Git Tutorial

This document is created by Yu Zhao.

This tutorial demonstrates git by stepping you through the process of setting up and initialising a Git repository and the process of using Git to manage a shared project source. By the end of this tutorial you’ll have an understanding of how git works, and know how to apply git to support a collaborative development process.

Git is a flexible tool able to support any number of release/version management processes. The process described in this tutorial is just ONE particular way of using Git. It is not the only way, and it isn’t the best way. Students are encouraged to explore different processes. Useful examples and discussions include:<http://git-scm.com/book/en/Git-Branching-Branching-Workflows>

<http://scottchacon.com/2011/08/31/github-flow.html>, and <http://nvie.com/posts/a-successful-git-branching-model/>

## User Accounts

You’ll need a Bitbucket account.

## Software Setup

You’ll need a standalone Git client. Git client installs can be found here<http://git-scm.com/downloads>.

Optional:

You can configure Git to SSH to transfer data. Setting-up SSH means you won’t need to enter your password each time you want to move data. Setting up SSH for Git is described here:<https://confluence.atlassian.com/display/BITBUCKET/Set+up+SSH+for+Git>

You can set up a few additional properties of git as described here:<http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup>.

You can also set up a diff tool to help you manage merge conflicts. Google ‘git merge tool’ to view possible merge tools and their pros and cons.

# Part One: Setting up a Team Git Repo

Each project within a team requires a shared Git repo (aka repository). This section steps you through the process of:

* Creating a project team. Teams are used by Bitbucket to manage user account access.
* Creating a shared repo for the team. This is shared space on the Bitbucket server.
* Initialising the repo and master branch. This create a repo that can be immediately used by team members.

These steps are only needed once per project per team.

## Creating a Project Team

To collaboration using Bitbucket you first need to create a Bitbucket team. You create a team from the main Bitbucket dashboard page.

The tutorial uses the following for the team name and ID.

|  |  |
| --- | --- |
| **Team Name** | elec5619 Project Team |
| **Team ID** | elec5619\_2015 |

Next, add team members via their e-mail address or Bitbucket username.

## Creating Shared Repo

To share and collaborate on a project you need to create a Bitbucket repo (aka repository) to hold copies of the shared project source. The repo will physically exist on a Bitbucket server and will be shared by all members of the team. As well as storing project source, Bitbucket provides a tracking system – to support change and bug tracking, and a wiki for documentation.

You create a new repo from the main Bitbucket Dashboard page. This tutorial uses the following:

|  |  |
| --- | --- |
| **Owner** | *elec5619* |
| **Name** | *TestRepo* |
| **Description** | *A rather testing repo.* |
| **Access Level** | Leave as: *private repository* |
| **Forking** | Leave as: *allow only private forks* |
| **Repository Type** | Leave as: *Git* |
| **Project Management** | Select:  - *Issue Tracking*  - *Wiki* |
| **Language** | *Java* |

Bitbucket will create an empty repo. Since we have assigned ownership of the repo to our project team, members of the team will be able to access and modify data held in the repo.

## Initialise the repo and master branch

Before we can do anything interesting with the repo it needs to be initialised and its master branch created. To initialise the repo enter the following at the command a line:

mkdir testrepo

cd testrepo

git init

Which creates an empty local repo in the **TestRepo** folder.

To link this local repo to the shared team repo, enter the following at a command line:

git remote add origin https://**xx**@bb.org/path/**TestRepo**.git

Where **xx** is the id of the owner of the repo, and **TestRepo** is the id of the repo we created on bitbucket. This command links the local repo to the remote repo, allowing the local repo to be updated by the remote repo and the remote repo to be updated by the local repo. This allows the changes to be shared with the team.

Next we create the master branch. The master branch holds a shared working version of the project. At anytime, it should be possible to build a working version of the project from the master branch. The master branch can only be created with an initial commit on the repo. To create a commit we need to add a file a to the repo.

A useful file to add at the start of a project is a .gitignore file. A .gitignore file tells Git which files, paths or path fragments should not be included in the repo.

To create an initial .gitignore file, first enter the following at the command line:

echo /target/ > .gitignore

echo .git >> .gitignore

You can add more items to the .gitignore file at any time. A good list of items is: <https://gist.github.com/octocat/9257657>

Next, we update the local repo and push our change back to the team’s shared repo. At the command line, enter the following:

git add .gitignore

git commit -m 'initial .gitignore'

git push origin master

At this point we have an initialised repo ready for use by the team. Next each member clones the repo and begin working on their part of the project.

# Part Two: Using Git to Collaborate

Once the shared repo has been created, each team member needs to joins the project by cloning the project repo locally. Once they have a local clone, they can start making changes to the project source and using git to manage those changes.

This section of the tutorial steps you through the process of:

* Joining the project and cloning the repo.
* Using Git to make and commit changes to the shared project source.

## Join the Project

You join a project by cloning the team’s shared repo. You only need to do this once when you join the project. Cloning copies the team’s shared master branch to your local machine. To clone a repo, enter the following at the command line:

git clone https://user@bb.org/path/repo.git

Which creates a clone of the repo in a new local subdirectory called repo.

Once you have local clone of the repo, you’re free to make and commit changes to the project source.

## Make and Commit Changes

Git uses branches as the mechanism for tracking changes made to the shared project source. Any changes we want to make to a repo must be made using a branch.

Conceptually branches contain a related set of changes – for example, a development iteration, a bug fix or a new feature – made to the project source. Branches are typically atomic – we commit all the changes in a branch or none of the changes in a branch. The amount of work or amount of time that makes up a branch is project dependent, and something your project team may need to agree on.

To make a change using a Git branch:

* Create a new branch to hold the change.
* Working on the change. While working on the change periodically commit the branch to the shared team server.
* When done merge the change into the master branch; make the changes available to the whole team.

### Create a Branch

To start work on a change, we create a new private branch to contain the changes. We need to create a new branch for every change we work on. Before we can create the new branch, we need to ensure our local master branch is up-to-date so that our branch starts on the latest project source.

At the command line enter:

git pull origin master

This fetches and merges into our local master branch any changes made to the remote master branch since our local master branch was last updated.

Next we create a new private branch. At the command line enter:

git branch new\_branch

Finally, to make the new branch our focus, at the command line enter:

git checkout new\_branch

Now, any changes we make will be made locally to new\_branch version of the project source.

### Work on Change

As we make changes to our local copy of the branch it is good practise to periodically commit and push our changes to the shared team repo. This makes a backup copy of our local branch, and provides other team members with an opportunity to review our changes before they are merged into the master branch.

To commit and push changes, at the command line enter:

git add .

git commit -m 'description of change'

git push origin new\_branch

### Merge and Share Change

Once we’ve completed our change, we need to merge our change into the master branch. We also need to somehow ensure that none of our changes break the state of the master branch.

First, we first update our local copy of the master branch. At the command line enter:

git checkout master

git pull origin master

This updates the local master branch with any changes made to the remote master branch since the local master was last updated from it and makes the master branch our focus.

Next, we merge our private branch into the master branch. At the command line enter:

git merge new\_branch

This command gets Git to merge new\_branch into master branch (the branch we are currently focused on). Unless there is a merge conflict, the result of the merge is an updated and committed master branch.

However, before we push the merged master branch back to the team’s shared repo, it is good practise to test the new master branch to ensure our changes haven’t introduced any bugs i.e we haven’t broken the build.

How to test the master branch really depends on the project you are building and how your team has agreed to do things. For example, you may ensure the master branch builds successfully, and that least your changes work before pushing the new master branch back to the team.

To push the new master branch back, at the command line enter:

git push origin master

Which updates the team’s shared repo master branch with our local master branch.

After you have updated the shared master with the branch changes, you can you can delete the local branch with the following:

git branch -d new\_branch

Before the rest of the team can access our changes, they need to update their local master branches.