Gerrit introduction

- Andy Cheng
- 2018/12/6

Outline

- Introduction
- Gerrit
- Gerrit Management Strategy
- Gerrit Code Review
- Git UI
- Git Commit Message
- Git Backup Solution
- Standard Operating Procedure for Gerrit Recovery
- Convert to svn from git
- Reference

Introduction

Purpose of this document:

We briefly introduce about how Gerrit work and management strategy of Gerrit.

Introduction

Statement 1: How we work with Gerrit

Statement 2: User under Gerrit.

Statement 3: Project management of Gerrit

Statement 4: Gerrit code review

Gerrit

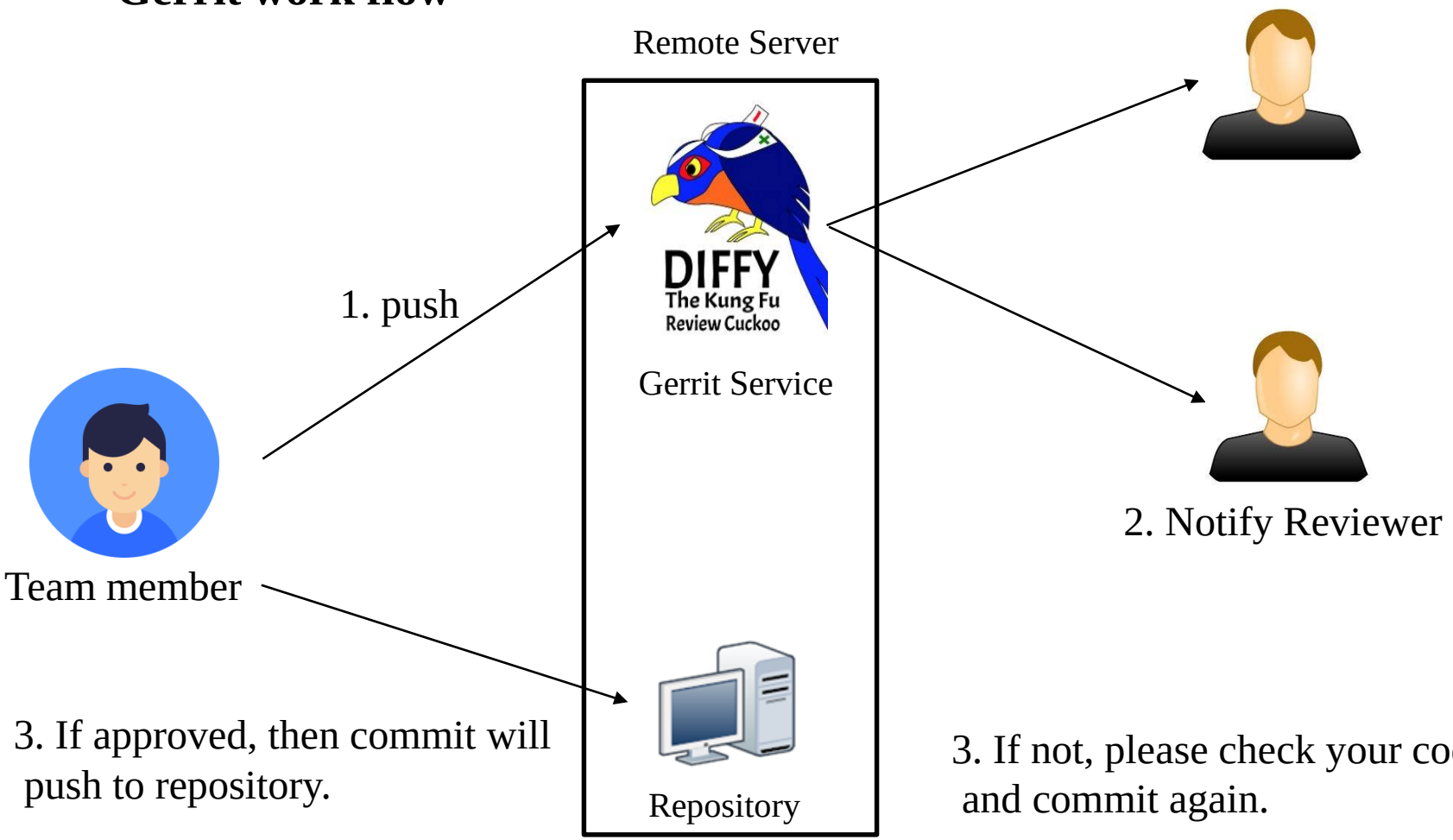
What is Gerrit ?

Gerrit is a web application based on git which is a tool for code review.

We work on local site and push commit to Gerrit, then it notifies all approvers to review your commit.

Gerrit

Gerrit work flow



Gerrit management strategy (for administrator)

Administrator of Gerrit

Administrator

1.Create account for user.

2.Assign team member to there group.

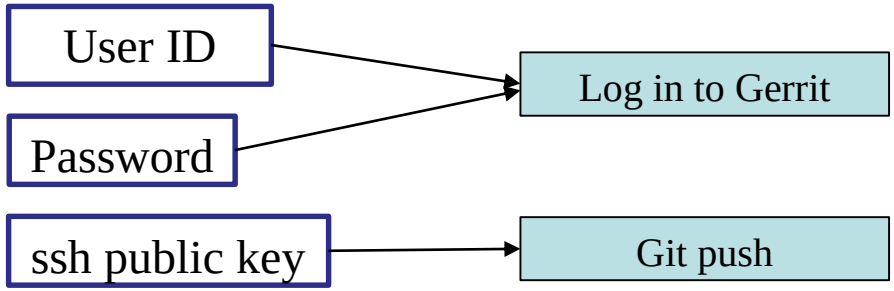
3.Manage the project

Note: only administrator could create & delete project

Gerrit management strategy (for administrator)

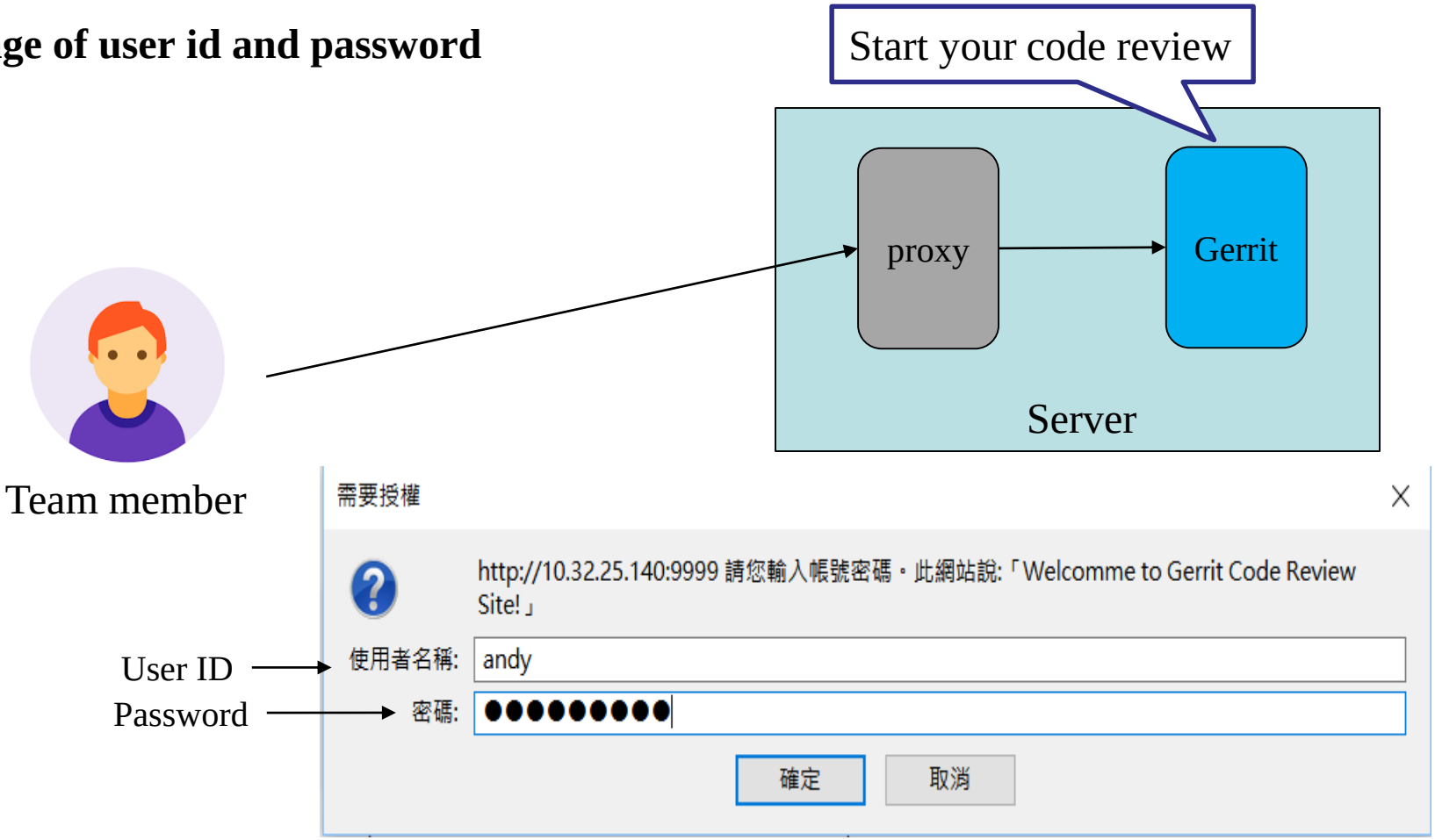
1.Create account for user

To create a account for user, it’s necessary for following items below.



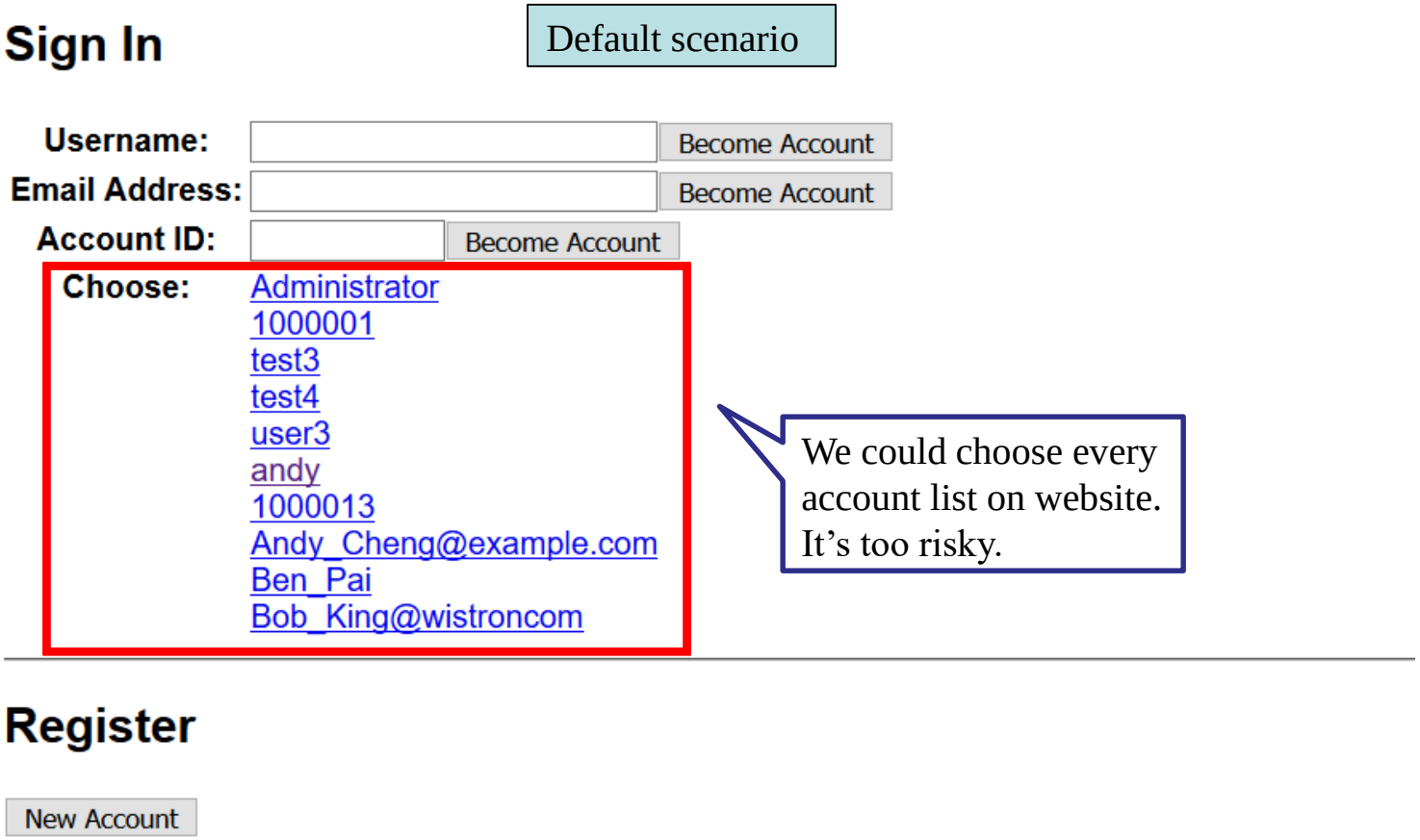
Gerrit management strategy (for administrator)

Usage of user id and password



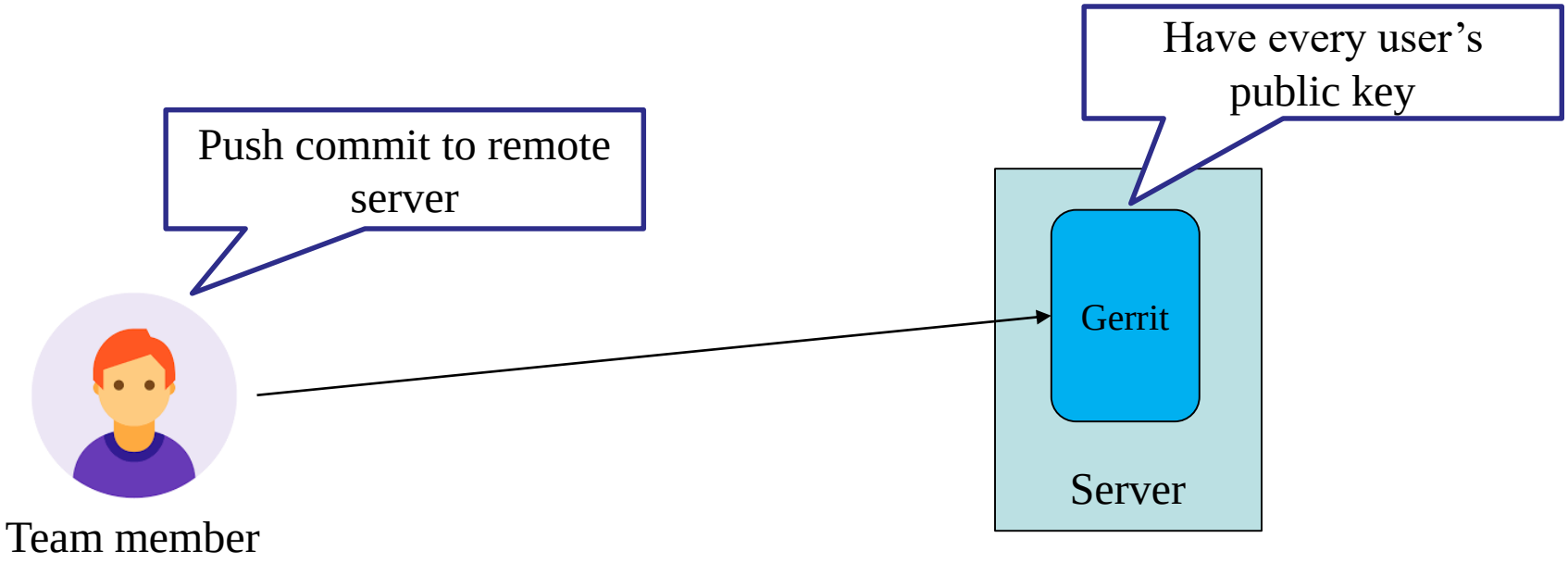
Gerrit management strategy (for administrator)

By default, Gerrit doesn't provide any protection for user account. So we need an authentication to assure user account won't be access by other members (proxy).



Gerrit management strategy (for administrator)

Usage of public key



Command:

```
push  ssh://User_ID@gerrit.com:29418/project  HEAD:refs/for/branch
```

Gerrit management strategy (for administrator)

Set ssh public key

First, every member need to make their ssh public key on linux system by using ssh-keygen –t rsa command and copy your key to administrator.

Our ssh connection use rsa algorithm.

Gerrit management strategy (for administrator)

Set ssh public key

```
ben@openbmc-server:~$ ssh-keygen -t rsa -C "Andy_CHeng@wistron.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ben/.ssh/id_rsa):
/home/ben/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ben/.ssh/id_rsa.
Your public key has been saved in /home/ben/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:rkKI8J9N5pza1M5xebyFd8V0TfJKDLYvWeS+5K1MwUQ Andy_CHeng@wist
The key's randomart image is:
+---[RSA 2048]----+
|              E  |
|             .   |
|            o + .|
|.          . O =.|
|.o .    S   . B *|
|. o . oo   o B =.|
|   o B..+ o * O o|
|    +o=+ o . X + |
|    .oo o   . =..|
+----[SHA256]-----+
```

We give every element by default (press enter).

Gerrit management strategy (for administrator)

Set ssh public key

```
ben@openbmc-server:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDEX/e2OXxYbVL8QVzLtMsahb1BhiPiLHOdZyPVTRcV
r9yBbZGNnxkZ9+fztZke27QQfT14EWoELWtv1JpsFk32vpG2nbPZtMs2/z41XEpLAFKjYzBTKANWJjrF
nYPnnvzbYArQJZBjTZnKcc4KgOO5yIMyzstHfG1SF2yGGsJR5gpzWqOnW7yur4okA1hnEscnkpGyBH2U
A1p6YuOG3upvRjELr4pLkWwvJ5JO5Irfi1DFcckQpPuXK+Ofd5HObIyp+uuO4u/f2oq2THVEPn/RE21h
mcL7y+oRIvx+e8S2Hxk5gjQvnn51Q4re9UCsKt7OqSrlDynQGHySBe/i1Fpn Andy_CHeng@wistron.
com
```

Copy this section to your administrator then setting is finished.

Gerrit management strategy (for administrator)

After create an account, user could open their browser and log into Gerrit by User ID and password which is set by user.

Gerrit management strategy (for project owner & administrator)

2.Assign team member to there group.

Administrators

Administrator that supervise the project

Non-Administrators

Ordinary User

Create specific group for your team member and assign to their group

Gerrit management strategy (for project owner & administrator)

Assign team member to there group.

All My Projects People Plugins Documentation

List Groups Create New Group

Groups

Filter

Group Name	Description
AMI_member	
Administrators	Gerrit Site Administrators
Non-Interactive Users	Users who perform batch actions on Gerrit
OpenBMC_member	
test2_user	
test_user	

Group for administrator

Group for ordinary user

Gerrit management strategy (for project owner & administrator)

How to delete group from Gerrit ?

Step 1: enter the database of Gerrit

```
ssh -p ${GERRIT_PORT} ${GERRIT_USER}@${GERRIT_HOST} gerrit gsql
```

Step 2: search GROUP_ID

```
select * from ACCOUNT_GROUP_NAMES;
```

Step 3: remove the group by GROUP_ID

```
delete from ACCOUNT_GROUP_NAMES where GROUP_ID=${GROUP_ID};
```

```
delete from ACCOUNT_GROUPS where GROUP_ID=${GROUP_ID};
```

Gerrit management strategy (for administrator)

3.Project management

As administrator, we have to do following of things below

- Create project for user
- Set permissions of project
- Assign a project owner

Gerrit management strategy (for project owner & administrator)

List of permission

Label Code_Review → Modify the range of score.

Abandon → Abandon code review which is not approved.

Add Patch Set → To let user add patch file.

Create Reference → Create reference to add other permission.

Create Annotated Tag → Push tag to branch.

Delete Reference → Delete reference in project.

Delete Changes → Remove code review from Gerrit.

Delete Drafts → Remove the drafts.

Delete Own Changes → Remove code review which is push by yourself.

Edit Assignee → Assign a assignee to code review.

Edit Hashtags → This category permits users to add or remove hashtags on a change that is uploaded for review.

Gerrit management strategy (for project owner & administrator)

List of permission

Edit Topic Name → Edit the topic of commit.

Forge Author Identity
Forge Committer Identity → These permission is used when applying patches from 3rd parties.
Forge Server Identity

Owner → Let user control wholr project permission.

Publish Drafts → Allow user publish their draft.

Rebase → Give user permission to rebase the branch.

Remove Reviewer → Remove reviewer who has been added.

Submit → Give user permission to merge into authoritative region.

View Drafts → User with this permission could view other member's draft.

Gerrit management strategy (for project owner & administrator)

Project permission

Normal User

For normal team members, we give them basic permission under the project.

Read → code review, clone.

Abandon → abandon code review request.

Add patch set → modify the code which has to be revised.

Gerrit management strategy (for project owner & administrator)

Project Owner

Project owner has every authority in the project. Major thing project owner has to do is list below.

Create tag → abandon code review request.

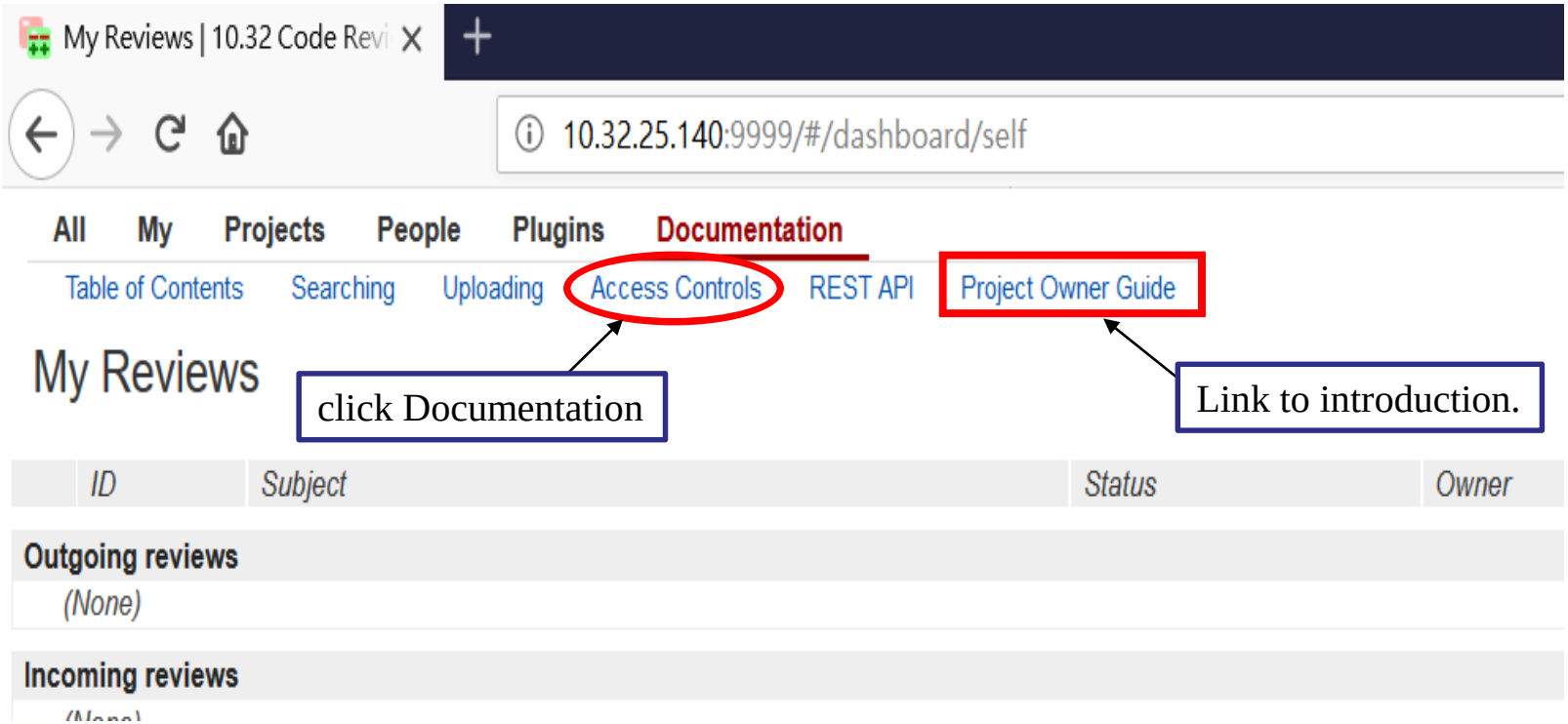
Submit → When commit has be approved, project owner should merge it into authoritative region.

Proxy → When project owner leave for any reason, you should give your permission to some of your team member temporarily.

Gerrit management strategy (for project owner & administrator)

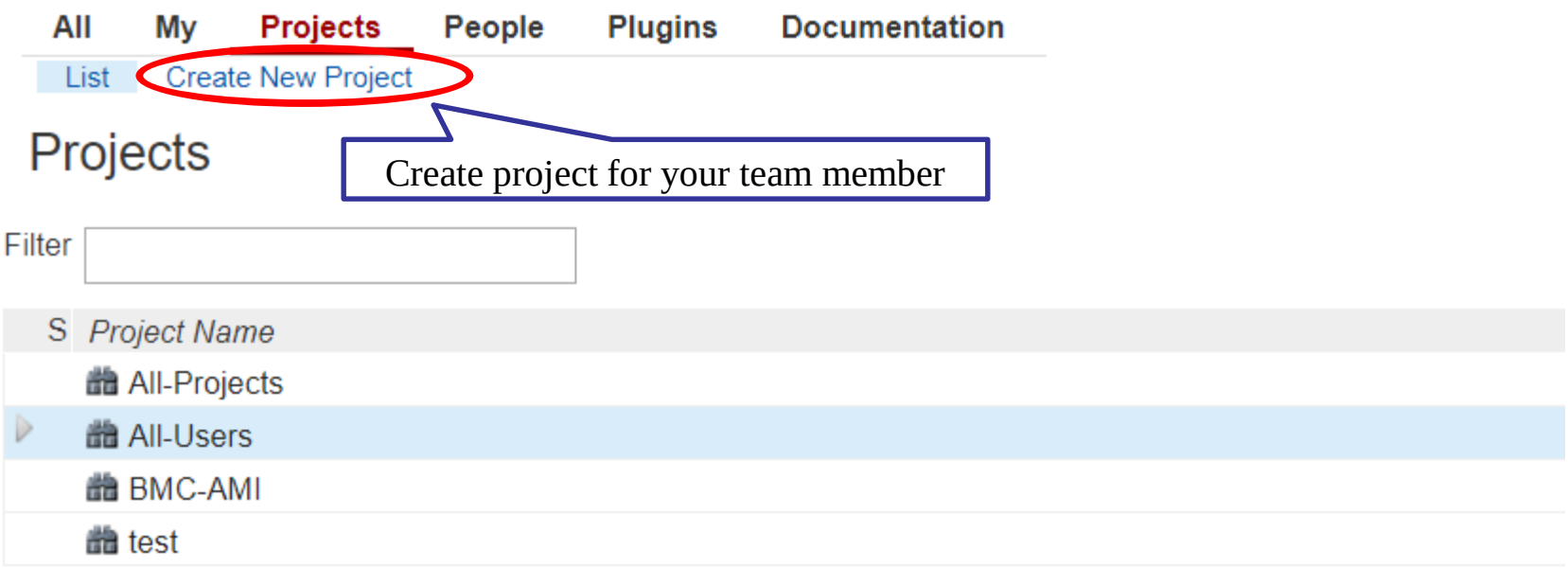
Project Owner

If you have any question about project owner, Gerrit web interface have full introduction about project owner.



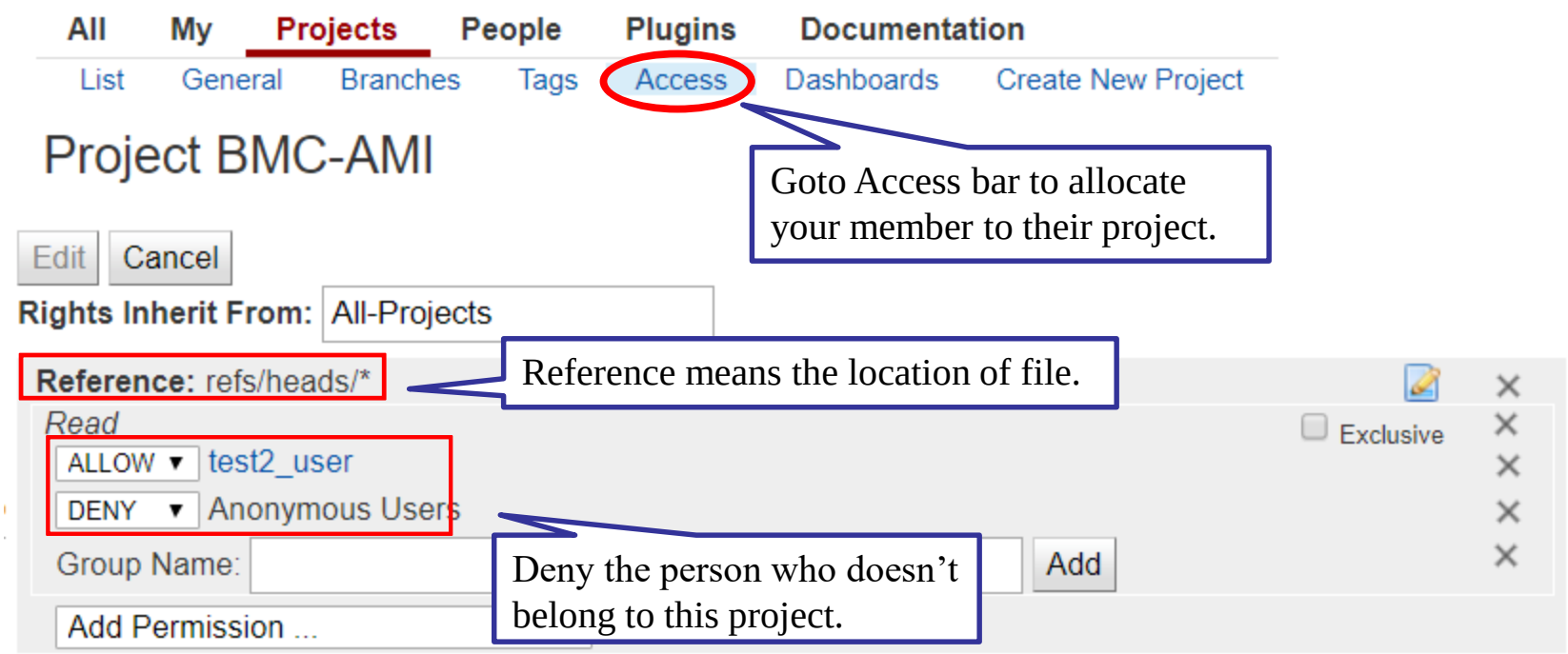
Gerrit management strategy (for project owner & administrator)

3.Project management



Gerrit management strategy (for project owner & administrator)

3.Project management



Gerrit management strategy (for project owner & administrator)

3.Project management

When every thing is setup, please go to desktop or any position you want your project locate and clone the project from Gerrit by following command.

```
git clone ssh://User ID@gerrit ip:29418/teset.git
```

```
andy_jr@openbmc-server:~$ git clone ssh://admin@10.32.25.140:29418/test.git
Cloning into 'test'...
remote: Counting objects: 12, done
remote: Finding sources: 100% (12/12)
remote: Total 14 (delta 0), reused 14 (delta 0)
Receiving objects: 100% (14/14), done.
Checking connectivity... done.
andy_jr@openbmc-server:~$ ls
5th_Oct_2018   Documents          GitEye              new_dev        repo
AMI            Downloads          GitEye.ini          p2             source
artifacts.xml  email.txt          gitkraken-amd64.deb Pictures       Templates
BMC-AMI.git    EOF                icon.xpm            plugins        test
configuration  examples.desktop   META-INF            Polaris_Plus   test3.pub
deploy-folder  features           Music               prj1           Videos
Desktop        gerrit_testsite    M71TB1(Wankang)     Public
```

Project you download

Gerrit Code Review (for every member)

How to request a code review ?

Gerrit is an web application based on git. So when we want to request for a code review, we have to push to HEAD:refs/for/your-branch-name (e.g. refs/for/master) by Git push ssh://User_ID@gerrit.com:29418/project HEAD:refs/for/master. Then Gerrit will inform approver to check it.

Gerrit Code Review (for every member)

Setup for code review.

Before getting start, we need to do following things first.

Step 1: Clone your project repository

```
git clone ssh://UserID@gerrit.com:29418/project.git
```

Step 2: Configure user.name and user.email

```
git config --global user.name andy
git config --global user.email Andy_Cheng@wistron.com
```

Step 3. Configure template of commit message.

```
git config --global commit.template ~/.gitmessage.txt
```

Gerrit Code Review (for every member)

Setup for code review.

Step 4: Configure template of commit message.

Make sure you create a .gitmessage.txt like this.

```
Subject line (try to keep under 50 characters)

Multi-line description of commit,
feel free to be detailed.

[Ticket: X]
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# modified:   lib/test.rb
#
```

Gerrit Code Review (for every member)

Setup for code review.

Step 5: Copy hook file from git repository

```
git rev-parse --git-dir
scp –p –P 29418 andy@gerrit.com:hooks/commit-msg .git/hooks
```

Every commit need an unique change-id which could automatically generate by hook file.

Gerrit Code Review (for every member)

Procedure of code review

Step 1: git add -A

Step 2: git commit

Step 3: git push ssh://User_ID@gerrit.com:29418/project HEAD:refs/for/master

Step 4: Then reviewer open up Gerrit for code review.

Gerrit Code Review (for every member)

Procedure of code review

1.Select button My

2.Go to Watched Changes

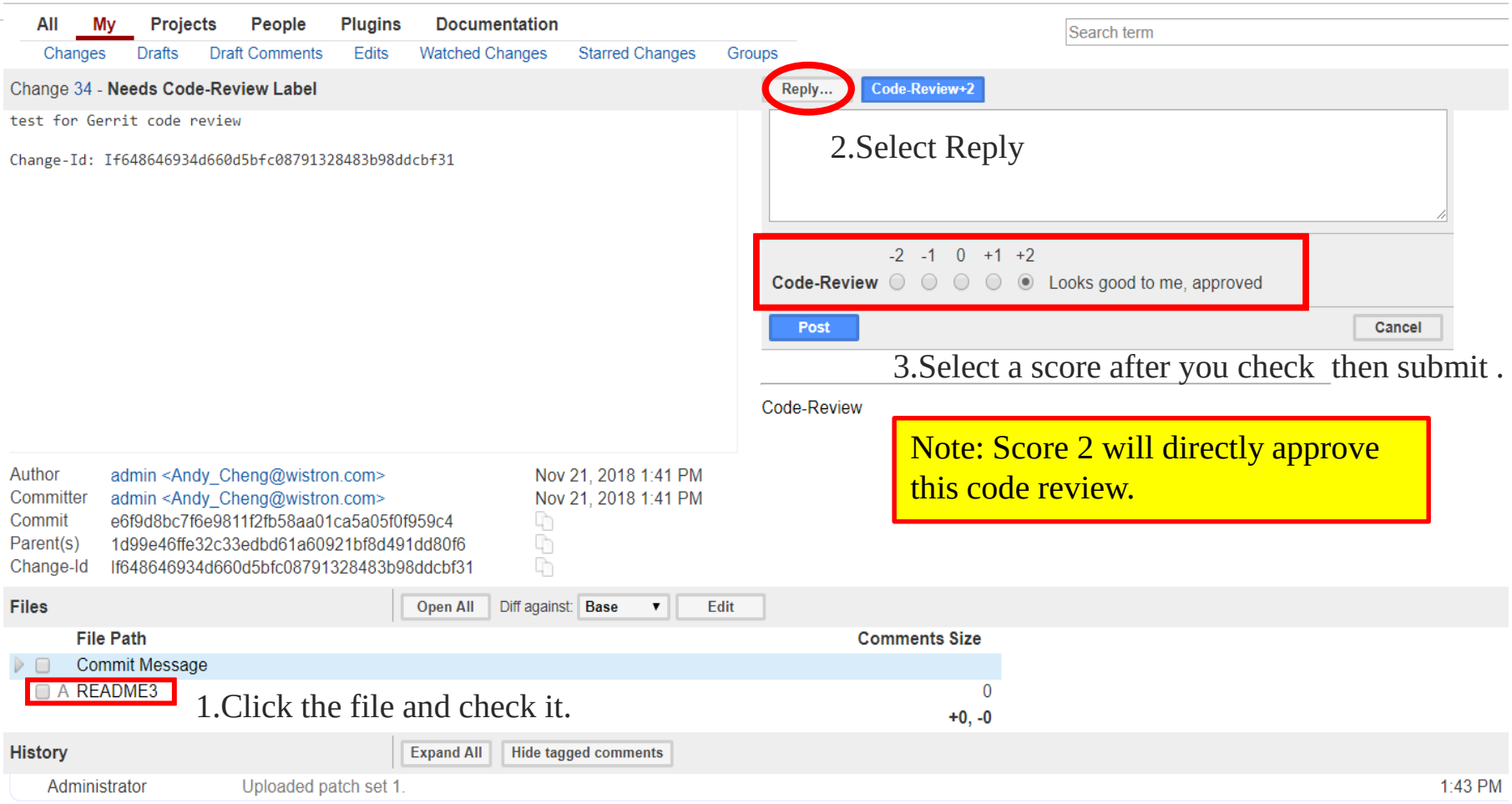
Search for is:watched is:open

	ID	Subject	Status	Owner
	34	test for Gerrit code review		Administrator

3.Then you see a request for code review

Gerrit Code Review (for every member)

Procedure of code review



Gerrit Code Review

What if we were rejected?

If our commit reject by approver, just do following step after you correct your mistake in local site.

Step 1. after you fix your error or anything else, just add to stage area again.
- git add -A

Step 2. commit to repository by postfix " –amend " , it will change your latest commit in your repo.
- git commit –amend

Step 3. Then push to Gerrit again.
- git push ssh://USER_ID@Gerrit.com:29418/PROJECT.git HEAD:refs/for/BRANCH.

Gerrit Code Review

What should approver do to reject commit ?

All My Projects People Plugins Documentation

Search term

Changes Drafts Draft Comments Edits Watched Changes Starred Changes Groups

Change 108 - **Needs Code-Review Label**

Reply... Code-Review+2

```
[Modify] add one line

Change-Id: I27e782deb51544092cc6ab48f8c26cbf4610c557
```

Owner Administrator
Assignee
Reviewers
Project Andy_test
Branch master
Topic
Strategy Merge if Necessary
Uploaded 26 seconds ago

Cherry Pick | Rebase | Abandon | Delete Change | Follow-Up

Code-Review

Select Delete change button to canceal this commit.

Author	admin <Andy_Cheng@wistron.com>	Mar 25, 2019 9:41 AM
Committer	admin <Andy_Cheng@wistron.com>	Mar 25, 2019 9:41 AM
Commit	543715ec94df76fcb2ca23cb87dea85dcf498d21	
Parent(s)	0b92774b1f9935943f52fe8558fdd5f17f3a7ef1	
Change-Id	I27e782deb51544092cc6ab48f8c26cbf4610c557	

Files | Open All | Diff against: Base | Edit

File Path	Comments Size
Commit Message	
README	2
	+2, -0

History | Expand All | Hide tagged comments

Administrator Uploaded patch set 1. 9:42

Git UI

- If you want to check history and every file on your git repository. Here are some tools about showing content by UI.

1.Using git command to generate httpd that listen port :1234

Step 1: git instaweb --httpd webrick

Step 2: Then open your browser and go to localhost:1234, it will show you all content of .git.

Git UI

Result after execution.

Applications Places

127.0.0.1 Git - .git/summary - Mozilla Firefox

127.0.0.1 Git - .git/sum

127.0.0.1:1234/?p=.git;a=summary

description Unnamed repository; edit this file 'description' to name the repository.
last change Thu, 8 Nov 2018 16:50:41 +0800 (16:50 +0800)

shortlog

2018-11-08	admin	**test** master | origin/HEAD | origin/master | v1.0 | v2.0 | v2.1	commit | commitdiff | tree | snapshot
2018-11-08	admin	**test for new branch** origin/ref/for/dev | origin/ref/for/master	commit | commitdiff | tree | snapshot
2018-11-08	admin	**test**	commit | commitdiff | tree | snapshot
2018-10-16	Administrator	**Initial empty repository**	commit | commitdiff | tree | snapshot

tags

2 weeks ago	**v2.1**	adding tag v2.1	tag | commit | shortlog | log
2 weeks ago	**v2.0**	adding tag v1.0	tag | commit | shortlog | log
2 weeks ago	**v1.0**	adding tag v1.0	tag | commit | shortlog | log

heads

3 weeks ago **master** shortlog | log | tree

remotes

origin

URL ssh://admin@10.32.25.140:29418/test.git

3 weeks ago	**dev**	shortlog | log | tree
3 weeks ago	**master**	shortlog | log | tree
3 weeks ago	**HEAD**	shortlog | log | tree
3 weeks ago	**ref/for/master**	shortlog | log | tree
3 weeks ago	**ref/for/dev**	shortlog | log | tree

Unnamed repository; edit this file 'description' to name the repository. Atom RSS

[andy_jr@open... andy_jr@openb... [12.0.1workspa... [andy_jr@open... 127.0.0.1 Git - .gi...

Git UI

Note: If user want to view other git repository, please stop the deamon previously by following command.

Step 1: git instaweb --httpd webrick --stop

Step 2: go to project which you attempt to view.

Step 3: active the deamon by git instaweb --httpd webrick.

Git UI

2. Install gitg on your linux system.

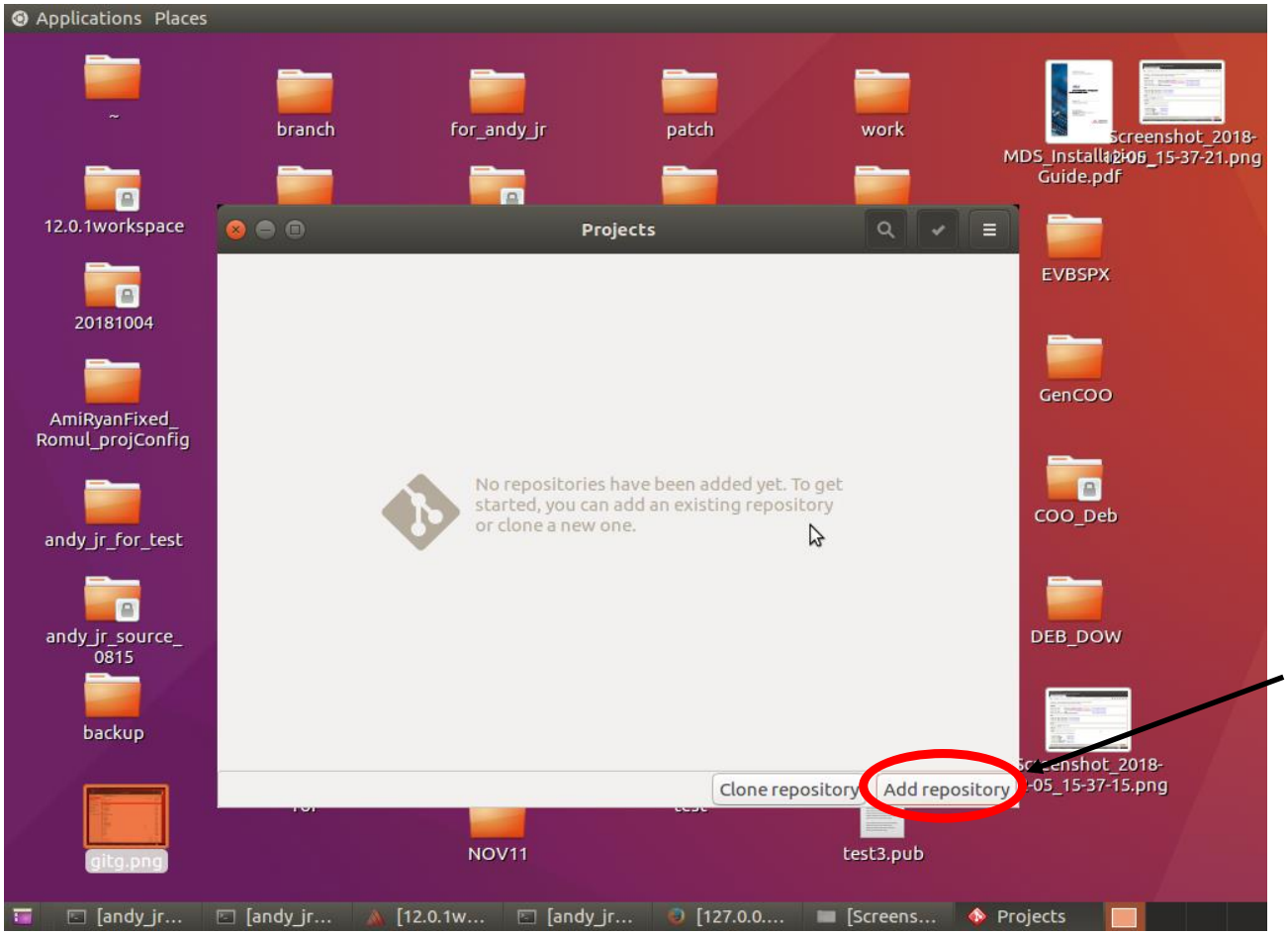
Step 1: download gitg from link.

Gitg: https://wiki.gnome.org/Apps/Gitg/

Step 2: Open gitg and add your .git repo.

Git UI

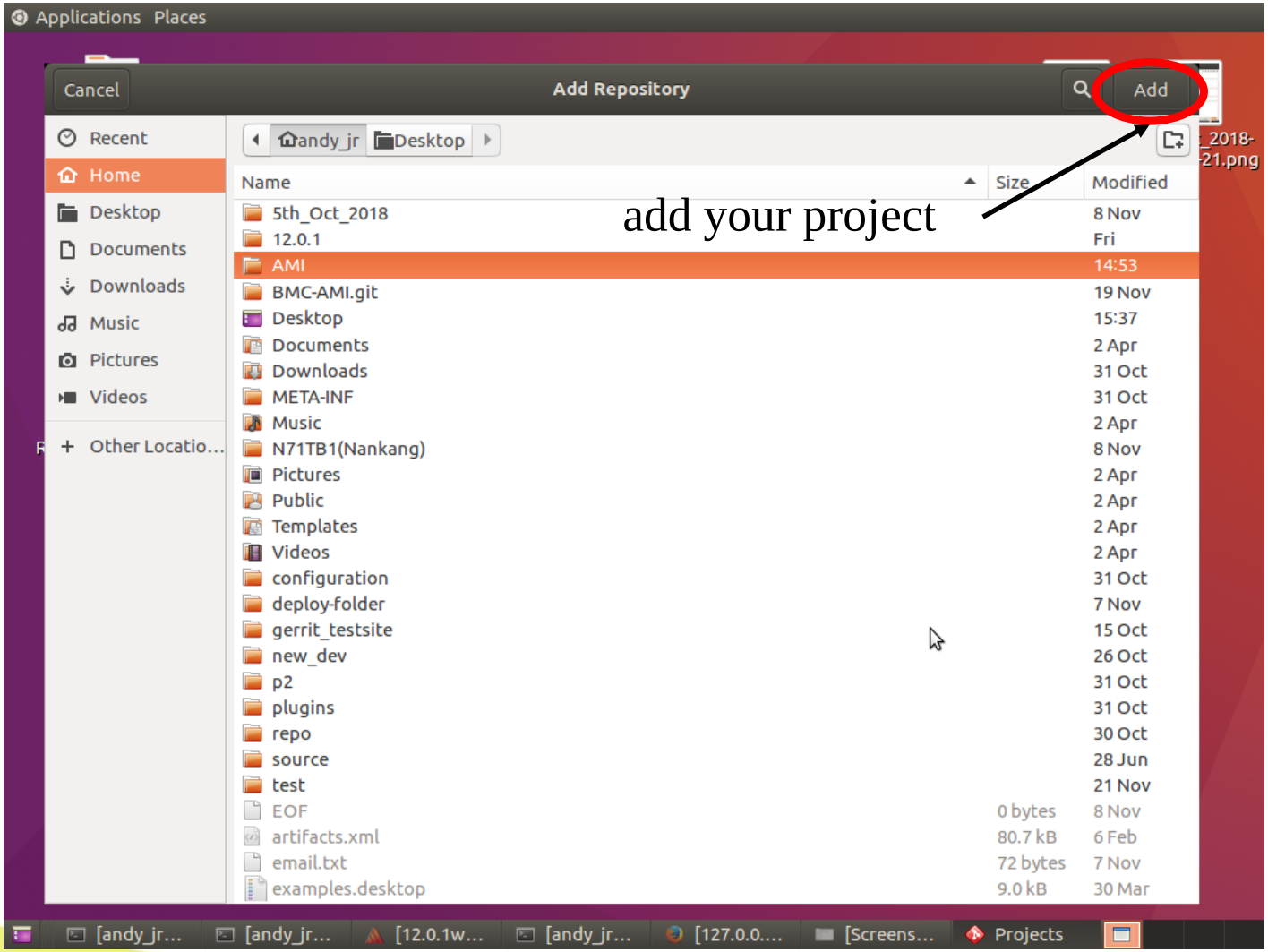
Start your gitg application.



Click add repository.

Git UI

Select your git repository and click Add button.



Git Commit Message

In order to let your commit clearly understand by every member, please follow the instruction below to formulate your commit message.

[Subject]:

with proper prefixed word at begin such as.

Feat: when you create new feature.

Fix: fix the bug.

Doc: upload some documents.

Style: reorganize your code.

Test: make test unit or benchmark for your code.

[Body]:

describe what and why you do this change in detail. Please make sure body is less than 72 characters.

Git Commit Message

Sample:

commit 42e769bdf4894310333942ffc5a15151222a87be
Author: andy<Andy_Cheng@wistron.com>
Date: Fri Jan 01 00:00:00 1982 -02000

[Feat] Create fan control feature for Mihawk.

We create our oem fan control algorithm to make sure our motherboard won’t over heat.

Git Backup Solution

How do we have a backup for git repository?

Step1. we clone the project on Gerrit server.

For example:

```
git clone –mirror ssh://admin@gerrit.com:29418/project.git
```

Step2. Then we use linux deamon called crontab to update the project regularly.

First, type the command as follow.

```
1.crontab -e
```

And add this line at the end

```
2. */5 * * * * cd ~/project.git; git remote update
```

This's mean that project will update in every 5minute

Git Backup Solution

Procedure of backup

We backup whole data (include database and .git repo) twice a day, which is 12 a.m. and 6 p.m. to make sure our firmware could properly be saved.

Check point 1

12 a.m.

database

.git

rsync

Remote update

Backup server

Check point 2

6 p.m.

database

.git

rsync

Remote update

Backup server

Git Backup Solution

Also , you could check the mail at /var/mail/ to confirm whether we have a backup.

Here is the notification from server.

```
Date: Tue, 20 Nov 2018 15:51:01 +0800
Message-Id: <201811200751.uAK7p1JJ004402@openbmc-server>
From: root@openbmc-server (Cron Daemon)
To: andy_jr@openbmc-server
Subject: Cron <andy_jr@openbmc-server> cd /home/andy_jr/BMC-AMI.git; git remote
update
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/andy_jr>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=andy_jr>

Fetching origin
```

Standard Operating Procedure for Gerrit Recovery

Assume we have properly backup all data and start to restore the Gerrit server, let's recover it step by step.

Step 1. Make sure Gerrit service doesn't on work by command

" sudo $GERRIT_HOME/bin/gerrit.sh stop "

Step 2. Cover the folder (cache, db and git)which we backup previously to the server we rebuild.

```
File  Edit  Setup  Control  Window  Help
andy_jr@openbmc-server:~/backup_data$ ll
total 20
drwxrwxr-x  5 andy_jr andy_jr 4096 Jan  2 15:42 ./
drwxr-xr-x 61 andy_jr andy_jr 4096 Jan  2 15:33 ../
drwxrwxr-x  2 andy_jr andy_jr 4096 Jan  2 15:33 cache/
drwxrwxr-x  2 andy_jr andy_jr 4096 Jan  2 15:33 db/
drwxrwxr-x  2 andy_jr andy_jr 4096 Jan  2 15:33 git/
andy_jr@openbmc-server:~/backup_data$ sudo cp -r {cache,db,git} $Gerrit_home
```

Note: $GERRIT_HOME is where your Gerrit directory is located e.g. /home/andy_jr/gerrit_site.

Standard Operating Procedure for Gerrit Recovery

Step 2.

cache and db directory store the user information.

git directory store the project data which is .git repository.

Step 3. Recreate the Lucene index for Gerrit by command

"java -jar $GERRIT_HOME/bin/gerrit.war reindex -d $GERRIT_HOME ".

If we don't take this step, Gerrit website won't show our review record we done previously.

```
andy_jr@openbmc-server:~/gerrit_testsite$ java -jar ~/gerrit_testsite/bin/gerrit
.war reindex -d ~/gerrit_testsite/
[2019-01-02 16:15:38,841] [main] INFO  com.google.gerrit.server.git.LocalDiskRep
ositoryManager : Defaulting core.streamFileThreshold to 2047m
[2019-01-02 16:15:39,795] [main] INFO  com.google.gerrit.server.cache.h2.H2Cache
Factory : Enabling disk cache /home/andy_jr/gerrit_testsite/cache
[2019-01-02 16:15:39,913] [main] INFO  com.google.gerrit.server.git.WorkQueue :
Adding metrics for 'WorkQueue' queue
[2019-01-02 16:15:39,919] [main] INFO  com.google.gerrit.server.git.WorkQueue :
Adding metrics for 'Index-Interactive' queue
[2019-01-02 16:15:39,923] [main] INFO  com.google.gerrit.server.git.WorkQueue :
Adding metrics for 'Index-Batch' queue
[2019-01-02 16:15:39,962] [main] INFO  com.google.gerrit.server.git.WorkQueue :
Adding metrics for 'ReceiveCommits' queue
[2019-01-02 16:15:39,962] [main] INFO  com.google.gerrit.server.git.WorkQueue :
Adding metrics for 'SendEmail' queue
Reindexing accounts:    100% (11/11)
Reindexed 11 documents in accounts index in 0.3s (43.3/s)
Reindexing groups:      100% (5/5)
Reindexed 5 documents in groups index in 0.1s (84.7/s)
Collecting projects:    6
Reindexing changes: projects: 100% (6/6), 1% (24/2203), done
Reindexed 24 documents in changes index in 2.1s (11.6/s)
```

Standard Operating Procedure for Gerrit Recovery

Step 4. Configure the Gerrit service. (httpd port and service ip)

We have to set gerrt.config file as same as we used before. Config file is located at $Gerrit_home/etc/ directory.

```
[gerrit]
        basePath = git
        serverId = 298ffa04-ef14-48bd-a8fb-988020d9f2e7
        canonicalWebUrl = http://openbmc-server:8080/
[database]
        type = h2
        database = /home/andy_jr/gerrit_testsite/db/ReviewDB
[index]
        type = LUCENE
[auth]
        type = DEVELOPMENT_BECOME_ANY_ACCOUNT
[receive]
        enableSignedPush = false
[sendemail]
        smtpServer = localhost
[container]
        user = andy_jr
        javaHome = /usr/lib/jvm/jdk1.8.0_171/jre
[sshd]
        listenAddress = *:29418
[httpd]
        listenUrl = http://10.32.25.115:8080
[cache]
        directory = cache
[plugins]
        allowRemoteAdmin = true
```

→ authentication we used is HTTP.

→ Your java environment.

→ Your current ip address and port that used for Gerrit service.

Standard Operating Procedure for Gerrit Recovery

Step 5. Eventually, Restart the Gerrit service by command

"sudo $Gerrit_home/bin/gerrit.sh restart"

And open browser to check whether every data been restored correctly, include review record, project and user information.

review record

All My Projects People Plugins Documentation

Open Merged Abandoned

status:merged

Search for status:merged

ID	Subject	Status	Owner	Project
86	[FIX] Fix the LICENSE	Merged	Administrator	openbmc
85	add source code	Merged	bob	Openbmc
84	remove idbutton from dts	Merged	bob	Openbmc
83	add dts	Merged	bob	Openbmc
82	first commit	Merged	bob	Openbmc
81	.git folder test	Merged	Administrator	test
80	test	Merged	Administrator	Openbmc
64	[ADD] add new line	Merged	Administrator	AMI
63	ADD new line.	Merged	Administrator	AMI
62	test	Merged	Administrator	AMI
61	Edit Project Config	Merged	Administrator	AMI
59	revision part 2	Merged	Administrator	AMI
58	revise the same part	Merged	Administrator	AMI
57	second commit	Merged	Administrator	AMI
56	initial commit	Merged	Administrator	AMI
47	test2	Merged	bob	test
46	testing	Merged	bob	test
22	test	Merged	Administrator	test
21	test	Merged	Administrator	test
19	test	Merged	Administrator	test
1	Edit Project Config	Merged	Administrator	test

Standard Operating Procedure for Gerrit Recovery

Project

All My Projects People Plugins Documentation

Search term

List Create New Project

Projects

Filter

S	Project Name	Project Description
	AMI	
	All-Projects	Access inherited by all other projects.
	All-Users	Individual user settings and preferences.
	Openbmc	
	openbmc	OpenBMC Distribution
	test	test for gerrit

Note: Clone the project and check git log isn't exist.

Standard Operating Procedure for Gerrit Recovery

User information

Sign In

Username: [ ] Become Account

Email Address: [ ] Become Account

Account ID: [ ] Become Account

Choose:

Administrator
1000001
test3
test4
user3
andy
1000013
Andy_Cheng@example.com
Ben_Pai
Bob_King@wistron.com
Timothy_Huang@wistron.com

Note: Log in and check ssh-key and email.

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 1. Retrieve a list of all Subversion committers

```
$ svn log -q https://svn.example.com/repository_name | \
awk -F '|' '/^r/ {sub("^ ", "", $2); sub(" $", "", $2); \
print $2" = "$2" <"$2">"}' | sort -u > authors-transform.txt
```

That will grab all the log messages, pluck out the usernames, eliminate any duplicate usernames, sort the usernames and place them into a "authors-transform.txt" file. Now edit each line in the file. For example, convert:

Andy_Cheng = Andy_Cheng@wistron.com

Note: repository_name is your folder name in svn (e.g. Niki)

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 2. Clone the Subversion repository using git-svn

```
git svn clone [SVN repo URL] --no-metadata -A authors-transform.txt --stdlayout ~/temp
```

Step 3. Convert svn:ignore properties to .gitignore

If your svn repo was using svn:ignore properties, you can easily convert this to a .gitignore file using:

```
cd ~/temp
git svn show-ignore > .gitignore
git add .gitignore
git commit -m 'Convert svn:ignore properties to .gitignore.'
```

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 4. Push repository to a bare git repository

First, create a bare repository and make its default branch match svn's "trunk" branch name.

```
git init --bare ~/new-bare.git
cd ~/new-bare.git
git symbolic-ref HEAD refs/heads/trunk

cd ~/temp
git remote add bare ~/new-bare.git
git config remote.bare.push 'refs/remotes/*:refs/heads/*'
git push bare
```

You can now safely delete the ~/temp repository.

Note: In above example, we create a directory called new-bare.git and initiate it by command "git init –bare".

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 5. Rename "trunk" branch to "master"

Your main development branch will be named "trunk" which matches the name it was in Subversion. You'll want to rename it to Git's standard "master" branch using:

```
cd ~/new-bare.git
git branch -m trunk master
```

Note: Traditionally, Master is major branch for .git repo.

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 6. Clean up branches and tags

git-svn makes all of Subversions tags into very-short branches in Git of the form "tags/name”. You’ll want to convert all those branches into actual Git tags using:

```
cd ~/new-bare.git
git for-each-ref --format='%(refname)' refs/heads/tags |
cut -d / -f 4 |
while read ref
do
  git tag "$ref" "refs/heads/tags/$ref";
  git branch -D "tags/$ref";
done
```

Note: $refs is tag named on svn

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 7. Move bare repository to central remote repository

Example of how to move your local bare repository to a gitolite repository:

```
mv new-bare.git repository_name.git
tar czvf repository_name.git.tar.gz repository_name.git/
scp repository_name.git.tar.gz remote_host:
ssh remote_host
tar xzvf repository_name.git.tar.gz
sudo chown -R git:staff repository_name.git/
cd repository_name.git/
find . -type f -exec chmod go= {} \;  # remove group and world permissions
find . -type d -exec chmod go= {} \;  # remove group and world permissions
cd ../
mv repository_name.git /Users/git/repositories/
```

Covert to svn from git

The following step is tutorial about transforming svn into git working space.

Step 8. Clone new local copy

```
mv old-svn-copy old-svn-copy.backup
git clone git@remote_host:repository_name.git
```

List all unversioned files from your old local svn repository and copy them to the new local git repository:

```
cd old-svn-copy.backup
git clean -dXn  # Using this command because the old copy was a git-svn clone
cp example-file.txt ../repository_name/  # copy all files and directories from the list that you need in the new loc
```

Step 9. Done

Reference

- **Gerrit Code Review**
  - https://gerrit-review.googlesource.com/Documentation/index.html#_developer
- **Introduction for git**
  - https://git-scm.com/about
- **Gerrit/trouble shooting**
  - https://www.mediawiki.org/wiki/Gerrit/Troubleshooting
- **How to write a git commit message**
  - https://chris.beams.io/posts/git-commit/

Thanks for your participation