GS LABS
GS GROUP

Публичный - L0

DRE Event Server

Руководство по установке

Индекс	2060-DREES-IG
Секретность	Публичный - L0
Ревизия	1.0
Статус	Согласован
Подразделение	Департамент по разработке сервисов
Компания	GS Labs

Содержание

1. Аннотация

Документ предназначен для технических специалистов, занимающихся установкой, настройкой и поддержкой системы мониторинга DRE Event Server (далее - DREES). Документ рассчитан на инженеров, обладающих специальными навыками и знаниями в области программного обеспечения.

2. Минимальные системные требования

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

1. Операционная система Ubuntu 18.04.
2. Многоядерный центральный процессор с тактовой частотой каждого ядра 2 ГГц (не менее 20-ти ядер).
3. Объем оперативной памяти 64 ГБ.
4. Не менее 2-ух жестких дисков емкостью не менее чем по 500 ГБ. 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.
6. Свободное место для папки временных файлов /tmp - 10 ГБ.

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

3. Развертывание сервисов DREES в кластере kubernetes

Кластер развёртывается по официальной инструкции (https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/).

Для установки DREES в имеющийся настроенный кластер Kubernetes используется процесс CI/CD, настраиваемый с помощью GitLab. Весь процесс описан в документе - https://gitlab.gs-labs.tv/automation/cd-templates (доступ ограничен).

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

Для развертывания EventServer используется CD конфигурация на базе Helm Chart и Helmfile. Репозиторий с конфигурацией: https://gitlab.gs-labs.tv/releases/providence (доступ ограничен).

CD конфигурация состоит из двух файлов: `helmfile.yaml` содержащий в себе список сервисов DREES, и `values.yaml` – файл с настройками конкретного релиза. В качестве примера в репозитории есть файл `defaults.yaml`.

Предполагается, что DREES будет доступен по доменному имени metrics.tricolor.tv, а так же будет поддерживать Swagger UI для тестирования.

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
```

Для балансировки сервисов между нодами, кластер использует механизм `nodeAffinity`, для которого нодам необходимо раздать разные лэйблы (label). Для конфигурации балансировки сервисов необходимо изменить поле `nodeAffinity` в используемом файле `values.yaml`

nodeAffinity

```
# Each service deploy to defined node, using this labels
nodeSelector:
  consumer:
    role.providence/consumer: ""
  events:
    role.providence/events: ""
  geoIp:
    role.providence/geo-ip: ""
  gateway:
    role.providence/gateway: ""
  envoy:
    role.providence/envoy: ""
```

4. Требования к окружению

Для развёртывания DREES предварительно необходимо соблюсти следующие требования:

1. Развернуть высоконагруженный кластер PostgreSQL с расширением TimescaleDB.
2. Разворачивать необходимо HA кластер Kubernetes.

4.1.1. Требования к PostgreSQL

PostgreSQL в данном продукте - это ключевой компонент. Чем быстрее база, тем больше клиентов DREES может обслуживать. Рекомендуется использовать расширение TimescaleDB для работы с Time-series - партиционирование таблиц, функции агрегации и т.д. Последовательность действий:

1. Развернуть PostgreSQL. Лучше - отказоустойчивый, но главное - быстрый.
2. Развернуть пулер соединений перед ним (опционально для тестовых зон).
3. Установить расширение TimescaleDB.
4. Создать БД для DREES (в примерах будем использовать имя "event-server").
5. Создать пользователя с доступом только к этой БД (в примерах будем использовать имя "event-server-user" с паролем "event-server-password").
6. Записать артефакты конфигурации для последующей настройки DREES:
    a. IP адрес и порт мастера (пулера).
    b. Имя БД.
    c. Логин и пароль пользователя.
    d. Имя схемы, используемой в БД (по умолчанию - "public")

4.1.2. Установка Nats Streaming и Etcd

Установка Nats Streaming и Etcd выполняется при включении ссылки на файл services.yaml из релизного проекта в helmfile проекта установки следующим образом:

```
helmfiles:
  - path: ingress.helmfile.yaml
  - path: ceph.helmfile.yaml
  - path: etcdvars.helmfile.yaml
    values:
    - production.yaml
{color:red}  - path: providence/services.yaml
    values:
    - providence/default.yaml
    - production.yaml{color}
  - path: providence/helmfile.yaml
    values:
      - providence/versions.gen.yaml
      - providence/default.yaml
      - production.yaml
```

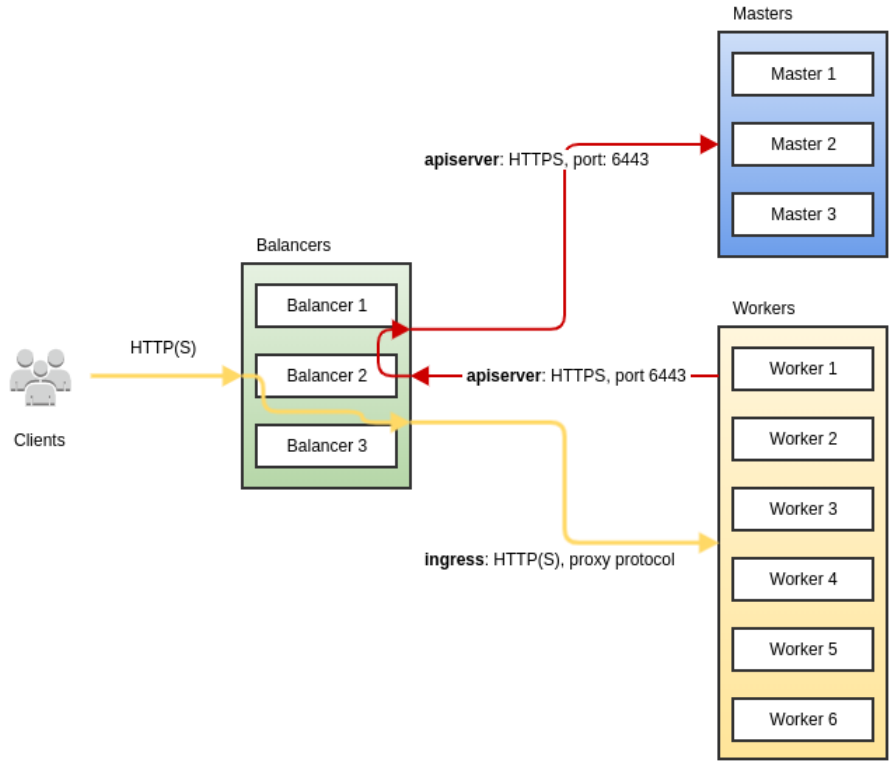
4.1.3. Распределение узлов

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

4.1.4. Общая схема

Перед Kubernetes кластером необходимо разместить балансировщики на базе haproxy для балансировки как внутреннего трафика kubernetes (workermaster), 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:



4.1.5. 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
    }
}
```

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.

Apiserver трафик

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

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.

4.1.6. 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.

Используемая версия Ingress Controller: 0.30.0

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_extra_args:
  - --maxmind-license-key=<MAXMIND_LICENSE_KEY>
ingress_nginx_configmap:
  use-forwarded-headers: "true"
  use-proxy-protocol: "true"
  http-snippet: |
    geoip2 /etc/nginx/geoip/GeoLite2-City.mmdb {
      $geoip2_city_id source=$http_x_forwarded_for city geoname_id;
      $geoip2_city_name source=$http_x_forwarded_for city names ru;
      $geoip2_subdivision_id source=$http_x_forwarded_for subdivisions 0 geoname_id;
      $geoip2_subdivision_name source=$http_x_forwarded_for subdivisions 0 names ru;
      $geoip2_latitude source=$http_x_forwarded_for location latitude;
      $geoip2_longitude source=$http_x_forwarded_for location longitude;
      $geoip2_city_country_code source=$http_x_forwarded_for country iso_code;
      $geoip2_time_zone source=$http_x_forwarded_for location time_zone;
    }
  main-snippet: |
    load_module /etc/nginx/modules/ngx_http_geoip2_module.so;
...
```

Обратите внимание, что в опциях используется ключ для скачивания базы данные гео-информации об IP-адресах MaxMind.

После этого необходимо выполнить эту команду для каждого worker узла:

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

4.1.7. Конфигурация TLS

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

В наличии необходимо иметь сертификат, подходящий к доменному имени metrics.tricolor.tv. Дла загрузки сертификата в кластер используйте команду:

kubectl tls

```
$ kubectl --namespace events providence secret tls tricolor.tv --cert <path_to_cert_file> --key
<path_to_key_file>
```

4.1.8. Конфигурация Envoy

Envoy используется для отказоустойчивой балансировки нагрузки между подами events server. Развертывается автоматически, при помощи helm чарта event server, в дополнительной конфигурации не нуждается.