-monitoring |

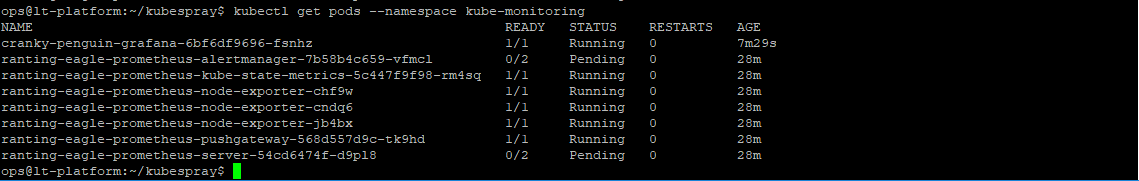


Рисунок 2.48 Установленные компоненты мониторинга

* + 1. Запросить секретное слово (пароль).

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

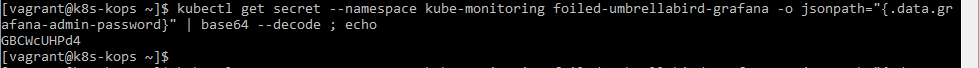


Рисунок 2.49 Секретное слово для входа в панель мониторинга grafana

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