链滴

通过 docker 搭建自用的 gitlab 服务

作者：remixjc
原文链接：https://ld246.com/article/1575856331460
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20190807.jpg?imageView2/1/w/960/h/540/interlace/1/q/100">
<h2 id="前言">前言</h2>
git 是当下如日中天的版本管理系统。现在如果不是工作在 git 版本管理系统之下，几乎都不好思和人打招呼了。有很多现成的互联网的 git 服务提供给大家使用，例如号称程序员社交网络的 GitHb，还有低调好用的 bitbucket 。这些给个人使用或者公司用来做开源使用都没有什么问题。但如果部门内推广使用就会涉及到代码不能公开或者额外的费用的问题。本人原来在部门内采用的是手工在 linux 服务器上来管理代码仓库。权限没法设置，也非常不方便。所以也一直很苦恼。
正好 gitlab 公司提供了 gitlab 社区版，看了看基本满足了部门内 git 管理的需求。gitlab 提供各种各样的安装方式，最方便的当然还是 docker 方式的安装，适合我这种不想多折腾的。抽空搭建一个。也趟了几个坑，将步骤记录如下，希望对其他有此需求的人有所帮助。
<h2 id="docker-安装">docker 安装</h2>
既然是基于 docker 来安装 gitlab ，首先是安装 docker 环境了。我是在 centos 7 的基础上安的。根据官网的指南
<h4 id="删除旧版本的-docker">删除旧版本的 docker</h4>
旧版本的 docker 的叫做 docker 或者 docker-engine，如果系统中已经安装旧版本，则需要删。通过一下命令删除旧的 docker 版本
<pre><code class="highlight-chroma"><span class="highlight
cl">yum remove docker docker-common docker-selinux docker-engine
复制代码
</code></pre>
<h4 id="增加-docker-yum-源">增加 docker yum 源</h4>
新的 docker 叫做 docker-ce ，如果第一次安装 docker-ce 需要设置 docker-ce 的 yum 源。如下的命令来增加 docker-ce 的 yum 源
<pre><code class="highlight-chroma"><span class="highlight
cl">yum install -y yum-utils device-mapper-persistent-data lvm2
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
yum-config-manager --enable docker-ce-edge
yum-config-manager --enable docker-ce-test
复制代码
</code></pre>
<h4 id="安装-dokcer-ce">安装 dokcer-ce</h4>
首先我们查看所有有效的 docker-ce 的版本
<pre><code class="highlight-chroma"><span class="highlight
cl">yum list docker-ce --showduplicates | sort -r
复制代码
</code></pre>
得到如下的列表
<pre><code class="highlight-chroma"><span class="highlight
cl">docker-ce.x86_64 17.12.0.ce-1.el7.centos docker-ce-test
docker-ce.x86_64 17.12.0.ce-1.el7.centos docker-ce-stable
docker-ce.x86_64 17.12.0.ce-1.el7.centos @docker-ce-stable
docker-ce.x86_64 17.12.0.ce-0.4.rc4.el7.centos docker-ce-test
docker-ce.x86_64 17.12.0.ce-0.3.rc3.el7.centos docker-ce-test
docker-ce.x86_64

17.12.0.ce-0.2.rc2.el7.centos docker-ce-test
docker-ce.x86_64
17.12.0.ce-0.1.rc1.el7.centos docker-ce-test
docker-ce.x86_64
17.11.0.ce-1.el7.centos docker-ce-test
docker-ce.x86_64
17.11.0.ce-0.4.rc4.el7.centos docker-ce-test
...
docker-ce.x86_64
17.03.1.ce-0.1.rc1.el7.centos docker-ce-test
docker-ce.x86_64
17.03.0.ce-1.el7.centos docker-ce-stable
复制代码
</code></pre>
从这里可以看到最新的稳定版本是 17.12.0.ce。我们用下面的命令选择安装该版本
<pre><code class="highlight-chroma"><span class="highlight
cl">yum install docker-ce-17.12.0.ce
复制代码
</code></pre>
<h4 id="自定义-docker-配置">自定义 docker 配置</h4>
因为众所周知的原因，访问 docker 的中央仓库下载 docker 镜像的速度非常慢。为了获得较好
体验，我们需要配置一个寻找一个国内的镜像加速服务。
默认情况下，docker 将运行的目录配置到了/var/lib/docker 目录下。通常这个目录是在 linux
根分区下，空间比较有限，所以我们需要将 docker 的运行目录配置到其他目录下。
镜像加速服务可以使用阿里云的镜像加速服务。注册阿里云的用户后，登录 <a href="https://ld
46.com/forward?goto=https%3A%2F%2Fcr.console.aliyun.com" target="_blank" rel="nofollo
ugc">https://cr.console.aliyun.com ，在管理控制台选择镜像加速服务，则出现如下的信息
这里红色打码部分就是你的阿里云镜像加速服务的地址。
<img src="https://ld246.com/images/img-loading.svg" alt="45ed10a771a1bf79db302f6
030236c.png" data-src="https://b3logfile.com/file/2019/12/45ed10a771a1bf79db302f630302
6c-783eafea.png?imageView2/2/interlace/1/format/jpg">
为了自定义 docker 的默认运行目录和镜像仓库地址，我们需要修改 /etc/docker/daemon.json
/p>
<pre><code class="highlight-chroma"><span class="highlight
cl">mkdir /etc/docker
vi /etc/docker/da
mon.json
复制代码
</code></pre>
输入如下内容
<pre><code class="highlight-chroma"><span class="highlight
cl">{
 "graph":"/u1/do
ker",
 "registry-mirrors"
["https://xxxxxx.mirror.aliyuncs.com"]
}
复制代码
</code></pre>
graph 定义 docker 运行的目录， registry-mirrors 定义了 docker 获取镜像的仓库的地址。</
>
<h4 id="启动-docker">启动 docker</h4>
执行如下的命令启动 docker 的服务
<pre><code class="highlight-chroma"><span class="highlight

cl">systemctl start docker
复制代码
</code></pre>
执行如下命令查看 docker 信息
<pre><code class="highlight-chroma"><span class="highlight
cl">docker info
复制代码
</code></pre>
可以看到如下的信息
<img src="https://ld246.com/images/img-loading.svg" alt="9a32c8fb2fdbd4ce54584e60
92ee3a.png" data-src="https://b3logfile.com/file/2019/12/9a32c8fb2fdbd4ce54584e60492ee
a-cc2ab922.png?imageView2/2/interlace/1/format/jpg">
可以看到 Docker Root Dir 和 RegistryMirrors 都变成了 /etc/docker/daemon.json 中配置的
容了
<h2 id="安装-gitlab">安装 gitlab</h2>
<h4 id="下载镜像">下载镜像</h4>
执行下面的命令，从 docker 的镜像仓库中下载 gitlab 社区版的镜像
<pre><code class="highlight-chroma"><span class="highlight
cl">docker pull gitlab/gitlab-ce:latest
复制代码
</code></pre>
镜像有 1g 多，所以需要等待一段时间
<h4 id="数据持久化保存">数据持久化保存</h4>
因为容器的数据是不能持久化保存的。所以我们需要用 docker volume 的方式将存储的数据映
到操作系统的目录中来。这样就算运行的容器崩溃，我们重新启动一个新的容器，原来容器中的数据
是不会丢失
我们建立了目录 /u1/gitlab 来保存 gitlab 容器中的数据
<h4 id="准备映射-sshd">准备映射 sshd</h4>
另外，为了 git 采用 ssh 协议来操作 git 仓库，我们将主机的 sshd 的 22 端口映射到 容器中去
将主机的 sshd 端口更改为 15678。这里因为 centos 7 更严格的安全机制，算是一个坑，需要按照
面的步骤去进行。
编辑文件 /etc/ssh/sshd_config，将其中的 #Port 22 注释去掉，将数字 22 更改为 15678
执行下面的命令重启 sshd 服务
<pre><code class="highlight-chroma"><span class="highlight
cl">systemctl restart sshd
复制代码
</code></pre>
运行下面的命令使 15678 端口可以对外提供服务。否则无法进行远程的 ssh 登录。
<pre><code class="highlight-chroma"><span class="highlight
cl">semanage port -a -t ssh_port_t -p tcp 15678
firewall-cmd --pe
manent --add-port=15678/tcp
firewall-cmd --rel
ad
复制代码
</code></pre>
<h4 id="运行-gitlab">运行 gitlab</h4>
<pre><code class="highlight-chroma"><span class="highlight
cl">docker run \
  --publish 443:4
3 --publish 80:80 --publish 22:22 \
  --name gitlab \
  --volume /u1/gi
lab/config:/etc/gitlab \

  --volume /u1/gitlab/logs:/var/log/gitlab \
  --volume /u1/gitlab/data:/var/opt/gitlab \
  gitlab/gitlab-ce
复制代码
</code></pre>
这里把主机的 443、80、22 端口直接转发到容器，同时利用 --volume /u1/gitlab/config:/etc/itlab 、 --volume /u1/gitlab/logs:/var/log/gitlab 、 --volume /u1/gitlab/data:/var/opt/gitlab 三个参数将 gitlab 的配置、数据和日志持久化到主机文件系统上来。
<h2 id="管理员配置-gitlab">管理员配置 gitlab</h2>
<h4 id="登录-gitlab">登录 gitlab</h4>
等待 docker 容器启动完成后，访问 http://ip 就进入 gitlab 访问界面。第一次访问是让我们修管理员密码。如下所示
<img src="https://ld246.com/images/img-loading.svg" alt="ea32e56846515eaf37471c979cdab3.png" data-src="https://b3logfile.com/file/2019/12/ea32e56846515eaf37471c9679cdb3-5207daf7.png?imageView2/2/interlace/1/format/jpg">
设置后管理员密码后，就进入登录页面，输入用户名 root 和刚才设置的密码就进入了 gitlab 的制台。如下图所示
<img src="https://ld246.com/images/img-loading.svg" alt="9db9a63278fb53858f1fceec195429.png" data-src="https://b3logfile.com/file/2019/12/9db9a63278fb53858f1fceecb19549-155a9309.png?imageView2/2/interlace/1/format/jpg">
<h4 id="创建组--group-">创建组（group）</h4>
gitlab 里面有三类对象：组（group）、项目（project）和用户 （people）。
为了方便管理，我们应该基于组来创建项目。一个项目就是一个 git 的仓库。基于组创建项目，后将用户设置合适的权限后加入到组里面。这样用户就有了组里面所有项目的对应权限。
点击 “Create a group” 链接，如下图所示创建一个“健康医疗开发组” 的组
<img src="https://ld246.com/images/img-loading.svg" alt="21ab4271e483cf7d210818e32fe8a6.png" data-src="https://b3logfile.com/file/2019/12/21ab4271e483cf7d210818e632fea6-89030e43.png?imageView2/2/interlace/1/format/jpg">
<h4 id="创建用户-people-">创建用户（people）</h4>
点击 “Add people” 链接，如下图所示创建一个 “yanggch” 的用户
<img src="https://ld246.com/images/img-loading.svg" alt="9fb0376e74ff24e37bab907d7a44ec.png" data-src="https://b3logfile.com/file/2019/12/9fb0376e74ff24e37bab907d77a44c-b2735271.png?imageView2/2/interlace/1/format/jpg">
因为还没有配置好邮件服务，所以还不能发送用户初始化密码的邮件。我们需要编辑用户，手动置一个密码。如下图所示。如果用户忘记了密码，充值密码也可以在这里进行。
<img src="https://ld246.com/images/img-loading.svg" alt="14c82f0f91807b111075192e99c698.png" data-src="https://b3logfile.com/file/2019/12/14c82f0f91807b111075192e099c68-dad68487.png?imageView2/2/interlace/1/format/jpg">
<h4 id="将用户加入组">将用户加入组</h4>
为了方便管理，需要将用户加入到对应的组里面。如下图所示，在组管理界面中，点击组的名称进入组用户设置界面。将刚才创建的用户 “yanggch” 加入到组 “健康医疗开发组”中，并且给他置为 “Master” 角色。只有 “Master” 或者 “Owner” 角色才能推送 git 的更新。
<img src="https://ld246.com/images/img-loading.svg" alt="e542bb2f0dafae5641e5b3df974e0e.png" data-src="https://b3logfile.com/file/2019/12/e542bb2f0dafae5641e5b3dfb974ee-e08748be.png?imageView2/2/interlace/1/format/jpg">
<h4 id="创建项目-project-">创建项目（project）</h4>
增加 gitlab 组的时候，为了让项目让组里面的人都能访问，注意要将项目建立在组之下。如下所示，在“健康医疗开发组”之下建立了 “redis_util” 的项目。
<img src="https://ld246.com/images/img-loading.svg" alt="2f7db4b25265591e487e9a63de67ea.png" data-src="https://b3logfile.com/file/2019/12/2f7db4b25265591e487e9a6a3de7ea-6ae414fa.png?imageView2/2/interlace/1/format/jpg">
这样项目建好之后就可以被组里的用户访问了。

<h2 id="客户端访问">客户端访问</h2>
<h4 id="安装-git-客户端">安装 git 客户端</h4>
这里演示在 window 上安装 git 客户端。首先根据 window 版本下载 git 客户端安装程序。这我下载的是
<pre><code class="highlight-chroma">https://github.com/git-for-windows/git/releases/download/v2.15.1.windows.2/Git-2.15.1.-64-bit.exe
复制代码
</code></pre>
<h4 id="命令行访问">命令行访问</h4>
安装后，在准备 clone 项目的目录下点击鼠标右键，出现下面的右键菜单。
<img src="https://ld246.com/images/img-loading.svg" alt="5924bd541bc663f90936c7e8b185b2.png" data-src="https://b3logfile.com/file/2019/12/5924bd541bc663f90936c7e48b15b2-56a05eab.png?imageView2/2/interlace/1/format/jpg">
点击 “Git Bash Here” 进入 git 命令行环境。我们会从 redis_util 项目中看到该项目基于 http 协议的 clone 命令是
<pre><code class="highlight-chroma">git clone http://7a45cd079bdc/healthcare_dev/redis_util.git
复制代码
</code></pre>
这里的 7a45cd079bdc 实际上是 docker 容器的机器名。实际执行的时候我们把这个字符串更成容器所在主机的 ip 地址即可。如下所示
<pre><code class="highlight-chroma">git clone http://10.110.2.50/healthcare_dev/redis_util.git
复制代码
</code></pre>
这种情况下会要求输入登录的用户名和密码。这里输入刚才创建的用户的用户名和密码即可。在用前，需要用这个用户登录 gitlab 控制台修改一下初始密码才能使用。
<img src="https://ld246.com/images/img-loading.svg" alt="7f4e601fa1d0faaae6fe2176c7b86a.png" data-src="https://b3logfile.com/file/2019/12/7f4e601fa1d0faaae6fe2176cf7b86a42beef5e.png?imageView2/2/interlace/1/format/jpg">
出现如下提示表示从 gitlab clone 项目成功
<pre><code class="highlight-chroma">Cloning into 'redis_util'...
warning: You appar to have cloned an empty repository.
复制代码
</code></pre>
<h4 id="免密码登录">免密码登录</h4>
刚才那种方式通过 http 协议和 gitlab 进行通信，每次都要输入用户名和密码，非常不方便。用可以设置通过 ssh 进行交互，将 ssh key 加入到用户的 sshkey 设置列表中。
参考“命令访问” 章节进入 git 的 bash 环境。执行下面的命令进入 ssh key 存储目录
<pre><code class="highlight-chroma">cd ~/.ssh
复制代码
</code></pre>
目录中 id_rsa.pub 是 ssh 访问的公钥。如果不存在则执行下面的命令生成
<pre><code class="highlight-chroma">ssh-keygen -t rsa
复制代码
</code></pre>
全部回车后，会生成 id_rsa.pub 文件。
将文件中的内容拷贝到剪贴板。然后通过前面创建的用户名和密码登录 gitlab 控制台。在下面界面中，将 id_rsa.pub 文件的内容填入文本框

保存后。再通过 ssh 协议操作 git 仓库，将不再需要输入用户名和密码。如下所示
<h4 id="推送一次提交">推送一次提交</h4>
首先配置当前仓库的用户名和用户邮箱配置
<pre><code class="highlight-chroma"><span class="highlight
cl">cd redis_util
git config --local
ser.name "yanggch"
git config --local
ser.email "yanggch@inspur.com"
复制代码
</code></pre>
然后在 redis_util 目录下加入一个 readme.txt，执行下面的命令提交并将更新推送到 gitlab 远
服务器
<pre><code class="highlight-chroma"><span class="highlight
cl">#将新文件加入到版本管理
git add readme.txt
#提交
git commit -m"
一次提交"
#将当前分支 maste
推送到远程仓库
git push
#以 master 分支
基础建立一个新的 dev 本地分支
git checkout -b d
v
#将本地仓库分支
送到远程仓库，在远程仓库建立对应的 dev 分支
git push --set-ups
ream origin dev
复制代码
</code></pre>
到 gitlab 控制台查看 redis_util 的状态。如下图所示
<img src="https://ld246.com/images/img-loading.svg" alt="12a3bf48312653af1ee42c73
ad89e9.png" data-src="https://b3logfile.com/file/2019/12/12a3bf48312653af1ee42c736ad89
9-aafaaccc.png?imageView2/2/interlace/1/format/jpg">
到此，我们就完成了一个公司级别的 gitlab 服务器的搭建工作。小伙伴们就可以在这个上面流
的进行开发了。
原文发表在简书中，<a href="https://ld246.com/forward?goto=https%3A%2F%2
www.jianshu.com%2Fp%2F8d8e6b45a514" target="_blank" rel="nofollow ugc">原始链接</a