GitCheatSheet

Git cheat sheet

Git setup

If you're a first time Git user, set up your global Git configuration first:

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Enabling color output is highly recommended:

```
git config --global color.ui auto
```

You can also create your own global *.gitignore* file in *$HOME*, and put rules for your editor temp files in there. Enable it like this:

```
git config --global core.excludesfile ~/.gitignore
```

For Git versions below 2.0: By default, *git push* without arguments pushes all local branches to existing branches with the same name on the remote. This is not what most users expect. It is recommended to change the *push.default* setting to *simple* or *upstream*, so only the current branch will be pushed to its upstream branch.

```
git config --global push.default simple
```

ASF committers can set their Apache username, so they don't have to enter it with every commit:

```
git config credential.username <username>
```

Cloning

To clone the repository:

```
git clone https://git-wip-us.apache.org/repos/asf/lucy.git
```

Standard workflow

```
git checkout <branch>
git pull --rebase
# Edit
git commit
git push
```

Working with branches

Create and switch to a local branch:

```
git checkout -b <branch>
```

Push local branch to remote for the first time:

```
git push -u origin <branch>
```

Use the *-u* option to automatically setup the remote tracking configuration.

Checkout a remote tracking branch:

```
git checkout --track origin/<branch>
```

Or with a recent Git version, simply:

```
git checkout <branch>
```

Delete a remote branch:

```
git push origin :<branch>
```

Merging branches and keeping linear history

If you want to merge a branch to *master* and keep linear history, copy the branch to a temporary branch, rebase the temporary branch onto *master*, then merge the rebased branch:

```
git checkout -b tmp <branch>
git rebase master
# Fix possible conflicts and review
git checkout master
git merge tmp
```

Then the temporary branch can be deleted:

```
git branch -d tmp
```

Merging Github pull requests

A simple way to get changes from Github is to run *git fetch* and then create a branch from *FETCH_HEAD*:

```
git fetch https://github.com/<user>/<repo> <branch>
git checkout -b pull-request FETCH_HEAD
```

Now you're on a local branch *pull-request* with all the commits from the pull request. You can review the changes and rebase them onto *master* if you want to keep linear history:

```
git rebase master
```

If everything looks good, merge to *master*:

```
git checkout master
git merge pull-request
```