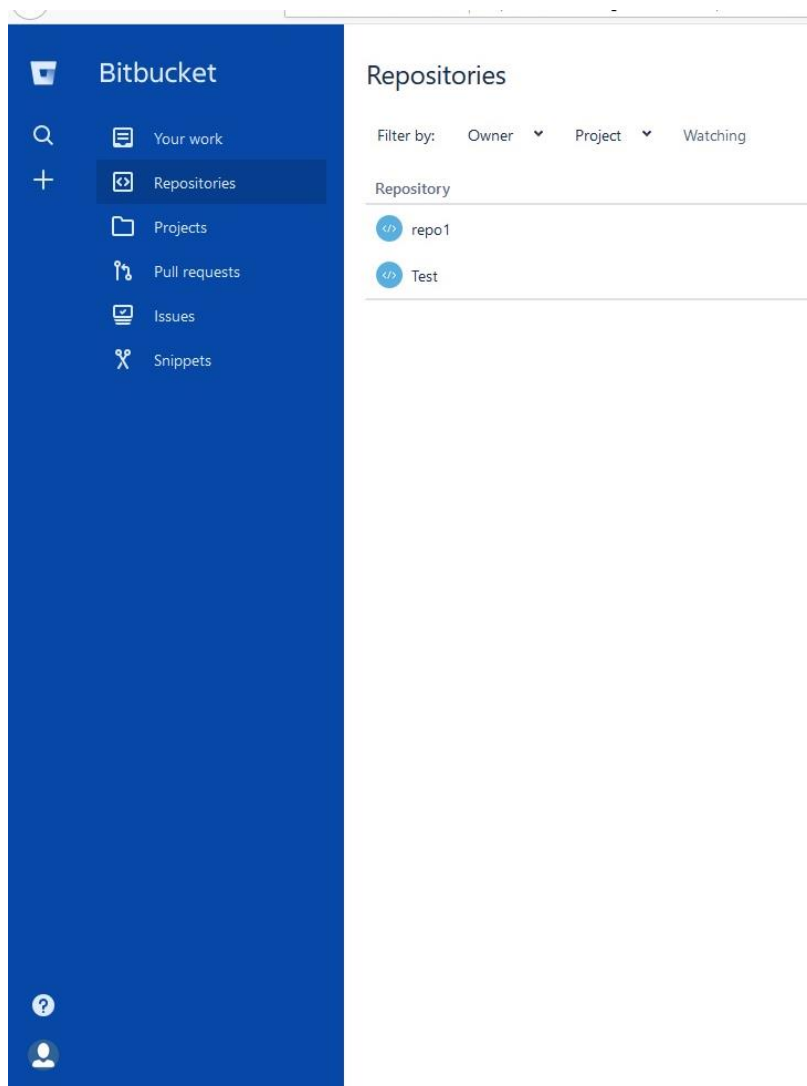


Bitbucket Bash Basics

Creating a repository

After having created a bitbucket account, the first thing you will want to do is create a repository. Creating a repository can be achieved by selecting the '+' symbol on the left hand side of the dashboard.



After selecting the “Repositoies” option, you will be presented with a screen similar to the following providing options for customisation of your repository.

Create a new repository

[Import repository](#)

Repository name *

ProjectOne

Access level

☒ This is a private repository

Include a README?

Yes, with a tutorial (for beginne... ▾

Version control system

☒ Git
☐ Mercurial

Advanced settings

Description

This is a repository that will be utilised for storing files for the parking facility application

Forking

Allow forks ▾

Project management

☒ Issue tracking
☒ Wiki

Language

Select language... ▾

Create repository

Cancel

On this screen you may provide a name, description and the mechanism by which your source will be managed. To make your repository publicly available you may adjust the options accordingly.

After having completed the customisation of the repository, you will be presented with a dashboard that allows you to perform a number of operations related to the management of the project for which you are working upon.

The dashboard for each given repository looks similar to the depiction below allowing the user to perform commits, create branches, review the works of others in addition to reporting issues, approving changes and finally merging those changes into the main branch.

The screenshot shows the Bitbucket interface for a repository named 'ProjectOne' by user 'jaymie'. On the left is a dark blue sidebar with navigation icons and a list of options: Source (selected), Commits, Branches, Pull requests, Pipelines, Deployments, Issues, Wiki, Downloads, Boards, and Settings. The main content area has a header with the repository name and a 'Take the next steps' section. This section includes a folder icon and instructions to clone the repository using the command: `git clone https://SeriousAcademic@bitbucket.org/SeriousAcademic/projectone.git`. Below this, it states the repository's purpose: 'This is a repository that will be utilised for storing files for the parking facility application'. A file browser shows the 'master' branch with a 'Filter files' input. A table lists the files in the repository:

Name	Size	Last commit	Message
README.md	2.56 KB	24 seconds ago	Initial commit

Below the table, the 'README.md' file is expanded, showing a heading 'Edit a file, create a new file, and clone from Bitbucket in under 2 minutes' and a paragraph: 'When you're done, you can delete the content in this README and update the file with details for others getting started with your repository.'

This screen provides you with some basic information to help you get started with your repository, perhaps most importantly, the URL for the repository. This URL will be the most crucial element when attempting to access the features from standalone source management interfaces and the command line.

Cloning a Repository

After the repository has been created you will want to add some files to the repository to be managed. This is achieved by first cloning the repository to your local machine, this has been shown using the BASH command prompt however many tools may be utilised for this including the use of Sourcetree. Whilst performing this option, a folder in which the repository resides must be stipulated.

```
MINGW64:/d
jaymi@DESKTOP-JUS7BLD MINGW64 /d
$ git clone https://SeriousAcademic@bitbucket.org/SeriousAcademic/projectone.git ProjectOne
Cloning into 'ProjectOne'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.

jaymi@DESKTOP-JUS7BLD MINGW64 /d
$
```

After you have cloned the repository to your local machine, you are able to add files to the repository within the file explorer being utilised. This should be done so in whatever may you wish, whether this is created by means of an IDE or through the dragging and dropping of files into the root folder.

The files that have been added to the repository folder may then be staged for adding to the repository. This means that the files will be recorded for addition to the repository creating change information for them. This can be achieved by the “**git add**” command. The asterisk appended to the example given, includes all new files added to the project, this may be amended to ignore or include only specific files.

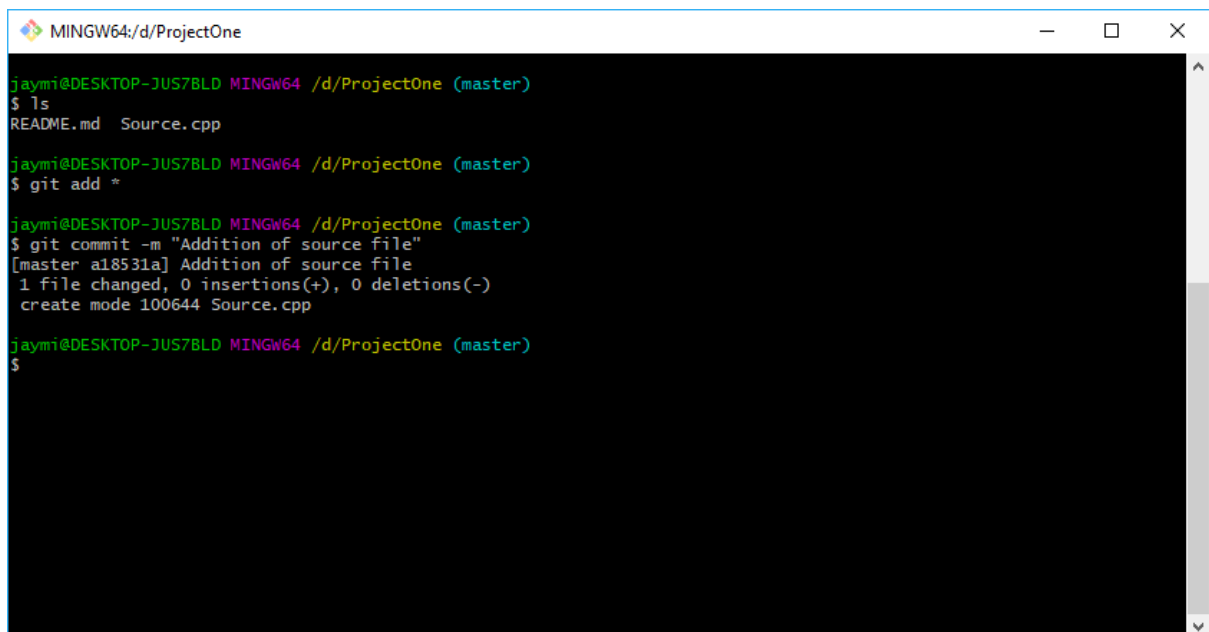
```
MINGW64:/d/ProjectOne
jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ ls
README.md  Source.cpp
jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git add *
jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$
```

Git Commit

The final stage that is required for a repository to track the changes of a file is the commit stage. The commit creates checksums of the files that you are adding to your project and provisions hash values for the current state of these files. To create this information the “git commit” command is utilised.

It is often the case that a comment is supplied with the commit operation, this may be achieved in many ways, but the simplest of all is through the use of the “-m” switch provided when initiating a commit.

Example : git commit -m “Document update”

A screenshot of a terminal window titled 'MINGW64; d/ProjectOne'. The terminal shows a series of commands and their outputs. First, the user runs 'ls', which lists 'README.md' and 'Source.cpp'. Then, they run 'git add .' to stage the files. Finally, they run 'git commit -m "Addition of source file"', which results in a commit hash 'a18531a' and a message 'Addition of source file'. The output also shows '1 file changed, 0 insertions(+), 0 deletions(-)' and 'create mode 100644 Source.cpp'. The prompt returns to '\$' after the commit.

```
jaymi@DESKTOP-JU57BLD MINGW64 /d/ProjectOne (master)
$ ls
README.md  Source.cpp

jaymi@DESKTOP-JU57BLD MINGW64 /d/ProjectOne (master)
$ git add .

jaymi@DESKTOP-JU57BLD MINGW64 /d/ProjectOne (master)
$ git commit -m "Addition of source file"
[master a18531a] Addition of source file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Source.cpp

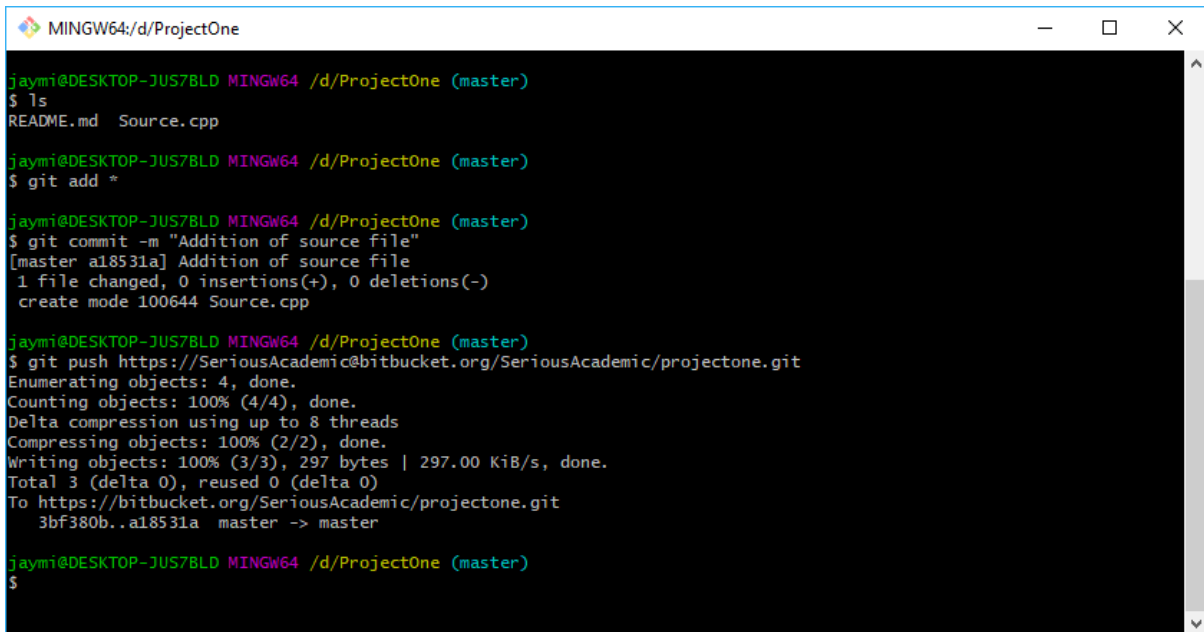
jaymi@DESKTOP-JU57BLD MINGW64 /d/ProjectOne (master)
$
```

This commit, ensures that changes are tracked for this file within the local repository. This means that whenever any changes occur to this file on the local machine, and subsequently committed, the changes will be stored within the git information store for this repository.

This does not mean however that the other users of the repository will be aware of the changes that you have made on your local machine, for this another command is utilised.

Git Push

To ensure that the other users of the repository have access or are aware of the changes that you have made to the code, the push command is used. The push command provides a mechanism by which you may transport the information related to your commits from your local machine to the public version of the repository. This is done so utilising the URL from the repository.

A screenshot of a terminal window titled 'MINGW64; d/ProjectOne'. The terminal shows a series of Git commands and their outputs. The user lists files, adds them to the repository, commits the changes with a message, and then pushes the commit to a remote repository on Bitbucket. The output shows the progress of the push, including object enumeration, counting, compression, and writing. The final output shows the commit being pushed to the master branch.

```
MINGW64; d/ProjectOne

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ ls
README.md  Source.cpp

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git add *

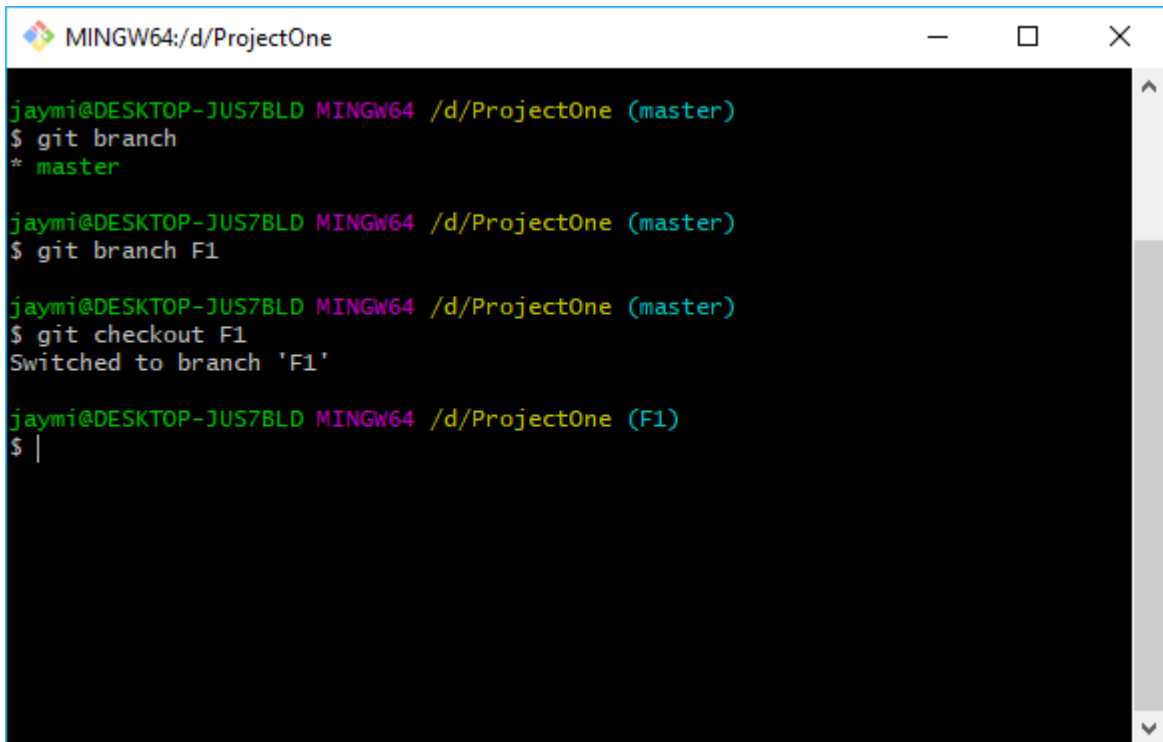
jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git commit -m "Addition of source file"
[master a18531a] Addition of source file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Source.cpp

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git push https://SeriousAcademic@bitbucket.org/SeriousAcademic/projectone.git
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://bitbucket.org/SeriousAcademic/projectone.git
3bf380b..a18531a master -> master

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$
```

Git Branch

When a new feature requires implementation, a branch is typically created from the main source tree to ensure sanitation and isolation of the new feature from the functional code in the repositories. When a branch is created in the local version of the repository, only the local user is aware of its existence, it is not until the changes are committed and you push the changes to the public server that they are visible to the rest of the team.

A screenshot of a terminal window titled 'MINGW64:/d/ProjectOne'. The terminal shows a series of Git commands being executed. The prompt is 'jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)'. The first command is '\$ git branch', which outputs '* master'. The second command is '\$ git branch F1'. The third command is '\$ git checkout F1', which outputs 'Switched to branch 'F1''. The final prompt is 'jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (F1)' followed by '\$ |'.

```
MINGW64:/d/ProjectOne

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git branch
* master

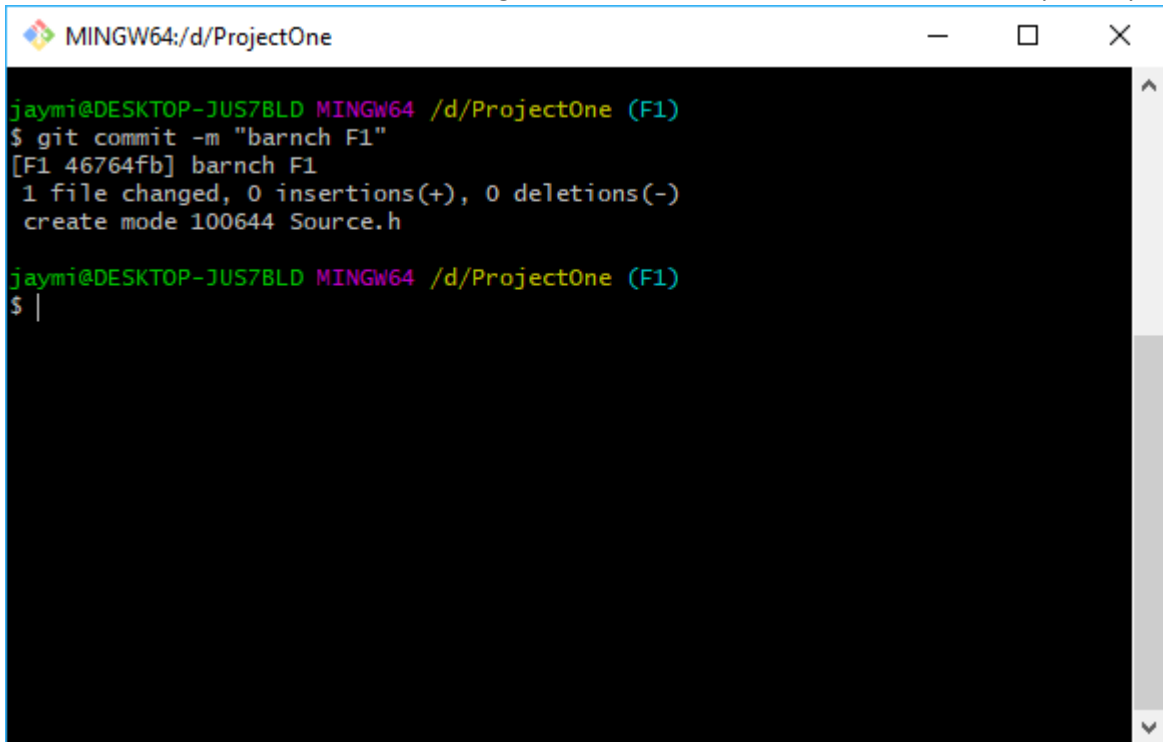
jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git branch F1

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (master)
$ git checkout F1
Switched to branch 'F1'

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (F1)
$ |
```

After the creation of a branch it must be checked out for it to be utilised. The example provided above performs both of these actions, not only creating the branch if it does not exist but also checking out the branch for utilisation.

After having added the required files and updated the information for the branch, in the same fashion as the main source tree, the changes should be added and committed to the repository.

A screenshot of a terminal window titled "MINGW64:/d/ProjectOne". The terminal shows a user named "jaymi" at a desktop named "JUS7BLD" in a "MINGW64" environment, working in the directory "/d/ProjectOne" on the "F1" branch. The user enters the command "\$ git commit -m 'barnch F1'". The output shows the commit hash "[F1 46764fb]", the branch name "barnch F1", and the commit details: "1 file changed, 0 insertions(+), 0 deletions(-)" and "create mode 100644 Source.h". The prompt returns to "\$ |".

```
MINGW64:/d/ProjectOne

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (F1)
$ git commit -m "barnch F1"
[F1 46764fb] barnch F1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Source.h

jaymi@DESKTOP-JUS7BLD MINGW64 /d/ProjectOne (F1)
$ |
```

This example shows the commit to the branch "F1" indicating the utilisation of this branch in place of the master branch of the source tree.

Pull Requests

After having added your changes to the repository, you will want to make the other members of your team aware that they need to review and approve the changes or additions that you have proposed. This is achieved through a pull request. The easiest means through which this may be accomplished is through the use of the options on the bitbucket dashboard, although recently commands have become available for use with command lines. Creation of a pull request has been depicted below.

jaymie / ProjectOne / Pull requests

Create a pull request

Source: SeriousAcademic / ProjectOne
Created 10 hours ago, updated 11 minutes ago
Branch: F1

Destination: SeriousAcademic/projectone
Branch: master

Title: branch F1

Description: **Aa** **B** **I** ... **Link** **Image** **Emoji** **More** **Feedback** **?**

Reviewers: Start typing to search for a user

Close branch: ☒ Close **F1** after the pull request is merged

[Create pull request](#)

Author	Commit	Message
jaymie	46764fb	branch F1

This image clearly shows the path of the changes that have been made and the intended destination upon approval. This form allows for the developer to provide further annotations for the changes that they have made in addition to the commit comments.

Git Commands

Git init – Initialises an empty repository in the folder where the command is called

Git add <option> – Adds files to the repository in which the command is called if the command is called within a repository

Git Commit – Commit the changes within the staging area to the git data store, recording any changes to said files.

Git push <location> <branch> - push the locally committed changes to the location and branch specified

Git pull – retrieve the latest version from the public store and checkout

Git log- retrieve the information relating to all commits for current branch

Git status – retrieve information relating to the current branch