CloudOS安装指南

[0. 名词解释](#_tzb8dzqhr8v7)

[1. 安装包结构](#_vxlgjdgn63lj)

[1.1 GCP](#_n5dj5uhp4pxy)

[1.2 DC - Cos Playbook](#_gczyi01geowf)

[1.3 Cluster - Simple Kube](#_jbjz62xaq8e2)

[2. 基本要求](#_8p4e27pmzfzc)

[3. 规划](#_4ublt7ilxd6d)

[4 Inventory](#_t7ak09ryncw3)

[4.1 GCP](#_am5grnr9f3d8)

[4.2 DC](#_f1jkc3l9wy7l)

[4.3 Kubernetes Cluster](#_n5eqjinnowgc)

[5. 安装](#_uruhms2kgxnw)

[5.1 准备](#_y0n4ftgxrec9)

[5.2 初始化](#_3003ieu3ebwk)

[5.3 安装 Global Control Plane（GCP）](#_dt8hztyh81ai)

[5.3.1 初始化SSH](#_55fb5ijon6kt)

[5.3.2 启动安装程序](#_xp3f5t8ka9ym)

[5.3.3 动手修改服务配置](#_f3hrbv40vd4u)

[5.4. 访问CloudOS](#_7vfvt5wbnpf3)

[5.5 添加DC和集群](#_d5sns4gq565m)

[5.5.1 添加DC](#_6g0dsgr1rs74)

[5.5.2 添加集群](#_r61s0psrimlw)

[5.5.3 移除DC](#_1gy3i0cfwmos)

[5.5.4 移除集群](#_ibpzagmsp3d0)

[5.6 安装DC](#_c11j3os9cx24)

[5.7 安装原生Kubernetes集群](#_qle6u8ftynic)

[5.8 在原生Kubernetes/OpenShift集群中安装CloudOS组件](#_on7tk78qltsc)

[5.8.1 初始化Ingress](#_5weuhjpgmjvm)

[6 测试用户](#_9vf5dud1v4l7)

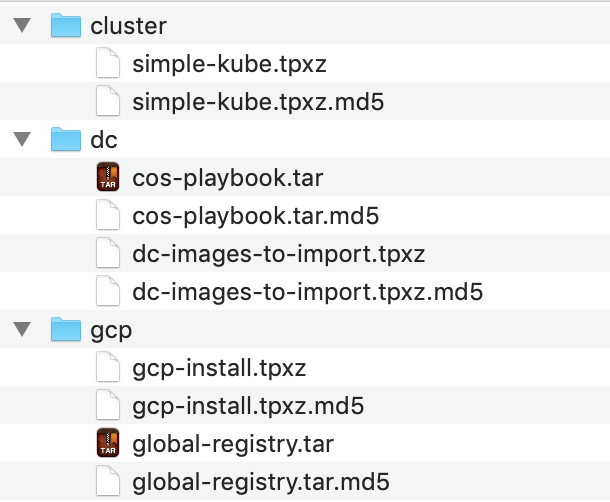
# 0. 名词解释

本节将列出文档中使用的名词和《行云POC环境虚拟机资源申请清单及网络策略》中各角色的对应关系。

|  |  |
| --- | --- |
| Global Control Plane(GCP) | 基础架构（包括的角色有：管理功能，内部DNS和内部代码服务） |
| Main.GCP | 管理功能 |
| DNS.GCP/Global DNS | 内部DNS |
| Git.GCP | 内部代码服务 |
| Ingress | 服务网关 |
| Warehouse | 镜像和日志服务 |

# 1. 安装包结构

安装包由3部分组成，分别是 gcp，dc 和 cluster，这3部分均可以独立运行。在开始完整前，请务必完成文件校验，防止文件损坏。安装包中的所有文件如下图所示，



## 1.1 GCP

如图所示，GCP安装程序包括两部分，即Global Registry和GCP-Install。Global Registry中存放了安装GCP的全部镜像和部分安装DC时需要的镜像。

GCP-Install是Main.GCP、DNS.GCP和Git.GCP的安装程序其使用方法将在后续章节中讲解。

在执行任何安装命令前，需要先部署Global Registry，大致步骤如下，

将 gcp/global-registry.tar 拷贝到任意Host的任意目录，解压gcp-registry.tar，在global\_registry目录下执行 run.sh ，Global Registry将会工作在该Host的5000端口上，即可以通过 HostIP:5000 来获取Global Registry中的镜像。

## 1.2 DC - Cos Playbook

DC的安装程序包括其主要安装逻辑 cos-playbook.tar 和 dc-image-to-import.tpxz。

cos-playbook.tar是一套Ansible Playbook，用来安装CloudOS的DC、Ingress设施和将Kubernetes/OpenShift集群加入CloudOS，它被封装在一个OCI镜像中。部署步骤如下，将 dc/cos-playbook.tar 拷贝到任意Host的任意目录，解压cos-playbook.tar，在cos-playbook目录下执行 run.sh ，将会直接接入cos-playbook的终端，在执行过程中修改过的文件也会被保存，再次在相同位置执行 run.sh 依然可以使用。该playbook的详细说明将在后续章节中讲解。

dc-image-to-import.tpxz中封装了只有在测试阶段才会使用的镜像，因此可以在安装完成后再导入CloudOS。由于CloudOS管理的DC可以自动同步Registry，因此只在其中一个DC中导入即可。导入的方法如下，使用 tar xf -C 命令将该包解压到某个指定的目录，在目标目录中使用docker load 读取镜像并Push镜像到registry.local。

## 1.3 Cluster - Simple Kube

simple-kube.tpxz 是一套用来安装原生Kubernetes集群的Ansible Playbook。和cos-playbook一样它也被封装在一个OCI镜像中。

将 cluster/simple-kube.tpxz 拷贝到任意Host的任意目录，解压cos-playbook.tpxz(tar xf)，在simple-kube目录下执行 run.sh ，将会直接接入simple-kube的终端，在执行过程中修改过的文件也会被保存，再次在相同位置执行 run.sh 依然可以使用。该playbook的详细说明将在后续章节中讲解。

# 2. 基本要求

**⚠️所有Host必须安装有systemd ( Docker 要求）**

**⚠️Linux Kernel版本不得低于3.10.0-693 (WeaveNet 要求)**

**⚠️执行安装程序的Host必须使用UTC时间**

**⚠️如需部署跨集群网络，需确保IaaS关闭“源/目标IP检查”**

**检查方法：**

**⚠️如需部署HA环境，需确保IaaS支持VRRP协议**

**检查方法：**

# 3. 规划

CloudOS由GCP，DC和Kubernetes Cluster3部分组成，在安装前需要做好各个部分的规划，可以使用下表来完成。

对于整个CloudOS而言，有一些全局的配置需要预先确定，如下表所述。

**CloudOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Domain Name** | **GCP VIP** | **Global DNS IP** | **Global Registry** |
| cloud.run.io | 192.168.15.56 | 192.168.15.58 | 192.168.15.56:5000 |

其中，

* **Domain Name**即CloudOS的域名，不能使用主域名否则安装GCP校验不通过
* **GCP VIP**即CloudOS GCP的入口IP，非HA环境即main节点（原infra）IP，
* **Global DNS IP**即CloudOS DNS节点IP，
* **Global Registry IP**即运行GCP registry的节点IP。

对于要使用到的每台Host都需要确定下表所述内容。

**Host**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Hostname** | **IP** | **Internet IP** | **User** | **Password**  **Key** | **Data Disk** | **Docker**  **Storage Driver** | **Primary**  **NIC** |
| main.gcp |  |  |  |  | /dev/sdb |  | ens160 |
| gitlab.gcp |  |  |  |  |  |  |  |
| dns.gcp |  |  |  |  |  |  |  |
| ingress |  |  |  |  |  |  |  |
| warehouse  (backup) |  |  |  |  |  |  |  |
| master |  |  |  |  |  |  |  |

其中，

* **Hostname**即主机名，可随意设置，
* **IP**即Host内网IP地址，
* **Internet IP**即Host的公网IP，
* **User/Password/Key**记录Host的鉴权信息，
* **Data Disk**即Host的数据盘信息，数据盘需要分区和挂载保持裸盘状态，
* **Docker Storage Driver**即Docker的存储驱动，其值可以是 **overlay2** 或 **direct-lvm**。Linux Kernel版本为4.0及以上时选择overlay2，其他情况下选择direct-lvm，注意 direct-lvm 要求必须配备数据盘，RHEL在应用了某个Kernel Patch的情况下可以使用overlay2，可灵活处理，
* **Primary NIC**即Host上内网IP地址绑定的网卡。

对于每个DC需要确定如下信息，

**DC**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DC Name** | **DC ID** | **Ingress IP(\*)** | **Ingress Internet**  **IP（\*）** | **Warehouse**  **IP** | **Tag** |
| DC-A | db081bc2-22fb-4a59-b6c0-313b5966f361 |  |  |  | key=value |
| DC-B |  |  |  |  |  |

其中，

* **DC Name** 即DC的名字，20字符以内的字符串即可，原则上可以支持中文，
* **DC ID** 即DC的ID，因为和集群的配置有关，安装完成后不可以修改，原则上可以是任意字符串，通常会使用UUID，生成方式 ：cat /proc/sys/kernel/random/uuid
* **Ingress IP**即接入服务的内网IP，即原gateway和ingress，可以有多个，目前第一个会被用来做DC之间通信的gateway，
* **Ingress Internet IP**即Ingress的公网IP，部署在公网是会用到，
* **Warehouse IP**即原registry（程序中体现为backup）的内网IP，其上部署了Registry，InfluxDB，ElasticSearch以及DC的DNS服务，
* **Tag**即标签，目前DC标签并没有实际的用处。

对于每个Kubernetes集群需要确定如下信息，

**Kubernetes Cluster**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cluster Name** | **Cluster ID** | **DC ID** | **Service CIDR** | **Pod**  **CIDR** | **Node**  **CIDR** | **VIP** | **Tag** |
| Custer-A1 | e6695d27-d15f-4c81-a37d-d8d0e0854650 | db081bc2-22fb-4a59-b6c0-313b5966f361 | 10.21.0.0/16 | 10.20.0.0/16 | 192.168.1.0/24 | 192.168.1.11 | key=value |
| Cluster-A2 | 5d561cc2-dcb8-492b-9e3b-3d184c07779c | db081bc2-22fb-4a59-b6c0-313b5966f361 | 10.22.0.0/16 | 10.23.0.0/16 |  |  |  |

其中，

* **Cluster Name**即集群的名字，20字符以内的字符串即可，原则上可以支持中文，
* **Cluster ID**即集群的ID，原则上可以是任意字符串，通常会使用UUID，
* **DC ID**即集群所在的DC的ID，
* **Service CIDR**和**Pod CIDR**为两个网段，他们不能相互覆盖，也不能和节点的网段覆盖，也不能和其他集群的CIDRs覆盖，其中Pod CIDR中的IP数量不能小于Service CIDR，且要基本能保证每个节点上可以同时使用200个IP，
* **Node CIDR**即节点所在子网网段，
* **VIP**即HA集群Master节点的VIP，需要处于**Node CIDR**中，
* **Tag**即标签，该标签和CloudOS任务调度有关，支持中文。

# 4 Inventory

对于Ansible Playbook来说，Inventory是唯一需要准备的配置。对于CloudOS而言，我们需要准备如下的Inventory。

## 4.1 GCP

GCP需要1份Inventory，用来做初始化工作。模板可以从cos-playbook的inventory/gcp-sample处获得，其中需要修改的地方如下， 其内容均来自第3节中的**Host**表。

1. 在[gcp]组中增加GCP中个Host的配置，
2. 设置 storage\_driver 变量，
3. 设置 data\_disk 变量，
4. 设置 global\_registry 变量。

## 4.2 DC

每个DC也需要1份Inventory，用来安装DC，并为集群的Inventory提供配置。模版可以从cos-playbook的inventory/sample处获得，其中需要修改的内容如下， 其内容均来自第3节中的**Host**表和**DC**表。

1. 在[ingress]组中增加当前DC的所有Ingress的配置，
2. 在[warehouse]组中增加当前DC的Warehouse的配置，
3. 设置 storage\_driver 变量，
4. 设置 data\_disk 变量，
5. 设置 global\_registry 变量,

- 在 ingress 和 warehouse的 /etc/docker/daemon.json中添加内容

{ "insecure-registries":["<global-registry-ip>:5000"] }

- inventory 文件中global\_registry的值设置为 <global-registry-ip>:5000

1. 设置 global\_dns 变量，
2. 设置 infra\_vip 变量，
3. 设置 domain 变量，
4. 设置 data\_center\_id 变量。

## 4.3 Kubernetes Cluster

每个集群需要2份Inventory，我们将它们命名为 cos-inventory 和 k8s-inventory 。cos-inventory用来将原生Kubernetes集群或OpenShift集群添加到指定DC，它需要以集群所在DC的Inventory为模板，在其中做出如下修改，其内容均来自第3节中的**Host**表和**Kubernetes Cluster**表。

1. 在[master]组中增加当前集群的所有Master节点的配置，
2. 在[storage]组中增加当前集群的Storage节点的配置，
3. 在[node]组中增加当前集群的Node节点的配置，
4. 设置 storage\_driver 变量，
5. 设置 data\_disk 变量，
6. 设置 global\_registry 变量，
7. 设置 cluster\_id 变量，
8. 设置 node\_nic 变量，
9. 设置 pod\_cidr 变量，
10. 设置 svc\_cidr 变量，
11. 设置 node\_cidr 变量，
12. 设置 all\_k8s\_cidrs 变量，该变量的值包括所有的Pod CIDR和Service CIDR，用空格分隔。

k8s-inventory用来安装原生Kubernetes集群。模板可以从simple-kube的inentory/sample处获得，其中需要修改的内容如下，其内容均来自第3节中的**Host**表和**Kubernetes Cluster**表。

1. 在[master]组中增加当前集群的所有Master节点的配置，
2. 在[node]组中增加当前集群的Node节点和Storage节点的配置，
3. 设置 pod\_cidr 变量，
4. 设置 svc\_cidr 变量，
5. 设置 apiserver\_vip 变量

# 5. 安装

## 5.1 准备

1. 所有Host都必须有可用的Yum/APT源，可以是互联网源，也可以是内网或者ISO源。
2. 部署Global Registry，参考1.1节。

[base]

name=CentOS7-base

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra

baseurl=http://10.10.14.74:9999/centos7

gpgcheck=1

enabled=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

## 5.2 初始化

使用第4节中所有为cos-playbook准备的Inventory文件（若仅是将已经安装好的Kubernetes/OpenShift集群加入CloudOS，则这些集群的Inventory文件应当被跳过），在 cos-playbook 中执行

|  |
| --- |
| ansible-playbook -i INVENTORY install-deps.yml |

其中，将 INVENTORY 替换成需要执行的Inventory文件，每次执行只能指定一个Inventory文件。

**⚠️属于相同DC中的集群，及它们所在的DC不可以同时执行。**

若执行失败，可这样清理，

|  |
| --- |
| ansible-playbook -i INVENTORY clean-deps.yml |

## 5.3 安装 Global Control Plane（GCP）

在校验完安装文件后，您需要将安装文件gcp/gcp-install.tpxz解压(tar xf)出来。安装程序中包含了大量的数据包和安装程序，其中installer即是安装程序。

### 5.3.1 初始化SSH

根据《行云POC环境虚拟机资源申请清单及网络策略》创建所有虚拟机后，您需要配置虚拟机的SSH服务，步骤如下。

|  |
| --- |
| AcceptEnv LC\_SUBDOMAIN LC\_JWT LC\_MYSQL\_SERVER LC\_API\_INFRA\_EIP LC\_MYSQL\_ROOT\_PASSWORD LC\_DNS\_RESOLVER LC\_GIT\_EIP LC\_GIT\_TOKEN LC\_INFRA\_IP LC\_LOCALHOST\_IP LC\_EIP LC\_INSTALLNAME LC\_REGISTRY |

1. 将以上内容内容添加到**每台**虚拟机的 **/etc/ssh/sshd\_config** 文件中，
2. 执行 systemctl restart sshd 重启ssh服务。

### 5.3.2 启动安装程序

安装程序有交互式和非交互式模式两种工作模式。我们建议您使用交互模式开始。在交互模式中，您会得到关于接下来要输入的内容的更多提示。

1. 直接在Linux shell中运行./installer， 交互模式便会启动。您会得到如下提示。

|  |
| --- |
| Global Registry(e.g. 192.168.1.11:5000): |

您需要输入Global Registry的位置，它由Global Registry所在Host的IP地址和端口组成。

|  |
| --- |
| Rename CloudOS[CloudOS]: |

接下来如果您需要将CloudOS命名为其他产品（e.g. OpenShift），则可以在这里设置，否则可跳过。

|  |
| --- |
| Subdomain for CloudOS: |

您需要为CloudOS提供一个子域名。 您可以通过这个子域名访问CloudOS和部署在CloudOS上的各种应用。该值对应《行云POC环境虚拟机资源申请清单及网络策略》表3中的域名。

1. 配置内部代码服务

输入域名后，安装程序会提示您输入代码服务的虚拟机信息。

|  |
| --- |
| IPv4 Address of the Git:  Public IPv4 Address of the Git:  User:  Password[SSH key will be used if leave this blank]:  SSH Key[~/.ssh/id\_rsa.Type "" to ignore defaults]: |

您需要依次输入，

* **内部代码服务**的IPv4地址，
* 其公网地址，若无公网地址可直接跳过，
* 用户名，必须是root，
* 密码，若不使用密码可直接跳过，
* SSH证书，如比使用证书则输入“”**。**

1. 配置管理功能

接下来安装程序会提示您输入Infrastructure的主机信息。

|  |
| --- |
| IPv4 Address of the Infrastructure Server:  Public IPv4 Address of the Infrastructure Server:  User:  Password[SSH key will be used if leave this blank]:  SSH Key[~/.ssh/id\_rsa.Type "" to ignore defaults]:  Would you like to install all components on the Infrastructure Server?[y/N] |

您需要依次输入，

* **管理功能**的IPv4地址，
* 其公网地址，若无公网地址可直接跳过，
* 用户名，必须是root，
* 密码，若不使用密码可直接跳过，
* SSH证书，如比使用证书则输入“”**，**
* N，若输入y，则DNS服务不能被安装到其他虚拟机。

1. 配置数据库服务

此处我们将数据库服务安装在**管理功能**节点上。您会看到如下提示。

|  |
| --- |
| IPv4 Address of the Database Server:  Public IPv4 Address of the Database Server:  User:  Password[SSH key will be used if leave this blank]:  SSH Key[~/.ssh/id\_rsa.Type "" to ignore defaults]: |

您需要依次输入，

* **管理功能**的IPv4地址，
* 其公网地址，若无公网地址可直接跳过，
* 用户名，必须是root，
* 密码，若不使用密码可直接跳过，
* SSH证书，如比使用证书则输入“”**。**

1. 配置内部DNS

您会看到如下提示。

|  |
| --- |
| IPv4 Address of the DNS resolver:  Public IPv4 Address of the DNS resolver:  User:  Password[SSH key will be used if leave this blank]:  SSH Key[~/.ssh/id\_rsa.Type "" to ignore defaults]: |

您需要依次输入，

* **内部DNS**的IPv4地址，
* 其公网地址，若无公网地址可直接跳过，
* 用户名，必须是root，
* 密码，若不使用密码可直接跳过，
* SSH证书，如比使用证书则输入“”**。**

设置完成后，安装程序将开始的数据传输和安装服务。安装进度和错误将同时输出到名为cloudos.log的日志文件和控制台。

### 5.3.3 动手修改服务配置

1. 修改发布域名：

将发布应用的后缀域名修改为与安装程序中设置的子域名相同，假定设置的子域名为cloud.abc.com，则在infra上做如下操作：

进去数据库容器：

docker exec -ti biz-db bash

查询mysql密码

env|grep YSQL\_ROOT\_PASSWORD

用查询出来的密码登录mysql数据库：

mysql -uroot -pXXXX         #XXXX为查询的密码

再执行如下的sql语句：

use service\_user;

delete from dict\_data  where data\_code = '.c.microservice.shop' and data\_name = '.c.microservice.shop';

delete from dict\_data  where data\_code = '.c.thecloudapps.cn' and data\_name = '.c.thecloudapps.cn';

update dict\_data  set data\_code ='.XXX' ,data\_name ='.XXX',remark='.XXX' where data\_code = '.c.cloudtogo.cn';

注:update语句中的XXX前面带有符号”.”，XXX的值为设置的子域名的值。

检验是否修改成功：

执行SQL命令：select \* from dict\_data;

若有data\_code值为.XXX的记录，则修改成功。

1. 修改registry地址：

在 main.gcp 上登录MySQL，将service\_registry.registry\_provider表中ID为default的记录的uri字段修改为某一个DC的Registry(通常是http://{{WarehouseIP}}:5001)，并重启 registry-agent 容器(SQL命令可参考 update service\_registry.registry\_provider set uri="YOUR REGISTRY" where id = 'default'，可以使用如下命令查询 select \* from service\_registry.registry\_provider where id = 'default';

修改完成后重启factory

1. 初始化sys-config服务配置：

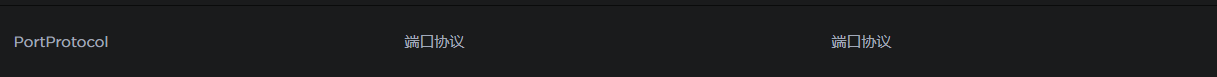
确认sys-config服务可用后，在main.gcp上执行 curl -v -H "Content-Type:application/json" -H "uid:admin" -X POST --data '{"map": { "bitbucket": true, "git": true, "github": true, "gitlab": true, "svn": true, "project": true, "team": true }, "name": "应用工厂", "website": "factory"}http://localhost:8028/web/config

1. 手动修改infra的/etc/hosts文件:

将registry.local映射为DC服务器IP，否则工厂创建镜像组件无法查找到系统内镜像。修改完成后重启infra中registry-agent容器

1. 添加通信协议：

系统启动后进入运维中心，添加字典类型“PortProtocol”：



在该类型下添加字典：



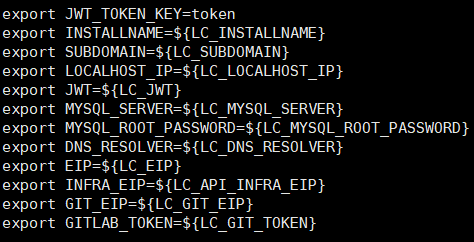
1. 修改Composer 支持 Service Mesh：

Composer目前仅支持从某个特定集群读取数据，且需要在集群安装完成后将集群的信息作为参数配置给Composer服务，方法为位于main.gcp的composer容器的设置环境变量 api\_path\_status=http://${CLUSTER\_MASTER\_HOST}:30235/graph，其中CLUSTER\_MASTER\_HOST为某个K8s集群的Master节点IP。具体修改方法：

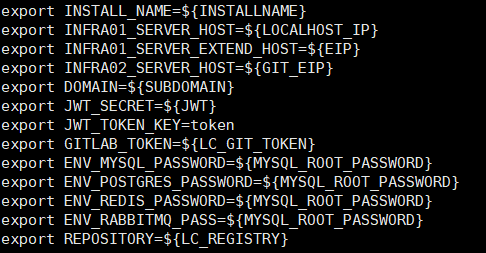
6.1.进入/root/.ctg/infra.pack.res目录

6.2.chmod +x gcp-images.sh 将gcp-images.sh文件设置为可执行

6.3.进入 install.sh 文件头部的环境变量，如：



和文件尾部的环境变量，如：



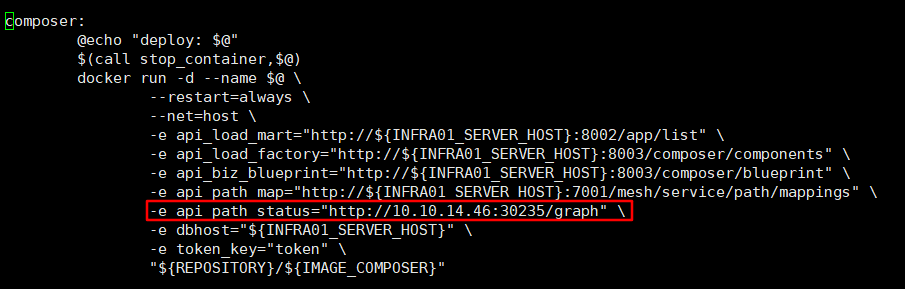
复制到manual.sh文件中，删除manual.sh文件的. "$(pwd)/install.sh"行

6.4.在manual.sh文件末尾添加：

export REPOSITORY=10.10.14.41:5000 (值为安装时Global Registry的地址和端口)

. "$(pwd)/gcp-images.sh"

make -f biz-install.makefile composer

6.5.修改biz-install.makefile，修改composer:下的环境变量值，修改为集群master地址

6．6.执行manual.sh重新安装composer，等composer启动完成即可

1. 测试服务，该服务的基础容器都已经部署，但仍然需要在集群完整完成后修改服务配置和数据库内容，具体方法见《测试服务容器修改》文档

### 5.3.4 重装gcp

gcp重新安装前先停止infra、gitlab、dns的所有docker容器docker stop $(docker ps -aq)，清除main.gcp上/var/lib/cloudtogo目录，清除git.gcp 的 /home/gitlab目录，清除dns.gcp 的 /etc/bind目录。再开始重新安装，否则可能出现che启动失败导致创建新用户没有创建gitlab账号的情况

## 5.4. 访问CloudOS

至此，CloudOS的GCP已经安装完成。您可以通过访问http://factory.[subdomain]和http://mart.[subdomain]来分别访问应用工厂和应用商店，也可以通过http://admin.[subdomain]在 系统配置 中为其他用户创建账号。超级管理员的用户名是**admin@ctg.cn**，密码为**admin123**。

## 5.5 添加DC和集群

尽管DC和集群还没有安装，但是我们可以先在GCP中将它们配置好。添加DC和集群都是通过Turtle服务的客户端完成。添加完成后，在运维中心即可看到离线的集群。

### 5.5.1 添加DC

在 main.gcp 上执行 docker exec -ti turtle sh 进入Turtle服务，在Turtle服务中执行如下命令。

|  |
| --- |
| cliturtle dc -a \  --tag K1=V1 --tag K2=V2 \  --gateway-primary-ip 10.10.10.10 \  --gateway-public-ip 112.10.11.11 \  --registry-ip 10.10.10.11 \  --id DC-ID --name DC-Name |

各参数说明如下，

|  |  |
| --- | --- |
| --tag | DC的Tag，格式为 Key=Value |
| --gateway-primary-ip | 第一个Ingress的内网IP |
| --gateway-public-ip | 第一个Ingress的公网IP，没有可略过 |
| --registry-ip | Warehouse节点内网IP |
| --id | DC ID |
| --name | DC名称 |

### 5.5.2 添加集群

在 main.gcp 上执行 docker exec -ti turtle sh 进入Turtle服务，在Turtle服务中执行如下命令。

|  |
| --- |
| cliturtle cluster -a \  --tag K3=V3 --tag K4=V4 \  --pod-cidr=10.20.0.0/16 --service-cidr=10.21.0.0/16 \  --dc-id=DC-ID --id=Cluster-ID --name=Cluster-Name |

各参数说明如下，

|  |  |
| --- | --- |
| --tag | 集群的Tag，格式为 Key=Value |
| --pod-cidr | Pod子网 |
| --service-cidr | Service子网 |
| --dc-id | 集群所在DC ID |
| --id | 集群ID |
| --name | 集群名称 |

### 5.5.3 移除DC

在 main.gcp 上执行 docker exec -ti turtle sh 进入Turtle服务，在Turtle服务中执行如下命令。

|  |
| --- |
| cliturtle dc -d --id DC-ID |

### 5.5.4 移除集群

在 main.gcp 上执行 docker exec -ti turtle sh 进入Turtle服务，在Turtle服务中执行如下命令。

|  |
| --- |
| cliturtle cluster -d --dc-id=DC-ID --id=Cluster-ID |

## 5.6 安装DC

使用第4.2节中为DC创建的Inventory文件，在 cos-playbook 中执行

|  |
| --- |
| ansible-playbook -i INVENTORY install-dc.yml |

其中，将 INVENTORY 替换成需要执行的Inventory文件。

若执行失败，可这样清理，

|  |
| --- |
| ansible-playbook -i INVENTORY clean-dc.yml |

注意，往kuberents集群中安装CloudOS组件时会往DC中写入集群集群信息，DC清理时这些数据也被清理。清理DC后要使DC中的集群可用，除了重新安装DC外，还需要重新执行5.8章节中的安装内容。

## 5.7 安装原生Kubernetes集群

使用第4.3节中为集群创建的k8s-inventory文件，在 simple-kube 中执行

|  |
| --- |
| ansible-playbook -i INVENTORY deploy-cluster.yml |

其中，将 INVENTORY 替换成需要执行的Inventory文件。

若执行失败，可这样清理，

|  |
| --- |
| ansible-playbook -i INVENTORY clean-cluster.yml |

## 5.8 在原生Kubernetes/OpenShift集群中安装CloudOS组件

如需CloudOS组件安装至Kubernetes/OpenShift集群，需要保证如下条件，

1. 在集群内可以使用registry.local从所属DC获取镜像，
2. Docker的log driver需要设置为json-file，
3. 确认Etcd使用的证书和认证文件位置，并替换cos-inventory中的 etcd\_certfile , etcd\_keyfile和etcd\_cafile 。

使用本文档安装的Kubernetes集群已经具备了上述条件，而OpenShift集群的调整方法可参考本节的附注。

使用第4.3节中为DC创建的cos-inventory文件，在 cos-playbook 中执行

|  |
| --- |
| ansible-playbook -i INVENTORY install-cluster.yml |

其中，将 INVENTORY 替换成需要执行的Inventory文件。

若执行失败，可这样清理，

|  |
| --- |
| ansible-playbook -i INVENTORY clean-cluster.yml |

### 5.8.1 初始化Ingress

在将第一个集群加入DC后，需要在Ingress Master上执行 curl http://127.0.0.1:1402/init ，但每个DC只可以执行一次。由于Ingress（hapa-master）在没有配置集群前不能正确工作，因此需要在安装完第一个集群后执行。

### 5.8.2 恢复WeaveNet

在master上执行 skc get po ，有可能会存在处于CrashLoopbackoff状态的weavenet，可以检查Pod中weave-npc容器的日志，若发现有类似 “ipset被kernel占用，无法删除（英文）”的日志，只能通过重启Pod所在Host恢复。

### 5.8.3 恢复Istio

在master上执行 kc get po -n istio-system，若存在不正常的Pod，删除即可。

注：为OpenShift Host添加registry.local Registry并修改log driver

在安装开始前，您需要做一些准备工作。如下，

1. 修改OpenShift/OKD集群中各节点的DNS配置，使其可以使用CloudOS的资源。修改**各个节点**的 /etc/dnsmasq.d/origin-upstream-dns.conf 文件，将其中的server的值改成**镜像和日志服务**虚拟机的IP地址，然后执行 systemctl restart dnsmasq 来重启DNS服务。
2. 修改OpenShift/OKD集群中各节点的Docker配置，使其可以使用CloudOS的镜像服务。修改**各个节点**的 /etc/containers/registries.conf 文件，在 [registries.insecure] 小结的registries中追加 registry.local 。如 registries = ['registry.local']，然后执行 systemctl enable registries && systemctl restart registries和systemctl restart docker 来使配置生效。
3. 修改**各个节点**的Docker配置使日志收集系统生效。修改/etc/sysconfig/docker文件，在OPTIONS=’’一行中追加--log-driver=json-file，然后执行systemctl restart docker来使配置生效。

# 6 测试用户

安装完成后，请不要使用admin@ctg.cn作为测试用户。您需要在运维中心创建新用户。