

Общество с ограниченной ответственностью «МетаПрайм»

УТВЕРЖДАЮ

Генеральный директор
ООО «МетаПрайм»

_____ А.В. Рожков

М.П.

«___» _____ 2023 г.

Автоматизированная информационная система
«Региональная цифровая социальная платформа»

полное наименование АС

РЦСП

сокращённое наименование АС

РУКОВОДСТВО ПО УСТАНОВКЕ КОМПОНЕНТОВ СИСТЕМЫ

На _____ листах

Аннотация

Данный документ представляет собой руководство по установке компонентов программного обеспечения «Региональная цифровая социальная платформа». Документ описывает программно-аппаратные требования, предъявляемые к серверному оборудованию и рабочих станций, порядок установки компонентов программного обеспечения, необходимого для функционирования Системы.

Содержание

Аннотация	2
Термины и определения.....	4
Принятые сокращения и обозначения	5
Общие сведения.....	6
1 Требования к программному обеспечению	7
1.1 Требование к программному обеспечению сервера.	7
1.2 Требования к программному обеспечению клиента	7
2 Требования к техническому обеспечению	7
2.1 Аппаратное обеспечение сервера	7
2.2 Аппаратное обеспечение клиента	7
3 Настройка программы	8
3.1 Установка и настройка PostgreSQL.....	8
3.1.1 Установка PostgreSQL 14.x на выделенную виртуальную машину.....	8
3.1.2 Настройка PostgreSQL	8
3.1.3 Установка и настройка pgbouncer на сервер с postgresql	10
3.2 Установка и настройка кластера k8s (kubernetes)	11
3.2.1 Установка k8s (kubernetes) – управляющие(master) узлы	11
3.2.2 Установка k8s (kubernetes) – рабочие узлы(work)	15
3.3 Установка Nexus с docker	16
3.3.1 Установка nexus.....	16
3.3.2 Установка docker	17
3.3.3 Настройка nexus-docker	20
3.4 Установка и настройка кластера kafka.....	21
3.4.1 Установка Java.....	21
3.4.2 Установка и настройка zookeeper	21
3.4.3 Установка Kafka.	22
3.4.4 Проверка кластера и служб.	24
3.5 Установка кластера minio.....	25
3.5.1 Установка minio на одной ноде	25
3.5.2 Установка распределенного minio.....	26
3.6 Настройка ELK для сбора логов(Не обязательна к установке)	27

Термины и определения

В руководстве применяются следующие термины с соответствующими определениями (Таблица 1).

Таблица 1– Список терминов и определений

Термин	Определение
1	2
Веб-браузер	Агент пользователя, позволяющий пользователю получать и читать гипертекстовую информацию, просматривать содержание гипертекстовых узлов (обычно веб-страницы), перемещаться от одного узла к другому и взаимодействовать с информационным наполнением ((ГОСТ Р ИСО 9241-151-2014)
Кластер	Массив взаимосвязанных серверов для одной системы или службы, задачи

Принятые сокращения и обозначения

В руководстве принимаются следующие сокращения и обозначения (Таблица 2).

Таблица 2– Список сокращений и определений

Сокращение (обозначение)	Значение сокращения (обозначения)
1	2
БД	База данных
СУБД	Система управления базами данных
Apache Kafka	Связующее программное обеспечение, ориентированное на обработку сообщений (брокер сообщений)
Apache Mesos	Проект с открытым исходным кодом для управления компьютерными кластерами
Apache ZooKeeper	Иерархическое хранилище значений ключей
Docker Registry	Программное обеспечение, обеспечивающее хранение и доставку до сервера Docker приложений
ELK	Elasticsearch, Kibana, Logstash – (англ.). Решение по сбору, анализу и хранению данных (логов).
JAVA	Объектно-ориентированный язык программирования и платформа вычислений
K8S/ Kubernetes	Открытое программное обеспечение для оркестровки контейнеризированных приложений – автоматизации их развёртывания, масштабирования и координации в условиях кластера.
MinIO	Сервер облачного хранилища
PgBouncer	Программа, управляющая пулом соединений PostgreSQL
PL/pgSQL	Procedural Language/Postgres Structured Query Language, процедурное расширение языка SQL, используемое в СУБД PostgreSQL
PostgreSQL	Свободная объектно-реляционная система управления базами данных
TCP/IP	Transfer Control Protocol/Internet Protocol – (англ.). Сетевая модель передачи данных, представленных в цифровом виде

Общие сведения

Система разработана на платформе специального программного обеспечения такого, как СУБД PostgreSQL с PgBouncer, веб-сервера Apache Kafka и ZooKeeper, объектного хранилища MinIO, веб-браузера, среды разработки OpenJDK, а также языка программирования Java.

1 Требования к программному обеспечению

Функционирование программных компонентов Системы требует предварительной установки ПО сервера и клиента.

1.1 Требование к программному обеспечению сервера.

Установка и эксплуатация программного обеспечения «Региональная цифровая социальная платформа» может выполняться на технических средствах под управлением ОС семейства Linux.

Операционная система должна позволять установить следующее программное обеспечение:

1. PostgreSQL (версия не ниже 14.4);
2. PgBouncer (версия не ниже 1.17.0);
3. Apache Zookeeper (версия не ниже 3.6.3);
4. Apache Kafka (версия не ниже 2.13-3.0.1);
5. MinIO (версия не ниже 2022-08-02T23-59-16Z);
6. OpenJDK (версия не ниже 1.8.0_342);
7. Chromium (версия не ниже 104.0.5112.101).

1.2 Требования к программному обеспечению клиента

В состав программного обеспечения клиента должно входить следующее ПО:

1. Windows (32/64 bit) версия не ниже 7, или Astra Linux или Linux;
2. веб-браузер:
 - Яндекс (рекомендовано);
 - Chrome - (официальная сборка 64-bit);
 - Chromium;
 - Firefox;
 - Спутник;
 - Атом.

2 Требования к техническому обеспечению

2.1 Аппаратное обеспечение сервера

Конфигурация сервера зависит от объема задач, но должна быть не хуже:

1. не менее 1 процессора с характеристиками: не менее 4 ядер; максимальная тактовая частота не менее 2.8 гигагерц;
2. оперативная память: общее количество слотов (DIMM) для установки модулей памяти не менее 12; объем установленной оперативной памяти не менее 32 Гб;
3. минимум 1 контроллер жестких дисков с характеристиками: интерфейс SAS со скоростью передачи информации 6 гигабит в секунду с поддержкой SATA со скоростью передачи информации 6 гигабит в секунду; не менее 6 портов;
4. должна быть установлена кэш-память (энергонезависимый флэш-кэш) объемом 2 гигабайта; поддержка уровней RAID 0, 1, 1+0, 5, 5+0, опционально 6, 6+0; поддерживаемое количество логических томов не менее 32;
5. рекомендуется использовать исключительно SSD, для работы в контроллерах дисков. В том числе и в RAID;
6. сетевая карта Ethernet 1000 Мбит/с.

2.2 Аппаратное обеспечение клиента

Конфигурация клиента должна быть не хуже:

1. процессор - Intel Core i3 2 ГГц или Ryzen 3 2 ГГц или лучше;
2. оперативная память - 4 Гбайт и выше;
3. жесткий диск – от 200 Мбайт свободного места на HDD;
4. сетевая карта Ethernet 100 Мбит/с.

3 Настройка программы

Для обеспечения работоспособности разработанных программных комплексов необходимо произвести установку и настройку следующего программного обеспечения:

1. СУБД PostgreSQL с PgBouncer;
2. кластер k8s;
3. nexus с docker;
4. кластер kafka;
5. кластер minioS3;
6. ELK.

3.1 Установка и настройка PostgreSQL

3.1.1 Установка PostgreSQL 14.x на выделенную виртуальную машину

Выполнить следующие команды:

```
sudo wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc
```

```
sudo apt-key add -
```

```
sudo echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" |sudo tee
/etc/apt/sources.list.d/pgdg.list
```

```
sudo apt-get update
```

```
sudo apt-get install -y postgresql-14 postgresql-contrib-14 postgresql-14-cron
```

Разрешение подключения с внешних ip

```
sudo nano /etc/postgresql/14/main/postgresql.conf
```

```
listen_addresses = '*'
```

Разрешение подключения с внешних и внутренних ip

```
sudo nano /etc/postgresql/14/main/pg_hba.conf
```

```
# "local" is for Unix domain socket connections only
```

```
local all all peer
```

```
# IPv4 local connections:
```

```
host all postgres 127.0.0.1/32 trust
```

```
host all postgres (Ваша подсеть к примеру 192.168.0.0)/16 md5
```

```
# IPv6 local connections:
```

```
# host all all ::1/128 scram-sha-256
```

```
# Allow replication connections from localhost, by a user with the
```

```
# replication privilege.
```

```
local replication all peer
```

```
host replication all 127.0.0.1/32 scram-sha-256
```

```
host replication all ::1/128 scram-sha-256
```

Для настройки доступа для нового пользователя с правами администратора выполнить следующие команды:

```
sudo -u postgres psql
```

```
create user iac with password '....';
```

```
alter user iac with superuser;
```

3.1.2 Настройка PostgreSQL

В файле postgresql.conf разрешить доступ с внешних ip и настроить логирование выполнив следующие действия:

```
sudo nano /etc/postgresql/14/main/postgresql.conf
```

Раскомментировать следующие строки и привести их к виду:

```
listen_addresses = '*'
```

```
port = 5432
```

```
max_connections = 300
```

```
log_destination = 'stderr'
```

```
logging_collector = on
```

```
log_directory = '/var/log/postgres'
```



```

log_filename = 'postgresql-%a.log'
log_truncate_on_rotation = on
log_rotation_age = 3d
log_rotation_size = 100
log_min_duration_statement = 0
log_line_prefix = '< %m > '
log_timezone = 'Europe/Moscow'
track_activity_query_size = 5000
datestyle = 'iso, dmy'
lc_messages = 'en_US.UTF-8'
lc_monetary = 'ru_RU.UTF-8'
lc_numeric = 'ru_RU.UTF-8'
lc_time = 'ru_RU.UTF-8'
timezone = 'Europe/Moscow'
default_text_search_config = 'pg_catalog.russian'
max_locks_per_transaction = 150
effective_cache_size = xxGB # чуть меньше 2/3 от доступного объема памяти
shared_buffers = xGB # 1/8 RAM или больше (1/6) (но не более 1/4)
maintenance_work_mem = xMB (рекомендуемый размер 1/4 RAM)
work_mem = xGB (рекомендуемый размер 1/20 RAM)
temp_buffers = xGB (рекомендуемый размер 1/20 RAM)
huge_pages = try
effective_io_concurrency = 200
max_worker_processes = 12
max_parallel_workers_per_gather = 4
max_parallel_maintenance_workers = 4
max_parallel_workers = 8
wal_level = logical
wal_buffers = 128MB
checkpoint_completion_target = 0.7
max_wal_size = 4GB
min_wal_size = 1GB
max_wal_senders = 20
max_replication_slots = 20
max_logical_replication_workers = 9
max_sync_workers_per_subscription = 5
enable_seqscan = on
random_page_cost = 1.1
default_statistics_target = 100
gin_fuzzy_search_limit = 0
max_locks_per_transaction = 150
jit = off
password_encryption = md5
ssl = off
effective_io_concurrency = 200
default_statistics_target = 100
log_min_messages = warning
log_min_error_statement = error
log_min_duration_statement = -1

```

Запустить pgsq1 и сделать автозагрузку сервиса выполнив следующие команды:

```
sudo systemctl start postgresql
```

```
sudo systemctl enable postgresql
sudo systemctl status postgresql
```

3.1.3 Установка и настройка pgbouncer на сервер с postgresql

Для установки pgbouncer выполнить следующие команды:

```
sudo apt-get install pgbouncer
```

Проверить установку pgbouncer командой:

```
sudo systemctl status pgbouncer
```

pgbouncer.service - pgbouncer PostgreSQL connection pooler

Loaded: loaded (/lib/systemd/system/pgbouncer.service; enabled; vendor preset: enabled)

Active: active (running) since Fri 2022-08-26 12:22:30 MSK; 1 months 14 days ago

Main PID: 987681 (pgbouncer)

Status: "stats: 0 хacts/s, 0 queries/s, in 198 B/s, out 903 B/s, хact 60136 μs, query 23700 μs, wait 22 μs"

Tasks: 2 (limit: 4541)

Memory: 7.1M

CGroup: /system.slice/pgbouncer.service

└─987681 /usr/sbin/pgbouncer /etc/pgbouncer/pgbouncer.ini

Открыть конфигурационный файл, выполнив команду:

```
sudo nano /etc/pgbouncer/pgbouncer.ini
```

где проверить и скорректировать настройки:

[databases]

* = host=localhost port=5432

[pgbouncer]

admin_users=postgres

logfile = /var/log/postgresql/pgbouncer.log

pidfile = /var/run/postgresql/pgbouncer.pid

listen_addr = *

listen_port = 6432

auth_type = md5

auth_file = /etc/pgbouncer/userlist.txt

pool_mode = transaction

server_reset_query = DISCARD ALL

ignore_startup_parameters = extra_float_digits

max_client_conn = 1000

default_pool_size = 180

Открыть конфигурационный файл следующей командой:

```
sudo nano /etc/init.d/pgbouncer
```

где проверить и скорректировать настройки:

добавить строчку 'ulimit -n 100000' в блок case "\$1" in новой строкой после строки

```
test -d $PIDDIR || install -d -o postgres -g postgres -m 2775 $PIDDIR
```

Открыть конфигурационный файл следующей командой:

```
nano /usr/lib/systemd/system/pgbouncer.service
```

где проверить и скорректировать настройки:

в раздел Service добавить строчку

```
LimitNOFILE=100000
```

проверим лимит максимально открытых файлов для пользователя postgres:

```
sudo su - postgres -c 'ulimit -n'
```

```
sudo nano /etc/security/limits.conf
```

где добавить строки:

```
* hard nofile 100000
```

```
* soft nofile 100000
```

 последний параметр 'session required pam_limits.so' - для ОС Ubuntu

sudo su - postgres -c 'ulimit -n'

Далее добавим пользователя в userlist

sudo nano /etc/pgbouncer/userlist.txt

"iac" "пароль" – кавычки нужны, пароль в md5 не сгенерированный, а взятый из postgresql

sudo su postgres

psql

select * from pg_shadow;

скопировать пароль md5 для iac

Перезапустить pgbouncer командой:

sudo systemctl daemon-reload

sudo systemctl restart pgbouncer

Проверить статус сервиса командой

sudo /etc/init.d/pgbouncer status

3.2 Установка и настройка кластера k8s (kubernetes)

3.2.1 Установка k8s (kubernetes) – управляющие(master) узлы

Для установки containerd выполнить следующие команды:

sudo -i

sudo apt install containerd

Для настройки containerd создать файл конфигурации командами:

sudo mkdir -p /etc/containerd

containerd config default > /etc/containerd/config.toml

Запустить containerd:

sudo systemctl enable containerd && systemctl start containerd && systemctl status containerd

Отключить брандмауэр или прописать разрешение служебной сетью Kubernetes 10.32.0.0/12:

sudo ufw disable или ufw allow from 10.32.0.0/12

Прописать имя хоста в /etc/hostname, если не прописано.

Прописать настраиваемые узлы в /etc/hosts, в том числе и универсальный ip – адрес всех управляющих серверов, единый для всех. К примеру 10.10.10.10 kuber (при условии, что остальные узлы 10.10.10.0-9,11-255)

Отключить swap - для этого комментируем строку в файле /etc/fstab

/swap.img none swap sw 0 0

В файле /etc/sysctl.conf раскомментировать строку:

net.ipv4.ip_forward=1

Далее в консоли воспроизвести команды:

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

br_netfilter

EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

EOF

Перезагрузить сервер

Выполнить следующие команды:

sudo modprobe overlay

sudo modprobe br_netfilter

Произвести инсталляцию утилит kubernetes (с ее помощью производится развертывание продукта), kubectl (управление системой) и управляющего процесса kubelet, выполнив следующие команды:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg
sudo echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Установка первого узла управляющей плоскости

Для обеспечения отказоустойчивости управляющей плоскости кластера установим инструмент keepalived командой:

```
sudo apt install keepalived
```

Создадим файл управления установкой для серверов управляющей плоскости

```
sudo nano /etc/keepalived/keepalived.conf:
```

```
vrrp_instance VI_101 {
    state BACKUP
    interface eth0
    virtual_router_id 101
    priority 100
    advert_int 1
    nopreempt
    authentication {
        auth_type AH
        auth_pass iecl10pee
    }
    virtual_ipaddress {
        172.16.1.240
    }
}
```

Здесь 172.16.1.240 – виртуальный ip-адрес, находящийся в той же подсети, что и ip-адреса хостов управляющей плоскости, единый для всех этих хостов, желательно отличаться от рабочей плоскости к примеру 10.10.10.10 ,а у рабочей 10.10.10.11.

auth_pass - должен отличаться хотябы на 1 символ от рабочей плоскости,так же как и vrrp_instance VI_* и virtual_router_id *

Запустить службу keepalived следующей командой:

```
sudo systemctl enable keepalived && systemctl start keepalived && systemctl status keepalived
```

Создать файл управления инсталляцией

```
sudo nano /opt/kubeadm-config.yaml:
```

```
apiVersion: kubeadm.k8s.io/v1beta3
kind: InitConfiguration
```

```

nodeRegistration:
localAPIEndpoint:
  advertiseAddress: "172.16.1.201"
---
apiVersion: kubeadm.k8s.io/v1beta3
kind: ClusterConfiguration
kubernetesVersion: stable
apiServer:
  certSANs:
    - "127.0.0.1"
    - "172.16.1.240"
    - "172.16.1.201"
    - "172.16.1.202"
    - "172.16.1.203"
networking:
  podSubnet: 10.32.0.0/12
  controlPlaneEndpoint: "172.16.1.240:6443"

```

Где advertiseAddress: "172.16.1.201" текущий сервер на котором проводится разворачивание k8s, на остальных будет 202-204 и т.д. В пункт certSANs прописываете все ваши будущие управляющие узлы. В строке controlPlaneEndpoint: "xxx.xxx.xxx.xxx:6443" прописать свой виртуальный ip-адрес, единый для всех управляющих узлов. Он так же прописывается в общий список.

После разворачивания k8s и всех узлов, этот файл нужно убрать из этой папки куда-либо в архив или в защищенное хранилище.

Провести развертывание:

```
cd /opt
```

```
sudo kubeadm init --config=kubeadm-config.yaml
```

В результате должен получиться приблизительно такой вывод:

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

You can now join any number of control-plane nodes by copying certificate authorities and service account keys on each node and then running the following as root:

```

kubeadm join 172.16.1.201:6443 --token 8ooy30.i84xjwv8azfpynrd \
  --discovery-token-ca-cert-hash
sha256:4c065256c63c5c7dd7f5f21fb00f18dd9c4b627b075b3add3a1a7 \
  --control-plane

```

Then you can join any number of worker nodes by running the following on each as root:

```

kubeadm join 172.16.1.201:6443 --token 8ooy30.i84xjwv8azfpynrd \
  --discovery-token-ca-cert-hash
sha256:4c065256c63c5c7dd7f5f21fb00f18dd9c4b627b075b3add3a1a7

```

Пользуясь указанными в блоке вывода командами, возможно установить в кластер kubernetes дополнительные узлы. В связи с этим вывод рекомендуется сохранить. Токены в составе команд действительны в течение 1 суток.

Если со времени инсталляции первого узла прошло более суток и токен устарел, можно получить новый, выполнив на нем команду:

```
sudo kubeadm token create --print-join-command
```

Настроить текущий узел для упрощения администрирования кластера из командной строки. Для этого выполнить следующие команды:

```
sudo mkdir -p ~/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf ~/.kube/config
```

Установить CNI-службу Weave.

Внимание! Перед установкой Weave убедитесь в том, что идентификатор узла кластера в файле /etc/machine-id для разных узлов кластера - разный! Он может оказаться одинаковым, если узлами кластера являются виртуальные машины, установленные методом клонирования. Weave по умолчанию выбирает подсеть для узла на основе этого идентификатора, что в случае идентичности идентификаторов на разных узлах приводит к конфликтам IP-адресов контейнеров.

Для установки выполнить следующие команды:

```
sudo kubectl --kubeconfig /etc/kubernetes/admin.conf apply -f
```

<https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml>

Установить остальные узлы управляющей плоскости. Выполняем всю инструкцию вплоть до раздела «Запускаем службу keepalived», но keepalived временно останавливаем командой:

```
sudo systemctl stop keepalived
```

После выполнить следующие задачи:

Копировать с первого узла на новый узел ключевые пары /etc/kubernetes/pki/ca.*, /etc/kubernetes/pki/front-proxy-ca.*, /etc/kubernetes/pki/sa.*, /etc/kubernetes/pki/etcd/ca.* в одноименные каталоги.

```
sudo kubeadm join xxx.xxx.xxx.xxx:6443 --token qu0cbm.32ublwswwkhdv417 --discovery-token-ca-cert-hash
```

```
sha256:c936b6b86007dbe5fbe953c71f4371c6311ae87c1fb5478bab00f8f2e59c2512 --control-plane
```

Здесь xxx.xxx.xxx.xxx - виртуальный ip-адрес управляющей плоскости, значения параметров --token и --discovery-token-ca-cert-hash берутся из вывода на консоль в результате установки первого узла либо в результате выполнения команды:

```
sudo kubeadm token create --print-join-command
```

Убедиться, что интеграция нового узла управляющей плоскости в кластер осуществилась, на Рисунок 1

```
sudo kubectl get nodes -o=wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
test-kuber-1	Ready	control-plane	69d	v1.25.3	172.16.1.201	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-1w	Ready	<none>	69d	v1.25.3	172.16.1.204	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-2	Ready	control-plane	69d	v1.25.3	172.16.1.202	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-2w	Ready	<none>	69d	v1.25.3	172.16.1.205	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-3	Ready	control-plane	69d	v1.25.3	172.16.1.203	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-3w	Ready	<none>	69d	v1.25.3	172.16.1.206	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9

Рисунок 1 – Осуществление интеграции нового узла управляющей плоскости в кластер

Настроить текущий узел для упрощения администрирования кластера из командной строки:

```
sudo mkdir -p ~/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf ~/.kube/config
```

```
sudo systemctl start keepalived
```

Далее если требуется больше управляющих узлов (обычно их минимум 3), повторяем инструкцию для вторичных узлов.

3.2.2 Установка k8s (kubernetes) – рабочие узлы(work)

Установить containerd, выполнив следующие команды:

```
sudo apt install containerd
```

Настроить containerd, для чего создать файл конфигурации командами:

```
sudo mkdir -p /etc/containerd
```

```
sudo containerd config default > /etc/containerd/config.toml
```

Запустить containerd, выполнив команду:

```
sudo systemctl enable containerd && systemctl start containerd && systemctl status containerd
```

Отключить брандмауэр или прописать разрешение служебной сетью Kubernetes 10.32.0.0/12.

```
sudo ufw disable или ufw allow from 10.32.0.0/12
```

Прописать имя хоста в /etc/hostname если не прописано

Прописать настраиваемые узлы в /etc/hosts в том числе и универсальный ip всех управляющих серверов ,единый для всех. К примеру 10.10.10.10 kuber(при условии, что остальные узлы 10.10.10.0-9,11-255)

Отключить swar. Для этого комментировать строку в файле /etc/fstab

```
# /swap.img none swap sw 0 0
```

В файле /etc/sysctl.conf раскомментировать строку:

```
net.ipv4.ip_forward=1
```

Далее в консоли выполнить команды:

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
```

```
br_netfilter
```

```
EOF
```

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
```

```
net.bridge.bridge-nf-call-iptables = 1
```

```
EOF
```

Перезапустить сервер, выполнив следующие команды:

```
sudo modprobe overlay
```

```
sudo modprobe br_netfilter
```

Произвести установку утилит kubeadm (с ее помощью производится развертывание продукта), kubectl (управление системой) и управляющего процесса kubelet, выполнив следующие команды:

```
sudo apt-get update && sudo apt-get install -y apt-transport-https ca-certificates curl
```

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
```

```
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
sudo echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
```

```
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update
```

```
sudo apt-get install -y kubelet kubeadm kubectl
```

```
sudo apt-mark hold kubelet kubeadm kubectl
```

Установить первый узел управляющей плоскости.

Для обеспечения отказоустойчивости управляющей плоскости кластера установить инструмент keepalived командой:

```
sudo apt install keepalived
```

Создать файл управления инсталляцией для серверов с рабочими узлами (рабочей плоскости) /etc/keepalived/keepalived.conf:

```

vrrp_instance VI_102 {
    state BACKUP
    interface eth0
    virtual_router_id 102
    priority 100
    advert_int 1
    nopreempt
    authentication {
        auth_type AH
        auth_pass iechl1pee
    }
    virtual_ipaddress {
        172.16.1.244
    }
}

```

Здесь 172.16.1.244 - виртуальный ip-адрес, находящийся в той же подсети, что и ip-адреса хостов рабочей плоскости, единый для всех этих хостов, желательно отличаться от управляющей плоскости к примеру 10.10.10.10 ,а у рабочей 10.10.10.11.

auth_pass - должен отличаться хотя бы на 1 символ от управляющей плоскости, так же как и vrrp_instance VI_* и virtual_router_id *

Активировать службу keepalived, но не запускать. Для этого использовать команду:

```
sudo systemctl enable keepalived
```

```

sudo kubeadm join xxx.xxx.xxx.xxx:6443 --token qu0cbm.32ublwswwkhdv417 --discovery-
token-ca-cert-hash
sha256:c936b6b86007dbe5fbe953c71f4371c6311ae87c1fb5478bab00f8f2e59c2512

```

Здесь xxx.xxx.xxx.xxx - виртуальный ip-адрес управляющей плоскости, значения параметров --token и --discovery-token-ca-cert-hash берутся из вывода на консоль в результате инсталляции первого узла либо в результате выполнения команды на управляющем узле:

```
sudo kubeadm token create --print-join-command
```

```
sudo kubectl get nodes -o=wide
```

Результат выполнения команды представлен на Рисунок 2

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
test-kuber-1	Ready	control-plane	69d	v1.25.3	172.16.1.201	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-1w	Ready	<none>	69d	v1.25.3	172.16.1.204	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-2	Ready	control-plane	69d	v1.25.3	172.16.1.202	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-2w	Ready	<none>	69d	v1.25.3	172.16.1.205	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-3	Ready	control-plane	69d	v1.25.3	172.16.1.203	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9
test-kuber-3w	Ready	<none>	69d	v1.25.3	172.16.1.206	<none>	Ubuntu 20.04.4 LTS	5.4.0-135-generic	containerd://1.5.9

Рисунок 2 – Результат выполнения команды

После, перезагрузить все рабочие узлы.

3.3 Установка Nexus с docker

3.3.1 Установка nexus

Выполнить следующие команды:

```
sudo apt install openjdk-8-jre-headless
```

```
sudo useradd -M -d /opt/nexus -s /bin/bash -r nexus
```

```
sudo echo "nexus ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/nexus
```

```
sudo wget https://download.sonatype.com/nexus/3/latest-unix.tar.gz
```

```
sudo tar -zxvf latest-unix.tar.gz -C /opt/
```

```
sudo mv /opt/nexus-3.30.1-01 /opt/nexus
```

```
sudo chown -R nexus:nexus /opt/nexus /opt/sonatype-work
```



```
sudo nano /opt/nexus/bin/nexus.rc
раскомментировать run_as_user="nexus"
sudo nano /etc/systemd/system/nexus.service
Создать юнит, выполнив следующие действия:
```

```
[Unit]
```

```
Description=nexus service
```

```
After=network.target
```

```
[Service]
```

```
Type=forking
```

```
LimitNOFILE=65536
```

```
ExecStart=/opt/nexus/bin/nexus start
```

```
ExecStop=/opt/nexus/bin/nexus stop
```

```
User=nexus
```

```
Restart=on-abort
```

```
[Install]
```

```
WantedBy=multi-user.target
```

```
Запустить сервис, выполнив команду:
```

```
sudo systemctl daemon-reload && systemctl start nexus && systemctl enable nexus
```

```
Скопировать сгенерированный пароль для admin:
```

```
nano /opt/sonatype-work/nexus3/admin.password
```

```
Перейти в браузере по ip-адрес-nexus и порт 8081(Прим. 0.0.0.0:8081)
```

В правом верхнем углу нажать кнопку «sign in», ввести логин «admin» и пароль, который был скопирован ранее.

Для использования nexus как хранилище docker registry требуется установить docker.

3.3.2 Установка docker

Выполнить следующие команды:

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

```
sudo apt update
```

```
sudo apt-cache policy docker-ce
```

```
sudo apt install docker-ce
```

```
Запустим сервис и проверим
```

```
sudo systemctl enable docker && systemctl status docker
```

```
Создать файл daemon.json в /etc/docker/
```

```
{
  "insecure-registries" : [
    "ip-адрес сервера nexus:8085"
  ]
}
```

Перезапустить docker командой:

```
sudo systemctl restart docker
```

Перейти в браузере по ip адресу nexus и порт 8081(Прим. 0.0.0.0:8081)

Зайти в настройки под учетной записью «admin» в раздел «repositories», далее нажать кнопку «create repositories» и выбрать docker(hosted) как показано на Рисунок 3.

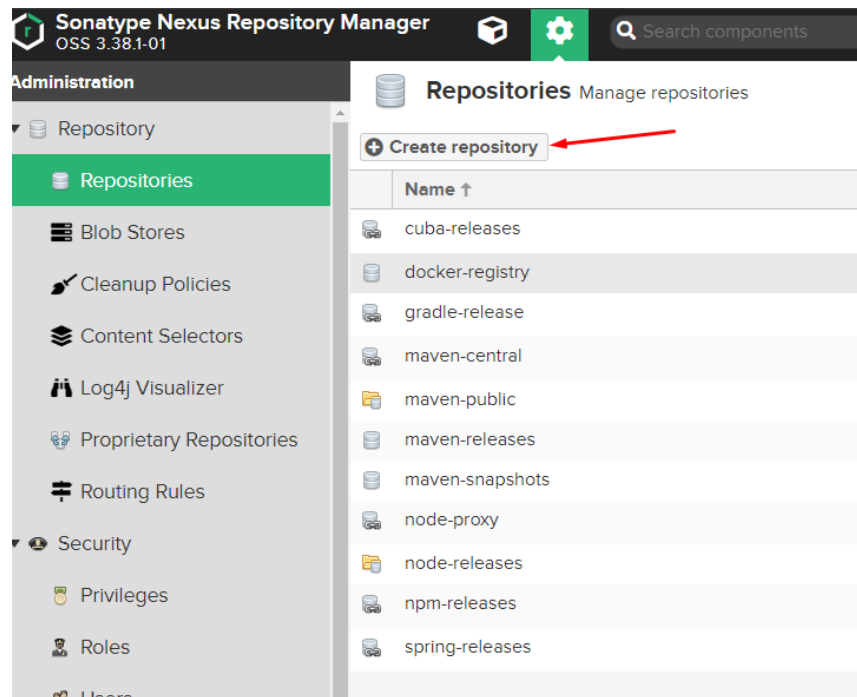


Рисунок 3 – Добавление репозитория

Указываем имя, порт и ставим галку V1 API как показано на Рисунок 4.

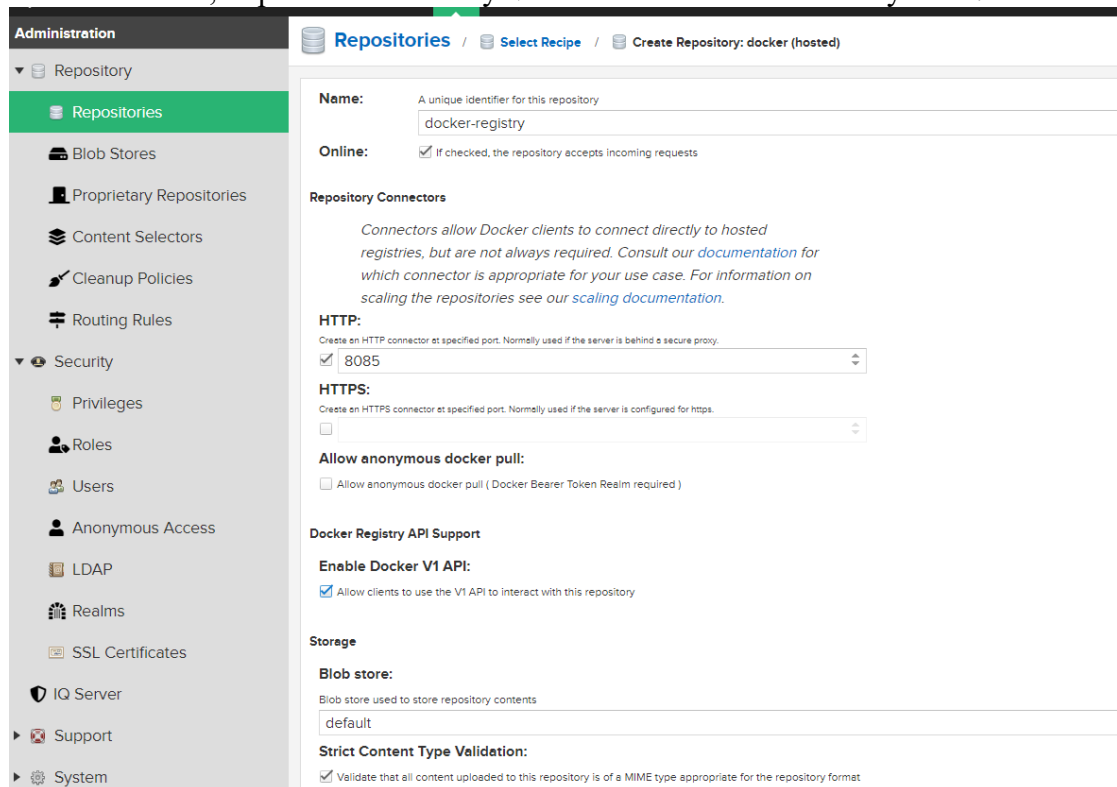


Рисунок 4 – Настройка docker

Далее внизу нажать кнопку «create».

В консоли сервера nexus произвести вход в nexus-docker следующей командой:
`sudo docker login -u admin ip-адрес-nexus:8085` и пароль admin

Создать задачи автоматизации nexus для очистки кэша, блоб и устаревших докер контейнеров. Для этого перейти в браузере по ip адресу nexus и порт 8081(Прим. 0.0.0.0:8081). Зайти в настройки из-под учетной записи «admin», в раздел «system-tasks». Нажать кнопку «create task» и выбрать тип таска и настройки как указано на Рисунок 5 – Рисунок 7.

Tasks / compact blob (Admin - Compact blob store)

Delete task Run Stop

Summary Settings

Task enabled:
☒ This flag determines if the task is currently active. To disable this task for a period of time, de-select this checkbox.

Task name:
 A name for the scheduled task
 compact blob

Notification email:
 The email address where an email will be sent in case that task execution fails

Blob store:
 Select the blob store to compact
 default

Task frequency:
 The frequency this task will run. Manual - this task can only be run manually. Once - run the task once at the specified date/time. Daily - run the task every day at the specified time. Weekly - run the task every week on the specified day at the specified time. Monthly - run the task every month on the specified day(s) and time. Advanced - run the task using the supplied cron string
 Daily

Start date:
 01/15/2019

Time to run this task:
 The time this task should start on days it will run in your time zone GMT+0300 (Москва, стандартное время).
 00:45

Save Discard

Рисунок 5 – Выполнение настройки

Tasks / clear (Maven - Delete SNAPSHOT)

Delete task Run Stop

Summary Settings

Task enabled:
☒ This flag determines if the task is currently active. To disable this task for a period of time, de-select this checkbox.

Task name:
 A name for the scheduled task
 clear

Notification email:
 The email address where an email will be sent in case that task execution fails

Repository:
 Select the Maven repository or repository group to remove snapshots from.
 (All Repositories)

Minimum snapshot count:
 Minimum number of snapshots to keep for one GAV.
 2

Snapshot retention (days):
 Delete all snapshots older than this, provided we still keep the minimum number specified.
 0

Remove if released:
☐ Delete all snapshots that have a corresponding release



Grace period after release (days):
 The grace period during which snapshots with an associated release will not be deleted.




Task frequency:
 The frequency this task will run. Manual - this task can only be run manually. Once - run the task once at the specified date/time. Daily - run the task every day at the specified time. Weekly - run the task every week on the specified day at the specified time. Monthly - run the task every month on the specified day(s) and time. Advanced - run the task using the supplied cron string
 Daily

Start date:
 01/15/2019

Time to run this task:
 The time this task should start on days it will run in your time zone GMT+0300 (Москва, стандартное время).
 00:00

Рисунок 6 – Выполнение настройки

 **Tasks** /  **docker-clean (Docker - Delete unused manifests and images)**

 Delete task  Run  Stop

[Summary](#) [Settings](#)

Task enabled:
☒ This flag determines if the task is currently active. To disable this task for a period of time, de-select this checkbox.

Task name:
 A name for the scheduled task

Notification email:
 The email address where an email will be sent if the condition below is met

Send notification on:
 Conditions that will trigger a notification email

Repository:
 Select the Docker repository to clean up

Deploy offset in hours:
 Manifests and images deployed within this period before the task starts will not be deleted

Task frequency:
 The frequency this task will run. Manual - this task can only be run manually. Once - run the task once at the specified date/time. Weekly - run the task every week on the specified day at the specified time. Monthly - run the task every month on the specified day(s)

Start date:

Time to run this task:
 The time this task should start on days it will run in your time zone GMT+0300 (Москва, стандартное время).

Рисунок 7 – Выполнение настройки

3.3.3 Настройка nexus-docker

Нажать на кнопку, указанной на Рисунок 8.

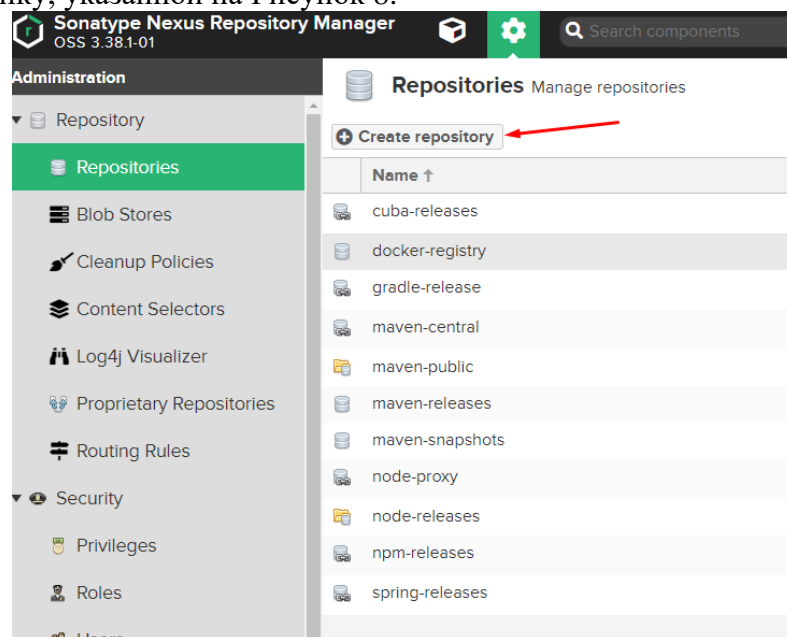


Рисунок 8 – Кнопка создания репозитория

После выбрать `docker(host)`.

После выставить такие же настройки. Порт любой не занятый, имя так же любое. `Https` лучше настраивать обратный прокси через `nginx`. Так как `nexus` работает на `eclipse jetty` нельзя использовать самоподписанные сертификаты.

Так же создать новый таск, иначе через какое-то время сервер переполнится.

После на сервере, где `nexus` выполнить команду:

```
sudo docker login -u admin 172.16.1.214:8085
```

А затем ввести пароль администратора `nexus`.

3.4 Установка и настройка кластера `kafka`

3.4.1 Установка `Java`

Выполнить следующие команды:

```
sudo apt install openjdk-8-jdk -y
```

Проверить работоспособность следующей командой:

```
java -version
```

```
openjdk version "1.8.0_342"
```

```
OpenJDK Runtime Environment (build 1.8.0_342-8u342-b07-0ubuntu1~20.04-b07)
```

```
OpenJDK 64-Bit Server VM (build 25.342-b07, mixed mode)
```

3.4.2 Установка и настройка `zookeeper`

Выполнить следующие команды:

```
cd /opt
```

```
sudo wget http://mirror.linux-ia64.org/apache/zookeeper/zookeeper-3.6.3/apache-zookeeper-
```

```
3.6.3-bin.tar.gz
```

```
sudo tar -zxvf apache-zookeeper-3.6.3-bin.tar.gz
```

Установить и сконфигурировать `zookeeper`:

Создать каталог `/etc/zookeeper`

Копировать в него содержимое каталога `/opt/apache-zookeeper-3.6.3-bin/conf`

Создать файл `zoo.cfg`, используя шаблон `zoo_sample.cfg`

Редактировать `/etc/zookeeper/zoo.cfg`:

```
sudo nano etc/zookeeper/zoo.cfg
```

```
dataDir=/var/lib/zookeeper
```

```
autopurge.snapRetainCount=3
```

```
autopurge.purgeInterval=1
```

```
server.1=172.16.1.241:2888:3888
```

```
audit.enable=true
```

```
4lw.commands.whitelist=stat
```

```
admin.enableServer=false
```

Последняя строка необходима, если необходимо запретить возможность администрирования `Zookeeper` с помощью `HTTP API`. По умолчанию данная возможность активна, в результате чего `Zookeeper` занимает порт `TCP 8080`, и возможны конфликты с другим программным обеспечением. Если же требуется сохранить возможность работы с этим `API`, рекомендуется сменить порт с `8080` на другой. Это делается путем замены строки

```
admin.enableServer=false
```

на строку:

```
admin.serverPort=xxxx.
```

Если в системе более одного узла `Zookeeper`, добавляются строки

```
# server.2=<ip-kafka-server-2>:2888:3888
```

```
# server.3=<ip-kafka-server-3>:2888:3888
```

по числу узлов с указанием `ip-адресов` этих узлов.

Создать каталог `/var/lib/zookeeper`

```
sudo mkdir /var/lib/zookeeper
```

Создать в каталоге файл myid

```
sudo nano /var/lib/zookeeper/myid
```

Внести туда цифру, соответствующую ip-адресу узла из строчек
"server.N=xxx.xxx.xxx.xxx:2888:3888" в файле zoo.cfg:

N

Создать файл /opt/apache-zookeeper-3.6.3-bin/bin/zkStart.sh

```
sudo nano /opt/apache-zookeeper-3.6.3-bin/bin/zkStart.sh
```

Содержимое:

```
#!/usr/bin/env bash
```

```
ZOO_LOG_DIR=/var/log/zookeeper ZOO_LOG4J_PROP='INFO,ROLLINGFILE'
```

```
/opt/apache-zookeeper-3.6.3-bin/bin/zkServer.sh start
```

Сделать zkStart.sh исполняемым (755).

Создать файл /etc/systemd/system/zk.service

Содержимое:

```
[Unit]
```

```
Description=Zookeeper Daemon
```

```
Documentation=http://zookeeper.apache.org
```

```
Requires=network.target
```

```
After=network.target
```

```
[Service]
```

```
Type=forking
```

```
WorkingDirectory=/opt/apache-zookeeper-3.6.3-bin
```

```
ExecStart=/opt/apache-zookeeper-3.6.3-bin/bin/zkstart.sh
```

```
ExecStop=/opt/apache-zookeeper-3.6.3-bin/bin/zkServer.sh stop
```

```
TimeoutSec=30
```

```
Restart=on-failure
```

```
[Install]
```

```
WantedBy=default.target
```

```
sudo nano /etc/zookeeper/conf/myid
```

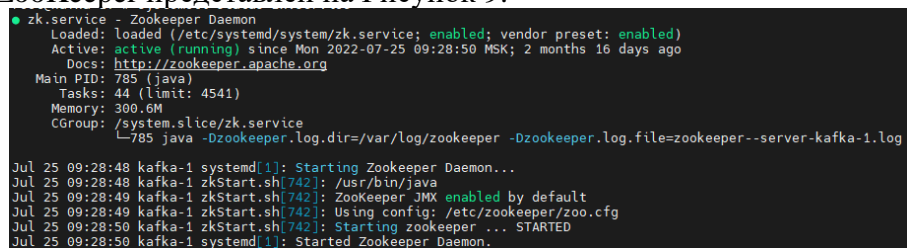
где заменить всё на "1" для первого сервера, для второго – на "2", для третьего – на "3")

Проверить работоспособность, выполнив команду:

```
sudo systemctl restart zookeeper
```

```
sudo systemctl status zookeeper
```

Статус ZooKeeper представлен на Рисунок 9.



```
● zk.service - Zookeeper Daemon
   Loaded: loaded (/etc/systemd/system/zk.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-07-25 09:28:50 MSK; 2 months 16 days ago
     Docs: http://zookeeper.apache.org
   Main PID: 785 (java)
    Tasks: 44 (limit: 4541)
   Memory: 300.6M
    CGroup: /system.slice/zk.service
           └─785 java -Dzookeeper.log.dir=/var/log/zookeeper -Dzookeeper.log.file=zookeeper--server-kafka-1.log

Jul 25 09:28:48 kafka-1 systemd[1]: Starting Zookeeper Daemon...
Jul 25 09:28:48 kafka-1 zkStart.sh[742]: /usr/bin/java
Jul 25 09:28:49 kafka-1 zkStart.sh[742]: ZooKeeper JMX enabled by default
Jul 25 09:28:49 kafka-1 zkStart.sh[742]: Using config: /etc/zookeeper/zoo.cfg
Jul 25 09:28:50 kafka-1 zkStart.sh[742]: Starting zookeeper ... STARTED
Jul 25 09:28:50 kafka-1 systemd[1]: Started Zookeeper Daemon.
```

Рисунок 9 – Статус ZooKeeper

```
telnet 172.16.1.241 2181
```

3.4.3 Установка Kafka.

Скачать архив и подготовить каталоги:

```
cd /opt/
```

```
sudo curl https://apache-mirror.rbc.ru/pub/apache/kafka/3.0.2/kafka_2.13-3.0.2.tgz
```

```
sudo cp ./kafka_2.13-3.0.2.tgz ./kafka.tgz
```

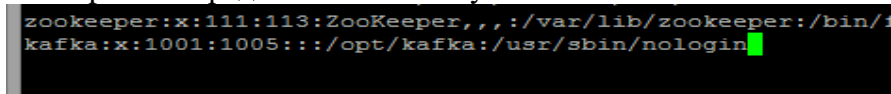
```
sudo rm ./kafka_2.13-3.0.2.tgz
```

```
sudo tar -xf kafka.tgz
```

```
sudo mv kafka_2.13-3.0.2 kafka
```

```
sudo useradd kafka
sudo nano /etc/passwd
```

где kafka:x:1000:1000::/home/kafka:/bin/sh – меняем
на kafka:x:1000:1000::/opt/kafka:/usr/sbin/nologin
Настройка /etc/passwd представлена на Рисунок 10.



```
zookeeper:x:111:113:ZooKeeper,,,:/var/lib/zookeeper:/bin/sh
kafka:x:1001:1005:::/opt/kafka:/usr/sbin/nologin
```

Рисунок 10 – Настройка /etc/passwd

```
sudo chown -R kafka:kafka /opt/kafka/
sudo chmod -R 755 /opt/kafka/
sudo mkdir /var/log/kafka-logs/
sudo chown -R kafka:kafka /var/log/kafka-logs/
sudo chmod -R 0755 /var/log/kafka-logs/
```

Настроить конфигурационный файлы службы server:

```
sudo nano /opt/kafka/config/server.properties
```

пример конфигурации server.properties:

```
broker.id=1(кластер, 1 сервер "1", второй "2" и т.д.)
listeners=PLAINTEXT:// 172.16.1.241:9092
advertised.listeners=PLAINTEXT:// 172.16.1.241:9092(кластер, 1 сервер
"172.16.1.241:9092" ,второй "172.16.1.242:9092" и т.д )
log.dirs=/var/log/kafka-logs
num.partitions=4
num.recovery.threads.per.data.dir=1
zookeeper.connect=172.16.1.241:2181(если кластер то на каждом сервере прописать
172.16.1.241:2181, 172.16.1.242:2181, 172.16.1.243:2181)
zookeeper.connection.timeout.ms=18000
group.initial.rebalance.delay.ms=0
```

Следующие настройки server.properties добавляются только при кластерном режиме из 3х серверов kafka :

```
default.replication.factor=3
min.insync.replicas=2
offsets.topic.replication.factor=3
transaction.state.log.replication.factor=3
transaction.state.log.min.isr=1
acks=all
```

Настроить конфигурационный файлы службы zookeeper:

```
sudo nano /opt/kafka/config/zookeeper.properties
```

пример конфигурации zookeeper.properties для каждого сервера кластера:

```
dataDir=/tmp/zookeeper
clientPort=2181
maxClientCnxns=0
admin.enableServer=false
```

Настроить остальные службы:

```
sudo nano /opt/kafka/config/producer.properties
sudo nano /opt/kafka/config/consumer.properties
sudo nano /opt/kafka/config/connect-distributed.properties
sudo nano /opt/kafka/config/connect-standalone.properties
```

куда внесём данные о кластере:

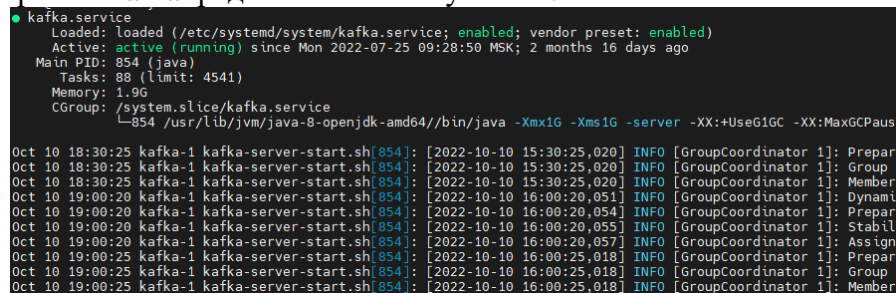
```
bootstrap.servers=172.16.1.241:9092(если кластер то на каждом сервере прописать
172.16.1.241:9092, 172.16.1.242:9092, 172.16.1.243:9092)
```

Сделать юнит, выполнив следующие команды:

```

sudo nano /etc/systemd/system/kafka.service
[Unit]
Requires=zookeeper.service
After=zookeeper.service
[Service]
Type=simple
User=kafka
Environment="JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/"
ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties
ExecStop=/opt/kafka/bin/kafka-server-stop.sh
Restart=on-abnormal
RestartSec=10
[Install]
WantedBy=multi-user.target
запустим и проверим:
sudo systemctl daemon-reload
sudo systemctl enable kafka.service
sudo systemctl start kafka.service
sudo systemctl status kafka.service
Статус сервиса Kafka представлен на Рисунок 11.

```



```

● kafka.service
   Loaded: loaded (/etc/systemd/system/kafka.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-07-25 09:28:50 MSK; 2 months 16 days ago
     Main PID: 854 (java)
        Tasks: 88 (limit: 4541)
       Memory: 1.9G
      CGroup: /system.slice/kafka.service
              └─854 /usr/lib/jvm/java-8-openjdk-amd64/bin/java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPaus

Oct 10 18:30:25 kafka-1 kafka-server-start.sh[854]: [2022-10-10 15:30:25,020] INFO [GroupCoordinator 1]: Prepar
Oct 10 18:30:25 kafka-1 kafka-server-start.sh[854]: [2022-10-10 15:30:25,020] INFO [GroupCoordinator 1]: Group
Oct 10 18:30:25 kafka-1 kafka-server-start.sh[854]: [2022-10-10 15:30:25,020] INFO [GroupCoordinator 1]: Member
Oct 10 19:00:20 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:20,051] INFO [GroupCoordinator 1]: Dynam
Oct 10 19:00:20 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:20,054] INFO [GroupCoordinator 1]: Prepar
Oct 10 19:00:20 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:20,055] INFO [GroupCoordinator 1]: Stabl
Oct 10 19:00:20 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:20,057] INFO [GroupCoordinator 1]: Assign
Oct 10 19:00:25 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:25,018] INFO [GroupCoordinator 1]: Prepar
Oct 10 19:00:25 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:25,018] INFO [GroupCoordinator 1]: Group
Oct 10 19:00:25 kafka-1 kafka-server-start.sh[854]: [2022-10-10 16:00:25,018] INFO [GroupCoordinator 1]: Member

```

Рисунок 11 – Статус сервиса Kafka

3.4.4 Проверка кластера и служб.

Проверить, что логи пишутся в /var/log/kafka-logs/, выполнив команду:

```
sudo ls -al /var/log/kafka-logs/
```

Проверить окружение командой:

```
sudo jps
```

```
15926 QuorumPeerMain
```

```
28408 Kafka
```

```
28925 Logstash
```

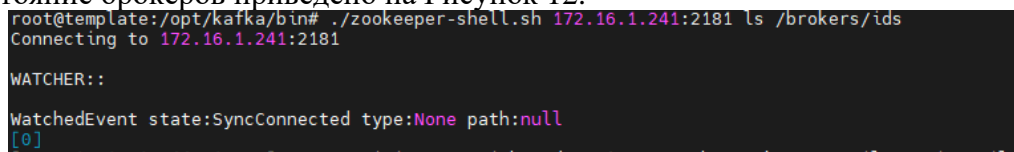
```
3182 Jps
```

Проверить, что все брокеры регистрируются в zookeeper командой:

```
cd /opt/kafka/
```

```
./bin/zookeeper-shell.sh 172.16.1.241:2181 ls /brokers/ids
```

Состояние брокеров приведено на Рисунок 12.



```

root@template:/opt/kafka/bin# ./zookeeper-shell.sh 172.16.1.241:2181 ls /brokers/ids
Connecting to 172.16.1.241:2181

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[0]

```

Рисунок 12 – Состояние брокеров

Создать топик тремя разделами и фактором репликации 3 (запустить с любого сервера кластера), выполнив команду:


```
sudo ./bin/kafka-topics.sh --create --bootstrap-server 172.16.1.241:9092 --topic example-topic
--partitions 3 --replication-factor 1
```

Создание топика приведено на Рисунок 13 и Рисунок 14.

```
root@template:/opt/kafka/bin# ./kafka-topics.sh --create --bootstrap-server 172.16.1.241:9092 --topic example-topic --partitions 3 --replication-factor 1
Created topic example-topic.
root@template:/opt/kafka/bin#
```

Рисунок 13 – Создание топика

проверим топик:

```
sudo ./bin/kafka-topics.sh --bootstrap-server 172.16.1.241:9092 --describe --topic example-
topic
```

```
root@template:/opt/kafka/bin# ./bin/kafka-topics.sh --bootstrap-server 172.16.1.241:9092 --describe --topic example-topic
Topic: example-topic TopicId: PL2ScnozRCWBgxVu0BU5mg PartitionCount: 3 ReplicationFactor: 1 Configs: segment.bytes=1073741824
Topic: example-topic Partition: 0 Leader: 0 Replicas: 0 Isr: 0
Topic: example-topic Partition: 1 Leader: 0 Replicas: 0 Isr: 0
Topic: example-topic Partition: 2 Leader: 0 Replicas: 0 Isr: 0
root@template:/opt/kafka/bin#
```

Рисунок 14 – Состояние топика

Проверить подписчиков командами:

```
sudo ./bin/kafka-console-consumer.sh --bootstrap-server 172.16.1.241:2181 --topic example-
topic --from-beginning
```

```
sudo ./bin/kafka-topics.sh --list --list --bootstrap-server 172.16.1.241:9092
```

```
sudo ./bin/kafka-console-producer.sh --broker-list 172.16.1.241:9092 --topic example-topic
```

Состояние подписчиков приведено на Рисунок 15.

```
root@template:/opt/kafka/bin# ./kafka-topics.sh --list --bootstrap-server 172.16.1.241:9092
consumer_offsets
example-topic
neatsandbox-log
publickey1
root@template:/opt/kafka/bin# ./kafka-console-producer.sh --broker-list 172.16.1.241:9092 --topic example-topic
>adadas
>^Croot@template:/opt/kafka/bin#
```

Рисунок 15 – Состояние подписчиков

3.5 Установка кластера minio

3.5.1 Установка minio на одной ноде

Убедитесь, что порт 9000 открыт для входящих соединений

1. Скачать и запустить

```
sudo wget https://dl.minio.io/server/minio/release/linux-amd64/minio
```

```
sudo chmod +x minio
```

```
sudo mv minio /usr/local/bin
```

```
cd /usr/local/bin
```

Открыть <http://127.0.0.1:9000> для проверки (или <http://<ip-address>:9000/>)

Создать пользователя, из-под которого будет запускаться сервис командами:

```
sudo useradd minio-user
```

```
sudo nano /etc/passwd
```

где дописать этому пользователю `"/usr/sbin/nologin"`

Сделать директорию и дать права пользователю minio командой:

```
sudo mkdir -p minio/data/
```

```
sudo chown -R minio-user:minio-user minio/
```

Создать конфигурационный файл с такими параметрами:

```
sudo nano /etc/default/minio
```

```
MINIO_VOLUMES="/minio/data"
```

```
MINIO_OPTS="--address :9000"
```

```
MINIO_ACCESS_KEY=HET7RAEFSSHREJTJ409GCRR
```

```
MINIO_SECRET_KEY=HET7RAEFSSHREJTJ409GCRR
```

ключи, которые можно заменить на свои, первое логин, второе пароль.

Сделать сервис, запустив команду:

```
sudo nano /etc/systemd/system/minio.service
```

Запустить сервис и проверить его, выполнив следующие команды:

```
sudo systemctl daemon-reload
sudo systemctl enable minio
sudo systemctl start minio
systemctl status minio
http://<ip-address>:9000/minio/login
```

3.5.2 Установка распределенного minio

Установить Minio в распределённом режиме, выполнив следующие команды:

```
sudo wget https://dl.minio.io/server/minio/release/linux-amd64/minio
sudo chmod +x minio
sudo mv minio /usr/local/bin
```

Создать пользователя, из-под которого будет запускаться сервис командой:

```
sudo useradd -r minio-user -s /sbin/nologin
```

Проверить, выполнив команду:

```
sudo nano /etc/passwd
```

где дописать этому пользователю "/usr/sbin/nologin" командой:

дать права пользователю minio командами:

```
sudo chown minio-user:minio-user /usr/local/bin/minio
sudo chown -R minio-user:minio-user /minio/data
```

Создать конфигурационный файл с такими параметрами:

```
sudo nano /etc/default/minio
```

```
MINIO_VOLUMES="http://10.1.250.107/minio/data          http://10.1.250.108/minio/data
http://10.1.250.109/minio/data http://10.1.250.116/minio/data"
MINIO_OPTS="--address :9000"
MINIO_ACCESS_KEY=11UWNFEGSH6875FDKB5OK
MINIO_SECRET_KEY=7mlSV8uu3ESGEaJ+QEGSES43icJU9pgstRAVtmz+A
```

MINIO_VOLUMES должно быть вида:

```
MINIO_VOLUMES="http://<ip-minio-server-1>/minio/data          http://<ip-minio-server-
2>/minio/data http://<ip-minio-server-3>/minio/data http://<ip-minio-server-4>/minio/data"
```

MINIO_ACCESS_KEY , MINIO_SECRET_KEY ключи, которые можно заменить на свои, первое логин, второе пароль.

Одинаковый файл конфигурации для всех серверов, скопировать данные в такие же созданные файлы на серверах.

Погрешность времени на серверах должны быть не более 3 секунд (в крайнем случае указать на всю одинаковую дату/время -#date MMDDhhmmYYYY.ss – date 070918422019.00)

Сделать сервис, выполнив команду:

```
sudo nano /etc/systemd/system/minio.service
```

Создать unit

```
[Unit]
```

```
Description=Minio
```

```
Documentation=https://docs.minio.io
```

```
Wants=network-online.target
```

```
After=network-online.target
```

```
AssertFileIsExecutable=/usr/local/bin/minio
```

```
[Service]
```

```
WorkingDirectory=/usr/local/
```

```
User=minio-user
```

```
Group=minio-user
```

```
PermissionsStartOnly=true
```

```
EnvironmentFile=-/etc/default/minio
```

```
ExecStartPre=/bin/bash -c "[ -n \"${MINIO_VOLUMES}\" ] || echo \"Variable
MINIO_VOLUMES not set in /etc/default/minio\""
```

```

ExecStart=/usr/local/bin/minio server $MINIO_OPTS $MINIO_VOLUMES
StandardOutput=journal
StandardError=inherit
# Specifies the maximum file descriptor number that can be opened by this process
LimitNOFILE=65536
# Disable timeout logic and wait until process is stopped
TimeoutStopSec=0
# SIGTERM signal is used to stop Minio
KillSignal=SIGTERM
SendSIGKILL=no
SuccessExitStatus=0
[Install]
WantedBy=multi-user.target
Запустить сервис и проверить его, выполнив команды:
sudo systemctl daemon-reload
sudo systemctl enable minio
sudo systemctl start minio

```

3.6 Настройка ELK для сбора логов(Не обязательна к установке)

Получить источник и произвести установку с помощью следующих команд:

```

wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.2-amd64.deb
sudo dpkg -i elasticsearch-7.17.2.deb

```

```

wget https://artifacts.elastic.co/downloads/kibana/kibana-7.17.2-amd64.deb
sudo dpkg -i kibana-7.17.2-amd64.deb

```

```

wget https://artifacts.elastic.co/downloads/logstash/logstash-7.17.2-amd64.deb
sudo dpkg -i logstash-7.17.2.deb

```

Правка sudo nano /etc/elasticsearch/elasticsearch.yml

```
network.host: 10.1.250.144
```

path.data: /var/lib/elasticsearch # директория для хранения данных (по умолчанию всё уже там, просто проверить)

Правка sudo nano /etc/kibana/kibana.yml

```
server.host: "10.1.250.144"
```

Правка sudo nano /etc/logstash/logstash.yml

```
http.host: "10.1.250.144"
```

Добавляем в автозагрузку и запускаем

```
sudo systemctl enable elasticsearch.service
```

```
sudo systemctl start elasticsearch.service
```

```
sudo systemctl enable kibana.service
```

```
sudo systemctl start kibana.service
```

```
sudo systemctl start logstash.service
```

```
sudo systemctl enable logstash.service
```

Проверить работоспособность следующими командами:

```
sudo systemctl status elasticsearch.service
```

```
sudo systemctl status kibana.service
```

```
sudo systemctl status logstash.service
```

Статус сервисов ELK представлен на Рисунок 16.

```

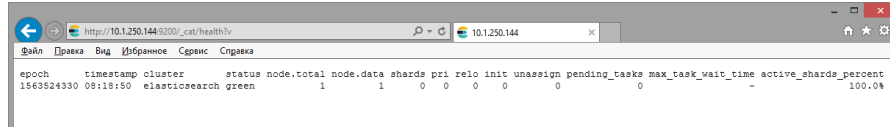
master@kafka-n1-dev:/opt/kafka$ sudo systemctl status logstash.service
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2019-07-19 11:12:19 MSK; 40s ago
     Main PID: 1101 (java)
       Tasks: 24 (limit: 4915)
   CGroup: /system.slice/logstash.service
           └─1101 /usr/bin/java -Xms1g -Xmx1g -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFr

Jul 19 11:12:19 kafka-n1-dev systemd[1]: Started logstash.
Jul 19 11:12:53 kafka-n1-dev logstash[1101]: Sending Logstash logs to /var/log/logstash which is now configured via
Jul 19 11:12:54 kafka-n1-dev logstash[1101]: [2019-07-19T11:12:54,327][INFO ][logstash.runner      ] Starting L
Jul 19 11:12:56 kafka-n1-dev logstash[1101]: [2019-07-19T11:12:56,907][ERROR][logstash.agent      ] Failed to
Jul 19 11:12:57 kafka-n1-dev logstash[1101]: [2019-07-19T11:12:57,730][INFO ][logstash.agent      ] Successful

```

Рисунок 16 - Статус сервисов ELK

Открыть ссылку через IE: [http:// 10.1.250.144:9200/_cat/health?v](http://10.1.250.144:9200/_cat/health?v)
 Статус сервисов в IE представлен на Рисунок 17.



epoch	timestamp	cluster	status	node.total	node.data	shards	pri	relo	init	unassign	pending_tasks	max_task_wait_time	active_shards_percent
1563524330	09:18:50	elasticsearch	green	1	1	0	0	0	0	0	0	0	100.0%

Рисунок 17 – Статус сервисов в IE

Настроить kafka.conf для logstash следующими командами:

```

sudo nano /etc/logstash/conf.d/kafka.conf
input {
  kafka {
    bootstrap_servers => "10.1.250.144:9092"
    topics => ["eais-log"]
    codec => json
  }
}
output {
  elasticsearch {
    hosts => ["10.1.250.144:9200"]
    index => "eais-log"
    workers => 1
  }
}

```

Логи logstash доступны в /var/log/logstash

Сброс лока индекса, при необходимости (например, автолок при нехватке места)
 доступен при помощи:

```

PUT eais-log/_settings
{
  "index": {
    "blocks": {
      "read_only_allow_delete": "false"
    }
  }
}

```