

What Are Codespaces?

GitHub Codespaces is a cloud-based development environment that allows you to code directly within your web browser or through Visual Studio Code. It provides a fully configured, containerized environment tailored to your project's needs, eliminating the hassle of local setup.

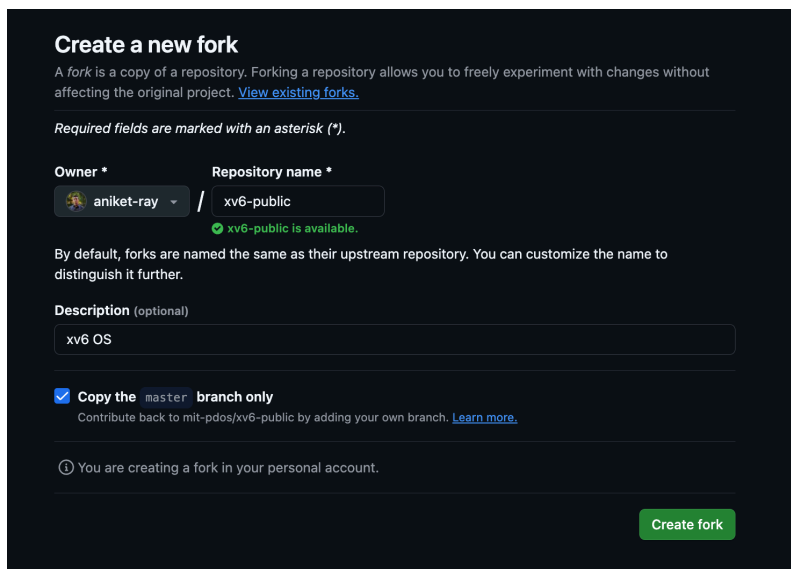
Prerequisites

1. A [GitHub account](#).
2. Access to GitHub Codespaces (available for individual accounts and organizations).

Steps to Create a Codespace

Forking xv6 from the original repository

1. Visit <https://github.com/mit-pdos/xv6-public>.
2. Fork the repository to your github account.

The image shows the 'Create a new fork' form on GitHub. At the top, it says 'Create a new fork' and explains that a fork is a copy of a repository. Below this, it states 'Required fields are marked with an asterisk (*)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' dropdown is set to 'aniket-ray'. The 'Repository name' dropdown is set to 'xv6-public', with a green checkmark indicating it is available. Below these, there is a 'Description (optional)' field with the text 'xv6 OS'. A checkbox labeled 'Copy the master branch only' is checked, with a note to 'Contribute back to mit-pdos/xv6-public by adding your own branch. Learn more.' At the bottom, there is a note: 'You are creating a fork in your personal account.' and a green 'Create fork' button.

3. Click on "Create fork".

Creating a Codespace

1. Open the link <https://github.com/codespaces>.
2. Navigate to the button "New codespace"
3. Choose the repository "<username>/xv6-public" that we forked in the previous step.

4. Keep the defaults:
Branch: master
Region: US East
Machine Type:
5. Click on "Create codespace".
6. A remote connection to the VM will be set up and a VSCode window should appear.

Running xv6 inside the codespace

1. Inside the VSCode terminal, run the following commands:

```
> $ sudo apt-get update  
> $ sudo apt-get -y upgrade  
> $ apt-get install -y qemu-system
```

2. Make sure you're inside the correct directory, i.e. `/workspaces/xv6-public`
3. Run:

```
> $ make  
> $ make qemu-nox
```

4. You should be able to access the xv6 interactive shell from the terminal.
5. Terminating xv6: `[ctrl + A] → X` [Press Ctrl+A, Release, Press X]

Running a hello-world program in xv6

This is not a graded task but it would help you understand the xv6 OS and assist you in the coming assignments. Write a program for xv6 that, when run, prints "Hello world" to the xv6 console. This can be broken up into a few steps:

1. Create a file in the xv6 directory named `hello.c`
2. Put code you need to implement printing "Hello world" into `hello.c`. All the C Standard Library header files might NOT be available in xv6. Take a look at other programs (such as: `echo.c`, `cat.c`, `wc.c`, `ls.c`) to learn the basic xv6 APIs available.
3. Edit the `Makefile` (inside the `xv6_public` directory) and the section `UPROGS` (which contains a list of programs to be built), and add a line to tell it to build your Hello World program. When you're done that portion of the Make file should look like:

```
UPROGS=\
```

```
_cat\  
_echo\  
_forktest\  
_grep\  
_init\  
_kill\  
_ln\  
_ls\  
_mkdir\  
_rm\  
_sh\  
_stressfs\  
_wc\  
_zombie\  
_hello\
```

4. Run `make` to build xv6, including your new program. You can use the same commands described above (make and make qemu-nox for building and running until you have compiled code).
5. Run `make qemu-nox` to launch xv6, and then type `hello` in the QEMU window. You should see "Hello world" printed out.

Saving your work on GitHub Codespaces

While GitHub Codespaces offers virtual machines with persistent storage, to protect your work from potential loss due to accidental deletion of the VM, we recommend using Git for version control and making regular incremental commits.

Word of Caution

When using GitHub Codespaces, be mindful of the free usage limits, which are currently set at **60 hours per month for a 2-core machine**. While this limit should be ample for this project, it's important to manage your time effectively to avoid exceeding it. To ensure you stay within your allocated hours and protect your work:

1. **Make Regular Commits:** Frequently commit your changes to keep a local record of your incremental work.
2. **Push to Your Repository:** Regularly push your commits to your GitHub repository to back up your work remotely.
3. **Shut Down the VM:** After each session, make sure to shut down your Codespace VM to stop accruing usage time.

By adhering to these practices, you not only safeguard your progress but also help prevent unnecessary consumption of your allotted Codespaces hours.