# An Informal Guide to Using Git*
# with the linux-pnfs git tree

Benny Halevy
bhalevy@panasas.com

Connectathon
May 14th, 2008

* Information is based on git version 1.5.3.3

# Getting help

- The `git` command is a swiss army knife
- `git <command>` is equivalent to `git-command`
- `man git` provides an overview of all commands.
- `man git-command` (or `git-command --help`) provides command specific information.

- Note: Expect change. git keeps evolving.

# Git URLs

- A git url specifies the location of a git tree.
- For example:
  - git://linux-nfs.org/~bhalevy/linux-pnfs.git
  - ssh://myserver.com/~me/git-dev/my-tree
  - /home/me/git-dev/my-tree

# Cloning

```
git clone git://linux-nfs.org/~bhalevy/linux-pnfs.git
```

- Cloning a remote tree creates a local copy of it in a sub-directory.
- By default, the remote tree is called "`origin`"
- All remote (a.k.a. tracking) branches are available via `origin/<branch>`

# Adding a remote

`git remote add <name> <url>`

- Adds a remote tree
- Allows you to refer to commits in remote branches `<name>/<branch>`
- Definition lives in `.git/config`

# Branching

`git checkout -b <local> <origin>/<remote>`

- Creates a branch called `<local>` pointing at `<origin>/<remote>` and checks it out.
- “`git branch <local> <origin>/<remote>`” just creates the branch.

# Updating a Remote Tree

`git remote update`

- Fetches all remote trees

`git fetch origin`

- Fetches just one remote tree

- Either way only your copy of the remote tree is affected.
- Any branches you created off of it are intact.

# Resetting to the updated remote

```
git checkout <local>
git reset –hard <origin>/<remote>
```

- Resets your `<local>` branch to point at `<origin>/<remote>`
- "`git reset`" can be used to reset to anywhere else in the tree.
- Note: Any changes you committed to `<local>` will be lost.

# Making changes

- Unlike other source control systems these is no central coordination.
- No need to “checkout” particular files you intend to work on.  Just throw your favorite editor at them.

```
git diff
```

- Shows you your pending changes.

# Committing Changes

`git add`

- Adds changes to the “index” for later commit.

`git diff –cached`

- Will show your cached changes

`git commit`

- Commits the cached changes to the tree.

`git commit -a`

- Adds and commits at the same time.

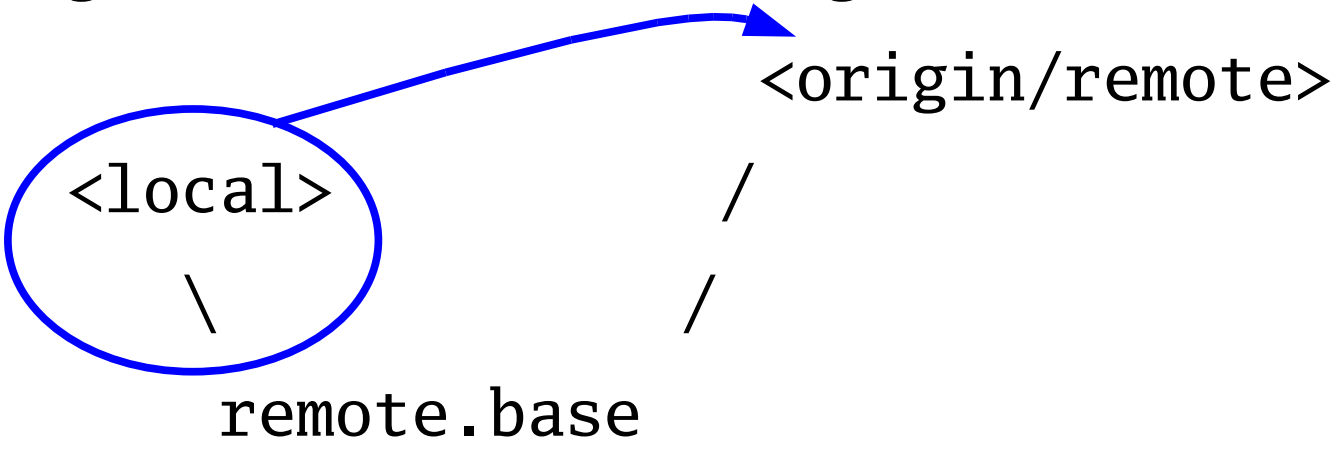
# Using tags

`git tag [-f] <tagname> <ref>`

- Makes a symbolic reference pointing at <ref>
- Useful for, e.g.:
  - "`git diff <tagname> HEAD`", or
  - "`git reset [--hard] <tagname>`", or
  - "`git checkout [-b <branch>] <tagname>`"

# Rebasing your changes

```
git checkout -b <local> <origin/remote>
   git commit...
git tag remote.base <origin/remote>
git fetch origin
git rebase --onto <origin/remote> remote.base <local>
```




- Each commit in the rebased range is “cherry picked” onto the new base and merged.
- Conflicts must be resolved per-commit.

# Questions?