目的

为我的 SOA，商业机器，我的产品/服务，提供架构支持。

第一步：实现微服务架构。

为我的 SOA，商业机器，我的产品/服务，提供架构支持。

第二步：在架构上，实现我的产品、服务。

一是产品服务要分层，相互调用，实现 SOA。

二是产品服务要自动化部署、发布，实现自动化管理。

第三步，通过网关，对外暴露我的产品和服务。

比如说通过手机就能访问到我的产品和服务。

而对外暴露的产品服务实现价值变现。

通过我自己的域名。

整体框架

多数据源	
数据访问	Mybatis，Druid，Mysql
服务注册/调用 20190107-0226 服务发现 服务调用	Consul+Template+Redis+Openresty DiscoveryClient RestTemplate
API 文档管理	Swagger2

健康检查	Actuator
缓存	Redis，Guava
声明式调用 Feign	
负载均衡（客户端）	Ribbon
服务熔断	Hystrix
国际化	
身份认证	
统一异常	
参数校验	
日志	ELK
配置中心	Spring Cloud Config
服务链路追踪	Sleth，Zipkin
跨域请求	
消息队列	Kafka
路由网关	Zuul
Security	
OAuth2	
容器云管理	Docker，K8S
自动化部署	Jenkins，GitLab

实现步骤

1、 框架工程

代码结构

迭代，

如上表格，选择实现顺序

2、 周边服务器

迭代

3、 流水线

代码仓库，

自动化部署、发布

最后，得到的是一套微服务框架，周边服务，和自动化流水线。

Mysql 服务器

主机 104/虚机 192.168.56.101 data1

Ubuntu 版本 ，端口 3306

root / 123

Sudo service mysql stop

Sudo service mysql start

Sudo service mysql restart

登陆 MySQL 数据库 mysql -u root -p

其中，-u 表示登陆的用户名，-p 表示登陆的密码

查看当前数据库 show databases;

使用某个数据库 use [databaseName];

查看当前数据库中的所有表 show tables;

虚机 101 启动时自启动。

桌面工具：Navicat Premium 12

主机 108/虚机 192.168.56.101 data1

Windows 启动 mysql

管理员身份启动 cmd

>net start mysql

>net stop mysql

安装路径：E:\mysql-8.0.13-winx64\bin

主机 101

Windows 启动 mysql

安装路径：E:\mysql-8.0.32-winx64\bin

安装过程，查看错误日志：:\mysql-8.0.32-winx64\data\DESKTOP-IL16HQJ.err

> mysql –version 查看版本 5.7.40

Root 用户：root

> mysqld -remove 删除实例，需要删除 data 文件夹内容

> mysqld -install 安装服务

> mysqld --initialize-insecure 初始化 data 文件夹

> net start mysql 启动实例，**注意 my.ini 的更新需要重启实例才能生效**

> mysql -u root -p 登录

mysql> use mysql; 选择数据库

mysql> alter user 'root'@'localhost' identified by 'root';

mysql> flush privileges; 刷新权限

https://blog.csdn.net/The_Ink/article/details/108034403?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-108034403-blog-124368620.pc_relevant_landingrelevant&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-108034403-blog-124368620.pc_relevant_landingrelevant&utm_relevant_index=1

https://blog.csdn.net/Z__learning/article/details/124368620

MySQL 压缩包方式安装，傻瓜式教学_mysql 压缩包安装教程_qiuchangyong 的博客-CSDN 博客

postgreSQL

安装路径：G:\Program Files\PostgreSQL\14\bin

superuser 密码：admin

master pw for pgAdmin: admin

user postgres: admin

启动：win + R，services.msc，找到 postgreSql，右键启动

或者 cmd net start postgreSql-10

版本：postgres (PostgreSQL) 14.1

注意：开机没有自启动

postgres -V

Jira

Jira 安装及配置----最详细的教程（测试木头人）_jira 安装配置教程_测试木头人的博客-CSDN 博客

Supported platforms | Administering Jira applications Data Center and Server 8.4 | Atlassian Documentation

硝烟中的 JIRA 和敏捷开发（三）-JIRA 证书 Licenses 更新延期_aezhaocq_的博客-CSDN 博客

JIRA REST API 使用说明 (devpod.cn)

版本：8.4.1

安装路径：G:\Program Files\Atlassian\jira

BGVD-AOXA-651Q-711C

quickersandcsu2023@126.com

WORK 看板 - Agile Board - Jira

Confluence

下载安装 confluence，win10 版本

安装路径：G:\Program Files\Atlassian\Confluence

数据存放路径：G:\Program Files\Atlassian\Application Data\Confluence

启动> start confluence

停止> stop confluence

windows10 系统 搭建 confluence6.3.1 详解_confluence windows_伯先知的博客-CSDN 博客

Database Setup For MySQL | Confluence Data Center and Server 8.0 | Atlassian Documentation

用户名 admin/admin，13794476201@163.com

主页面 - Confluence

Consul

安装：主机 104/虚机 101，data1

安装路径：/usr/local/bin/consul

& sudo consul agent -server -bootstrap-expect 1 -data-dir /tmp/consul -ui -bind=192.168.1.101 -client=0.0.0.0

验证：192.168.1.101:8500

虚机 101 启动时自启动，见 etc/rc.local。

可能连接不上，需要等 10 分钟。

可以用 netstat，ifconfig 查看。

MSB-rest 服务

监控 **Consul**，定时调用 Consul 的服务查询接口（HTTP API），更新 redis 数据。

问题：这种方案会有延时，当服务信息发生改变时，更新不及时。

方案优化：使用 **Consul-Template** 监听服务变化，修改 Nginx 的配置文件，执行 reload。

问题：执行 reload 会导致 Nginx 的性能降低，特别是配置文件频繁更改的场景。

方案优化：监听 Consul 的事件通知，获取服务信息，使用 **Redis** 持久化服务注册信息。通过 **OpenResty** 动态加载注册信息。

consul-template -consul-addr 192.168.56.101:8500 -template "tmpltest.ctmpl:luajit ./h.lua" -once

OpenResty

安装：主机 104/虚机 101，data1

路径：/home/www

nginx -p `pwd`/ -c conf/nginx.conf

验证 http://www.txw.com:9000/redirect

日志路径是 home/www/logs/error.log。

虚机 101 启动时自启动。

Nginx

安装：主机 108/虚机 192.168.56.100，master

Redis 服务器

主机 104/虚机 192.168.56.101 data1

安装：主机 104/虚机 101，data1

redis-server -v

修改密码 requirepass 123（重启失效）

$ redis-cli shutdown

$ sudo redis-server /etc/redis/redis.conf

查看 redis 进程

$ ps -ef | grep redis

$ redis-cli 查看终端

> Keys * 查看所有键

开机服务自启动

连接工具：RedisDesktopManager

如果远程无法连接，尝试 ping 192.168.1.104 打通路由。

Redis 集群

安装：主机 108/虚机 102，Kafka

redis-cli --cluster create 192.168.1.102:7001 192.168.1.102:7002 192.168.1.102:7003

192.168.1.102:7004 192.168.1.102:7005 192.168.1.102:7006

关闭集群脚本 src/shutdown.sh

./shutdown.sh

Docker

安装：主机 108/虚机 192.168.56.100

主机 108/虚机 101，data1

安装 JDK

Tar 包路径：/mnt/software/jdk8

安装路径：/opt/

虚机 101 启动时自启动。

Docker Registry

安装：主机 108/虚机 192.168.56.100 容器安装

sudo docker run -d -p 5000:5000 -v /opt/registry:/var/lib/registry registry

验证：http://192.168.56.100:5000/v2/

http:// 192.168.56.100:5000/v2/txw/java/tags/list

http:// 192.168.56.100:5000/v2/_catalog

Docker 的本地镜像库

Gitlab

安装：主机 108/虚机 192.168.56.100 master 容器安装

sudo docker run -d -h gitlab.txw.com -p 22:22 -p 8929:8929 -v /etc/gitlab/:/etc/gitlab/ -v /var/log/gitlab/:/var/log/gitlab/ -v /var/opt/gitlab/:/var/opt/gitlab/ --name gitlab gitlab/gitlab-ce

访问地址：

http://192.168.1.100:8929

http://gitlab.txw.com:8929/txw-framework

管理员密码 root/12345678

因为需要映射 22 端口，所以需要把 100 宿主机上的 sshd 服务关闭。

不然本机推送代码到远程仓库 git push 时，会报错。

虚机开机自启动

sudo docker ps -a

sudo docker exec -it CONTAINER_ID /bin/bash

进入容器， exit 退出容器

Docker restart CONTAINER_ID

Jenkins

安装：主机 108/虚机 192.168.56.100 容器安装

验证：http://192.168.56.100:8080

sudo docker run -d -p 8080:8080 -u root -v /var/run/docker.sock:/var/run/docker.sock -v /var/jenkins_home:/var/jenkins_home --link gitlab:gitlab.txw.com --name jenkins jenkins/docker

因为需要从 gitlab 拉取代码，所以增加--link gitlab:gitlab.txw.com。

admin/admin

Git

在 windows 本机安装 Git。

配置文件：C:\Users\admin/.ssh/id_rsa

Kafka

主机 108/虚机 102

安装路径：/usr/local/services/kafka_2.12-2.1.1

$ cd /usr/local/services/kafka_2.12-2.1.1

$ sudo bin/zookeeper-server-start.sh -daemon config/zookeeper.properties

$ sudo bin/kafka-server-start.sh -daemon config/server.properties

bin/kafka-server-stop.sh config/server.properties

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

注：开机后需要手动运行 kafka（以上加粗命令行）。

有问题可以直接看日志，logs/zookeeper.out

测试命令

bin/kafka-console-producer.sh --broker-list 192.168.1.102:9092 --topic test

注：如果没有提前建立 topic test，则会自动创建。

bin/kafka-console-consumer.sh --bootstrap-server 192.168.1.102:9092 --topic test --from-beginning

Kafka Manager

主机 108/虚机 102

安装路径：/usr/local/services/kafka-manager

启动服务

nohup bin/kafka-manager -Dconfig.file=conf/application.conf -Dheep.port=9000 &

注：需先删除 RUNNING_PID 文件

验证：http://192.168.1.102:9000

Add Cluste

ZipKin

主机 104/虚机 101

虚机 101 启动时自启动。

验证：http://192.168.1.101:9411

Elasticsearch

安装：主机 108/虚机 102，kafka

路径：/usr/local/services/elasticsearch/bin

启动：./elasticsearch

```
设置 zipkin：
$ STORAGE_TYPE=elasticsearch ES_HOSTS=http://192.168.1.102:9200 java -jar zipkin.jar
```

验证：http://192.168.1.102:9200/

版本：6.6.2

Elasticsearch Head

安装：主机 108/虚机 102，kafka

路径：cd /usr/local/services/elasticssearch-head-master

启动：npm run start

验证：http://192.168.1.102:9100/

Kibana

安装：主机 108/虚机 102，kafka

目录：/usr/local/services/kibana/bin

启动：./kibana

验证：http://192.168.1.102:5601

手册：https://www.elastic.co/guide/cn/kibana/current/tutorial-discovering.html

Logstash

安装：主机 108/虚机 102，kafka

目录：/usr/local/services/logstash-7.6.1/bin

测试：./logstash -e 'input { stdin { } } output { stdout {} }'

启动： ./logstash -f **logstash.conf**

Maven 服务器

如何搭建？是否需要？是否会影响 Devop？

K8s 集群

安装：主机 108/虚机 110，kube-master，虚机 111，Kube-node

kubectl get pods -n kube-system

root/kubeadm.yaml

swapoff -a

kubeadm reset // 可选

kubeadm config print init-defaults // 显示默认配置文件

kubeadm init --config /root/kubeadm.yaml

kubeadm init --apiserver-advertise-address 192.168.1.110 --pod-network-cidr 10.244.0.0/16

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

mkdir /etc/kubernetes/manifests/my.conf/

cp -r /root/kube-flannel.yml /etc/kubernetes/manifests/my.conf/kube-flannel.yml

kubectl create -f /etc/kubernetes/manifests/my.conf/kube-flannel.yml

验证

Kubectl get cs

Kubectl get nodes

kubectl get pod --all-namespaces

MongoDB

主机 104/虚机 192.168.56.101 data1

安装：主机 104/虚机 101，data1

查看进程 pgrep mongo -a

杀掉进程 pkill mongod

dbpath=/var/db

logpath=/var/log/mongodb/mongodb.log

/etc/mongodb.conf

注：以上路径和文件的权限 sudo chmod 777 *

sudo service mongodb status

sudo service mongodb stop

sudo service mongodb start

sudo service mongodb restart

安装目录/usr/local/mongodb

启动服务 mongod -f /etc/mongodb.conf

关闭服务 mongod --dbpath /var/lib/mongo --logpath /var/log/mongodb/mongod.log

–shutdown

```
>db.shutdownServer()
```

shell 命令模式 mongo

show dbs

use admin

db.createUser({user:'root',pwd:'202007',roles:[{role:'root',db:'admin'}]})

db.auth('root','202007')

show users

服务器&安装服务整理

1、物理机 192.168.1.104，win10，笔记本

1.1）虚拟机 master，192.168.1.100

1.2）虚拟机 data1，192.168.1.101：redis

2、物理机 192.168.1.108，win10，台式机

2.1）虚拟机 master，192.168.1.100：Gitlab，Jenkins，Registry

2.2）虚拟机 kafka，192.168.1.102：kafka，ES，Kibana，Logstash

2.3）虚拟机 kube-master

2.4）虚拟机 kube-slave

开发环境：主机 104/虚机 101

生产环境：主机 108/虚机 100,101,102

演示环境同生产环境

虚机 100（master）：Jenkins，Gitlab，Docker，Docker Registry，Nginx

虚机 101（data1）：Redis，Mysql，Zipkin，Consul，Template，Openresty，Docker

虚机 102（kafka）：Kafka，ES，Kibana

说明：

1、平时使用开发环境，只需要启动虚机 101。

2、如果需要使用持续集成和构建，需要启动主机 108 和虚机 100。

3、如果需要使用 Kafka，需要启动主机 108 和虚机 102。

4、完整的生产环境和演示环境，需要启动主机 108，虚机 100,101 和 102。

并且开启路由器的虚拟 IP 映射和虚机的 NAT 的端口映射。

持续构建&持续部署

1、 Gitlab 新增一个项目。

新增一个项目 Project，比如 helloworld。

Gitlab 生成远程仓库 gitlab.txw.com:root/helloworld.git

2、 开发机 Git 提交代码。

按照 Gitlab 项目页面的提示，提交代码到远程仓库：git push -u origin master

Existing folder

```
cd existing_folder
git init
git remote add origin git@gitlab.txw.com:txw-framework/txw-auth.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

初始化：git init

添加远程仓库: git remote add origin git@gitlab.txw.com:root/ helloworld.git

git remote add origin **git@gitlab.txw.com:txw-framework/txw-admin-server.git**

3、 Jenkins 从 Gitlab 拉取代码。

Jenkins 安装 Git，安装插件。

Jenkins 新建一个 maven 项目。

配置源码管理：Repository URL

http://gitlab.txw.com:8929/root/spring-boot-helloworld.git

Credentials：root/12345678

4、 Jenkins 构建。

Jenkins 安装 maven，安装插件。

执行 Maven 命令 clean package。

构建目录：/var/jenkins_home/workspace/spring-boot-helloword 构建

得到 Jar 包：/target/spring-boot-helloworld-1.0.jar

Jar 包名称默认为{project.artifactId}-{project.version}

5、 Jenkins 打镜像。

Jenkins 安装了 docker-maven-plugin 插件。

Jenkins 容器内安装了 docker。

代码里面有 Dockerfile。

Jenkins 把 Dockerfile 和 Jar 包都拷贝到同一个目录下: /var/jenkins_home/workspace/spring-boot-helloword 构建/target/docker/。

然后执行构建命令。

Dockerfile

From 192.168.56.100:5000/txw/java

ADD spring-boot-helloworld-1.0.jar app.jar

ENV JAVA_HOME opt/jdk

ENV PATH $PATH:$JAVA_HOME/bin

CMD ["java", "-jar", "app.jar"]

得到镜像，名称：192.168.56.100:5000/com.neo/spring-boot-helloworld:1.0

6、Jenkins 推送镜像到 Registry。

Jenkins 设置 docker:push

通过 POM.xml 文件中的设置，推送到镜像库 Pushing 192.168.56.100:5000/com.neo/spring-boot-helloworld:1.0

7、Jenkins 拉取镜像，启动容器。

Jenkins 配置了 SSH 机器（192.168.56.101）。

Jenkins 设置 Send build artifacts oven SSH

通过配置的 SSH 机器(192.168.56.101), 执行 shell 脚本/mnt/script/service_docker_start.sh。

完成容器删除，从 Registry 拉取镜像，并启动容器。