Getting started with gem5

In this section, we will get familiar 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

- > 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.

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



Using codespaces

- We will be using the "bootcamp environment"
 - Note: That's also where the source for these slides are

Step 1: Go to the classroom https://classroom.github.com/...

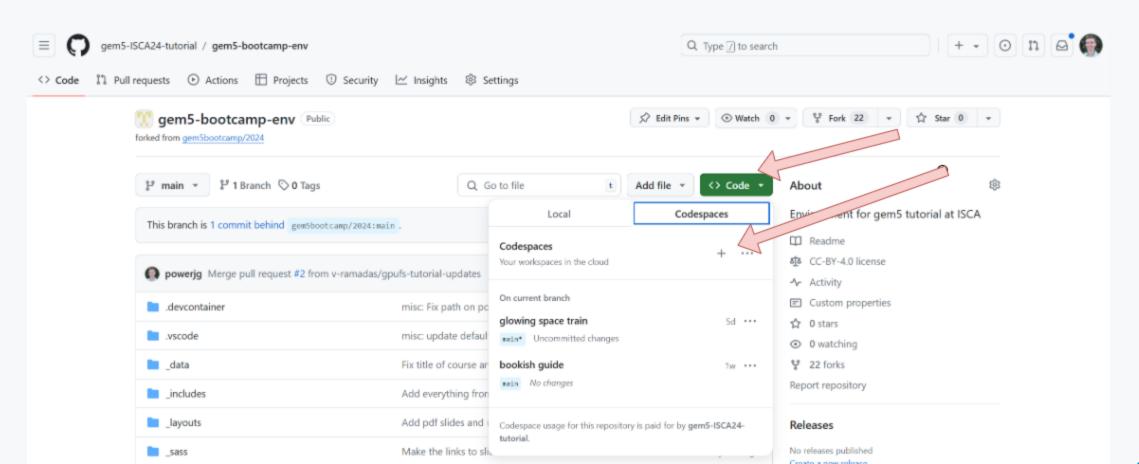
You need to be in the github organization (via the classroom) to get free codespaces.



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-2024/



Using codespaces 3

Step 3: Wait for the environment to load.

Screenshot



Building gem5

- > scons build/ALL/gem5.opt
- 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.

```
CXX] ALL/python/m5/ext/pystats/serializable_stat.py.cc -> .o
     CXX] ALL/python/m5/ext/pystats/abstract_stat.py.cc -> .o
     CXX] ALL/python/m5/ext/pystats/group.py.cc -> .o
     CXX] ALL/python/m5/ext/pystats/simstat.py.cc -> .o
     CXX] ALL/python/m5/ext/pystats/statistic.py.cc -> .0
     CXX] ALL/python/m5/ext/pystats/storagetype.py.cc -> .o
     CXX] ALL/python/m5/ext/pystats/timeconversion.py.cc -> .o
     CXX] ALL/python/m5/ext/pystats/jsonloader.py.cc -> .o
     CXX] ALL/python/m5/stats/gem5stats.py.cc -> .o
     CXX] src/python/embedded.cc -> ALL/python/embedded.o
     CXX] src/python/importer.cc -> ALL/python/importer.o
     CXX] ALL/python/m5ImporterCode.cc -> .o
     CXX] src/python/pybind11/core.cc -> ALL/python/pybind11/core.o
     CXX] src/python/pybind11/debug.cc -> ALL/python/pybind11/debug.o
     CXX] src/python/pybind11/event.cc -> ALL/python/pybind11/event.o
     CXX] src/python/pybind11/object_file.cc -> ALL/python/pybind11/object_file.o
[CONFIG H] HAVE_HDF5, 1 -> ALL/config/have_hdf5.hh
     CXX] src/python/pybind11/stats.cc -> ALL/python/pybind11/stats.o
     CXX] ALL/python/m5/objects/SimObject.py.cc -> .o
[SO Param] m5.objects.SimObject.SimObject -> ALL/python/ m5/param SimObject.cc
```

Let's start by writing a simulation configuration

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

This template code is available in the <u>materials/</u> directory. Open the materials/01-basic.py file and start editing.



Let's be lazy and use a prebuild board

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.

Source is available: src/python/gem5/prebuilt/demo/x86 demo board.py.



Let't load some software

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

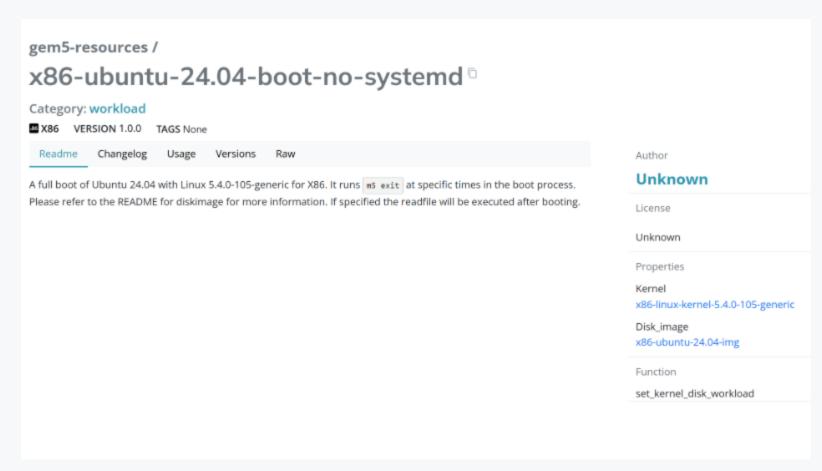
- obtain_resource downloads the files needed to run workload
 - Boots Ubuntu without systemd then exits the simulation
 - Downloads disk image, kernel, and sets default parameters

See the <u>gem5 resource page</u>



gem5 resources web portal

Link





Now, let's create a simulator to actually run

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



That's it!

```
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 materials/01-basic.py



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.

