Git 102: Using git with servers

https://developer.cisco.com/learning/lab/git-servers/step/1

Introduction

This Learning Lab provides you with an introduction to using Git servers like GitHub.

Prerequisites

- A GitHub account, which you can create for free at https://github.com.
- Completion of the Git Intro Learning Lab.
- Completion of the Git Branching Learning Lab.
- A Git client on your workstation. If you are a DevNet event - such as at CiscoLive! - the workstations are pre-installed with `git`.

Git servers

Git was designed to be a distributed and decentralized version control system. Each team member keeps a replica of the code repository on their workstation. To enable users to share changes, a repo can also be stored remotely.

A Git server can be as simple as storing a repository on a file server. But a Git server can exist in several forms using Git protocol, HTTP, and SSH access to the remote Git repo. We will not cover all of these options in this tutorial.

GitHub, Bitbucket, Gitlab

There are many popular Git services out there, including GitHub, Bitbucket from Atlassian, and Gitlab. There are commonalities among them, and so we will cover the main use cases. Because it is readily accessible, we'll use GitHub in these examples, but don't take this as an endorsement per se. Use the service that suits your needs.

Next: Creating a remote repository

Creating a remote repository

If you don't have a GitHub account, you need to create one. And feel free to try out Bitbucket or Gitlab, if you prefer. While the steps may be slightly different, the concepts are the same.

To create a repository on GitHub:

1. Click the **New repository** button.
2. For the repository name, enter `git-intro`. Enter any description that you like. Do not initialize the repo with a README.
3. Click **Create repository**. You should see a screen that provides you with Quick Setup information.

   In the Git Intro lab, you created a new repository locally. Note the following commands provided by GitHub:

```
git remote add origin git@github.com:<your_username>/git-intro.git
git push -u origin main
```

4. Back on the workstation, in your local `git-intro` repository, copy and paste those commands into the command line.

   Once you press Enter, you should see a response message similar to the following:

```
Counting objects: 9, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 782 bytes | 0 bytes/s, done.
Total 9 (delta 2), reused 0 (delta 0)
To git@github.com:<your_username>/<your_repo>.git
* [new branch]      main -> main
Branch main set up to track remote branch main from origin.
```

You've now successfully shared your local code into a remote repository!

Next: Stage, commit, push, pull

Stage, commit, push, pull

Now that you've pushed your code, you have the elements of what you need to work collaboratively using Git. A collaborative development cycle usually includes the following stages:

1. Work on your code
2. Stage and commit your changes: `git add <files> && Git commit -m <your message>`
3. Push your changes: `git push origin main`

4. If others are working on the same project, pull their changes from the server with `git pull origin main`

You may be wondering about `origin main`. If you recall from the previous step, we created the remote with:

```
git remote add origin git@github.com:<your_username>/<your_repo>.git
```

You can see `origin` is an alias for the remote server, and `main` is the name of the branch into which you are pushing or pulling.

Next: Pull requests

Pull requests

Git uses concept of pull requests to submit repository changes for review and approval before merging them.

For example, you may identify a bug in a code repository owned by another user. In other version control systems, you would submit a patch to the repository owner. With Git you can submit a pull request.

Git has the built-in feature `git request-pull`, but since we're working on GitHub, we will use GitHub's version.

Note: In this exercise, you will be creating and merging a pull request against your own repository. In standard development scenarios, you will create pull requests against other users' repositories, and you should not merge your own pull requests unless the owner has given you permission to do so.

Creating a pull request

Pull requests can be used with branches or with forks. You will learn about forks in the next section. Follow these steps to create a pull request with a branch.

1. In your local `git-intro` repository, ensure that you are on the `main` branch and that it is up to date with the remote server.

```
2. $ git checkout main
3. $ git pull origin main
```

4. Create a new branch called `shakespeare`.
5. Create a file called `second.txt` and add the following text to it:

```
6.  All the world's a stage,
7.  And all the men and women merely players:
8.  They have their exits and their entrances;
9.  And one man in his time plays many parts,
10. His acts being seven ages.
```

11. Commit the change.
12. Push the shakespeare branch to the server.

```
13. $ git push origin shakespeare
```

14. In your browser, navigate to the repository page. You should see a notice that says shakespeare had recent pushes less than a minute ago. If you do not see this notice, click on **Branches** and find the shakespeare branch in the list.

    Click **Compare & pull request** or **New pull request**. You will now see the **Open a pull request** page.
15. Enter a brief description for the pull request and click **Create pull request**.

Git creates your pull request, and you can now view it.

Merging a pull request

When a pull request is created, GitHub will indicate whether the branch code has any conflicts with the base code. If there are none, you will see a green **Merge pull request** button. Click that button to merge the code from shakespeare into main.

If you view the code in the remote repository, you will now see second.txt in the repository.

Forking a repository

Another method of creating a pull request is to create a fork of a remote repository. For a tutorial on forking, refer to GitHub's Forking Projects guide.

Summary

Good job! You have learned some of the fundamentals of Git in a server environment, including:

- How to create a repo on GitHub
- How to fork a repo
- How to create a pull request