



acmeair 航班订票系统

基于 Service Stage 实现应用微服务化 及上云案例

文档版本 01

发布日期 2017-07-21

华为技术有限公司



版权所有 © 华为技术有限公司 2017。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<http://www.huawei.com>

客户服务邮箱：support@huawei.com

客户服务电话：4008302118

目录

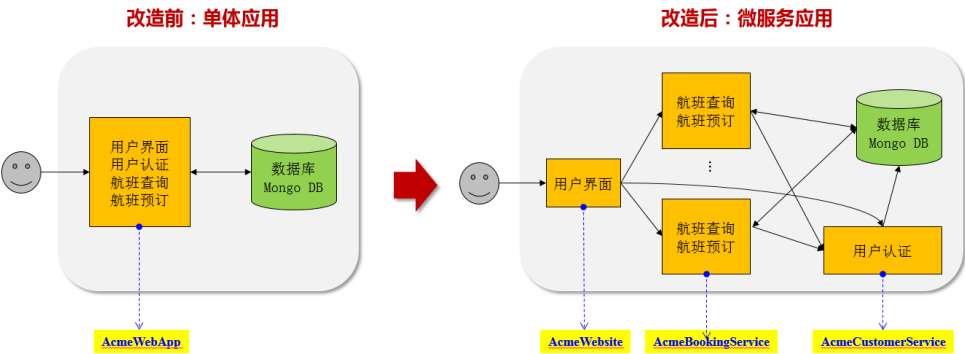
1 背景	1
2 acmeair 拆分微服务方案设计	2
3 acmeair 基于 SDK 的微服务开发	4
4 acmeair 测试	7
4.1 环境准备	8
4.2 导入项目并构建	8
4.2.1 将 acmeair 项目导入 eclipse	8
4.2.2 执行构建命令	9
4.3 启动服务	9
4.3.1 service-center	9
4.3.2 acmeair-booking	10
4.3.3 acmeair-customer	10
4.3.4 acmeair-website	11
4.4 前台访问	12
5 acmeair 发布到 Service Stage	13
5.1 环境准备	14
5.2 创建资源	14
5.3 制作镜像并上传	15
5.4 应用部署	17
5.4.1 手动部署	17
5.4.2 自动部署	18
5.5 应用上线	19
6 acmeair 的运维和治理	20

1 背景

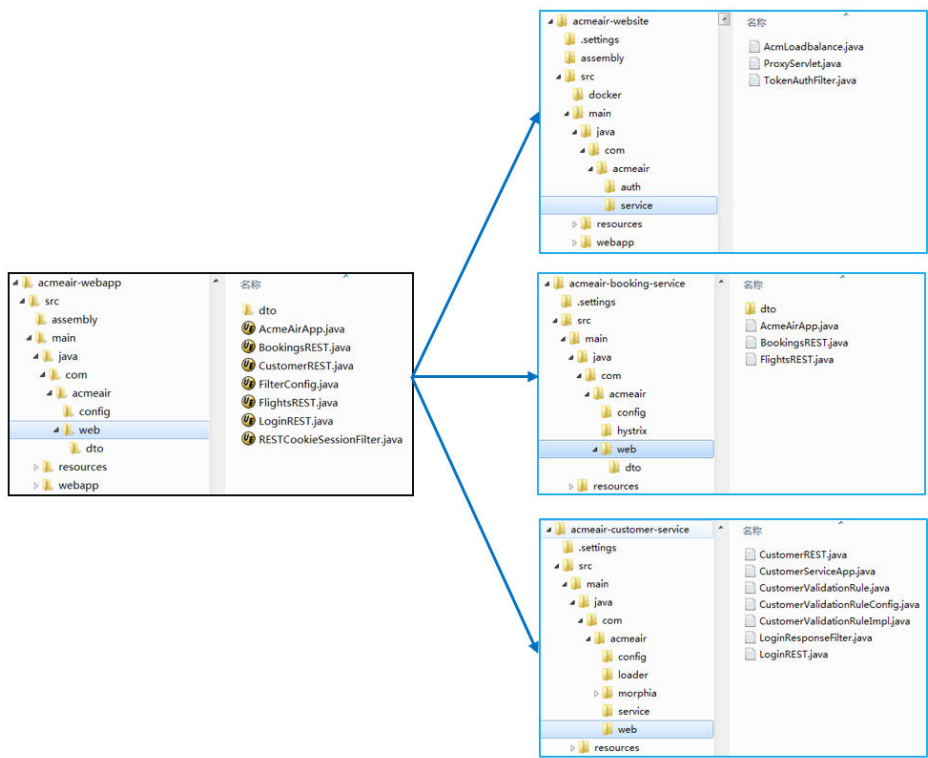
- **Service Stage:** 微服务云应用平台，它是面向企业及开发者的一站式DevOps平台服务，支持基于微服务的应用开发、治理、部署及运维监控的全生命周期管理，并提供大规模容器集群管理及中间件服务等平台能力，帮助用户快速构建云分布式应用。
- 本指导以acmeair demo为例，演示如何使用ServiceComb（华为微服务框架开源版本）实现应用的微服务化，并利用Service Stage云平台实现应用的云化。
- 若想详细了解acmeair微服务化和上云整个流程请阅读本指导所有章节，如果只想查看上云效果，请直接参考第5、第6章节，采用自动部署一键式操作。

2 acmeair 拆分微服务方案设计

步骤1 应用架构设计，划分微服务，识别服务依赖关系。



步骤2 创建微服务，单体应用->微服务应用。

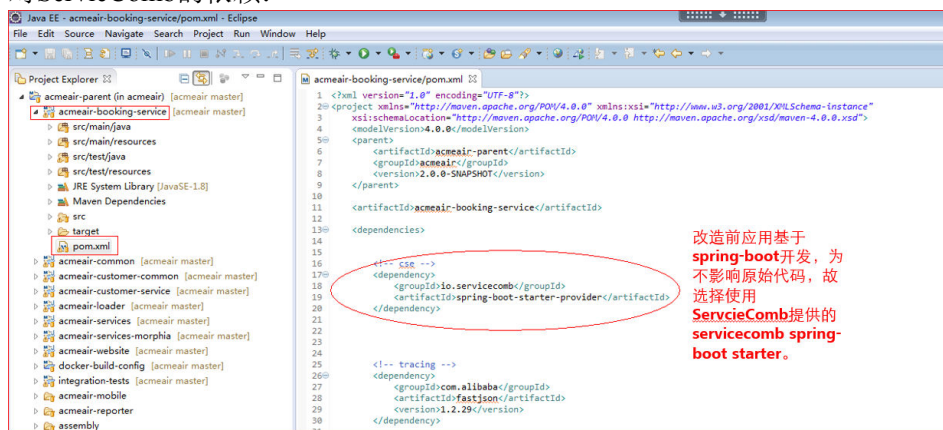


----结束

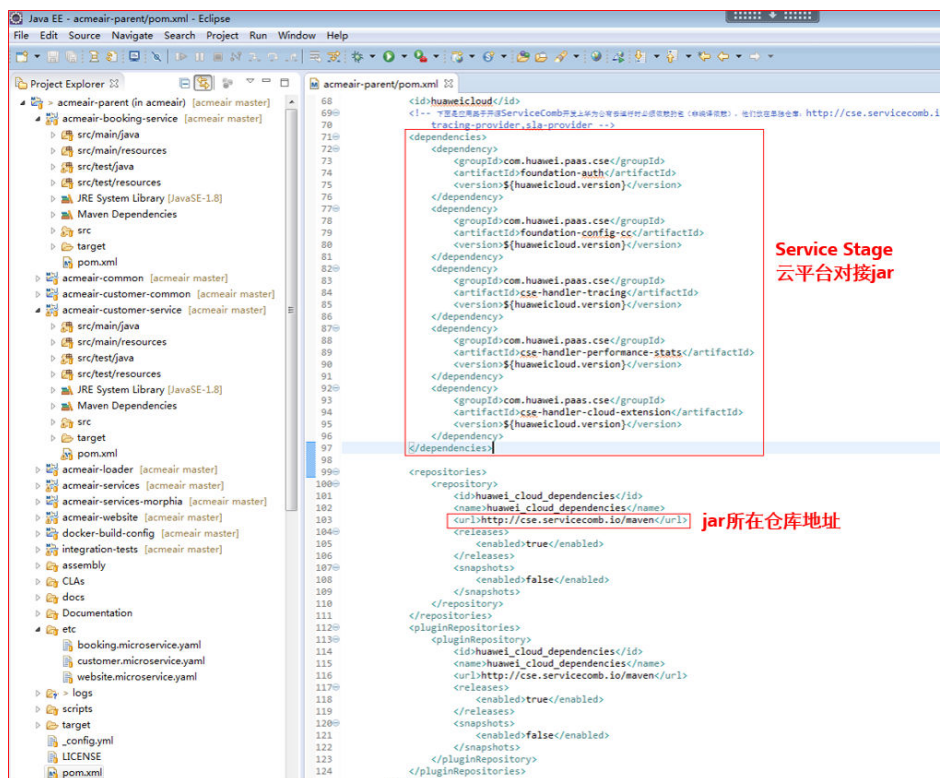
3 acmeair 基于 SDK 的微服务开发

步骤1 POM引入对ServiceComb和服务 Stage的依赖。

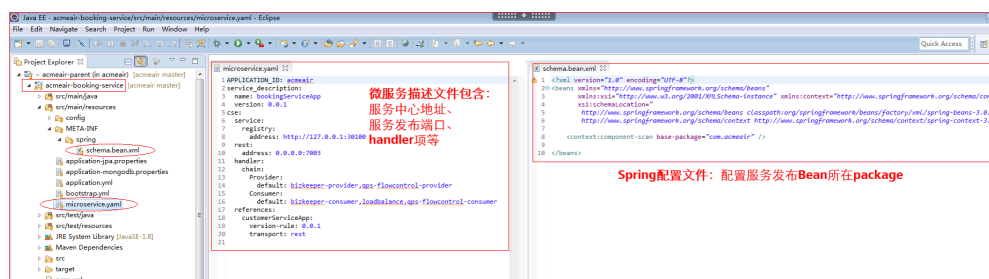
1. 对ServiceComb的依赖:



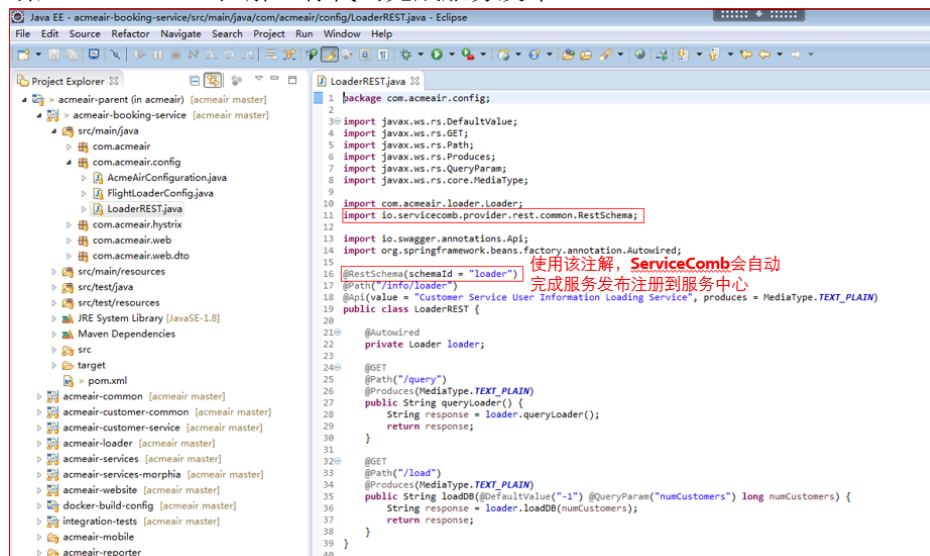
2. 对Service Stage的依赖:



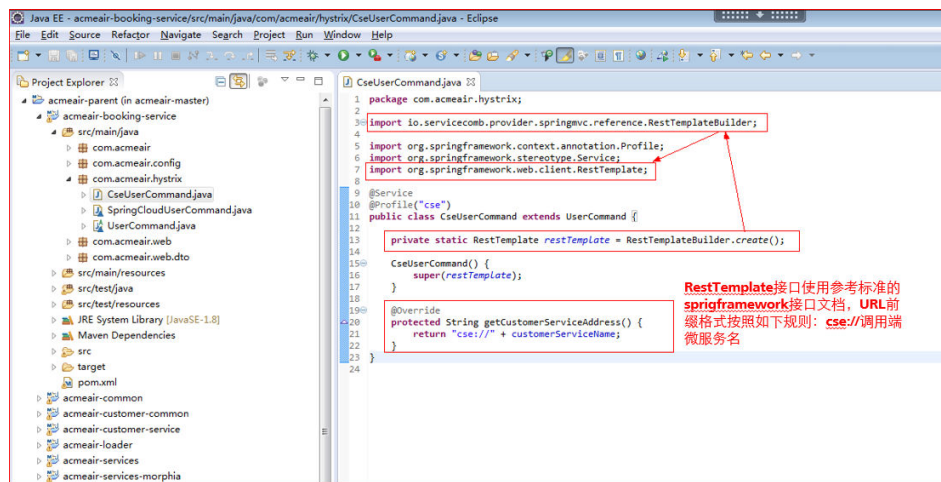
步骤2 增加微服务描述文件和spring配置文件。



步骤3 引入ServiceComb注解一行代码完成服务发布。



步骤4 使用ServiceComb接口简单创建RestTemplate完成服务消费（也支持注解方式进行消费）。



----结束

4 acmeair 测试

本章节介绍在windows 64位环境下进行acmeair的测试。

[4.1 环境准备](#)

[4.2 导入项目并构建](#)

[4.3 启动服务](#)

[4.4 前台访问](#)

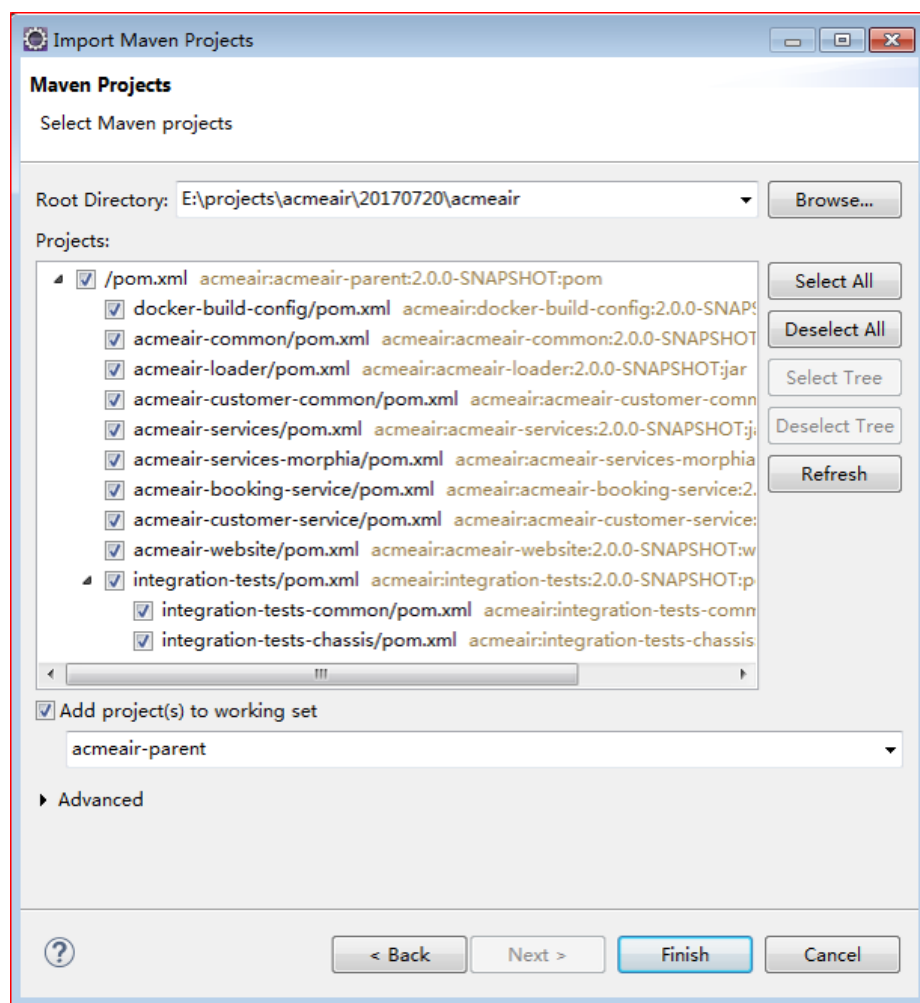
4.1 环境准备

1. JDK 1.8
2. maven 3.x
3. eclipse
4. service-center: <https://github.com/ServiceComb/service-center/releases>
下载service-center-x.x.x-x-windows-amd64.zip，解压到任意目录。
5. mongodb（可选）
6. 下载acmeair源码

4.2 导入项目并构建

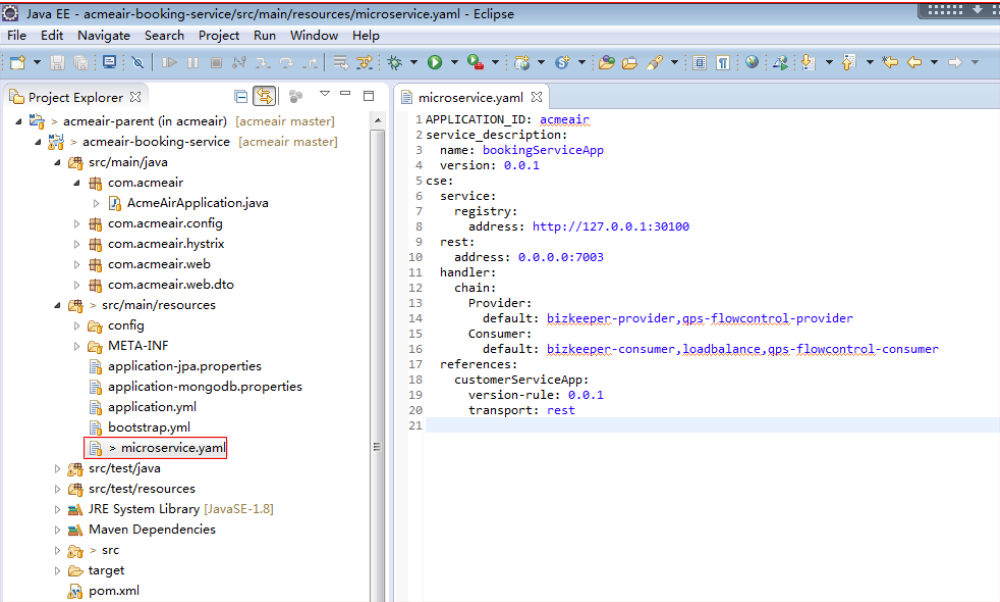
4.2.1 将 acmeair 项目导入 eclipse

Import -> Maven:Existing Maven Projects，选择代码所在的目录，finish导入。

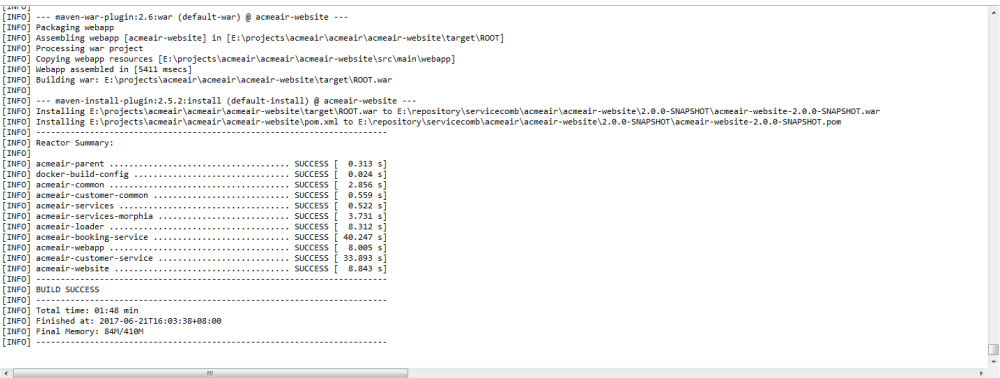


4.2.2 执行构建命令

项目根目录下执行mvn install命令，各服务的微服务描述文件为：src/main/resources/microservice.yaml。以acmeair-booking-service为例：



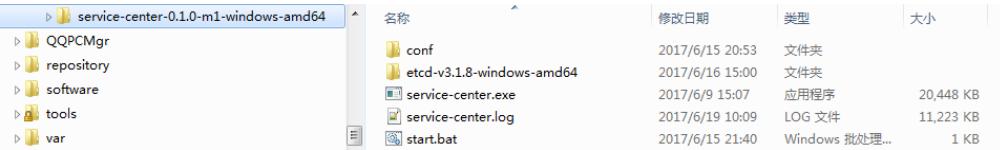
控制台输出BUILD SUCCESS表示项目构建成功



4.3 启动服务

4.3.1 service-center

找到service-center-0.1.0-m1-windows-amd64解压目录，双击start.bat。



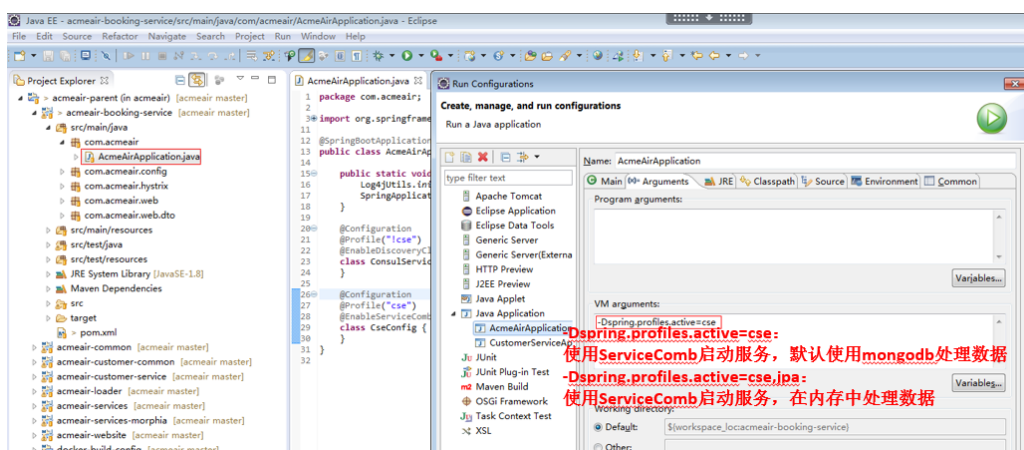
```

C:\Windows\system32\cmd.exe
gistry>390 - service_center."starting manager server in proxy mode"
2017-06-21 11:16:35:24.385162+08:00 DEBUG service_center 9872 github.com/servicecomb/service-center/server/core/registry/etcd/remote_etcd_plugin.go NewRe
gistry()>420 - service_center."refreshManagerClusterInterval is 30"
2017-06-21 11:16:35:24.388852+08:00 WARN service_center 9872 github.com/servicecomb/service-center/server/core/registry/etcd/remote_etcd_plugin.go NewRe
gistry()>428 - service_center."get etcd client [127.0.0.1:2379] completed."
2017-06-21 11:16:35:24.3108392+08:00 INFO service_center 9872 github.com/servicecomb/service-center/server/service/notification/notification_service.go S
tartNotifyService()>176 - service_center."notify service is ready"
2017-06-21 11:16:35:24.3108392+08:00 DEBUG service_center 9872 github.com/servicecomb/service-center/server/core/registry/etcd/remote_etcd_plugin.go Watc
h()>352 - service_center."start to watch key /cse-sr/inst/files/default/"
2017-06-21 11:16:35:24.3108392+08:00 DEBUG service_center 9872 github.com/servicecomb/service-center/server/core/registry/etcd/remote_etcd_plugin.go Watc
h()>352 - service_center."start to watch key /cse-sr/tenant/"
2017-06-21 11:16:35:24.3127932+08:00 INFO service_center 9872 github.com/servicecomb/service-center/etcdsync/mutex.go Lock()>128 - service_center."Trying
to create a lease /cse-sr/inst/files/default/1/1 120s"
2017-06-21 11:16:35:24.3206092+08:00 INFO service_center 9872 github.com/servicecomb/service-center/server/core/registry/etcd/remote_etcd_plugin.go PutNo
Override()>214 - service_center."response /cse-sr/etcdsync/servicecenter, id=szxbz1015-9872-20170621-16:35:24.3127932"
2017-06-21 11:16:35:24.3215862+08:00 WARN service_center 9872 github.com/servicecomb/service-center/etcdsync/mutex.go Lock()>138 - service_center."Create
Lock OK, key=/cse-sr/etcdsync/servicecenter, id=szxbz1015-9872-20170621-16:35:24.3127932"
2017-06-21 11:16:35:24.3245172+08:00 WARN service_center 9872 github.com/servicecomb/service-center/server/api/api_server.go registryService()>154 - serv
ice_center."service center service already registered, service id 1"
2017-06-21 11:16:35:24.3264712+08:00 WARN service_center 9872 github.com/servicecomb/service-center/etcdsync/mutex.go Unlock()>188 - service_center."Dete
le Lock OK, key=/cse-sr/etcdsync/servicecenter, id=szxbz1015-9872-20170621-16:35:24.3127932"
2017-06-21 11:16:35:24.3294022+08:00 INFO service_center 9872 github.com/servicecomb/service-center/server/service/instances.go Register()>74 - service_c
enter."Start register a new instance for service 1"
2017-06-21 11:16:35:24.3381952+08:00 DEBUG service_center 9872 github.com/servicecomb/service-center/server/service/instances.go Register()>125 - service
_center."Instance ID 111"
2017-06-21 11:16:35:24.3401492+08:00 DEBUG service_center 9872 github.com/servicecomb/service-center/server/service/instances.go Register()>178 - service
_center."Start register instance: /cse-sr/inst/files/default/1/1 instanceId:\111 serviceId:\111 endpoints:\rest://127.0.0.1:30100\
hostName:\service-center-10-229-33-225\ status:\UP\ healthCheck:\mode:\push\ interval:30 times:3 > timestamp:\1498034124\ stage:\prod\ . le
ase: /cse-sr/inst/leases/default/default/1/1 120s"
2017-06-21 11:16:35:24.3430602+08:00 WARN service_center 9872 github.com/servicecomb/service-center/server/service/notification/notification_service.go f
unc()>251 - service_center."notification service catch instance 1/1 (UPDATE) event, msg: key information changed"
2017-06-21 11:16:35:24.3450342+08:00 INFO service_center 9872 github.com/servicecomb/service-center/server/core/registry/etcd/remote_etcd_plugin.go PutNo
Override()>214 - service_center."response /cse-sr/tenant/default false 252"
2017-06-21 11:16:35:24.3460112+08:00 WARN service_center 9872 github.com/servicecomb/service-center/server/service/instances.go Register()>226 - service
_center."register instance for service 1, service-center-10-229-33-225(1) successfully from remote ."
2017-06-21 11:16:35:24.3469802+08:00 DEBUG service_center 9872 github.com/servicecomb/service-center/server/service/dependency/dependency_service.go GetC
onsumersInCache()>87 - service_center."Query service consumers in cache 1"
2017-06-21 11:16:35:24.3479652+08:00 WARN service_center 9872 github.com/servicecomb/service-center/common/cron/cron.go Do()>151 - service_center."add jo
b(0x04021ec000) JOBtype: 1(seconds), interval: 30, delay: 0, at: , task: github.com/servicecomb/service-center/server/api.(*APIServer).github.com/s
ervicecomb/service-center/server/api.doAPIHeartBeat-fm."
2017-06-21 11:16:35:24.3479652+08:00 WARN service_center 9872 github.com/servicecomb/service-center/util/rest/server.go ListenAndServe()>58 - service_c
enter."Listen on server 127.0.0.1:30100"
2017-06-21 11:16:35:24.3499192+08:00 INFO service_center 9872 github.com/servicecomb/service-center/server/api/api_server.go Func()>256 - service_center
."api server is ready"

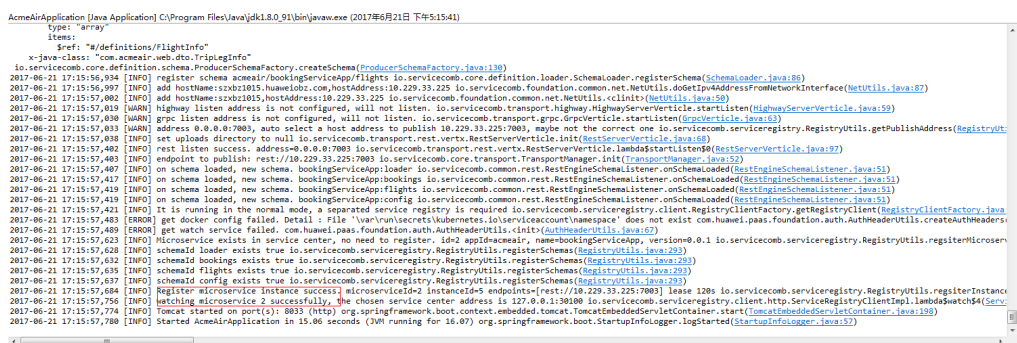
```

4.3.2 acmeair-booking

AcmeairApplication:Run As - - Run Configurations...

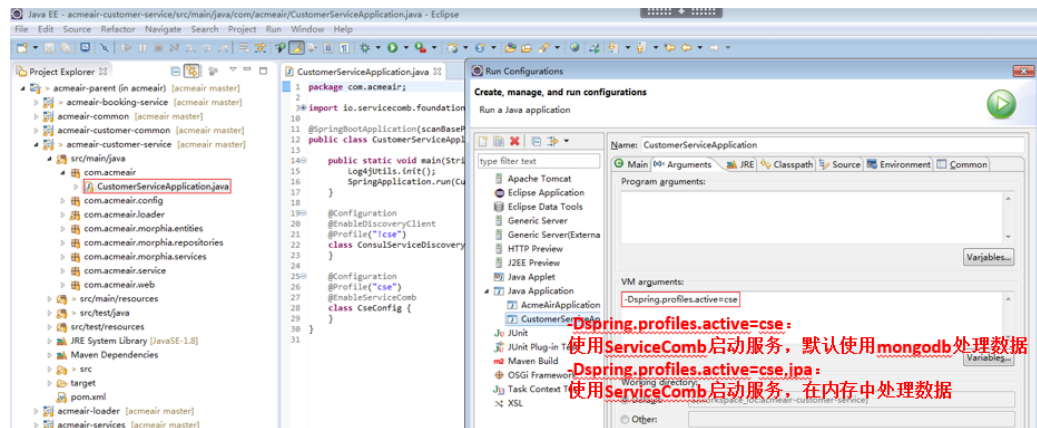


watching microservice 2 successfully, 服务注册成功。



4.3.3 acmeair-customer

CustomerServiceApplication:Run As - - Run Configurations...

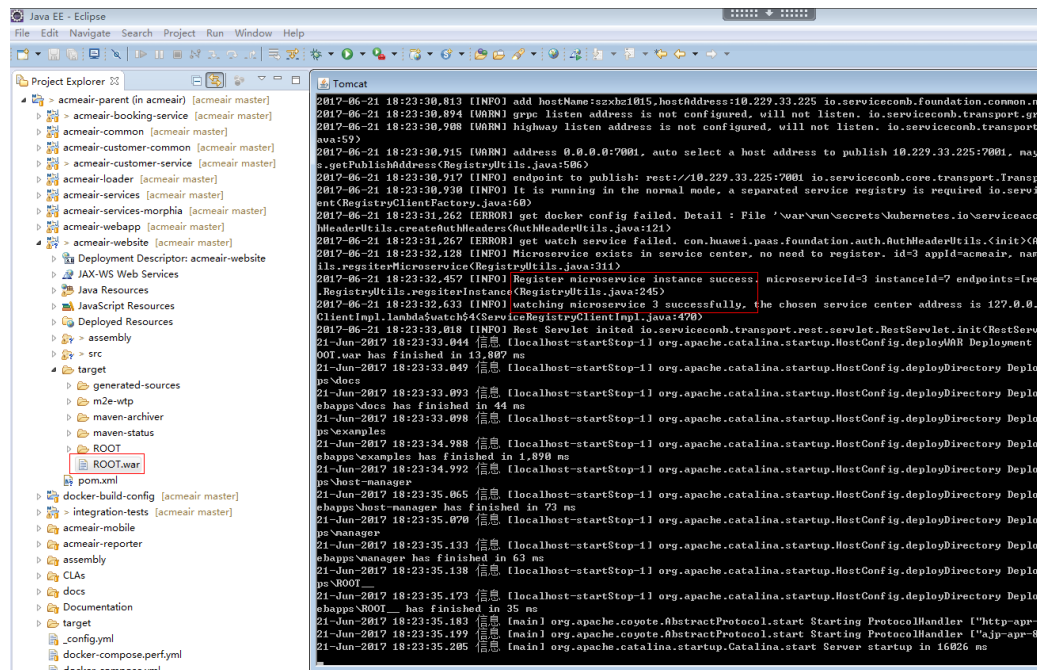


watching microservice 4 successfully, 服务注册成功。



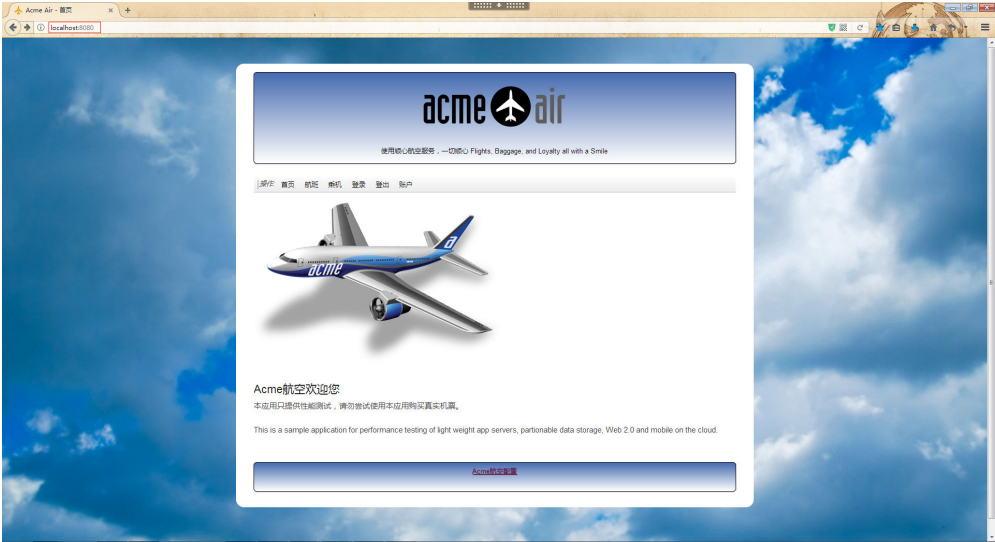
4.3.4 acmeair-website

将acmeair-website\target\ROOT.war复制到tomcat webapps文件夹下，启动tomcat。



4.4 前台访问

步骤1 浏览器地址栏输入<http://localhost:8080>。



步骤2 单击网页最下端Acme航空配置。



----结束

5 acmeair 发布到 Service Stage

本章节演示如何在linux环境下制作镜像以及如何利用Service Stage实现云化。

[5.1 环境准备](#)

[5.2 创建资源](#)

[5.3 制作镜像并上传](#)

[5.4 应用部署](#)

[5.5 应用上线](#)

5.1 环境准备

1. linux
2. docker 1.11.2（当前只支持该版本）
3. maven 3.x
4. JDK 1.8+
5. 注册Service Stage账号。

5.2 创建资源

步骤1 创建名称为acmeair的集群。

请参考http://support.huaweiclouds.com/usermanual-servicestage/zh-cn_topic_0053496188.html



步骤2 添加节点。

请参考http://support.huaweiclouds.com/usermanual-servicestage/zh-cn_topic_0053443149.html

* 节点类型: ☒ 通用型 ☐ 内存优化型
通用型实例提供均衡的计算、存储以及网络配置,适用于大多数的使用场景。通用型实例可用于web服务器、开发测试环境以及小型数据库应用等场景。

* 是否选择EIP: ☐ 是 ☒ 否
选择使用弹性IP时,将为创建的节点绑定弹性IP地址,请确保弹性IP地址配额充足。

* CPU: 1核 2核 4核 8核 16核 32核

* 内存: 2GB 4GB 8GB
为了保证性能体验,Windows2008/2012系统建议选择2GB及以上内存
选择的规格为: c2.large | 2核 | 4GB

* 磁盘: 系统盘 普通IO 40GB ?
+ 增加一块数据盘 您还可以增加10块磁盘(云硬盘)

* 密钥对: KeyPair-39a7 查看秘钥

* 购买数量: 1 台 您还可以增加15个节点

当前配置:	节点类型:	通用型	是否选择EIP:	否
	选择的规格为:	c2.large 2核 4GB	购买数量:	1
	磁盘:	系统盘 普通IO 40GB		
	云服务器费用:	¥ 0.457/小时	EIP费用:	--

----结束

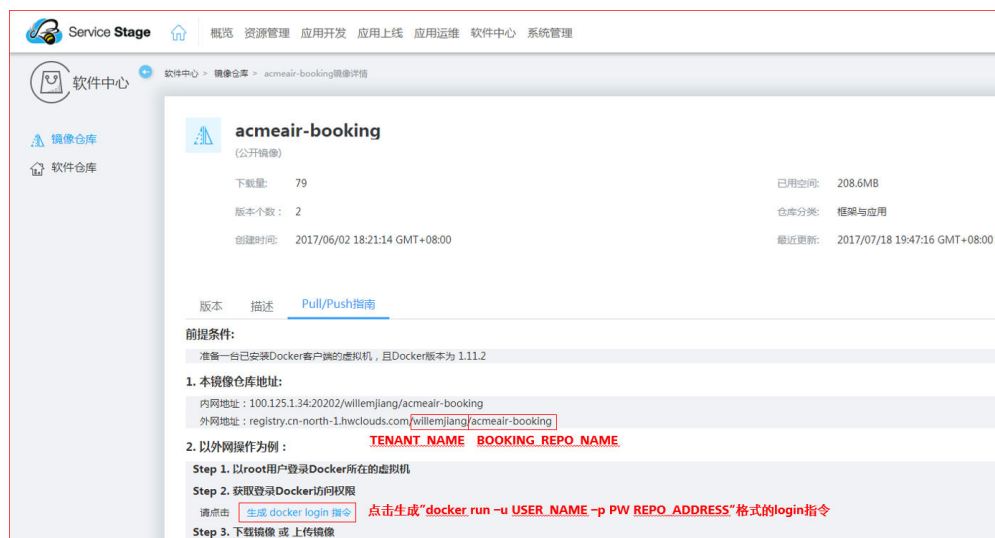
5.3 制作镜像并上传

步骤1 创建镜像仓库。

请参考http://support.huaweiclouds.com/usermanual-servicestage/zh-cn_topic_0055101643.html



步骤2 根据docker login指令修改acmeair/scripts/release_images_to_huaweicloud.sh并运行。



```
[root@acmeair-bzx8a acmeair]# vi scripts/release_images_to_huaweicloud.sh

#how to use(linux): 1.git clone https://github.com/TankTian/acmeair 2. Set the following variables 3. ./release_images_to_huaweicloud.sh
#huawei cloud website : https://servicestage.hwclouds.com/

#config example
#TARGET_VERSION=0.0.1
#ORIGIN_VERSION=2.0.0-SNAPSHOT
#TENANT_NAME=xxxxxxxxxxxx
#REPO_ADDRESS=registry.cn-north-1.hwclouds.com
#USER_NAME=xxxxxx
#PW=xxxxxxx
#CUSTOMER_REPO_NAME=acmeair-customer
#BOOKING_REPO_NAME=acmeair-booking
#WEBSITE_REPO_NAME=acmeair-website

-----huawei cloud images repo save version.
-----images version been made "mvn -Pdocker".
-----huawei cloud tenant name.
-----huawei cloud images repo address.
-----user name: login huawei cloud images repo.
-----password: login huawei cloud images repo.
-----customer repo name ,created by huawei cloud.
-----booking repo name ,created by huawei cloud.
-----website repo name ,created by huawei cloud.

CUR_PATH=`pwd`
ROOT_PATH=${CUR_PATH}/../
cd ${ROOT_PATH}
docker rmi -f $(docker images|grep acmeair-customer-service|grep $ORIGIN_VERSION |awk '{print $3}')
docker rmi -f $(docker images|grep acmeair-booking-service|grep $ORIGIN_VERSION |awk '{print $3}')
docker rmi -f $(docker images|grep acmeair-website|grep $ORIGIN_VERSION |awk '{print $3}')
mvn clean install -Phuaweicloud -Pdocker -Dmaven.test.skip=true
docker tag acmeair-customer-service:$ORIGIN_VERSION ${REPO_ADDRESS}/${TENANT_NAME}/${CUSTOMER_REPO_NAME}:$TARGET_VERSION
docker tag acmeair-booking-service:$ORIGIN_VERSION ${REPO_ADDRESS}/${TENANT_NAME}/${BOOKING_REPO_NAME}:$TARGET_VERSION
docker tag acmeair-website:$ORIGIN_VERSION ${REPO_ADDRESS}/${TENANT_NAME}/${WEBSITE_REPO_NAME}:$TARGET_VERSION
docker login -u ${USER_NAME} -p ${PW} ${REPO_ADDRESS}
docker push ${REPO_ADDRESS}/${TENANT_NAME}/${CUSTOMER_REPO_NAME}:$TARGET_VERSION
docker push ${REPO_ADDRESS}/${TENANT_NAME}/${BOOKING_REPO_NAME}:$TARGET_VERSION
docker push ${REPO_ADDRESS}/${TENANT_NAME}/${WEBSITE_REPO_NAME}:$TARGET_VERSION
```

1. -Phuaweicloud激活id为huaweicloud的profile，实现以下两个功能：

1) 下载商业版本的jar包：

```
<profile>
<id>huaweicloud</id>
<!-- 下面是应用基于开源ServiceComb开发上华为公有云运行时必须依赖的包（非编译依赖），他们放在单独仓库：http://cse.servicecomb.io/maven
tracing-provider,sla-provider -->
<dependencies>
<dependency>
<groupId>com.huawei.paas.cse</groupId>
<artifactId>foundation-auth</artifactId>
<version>${huaweicloud.version}</version>
</dependency>
<dependency>
<groupId>com.huawei.paas.cse</groupId>
<artifactId>foundation-config-cc</artifactId>
<version>${huaweicloud.version}</version>
</dependency>
<dependency>
<groupId>com.huawei.paas.cse</groupId>
<artifactId>cse-handler-tracing</artifactId>
<version>${huaweicloud.version}</version>
</dependency>
<dependency>
<groupId>com.huawei.paas.cse</groupId>
<artifactId>cse-handler-performance-stats</artifactId>
<version>${huaweicloud.version}</version>
</dependency>
</dependencies>
<repositories>
<repository>
<id>huawei_cloud_dependencies</id>
<name>huawei_cloud_dependencies</name>
<url>http://cse.servicecomb.io/maven</url>
<releases>
<enabled>true</enabled>
</releases>
<snapshots>
<enabled>false</enabled>
</snapshots>
</repository>
</repositories>
</profile>
```

商业版本的jar包，应用利用该jar包使用Service Stage service center/config center/调用链等功能

商业版本jar包的仓库地址

```

APPLICATION.ID: acmeair
service_description:
  name: bookingServiceApp
  version: 0.0.1
cse:
  service:
    registry:
      address: https://100.125.1.34:30100 公有云服务中心的地址
    monitor:
      client:
        serverUri: https://100.125.1.34:30109
  rest:
    address: 0.0.0.0:7003
  handler:
    chain:
      Provider:
        default: bizkeeper-provider,perf-stats,tracing-provider,sla-provider,qps-flowcontrol-provider
      Consumer:
        default: bizkeeper-consumer,loadbalance,perf-stats,tracing-consumer,sla-consumer,qps-flowcontrol-consumer
  references:
    customerServiceApp:
      version-rule: 0.0.1
      transport: rest
  config:
    client:
      serverUri: https://100.125.1.34:30103 公有云配置中心地址

```

```
<plugin>
<groupId>io.fabric8</groupId>
<artifactId>docker-maven-plugin</artifactId>
<configuration>
  <images>
    <image>
      <name>${project.artifactId}:${project.version}</name>
      <alias>${project.artifactId}</alias>
      <build>
        <from>openjdk:8-jre-alpine</from>
        <ports>
          <port>7072</port>
          <port>8082</port>
          <port>9992</port>
        </ports>
      </build>
    </image>
  </images>
  <assembly>
    <mode>tar</mode>
    <descriptor>${root.basedir}/assembly/docker_assembly.xml</descriptor>
  </assembly>
  <entryPoint>
    <shell>java $JAVA_OPTS -Dspring.profiles.active=cse -Dspring.data.mongodb.database=acmeair -Dspring.data.mongodb.host=mongodb
      -Dspring.data.mongodb.port=32701 -jar /maven/acmeair/${JAR_NAME}-${project.version}-exec.jar</shell>
    </entryPoint>
  </build>
</image>
</configuration>
```

5.4 应用部署

5.4.1 手动部署

文档版本 01 (2017-07-21)

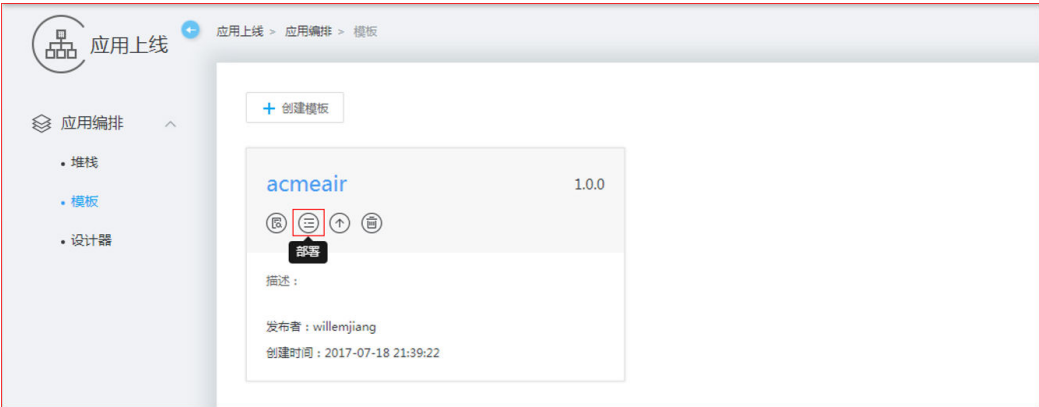


5.4.2 自动部署

本地上传acmeair/scripts/acmeair-blueprint-deploy-template-v1.tar.gz自动部署模板并部署。

请参考：http://support.hwclouds.com/usermanual-servicestage/zh-cn_topic_0066615707.html





5.5 应用上线

应用部署完成后，在acmeair集群下会有如下应用列表：

集群: acmeair							
<div>+ 创建应用 移至应用组 删除</div>							
<div>全部状态 应用名称搜索</div>							
<input type="checkbox"/>	名称	状态	应用组	应用类型	类型	实例个数(正常/全部)	创建时间
<input type="checkbox"/>	acmeair-customer	运行中		无状态应用	容器	1/1	2017/07/18 17:26:29 GMT...
<input type="checkbox"/>	acmeair-booking	运行中		无状态应用	容器	1/1	2017/07/18 17:26:29 GMT...
<input type="checkbox"/>	mongodb	运行中		无状态应用	容器	1/1	2017/07/18 17:26:16 GMT...
<input type="checkbox"/>	acmeair-website	运行中		无状态应用	容器	1/1	2017/07/18 17:26:16 GMT...
<div>10 总数: 4</div>							

点击应用名称进入应用详情页面，点击访问地址访问应用。



6 acmeair 的运维和治理

目的：保障应用平稳运行。

- 微服务状态监控
请参考http://support.hwclouds.com/usermanual-servicestage/zh-cn_topic_0053555506.html



- 按需弹性伸缩
请参考http://support.hwclouds.com/usermanual-servicestage/zh-cn_topic_0054051691.html



- 微服务可视化治理
请参考http://support.hwclouds.com/usermanual-servicestage/zh-cn_topic_0053560172.html

