GS LABS
GS GROUP

Публичный - L0

DRE Remote Control Service

Руководство по установке

Индекс	Cloud-IG
Конфиденциальность	Публичный - L0
Ревизия	1.0
Статус	Согласован

Содержание

1. Аннотация

Документ предназначен для технических специалистов, занимающихся установкой, настройкой и поддержкой сервиса DRE Remote Control Service (далее в документе используется условное наименование - облачный сервис Cloud). Документ рассчитан на инженеров, обладающих специальными навыками и знаниями в области программного обеспечения.

> *Данный документ опубликован исключительно с целью изучения системных требований для установки продукта, а также ознакомления с последовательностью и деталями процесса установки. Реальная установка продукта производится с использованием внутренних репозиториев ООО "Цифра", доступ к которым предоставляется заказчику по запросу.*

2. Минимальные системные требования

Для установки сервиса необходимо наличие не менее 3 серверов с разными именами (hostname): master, worker1, worker2. Общее количество серверов должно быть нечетным.
Серверы должны удовлетворять следующим требованиям:

1. Операционная система ubuntu-20.04-server-amd64 (с установленным пакетом sudo).
2. Многоядерный центральный процессор с тактовой частотой каждого ядра 2 ГГц (минимум 20 ядер).
3. Объем оперативной памяти 64 ГБ.
4. Не менее 2 жестких дисков емкостью 500 ГБ или больше. Рекомендуется наличие на каждой ноде помимо основного дискового пространства с ОС 1-го диска SSD или NVMe и 9-ти дисков HDD (SATA, SAS), не собранных в RAID и не форматированных.
5. Два интерфейса Ethernet 100 и 1000 Base-T с поддерживаемой пропускной способностью 100 и 1000 Мбит/сек соответственно. Один предназначен для сети поддержки, второй используется для вывода генерируемого транспортного потока.

Установка должна производиться с дополнительного Ubuntu-сервера, не имеющего отношения к будущему кластеру. Требования к объему ресурсов дополнительного сервера отсутствуют.

Корректная работа сервиса гарантируется на версиях ОС Ubuntu 20.04

3. Развертывание приложения в кластере kubernetes

Кластер развёртывается по официальной инструкции kubernetes.

Для установки сервиса в имеющийся настроенный кластер Kubernetes используется процесс CI/CD, настраиваемый с помощью GitLab.

Все действия возможно производить на локальной машине или на любом Ubuntu-сервере с доступом через консоль от имени любого пользователя.

3.1. Требования к окружению

Для развёртывания сервиса нужно соблюсти следующие требования:

1. Развернуть высоконагруженный кластер PostgreSQL.
2. Развернуть кластер NATS с использованием nats-operator
3. Развертывать необходимо на кластер Kubernetes.

3.1.1. Требования к NATS

Для работы продукта необходим кластер NATS, развернутый при помощи nats-operator.

Пример helmfile для установки оператора в кластер:

nats-operator.yaml

```
helmDefaults:
  kubeContext: integration

repositories:
  - name: geek-cookbook
    url: https://geek-cookbook.github.io/charts

releases:
  - name: nats-operator
    namespace: data
    chart: geek-cookbook/nats-operator
    version: 0.1.3
    labels:
      app: nats-operator
      level: infra
    values:
    - cluster:
        auth:
          enabled: false
        metrics:
          enabled: true
```

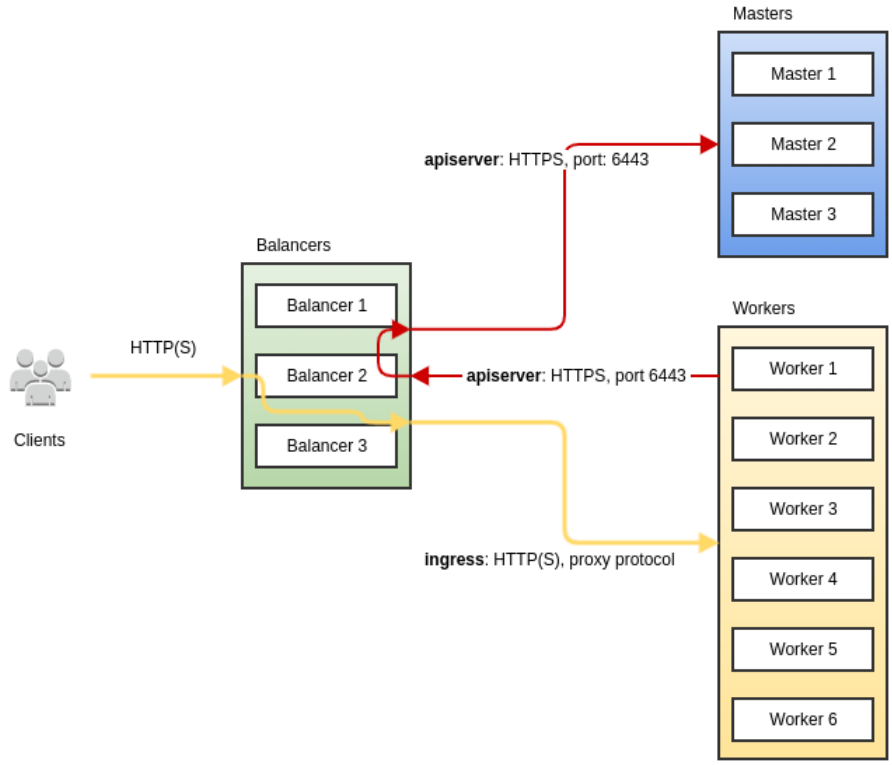
3.1.2. Распределение узлов

Необходимо разделить узлы логически на две роли:

Роль	Компоненты	Пример распределения ресурсов	Требования к количеству узлов	Требования к физическому местоположению	Резервное копирование
Master	• apiserver • control plane • etcd	• 2 CPU cores • 2GB RAM	1. Не менее трёх 2. Нечётное количество	Распределить по физическим машинам. Рекомендуется 1 физическая машина - 1 master нода.	Каждый час - полный snapshot etcd. (Документация https://etcd.io/docs/v3.3.12/op-guide/recovery/)
Worker	• kubelet	• 4 CPU cores • 16GB RAM	1. Не менее двух	Распределить по физическим машинам. Рекомендуется 1 физическая машина - 2 worker-ноды.	Не требуется

3.1.3. Общая схема

Перед Kubernetes кластером необходимо разместить балансировщики на базе haproxy для балансировки как внутреннего трафика kubernetes (worker↔master), так и внешнего (ingress). Балансировщики должны иметь общий Virtual IP адрес, который настраивается через keepalived. Этот адрес должен использоваться как адрес мастера при создании кластера. Таким образом, в случае отказа мастера, трафик с воркеров будет успешно достигать одного из мастеров. Схематически процесс изображен ниже:



3.1.4. Балансировка

Конфигурация haproxy (на тестовом кластере, реальные параметры подбираются исходя из доступных вычислительных мощностей):

- 3 Master узла
- 6 Worker узлов
- 3 балансера

haproxy.cfg

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGCM:
RSA+AES:!aNULL:!MD5:!DSS
        ssl-default-bind-options no-sslv3

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

frontend k8s-api
  bind :6443
  mode tcp
  option tcplog
  timeout client  86400000
  default_backend k8s-api

frontend k8s-ingress-http
  bind :80
  mode tcp
  option tcplog
  default_backend k8s-ingress-http

frontend k8s-ingress-https
  bind :443
  mode tcp
```

```
  option tcplog
  default_backend k8s-ingress-https

backend k8s-api
  mode tcp
  option tcplog
  option tcp-check
  balance roundrobin
  default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250 maxqueue 256 weight 100
  timeout server  86400000
  server kube-master-1 10.128.55.63:6443 check
  server kube-master-2 10.128.55.64:6443 check
  server kube-master-3 10.128.55.66:6443 check

backend k8s-ingress-http
  mode tcp
  option tcplog
  option tcp-check
  balance roundrobin
  default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250 maxqueue 256 weight 100
  # 80 - Port of http nginx-ingress service
  server kube-worker-1 10.128.55.61:80 check send-proxy
  server kube-worker-2 10.128.55.56:80 check send-proxy
  server kube-worker-3 10.128.55.51:80 check send-proxy
  server kube-worker-4 10.128.55.68:80 check send-proxy
  server kube-worker-5 10.128.55.62:80 check send-proxy
  server kube-worker-6 10.128.55.54:80 check send-proxy

backend k8s-ingress-https
  mode tcp
  option tcplog
  option tcp-check
  balance roundrobin
  default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250 maxqueue 256 weight 100
  # 443 - Port of https nginx-ingress service
  server kube-worker-1 10.128.55.61:443 check send-proxy
  server kube-worker-2 10.128.55.56:443 check send-proxy
  server kube-worker-3 10.128.55.51:443 check send-proxy
  server kube-worker-4 10.128.55.68:443 check send-proxy
  server kube-worker-5 10.128.55.62:443 check send-proxy
  server kube-worker-6 10.128.55.54:443 check send-proxy
```

keepalived.conf

```
vrrp_script haproxy-check {
    script "/usr/bin/killall -0 haproxy"
    interval 2
    weight 20
}

# VIP address: 10.128.55.67
# balancer-1 (this): 10.128.55.65
# balancer-2: 10.128.55.60
# balancer-3: 10.128.55.58

vrrp_instance haproxy-vip {
    state BACKUP
    priority 101
    interface ens18 # ethernet interface name
    virtual_router_id 47
    advert_int 3

    unicast_src_ip 10.128.55.65 # IP of this balancer
    unicast_peer {
        # IPs of other balancers
        10.128.55.60
        10.128.55.58
    }

    virtual_ipaddress {
        10.128.55.67/24
    }

    track_script {
        haproxy-check weight 20
    }
}
```

3.1.4.1. **Входящий трафик**

Во внешнюю сеть должны быть открыты только эти порты:

- 80;
- 443;

В тестовых средах дополнительно может быть открыт порт 6443 для работы с kubernetes кластером.

3.1.4.2. **Apiserver трафик**

Весь трафик Kubernetes apiserver передаётся по порту 6443 и направлен на узлы с ролью Master.

3.1.4.3. **Ingress трафик**

Весь Ingress трафик передаётся по портам 80 и 443. SSL termination на балансировщике не производится, вместо этого Ingress трафик передаётся в виде proxy protocol до узлов с ролью Worker.

3.1.5. Конфигурация Ingress

Ingress Controller, который используется нами - это nginx-ingress, и для правильной его работы с proxy protocol, в его конфигурации потребуется установка дополнительных опций. Опции Kubespray:

kubespray/addons/all.yaml

```
...
ingress_nginx_enabled: true
ingress_nginx_nodeselector:
  node-role.kubernetes.io/ingress: "" # Only run ingress DaemonSet on labeled nodes
ingress_nginx_host_network: true
ingress_nginx_tolerations: []
ingress_nginx_insecure_port: 80
ingress_nginx_secure_port: 443
ingress_nginx_configmap:
  use-forwarded-headers: "true"
  use-proxy-protocol: "true"
...
```

После этого необходимо выполнить эту команду для каждого worker узла:

```
kubectl label node <node> node-role.kubernetes.io/ingress=
```

3.1.6. Конфигурация TLS

Nginx не поддерживает автоматическое определение протокола между HTTP1.1 и HTTP/2 если не используется TLS. Именно поэтому установка сервиса невозможна в среду с отсутствующим TLS шифрованием.

В наличии необходимо иметь сертификат, подходящий к доменному имени оператора. Для загрузки сертификата в кластер используйте команду:

kubectl tls

```
$ kubectl --namespace events smart-home-cloud secret tls [domain.name] --cert <path_to_cert_file> --key
<path_to_key_file>
```

3.2. Развёртывание внутри Kubernetes

Для установки сервиса в имеющийся настроенный кластер Kubernetes используется процесс CI/CD, настраиваемый с помощью GitLab. Ссылка на документ с описанием всего процесса предоставляется по запросу заказчика.

Ссылка на репозиторий для развёртывания продукта предоставляется по запросу заказчика.

Состав репозитория:

- helmfile.yaml - главный конфигурационный файл утилиты helmfile.
- default.yaml - файл с values окружения утилиты helmfile.
- values - папка с values для каждого чарта; они являются шаблонными и забирают значения из values окружения (файла default.yaml).
- versions.yaml - файл с версиями компонентов.
- limitation - папка с values ресурсов подов.

Конфигурирование сервисов осуществляется путем изменения значений параметров в helm-файле.

Набор параметров и выставляемых значений может меняться в соответствии с требованиями и задачами заказчика. Описание специфических параметров для Cloud приведено в Руководстве администратора.

3.2.1. Конфигурация сервисов

Конфигурацию сервисов по умолчанию можно посмотреть в файле *default.yaml* проекта (ссылка предоставляется по требованию заказчика).
Перечень доступных параметров переменных окружения сервисов продукта и их начальных значений предоставляется заказчику по требованию.