41 | 软件部署实战（中）：IAM 系统生产环境部署实战

2021-08-28 孔令飞

《Go 语言项目开发实战》 课程介绍 >

讲述：孔令飞
时长 11:55 大小 10.93M

你好，我是孔令飞。

上一讲，我介绍了 IAM 部署用到的两个核心组件，Nginx 和 Keepalived。那么这一讲，我们就来看下，如何使用 Nginx 和 Keepalived 来部署一个高可用的 IAM 应用。下一讲，我再介绍下 IAM 应用安全和弹性伸缩能力的构建方式。

这一讲，我们会通过下面四个步骤来部署 IAM 应用：

1. 在服务器上部署 IAM 应用中的服务。
2. 配置 Nginx，实现反向代理功能。通过反向代理，我们可以通过 Nginx 来访问部署在内网的 IAM 服务。

3. 配置 Nginx，实现负载均衡功能。通过负载均衡，我们可以实现服务的水平扩缩容，使 IAM 应用具备高可用能力。
4. 配置 Keepalived，实现 Nginx 的高可用。通过 Nginx + Keepalived 的组合，可以实现整个应用架构的高可用。

部署 IAM 应用

部署一个高可用的 IAM 应用，需要至少两个节点。所以，我们按照先后顺序，分别在`10.0.4.20`和`10.0.4.21`服务器上部署 IAM 应用。

在`10.0.4.20`服务器上部署 IAM 应用

首先，我来介绍下如何在`10.0.4.20`服务器上部署 IAM 应用。

我们要在这个服务器上部署如下组件：

iam-apiserver

iam-authz-server

iam-pump

MariaDB

Redis

MongoDB

这些组件的部署方式，03 讲 有介绍，这里就不再说明。

此外，我们还需要设置 MariaDB，给来自于`10.0.4.21`服务器的数据库连接授权，授权命令如下：

复制代码

```
$ mysql -hlocalhost -P3306 -uroot -proot # 先以root用户登陆数据库
MariaDB [(none)]> grant all on iam.* TO iam@10.0.4.21 identified by 'iam1234';
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.000 sec)
```

在10.0.4.21服务器上部署 IAM 应用

然后，在10.0.4.21服务器上安装好 iam-apiserver、iam-authz-server 和 iam-pump。这些组件通过10.0.4.20 IP 地址，连接10.0.4.20服务器上的 MariaDB、Redis 和 MongoDB。

配置 Nginx 作为反向代理

假定要访问的 API Server 和 IAM Authorization Server 的域名分别为iam.api.marmotedu.com和iam.authz.marmotedu.com，我们需要分别为 iam-apiserver 和 iam-authz-server 配置 Nginx 反向代理。整个配置过程可以分为 5 步（在10.0.4.20服务器上操作）。

第一步，配置 iam-apiserver。

新建 Nginx 配置文件/etc/nginx/conf.d/iam-apiserver.conf，内容如下：

复制代码

```
server {
    listen       80;
    server_name  iam.api.marmotedu.com;
    root         /usr/share/nginx/html;
    location / {
      proxy_set_header X-Forwarded-Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_pass  http://127.0.0.1:8080/;
      client_max_body_size 5m;
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

有几点你在配置时需要注意，这里说明下。

server_name需要为iam.api.marmotedu.com，我们通过iam.api.marmotedu.com访问 iam-apiserver。

iam-apiserver 默认启动的端口为8080。

由于 Nginx 默认允许客户端请求的最大单文件字节数为1MB，实际生产环境中可能太小，所以这里将此限制改为 5MB（client_max_body_size 5m）。如果需要上传图片之类的，可能需要设置成更大的值，比如50m。

server_name 用来说明访问 Nginx 服务器的域名，例如curl -H 'Host: iam.api.marmotedu.com' http://x.x.x.x:80/healthz，x.x.x.x为 Nginx 服务器的 IP 地址。

proxy_pass 表示反向代理的路径。因为这里是本机的 iam-apiserver 服务，所以 IP 为127.0.0.1。端口要和 API 服务端口一致，为8080。

最后还要提醒下，因为 Nginx 配置选项比较多，跟实际需求和环境有关，所以这里的配置是基础的、未经优化的配置，在实际生产环境中需要你再做调节。

第二步，配置 iam-authz-server。

新建 Nginx 配置文件/etc/nginx/conf.d/iam-authz-server.conf，内容如下：

复制代码

```
server {
    listen       80;
    server_name  iam.authz.marmotedu.com;
    root         /usr/share/nginx/html;
    location / {
      proxy_set_header X-Forwarded-Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_pass  http://127.0.0.1:9090/;
      client_max_body_size 5m;
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
```

```
20 }
```

下面是一些配置说明。

server_name 需要为`iam.authz.marmotedu.com`，我们通过`iam.authz.marmotedu.com`访问 iam-authz-server。

iam-authz-server 默认启动的端口为`9090`。

其他配置跟`/etc/nginx/conf.d/iam-apiserver.conf`一致。

第三步，配置完 Nginx 后，重启 Nginx：

复制代码

```
1 $ sudo systemctl restart nginx
```

第四步，在 /etc/hosts 中追加下面两行：

复制代码

```
1 127.0.0.1 iam.api.marmotedu.com
2 127.0.0.1 iam.authz.marmotedu.com
```

第五步，发送 HTTP 请求：

复制代码

```
1 $ curl http://iam.api.marmotedu.com/healthz
2 {"status":"ok"}
3 $ curl http://iam.authz.marmotedu.com/healthz
4 {"status":"ok"}
```

我们分别请求 iam-apiserver 和 iam-authz-server 的健康检查接口，输出了`{"status":"ok"}`，说明我们可以成功通过代理访问后端的 API 服务。

在用 curl 请求`http://iam.api.marmotedu.com/healthz`后，后端的请求流程实际上是这样的：

1. 因为在/etc/hosts中配置了127.0.0.1 iam.api.marmotedu.com，所以请求http://iam.api.marmotedu.com/healthz实际上是请求本机的 Nginx 端口（127.0.0.1:80）。
2. Nginx 在收到请求后，会解析请求，得到请求域名为iam.api.marmotedu.com。根据请求域名去匹配 Nginx 的 server 配置，匹配到server_name iam.api.marmotedu.com;配置。
3. 匹配到 server 后，把请求转发到该 server 的proxy_pass路径。
4. 等待 API 服务器返回结果，并返回客户端。

配置 Nginx 作为负载均衡

这门课采用 Nginx 轮询的负载均衡策略转发请求。负载均衡需要至少两台服务器，所以会分别在10.0.4.20和10.0.4.21服务器上执行相同的操作。下面我分别来介绍下如何配置这两台服务器，并验证配置是否成功。

10.0.4.20服务器配置

登陆10.0.4.20服务器，在/etc/nginx/nginx.conf中添加 upstream 配置，配置过程可以分为 3 步。

第一步，在/etc/nginx/nginx.conf中添加 upstream：

复制代码

```
http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
```

```
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;
    upstream iam.api.marmotedu.com {
        server 127.0.0.1:8080
        server 10.0.4.21:8080
    }
    upstream iam.authz.marmotedu.com {
        server 127.0.0.1:9090
        server 10.0.4.21:9090
    }
}
```

配置说明：

upstream 是配置在`/etc/nginx/nginx.conf`文件中的`http{ … }`部分的。

因为我们要分别为 iam-apiserver 和 iam-authz-server 配置负载均衡，所以我们创建了两个 upstream，分别是`iam.api.marmotedu.com`和`iam.authz.marmotedu.com`。为了便于识别，upstream 名称和域名最好保持一致。

在 upstream 中，我们需要分别添加所有的 iam-apiserver 和 iam-authz-server 的后端（`ip:port`），本机的后端为了访问更快，可以使用`127.0.0.1:<port>`，其他机器的后端，需要使用`<内网>:port`，例如`10.0.4.21:8080`、`10.0.4.21:9090`。

第二步，修改 proxy_pass。

修改`/etc/nginx/conf.d/iam-apiserver.conf`文件，将`proxy_pass`修改为：

复制代码

```
proxy_pass http://iam.api.marmotedu.com/;
```

修改`/etc/nginx/conf.d/iam-authz-server.conf`文件，将`proxy_pass`修改为：

复制代码

```
proxy_pass http://iam.authz.marmotedu.com/;
```

当 Nginx 转发到`http://iam.api.marmotedu.com/`域名时，会从`iam.api.marmotedu.com` upstream 配置的后端列表中，根据负载均衡策略选取一个后端，并将请求转发过去。转发`http://iam.authz.marmotedu.com/`域名的逻辑也一样。

第三步，配置完 Nginx 后，重启 Nginx：

复制代码

```
$ sudo systemctl restart nginx
```

最终配置好的配置文件，你可以参考下面这些（保存在configs/ha/10.0.4.20目录下）：

nginx.conf：configs/ha/10.0.4.20/nginx.conf。

iam-apiserver.conf：configs/ha/10.0.4.20/iam-apiserver.conf。

iam-authz-server.conf：configs/ha/10.0.4.20/iam-authz-server.conf。

`10.0.4.21`服务器配置

登陆`10.0.4.21`服务器，在`/etc/nginx/nginx.conf`中添加 upstream 配置。配置过程可以分为下面 4 步。

第一步，在`/etc/nginx/nginx.conf`中添加 upstream：

复制代码

```
http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
```

```
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;
    upstream iam.api.marmotedu.com {
        server 127.0.0.1:8080
        server 10.0.4.20:8080
    }
    upstream iam.authz.marmotedu.com {
        server 127.0.0.1:9090
        server 10.0.4.20:9090
    }
}
```

upstream 中，需要配置10.0.4.20服务器上的 iam-apiserver 和 iam-authz-server 的后端，例如10.0.4.20:8080、10.0.4.20:9090。

第二步，创建/etc/nginx/conf.d/iam-apiserver.conf**文件**（iam-apiserver 的反向代理 + 负载均衡配置），内容如下：

复制代码

```
server {
    listen       80;
    server_name  iam.api.marmotedu.com;
    root         /usr/share/nginx/html;
    location / {
      proxy_set_header X-Forwarded-Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_pass  http://iam.api.marmotedu.com/;
      client_max_body_size 5m;
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

第三步，创建/etc/nginx/conf.d/iam-authz-server文件（iam-authz-server 的反向代理 + 负载均衡配置），内容如下：

复制代码

```
server {
    listen       80;
    server_name  iam.authz.marmotedu.com;
    root         /usr/share/nginx/html;
    location / {
      proxy_set_header X-Forwarded-Host $http_host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_pass  http://iam.authz.marmotedu.com/;
      client_max_body_size 5m;
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}

```

第四步，配置完 Nginx 后，重启 Nginx：

复制代码

```
$ sudo systemctl restart nginx
```

最终配置好的配置文件，你可以参考下面这些（保存在configs/ha/10.0.4.21目录下）：

nginx.conf：configs/ha/10.0.4.21/nginx.conf。

iam-apiserver.conf：configs/ha/10.0.4.21/iam-apiserver.conf。

iam-authz-server.conf：configs/ha/10.0.4.21/iam-authz-server.conf。

测试负载均衡

上面，我们配置了 Nginx 负载均衡器，这里我们还需要测试下是否配置成功。

第一步，执行测试脚本（ test/nginx/loadbalance.sh ）：

复制代码

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

for domain in iam.api.marmotedu.com iam.authz.marmotedu.com
do
  for n in $(seq 1 1 10)
  do
    echo $domain
    nohup curl http://${domain}/healthz &>/dev/null &
  done
done
```

第二步，分别查看 iam-apiserver 和 iam-authz-server 的日志。

这里我展示下 iam-apiserver 的日志（iam-authz-server 的日志你可自行查看）。

10.0.4.20服务器的 iam-apiserver 日志如下图所示：

```
2020-10-13 09:55:47.546 INFO    apiserver/grpc.go:26    Start grpc server at 0.0.0.0:8081
2020-10-13 09:55:47.546 INFO    server/genericapiserver.go:136  Start to listening the incoming requests on
80
2020-10-13 09:55:47.546 INFO    server/genericapiserver.go:153  Start to listening the incoming requests on
443
2020-10-13 09:55:47.546 INFO    server/genericapiserver.go:211  Waiting for the router, retry in 1 second.
[GIN] 2020/10/13 - 09:55:48 | 200 |      65.964µs |       127.0.0.1 | GET      "/healthz"
2020-10-13 09:55:48.547 INFO    server/genericapiserver.go:203  The router has been deployed successfully.
[GIN] 2020/10/13 - 09:56:31 | 200 |        64.1µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      49.813µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      25.328µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      24.396µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      36.358µs |        10.0.4.2 | GET      "/healthz"
```

10.0.4.21服务器的 iam-apiserver 日志如下图所示：

```
2020-10-13 09:55:41.080 INFO    apiserver/grpc.go:26    Start grpc server at 0.0.0.0:8081
2020-10-13 09:55:41.080 INFO    server/genericapiserver.go:136  Start to listening the incoming requests on
80
2020-10-13 09:55:41.080 INFO    server/genericapiserver.go:153  Start to listening the incoming requests on
443
2020-10-13 09:55:41.080 INFO    server/genericapiserver.go:211  Waiting for the router, retry in 1 second.
[GIN] 2020/10/13 - 09:55:42 | 200 |      77.536µs |       127.0.0.1 | GET      "/healthz"
2020-10-13 09:55:42.081 INFO    server/genericapiserver.go:203  The router has been deployed successfully.
[GIN] 2020/10/13 - 09:56:31 | 200 |     121.787µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      55.094µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      24.195µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |       21.73µs |        10.0.4.2 | GET      "/healthz"
[GIN] 2020/10/13 - 09:56:31 | 200 |      76.293µs |        10.0.4.2 | GET      "/healthz"
```

通过上面两张图，你可以看到10.0.4.20和10.0.4.21各收到5个/healthz请求，说明负载均衡配置成功。

配置 Keepalived

在 40 讲，我们分别在10.0.4.20和10.0.4.21服务器上安装了 Keepalived。这里，我来介绍下如何配置 Keepalived，实现 Nginx 的高可用。为了避免故障恢复时，VIP 切换造成的服务延时，这一讲采用 Keepalived 的非抢占模式。

配置 Keepalived 的流程比较复杂，分为创建腾讯云 HAVIP、主服务器配置、备服务器配置、测试 Keepalived、VIP 绑定公网 IP 和测试公网访问六大步，每一步中都有很多小步骤，下面我们来一步步地看下。

第一步：创建腾讯云 HAVIP

公有云厂商的普通内网 IP，出于安全考虑（如避免 ARP 欺骗等），不支持主机通过 ARP 宣告 IP 。如果用户直接在keepalived.conf文件中指定一个普通内网 IP 为 virtual IP，当 Keepalived 将 virtual IP 从 MASTER 机器切换到 BACKUP 机器时，将无法更新 IP 和 MAC 地址的映射，而需要调 API 来进行 IP 切换。所以，这里的 VIP 需要申请腾讯云的 HAVIP。

申请的流程可以分为下面 4 步：

1. 登录私有网络控制台。
2. 在左侧导航栏中，选择【IP 与网卡】>【高可用虚拟 IP】。
3. 在 HAVIP 管理页面，选择所在地域，单击【申请】。
4. 在弹出的【申请高可用虚拟 IP】对话框中输入名称，选择 HAVIP 所在的私有网络和子网等信息，单击【确定】即可。

这里选择的私有网络和子网，需要和10.0.4.20、10.0.4.21相同。HAVIP 的 IP 地址可以自动分配，也可以手动填写，这里我们手动填写为 10.0.4.99。申请页面如下图所示：

第二步：主服务器配置

进行主服务器配置，可以分为两步。

首先，修改 Keepalived 配置文件。

登陆服务器10.0.4.20，编辑/etc/keepalived/keepalived.conf，修改配置，修改后配置内容如下（参考：configs/ha/10.0.4.20/keepalived.conf）：

复制代码

```
# 全局定义，定义全局的配置选项
global_defs {
# 指定keepalived在发生切换操作时发送email，发送给哪些email
# 建议在keepalived_notify.sh中发送邮件
  notification_email {
    acassen@firewall.loc
  }
  notification_email_from Alexandre.Cassen@firewall.loc # 发送email时邮件源地址
    smtp_server 192.168.200.1 # 发送email时smtp服务器地址
    smtp_connect_timeout 30 # 连接smtp的超时时间
    router_id VM-4-20-centos # 机器标识，通常可以设置为hostname
    vrrp_skip_check_adv_addr # 如果接收到的报文和上一个报文来自同一个路由器，则不执行检
    vrrp_garp_interval 0 # 单位秒，在一个网卡上每组gratuitous arp消息之间的延迟时间，
    vrrp_gna_interval 0 # 单位秒，在一个网卡上每组na消息之间的延迟时间，默认为0
}
# 检测脚本配置
vrrp_script checkhaproxy
{
  script "/etc/keepalived/check_nginx.sh" # 检测脚本路径
    interval 5 # 检测时间间隔（秒）
    weight 0 # 根据该权重改变priority，当值为0时，不改变实例的优先级
}
# VRRP实例配置
vrrp_instance VI_1 {
  state BACKUP  # 设置初始状态为'备份'
    interface eth0 # 设置绑定VIP的网卡，例如eth0
    virtual_router_id 51  # 配置集群VRID，互为主备的VRID需要是相同的值
    nopreempt               # 设置非抢占模式，只能设置在state为backup的节点上
    priority 100 # 设置优先级，值范围0～254，值越大优先级越高，最高的为master
    advert_int 1 # 组播信息发送时间间隔，两个节点必须设置一样，默认为1秒
# 验证信息，两个节点必须一致
    authentication {
      auth_type PASS # 认证方式，可以是PASS或AH两种认证方式
        auth_pass 1111 # 认证密码
    }
  unicast_src_ip 10.0.4.20  # 设置本机内网IP地址
    unicast_peer {
      10.0.4.21             # 对端设备的IP地址
    }
# VIP，当state为master时添加，当state为backup时删除
  virtual_ipaddress {
    10.0.4.99 # 设置高可用虚拟VIP，如果是腾讯云的CVM，需要填写控制台申请到的HAVIP地址。
  }
  notify_master "/etc/keepalived/keepalived_notify.sh MASTER" # 当切换到master状
    notify_backup "/etc/keepalived/keepalived_notify.sh BACKUP" # 当切换到backu
    notify_fault "/etc/keepalived/keepalived_notify.sh FAULT" # 当切换到fault状
```

```
    notify_stop "/etc/keepalived/keepalived_notify.sh STOP" # 当切换到stop状态时
    garp_master_delay 1     # 设置当切为主状态后多久更新ARP缓存
    garp_master_refresh 5    # 设置主节点发送ARP报文的时间间隔
    # 跟踪接口，里面任意一块网卡出现问题，都会进入故障(FAULT)状态
    track_interface {
      eth0
    }
  # 要执行的检查脚本
  track_script {
    checkhaproxy
  }
}
```

这里有几个注意事项：

确保已经配置了 garp 相关参数。因为 Keepalived 依赖 ARP 报文更新 IP 信息，如果缺少这些参数，会导致某些场景下主设备不发送 ARP，进而导致通信异常。garp 相关参数配置如下：

复制代码

```
garp_master_delay 1
garp_master_refresh 5
```

确定没有采用 strict 模式，即需要删除 vrrp_strict 配置。

配置中的/etc/keepalived/check_nginx.sh和/etc/keepalived/keepalived_notify.sh脚本文件，可分别拷贝自scripts/check_nginx.sh和scripts/keepalived_notify.sh。

然后，重启 Keepalived：

复制代码

```
$ sudo systemctl restart keepalived
```

第三步：备服务器配置

进行备服务器配置也分为两步。

首先，修改 Keepalived 配置文件。

登陆服务器10.0.4.21，编辑/etc/keepalived/keepalived.conf，修改配置，修改后配置内容如下（参考：configs/ha/10.0.4.21/keepalived.conf）：

复制代码

```
# 全局定义，定义全局的配置选项
global_defs {
# 指定keepalived在发生切换操作时发送email，发送给哪些email
# 建议在keepalived_notify.sh中发送邮件
  notification_email {
    acassen@firewall.loc
  }
  notification_email_from Alexandre.Cassen@firewall.loc # 发送email时邮件源地址
    smtp_server 192.168.200.1 # 发送email时smtp服务器地址
    smtp_connect_timeout 30 # 连接smtp的超时时间
    router_id VM-4-21-centos # 机器标识，通常可以设置为hostname
    vrrp_skip_check_adv_addr # 如果接收到的报文和上一个报文来自同一个路由器，则不执行检
    vrrp_garp_interval 0 # 单位秒，在一个网卡上每组gratuitous arp消息之间的延迟时间，
    vrrp_gna_interval 0 # 单位秒，在一个网卡上每组na消息之间的延迟时间，默认为0
}
# 检测脚本配置
vrrp_script checkhaproxy
{
  script "/etc/keepalived/check_nginx.sh" # 检测脚本路径
    interval 5 # 检测时间间隔（秒）
    weight 0 # 根据该权重改变priority，当值为0时，不改变实例的优先级
}
# VRRP实例配置
vrrp_instance VI_1 {
  state BACKUP  # 设置初始状态为'备份'
    interface eth0 # 设置绑定VIP的网卡，例如eth0
    virtual_router_id 51  # 配置集群VRID，互为主备的VRID需要是相同的值
    nopreempt               # 设置非抢占模式，只能设置在state为backup的节点上
    priority 50 # 设置优先级，值范围0～254，值越大优先级越高，最高的为master
    advert_int 1 # 组播信息发送时间间隔，两个节点必须设置一样，默认为1秒
# 验证信息，两个节点必须一致
    authentication {
      auth_type PASS # 认证方式，可以是PASS或AH两种认证方式
        auth_pass 1111 # 认证密码
    }
  unicast_src_ip 10.0.4.21  # 设置本机内网IP地址
    unicast_peer {
      10.0.4.20             # 对端设备的IP地址
    }
# VIP，当state为master时添加，当state为backup时删除
  virtual_ipaddress {
    10.0.4.99 # 设置高可用虚拟VIP，如果是腾讯云的CVM，需要填写控制台申请到的HAVIP地址。
  }
```

```
  notify_master "/etc/keepalived/keepalived_notify.sh MASTER" # 当切换到master状
    notify_backup "/etc/keepalived/keepalived_notify.sh BACKUP" # 当切换到backu
    notify_fault "/etc/keepalived/keepalived_notify.sh FAULT" # 当切换到fault状
    notify_stop "/etc/keepalived/keepalived_notify.sh STOP" # 当切换到stop状态时
    garp_master_delay 1     # 设置当切为主状态后多久更新ARP缓存
    garp_master_refresh 5    # 设置主节点发送ARP报文的时间间隔
    # 跟踪接口，里面任意一块网卡出现问题，都会进入故障(FAULT)状态
    track_interface {
      eth0
    }
  # 要执行的检查脚本
  track_script {
    checkhaproxy
  }
}
```

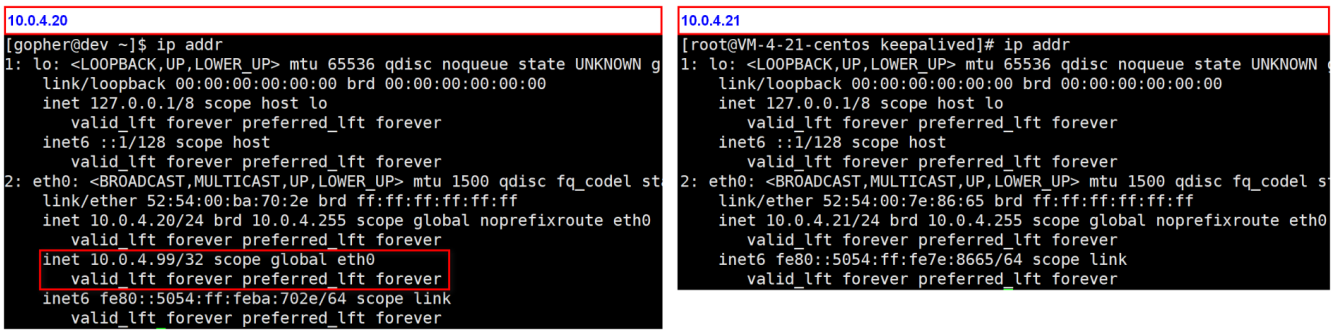
然后，重启 Keepalived：

复制代码

```
$ sudo systemctl restart keepalived
```

第四步：测试 Keepalived

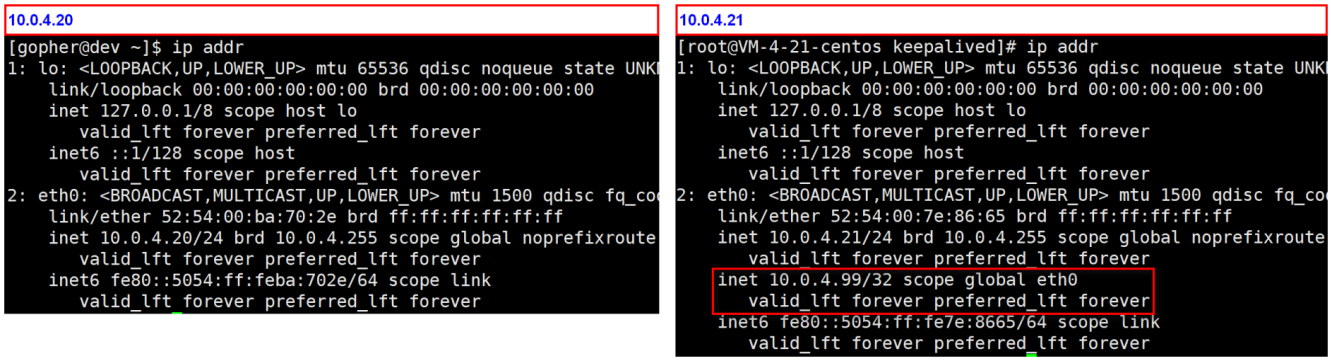
上面的配置中，10.0.4.20的优先级更高，所以正常情况下10.0.4.20将被选择为主节点，如下图所示：



接下来，我们分别模拟一些故障场景，来看下配置是否生效。

场景 1：Keepalived 故障

在10.0.4.20服务器上执行sudo systemctl stop keepalived模拟 Keepalived 故障，查看 VIP，如下图所示：



可以看到，VIP 从10.0.4.20服务器上，漂移到了10.0.4.21服务器上。查看/var/log/keepalived.log，可以看到10.0.4.20服务器新增如下一行日志：

```
1 [2020-10-14 14:01:51] notify_stop
```

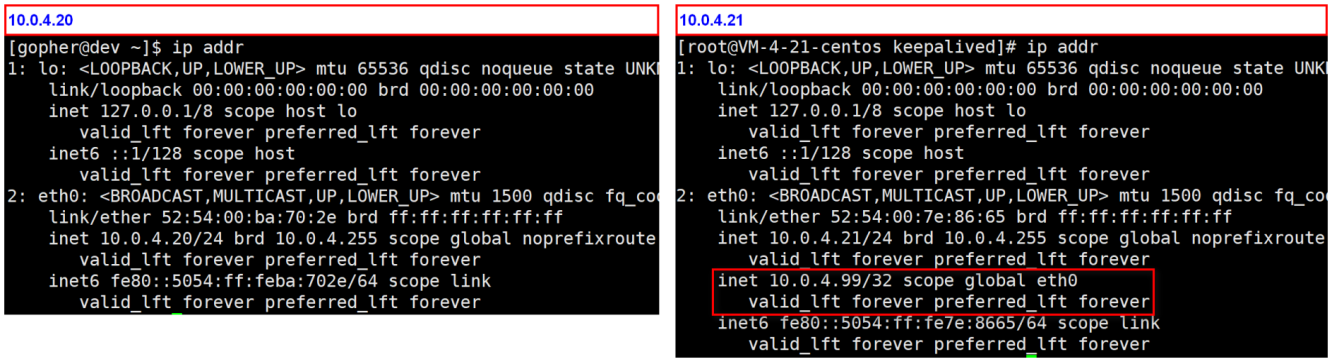
10.0.4.21服务器新增如下日志：

```
1 [2020-10-14 14:01:52] notify_master
```

场景 2：Nginx 故障

在10.0.4.20和10.0.4.21服务器上分别执行sudo systemctl restart keepalived，让 VIP 漂移到10.0.4.20服务器上。

在10.0.4.20服务器上，执行 sudo systemctl stop nginx 模拟 Nginx 故障，查看 VIP，如下图所示：



可以看到，VIP 从10.0.4.20服务器上，漂移到了10.0.4.21服务器上。查看/var/log/keepalived.log，可以看到10.0.4.20服务器新增如下一行日志：

```
1 [2020-10-14 14:02:34] notify_fault
```

复制代码

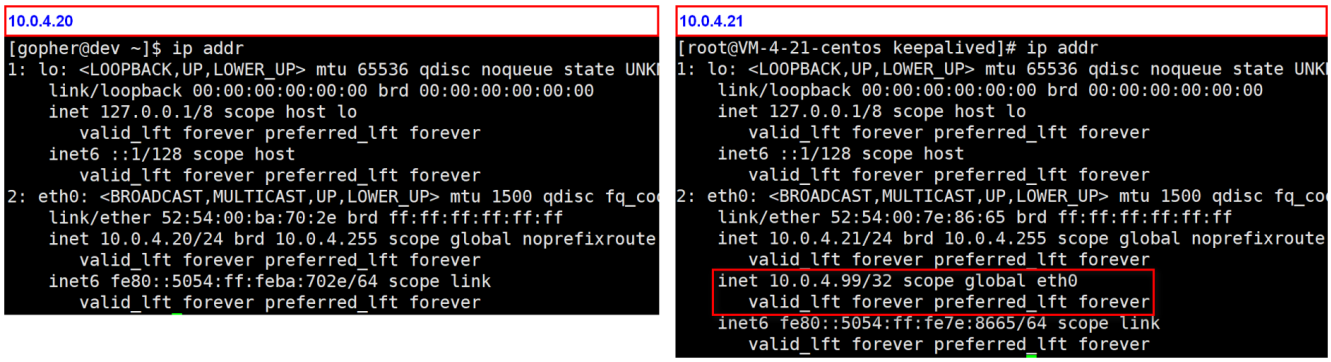
10.0.4.21 服务器新增如下日志：

```
1 [2020-10-14 14:02:35] notify_master
```

复制代码

场景 3：Nginx 恢复

基于**场景 2**，在10.0.4.20服务器上执行sudo systemctl start nginx恢复 Nginx，查看 VIP，如下图所示：



可以看到，VIP 仍然在10.0.4.21服务器上，没有被10.0.4.20抢占。查看/var/log/keepalived.log，可以看到10.0.4.20服务器新增如下一行日志：

```
1 [2020-10-14 14:03:44] notify_backup
```

复制代码

10.0.4.21服务器没有新增日志。

第五步：VIP 绑定公网 IP

到这里，我们已经成功配置了 Keepalived + Nginx 的高可用方案。但是，我们的 VIP 是内网，还不能通过外网访问。这时候，我们需要将 VIP 绑定一个外网 IP，供外网访问。在腾讯云上，可通过绑定弹性公网 IP 来实现外网访问，需要先申请公网 IP，然后将 VIP 绑定弹性公网 IP。下面我来讲讲具体步骤。

申请公网 IP：

1. 登录**私有网络控制台。**
2. 在左侧导航栏中，选择【IP 与网卡】>【弹性公网 IP】。
3. 在弹性公网 IP 管理页面，选择所在地域，单击【申请】。

将 VIP 绑定弹性公网 IP：

1. 登录**私有网络控制台。**
2. 在左侧导航栏中，选择【IP 与网卡】>【高可用虚拟】。
3. 单击需要绑定的 HAVIP 所在行的【绑定】。
4. 在弹出界面中，选择需要绑定的公网 IP 即可，如下图所示：

绑定的弹性公网 IP 是`106.52.252.139`。

这里提示下，腾讯云平台中，如果 HAVIP 没有绑定实例，绑定 HAVIP 的 EIP 会处于闲置状态，按¥0.2/小时 收取闲置费用。所以，你需要正确配置高可用应用，确保绑定成功。

第六步：测试公网访问

最后，你可以通过执行如下命令来测试：

```
$ curl -H"Host: iam.api.marmotedu.com" http://106.52.252.139/healthz -H"iam.ap
{"status":"ok"}
```

可以看到，我们可以成功通过公网访问后端的高可用服务。到这里，我们成功部署了一个可用性很高的 IAM 应用。

总结

今天，我主要讲了如何使用 Nginx 和 Keepalived，来部署一个高可用的 IAM 应用。

为了部署一个高可用的 IAM 应用，我们至少需要两台服务器，并且部署相同的服务 iam-apiserver、iam-authz-server、iam-pump。而且，选择其中一台服务器部署数据库服务：MariaDB、Redis、MongoDB。

为了安全和性能，iam-apiserver、iam-authz-server、iam-pump 服务都是通过内网来访问数据库服务的。这一讲，我还介绍了如何配置 Nginx 来实现负载均衡，如何配置 Keepalived 来实现 Nginx 的高可用。

课后练习

1. 思考下，当前部署架构下如果 iam-apiserver 需要扩容，可以怎么扩容？
2. 思考下，当 VIP 切换时，如何实现告警功能，给系统运维人员告警？

欢迎你在留言区与我交流讨论，我们下一讲见。

分享给需要的人，Ta订阅后你可得 2 元现金奖励

赞 0 提建议

上一篇 40 | 软件部署实战（上）：部署方案及负载均衡、高可用组件介绍

下一篇 42 | 软件部署实战（下）：IAM系统安全加固、水平扩缩容实战

精选留言（1）

写留言

pedro
2021-08-29

iamctl 好用的不行，已经沉淀为了自己的 pctl 了，这不是抄袭，这是模仿～

作者回复: 优秀，哈哈哈

1