前期准备

环境：阿里云 ESC，系统 CentOS-7.2，1 核 1G，带宽 5M
1.查看防火墙状态：
firewall-cmd --state

2.如果防火墙显示 not running，启动防火墙：
systemctl enable firewalld
systemctl start firewalld

3.防火墙启动成功显示 running。

4.安装 vim：
rpm -qa|grep vim
yum -y install vim*

5.安装 Java JDK-8
查看 Java 版本：
yum -y list java*
选择其中一个版本安装：
yum install java-1.8.0-openjdk*
验证是否成功：
java -version
（使用 yum 安装的默认路径为：/usr/lib/jvm）
设置环境变量：
vim /etc/profile
在文件最后加入如下内容
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.121-0.b13.el7_3.x86_64
PATH=$PATH:$JAVA_HOME/bin
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib.tools.jar
Export JAVA_HOME CLASSPATH PATH
让修改生效(注意半角句点后有空格)：
. /etc/profile
使用 HelloWold 程序验证可用性。

6.安装 Docker
使用官方脚本自动安装：
curl -sSL https://get.docker.com/ | sh

进入 Docker 世界

基础操作

1.安装 Docker
curl -sSL https://get.docker.com/ | sh

2.启动 Docker 引擎
systemctl enable docker
systemctl start docker

3.建立 Docker 用户组：
默认情况下，docker 命令会使用 Unix socket 与 Docker 引擎通讯。而只有 root 用户和 docker 组的用户才可以访问 Docker 引擎的 Unix socket。出于安全考虑，一般 Linux 系统上不会直接使用 root 用户。因此，更好地做法是将需要使用 docker 的用户加入 docker 用户组
查看是否存在 docker 用户组：
grep docker /etc/group
如果存在，将 user 用户加入改组：
usermod -aG docker user
如果不存在，首先需要建立 docker 组：
groupadd docker

4.基础命令：

查看本地所有镜像：
docker images

搜索镜像（如：tomcat）：
docker search tomcat

安装镜像（如：tomcat）：
docker pull tomcat

退出该镜像：
exit

查看正在运行的容器：
docker ps

查看最近终止的容器：
docker ps -l

查看所有容器：
docker ps -a

启动运行容器（javaweb 为容器名）：
docker start javaweb

终止容器：
docker stop javaweb

保存和加载镜像
保存镜像到一个 tar 包：
docker save ubuntu:latest > ubuntu_save.tar
加载一个 ta 包格式的镜像(之后的镜像信息都在，包括名字)：
docker load < ubuntu_save.tar

运行 CentOS 镜像

搭建 JavaWeb 运行容器（测试成功！可运行使用）

就是通过 CentOS 镜像，创建一个 JavaWeb 运行环境的容器。
1.进入 CentOS 镜像：
docker run -i -t -v /opt/software/:/mnt/software/ centos /bin/bash
其中，-i 表示以交互模式运行容器。
-t 表示进入容器后会进入其命令行。
-v 表示指定容器挂载的目录，格式：<宿主机目录>:<容器目录>

通过挂载宿主和镜像的目录，可以共享文件。

2.在镜像中安装 Tomcat（过程：略）

3.解压到本地：/usr/local/tomcat

4.编写运行脚本：
vim /root/run.sh
内容如下：
#!/bin/bash
source ~/.bashrc
sh /usr/local/tomcat/bin/catalina.sh run

5.设置运行的环境变量：
vim ~/.bashrc

在最后加入如下内容：
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.121-0.b13.el7_3.x86_64
export PATH=$PATH:$JAVA_HOME

6.使该环境变量生效：
source ~/.bashrc

7.为脚本添加执行权限：
chmod u+x /root/run.sh

8.执行命令退出容器
exit

9.查看当前运行的容器列表：
docker ps -a
记下 CentOS 镜像的容器 ID：3b826adbe0b5

10.根据该容器 ID 创建一个新的 JavaWeb 镜像：
docker commit 3b826adbe0b5 hankchan/javaweb:0.1
其中，3b826adbe0b5 是容器 ID，hankchan/javaweb:0.1 是生成的镜像名。

11.基于 JavaWeb 镜像运行容器（创建一个新的容器，名为 javaweb）：
docker run -d -p 58080:8080 --name javaweb hankchan/javaweb:0.1 /root/run.sh
其中，-d 表示以“守护模式”执行/root/run.sh 脚本，此时 Tomcat 控制台不会出现在输出终端上。
-p 表示宿主机与容器的端口映射，此时将容器内部的 8080 端口映射为宿主机的 58080 端口，这样就向外界暴露了 58080 端口，可通过 Docker 网桥来访问容器内部的 8080 端口了。
--name：表示容器名称，用一个有意义的名称命名即可，如这里使用 javaweb

容器运行时，进入容器：
docker exec -i -t <containerID> bash

12.开放宿主机的 58080 端口，就可以通过宿主机的该端口访问到该容器中运行到 Tomcat 了。

13.安装 Nginx 服务器：
首先需要安装编译工具及库文件：
yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel
安装 PCRE：
wget http://downloads.sourceforge.net/project/pcre/pcre/8.35/pcre-8.35.tar.gz
tar zxvf pcre-8.35.tar.gz
cd pcre-8.35
编译安装：

./configure
make && make install
验证版本：
pcre-config --version

下载 Nginx：
wget http://nginx.org/download/nginx-1.9.9.tar.gz
解压：
zar zxvf nginx-1.9.9.tar.gz

编译安装：
cd nginx-1.9.9
./configure --prefix=/usr/local/nginx --with-http_stub_status_module --with-http_ssl_module --with-pcre=/usr/local/pcre-8.35
其中，--prefix 是编译 nginx 后的目录地址；另外注意要指定 pcre 的安装目录。
make
make install

配置 nginx.conf 配置文件：（略）
检查配置文件正确性：
/usr/local/nginx/sbin/nginx -t
重新载入配置文件：
/usr/local/nginx/sbin/nginx -s reload

启动 Nginx：
/usr/local/nginx/sbin/nginx
重启 Nginx：
/usr/local/nginx/sbin/nginx -s reopen
停止 Nginx：
/usr/local/nginx/sbin/nginx -s stop

经过考虑，决定使用 Nginx 镜像代替机器安装 Nginx 服务（配置监听端口都为 80）。

安装 Nginx 镜像：
docker pull nginx

启动 Nginx
docker run --name webserver -d -p 80:80 nginx

(可忽略)使用 Dockerfile 定制镜像

Dockerfile 是一个文本文件，其内包含了一条条的指令(Instruction)，每一条指令构建一层，因此每一条指令的内容，就是描述该层应当如何构建。

定制 Nginx 示例

编写 Dockerfile 文件

1.在一个空白目录中，建立一个文本文件，命名为：Dockerfile：
mkdir mynginx
cd mynginx
touch Dockerfile
其中内容为：
FROM nginx
RUN echo '<h1>Hello, Docker!</h1>' > /usr/share/nginx/html/index.html
这个 Dockerfile 很简单，一共就两行。涉及到了两条指令，FROM 和 RUN。

FROM 命令：指定基础镜像，在 Dockerfile 中，FROM 是必备的，并且必须是第一条指定。
RUN 命令：是用来执行命令行命令的，有两种格式。
1. shell 格式：RUN <命令>
2. exec 格式：RUN [“可执行文件”, “参数 1”, “参数 2”]
由于每条指令都会建立一层，如果将层次浪费在无意义的操作，很容易出问题，也太臃肿。正确的写法应该是类似以下这种：

```
FROM debian:jessie

RUN buildDeps='gcc libc6-dev make' \

    && apt-get update \

    && apt-get install -y $buildDeps \

    && wget -O redis.tar.gz "http://download.redis.io/releases/redis-3.2.5.tar.gz" \

    && mkdir -p /usr/src/redis \

    && tar -xzf redis.tar.gz -C /usr/src/redis --strip-components=1 \

    && make -C /usr/src/redis \

    && make -C /usr/src/redis install \

    && rm -rf /var/lib/apt/lists/* \

    && rm redis.tar.gz \

    && rm -r /usr/src/redis \

    && apt-get purge -y --auto-remove $buildDeps
```

一条命令完成一个目的。不要拆分开成为多个 RUN 语句。
Dockerfile 支持 Shell 类的行尾添加 \ 的命令换行方式，以及行首 # 进行注释的格式。

构建镜像

在 Dockerfile 所在目录下执行构建命令：
docker build -t nginx:v2 .
其中，nginx:v2 是新生成的镜像名称和版本。注意语句最后是空格加句点结尾！

其他构建方法

从标准输入中读取上下文压缩包进行构建：
docker build - < context.tar.gz
如果发现标准输入的文件格式是 gzip、bzip2 以及 xz 的话，将会使其为上下文压缩包，直接将其展开，将里面视为上下文，并开始构建。

部署基于 EDAS 的应用

部署内容说明

1. EDAS 配置中心服务：edas-config-center（jmenv.tbsite.net）
　　由于 edas 配置中心需要单独占用 8080 端口，我们选择一个 CentOS 镜像作为 edas-config-center 的运行容器，端口映射设定为：58080:8080。采用另外一种实现方式，即：直接使用本地内网其他机器上的配置中心（10.148.16.72）即可。

2. 服务消费者 docker-web 应用
　　使用基础 CentOS 镜像制作而成的容器作为 docker-web 应用运行环境，端口映射设定为：58081:8080。
　　该容器中需要配置 Ali-Tomcat 并且修改 hosts 加入 edas-config-center 的地址。

3. 服务提供者 docker-service 应用和 docker-service2 应用
　　使用基于 CentOS 的镜像生成两个容器，用于运行这两个服务提供者应用，端口映射设定为：58082:8080 和 58083:8080。
　　注意：war 包应用（包括消费者和提供者）基于 EDAS-HSF，需要运行在 AliTomcat 环境下，所以自定义一个包含 AliTomcat 的基础运行环境镜像，用于部署应用。每个容器启动时，必须将配置中心 IP 映射加入 hosts（直接修改容器的

hosts 文件不管用，启动时使用--add-host=[host:ip]命令）。

4. Nginx 服务器

使用官方提供的 Nginx 镜像作为运行容器即可。在其中配置上述应用的反向代理。也可以不使用容器运行 Nginx，后面看具体情况。

5. 为每个容器分配一个虚拟 IP

由于 EDAS-HSF 注册中心和服务提供者消费者等基于路由，且容器基于 CentOS 镜像，那么就为每个容器分配一个虚拟 IP。Docker 默认会为容器分配动态子网 IP，使用端口映射可以从外访问容器。可以不手动分配 IP。

war 包应用上传方法：

rz

如果没有 rz 命令，先执行安装：

yum install lrzsz -y

在 XShell 运行 rz 命令，会弹出窗口，选择文件即可上传到当前目录中。

端口操作

开放端口：

firewall-cmd --add-port=8080/tcp

开放永久端口：

firewall-cmd --zone=public --add-port=8080/tcp --permanent

firewall-cmd --reload

关闭端口：

firewall-cmd --remove-port=8080/tcp

一些问题

安装试运行 edas-config-center：

在使用 Open JDK 运行时，一直报错提示 JDK 版本无法达到要求。

改用 Java JDK 不存在该问题。

安装 JavaJDK：

1.下载 JavaJDK 的 rpm 包：

下载地址：http://www.oracle.com/technetwork/java/javase/downloads/index.html

2.将 rpm 包拉进系统中，执行安装：

rpm -ivh jdk-8u121-linux-x64.rpm

JDK 默认安装在/usr/java 中。

3.配置环境变量：

vi /etc/profile

在文本最后加入以下内容：

export JAVA_HOME=/usr/java/jdk1.8.0_121
export JRE_HOME=/usr/java/jdk1.8.0_121/jre
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
4.保存并退出，然后使修改生效：
source /etc/profile
5.测试是否成功!

自定义 Ali-Tomcat 容器镜像

由于前面的尝试失败，现在尝试两种方式（第二种为 Dockerfile 方式）定制一个包含 JavaJDK 和 Ali-Tomcat 的镜像。用于后续部署基于 EDAS-HSF 的 web 应用。

使用非 Dockerfile 方式（镜像：alitomcat:1.2）

下面开始基于 CentOS 镜像定制 Alitomcat 镜像。

安装 JDK

1.使用交互模式进入 CentOS 镜像
docker run -i -t -v /opt/software/:/mnt/software/ centos /bin/bash
2.安装配置 JavaJDK 环境（CentOS 原始镜像不会有 JDK）
rpm -ivh jdk1.8.0_121.rpm
JDK 默认安装在/usr/java 中。
3.配置环境变量：
vi /etc/profile
在文本最后加入以下内容：
export JAVA_HOME=/usr/java/jdk1.8.0_121
export JRE_HOME=/usr/java/jdk1.8.0_121/jre
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
4.保存并退出，然后使修改生效：
source /etc/profile
5.测试是否成功!

安装 AliTomcat

1.将 taobao-tomcat 压缩包解压到/usr/local/目录下
tar -zxvf taobao-tomcat.tgz
2.删除压缩包
rm -rf taobao-tomcat.tgz
3.将 taobao-tomcat 重命名为：alitomcat
mv taobao-tomcat/ alitomcat

配置运行环境变量

1.设置环境变量：
vi ~/.bashrc
加入以下内容，保存退出：
export JAVA_HOME=/usr/java/jdk1.8.0_121
export PATH=$PATH:$JAVA_HOME/bin
2.使修改生效：
source ~/.bashrc
编写运行脚本
1.创建脚本文件：
touch /root/run.sh
vi /root/run.sh
加入内容如下：
#!/bin/bash
source ~/.bashrc
sh /usr/local/alitomcat/bin/startup.sh run
2.为脚本添加执行权限：
chmod u+x /root/run.sh
退出当前镜像容器
exit
创建新镜像
1.获取刚刚运行的容器 ID：
docker ps -a
容器 ID：a78d9b71d4e8
2.创建镜像：
docker commit a78d alitomcat:1.0
3.查看镜像库：
docker images
可以看到最新创建的 alitomcat 的 1.0 版本的镜像
测试该镜像的容器实例可用性
1.守护模式(-d)运行一个容器（名为：webapp_1）：
docker run -d -p 58081:8080 --name webapp_1 alitomcat:1.0 /root/run.sh
2.查看运行中的容器：
docker ps
发现该容器停止了！！
查看日志：
docker logs <containerID>
发现容器中的 Ali-Tomcat 是正常启动过的。
原来，对于容器来说，容器中的应用如果不再运行，容器会自动退出。因为 AliTomcat 默认是没有“欢迎首页”的，所以启动后容器认为该应用不再运行，自动退出。
解决办法有两种：一是在 AliTomcat 的应用部署路径（alitomcat/deploy/）中新建一个 index.html 页面；另一个是通过 catalina.sh 脚本启动 tomcat。
重新尝试启动容器，发现容器依旧在运行。
docker start <containerID>

部署方法
1.将宿主机上的 war 包地址映射到容器中的地址：
docker run -d -p 58082:8080 -v /hankchan101/war:/usr/local/alitomcat/deploy --name docker-service alitomcat:1.2 /root/run.sh

使用 Dockerfile 方式自定义镜像

。。。

网络虚拟 IP 分配

1.安装网桥设置工具
yum install bridge-utils

在安装 Docker 后，可以看到多出了一个 docker0 的网桥：
brctl show
测试端口连通性
nc -z -w 1 180.97.33.107 8080

向容器 hosts 中加入配置，在 docker run 中附加命令：
--add-host=jmenv.tbsite.net:<ip>
如：
docker run --add-host=jmenv.tbsite.net:10.148.16.72 -d -p 58081:8080 -v /opt/software/test/serivce/:/usr/local/alitomcat/deploy --name service alitomcat:1.0 /root/run.sh

配置 EDAS-Config-Center

主要难点是关于 hosts 文件映射的 ip 问题。
如果要在宿主机直接运行 edas-config-center，并将 hosts 的域名指向宿主机，无法正常被应用容器映射到。另一个思路是将 edas-config-center 容器化，然后为每个容器分配虚拟 IP（或者静态 IP），如此就能够正常使用配置中心。

最终解决方案

直接利用其它机器部署 EDAS 配置中心，如之前在 10.148.16.72 上已经部署过，直接使用就好。不要将应用和配置中心放在同一台机器。

容器编排