Docker 安装 GitLab 说明指引文档

目录

姓名	日期	版本	说明
Liangjl	2019-05-27	V.0.0.1	初稿创建

1、 前言

1、此文档主要是在 Docker 上面安装 GitLab，GitLab 是一个用于仓库管理系统的开源项目，使用 Git 作为代码管理工具，并在此基础上搭建起来的 web 服务。

2、 GitLab 是利用 Ruby on Rails 一个开源的版本管理系统，实现一个自托管的 Git 项目仓

库，可通过 Web 界面进行访问公开的或者私人项目。它拥有与 Github 类似的功能，能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序(Wall)进行交流。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。

3、 注意：由于公司项目用到 Docker，这里也以 Docker 为例进行详解，没有 Docker 的童鞋们先补习一下 Docker 方面的知识点。

4、 此文档不阐述 Docker 安装的过程,需要了解的同学们请参考”CentOS Docker 安装”

2、Docker 的简单认知

2.1 Docker 安装 GitLab 的好处

在 docker 上安装 Gitlab 的好处，由于网上很多活雷锋把整个 gitlab 所依赖的软件都一起打包好成镜像(Images)，我们只需要了解些 docker 的简单命令或安装过程中修改些配置即可

2.2 Docker 有哪些优势

Docker 五大优势：持续集成、版本控制、可移植性、隔离性和安全性

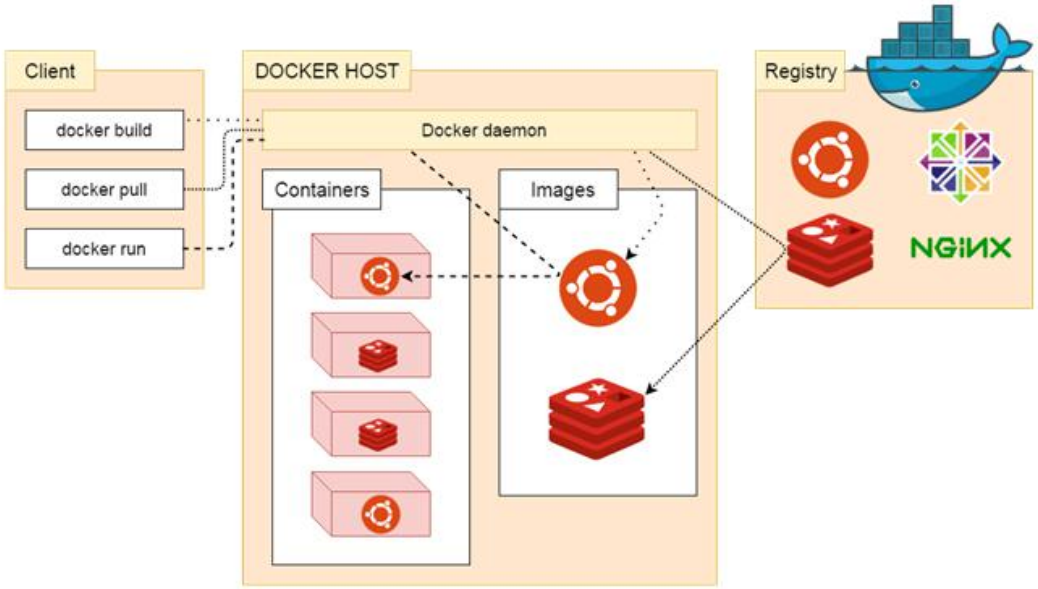
对比传统虚拟机总结

特性	容器	虚拟机
启动	秒级	分钟级
硬盘使用	一般为 MB	一般为 GB
性能	接近原生	弱于
系统支持量	单机支持上千个容器	一般几十个

2.3、Docker 系统架构

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程 API 来管理和创建 Docker 容器。
Docker 容器通过 Docker 镜像来创建。
容器与镜像的关系类似于面向对象编程中的对象与类。

Docker	面向对象
容器	对象
镜像	类



标题	说明
镜像 (Images)	Docker 镜像是用于创建 Docker 容器的模板。
容器 (Container)	容器是独立运行的一个或一组应用。
客户端 (Client)	Docker 客户端通过命令行或者其他工具使用 Docker API (https://docs.docker.com/reference/api/docker_remote_api) 与 Docker 的守护进程通信。
主机(Host)	一个物理或者虚拟的机器用于执行 Docker 守护进程和容器。
仓库 (Registry)	Docker 仓库用来保存镜像，可以理解为代码控制中的代码仓库。Docker Hub(https://hub.docker.com) 提供了庞大的镜像集合供使用。
Docker Machine	Docker Machine是一个简化Docker安装的命令行工具，通过一个简单的命令行即可在相应的平台上安装Docker，比如VirtualBox、 Digital Ocean、Microsoft Azure。

3、安装前准备工作

3.1 Linux 版本

1、Linux 的版本是以 Centos7 为主.,命令如下：

```
cat /proc/version
```

2、命令输出的结果信息

```
[root@localhost /]# cat /proc/version
Linux version 3.10.0-957.12.2.el7.x86_64
[root@localhost /]#
```

3.2 Docker 版本

1、查看 docker 的版本命令

```
docker    version
```

2、命令输出的结果信息

```
[root@localhost /]# docker version
Client:
 Version:           18.09.6
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        481bc77156
 Built:             Sat May  4 02:34:58 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.6
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.8
  Git commit:       481bc77
  Built:            Sat May  4 02:02:43 2019
  OS/Arch:          linux/amd64
  Experimental:     false
[root@localhost /]#
```

3.3 有哪些加速器服务进行选择呢？

1、目前有阿里、腾讯、网易云这几个巨头的容器镜像服务做得比较好，服务比较稳定，这里推荐阿里云的容器镜像服务，而且阿里在国内的技术毋庸置疑，而且经过安装过程中其实

比较顺利，如果使用其他的话不敢保证，在过程中遇到 docker 中文社区拉下来的镜像安装 gitlab 各种报错。

2、容器镜像服务

地址：https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors

3、容器镜像服务注意事项

如果没有账号的童鞋们可以通过注册一个或者淘宝号可以登录。

3.4 加速器服务配置步骤

由于 centos7 安装 Docker 之后是无 daemon.json 文件，需要自己手动创建一个

1、创建一个目录如下命令：

```
sudo mkdir -p /etc/docker
```

2、追加加速器地址到 daemon.json 文件里面命令：

```
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://fxde.mirror.aliyuncs.com"]
}
EOF
```

3、重新加载 daemon 文件命令：

```
sudo systemctl daemon-reload
```

4、重启 docker 服务命令：

```
sudo systemctl restart docker
```

4、安装 Gitlab 准备工作

4.1 获取 gitlab 镜像包

```
docker pull gitlab/gitlab-ce
```

这里从阿里镜像服务里面拉取 gitlab 镜像有点大，需要耐心等待,下载完镜像之后通过 docker 命令可以看到它的大小,可以看到它的文件有 1.85G,为什么有那么大呢？ 因为 gitlab 集成了很多依赖软件

4.2 在本机准备 gitlab 工作目录

```
mkdir -p /home/gitlab/config    创建 config 目录
mkdir -p /home/gitlab/logs      创建 logs 目录
mkdir -p /home/gitlab/data      创建 dat 目录
```

4.2 运行脚本启动 GitLab

```
docker run --detach \
    --hostname 192.168.1.106 \
    --publish 7001:443 --publish 7002:80 --publish 7003:22 \
    --name gitlab --restart always \
    --volume /home/gitlab/config:/etc/gitlab \
    --volume /home/gitlab/logs:/var/log/gitlab \
    --volume /home/gitlab/data:/var/opt/gitlab 8e28c88b6a21
```

参数说明：

参数名称	参数说明

detach	指定容器运行于前台还是后台
hostname	指定主机地址，如果有域名可以指向域名
publish	指定容器暴露的端口,左边的端口代表宿主机的端口，右边的是代表容器的端口
name	给容器起一个名字，
restart always	重启
volume	数据卷，在 docker 中是最重要的一个知识点.

备注：8e28c88b6a21 代表阿里云拉下的镜像 Id 这里只列举上面脚本的，详情请看官方文档.
https://docs.docker.com/engine/reference/commandline/docker/

4.3 修改 gitlab.rb 配置文件

按上面的方式，gitlab 容器运行没问题，但在 gitlab 上创建项目的时候，生成项目的 URL 访问地址是按容器的 hostname 来生成的，也就是容器的 id。作为 gitlab 服务器，我们需要一个固定的 URL 访问地址，于是需要配置 gitlab.rb（宿主机路径：/home/gitlab/config/gitlab.rb）配置有三个参数如：

```
external_url 'http://192.168.1.106'
gitlab_rails['gitlab_ssh_host'] = '192.168.1.106'
gitlab_rails['gitlab_shell_ssh_port'] = 703
```

4.4 进去 gitlab 容器重启服务

```
docker exec -it gitlab /bin/bash    进去 gitlab 容器的命令
gitlab-ctl reconfigure    重置 gitlab 客户端的命令
```

由于我们运行是使用数据卷参数进行运行的，宿主机的 gitlab.rb 文件修改了，gitlab 的文件会跟着改，但是容器的文件不会跟着生效，必须要进去容器里面进行命令执行，重置配置文件比较耗费时间，需要耐心等待，如果时间比较短说明成功率不高，而且进去容器之后就退出啦。

```
[root@localhost /]# docker exec -it gitlab /bin/bash
root@192:/# gitlab-ctl reconfigure
Starting Chef Client, version 13.6.4
resolving cookbooks for run list: ["gitlab"]
Synchronizing Cookbooks:
  - gitlab (0.0.1)
  - package (0.1.0)
  - postgresql (0.1.0)
  - redis (0.1.0)
  - registry (0.1.0)
  - mattermost (0.1.0)
  - consul (0.1.0)
  - gitaly (0.1.0)
  - letsencrypt (0.1.0)
  - nginx (0.1.0)
  - runit (4.3.0)
  - acme (3.1.0)
```

```
    * ruby_block[disable grafana] action run (skipped due to only_if)
     (up to date)
Recipe: gitlab::deprecate-skip-auto-migrations
  * file[/etc/gitlab/skip-auto-reconfigure] action create (skipped due to only_if)
  * ruby_block[skip-auto-migrations deprecation] action run (skipped due to only_if)

Running handlers:
Running handlers complete
Chef Client finished, 4/598 resources updated in 01 minutes 04 seconds
gitlab Reconfigured!
```

备注：如上信息如果出现了，恭喜你在 docker 安装 Gitlab 成功通过了.

4.5 重启 gitlab 容器命令

docker restart gitlab 命令

[root@localhost /]#docker restart gitlab　　这里重启容器也需要耐心等待.

4.7 检查启动信息

docker ps 命令

[root@localhost /]# docker ps

```
-bash: doccker: command not found
[root@localhost /]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                    PORTS                                                                NAMES
fbf2081f9773        8e28c88b6a21        "/assets/wrapper"        6 hours ago         Up 6 hours (healthy)      0.0.0.0:7003->22/tcp, 0.0.0.0:7002->80/tcp, 0.0.0.0:7001->443/tcp    gitlab
7d567701da7f        redis:3.2           "docker-entrypoint.s…"   13 hours ago        Up 13 hours               0.0.0.0:6379->6379/tcp                                               gifted_grothendieck
[root@localhost /]#
```

4.6 再查看本机端口状态

netstat -tnl 命令

```
[root@localhost /]# netstat -tnl
```

```
[root@localhost /]# netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
tcp6       0      0 :::6379                 :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::7001                 :::*                    LISTEN
tcp6       0      0 :::7002                 :::*                    LISTEN
tcp6       0      0 :::7003                 :::*                    LISTEN
[root@localhost /]#
```

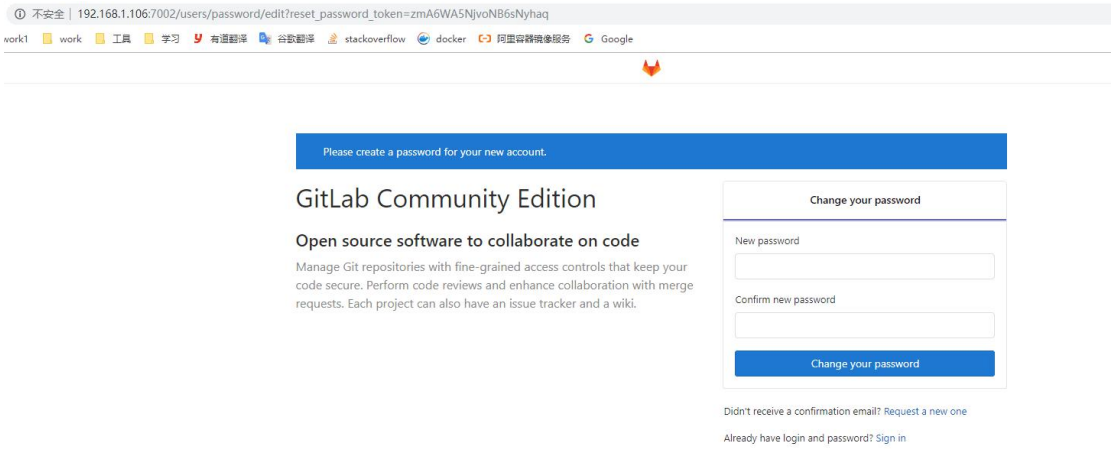
4.7 GitLab 命令

```
gitlab-ctl reconfigure      // 重新应用 gitlab 的配置
gitlab-ctl restart          // 重启 gitlab 服务
gitlab-ctl status           // 查看 gitlab 运行状态
gitlab-ctl stop             // 停止 gitlab 服务
gitlab-ctl tail             // 查看 gitlab 运行日志
```

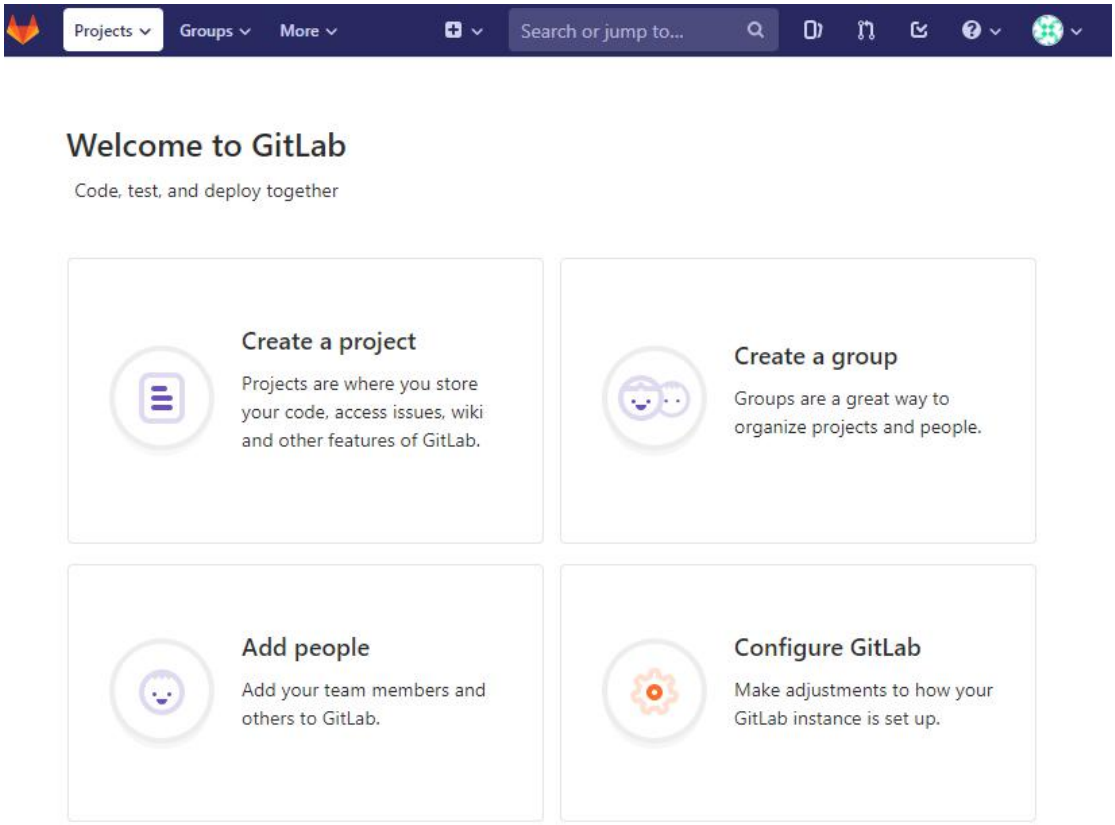
4、打开 GiltLab

5.1 打开浏览器

浏览器输入 http://192.168.1.106:7002，如果出现了此界面无问题了，
由于 gitlab 安装之后需要重置密码，



5.2 GitLab 主界面



5、总结与建议

1 、以上问题都是根据搭建 GitLab 实际情况进行总结整理，除了技术问题查很多网上资料

通过进行学习之后梳理。

2、 在学习过程中也遇到很多困难和疑点，如有问题或误点，望各位老司机多多指出或者提出建议。本人会采纳各种好建议和正确方式不断完善现况，人在成长过程中的需要优质的养料。

3、 希望此文章能帮助各位老铁们更好去了解如何在 Docker 里面安装 GitLab，也希望自己看了此文档或者通过找资料进行手动安装效果会更好。