**Git**:

Git is a free and open source version control system, originally created by Linus Torvalds in 2005.

Version Control System (VCS) is software that helps software developers to work together and maintain

a complete history of their work.

Git does not use a centralised server.

Git runs on major operating system like OS X, Windows, Unix & Linux.

Git also has excellent support for branching, merging, and rewriting repository history, which has lead to many innovative and powerful workflows and tools. Pull requests are one such popular tool that allow teams to collaborate on Git branches and efficiently review each others code. Git is the most widely used version control system in the world today and is considered the modern standard for software development.

**Advantages of Git:**

Everything is local(almost)

Fast

Every clone/branch is a backup

Work offline

No single point of failure

Lightweight

Branching is cheap and merging is easy

Distributed

Every file and commit is checksummed.

Free and open source.

**Listed below are the functions of a VCS:**

* Allows developers to work simultaneously.
* Does not allow overwriting each other’s changes.
* Maintains a history of every version.

Let us see the basic workflow of Git.

1. You modify a file from the working directory.
2. You add these files to the staging area.
3. You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.

Git Directory Repository

**Staging Area**

**Working Directory**

**Stage files**

**Commit**

**Checkout project**

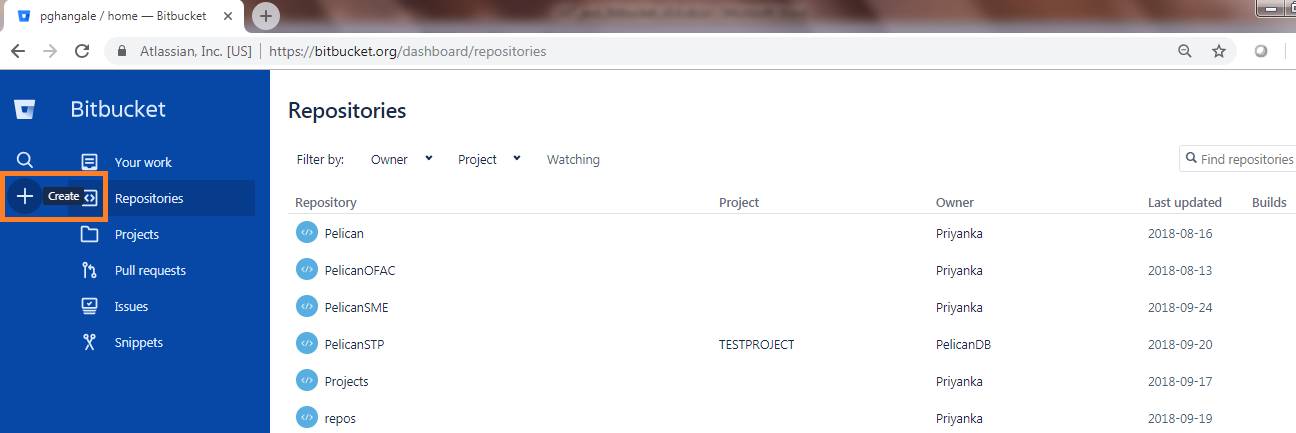
**Bitbucket**:

Bitbucket is a web-based version control repository hosting service owned by Atlassian, for source code and development projects that use either Mercurial or Git revision control systems.

**Basic commands of Git:**

1. Do the following to create your repository:

From Bitbucket, click the + icon in the global sidebar and select Repository.



|  |
| --- |
|  |

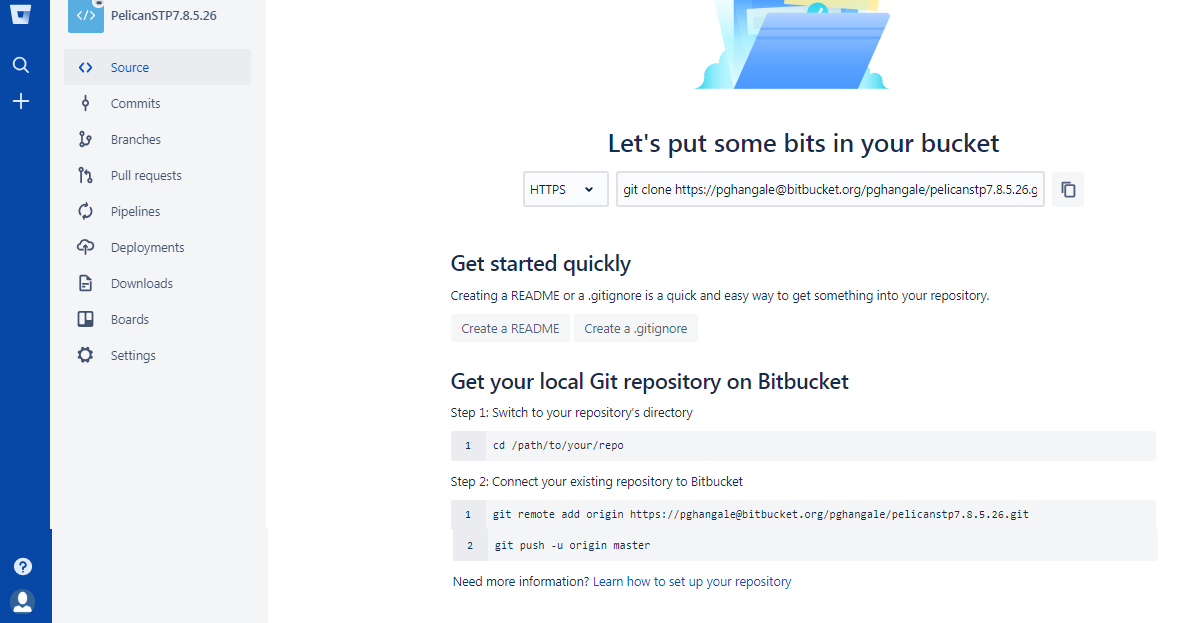
Bitbucket displays the Create a new repository page.

|  |
| --- |
|  |

1. Enter **PelicanSTP7.8.5.26** for the **Name** field. Bitbucket uses this Name in the URL of the repository.

For example, if the user pghangale has a repository called awesome\_repo, the URL for that repository would be **https://bitbucket.org/pghangale/awesome\_repo**.

1. For **Access level**, leave the **This is a private repository box** checked. A private repository is only visible to you and those you give access to.
2. If this box is unchecked, everyone can see your repository.
3. Pick Git for the Repository type. You can't change the repository type after you click Create repository.
4. Click Create repository. Bitbucket creates your repository and displays its Overview page.



1. Click + from the global sidebar for common actions for a repository. Click items in the navigation sidebar to see what's behind each one, including Settings to update repository details and other settings.
2. When you click the Commits option in the sidebar, you find that you have no commits because you have not created any content for your repository. Your repository is private and you have not invited anyone to the repository, so the only person who can create or edit the repository's content right now is you, the repository owner.

**Copy your Git repository and add files:**

Now that you have a place to add and share your space station files, you need a way to get to it from your local system. To set that up, you want to copy the Bitbucket repository to your system. Git refers to copying a repository as "cloning" it. When you clone a repository, you create a connection between the Bitbucket server (which Git knows as origin) and your local system.

1. Clone your repository to your local system.

Open a browser and a terminal window from your desktop. After opening the terminal window, do the following:

Navigate to your home directory.

|  |
| --- |
| C:\Users\pghangale>**pushd D:\bitbucket\TEST** |

1. Create a directory to contain your repositories.

|  |
| --- |
| D:\bitbucket\TEST>**mkdir PelicanSTP7.8.5.26** |

1. From the terminal, update the directory you want to work in to your new repos directory.

|  |
| --- |
| D:\bitbucket\TEST>**cd D:\bitbucket\TEST\PelicanSTP7.8.5.26** |

1. From Bitbucket, go to your PelicanSTP7.8.5.26 repository.
2. Click the + icon in the global sidebar and select Clone this repository.
3. Bitbucket displays a pop-up clone dialog. By default, the clone dialog sets the protocol to HTTPS or SSH, depending on your settings. For the purposes of this tutorial, don't change your default protocol.

|  |
| --- |
|  |

1. Copy the highlighted clone command.

|  |
| --- |
|  |

**Command**: git clone https://pghangale@bitbucket.org/pghangale/PelicanSTP7.8.5.26.git

1. From your terminal window, paste the command you copied from Bitbucket and press Return.
2. Enter your Bitbucket password when the terminal asks for it. If you created an account by linking to Google, use your password for that account.

|  |
| --- |
| D:\bitbucket\TEST\PelicanSTP7.8.5.26>git clone https://pghangale@bitbucket.org/p  ghangale/PelicanSTP7.8.5.26.git  Cloning into 'PelicanSTP7.8.5.26'...  Password for 'https://pghangale@bitbucket.org':  warning: You appear to have cloned an empty repository. |

|  |
| --- |
|  |

**Add a file to your local repository and put it on Bitbucket:**

With the repository on your local system, it's time to get to work. You want to start keeping track of all your space station locations. To do so, let's create a file about all your locations.

1. Go to your terminal window and navigate to the top level of your local repository.

|  |
| --- |
| **cd D:\bitbucket\TEST\PelicanSTP7.8.5.26** |

1. Enter the following line into your terminal window to create a new file with content.

|  |
| --- |
| **echo "Earth's Moon" >> locations.txt** |

If the command line doesn't return anything, it means you created the file correctly!

1. Get the status of your local repository. The git status command tells you about how your project is progressing in comparison to your Bitbucket repository.

At this point, Git is aware that you created a new file, and you'll see something like this:

|  |
| --- |
| $ **git status**  On branch master  No commits yet  Untracked files:  (use "git add <file>..." to include in what will be committed)  locations.txt  nothing added to commit but untracked files present (use "git add" to track)  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

The file is untracked, meaning that Git sees a file not part of a previous commit. The status output also shows you the next step: adding the file.

1. Tell Git to track your new locations.txt file using the git add command. Just like when you created a file, the git add command doesn't return anything when you enter it correctly.

|  |
| --- |
| **git add locations.txt** |

The git add command moves changes from the working directory to the Git staging area. The staging area is where you prepare a snapshot of a set of changes before committing them to the official history.

1. Check the status of the file.

|  |
| --- |
| $ **git status**  On branch master  No commits yet  Changes to be committed:  (use "git rm --cached <file>..." to unstage)  new file: locations.txt  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

1. Now you can see the new file has been added (staged) and you can commit it when you are ready. The git status command displays the state of the working directory and the staged snapshot.
2. Issue the git commit command with a commit message, as shown on the next line. The -m indicates that a commit message follows.

|  |
| --- |
| $ **git commit -m "Initial commit"**  [master (root-commit) 89d8041] Initial commit  1 file changed, 1 insertion(+)  create mode 100644 locations.txt  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

The git commit takes the staged snapshot and commits it to the project history. Combined with git add, this process defines the basic workflow for all Git users.

Up until this point, everything you have done is on your local system and invisible to your Bitbucket repository until you push those changes.

* Learn a bit more about Git and remote repositories.
  + Git's ability to communicate with remote repositories (in your case, Bitbucket is the remote repository) is the foundation of every Git-based collaboration workflow.
  + Git's collaboration model gives every developer their own copy of the repository, complete with its own local history and branch structure. Users typically need to share a series of commits rather than a single changeset. Instead of committing a changeset from a working copy to the central repository, Git lets you share entire branches between repositories.
  + You manage connections with other repositories and publish local history by "pushing" branches to other repositories. You see what others have contributed by "pulling" branches into your local repository.

Go back to your local terminal window and send your committed changes to Bitbucket using git push origin master. This command specifies that you are pushing to the master branch (the branch on Bitbucket) on origin (the Bitbucket server).

You should see something similar to the following response:

|  |
| --- |
| $ **git push origin master**  Enumerating objects: 3, done.  Counting objects: 100% (3/3), done.  Writing objects: 100% (3/3), 229 bytes | 114.00 KiB/s, done.  Total 3 (delta 0), reused 0 (delta 0)  To https://bitbucket.org/pghangale/PelicanSTP7.8.5.26.git  \* [new branch] master -> master  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

Your commits are now on the remote repository (origin).

1. Go to your PelicanSTP7.8.5.26 repository on Bitbucket.
2. If you click Commits in the sidebar, you'll see a single commit on your repository. Bitbucket combines all the things you just did into that commit and shows it to you.
3. If you click Source in the sidebar, you'll see that you have a single source file in your repository, the **locations.txt** file you just added.

|  |
| --- |
|  |

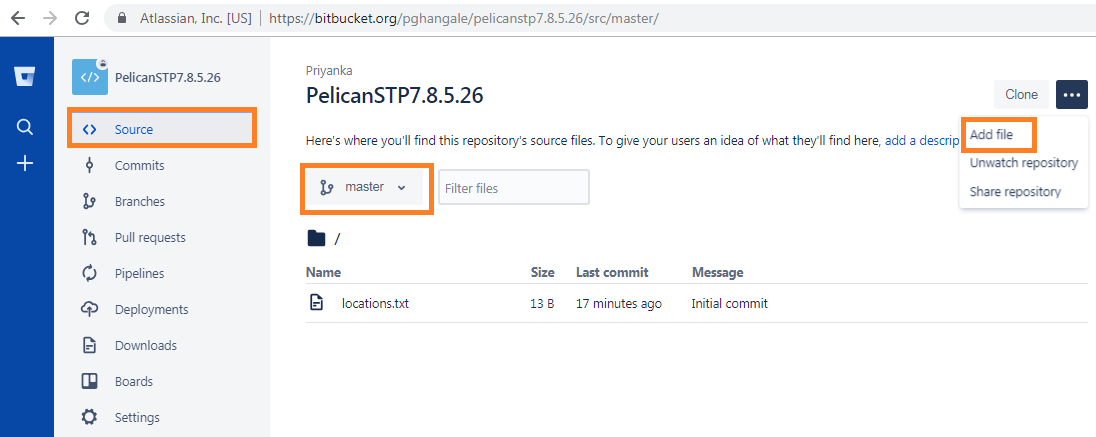
**Pull changes from your Git repository on Bitbucket Cloud:**

Next on your list of space station administrator activities, you need a file with more details about your locations. Since you don't have many locations at the moment, you are going to add them right from Bitbucket.

**Create a file in Bitbucket:**

To add your new locations file, do the following:

1. From your PelicanSTP7.8.5.26 repository, click Source to open the source directory. Notice you only have one file, locations.txt, in your directory.



A. **Branch selection**: Pick the branch you want to view.

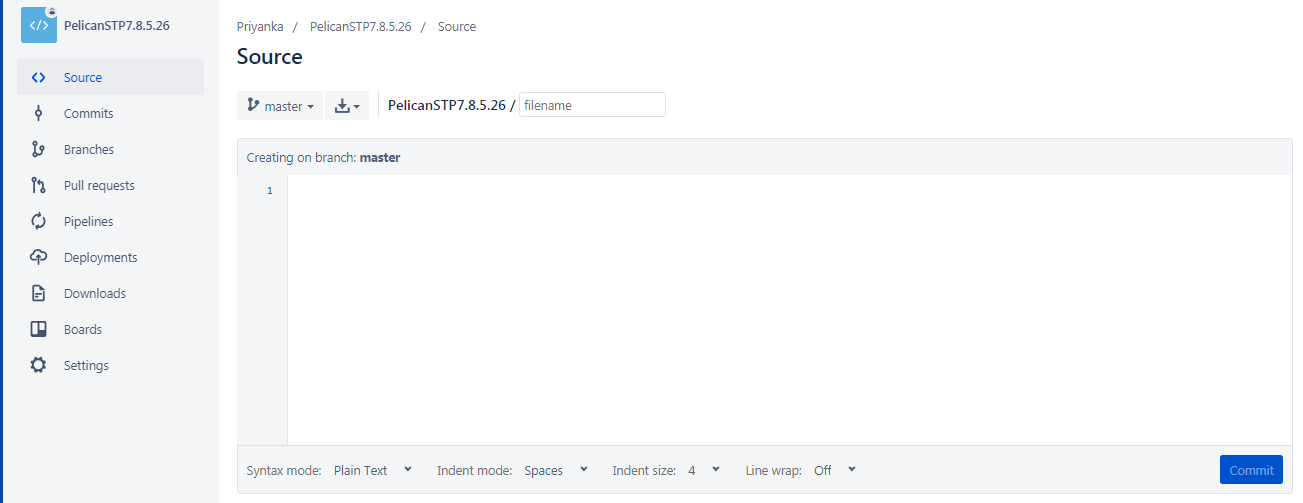
B. **Source page**: Click the link to open this page.

C. **Add file button**: Edit and create a file in Bitbucket.

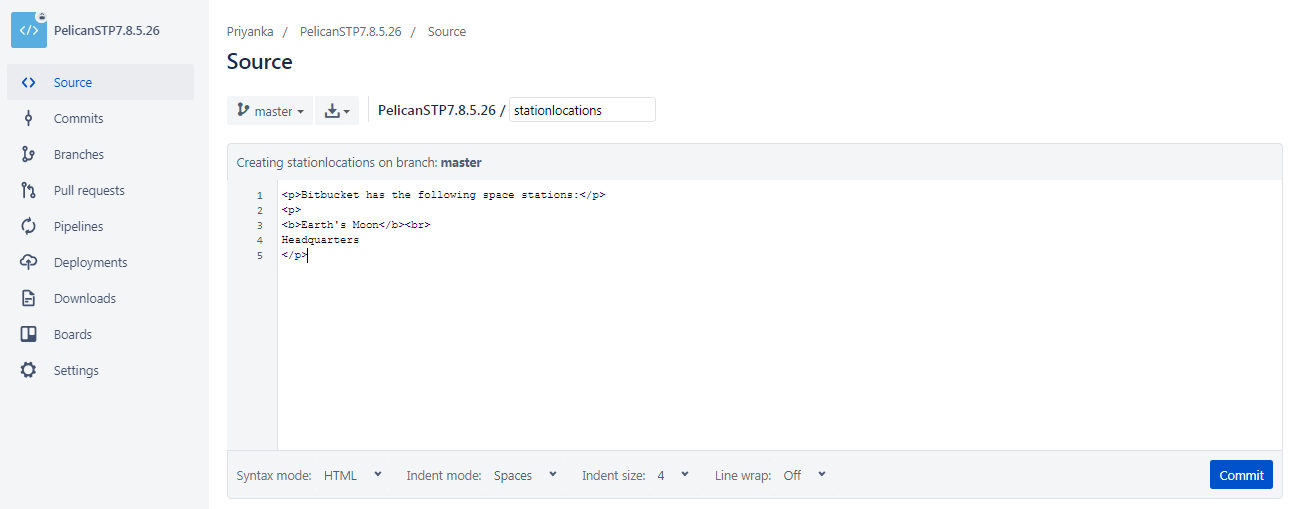
D. **Source file area**: View the directory of files in Bitbucket.

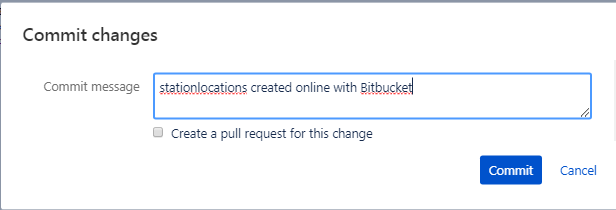
2. From the **Source** page, click **New** **file** in the top right corner. This button only appears after you have added at least one file to the repository.

A page for creating the new file opens, as shown in the following image.



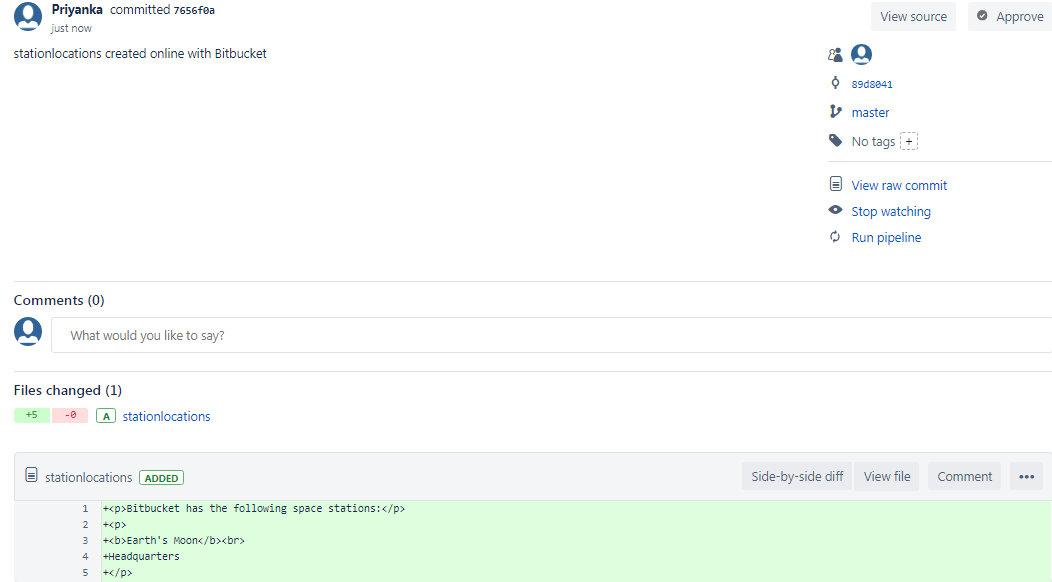
1. **Branch with new file**: Change if you want to add file to a different branch.
2. **New file area**: Add content for your new file here.
3. Enter stationlocations in the filename field.
4. Select HTML from the Syntax mode list.
5. Add the following HTML code into the text box:





1. Click Commit. The Commit message field appears with the message: **stationlocations created online with Bitbucket**.
2. Click Commit under the message field.

You now have a new file in Bitbucket! You are taken to a page with details of the commit, where you can see the change you just made:



If you want to see a list of the commits you've made so far, click commits in the sidebar.

**Pull changes from a remote repository:**

Now we need to get that new file into your local repository. The process is pretty straight forward, basically just the reverse of the push you used to get the locations.txt file into Bitbucket.

To pull the file into your local repository, do the following:

1. Open your terminal window and navigate to the top level of your local repository.
2. Enter the git pull --all command to pull all the changes from Bitbucket. (In more complex branching workflows, pulling and merging all changes might not be appropriate.) Enter your Bitbucket password when asked for it. Your terminal should look similar to the following:

|  |
| --- |
| $ **git pull --all**  Fetching origin  remote: Counting objects: 3, done.  remote: Compressing objects: 100% (3/3), done.  remote: Total 3 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (3/3), done.  From https://bitbucket.org/pghangale/PelicanSTP7.8.5.26  89d8041..7656f0a master -> origin/master  Updating 89d8041..7656f0a  Fast-forward  stationlocations | 5 +++++  1 file changed, 5 insertions(+)  create mode 100644 stationlocations  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

|  |
| --- |
|  |

The git pull command merges the file from your remote repository (Bitbucket) into your local repository with a single command.

Navigate to your repository folder on your local system and you'll see the file you just added.

**Use a Git branch to merge a file:**

Learning branches will allow you to update your files and only share the information when you're ready.

Branches are most powerful when you're working on a team. You can work on your own part of a project from your own branch, pull updates from Bitbucket, and then merge all your work into the main branch when it's ready.

A branch represents an independent line of development for your repository. Think of it as a brand-new working directory, staging area, and project history. Before you create any new branches, you automatically start out with the main branch (called master).

**Git branching**:

**Create a branch and make a change.**

It's important to understand that branches are just pointers to commits. When you create a branch, all Git needs to do is create a new pointer—it doesn’t create a whole new set of files or folders.

To create a branch, do the following:

1. Go to your terminal window and navigate to the top level of your local repository using the following command:

|  |
| --- |
| cd D:\bitbucket\TEST\PelicanSTP7.8.5.26\PelicanSTP7.8.5.26 |

1. Create a branch from your terminal window.

|  |
| --- |
| $ **git branch future-plans**  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

The repository history remains unchanged. All you get is a new pointer to the current branch. To begin working on the new branch, you have to check out the branch you want to use.

1. Checkout the new branch you just created to start using it.

|  |
| --- |
| $ **git** **checkout future-plans**  Switched to branch 'future-plans'  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (future-plans) |

1. Search for the **PelicanSTP7.8.5.26** folder on your local system and open it. You will notice there are no extra files or folders in the directory as a result of the new branch.
2. Open the stationlocations file using a text editor.
3. Make a change to the file by adding another station location:

|  |
| --- |
| <p>Bitbucket has the following space stations:</p>  <p>  <b>Earth's Moon</b><br>  Headquarters  </p>  <p>  <b>Mars</b><br>  Recreation Department  </p> |

1. Save and close file.
2. Enter git status in the terminal window. You will see something like this:

|  |
| --- |
| $ **git status**  On branch future-plans  Changes not staged for commit:  (use "git add/rm <file>..." to update what will be committed)  (use "git checkout -- <file>..." to discard changes in working directory)  deleted: stationlocations  Untracked files:  (use "git add <file>..." to include in what will be committed)  stationlocations.txt  no changes added to commit (use "git add" and/or "git commit -a")  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (future-plans) |

Notice the On branch future-plans line? If you entered git status previously, the line was on branch master because you only had the one master branch. Before you stage or commit a change, always check this line to make sure the branch where you want to add the change is checked out.

1. Stage your file.

|  |
| --- |
| $ **git add stationlocations.txt**  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (future-plans) |

1. Enter the git commit command in the terminal window, as shown with the following:

|  |
| --- |
| $ **git** **commit stationlocations.txt -m "making a change in a branch"**  [future-plans a64e342] making a change in a branch  1 file changed, 9 insertions(+)  create mode 100644 stationlocations.txt  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (future-plans) |

Now it's time to merge the change that you just made back into the master branch.

**To complete a fast-forward merge do the following:**

1. Go to your terminal window and navigate to the top level of your local repository.
2. Enter the git status command to be sure you have all your changes committed and find out what branch you have checked out.
3. Switch to the master branch.

|  |
| --- |
| $ **git checkout master**  Switched to branch 'master'  D stationlocations  Your branch is up to date with 'origin/master'.  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

1. Merge changes from the future-plans branch into the master branch. It will look something like this:

|  |
| --- |
| $ **git merge future-plans**  Updating 7656f0a..a64e342  Fast-forward  stationlocations.txt | 9 +++++++++  1 file changed, 9 insertions(+)  create mode 100644 stationlocations.txt  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

You've essentially moved the pointer for the master branch forward to the current head and your repository looks something like the fast forward merge above.

1. Because you don't plan on using future-plans anymore, you can delete the branch.

|  |
| --- |
| **git branch -d future-plans** |

When you delete future-plans, you can still access the branch from master using a commit id. For example, if you want to undo the changes added from future-plans, use the commit id you just received to go back to that branch.

1. Enter git status to see the results of your merge, which show that your local repository is one ahead of your remote repository. It will look something like this:

|  |
| --- |
| $ **git status**  On branch master  Your branch is ahead of 'origin/master' by 1 commit.  (use "git push" to publish your local commits)  Changes not staged for commit:  (use "git add/rm <file>..." to update what will be committed)  (use "git checkout -- <file>..." to discard changes in working directory)  deleted: stationlocations  no changes added to commit (use "git add" and/or "git commit -a")  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

Here's what you've done so far:

Created a branch and checked it out

Made a change in the new branch

Committed the change to the new branch

Integrated that change back into the main branch

Deleted the branch you are no longer using.

Next, we need to push all this work back up to Bitbucket, your remote repository.

**Push your change to Bitbucket:**

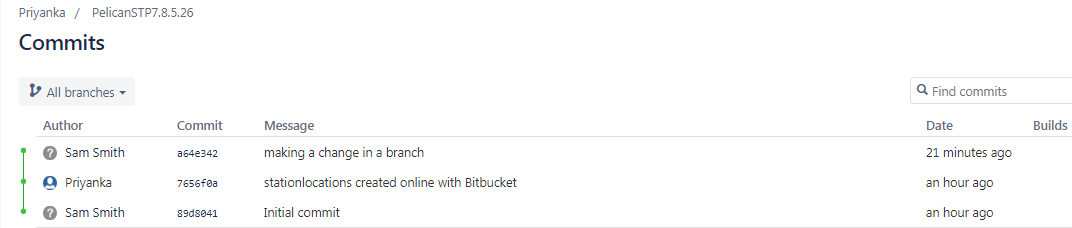
You want to make it possible for everyone else to see the location of the new space station. To do so, you can push the current state of your local repository to Bitbucket.

Here's how to push your change to the remote repository:

1. From the repository directory in your terminal window, enter git push origin master to push the changes. It will result in something like this:

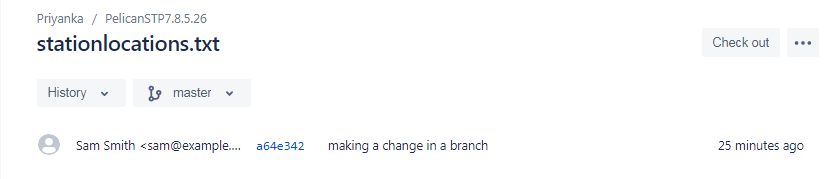
|  |
| --- |
| $ **git push origin master**  Enumerating objects: 4, done.  Counting objects: 100% (4/4), done.  Delta compression using up to 4 threads.  Compressing objects: 100% (3/3), done.  Writing objects: 100% (3/3), 420 bytes | 140.00 KiB/s, done.  Total 3 (delta 0), reused 0 (delta 0)  To https://bitbucket.org/pghangale/PelicanSTP7.8.5.26.git  7656f0a..a64e342 master -> master  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/PelicanSTP7.8.5.26/PelicanSTP7.8.5.26 (master) |

1. Click the Overview page of your Bitbucket repository, and notice you can see your push in the Recent Activity stream.
2. Click Commits and you can see the commit you made on your local system. Notice that the change keeps the same commit id as it had on your local system.



You can also see that the line to the left of the commits list has a straight-forward path and shows no branches. That’s because the future-plans branch never interacted with the remote repository, only the change we created and committed.

1. Click **Branches** and notice that the page has no record of the branch either.
2. Click **Source**, and then click the stationlocations file. You can see the last change to the file has the commit id you just pushed.
3. Click the file history list to see the changes committed for this file, which will look similar to the following figure.



**Learn about code review in Bitbucket Cloud:**

Create a repository and add a teammate/ Clone and make a change on a new branch /If you're using command line /If you're using Sourcetree.

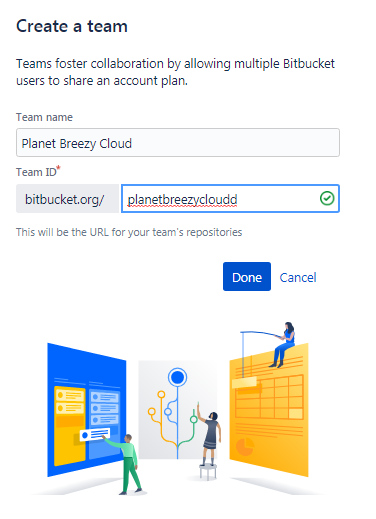
**Create a repository and add a teammate:**

You just arrived at the Bitbucket space station and it's time to go through the orientation process, part of which involves making updates to your welcome package and getting them approved. To get you started, we'll walk you through creating a team repository with some content and giving someone access.

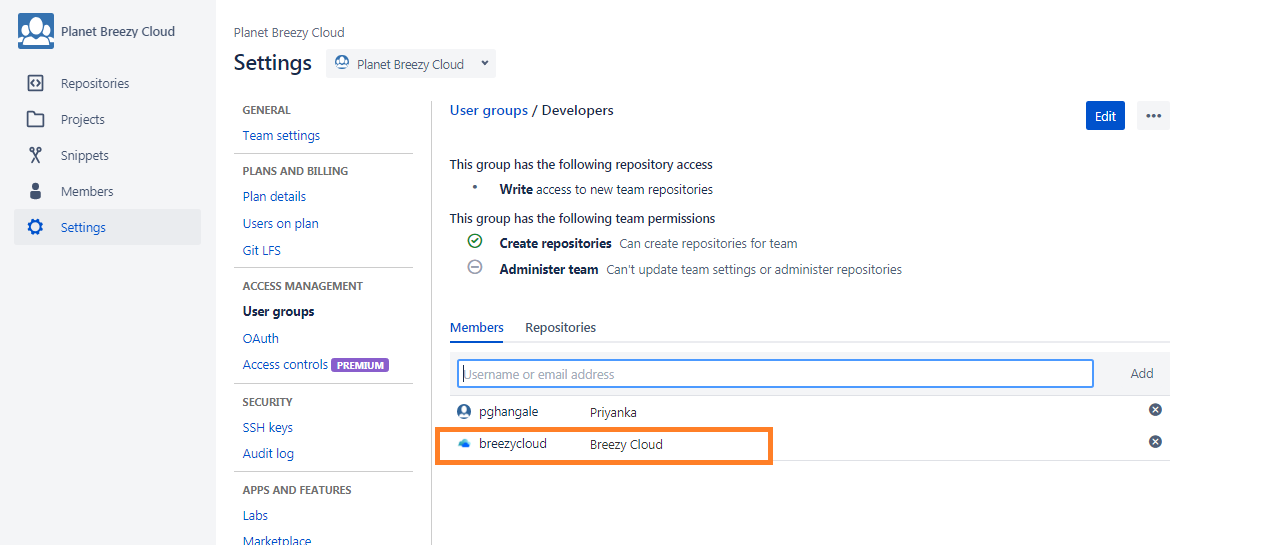
**Create a team and add a teammate:**

Start by creating a team for your repository and teammate. No need to have a teammate for this tutorial. For our purposes, you'll make a new friend that goes by the username **breezycloud**.

1. Click + in the global sidebar and select Team under Create.
2. Enter a Team name you'd like to use. If you can't think of any, we suggest starting with Planet followed by a number or name, for example Planet Breezy Cloud or Planet 727. If you pick a team name that already exists, you may need to edit the Team ID because all IDs must be unique.



1. Click Done.
2. You'll land on your team Repositories page. From there, click Settings.
3. Click User groups under Access management. You'll see that you have only two groups and groups.
4. Click the Developers group. From there, you'll see a place to add new members.
5. Search for breezycloud in the Username or email address field, pick that user, and click Add.



Now when you create a pull request for your future repository, you'll have someone to review it!

**Create a repository with some content:**

No worries about putting a whole bunch of code together for this repository. We'll provide you with some to start.

1. Click + in the global sidebar and select Repository under Create.
2. Make sure the team you created is the repository Owner.
3. Enter anything you want for the Project name and Repository name. If you can't think of anything, how about Welcome package and First impressions, respectively.
4. From Include a README?, select either of the Yes options.
5. From Version control system, pick an option for the type of repository you want to create. If you're not sure, keep as is.
6. Click Create repository and you'll land on the Source view of your brand, new repository.
7. From Source, select > Add file.
8. Name the file survey.html, then copy this code and paste it into the main text area.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <style media="screen" type="text/css">  body {  margin: auto;  width: 700px;  color: #FFFFFF;  font-family: Arial, sans-serif;  background-color: #172B4D;  }  body>h1 {  margin: 50px;  font-size: 50px;  text-align: center;  color: #0052CC;  }  </style>  </head>  <body>  <h1>Team up in space</h1>  <p>  Welcome to the team! You've made it this far so we know that you've got the potential to do great things. Because you're going to be collaborating with other awesome people, anything you add needs to be code reviewed and approved. That's just how a team works! You should have already created a branch and checked it out locally. If you haven't, go back to the tutorial and do that now. We'll be here.  </p>  <p>  Because you're on your own branch, you can go crazy. Spice up this file any way you like. Add more files to this repository if you see fit. If want to take it slow and are just here to learn about pull requests, you can use this opportunity to fill out our short questionaire.  </p>  <br>  <p>  <b>Question 1</b>: Have you used pull requests before?  </p>  <p>  <b>Answer 1</b>: \*\*\*\* Your answer here \*\*\*\*  </p>  <p>  <b>Question 2</b>: Why do you want to learn about code review?  </p>  <p>  <b>Answer 2</b>: \*\*\*\* Your answer here \*\*\*\*  </p>  <p>  <b>Question 3</b>: Who do you plan to work with on Bitbucket?  </p>  <p>  <b>Answer 3</b>: \*\*\*\* Your answer here \*\*\*\*  </p>  </body>  </html> |

1. Click Commit and then commit again from the dialog.

Your repository is looking pretty good now. Take a look around if you feel like it. If you click Settings and go to the User and groups access page, you'll see that your breezycloud teammate you added has repository access through the Developers group.

**Clone and make a change on a new branch:**

In a typical team context, you'd most likely already have the repository cloned before creating a branch. So that's what we're going to do first before you set up your own branch.

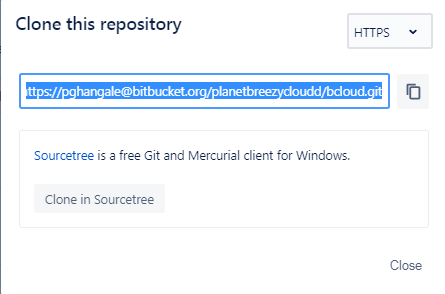
If you're using command line

**Clone your repository to your local system:**

Let's get it onto your local system so that you can really start working on it.

1. From the repository, click the Clone button in the top right.

Bitbucket displays the Clone this repository dialog. By default, the clone dialog sets the protocol to HTTPS or SSH, depending on your settings. As a result, you don't need to change your default protocol.



1. Copy the clone command.
2. From a terminal window, change into the local directory where you want to clone your repository.
3. Paste the command you copied from Bitbucket, for example:

|  |
| --- |
| $ **git clone https://pghangale@bitbucket.org/planetbreezycloudd/Bcloud.git**  Cloning into 'Bcloud'...  remote: Counting objects: 6, done.  remote: Compressing objects: 100% (4/4), done.  remote: Total 6 (delta 0), reused 0 (delta 0)  Unpacking objects: 100% (6/6), done.  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST |

**Create a branch and pull in locally:**

Now that your repository is all set up, next comes the fun part. You can create branches locally or through Bitbucket. Let's create one from Bitbucket for the purposes of this tutorial.

1. Click Branches from the left navigation. You'll see that you already have one branch — your main branch, master.
2. Click Create a branch in the top right corner.
3. After you create a branch, you need to check it out on your local system. Bitbucket provides you with a fetch and checkout command that you can copy and paste into your command line, similar to the following:

|  |
| --- |
| $ **git fetch && git checkout my-updates**  From https://bitbucket.org/planetbreezycloudd/Bcloud  \* [new branch] my-updates -> origin/my-updates  Switched to a new branch 'my-updates'  Branch 'my-updates' set up to track remote branch 'my-updates' from 'origin'.  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/Bcloud (my-updates) |

As you can see, you've switched to your new branch locally, allowing you to work on and push that separate line of code.

**Make a change to the branch:**

1. Open the survey.html file (or whatever you named it) with a text editor.
2. Make your changes, big or small, and then save and close the file.
3. From your terminal window, you should still be in the repository directory unless you've changed something. Display the status of the repository with git status. You should see the survey.html file you modified. If you added or modified other files, you'll see those as well.

|  |
| --- |
| $ **git status**  On branch my-updates  Your branch is up to date with 'origin/my-updates'.  nothing to commit, working tree clean  pghangale@ACEINLTPTI05 MINGW64 /d/bitbucket/TEST/Bcloud (my-updates) |

1. Add your changes locally with git add <filename>:

|  |
| --- |
| **git add survey.html** |

1. Commit your changes locally with git commit -m "your commit message":

|  |
| --- |
| $ **git commit -m "Answered Questions"** |

1. Enter git push origin <branch\_name> to push the changes to your branch on Bitbucket, and enter your password to finish pushing changes.

|  |
| --- |
| **$ git push origin my-updates** |

1. From Bitbucket, click the **Source** page of your repository. You should see both branches in the dropdown. Any other commits you make to my-updates will also appear on that branch.