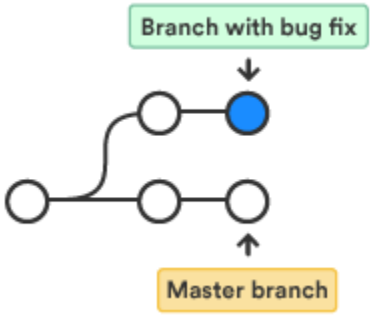
Use a Git branch to merge a file

Being a space station administrator comes with certain responsibilities. Sometimes you'll need to keep information locked down, especially when mapping out new locations in the solar system. Learn about branches to update your files separately from the main source and only share your changes when you're ready.

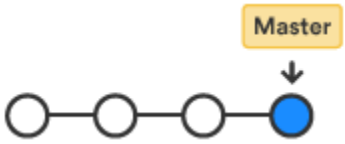
A branch represents an independent line of development for your repository. Think of it as a brand-new working directory, staging area, and project history. Before you create any new branches, you automatically start out with the main branch (called `master`). For a visual example, this diagram shows the `master` branch and the other branch with a bug fix update.



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, pulling updates from `master` if necessary, and then you'll merge all your work into `master` when you're ready. Our documentation includes more explanation of why you would want to use branches.

Step 1. Create a branch and make a change

Create a branch where you can add future plans for the space station that you aren't ready to commit. When you are ready to make those plans known to all, you can merge the changes into your Bitbucket repository and then delete the no-longer-needed branch.
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. Before you begin, your repository looks like this:



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:

   macOS / Linux / Git Bash

   ```
   $ cd ~/repos/bitbucketstationlocations/
   ```

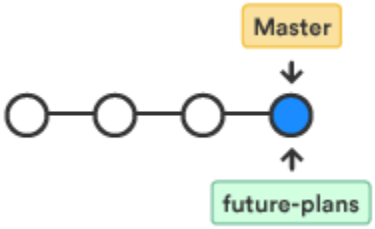
   Windows Command Prompt

   ```
   $ cd repos\bitbucketstationlocations\
   ```

2. Create a branch from your terminal window.

   ```
   $ git branch future-plans
   ```

   This command creates a branch but does not switch you to that branch, so your repository looks something like this:

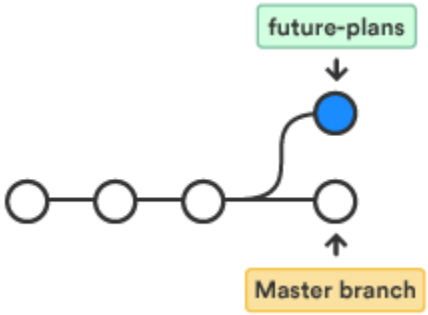
   The repository history remains unchanged. All you get is a new pointer to the current branch. To begin working on the new branch, you have to check out the branch you want to use.

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

   ```
   $ git checkout future-plans
   Switched to branch 'future-plans'
   ```

   The `git checkout` command works hand in hand with `git branch` . Because you are creating a branch to work on something new, every time you create a new branch (with `git branch`), you want to make sure to check it out (with `git checkout`) if you're

going to use it. Now that you've checked out the new branch, your Git workflow looks something like this:



4. Search for the `bitbucketstationlocations` 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.
5. Open the `stationlocations` file using a text editor.

6. 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>
```

12. Save and close the file.

13. 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 <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
    modified: stationlocations
no changes added to commit (use "git add" and/or "git commit -a")
```

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.

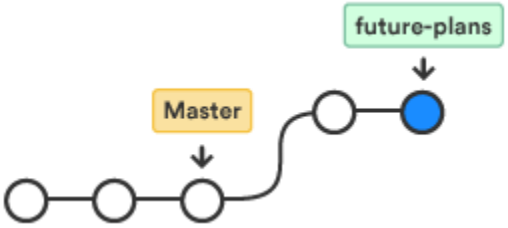
14. Stage your file.

```
$ git add stationlocations
```

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

```
$ git commit -m 'making a change in a branch'
[future-plans e3b7732] making a change in a branch
 1 file changed, 4 insertions(+)
```

With this recent commit, your repository looks something like this:

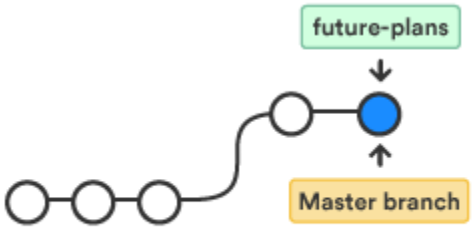


Now it's time to merge the change that you just made back into the `master` branch.

Step 2. Merge your branch: fast-forward merging

Your space station is growing, and it's time for the opening ceremony of your Mars location. Now that your future plans are becoming a reality, you can merge your `future-plans` branch into the main branch on your local system.

Because you created only one branch and made one change, use the fast-forward branch method to merge. You can do a fast-forward merge because you have a linear path from the current branch tip to the target branch. Instead of "actually" merging the branches, all Git has to do to integrate the histories is move (i.e., "fast-forward") the current branch tip up to the target branch tip. This effectively combines the histories, since all of the commits reachable from the target branch are now available through the current one.



This branch workflow is common for short-lived topic branches with smaller changes and are not as common for longer-running features.

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.

   macOS / Linux / Git Bash

   ```
   $ cd ~/repos/bitbucketstationlocations/
   ```

   Windows Command Prompt

   ```
   $ cd repos\bitbucketstationlocations\
   ```

2. Enter the `git status` command to be sure you have all your changes committed and find out what branch you have checked out.

   ```
   $ git status
   On branch future-plans
   nothing to commit, working directory clean
   ```

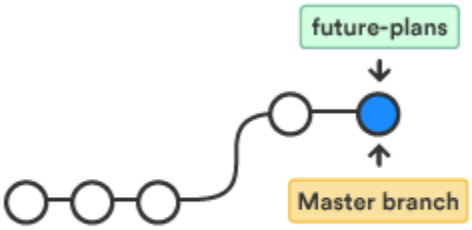
3. Switch to the `master` branch.

```
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
```

4. M erge changes from the `future-plans` branch into the `master` branch. It will look something like this:

   ```
   $ git merge future-plans
   Updating fcbeeb0..e3b7732
   Fast-forward
    stationlocations | 4 ++++
    1 file changed, 4 insertions(+)
   ```

   You've essentially moved the pointer for the `master` branch forward to the current head and your repository looks something like this:



5. Because you don't plan on using `future-plans` anymore, you can delete the branch.
   ```
   $ git branch -d future-plans
   Deleted branch future-plans (was e3b7732).
   ```

   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.

6. 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)
   nothing to commit, working directory clean
   ```

Here's what you've done so far:

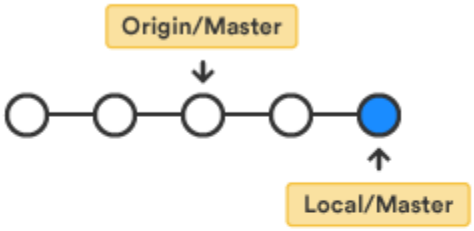
- Created a branch and checked it out
- Made a change in the new branch
- Committed the change to the new branch
- Integrated that change back into the main branch
- Deleted the branch you are no longer using.

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

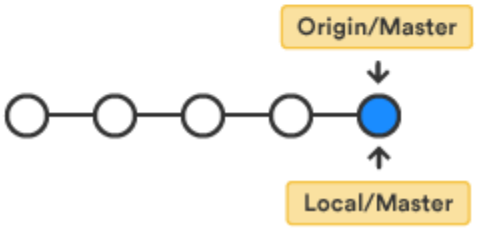
Step 3. Push your change to Bitbucket

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.
This diagram shows what happens when your local repository has changes that the central repository does not have and you push those changes to Bitbucket .

Before pushing



After pushing



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
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 401 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://emmap1@bitbucket.org/emmap1/bitbucketstationlocations.git
   fcbeeb0..e3b7732 master -> master
```

2. Click the **Overview** page of your Bitbucket repository, and notice you can see your push in the **Recent Activity** stream.
3. 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.

Author	Commit	Message
Emma Paris	c7cf7b7	making a change in a branch
Emma Paris	fccfa6f	stationlocations created online with Bitbucket
Emma Paris	5424bbf	Initial commit

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.

4. Click **Branches** and notice that the page has no record of the branch either.
5. Click **Source**, and then click the `stationlocations` file.
   You can see the last change to the file has the commit id you just pushed.

6. Click the file history list to see the changes committed for this file, which will look similar to the following figure.