IVI gerrit manual for git user

Version: 0.1 (not official)
Author: son.nguyen

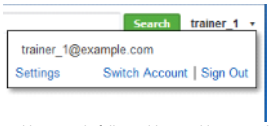
Tuesday, July 12, 2016 2:29 PM

Requirements and background knowledge:

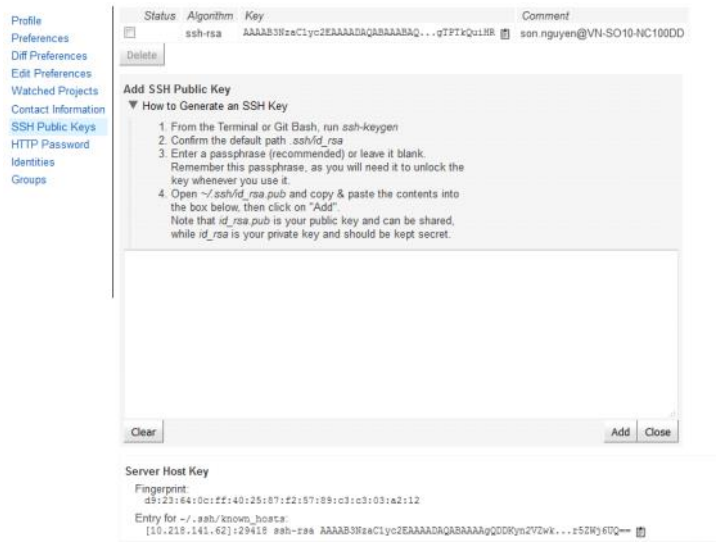
- git document:
  - For studious guy:
    - https://git-scm.com/doc
    - https://www.atlassian.com/git/tutorials/
  - For lazy guy like me: http://rogerdudler.github.io/git-guide/
  - For guy who love playing game - (like me again): https://try.github.io
- download and install git: https://git-scm.com/download

Optional:
- https://tortoisegit.org/ very useful git with GUI
- create a personal account and repository on github.com and play around (do it at home, don't push company code to github :))

1. Login into gerrit webpage:
   a. http://10.218.141.62:8080
   b. Choose: trainer_1@example.com account (listed after login textboxes)
2. Go to setting page:

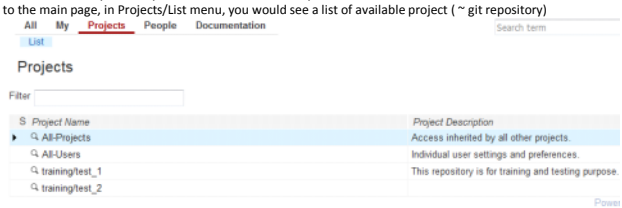


3. In SSH Public Keys tab, follow Add SSH Public Key guide to add new SSH key



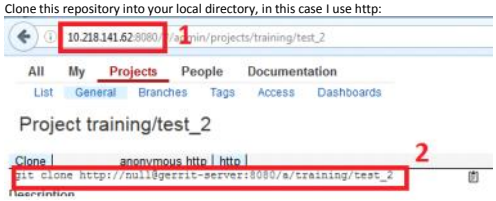
Note that you can add multiple SSH key into the same account, each computer should have only one identical SSH key, do not remove other keys unless you are the owner of this key.

4. Back to the main page, in Projects/List menu, you would see a list of available project (~ git repository)



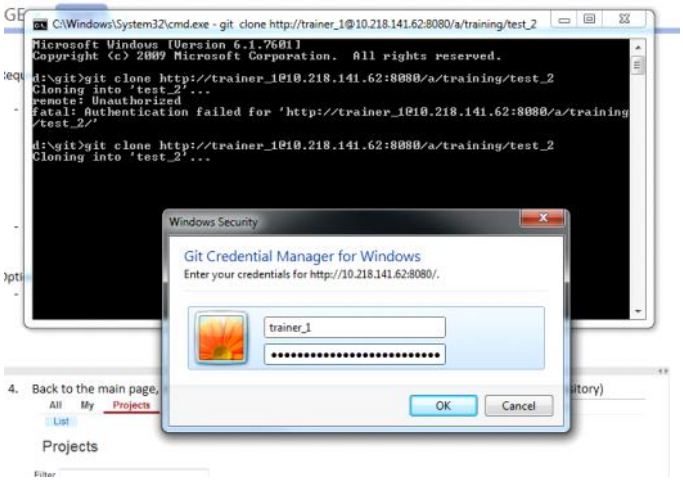
5. Choose training/test_1 project (I accidently cropped the next images from test_2, please notice the differences)

5. Clone this repository into your local directory, in this case I use http:



In the git bash command prompt, copy the http git clone command and edit:
null -> trainner_1
gerrit-server -> IP of gerrit server (box 1)

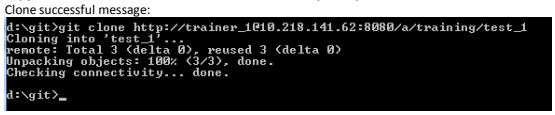
In case a login box appears like this, don't close, process the next step



7. Back to gerrit User setting page, HTTP Password tab, you would see

Copy the Username and Password into Windows Security Prompt windows.

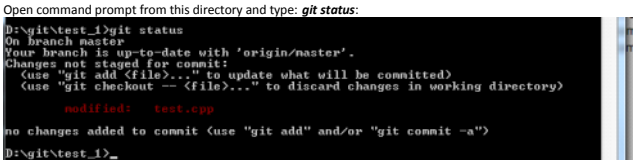
8. Clone successful message:



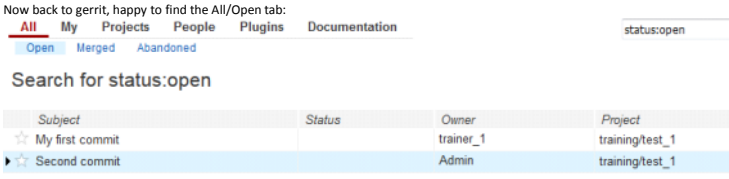
Note: The step 3 is necessary if you pull and push the code using SSH, gerrit and your local git will verify your computer using this SSH key.

---

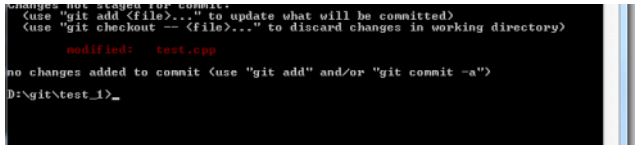
9. Now make some changes in your local repository.
   a. In the test_1 folder (in your computer directory), you would see a test.cpp file, this file is cloned from the trainning/test_1 repository.
   b. Make some changes on this file using any text editor.
   c. Open command prompt from this directory and type: ***git status***:



12. Now back to gerrit, happy to find the All/Open tab:

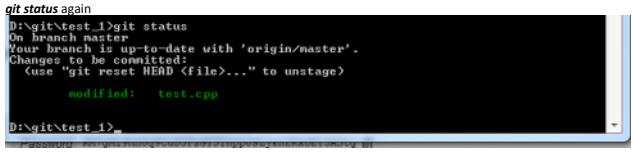


See your commit? yes, exactly your commit. The most recent commit will be on top.

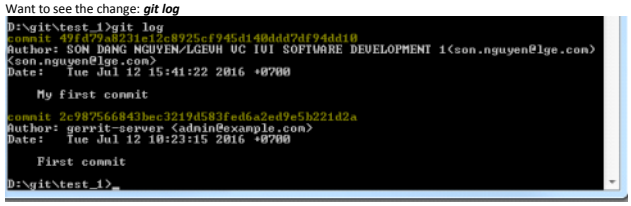


This means that you have modification(s) on your local branch and have not committed the change(s)

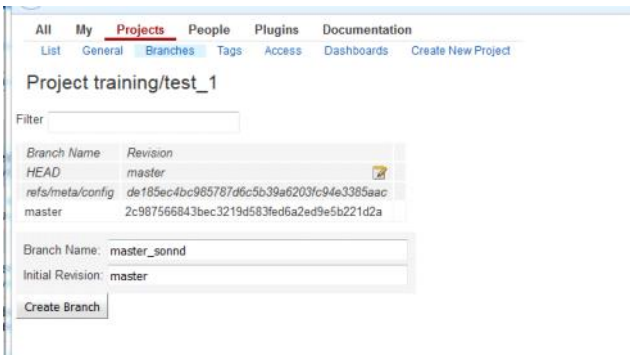
d. Add the change to the commit: ***git add ****
e. Notice that * means you will add all the file|changes|modification into commit ready state, you can choose single file or specific files with commands like:
   i. *git add *.txt*
   ii. *git add myNewFile.h*
f. ***git status*** again



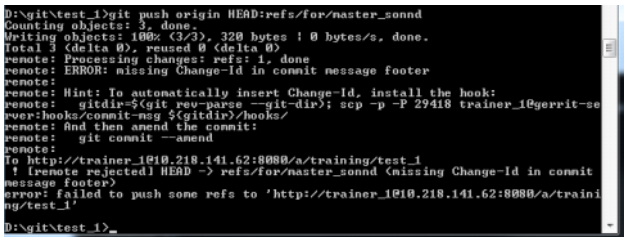
10. Now commit the code:
    a. ***git commit -m "My first commit"*** with -m means your commit message
    b. Now your modification(s) are safe (maybe) on your local repository
    Want to see the change: ***git log***



11. Push to gerrit
    a. Now back to gerrit webpage, on Projects, choose *training/test_1* project, move to *Branches* tab. (Try to find it yourself, honestly after 2 years of pushing my dumb code to gerrit I still unable to understand the logical of those menus and tabs, rofl)

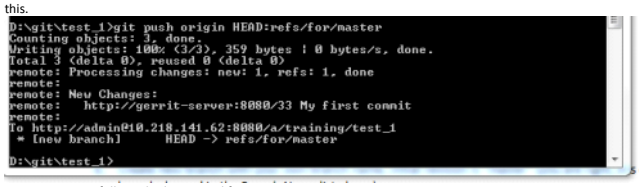


b. Create new branch with your own branch name, initial revision is "master" (you are right, it is exactly the master branch showed in the Branch Name list above).
c. Now push your local changes into gerrit:
git push origin HEAD:refs/for/master_sonnd *<- see the logic? Yeah, not too hard.*

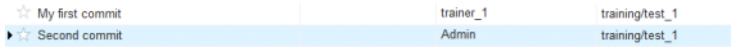


See the error? Congratulation, you are on track, follow the Hint on the error message and commit again. Believe me, you don't have to research the reason that cause the error, just follow the Hint. (ah, yes, replace**gerrit-server** with gerrit server address, and remember to use **git bash**)

d. Now, back to the part c, look at the code, you will ask why**origin**? Ok, type **git remote -v**, it show that **origin** is just the name of the remote repository, you can actually change it to whatever you want.
e. What is ***HEAD:refs/for/master_sonnd*** **?** I don't know the story behind, but next section I will show you why you need this.

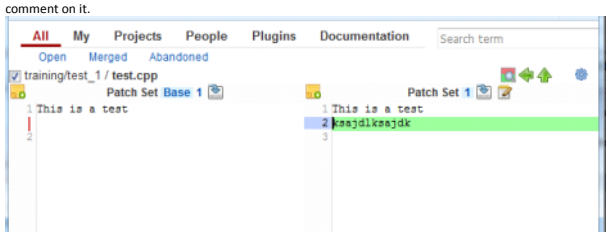


Now you successfully pushed your modification into gerrit.

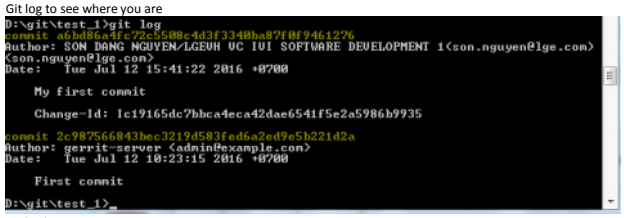


See your commit? yes, exactly your commit. The most recent commit will be on top.

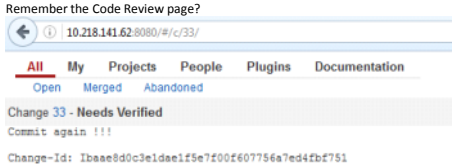
13. Take a look at Code review page by click on the commit. Check the file, it will show you what you had changed. You can even comment on it.



14. Oh, loooook! I made it wrong! The second line should be a readable text. What should I do? Don't worry (until you tell the leader or sub-leader come here and review your code, lol they will ask you much)
15. Now go back to your local repo, imagine the git branches and commits just like a linked-list. You go one wrong step, remove it, go back and take another step.
    a. Git log to see where you are



    b. Go back 1 commit: ***git reset HEAD~1***
    c. Edit file (9.b)
    d. Add the changes (9.c)
    e. Commit it again
    f. Now wait, the push command changed. Remember the last command? (***git push origin HEAD:refs/for/master_sonnd***) Now you have to update the change you already committed, how to do that?
    git push origin HEAD:refs/changes/33
    e. Yes, you should ask where the number **33** came from?
    Remember the Code Review page?



See the number 33?
Again, back to the 11.e, see the magic number? Nice!

Then we almost done the basic.

This part I will show you some commands related to git that I usually use:

- git add
- git status
- git clone
- git fetch
- git push
- git commit
- git reset --h
- git clean -dfx

- google (lol)


Follow the Hint,
and It works
Magic...

Install and setting gerrit system

Version: 0.1 (not official)
Author: son.nguyen

Thursday, July 07, 2016 9:40 AM

Requirement:

- Linux Ubuntu
- Java

References:
https://www.digitalocean.com/community/tutorials/how-to-install-gerrit-on-an-ubuntu-cloud-server
https://gerrit-review.googlesource.com/Documentation/install-quick.html
https://www.digitalocean.com/community/tutorials/how-to-install-java-on-ubuntu-with-apt-get

Maintenance:

- Restart: ~/gerrit/bin/gerrit.sh restart
- Stop: ~/gerrit/bin/gerrit.sh stop
- Start: ~/gerrit/bin/gerrit.sh start

1. Install packages:
   - sudo apt-get update
   - sudo apt-get install default-jre git git

2. Get source:

   - Create gerrit user and become the user:
     - *sudo adduser gerrit*
     - *sudo su gerrit*
   - Download gerrit:
     - cd to user home and mkdir a folder to save gerrit source: *mkdir ~/gerrit_source*, cd to gerrit_source folder.java
     - Download gerrit, check out the last stable or recommended version:
       - *wget* https://www.gerritcodereview.com/download/gerrit-2.12.3.war

```
gerrit@sonnd-VirtualBox:~/gerrit_source$ wget https://www.gerritcodereview.com/download/gerrit-2.2.2.war
--2016-07-07 09:46:47--  https://www.gerritcodereview.com/download/gerrit-2.2.2.war
Resolving www.gerritcodereview.com (www.gerritcodereview.com)... 216.58.199.110, 2404:6800:4005:803::200e
Connecting to www.gerritcodereview.com (www.gerritcodereview.com)|216.58.199.110|:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://gerrit-releases.storage.googleapis.com/gerrit-2.2.2.war [following]
--2016-07-07 09:46:48--  https://gerrit-releases.storage.googleapis.com/gerrit-2.2.2.war
Resolving gerrit-releases.storage.googleapis.com (gerrit-releases.storage.googleapis.com)... 216.58.199.112, 2404:6800:4005:803::2010
Connecting to gerrit-releases.storage.googleapis.com (gerrit-releases.storage.googleapis.com)|216.58.199.112|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27146885 (26M) [application/octet-stream]
Saving to: 'gerrit-2.2.2.war'

100%[=========================================================================================================================>] 27,146,885  469KB/s   in 57s

2016-07-07 09:47:46 (469 KB/s) - 'gerrit-2.2.2.war' saved [27146885/27146885]

gerrit@sonnd-VirtualBox:~/gerrit_source$ ls
gerrit-2.2.2.war
```

3. Initialize the site:

   - check out the name of .war file
   - *java -jar **gerrit-2.2.2.war** init --batch -d ~/gerrit*

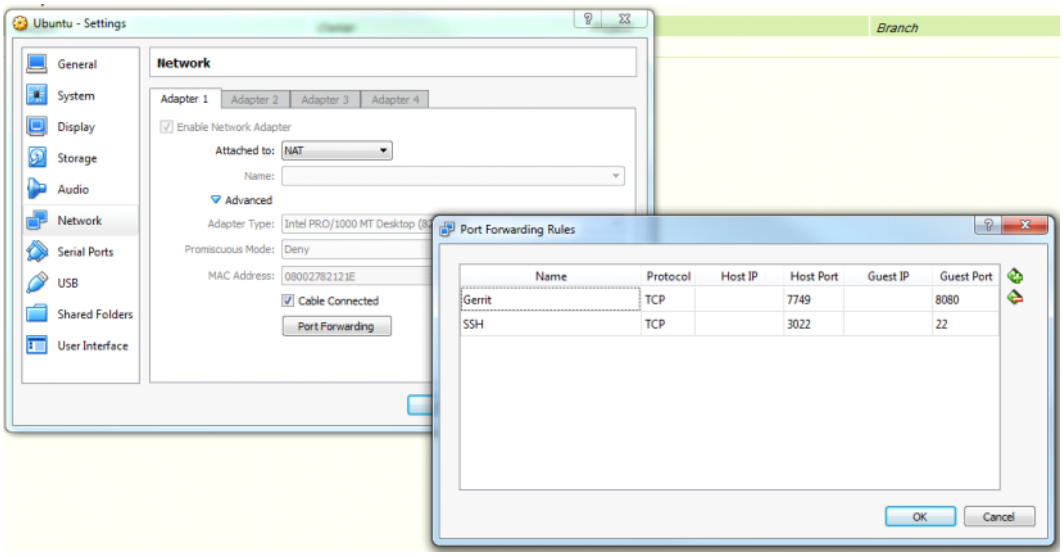
```
gerrit@sonnd-VirtualBox:~/gerrit_source$ ls
gerrit-2.2.2.war
gerrit@sonnd-VirtualBox:~/gerrit_source$ java -jar gerrit-2.2.2.war init --batch -d ~/gerrit
Generating SSH host key ... rsa(simple)... done
Initialized /home/gerrit/gerrit
Executing /home/gerrit/gerrit/bin/gerrit.sh start
Starting Gerrit Code Review: OK
gerrit@sonnd-VirtualBox:~/gerrit_source$
```

   - *When the init is complete, you can review your settings in the file '**$site_path**/etc/gerrit.config'* and exit gerrit user by *'exit'*

4. SSH check existing:
   - Check whether there are any ssh keys already. You're looking for two files, id_rsa and id_rsa.pub:
     user@host:~$ ls .ssh
     authorized_keys config id_rsa id_rsa.pub known_hosts
   - If you have the files, you may skip the key generating step.
   - If you don't see the files in your listing, your will have to generate rsa keys for your ssh sessions:

5. Generate SSH key:
   Please don't generate new keys if you already have a valid keypair! They will be overwritten!
   - *ssh-keygen*
     Note that the ssh-keygen will ask multiple times for passphrase, set null by press enter for default. Normally you will not need the passphrase.

6. NAT VirtualMachine Ethernet port
   - This step allows you access gerrit from outside the virtual machine



   - On Network setting tabs, create new port forwarding like the "Gerrit" one in the figure. Note that you can set the Host Portwhatever you like.

7. Access gerrit from your machine:
   - http://127.0.0.1:7749 (or whatever port you set)

8. Add SSH key:
   Log into the gerrit web, in the User setting page, open SSH Public Keys to add SSH generated from step 5. Usually in Windows, the id_rsa.pub located on c:\Users\[username]\.ssh\ folder, in Linux: ~/.ssh folder

IVI gerrit admin manual

Version: 0.1 (not official)
Author: son.nguyen

Tuesday, July 12, 2016 2:16 PM

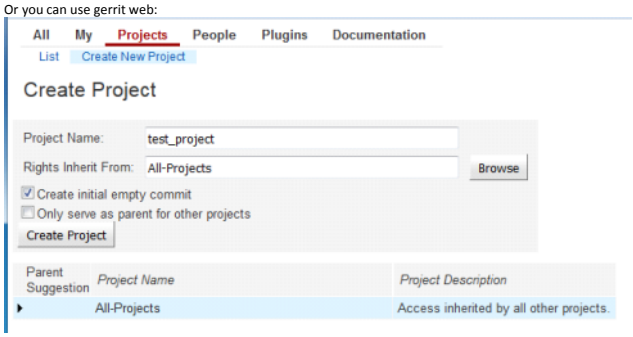
- **Create new project from scratch:**

If you choose to create a new repository from scratch, it's easier for you to create a project with an initial commit in it. That way first time setup between client and server is easier.

This is done via the SSH port:

```
user@host:~$ ssh -p 29418 user@localhost gerrit create-project test_project --empty-commit
user@host:~$
```

Or you can use gerrit web:



This will create a repository that you can clone to work with.
Note: If you create project from scratch, "**My first change**" is your next step.

- **My first change**

Download a local clone of the repository and move into it

```
user@host:~$ git clone ssh://user@localhost:29418/demo-project
Cloning into demo-project...
remote: Counting objects: 2, done
remote: Finding sources: 100% (2/2)
remote: Total 2 (delta 0), reused 0 (delta 0)
user@host:~$ cd demo-project
user@host:~/demo-project$
```

Then make a change to it and upload it as a reviewable change in Gerrit.

```
user@host:~/demo-project$ date > testfile.txt
user@host:~/demo-project$ git add testfile.txt
user@host:~/demo-project$ git commit -m "My pretty test commit"
[master ff643a5] My pretty test commit
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 testfile.txt
user@host:~/demo-project$
```

Usually when you push to a remote git, you push to the reference '/refs/heads/branch', but when working with Gerrit you have to push to a virtual branch representing "code review before submission to branch". This virtual name space is known as /refs/for/<branch>

```
user@host:~/demo-project$ git push origin HEAD:refs/for/master
Counting objects: 4, done.
Writing objects: 100% (3/3), 293 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: New Changes:
remote:   http://localhost:8080/1
remote:
To ssh://user@localhost:29418/demo-project
 * [new branch]      HEAD -> refs/for/master
user@host:~/demo-project$
```

You should now be able to access your change by browsing to the http URL suggested above,
http://localhost:8080/1

- **Already existing project on your local**

The other alternative is if you already have a git project that you want to try out Gerrit on. First you have to create the project. This is done via the SSH port:

```
user@host:~$ ssh -p 29418 user@localhost gerrit create-project demo-project
user@host:~$
```

You need to make sure that at least initially your account is granted "Create Reference" privileges for the refs/heads/* reference. This is done via the web interface in the Admin/Projects/Access page that correspond to your project.

After that it's time to upload the previous history to the server:

```
user@host:~/my-project$ git push ssh://user@localhost:29418/demo-project *:*
Counting objects: 2011, done.
Writing objects: 100% (2011/2011), 456293 bytes, done.
Total 2011 (delta 0), reused 0 (delta 0)
To ssh://user@localhost:29418/demo-project
 * [new branch]      master -> master
user@host:~/my-project$
```

This will create a repository that you can clone to work with.

References:

https://gerrit-review.googlesource.com/Documentation/install-quick.html