中国平安 PING AN
保险·银行·投资

Weblogic容器化研究

平安产险科技中心
技术架构组
2017年12月

中国平安 PING AN
保险 · 银行 · 投资

目录
Content

一、背景简述

1.1、研究背景：

- 产险大量应用系统运行在weblogic中
- Oracle官方只提供最新版本weblogic容器化的指引
- 急需一套从镜像构建到服务编排的自动化全流程解决方案

1.2、研究目标：

- 实现自动化构建基于weblogic的应用镜像
- 实现现有系统的容器化测试以及EJB服务在容器中的运行测试
- 实现基于容器的服务编排（扩容、缩容、滚动升级）

2.1基于Weblogic 的应用镜像构建概要

1、基础镜像构建：

- 构建weblogic基础镜像并创建domain
- Weblogic镜像优化
- 自定义配置参数化

2、基于基础镜像配置weblogic资源：

- Wlst介绍
- Online模式配置资源及参数化
- Offline模式配置资源及参数化

3、构建基于weblogic的应用镜像：

- Wlst offline部署应用
- 应用配置参数暴露及启动方式

2.2 Weblogic 基础镜像构建

```
# Pull base image
# ---------------
FROM oraclelinux:7

# Environment variables required for this build (do NOT change)
# -------------------------------------------------------------
ENV JAVA_HOME /u01/jdk1.6.0_45

# Setup required packages (unzip), filesystem, and oracle user
# ------------------------------------------------------------
RUN mkdir -p /u01/oracle  && \
    chmod a+xr /u01 && \
    mkdir -p /u01/jdk1.6.0_45

COPY oracle /u01/oracle
COPY jdk1.6.0_45 /u01/jdk1.6.0_45
# Install and configure Oracle JDK
# --------------------------------
WORKDIR /u01

ENV PATH $PATH:/u01/jdk1.6.0_45/bin

CMD [ "/u01/oracle/user_projects/domains/base_domain/startWebLogic.sh" ]
```

2.3 Weblogic 基础镜像构建

1、静默安装方式

- 传统的weblogic安装，首先安装jdk，然后使用java命令运行weblogic安装程序进行安装
- 这里参考平安科技同事的文章：http://www.cnblogs.com/ericnie/p/7866686.html

2、上述方式构建镜像的不足及解决方法

- 不足：基于上述方法构建的镜像体积很大（达到2个多G），这样对网络会造成很大压力
- 解决方法：

a) 因为java应用的运行只需要配置好path与classpath即可运行，基于这样的原理，将上述镜像中的weblogic目录和jdk目录copy到虚拟机，然后直接通过COPY的方式构建镜像，配置环境变量后weblogic服务正常启动，镜像大小减小到1G；

b) 构建目录结构以及dockerfile如下：

```
[root@SZD-L0077282 weblogic1036]# ls
dockerfile  jdk1.6.0_45  oracle
```

dockerfile

2.4 Weblogic自定义配置的参数化

1、java应用的主要自定义参数：

- JVM内存参数：java应用启动都需要定制内存堆大小，gc策略等参数以获取更好的性能
- 自定义classpath：有些应用启动前需要JVM提前加载一些容器级别的自定义配置及jar包
- 自定义的-Dxxx java options：除了上述配置以外，有些应用还需要配置额外的环境参数，通过-Dxxx=yyy的方式加载。

通过修改weblogic的启动脚本startWebLogic.sh，添加自定义参数的环境变量；后续只需要向容器注入环境变量或通过启动脚本初始化环境变量即可，后面详细介绍。

```
#user custom weblogic java options start
if [ "${USER_MEM_ARGS}" != "" ] ; then
  MEM_ARGS="${USER_MEM_ARGS}"
fi

if [ "${USER_CLASSPATH}" != "" ] ; then
  CLASSPATH="${USER_CLASSPATH}:${CLASSPATH}"
fi

if [ "${USER_JAVA_OPTIONS}" != "" ] ; then
  JAVA_OPTIONS="${USER_JAVA_OPTIONS} ${JAVA_OPTIONS}"
fi
```

2.5 Weblogic wlst工具介绍

1、WLST： WebLogic Scripting Tool

- 提供除web界面方式以外的另外一种管理配置weblogic domain及server的方式
- WLST 脚本环境基于 Java 脚本解释器 Jython
- WLST 提供 online与offline两种配置方式（有些配置只能用online）：online模式是通过连接到运行中的weblogic管理服务进行配置，offline模式是在weblogic服务停止状态下读取domain的配置，通过命令及脚本方式直接修改某些参数

2、WLST在容器镜像构建过程中的用法

- 容器镜像构建过程中主要使用offline模式进行配置
- 必须使用online方式配置的资源（比如定制的AuthenticationProvider）通过如下方式进行定制化：
  - 基于基础镜像启动使用主机网络的容器并加装需要的资源到容器中
  - 启动weblogic，通过web控制台配置资源，保存后重启weblogic（有些资源重启后才生效）
  - 资源生效后，停止weblogic服务，修改domain的config.xml 将环境相关的配置使用占位符替换，后续部署应用的时候根据不同环境将占位符替换成不同值。

二、镜像构建研究

2.6 Online 模式配置weblogic 定制 AuthenticationProvider

1、启动容器：

```
docker run --name weblogic1036-jdk16045 --net=host -v
/opt/zhaoyanbin442/weblogic1036_PASProvider:/u01/data -i
hub.yun.paic.com.cn/zhaoyanbin442/weblogic1036-jdk16045:1.0.0 bash
```

2、进入控制台配置资源：

参数化配置项：

```
<realm>
  <sec:authentication-provider xmlns:ext="http://xmlns.oracle.com/weblogi
    <sec:name>PASAuthenticatorSix</sec:name>
    <sec:control-flag>OPTIONAL</sec:control-flag>
    <ext:mapping-instance>ALL</ext:mapping-instance>
    <ext:enable-get-user-roles>true</ext:enable-get-user-roles>
    <ext:app-series>PA003</ext:app-series>
    <ext:ldap-server>${ldap-server}</ext:ldap-server>
    <ext:security-principal>cn=realm,cn=Users,dc=paic,dc=com,dc=cn</ext:s
    <ext:schema-property>user.filter=(&amp;amp;(uid=%u)(pafaisactive=true
    <ext:schema-property>user.dn=ou=people,o=paic.com.cn</ext:schema-prop
    <ext:schema-property>membership.filter=(&amp;amp;(uniquemember=%M)(ob
    <ext:schema-property>group.filter=(&amp;amp;(cn=%g)(objectclass=group
    <ext:schema-property>group.dn=ou=services,o=paic.com.cn</ext:schema-p
    <ext:ldap-port>${ldap-port}</ext:ldap-port>
    <ext:security-credential-encrypted>${credential-value}</ext:security-
  </sec:authentication-provider>
```

Docker commit 方式基于容器创建镜像：

```
Docker commit  xxxx    yyyyy:tag
```

2.7 Offline 模式配置weblogic jdbc 数据源

1、编写WLST python脚本：

- Oracle 官方docker相关资源：
  https://github.com/oracle/docker/tree/master/OracleWebLogic
- 这里的jdbc 数据源创建脚本，参见右侧附件。

ds-deploy.py

2、通过properties属性文件进行参数化配置：

```
#jdbc resource
dsname=centerDS
dsdbname=d3epcis;create=true
dsjndiname=jdbc/iCore_cache_center/DefaultDS
dsdriver=oracle.jdbc.OracleDriver
dsurl=jdbc:oracle:thin:@10.20.129.46:1526:d3epcis
dsusername=icacheopr
dspassword=paic1234
dstestquery=SQL SELECT 1 FROM DUAL
dsmaxcapacity=1
```

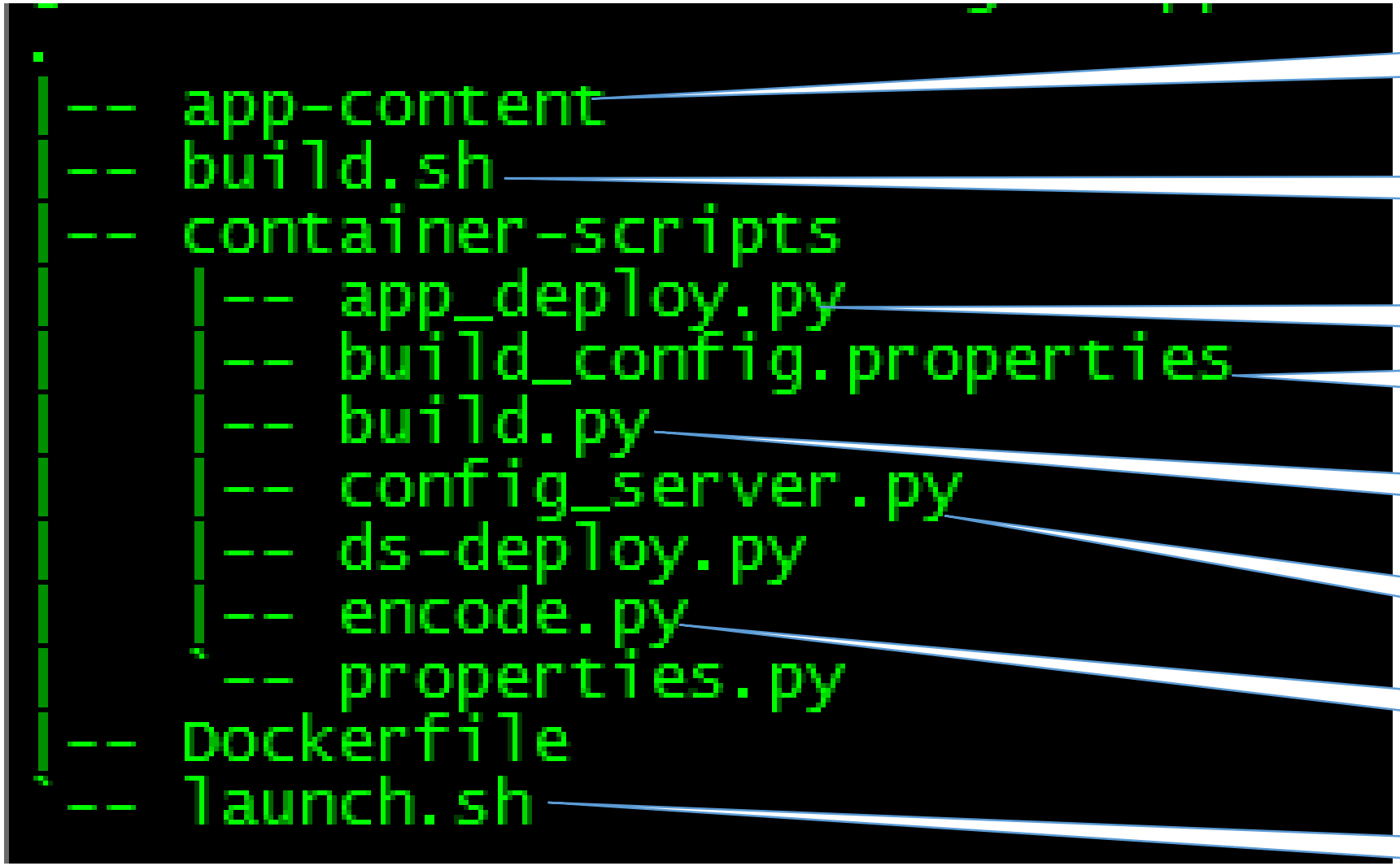
3、dockerfile中通过RUN 运行wlst工具创建数据源：

采坑：

1. Dockerfile中配置环境变量，避免命令找不到： ENV PATH /u01/oracle/wlserver_10.3/common/bin:$PATH
2. Wlst脚本中添加参数，解决执行过程异常慢的问题 JVM_ARGS="-Djava.security.egd=file:///dev/urandom

RUN wlst -loadProperties /u01/oracle/datasource.properties /u01/oracle/ds-deploy.py

2.8 基于weblogic的应用镜像构建工作空间目录介绍



应用资源目录

镜像构建脚本

应用部署脚本

部署配置文件

应用部署主程序

配置监听端口

Weblogic加密工具

容器启动脚本

脚本已上传到内git服务：http://10.11.112.66/docker-container-scripts/weblogic-container-scripts.git

2.9 基于weblogic的应用镜像构建

1、应用部署的offline wlst脚本：

```
cd('/')
app = create(app_name, 'AppDeployment')
app.setSourcePath(appdir +  app_pkg_file)
app.setStagingMode('nostage')
assign('AppDeployment', app_name, 'Target', admin_server_name)
```


app_deploy.py


build.py

2、以icore-cache为例的全流程应用部署介绍：

- 单向资源配置或应用部署模块化
- 在配置文件按需设置构建开关，通过build.py进行构建调度


build_config.properties

3、根据应用需求定制启动脚本并暴露运行时参数给容器调度平台：

- Icore-cache 需要的配置参数为：CACHE_LOG、INSTANCE_NAME、CACHE_CONFIG
- 在启动脚本launch.sh 中配置上述参数与weblogic基础镜像定制参数关系
- 容器拉起的时候只需要注入相应的环境变量即可实现不同的运行状态


launch.sh

2.10 容器镜像的分层与共享

在空白机器上单独拉取weblogic基础镜像，耗时49秒

```
[root@SZD-L0079575 ~]# date && docker pull hub.yun.paic.com.cn/zhaoyanbin442/weblogic1036-jdk16045-auth:1.0.0 && date
2017年 12月 16日 星期六 14:25:13 CST
Trying to pull repository hub.yun.paic.com.cn/zhaoyanbin442/weblogic1036-jdk16045-auth ...
1.0.0: Pulling from hub.yun.paic.com.cn/zhaoyanbin442/weblogic1036-jdk16045-auth
14964297f38e: Pull complete
5a90add62b36: Pull complete
c6f650c283e2: Pull complete
d675459288f4: Pull complete
bdb0a3e111b1: Pull complete
Digest: sha256:b619fa7d976e9989b115debcf69bd7f7cfc132d141b6596b89755b4f0ea845fd
Status: Downloaded newer image for hub.yun.paic.com.cn/zhaoyanbin442/weblogic1036-jdk16045-auth:1.0.0
2017年 12月 16日 星期六 14:26:02 CST
```

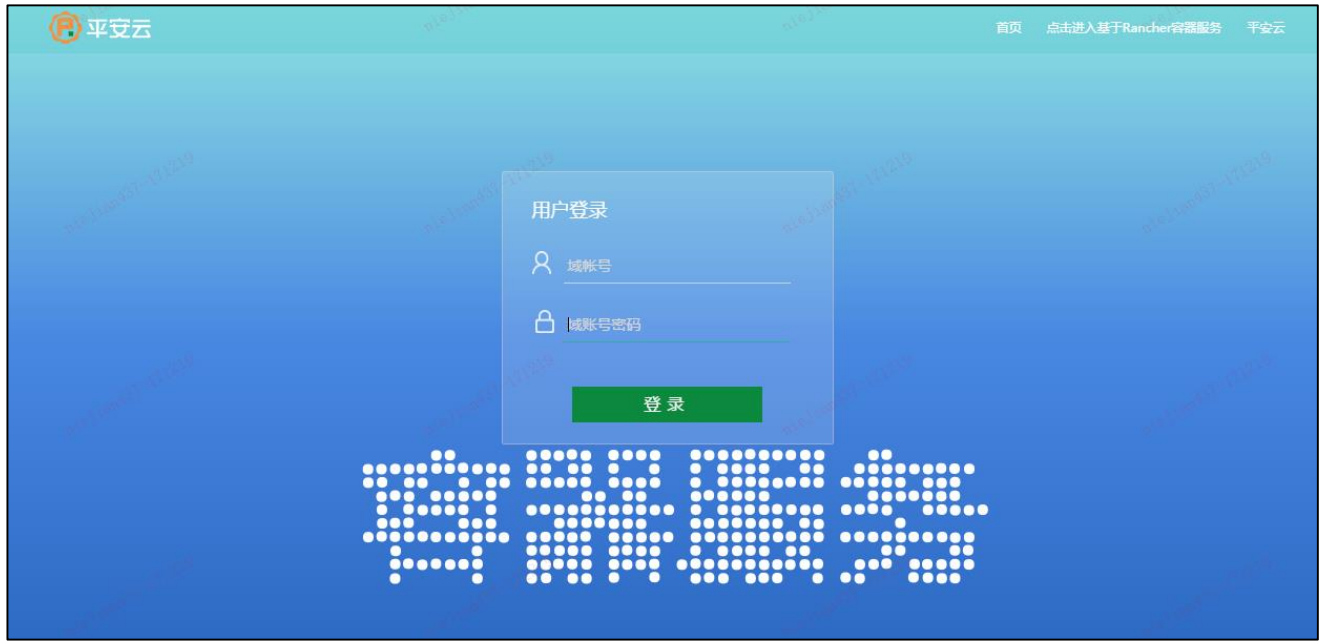
Weblogic 基础镜像拉取后再拉取应用镜像，耗时3秒

```
[root@SZD-L0079575 ~]# date && docker pull hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.0 && date
2017年 12月 16日 星期六 14:26:32 CST
Trying to pull repository hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache ...
1.1.0: Pulling from hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache
14964297f38e: Already exists
5a90add62b36: Already exists
c6f650c283e2: Already exists
d675459288f4: Already exists
bdb0a3e111b1: Already exists
98904c6bec1a: Pull complete
16438e078f66: Pull complete
9111414e2067: Pull complete
6e67a33fdb47: Pull complete
46df2dc8cf59: Pull complete
Digest: sha256:3937d2de0a43540cd55c0ee3b8cb056e840b3b7cabb1da45d83450817e88c5dc
Status: Downloaded newer image for hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.0
2017年 12月 16日 星期六 14:26:35 CST
```

提示：weblogic 应用可以通过对基础镜像预先作，预拉取或者预推送的方式提高应用部署的敏捷度，降低网络流量

三、服务编排

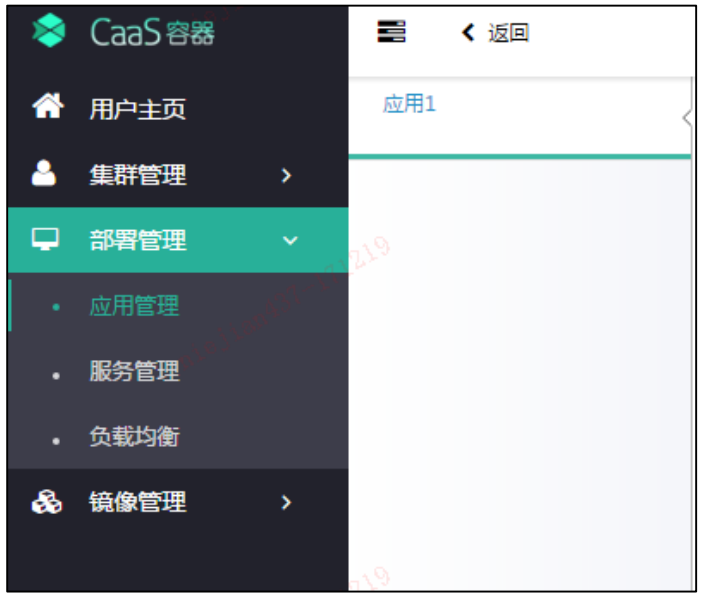
3.1 以icore-cache为例介绍将应用镜像部署为服务（一）



1、登录云平台CaaS容器服务系统（http://caas.paic.com.cn)

2、创建集群，加入节点(略)

3 、基于应用镜像创建icore-cache服务



三、服务编排

3.2 以icore-cache为例介绍将应用镜像部署为服务（二）

服务基本配置

服务名称 weblogic-icore-cache

资源组 基础架构组短期研究组

所属应用 tomcat

所属集群 容器测试

创建icore-cache服务

数据卷

本地磁盘 weblogic-data /app/mysqldata/weblogic

挂载外置日志目录到容器

添加

三、服务编排

3.3 以icore-cache为例介绍将应用镜像部署为服务（三）

容器配置

容器名称 weblogic-icore-cache *

镜像 选择镜像 zhaoyanbin442/weblogic-icore-cache 更多选项>>

1.1.0

数据卷 weblogic-data /u01/oracle/cache_log

添加

环境变量

变量 INSTANCE_NAME 值 icore-cache-docker-a

变量 CACHE_CONFIG 值 /u01/oracle/application/cach

变量 CACHE_LOG 值 /u01/oracle/cache_log

添加

选择之前构建的应用镜像

配置数据卷挂载到容器目录

配置运行时环境变量

三、服务编排

3.4 以icore-cache为例介绍将应用镜像部署为服务（四）

访问配置

实例数量 - 1 +

1 Ant B

访问模式 集群外访问 *

容器不直接暴露端口,将服务端口映射到集群节点上大于30000的随机端口,通过【任意节点IP】+【映射端口】进行访问.

服务端口*	容器端口	协议
7001	7001	TCP

增加端口

配置服务网络访问

服务信息

基础信息

服务名 weblogic-icore-cache	服务状态 创建完成	所属应用 tomcat
所属集群 容器测试	区域 SZD	创建人 zhaoyanbin442
创建时间 2017-12-15 17:44:57	访问模式 集群外访问	服务IP 172.254.180.248

启动了一个实例的服务

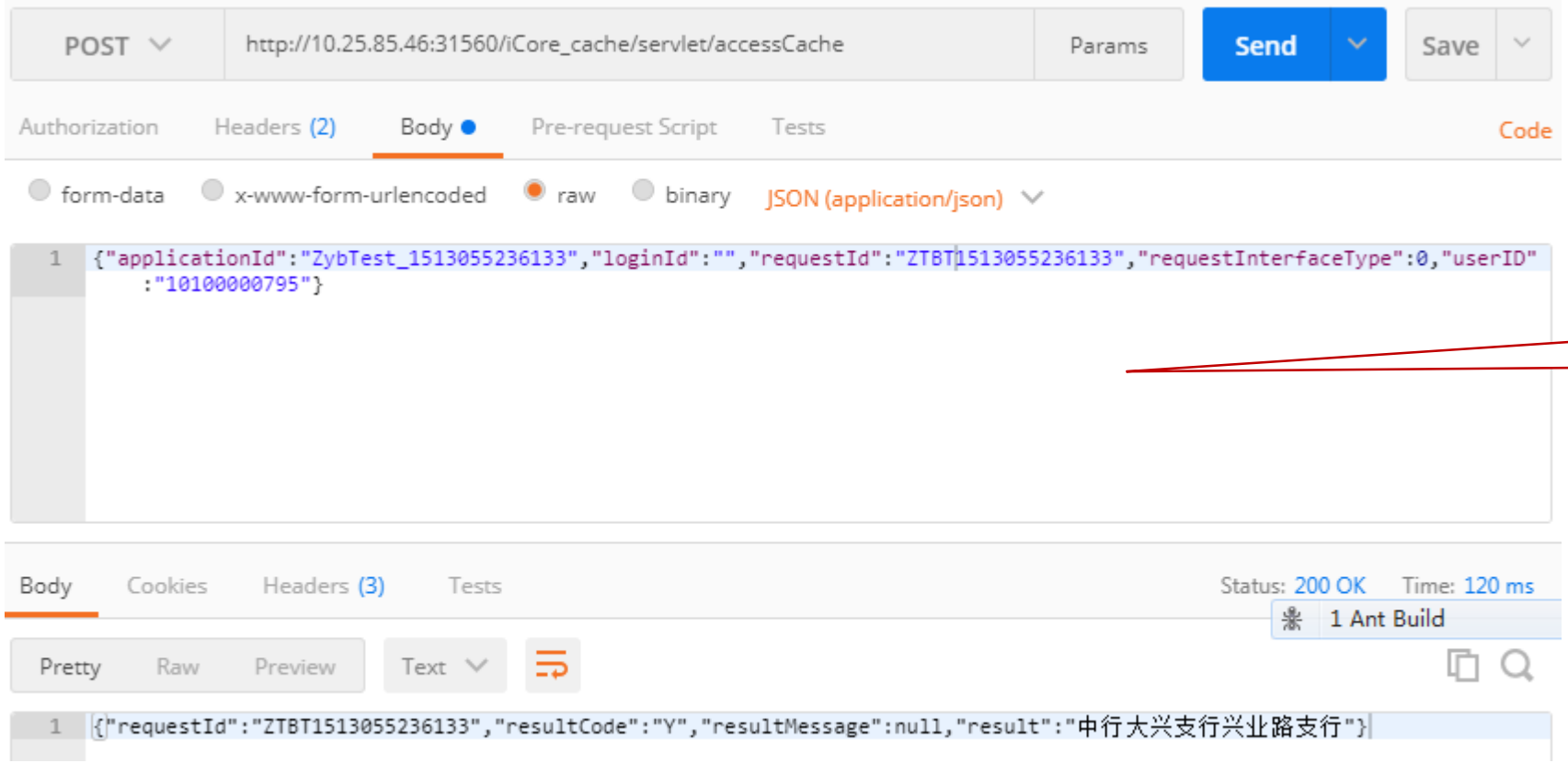
容器组 | 事件 | 数据卷 | 端口

负载均衡

容器组名称	运行状态	ip	宿主机IP
weblogic-icore-cache-2788741733-m...	运行中	172.1.61.2	10.25.85.46

三、服务编排

3.5 以icore-cache为例介绍将应用镜像部署为服务（五）



通过nodeport方式测试成功

3.5 基于容器的EJB调用方式研究

```java
@Stateless(mappedName = "HelloWorldBean")
@Remote({ HelloWorld.class })
public class HelloWorldBean implements HelloWorld {
    public String sayHello(String name) {
        return name + " say hello.";
    }
}
```

```java
InitialContext ic = new InitialContext(prop);
long start = System.currentTimeMillis();
HelloWorld h = (HelloWorld) ic.lookup("HelloWorldBean#com.zyb.ejb.service.HelloWorld");
String rrr = h.sayHello(holleStr);
long end  = System.currentTimeMillis();
rrr = rrr+":"+init+":"+start+":"+end+":init-start:"+(start-init)+":start-end:"+(end-start)+":init-end:"+(end-init);
OutputStream outputStream = response.getOutputStream();
response.setHeader("content-type", "text/html;charset=UTF-8");
byte[] dataByteArr = rrr.getBytes("UTF-8");
outputStream.write(dataByteArr);
outputStream.close();
```

EJB首次调用，初始化很耗时

节点	pod	访问方式	次数	初始化时间(毫秒)	调用时间	总时间	
k8s外部署容器主机网络访问pod		pod ip	第一次	15076	36	15112	
			第二次	5	3	8	
		service ip	第一次	15023		15025	
			第二次	5	4	9	
同节点	同pod	localhost	第一次	105	111	216	
			第二次	7	6	13	
同节点	跨pod	pod ip	第一次	10122	106	10228	
			第二次	9	5	14	
同节点	跨pod	service ip	第一次	20032	4	20036	
			第二次	6	3	9	
同节点	跨pod	service 域名	第一次	5	4	9	上面的测试已经完成建链

初步结论：

K8S内部可以通过POD内、POD间POD ip地址、POD间service ip及域名方式访问；鉴于POD ip的不确定性，建议使用service的方式进行调用（支持负载均衡）

3.6 基于K8S容器服务的编排—扩容缩容

1、扩容：kubectl -s 10.25.65.209:8080 --namespace=szd-0879950f scale --replicas=2 deploy/weblogic-icore-cache

```
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get pod -o wide|grep weblogic
weblogic-icore-cache-2788741733-m89w3   1/1       Running   0          17h       172.1.61.2   10.25.85.46
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f scale --replicas=2 deploy/weblogic-icore-cache
deployment "weblogic-icore-cache" scaled
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get pod -o wide|grep weblogic
weblogic-icore-cache-2788741733-m89w3   1/1       Running   0          17h       172.1.61.2   10.25.85.46
weblogic-icore-cache-2788741733-tkwd9   1/1       Running   0          0s        172.1.58.2   10.25.85.49
```

容器组名称	运行状态	ip	宿主机IP
weblogic-icore-cache-2788741733-m...	运行中	172.1.61.2	10.25.85.46
weblogic-icore-cache-2788741733-tk...	运行中	172.1.58.2	10.25.85.49

2、缩容：kubectl -s 10.25.65.209:8080 --namespace=szd-0879950f scale --replicas=1 deploy/weblogic-icore-cache

```
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get pod -o wide|grep weblogic
weblogic-icore-cache-2788741733-m89w3   1/1       Running   0          17h       172.1.61.2   10.25.85.46
weblogic-icore-cache-2788741733-tkwd9   1/1       Running   0          1m        172.1.58.2   10.25.85.49
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f scale --replicas=1 deploy/weblogic-icore-cache
deployment "weblogic-icore-cache" scaled
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get pod -o wide|grep weblogic
weblogic-icore-cache-2788741733-m89w3   1/1       Running       0          17h       172.1.61.2   10.25.85.46
weblogic-icore-cache-2788741733-tkwd9   0/1       Terminating   0          1m        <none>       10.25.85.49
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get pod -o wide|grep weblogic
weblogic-icore-cache-2788741733-m89w3   1/1       Running       0          17h       172.1.61.2   10.25.85.46
weblogic-icore-cache-2788741733-tkwd9   0/1       Terminating   0          1m        <none>       10.25.85.49
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get pod -o wide|grep weblogic
weblogic-icore-cache-2788741733-m89w3   1/1       Running   0          17h       172.1.61.2   10.25.85.46
```

容器组名称	运行状态	ip	宿主机IP
weblogic-icore-cache-2788741733-m...	运行中	172.1.61.2	10.25.85.46

提示：

1. 除了kubectl 命令外直接curl k8s api也能实现同样功能
2. 可以通过定制策略或者结合第三方监控实现自动伸缩

3.7 基于K8S容器服务的编排—滚动升级（一）

创建新版本镜像：

sh build.sh

docker tag weblogic-icore-cache hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.1

docker push hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.1

```
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get deployment -o wide
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE       CONTAINER(S)           I
MAGE(S)                                                SELECTOR
icore-emcs-nginx      1         1         1            1           4d        icore-emcs-nginx       h
ub.yun.paic.com.cn/library/nginx:1.13.3                caas_service=icore-emcs-nginx
weblogic-icore-cache  1         1         1            1           1d        weblogic-icore-cache   h
ub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.0    caas_service=weblogic-icore-cache
```

kubectl -s 10.25.65.209:8080 --namespace=szdgt f-0879950f get deployment weblogic-icore-

```
spec:
  replicas: 1
  selector:
    matchLabels:
      caas_service: weblogic-icore-cache
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
```

超过期望的Pod数量的最大个数

升级过程中不可用Pod的最大数量

滚动升级，还支持recreate

3.8 基于K8S容器服务的编排—滚动升级（二）

kubectl -s 10.25.65.209:8080 --namespace=szd-0879950f set image deploy weblogic-icore-cache weblogic-icore-cache=hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.1 --record

```
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f set image deploy weblogi
c-icore-cache weblogic-icore-cache=hub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.1 --reco
rd
deployment "weblogic-icore-cache" image updated
[root@SZD-L0077283 ~]#
[root@SZD-L0077283 ~]#
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get deployment -o wide
NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE       CONTAINER(S)            I
MAGE(S)                                                 SELECTOR
icore-emcs-nginx        1         1         1            1           4d        icore-emcs-nginx        h
ub.yun.paic.com.cn/library/nginx:1.13.3                 caas_service=icore-emcs-nginx
weblogic-icore-cache    1         1         1            1           1d        weblogic-icore-cache    h
ub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.1   caas_service=weblogic-icore-cache
wechatmgt-mysql         1         1         1            1           31d       wechatmgt-mysql         h
```

完成升级，可以通过命令查看版本历史

```
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f rollout history deployme
nt/weblogic-icore-cache
deployments "weblogic-icore-cache"
REVISION        CHANGE-CAUSE
1               <none>
2               kubectl set image deploy weblogic-icore-cache weblogic-icore-cache=hub.yun.paic.com.cn
/zhaoyanbin442/weblogic-icore-cache:1.1.1 --server=10.25.65.209:8080 --namespace=szd-0879950f --record
=true
```

3.9 基于K8S容器服务的编排—滚动升级（三）

可以执行rollout undo做回滚：kubectl -s 10.25.65.209:8080 --namespace=szd-0879950f rollout undo deployment/weblogic-icore-cache --to-revision=1

```
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f rollout undo deployment/
weblogic-icore-cache --to-revision=1
deployment "weblogic-icore-cache" rolled back
[root@SZD-L0077283 ~]#
[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f rollout history deployme
nt/weblogic-icore-cache
deployments "weblogic-icore-cache"
REVISION        CHANGE-CAUSE
2               kubectl set image deploy weblogic-icore-cache weblogic-icore-cache=hub.yun.paic.com.cn
/zhaoyanbin442/weblogic-icore-cache:1.1.1 --server=10.25.65.209:8080 --namespace=szd-0879950f --record
=true
3               <none>

[root@SZD-L0077283 ~]# kubectl  -s 10.25.65.209:8080 --namespace=szd-0879950f get deployment -o wide
NAME                 DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE       CONTAINER(S)            I
MAGE(S)                                                SELECTOR
icore-emcs-nginx     1         1         1            1           4d        icore-emcs-nginx        h
ub.yun.paic.com.cn/library/nginx:1.13.3                caas_service=icore-emcs-nginx
weblogic-icore-cache   1         1         1            1           1d        weblogic-icore-cache    h
ub.yun.paic.com.cn/zhaoyanbin442/weblogic-icore-cache:1.1.0   caas_service=weblogic-icore-cache
```

3.10 容器平台后续的roadmap

- 公有云CaaS服务界面上已经有扩容缩容选项
- 现有的私有云的CaaS容器服务暂时还没有在界面上有选项,后台通过K8S支持
- CaaS计划在2018Q1界面上支持Scale,滚动升级等功能

四、问题与不足

问题	描述
编排	容器平台界面还不支持扩容、缩容；**计划2018Q1支持** 还不支持VM标签以及亲和、反亲和部署
应用升级	容器平台界面还不支持容器服务的滚动升级；**计划2018Q1支持**
安全密钥管理	容器平台界面还不支持K8S Secrets 管理秘钥等参数；**目前平安内部使用cyberark**
应用状态	还不支持有状态应用部署，DaemonSet部署

参考资料：

官方容器资料：https://github.com/oracle/docker/tree/master/OracleWebLogic
官方wlst资料：https://docs.oracle.com/cd/E13222_01/wls/docs90/config_scripting/quick_ref.html
启动卡顿问题：http://www.cnblogs.com/ericnie/p/5186112.html
容器weblogic安装：http://www.cnblogs.com/ericnie/p/7866686.html
Wlst资料：http://middleware123.com/weblogic/docs100/config_scripting/reference.html
控制台wlst脚本录制：http://www.acfun.cn/v/ac3985185

同呼吸、共命运、心连心

中国平安 PING AN

保险 · 银行 · 投资