**Linux下对Docker的安装和应用**

学习文档

2020-06-23

目录

[1、 Docker教程 1](#_Toc31080)

[2、 如何在Linux环境中安装Docker 1](#_Toc21885)

[3、 Docker的常用命令 3](#_Toc25275)

[4、 Docker下安装MySql 5](#_Toc23420)

[5、 Docker下安装Oracle 7](#_Toc16634)

[6、 Docker下安装redis 11](#_Toc19887)

[7、 Docker下安装tomcat，多个tomcat同时启动 15](#_Toc10930)

[8、 Docker下安装nginx实现负载均衡 16](#_Toc6705)

[9、 Docker下安装keepalived模拟高可用 21](#_Toc18513)

[10、 Docker下安装RabbitMQ 21](#_Toc22409)

# **Docker教程**

可参考菜鸟教程，较为详细，以下是教程链接地址

<https://www.runoob.com/docker/docker-tutorial.html>

# **如何在Linux环境中安装Docker**

## 查看Linux核心版本，3.10版本及以上才可以安装docker。

uname -r

## 更新yum包

yum update

## 查看docker是否曾经安装过

whereis docker

如果安装过，则删除之前的版本

yum remove docker docker-common docker-selinux docker-engine

## 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

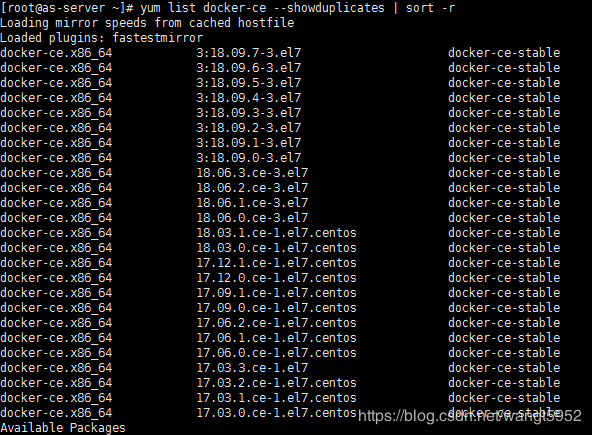
yum install -y yum-utils device-mapper-persistent-data lvm2

## 设置yum源

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

## 查看docker版本，一般使用稳定版

yum list docker-ce --showduplicates | sort -r



## 安装docker

默认安装最新版本

yum install docker-ce

安装某特定版本需增加版本号（如18.06.3.ce-3.el7）

yum install docker-ce-18.06.3.ce

## 启动docker

systemctl start docker

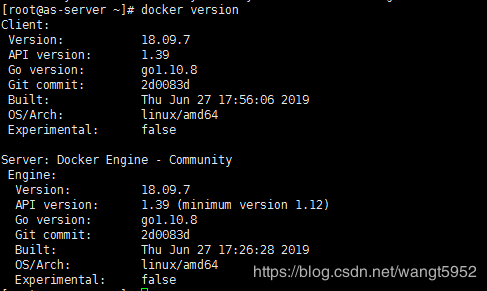
## 开机启动

systemctl enable docker

## 验证安装是否成功

docker version

如下存在Client和Server则成功



# **Docker的常用命令**

## 基础命令

查看docker版本信息

Docker -v

## 镜像相关命令

查看镜像：

docker images

docker search nginx #搜索所有nginx相关的镜像列表（模糊查询）

移除镜像：

docker rmi 镜像名称/镜像ID

docker rmi -f 镜像名称/镜像ID #强制删除有容器在运行中的镜像

拉取镜像

docker pull nginx #拉取nginx镜像

docker pull nginx:stable #拉取tag为stable的nginx

## 容器相关命令

查询容器运行进程：

docker ps

强制中断容器运行进程：

docker kill -s 容器名称

移除容器：容器必须是停止状态

docker rm 容器名称

查看容器内部信息

docker inspect 容器名称

停止某个容器

docker stop 容器名称

启动某个容器

docker start 容器名称

重启 -t 指定时间

docker restart (-t) name[name...]

输出指定容器日志信息

docker logs(-$) name

-f　　跟踪日志输出

-t　　显示时间戳 类似 tail -f

--tail  在日志的末尾输出指定行数日志（默认所有日志）

启动一个伪终端以交互式的方式进入某个容器

docker exec -ti 容器名称

exit #退出容器

# **[Docker下安装MySql](https://www.cnblogs.com/seethrough/p/9010522.html)**

## **拉取 Docker Hub 官方提供的mysql镜像**

docker pull mysql:5.7

## **运行容器**

docker run --name mysql -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306 -d mysql:5.7

含义

* --name mysql 创建的容器名称
* -e MYSQL\_ROOT\_PASSWORD=root 使用root作为root账号的密码
* -p 3306:3306 容器的3306端口映射到宿主机器
* -d 后台运行容器
* mysql:5.7 镜像名称

## **进入Mysql容器**

docker exec -it mysql bash

## **开启远程连接**

登录

mysql -u root -p"root"

进行授权

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;

重载授权表

FLUSH PRIVILEGES;

退出

exit;

## **大小写忽略**

到这里，如果防火墙没有问题，已经可以正常访问数据库了，但是默认的mysql镜像是严格检查大小写，因此我们需要对配置文件做设置，官方推荐的做法是，映射配置文件夹到宿主机，我这边使用的做法是把配置文件拷贝到宿主机，修改完，覆盖回去。

拷贝容器中的文件

docker cp mysql:./etc/mysql/mysql.conf.d/mysqld.cnf /home/custom/mysqld.cnf

[mysqld] 底部新增

lower\_case\_table\_names=1

文件拷贝到容器

docker cp /home/custom/mysqld.cnf mysql:./etc/mysql/mysql.conf.d/mysqld.cnf

## **重启容器**

docker restart mysql

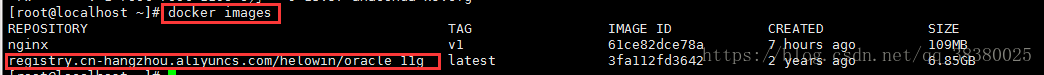
# **Docker下安装Oracle**

## 拉取镜像

docker pull registry.cn-hangzhou.aliyuncs.com/helowin/oracle\_11g

## 下载完成后 查看镜像

docker images



## 创建容器

docker run -d -p 1521:1521 --name oracle11g registry.cn-hangzhou.aliyuncs.com/helowin/oracle\_11g

注意：这里说一下，命令后面的地址一定要是你下载的镜像地址也就是你拉取镜像名字，否则会出现名字已存在等问题！

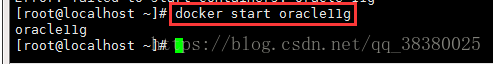
如果空间不足可以清理数据：

删除所有dangling数据卷（即无用的Volume）：

docker volume rm $(docker volume ls -qf dangling=true)

## 启动容器

docker start oracle11g



## 进入镜像进行配置

**（1）进入oracle的命令环境中。**

docker exec -it oracle11g bash

IMG_256

**（2）切换到oracle数据库的root 用户下，输入密码**

# su root

Password：helowin

IMG_256

1. **编辑profile文件配置ORACLE环境变量**

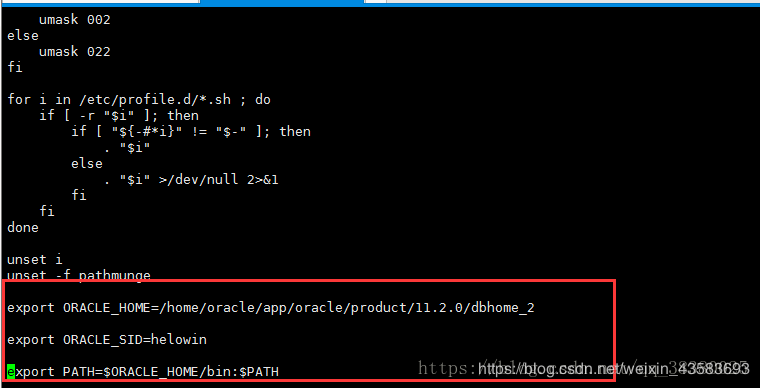
# vi /etc/profile

编辑文件内容

export ORACLE\_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome\_2

export ORACLE\_SID=helowin

export PATH=$ORACLE\_HOME/bin:$PATH



**（4）创建软连接**

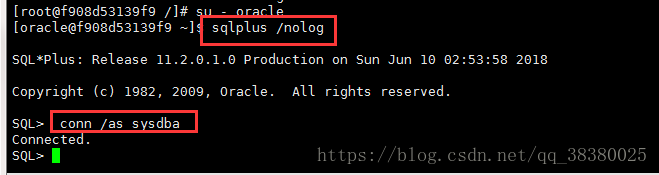
# ln -s $ORACLE\_HOME/bin/sqlplus /usr/bin

切换到oracle 用户

这里还要说一下，一定要写中间的内条 - 必须要，否则软连接无效

IMG_256

**（5）登录sqlplus–修改sys、system用户密码–创建用户**



修改密码：

alter user system identified by system;

alter user sys identified by sys;

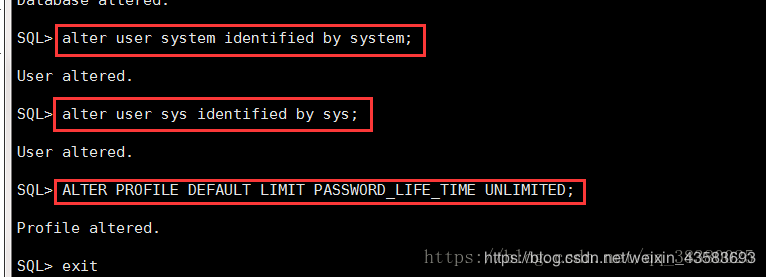
ALTER PROFILE DEFAULT LIMIT PASSWORD\_LIFE\_TIME UNLIMITED;

创建用户

create user test identified by test;

并给用户赋予权限

grant connect,resource,dba to test;



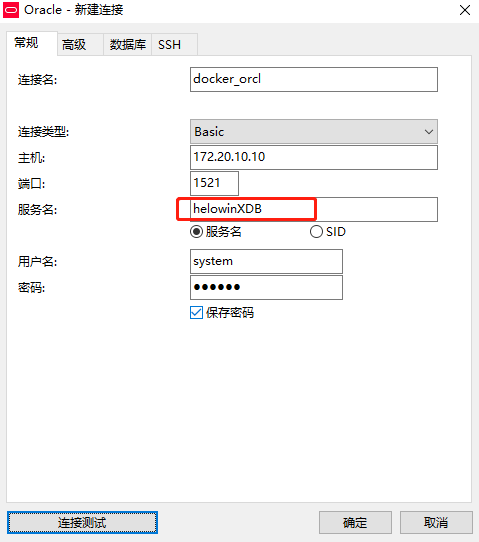
过程中出现的报错需要输入以下内容：

输入：alter database mount;

输入 ：alter database open;

## 客户端工具登录

注意服务名必须输入helowinXDB

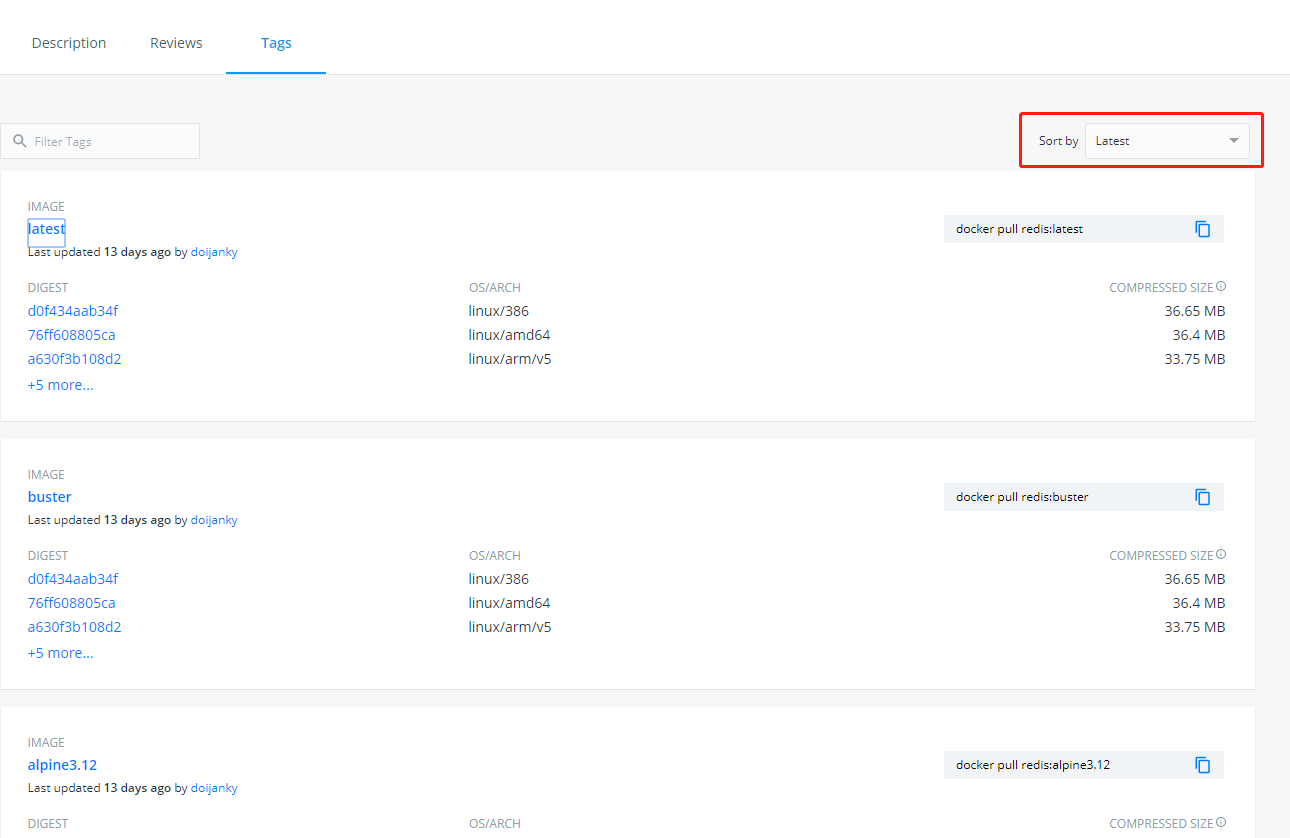


# **Docker下安装redis**

## 查看可用的 Redis 版本

访问 Redis 镜像库地址： [https://hub.docker.com/\_/redis?tab=tags](https://hub.docker.com/_/redis?tab=tags" \t "https://www.runoob.com/docker/_blank)。

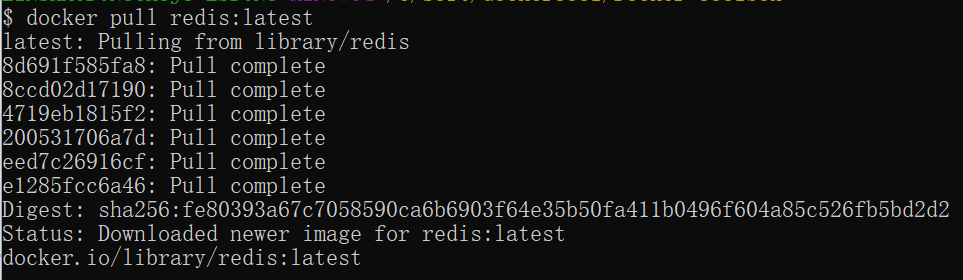
可以通过 Sort by 查看其他版本的 Redis，默认是最新版本 **redis:latest**。



## 拉取最新版的 Redis 镜像

这里我们拉取官方的最新版本的镜像：

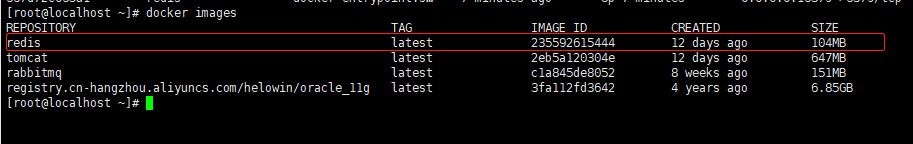
$ docker pull redis:latest

[](https://www.runoob.com/wp-content/uploads/2016/06/docker-redis3.png)

## 查看本地镜像

使用以下命令来查看是否已安装了 redis：

$ docker images



在上图中可以看到我们已经安装了最新版本（latest）的 redis 镜像。

## 运行容器

安装完成后，我们可以使用以下命令来运行 redis 容器：

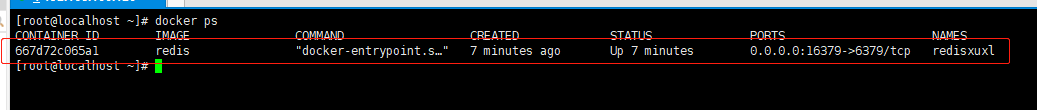
$ docker run -itd --name redisxuxl -p 16379:6379 redis

参数说明：

**-p 16379:6379**：映射容器服务的 16379 端口到宿主机的 6379 端口。外部可以直接通过宿主机ip:16379 访问到 Redis 的服务。

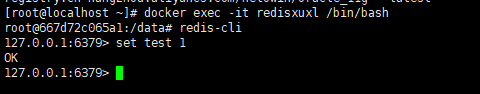
## 安装成功

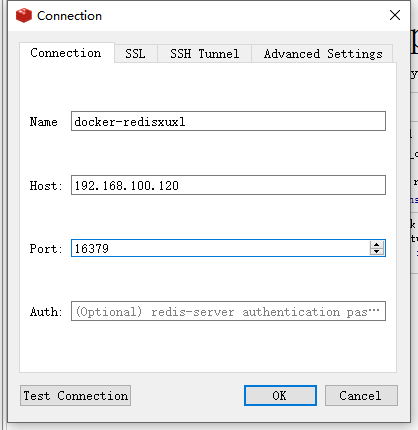
最后我们可以通过 docker ps 命令查看容器的运行信息：

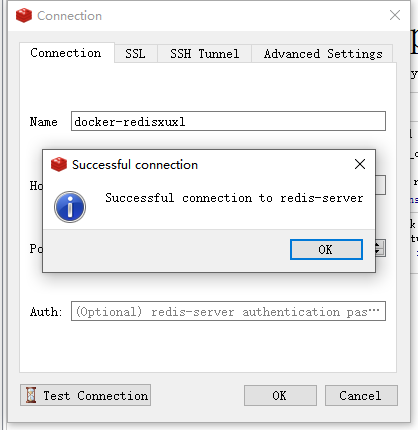


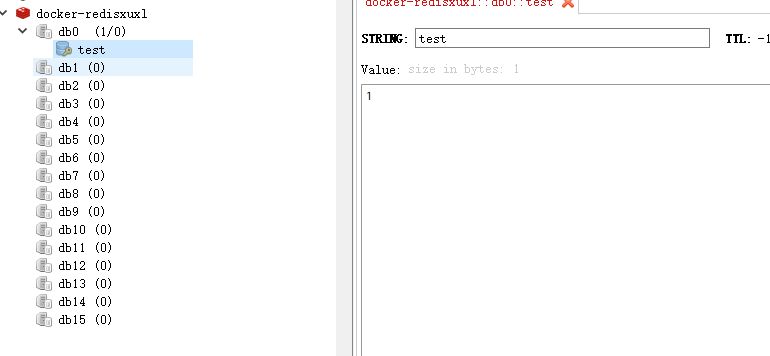
接着我们通过 redis-cli 连接测试使用 redis 服务。

$ docker exec -it redisxuxl /bin/bash









# **Docker下安装tomcat，多个tomcat同时启动**

## 拉取镜像

docker pull tomcat:latest

## 创建容器

docker run --name mytomcat1 -d -p 8081:8080 tomcat:8

docker run --name mytomcat2 -d -p 8082:8080 tomcat:8

## 拷贝war包到容器

进入容器

docker exec -it mytomcat1 /bin/bash

退出容器

exit

拷贝war包

docker cp /usr/local/web/test.war mytomcat1:/usr/local/tomcat/webapps

## 启动和关闭容器

docker start mytomcat1

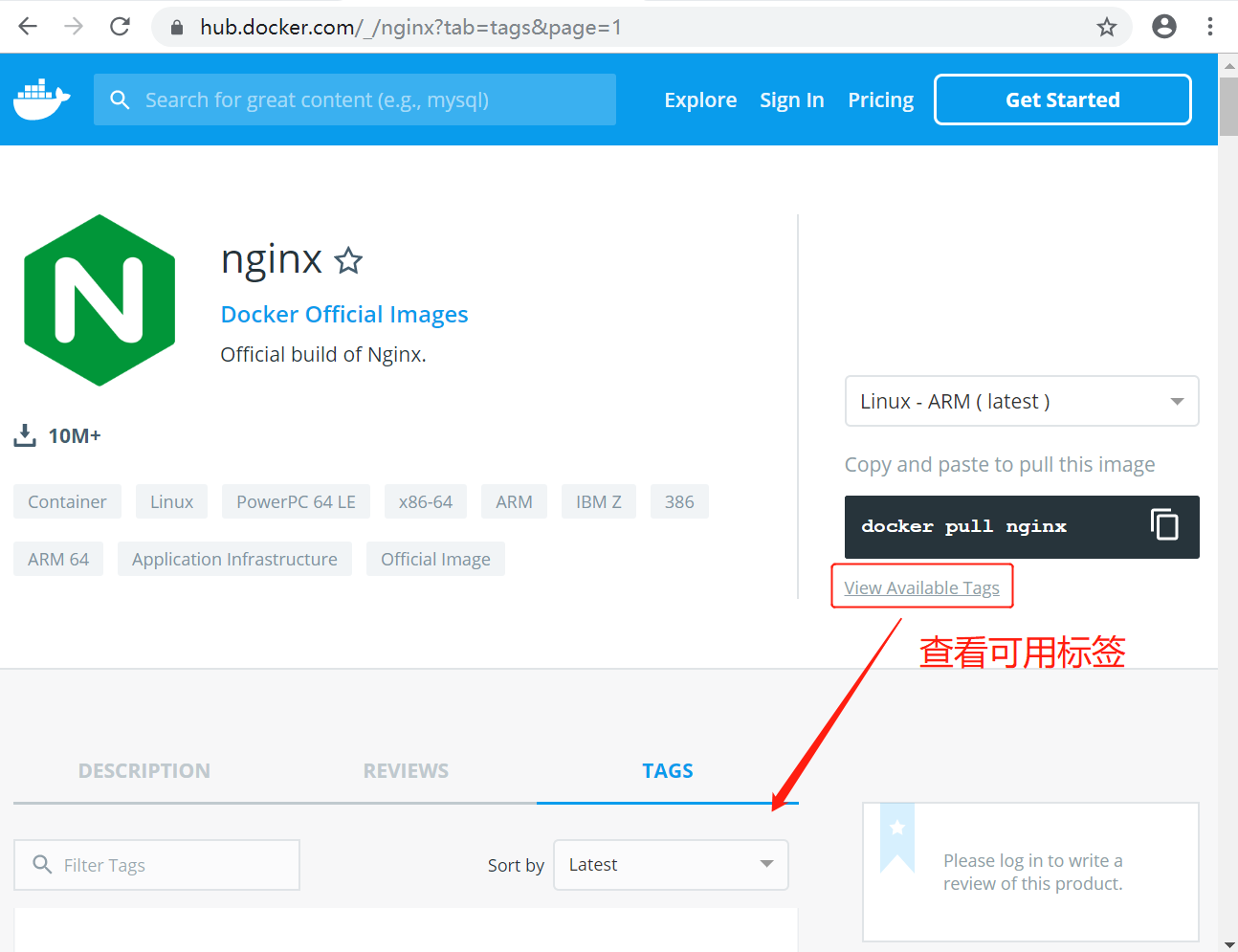
docker stop mytomcat1

# **Docker下安装nginx实现负载均衡**

## 查看可用的 Nginx 版本

访问 Nginx 镜像库地址： [https://hub.docker.com/\_/nginx?tab=tags](https://hub.docker.com/_/nginx?tab=tags" \t "https://www.runoob.com/docker/_blank)。

可以通过 Sort by 查看其他版本的 Nginx，默认是最新版本 **nginx:latest**。

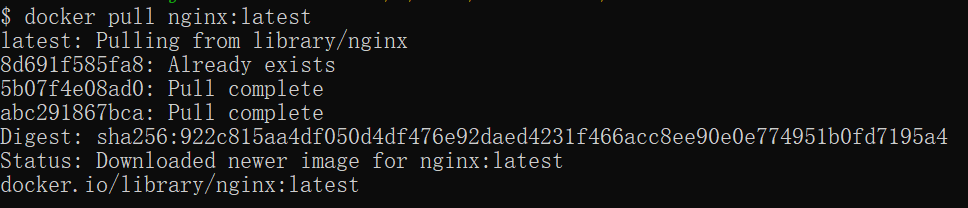
[](https://www.runoob.com/wp-content/uploads/2016/06/docker-nginx1.png)

你也可以在下拉列表中找到其他你想要的版本：

## 取最新版的 Nginx 镜像

这里我们拉取官方的最新版本的镜像：

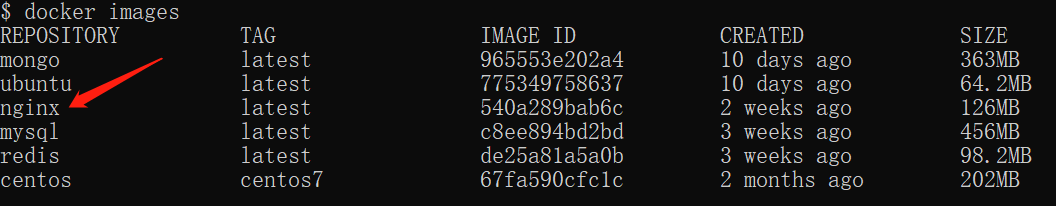
$ docker pull nginx:latest

[](https://www.runoob.com/wp-content/uploads/2016/06/docker-nginx3.png)

## 查看本地镜像

使用以下命令来查看是否已安装了 nginx：

$ docker images

[](https://www.runoob.com/wp-content/uploads/2016/06/docker-nginx4.png)

在上图中可以看到我们已经安装了最新版本（latest）的 nginx 镜像。

## 运行容器

安装完成后，我们可以使用以下命令来运行 nginx 容器：

$ docker run --name nginx-test -p 8080:80 -d nginx

参数说明：

* **--name nginx-test**：容器名称。
* **-p 8080:80**： 端口进行映射，将本地 8080 端口映射到容器内部的 80 端口。
* **-d nginx**： 设置容器在在后台一直运行。

[IMG_258](https://www.runoob.com/wp-content/uploads/2016/06/docker-nginx5.png)

## 安装成功

最后我们可以通过浏览器可以直接访问 8080 端口的 nginx 服务：



## 配置负载均衡

nginx的配置文件在容器的/etc/nginx/所以这里为了方便 先把配置文件拷贝出来

docker cp nginx-test:/etc/nginx/nginx.conf nginx.conf

vi nginx.conf修改配置文件

user nginx;

worker\_processes 1;

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65;

#gzip on;

upstream admin {

server ip:9091;

server ip:9090;

}

include /etc/nginx/conf.d/\*.conf;

}

upstream admin 下面配置的就是两个web应用所在的的ip地址以及端口号

注意下面的include /etc/nginx/conf.d/\*.conf;

意思是导入的配置文件 在 /etc/nginx/conf.d/\*.conf;下面

继续将 /etc/nginx/conf.d/\*.conf;下面的配置文件拷贝出来修改

docker cp nginx-test:/etc/nginx/conf.d/default.conf default.conf

打开后编辑

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log /var/log/nginx/host.access.log main;

location / {

proxy\_pass http://admin;

root /usr/share/nginx/html;

index index.html index.htm;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000;

# fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

# include fastcgi\_params;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

在location / {}里面添加proxy\_pass http://admin;  添加后wq保存

然后将两个配置文件拷贝回去

docker cp nginx.conf nginx-test:/etc/nginx/nginx.conf

docker cp default.conf nginx-test:/etc/nginx/conf.d/default.conf

重启nginx

docker restart nginx-test

# **Docker下安装keepalived模拟高可用**

方法一：两台linux服务器下安装keepalived+nginx

方法二：docker下安装centos，模拟跨主机

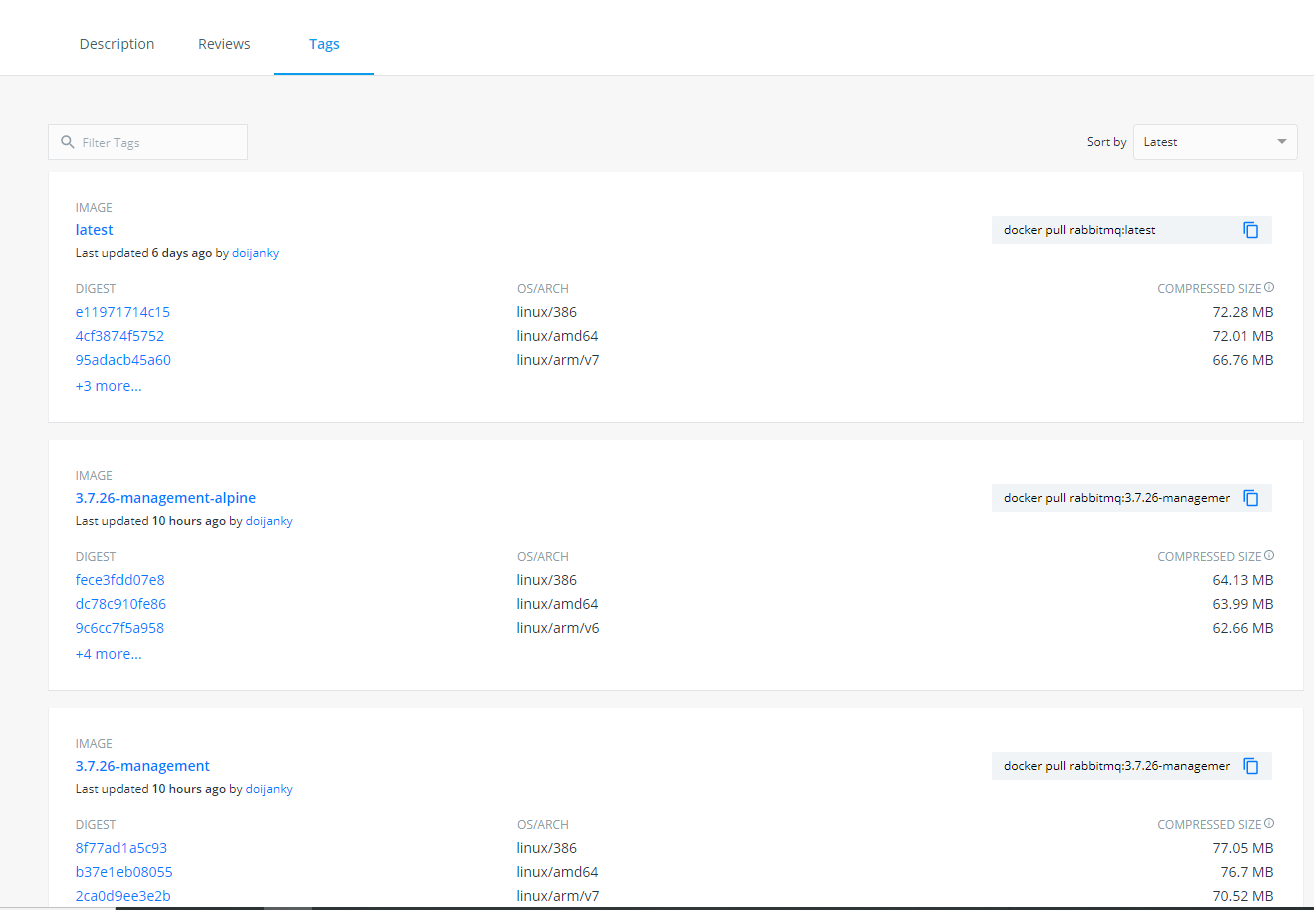
<https://www.cnblogs.com/jinjiangongzuoshi/p/9313438.html>

# **Docker下安装RabbitMQ**

## 查看可用rabbitMQ版本

进入docker hub镜像仓库地址：<https://hub.docker.com/_/rabbitmq?tab=tags>

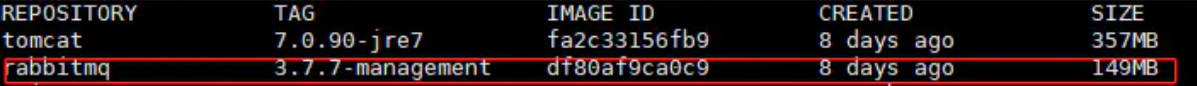
我们选择带有“mangement”的版本（**包含web管理页面**）



## 拉取镜像

docker pull rabbitmq:3.7.7-management

使用：docker images 查看所有镜像



## 根据下载的镜像创建和启动容器

docker run -d --name rabbitmq3.7.7 -p 5672:5672 -p 15672:15672 -v `pwd`/data:/var/lib/rabbitmq --hostname myRabbit -e RABBITMQ\_DEFAULT\_VHOST=my\_vhost -e RABBITMQ\_DEFAULT\_USER=admin -e RABBITMQ\_DEFAULT\_PASS=admin df80af9ca0c9

说明：

-d 后台运行容器；

--name 指定容器名；

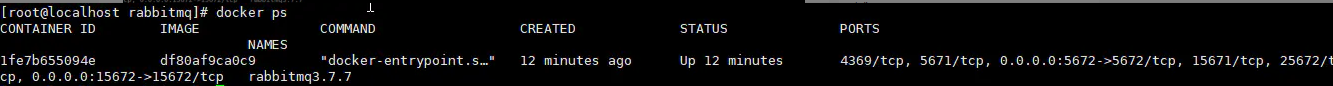
-p 指定服务运行的端口（5672：应用访问端口；15672：控制台Web端口号）；

-v 映射目录或文件；

--hostname  主机名（RabbitMQ的一个重要注意事项是它根据所谓的 “节点名称” 存储数据，默认为主机名）；

-e 指定环境变量；（RABBITMQ\_DEFAULT\_VHOST：默认虚拟机名；RABBITMQ\_DEFAULT\_USER：默认的用户名；RABBITMQ\_DEFAULT\_PASS：默认用户名的密码）

## 使用命令：docker ps 查看正在运行容器



## 可以使用浏览器打开web管理端：http://Server-IP:15672

