


Working with Gerrit

Lesson 02

April 22, 2016

Proprietary and Confidential

- 1 -

  
CONSULTING. TECHNOLOGY. OUTSOURCING

## Lesson Objectives

- Understanding Code review
- Performing Code review using Gerrit



## Understanding Code Review

### ➤ Code Review

- Part of the development process
- Prerequisite for merging the code into target branch
- It is seen as *team programming*, where code is shared all members of the development team

### ➤ Roles in Code Review

- **Contributor** : User with the ability to access at least one branch of the repository in read mode and with the ability to upload a new change
- **Reviewer** : Team member who can express a score on a given change, expressed as a label + numeric value (positive, negative, or neutral)
- **Committer** : Team member who can thorough feedback on the code and provide the final word (either positive or negative) on a proposed change
- **Maintainer** : *This role involves administering and monitoring the project, but who then has no active role in the Code Review process.*

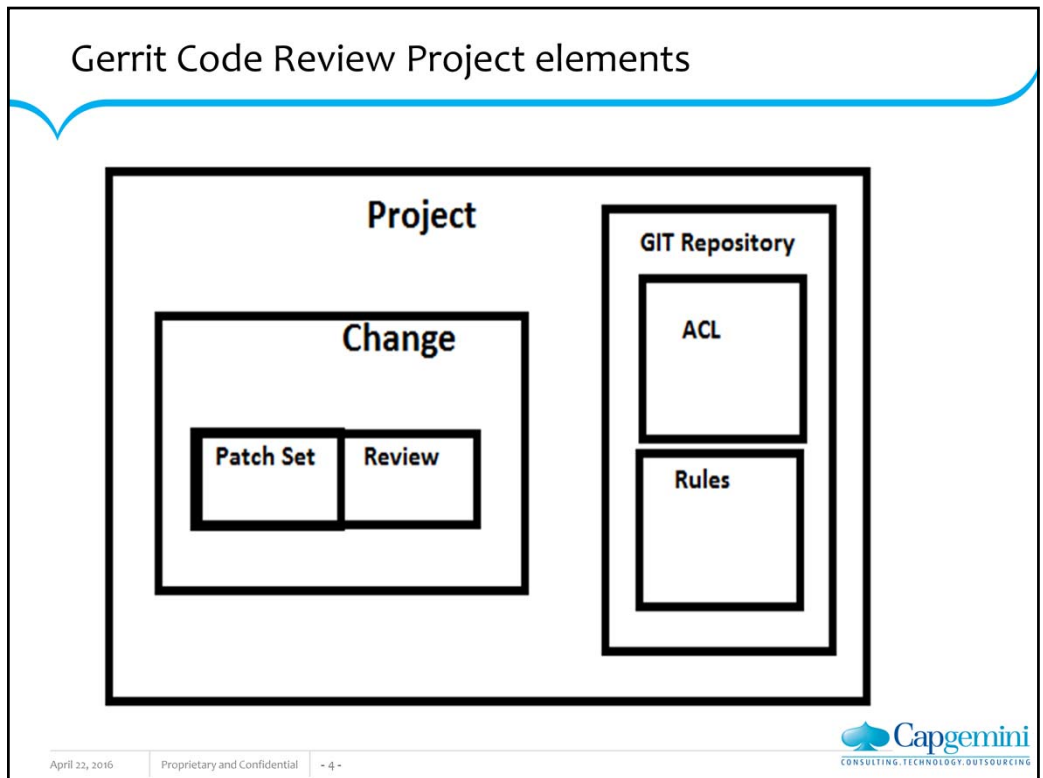
April 22, 2016

Proprietary and Confidential

- 3 -



Reviewer Team members and contributors are typically granted the ability to express a positive (+1) or negative (-1) review score in addition to simple comments on the code or on the entire change.



### Performing Code Review : steps .....

- Initialize the commit-msg hooks
- The contributor creates a commit & pushes to the Gerrit review branch
- The contributor invites one or more reviewers to look at the code and provide their feedback
- Reviewers provide feedback, with comments, code change and positive or negative score, ranging from -2 to +2
- A change with score less than minimum score to be approved, it will need to be reworked by its contributor
- Types of reworking can be **rebase, amend**
- Committer submits a change once it satisfies the minimum conditions for approval, it automatically triggers the merge & Gerrit merges

April 22, 2016

Proprietary and Confidential

- 5 -



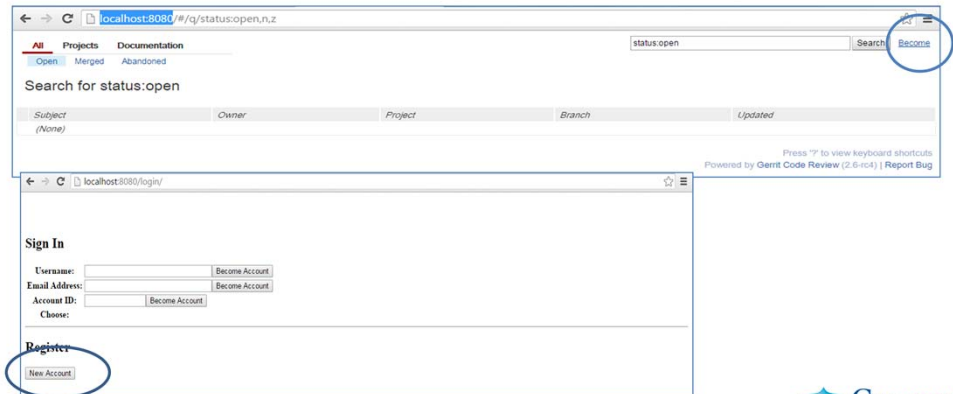
The contributor creates a commit and assigns a new unique global **Change-Id** through the commit-msg hooks previously downloaded from Gerrit. The contributor pushes to the Gerrit review branch, creating a new change for review. Gerrit assigns a unique URL in order to access and review the change using its web-UI. This step is optional. Continuous Integration builds and verifies the change: the builder fetches and triggers a build to check if the change actually works. Feedback is reported back as a positive (code builds and works) or negative (code doesn't work or build is broken) score. For existing changes with additional patch-sets, Gerrit automatically notifies all of the existing reviewers and watchers that a new review is required.

Gerrit has a set of rules for getting a change approved based on the overall feedback and scoring received. Default Gerrit rules require at least one Code Review/ + 2. Rules can be tailored to each project's needs and can even have a different set of approval conditions or be based on additional custom labels. When a change does not reach the minimum score to be approved, it will need to be reworked by its contributor.

Gerrit merges the change onto its target branch, as a consequence of the previous submit operation. Should the merge fail because of a conflict, the change has to be reworked again and validation has to resume.

## Creating a User profile for Gerrit UI

- Selecting single tag takes the following syntax
  - <http://localhost:8080>
- Select **'Become'** and then **'New Account'**



Add the notes here.

## Creating a User profile for Gerrit (continued)

### ➤ Gerrit 'New Account' registration

- Fill in the form as shown below

The screenshot shows the Gerrit 'New Account' registration page. The page has a navigation bar at the top with links: All, My, Projects, People, Plugins, Documentation. Below the navigation bar, there are tabs: Changes, Drafts, Draft Comments, Watched Changes, Starred Changes. The main heading is 'Welcome to Gerrit Code Review'. Below this, there is a section titled 'Please review your contact information:' with a paragraph explaining that the following contact information was automatically obtained when you signed-in to the site. The form contains the following fields: Full Name (Contributor1), Preferred Email (contributor1@mydomain.com), and a 'Register New Email...' button. There are three numbered annotations: '1' points to the 'Register New Email...' button, '2' points to the 'Save Changes' button, and '3' points to the 'Username' field. Below the contact information section, there is a section titled 'Register an SSH public key' with a paragraph explaining that Gerrit Code Review uses public-key cryptography and SSH to authenticate you during git's push and pull commands to hosted projects. Below this, there is a section titled 'Add SSH Public Key' with a link to 'How to Generate an SSH Key' and a text area for the public key. At the bottom, there is a 'Server Host Key' section with a fingerprint and a link to 'How to Generate an SSH Key'. The page footer contains the Capgemini logo and the text 'CONSULTING. TECHNOLOGY. OUTSOURCING'.

Add the notes here.

## Walkthrough

### ➤ Performing Gerrit Code Review



April 22, 2016 | Proprietary and Confidential | - 8 -

 **Capgemini**  
CONSULTING. TECHNOLOGY. OUTSOURCING

Add the notes here.



## References

### Code review

➤ <https://gerrit-review.googlesource.com/Documentation/dev-readme.html>

April 22, 2016

Proprietary and Confidential

~ 9 ~



Add the notes here.

## Summary

- Understanding Code review
- Performing Code review using Gerrit



April 22, 2016

Proprietary and Confidential

• 10 •

 **Capgemini**  
CONSULTING. TECHNOLOGY. OUTSOURCING

Add the notes here.