从Docker到Kubernetes企业应用实战

个人介绍

讲师：李振良

资深运维工程师，51CTO知名博主。曾就职在IDC，大数据，金融行业，现任职360公司，经重重磨练，具备丰富的运维实战经验。

技术博客：http://blog.51cto.com/lizhenliang

DevOps技术栈

专注于分享DevOps工具链及经验总结。

微信扫一扫

Docker/K8s技术学员群：397834690

课程目录

案例一 整套项目打包部署

Docker Compose

1、介绍

Compose是一个定义和管理多容器的工具，使用Python语言编写。使用Compose配置文件描述多个容器应用的架构，比如使用什么镜像、数据卷、网络、映射端口等；然后一条命令管理所有服务，比如启动、停止、重启等。

2、安装

```
curl -L https://github.com/docker/compose/releases/download/1.15.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

或者

```
pip install docker-compose
```

3、YAML文件格式及编写注意事项

YAML是一种标记语言很直观的数据序列化格式，可读性高。类似于XML数据描述语言，语法比XML简单的很多。

YAML数据结构通过缩进来表示，连续的项目通过减号来表示，键值对用冒号分隔，数组用中括号括起来，hash用花括号括起来。

YAML文件格式注意事项：

1. 不支持制表符tab键缩进，需要使用空格缩进
2. 通常开头缩进2个空格
3. 字符后缩进1个空格，如冒号、逗号、横杆
4. 用井号注释
5. 如果包含特殊字符用单引号引起来
6. 布尔值（true、false、yes、no、on、off）必须用引号括起来，这样分析器会将他们解释为字符串。

案例一 整套项目打包部署

Compose应用案例

配置文件常用字段：

字段	描述

build
  dockerfile
  context
	指定Dockerfile文件名
	构建镜像上下文路径

image	指定镜像
command	执行命令，覆盖默认命令
container_name	指定容器名称，由于容器名称是唯一的，如果指定自定义名称，则无法scale
deploy	指定部署和运行服务相关配置，只能在Swarm模式使用
environment	添加环境变量
networks	加入网络，引用顶级networks下条目
ports	暴露端口，与-p相同，但端口不能低于60
volumes	挂载宿主机路径或命名卷。如果是命名卷在顶级volumes定义卷名称
restart	重启策略，默认no，always|on-failure|unless-stopped
hostname	容器主机名

官方文档：https://docs.docker.com/compose/compose-file

案例一 整套项目打包部署

Compose应用案例

常用命令：

字段	描述
build	重新构建服务
ps	列出容器
up	创建和启动容器
exec	在容器里执行命令
scale	指定一个服务容器启动数量
top	显示容器进程
logs	查看容器输出
down	删除容器、网络、数据卷和镜像
stop/start/restart	停止/启动/重启服务

案例一 整套项目打包部署

Compose应用案例

- 一键部署LNMP网站平台
- 一键部署Nginx反向代理Tomcat集群

案例二 构建持续集成环境

1、CI/CD介绍

2、发布流程设计

3、部署Git服务器

4、部署Harbor镜像仓库（已完成）

5、构建业务基础镜像

6、测试服务器安装Docker

7、Jenkins安装

8、Jenkins基本配置

9、Jenkins创建项目

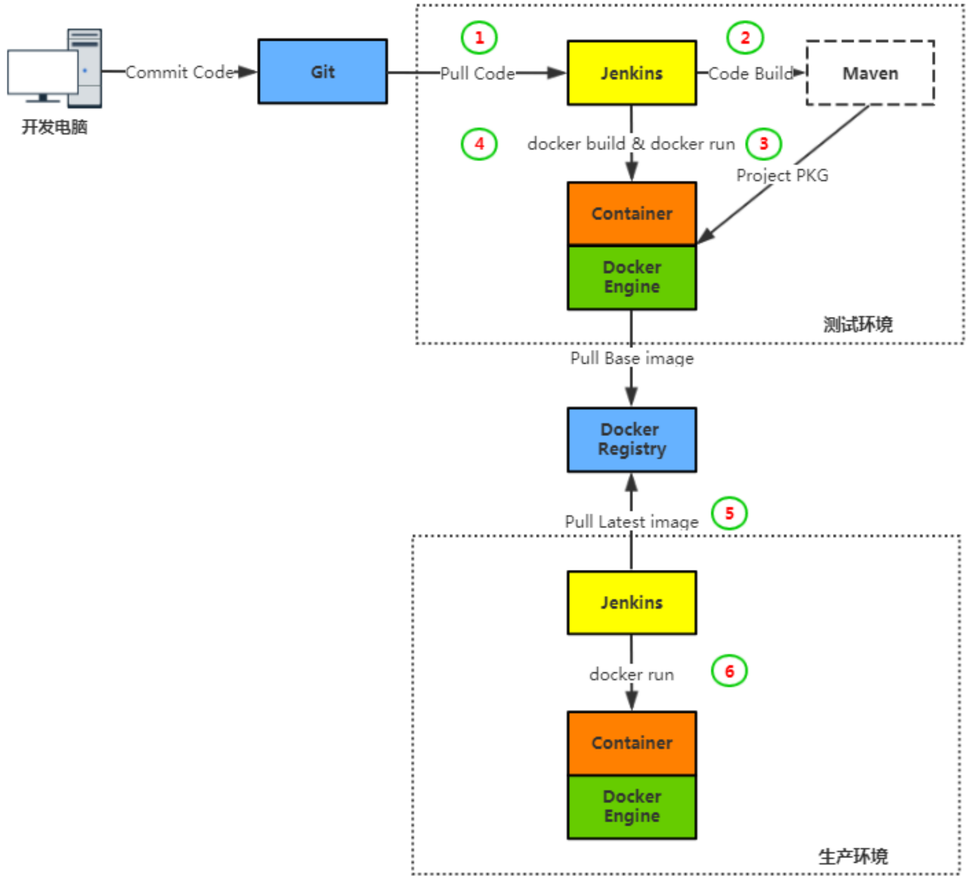
10、测试

案例二 构建持续集成环境

CI（持续集成）/CD（持续交付/持续部署）

案例二 构建持续集成环境

发布流程设计



发布流程设计

服务器	IP
Jenkins服务器	192.168.0.211
Docker服务器	192.168.0.212
Git/Harbor	192.168.0.213

工具	版本
CentOS	7.4_x64
Maven	3.5
Tomcat	8
JDK	1.8
Jenkins	2.6
Docker CE	17.06

案例二 构建持续集成环境

部署Git服务器

1、安装Git

```
# yum install git
```

2、创建Git用户

```
# useradd git
# passwd git
```

3、创建仓库

```
# su - git
# mkdir app.git
# git -bare init
```

案例二 构建持续集成环境

测试服务器安装Docker

1、安装Docker

```
yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
yum install docker-ce
```

2、配置官方国内镜像仓库与添加私有仓库可信任

```
cat > /etc/docker/daemon.json << EOF
{
  "registry-mirrors": [ "https://registry.docker-cn.com"],
  "insecure-registries":["192.168.0.213:5000"]
}
EOF
systemctl restart docker
```

案例二 构建持续集成环境

Jenkins安装

```
FROM jenkins

USER root

RUN echo '' > /etc/apt/sources.list.d/jessie-backports.list && \
  wget http://mirrors.163.com/.help/sources.list.jessie -O /etc/apt/sources.list

RUN apt-get update && apt-get install -y git libltdl-dev
```

```
docker run -d \
--name jenkins \
-p 8080:8080 \
-v /var/jenkins_home/:/var/jenkins_home \
-v /usr/local/apache-maven-3.5.0:/usr/local/maven \
-v /usr/local/jdk1.8.0_45:/usr/local/jdk \
-v /var/run/docker.sock:/var/run/docker.sock \
-v $(which docker):/usr/bin/docker \
-v ~/.ssh:/root/.ssh \
 jenkins:v1
```

案例二 构建持续集成环境

Jenkins基本配置与创建项目

系统设置
全局设置&路径

添加Publish Over SSH远程主机（Docker主机）

Global Tool Configuration
Configure tools, their locations and automatic installers.

配置Maven、JDK、Git环境

案例三 容器服务自动注册与发现

Consul

1、介绍

Consul是一个分布式、高可用性，在基础设施中发现和配置服务的工具。

2、安装

下载二进制Consul包：https://www.consul.io/downloads.html

```
# unzip consul_0.9.2_linux_amd64.zip
# mv consul /usr/bin
```

3、部署

```
# consul agent \
-server \
-bootstrap \
-ui \
-data-dir=/var/lib/consul-data \
-bind=192.168.0.211 \
-client=0.0.0.0 \
-node=server01
```

案例三 容器服务自动注册与发现

Consul

查看集群信息：

```
consul members
consul info |grep leader
consul catalog services
```

通过HTTP API获取集群信息：

```
curl 127.0.0.1:8500/v1/status/peers            # 集群server成员
curl 127.0.0.1:8500/v1/status/leader           # 集群Raft leader
curl 127.0.0.1:8500/v1/catalog/services        # 注册的所有服务
curl 127.0.0.1:8500/v1/catalog/services/nginx  # 服务信息
curl 127.0.0.1:8500/v1/catalog/nodes           # 集群节点详细信息
```

服务注册：

```
curl -X PUT -d \
'{"id": "jetty","name": "service_name","address": "192.168.0.212","port": 8080,"tags": ["test"],"checks": [{"http":
"http://192.168.0.212:8080/","interval": "5s"}]}' \
http://192.168.0.211:8500/v1/agent/service/register
```

案例三 容器服务自动注册与发现

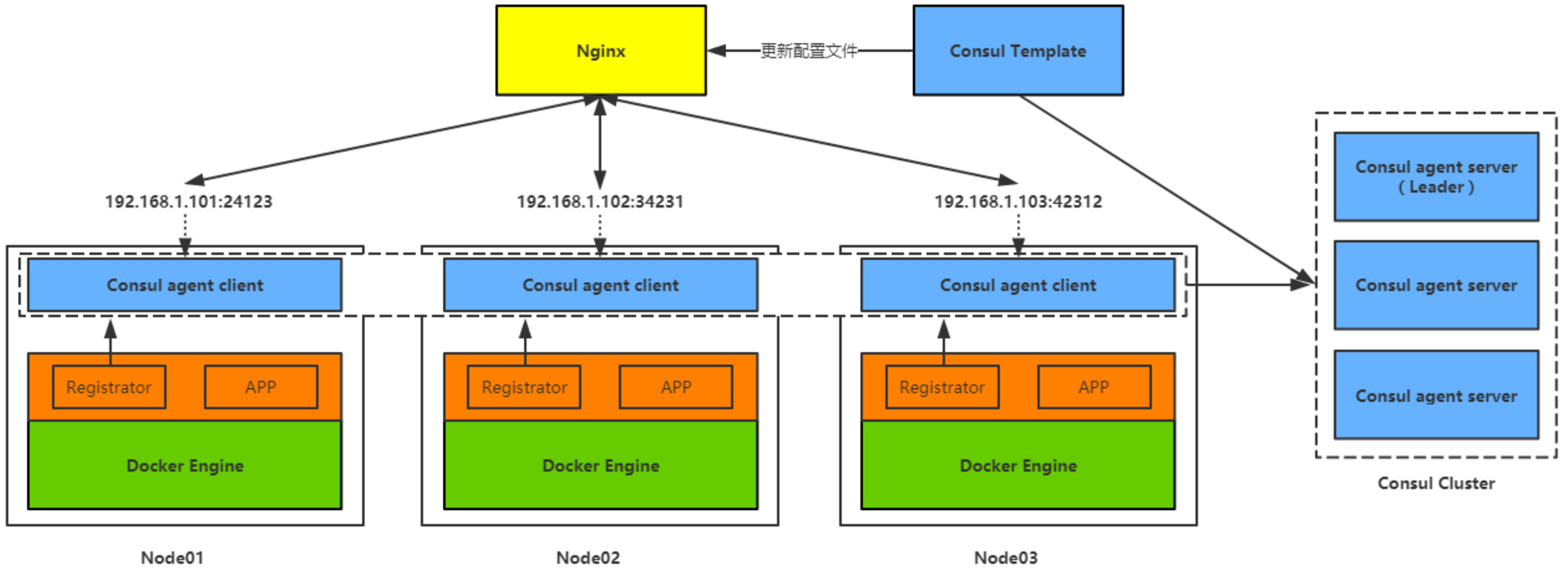
Docker+Registrator+Consul实现容器服务自动加入Nginx集群

consul-template：一个守护程序，用于实时查询consul集群数据，并更新文件系统上的任意数量的指定模板，生成配置文件，更新完成后可以选择运行任何Shell命令。

gliderlabs/registrator：检查容器运行状态自动注册和注销Docker容器的服务到服务配置中心。目前支持Consul、etcd和SkyDNS2。

https://github.com/hashicorp/consul-template

https://releases.hashicorp.com/consul-template/0.19.3/consul-template_0.19.3_linux_amd64.zip



案例三 容器服务自动注册与发现

Docker+Registrator+Consul实现容器服务自动加入Nginx集群

Docker主机启动注册器：

```
# docker run -d \
--name=registrator \
--net=host \
-v /var/run/docker.sock:/tmp/docker.sock \
--restart=always \
gliderlabs/registrator:latest \
-ip=192.168.0.212 \
consul://192.168.0.211:8500
```

案例三 容器服务自动注册与发现

Docker+Registrator+Consul实现容器服务自动加入Nginx集群

Nginx负载均衡节点：

```
# vi nginx.ctmpl
upstream http_backend {
    ip_hash;
    {{range service "nginx"}}
    server {{ .Address }}:{{ .Port }};
    {{ end }}
}
server {
    listen 80;
    server_name localhost;
    location / {
         proxy_pass http://http_backend;
    }
}

# consul-template \
-consul-addr 192.168.0.211:8500 \
-template "./nginx.ctmpl:/usr/local/nginx/conf/nginx.conf:/usr/local/nginx/sbin/nginx -s reload" \
-log-level=info
```

谢谢