# 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.

## Obtaining a Git Project

### Creating a New Git Project

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

Then, ask the sysadmin to put your tree onto the lab server.

If your tree will not be accessed by anybody else, there is no need to talk to the sysadmin or to specifiy `--shared`.

### Retrieving an Existing Git Project

To get the source code initially use `git clone`:

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

e.g.

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

# Standard Usage

**Develop locally, within your own trees, and push (or request to pull) only generally-useful, tried-and-proven patch sets into the main tree.**

## Example usage

for local commits use just something like

```
git commit -a -m "Commit Message"
```

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/
```

# For Releases

push the local commits to the server (master tree) - as mentioned above only generally-useful, tried-and-proven code.

```
git push origin master
```

to get updates from the server

```
git pull origin master
```

# References

howto think like git: http://gitref.org/

this may also be a good starting point:
http://www.kernel.org/pub/software/scm/git/docs/everyday.html

1)

example taken from http://linuxwireless.org/en/developers/Documentation/git-guide