Gerrit Code Review

First steps with Git and Code Review Workflow

GerritForge

Luca Milanesio
luca@gerritforge.com

10 git commands commonly used in Gerrit Workflow

GerritForge

1. git clone
2. git fetch
3. git checkout
4. git add
5. git commit --amend
6. git reset
7. git rebase
8. git merge
9. git cherry-pick
10. git remote

Git refresh – Hands-on Lab#1

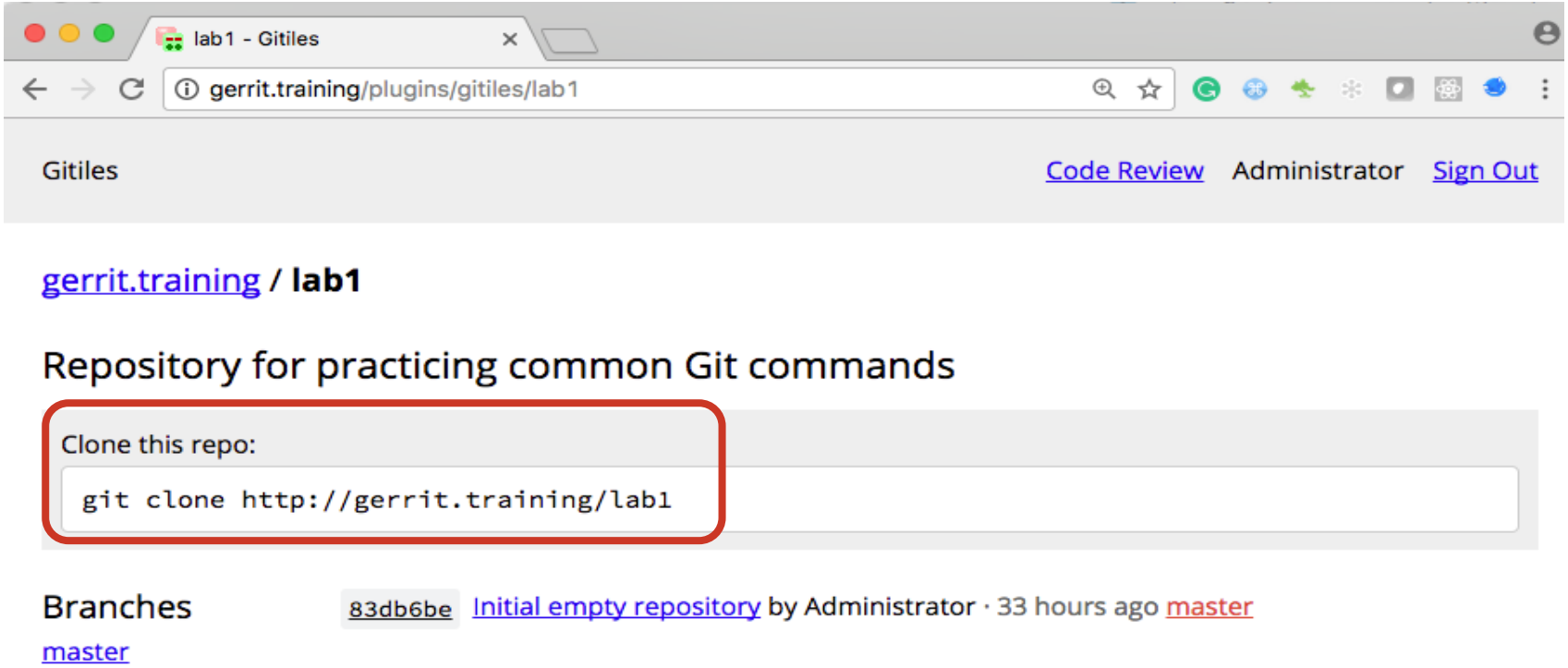
GerritForge

Practicing advanced Git commands

@gitenterprise @gerritreview

Lab#1: Gerrit Repositories Browser: Gitiles

GerritForge



Lab#1: Clone the repository from Gerrit

GerritForge

```
$ git clone http://gerrit.training/lab1
Cloning into 'lab1'...
remote: Counting objects: 2, done
remote: Finding sources: 100% (2/2)
remote: Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
Checking connectivity... done.

$ cd lab1
(master) $
```

Lab#1: Create a feature branch and add one commit

GerritForge

```
(master) $ git checkout -b feature/my-true-story
Switched to a new branch 'feature/my-true-story'

(feature/my-true-story) $ echo "This is me" > my-true-story.md

(feature/my-true-story) $ git add . && git commit -m "My true story"
[feature/my-true-story 0368861] My true story
 1 file changed, 1 insertion(+)
 create mode 100644 my-true-story.md
```

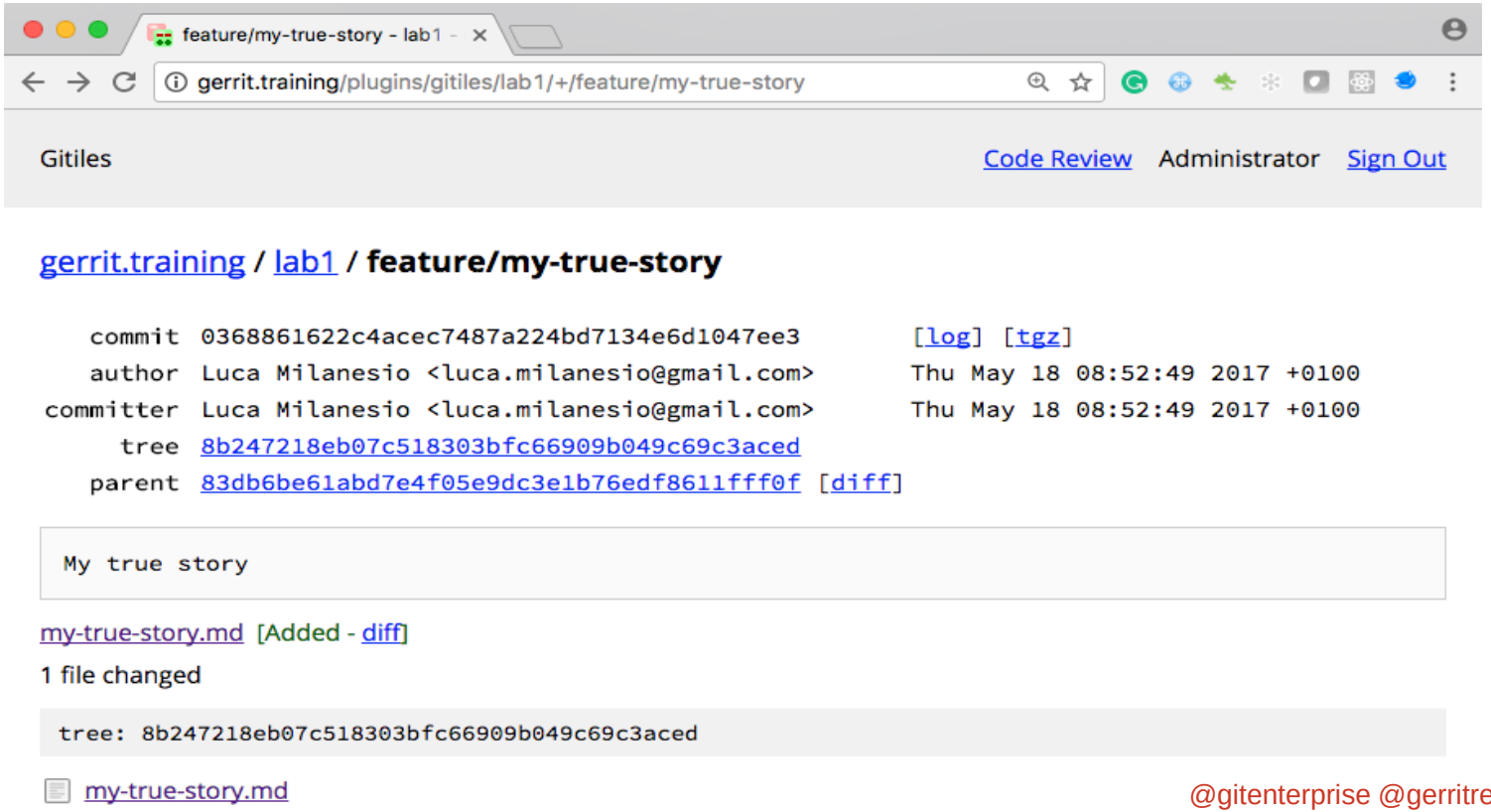
Lab#1: Push to Gerrit

GerritForge

```
(feature/my-true-story) $ git push --set-upstream origin feature/my-true-story
Username for 'http://gerrit.training': admin
Password for 'http://admin@gerrit.training': secret
Counting objects: 3, done.
Writing objects: 100% (3/3), 266 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Processing changes: refs: 1, done
To http://gerrit.training/lab1
 * [new branch]      feature/my-true-story -> feature/my-true-story
Branch feature/my-true-story set up to track remote branch feature/my-true-story from origin.
```

Lab#1: See feature branches in Gitiles

GerritForge



@gitenterprise @gerritreview

Lab#1: Another developer pushes a feature branch

GerritForge

```
(master) $ git checkout -b feature/another-story
Switched to a new branch 'feature/another-story'

(feature/another-story) $ echo "This is you" > another-story.md

(feature/another-story) $ git add . && git commit –m "Another story"
[feature/my-true-story 036bc6d] Another story
 1 file changed, 1 insertion(+)
 create mode 100644 another-story.md

(feature/another-story) $ git push origin feature/another-story
```

Lab#1: Working in "detached HEAD" mode

GerritForge

```
$ git fetch origin feature/another-story
From http://gerrit.training/lab1
 * branch            feature/another-story -> FETCH_HEAD

$ git checkout FETCH_HEAD
Note: checking out 'FETCH_HEAD'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 09385fc... Another story
```

@gitenterprise @gerritreview

Lab#1 – Amending a commit in "detached" HEAD

GerritForge

```
((09385fc...))$ echo "yet another word" >> another-story.md

((09385fc...))$ git commit -a --amend -m "Amended commit"
[detached HEAD bc4970b] Amended commit
 Date: Thu May 18 23:35:12 2017 +0100
 1 file changed, 1 insertion(+)
 create mode 100644 another-story.md

((bc4970b...))$ git push -f origin HEAD:feature/another-story
Counting objects: 3, done.
Writing objects: 100% (3/3), 269 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Processing changes: refs: 1, done
To http://gerrit.training/lab1
 + 09385fc...bc4970b HEAD -> feature/another-story (forced update)
```

Lab#1 – Fetching last HEAD of a branch

GerritForge

```
((09385fc...))$ git checkout feature/another-story
Previous HEAD position was c14001a... Amended commit
Switched to branch 'feature/another-story'
Your branch and 'origin/feature/another-story' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

(feature/another-story)$ git fetch origin
From http://gerrit.training/lab1
 * branch            feature/another-story -> feature/another-story (forced update)

(feature/another-story)$ git reset --hard origin/feature/another-story
HEAD is now at c14001a Amended commit
```

Lab#1 – Rebase a feature branch against master (1)

GerritForge

```
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

(master)$ echo foo > bar
(master)$ git add bar && git commit -m foo
[master 52c1df5] foo
 1 file changed, 1 insertion(+)
 create mode 100644 bar

(master)$ git push
Counting objects: 3, done.
Writing objects: 100% (3/3), 237 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Processing changes: refs: 1, done
To http://gerrit.training/lab1
   83db6be..52c1df5  master -> master
```

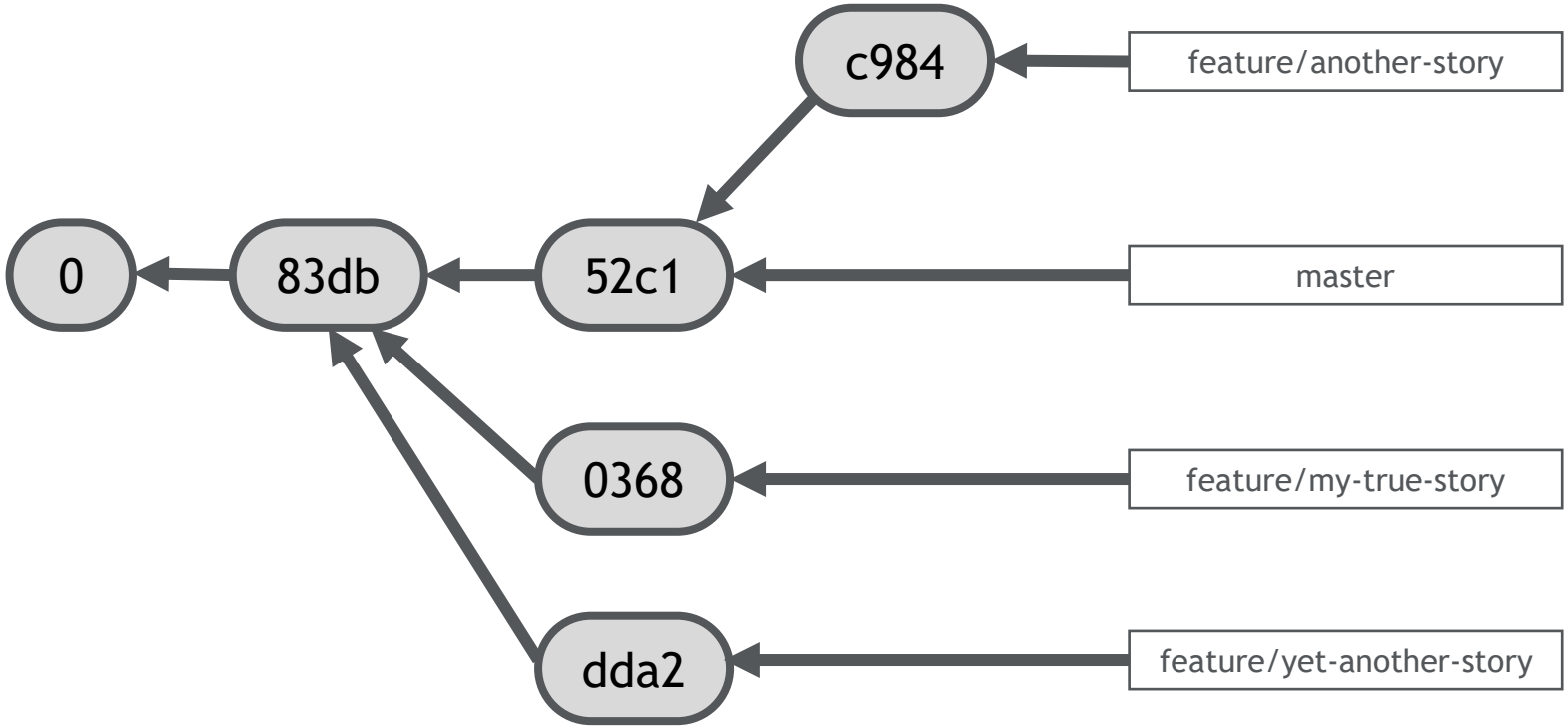
Lab#1 – Rebase a feature branch against master (2)

GerritForge

```
$ git checkout feature/another-story
Switched to branch 'feature/another-story'
Your branch is up-to-date with 'origin/feature/another-story'.

(feature/another-story)$ git fetch && git rebase origin/master
First, rewinding head to replay your work on top of it...
Applying: Amended commit

(feature/another-story)$ git push -f origin feature/another-story
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 316 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Processing changes: refs: 1, done
To http://gerrit.training/lab1
 + c14001a...c9841bf feature/another-story -> feature/another-story (forced update)
```

Lab#1 – Cherry-pick commits between branches (1)

GerritForge



Lab#1 – Cherry-pick commits between branches (2)

GerritForge

```
(feature/another-story)$ git cherry-pick 0368
[feature/another-story 31911f1] My true story
 Date: Thu May 18 08:52:49 2017 +0100
 1 file changed, 1 insertion(+)
 create mode 100644 my-true-story.md

(feature/another-story)$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 332 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Processing changes: refs: 1, done
To http://gerrit.training/lab1
   c9841bf..31911f1  feature/another-story -> feature/another-story
```

Lab#1 – Cherry-pick commits between branches (3)

GerritForge

Lab#1 – Working with remotes

GerritForge

```
$ git remote -v
origin	http://gerrit.training/lab1 (fetch)
origin	http://gerrit.training/lab1 (push)

$ git remote set-url --push origin ssh://admin@gerrit.training:29418/lab1
$ git remote -v
origin	http://gerrit.training/lab1 (fetch)
origin	ssh://admin@gerrit.training:29418/lab1 (push)

$ git remote add sshorigin ssh://admin@gerrit.training:29418/lab1
$ git remote -v
origin	http://gerrit.training/lab1 (fetch)
origin	ssh://admin@gerrit.training:29418/lab1 (push)
sshorigin	ssh://admin@gerrit.training:29418/lab1 (fetch)
sshorigin	ssh://admin@gerrit.training:29418/lab1 (push)
```

GerritForge

Gerrit Code Review Workflow

Code Review Workflow (0)

GerritForge

CI Build: OK

A1

origin/branch

Code Review Workflow (1)

GerritForge

CI Build: OK

A1

origin/branch

clone

A1

branch

@gitenterprise @gerritreview

Code Review Workflow (2)

GerritForge

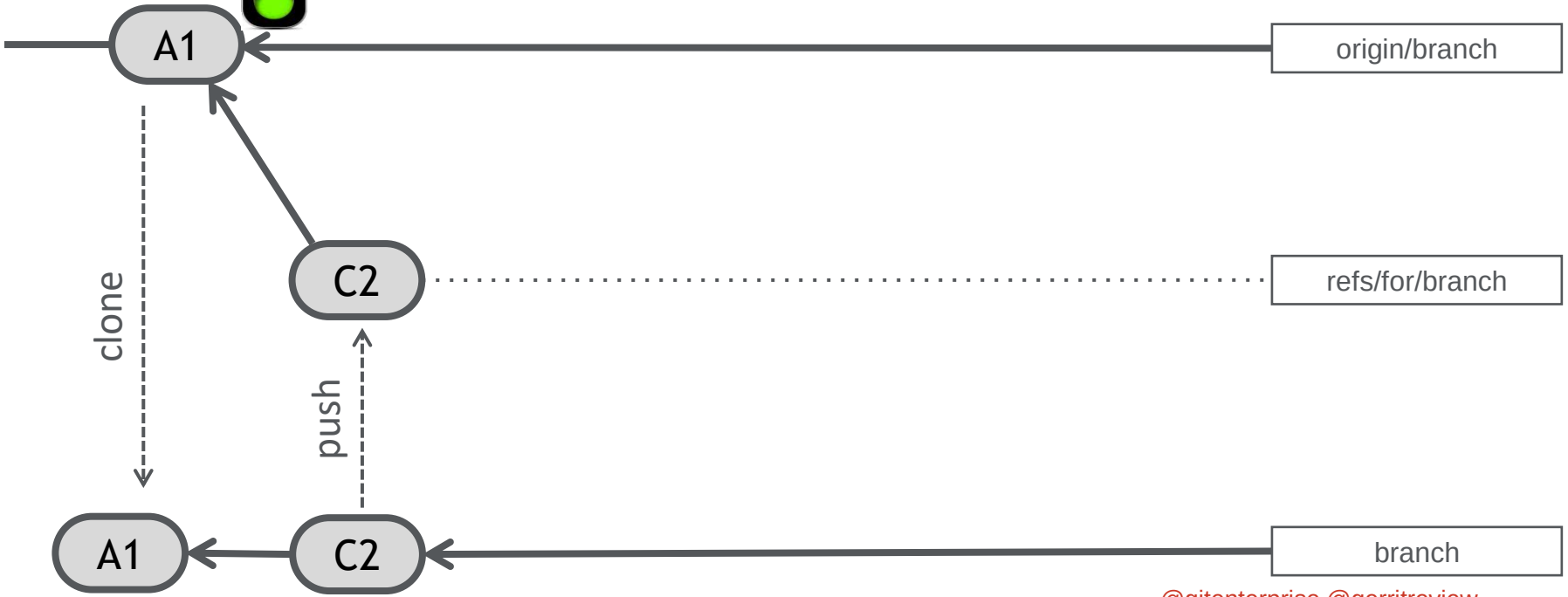
CI Build: OK

@gitenterprise @gerritreview

Code Review Workflow (3)

GerritForge

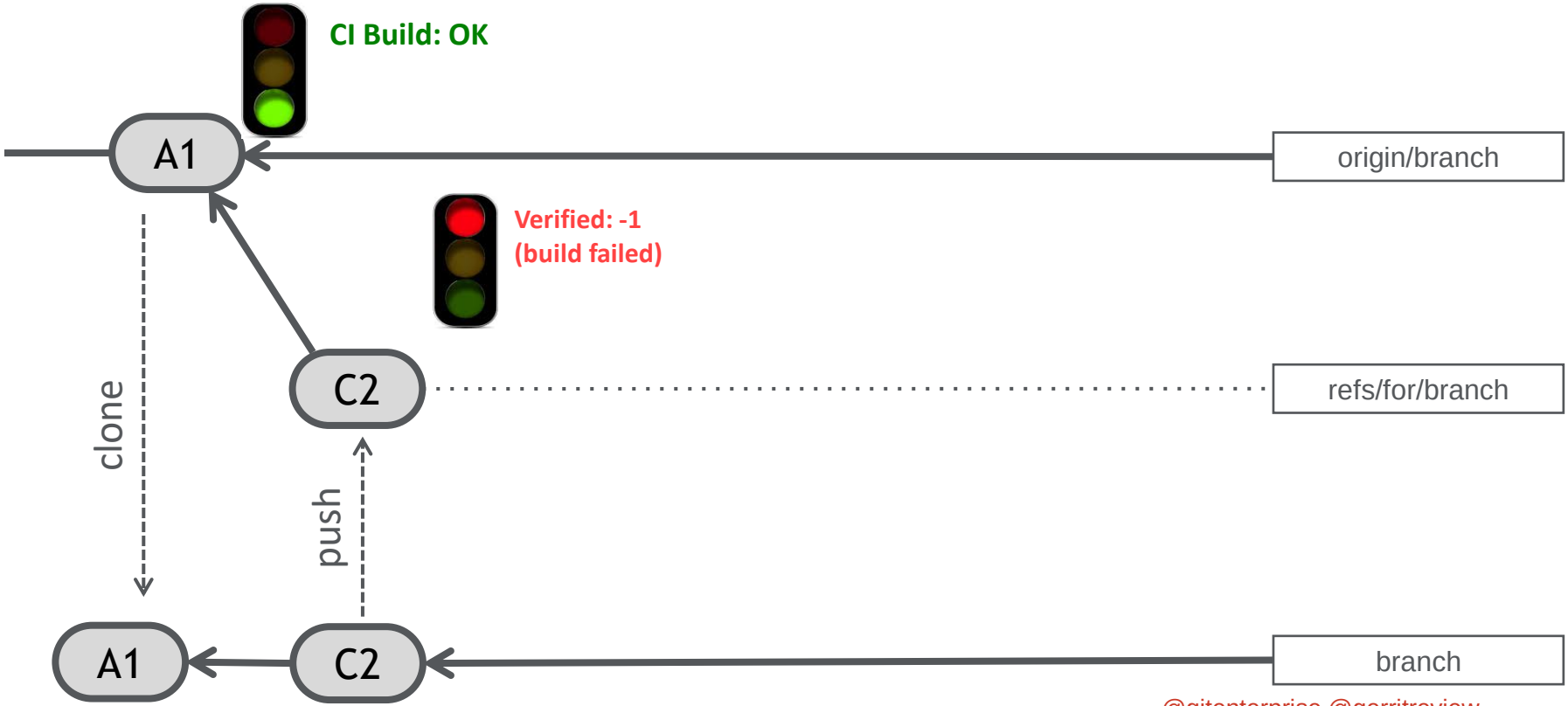
CI Build: OK



@gitenterprise @gerritreview

Code Review Workflow (4)

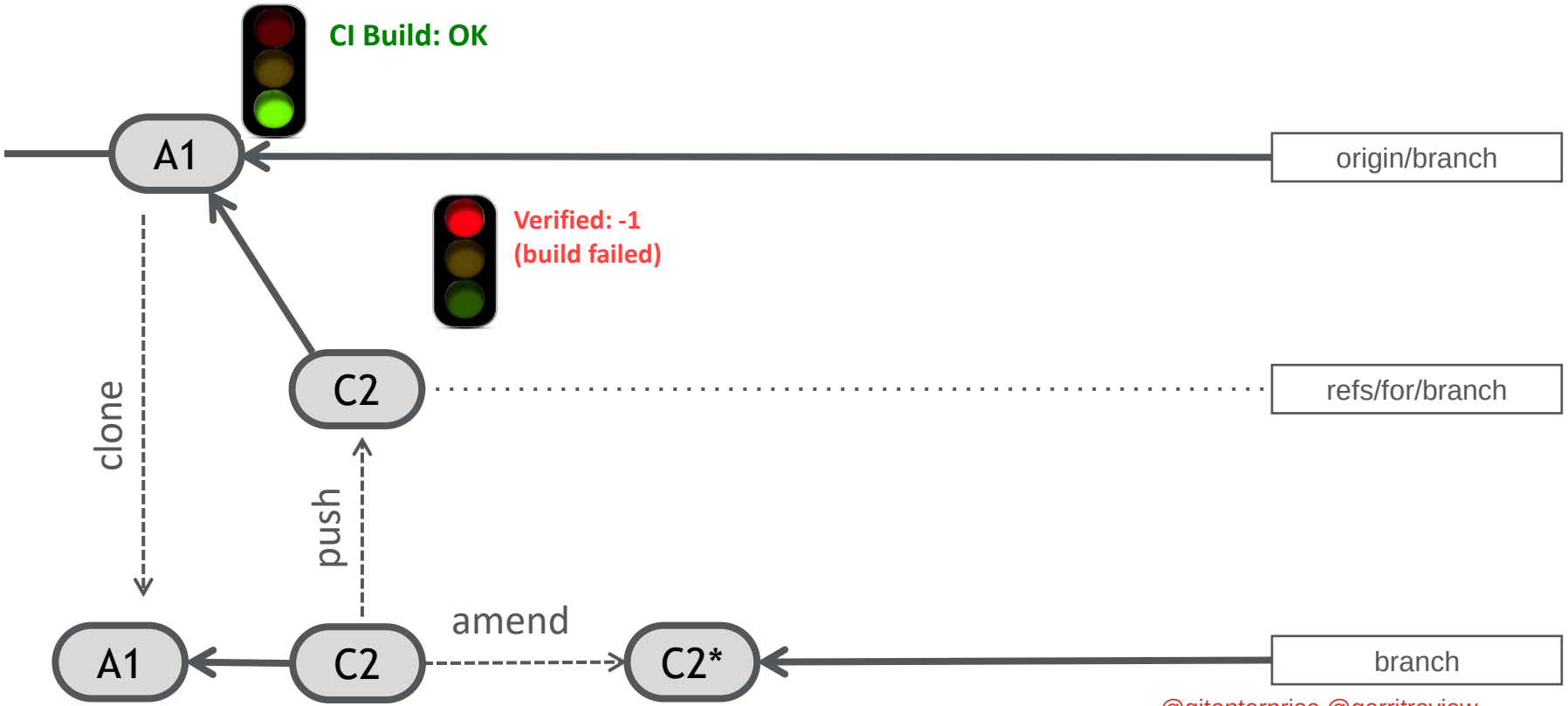
GerritForge



@gitenterprise @gerritreview

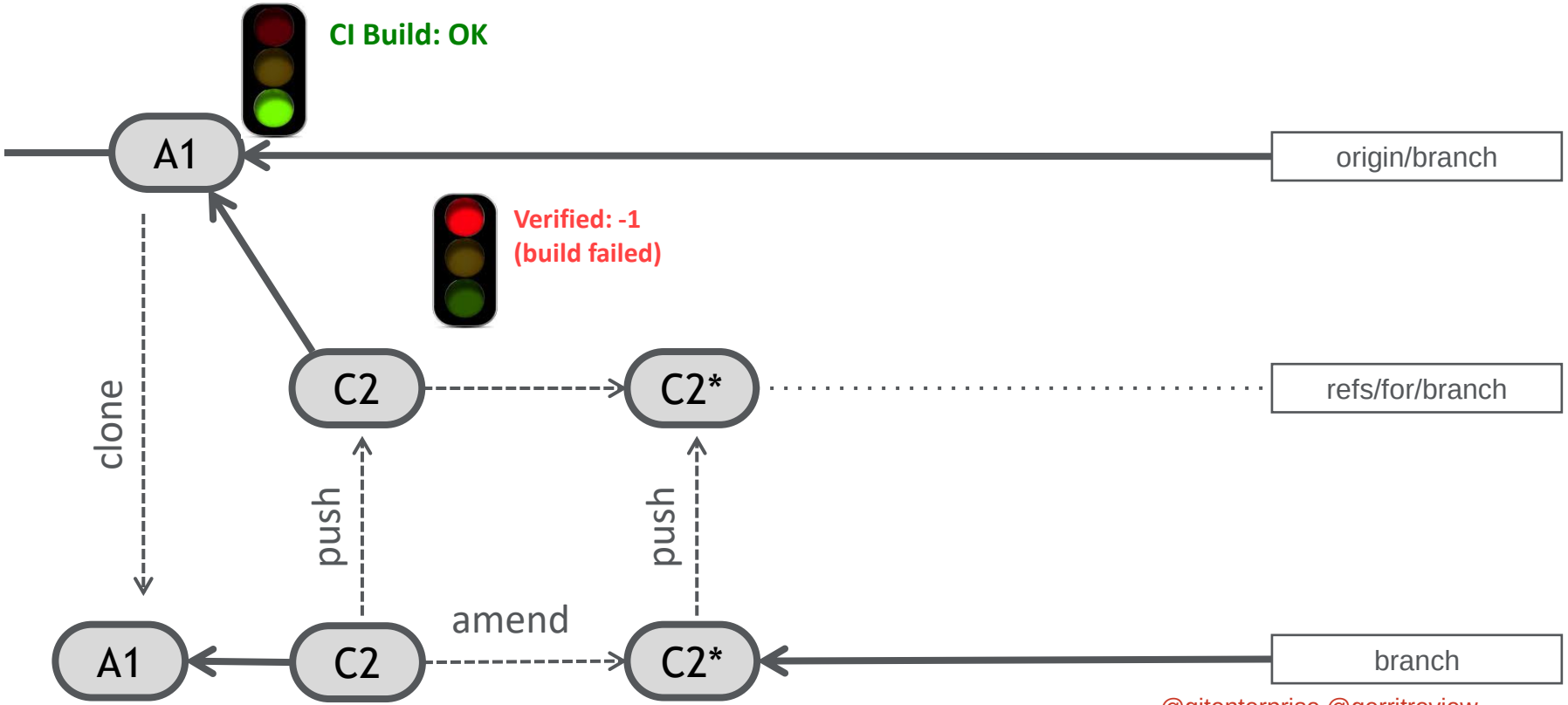
Code Review Workflow (5)

GerritForge



Code Review Workflow (6)

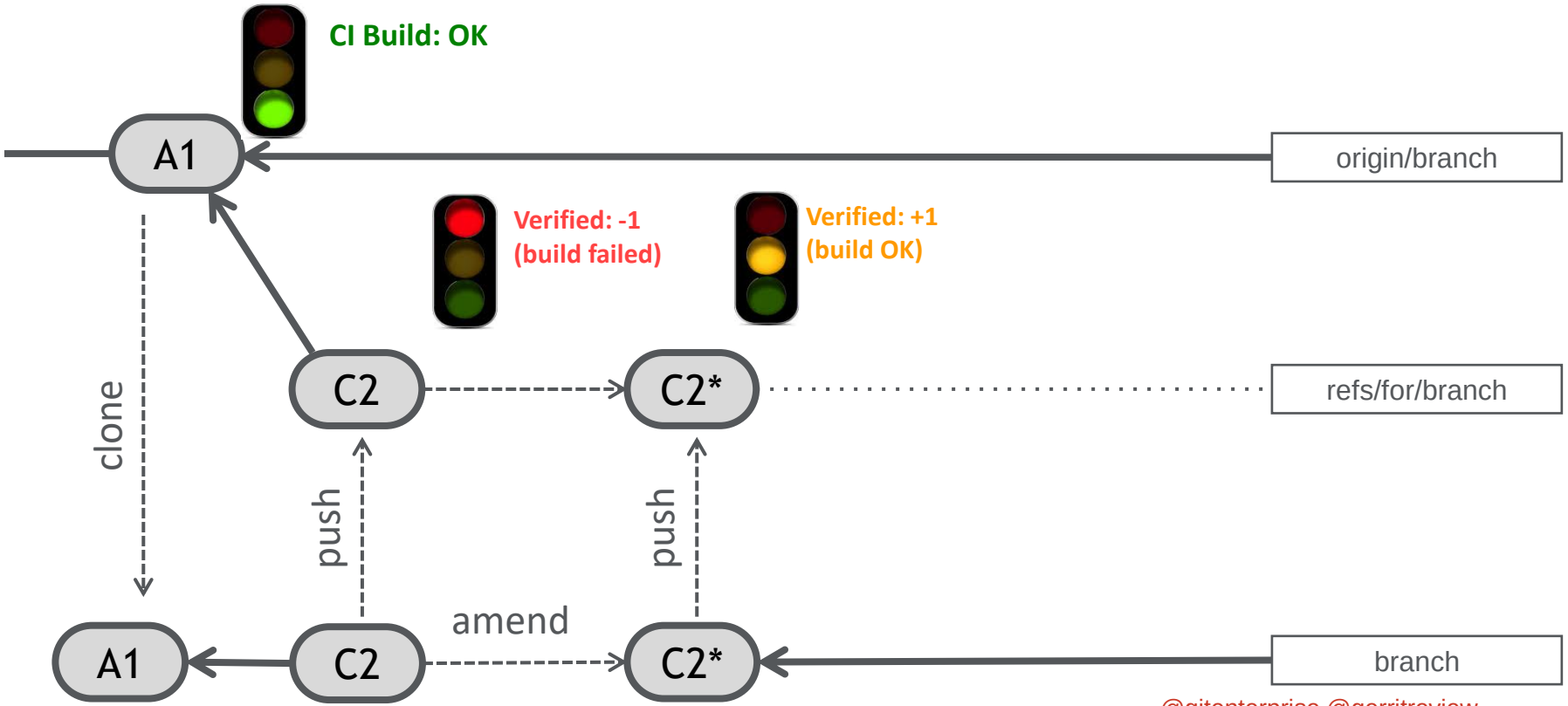
GerritForge



@gitenterprise @gerritreview

Code Review Workflow (7)

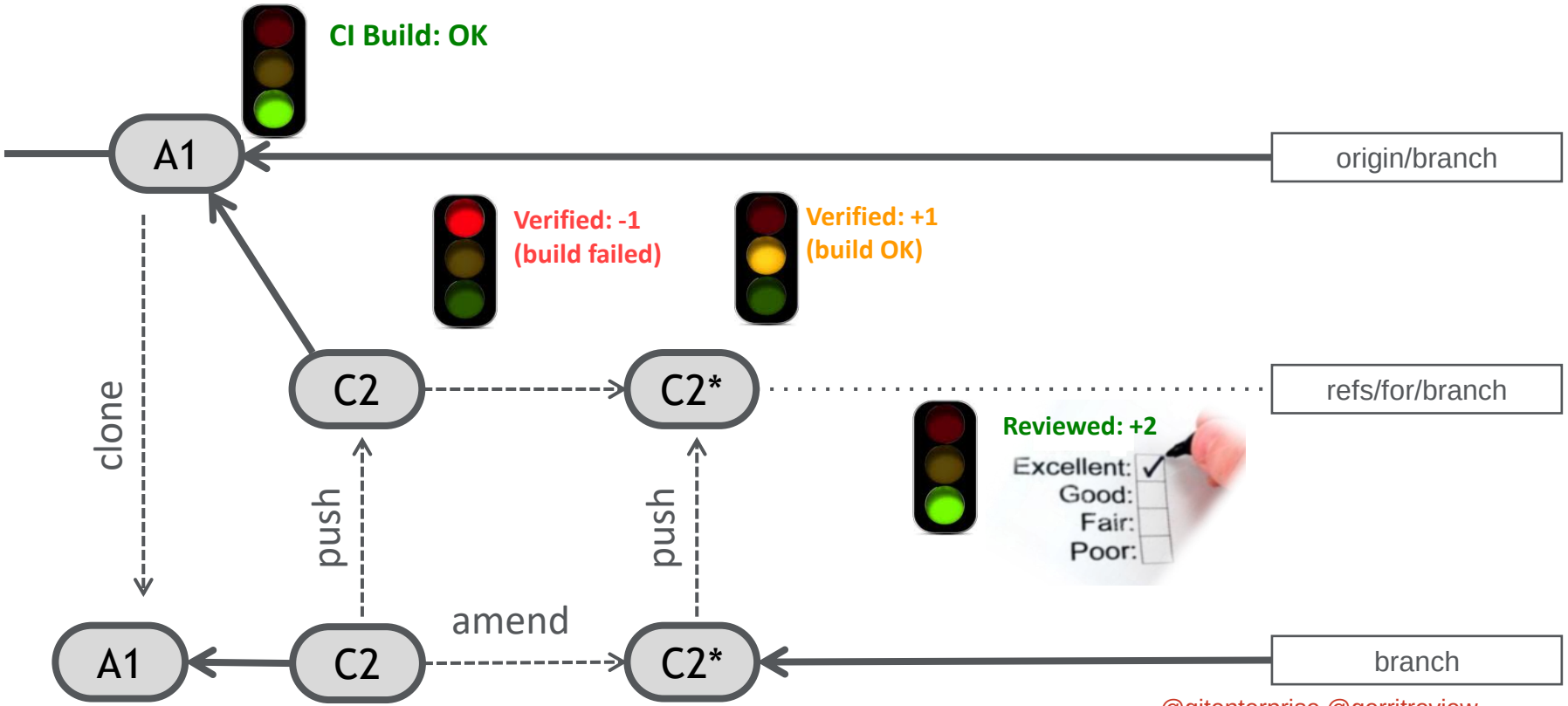
GerritForge



@gitenterprise @gerritreview

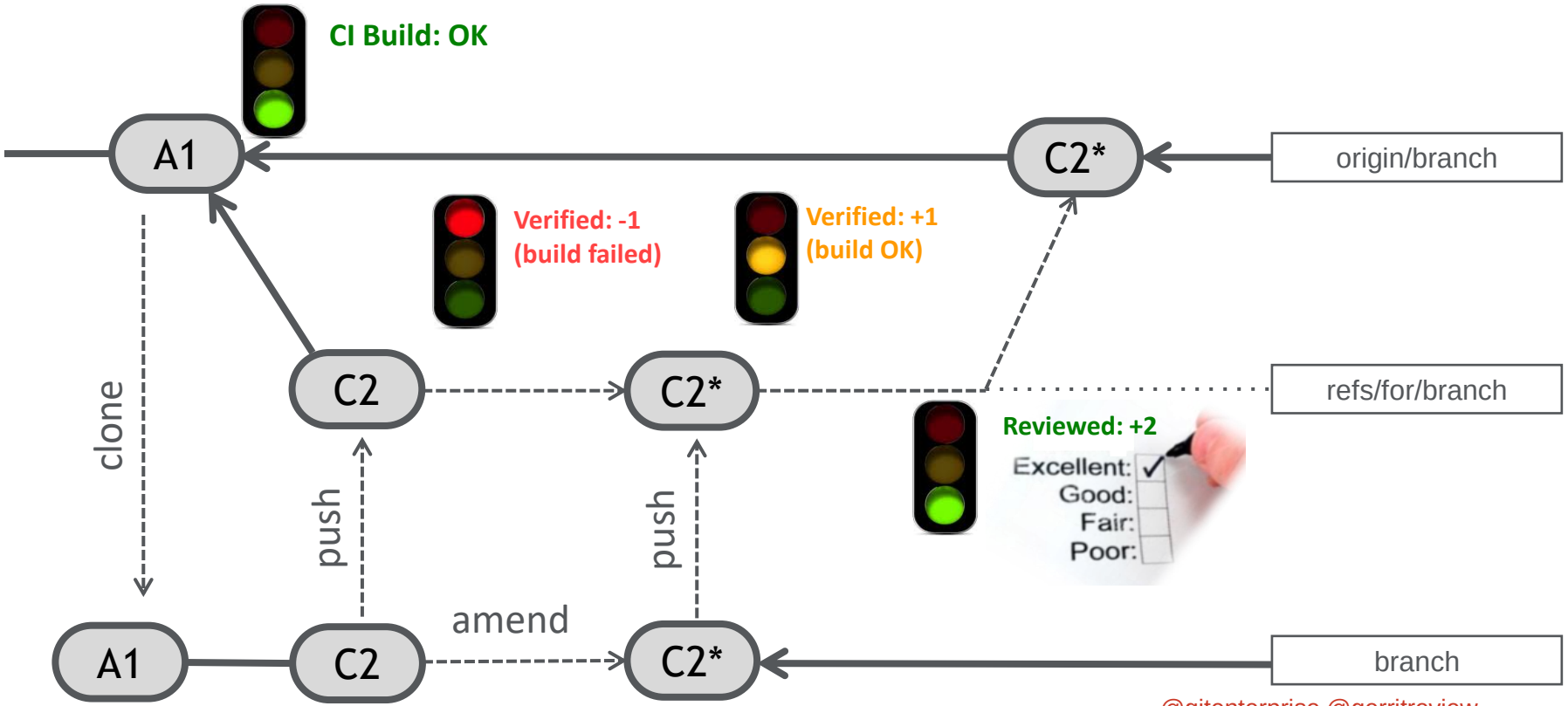
Code Review Workflow (8)

GerritForge



@gitenterprise @gerritreview

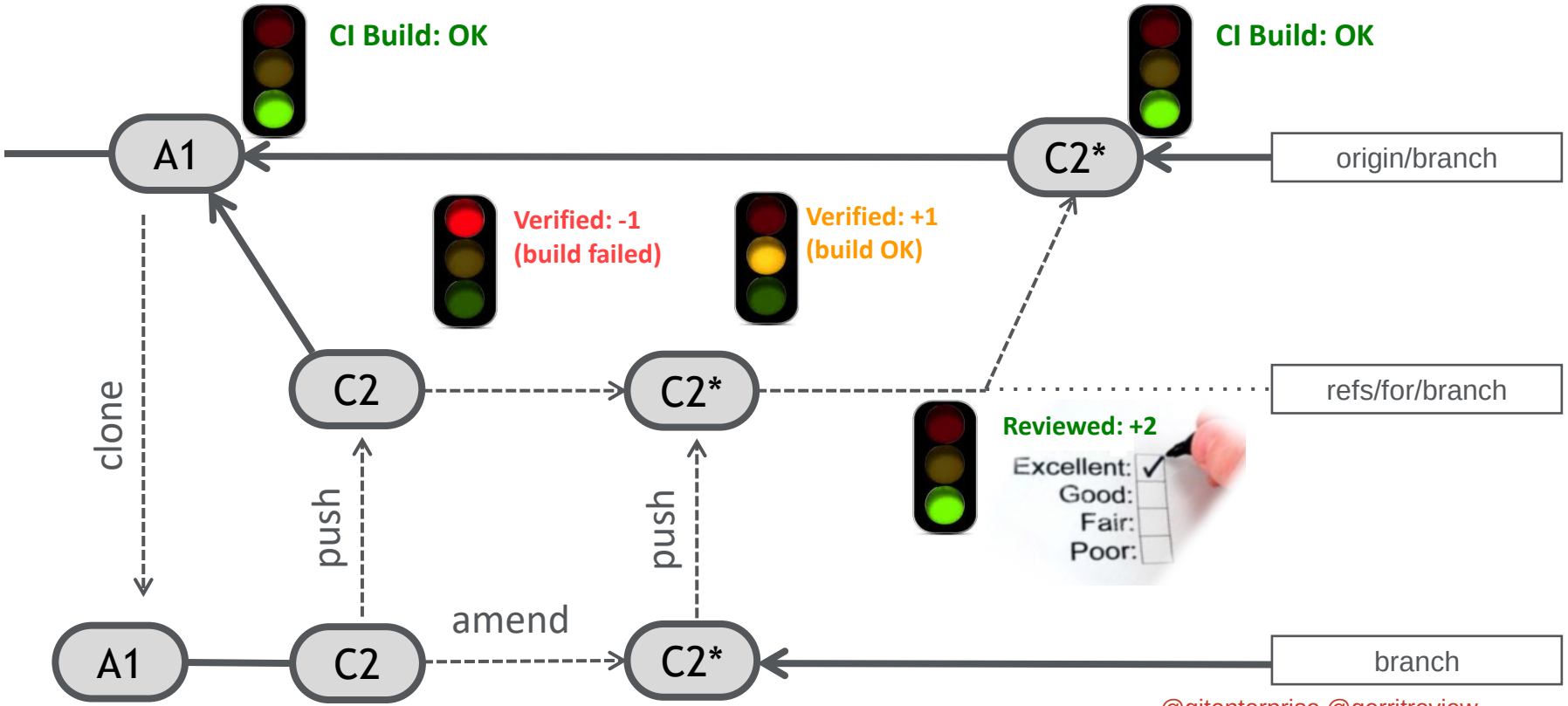
Code Review Workflow (9)



@gitenterprise @gerritreview

Code Review Workflow (10)

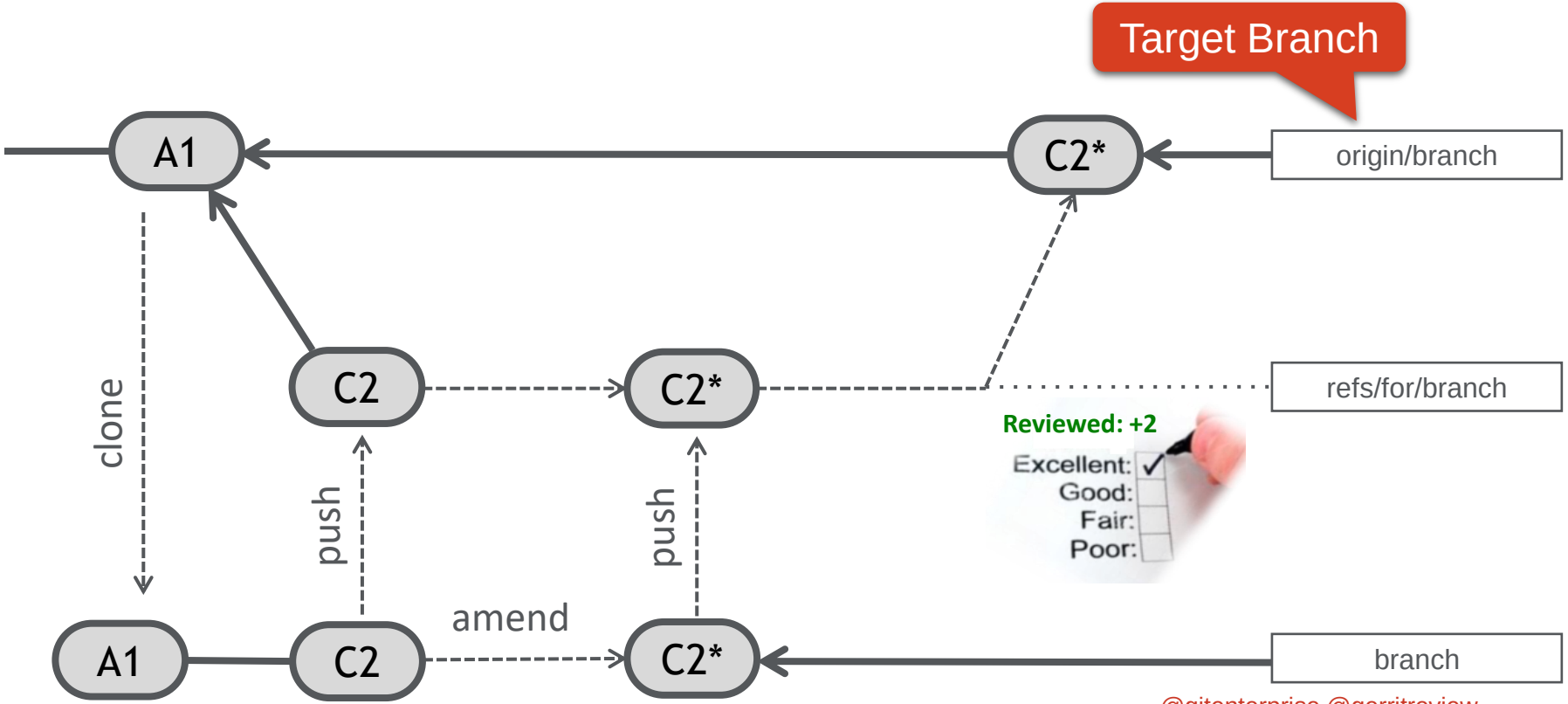
GerritForge



@gitenterprise @gerritreview

Gerrit concepts: target branch

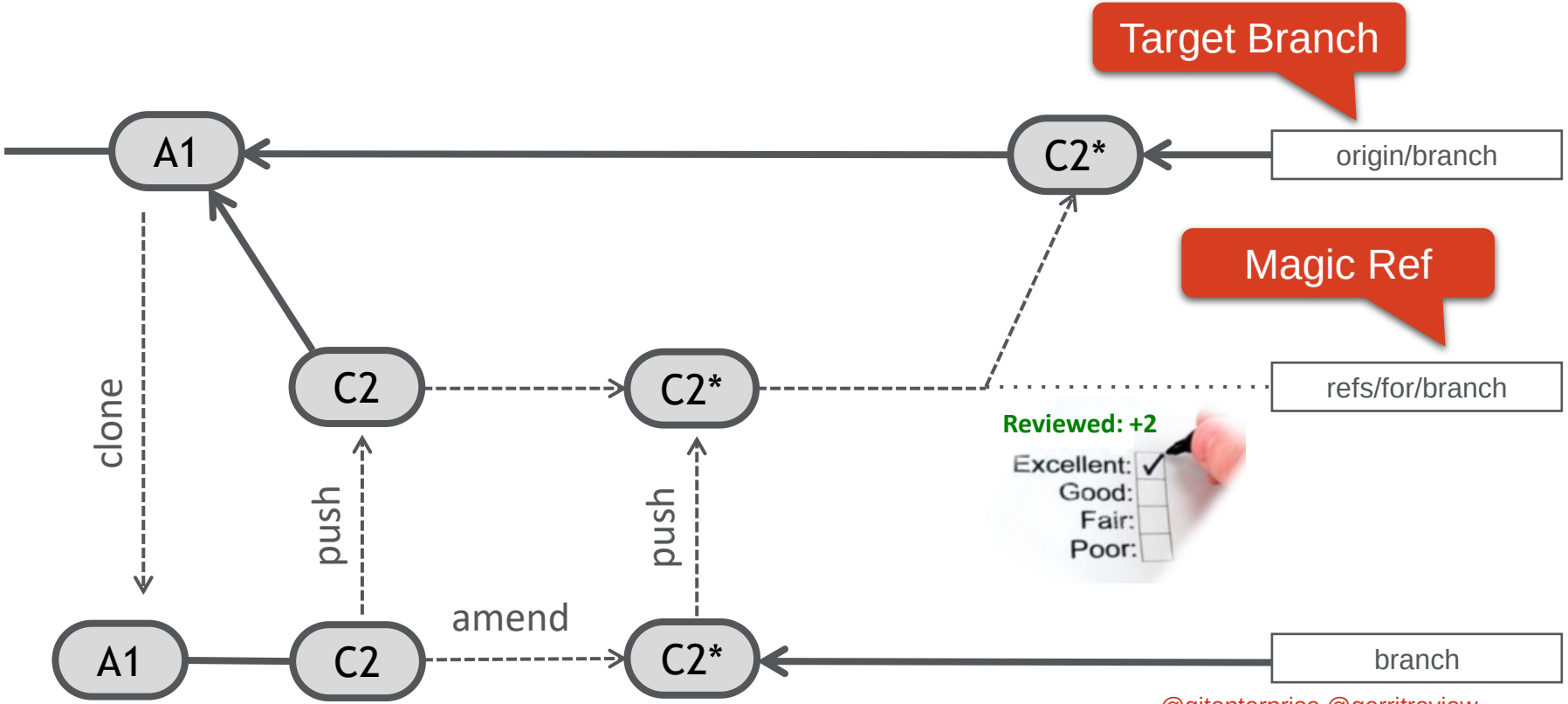
GerritForge



@gitenterprise @gerritreview

Gerrit concepts: magic ref

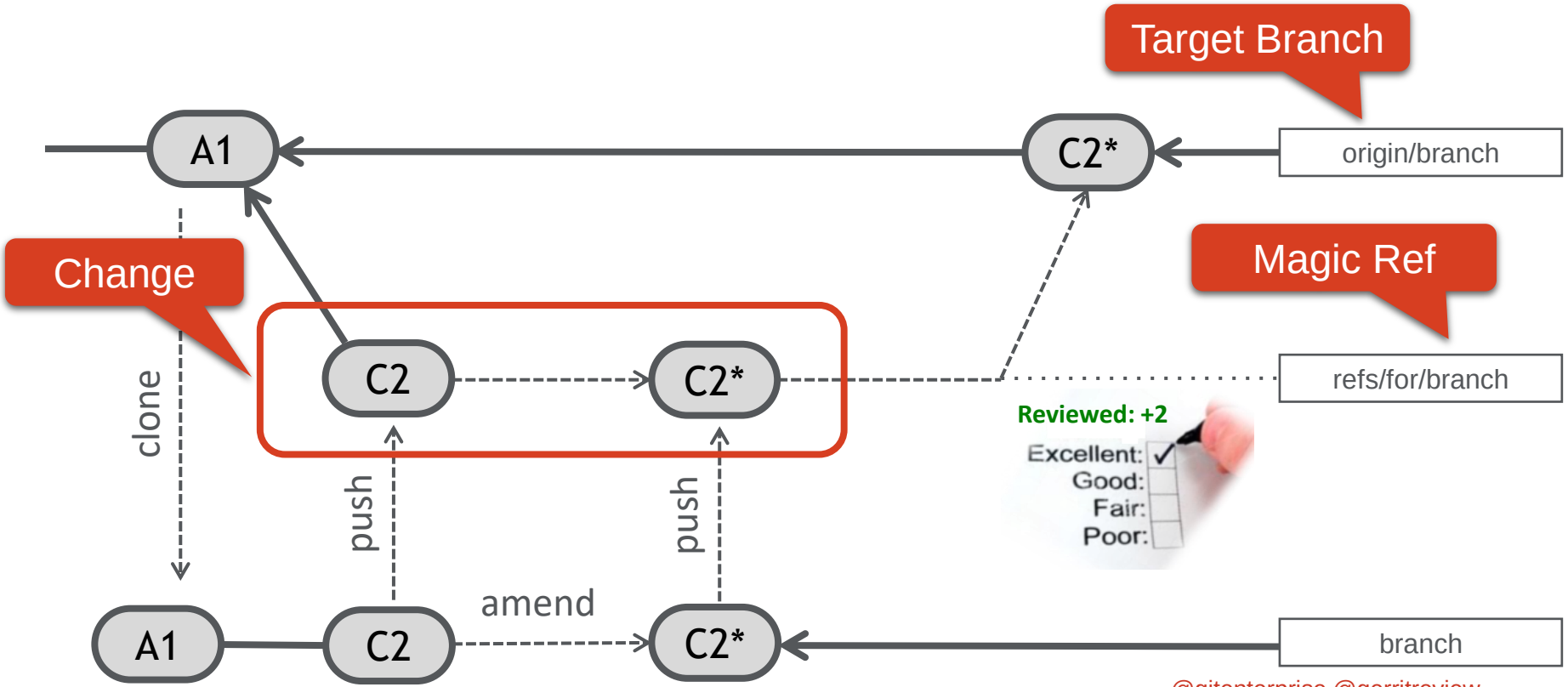
GerritForge



@gitenterprise @gerritreview

Gerrit concepts: change (1)

GerritForge



Gerrit concepts: change (2)

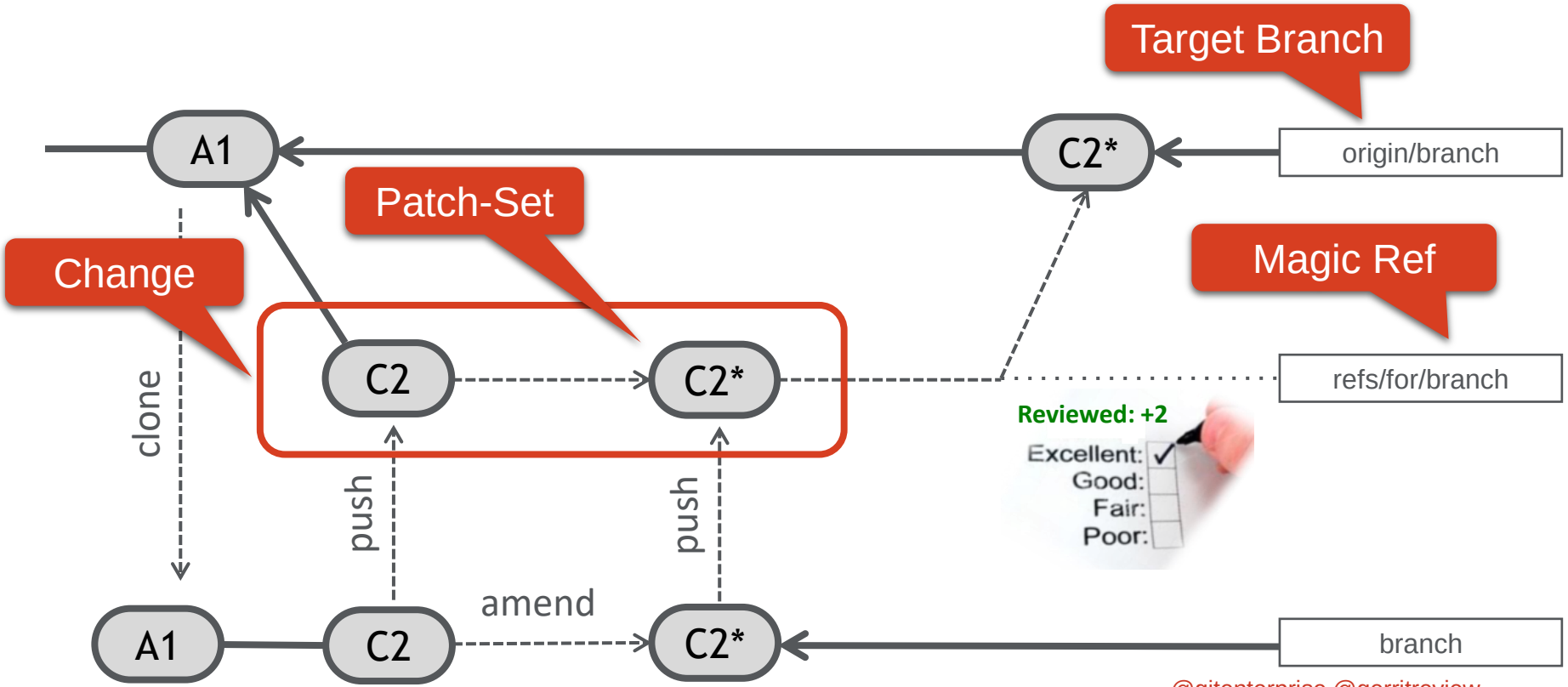
GerritForge

- Assigned by the Git client
- Generated by a Gerrit hook
- Appended to the Git commit footer meta-data
- Kept across multiple amendments to the commit

```
Commit headline for review

This is the Git commit message description
possibly multi-line

Change-Id: I14343c9ef7445558111ccfc345d5e9eb1d60529a
```

Gerrit concepts: patch-set (1)

GerritForge

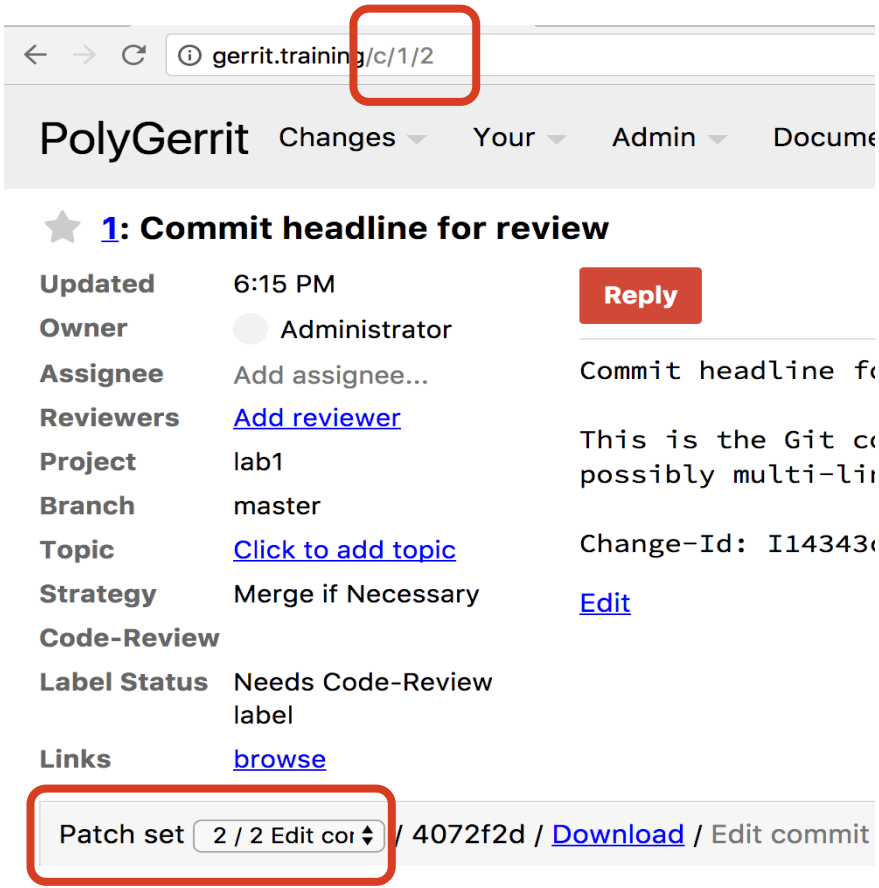


@gitenterprise @gerritreview

Gerrit concepts: patch-set (2)

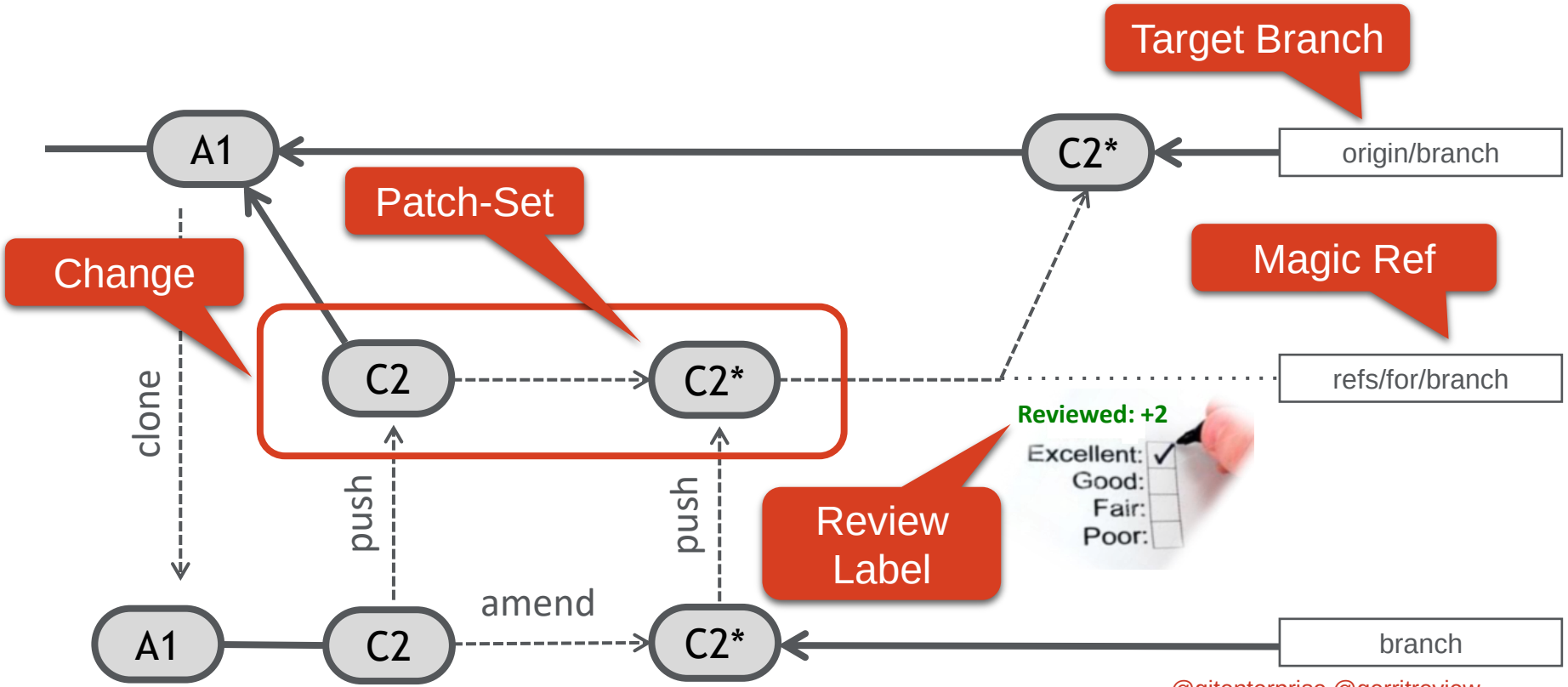
GerritForge

- Assigned by Gerrit Server
- Unique for every commit
- Identified by
  - Change Number (sequence)
  - Patch Number (sequence)



Gerrit concepts: review label and score (1)

GerritForge

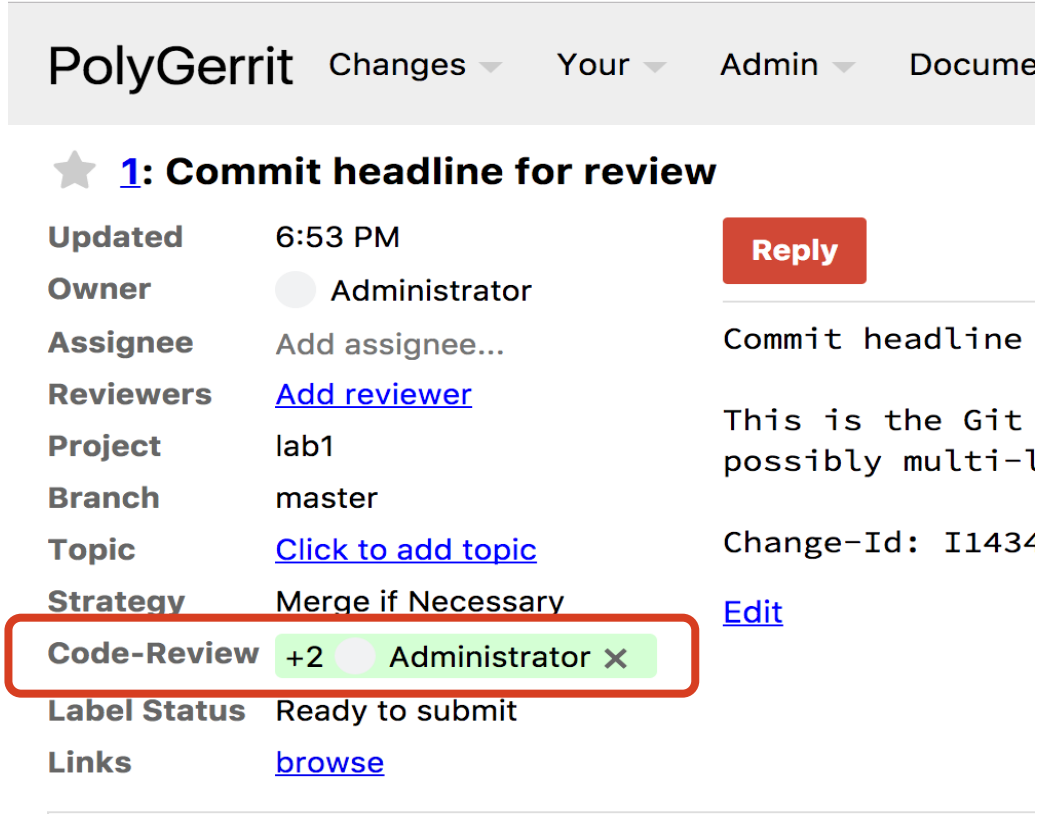


@gitenterprise @gerritreview

Gerrit concepts: review label and score (2)

GerritForge

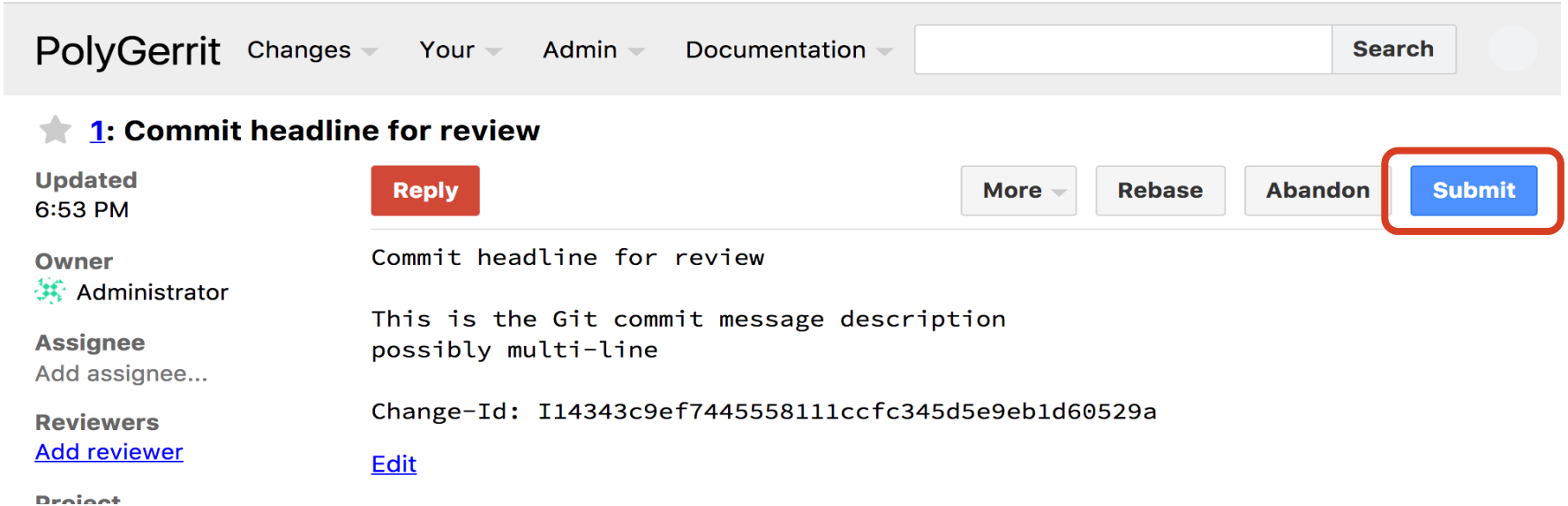
- Label = category of review
- Default Labels:
  - Code-Review (human)
  - Verified (build)
- Score types
  - Neutral (comments)
  - Positive (good)
  - Negative (changes needed)
- Special Scores
  - Max = approved
  - Min = veto



Gerrit concept: Submit / Merge

GerritForge

- Last patch-set ➔ target branch
- Enabled by rules on review
- Default Submit rules: Code-Review = +2 and Verified = +1



GerritForge

Gerrit Code Review building blocks:

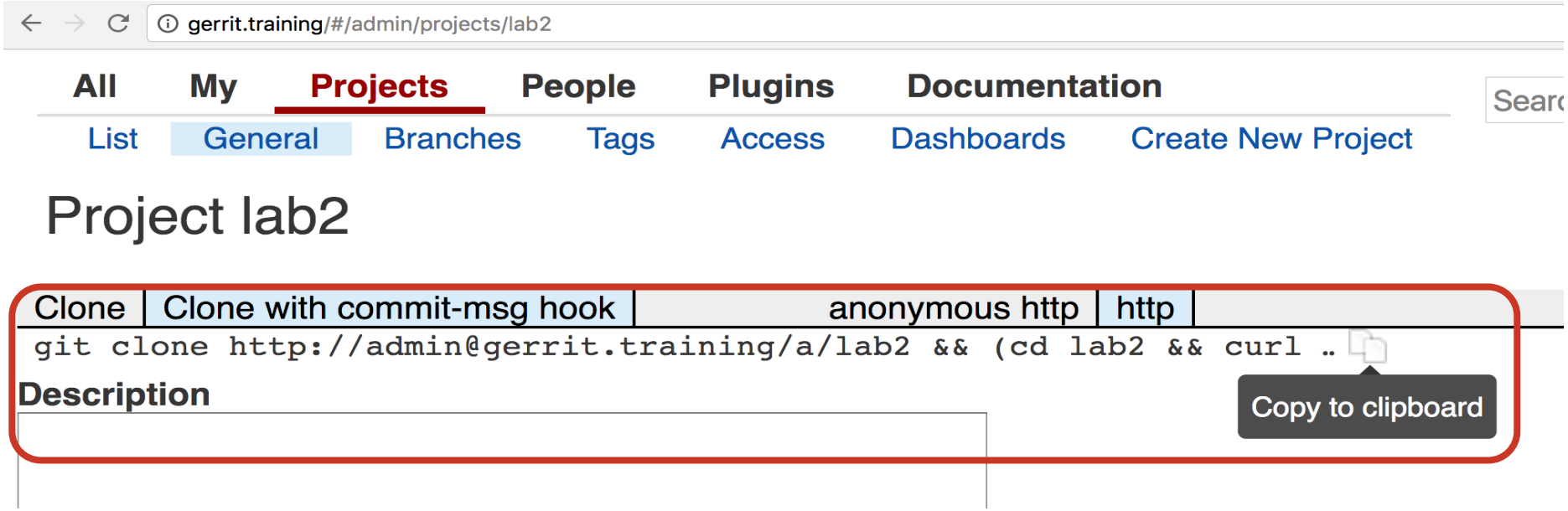
1. Advanced Git Server
2. Review API
3. Extensible Workflow

GerritForge

First Code Review Workflow

Lab#2: Gerrit Project Page

GerritForge



Lab#2: Clone the repository with commit-msg hook

GerritForge

```
$ git clone http://admin@gerrit.training/a/lab2 && (cd lab2 && curl -kLo `git rev-parse --git-dir`/hooks/commit-msg http://admin@gerrit.training/tools/hooks/commit-msg; chmod +x `git rev-parse --git-dir`/hooks/commit-msg)
Cloning into 'lab2'...
remote: Counting objects: 2, done
remote: Finding sources: 100% (2/2)
Unpacking objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 0 (delta 0)
Checking connectivity... done.
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4691  100  4691    0     0   527k      0 --:--:-- --:--:-- --:--:--  572k

$ cd lab2
(master) $
```

Lab#2: Create a local commit and push for review

GerritForge

```
(master)$ echo "My first review" > first-review
(master)$ git add . && git commit -m "My first review"
[master 4ee8383] My first review
 1 file changed, 1 insertion(+)
 create mode 100644 first-review

(master)$ git push origin HEAD:refs/for/master
Counting objects: 3, done.
Writing objects: 100% (3/3), 309 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: Processing changes: new: 1, refs: 1, done
remote:
remote: New Changes:
remote:   http://gerrit.training/2 My first review
remote:
To http://admin@gerrit.training/a/lab2
 * [new branch]      master -> refs/for/master
```

Lab#2: Push for review explained

GerritForge

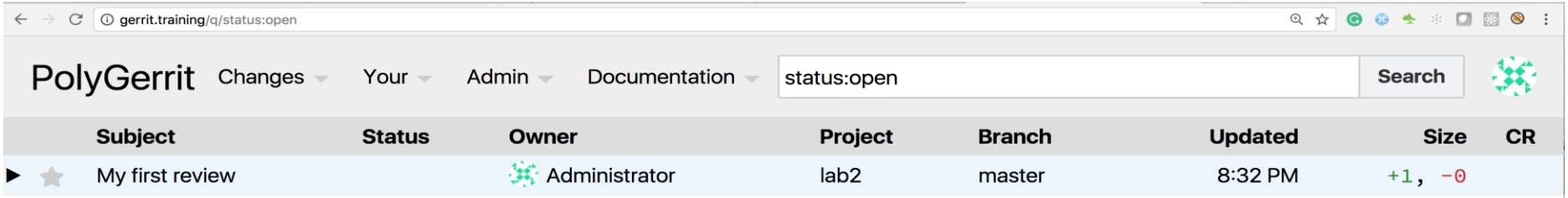
```
(master)$ echo "My first review" > first-review
(master)$ git add . && git commit -m "My first review"
[master 4ee8383] My first review
 1 file changed, 1 insertion(+)
 create mode 100644 first-review

(master)$ git push origin HEAD:refs/for/master
Counting objects: 3, done.
Writing objects: 100% (3/3), 309 bytes/s, done.
Total
remo
remo
remote: New Changes:
remote:   http://gerrit.training/2 My first review
remote:
To http://admin@gerrit.training/a/lab2
 * [new branch]      master -> refs/for/master
```

ref/for/master = magic ref for reviews targeting master

HEAD = latest commit of current branch

Lab#2: List of open changes

GerritForge



Commit headline

Commit author

Gerrit project

Target branch

Lab#2: Change details (1)

GerritForge

@gitenterprise @gerritreview

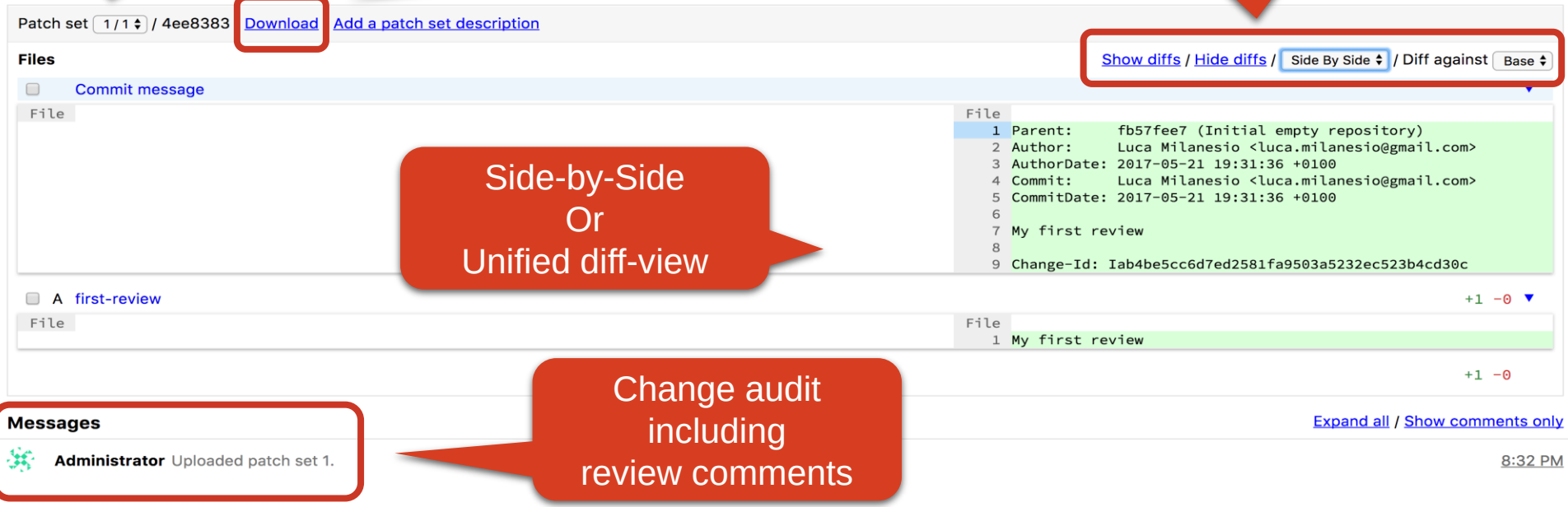
Lab#2: Change details (2)

GerritForge

Current Patch-Set

Git commands to fetch the Patch-Set

Diff view controls

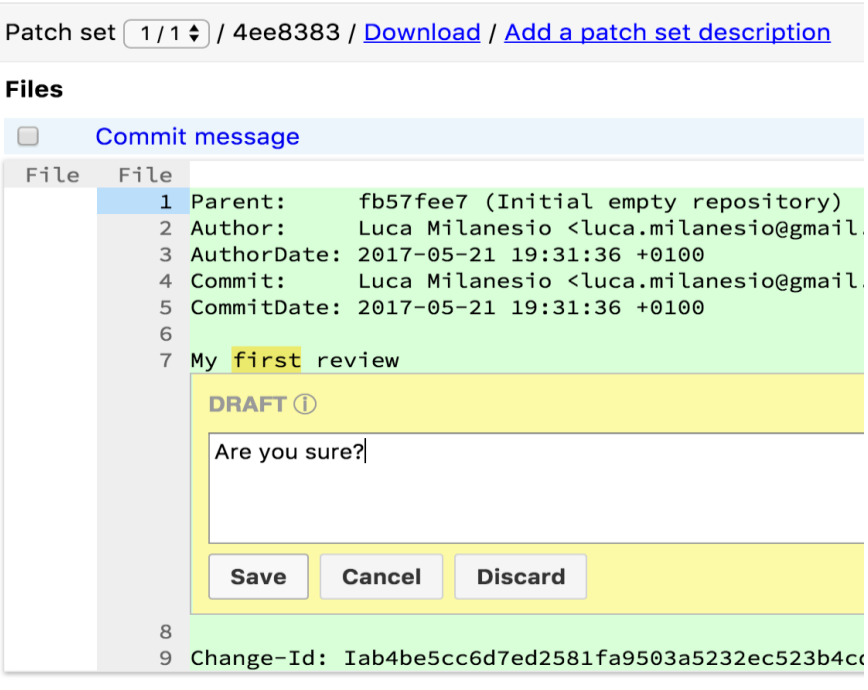


Side-by-Side
Or
Unified diff-view

Change audit
including
review comments

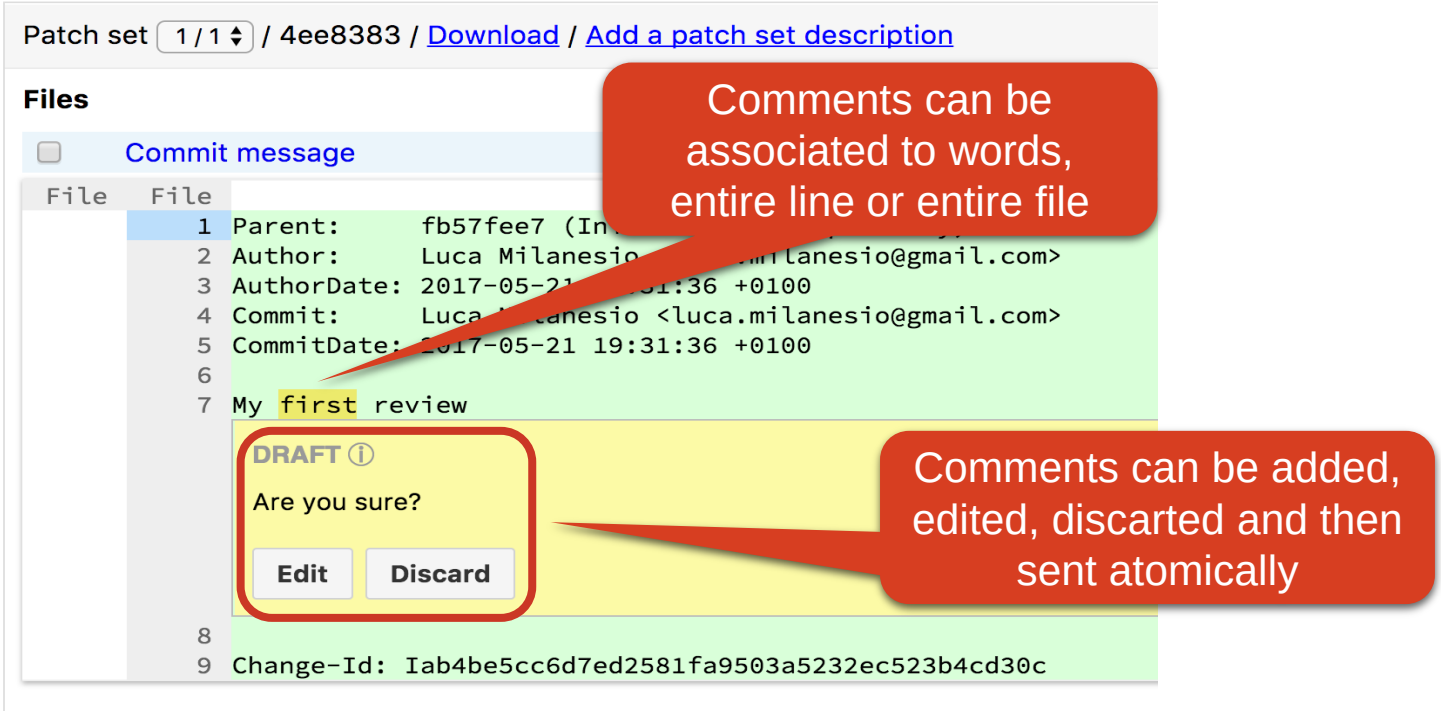
@gitenterprise @gerritreview

Lab#2: Adding in-line comments



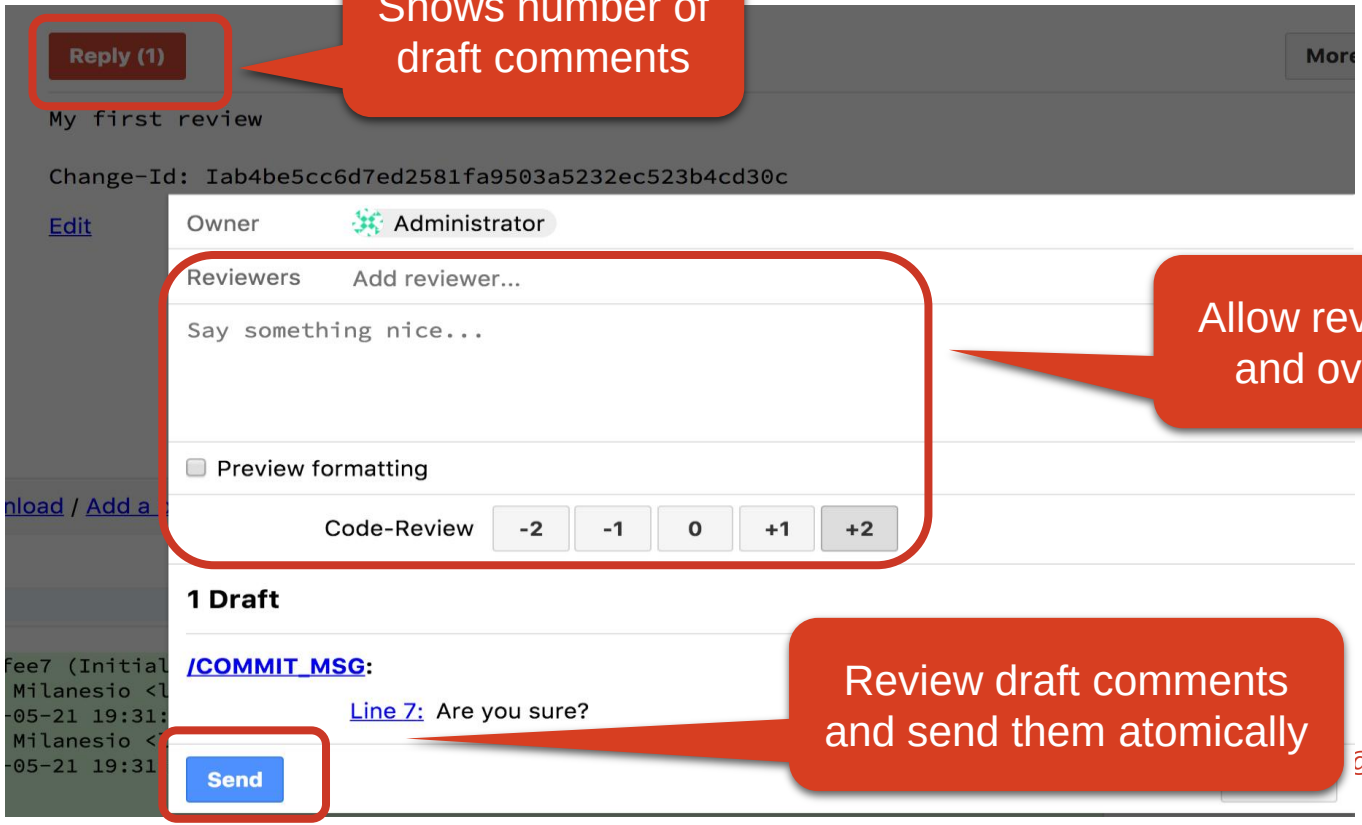
Lab#2: Draft comments review

GerritForge



Lab#2: Sending draft comments

GerritForge



@gitenterprise @gerritreview

Lab#2: Review notification

GerritForge

- All reviewers receive notification via e-mail (see example)

David Pursehouse (Gerrit)
Change in gerrit[stable-2.14]: Bazel: Allow plugins to non-transitively depend on prolog rules
To: David Ostrovsky, Luca Milanesio,
Reply-To: change-107003@gerrit-review.googlesource.com

David Pursehouse **posted comments** on this change.

View Change

Patch set 2: Code-Review +2

(1 comment)

File gerrit-plugin-api/BUILD:

Patch Set #2, Line 22: `"//gerrit-server/src/main/prolog:common",`

Unrelated to this change, because it's like this in other places too, but why do we need to explicitly specify "src/main" here?

To view, visit change 107003. To unsubscribe, visit settings.

eview

Lab#2: Reply to comments and keep the discussion thread

Messages Expand all / Show comments only

Administrator Uploaded patch set 1. 8:32 PM

Administrator 10:43 PM
Patch Set 1: Code-Review+2

(1 comment)

/COMMIT_MSG:

Line 7: Are you sure?

Reply

Administrator 10:43 PM
Patch Set 1:

(1 comment)

/COMMIT_MSG:

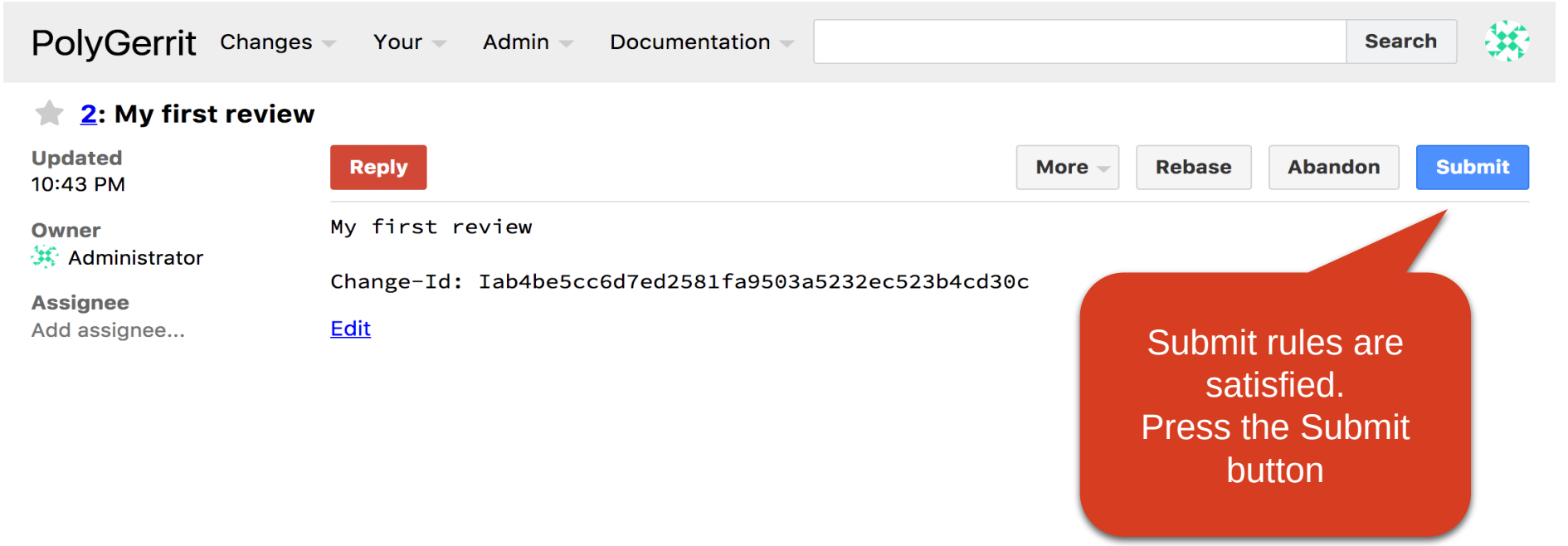
Line 7: Done

Reply

Change messages keep the message thread

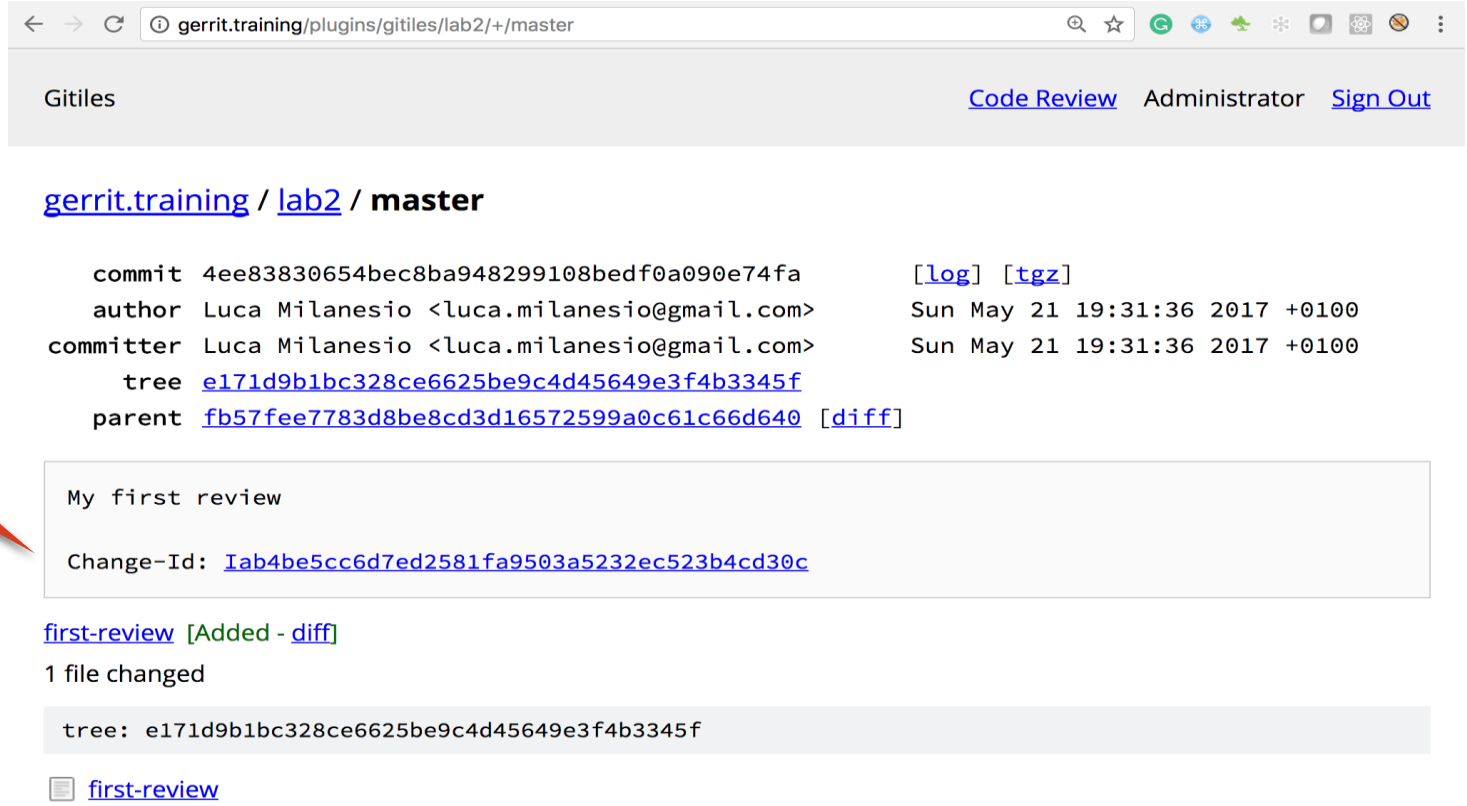
Lab#2: Submit the change

GerritForge



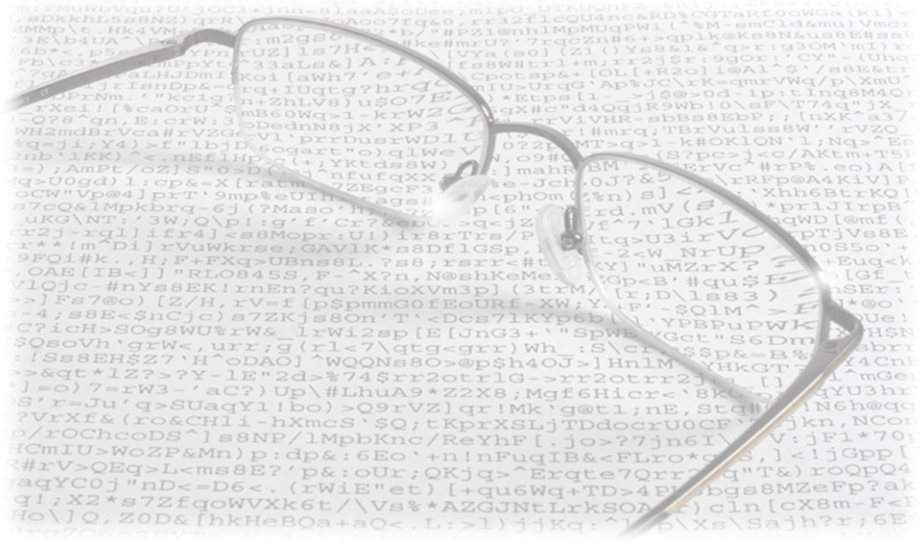
Lab#2: See the change in code browser (Gitiles)

GerritForge



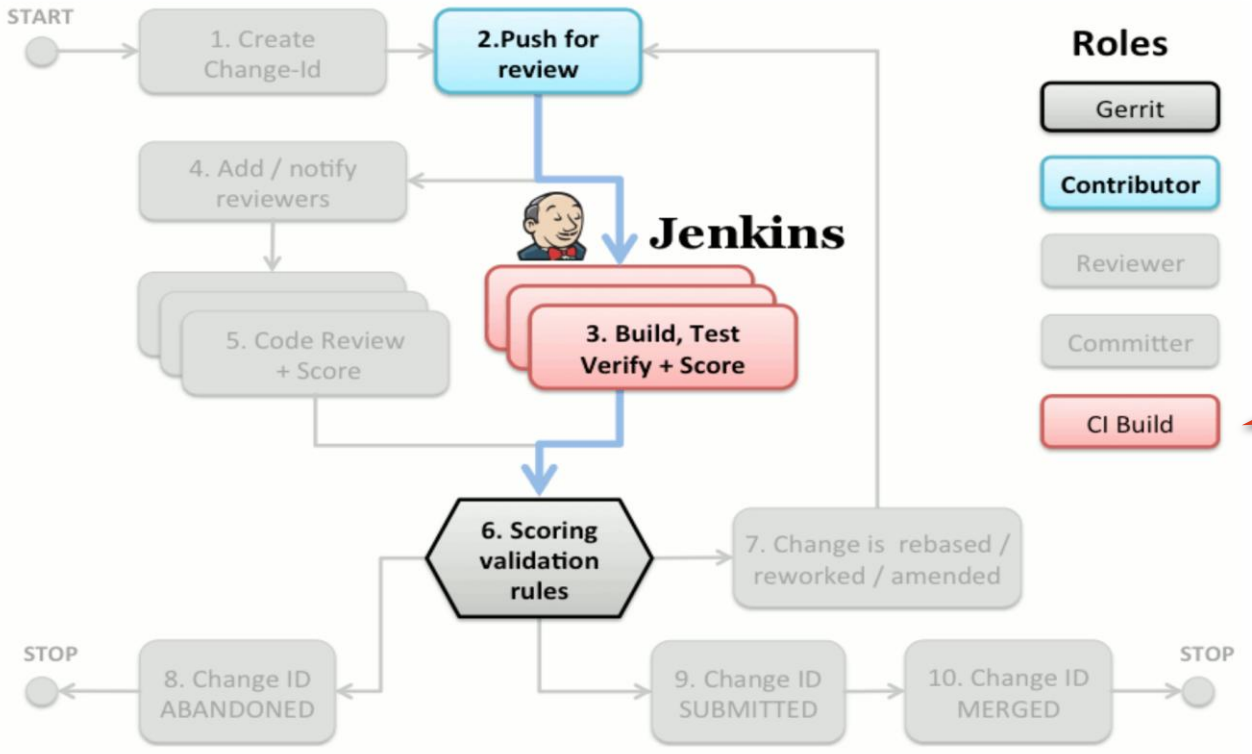
Keeps reference to Change-Id

GerritForge



Validating Commits Automatically

Jenkins Build Validation



Gives Verified +1 / -1 based on the Build and Tests

Automatic scoring

GerritForge

- Post Change / Patch-Set upload trigger
  - Triggers Jenkins build on new Changes
  - Triggers Jenkins build on new Patch-Sets
- Feedback as auto-review
  - Automatic scoring when build completed
  - Score ➔ Submit rule evaluation

Atlassian Jira validation

GerritForge

Create references

Git push

PRJ-87

Jira User

Developer:
`PRJ-87 My code fix`

@gitenterprise @gerritreview

Pre-push validation

GerritForge

- Validate commit during the push
  - Commit message restrictions (regex)
  - Existence of the Jira ticket
- Cross-reference Change to Jira ID
  - Change ➔ Jira
  - Jira ➔ Change
  - Change audit-trail in Jira
- Blocks creation of Change / Patch-Set / Commits
  - Push failed
  - Failure description

Content validation

GerritForge

Check commit policies

Project Onwer
Assign project policies

@gitenterprise @gerritreview

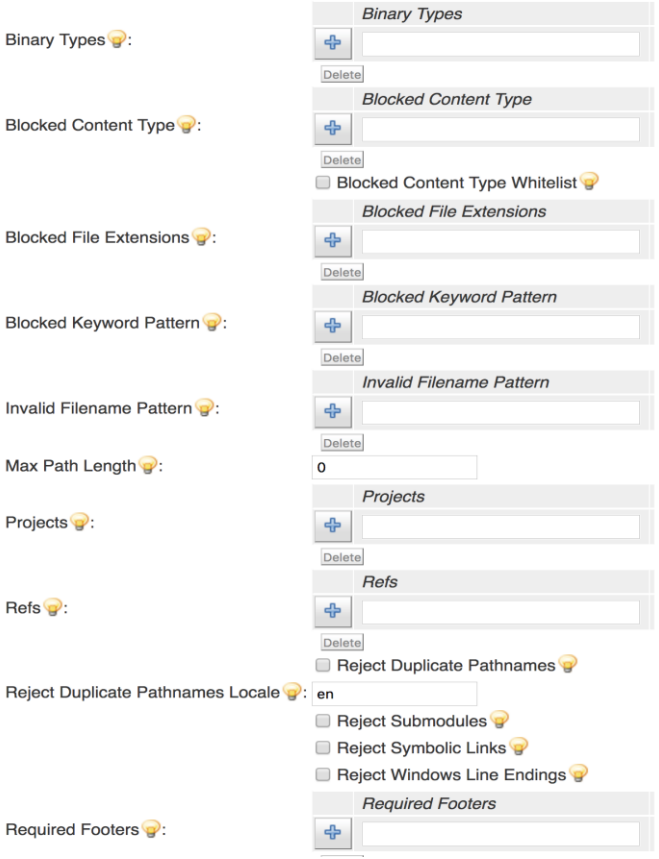
Git push

Developer:
`file1.java`
`my-super-big-file.dat`

Content Validation during push

GerritForge

- Project owner access
- Define blocks
- Require footer
- Block keywords



@gitenterprise @gerritreview

GerritForge

Experiment Jenkins, Jira and Content Validations

Questions?

GerritForge

Q&A

@gitenterprise @gerritreview

Amend Changes – Hands-on Lab#4

GerritForge

Add additional patches to existing Changes

Amend Changes – Hands-on Lab#5

GerritForge

Create dependent Changes

@gitenterprise @gerritreview

Amend Changes – Hands-on Lab#6

GerritForge

Server-side Change rebase and conflict resolution

@gitenterprise @gerritreview

Code Review Etiquette

GerritForge

Etiquette: Making Changes

GerritForge

- Each Change should add something **USABLE**
  - **Change = complete feature**
    OR
  - **Change = complete bug-fix**

- **Each Change should be STABLE**

Etiquette: Making Changes

GerritForge

- Each change should focus on ONLY ONE THING
  - do not mix features, bug-fixes

- Why is this bad?
  - You need the bug-fix in another branch.
    (git cherry-pick is not easy)
  - The feature broke something.
    (git revert cannot be used)
  - It's more difficult to review.

Etiquette: Making Changes

GerritForge

- Push only Changes that are **READY**
  - Q: Who wants to merge unfinished changes?
  - Q: Who wants to review unfinished changes?

- It is unclear for reviewers
  - what is an issue and
  - what is simply not done yet

Etiquette: Making Changes

GerritForge

- Explain the WHY for the Change
  - what was changed can be seen from the diff

- Example: is this commit message good or bad?
  "Disable category GET API"
  - Commit Message without motivation
  - No info about why this had to be disabled.
    … git blame gets less useful ☹

Etiquette: Making Changes

GerritForge

- Make BIG features as series of SMALL Changes
- Each Change adds something usable
- Mark the change series as a topic

Final wrap-up

GerritForge

Q&A

@gitenterprise @gerritreview