

内容目录

Docker 和 Jenkins 的安装与使用.....	1
1.使用存储库安装安装 docker.....	1
1.1 设置存储库.....	1
1.2 安装 DOCKER CE.....	3
1.3 升级 DOCKER CE.....	4
1.4 卸载旧版本.....	4
1.5 支持的存储驱动.....	4
1.6 AUFS 的额外步骤.....	4
1.7 配置 Docker 以在启动时启动.....	5
1.8 Docker 容器镜像删除.....	5
2.Jenkins 的安装.....	6
2.1 下载 Jenkins.....	6
2.2 创建 jenkins 文件夹.....	6
2.3 启动.....	7
2.4 初次使用获取管理员密码.....	7
2.5 初始插件下载.....	8
2.6 创建第一个管理员用户.....	9
3.使用 jenkins 构建 job.....	10
3.1 新建一个软件项目.....	10
3.2 构建任务.....	11
3.3 定时启动一个 job 遇到的问题.....	12
3.4 解决方法.....	12

Docker 和 Jenkins 的安装与使用

1.使用存储库安装安装 docker

大多数用户 设置 docker 的存储库(repository)并从中进行安装，以便于安装和升级任务。这是推荐的方法。

在新主机上首次安装 Docker CE 之前，需要设置 Docker 存储库。之后，您可以从存储库安装和更新 Docker。

1.1 设置存储库

1.更新 apt 包索引：

```
$ sudo apt-get update
```

2.安装包以允许 `apt` 通过 HTTPS 使用存储库：

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

3.添加 Docker 的官方 GPG 密钥：

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88 通过搜索指纹的最后 8 个字符，验证您现在拥有带指纹的密钥。

```
$ sudo apt-key fingerprint 0EBFCD88

pub 4096R/0EBFCD88 2017-02-22

    Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

uid          Docker Release (CE deb) <docker@docker.com>

sub 4096R/F273FCD8 2017-02-22
```

4.使用以下命令设置**稳定**存储库。即使您还想从**边缘**或**测试**存储库安装构建，您始终需要**稳定的**存储库。要添加**边缘**或 **测试**存储库，请在下面的命令中的单词后添加单词或（或两者）。

`edgeteststable`

注意：下面的 `lsb_release -cs` 子命令返回您的 Ubuntu 发行版的名称，例如 `xenial`。有时，在像 Linux Mint 这样的发行版中，您可能需要更改 `$(lsb_release -cs)` 为父 Ubuntu 发行版。例如，如果您正在使用 `Linux Mint Rafaela`，则可以使用 `trusty`。

```
$ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
```

```
$(lsb_release -cs) \
stable"
```

注意：从 Docker 17.06 开始，稳定版本也会被推送到**边缘并测试**存储库。
了解**稳定和边缘**渠道。

1.2 安装 DOCKER CE

1.更新 apt 包索引。

```
$ sudo apt-get update
```

2.安装最新版本的 Docker CE，或转到下一步安装特定版本：

```
$ sudo apt-get install docker-ce
```

有多个 Docker 存储库？

如果您启用了多个 Docker 存储库，则在未指定 `apt-get install` 或 `apt-get update` 命令中的版本的情况下安装或更新始终会安装尽可能高的版本，这可能不适合您的稳定性需求。

3.要安装特定版本的 Docker CE，请列出 repo 中的可用版本，然后选择并安装：

一个。列出您的仓库中可用的版本：

```
$ apt-cache madison docker-ce
```

```
docker-ce | 18.03.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable
amd64 Packages
```

湾 按其完全限定的包名称安装特定版本，例如，包名称（`docker-ce`）“=” 版本字符串（第 2 列）`docker-ce=18.03.0~ce-0~ubuntu`。

```
$ sudo apt-get install docker-ce=<VERSION>
```

Docker 守护程序自动启动。

4.通过运行 `hello-world` 映像验证是否正确安装了 Docker CE。

```
$ sudo docker run hello-world
```

此命令下载测试映像并在容器中运行它。当容器运行时，它会打印一条信息性消息并退出。

Docker CE 已安装并正在运行。该 `docker` 组已创建，但未向其添加任何用户。您需要使用它 `sudo` 来运行 Docker 命令。继续 [Linux postinstall](#) 以允许非特权用户运行 Docker 命令和其他可选配置步骤。

1.3 升级 DOCKER CE

要升级 Docker CE，请先运行 `sudo apt-get update`，然后按照 [安装说明](#) 选择要安装的新版本。

1.4 卸载旧版本

较旧版本的 Docker 被称为 `docker` 或 `docker-engine`。如果已安装，请卸载它们：

```
$ sudo apt-get remove docker docker-engine docker.io
```

如果 `apt-get` 报告没有安装这些软件包，则可以。

`/var/lib/docker/` 保留包括图像，容器，卷和网络在内的内容。现在调用 Docker CE 包 `docker-ce`。

1.5 支持的存储驱动

Ubuntu 上的 Docker CE 支持 `overlay2` 和 `aufs` 存储驱动程序。

- 对于 Linux 内核版本 4 及更高版本的新安装，`overlay2` 支持并首选 `aufs`。
- 对于 Linux 内核的版本 3，`aufs` 支持，因为该内核版本不支持 `overlay` 或 `overlay2` 驱动程序。

如果您需要使用 `aufs`，您需要做以下概述的其他准备工作。

1.6 AUFS 的额外步骤

对于 Ubuntu 16.04 及更高版本，Linux 内核包括对 OverlayFS 的支持，Docker CE `overlay2` 默认使用存储驱动程序。如果需要使用 `aufs`，则需要手动配置。见 [aufs](#)。

1.7 配置 Docker 以在启动时启动

大多数当前的 Linux 发行版（RHEL，CentOS，Fedora，Ubuntu 16.04 及更高版本）用于 `systemd` 管理系统启动时启动的服务。Ubuntu 14.10 及以下使用 `upstart`。

`systemd`

```
$ sudo systemctl enable docker
```

要禁用此行为，请 `disable` 改用。

```
$ sudo systemctl disable docker
```

如果需要添加 HTTP 代理，为 Docker 运行时文件设置不同的目录或分区，或进行其他自定义，请参阅 [自定义 systemd Docker 守护程序选项](#)。

`upstart`

Docker 自动配置为在启动时启动 `upstart`。要禁用此行为，请使用以下命令：

```
$ echo manual | sudo tee /etc/init/docker.override
```

`chkconfig`

```
$ sudo chkconfig docker on
```

1.8 Docker 容器镜像删除

1. 停止所有的 container，这样才能够删除其中的 images：

```
docker stop $(docker ps -a -q)
```

如果想要删除所有 container 的话再加一个指令：

```
docker rm $(docker ps -a -q)
```

2. 查看当前有什么 images

`docker images`

3.删除 images，通过 image 的 id 来指定删除谁

`docker rmi <image id>`

想要删除 untagged images，也就是那些 id 为<None>的 image 的话可以用

`docker rmi $(docker images | grep "^<none>" | awk "{print $3}")`

要删除全部 image 的话

`docker rmi $(docker images -q)`

2.Jenkins 的安装

2.1 下载 Jenkins

`sudo docker pull jenkins`

2.2 创建 jenkins 文件夹

创建 jenkins 文件夹，用于和容器内文件夹做磁盘挂载

`mkdir ~/jenkins`

修改 jenkins 文件夹的权限归属

`sudo chown -R 1000:1000 jenkins/`

在安装 jenkins 时候，挂在文件夹 ``~/jenkins/`` 的归属用户 id 必须是 1000，否则会抛出无操作权限异常。

2.3 启动

```
sudo docker run -itd -p 8080:8080 -p 50000:50000 --name jenkins  
--privileged=true -v ~/jenkins:/var/jenkins_home jenkins
```

参数:

- `--rm`: 可选。在 Docker 容器(jenkinsci/blueocean 的实例)关闭时自动删除。
- `-d`: 可选, detached 后台模式。后台运行 jenkinsci/blueocean 容器并输出容器的 ID。如果不指定这个选项, Docker 会把日志输出到终端窗口。
- `-p 8080:8080:publishes` 发布端口。将发布的 jenkinsci/blueocean 容器的 8080 端口映射到宿主机的 8080 端口。第一个数字代表宿主机的端口, 最后一个数字代表容器的端口。因此如果你指定 `-p 49000:8080`, 表示可以通过访问本地主机的 49000 端口来访问 Jenkins。• `-p 50000:50000`: 可选。将发布的 jenkinsci/blueocean 容器的 50000 端口映射到宿主机的 50000 端口。只有在其他主机设置了一个或多个基于 JNLP 的 Jenkins 代理程序, 而这些代理程序又与 jenkinsci/blueocean 容器(作为主 Jenkins 服务器“Jenkins master”)进行交互时才需要这个配置。默认情况下, 基于 JNLP 的 Jenkins 代理通过 TCP 端口 50000 与 Jenkins master 进行通信。可以参考 Configure Global Security 页面更改 Jenkins master 上的端口号。如果要更改 Jenkins 主机的 JNLP 代理的 TCP 端口值(假设是 51000), 那就需要重新运行 Jenkins(通过 `docker run ...` 命令)并指定此“publishes”选项 `-p 52000:51000`, 其中最后一个值与 Jenkins master 上的这个更改值相匹配, 第一个值是 Jenkins master 的宿主机上的端口号, 基于 JNLP 的 Jenkins 代理通过它(52000)与 Jenkins master 进行通信。
- `-v jenkins-data:/var/jenkins_home`: 可选, 但是建议使用。将容器的目录 `/var/jenkins_home` 映射到 Docker volume 卷并命名为 `jenkins-data`。如果这个卷不存在, 那么这个 `docker run` 命令会自动创建卷。如果你希望每次重启 Jenkins 时能持久化 Jenkins 的状态, 则必须使用这个参数。如果没有指定, 则每次 Jenkins 重启时都会初始化为新的实例。
注意: `jenkins-data` 卷可以通过 `docker volume create` 命令独立创建:
`docker volume create jenkins-data`
除了将容器的目录 `/var/jenkins_home` 映射到 Docker 卷外, 也可以映射到宿主机的本地文件系统, 例如, 通过 `-v $HOME/jenkins:/var/jenkins_home` 可以将容器的目录 `/var/jenkins_home` 映射到宿主机 `$HOME` 目录下的 `jenkins` 子目录中, 通常是 `/Users/<your-username>/jenkins` 或 `/home/<your-username>/jenkins`。
- `-v /var/run/docker.sock:/var/run/docker.sock`: 可选。`/var/run/docker.sock` 表示 Docker 守护进程监听的 Unix 套接字。这个映射使得 jenkinsci/blueocean 容器可以和 Docker 守护进程通信, 如果 jenkinsci/blueocean 容器需要实例化其他 Docker 容器时这种通信是必须的。如果使用 `docker` 参数(例如 `agent { docker { ... } }`)运行语法中包含 代理 部分的声明式管道, 则此选项是必需的。更多资料参考 Pipeline Syntax 页面
- `--privileged=true` 参数给容器加特权。

2.4 初次使用获取管理员密码

在浏览器输入 “localhost:8080” 进入 Jenkins, 首次进入需要获取管理员的密码, 如图:

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

获取密码

在 jenkins 启动的时候，我们设置了文件夹的挂在，所以我们直接可以在本地 jenkins 目录下查看密码

```
cat ~/jenkins/secrets/initialAdminPassword
```

或者在没有挂在磁盘时，获取密码，

```
sudo docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

2.5 初始插件下载

进入如下界面，选择适合自己的插件。



Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

开始下载插件：

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	⚙ Credentials Binding	Folders
⚙ Timestampers	⚙ Workspace Cleanup	⚙ Ant	⚙ Gradle	** bouncycastle API
⚙ Pipeline	⚙ GitHub Branch Source	⚙ Pipeline: GitHub Groovy Libraries	⚙ Pipeline: Stage View	** Structs
⚙ Git	⚙ Subversion	⚙ SSH Slaves	○ Matrix Authorization Strategy	** Script Security
⚙ PAM Authentication	⚙ LDAP	⚙ Email Extension	⚙ Mailer	** Pipeline: Step API
				** SCM API
				** Pipeline: API
				** JUnit
				OWASP Markup Formatter
				** Token Macro
				Build Timeout

2.6 创建第一个管理员用户

Getting Started

Create First Admin User

用户名:

majc0806

密码:

确认密码:

全名:

电子邮件地址:

确认之后：

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

点击 Start using Jenkins 之后就来到了首页：

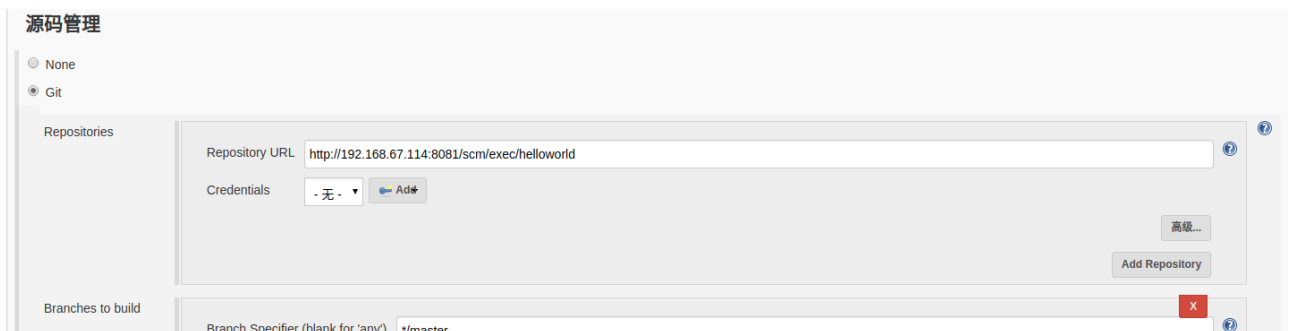


3.使用 jenkins 构建 job

3.1 新建一个软件项目



在码源管理选择 Git:



其他选项暂时不选，点击保存。



3.2 构建任务

在没有构建之前，工作区是空的；

可以选择立即构建选项来构建任务；

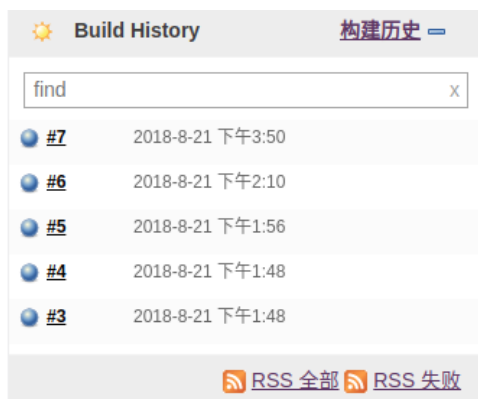


构建完成之后，可以在工作区看到，已经从 gerrit 上拉取了最新的代码。

Workspace of Myjob on master



并且可以在页面的左下角看到构建历史(Build History)

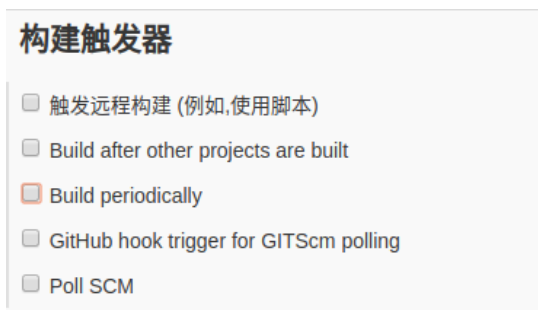


3.3 定时启动一个 job 遇到的问题

近期在配置 jenkins 定时任务时，发现未生效，并没有按时触发任务

解决思路：

1、先查看下我们的定时任务有没有选择正确，如下说明：



Poll SCM：定时检查源码变更，如果有更新就 checkout 最新 code 下来，然后执行构建动作。

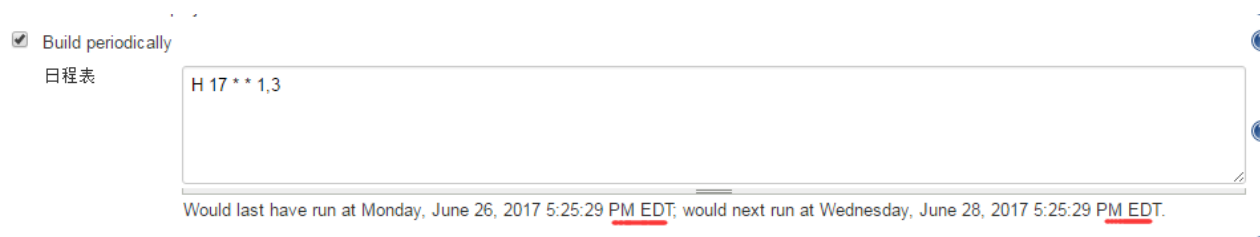
如果没有更新就不会执行构建

Build periodically：周期进行项目构建（源码是否发生变化没有关系）

所以如果没有配 GIT 或 SVN 的话，周期执行就用 Build periodically

2、选择了 Build periodically 后，还是未生效

有时候 jenkins 在 linux 下部署的话，就会存在时区问题一说，jenkins 构建在启动时加入 time-zone 时区可以解决，但在周期构建时，特别要注意时区并转换，如下图



写完后，jenkins 会提示什么时候运行，这时要注意写的是何时时区，如图中用的时区是 EDT，也就是美国东部时区，那定时跑时肯定与北京时间有差别，所以可以通过时区转换来完成；

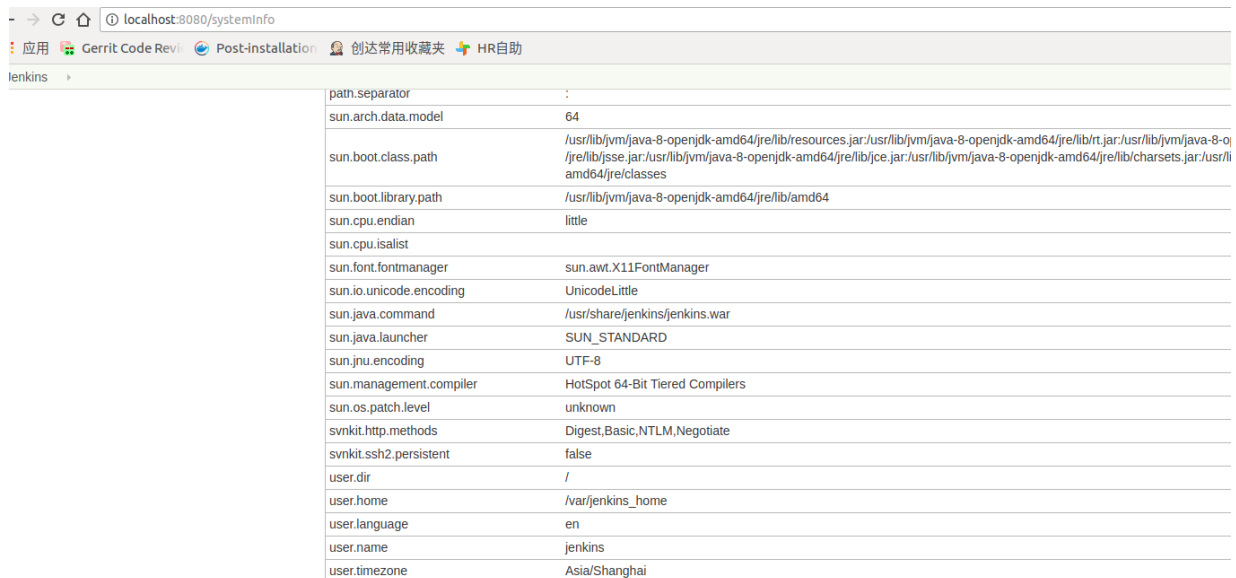
3.4 解决方法

jenkins 官方修改时区的方法。但是基本都是通过修改 jenkins、java 的参数来达到目的的。目前使用 docker 容器没办法处理。

最终解决方法，在 docker 启动容器时加上参数，最终的命令如下：

```
sudo docker run -itd -p 8080:8080 -p 50000:50000 --name jenkins --privileged=true -v  
~/jenkins:/var/jenkins_home -v /etc/timezone:/etc/timezone jenkins
```

jenkins 启动之后，在 url 定向到 localhost:8080/systemInfo, 可以看到系统属性最下方的 user.timezone 变为了 Asia/Shanghai



path.separator	:
sun.arch.data.model	64
sun.boot.class.path	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/resources.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/rt.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jsse.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jce.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/charsets.jar:/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/classes
sun.boot.library.path	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/amd64
sun.cpu.endian	little
sun.cpu.isalist	
sun.font.fontmanager	sun.awt.X11FontManager
sun.io.unicode.encoding	UnicodeLittle
sun.java.command	/usr/share/jenkins/jenkins.war
sun.java.launcher	SUN_STANDARD
sun.jnu.encoding	UTF-8
sun.management.compiler	HotSpot 64-Bit Tiered Compilers
sun.os.patch.level	unknown
svnkit.http.methods	Digest,Basic,NTLM,Negotiate
svnkit.ssh2.persistent	false
user.dir	/
user.home	/var/jenkins_home
user.language	en
user.name	jenkins
user.timezone	Asia/Shanghai

这个时候就可以设置定时任务了。