Outline for Git
CAT&DOG Git study 2013
Day 1

Eon Jeong

March 31, 2013

Contents

Contents

About revision control

Common concepts

About Git

Websites

Assignment

Section 1

About revision control

VCS

*V*ersion *c*ontrol *s*ystem

- ▶ Or alternatively, *revision control* or *source code control*

VCS

*V*ersion *c*ontrol *s*ystem

▶ Or alternatively, *revision control* or *source code control*

▶ Similar to what you may have experienced as:

    History features found in wikis or documentations, e.g. Wikipedia

    Undo/redo features in most text editors (including even Windows notepad), word processor, spreadsheets, etc.

Why do we need a VCS?

What if when you are writing a document or a
software,

- you are hacked by super mighty 127.0.0.1
- then lose everything. :P

Why do we need a VCS?

What if when you are writing a document or a
software,

- you are hacked by super mighty 127.0.0.1
- then lose everything. :P
- or just made a huge wrong modification;
- how about writing something together?

Note that any data or transmission can be very
volatile with our electric computers.

What VCS does

Assume you are translating an article about something the readers aren't aware of.

What VCS does

Assume you are translating an article about something the readers aren't aware of.

- As you have more translations day by day,

What VCS does

Assume you are translating an article about
something the readers aren't aware of.

- As you have more translations day by day,
- some sections should be reviewed later;

What VCS does

Assume you are translating an article about
something the readers aren't aware of.

► As you have more translations day by day,

► some sections should be reviewed later;

► some need to be reorganized totally;

What VCS does

Assume you are translating an article about something the readers aren't aware of.

- ▶ As you have more translations day by day,
- ▶ some sections should be reviewed later;
- ▶ some need to be reorganized totally;
- ▶ some are candidates, now unused but to be merged;

What VCS does

Assume you are translating an article about something the readers aren't aware of.

- As you have more translations day by day,
- some sections should be reviewed later;
- some need to be reorganized totally;
- some are candidates, now unused but to be merged;
- or tomorrow you may want to revert the changes you made today.

What VCS does

Assume you are translating an article about something the readers aren't aware of.

- As you have more translations day by day,
- some sections should be reviewed later;
- some need to be reorganized totally;
- some are candidates, now unused but to be merged;
- or tomorrow you may want to revert the changes you made today.

Likewise, development is a hard thing. IDEs such as Visual Studio or Eclipse have functions like undo/redo. **It ain't enough though.**

Section 2

Common concepts

Repository

A typical *version control **repository*** is whatever includes a historical set of objects (files, folders and references onto them), managed by a VCS.

Repository

A typical *version control* **repository** is whatever includes a historical set of objects (files, folders and references onto them), managed by a VCS.

- Hence, **VCS** is a kind of DBMS around version control operations.

Repository

A typical *version control* **repository** is whatever includes a historical set of objects (files, folders and references onto them), managed by a VCS.

- Hence, **VCS** is a kind of DBMS around version control operations. (Thus we can assume a good one should follow *ACID*.)

Repository

A typical *version control* **repository** is whatever includes a historical set of objects (files, folders and references onto them), managed by a VCS.

- ▶ Hence, **VCS** is a kind of DBMS around version control operations. (Thus we can assume a good one should follow *ACID*.)
- ▶ **Local** means the repository or its copy is on the workspace.

Repository

A typical *version control* **repository** is whatever includes a historical set of objects (files, folders and references onto them), managed by a VCS.

- Hence, **VCS** is a kind of DBMS around version control operations. (Thus we can assume a good one should follow *ACID*.)
- **Local** means the repository or its copy is on the workspace.
- **Server** or **remote** means the repository can be accessed via network.

Repository

A typical *version control **repository*** is whatever includes a historical set of objects (files, folders and references onto them), managed by a VCS.

- Hence, **VCS** is a kind of DBMS around version control operations. (Thus we can assume a good one should follow *ACID*.)
- **Local** means the repository or its copy is on the workspace.
- **Server** or **remote** means the repository can be accessed via network.
- **Centralized** means every client should work on the same repository on a server.

Repository

A typical *version control **repository*** is whatever
includes a historical set of objects (files, folders and
references onto them), managed by a VCS.

- Hence, **VCS** is a kind of DBMS around version
  control operations. (Thus we can assume a
  good one should follow *ACID*.)
- **Local** means the repository or its copy is on the
  workspace.
- **Server** or **remote** means the repository can be
  accessed via network.
- **Centralized** means every client should work on
  the same repository on a server.
- **Distributed** means there are local repository
  which may have some references to other
  repos.

Revision

A **revision** is the status of contents in the repository at a specific timepoint.

Revision

A **revision** is the status of contents in the repository at a specific timepoint.

- **Commit** is task making a new revision, or the revision itself.

Revision

A **revision** is the status of contents in the repository at a specific timepoint.

- ▶ **Commit** is task making a new revision, or the revision itself.
- ▶ **Checkout** is task switching the repository to one of its revisions.

Revision

A **revision** is the status of contents in the repository at a specific timepoint.

- ▶ **Commit** is task making a new revision, or the revision itself.
- ▶ **Checkout** is task switching the repository to one of its revisions.
- ▶ **Difference**, or just simply **diff**, is *changeset* between two or more revisions.

Revision

A **revision** is the status of contents in the repository at a specific timepoint.

- ▶ **Commit** is task making a new revision, or the revision itself.
- ▶ **Checkout** is task switching the repository to one of its revisions.
- ▶ **Difference**, or just simply **diff**, is *changeset* between two or more revisions.
- ▶ **Head** is a reference pointing the last committed (or the last checked out) revision.

Source tree

Source tree, or *stream*, is the relationship of order among whole revisions of a repository.

Source tree

Source tree, or *stream*, is the relationship of order among whole revisions of a repository.

- **Branch**, or *codeline*, is a named timeline of revisions with start and end, each is committed on the base of the previous revision of the branch.

Source tree

Source tree, or *stream*, is the relationship of order among whole revisions of a repository.

- **Branch**, or *codeline*, is a named timeline of revisions with start and end, each is committed on the base of the previous revision of the branch.
- **Trunk**, **master branch** or the *mainline* means a unique unnamed branch of the repository. It commonly contains revs from approved diffs.

Source tree

Source tree, or *stream*, is the relationship of order among whole revisions of a repository.

- ► **Branch**, or *codeline*, is a named timeline of revisions with start and end, each is committed on the base of the previous revision of the branch.
- ► **Trunk**, **master branch** or the *mainline* means a unique unnamed branch of the repository. It commonly contains revs from approved diffs.
- ► **Merging** two or more branches cause them share one ongoing revision.

Source tree

Source tree, or *stream*, is the relationship of order among whole revisions of a repository.

- ▶ **Branch**, or *codeline*, is a named timeline of revisions with start and end, each is committed on the base of the previous revision of the branch.
- ▶ **Trunk**, **master branch** or the *mainline* means a unique unnamed branch of the repository. It commonly contains revs from approved diffs.
- ▶ **Merging** two or more branches cause them share one ongoing revision.
- ▶ **Tag**, or *label*, is a meaningful snapshot with name, usually a version number.

Source tree

Source tree, or *stream*, is the relationship of order among whole revisions of a repository.

- **Branch**, or *codeline*, is a named timeline of revisions with start and end, each is committed on the base of the previous revision of the branch.
- **Trunk**, **master branch** or the *mainline* means a unique unnamed branch of the repository. It commonly contains revs from approved diffs.
- **Merging** two or more branches cause them share one ongoing revision.
- **Tag**, or *label*, is a meaningful snapshot with name, usually a version number.
- **Fork** means copying a repo to maintain it independently. The word represents relation among related branches, each having its tail on another branch and its head far forward.

A typical Git 6repository

A typical Git 6repository

1. **revision**, diff, head

A typical Git 6repository

Git Study Day 1
Outline

Eon Jeong

Contents

About revision
control

Common
concepts

About Git

Websites

Assignment

1. **revision**, diff, head
2. **source tree**, branch, merge

A typical Git 6repository

Git Study Day 1
Outline

Eon Jeong

Contents

About revision
control

Common
concepts

About Git

Websites

Assignment

1. **revision**, diff, head
2. **source tree**, branch, merge
3. ???
4. PROFIT!

Section 3

About Git

Git

Git Study Day 1
Outline

Eon Jeong

Contents

About revision
control

Common
concepts

About Git

Websites

Assignment

- git-scm.com
- Linus Torvalds, 2005 ˜
- written in C, Perl and shell scripts

How to use Git

- *As-is (I suggest)*
    - `git <command>` in terminal
    - many direct features associated with software configuration management (SCM) issues
    - Git is fast!
- Or with GUI (front-ends)
    - Git-GUI
    - SourceTree
    - SmartGitHg
    - GitHub for Windows (or GitHub for Mac)
    - TortoiseGit

Section 4

Websites

GitHub

- github.com
- 2008 ˜
- free public Git repository for any size of teams
- fast fork, pull request, tree graph for forks, issue tracker, wiki
- star, watch
- 'flavored' markdown
- template repository (called Gist)
- web page hosting
- similar: Gitorious

Bitbucket

Git Study Day 1
Outline

Eon Jeong

Contents

About revision
control

Common
concepts

About Git

Websites

Assignment

- <u>bitbucket.org</u>
- 2008 ˜
- free public/private Git/Mercurial repository for small teams
- fast fork, pull request, tree graph, issue tracker, wiki
- star, fast diff
- similar: Launchpad

Others

John Mang (2010˜) supports only SVN, not Git.

Google Code (2005˜) supports Git as well as CVS and Subversion.

Sourceforge (1999˜) supports Git, Mercurial, Bazaar, Subversion, CVS.

KLDP.net (2002˜) supports same VCSs as Google code; Naver Developer Center uses the same engine, nForge.

First week's assignment

1. **Install Git**
   - ▶ Get *MSysGit* to use Git in command line, Windows.
   - ▶ If you want a good GUI, Don't use *TortoiseGit*.
2. **Sign up GitHub**
   - ▶ This is necessary for the upcoming second assignment.