Getting started with gem5

In this section, we will get familiar with the tutorial's codespace environment and run our first gem5 simulation.

Let's hit the ground running

This example will show

1. How someone obtains gem5.
2. How you build it.
3. Running a very basic "Hello World" simulation.

- Getting and compiling gem5 is often the hardest part.
- There's a lot of complicated things happening behind the scenes. We will explain them later.

Typical Downloading

gem5 is not your typical software project where you can easily download a binary.
I.e., `apt install gem5` will not work.

The main way gem5 is distributed is as source code that you have to build.

```
git clone https://github.com/gem5/gem5
cd gem5
```

> There are two main branches in the gem5 repository:
> **stable**: The default branch for gem5. Updated at stable releases. Currently v24.0 (As of August 2024).
> **develop**: The branch in which new features, improvements, etc. are added regularly for the next release.

In this tutorial we're going to use codespaces with a repo which includes some example materials.
Though all the gem5 code is v24.0.

gem5 versions

On the **stable** branch, there are *tags* for each release.

We release gem5 roughly 2-3 times per year.
We don't have a strict schedule or feature or bug fix goal for releases.

The releases are named for the year and number.

E.g., the most recent gem5 release, v24.0, was the first release in 2024.

The full version string is `v24.0.0.0`.
The last two numbers are for

- Minor releases (these happen rarely when a major bug is found).
- Hotfix releases: These are for "small" bugs that are found after a release.

See CONTRIBUTING.md for more information.

Using codespaces

- We will be using the "bootcamp environment"
  - Note: That's also where the source for these slides are
  - You will be doing all of your development in the repo found at https://github.com/gem5bootcamp/latin-america-2024.

These slides and are available at https://gem5bootcamp.github.io/latin-america-2024/ for you to follow along.

> **Step 1:** Go to the classroom https://classroom.github.com/a/zIM1ysn_

You need to be in the GitHub organization (via the classroom) to get free codespaces.

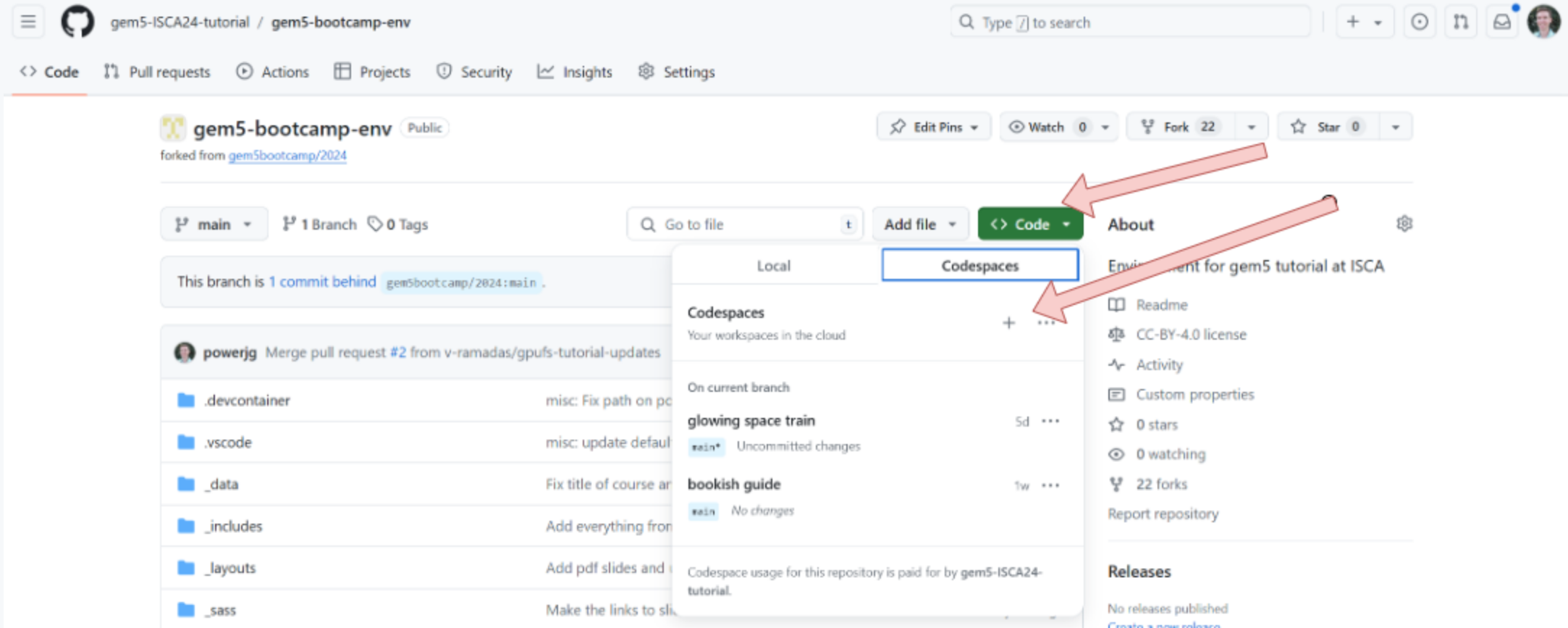
We strongly recommend using codespaces for the bootcamp.

This guarantees everyone is using the same environment and will make debugging easier.

Using codespaces 2

AFTER joining the classroom, you can go to the repository and click on the green "Code" button. Again, note that this is the repo where the slides are.

https://github.com/gem5bootcamp/latin-america-2024/



Using codespaces 3

Step 3: Wait for the environment to load.

You can also open it in your local VS Code if you install the Codespaces extension.
(If you do this, the extensions will not be installed automatically on your local VS Code.)

Navigating the repository

- `gem5/`
  - The gem5 source code (v24.0). A sub-repository
- `gem5-resources/`
  - Source code for gem5's resources (workloads, disks, etc.). Also a sub-repository
- `slides/`
  - Markdown version of these slides. Used to build the website/slides.
  - You can also preview the slides in VS Code.
- `materials/`
  - Python scripts and other materials for the class examples.
  - Completed examples are in the `completed` directories.
- `homework/`
  - Homework assignments to be completed outside of the workshop.

Both the slides and materials are broken down into sections and lessons.
We use numbering to keep them in order.

Building gem5

- To build the gem5 binary you need to use SCons. The following code is an example of how you would build the optimized version of gem5.
  - SCons is a software construction tool like Make that makes it easy to put together libraries and environments.

> Don't do this right now!

```
scons build/ALL/gem5.opt -j [number of cores]
```

- This takes a while (10-15 minutes with 16 cores, ~1hr on 1 core).
- If you're using codespaces, we have prebuilt binaries for you.
- We'll talk more about the build system and options later.

<script src="https://asciinema.org/a/6rAd24brgGqb3Sj8Kmvy1msaG.js" id="asciicast-6rAd24brgGqb3Sj8Kmvy1msaG" async="true"></script>

First Exercise

In this exercise, you will get familiar with the codespace environment and run your first gem5 simulation.

Steps:

1. Open the codespace and configure the codespace to show the slides
2. Write a simple gem5 runscript
3. Run gem5
4. Explore the output

Questions

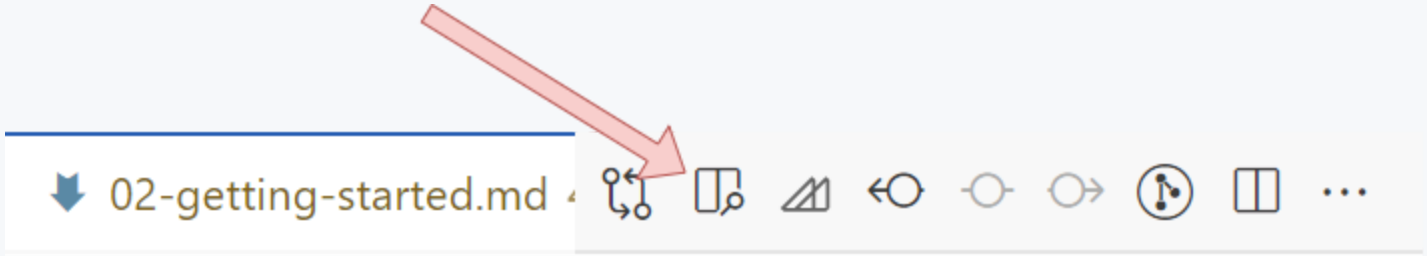
1. How can you tell that Linux began booting?
2. How many instructions executed?

Step 1: Open the codespace

While we do this, feel free to follow along in the slides.

The slides (found in `slides/01-Introduction/03-getting-started.md`) contain the code snippets we will be using. You can copy-paste from there if you get behind.

Press the "Preview" button in VS Code to see a rendered version of the slides locally.



Step 2: simulation configuration

Create a new directory: `exercises/01-Introduction/03-getting-started`.

Create a new python file: `basic.py` in the `exercises/01-Introduction/03-getting-started` directory.

Import the `X86DemoBoard` class and the `obtain_resource` and `Simulator` functions.

Step 2: Answer

```
from gem5.prebuilt.demo.x86_demo_board import X86DemoBoard
from gem5.resources.resource import obtain_resource
from gem5.simulate.simulator import Simulator
```

Step 3: Let’s be lazy and use a prebuilt board

Create an instance of the `X86DemoBoard` class.

Source is available: src/python/gem5/prebuilt/demo/x86_demo_board.py.

Step 3: Answer

```
board = X86DemoBoard()
```

The X86DemoBoard has the following properties:

- Single Channel DDR3, 2GB Memory.
- A 4 core 3GHz processor (using gem5's "timing" model).
- A MESI Two Level Cache Hierarchy, with 32kB data and instruction case and a 1MB L2 Cache.
- Will be run as a Full-System simulation.

Step 4: Load some software

Let's run a simple workload that boots Ubuntu 24.04 without systemd for x86.

You can search the workloads available on the gem5 resources page.

Use the `obtain_resource` function to download the workload.

You can find more information about the `obtain_resource` function in the gem5 source code.

Step 4: Answer

```
board.set_workload(
    obtain_resource("x86-ubuntu-24.04-boot-no-systemd")
)
```

- `obtain_resource` downloads the files needed to run workload
  - Boots Ubuntu without systemd then exits the simulation
  - Downloads disk image, kernel, and sets default parameters

See the gem5 resource page.

Step 5: Create a simulator object and run the simulation

Create a simulator object and run the simulation for 20 billion ticks (20 ms).

You can find more information about the `Simulator` class in the gem5 source code.

You should pass one parameter to the `Simulator`, which is the board you want to run.

Make sure you only run it for 20ms, otherwise it will take a long time.

Step 5: Answer

```
sim = Simulator(board)
sim.run(20_000_000_000) # 20 billion ticks or 20 ms
```

Step 6: Run the simulation

Use the `gem5-mesi` workload to run your python script.

Remember, gem5 is just a Python interpreter.

Step 6: Answer

```
from gem5.prebuilt.demo.x86_demo_board import X86DemoBoard
from gem5.resources.resource import obtain_resource
from gem5.simulate.simulator import Simulator
board = X86DemoBoard()
board.set_workload(
    obtain_resource("x86-ubuntu-24.04-boot-no-systemd")
)
sim = Simulator(board)
sim.run(20_000_000_000) # 20 billion ticks or 20 ms
```

To run it:

```
gem5-mesi basic.py
```

Step 7: Explore the output

gem5 will create a new directory called `m5out/` in the current directory.

In this directory, you will find the output of the simulation.

Step 7: Answer

gem5 has a lot of output.
It's both verbose on stdout, but also writes many files in `m5out/`.

gem5's output

In `m5out/` you'll see:

- `stats.txt`: The statistics from the simulation.
- `board.pc.com_1.device`: The console output from the simulation.
- `citations.bib`: Citations for the models and resources used.
- `config.ini/json`: The configuration file used.
- `config*.pdf/svg`: A visualization of the configuration for the system and the caches.

Exercise 1: Questions

1. How can you tell that Linux began booting?
2. How many instructions executed?

Take aways

- `gem5` is a Python interpreter.
- The *interface* to `gem5` is Python scripts.
- `gem5` contains many Python libraries.
  - All of the models in gem5 (e.g., caches, CPUs, etc.).
  - The standard library (stdlib)
- The output of gem5 is in `m5out/` by default.
  - Details of configuration
  - Other output
  - **Statistics** (the most important part)
- The Codespaces environment is configured to make things easy.
  - You'll need to do some work to set up your own environment.