

# 基于阿里云ECS与ACR的容器镜像管理实验

---

## 实验要求

1. 获取阿里云ECS服务器，基于 ECS 安装Docker引擎，同时开通阿里云容器镜像服务ACR。
2. 练习ECS基础操作，包括远程连接、重置密码、创建普通用户、禁用 root 账户SSH远程登录等。
3. 练习Docker常用命令，包括拉取镜像、启动容器、查看容器/镜像、进入容器、删除容器等。
4. 撰写Dockerfile文件，实现基于Dockerfile构建Docker容器镜像。
5. 应用阿里云容器镜像服务ACR，基于 ECS 实现自定义容器镜像的推送与拉取。

## 实验环境

1. 阿里云云服务器ECS（Ubuntu Server）： Docker
2. 阿里云容器镜像服务（ACR）

## 实验步骤

### 构建实验环境

#### 准备云服务器ECS

##### 购买阿里云ECS

登录阿里云，访问云服务器ECS产品首页（<https://www.aliyun.com/product/ecs>），点击[立即购买]，根据页面提示，按需选择相关配置后付费购买。其中操作选择Ubuntu，安全组配置默认应仅允许开启SSH（22）与HTTP（80）与HTTPS（443）端口。

##### 阿里云高校计划

通过[阿里云高校计划（<https://university.aliyun.com/plan/student>）]，阿里云提供免费获取ECS的途径。

#### 在ECS安装Docker引擎

访问Docker官方页面（<https://docs.docker.com/engine/install/ubuntu/>），根据提示在阿里云服务器ECS上安装Docker。本实验系列设定ECS采用Ubuntu系统。 点击实例 -> 远程连接 -> 立即登录

```
WORKBENCH 文件 编辑 视图 实例 会话 功能 帮助 数据库 终端助手 简体中文
1_root@iZ2vcfk3gmwxxap6tnf... x
西南1(成都)
2_root@iZ2vcfk3gmwxxap6tnf5Z: ~ x
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-122-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Tue Oct 15 08:00:38 PM CST 2024

System load:  0.03          Processes:    122
Usage of /:   6.4% of 39.01GB Users logged in:  0
Memory usage: 15%          IPv4 address for eth0: 172.
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Welcome to Alibaba Cloud Elastic Compute Service !

Last login: Tue Oct 15 20:00:39 2024 from 100.104.65.255
root@iZ2vcfk3gmwxxap6tnf5Z:~#
```

创建成功。

😭😭😭:exploding\_head: 安装Docker一直出现网络问题，一直无法安装成功。换了个阿里云的镜像源，试过挂梯子，也都还是不行。

### 💡 问题解决

1. 经过复盘排查，也跟其他同学的每一步进行对比过后，发现是因为在前面购买阿里云的ECS时，选择了一年的套餐而不是三个月的，问了客服才知道这两个的区别就包括三个月的有公网IP和带宽，而一年的没有公网IP和带宽，所以在安装Docker时，需要先开通公网IP和带宽。
2. 由于我现在的服务器公网带宽是0，所以无法通过公网连接，参考这个文档将带宽调整大于0，系统才会分配公网IP使用。 <https://help.aliyun.com/zh/ecs/user-guide/modify-the-bandwidth-configurations?spm=a2c4g.11186623.0.i4>
3. 此时Docker就顺利安装成功了。
4. 但是在检查Docker版本时证明安装成功之后，试着拉去一个hello-world的镜像时又出现了问题。如下图所示：

```
root@iZ2vcfk3gmwxxap6tnf5Z:~# docker --version
Docker version 27.3.1, build ce12230
root@iZ2vcfk3gmwxxap6tnf5Z:~# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
docker: 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).
See 'docker run --help'.
```

★本问题解决方法：通过修改`daemon.json`文件来配置多个国内镜像源，可以有效提高Docker镜像拉取的成功率。配置多个镜像源（如网易、腾讯云、阿里云等）会增加拉取镜像的稳定性，特别是在网络状况不佳时。

操作步骤：

1. 编辑`/etc/docker/daemon.json`文件，添加国内的镜像源。
2. 保存并退出编辑器（Ctrl + X, 然后 Y 再 Enter）。
3. 重新加载Docker守护进程配置：

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

4. 测试拉取镜像：

```
sudo docker pull hello-world
```

这样就可以从配置的国内镜像源中拉取镜像，避免网络问题导致的拉取失败。

```
root@iZt1z1z1z1z1z1z1z1z1Z:~# sudo docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

成功。

基于阿里云容器镜像服务ACR配置镜像仓库

登陆阿里云，访问容器镜像服务ACR产品首页（<https://www.aliyun.com/product/acr>），点击[管理控制台]进入实例列表，了解容器镜像服务个人版与企业版的主要差异。本实验中，创建使用个人实例即可。具体使用时，需完成三个基本步骤：

- 1. 设置访问密码。用户名：阿里云账户全名。密码：8-32位，必须包括字母、符号或数字中的至少两项
- 2. 创建命名空间 全局命名空间(不可重复) 长度2-30位，可填写小写英文字母、数字、可使用的分隔符(分隔符不能在首位和末位)
- 3. 创建镜像仓库。仓库信息:选择上一步创建的[命名空间]，设置[仓库名称]-本实验使用名称 npu(Nginx-Python-Ubuntu)，撰写[仓库摘要]，默认设置仓库为私有而非公开。◦代码源:本实验选择[本地仓库]，即[通过命名行推送镜像到镜像仓库]，后续实验学习使用流水线时，则选择真实代码源。

← npu

西南1（成都） | 私有 | 本地仓库 | ✓ 正常

部署

基本信息

触发器

镜像版本

基本信息

编辑

仓库名称	npu 复制	公网地址	npz-y48vqfhwj3qazsh.cn-shanghai.personal.aliyuncs.com 复制
仓库地域	西南1（成都）	专有网络	vpc-y48vqfhwj3qazsh-vpc.cn-shanghai.personal.aliyuncs.com 复制
仓库类型	私有	摘要	云计算与中间件实验
代码仓库	无		

操作指南

制品描述

1. 登录阿里云Docker Registry

```
1. docker login --username=000000000000 npz-y48vqfhwj3qazsh.cn-shanghai.personal.aliyuncs.com
```

用于登录的用户名为阿里云账号全名，密码为开通服务时设置的密码。  
您可以在访问凭证页面修改凭证密码。

2. 从Registry中拉取镜像

```
1. docker pull npz-y48vqfhwj3qazsh.cn-shanghai.personal.aliyuncs.com/nginx_1/nginx [镜像版本号]
```

3. 将镜像推送到Registry

```
1. docker login --username=000000000000 npz-y48vqfhwj3qazsh.cn-shanghai.personal.aliyuncs.com
2. docker tag [ImageId] npz-y48vqfhwj3qazsh.cn-shanghai.personal.aliyuncs.com/nginx_1/nginx [镜像版本号]
3. docker push npz-y48vqfhwj3qazsh.cn-shanghai.personal.aliyuncs.com/nginx_1/nginx [镜像版本号]
```

练习ECS基础操作

云服务器新手上路

完成阿里云起实验室实验项目ECS云服务器新手上路

<https://developer.aliyun.com/adc/scenario/410e5b6a852f4b4b88bf74bf4c197a57>，主要开展控制台管理实践。包括重置实例密码，远程连接等。

### 创建配置普通用户

在本地系统基于 SSH 访问 ECS 公网地址，使用 root 账号登陆，执行 useradd 命令添加普通用户：使用 passwd 命令设置/重置用户密码。

```
# -m 创建用户主目录; -s 设定 Shell
sudo useradd -m -s /usr/bin/bash <username>
# 设置登陆密码
passwd <username>
# 编辑 /etc/group, 将指定用户加入 sudo 组, 提供 sudo 权限
sudo usermod -G sudo <username>
# 编辑 /etc/group, 将制定用户加入 docker 组, 提供 docker 权限
sudo usermod -G docker <username>
```

完成上述配置后，使用新建用户进行 SSH 登陆验证，且确认 `sudo su` 命令可以正常切换到 root 用户。

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# sudo useradd -m -s /usr/bin/bash beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# passwd beryl
New password:
Retype new password:
passwd: password updated successfully
root@iZ2vcfk3gmwgxxap6tntf5Z:~# sudo usermod -G sudo beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# sudo usermod -G docker beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~#
```

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

```
Welcome to Alibaba Cloud Elastic Compute Service !
```

```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ sudo su
[sudo] password for beryl:
beryl is not in the sudoers file. This incident will be reported.
```

发现beryl并没有在sudo组里，所以没有sudo权限。★**问题解决:** 使用 root 用户添加 beryl 到 sudoers 文件

1. **登录为 root 用户:** 如果你可以使用 root 用户登录，运行以下命令切换到 root 用户:

```
su -
```

然后输入 root 用户的密码。

2. **编辑 `sudoers` 文件**：在 `root` 用户模式下，运行以下命令打开 `sudoers` 文件：

```
visudo
```

这会打开一个安全的编辑器，用于编辑 `sudoers` 文件。

3. **添加用户到 `sudoers` 文件**：在文件中找到类似 `# User privilege specification` 的部分，添加以下行，将 `beryl` 用户加入到 `sudoers`：

```
beryl ALL=(ALL:ALL) ALL
```

这会赋予 `beryl` 用户执行任何 `sudo` 命令的权限。

4. **保存并退出**：保存修改（在 `visudo` 中通常按 `Ctrl+X`，然后按 `Y` 确认保存），然后退出编辑器。

5. **重新登陆并验证**

```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ sudo su
[sudo] password for beryl:
root@iZ2vcfk3gmwgxxap6tntf5Z:/home/beryl# whoami
root
root@iZ2vcfk3gmwgxxap6tntf5Z:/home/beryl#
```

成功。

## 禁止 `root` 远程登录

编辑文件 `/etc/ssh/sshd_config`，添加配置 `PermitRootLogin no`，然后重启 SSH 服务。

可以使用以下命令重启 SSH 服务：

```
$ sudo service sshd restart
#或者执行 sudo systemctl restart sshd
```

```
UseDNS no
SyslogFacility AUTHPRIV
PermitRootLogin no
PasswordAuthentication yes
```

```
root@iZ2vcfk3gmwgxxap6tntf5Z:/home/beryl# sudo nano /etc/ssh/sshd_config
root@iZ2vcfk3gmwgxxap6tntf5Z:/home/beryl# sudo systemctl restart sshd
root@iZ2vcfk3gmwgxxap6tntf5Z:/home/beryl# sudo systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-10-28 15:38:37 CST; 25s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 48696 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 48698 (sshd)
    Tasks: 1 (limit: 1917)
   Memory: 1.7M
      CPU: 112ms
   CGroup: /system.slice/ssh.service
           └─48698 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Oct 28 15:38:37 iZ2vcfk3gmwgxxap6tntf5Z systemd[1]: Starting OpenBSD Secure Shell serv>
Oct 28 15:38:37 iZ2vcfk3gmwgxxap6tntf5Z sshd[48698]: Server listening on 0.0.0.0 port >
Oct 28 15:38:37 iZ2vcfk3gmwgxxap6tntf5Z sshd[48698]: Server listening on :: port 22.
Oct 28 15:38:37 iZ2vcfk3gmwgxxap6tntf5Z systemd[1]: Started OpenBSD Secure Shell serve>
Oct 28 15:38:38 iZ2vcfk3gmwgxxap6tntf5Z sshd[48699]: pam_unix(sshd:auth): authenticati>
Oct 28 15:38:39 iZ2vcfk3gmwgxxap6tntf5Z sshd[48699]: Failed password for root from 47.>
Oct 28 15:38:39 iZ2vcfk3gmwgxxap6tntf5Z sshd[48699]: error: Received disconnect from 4>
Oct 28 15:38:39 iZ2vcfk3gmwgxxap6tntf5Z sshd[48699]: Disconnected from authenticating >
```

## 练习Docker常用命令

本实验以高性能HTTP&反向代理Web服务器 Nginx1.24.0 为例，练习Docker常用命令。

### 拉取容器镜像


访问 Docker Hub (<https://hub.docker.com/>)，查找 Nginx ([https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx))，查看镜像名称。  
或者使用命令 `docker search` 在Docker Hub 查询镜像。



```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
a480a496ba95: Pull complete
f3ace1b8ce45: Pull complete
11d6fdd0e8a7: Pull complete
f1091da6fd5c: Pull complete
40eea07b53d8: Pull complete
6476794e50f4: Pull complete
70850b3ec6b2: Pull complete
Digest: sha256:28402db69fec7c17e179ea87882667f1e054391138f77ffaf0c3eb388efc3ffb
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
nginx            latest       3b25b682ea82  3 weeks ago    192MB
hello-world      latest       d2c94e258dcb  18 months ago  13.3kB
```

基于镜像创建并运行容器

简单创建运行

首先使用 docker run 进行简单创建与运行。 随后使用命令 docker ps 可以查看当前正在运行的容器信息,使用 docker ps -a可以查看全部（运行和非运行的都可以查看） 此时使用浏览器访问该ECS服务器公网IP，即 <http://<ECS 公网IP>>  docker run

挂载宿主目录

实际应用时，对于容器中部署的服务而言，通常需要把它们所依赖的配置文件、数据文件进行持久化存储，否则一旦容器被销毁，这些文件也将会随之销毁。要满足此类需求，需要在容器创建运行时，将宿主系统中用于存储配置文件、代码文件、数据文件的目录甚至具体文件映射到容器中的对应目录或文件，即，让容器在创建启动时，从宿主系统中指定的目录文件加载配置文件，在生成数据时则写入宿主系统中指定的目录文件。如果直接访问容器目录呢？

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker exec -it nginx_beryl /bin/bash
root@2a55ca92a773:/# ls
bin    docker-entrypoint.d  home    media    proc    sbin    tmp
boot  docker-entrypoint.sh lib     mnt      root    srv     usr
dev    etc                  lib64   opt      run     sys     var
root@2a55ca92a773:/# cd etc
root@2a55ca92a773:/etc# ls
adduser.conf      environment  kernel      pam.conf     rmt
alternatives      fonts       ld.so.cache pam.d         security
apt               fstab       ld.so.conf  passwd       selinux
bash.bashrc       gai.conf    ld.so.conf.d  passwd-      shadow
bindresvport.blacklist group        libaudit.conf  profile     shadow-
ca-certificates   group-      localtime    profile.d    shells
ca-certificates.conf gshadow     login.defs    rc0.d        skel
cron.d            gshadow-    logrotate.d   rc1.d        ssl
cron.daily        gss         mke2fs.conf   rc2.d        subgid
debconf.conf      host.conf   motd          rc3.d        subuid
debian_version    hostname    mtab          rc4.d        systemd
default           hosts       nginx         rc5.d        terminfo
deluser.conf      init.d      nsswitch.conf rc6.d        timezone
dpkg              issue       opt           rcS.d        update-motd.d
e2scrub.conf      issue.net   os-release    resolv.conf  xattr.conf
root@2a55ca92a773:/etc# cd nginx
root@2a55ca92a773:/etc/nginx# ls
conf.d  fastcgi_params  mime.types  modules  nginx.conf  scgi_params  uwsgi_params
```

退出容器，你可以输入 exit 命令退出容器的命令行。

本实验将为Nginx服务的配置文件、日志文件以及Web服务的根目录分别建立持久化映射。nginx\_beryl 容器中的三个固有目录与宿主的目录映射关系定义为：

持久化内容	容器目录	宿主目录
总体配置	/etc/nginx/nginx.conf	/opt/docker/nginx/nginx.conf
服务配置	/etc/nginx/conf.d	/opt/data/nginx/conf.d
日志文件	/var/log/nginx	/opt/log/nginx
Web服务	/usr/share/nginx/html	/opt/share/nginx/html

在 ECS 上执行以下命令创建三个宿主目录：

```
$ mkdir -p /opt/docker/nginx/conf.d
$ mkdir -p /opt/docker/nginx/logs
$ mkdir -p /opt/docker/nginx/html
# -p, --parents: 如果路径中任意一级父目录不存在，则创建
```



从先前启动的 nginx\_beryl 容器中将配置文件复制到宿主目录（否则目录挂载后，若容器找不到配置文件，服务不能正常运行）。

```
sudo docker cp nginx_beryl:/etc/nginx/nginx.conf /opt/docker/nginx/  
sudo docker cp nginx_beryl:/etc/nginx/conf.d /opt/docker/nginx/  
sudo docker cp nginx_beryl:/usr/share/nginx/html /opt/docker/nginx/
```

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# sudo mkdir -p /opt/docker/nginx/conf.d  
sudo mkdir -p /opt/docker/nginx/logs  
sudo mkdir -p /opt/docker/nginx/htmlroot@iZ2vcfk3gmwgxxap6tntf5Z:~#  
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker cp nginx_beryl:/etc/nginx/nginx.conf /opt/do  
cker/nginx/  
Successfully copied 2.56kB to /opt/docker/nginx/  
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker cp nginx_beryl:/etc/nginx/conf.d /opt/docker  
/nginx/  
Successfully copied 3.58kB to /opt/docker/nginx/  
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker cp nginx_beryl:/usr/share/nginx/html /opt/do  
cker/nginx/  
Successfully copied 4.1kB to /opt/docker/nginx/  
root@iZ2vcfk3gmwgxxap6tntf5Z:~#
```

为重新创新运行挂在目录的容器，需要先停止并删除当前的 nginx 容器。

#### 1. 首先停止运行当前 nginx 容器：

```
docker stop nginx_beryl # 除容器名称，亦可使用容器 ID 标识容器：docker stop  
a04d26310798
```

#### 2. 此时 `docker ps` 已查询不到已经停止运行的 nginx 容器，但是应用参数 `-a` 即可查看全部正在运行与停止运行的容器。停止的容器状态被标注为 `Exited`，同时端口没有发布（映射、记录）。

```
# docker ps -a  
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS   NAMES  
a04d26310798   nginx    ...                      9 minutes ago   Exited (0)      4  
minutes ago   nginx
```

#### 3. 随后删除停止运行的 nginx 容器。删除后通过 `docker ps -a` 查看，以无 nginx 容器踪影。

```
# docker rm nginx  
nginx  
≤ docker ps -a#自行观察
```

重新运行挂载宿主目录的 nginx 容器 重新设置执行 `docker run`，完成相关文件与目录的挂载。

```
# -v, --volume list: 挂载(绑定)目录: <宿主目录>:<容器目录>
$ docker run -d --restart=always --name nginx_beryl -p 80:80 -v
/opt/docker/nginx/nginx.conf:/etc/nginx/nginx.conf -v
/opt/docker/nginx/conf.d:/etc/nginx/conf.d -v
/opt/docker/nginx/html:/usr/share/nginx/html -v
/opt/docker/nginx/logs:/var/log/nginx nginx
```

在终端的操作如图:

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker stop nginx_beryl
nginx_beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
PORTS         NAMES
2a55ca92a773   nginx     "/docker-entrypoint..." 38 hours ago   Exited (0) 15 seconds ago
nginx_beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker rm nginx_beryl
nginx_beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS         NAMES
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker run -d --restart=always --name nginx_beryl -p 80:80 -v /opt/docker/nginx/nginx.conf:/etc/nginx/nginx.conf -v /opt/docker/nginx/conf.d:/etc/nginx/conf.d -v /opt/docker/nginx/html:/usr/share/nginx/html -v /opt/docker/nginx/logs:/var/log/nginx nginx
082d317f8220e1a2346ce97885958ed98a44088a6be85b28e156de2cd5f5c9fc
```

停止、启动以及进出容器

刚才已经体验过停止容器操作，此时再次使用该命令关闭刚创建的挂载了宿主目录的nginx\_beryl 容器，随后使用 docker start 命令启动该容器

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS         NAMES
082d317f8220   nginx     "/docker-entrypoint..." 4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp nginx_beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker stop nginx_beryl
nginx_beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS         NAMES
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker start nginx_beryl
nginx_beryl
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS         NAMES
082d317f8220   nginx     "/docker-entrypoint..." 5 minutes ago   Up 6 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp nginx_beryl
```

启动后，换命令行方式快速在ECS进行Nginx服务验证。

```

root@iZ2vcfk3gmwgxxap6tntf5Z:~# curl 127.0.0.1:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
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>

```

使用 `docker exec` 命令进入容器。该命令本身定义为在容器的内部运行一条指定命令，通过指定命令为 Shell 程序，如 `/bin/bash`，配合应用参数 `-it`，可实现进入容器进行命令行交互式操作。

```

docker exec -it nginx_beryl /bin/bash
# -i--interactive, 保持交互模式
# -t--tty, 分配一个伪终端(模拟终端)
# nginx_beryl, 容器名称
# /bin/bash, 待执行的命令, 进入 bash shell

```

```

</html>
root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker exec -it nginx_beryl /bin/bash
root@082d317f8220:/#

```

进入容器后，可以执行容器支持的命令操作。尝试进入 `/usr/share/nginx/html` 目录，创建 `test.html` 文件。因容器不支持 `vim` 命令，故使用 `echo` 命令创建文件。浏览器访问该文件将展示 `h1` 大小的 `Hello000000, world`。

完成容器内文件创建后，用 `exit` 命令退出容器。查看宿主目录 `ls /opt/docker/nginx/html`，可见容器内创建的 `test.html` 文件已成功持久化存储到宿主系统。此时使用 `curl` 访问 ECS 本地 URL `127.0.0.1:80/test.html`，可成功输出文件内容。

容器内部执行指令：

```

cd /usr/share/nginx/html
ls
echo "<h1>Hello000000, world</h1>"> test.html
ls

```

```
cat test.html
exit
```

```
root@082d317f8220:/# cd /usr/share/nginx/html
root@082d317f8220:/usr/share/nginx/html# ls
50x.html  index.html
root@082d317f8220:/usr/share/nginx/html# echo "<h1>Hellooooo, world</h1>" > test.html
root@082d317f8220:/usr/share/nginx/html# ls
50x.html  index.html  test.html
root@082d317f8220:/usr/share/nginx/html# cat test.html
<h1>Hellooooo, world</h1>
root@082d317f8220:/usr/share/nginx/html# exit
exit
```

ECS执行指令：

```
ls /opt/docker/nginx/html
curl 127.0.0.1:80/test.html
```

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# ls /opt/docker/nginx/html
50x.html  index.html  test.html
root@iZ2vcfk3gmwgxxap6tntf5Z:~# curl 127.0.0.1:80/test.html
<h1>Hellooooo, world</h1>
```

完成以上实验后请自行使用浏览器访问 <http://<ECS公网地址>/test.html>，观察 h1 大小Hellooooo, world

## 查看容器日志

实际应用时，可能会发现容器内的服务并没有正常工作，或者需要查看容器内服务的日志输出。此时需要采用 `docker logs` 查看容器的日志输出。

```
docker logs nginx_beryl
```

```
47.109.53.146root@iZ2vcfk3gmwgxxap6tntf5Z:~# docker logs nginx_beryl
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform
configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.
sh
10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform
configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.
sh
10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
root@iZ2vcfk3gmwgxxap6tntf5Z:~#
```

## 删除容器镜像

实际应用时，对于不使用的容器镜像执行删除，可以节约系统空间（云服务器的空间）

```
docker ps
docker rmi <镜像名># 等价于docker image rm <镜像名>; rmi: remove images
```

```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
082d317f8220   nginx     "/docker-entrypoint..." 10 hours ago   Up 9 hours    0.0.0
.0:80->80/tcp, :::80->80/tcp   nginx_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker stop nginx_beryl
nginx_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
082d317f8220   nginx     "/docker-entrypoint..." 10 hours ago   Exited (0) 8 seconds ago   nginx_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rm nginx_beryl
nginx_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:28402db69fec7c17e179ea87882667f1e054391138f77ffaf0c3eb388efc3ffb
Deleted: sha256:3b25b682ea82b2db3cc4fd48db818be788ee3f902ac7378090cf2624ec2442df
Deleted: sha256:3e8a4396bcd62aeb916ec1e4cf64500038080839f049c498c256742dd842334
Deleted: sha256:8dd6a711fbdd252eba01f96630aa132c4b4e96961f09010fbdb11869865f994
Deleted: sha256:9368c52198f80c9fb87fc3eaf7770afb7abb3bfd4120a8defd8a8f1a68ff375d
Deleted: sha256:46834c975bf2d807053675d76098806736ee94604c650aac5fe8b5172ab008c8
Deleted: sha256:6e433330e8b1553bee0637fac3b1e66c994bb2c0cab7b2372d2584171d1c93d8
Deleted: sha256:fbcb11fa4a4aff4cf0bfd963c49e2c416ff8047c9f84c2dc9328d3b833f1118d
Deleted: sha256:98b5f35ea9d3eca6ed1881b5fe5d1e02024e1450822879e4c13bb48c9386d0ad
```

构建 Docker 镜像容器

本实验以 Ubuntu 镜像作为基础镜像，拟制作一个包含 Python 基础环境以及 nginx 服务器的新镜像。整个制作过程包含两大步骤：一是写 Dockerfile 文件；二是使用 `docker build` 命令基于 Dockerfile 创建镜像。

编写 Dockerfile 文件

Dockerfile 是制作 Docker 容器镜像的基础。一般而言，Dockerfile 文件指令逻辑应按照以下模式建立：选择合适的基础镜像、安装基础工具与依赖、添加其他应用、清理缓存、声明镜像端口暴露情况、设置默认启动命令。

具体实验步骤如下：

创建一个本地实验文件夹，并进入新创建的实验文件夹

```
ls # 查看当前有什么文件夹
mkdir beryl_docker_exp #创建一个自己的实验文件夹
ls #再次查看
cd beryl_docker_exp
```



再次创建一个用来存放文件的文件夹，并进入新创建的存放文件的文件夹

```
ls #查看当前有什么文件夹
mkdir beryl_file #创建一个自己的实验文件夹
ls #再次查看
```

然后，创建 Dockerfile 文件，并编写相关内容。

```
touch Dockerfile #创建 Dockerfile 文件
vim Dockerfile #进入后进行编写内容
```

写入下面的内容：

```
# 使用 Ubuntu 作为基础镜像
FROM ubuntu:22.04
# 维护者信息，请自行更换为学生信息
LABEL maintainer="<3049736719@qq.com>"

# 设置工作目录
WORKDIR /app

# 安装 nginx & Python
RUN apt-get update && apt-get install -y
nginx
python3
python3-distutils

# 清理 apt 软件包缓存
RUN rm -rf /var/lib/apt/lists/*

# 声明暴露端口
EXPOSE 80

# 设置启动命令
CMD ["nginx", "-g", "daemon off;"]
```

输入 :wq, 然后点击 ENTER。

```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ ls
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ mkdir beryl_docker_exp
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ ls
beryl_docker_exp
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ cd beryl_docker_exp
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp$ ls
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp$ mkdir beryl_file
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp$ ls
beryl_file
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp$ touch Dockerfile
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp$ vim Dockerfile
```

## 基于 Dockerfile 创建容器镜像及后续操作

基于 Dockerfile, 使用 `docker build` 命令创建容器镜像。创建完成后, 通过 `docker images` 观察新创建镜像的大小。

```
# -t, tag: 镜像名称 (REPOSITORY:TAG)
# npu: Nginx Python Ubuntu, 自定义镜像名称
#.: PATH, 执行命令的上下文路径, 构造过程中可以引用该上下文中路径中的任何文件
docker build -t npu. # 创建镜像
docker images #查看新建镜像的大小
```

```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp/beryl_file$ vim Dockerfile
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp/beryl_file$ docker build -t npu .
[+] Building 65.1s (8/8) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 466B                                0.0s
=> [internal] load metadata for docker.io/library/ubuntu:22.04    17.9s
=> [internal] load .dockerignore                                    0.0s
=> => transferring context: 2B                                       0.0s
=> [1/4] FROM docker.io/library/ubuntu:22.04@sha256:0e5e4a57c2499249aafc3b 12.0s
=> => resolve docker.io/library/ubuntu:22.04@sha256:0e5e4a57c2499249aafc3b4 0.0s
=> => sha256:0e5e4a57c2499249aafc3b40fcd541e9a456aab7296681 6.69kB / 6.69kB 0.0s
=> => sha256:3d1556a8a18cf5307b121e0a98e93f1ddf1f3f8e092f1fddfd 424B / 424B 0.0s
=> => sha256:97271d29cb7956f0908cfb1449610a2cd9cb46b004ac8a 2.30kB / 2.30kB 0.0s
=> => sha256:6414378b647780fee8fd903ddb9541d134a1947ce092 29.54MB / 29.54MB 9.9s
=> => extracting sha256:6414378b647780fee8fd903ddb9541d134a1947ce092d08bdeb 1.9s
=> [2/4] WORKDIR /app                                              0.4s
=> [3/4] RUN apt-get update && apt-get install -y nginx python3 python3-di 32.6s
=> [4/4] RUN rm -rf /var/lib/apt/lists/*                            0.4s
=> exporting to image                                              1.6s
=> => exporting layers                                              1.5s
=> => writing image sha256:e6ee475e1f113ebed44429dd8e6f530f792e59df92a2c4d2 0.0s
=> => naming to docker.io/library/npu                              0.0s
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp/beryl_file$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
npu	latest	e6ee475e1f11	2 minutes ago	221MB
hello-world	latest	d2c94e258dcb	18 months ago	13.3kB

运行并访问 80 端口以检测 Nginx 服务。随后进入容器检测 Python 解析能力。

```
docker run -d --name npu_beryl -p 80:80 npu
curl 127.0.0.1 # 等价于 127.0.0.1:80, 或者 http://127.0.0.1:80
```

```
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp/beryl_file$ docker run -d --name npu_beryl -p 80:80 npu
e74b8e88f8550360c6124f9a0b040bd6a0107c449863d1f70c7f2f0bd368ed03
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~/beryl_docker_exp/beryl_file$ curl 127.0.0.1
<!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>
```

进入 `npu_beryl` 容器, 并检查 Python 的解析能力。

```
docker exec -it npu_beryl /bin/bash
python3
```

在 Python 解释器环境下输入下面内容:

```
print("hello world")
```

```
bery1@iZ2vcfk3gmwgxxap6tntf5Z:~/bery1_docker_exp/bery1_file$ docker exec -it npu_bery1 /bin/bash
root@e74b8e88f855:/app# ls
root@e74b8e88f855:/app# python3
Python 3.10.12 (main, Sep 11 2024, 15:47:36) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world!!!!")
hello world!!!!
>>> exit()
root@e74b8e88f855:/app# exit
exit
```

## ACR镜像管理与应用

实际应用时，开发者通常需要配置持续集成与持续部署流水线。在「构建镜像」阶段将代码制品打包到基础镜像后推送至特定镜像仓库（Registry）；而在「服务部署」阶段则从镜像仓库拉取该镜像并部署于特定 ECS 服务器或集群。本实验主要练习基于阿里云容器镜像服务与命令行 Docker 指令的镜像推送与拉取，为后续流水线实验奠定基础。

### 推送容器镜像至ACR

重新标记刚创建的镜像 `npu`，以满足推送该镜像至 ACR 的要求。镜像的名称包括「镜像仓库(Registry)与[标记(Tag)」两部分。此时要推送的镜像仓库为阿里云容器镜像服务 ACR 中所创建的仓库。

访问先前在[阿里云 ACR 控制台](#)创建的镜像仓库 `npu`，查询镜像仓库 URL。

1. 登陆地址: `registry.<region>.aliyuncs.com`
2. 仓库地址: `registry.<region>.aliyuncs.com/<namespace>/npu`

使用仓库地址重新标记镜像 `npu`。

```
docker tag npu registry.<region>.aliyuncs.com/<namespace>/npu
docker images|grep npu #自行观察
```

然后输入下面指令：

```
docker tag <镜像名> registry.<region>.aliyuncs.com/<namespace>/npu
docker push registry.<region>.aliyuncs.com/<namespace>/npu
```

```

beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ sudo docker login --username=歆毓xinyu crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com
[sudo] password for beryl:
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/#credential-stores

Login Succeeded

beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker tag npu crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu:latest
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker push crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu:latest
The push refers to repository [crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu]
3e1d9bba9fbf: Pushed
5b7b31c9a1a9: Pushed
8b7ebfc365d2: Pushed
2573e0d81582: Pushed
latest: digest: sha256:ccc875fc256f9b48b5865aaf8325588ef80382aab230a77af0a536668d92d714 size: 1154

```

## 拉取容器镜像至ECS

为测试从阿里云 ACR 拉取到ECS的容器镜像是否能够正常工作，首先需要删除 ECS 本地相关的容器以及镜像，以免干扰测试。

```

beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
e74b8e88f855   npu       "nginx -g 'daemon of..." 4 hours ago    Up 4 hours    0.0.0.0:80->80/tcp, :::80->80/tcp   npu_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker stop npu_beryl
npu_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
e74b8e88f855   npu       "nginx -g 'daemon of..." 4 hours ago    Exited (0) 12 seconds ago    npu_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rm npu_beryl
npu_beryl
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rmi npu
Untagged: npu:latest

```

使用以下指令查看还有哪些相关镜像,并继续删除:

```

beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker images
REPOSITORY                                TAG      IMAGE ID      CREATED      SIZE
crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu    latest   e6ee475e1f11   4 hours ago  221MB
registry.cn-chengdu.aliyuncs.com/myexp_1/npu    latest   e6ee475e1f11   4 hours ago  221MB
hello-world                                     latest   d2c94e258dcb   18 months ago 13.3kB
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rmi crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu
Untagged: crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu:latest
Untagged: crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu@sha256:ccc875fc256f9b48b5865aaf8325588ef86081eab108a77aef8a100d8a90a71a
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rmi registry.cn-chengdu.aliyuncs.com/myexp_1/npu
docker: 'registry.cn-chengdu.aliyuncs.com/myexp_1/npu' is not a docker command.
See 'docker --help'
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker rmi registry.cn-chengdu.aliyuncs.com/myexp_1/npu
Untagged: registry.cn-chengdu.aliyuncs.com/myexp_1/npu:latest
Deleted: sha256:e6ee475e1f113ebd44429dd8e6f530f792e59df92a2c4d20402f5e41c8db718
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world     latest   d2c94e258dcb   18 months ago 13.3kB

```

然后，在ECS上执行拉取镜像操作，随后创建运行容器并测试服务效果。



```

beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker pull crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu
Using default tag: latest
latest: Pulling from myexp_1/npu
7478e0ac0f23: Already exists
3e100e2c4241: Already exists
244fe29af1d0: Already exists
3cf18978a2c8: Already exists
Digest: sha256:ccc875fc256f9b48b5865aaf8325588ef80382aab230a77af0a536668d92d714
Status: Downloaded newer image for crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu:latest
crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu:latest
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker images
REPOSITORY                                                                 TAG          I
IMAGE ID          CREATED          SIZE
crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu  latest      e
6ee475e1f11       4 hours ago     221MB
hello-world       latest         d
2c94e258dcb       18 months ago  13.3kB

beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ docker run -d --name npu_beryl -p 80:80 crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu
d6d86d7630118bba958669f09ad416b9417add40227e8f84ba70baba5a015ca8
beryl@iZ2vcfk3gmwgxxap6tntf5Z:~$ curl 127.0.0.1
<!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>

```