

# CollabNet

## Git/Gerrit with TeamForge

Secure, Scalable, Standards-Compliant for the Enterprise

Johannes Nicolai

Director of Engineering CollabNet

 **git** enterprise-grade

# Agenda

---

- Introduction to Git, Gerrit and Jenkins
- Git/Gerrit's Place in the TeamForge Universe
  - Traceability
  - Searchability
  - Scalability
  - Security
- Gerrit Signature Features
  - History Protection
  - Code Quality Gates
- Q & A

# Introduction of Technologies

- Git
  - Most popular emerging distributed SCM
- Jenkins
  - Most popular open source Continuous Integration
- Gerrit Code Review
  - Widely used code-review tool and Git server backend
- CollabNet TeamForge®
  - Provides an unparalleled smooth integration of all these tools into your software development process

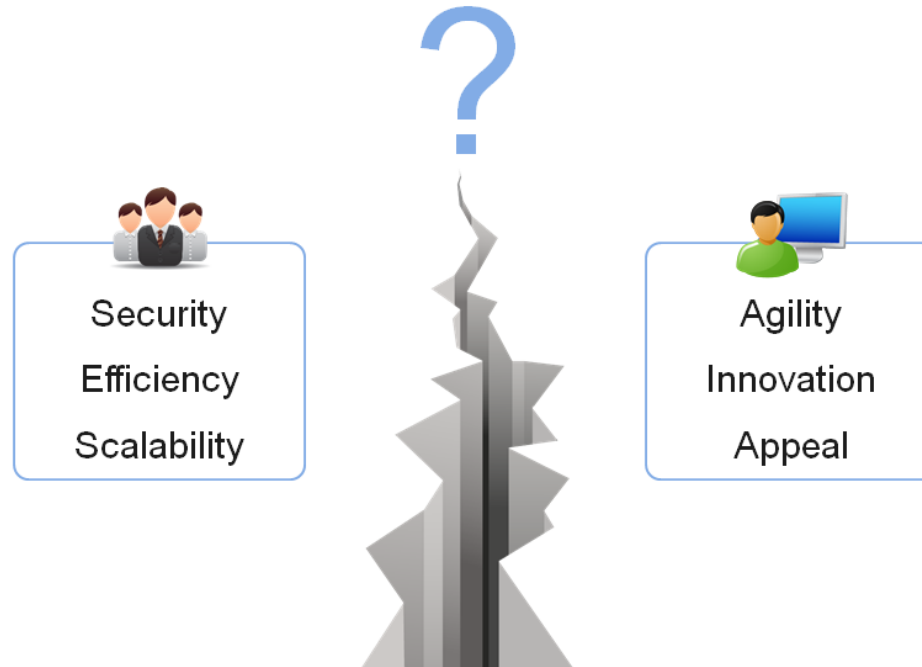


# Git = Leading DVCS (Distributed Version Control System)

- Open source distributed SCM, used notably by:
  - Linux (it was originally developed for that very purpose)
  - Eclipse foundation
  - Android (based on Gerrit Code Review)
- Why Git?
  - All operations are local
    - Search, log, history, branching and merging
    - Offline development is possible
  - Powerful merging strategies (merge, rebase, cherry-pick)



# Git Popularity Continues to Grow, But....



The adoption of DVCS has accelerated in small teams, but is moving more slowly in enterprise settings.

Source: Gartner, 27 July 2011 ID Number: G00214153; Hype Cycle for Application Development, 2011; Ian Finley, Mike Blechar

# Gerrit = (Widely Used) Git Server + Code Review Tool

- 100% pure Java SSH and HTTP Git backend
  - Scalable and suitable for large Enterprises
- Web-based and Command Line administration
  - Users and Groups
  - Project and branch security (read and write)
- Collaboration and Code-review
  - B2B integration at code-level
  - Communicate and share code knowledge in the Team
  - Enforce workflow and code quality
- The most successful open source code review
  - Android OS
  - Eclipse, OpenStack, Qt, CouchDB and many others



# Who's Using Gerrit in Production Today?

**SONY**



**ebay**

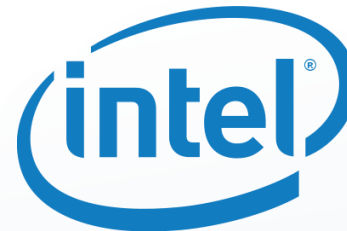
**QUALCOMM**



**Google**

**BAE SYSTEMS**

**GARMIN**



**TOMTOM**



source: Wikipedia

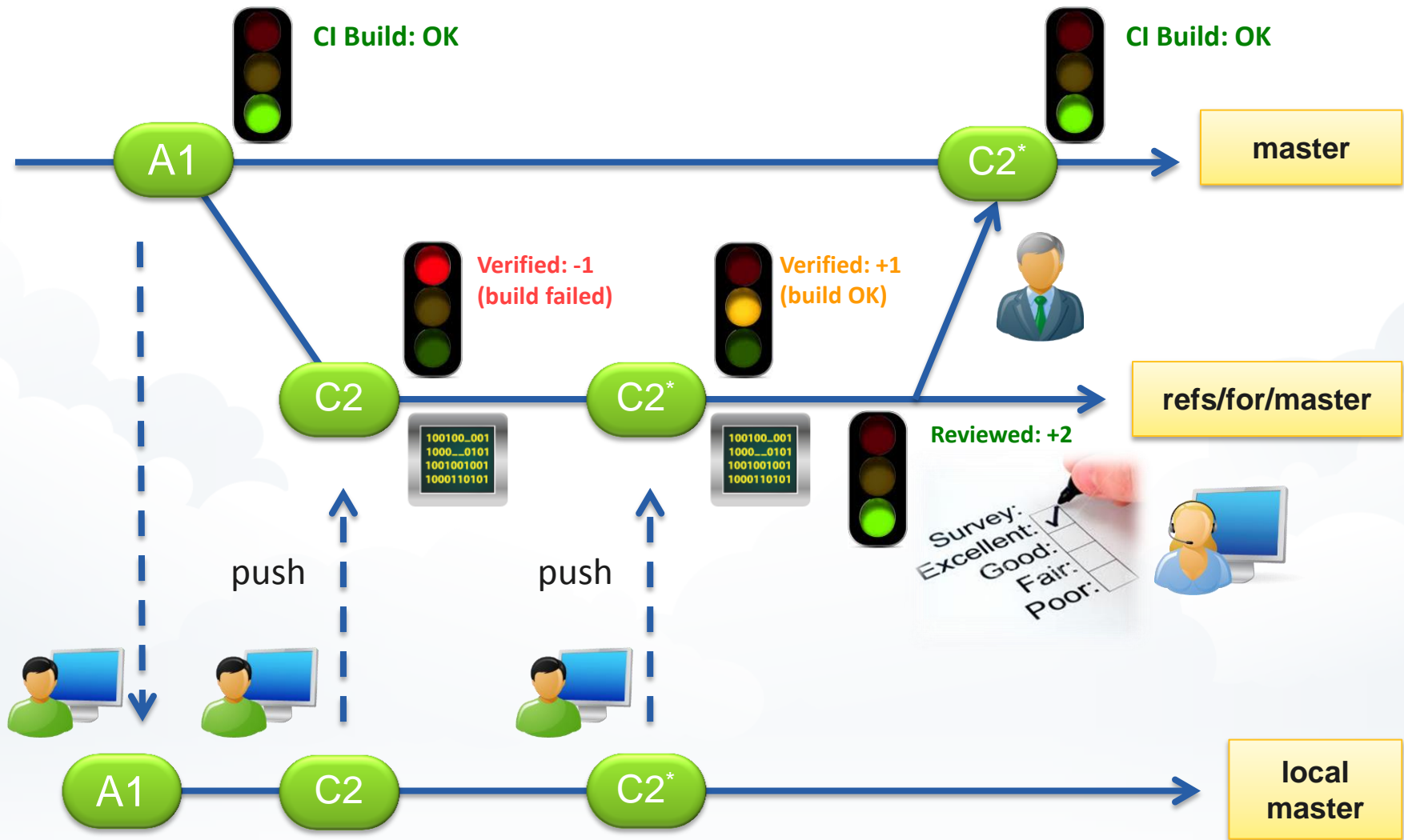
# Jenkins = Most popular Open Source CI



# Jenkins

- Has hundreds of plugins to
  - Interact with various SCMs (Git, SVN, CVS, ...)
  - Build software (ant, maven, gradle, make)
  - Run unit/integration tests (JUnit, Selenium, ...)
  - Perform static code analysis (findbugs, checkstyle, PMS, Sonar, ...)
- Works brilliantly together with Gerrit over Gerrit Trigger Plugin
  - Keeps ssh connection open to learn about new review requests immediatly
  - Builds and verifies all configures quality gates (tests, coding conventions, code KPIs, you name it)
  - Sets ‘verified’ flag in review request according to result of verification build (either +1 or -1)
  - If Jenkins sets -1, then further code review isn’t possible, developer has to either re-work or abandon change

# Tackling half baked review requests: Marrying Gerrit and Jenkins



# TeamForge = Enterprise-Grade Git Mgmt. + ALM



- ✓ Basic SCM features
- ✓ jGit engine
- ✓ Native engine
- ✓ GitWeb



## Gerrit

- ✓ Advanced Git security
- ✓ Git Projects organization
- ✓ Code-review
- ✓ Replication

## TeamForge.

- ✓ 24/7 Support
- ✓ Git/Gerrit Training
- ✓ Standards-Compliance
- ✓ Scalability
- ✓ ALM Integration
- ✓ History Protection
- ✓ Code Quality

With TeamForge and Gerrit, Git is now ready for the enterprise. Realize all the benefits of Git, without compromising governance, security and compliance.

# TeamForge = Enterprise-Grade Git Mgmt. + ALM



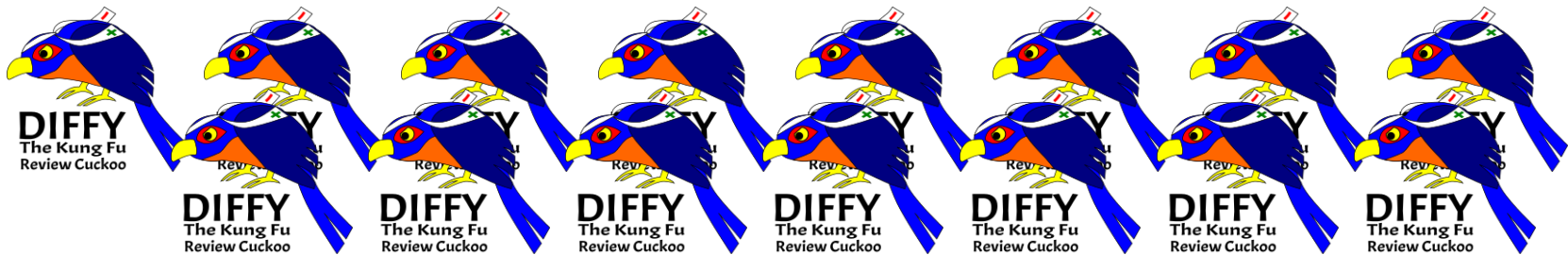
Integration of existing tools



Security, standards and audit compliance



Scalability (and manageability) at



# TeamForge = Enterprise-Grade Git Mgmt. + ALM



**Searchability** – across TF artifacts, including Code Search across multiple repositories, gerrit instances and other SCM tools (svn, cvs, perforce...)


**Traceability** – associating the source code with requirements, issues, documents, tasks and other artifacts.



**RBAC** – Role Based Access Control - for each application's concepts, documents, file releases, trackers, and discussion forums, you can assign permissions globally based on user roles

# Simple RBAC - TeamForge

## Source Code Permissions

 Selected permissions in this section get inherited to subprojects.

Source Code Admin:  (Administer / Create / Edit / View)



Delete/View:  All Repositories

Commit/View:  All Repositories

View only:  All Repositories

## Permissions for Specific Repositories

 Permissions in this section are not inherited to subprojects.

Repositories	Permissions
demo_repo	<input checked="" type="radio"/> No Access <input type="radio"/> View Only <input type="radio"/> View and Commit <input type="radio"/> View, Commit and Delete
Publishing	<input checked="" type="radio"/> No Access <input type="radio"/> View Only <input type="radio"/> View and Commit <input type="radio"/> Path-based Permissions  

# Gerrit Access Rights- Not simple at all

<b>Forge Server Identity</b>	<input type="checkbox"/> Exclusive
ALLOW TF: demo: Admin	
<b>Push</b>	<input type="checkbox"/> Exclusive
ALLOW TF: demo: Admin	<input checked="" type="checkbox"/> Force Push
<b>Label Code-Review</b>	<input type="checkbox"/> Exclusive
-2 +2 TF: demo: Admin	
-2 +2 TF: demo: Developer	
-2 +2 TF: demo: Release Manager	
-1 +1 TF: demo: Observer	
<b>Label Verified</b>	<input type="checkbox"/> Exclusive
-1 +1 TF: demo: Admin	
-1 +1 TF: demo: Developer	
-1 +1 TF: demo: Release Manager	
<b>Submit</b>	<input type="checkbox"/> Exclusive
ALLOW TF: demo: Admin	
ALLOW TF: demo: Developer	
ALLOW TF: demo: Release Manager	
<b>View Drafts</b>	<input type="checkbox"/> Exclusive
ALLOW TF: demo: Admin	
ALLOW TF: demo: Developer	
ALLOW TF: demo: Observer	
ALLOW TF: demo: Release Manager	
<b>Delete Drafts</b>	<input type="checkbox"/> Exclusive
ALLOW TF: demo: Admin	

# RepoCategories: Bridging the RBAC gap

## Motivation:

Shield most users from Gerrit's complexities by using pre-defined repo categories and generic TeamForge RBAC model with its SCM permissions.

without hiding advanced features for power users

## Implementation:

- Map TeamForge SCM permissions to Gerrit Access Rights.
- Provide pre-defined **code review policies** so anyone can start to work with Gerrit within a few steps.
- **Code review policies** are applied per repository.

# RepoCategories: Bridging the RBAC gap

## Edit Repository

Directory Name:\*/ `/gitroot/demo_repo`

Repository Name:\*/

Description:

Server: **Git**

Repository Category

- Default - no review
- Optional review
- Mandatory review
- Custom
- User defined:

Protect History:

# TeamForge = Enterprise-Grade Git Mgmt. + ALM



- ✓ Basic SCM features
- ✓ jGit engine
- ✓ Native engine
- ✓ GitWeb



## Gerrit

- ✓ Advanced Git security
- ✓ Git Projects organization
- ✓ Code-review
- ✓ Replication

## TeamForge.

- ✓ 24/7 Support
- ✓ Git/Gerrit Training
- ✓ Standards-Compliance
- ✓ Scalability
- ✓ ALM Integration
- ✓ History Protection
- ✓ Code Quality

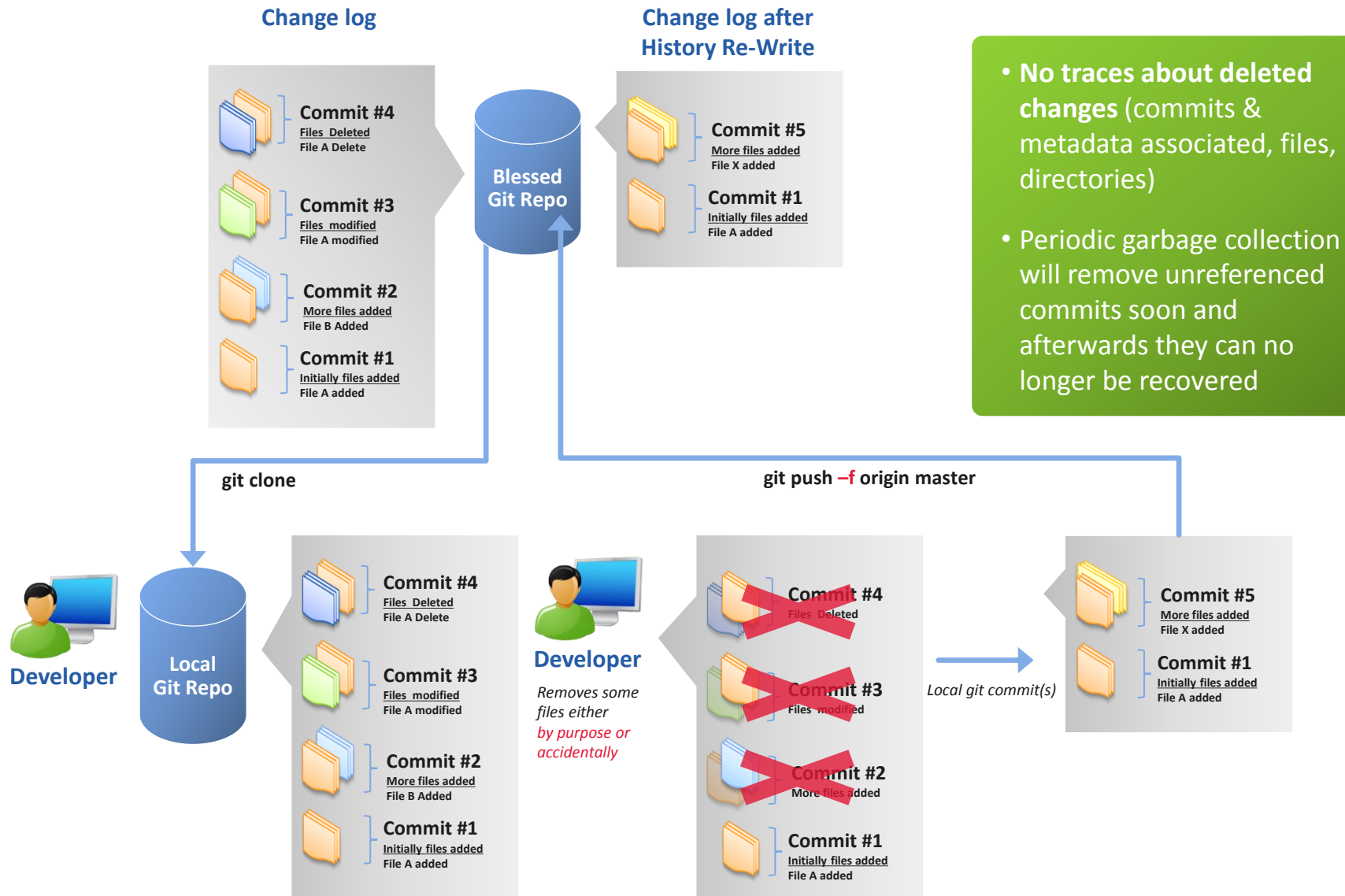
With TeamForge and Gerrit, Git is now ready for the enterprise. Realize all the benefits of Git, without compromising governance, security and compliance.

# Git History Rewrite – Defined

---

“We define History Rewrite as non-fast forward updates of remote refs and its associated objects. This happens whenever a branch in the remote repository gets deleted, previously pushed commits get amended/tree filtered and forcefully re-pushed, or a remote branch/tag is pointed to an entire different commit history.”

# Git History Rewrite – Functionality in Detail



- No traces about deleted changes (commits & metadata associated, files, directories)
- Periodic garbage collection will remove unreferenced commits soon and afterwards they can no longer be recovered

# Git History Rewrite – No Traces Left Behind!

github | Search or Type a Command | Explore Gist Blog Help | dharmsheta

dharmsheta / Spoon-Knife-1

Code | Network | Pull Requests | Graphs | Admin

branch: master | Files | Commits | Branches | Tags | Downloads

Spoon-Knife-1 / Commit History

- Apr 21, 2011: Merged pull request #9.
- Apr 01, 2011: Added hidden double rainbow reference; added title
- Mar 26, 2011: lowercase doctype html, uppercase untitled, no width/height on img, s...
- Mar 02, 2011: doctype html
- Feb 25, 2011: + Heh.
- Feb 24, 2011: Updated the name and readme
- Jan 27, 2011: First commit

Code Change History

Removing top 6 commits locally

Creating new commit locally

Pushing all changes to remote

```
sheta@SHETA-THINK ~/Spoon-Knife-1 (master)
$ git reset --hard HEAD~6
HEAD is now at 090cf8d First commit
```

```
sheta@SHETA-THINK ~/Spoon-Knife-1 (master)
$ echo "deleted">>history.gone && git add history.gone

sheta@SHETA-THINK ~/Spoon-Knife-1 (master)
$ git commit -m "Rewriting History"
[master 2358618] Rewriting History
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 history.gone

sheta@SHETA-THINK ~/Spoon-Knife-1 (master)
$ git push -f origin master
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 352 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:dharmsheta/Spoon-Knife-1.git
+ bdd3996...2358618 master -> master (forced update)
```

No traces about deleted 6 commits in Web UI

github | Search or Type a Command | Explore Gist Blog Help | dharmsheta

dharmsheta / Spoon-Knife-1

Code | Network | Pull Requests | Graphs | Admin

branch: master | Files | Commits | Branches | Tags | Downloads

Spoon-Knife-1 / Commit History

- Nov 14, 2012: Rewriting History
- Jan 27, 2011: First commit

# Git History Rewrite – This Can Happen to You, Too!

## Single ‘space’ wipes entire history:

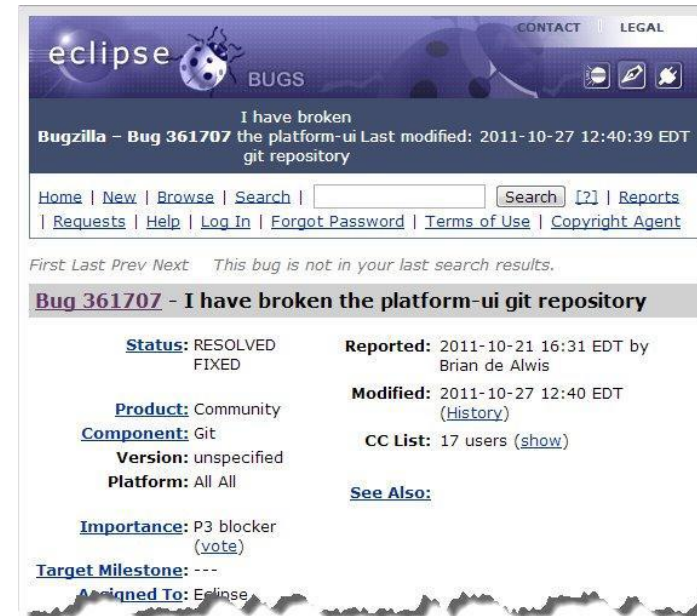
- git push origin production-quick-fix:stable  
*instead of*
- git push origin production-quick-fix :stable

## No traces left behind:

- No alerts
- No recovery (unless noticed quickly, via administrators)

## Happens even to the experts:

- [https://bugs.eclipse.org/bugs/show\\_bug.cgi?id=361707](https://bugs.eclipse.org/bugs/show_bug.cgi?id=361707)
- Eclipse Foundation lost almost all their branches and tags accidentally, garbage collection pruned unreferenced commits, administrators had to ask developers for a recent local backup
- <http://jenkins-ci.org/content/summary-report-git-repository-disruption-incident-nov-10th>
- 186 repositories from Jenkins Open Source Project got reset on GitHub



The screenshot shows the Eclipse Bugzilla interface for bug 361707. The title is "I have broken the platform-ui git repository". The bug is marked as "RESOLVED FIXED". It was reported on 2011-10-21 and modified on 2011-10-27. The product is "Community", component is "Git", and version is "unspecified". The importance is "P3 blocker". The bug is assigned to "Eclipse".

**Bug 361707 - I have broken the platform-ui git repository**

**Status:** RESOLVED  
FIXED

**Reported:** 2011-10-21 16:31 EDT by Brian de Alwis

**Product:** Community

**Component:** Git

**Version:** unspecified

**Platform:** All All

**Modified:** 2011-10-27 12:40 EDT (History)

**CC List:** 17 users ([show](#))

**See Also:**

**Importance:** P3 blocker (vote)

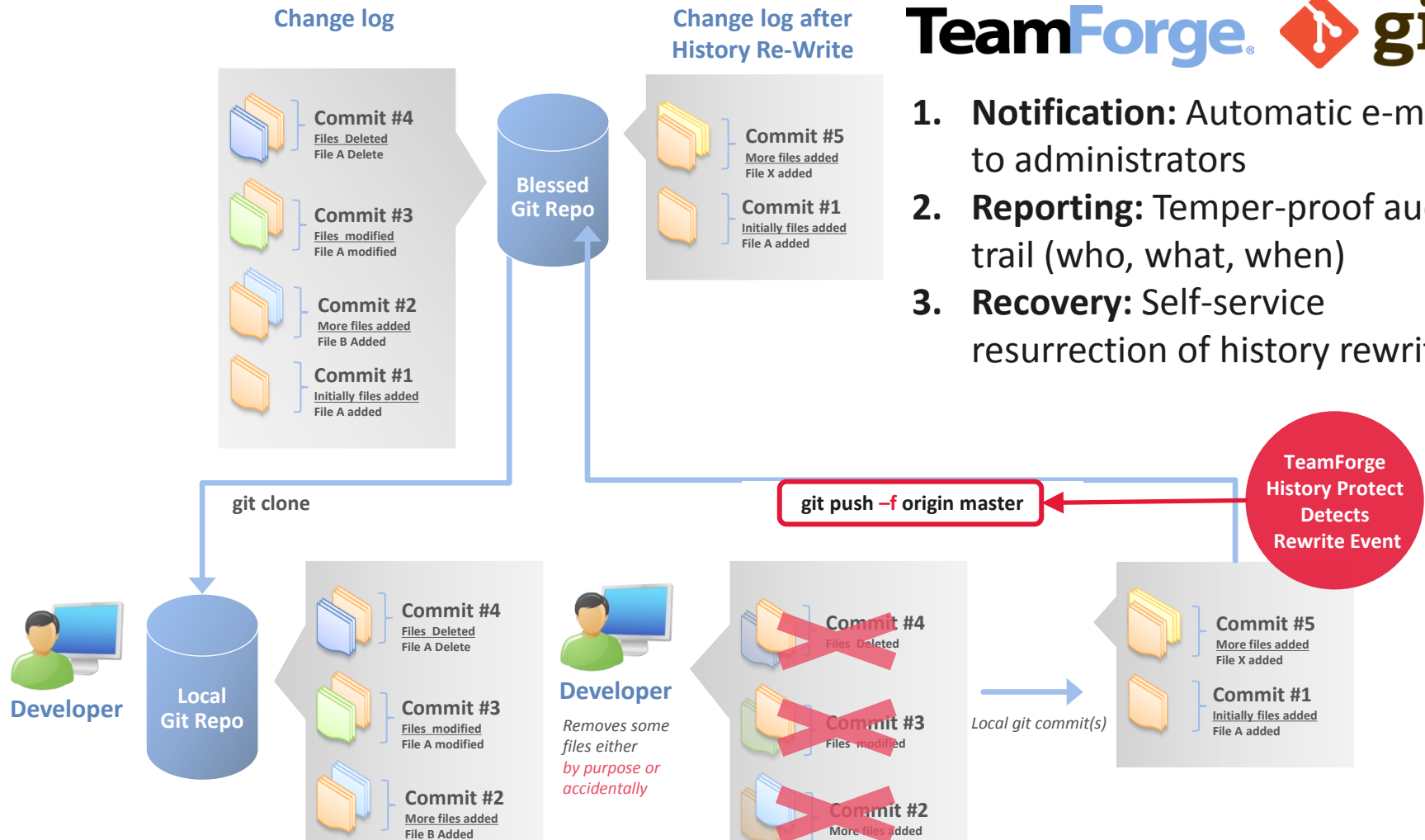
**Target Milestone:** ---

**Assigned To:** Eclipse

# Introducing Git History Protection



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite



TeamForge automatically secures history snapshots on 'blessed' master repository (under "refs/rewrite" or "refs/deleted"), for auditing & recovery.

# TeamForge = Enterprise-Grade Git Mgmt. + ALM



- ✓ Basic SCM features
- ✓ jGit engine
- ✓ Native engine
- ✓ GitWeb



## Gerrit

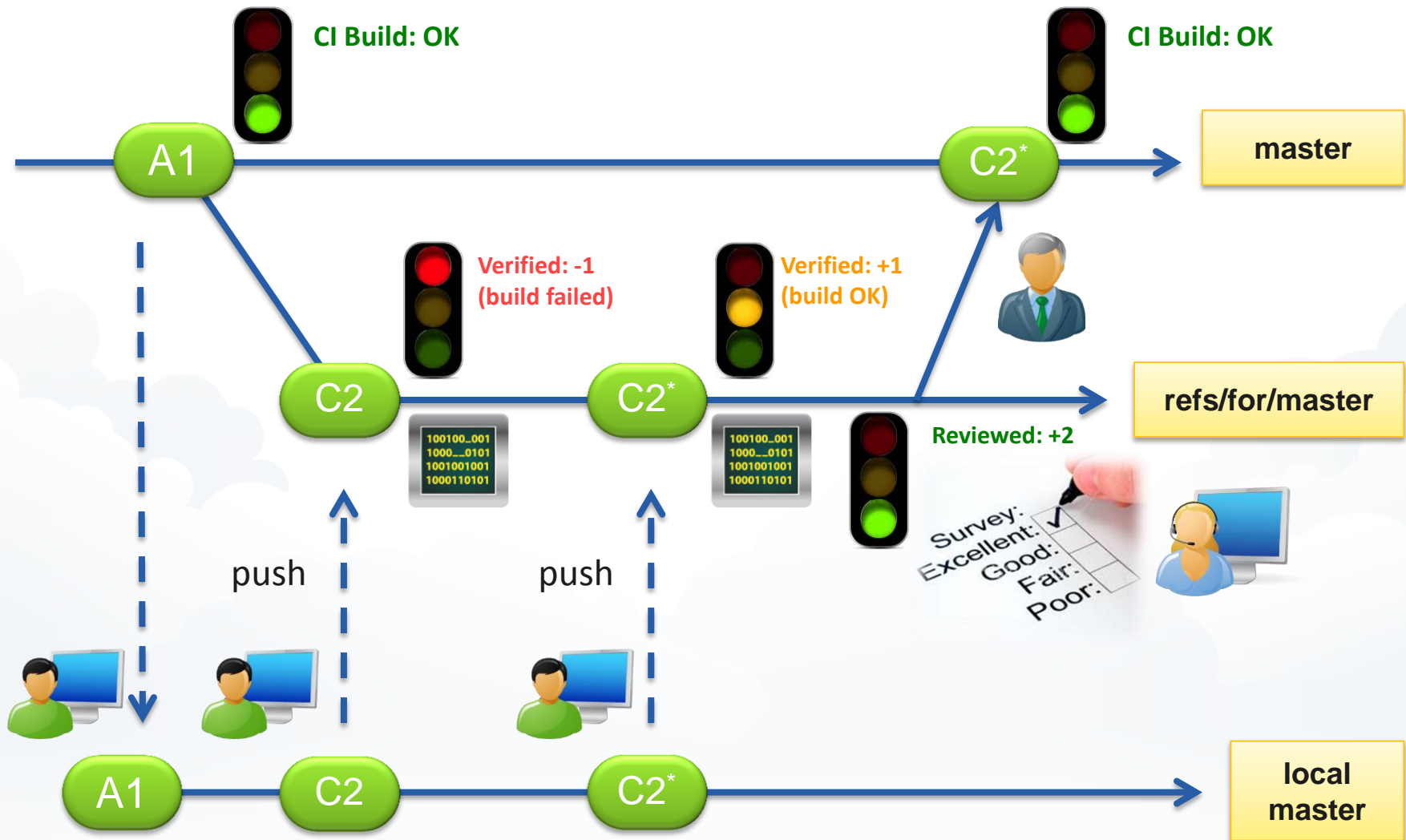
- ✓ Advanced Git security
- ✓ Git Projects organization
- ✓ Code-review
- ✓ Replication

## TeamForge.

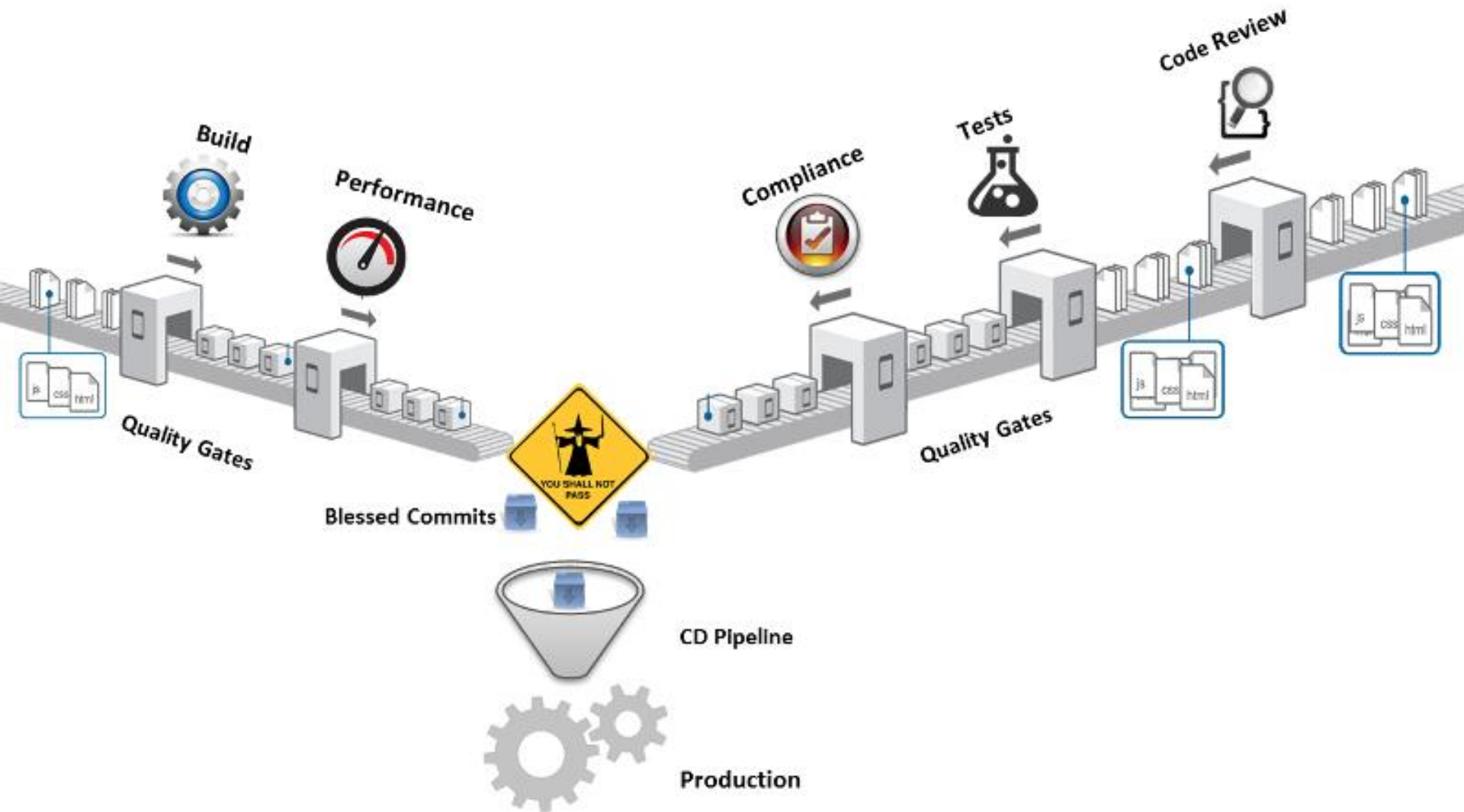
- ✓ 24/7 Support
- ✓ Git/Gerrit Training
- ✓ Standards-Compliance
- ✓ Scalability
- ✓ ALM Integration
- ✓ History Protection
- ✓ Code Quality

With TeamForge and Gerrit, Git is now ready for the enterprise. Realize all the benefits of Git, without compromising governance, security and compliance.

# Tackling half baked review requests: Marrying Gerrit and Jenkins



# Gerrit – Code Quality Gate Wizard – General Concept

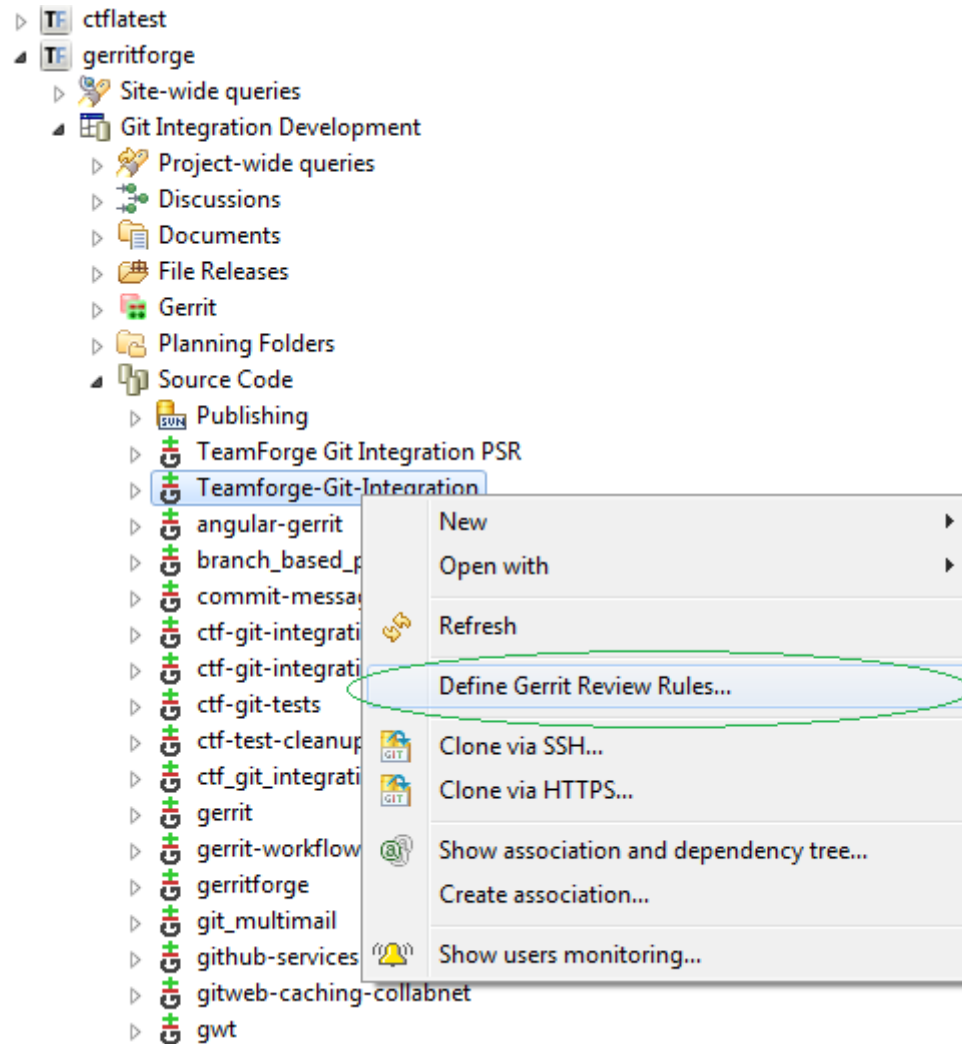


# Code Quality Gate Wizard

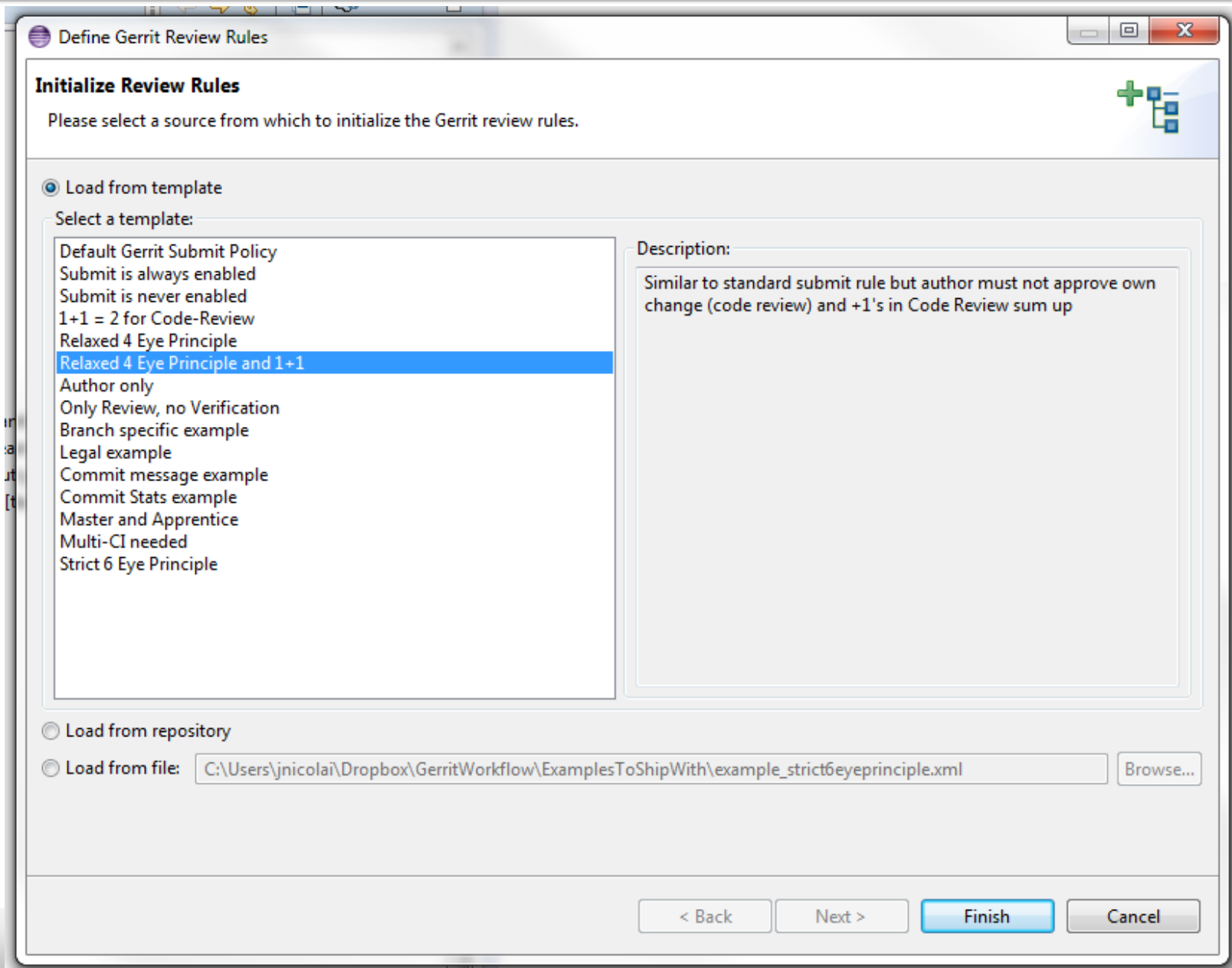
## Code Quality Gate Wizard for Gerrit (Gerrit 2.8+)

- comes with a bunch of predefined policies and lets you graphically design your own quality gates as easy as defining email filter rules.
- Examples:
  - Four-eye peer review
  - Legal has to approve copyright file changes
  - Senior staff has to approve the work of juniors
  - Democratic feature voting
- More details under <http://blogs.collab.net/git>

# Gerrit – Code Quality Gate Wizard – How to launch



# Gerrit – Code Quality Gate Wizard – Predefined templates



# Gerrit – Code Quality Gate Wizard – Built In Test Mode

**Test Submit Rules**

Select or enter a Gerrit change and click Finish to test your submit rules.

Select a change:

- My watched changes
- My open changes
- All open changes
  - 1980 - Throw FakeRepoConfigWriteException [teamforge]
  - 1985 - Return Optional.absent() instead of null [teamforge]
  - 2021 - Demonstrate how to use the engine without gerrit. [gerrit-wo]
  - 2022 - Adjusted tests to work with TeamForge7.2 [teamforge]**
  - 2028 - ChangeDetailFilter tests added. [gerrit-wo]
  - 2029 - CommitDetailFilter test added. [gerrit-wo]

Or enter a change ID:  
164ab67e19715a105ed7969e3eed0337a26ec9e9e

Results:

Rule	Action
Verified-Veto-Blocks-Submit	BLOCK
Code-Review	MAY
Verified	MAY

Finish Cancel

# Gerrit – Code Quality Gate Wizard – Based on existing changes

**Add Change-Based Submit Rule**

**Change-Based Submit Rule**  
Select or enter a Gerrit change upon which to base the new submit rule.

Display name:  
Rule-based-on-change-2041

Action:  
Satisfied: allow Not satisfied:

Select a change:

- My changes
- My watched changes
- My open changes
  - 2031 - Changed copyright notice from 2013 to 2014 [teamforge]
  - 2032 - Fix infinite loop if TF 7.2 license got expired [teamforge]
  - 2041 - Experimental change for experimental branch [teamforge]**
- All open changes

New Query...

Or enter a change ID:  
I4db4a15dbb58f11bb56c05b69be25bc2a964e794

Choose what to copy from the selected change:

- Voting details
- Commit and change details
- Commit stats
- User and group details

Finish Cancel

# Gerrit – Code Quality Gate Wizard – Great Flexibility

The screenshot displays the Gerrit Code Quality Gate Wizard interface. At the top, a table lists existing rules:

Display Name	Satisfied Action	Not Satisfied Action	Voting Conditions	Filters
Code-Review-From-Non-Author-And-...	allow		Code-Review (ignoreAuthorVotes,ignoreCommitterVotes,minSum=2),Verified 1 ( )	
Code-Review-Veto-Blocks-Submit	block		Code-Review -2	
Verified-Veto-Blocks-Submit	block		Verified -1	

Below the table are buttons: "Add Rule Manually", "Add Default Policy Rules", "Add Change-Based Rule", and "Deploy to Gerrit".

Three dialog boxes are overlaid on the interface:

- Change Voting Condition:** Configures voting category (Code-Review), value, filters, voter groups, and vote counts (Min: 2, Max: ). Includes checkboxes for "Ignore author votes" and "Ignore committer votes".
- Change Submit Rule:** Configures display name (Code-Review-From-Non-Author-And-Verified-To-Submit), action (allow), and voting conditions (Code-Review, Verified 1).
- Select Users:** Lists users to add: Balaji Kandasamy (kbalaji), Dariusz Luksza (dluksza), Dharmesh Sheta (sheta), Eryk Szymanski (eszymanski), Jacek Centkowski (jcentkowski), and jayaprakash (jayaprakash).

# TeamForge = Enterprise-Grade Git Mgmt. + ALM



- ✓ Basic SCM features
- ✓ jGit engine
- ✓ Native engine
- ✓ GitWeb



## Gerrit

- ✓ Advanced Git security
- ✓ Git Projects organization
- ✓ Code-review
- ✓ Replication

## TeamForge.

- ✓ 24/7 Support
- ✓ Security
- ✓ Standards-Compliance
- ✓ Scalability
- ✓ ALM Integration
- ✓ History Protection
- ✓ Code Quality

With TeamForge and Gerrit, Git is now ready for the enterprise. Realize all the benefits of Git, without compromising governance, security and compliance.

# Learn More

## 3-Minute Video on History Protection

<http://www.collab.net/products/teamforge/git-for-the-enterprise>



## Git Blogs

<http://blogs.collab.net/git>



## Software Downloads (TeamForge, Git + clients)

<http://www.collab.net/git>



# Q&A

# Feature deep dive: Simple RBAC

## Pre defined & user-defined categories

- **Default:** All Gerrit review features are turned off
- **Mandatory Review:** All code changes have to be reviewed
- **Optional Review:** Review feature is turned on but can be bypassed if necessary
- **Custom:** Access rights have to be set manually in Gerrit Web UI and will not be overridden by TeamForge
- **User-defined:** Possibility to define your own access rights controlled by TeamForge

# Git History Rewrite – Example: Accidental Branch Deletion

github | Search or Type a Command | Explore Gist Blog Help | dharmsheta

dharmsheta / Spoon-Knife-1  
forked from octocat/Spoon-Knife

Code | Network | Pull Requests | Graphs | Admin

This repo is for demonstration purposes only. Comments and issues may not be responded to. — Read more

Clone in Windows | ZIP | HTTP | SSH | Git Read-Only

branch: releasebranch | Files | Commits | Branches

Switch branches/tags

Filter branch/tags

Branches | Tags

- master
- releasebranch
- stable

5 commits

latest commit 9c9ac2f048

Prepare a fix in local branch which is to be pushed to *stable* branch on remote.

Use this syntax for PUSH to remote

```
sheta@SHETA-THINK ~/Spoon-Knife-1 (production-quick-fix)
$ echo "fix">>fix.patch && git add fix.patch

sheta@SHETA-THINK ~/Spoon-Knife-1 (production-quick-fix)
$ git commit -m "quick fix for production"
[production-quick-fix c951efd] quick fix for production
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 fix.patch

sheta@SHETA-THINK ~/Spoon-Knife-1 (production-quick-fix)
$ git push origin production-quick-fix:stable
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:dharmsheta/Spoon-Knife-1.git
2358618..c951efd production-quick-fix -> stable
```

Typo in push command...(accidentally added a space)

Results in branch deletion. No trace on Web UI on Git server side (*stable* branch got deleted)

```
sheta@SHETA-THINK ~/Spoon-Knife-1 (production-quick-fix)
$ echo "fix REWORKED">>fix.patch && git add fix.patch

sheta@SHETA-THINK ~/Spoon-Knife-1 (production-quick-fix)
$ git commit -m "quick fix for production reworked"
[production-quick-fix a70264e] quick fix for production reworked
1 files changed, 1 insertions(+), 0 deletions(-)

sheta@SHETA-THINK ~/Spoon-Knife-1 (production-quick-fix)
$ git push origin production-quick-fix :stable
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:dharmsheta/Spoon-Knife-1.git
- [deleted]      stable
* [new branch]  production-quick-fix -> production-quick-fix
```

github | Search or Type a Command | Explore Gist Blog Help | dharmsheta

dharmsheta / Spoon-Knife-1  
forked from octocat/Spoon-Knife

Code | Network | Pull Requests | Graphs | Admin

This repo is for demonstration purposes only. Comments and issues may or may not be responded to. — Read more

Clone in Windows | ZIP | HTTP | SSH | Git Read-Only | git@github.com:dharmsheta/Spoon-Knife-1.git | Read+Write access

branch: releasebranch | Files | Commits | Branches | Tags | Downloads

Switch branches/tags

Filter branch/tags

Branches | Tags

- production-quick-fix
- releasebranch
- secondbranch

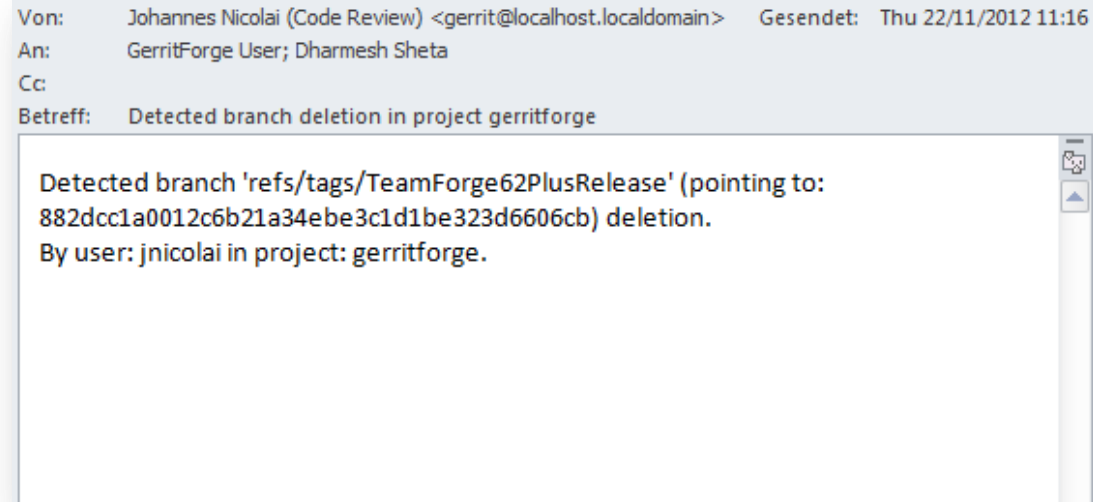
5 commits

latest commit 9c9ac2f048

# Git History Protection – Alerting



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite



Whenever history gets ‘re-written’, an email gets sent out to ‘Gerrit Administrators’ containing details about old HEAD of branch and new HEAD after ‘re-write’.

# Git History Protection – Reporting (Audit Log)



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite

The screenshot shows the TeamForge web interface. At the top, it says "COLLABNET TeamForge" and "Logged in as: Dharmesh Sheta (sheta)". Below the navigation bar, there's a "Site Administration" section with various icons for Projects, Reports, Project Groups, Users, User Groups, Roles, etc. A green callout bubble points to the "System Logs" section, stating: "Audit Log containing all History Rewrite Events is accessible through TeamForge Web UI".

Under "System Tools > System Logs", there's a "System Tools Menu" with options like Server Status, System Logs, Build Information, Ad Hoc Database Query, Broadcast Message, and Custom Event Handlers. The "System Logs" option is selected, showing a list of log files:

- connections.log (0 bytes)
- default-james.log (580 bytes)
- default-search.log (17700388 bytes)
- dnsserver.log (104824 bytes)
- gerrit
- gerrit-synch.audit.log (0 bytes)
- gerrit-synch.pid (6 bytes)
- gerrit-synch.system.log (16391 bytes)
- gerrit.audit.log (6043 bytes)

A red arrow points to the "gerrit.audit.log" file. To the right, a preview of the audit log content is shown, detailing a history rewrite event:

```
refs/heads/Q1-feature-release with old id
4824f7e0b8c347bc711c1cd0805a3390f263eaf was rewritten to
518b1feb5cec61b9a93b648d825e3b65e57a18 |REWRITE BRANCH| [] |
012-11-22 04:25:25,144 | {1353587116415}|sheta|gerritforge|Backup
branch refs/rewrite/20121122041859-Q1-feature-release-
518b1feb5cec61b9a93b648d825e3b65e57a18-sheta pointing to
4824f7e0b8c347bc711c1cd0805a3390f263eaf was deleted|DELETE
BACKUP| [] |
012-11-22 04:25:25,144 | {1353587123912}|sheta|gerritforge|Branch
refs/heads/Q1-feature--release pointing to
518b1feb5cec61b9a93b648d825e3b65e57a18 was deleted.|DELETE
BRANCH| [] |
012-11-22 04:25:55,144 | {1353587129684}|sheta|gerritforge|Backup
branch refs/delete/20121122042523-Q1-feature-release--sheta
pointing to b518b1feb5cec61b9a93b648d825e3b65e57a18 was
deleted|DELETE BACKUP| [] |
012-11-22 05:49:25,144 | {1353592154609}|sheta|gerritforge|Branch
refs/heads/Q1-feature-release with old id
4824f7e0b8c347bc711c1cd0805a3390f263eaf was rewritten to
2202f191b44e593a10bf937dc52d1dd827e388 |REWRITE BRANCH| [] |
```

# Git History Protection – Reporting (Web UI)



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite

The screenshot shows the TeamForge web interface for reporting Git history protection. The page title is "Project gerritforge". The main content area displays a table of history rewrites. The table has columns for Type, Reference Name, User Name, Rewrite date, Original Commit ID, New Commit ID, and Actions. Two rows are visible: one for a REWRITE operation by sheta on 2012 Nov 22 05:49:14, and one for a DELETE operation by jnicolai on 2012 Nov 22 01:43:31. The Actions column for each row contains links for "Delete permanently" and "Resurrect".

	Type	Reference Name	User Name	Rewrite date	Original Commit ID	New Commit ID	Actions
<a href="#">General</a> <a href="#">Branches</a> <a href="#">Access</a> <a href="#">Rewritten history</a>	REWRITE	Q1-feature-release	sheta	2012 Nov 22 05:49:14	a4824f7e0b8c347bc711c1cd0805a3390f263eaf	72202f191b4a4e593a10bf837dc52d1dd827e388	<a href="#">Delete permanently</a> <a href="#">Resurrect</a>
	DELETE	GitScmAdapter	jnicolai	2012 Nov 22 01:43:31	fe35b1d308622cf6686ec77c43a4770edcb615e7		<a href="#">Delete permanently</a> <a href="#">Resurrect</a>

# Git History Protection – Reporting (Command Line)



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite

```
git ls-remote origin
fe35b1d308622cf6686ec77c43a4770edcb615e7 refs/delete/20121122014331-GitScmAdapter--jnicolai
a4824f7e0b8c347bc711c1cd0805a3390f263eaf refs/rewrite/20121122054914-Q1-feature-release-72202f
837dc52d1dd827e388-sheta
```

# Git History Protection – Reporting (Eclipse)



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite

```
Git Repository Exploring - gerics-ctfsynch/src/main/java/com/gerics/c... TT62Personal - VMware Player File
File Edit Source Refactor Navigate Search Project Run Window Help
Git Repositories
  gerrit [tbd] - /root/workspace/gerrit/.git
  gerritforge [Q1-feature-release] - /root/workspace/gerritforge/.git
  Branches
    Local
    Remote Tracking
      origin/backagain 9b5fb94 Revert "For now short-circuiting partial-refresh request to act as full request"
      origin/cobertura 1e3b588 put sonar properties into gerritforge/pom.xml
      origin/ctf-6.1.1 c81440c #120 Exclude removed roles from getUserRoleList()
      origin/ctf-6.2 90f1ec2 Revert "Create a new RPC Session whenever a new Synch cycle is initiated."
      origin/git-scm-adapter 115b790 Merge "Added stuff needed to patch existing GerritForge running with TeamForge 6.
      origin/GitScmAdapter fe35b1d For artifact 110923 - added line sent by Luca: detail.setIncludes(loadIncludes());
      origin/init-better-rpm 3ed485f Update RPM setup
      origin/integration_test_suite ab42a2b indent fixed
      origin/LMIT-7.0 86c4702 Version set to 1.0.7
      origin/master a4824f7 Fix typos and improve mail message in HistoryGuardSender
      origin/mergeathon 9b5fb94 Revert "For now short-circuiting partial-refresh request to act as full request"
      origin/protect-history 3f6ee87 Fixed problem where setHasHistoryProtection was not called.
      origin/Q1-feature-release 72202f1 Change default diff size limit to 25k
      origin/refs/delete/20121122014329-init-better-rpm--jnicolai 3ed485f Update RPM setup
      origin/refs/delete/20121122014330-git-scm-adapter--jnicolai 115b790 Merge "Added stuff needed to patch existing
      origin/refs/delete/20121122014331-GitScmAdapter--jnicolai fe35b1d For artifact 110923 - added line sent by Luca: d
      origin/refs/delete/20121122014332-cobertura--jnicolai 1e3b588 put sonar properties into gerritforge/pom.xml
      origin/refs/delete/20121122014333-backagain--jnicolai 9b5fb94 Revert "For now short-circuiting partial-refresh reque
      origin/refs/delete/20121122014333-sudo--jnicolai ca4c1d7 make use of the new sudo call in ScmListener
      origin/refs/delete/20121122021621-refs/tags/TeamForge62PlusRelease--jnicolai 332fa92 Artifact130337:Fixed NPE w
      origin/refs/rewrite/20121122054914-Q1-feature-release-72202f191b4a4e593a10bf837dc52d1dd827e388-sheta a482
```

# Git History Protection – Recovery (Web UI)



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite

The screenshot shows the TeamForge web interface for the 'Project gerritforge'. It displays a table of history rewrites with columns for Type, Reference Name, User Name, Rewrite date, Original Commit ID, and New Commit ID. A dialog box titled 'Resurrect branch' is open, showing details for a rewrite of the 'Q1-feature-release' branch by user 'sheta' on 2012 Nov 22 05:49:14. The dialog includes a text input field with the value 'restored\_Q1-feature-releas' and 'OK' and 'Cancel' buttons.

Type	Reference Name	User Name	Rewrite date	Original Commit ID	New Commit ID	Actions
REWRITE	Q1-feature-release	sheta	2012 Nov 22 05:49:14	a4824f7e0b8c347bc711c1cd0805a3390f263eaf	72202f191b4a4e593a10bf837dc52d1dd827e388	<a href="#">Delete permanently</a> <a href="#">Resurrect</a>
DELETE	GitScmAdapter	jnicolai	2012 Nov 22 01:43:31	fe35b1d308622cf6686ec77c43a4770edcb615e7		<a href="#">Delete permanently</a> <a href="#">Resurrect</a>

# Git History Protection – Recovery (Command Line)



1. **Notification:** Automatic e-mail to administrators
2. **Reporting:** Temper-proof audit-trail (who, what, when)
3. **Recovery:** Self-service resurrection of history rewrite

Alternatively users having permissions to create a new branch can restore history by using their git client

## Project demogitrepo

General	Branch Name	Revision	
Branches	HEAD	master	<a href="#">(gitweb)</a>
Access	master	03412aed0c60a3e4d480ac3d135369431645ab25	<a href="#">(gitweb)</a>
Rewritten history	<input type="checkbox"/> rescued_after_rewrite	1814a1b0a4f1351db62a5b5fd74ceff87c3c2076	<a href="#">(gitweb)</a>
	Delete		



**Resurrected** branch via command line is now available for all user having atleast read access(view only) for this Git repository

Run this command on git command line

```
1 $ git ls-remote origin
03412aed0c60a3e4d480ac3d135369431645ab25 HEAD
03412aed0c60a3e4d480ac3d135369431645ab25 refs/heads/master
2 1814a1b0a4f1351db62a5b5fd74ceff87c3c2076 refs/rewrite/20121102215850-master-03412aed0c60a3e4d480ac3d135369431645ab25-david
```

Copy HEAD SHA1 of branch created after history 're-write'

Fetch SHA1 of rewritten history from server

```
3 $ git fetch origin refs/rewrite/20121102215850-master-03412aed0c60a3e4d480ac3d135369431645ab25-david
```

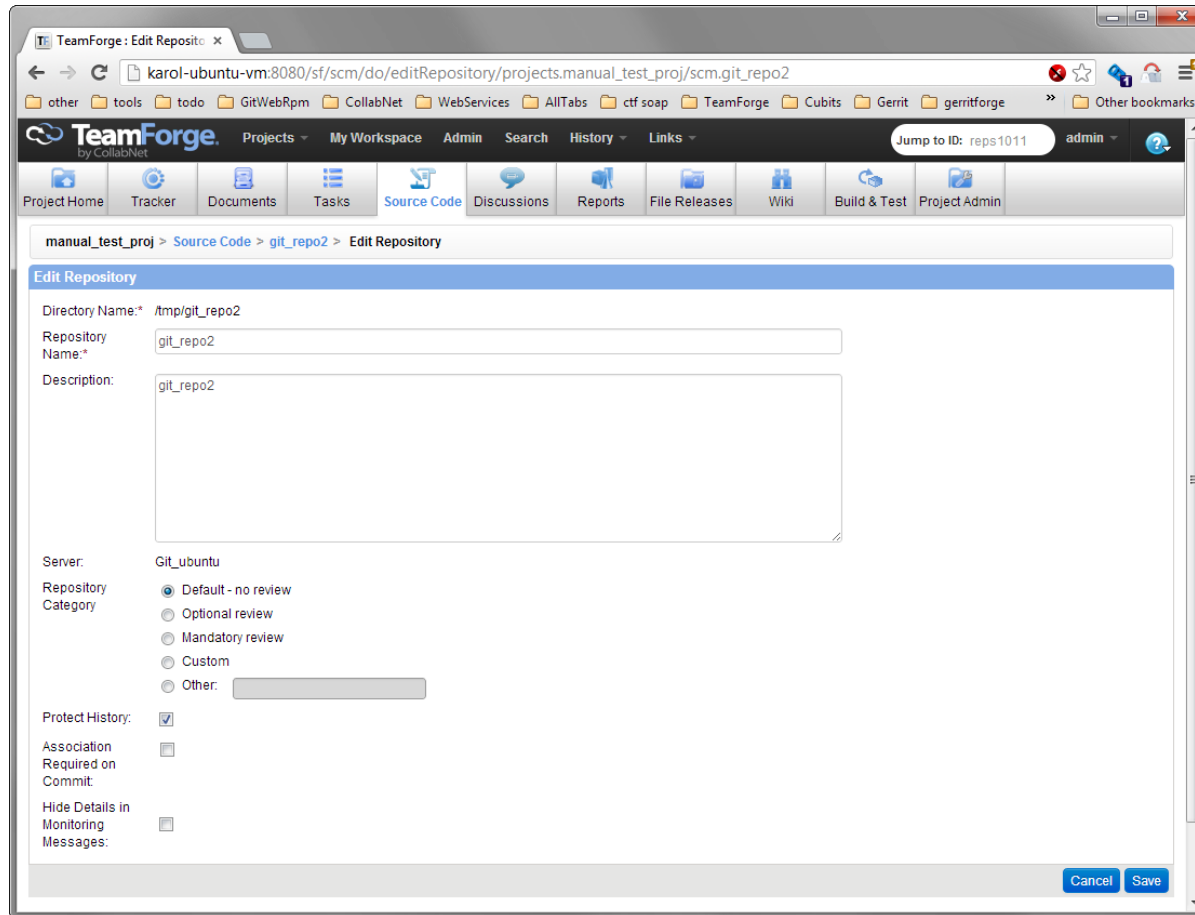
Use copied SHA1 to create new local branch in Git

```
4 $ git checkout -b rescued_from_commandline 1814a1b0a4f1351db62a5b5fd74ceff87c3c2076
Switched to a new branch 'rescued_from_commandline'
```

Push this local branch to remote 'blessed repository'

```
5 $ git push origin rescued_from_commandline:rescued_after_rewrite
Total 0 (delta 0), reused 0 (delta 0)
To ssh://david@tf62personal.potsdam.collab.net:29418/demogitrepo
* [new branch] rescued_from_commandline -> rescued_after_rewrite
```

# Git History Protection – Administration



Gerrit Config-Option allows you to protect all Git repositories hosted by TeamForge: Not even TeamForge Site Admins can override

```
[gerrit]
basePath = /gitroot
forceHistoryProtection = true
[database]
```

# Git History Rewrite – Why Blocking is NOT the Answer

## Legitimate use cases

### – Developers

- Deleting accidentally committed file(s)/change(s)
- Change appearance of commits
  - squashing multiple commits into unified single commit
  - change order of commits

### – Build / Release managers / Developers

- Removing Copyrights/ Intellectual Property(IP) related resources from code base
- Removing large file(s)
- Removing feature branch(es) created temporarily and already merged

## Not (so) Legitimate use cases

### – Developers / Build / Release Managers

- Remove somebody else's changes without leaving any trace
- Pretending someone else's change as own (forgery)
- Accidentally removing branches



# Git History Rewrite – Why Blocking is NOT the Answer

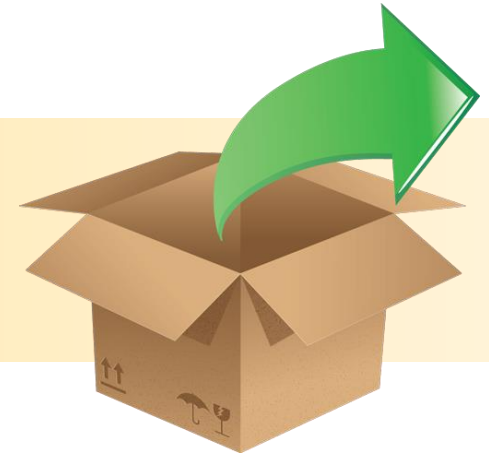
## Legitimate use cases

### – Developers

- Deleting accidentally committed file(s)/change(s)
- Change appearance of commits
  - squashing multiple commits into unified single commit
  - change order of commits

### – Build / Release managers / Developers

- Removing Copyrights/ Intellectual Property(IP) related resources from code base
- Removing large file(s)
- Removing feature branch(es) created temporarily and already merged



## Not (so) Legitimate use cases

### – Developers / Build / Release Managers

- Remove somebody else's changes without leaving any trace
- Pretending someone else's change as own (forgery)
- Accidentally removing branches

# Git History Rewrite – Why Blocking is NOT the Answer

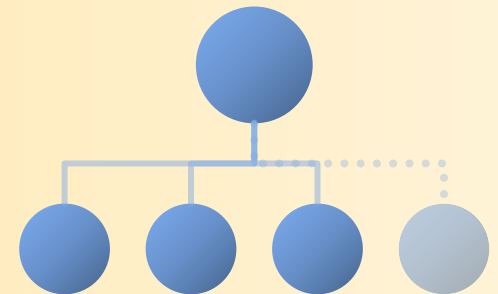
## Legitimate use cases

### – Developers

- Deleting accidentally committed file(s)/change(s)
- Change appearance of commits
  - squashing multiple commits into unified single commit
  - change order of commits

### – Build / Release managers / Developers

- Removing Copyrights/ Intellectual Property(IP) related resources from code base
- Removing large file(s)
- Removing feature branch(es) created temporarily and already merged



## Not (so) Legitimate use cases

### – Developers / Build / Release Managers

- Remove somebody else's changes without leaving any trace
- Pretending someone else's change as own (forgery)
- Accidentally removing branches

# Git History Protection - Summary

- TeamForge + Gerrit is a powerful foundation for Enterprise Git
- Git History Rewrite covers branch deletions and forced pushes, two very powerful, often needed but quite dangerous operations which might lead to data loss and tampering
- TeamForge 6.2 with Gerrit introduced History Protection
- History protection ensures that potentially unnoticed events, such as remote branch deletions and forced pushes, are now detected and fully recoverable, any modifications to Git code and repositories are securely recorded with tamper-proof audit logs
- Recovery is possible from a Web interface at the push of a button, and using an ordinary Git client, IT operations teams don't have to be involved



# Feature deep dive: Notifications

## Motivation:

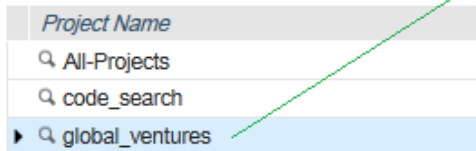
Bring the git push notifications mechanism from the gitmultimail script to TeamForge

Step 1: Navigate to Projects->List menu



## Projects

Filter



Powered by  CollabNet.

Step 2: Select source code repository and then navigate to Notification menu



## Project global\_ventures



Powered by  CollabNet.

# Feature deep dive: Notifications

**TeamForge** by CollabNet

Projects ▾ My Workspace Admin History ▾ More ▾

Project Home Trackers Source Code Build & Test File Releases Documents Wiki Discussions Project Admin

Brokerage System (Sample) / Discussion / Development / [global\_ventures] annotated tag v1.0 created (now e844e03) / List of Posts

Forum Topic - [global\_ventures] annotated tag v1.0 created (now e844e03): (3 Items) 🚩

View: 25 Posts ▾ as Nested - by Reply ▾ Update

**TeamForge Administrator**  
02/28/2014 10:20 AM  
post1012

**[global\_ventures] annotated tag v1.0 created (now e844e03)**  
This is an automated email from the git hooks/post-receive script.

product\_developer pushed a change to annotated tag v1.0 in repository global\_ventures.

at e844e03 (tag)  
tagging fa3904125c990bd7397ed63c2dc514c7db4a9282 (commit)  
tagged by Nancy  
on Fri Feb 28 10:20:10 2014 +0530

- Log -----  
World Class Features A and B Released  
-----

This annotated tag includes the following new commits:

new 3e4ce82 World Class Feature A  
new fa39041 World Class Feature B

The 2 revisions listed above as "new" are entirely new to this repository. The revisions listed as "adds" were already present in the repository and have only been added to this reference.

--  
To stop receiving notification emails like this one, please contact the administrator of this repository.

**TeamForge Administrator** [global\_ventures] 01/02: World Class Feature A

**TeamForge Administrator** [global\_ventures] 02/02: World Class Feature B

*Annotations:*  
- Topic header: points to the forum topic breadcrumb.  
- Summary of changes introduced with this RefUpdate: points to the commit information block.  
- Subsequent commits details: points to the list of new commits.

# Appendix – History Rewrite, versus Git Reflog

	Git relog	TeamForge Git Integration with 'History Protect'
<b>Accessibility</b>	Requires direct access to file system on server where 'blessed Git repository' is hosted which is very unlikely in huge organizations and will keep the server administrators busy	'Self –Service' approach. Users with appropriate permissions in TeamForge can find out/resurrect deleted/rewritten branches by themselves, decreasing work load of server administrators. Gerrit Administrators can also permanently delete selected branches/tags.
<b>Signal-to- Noise Ratio</b>	<p>reflog records &lt;all&gt; changes in the repository</p> <ul style="list-style-type: none"> <li>• Any push (also ordinary fast forward)</li> <li>• Any merge</li> <li>• Any Branch creation/deletion</li> <li>• Any Tag creation / deletion</li> </ul> <p>Finding out about history rewrites/deleted branches is like searching for a needle in a haystack</p>	<p>History Protect only reports</p> <ul style="list-style-type: none"> <li>• Deleted branches/tags</li> <li>• History rewrites (non fast forward pushes)</li> </ul>
<b>Notification</b>	No notification	<ul style="list-style-type: none"> <li>• Email to Gerrit Administrators</li> <li>• Audit log entry whenever branch/tags gets               <ul style="list-style-type: none"> <li>• Deleted</li> <li>• Re-written (non fast forward)</li> <li>• Resurrected</li> <li>• Permanently deleted</li> </ul> </li> </ul>
<b>Ease of use</b>	<ul style="list-style-type: none"> <li>• Only manually configurable by administrator having file system access</li> <li>• To be configured for each and every repository</li> <li>• Restoring requires running git commands on server</li> </ul>	<ul style="list-style-type: none"> <li>• Pre-configured in TeamForge/Gerrit</li> <li>• Configurable for all repositories by setting site-wide config option or on a per repository basis</li> <li>• User with appropriate permission can restore history using Gerrit WebUI / Git client</li> </ul>
<b>Protection against object pruning/reflog expiration</b>	RefLog expiration and gc pruning settings have to be manually configured by server administrator. Only possibility to not lose commits no longer referenced in a branch is to set both values to <unlimited> which will consume huge amounts of disk space, slows down garbage collection and does not allow to permanently delete specific commits (all or nothing).	Preserved commits will never be pruned by garbage collection unless permanently removed using Gerrit Web UI. No need to keep a large ref log. Garbage collection will run faster since all commits are still referenced in the repo.