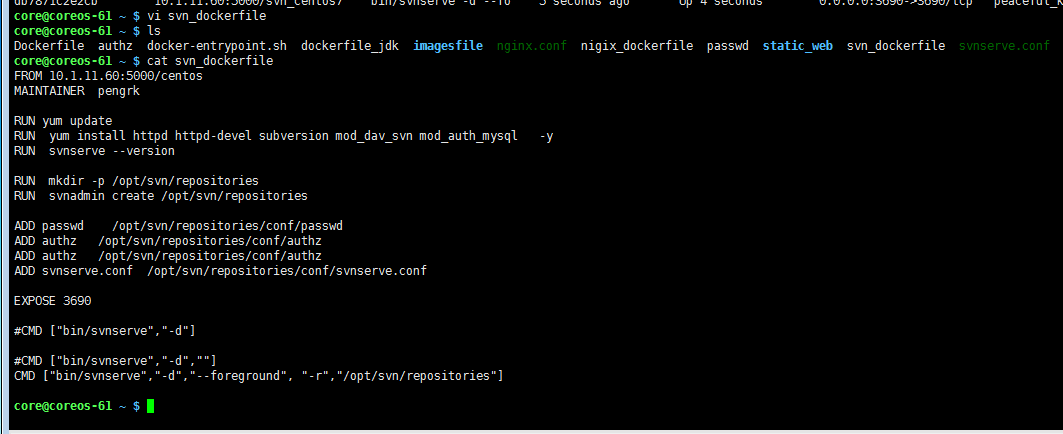
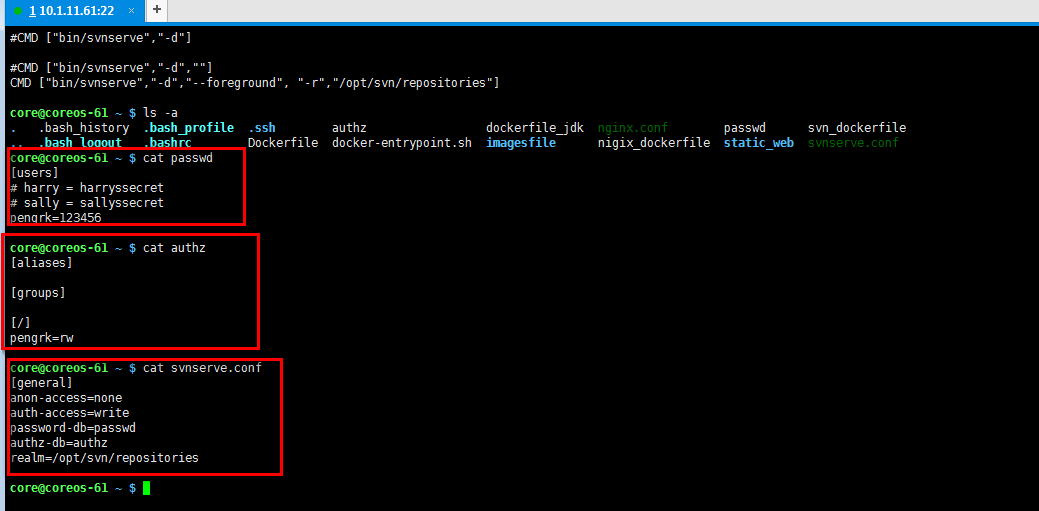
## 创建svn\_dockerfile



|  |
| --- |
| FROM 10.1.11.60:5000/centos  MAINTAINER pengrk  RUN yum update  RUN yum install httpd httpd-devel subversion mod\_dav\_svn mod\_auth\_mysql -y  RUN svnserve --version  RUN mkdir -p /opt/svn/repositories  RUN svnadmin create /opt/svn/repositories  ADD passwd /opt/svn/repositories/conf/passwd  ADD authz /opt/svn/repositories/conf/authz  ADD authz /opt/svn/repositories/conf/authz  ADD svnserve.conf /opt/svn/repositories/conf/svnserve.conf  EXPOSE 3690  CMD ["bin/svnserve","-d","--foreground", "-r","/opt/svn/repositories"] |

## 创建dockfile中copy的文件



passwd

|  |
| --- |
| [users]  # harry = harryssecret  # sally = sallyssecret  pengrk=123456 |

authz

|  |
| --- |
| [aliases]  [groups]  [/]  pengrk=rw |

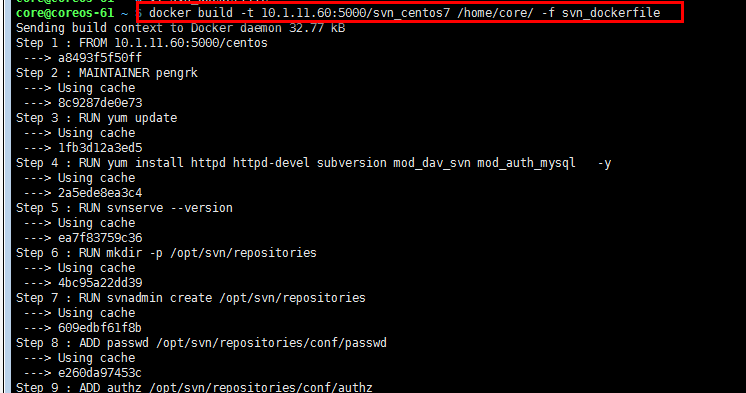
svnserve.conf

文件中要每行要顶格。不然访问不了

|  |
| --- |
| [general]  anon-access=none  auth-access=write  password-db=passwd  authz-db=authz  realm=/opt/svn/repositories |

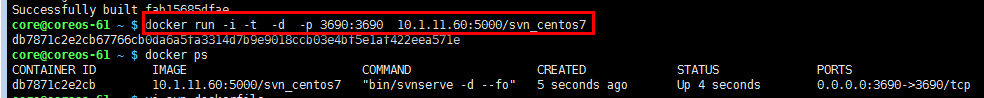
## build镜像

docker build -t 10.1.11.60:5000/svn\_centos7 /home/core/ -f svn\_dockerfile



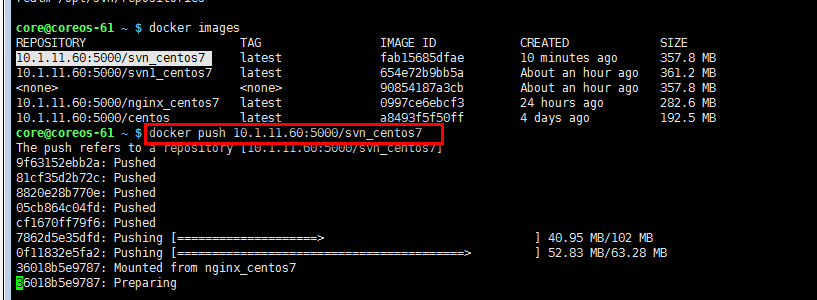
## run

docker run -i -t -d -p 3690:3690 10.1.11.60:5000/svn\_centos7



## push

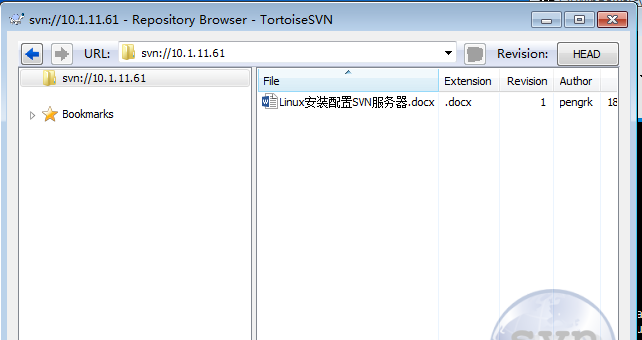
docker push 10.1.11.60:5000/svn\_centos7



yum install net-tools

## 测试

Docker的服务是10.1.11.61，其svn的地址为svn://10.1.11.61



https://github.com/lejingw/docker-svnserver/blob/master/Dockerfile

## 参考1：

http://www.cnblogs.com/puloieswind/p/5856326.html

**1. 安装SVN服务器：**

检查是否已安装

# rpm -qa subversion

安装SVN服务器

# yum install httpd httpd-devel subversion mod\_dav\_svn mod\_auth\_mysql

验证安装

# cd /etc/httpd/modules

# ls | grep svn

mod\_authz\_svn.so

mod\_dav\_svn.so

查看版本

# svnserve --version

**2. 代码库创建：**

安装完成后要建立SVN库

# mkdir -p /opt/svn/repositories

# vi

执行后，自动建立repositories库，查看/opt/svn/repositories文件夹包含了conf，db，format，hooks，locks，README.txt等文件，说明一个SVN库已经建立。

**3. 配置版本库：**

进入上面conf文件夹下，进行配置：

**a. 用户密码passwd配置：**

# vi + passwd //+表示光标放在文件最低端

修改passwd为一下内容：

[users]

# harry = harryssecret

# sally = sallyssecret

zhoulf=123456

**b. 权限控制authz配置：**

# vi + authz

设置哪些用户可以访问哪些目录，向authz文件追加以下内容：

[/]

zhoulf=rw //给该用户访问所有库的权限  
 或  
[repositories:/project] //repository库的根目录权限  
zhoulf=rw

/ 表示根目录及以下，根目录是svnserve启动时指定的，我们指定的是/opt/svn；/ 就是指对全部版本库都具有权限

repositories：/ 表示对库repositories的根目录设置权限

PS：

\* 权限配置文件中出现的用户名必须已在用户配置文件中定义。

\* 对权限配置文件的修改立即生效，不必重启svn。

**c. 服务svnserve.con配置：**

# vi + svnserve.conf

添加一下内容：

[复制代码](javascript:void(0);)

[general]

#匿名访问的权限，可以是read,write,none,默认为read

anon-access=none

#使授权用户有写权限

auth-access=write

#密码数据库的路径

password-db=passwd

#访问控制文件

authz-db=authz

#认证命名空间，subversion会在认证提示里显示，并且作为凭证缓存的关键字

realm=/opt/svn/repositories

[复制代码](javascript:void(0);)

这里注意各标签不能错，也不能有重复，不然无法连接。

**d. 配置防火墙端口（如果需要）：**

不一定每个人都需要设置，可以先测试后再看是否需要打开端口

# vi /etc/sysconfig/iptables

添加一下内容：

-A INPUT -m state --state NEW -m tcp -p tcp --dport 3690 -j ACCEPT

保存后重启防火墙

# service iptables restart

**4. 查看：**

**a. 启动SVN**

# svnserve -d -r /opt/svn/repositories

**b. 查看SVN进程**

# ps -ef|grep svn|grep -v grep

root 12538 1 0 14:40 ? 00:00:00 svnserve -d -r /opt/svn/repositories

**c. 检测SVN端口**

# netstat -ln |grep 3690

tcp 0 0 0.0.0.0:3690 0.0.0.0:\* LISTEN

**5. 停止重启SVN：**

# killall svnserve //停止

# svnserve -d -r /opt/svn/repositories // 启动

**6. 测试连接：**

使用TortoiseSVN进行测试：

1. SVN服务启动后，需要使用客户端测试连接：

　　　　客户端连接地址：svn://192.168.15.231；然后，输入用户名密码；

2. 新建一个文件夹，即本地的库文件夹，右键checkout，将会得到一个隐藏文件夹.svn；

3. 在此文件夹中放入项目内容，然后右键点击commit，就可以上传本地项目了。

CMD ["nginx", "-g", "daemon off;"]

yum install net-tools

## 参考2：

通常，很多Linux发行版已经自带了svn软件，安装系统时候选择即可。因此，本文并不讲解如何安装svn相关软件，而是重点总结一下如果使用svnserve服务器来构建自己的代码仓库，并且允许别人远程访问。

      本文的整个内容参考了svn手册。 在svnserve的配置方面参考了以下链接的内容：

<http://blog.tom.com/silingshengren/article/1331.html>。

     本文欢迎自由转载，但请标明出处和本文链接，并保持本文的完整性。

     CU: Godbach。

     Apr 6, 2009

一、新建并初始化版本库

1.新建代码仓库

# svnadmin create /home/myrepos

这样，我们就在/home目录下新建了版本库myrepos。

2.初始化代码仓库

#mkdir -p code/{branches,tags,trunk}

按照svn手册上的建议，我们版本库的布局采用branches/tags/trunk这样的三个目录。  
然后可以将代码直接拷贝的trunk目录，也可以随后采用svn add的方式添加。

#svn import code file: ///home/myrepos

将我们构造好的版本库布局（以及代码，如果已经拷贝到trunk下的话）提交到版本库。  
由于这里我们是在本地操作，因此可以直接使用file协议方式提交。  
格式为：file:// + 版本库绝对的路径

OK，到这里我们已经构建了一个本地版本库，可以再本地使用svn ci的方法得到版本库  
的副本了。如果前面只是提交了版本库布局，而没有提交代码的话，则可以通过如下命  
令，提交版本代码：

#svn co file: ///home/myrepos/trunk repos

通常只从主线目录trunk下取出源码，放在了新建的repos目录。然后可以将源码拷贝  
到repos目录。

执行以下两条命令将代码添加的版本库。

#svn add \*

#svn ci \* -m "Committed codes"

首先需要将拷贝的代码加入到当前的副本中，然后将代码提交到版本库。

二、配置并启动svnserve

1. 配置svnserve

这一步主要完成远程用户访问版本的权限。

在版本库目录下有个conf目录：home/myrepos/conf，该目录包含三个文件：  
authz，passwd，svnserve.conf。

（1）svnserve.conf

此文件重点的设置内容如下：

anon-access = read

auth-access = write

password-db = passwd

authz-db = authz

其中，anon-acces s和auth-access分别代表设置匿名用户和授权用户的访问权限。  
read代表只读，write代表可读可写，还有一个none，代表禁止访问。如果不想让匿名用户  
访问的话，可以设置anon-access = none。

password-db为保存版本库的用户名和密码，通常为conf目录下的passwd文件。

authz-db指向的文件，主要用于将用户进行分组，并赋予不同的组不同的权限。  
譬如有多个目录时，设置某个组的用户只可以访问其中的那些目录。

（2）passwd

如前面介绍，这个文件的主要内容如下

[users]

harry = harrysecret

sally = sallyssecret

以上两行就是可以访问版本库的用户名和密码。格式为：user = passwd。

（3）authz

[groups]

# harry\_and\_sally = harry,sally

# [/foo/bar]

# harry = rw

# \* =

# [repository:/baz/fuz]

# @harry\_and\_sally = rw

# \* = r

以上就是authz文件的格式。先给用户分组，然后分别设置每个组的工作目录以及权限等。

本文的讲解并未用到authz的配置。

2. 启动svnserve

#svnserve -d -r /home/

svnserve将以守护进程的方式运行。同时，通过-r选项指定了版本库的目录。这个目录可以  
是版本库的上级目录（/home），也可以是版本库的目录（/home/myrepos）。具体的区  
别会在后面讲到。当然，也可以不用-r指定。

三、远程访问svnserve。

假设版本服务器的地址为10.1.1.1，那么用户可以在远程linux端输入命令：

#svn checkout svn://10.1.1.1/myrepos myrepos

这样版本库的所有内容都被check出来，存放在当前目录myrepos下。这里相当于使用svn  
协议，不用搭建http服务器。

如果启动svnserve的时候，-r选项指定的是版本库的目录，譬如/home/myrepos，那么检出  
版本的时候，执行的命令为：

#svn checkout svn://10.1.1.1/ myrepos

如果启动svnserve是没有指定-r选项，则要输入版本库的绝对路径：

#svn checkout svn://10.1.1.1/home/myrepos myrepos

四、容易出现的问题

在访问svnserve时，遇到了不少的问题，大致总结一下：

1. 这一步最容易避免，一定要记得启动svnserve。

2. 记得要正确的修改配置文件svnserve.conf和passwd。

3. 保证版本库服务器没有配置限制远程访问svnserve的规则。有时在访问svn服务器的时候，会提示服务器积极拒绝链接。简单起见，可以看看iptables -L INPUT和iptables -L OUTPUT有没有限制访问的规则。

4. 要保证输入版本库的URL正确。

后记：

1. 版本库的初次导入可以采用多种方式：

（1）直接构建好版本库目录，并将源码拷贝的对应的位置，然后svn import即可；

（2）先用svn checkout取出版本，然后构建目录和源码，执行svn add 并且svn commit即可。

2. 导入或者提交文件类型的控制

    svn客户端是可以配置忽略哪些些文件，比如说我们不想提交\*.o之类的文件。那么可以通过设置svn:ignore或者global-ignore来进行过滤相关的文件。通常默认的情况就会过滤掉\*.o, \*.so,\*.a等被svn认为是中间生成的文件。我本人使用的客服端不管是WIN的还是Linux的都是这种默认情况。

    但是，如果有时基于某种需要，或者是导入一个别人之前维护的工程，里面有些\*.so，\*.a文件必须保留，那么如果在第一次import的时候全部导入呢？曾经问过有些朋友说，svn add或者svn import是可以直接把这种文件导入到版本库的。但是我个人进行测试的时候，确实导入不进去，除非你直接svn add这个文件，然后提交应该就可以了。

    如果这样的文件比较多，而且比较分散，那么如何解决呢？看一下svn import和svn add这两个命令的帮助，都有--no-ignore选项。这个选项的功能就是执行相应动作的时候，不忽略任何文件。那么执行以下两条命令：

svn import [source] [repository] --no-ignore

或

svn add [source] --no-ignore

会自动将source目录下所有的内容全部import或者add。

    因此，通过--no-ignore选项，可以满足提交一些特殊文件的需求。

3. 用一个旧版本的svnadmin创建的版本库，直接拷贝到另外一台svn版本比较高的PC上使用时，发现提示数据库版本不匹配。

解决的方法：

（1）导出旧的版本库内容，包含版本信息

具体使用的命令为svnadmin dump, 详细使用方法参考手册。该命令的格式为：

svnadmin dump REPOS\_PATH [-r LOWER[:UPPER]] [--incremental]

如果我将版本库完整的信息都倒出来的话，执行如下命令：

# svnadmin dump old\_repos > repos.dump

默认版本库dump的时候是输出在终端的，通过重定下输出到repos.dump文件里面。该文件实际是文本文件。

（2）导入旧版本库的内容

在需要导入的位置，新建一个版本库

# svnadmin create new\_repos

然后将repos.dump文件导入到新的版本库：

# svnadmin load new\_repos < repos.dump

该命令成功执行完毕之后，新的版本库中已经包含了old\_repos版本库中的所有版本内容。

svnadmin dump和svnadmin load命令都支持导入/导出某个指定revsion或者某个指定范围的所有revision。

该部分后记参考了如下连接，在此表示感谢：

http://hi.baidu.com/primechen/blog/item/71a7a1dc2b981fe877c63841.html