Git

Git is an extremely powerful and flexible revision control system, and using it effectively requires adherence to conventions.

Usage Policy

Here is a proposal; comments and adjustments are welcome.

- A distinct tree / a repository - is maintained for each independent (software, paper, …) project.
- For each project, the mainline (master) tree is hosted here at the iis uibk servers.
- Contributors develop locally and maintain their own trees, and push (or request to pull) only generally-useful, tried-and-proven patch sets into the main tree.
- Where appropriate, multiple external developers can exchange patches among each others before committing to the mainline.
- Branches are used to maintain multiple releases simultaneously while developing new features in the trunk. As long as we will not generally have formal releases, there will be little or no need for branches.

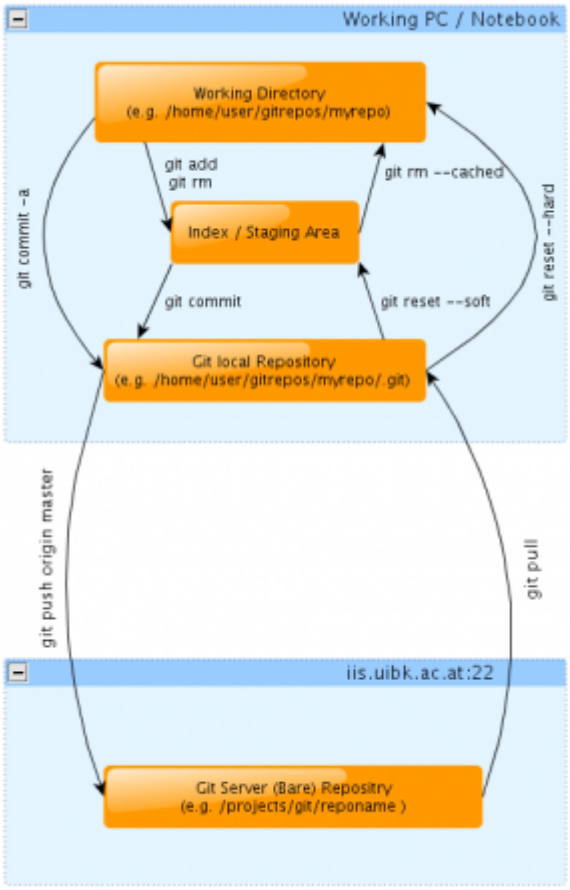
Basic Workflow

Retrieve an IIS Git tree for you to work on:

```
git clone ssh://iis.uibk.ac.at:2222/projects/git/PROJECTNAME
```

Prepend the hostname with USERNAME@ if your Git username does not match your local username.

Now make your edits. To see the status of your files with respect to your repository, do `git status`.



For each reasonable unit of changes, tell Git that you want to keep it, and then commit it to your local tree:

```
git add FILENAME
git commit -m "brief documentation of your changes"
```

Each self-contained, fully-functional set of changes that you want to make public should be pushed upstream to the IIS master tree:

```
git push
```

At this point you are free to delete your local Git tree.

In the meantime, you can retrieve updates from the IIS master tree with `git pull`.

For more information, see the Git Reference and Everyday Git.

More Advanced Hints

Creating a New Git Project

To create a shared Git repository, initialize it on the lab server. Then, clone and populate it.

To create your personal Git repository for your own, local use only:

```
cd PROJECTROOT
git init --shared
git add .
git commit -m "initial import"
```

textmode tool for git

a nice (commandline) tool for viewing commits, logs, diffs and other changes is tig. to install tig on a debian based machine

```
aptitude install tig
```

change into the git working directory and start

```
tig
```

Patches

Generate Patches and use Peer-to-peer patch exchanges [1)]

```
git format-patch --cover-letter -o some-dir
d8a285c8f83f728be2d056e6d4b0909972789d51..9202ec15da36ca060722c363575e0e390d
85fb71
# this is equivalent to, this is the short form
git format-patch --cover-letter -n -o some-dir d8a28..9202e
```

Where d8a28 was the last commit before you started hacking and 9202e is the current head, meaning the commit ID of your latest commit.

For renaming files add “-M” to the git-format-patch arguments then patches wont create removals and adds for a simple rename.

Sending Patches:

```
git send-email --no-chain-reply-to --from "My Name <my.name@uibk.ac.at>" --
to recipient@domain some-directory/
```

[1)]
example taken from http://linuxwireless.org/en/developers/Documentation/git-guide

From:
https://iis.uibk.ac.at/ - **IIS**

Permanent link:
https://iis.uibk.ac.at/collab/git?rev=1302510921

Last update: **2018/09/03 14:57**