

•Basic version control with Git and bitbucket.org

This tutorial demonstrates how to set up and use Git as a version control system for source code when multiple developers has to work together on the same project.

Repository: a receptacle or place where things are deposited, stored, or offered for sale.

Bitbucket allows for private repositories with up to 5 users. It is an excellent way to share source code for your project. Another option is github.com however this service does not offer free private repositories. Bitbucket is used in this tutorial.

The tutorial demonstrates using Git with the command prompt. If you are more comfortable with a graphical user interface there are several to choose from (eg. EGit for Eclipse).

Only the most basic operations needed for multiple developers to work together are shown here. For a more complete tutorial of the many features of Git go here:

<http://www.vogella.com/articles/Git/article.html>

Prerequisites

Git must be installed on the computer (<http://git-scm.com/downloads>)

You must have an account with bitbucket (www.bitbucket.org)

Setting up a remote repository with bitbucket.org

Only one developer has to do the following initial setup.

- Sign up with bitbucket.org and create a new repository. Select a name and language for the repository. You can also choose to make the repository private.

It should look like the following:

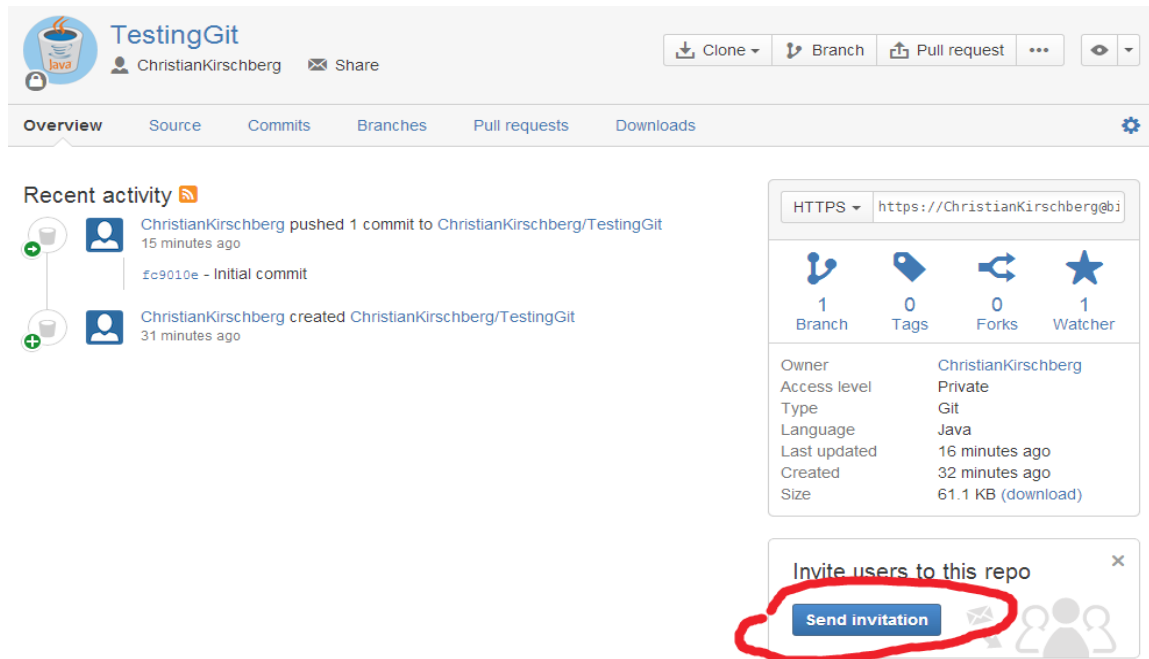
Create a new repository

You can also [import a repository](#)

The screenshot shows the Bitbucket 'Create a new repository' form. The form is divided into two main sections. The left section contains the repository configuration fields: 'Name*' (TestingGit), 'Description' (empty text area), 'Access level' (checked for 'This is a private repository'), 'Forking' (dropdown menu set to 'Allow only private forks'), 'Repository type' (radio buttons for 'Git' and 'Mercurial', with 'Git' selected), 'Project management' (checkboxes for 'Issue tracking' and 'Wiki'), and 'Language' (dropdown menu set to 'Java'). At the bottom of this section are 'Create repository' and 'Cancel' buttons. The right section contains two informational boxes. The top box is titled 'New to Bitbucket?' and contains the text 'Learn the basics of using Git and Mercurial by exploring the Bitbucket 101.' with a graduation cap icon. The bottom box is titled 'Working in a team?' and contains the text 'Create a team account to consolidate your repos and organize your team's work' with an icon of three people.

Invite the other developers to participate in this repository by clicking the link. Fill out each developers git username or the email they signed up with. Grant them the rights needed (Write or

admin access in this example):



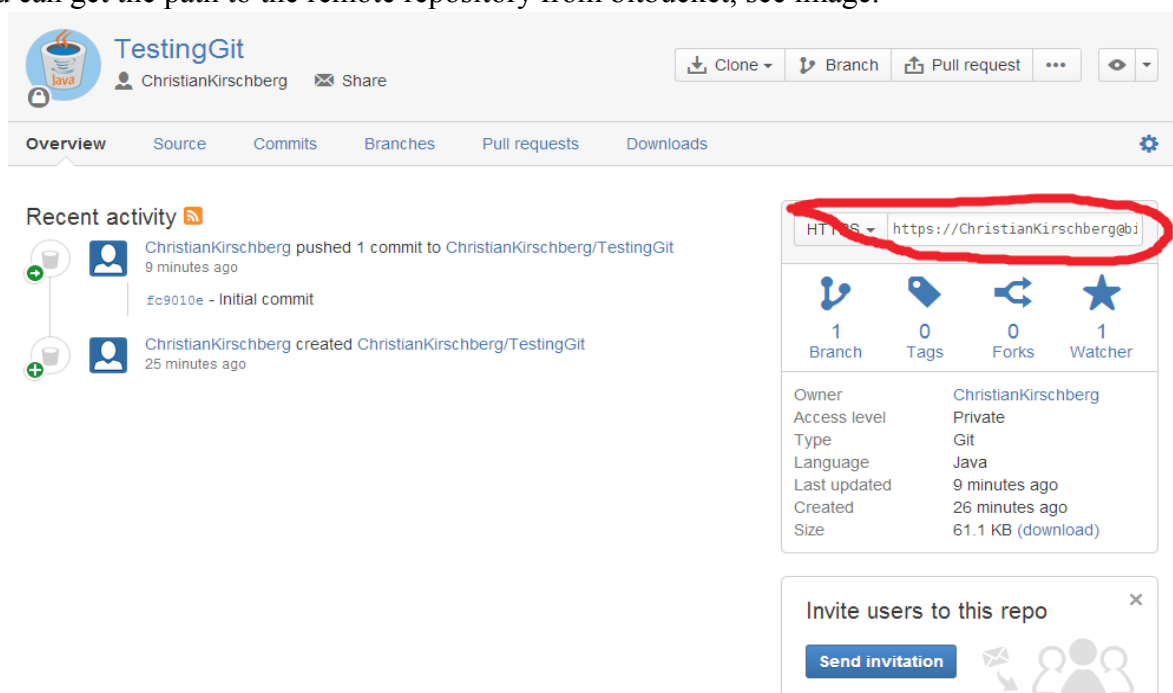
Your shared (remote) repository is setup and ready to use.

Now, let's create your local repository.

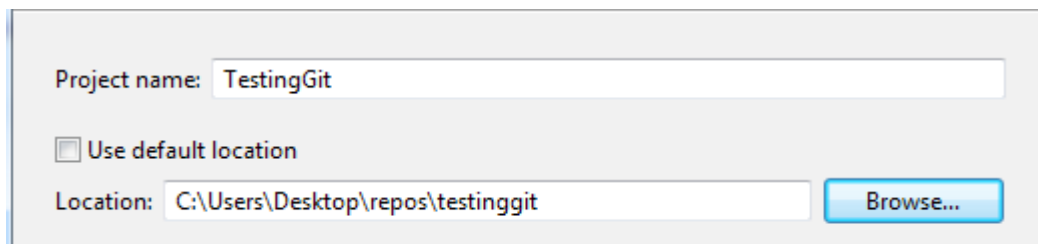
Navigate to the path on your pc/mac where you want your local repository. (Do not put this in a shared (dropbox) folder that other developers also use as their local repository)

In Git Bash call the following commands:

- git init (creates a new local repository in your current path)
- git remote add origin [path to shared repository] (eg git remote add origin https://ChristianKirschberg@bitbucket.org/ChristianKirschberg/gittest1.git)
- You can get the path to the remote repository from bitbucket, see image.



- Go to Eclipse and create a new project. Select the location to correspond to the path where you just created your local repository.



- Create a class (or a readme file) for the project.

Add the new class (or readme file) to the local repository, commit it and push it to the shared repository.

- git add .project
- git add src/*.java (change path and files to add accordingly)
- git commit -m "Initial commit"
- git push -u origin master
- Enter password for the shared repository if needed.

```
Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git add .project
```

```
Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git add src/*.java

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git commit -m "Initial commit"
[master (root-commit) fc9010e] Initial commit
1 file changed, 9 insertions(+)
create mode 100644 src/Run.java

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git push -u origin master
Password for 'https://ChristianKirschberg@bitbucket.org':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 355 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://ChristianKirschberg@bitbucket.org/ChristianKirschberg/testinggit.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$
```

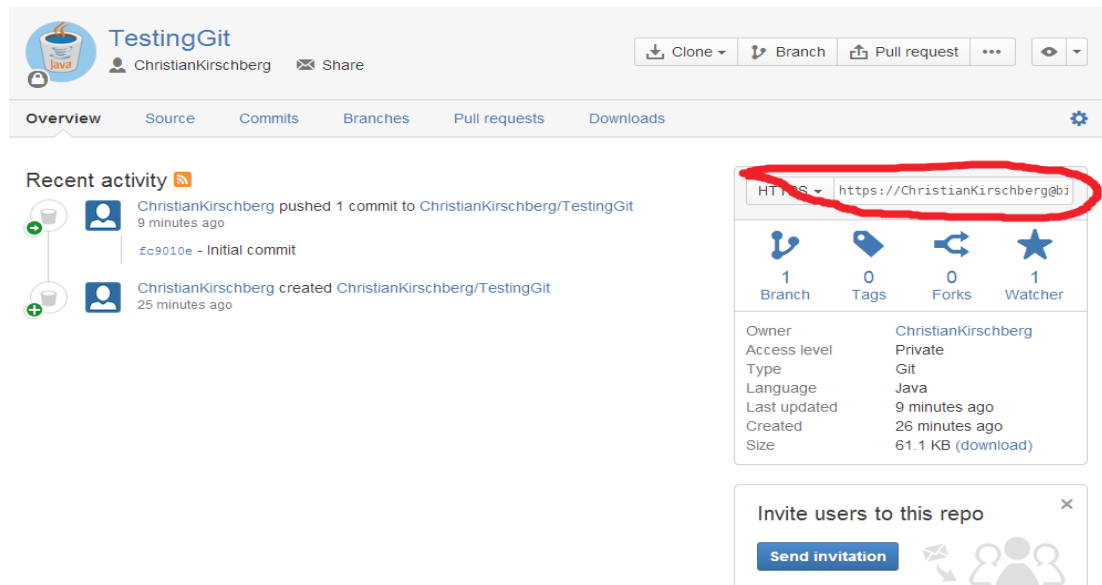
Now your project is in the shared repository.

Other developers can now do the following to get a local copy of the project that was just

created.

Navigate to the path on your pc/mac where you want your local repository. (Do not put this in a shared (dropbox) folder that other developers also use as their local repository). Then copy the remote repository's content to the local repository you are about to create with:

- git clone [path to shared repository] (owner of repository can find it here, see image)



eg. `git clone https://ChristianKirschberg@bitbucket.org/ChristianKirschberg/testinggit.git`

notice the first part is the user. Change this to your bitbucket user eg.

`git clone https://jjensengit@bitbucket.org/ChristianKirschberg/testinggit.git`

- Enter password if needed.

If the clone command fails with the "Authentication failed..." message, the owner did not grant you rights to the repository.

- Now import the project in Eclipse by selecting the Import ->Existing projects into workspace ? browse select root directory and select the path where you cloned to your local repository.

- The project will have a red error mark since you must add the java jre. To do this, right click the project ? properties ? Java build path ? Order and export ? Select the JRE System Library and click ok.

Now you should have a working project with the first class.

Do the same for all developers.

In the next section, user 1 will write new code and user 2 will get the updated version.

User 1:

- Add a new class in Eclipse.

In Git Bash, add, commit and push with the following commands:

- `git add src/*.java`

- git commit -m "Added class cat"
- git push -u origin master
-

```

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git add src/*.java

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git commit -m "Added class"
[master 079bb81] Added class
1 file changed, 4 insertions(+)
create mode 100644 src/Cat.java

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$ git push -u origin master
Password for 'https://ChristianKirschberg@bitbucket.org':
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 375 bytes | 0 bytes/s, done.
Total 4 (delta 0); reused 0 (delta 0)
To https://ChristianKirschberg@bitbucket.org/ChristianKirschberg/testinggit.git
   ab73d01..079bb81  master -> master
Branch master set up to track remote branch master from origin.

Desktop@DESKTOP-PC /c/Users/desktop/repos/testinggit (master)
$

```

User 2:

To get new changes issue the pull command with:

- git pull (if you get an error saying "fatal: not a git repository..." you are probably not in the correct directory).
- Go to Eclipse. The new class should appear.

In this section both users will edit code and we will solve a merge conflict.

In the example user 1 adds the following code to the project:

```

public class Cat {
    private String name;

    public Cat(String n)
    {
        this.name = n;
    }
}

```

User 2 adds the following code to the project:

```

public class Cat {
    private String name;

    public Cat(String name)
    {
        this.name = name;
    }
}

```

Now user 1, adds, commits and pushes the changes by running the following commands:

- git add src/*.java

- git commit -m "Added class cat"
- git push -u origin master

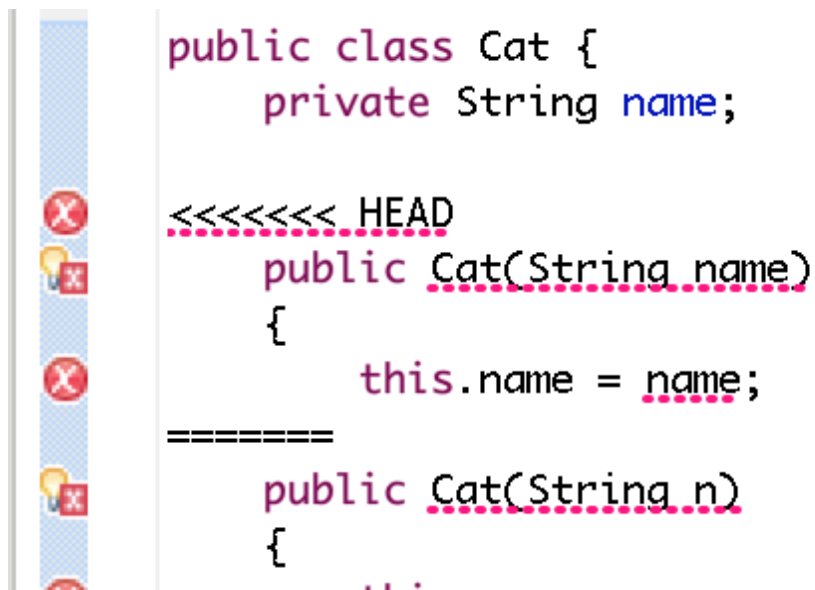
Now user 2 does the same. Notice the message from Git when user 2 tries to push the changes "master ? mater (fetch first)". The push fails because another user made changes. It is now up to user 2 to pull the changes and merge the code before finally pushing the code to the shared repository.

First pull the version from user 1 by calling the following command:

- git pull (You will get the message "CONFLICT (content): Merge conflict in src/Cat.java".)

Go to Eclipse to see the code. It will look like this (don't be scared, it is ok :-)):

—



```

public class Cat {
    private String name;

<<<<<< HEAD
    public Cat(String name)
    {
        this.name = name;
=====
    public Cat(String n)
    {
        ...
  
```

Simply remove the code you do not want. In this case we select one version, since user 1 and 2 did the same work. In many cases you would want to keep both users' work.

After editing, add, commit and push the merged changes with the following commands:

- git add src/*.java
- git commit -m "Solved merge conflict in Cat class"
- git push -u origin master

User 1 can now pull the shared version of the code by calling:

- git pull