Консалтинг

Руководство по подготовке инфраструктуры для установки Т1 аналитический CRM

Оглавление

1. Глоссарий

№	Термин	Значение
1	Apache Kafka	Распределённый программный брокер сообщений. Спроектирован как распределённая, горизонтально масштабируемая система, обеспечивающая наращивание пропускной способности, как при росте числа и нагрузки со стороны источников, так и количества систем-подписчиков. Подписчики могут быть объединены в группы. Поддерживается возможность временного хранения данных для последующей пакетной обработки. Одной из особенностей реализации инструмента является применение техники, сходной с журналами транзакций, используемыми в системах управления базами данных.
2	MinIO	Высокопроизводительное объектное хранилище. Оно может обрабатывать неструктурированные данные, такие как фотографии, видео, файлы журналов, резервные копии и изображения контейнеров.
3	Redis	Резидентная система управления базами данных класса NoSQL, работающая со структурами данных типа «ключ — значение». Используется как для баз данных, так и для реализации кэшей, брокеров сообщений. Ориентирована на достижение максимальной производительности на атомарных операциях
4	PostgreSQL	Свободная объектно-реляционная система управления базами данных (СУБД).
5	Opensearch	Набор технологий, позволяющих веб-сайтам и поисковым системам публиковать результаты поиска в форматах, удобных для распространения и сбора.
6	Fluent-Bit	Программное обеспечение для сбора данных. Анализирует журналы событий, журналы приложений и потоки переходов
7	Grafana	Gрограммная система визуализации данных, ориентированная на данные систем ИТ-мониторинга. Позволяет конечным пользователям строить сложные панели мониторинга.
8	Nginx	Прокси-сервер.
9	Polymatica	Аналитическая платформа сбора и визуализации данных, отчетов и графиков.
10	Kubernetes	Программное обеспечение для оркестровки контейнеризированных приложений, автоматизации их развёртывания, масштабирования и координации в условиях кластера.
11	Helm	Менеджер пакетов Helm - инструмент упаковки, настройки и развертывания приложений и служб в кластерах Kubernetes.

2. Информация по модулю Маркетинг перед установкой

Так как "T1 Аналитический CRM" является модулем системы T1CRM и представляет из себя приложение на основе микро-сервисной архитектуры. Для его установки необходима развернутая система T1CRM.

Установка производится в Kubernetes системы T1CRM и осуществляется с использованием пакетного менеджера Helm. Минимальные требования: предустановленное ядро системы T1CRM и зарезервированные ресурсы в размере 2 процессорных ядра и 4 гигабайта оперативной памяти.

Дополнительные ресурсы рассчитываются в зависимости от количества пользователей системы и предполагаемых процессов.

Для аутентификации используется Opensource решение на базе Keycloak, входящее в состав T1CRM. Для авторизации используется ролевая модель, реализованная в ядре T1CRM.

Для осуществления рассылок необходимо предоставить учетные данные для подключения к почтовому серверу, Sms-шлюзу, whatsapp-аккаунту. Для дополнительных интеграций можно использовать входящий в состав T1CRM инструмент как Kafka, либо же использовать кастомные решения на основе T1CRM.

Для установки модуля маркетинга каждому клиенту предоставляются индивидуальные скрипты установки модуля поверх описанной инфраструктуры в разделе 3.

3. Настройка инфраструктуры

3.1. Apache Kafka (брокер сообщений)

Предварительные требования:

Чтобы выполнить описанные ниже шаги, вам потребуется следующее:

- Сервер с операционной системой Oracle Linux 8.7, имеющий не менее 4 ГБ ОЗУ.
- Java 8+, установленный на вашем сервере:

```
dnf update -y
dnf install java-11-openjdk-devel -y
```

Порядок действий:

1. **Создание пользователя для Kafka**

1.1. Выполните вход с помощью пользователя без прав root с привилегиями sudo и создайте пользователя kafka с помощью команды useradd:

```
sudo useradd kafka
```

2. **Загрузка и извлечение двоичных файлов Kafka**

2.1. Предварительно необходимо скачать архив с двоичными файлами Kafka и скопировать целевой(-ые) на сервер(-ы) в директорию /tmp используя протокол SCP (WinSCP в OS Windows, scp в Linux).

2.2. Переходим в базовую директорию для установки Kafka, разархивируем скопированный на серверы архив и настроим директорию /opt/kafka

```
sudo cd /opt/
sudo wget https://archive.apache.org/dist/kafka/3.4.0/kafka_2.13-
3.4.0.tgz
sudo tar -xvzf kafka_2.13-3.4.0.tgz
```

```
sudo chown -R kafka:kafka /opt/kafka_2.13-3.4.0
sudo ln -s /opt/kafka_2.13-3.4.0 /opt/kafka
```

3. **Создание директории журнала**

```
sudo mkdir -p /var/log/kraft-combined-logs
sudo chown -R kafka:kafka /var/log/kraft-combined-logs
```

4. **Настройка сервера (кластера) Kafka**

4.1. Файл конфигураций

4.2. Опции конфигурации Kafka указаны в файле /opt/kafka/config/kraft/server.properties. Рекомендуется сохранить дефолтную версию конфигурации

```
sudo su - kafka
cd /opt/kafka/config/kraft/
cp server.properties server.properties.default
```

4.3. Apache Kafka может работать, как SingleNode режиме, так и в режиме кластера.

4.4. Для настройки в режиме SingleNode, необходимо заменить содержимое файла /opt/kafka/config/kraft/server.properties на содержимое из примера ниже.

4.5. В стр. 6 заменить 172.20.10.5 на IP-адрес текущей машины.

```
process.roles=broker,controller
node.id=1
controller.quorum.voters=1@localhost:9093
listeners=PLAINTEXT://:9092,CONTROLLER://:9093
inter.broker.listener.name=PLAINTEXT
advertised.listeners=PLAINTEXT://172.20.10.5:9092
controller.listener.names=CONTROLLER
listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,S
SL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
log.dirs=/var/log/kraft-combined-logs
num.partitions=1
num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
log.retention.hours=168
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
```

5. **Форматирование директории журнала**

```
cd /opt/kafka

./bin/kafka-storage.sh random-uuid

./bin/kafka-storage.sh format -t (ВЫВОД_ПРЕДЫДУЩИЙ_КОМАНДЫ) -c
/opt/kafka/config/kraft/server.properties
```

6. **Создание файлов элементов systemd**

```
sudo bash -c 'cat << EOF > /etc/systemd/system/kafka.service
[Unit]
```

```
Description=Apache Kafka server (broker)
Documentation=http://kafka.apache.org/documentation.html
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
Type=simple
User=kafka
Environment="KAFKA_START_JMX=-Dcom.sun.management.jmxremote.port=12345 -
Dcom.sun.management.jmxremote.rmi.port=12345 -
Djava.rmi.server.hostname=$(hostname -i)"
ExecStart=/bin/sh -c '/opt/kafka/bin/kafka-server-start.sh
/opt/kafka/config/kraft/server.properties > /opt/kafka/kafka.log 2>&1'
ExecStop=/opt/kafka/bin/kafka-server-stop.sh

[Install]
WantedBy=multi-user.target
EOF'
```

7. **Запуск сервера Kafka**

```
sudo systemctl daemon-reload
sudo systemctl enable kafka.service
sudo systemctl start kafka.service
```

8. **Проверка работоспособности сервера/кластера Kafka**

8.1. Проверяем состояние сервиса Kafka после запуска следующей командой:

```
sudo systemctl status kafka.service
```

8.2. Сервис должен быть в состоянии active (running) и журнале запуска не должно быть ошибок.

8.3. Настройка брандмауэра:

```
firewall-cmd --add-port=9092/tcp -permanent
firewall-cmd --reload
```

3.2. MinIO (Объектное хранилище, опционально для модуля Маркетинг)

Порядок действий:

1.1. **Создание пользователя для MinIO**

1.1.1. Выполните вход с помощью пользователя без прав root с привилегиями sudo и создайте пользователя minio-user с помощью команды useradd:

```
groupadd -r minio-user
useradd -M -r -g minio-user minio-user
```

1.2. **Загрузка и установка пакета MinIO**

1.2.1. Предварительно необходимо скачать rpm-пакет MinIO и скопировать на целевой(-ые) сервер(-ы) в директорию /tmp используя протокол SCP (WinSCP в OS Windows, scp в Linux).

```
wget https://dl.min.io/server/minio/release/linux-amd64/archive/minio-
20230112020616.0.0.x86_64.rpm -O minio.rpm -O /tmp/minio.rpm

sudo dnf install /tmp/minio.rpm
```

1.3. Настройка сервера MinIO

1.3.1. Пример конфигурации сервиса /etc/systemd/system/minio.service:

1.3.2. (Опции конфигурации MinIO по умолчанию указаны в файле /etc/default/minio в данном примере изменен путь к конфигурационному файлу minio.conf)

```
[Unit]
Description=MinIO
Documentation=https://docs.min.io
Wants=network-online.target
After=network-online.target
AssertFileIsExecutable=/usr/local/bin/minio

[Service]
WorkingDirectory=/usr/local

User=minio-user
Group=minio-user
ProtectProc=invisible

EnvironmentFile=-/opt/minio/minio.conf
ExecStartPre=/bin/bash -c "if [ -z \"${MINIO_VOLUMES}\" ]; then echo
\"Variable MINIO_VOLUMES not set in /etc/default/minio\"; exit 1; fi"
ExecStart=/usr/local/bin/minio server $MINIO_OPTS $MINIO_VOLUMES

Restart=always

LimitNOFILE=1048576

TasksMax=infinity

TimeoutStopSec=infinity
SendSIGKILL=no

[Install]
WantedBy=multi-user.target
```

1.3.3. Конфигурации Minio:

```
sudo bash -c 'cat << EOF > /opt/minio/minio.conf
MINIO_VOLUMES=/opt/minio/data/
MINIO_ROOT_USER=minioadmin #имя пользователя
MINIO_ROOT_PASSWORD=minio-secret-key-CHANGE-ME #пароль
MINIO_SERVER_URL="https://IP_reverse_proxy:9000"
EOF'
```

1.3.4. Назначения владельца директории:

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

1.3.5. Обновление, включение сервиса в автозагрузку и запуск Minio:

```
sudo systemctl daemon-reload
sudo systemctl enable minio.service
sudo systemctl restart minio.service
```

1.3.6. Настройка брандмауэра:

```
firewall-cmd --add-port=9000-9001/tcp --permanent
firewall-cmd --reload
```

1.4. Проверка работоспособности сервера/кластера MinIO

1.4.1. Проверяем состояние сервиса MinIO после запуска следующей командой:

```
sudo systemctl status minio.service
```

3.3. СУБД Redis

Порядок действий:

1.1. **Загрузка и установка пакета**

1.1.1. Redis доступен по умолчанию в RPM-based репозиториях и может быть установлен с помощью команды:

RHEL/CentOS/OracleLinux - version 8

```
sudo yum install epel-release
sudo dnf module enable redis:6 -y
```

RHEL/CentOS/OracleLinux - version 9

```
sudo dnf install redis -y
```

1.2. **Конфигурация Redis**

1.2.1. Пример файла конфигурации /etc/redis.conf:

```
# bind 127.0.0.1 172.20.11.8
protected-mode no
port 6379
supervised systemd
tcp-backlog 511
timeout 0
tcp-keepalive 300
daemonize no
pidfile /var/run/redis_6379.pid
loglevel notice
logfile /var/log/redis/redis.log
databases 16
always-show-logo no
set-proc-title yes
proc-title-template "{title} {listen-addr} {server-mode}"
stop-writes-on-bgsave-error yes
rdbcompression yes
rdbchecksum yes
dbfilename dump.rdb
rdb-del-sync-files no
dir /var/lib/redis
replica-serve-stale-data yes
replica-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-diskless-load disabled
repl-disable-tcp-nodelay no
replica-priority 100
acllog-max-len 128
lazyfree-lazy-eviction no
lazyfree-lazy-expire no
lazyfree-lazy-server-del no
replica-lazy-flush no
lazyfree-lazy-user-del no
lazyfree-lazy-user-flush no
oom-score-adj no
oom-score-adj-values 0 200 800
```

```
disable-thp yes
appendonly no
appendfilename "appendonly.aof"
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
aof-load-truncated yes
aof-use-rdb-preamble yes
lua-time-limit 5000
slowlog-log-slower-than 10000
slowlog-max-len 128
latency-monitor-threshold 0
notify-keyspace-events ""
hash-max-ziplist-entries 512
hash-max-ziplist-value 64
list-max-ziplist-size -2
list-compress-depth 0
set-max-intset-entries 512
zset-max-ziplist-entries 128
zset-max-ziplist-value 64
hll-sparse-max-bytes 3000
stream-node-max-bytes 4096
stream-node-max-entries 100
activerehashing yes
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit replica 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
hz 10
dynamic-hz yes
aof-rewrite-incremental-fsync yes
rdb-save-incremental-fsync yes
jemalloc-bg-thread yes
```

1.3. Запуск Redis и проверка работоспособности

```
sudo systemctl enable redis
sudo systemctl start redis.service
sudo systemctl status redis.service
```

1.4. Настройка брэндмаура

```
sudo firewall-cmd --add-port=6379/tcp --permanent
sudo firewall-cmd --reload
```

3.4. СУБД PostgreSQL

Порядок действий:

1.1. Установка PostgreSQL

RHEL/CentOS/OracleLinux - version 8

```
sudo dnf update
dnf module enable postgresql:12 -y
dnf install postgres
dnf install postgresql-contrib.x86_64
```

RHEL/CentOS/OracleLinux - version 9

```
sudo dnf update
sudo dnf install -y
https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-
redhat-repo-latest.noarch.rpm

sudo dnf -qy module disable postgresql
sudo dnf install -y postgresql12-server
sudo /usr/pgsql-12/bin/postgresql-12-setup initdb
```

1.2. Инициализация и запуск

```
sudo /usr/pgsql-12/bin/postgresql-12-setup initdb
sudo systemctl enable —now postgresql-12.service
```

1.3. Конфигурация PostgreSQL

```
sudo su - postgres
psql -c "ALTER SYSTEM SET port = 5432;"
psql -c "ALTER SYSTEM SET max_connections = 10000;"
psql -c "ALTER SYSTEM SET shared_buffers = '1024MB';"
psql -c "ALTER SYSTEM SET listen_addresses = '*';"
psql -c "ALTER SYSTEM SET wal_level = 'logical';"
psql -c "ALTER SYSTEM SET synchronous_commit = 'on';"
psql -c "ALTER SYSTEM SET max_wal_senders = 20;"
psql -c "ALTER SYSTEM SET max_replication_slots = 20;"
psql -c "ALTER SYSTEM SET wal_keep_segments = 10;"
```

1.4. Конфигурация pg_hba.conf

```
sudo bash -c 'cat << EOF > /var/lib/pgsql/12/data/pg_hba.conf
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             all                                     trust
host    all             all             127.0.0.1/32            md5
host    all             all             0.0.0.0/0               md5
host    all             all             ::1/128                 md5
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            ident
host    replication     all             ::1/128                 ident
EOF'
```

1.5. Запуск СУБД Postgres и проверка состояния.

```
sudo systemctl restart postgresql-12.service
```

3.5. Opensearch и Fluent-Bit (Система визуализации и сбора логов, опционально для модуля Маркетинг)

Порядок действий:

1.1. Подготовка

1.1.1. Opensearch использует `mmapfs` для хранения индексов. Ограничения операционной системы, будет слишком низкими. Увеличение максимального предела:

```
sudo -i
echo "vm.max_map_count=262144" >> /etc/sysctl.conf
sysctl -p
```

1.1.2. Настройка брандмауэра:

```
sudo firewall-cmd --add-port=9200/tcp --add-port=9300/tcp --permanent
sudo firewall-cmd --reload
```

1.2. Установка Opensearch

```
sudo dnf install -y
https://artifacts.opensearch.org/releases/bundle/opensearch/2.9.0/opensea
rch-2.9.0-linux-x64.rpm
```

1.2.1. Запуск Opensearch

```
sudo systemctl enable --now opensearch.service
```

1.2.2. Проверка состояния

```
curl -X GET https://localhost:9200 -u 'admin:admin' --insecure
```

1.3. Настройка конфигураций

1.3.1. Конфигурация настраивается в зависимостей от особенностей той или иной инфраструктуры. Основные опции конфигурации Opensearch указаны в файле `/etc/opensearch/opensearch.yml`.

1.3.2. Создание резервной копии конфигурации:

```
sudo cp /etc/opensearch/opensearch.yml
/etc/opensearch/opensearch.yml.default
```

1.3.3. Настройка доступа:

```
sudo sed -i 's/#network.host: 192.168.0.1/network.host: 0.0.0.0/g'
/etc/opensearch/opensearch.yml
sudo sed -i '67a\discovery.type: single-node'
/etc/opensearch/opensearch.yml
```

1.3.4. Генерация хэш пароля для пользователей admin и kibanaserver:

```
cd /usr/share/opensearch/plugins/opensearch-security/tools
OPENSEARCH_JAVA_HOME=/usr/share/opensearch/jdk ./hash.sh
```

1.3.5. Приводим файл `/etc/opensearch/opensearch-security/internal_users.yml` к виду, заменяя хэши паролей на полученные на предыдущем шаге:

```
---
# This is the internal user database
# The hash value is a bcrypt hash and can be generated with
plugin/tools/hash.sh

_meta:
  type: "internalusers"
  config_version: 2

# Define your internal users here

## Demo users

admin:
  hash: "<NEW_HASH>"
  reserved: true
  backend_roles:
  - "admin"
  description: "Admin user"

kibanaserver:
  hash: "<NEW_HASH>"
  reserved: true
  description: "OpenSearch Dashboards user"
```

1.3.6. Перезапускаем opensearch:

```
sudo systemctl restart opensearch.service
```

1.3.7. Применяем конфигурацию:

```
cd /usr/share/opensearch/plugins/opensearch-security/tools

OPENSEARCH_JAVA_HOME=/usr/share/opensearch/jdk ./securityadmin.sh -cd
/etc/opensearch/opensearch-security/ -cacert /etc/opensearch/root-ca.pem
```

```
-cert /etc/opensearch/kirk.pem -key /etc/opensearch/kirk-key.pem -icl -
nhnv
```

1.3.8. Проверяем новый пароль:

```
curl https://localhost:9200 -u admin:yournewpassword -k
```

1.4. Установка opensearch-dashboards (analog to Kibana)

1.4.1. Настройка брандмауэра:

```
sudo firewall-cmd --add-port=5601/tcp --permanent
sudo firewall-cmd --reload
```

1.4.2. Установка opensearch-dashboards:

```
sudo dnf install -y
https://artifacts.opensearch.org/releases/bundle/opensearch-
dashboards/2.9.0/opensearch-dashboards-2.9.0-linux-x64.rpm
```

1.4.3. Запуск opensearch-dashboards:

```
sudo systemctl enable --now opensearch-dashboards.service
```

1.4.4. Создаем резервную копию файла конфигурации:

```
sudo cp /etc/opensearch-dashboards/opensearch_dashboards.yml
/etc/opensearch-dashboards/opensearch_dashboards.yml.default
```

1.4.5. Копируем сертификаты:

```
sudo cp /etc/opensearch/kirk* /etc/opensearch/root-ca.pem
/etc/opensearch-dashboards

sudo chown -R opensearch-dashboards:opensearch-dashboards
/etc/opensearch-dashboards
```

1.4.6. Настраиваем подключение к opensearch:

```
sudo sed -i 's/# server.host: "localhost"/server.host: "0.0.0.0"/g'
/etc/opensearch-dashboards/opensearch_dashboards.yml
sudo sed -i 's/opensearch.ssl.verificationMode:
none/opensearch.ssl.verificationMode: full/g' /etc/opensearch-
dashboards/opensearch_dashboards.yml
sudo bash -c 'cat <<EOF >> /etc/opensearch-
dashboards/opensearch_dashboards.yml
server.ssl.enabled: false
server.ssl.certificate: "/etc/opensearch-dashboards/kirk.pem"
server.ssl.key: "/etc/opensearch-dashboards/kirk-key.pem"
opensearch.ssl.certificateAuthorities: "/etc/opensearch-dashboards//root-
ca.pem"
EOF'
```

1.4.7. Меняем пароль от kibanaserver:

```
# Заменить <NEW_KIBANA_PASS>
sudo sed -i 's/opensearch.password: kibanaserver/opensearch.password:
<NEW_KIBANA_PASS>/g' /etc/opensearch-dashboards/opensearch_dashboards.yml
```

1.4.8. Перезапускаем opensearch-dashboards:

```
sudo systemctl restart opensearch-dashboards.service
```

1.5. Установка Fluent-Bit

1.5.1. Опции конфигурации Fluent-Bit по умолчанию указаны в файле `/etc/fluent-bit/fluent-bit.conf`.

1.5.2. Добавляем репозиторий:

```
sudo bash -c 'cat /etc/yum.repos.d/fluent-bit.repo
[fluent-bit]
name = Fluent Bit
baseurl = https://packages.fluentbit.io/centos/\$releasever/
gpgcheck=1
gpgkey=https://packages.fluentbit.io/fluentbit.key
repo_gpgcheck=1
enabled=1
EOF'
```

1.5.3. Устанавливаем Fluent-Bit:

```
sudo dnf install -y fluent-bit
```

1.5.4. Запускаем Fluent-Bit:

```
sudo systemctl enable --now fluent-bit.service
```

1.5.5. Создаем резервную копию файла конфигурации:

```
sudo cp /etc/fluent-bit/fluent-bit.conf /etc/fluent-bit/fluent-bit.conf.default
```

1.5.6. Настраиваем Fluent-Bit:

```
sudo bash -c 'cat << EOF > /etc/fluent-bit/fluent-bit.conf
[SERVICE]
    Flush        3
    Daemon       Off
    Log_Level    info
    HTTP_Server  On
    HTTP_Listen  0.0.0.0
    HTTP_Port    2020
    storage.metrics on
[INPUT]
    Name tcp
    Listen 0.0.0.0
    Port 30764
    Chunk_Size 32
    Buffer_Size 128
    Format json
[OUTPUT]
    Name es
    Match *
    Host 127.0.0.1 # Opensearch host
    Port 9200
    HTTP_User <OPENSEARCH_USER>
    HTTP_Passwd <OPENSEARCH_USER_PASSWORD>
    Suppress_Type_Name On
    tls On
    tls.verify Off
    Logstash_Format On
    Logstash_Prefix node
    Retry_Limit False
    Replace_Dots On
    Trace_Error on
EOF'
sudo systemctl restart fluent-bit.service
```

1.5.7. Перезапускаем Fluent-Bit:

```
sudo systemctl restart fluent-bit.service
```

3.6. Grafana (Система визуализации данных мониторинга, опционально для модуля Маркетинг)

Порядок действий:

1.1. **Загрузка и установка пакета**

```
wget https://dl.grafana.com/oss/release/grafana-9.3.2-1.x86_64.rpm
sudo dnf install grafana-9.3.2-1.x86_64.rpm
```

1.2. **Настройка и запуск сервера**

1.2.1. Опции конфигурации указаны в файле `/etc/grafana/grafana.ini`. Обычно конфигурации по умолчанию хватает для базовой работы Grafana сервера, тем более основные настройки выполняется через веб-интерфейс сервера, который после запуска сервера будет доступен по веб-адресу: http://ip_server:3000

1.2.2. Активируем и запустим сервис данными командами:

```
sudo systemctl enable grafana-server

sudo systemctl restart grafana-server
```

1.3. **Проверка работоспособности сервера**

1.3.1. Проверяем состояние сервиса после запуска следующей командой:

```
sudo systemctl status grafana-server

ss -ntlp | grep 3000
```

3.7. *NGINX Reverse Proxy (реверс-прокси)*

Порядок действий:

1.1. **Загрузка и установка пакета**

1.1.1. Nginx доступен по умолчанию в RPM-based репозиториях и может быть установлен на все узлы с помощью команды: sudo dnf install nginx -y

1.2. **Настройка сервера (кластера)**

1.2.1. Файлы конфигурации находятся в директории `/etc/nginx/conf.d/` и имеют расширение `.conf`

1.2.2. Список серверов, которые необходимо завернуть на Reverse Proxy и файлы конфигурации nginx для них.

1.2.3. Важно заменить {{ full_domain_name }} на полное доменное имя стенда и также обозвать название сертификатов.

1.2.4. Фронт продукта:

```
server 172.20.10.2:80 fail_timeout=2s;

  server 172.20.10.3:80 fail_timeout=2s;

  server 172.20.10.4:80 fail_timeout=2s;

}

server {
```

```
  listen 80;

  include /etc/nginx/default.d/*.conf;
  location / {

    proxy_pass http://t1crm-frontend/;

    proxy_set_header Host $http_host;

    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_set_header X-Real-IP $remote_addr;

  }

}

server {

  listen 443 ssl;

  include /etc/nginx/default.d/*.conf;

  server_name {{ full_domain_name }};

  ssl_certificate         /etc/pki/tls/certs/{{ full_domain_name }}.crt;

  ssl_certificate_key     /etc/pki/tls/private/{{ full_domain_name }}.key;

  client_max_body_size 30M;

  access_log  /var/log/nginx/t1crm-upstream.log  upstreamlog;


  location / {

    if ($request_method ~* "(GET|POST)") {

      add_header "Access-Control-Allow-Origin" * always;

    }

    if ($request_method = OPTIONS ) {

      add_header "Access-Control-Allow-Origin" * always;

      add_header "Access-Control-Allow-Methods" "GET, POST, OPTIONS,
HEAD";

      add_header "Access-Control-Allow-Headers" "Authorization, Origin,
X-Requested-With, Content-Type, Accept, Access-Control-Allow-Origin";

      return 200;

    }

    proxy_pass http://t1crm-frontend;

    proxy_set_header Host $http_host;
```

```
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
  }
}
```

1.2.5. Grafana:

```
 upstream t1crm-grafana {
  server 172.20.10.11:3000;
}
server {
  listen 3000 ssl;
  include /etc/nginx/default.d/*.conf;
  server_name {{ full_domain_name }};
  ssl_certificate         /etc/pki/tls/certs/{{ full_domain_name }}.crt;
  ssl_certificate_key     /etc/pki/tls/private/{{ full_domain_name }}.key;
  client_max_body_size 30M;
  access_log  /var/log/nginx/t1crm-grafana.log  upstreamlog;

  location / {
    if ($request_method ~* "(GET|POST)") {
      add_header "Access-Control-Allow-Origin" * always;
    }
    if ($request_method = OPTIONS ) {
      add_header "Access-Control-Allow-Origin" * always;
      add_header "Access-Control-Allow-Methods" "GET, POST, OPTIONS,
HEAD";
      add_header "Access-Control-Allow-Headers" "Authorization, Origin,
X-Requested-With, Content-Type, Accept, Access-Control-Allow-Origin";
      return 200;
    }

    proxy_pass http://t1crm-grafana/;
```

```
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
  }
}
```

1.2.6. Kibana:

```
upstream t1crm-kibana {
  server 172.20.10.11:5601;
}
server {
  listen 5601 ssl;
  include /etc/nginx/default.d/*.conf;
  server_name {{ full_domain_name }};
  ssl_certificate         /etc/pki/tls/certs/{{ full_domain_name }}.crt;
  ssl_certificate_key     /etc/pki/tls/private/{{ full_domain_name }}.key;
  client_max_body_size 30M;
  access_log  /var/log/nginx/t1crm-kibana.log  upstreamlog;

  location / {
    if ($request_method ~* "(GET|POST)") {
      add_header "Access-Control-Allow-Origin" * always;
    }
    if ($request_method = OPTIONS ) {
      add_header "Access-Control-Allow-Origin" * always;
      add_header "Access-Control-Allow-Methods" "GET, POST, OPTIONS,
HEAD";
      add_header "Access-Control-Allow-Headers" "Authorization, Origin,
X-Requested-With, Content-Type, Accept, Access-Control-Allow-Origin";
      return 200;
    }
```

```
    proxy_pass https://t1crm-kibana/;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
  }
}
```

1.2.7. MiniO:

```
upstream t1crm-minio {
  server 172.20.10.9:9001;
}
server {
  listen 9001 ssl;
  include /etc/nginx/default.d/*.conf;
  server_name corpcrm.t1crm.ru;
  ssl_certificate         /etc/pki/tls/certs/corpcrm.t1crm.ru.crt;
  ssl_certificate_key     /etc/pki/tls/private/corpcrm.t1crm.ru.key;
  client_max_body_size 30M;
  access_log  /var/log/nginx/t1crm-minio.log  upstreamlog;
  location / {
    if ($request_method ~* "(GET|POST)") {
      add_header "Access-Control-Allow-Origin" * always;
    }
    if ($request_method = OPTIONS ) {
      add_header "Access-Control-Allow-Origin" * always;
      add_header "Access-Control-Allow-Methods" "GET, POST, OPTIONS,
HEAD";
      add_header "Access-Control-Allow-Headers" "Authorization, Origin,
X-Requested-With, Content-Type, Accept, Access-Control-Allow-Origin";
      return 200;
    }
    proxy_pass http://t1crm-minio/;
```

```
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
  }
}
```

1.2.8. Polymatica:

```
upstream polymatica {
  server 172.20.10.10:443;
}




server {
  listen 9443 ssl;
  include /etc/nginx/default.d/*.conf;
#  access_log  /var/log/nginx/polymatica.log;

  server_name t1crm-test.tsc.ts;
  ssl_certificate        /etc/pki/tls/certs/corpcrm.t1crm.ru.crt;
  ssl_certificate_key    /etc/pki/tls/private/corpcrm.t1crm.ru.key;

  client_max_body_size 100M;

  location / {

    if ($request_method ~* "(GET|POST)") {
      add_header "Access-Control-Allow-Origin" * always;
    }

    if ($request_method = OPTIONS ) {
```

```
      add_header "Access-Control-Allow-Origin" * always;

      add_header "Access-Control-Allow-Methods" "GET, POST, OPTIONS,
HEAD";

      add_header "Access-Control-Allow-Headers" "Authorization, Origin,
X-Requested-With, Content-Type, Accept, Access-Control-Allow-Origin";

      return 200;

    }



    proxy_pass https://polymatica/;

    proxy_set_header Host $http_host;

    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    proxy_set_header X-Real-IP $remote_addr;

  }

}
```

1.2.9. TLS сертефикат и ключ должны находится согласно путям указанных в конфигах.

1.2.10. Настройка брандмауэра:

```
sudo firewall-cmd --add-port=5601/tcp --add-port=3000/tcp --add-
port=9001/tcp --add-port=80/tcp --add-port=443/tcp --permanent
sudo firewall-cmd --reload
```

1.2.11. После применения всех вышеуказанный настроек активируем и запустим сервис **nginx** данными командами:

```
sudo systemctl enable  nginx
sudo systemctl restart nginx
```

1.3. **Проверка работоспособности сервера/кластера**

1.3.1. Проверяем состояние сервиса после запуска следующей командой:

```
sudo systemctl status nginx

ss -ntlp | egrep '\:80|\:443|\:3000|\:5601|\:8443|\:9001'
```

3.8. Kubernetes (система оркестрации контейнеров и микро-сервисов)

Предварительные требования:

Для установки кластера k8s понадобится:

- Один сервер с установленной операционной системой семейства Linux для **Kubespray**
- Пять серверов для кластера **Kubernetes**

HOSTNAME	TASK	IP ADDRESS
master	k8s-master	172.20.10.2
node1	k8s-node1	172.20.10.3

HOSTNAME	TASK	IP ADDRESS
node2	k8s-node2	172.20.10.4
node3	k8s-node3	172.20.10.5
node4	k8s-node4	172.20.10.6

Примечание: все операции развертывания кластера будут производится с помощью Ansible playbook Kubespray.

Порядок действий:

1.1. **Установка и настройка плейбука Kubespray**

1.1.1. Контроллер Kubespray — это сервер, на которой вы установите среду Kubespray и с которой у вас будет доступ ко всем серверам, которые станут частью вашего кластера. Желательно, чтобы этот контроллер также не был частью вашего кластера. На этом контроллере, следуя официальной документации, мы клонируем репозиторий Kubespray и установим ansible сначала в виртуальной среде Python - отдельном рабочем пространстве, похожем на контейнер, который инкапсулирует все необходимые версии программного обеспечения в одну.

```
git clone --depth=1 https://github.com/kubernetes-sigs/kubespray.git



KUBESPRAYDIR=$(pwd)/kubespray

VENVDIR="$KUBESPRAYDIR/.venv"

virtualenv  --python=$(which python3) $VENVDIR



source $VENVDIR/bin/activate

cd $KUBESPRAYDIR

pip install -r requirements.txt
```

1.2. **Установка и настройка плейбука Kubespray**

1.2.1. Самым первым шагом при использовании Kubespray является определение инвентаризации — основной концепции Ansible, которая представляет собой список серверов с именами хостов или IP-адресами и ролью, которую они играют в вашей установке.

1.2.2. Сделать копию конфигурации из inventory/sample в inventory/t1crm и настроить в соответствии с конфигурацией нашего кластера. (пример настроенных файлов приложен к инструкции в отдельном архиве inventory.zip)

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

declare -a IPS=(172.20.10.2 172.20.10.3 172.20.10.4 172.20.10.5
172.20.10.6)

sudo pip3 install -U pip setuptools wheel
pip3 install ruamel_yaml
```

```
pip3 install -r requirements-2.11.txt

CONFIG_FILE=inventory/mycluster/hosts.yaml python3
contrib/inventory_builder/inventory.py ${IPS[@]}
```

1.3. **Установка Развертывание кластера Kubernetes**

1.3.1. Установка кластера начинается с запуска Ansible-плейбука cluster.yaml. Процесс развертывания кластера Kubernetes в данной конфигурации займет примерно 20-30 минут.

```
sudo dnf install epel-release

sudo dnf install ansible

ansible-playbook -i inventory/t1crm/inventory.ini cluster.yml -b -v --
private-key=~/.ssh/id_rsa -K
```

1.3.2. Приме результата успешного выполнения плейбука Kubespray:

```
PLAY RECAP
*********************************************************************
**********************************************

localhost                  : ok=3    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0

node-1             : ok=709  changed=33   unreachable=0    failed=0
skipped=1243 rescued=0    ignored=4

node-2             : ok=507  changed=6    unreachable=0    failed=0
skipped=770  rescued=0    ignored=2

node-3             : ok=507  changed=6    unreachable=0    failed=0
skipped=770  rescued=0    ignored=2
```

1.4. **Подключение к кластеру и проверка работоспособности**

1.4.1. Для работы с кластером необходима утилита kubectl, в идеале с той же версией, и файл kubeconfig. Установка Kubespray помещает их на узлы контроллера. Таким образом, вы можете выполнить SSH-вход в свой кластер и работать оттуда или использовать контроллер kubespray, на который вы вручную можете установить утилиту kubectl.

1.4.2. Для проверки работоспособности кластера Kubernetes, подключитесь к master-серверу и выполните следующие команды:

```
kubectl cluster-info

kubectl get nodes

kubectl get pods -A
```

Вы должны увидеть статус кластера, все узлы и все запущенные pod-ы.

3.9. Установка менеджера пакетов Helm (опционально).

Порядок действий:

1. Загрузите нужную версию.

```
wget https://get.helm.sh/helm-v3.12.0-linux-amd64.tar.gz
```

2. Распакуйте архив.

```
tar -zxvf helm-v3.0.0-linux-amd64.tar.gz
```

3. Найдите helm двоичный файл в распакованном каталоге и переместите его в нужное место

```
mv linux-amd64/helm /usr/ bin/helm
```

Проверка:

```
helm version
```

В результате мы должны получить версию установленного Helm.

```
version.BuildInfo{Version:"v3.12.0",
GitCommit:"c9f554d75773799f72ceef38c51210f1842a1dea",
GitTreeState:"clean", GoVersion:"go1.20.3"}
```