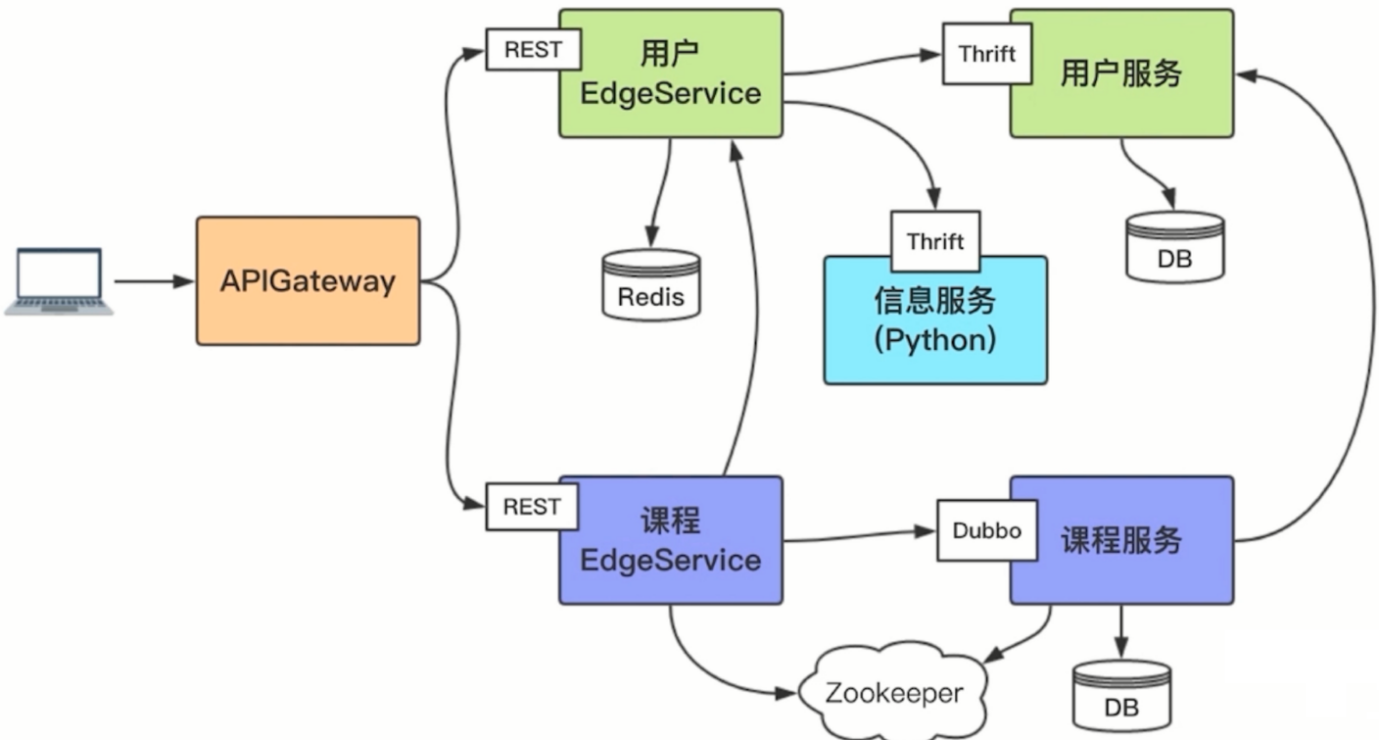
# Docker+Kubernetes(k8s)微服务容器化实践

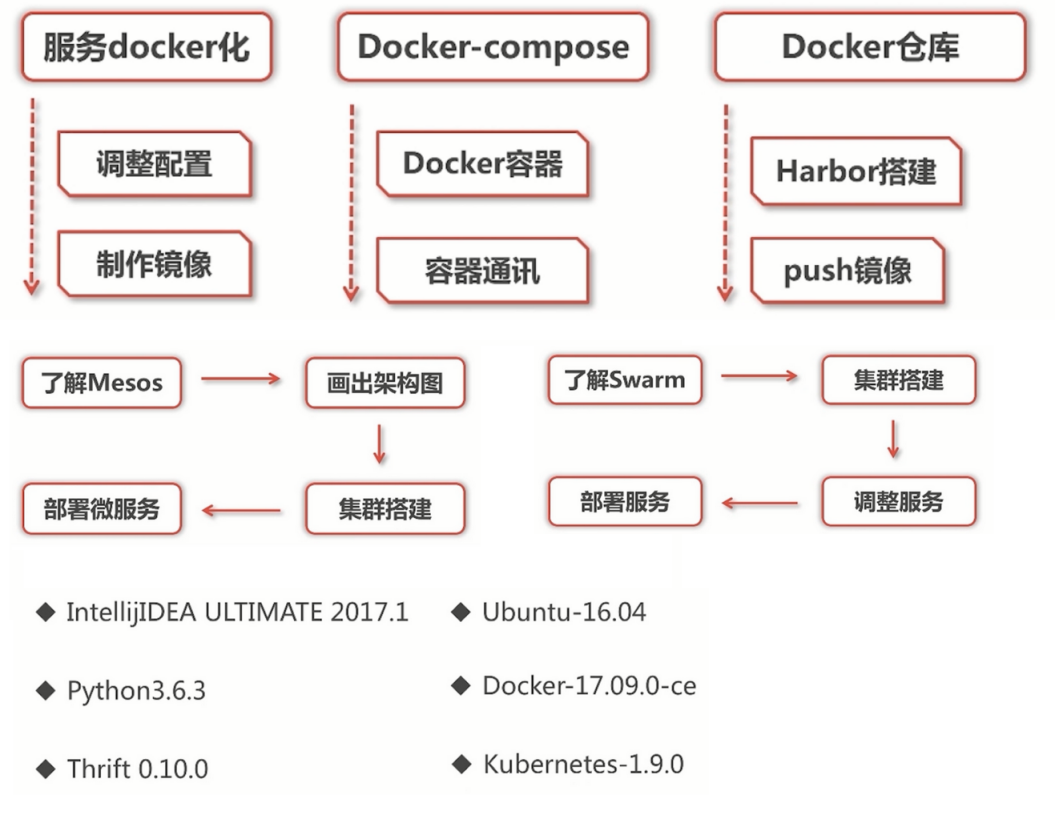
## 第一章 初识微服务

### 1.1概述

（1）Web应用整体框架

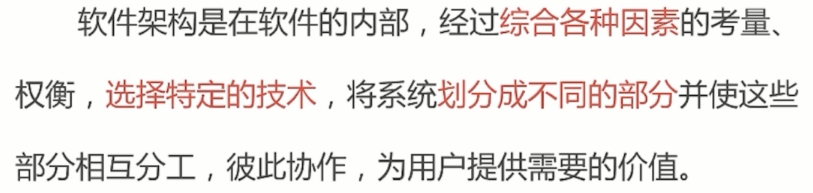


（2）容器化与DevOps工具

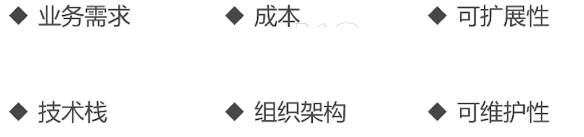


### 1.2软件架构的进化

·软件架构定义



·影响因素



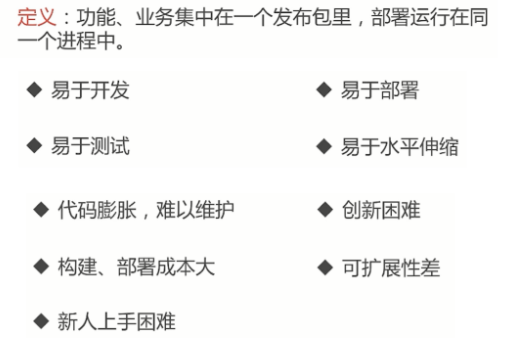
·软件架构进化

·一层架构

·MVC

·dubbo

·单体架构



### 1.3微服务

使用一套小服务来开发单个应用的方法，每个服务运行在独立的进程里，一般采用轻量级的通讯机制互联，并且他们可以通过自动化的方式部署。

·多微才算微

·代码量？

·开发时间

·不可度量

·微服务特征

·单一职责

·轻量级通讯

·隔离性

·有自己的数据

·技术多样性

（1）Grpc 使用HTTP2,序列化效率最高

（2）Dubbo使用的是python序列化

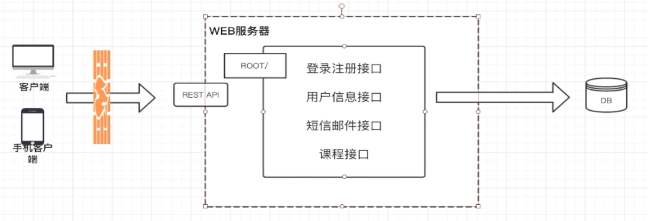
·微服务诞生背景

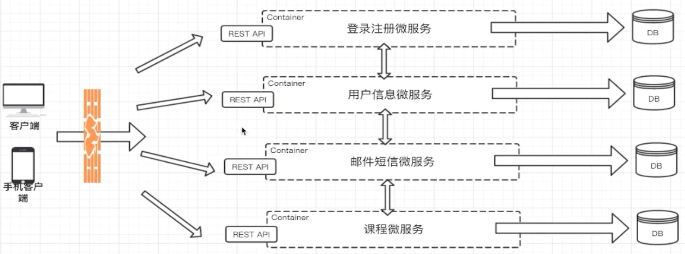
·互联网行业的快速发展

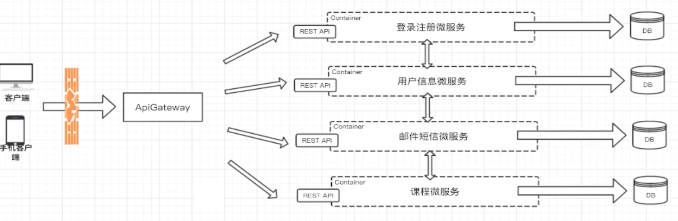
·敏捷开发，精益方法深入人心

·容器技术的成熟

### 1.4画出微服务架构图







### 1.5微服务架构的优势

·独立性

·敏捷性

·技术栈灵活

·高效团队

不足：

·额外的工作

·数据一致性

·沟通成本（接口变更）

## 微服务带来的问题及解决方案分析

### 2.1微服务结构带来的问题

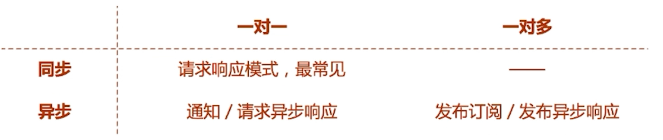
·微服务间如何通讯

·微服务如何发现彼此

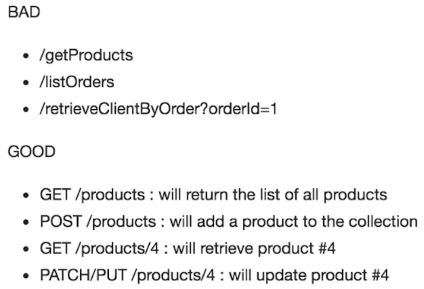
·微服务怎样部署？更新？扩容？

### 2.2微服务间如何通讯

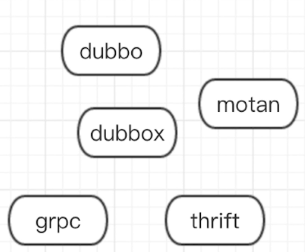
·从通讯模式角度考虑



·REST API



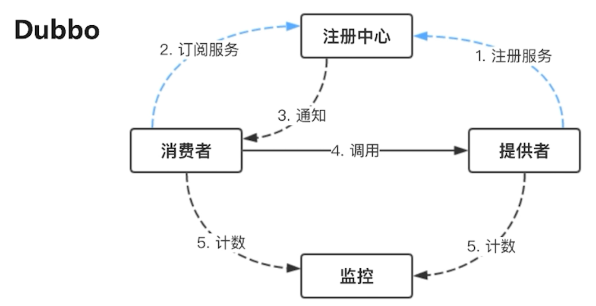
·RPC框架

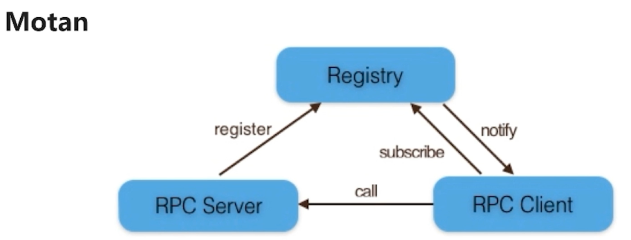


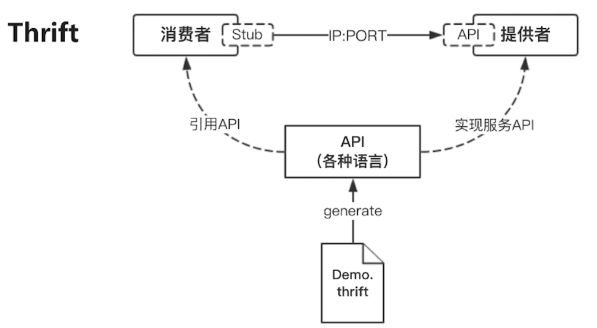
·MQ

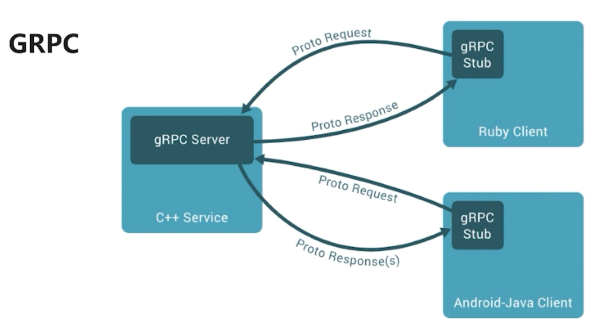
·如何选择RPC框架

1. I/O、线程调度模型
2. 序列化方式，二进制、JSON：影响RPC通讯效率
3. 多语言支持
4. 服务治理





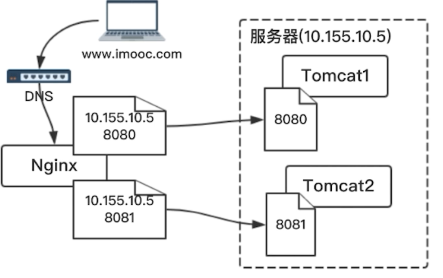




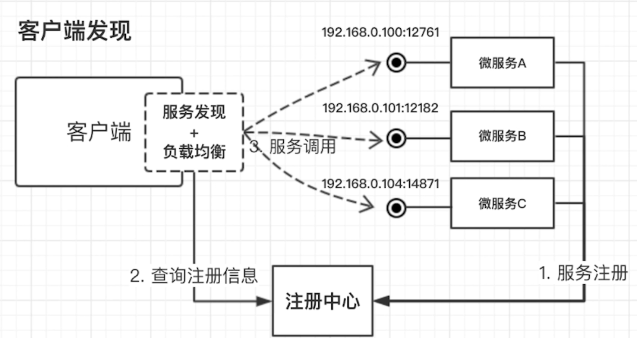


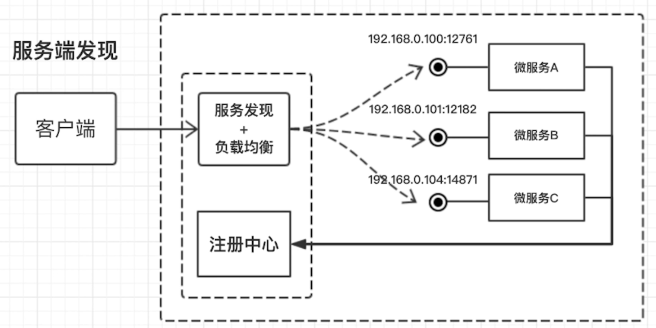
### 2.3服务发现

·传统服务VS微服务



·服务发现





·服务部署、更新和扩容

·服务编排

·流行的服务编排工具



### 2.4 SpringBoot & SpringCloud

·SpringBoot与微服务

·SpringBoot核心功能

·独立运行 java -jar xxx.jar

·内嵌web服务器

·简化配置

·准生产的应用监控

·微服务SpringCloud

·统一配置管理

·服务注册与发现

·服务间调用

·负载均衡

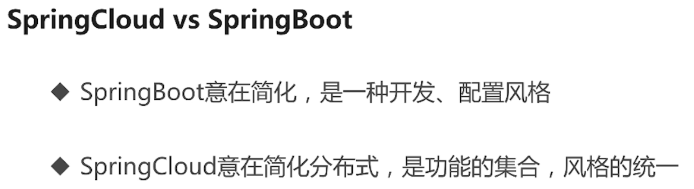
·分布式锁

·分布式Session

·深入理解

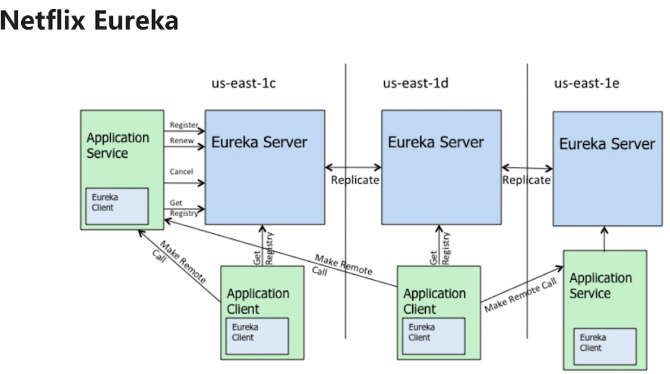
·一系列框架

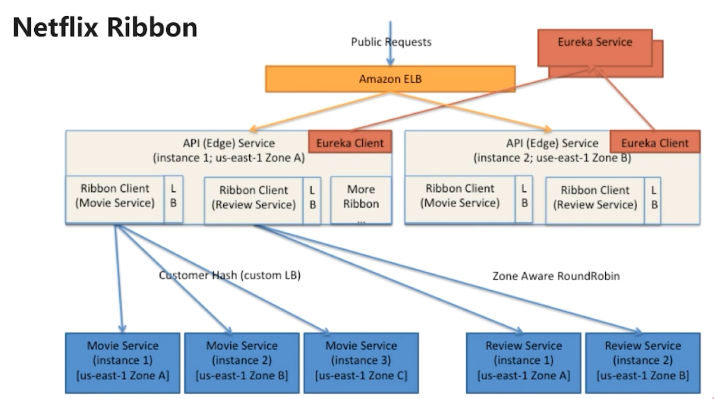
·简化java的分布式开发



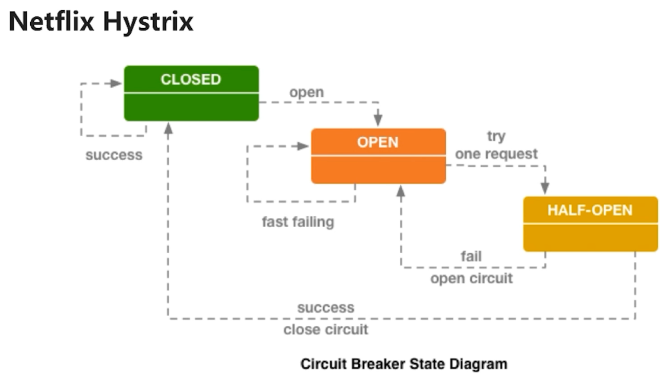
·SpringCloud核心组件

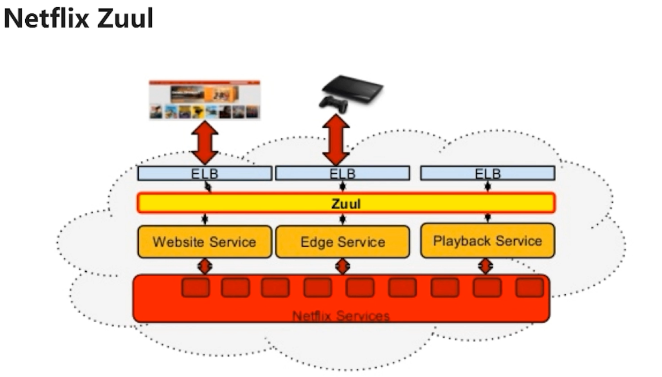


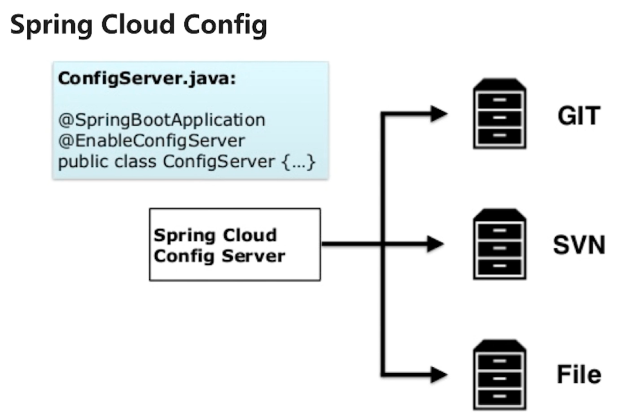




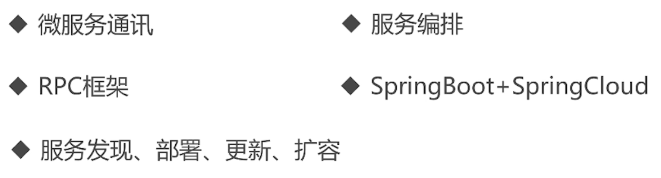
·Edge Service：Web RESTAPI







### 2.5小节



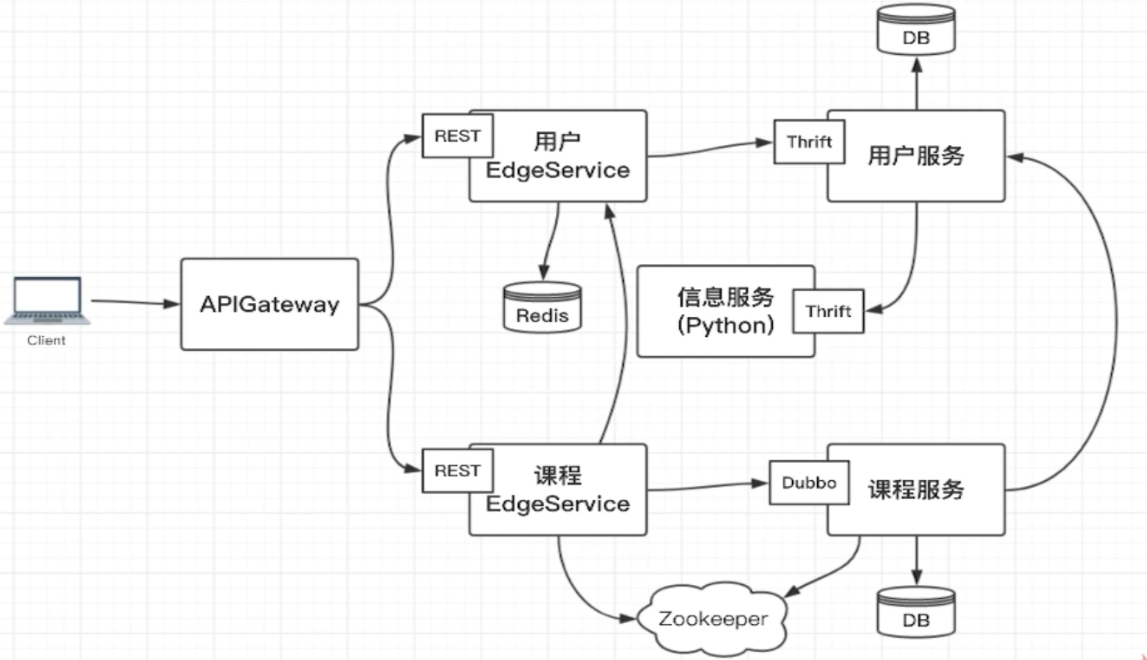
## 微服务开发

### 3.1微服务业务分析

·业务场景

·用户可以注册和登录

·登录用户可以对课程进行CRUD



### 3.2 Thirft安装和验证

|  |
| --- |
| namespace java com.byf.thrift.demo  namespace py thrift.demo  service DemoService {  void sayHello(1:string name);  } |

|  |
| --- |
| C:\Users\BYF\Downloads\thrift-test>thrift-0.12.0.exe --gen java demo.thrift  C:\Users\BYF\Downloads\thrift-test>thrift-0.12.0.exe --gen py demo.thrift |

|  |
| --- |
| C:\Users\BYF\Downloads\thrift-test 的目录  2019/09/22 14:57 <DIR> .  2019/09/22 14:57 <DIR> ..  2019/09/22 14:56 120 demo.thrift  2019/09/22 14:56 <DIR> gen-java  2019/09/22 14:57 <DIR> gen-py  2019/09/22 14:52 3,887,104 thrift-0.12.0.exe |

### 3.3 Python开发信息服务

|  |
| --- |
| namespace java com.byf.thrift.message namespace py message.api  service MessageService {  bool sendMobileMessage(1:string mobile, 2:string message);  bool sendEmailMessage(1:string email, 2:string message); } |

|  |
| --- |
| G:\MicroService\microservice\message-thrift-python-service\thrift>thrift-0.12.0.exe --gen py -out ../ message.t  hrift |

idea installed package failed目录缺少packaging\_tool.py

Intellij IDEA安装SDKs失败：

将目录F:\Program Files\JetBrains\IntelliJ IDEA 2019.1.1\plugins\python\helpers

拷贝至Intellij IDEA配置目录：C:\Users\BYF\.IntelliJIdea2019.1\config

|  |
| --- |
| **if** \_\_name\_\_ == **'\_\_main\_\_'**:  handler = MessageServiceHandler()  processor = MessageService.Processor(handler)  transport = TSocket.TServerSocket(**None**, **"9090"**)  tfactory = TTransport.TFramedTransportFactory()  pfactory = TBinaryProtocol.TBinaryProtocolFactory()   server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)  **print** (**"python thrift server start"**)  server.serve()  **print** (**"python thrift server exit"**) |
| org.apache.thrift.transport.TTransportException: java.net.ConnectException: Connection refused: connect  客户端连接时报错，修改None为具体IP地址 |
| **if** \_\_name\_\_ == **'\_\_main\_\_'**:  handler = MessageServiceHandler()  processor = MessageService.Processor(handler)  transport = TSocket.TServerSocket(**"127.0.0.1"**, **"9090"**)  tfactory = TTransport.TFramedTransportFactory()  pfactory = TBinaryProtocol.TBinaryProtocolFactory()   server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)  **print** (**"python thrift server start"**)  server.serve()  **print** (**"python thrift server exit"**) |

### 3.4 dubbo上手示例

<http://dubbo.apache.org/en-us/docs/user/quick-start.html>

|  |
| --- |
| <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  xmlns="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd"**>   *<!-- provider's application name, used for tracing dependency relationship -->* <**dubbo:application name="demo-provider"**>  <**dubbo:parameter key="qos.enable" value="true"**/>  <**dubbo:parameter key="qos.accept.foreign.ip" value="false"**/>  <**dubbo:parameter key="qos.port" value="33333"**/>  </**dubbo:application**>  *<!-- use multicast registry center to export service -->* <**dubbo:registry address="multicast://224.5.6.7:1234?unicast=false"**/>  *<!-- use dubbo protocol to export service on port 20880 -->* <**dubbo:protocol name="dubbo" port="20880"**/>  *<!-- service implementation, as same as regular local bean -->* <**bean id="demoService" class="org.apache.dubbo.demo.provider.DemoServiceImpl"**/>  *<!-- declare the service interface to be exported -->* <**dubbo:service interface="org.apache.dubbo.demo.DemoService" ref="demoService"**/> </**beans**> |

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  xmlns="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd"**>   *<!-- consumer's application name, used for tracing dependency relationship (not a matching criterion),  don't set it same as provider -->* <**dubbo:application name="demo-consumer"**>  <**dubbo:parameter key="qos.enable" value="true"** />  <**dubbo:parameter key="qos.accept.foreign.ip" value="false"** />  <**dubbo:parameter key="qos.port" value="33333"** />  </**dubbo:application**>  *<!-- use multicast registry center to discover service -->  <!--<dubbo:registry address="multicast://224.5.6.7:1234?unicast=false" check="false" />-->  <!-- <dubbo:registry valid="false" check="false" zookeeperProtocol="false" id="org.apache.dubbo.config.RegistryConfig" prefix="dubbo.registries." />-->  <!-- generate proxy for the remote service, then demoService can be used in the same way as the  local regular interface -->* <**dubbo:reference id="demoService" check="false" interface="org.apache.dubbo.demo.DemoService" url="127.0.0.1:20880"**/> </**beans**> |

-Ddubbo.application.qos.enable=true -Ddubbo.application.qos.port=33333 -Ddubbo.application.qos.accept.foreign.ip=false

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>dubbo-demo</**artifactId**>  <**groupId**>com.byf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-consumer</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>5.0.8.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.dubbo</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.7.2</**version**>  </**dependency**>  </**dependencies**>  </**project**> |

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>dubbo-demo</**artifactId**>  <**groupId**>com.byf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-provider</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>5.0.8.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.dubbo</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.7.2</**version**>  </**dependency**>  </**dependencies**>  </**project**> |

|  |
| --- |
| *###set log levels###* **log4j.rootLogger**=**info, stdout** *###output to the console###* **log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.Target**=**System.out log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**[%d{dd/MM/yy hh:mm:ss:sss z}] %t %5p %c{2}: %m%n** |

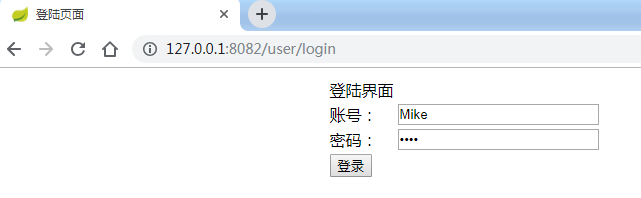
### 3.5课程服务

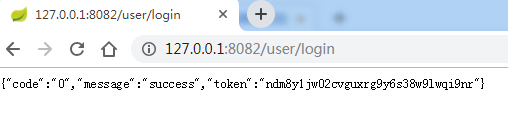
|  |
| --- |
| namespace java com.byf.thrift.user  struct UserInfo {  1:i32 id,  2:string username,  3:string password,  4:string realName,  5:string mobile,  6:string email,  7:string intro,  8:i32 stars }  service UserService {  UserInfo getUserById(1:i32 id);  UserInfo getTeacherById(1:i32 id);   UserInfo getUserByName(1:string username);  void registerUser(1:UserInfo userInfo); } |

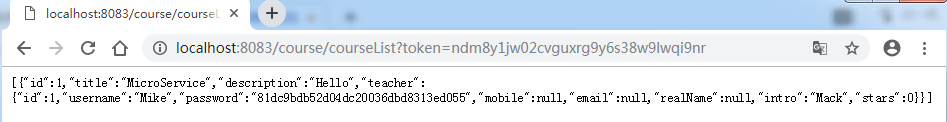
G:\MicroService\microservice\user-thrift-service-api\thrift>thrift-0.12.0.exe --gen java -out ../src/main/java user

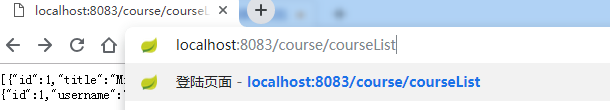
\_service.thrift

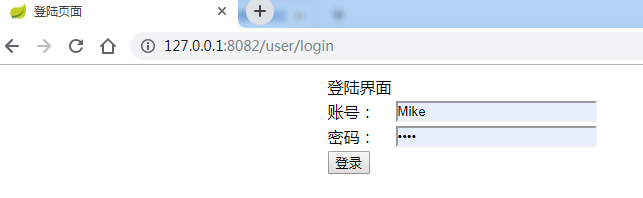
|  |
| --- |
| java.lang.IllegalStateException: Failed to check the status of the service com.byf.course.service.ICourseService. No provider available for the service com.byf.course.service.ICourseService from the url zookeeper://127.0.0.1:2181/com.alibaba.dubbo.registry.RegistryService?application=course-service&dubbo=2.5.3&interface=com.byf.course.service.ICourseService&methods=courseList&pid=7732&side=consumer&timestamp=1569339917977 to the consumer 192.168.0.107 use dubbo version 2.5.3 |
| 原因已找到。错误在CourseServiceImpl类中@Service注解用的是dubbo类中的，不是Spring中的！！！一定要注意 com.alibaba.dubbo.config.annotation.Service |











## 第4章微服务编排前奏

### 4.1服务Docker化

idea maven package无法将依赖的jar包打入一个jar包

|  |
| --- |
| <**plugin**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-maven-plugin</**artifactId**>  <**executions**>  <**execution**>  <**goals**>  <**goal**>repackage</**goal**>  </**goals**>  </**execution**>  </**executions**> </**plugin**> |

Centos7上安装docker

|  |
| --- |
| **一、安装docker** 1、Docker 要求 CentOS 系统的内核版本高于 3.10 ，查看本页面的前提条件来验证你的CentOS 版本是否支持 Docker 。  通过 **uname -r**命令查看你当前的内核版本  $ uname -r  2、使用 root 权限登录 Centos。确保 yum 包更新到最新。  $ sudo yum update  3、卸载旧版本(如果安装过旧版本的话)  $ sudo yum remove docker docker-common docker-selinux docker-engine  4、安装需要的软件包， yum-util 提供yum-config-manager功能，另外两个是devicemapper驱动依赖的  $ sudo yum install -y yum-utils device-mapper-persistent-data lvm2  5、设置yum源  $ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo  IMG_256  6、可以查看所有仓库中所有docker版本，并选择特定版本安装  $ yum list docker-ce --showduplicates | sort -r  IMG_257  7、安装docker  $ sudo yum install docker-ce #由于repo中默认只开启stable仓库，故这里安装的是最新稳定版17.12.0  $ sudo yum install <FQPN> # 例如：sudo yum install docker-ce-17.12.0.ce  IMG_258  8、启动并加入开机启动  $ sudo systemctl start docker  $ sudo systemctl enable docker  9、验证安装是否成功(有client和service两部分表示docker安装启动都成功了)  $ docker version  IMG_259 |

拉取镜像

|  |
| --- |
| Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: TLS handshake timeout  [root@localhost /etc/docker]#vi /etc/docker/daemon.json  {  "registry-mirrors":["https://6kx4zyno.mirror.aliyuncs.com"]  }  [root@localhost /etc/docker]#systemctl restart docker.service  [root@localhost /etc/docker]#systemctl status docker.service |

|  |
| --- |
| Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]    -d, --detach=false 指定容器运行于前台还是后台，默认为false  -i, --interactive=false 打开STDIN，用于控制台交互  -t, --tty=false 分配tty设备，该可以支持终端登录，默认为false  -u, --user="" 指定容器的用户  -a, --attach=[] 登录容器（必须是以docker run -d启动的容器）  -w, --workdir="" 指定容器的工作目录  -c, --cpu-shares=0 设置容器CPU权重，在CPU共享场景使用  -e, --env=[] 指定环境变量，容器中可以使用该环境变量  -m, --memory="" 指定容器的内存上限  -P, --publish-all=false 指定容器暴露的端口  -p, --publish=[] 指定容器暴露的端口  -h, --hostname="" 指定容器的主机名  -v, --volume=[] 给容器挂载存储卷，挂载到容器的某个目录  --volumes-from=[] 给容器挂载其他容器上的卷，挂载到容器的某个目录  --cap-add=[] 添加权限，权限清单详见：http://linux.die.net/man/7/capabilities  --cap-drop=[] 删除权限，权限清单详见：http://linux.die.net/man/7/capabilities  --cidfile="" 运行容器后，在指定文件中写入容器PID值，一种典型的监控系统用法  --cpuset="" 设置容器可以使用哪些CPU，此参数可以用来容器独占CPU  --device=[] 添加主机设备给容器，相当于设备直通  --dns=[] 指定容器的dns服务器  --dns-search=[] 指定容器的dns搜索域名，写入到容器的/etc/resolv.conf文件  --entrypoint="" 覆盖image的入口点  --env-file=[] 指定环境变量文件，文件格式为每行一个环境变量  --expose=[] 指定容器暴露的端口，即修改镜像的暴露端口  --link=[] 指定容器间的关联，使用其他容器的IP、env等信息  --lxc-conf=[] 指定容器的配置文件，只有在指定--exec-driver=lxc时使用  --name="" 指定容器名字，后续可以通过名字进行容器管理，links特性需要使用名字  --net="bridge" 容器网络设置:  bridge 使用docker daemon指定的网桥  host //容器使用主机的网络  container:NAME\_or\_ID >//使用其他容器的网路，共享IP和PORT等网络资源  none 容器使用自己的网络（类似--net=bridge），但是不进行配置  --privileged=false 指定容器是否为特权容器，特权容器拥有所有的capabilities  --restart="no" 指定容器停止后的重启策略:  no：容器退出时不重启  on-failure：容器故障退出（返回值非零）时重启  always：容器退出时总是重启  --rm=false 指定容器停止后自动删除容器(不支持以docker run -d启动的容器)  --sig-proxy=true 设置由代理接受并处理信号，但是SIGCHLD、SIGSTOP和SIGKILL不能被代理 |

|  |
| --- |
| [root@localhost /etc/docker]#docker pull openjdk:8-jre  8-jre: Pulling from library/openjdk  092586df9206: Pull complete  ef599477fae0: Pull complete  4530c6472b5d: Pull complete  f68761487d74: Pull complete  44e59d0ee089: Pull complete  81c7d41ff281: Pull complete  Digest: sha256:b2096eb6ac13c88e431e100db54c470f48c4228b1f2503a8419dbb9a7d3df8f6  Status: Downloaded newer image for openjdk:8-jre  docker.io/library/openjdk:8-jre  [root@localhost /etc/docker]#docker images  REPOSITORY TAG IMAGE ID CREATED SIZE  openjdk 8-jre fd62519d2906 2 weeks ago 246MB |

|  |
| --- |
| Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: TLS handshake timeout  [root@localhost /etc/docker]#vi daemon.json  {  "registry-mirrors":["https://6kx4zyno.mirror.aliyuncs.com"]  }  [root@localhost /etc/docker]#systemctl restart docker.service  [root@localhost /etc/docker]#systemctl status docker.service  docker run -it --name=jdk8 --hostname=centos --entrypoint bash openjdk:8-jre  192.168.1.108  [root@localhost /soft/microservice/user-thrift-service]#ll  total 4  -rw-r--r--. 1 root root 175 Sep 28 18:08 Dockerfile  drwxr-xr-x. 2 root root 50 Sep 28 18:08 target  [root@localhost /soft/microservice/user-thrift-service]#docker build -t user-service:latest .  Sending build context to Docker daemon 12.07MB  Step 1/4 : FROM openjdk:8-jre  ---> fd62519d2906  Step 2/4 : MAINTAINER BYF baiyifan@163.com  ---> Running in bcc414d0ec59  Removing intermediate container bcc414d0ec59  ---> 2c6cc8a8208b  Step 3/4 : COPY target/user-thrift-service-1.0-SNAPSHOT.jar /user-service.jar  ---> 33ddd61c0254  Step 4/4 : ENTRYPOINT ["java", "-jar", "/user-service.jar"]  ---> Running in f467607aac7c  Removing intermediate container f467607aac7c  ---> 8edde3390dc2  Successfully built 8edde3390dc2  Successfully tagged user-service:latest  [root@localhost /soft/microservice/user-thrift-service]#docker images  REPOSITORY TAG IMAGE ID CREATED SIZE  user-service latest 8edde3390dc2 5 seconds ago 258MB  openjdk 8-jre fd62519d2906 2 weeks ago 246MB  docker rm user-service  docker run -d --name=user-service --hostname=centos user-service:latest --mysql.address=192.168.1.108  docker rm user-edge-service  docker run -d --name=user-edge-service --hostname=centos user-edge-service:latest --redis.address=192.168.1.108  docker rm course-service  docker run -d --name=course-service --hostname=centos course-service:latest --zookeeper.address=192.168.1.108 --mysql.address=192.168.1.108  docker stop course-edge-service  docker rm course-edge-service  docker run -d --name=course-edge-service --hostname=centos course-edge-service:latest --zookeeper.address=192.168.1.108  docker run -it --name=course-edge-service --hostname=centos course-edge-service:latest --zookeeper.address=192.168.1.108  docker stop api-gateway-zuul  docker rm api-gateway-zuul  docker run -d --name=api-gateway-zuul --hostname=centos api-gateway-zuul:latest  docker run -it --name=api-gateway-zuul --hostname=centos api-gateway-zuul:latest |

### 4.2 Docker下的服务通信

|  |
| --- |
| 带token请求详情，问题  This application has no explicit mapping for /error, so you are seeing this as a fallback.  Mon Mar 25 14:40:25 UTC 2019 There was an unexpected error (type=Internal Server Error, status=500). Failed to invoke the method courseList in the service com.imooc.course.service.ICourseService. Tried 3 times of the providers [192.168.48.4:20880] (1/1) from the registry 192.168.1.10:2181 on the consumer 192.168.48.6 using the dubbo version 2.5.3. Last error is: Invoke remote method timeout. method: courseList, provider: dubbo://192.168.48.4:20880/com.imooc.course.service.ICourseService?anyhost=true&application=course-service&check=false&dubbo=2.5.3&interface=com.imooc.course.service.ICourseService&methods=courseList&pid=1&revision=1.0-SNAPSHOT&side=consumer&timestamp=1553524794080, cause: Waiting server-side response timeout by scan timer. start time: 2019-03-25 14:40:24.672, end time: 2019-03-25 14:40:25.686, client elapsed: 0 ms, server elapsed: 1014 ms, timeout: 1000 ms, request: Request [id=2, version=2.0.0, twoway=true, event=false, broken=false, data=RpcInvocation [methodName=courseList, parameterTypes=[], arguments=[], attachments={path=com.imooc.course.service.ICourseService, interface=com.imooc.course.service.ICourseService, version=0.0.0}]], channel: /192.168.48.6:54198 -> /192.168.48.4:20880  老师，您看的出这个可能怎么引发的吗，带token请求列表的时候 |
| 超时，一般是网络问题，在容器中测测网络 ip和端口 |

|  |
| --- |
| version: '3'  networks:  default:  external:  name: byf-network  services:  message-service:  image: message-service:latest  ports:  - 9090:9090    user-service:  image: user-service:latest  command:  - "--mysql.address=192.168.1.103"  ports:  - 7911:7911  - 17911:17911  user-edge-service:  image: user-edge-service:latest  links:  - user-service  - message-service  command:  - "--redis.address=192.168.1.103"  ports:  - 8082:8082  - 18082:18082  course-service:  image: course-service:latest  links:  - user-service  command:  - "--mysql.address=192.168.1.103"  - "--zookeeper.address=192.168.1.103"  ports:  - 20880:20880    course-edge-service:  image: course-edge-service:latest  links:  - user-edge-service  command:  - "--zookeeper.address=192.168.1.103"  ports:  - 8083:8083  api-gateway-zuul:  image: api-gateway-zuul:latest  links:  - course-edge-service  - user-edge-service  ports:  - 80:80 |
| docker network create byf-network  docker-compose up -d  docker-compose down |
| curl -L https://get.daocloud.io/docker/compose/releases/download/1.22.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose  chmod +x /usr/local/bin/docker-compose |

### 4.3 镜像仓库

Harbor安装：

<https://www.cnblogs.com/bolingcavalry/p/11565901.html>

<https://storage.googleapis.com/harbor-releases/release-1.9.0/harbor-offline-installer-v1.9.0.tgz>

|  |
| --- |
| [root@www /soft]#tar -zxf harbor-offline-installer-v1.9.0.tgz  [root@www /soft]#ln -s harbor harbor  harbor/ harbor-offline-installer-v1.9.0.tgz  [root@www /soft]#cd harbor  [root@www /soft/harbor]#ll  total 607396  -rw-r--r--. 1 root root 621937309 Sep 11 16:10 harbor.v1.9.0.tar.gz  -rw-r--r--. 1 root root 5805 Sep 11 16:10 harbor.yml  -rwxr-xr-x. 1 root root 5088 Sep 11 16:10 install.sh  -rw-r--r--. 1 root root 11347 Sep 11 16:10 LICENSE  -rwxr-xr-x. 1 root root 1748 Sep 11 16:10 prepare  [root@www /soft/harbor]#vi harbor.yml  Hostname 192.168.1.101  ...  [root@www /soft/harbor]#./install.sh  [Step 0]: checking installation environment ...  Note: docker version: 19.03.2  Note: docker-compose version: 1.22.0  [Step 1]: loading Harbor images ...  b80136ee24a4: Loading layer [==================================================>]  1326179ce332: Loading layer [==================================================>] 47.84MB/47.84MB  Loaded image: goharbor/notary-server-photon:v0.6.1-v1.9.0  [Step 2]: preparing environment ...  prepare base dir is set to /soft/harbor  Generated configuration file: /config/log/logrotate.conf  Generated configuration file: /config/log/rsyslog\_docker.conf  Generated configuration file: /config/nginx/nginx.conf  Generated configuration file: /config/core/env  Generated configuration file: /config/core/app.conf  Generated configuration file: /config/registry/config.yml  Generated configuration file: /config/registryctl/env  Generated configuration file: /config/db/env  Generated configuration file: /config/jobservice/env  Generated configuration file: /config/jobservice/config.yml  Generated and saved secret to file: /secret/keys/secretkey  Generated certificate, key file: /secret/core/private\_key.pem, cert file: /secret/registry/root.crt  Generated configuration file: /compose\_location/docker-compose.yml  Clean up the input dir  [Step 3]: starting Harbor ...  Creating network "harbor\_harbor" with the default driver  Creating harbor-log ... done  Creating harbor-db ... done  Creating redis ... done  Creating harbor-portal ... done  Creating registry ... done  Creating registryctl ... done  Creating harbor-core ... done  Creating harbor-jobservice ...  Creating nginx ... error  Creating harbor-jobservice ... done  f1ac1908d99a5f8f2598285360c017a891b549e3f77799184380728b413a): Bind for 0.0.0.0:80 failed: port is already allocated  ERROR: for proxy Cannot start service proxy: driver failed programming external connectivity on endpoint nginx (1dbaf1ac1908d99a5f8f2598285360c017a891b549e3f77799184380728b413a): Bind for 0.0.0.0:80 failed: port is already allocated  ERROR: Encountered errors while bringing up the project.  [root@www /soft/harbor]#docker-compose down  Stopping nginx ... done  Stopping harbor-log ... done  Removing harbor-jobservice ... done  Removing nginx ... done  Removing harbor-core ... done  Removing registryctl ... done  Removing harbor-db ... done  Removing harbor-portal ... done  Removing registry ... done  Removing redis ... done  Removing harbor-log ... done  Removing network harbor\_harbor  [root@www /soft/harbor]#docker-compose up -d  Creating network "harbor\_harbor" with the default driver  Creating harbor-log ... done  Creating registryctl ... done  Creating registry ... done  Creating redis ... done  Creating harbor-db ... done  Creating harbor-portal ... done  Creating harbor-core ... done  Creating harbor-jobservice ... done  Creating nginx ... done |

|  |
| --- |
| [root@www /soft/harbor]#vi /lib/systemd/system/docker.service  ...  ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --insecure-registry 192.168.1.101  ...  [root@www /soft/harbor]#vi /etc/sysconfig/docker  [root@www /soft/harbor]#systemctl daemon-reload  [root@www /soft/harbor]# systemctl restart docker  [root@www /soft/harbor]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  5d20e7458b65 goharbor/nginx-photon:v1.9.0 "nginx -g 'daemon of…" 23 minutes ago Up Less than a second (health: starting) 0.0.0.0:80->8080/tcp nginx  a6d19143d646 goharbor/harbor-log:v1.9.0 "/bin/sh -c /usr/loc…" 23 minutes ago Up 5 seconds (health: starting) 127.0.0.1:1514->10514/tcp harbor-log  [root@www /soft/harbor]#docker push www.microservice.com/micro-service/openjdk:8-jre  The push refers to repository [www.microservice.com/micro-service/openjdk]  Get https://www.microservice.com/v2/: dial tcp 192.168.1.101:443: connect: connection refused  [root@www /soft/harbor]#docker-compose down  Stopping harbor-jobservice ... done  Stopping nginx ... done  Stopping harbor-core ... done  Stopping harbor-portal ... done  Stopping registryctl ... done  Stopping harbor-db ... done  Stopping registry ... done  Stopping redis ... done  Stopping harbor-log ... done  Removing harbor-jobservice ... done  Removing nginx ... done  Removing harbor-core ... done  Removing harbor-portal ... done  Removing registryctl ... done  Removing harbor-db ... done  Removing registry ... done  Removing redis ... done  Removing harbor-log ... done  Removing network harbor\_harbor  [root@www /soft/harbor]#vi /lib/systemd/system/docker.service  [root@www /soft/harbor]#systemctl daemon-reload  [root@www /soft/harbor]#systemctl restart docker  [root@www /soft/harbor]#docker push www.microservice.com/micro-service/openjdk:8-jre  The push refers to repository [www.microservice.com/micro-service/openjdk]  Get https://www.microservice.com/v2/: dial tcp 192.168.1.101:443: connect: connection refused  [root@www /soft/harbor]#docker-compose down  Stopping nginx ... done  Stopping harbor-jobservice ... done  Stopping harbor-core ... done  Stopping registryctl ... done  Stopping registry ... done  Stopping harbor-db ... done  Stopping harbor-portal ... done  Stopping redis ... done  Stopping harbor-log ... done  Removing nginx ... done  Removing harbor-jobservice ... done  Removing harbor-core ... done  Removing registryctl ... done  Removing registry ... done  Removing harbor-db ... done  Removing harbor-portal ... done  Removing redis ... done  Removing harbor-log ... done  Removing network harbor\_harbor  [root@www /soft/harbor]#vi /lib/systemd/system/docker.service  [root@www /soft/harbor]#docker push www.microservice.com/micro-service/openjdk:8-jre  The push refers to repository [www.microservice.com/micro-service/openjdk]  Get https://www.microservice.com/v2/: dial tcp 192.168.1.101:443: connect: connection refused  [root@www /soft/harbor]#docker tag openjdk:8-jre 192.168.1.101/micro-service/openjdk:8-jre  [root@www /soft/harbor]#docker push 192.168.1.101/micro-service/openjdk:8-jre  The push refers to repository [192.168.1.101/micro-service/openjdk]  b690201c509b: Preparing  6a00eae1d01d: Preparing  cb5d7b073fde: Preparing  8bacec4e3446: Preparing  26b1991f37bd: Preparing  55e6b89812f3: Waiting  denied: requested access to the resource is denied  [root@www /soft/harbor]#docker login 192.168.1.101  Username: Mike  Password:  WARNING! Your password will be stored unencrypted in /root/.docker/config.json.  Configure a credential helper to remove this warning. See  https://docs.docker.com/engine/reference/commandline/login/#credentials-store  Login Succeeded  [root@www /soft/harbor]#docker push 192.168.1.101/micro-service/openjdk:8-jre  The push refers to repository [192.168.1.101/micro-service/openjdk]  b690201c509b: Pushed  6a00eae1d01d: Pushed  cb5d7b073fde: Pushed  8bacec4e3446: Pushed  26b1991f37bd: Pushed  55e6b89812f3: Pushed  8-jre: digest: sha256:79b85320bc3db70dd2a7ab11ba1cf2be4c0401137b6d27e957b9a6ebbd12b3a3 size: 1582  [root@www /soft/harbor]# |

镜像打标签提交至harbor

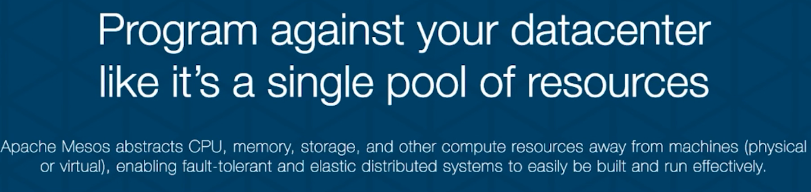
|  |
| --- |
| [root@www /soft/harbor]#docker tag python:3.6 192.168.1.101/micro-service/python:3.6  [root@www /soft/harbor]#docker push 192.168.1.101/micro-service/python:3.6  The push refers to repository [192.168.1.101/micro-service/python]  8a6c8639bc7e: Pushed  e6758511cf63: Pushed  658a7c8576fd: Pushed  69e209e74949: Pushed  3bfeb766f97b: Pushed  ea1227feeccb: Pushed  9cae1895156d: Pushed  52dba9daa22c: Pushed  78c1b9419976: Pushed  3.6: digest: sha256:3046835b0ec654bc86caf7b4214c3d890bf4e6f636a97036c16ae046ef30bb76 size: 2217 |

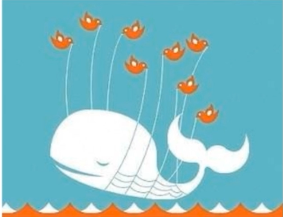
### 4.4服务编排

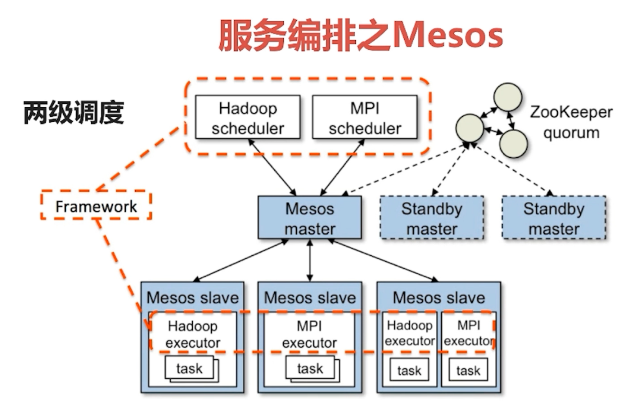


## 第5章 服务编排-Mesos

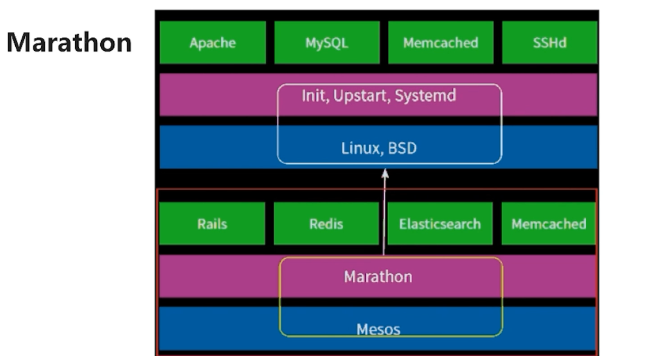
### 5.1了解Mesos

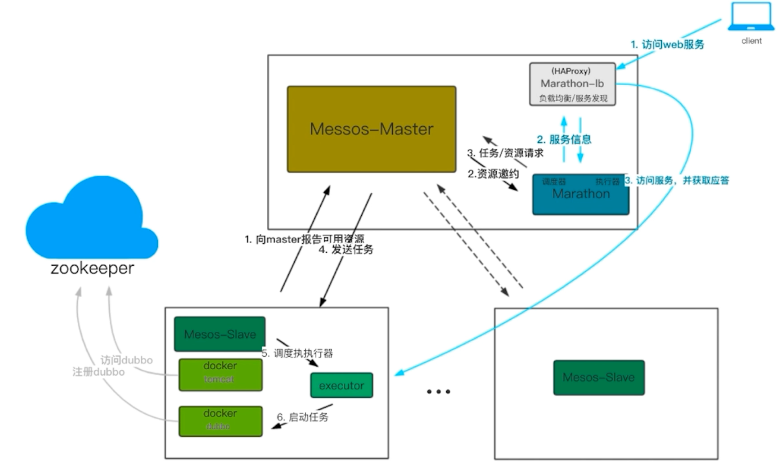








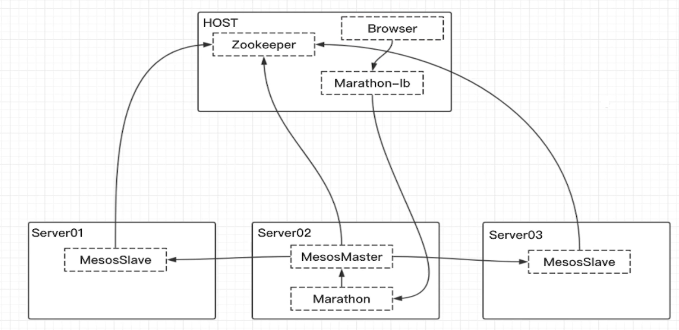








### 5.2Mesos集群结构图



### 5.3使用docker部署Mesos+Zookeeper+Marathon

**导语**

Mesos 抽象资源和调度任务;优秀的集群资源调度平台

Marathon Mesos 上层组件，长期运行保障

Zookeeper Mesos依赖组件

**一、使用docker进行部署**

1、拉取相关镜像

$ docker pull zookeeper

$ docker pull mesosphere/mesos-master:1.5.2

$ docker pull mesosphere/mesos-slave:1.5.2

$ docker pull mesosphere/marathon:latest

2、安装各个服务

(1) Zookeeper 组件

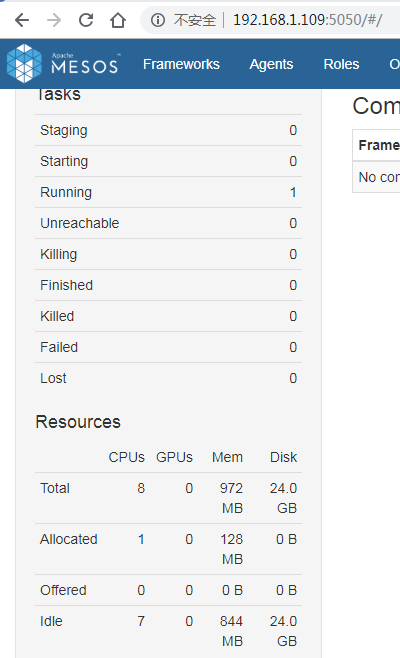
docker run -d --name zookeeper-server --restart=always -p 2181:2181 zookeeper:latest

无法删除：

docker update --restart=no zookeeper-server

(2) Mesos 主节点

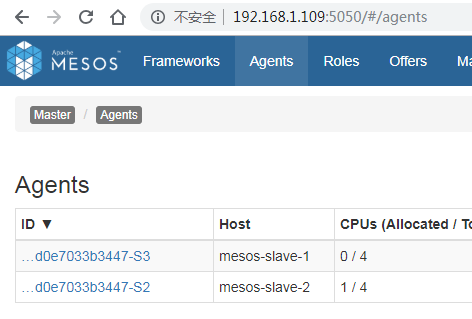
|  |
| --- |
| [root@mesos-master /root]#cat /soft/mesos/mesos-master.sh  #! /bin/bash  docker run -d --net=host \  --name mesos-master \  -e "MESOS\_HOSTNAME=www.mesos-master.com" \  -e MESOS\_PORT=5050 \  -e MESOS\_ZK=zk://192.168.1.103:2181/mesos \  -e MESOS\_QUORUM=1 \  -e MESOS\_REGISTRY=in\_memory \  -e MESOS\_LOG\_DIR=/var/log/mesos \  -e MESOS\_WORK\_DIR=/var/tmp/mesos \  -v "/var/log/mesos:/var/log/mesos" \  -v "/var/tmp/mesos:/var/tmp/mesos" \  mesosphere/mesos-master:1.5.2 |



(3) Mesos 从节点

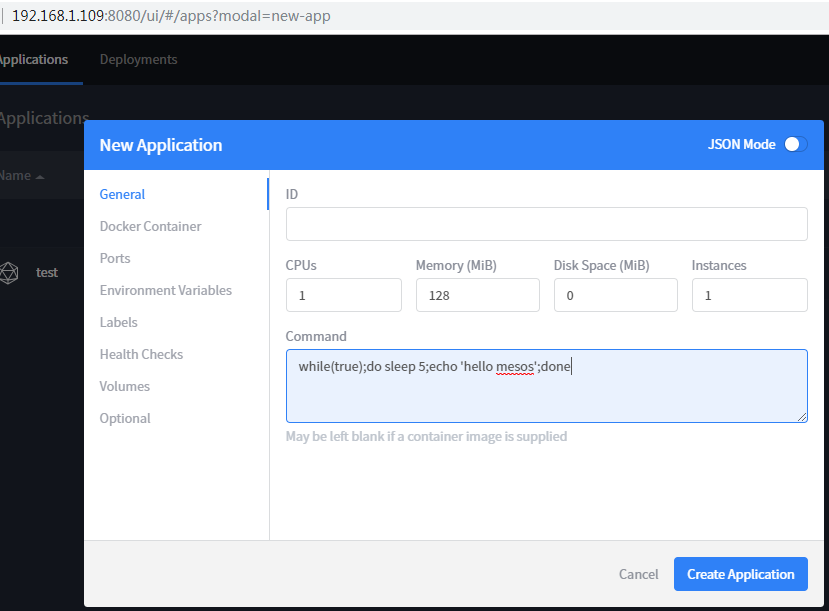
|  |
| --- |
| [root@mesos-slave-1 /soft/mesos]#cat /soft/mesos/mesos-slave.sh  #! /bin/bash  docker run -d --net=host \  --name mesos-slave-1 \  --privileged \  -e MESOS\_PORT=5051 \  -e MESOS\_MASTER=zk://192.168.1.103:2181/mesos \  -e MESOS\_SWITCH\_USER=0 \  -e MESOS\_CONTAINERIZERS=docker,mesos \  -e MESOS\_LOG\_DIR=/var/log/mesos \  -e MESOS\_WORK\_DIR=/var/tmp/mesos \  -e MESOS\_SYSTEMD\_ENABLE\_SUPPORT=false \  -v "/var/log/mesos-sl:/var/log/mesos" \  -v "/var/tmp/mesos-sl:/var/tmp/mesos" \  -v /var/run/docker.sock:/var/run/docker.sock \  -v /sys:/sys \  -v /usr/bin/docker:/usr/bin/docker \  mesosphere/mesos-slave:1.5.2 |

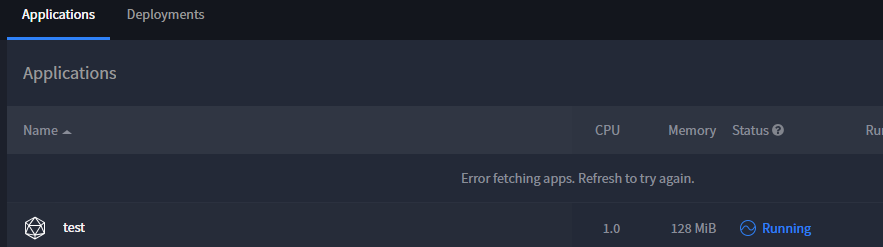
|  |
| --- |
| [root@mesos-slave-2 /soft/mesos]#cat /soft/mesos/mesos-slave.sh  #! /bin/bash  docker run -d --net=host \  --name mesos-slave-2 \  --privileged \  -e MESOS\_PORT=5051 \  -e MESOS\_MASTER=zk://192.168.1.103:2181/mesos \  -e MESOS\_SWITCH\_USER=0 \  -e MESOS\_CONTAINERIZERS=docker,mesos \  -e MESOS\_LOG\_DIR=/var/log/mesos \  -e MESOS\_WORK\_DIR=/var/tmp/mesos \  -e MESOS\_SYSTEMD\_ENABLE\_SUPPORT=false \  -v "/var/log/mesos-sl:/var/log/mesos" \  -v "/var/tmp/mesos-sl:/var/tmp/mesos" \  -v /var/run/docker.sock:/var/run/docker.sock \  -v /sys:/sys \  -v /usr/bin/docker:/usr/bin/docker \  mesosphere/mesos-slave:1.5.2 |

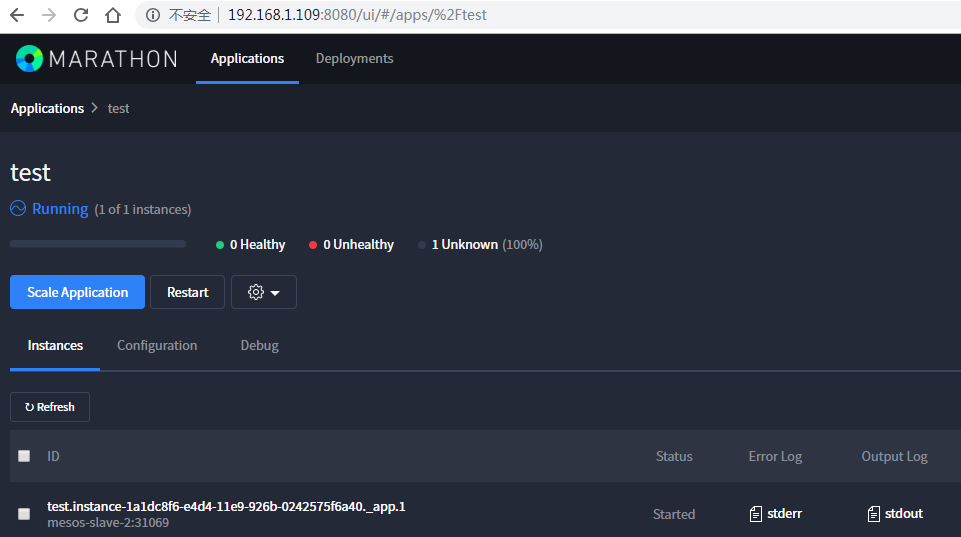


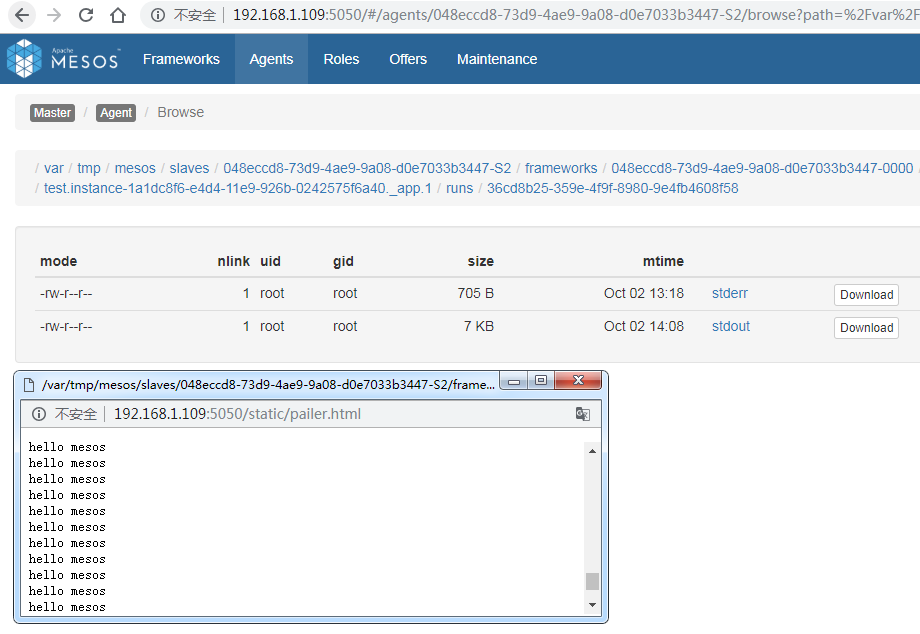
(4) Marathon

|  |
| --- |
| [root@mesos-master /root]#cat /soft/mesos/marathon.sh  #! /bin/bash  docker run --net=host -d \  --name marathon \  mesosphere/marathon \  --master zk://192.168.1.103:2181/mesos \  --zk zk://192.168.1.103:2181/marathon |



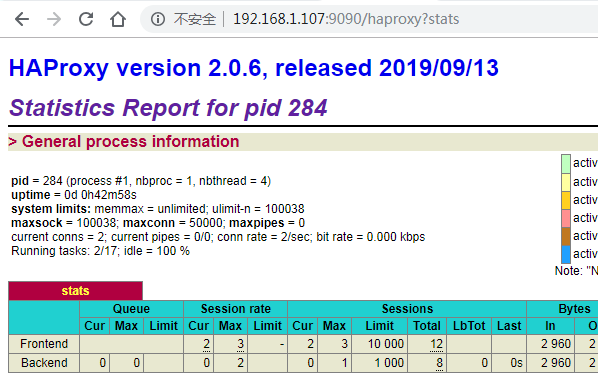






(5)Marathon-lb

|  |
| --- |
| docker pull mesosphere/marathon-lb:latest  [root@mesos-master /soft/mesos]#cat marathon-lb.sh  #! /bin/bash  docker stop marathon-lb  docker rm marathon-lb  docker run -d --net=host \  --name marathon-lb \  -e PORTS=9090 mesosphere/marathon-lb:latest sse --group external --marathon http://192.168.1.107:8080 |



注意：

|  |
| --- |
| [root@mesos-slave /root]#docker images  Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?  [root@mesos-slave /root]#systemctl daemon-reload  [root@mesos-slave /root]#systemctl restart docker.service  [root@mesos-master /soft/mesos]#sudo systemctl enable docker  [root@mesos-slave /root]#docker images  REPOSITORY TAG IMAGE ID CREATED SIZE  [root@mesos-slave /root]#docker pull mesosphere/mesos-master:1.5.2  Error response from daemon: Get https://registry-1.docker.io/v2/mesosphere/mesos-master/manifests/1.5.2: Get https://auth.docker.io/token?scope=repository%3Amesosphere%2Fmesos-master%3Apull&service=registry.docker.io: net/http: request canceled (Client.Timeout exceeded while awaiting headers)  [root@mesos-slave /root]#cd /etc/docker  [root@mesos-slave /etc/docker]#vi daemon.json  [root@mesos-slave /etc/docker]#systemctl restart docker.service  [root@mesos-slave /etc/docker]#docker pull mesosphere/mesos-master:1.5.2  1.5.2: Pulling from mesosphere/mesos-master |
| yum list mesos-master --showduplicates | sort -r  docker search mesos-master |
| 192.168.1.103 www.microservice.com  192.168.1.109 www.mesos-master.com  192.168.1.104 mesos-slave-1  192.168.1.114 mesos-slave-2 |

虚拟机静态ip无法访问百度

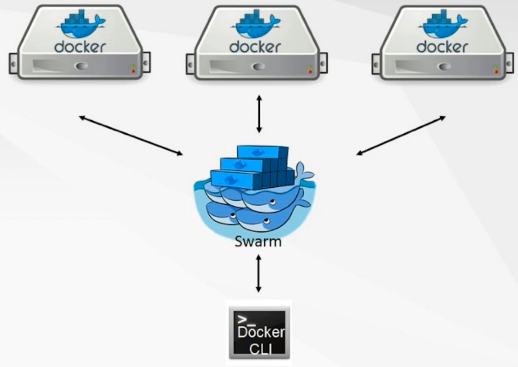
|  |
| --- |
| [root@mesos-master /soft/mesos]#curl www.baidu.com  curl: (6) Could not resolve host: www.baidu.com; Unknown error  [root@mesos-master /soft/mesos]#vi /etc/resolv.conf  nameserver 192.168.1.1 |

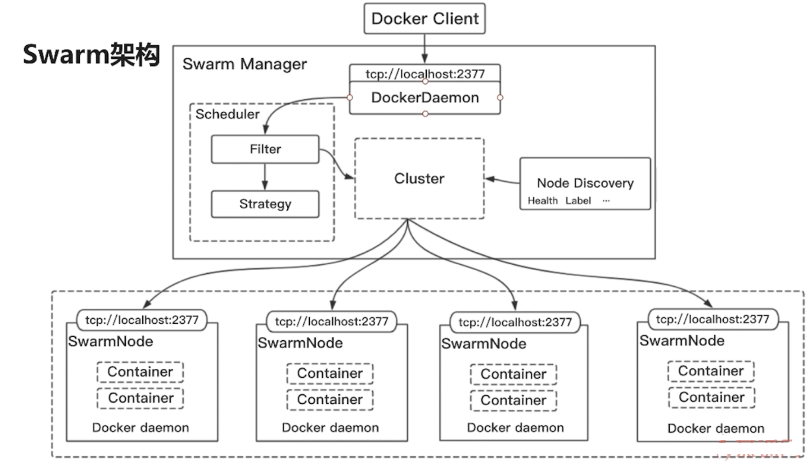
下载不到对应版本的docker镜像

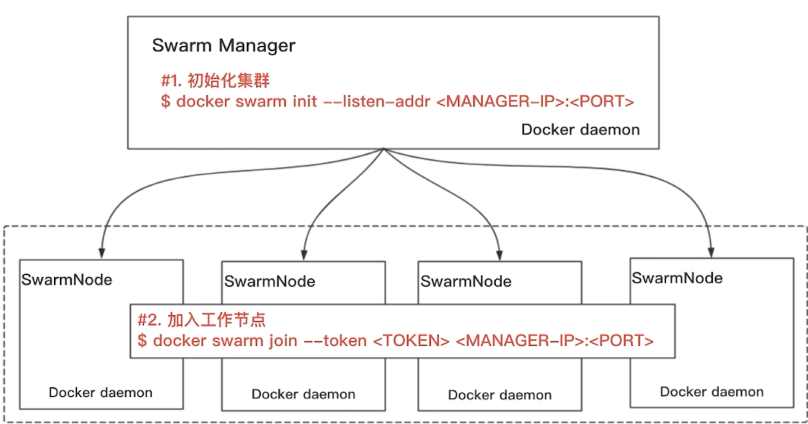
|  |
| --- |
| [root@mesos-master /etc/docker]#docker pull marathon-lb:1.5.2  Error response from daemon: Get https://registry-1.docker.io/v2/: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)  [root@mesos-master /etc/docker]#docker pull mesosphere/marathon-lb |

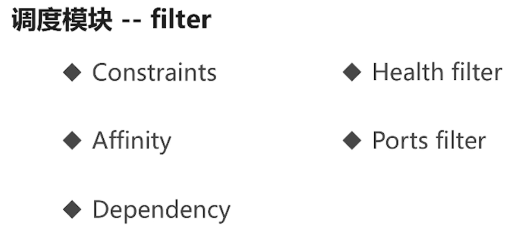
## 第6章 服务编排-DockerSwarm

### 6.1了解Docker Swarm



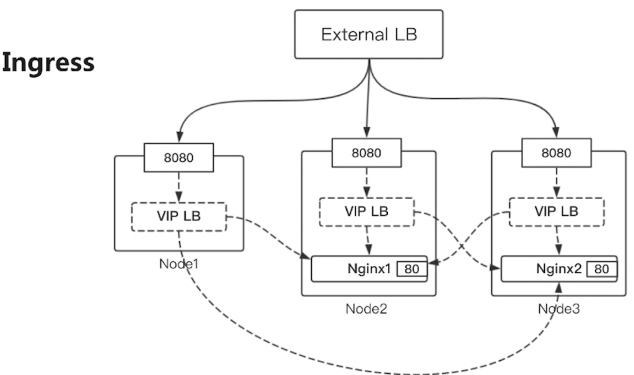


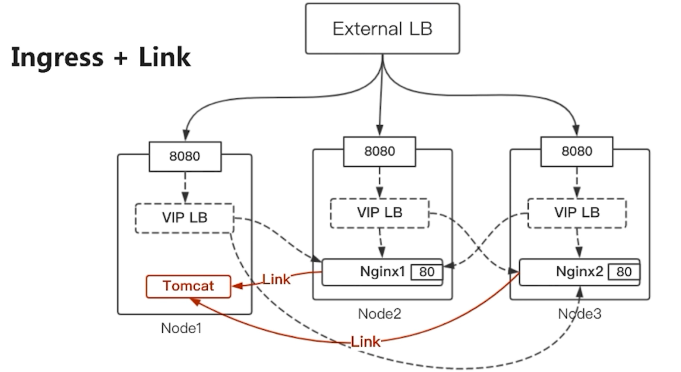


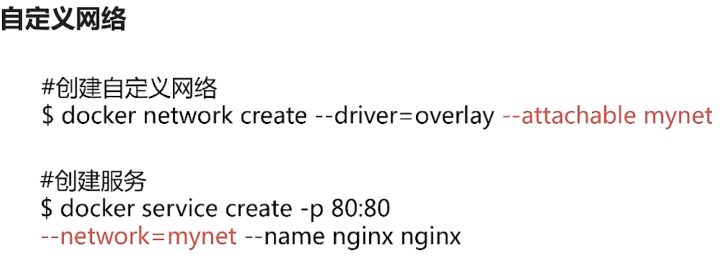


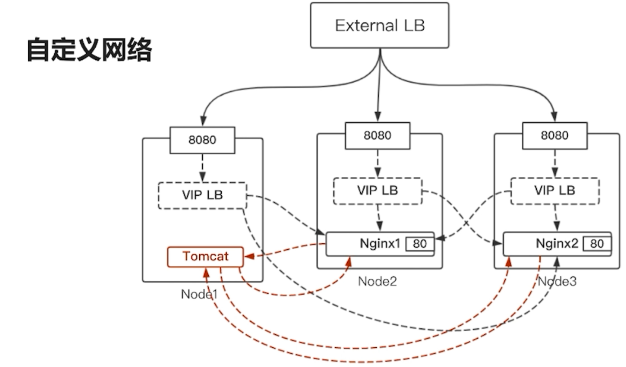




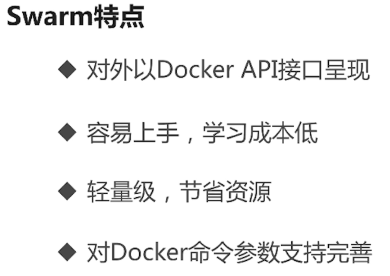












### 6.2 集群环境搭建

简单集群搭建

|  |
| --- |
| [root@mesos-master /root]#docker network ls  NETWORK ID NAME DRIVER SCOPE  319ecddba219 bridge bridge local  2d51bc10e38b host host local  48777f6aaa2a none null local  [root@mesos-master /root]#ifconfig  ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  inet 192.168.1.107 netmask 255.255.255.0 broadcast 192.168.1.255  [root@mesos-master /root]#docker swarm init --advertise-addr 192.168.1.107  Swarm initialized: current node (42vxbjooimyvrmbfb8fer5ktu) is now a manager.  To add a worker to this swarm, run the following command:  docker swarm join --token SWMTKN-1-1aqabl72lcr8wwkg5s5nwln3j006r6jdy3lv3v8uwgxa08tw5d-4zwtumudw01pz1tdjfwyxse93 192.168.1.107:2377  docker swarm join --token SWMTKN-1-3f3n32urnry3qqn6oe1j60nuokbacp7sl8ugq2ohlekrf8111q-79iw1eftfoh3ka023z2fukqlv 192.168.0.107:2377  To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  [root@mesos-master /root]#docker swarm leave --help  Usage: docker swarm leave [OPTIONS]  Leave the swarm  Options:  -f, --force Force this node to leave the swarm, ignoring warnings  [root@mesos-master /root]#docker swarm leave --force  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  [root@mesos-master /root]#docker network ls  NETWORK ID NAME DRIVER SCOPE  319ecddba219 bridge bridge local  13d402ecbc03 docker\_gwbridge bridge local  2d51bc10e38b host host local  lbsgwmsc9byj ingress overlay swarm  48777f6aaa2a none null local |

|  |
| --- |
| [root@mesos-slave-1 /root]#docker swarm join --token SWMTKN-1-1aqabl72lcr8wwkg5s5nwln3j006r6jdy3lv3v8uwgxa08tw5d-4zwtumudw01pz1tdjfwyxse93 192.168.1.107:2377  This node joined a swarm as a worker. |
| [root@mesos-slave-2 /root]#docker swarm join --token SWMTKN-1-1aqabl72lcr8wwkg5s5nwln3j006r6jdy3lv3v8uwgxa08tw5d-4zwtumudw01pz1tdjfwyxse93 192.168.1.107:2377  This node joined a swarm as a worker. |

高可用swarm集群

|  |
| --- |
| [root@mesos-master /root]#docker node ls  ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION  42vxbjooimyvrmbfb8fer5ktu \* mesos-master Ready Active Leader 19.03.2  3e3mop4js837mhwzbws0kbql0 mesos-slave-1 Ready Active 19.03.2  4fwdl42njad9p43ryvrwkl9q6 mesos-slave-2 Ready Active 19.03.2  [root@mesos-master /root]#docker node promote mesos-slave-1  Node mesos-slave-1 promoted to a manager in the swarm.  [root@mesos-master /root]#docker node ls  ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION  42vxbjooimyvrmbfb8fer5ktu \* mesos-master Ready Active Leader 19.03.2  3e3mop4js837mhwzbws0kbql0 mesos-slave-1 Ready Active Reachable 19.03.2  4fwdl42njad9p43ryvrwkl9q6 mesos-slave-2 Ready Active 19.03.2  [root@mesos-master /root]#docker node promote mesos-slave-2  Node mesos-slave-2 promoted to a manager in the swarm.  [root@mesos-master /root]#docker node ls  ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION  42vxbjooimyvrmbfb8fer5ktu \* mesos-master Ready Active Leader 19.03.2  3e3mop4js837mhwzbws0kbql0 mesos-slave-1 Ready Active Reachable 19.03.2  4fwdl42njad9p43ryvrwkl9q6 mesos-slave-2 Ready Active Reachable 19.03.2 |

创建测试服务

|  |
| --- |
| [root@mesos-master /root]docker service create --name test1 --detach=false alpine ping [www.baidu.com](http://www.baidu.com)  [root@mesos-master /root]docker service ls  [root@mesos-master /root] docker service inspect test1  [root@mesos-master /root]docker service logs -f test1 |
| [root@mesos-master /root]#docker service create --name nginx --detach=false nginx  ozbf3fvca9ogxptpviz3ewycs  overall progress: 1 out of 1 tasks  1/1: running [==================================================>]  verify: Service converged  [root@mesos-master /root]#docker service ls  ID NAME MODE REPLICAS IMAGE PORTS  ozbf3fvca9og nginx replicated 1/1 nginx:latest  8bj7xqdqyolz test1 replicated 1/1 alpine:latest |
| docker service update --publish-add 8080:80 --detach=false nginx |







注意：如果发现某一台无法访问，查看该机器上是否运行docker容器

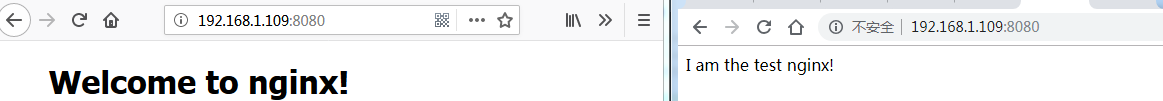
|  |
| --- |
| [root@mesos-slave-1 /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  [root@mesos-master /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  f02c89c5e40b nginx:latest "nginx -g 'daemon of…" 15 minutes ago Up 15 minutes 80/tcp nginx.1.p9hb1v39v43z0yljtuge5ey7r  [root@mesos-slave-2 /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  d7b4b6d2f7cc nginx:latest "nginx -g 'daemon of…" 10 minutes ago Up 10 minutes 80/tcp nginx.3.olxa21dg9phrmem74h76ew4bd  4eef01d13b27 nginx:latest "nginx -g 'daemon of…" 10 minutes ago Up 10 minutes 80/tcp nginx.2.qocbr1q4ekdfdsj28wc5sykm7 |

|  |
| --- |
| [root@mesos-master /root]#docker service scale nginx=3 --detach=false  nginx scaled to 3  overall progress: 3 out of 3 tasks  1/3: running [==================================================>]  2/3: running [==================================================>]  3/3: running [==================================================>]  verify: Service converged  [root@mesos-master /root]#docker service ls  ID NAME MODE REPLICAS IMAGE PORTS  ozbf3fvca9og nginx replicated 3/3 nginx:latest \*:8080->80/tcp  8bj7xqdqyolz test1 replicated 1/1 alpine:latest |

高可用测试

|  |
| --- |
| [root@mesos-master /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  22e2042b887f nginx:latest "nginx -g 'daemon of…" 11 minutes ago Up 11 minutes 80/tcp nginx.3.lnk1gnvk8mwzuh46qqjr1hkz8  effd7a96c8d2 alpine:latest "ping www.baidu.com" 48 minutes ago Up 48 minutes test1.1.17dikb7fi4q36dy6f7gidhdat  [root@mesos-master /root]#docker exec -it 22e2 bash  root@22e2042b887f:/# vi /usr/share/nginx/html/index.html  bash: vi: command not found  root@22e2042b887f:/# echo "I am the test nginx!" > /usr/share/nginx/html/index.html |

更换浏览器，不同会话



|  |
| --- |
| [root@mesos-master /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  22e2042b887f nginx:latest "nginx -g 'daemon of…" 19 minutes ago Up 19 minutes 80/tcp nginx.3.lnk1gnvk8mwzuh46qqjr1hkz8  effd7a96c8d2 alpine:latest "ping www.baidu.com" 56 minutes ago Up 56 minutes test1.1.17dikb7fi4q36dy6f7gidhdat  [root@mesos-master /root]#docker exec -it effd7a96c8d2 sh  / # ping nginx  ping: bad address 'nginx'  / # wget 192.168.1.109:8080  Connecting to 192.168.1.109:8080 (192.168.1.109:8080)  index.html 100% |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 612 0:00:00 ETA  / # cat index.html  <!DOCTYPE html>  <html>  <head>  <title>Welcome to nginx!</title>  <style>  body {  width: 35em;  margin: 0 auto;  font-family: Tahoma, Verdana, Arial, sans-serif;  }  </style>  </head>  <body>  <h1>Welcome to nginx!</h1>  <p>If you see this page, the nginx web server is successfully installed and  working. Further configuration is required.</p>  <p>For online documentation and support please refer to  <a href="http://nginx.org/">nginx.org</a>.<br/>  Commercial support is available at  <a href="http://nginx.com/">nginx.com</a>.</p>  <p><em>Thank you for using nginx.</em></p>  </body>  </html> |
| [root@mesos-master /root]#docker service rm nginx  nginx  [root@mesos-master /root]#docker ps -a  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES |

创建自定义网络模式

|  |
| --- |
| [root@mesos-master /root]#docker network ls  NETWORK ID NAME DRIVER SCOPE  319ecddba219 bridge bridge local  13d402ecbc03 docker\_gwbridge bridge local  2d51bc10e38b host host local  lbsgwmsc9byj ingress overlay swarm  48777f6aaa2a none null local  [root@mesos-master /root]#docker network create -d overlay byf-overlay  rre911vv79n1xifimxyrsk3ku  [root@mesos-master /root]#docker network ls  NETWORK ID NAME DRIVER SCOPE  319ecddba219 bridge bridge local  rre911vv79n1 byf-overlay overlay swarm  13d402ecbc03 docker\_gwbridge bridge local  2d51bc10e38b host host local  lbsgwmsc9byj ingress overlay swarm  48777f6aaa2a none null local  [root@mesos-master /root]#docker service ls  ID NAME MODE REPLICAS IMAGE PORTS  ozbf3fvca9og nginx replicated 3/3 nginx:latest \*:8080->80/tcp  8bj7xqdqyolz test1 replicated 1/1 alpine:latest  [root@mesos-master /root]#docker service rm nginx test1  nginx  test1  [root@mesos-master /root]#docker service ls  ID NAME MODE REPLICAS IMAGE PORTS  [root@mesos-master /root]# |

在自定义网络创建服务

|  |
| --- |
| [root@mesos-master /root]#docker service create --network byf-overlay --name nginx -p 8080:80 --detach=false nginx  mi4vcn54n3r4v05s1hij91xkj  overall progress: 1 out of 1 tasks  1/1: running [==================================================>]  verify: Service converged  [root@mesos-master /root]# docker service ls  ID NAME MODE REPLICAS IMAGE PORTS  mi4vcn54n3r4 nginx replicated 1/1 nginx:latest \*:8080->80/tcp |
| [root@mesos-master /root]#docker service create --network byf-overlay --name alpine --name alpine --detach=false alpine:latest ping www.baidu.com  g74c55kn70dl09hrrvuf5enex  overall progress: 1 out of 1 tasks  1/1: running [==================================================>]  verify: Service converged |

验证网络

|  |
| --- |
| [root@mesos-slave-1 /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  9075913834d9 alpine:latest "ping www.baidu.com" 2 minutes ago Up 2 minutes alpine.1.0o3xt27tfhz583hedkjzp8zz5  [root@mesos-slave-1 /root]#docker exec -it 9075913834d9 sh  / # ping nginx  PING nginx (10.0.0.2): 56 data bytes  64 bytes from 10.0.0.2: seq=0 ttl=64 time=0.507 ms  64 bytes from 10.0.0.2: seq=1 ttl=64 time=0.188 ms  ^C  --- nginx ping statistics ---  2 packets transmitted, 2 packets received, 0% packet loss  round-trip min/avg/max = 0.188/0.347/0.507 ms  / # wget nginx  Connecting to nginx (10.0.0.2:80)  index.html 100% |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 612 0:00:00 ETA  / # cat index.html  <!DOCTYPE html>  <html>  <head> |

|  |
| --- |
| [root@mesos-master /root]#docker service inspect nginx  "EndpointSpec": {  "Mode": "vip",  "Ports": [  {  "Protocol": "tcp",  "TargetPort": 80,  "PublishedPort": 8080,  "PublishMode": "ingress"  }  ]  } |
| [root@mesos-master /root]#docker service inspect alpine  "Endpoint": {  "Spec": {  "Mode": "vip"  },  "VirtualIPs": [  {  "NetworkID": "rre911vv79n1xifimxyrsk3ku",  "Addr": "10.0.0.218/24"  }  ]  } |

（1）通过ip（ingress），服务名（overlay模式）访问

（2）自定义网络模式下，swarm mode会给每一个service分配一个dns entry （网络一定要是overlay），当通过名字访问就会通过docker节点的dns server 然后load balance都对应的服务。

（3）Dnsrr模式

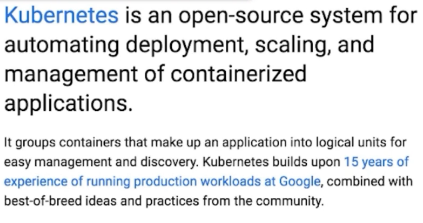
|  |
| --- |
| [root@mesos-master /root]#docker service create --name nginx-b --endpoint-mode dnsrr --detach=false nginx  7vzpe76e4tvfn1zxrne4uojkq  overall progress: 1 out of 1 tasks  1/1: running [==================================================>]  verify: Service converged |
| [root@mesos-slave-1 /root]#docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  211244f329a4 alpine:latest "ping www.baidu.com" 29 minutes ago Up 29 minutes alpine.1.lhfalnhvjbnkauen1sm8zdfmy  [root@mesos-slave-1 /root]#docker exec -it 211 sh  / # ping nginx  PING nginx (10.0.0.2): 56 data bytes  64 bytes from 10.0.0.2: seq=0 ttl=64 time=1.245 ms  64 bytes from 10.0.0.2: seq=1 ttl=64 time=0.162 ms  ^C  --- nginx ping statistics ---  2 packets transmitted, 2 packets received, 0% packet loss  round-trip min/avg/max = 0.162/0.703/1.245 ms  / # ping nginx-b  ping: bad address 'nginx-b' |
| [root@mesos-master /root]#docker service update --network-add byf-overlay --detach=false nginx-b  nginx-b  overall progress: 1 out of 1 tasks  1/1: running [==================================================>]  verify: Service converged |
| / # ping nginx-b  PING nginx-b (10.0.0.10): 56 data bytes  64 bytes from 10.0.0.10: seq=0 ttl=64 time=1.223 ms  64 bytes from 10.0.0.10: seq=1 ttl=64 time=0.993 ms  ^C  --- nginx-b ping statistics ---  2 packets transmitted, 2 packets received, 0% packet loss  round-trip min/avg/max = 0.993/1.108/1.223 ms |

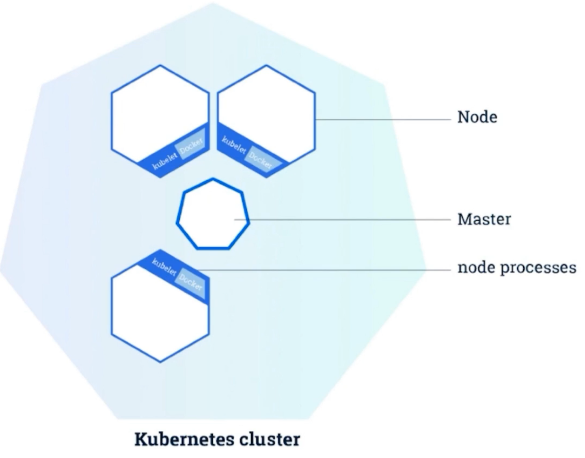
Docker stack创建服务

|  |
| --- |
| [root@mesos-master /soft/docker-swarm]#vi service.yml  version: "3.4"  services:  alpine:  image: alpine  command:  - "ping"  - "www.baidu.com"  networks:  - "byf-overlay"  deploy:  endpoint\_mode: dnsrr  replicas: 2  restart\_policy:  condition: on-failure  resources:  limits:  cpus: "0.1"  memory: 50M  depends\_on:  - nginx  nginx:  image: nginx  networks:  - "byf-overlay"  ports:  - "8080:80"  networks:  byf-overlay:  external: true |
| [root@mesos-master /soft/docker-swarm]#docker stack deploy -c service.yml ms  Creating service ms\_nginx  Creating service ms\_alpine  [root@mesos-master /soft/docker-swarm]#docker service ls  ID NAME MODE REPLICAS IMAGE PORTS  ucr6b2x71krz ms\_alpine replicated 2/2 alpine:latest  bv73spjj8706 ms\_nginx replicated 1/1 nginx:latest \*:8080->80/tcp  [root@mesos-master /soft/docker-swarm]#docker stack ps test  ID NAME IMAGE NODE DESIRED STATE CURRENT STATE ERROR PORTS  81gl5mjdjr4h test\_nginx.1 nginx:latest mesos-slave-2 Running Running less than a second ago  ei321ij49lri test\_alpine.1 alpine:latest mesos-slave-1 Running Running less than a second ago  4pe2mmzfcnqy test\_alpine.2 alpine:latest mesos-master Running Running 16 seconds ago  [root@mesos-master /soft/docker-swarm]#docker stack services test  ID NAME MODE REPLICAS IMAGE PORTS  s9obw8bbybvn test\_alpine replicated 2/2 alpine:latest  vihiudf5eomf test\_nginx replicated 1/1 nginx:latest \*:8080->80/tcp |
| [root@mesos-master /soft/docker-swarm]#docker exec -it 1a34cdb77bad sh  / # ping nginx  PING nginx (10.0.0.16): 56 data bytes  64 bytes from 10.0.0.16: seq=0 ttl=64 time=0.547 ms  64 bytes from 10.0.0.16: seq=1 ttl=64 time=0.290 ms  ^C  --- nginx ping statistics ---  2 packets transmitted, 2 packets received, 0% packet loss  round-trip min/avg/max = 0.290/0.418/0.547 ms  [root@mesos-master /soft/docker-swarm]#docker service inspect test\_alpine  "EndpointSpec": {  "Mode": "dnsrr"  } |
| [root@mesos-master /soft/docker-swarm]#docker stack rm test  Removing service test\_alpine  Removing service test\_nginx |

## 第7章服务编排-Kubernetes

### 7.1了解kubernetes

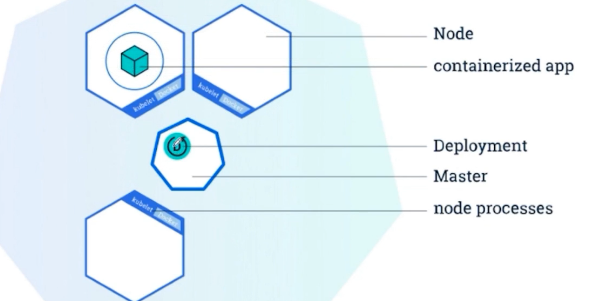




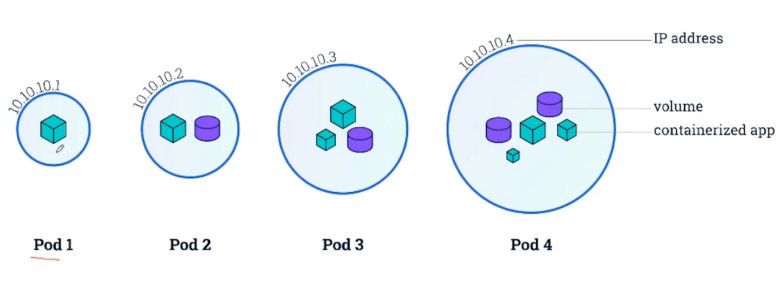
七边形：master

六边形：node

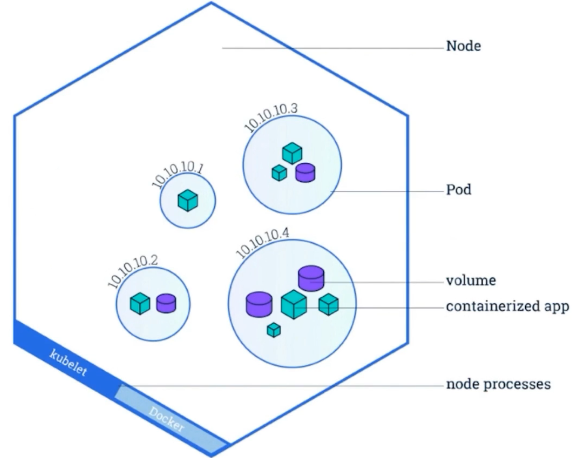
类似于mesos的master节点和worker节点

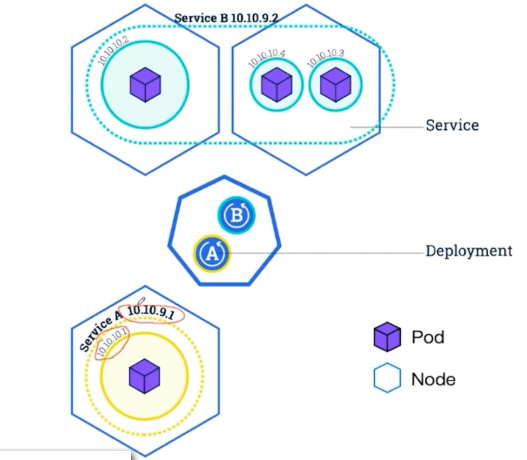


Containerized app



容器和共享存储





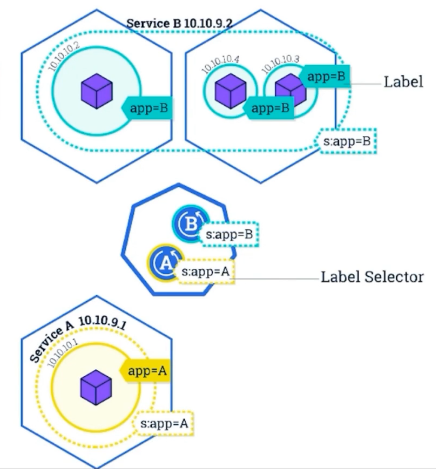
部署Service（通过标签关联Pod）

Service和Pod之间的关系松耦合

Pod：中运行多个容器

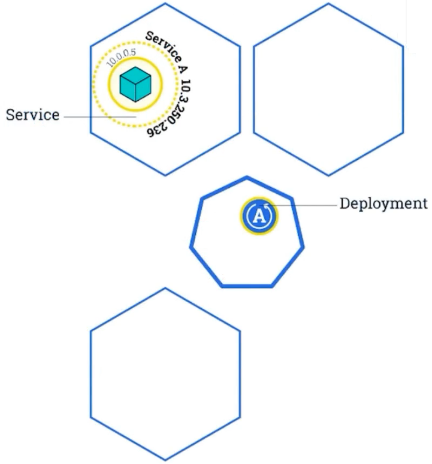
Service：里包含多个Pod

Deployment：可以部署Service或Pod

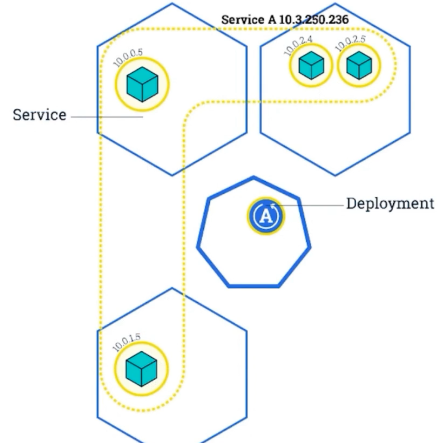


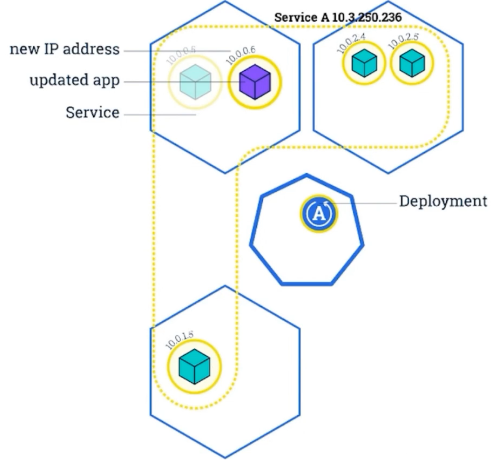
扩容Pod

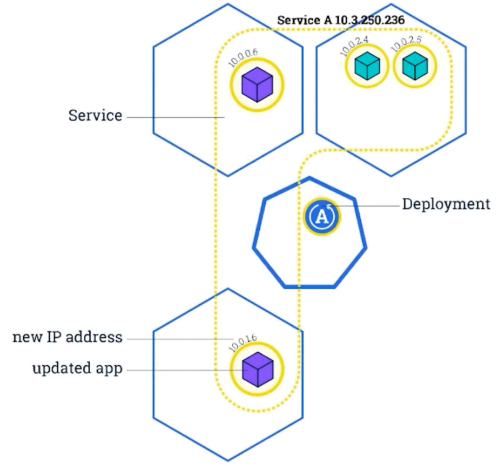
Service划分Pod逻辑组

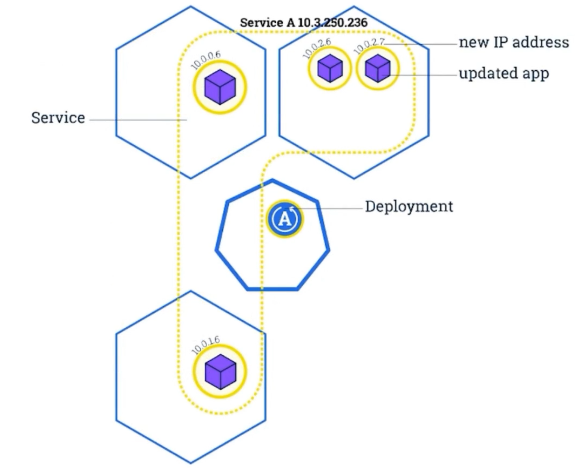


Service通过EndPoint感知Pod

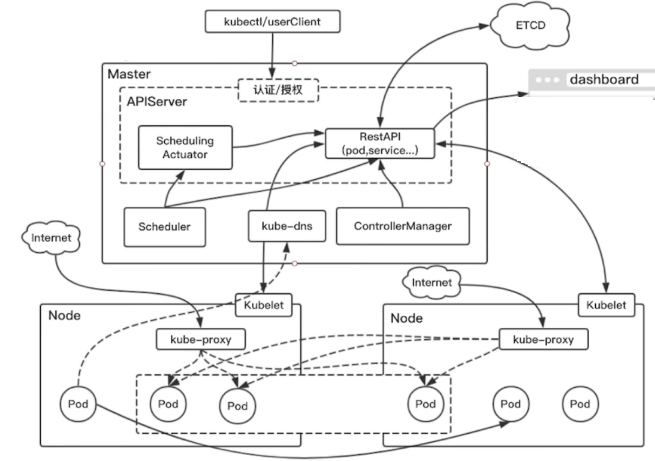








Kubernets整体架构



Master：K8S的核心模块

APISever：提供资源操作唯一入口，并且提供认证/授权，和K8S的访问控制

Kubectl/userClient：通过http请求，以restAPI方式访问APIServer，从而实现对整个集群的控制

ControllerManages：负责维护集群的状态，比如故障检测，扩缩容，滚动更新等待

Scheduler：负责资源调度，按照调度策略把Pod调度到Node节点上

ETCD：一致性存储，集群，Pod,Service信息，所有K8S需要持久化的数据存放位置

Kube-dns：负责整个集群的DNS服务（非必须，可通过名字访问）

DashBorad：提供集群相关数据的展示

Node：工作节点

Kubelet：负责维护当前节点上的生命周期，Volume，网络

Kube-proxy：负责Service，提供内部的服务发现和负载均衡

一个Pod 部署的全过程：