Developing locally is always tricky; this guide seeks to provide students with multiple ways to run code in the course, lest one method not work. Please try these in the order in which they are presented, as the methods below increase in complexity. If you have another way of running things that you think should be added to this guide, let us know!

---

# 1 Local Development

To get set up for local development, check out the **Setup** section of our Guide to Golang. To validate, ensure that `go version` prints out a version of 1.13 or higher. Common issues with getting set up on Golang usually include correctly setting your PATH and GOPATH environment variables. Check these values to ensure that Go is being picked up properly by your shell.

Also make sure you have Git installed. We use Git as a version control system. Install Git here or with your package manager of choice.

Aside from getting set up on Golang and Git, you may want to consider getting familiar with a text editor. The following are options for you to peruse.

- VSCode
- Atom
- Sublime Text
- Vim
- Emacs

Note: We really like VSCode for Golang development. Golang is a language where formatting counts. The Go extension for VSCode has lots of great macros to save you time and keep you from making unnecessary mistakes.

If you are on an Apple m1 computer, then use the 1.16 installation for Darwin on ARM. There should not be any major differences between 1.16 and 1.13. Unfortunately, m1 support did not start until 1.16.

If you are on an Apple Intel computer, then use the 1.13 installation for Darwin on x86.

---

# 2 Department Machine Development

Another option is to use a department machine. Since your code should already be hosted on GitHub, this option should be relatively easy to set up. Moreover, Golang 1.13 is already installed on department machines, meaning you won't have to do any setup!

If you've never SSH'd into a machine before, check out the Brown CS Department's guide here. Then, `git clone` your repository in your userspace, verify that everything is working, and code away!

The only thing to remember is: **make sure you permission your code correctly so nobody else can read it!!!!** Failure to permission code correctly is a violation of our academic code - see our syllabus for more.

---

# 3 Dockerized Development

In case local development isn't working for you, we provide a Dockerfile that *should* work throughout the course. If the Dockerfile stops working, come to TA hours or post on Campuswire, and we'll try to get it working again. Install Docker here (you don't need Docker Desktop, but it doesn't hurt).

Download our Dockerfile here. To run it, you'll need to first **build the image**, and then **run a container** on that image. Drop the Dockerfile into your project directory (if it isn't already in the stencil), then run the following two to build and run:

- `docker build -t bumblebase:latest .`
- `docker run --rm -v $PWD/data:/app/data -it bumblebase:latest`

---

# Feedback

As this is a new course, we appreciate any feedback you have for us! If you enjoyed this assignment, hated this assignment, or have other thoughts to share, please do so here!