# Start configuring and initializing

So that you can get credit (or blame) for the work that you've contributed, you need to associate your name and email address with your work. To do that in Git, you use the git config command.

git config --global user.name "Sample User"

git config --global user.email sample@example.com

git config --global core.pager cat

Now you're all set to start contributing your code.

In the terminal window, make sure you are in the **src** folder:

Step 1 : You need to associate your name and email address with your work

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ git config --global user.name Saby2002

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ git config --global user.email sabyasachikar24@gmail.com

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ git config --global core.pager cat

Step 2 : Make a directory called **git-intro** and change directory into it:

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ pwd

/c/Users/sakar

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ cd ~/src

bash: cd: /c/Users/sakar/src: No such file or directory

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ mkdir desktop/git-intro

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ cd git-intro

bash: cd: git-intro: No such file or directory

sakar@N-20HEPF0Z6AE0 MINGW64 ~

$ cd desktop/git-intro

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro

$ ls

Step 3 : Next, initialize your first repository:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro

$ git init

A message similar to the following one should display:

Initialized empty Git repository in C:/Users/sakar/Desktop/git-intro/.git/

You have created a local repo within your project contained in the hidden directory .git. This is where all of your change history is located.

# Try the git status command

As you work on your project, you will want to check to see which files have changed. This is helpful when you are committing files to the repo, and you don't want to commit all of them.

Still in the **git-intro** directory, type the command git status.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git status

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

This message tells you:

1. That you are on branch master (we'll discuss branches later).
2. That the commit message is Initial commit.
3. That there is nothing changed to commit.

You will see that the status of your repo changes once we add files and start making changes.

# Add files using the git add command

We have an initialized, but empty repository. In our source directory, ~/src/git-intro, create a file DEVASC.txt:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ echo "I am on my way to pass the Cisco DEVASC exam." > DEVASC.txt

Then list the files in the directory and make sure the DEVASC.txt file was successfully created:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ ls

DEVASC.txt

Type git status again.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

DEVASC.txt

nothing added to commit but untracked files present (use "git add" to track)

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

Notice that git found the new file in the directory, and knows that it's not tracked.

Next, we need to add DEVASC.txt to "stage" your work, and then "commit" your work. Staging is an intermediate phase of the process, providing the opportunity to gather your changes before you "commit" them to the repository.

You can stage multiple changes to the same file before committing, so you can organize your work. As you can see in the status, you need to execute git add <file> in order to stage your work.

Let's do that next git add DEVASC.txt.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git add DEVASC.txt

warning: LF will be replaced by CRLF in DEVASC.txt.

The file will have its original line endings in your working directory

Execute git status again.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: DEVASC.txt

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

You'll notice that there are now "staged" changes in the form of new file: DEVASC.txt.

# Let Git know you want to track current changes with the git commit command

Now that you've staged your changes, you'll need to commit them in order to let Git know you want to start tracking those changes. Commit your changes by executing git commit -m "Added my wish".

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git commit -m "Added my wish"

[master (root-commit) 646d29b] Added my wish

1 file changed, 1 insertion(+)

create mode 100644 DEVASC.txt

The -m "<message>" switch enables you to add a message explaining the changes you've made. For now, we'll just use a simple message. (Later, when you get into hard-core projects, you'll find that there are those who have ... strong opinions on what a commit message should look like.)

You will see git reply with your commit message and additional information:

Note that the number and letter combination 6a6ee56 contained in [master (root-commit) 6a6ee56]. Every commit is identified by a unique SHA1 hash, so of course you will have a different commit ID than what you see here.

[master (root-commit) **646d29b**] Added my wish

# Try the git log command to see what has been tracked

In some cases, you may want to review the commit history for a repo. To view the most recent commits, you use the command git log. Try that now.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git log

commit 646d29b5d3b1088f5f842d1991a65c587572874f (HEAD -> master)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:32:52 2020 +0530

Added my wish

You should see a commit ID followed by the Author and Date information.

In our previous step, we looked at the commit hash that git presented to us: 6a6ee56. You should notice that 6a6ee56 is the first 7 characters of the full commit hash above: 6a6ee56a5d1e5c466a299eacff1bf36c98e44d71.

That may seem like a lot of work just to make a commit, but in time it will become routine for you. Additionally, there are many git GUI clients that can automate many of these steps.

Moving forward, let's make a change to DEVASC.txt. Back in your text editor, add a new line to your file, that says: At this rate I will be ready for the exam in no time.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ echo "At this rate I will be ready for the exam in no time.">>DEVASC.txt

Now when you perform git status you will see that there are changes in your file.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: DEVASC.txt

no changes added to commit (use "git add" and/or "git commit -a")

You need to stage your file again, by executing git add, followed by another git commit:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git add DEVASC.txt

warning: LF will be replaced by CRLF in DEVASC.txt.

The file will have its original line endings in your working directory

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git commit -m "Add observation"

[master eccab85] Add observation

1 file changed, 1 insertion(+)

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git log

commit eccab85ee472da40f3c303d5d46939c5719d2c3b (HEAD -> master)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:38:08 2020 +0530

Add observation

commit 646d29b5d3b1088f5f842d1991a65c587572874f

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:32:52 2020 +0530

Added my wish

Now we have two commits, and we can compare one to the other.

# Compare two commits with the git diff command

Now that you have a few entries in the log, you can review changes between commits. You can do that with the command git diff <commit> <commit>. Use the git log command to find two commits to compare, and then copy the first seven characters from the long ID.

Paste in and replace the commit hashes with the hashes of your own commits and run the git diff like below:

git diff eccab85 646d29b

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git diff eccab85 646d29b

**diff --git a/DEVASC.txt b/DEVASC.txt**

**index d3ec66f..66a169e 100644**

**--- a/DEVASC.txt**

**+++ b/DEVASC.txt**

@@ -1,2 +1 @@

I am on my way to pass the Cisco DEVASC exam.

-At this rate I will be ready for the exam in no time.

# Splitting work into branches with the git branch command

Let's say you have a new feature that you're working on. You might have to refactor some of your code, and you want to be able to checkin your changes as you make progress. However, you don't want to box yourself in a corner and prevent easy changes to your production code. In order to manage this type of situation, you need to isolate your changes, and so you should use git branching.

Branching is pretty simple, you just execute the command git branch <branch name>. What that does, is to allow you to make changes in an area that won't affect the master branch. One convention in git is that the main branch is typically named master.

Now you're basically operating in a parallel universe until you merge your changes back into master. Let's create a new branch called exam.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git branch exam

In order to see all the branches that have been created, you can use git branch.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git branch

exam

\* master

The output of git branch in our case should look like the following:

You can see that we have 2 branches: exam and master. Now you need to switch to your branch. To switch between branches you can use git checkout <branch name>.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git checkout exam

Switched to branch 'exam'

Add some text to your DEVASC.txt file with some exam related musings.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (exam)

$ echo "I am a git master!" >>DEVASC.txt

Add and commit your change in this branch. , and then checkout the master branch.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (exam)

$ git add DEVASC.txt

warning: LF will be replaced by CRLF in DEVASC.txt.

The file will have its original line endings in your working directory

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (exam)

$ git commit -m "Added a commit in the exam branch"

[exam afc8b63] Added a commit in the exam branch

1 file changed, 1 insertion(+)

Once you have done that, checkout the master branch and then merge the content from the exam branch.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (exam)

$ git checkout master

Switched to branch 'master'

The output of that command should be similar to the following output:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git merge exam

Updating eccab85..afc8b63

Fast-forward

DEVASC.txt | 1 +

1 file changed, 1 insertion(+)

Open DEVASC.txt and you'll see that the merge was successful. You can delete your branch by doing the following:

Branches are often used when implementing new features or hot fixes. They can be submitted for review by team members, and then once verified, can be pulled into the main codebase.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git branch -d exam

Deleted branch exam (was d7ea076).

# Handling merge conflicts

At some point, you're going to have a merge conflict. This is where you may have made overlapping changes to a file, and git can't automatically merge the changes.

Go ahead and create a new branch named test and switch to it.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git branch test

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git checkout test

Switched to branch 'test'

Make a change to DEVASC.txt and replace master in I am a git master! with novice.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (test)

$ sed -i 's/master/novoice/' DEVASC.txt

Commit your change.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (test)

$ git commit -a -m 'modified wish'

warning: LF will be replaced by CRLF in DEVASC.txt.

The file will have its original line endings in your working directory

[test 6700e42] modified wish

1 file changed, 1 insertion(+), 1 deletion(-)

Now switch back to the master branch.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (test)

$ git checkout master

Switched to branch 'master'

Make a change to the same line by modifying master in I am a git master! to beginner this time and commit your change

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ sed -i 's/master/beginner/' DEVASC.txt

Try to merge the branches with git merge test:

You should see a message like the following one:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git commit -a -m 'changed master to beginner'

warning: LF will be replaced by CRLF in DEVASC.txt.

The file will have its original line endings in your working directory

[master d753a53] changed master to beginner

1 file changed, 1 insertion(+), 1 deletion(-)

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ git merge test

Auto-merging DEVASC.txt

CONFLICT (content): Merge conflict in DEVASC.txt

Automatic merge failed; fix conflicts and then commit the result.

If you view the DEVASC.txt file:

This is helping you see where the HEAD version (aka master in this case) is conflicting with the test branch version.

You have to fix this manually, generally you'd use a merge tool, but we can do this using the vim text editor. Start editing the DEVASC.txt file with vim.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master|MERGING)

$ cat DEVASC.txt

I am on my way to pass the Cisco DEVASC exam.

At this rate I will be ready for the exam in no time.

<<<<<<< HEAD

I am a git beginner!

=======

I am a git novoice!

>>>>>>> test

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master|MERGING)

$ vim DEVASC.txt

<<<<<<< HEAD

=======

I am a git novice!

>>>>>>> test

Save your file in vim (by pressing the ESC key and then : followed by wq and ENTER), and then commit.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master|MERGING)

$ git commit -a -m 'manually merged from branch test'

[master 9a92812] manually merged from branch test

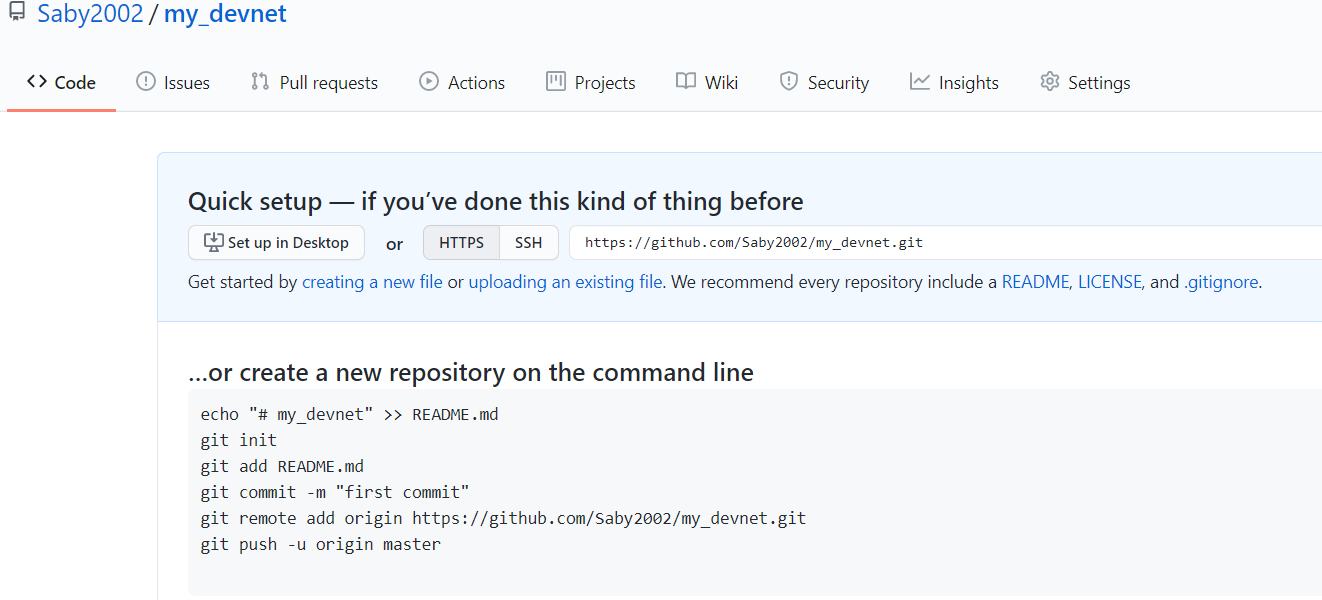
# Add a new repo in GitHub and create a remote with the git remote command

So far all the changes you have made to your file are stored on your local machine. This is great for keeping track of changes but the real advantages of Git are on display when using it as a distributed version control system. Each team member keeps a replica on their workstation. So it's easier to share changes, the repo can be stored remotely as well. A Git server can be as simple as storing your repo on a file server. There are quite a few popular Git services, including GitHub, Stash from Atlassian, and GitLab.

Because it is readily accessible, we'll use GitHub in these examples. If you don't have a GitHub account, you will need to create one. And feel free to try out Stash or Gitlab, if you prefer; while the steps may be slightly different, the concepts are the same.

On GitHub.com:

1. Log in to GitHub.
2. Find the "New repository" button or click the + icon in the upper right corner.
3. Provide a name and description of your repository.
4. Set the repository to Private or Public. You can read more about the settings in [About repositories in the GitHub help](https://help.github.com/en/github/creating-cloning-and-archiving-repositories/about-repositories).
5. Keep the default settings for everything else, so do not initialize the repo with a README file.
6. Click the "Create repository" button.



1. You should see a screen that provides you with Quick Setup information.

Back in the course terminal, in your repository, you will create a new directory called effective-octo-system. Copy the DEVASC.txt file into the directory, and then initialize Git.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro (master)

$ cd my\_DevOps/

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ cp ../DevOps.txt .

cp: cannot stat '../DevOps.txt': No such file or directory

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ cp ../DEVASC.txt .

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ ls

DEVASC.txt

Next, replace Sample User and sample@example.com with your real GitHub.com account details and make sure you copy and paste the git config commands below in the terminal.

We are doing this step because you will have to provide your real GitHub username and email for the git config --global user.name <GitHub.com username> and git config --global user.email <GitHub.com email> in this new repo.

Next, replace Sample User and sample@example.com with your real GitHub.com account details and make sure you copy and paste the git config commands below in the terminal.

We are doing this step because you will have to provide your real GitHub username and email for the git config --global user.name <GitHub.com username> and git config --global user.email <GitHub.com email> in this new repo.

git config --global user.name "Sample User"

git config --global user.email sample@example.com

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git config --global user.name Saby2002

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git config --global user.email sabyasachikar24@gmail.com

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$

After that you'll want to replace <your\_username> and <your\_repo> values with your actual GitHub.com details and then copy and paste the commands into the command line:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git remote add origin https://github.com/Saby2002/my\_devnet.git

To explain this longer command, you are adding a Git remote named "origin" that points to the newly created repository on GitHub. You can verify that you have a new remote by running the git remote command with the --verbose parameter:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git remote --verbose

origin https://github.com/Saby2002/my\_devnet.git (fetch)

origin https://github.com/Saby2002/my\_devnet.git (push)

Now, if you look at the history for this new repository in the new effective-octo-system folder, you'll see that there are no commits so Git replies with an error message.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git log

commit 9a928124ce7a5e77806e51225d7a8286ff67914f (HEAD -> master)

Merge: d753a53 6700e42

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:50:12 2020 +0530

manually merged from branch test

commit d753a531eb30e70fe6f85aa7fafc603c9ac65df1

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:49:05 2020 +0530

changed master to beginner

commit 6700e42d01e1e997e511846c92339e08bf5433dc (test)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:48:17 2020 +0530

modified wish

commit afc8b63af8b5b8ef75b84392d0edb6f881d6ad83 (exam)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:44:01 2020 +0530

Added a commit in the exam branch

commit eccab85ee472da40f3c303d5d46939c5719d2c3b

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:38:08 2020 +0530

Add observation

commit 646d29b5d3b1088f5f842d1991a65c587572874f

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:32:52 2020 +0530

Added my wish

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git add DEVASC.txt

Here is what you'll see in return, that a file has changed and that it has some lines inserted.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git commit -m "Added the initial file"

[master 0a69152] Added the initial file

1 file changed, 7 insertions(+)

create mode 100644 my\_DevOps/DEVASC.txt

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git log

commit 0a6915299ae93b1a24c98d857bf9438f1d37b7be (HEAD -> master)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Wed Aug 19 01:12:44 2020 +0530

Added the initial file

commit 9a928124ce7a5e77806e51225d7a8286ff67914f

Merge: d753a53 6700e42

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:50:12 2020 +0530

manually merged from branch test

commit d753a531eb30e70fe6f85aa7fafc603c9ac65df1

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:49:05 2020 +0530

changed master to beginner

commit 6700e42d01e1e997e511846c92339e08bf5433dc (test)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:48:17 2020 +0530

modified wish

commit afc8b63af8b5b8ef75b84392d0edb6f881d6ad83 (exam)

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:44:01 2020 +0530

Added a commit in the exam branch

commit eccab85ee472da40f3c303d5d46939c5719d2c3b

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:38:08 2020 +0530

Add observation

commit 646d29b5d3b1088f5f842d1991a65c587572874f

Author: Saby2002 <sabyasachikar24@gmail.com>

Date: Tue Aug 18 23:32:52 2020 +0530

Added my wish

And to be very sure you've committed everything and Git knows all about it, try the git status command again.

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git status

On branch master

nothing to commit, working tree clean

The phrase "working tree clean" means that Git has compared your file listing to what you've told Git, and it's a clean slate with nothing new to report.

Now, you've created that new repository on GitHub, how about sending this file to it? To do so, you will use the git push command, tell it to update the "origin" remote with the "master" branch.

Once you hit enter, you need to enter your GitHub username and password (whether the repo you created is private or public).

Once you've entered your username and password, you should see a response message like the following one:

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$ git push origin master

Enumerating objects: 20, done.

Counting objects: 100% (20/20), done.

Delta compression using up to 4 threads

Compressing objects: 100% (13/13), done.

Writing objects: 100% (20/20), 1.73 KiB | 354.00 KiB/s, done.

Total 20 (delta 5), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (5/5), done.

To https://github.com/Saby2002/my\_devnet.git

\* [new branch] master -> master

sakar@N-20HEPF0Z6AE0 MINGW64 ~/desktop/git-intro/my\_DevOps (master)

$

## Troubleshooting Git errors

What if you don't see the new branch message? There are several areas to check, and fixes for errors you may see.

**Error: remote: Invalid username or password. fatal: Authentication failed for '**<https://github.com/username/devasc-test.git/'>

If you see this error, double-check:

* Did you enter the correct username and password?
* Did you do the git config commands using your actual username and email address? Try git config -l to list the settings that Git has for you.
* Does the remote repository have a typo in it, causing Git to say you don't have access? Try git remote -v to list what "origin" is at this very moment.
* If you're using two-factor authentication, did you enter a personal access token for the password entry?

**Error: The authenticity of host 'github.com (13.114.40.48)' can't be established. RSA key fingerprint is SHA256:nThbg6...**

If you see this error, it means that you copied the SSH reference rather than the HTTPS reference when you set up the remote with the git remote add origin <https://github.com/<username>/<repo-name>.git. Check it with the git remote -v command. If you're working in the browser-based developer environment, you should remove the SSH remote and add again with the https reference. Try git remote rm origin and then git remote -v and then run the git remote add origin ... command again.

**Error: fatal: 'origin' does not appear to be a git repository**

The rest of the error message is:

fatal: Could not read from remote repository.

Please make sure you have the correct access rights

and the repository exists.

This error could mean that there is no remote set up already that is named origin. Check what the remote is named, or if it exists, with the git remote -v command. If you have nothing listed, go back to the git remote add origin ... steps and redo them.

Once you have that new branch, you've now successfully shared your code into an online repository to work with other coders!

## Git workflows

Learning Git takes time and practice, including a lot of vocabulary words that are specific to Git. With this walkthrough, you have the elements of what you need to work collaboratively using Git. Ultimately, you'll be in a cycle that looks like this:

* Work on your code
* Stage and commit your changes: git add <files> && git commit -m <your message>
* Push your changes: git push <remote> <branch>
* If others are working on the same project, you will want to pull their changes from the server, then repeat the cycle (pulling looks like git pull <remote> <branch>)

You may be wondering about <remote> <branch>, which in the example was set for origin master. If you recall, you created the remote with this command:

git remote add origin https://github.com/<your\_username>/<your\_repo>.git

Since you named it "origin" in the command, origin is the alias for the remote server.

As for master, that is the name of the branch into which you are pushing. Typically master is treated as a special branch that all collaborators want to move changes into. You can read much more about different workflows in this tutorial, [Comparing workflows](https://www.atlassian.com/git/tutorials/comparing-workflows).

## Summary

That's a lot of Git commands in one jam-packed session, but that's it, you're finished. Feel free to practice more in your own development environment.

Admin@DESKTOP-IF2O7E9 MINGW64 ~/Desktop/git-intro/my\_DevOps (master)

$ ls -al

total 5

drwxr-xr-x 1 Admin 197121 0 Aug 20 11:45 ./

drwxr-xr-x 1 Admin 197121 0 Aug 20 11:51 ../

-rw-r--r-- 1 Admin 197121 162 Aug 20 11:45 DEVASC.txt

Admin@DESKTOP-IF2O7E9 MINGW64 ~/Desktop/git-intro/my\_DevOps (master)

$ git init

Initialized empty Git repository in C:/Users/Admin/Desktop/git-intro/my\_DevOps/.git/

Admin@DESKTOP-IF2O7E9 MINGW64 ~/Desktop/git-intro/my\_DevOps (master)

$ ls -al

total 9

drwxr-xr-x 1 Admin 197121 0 Aug 20 12:10 ./

drwxr-xr-x 1 Admin 197121 0 Aug 20 11:51 ../

drwxr-xr-x 1 Admin 197121 0 Aug 20 12:10 .git/

-rw-r--r-- 1 Admin 197121 162 Aug 20 11:45 DEVASC.txt