Module 3: Git and GitHub

As you learn to code, and as we strive to make our, and your, AI explorations as open and reproducible as possible, we also believe that it is important to use industry-standard tools to improve your own coding practices and to share code, data, and AI models–part of ethical AI is open and reproducible code! Version control systems provide indispensable tools that programmers and data scientists rely on for daily work. Version control systems also make it easy to share code and AI models. git is the leading version control software and GitHub.com is one of the main online hosting sites for git repositories (though there are others). We have built *Practicum AI* using git and GitHub and feel that understanding these tools is so fundamental to learning AI that we start using GitHub even before learning Python!

Module 3 Objectives:

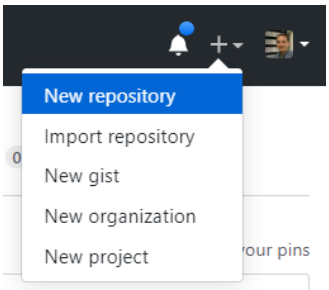
a. Navigate GitHub, including cloning and creating repositories and working with them in a Jupyter environment.
b. Use a git workflow to edit a file, add the file, commit the changes, and push changes to a remote repository.
c. Navigate the GitHub discussion board and issue system for getting help and reporting problems.
d. Explain why version control is an important part of AI development.
    i. Understand some ethical considerations concerning version control in software development.
    ii. Understand some ethical considerations concerning code transparency and reproducibility.
e. Understand what SSH keys are, and why they are important.

Step 1: Create your own Git repository

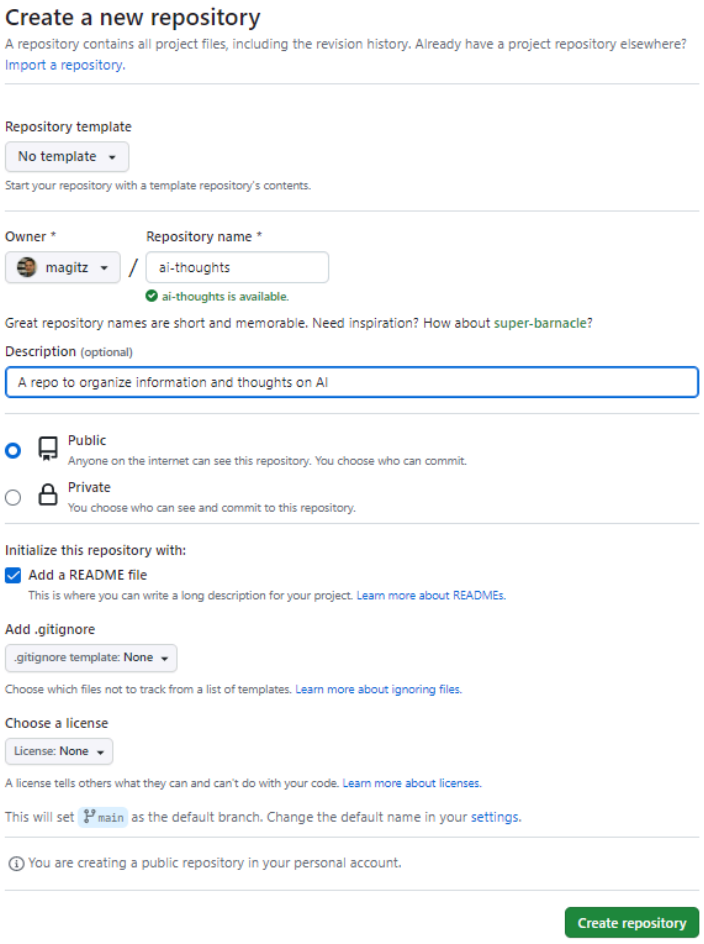
In Module 2, you learned how to create a repository from a template. The *Practicum AI* team had created the repository and set it to be a template for you to be able to easily create your own copy to use. In this exercise, we want to start from scratch, with you creating your own repository on GitHub. For this exercise, you will start a repository for keeping information on your AI journey.

We will start by creating a repo (repository is a really long word…let's call it a repo from now on). While there are many ways to create a repo, one of the easiest is on the GitHub.com site.

- Navigate to https://github.com/ and login if needed.
- To create a new repository, either click the New button on the left, or the +-Dropdown menu on the right in the top menu bar and select "New repository".
- Using the image below as a guide:

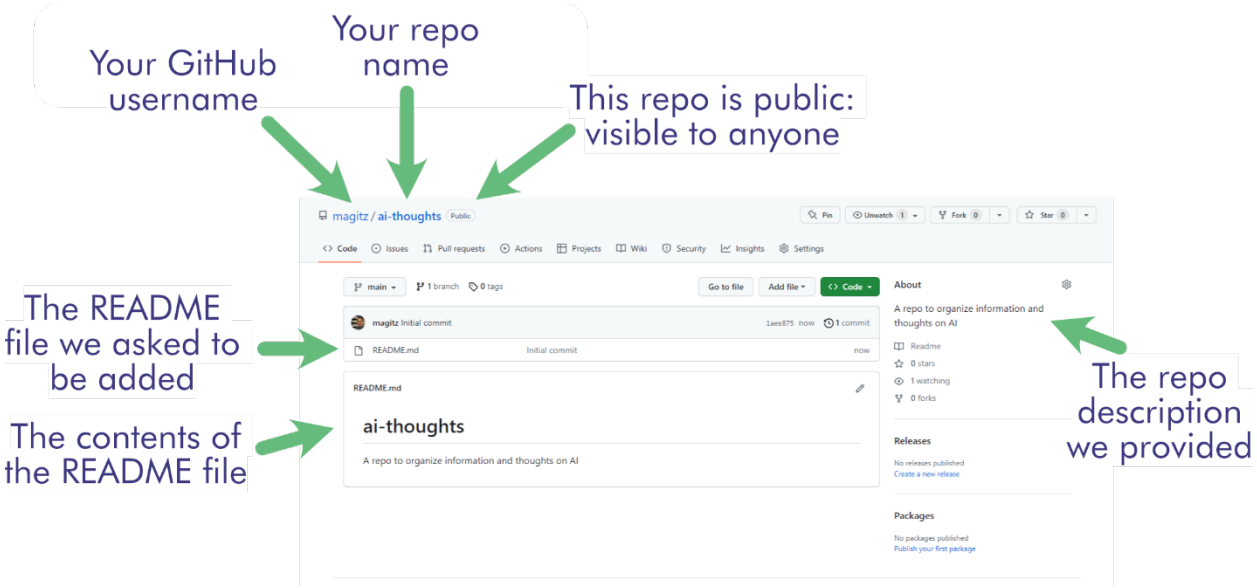


- Select the Owner if needed
- **Repository name** can be just about whatever you want, for this exercise, we suggest: **ai-thoughts**
- **Description**: Add a brief description of what this repo will be used for. This can always be edited later, but something like: **A repo to organize information and thoughts on AI**
- **Public or Private**: Sharing a Public repository is much simpler than sharing a Private one (you can share Public repositories with just the site link). Unless the repository needs to be Private, consider keeping them Public (the GitHub default setting).



- **Check the "Add a README file" checkbox.**
- **Click the "Create Repository" button**

- Explore the layout of the GitHub repo page. There is lots of information, and we will look at more details later, but notice a few things:
  - The top bar has linked information about the repo owner and name. It also shows if the repo is public (open to anyone to view) or private (limited to certain users).
  - Because we checked the "Add a README file" checkbox, the repo has a file called README.md and its contents are displayed. Any guesses on what the ".md" file extension is?
    - The ".md" file extension means the file is a Markdown file. Yes, the same Markdown that we learned a little about in Jupyter Markdown cells in Module 2.
    - Every repo should have a README.md file, and that file should describe what the repo is for.
    - The rendered version of the Markdown will be displayed below the file list on the repo's site. This is a useful way to let people know what they should expect to find in your repo!



- We will return to this repo later, but first, we need to set up your GitHub account to be able to work with the system you are using.

Step 2: Configure GitHub Account for Remote Access

Using Atlas requires that you have a SCINet account.

Note: You will need a GitHub account for this. You can create your account at https://github.com/ Click the **Sign up** button.

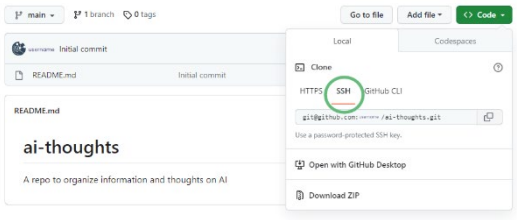
Please note that for official USDA work, you may *not* use a free GitHub account. Instead, you must use USDA's GitHub Enterprise Cloud platform. Your unit can purchase GitHub Enterprise Cloud licenses through the SLIM system, which is also used for purchasing other centrally managed software. If you are only using GitHub for the Practicum AI course, you may use a free account.

- Follow the same steps you used in Module 2 to launch a Jupyter session.
- Before we clone *your* new GitHub repository, we want to do some configuration to allow you to be able to push changes that you make to the repository on Atlas back to the repository on GitHub to keep the two copies synchronized.
- Open the notebook **02_git_configuration_ATLAS.ipynb**
- Follow the directions to create SSH keys and add the public key to your GitHub account.

Step 3: Open your ai-thoughts repo and add some notes

Now that you have connected your GitHub account with Atlas, you can make changes to files in those environments and have those changes appear in GitHub.

- Revisit your ai-thoughts repository created in step 1 above. You can login at https://github.com/, and it should be listed on the left-hand side of page under Top Repositories. If not, add your GitHub username and the name of the repo ("ai-thoughts") to the URL.
- Clone your "ai-thoughts" repo, this time using the SSH link now that your keys have been configured:
    - **Click the green <> Code button**.
    - Make sure the **SSH button** is selected (not the HTTPS),
    - **Copy** the line of text in the box using the copy button



- Connect to your Jupyter session or start a new one.
- As you did in Module 1 of this course, click on the **Terminal button** and type: `git clone <paste the URI>`
- **Using the File Browser in the left hand side of JupyterLab, navigate into the ai-thoughts folder**—this is your local copy of the repository.
- **Open the README.md file**
    - A README file is used to provide information about the repo. Every repo should generally contain a README file.
    - The ".md" file extension means that this file is written in Markdown—the same syntax used in the Markdown cells of Jupyter notebooks (Markdown has become very popular as a lightweight text formatting tool).
- Let's change the title of this file to something more interesting, like "My Amazing Thoughts and Resources on Artificial Intelligence" (remember the title is the line at the top with the level 1 header indicated by the single #)
- Next add a level 2 header and a list of some things you want to do with AI. Here's an example:

```
## Things I want to do with AI

* Have fun
* Take over the world
```

- OK, let's save these changes (File > Save Markdown File)
- Click back on the **terminal.**
- If you haven't, **change directories into the ai-thoughts repo**: `cd ai-thoughts`
- Type: `git status`
- The output should look similar to this:

```
[username@Atlas-0044 ai-thoughts]$ git status
# On branch main
# 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:   README.md
#
no changes added to commit (use "git add" and/or "git commit -a")
[username@Atlas-0044 ai-thoughts]$
```

- The changes were made in the **working directory**, and git shows that the README.md file has been modified.
- To **stage** the changes, type: `git add README.md`
- If you run `git status` again, you will see that the last line about no changes added to commit is now gone.
- To **commit** the changes, type: `git commit -m "Adding initial goals"`
  - The `-m` part is the commit message. Commit messages should describe what changes you have made and should be relatively short (40 characters)
- Run `git status` one more time:

```
[username@Atlas-0044 ai-thoughts]$ git status
# On branch main
# Your branch is ahead of 'origin/main' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
```

  - Note that your branch is ahead of "origin/main" by 1 commit—that means your local repo has your new commit that has not been pushed to the origin (GitHub in our case).
- To **push** your changes, type: `git push`
- Return to your repo in GitHub and click refresh on the page.
  - You should now see the new version of your notes on AI! Congratulations! You have made your first git commit and push!

This is the typical git workflow you will use throughout the *Practicum AI* series as well as in real coding work.

1. Create a repo on GitHub—it may be empty or from one of our templates like you did in Module 2
2. Clone that repo to Atlas
3. **Add** and **commit** changes, and **push** those changes to GitHub

This keeps everything in sync and allows you to have your code where you can share it with others, collaborate, and more.

Git and GitHub have a lot more features, and GitHub has some great tutorials to learn how to use some of them. We encourage you to explore, but what you have done so far will get you a long way!

Step 4: What to do when things go wrong?

Hopefully, everything you did worked perfectly! But let's be real…things seldom work perfectly ☹.

We will cover troubleshooting code in many of our modules, but we also have a discussion board on GitHub where you can post questions, get help, share with other students, and see what others are doing with their new AI knowledge.

Vitis the Practicum AI discussion boards!

Ran into a problem? Post a question.

Have feedback to improve our content? Post in the Ideas and Feedback board

Explore the options and get ready to tackle our next course, Python for AI!