But it works for me!
How to share research code

Aparna Bhaskaran, Prabha Acharya, Ryan Tam

SCSN

Brown Bag Seminar

September 3, 2025

Code development workflow

- Version control
- Documentation
- Coding best practices
- Testing
- Portability + scalability
- Running research code at SCSN

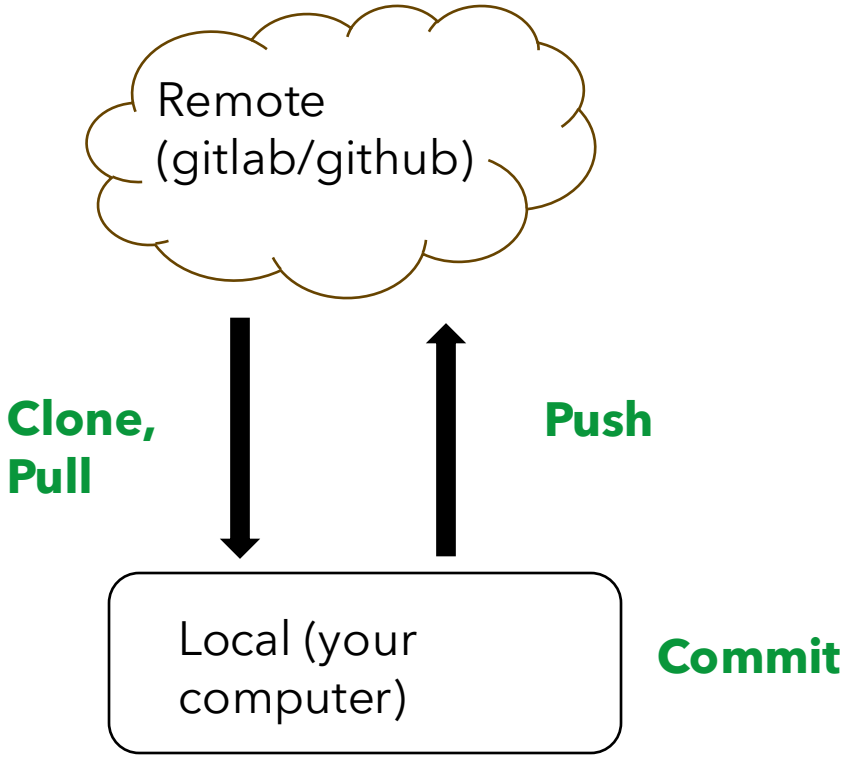
Version control

- Version control your source code – how many folks use git or another code management system?
  - Start now
  - Do it for new as well as existing code
  - Do it for one off scripts(!) or long running research projects
  - Your future self will thank you

Why use git platforms

- Open-source platform built around git. They include a comprehensive set of tools and features to manage git repositories, project planning, documentation, issue tracking, continuous integration/continuous deployment (CI/CD), etc.
- Free tier of gitlab/github/bitbucket as your remote repository – how many folks use them?
- Advantages
  - Share your code easily (and accept contributions)
  - Ticketing system – create tickets to track code changes
  - Configured, secure and shared development environment (codespaces, workspaces)
  - Many more…

Commit vs Push



- gitlab/github = remote, local = your computer
- Commit vs Push
  - Commit – happens locally, on your computer
  - Push – send the change made on your computer to gitlab/github

Document

- Documentation is communication (with users and your future self)
- Write requirements document, keep it up-to-date
- Provide a well formatted README in your repo describing your project
- Include any pre-requisites and dependencies needed to use the project in the README
- Include how to compile and/or run the code. Bonus: include “one click” setup and run (could be a container, setup script, pip install, etc.)

Document

- Include sample configuration parameters or files, if your project needs them
- Include working examples with sample data
- Include how to make contributions to the code
- Bonus : setup auto-generation of documentation (doxygen, makethedocs)
- Commented code is well documented code

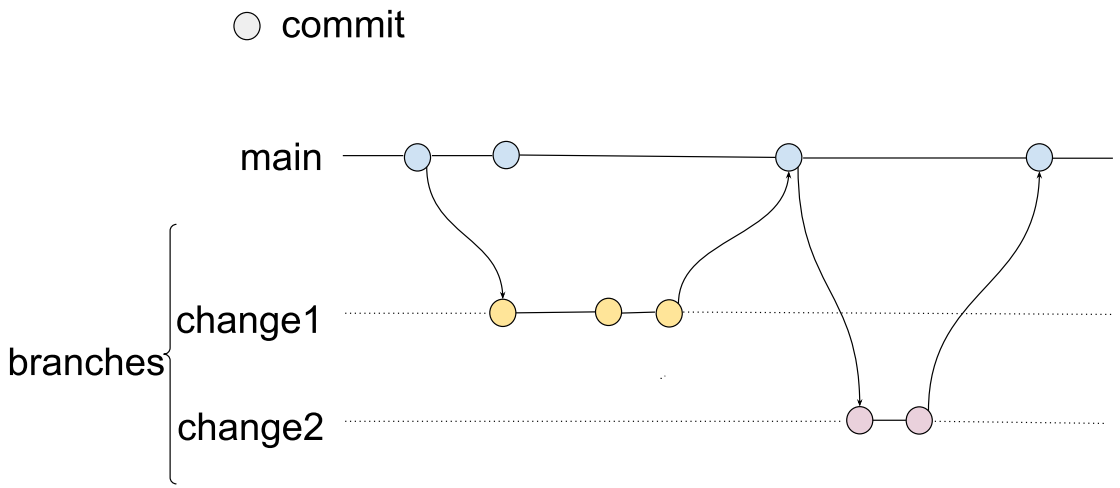
Code development best practices

- Make your code modular
  - split long code chunks into functions,
  - use classes where appropriate,
  - avoid hard coding, use configurations instead
- Follow style and coding conventions for your programming language (For e.g. PEP8), consistency is key
- Add comments to your code
- Add test(s) as you add code
- Use AI augmented IDE (for e.g. cursor) for development

Code development best practices

- Create tickets/issues for code changes
- Create a separate branch for making code changes, merge into `main` branch. Ensure `main` always is a working and tested version of your code.
- Commit your code often
  - Split changes into individual commits – makes it easy to track code changes, makes reverts easier
- Good commit messages
  - Mention intent along with the contents of the commit i.e. why the code change was made and what the change was.
- Push frequently

Git workflow



Testing

- Well tested code is reliable code / trendy code / popular code
- Use requirements document to create tests!
- Write unit and integration tests
  - Unit tests test individual functions of your code
  - Integration tests are end-to-end tests that test the functionality of your entire code
- Run your tests before you push to the git platform
- **Testing provides confidence to you (and your users/collaborators) that your code works as expected**

CI/CD - Continuous integration / continuous delivery or deployment

- CI is the practice of frequently pushing changes from developers to a central repo
  - tests are triggered automatically after each push
  - allows for early detection of bugs
- With CD, code changes that pass all automated checks are automatically deployed to a staging environment for final testing and approval
- Pipeline (Build ==> **Test** ==> Deploy)
- Combined, they help streamline the software development process
- **CI/CD gives confidence to your collaborators to contribute effectively**

Use cases

- PhaseNet - https://github.com/AI4EPS/PhaseNet
- ObsPy - https://github.com/obspy/obspy
- Seisbench - https://github.com/seisbench/seisbench
- Many more...

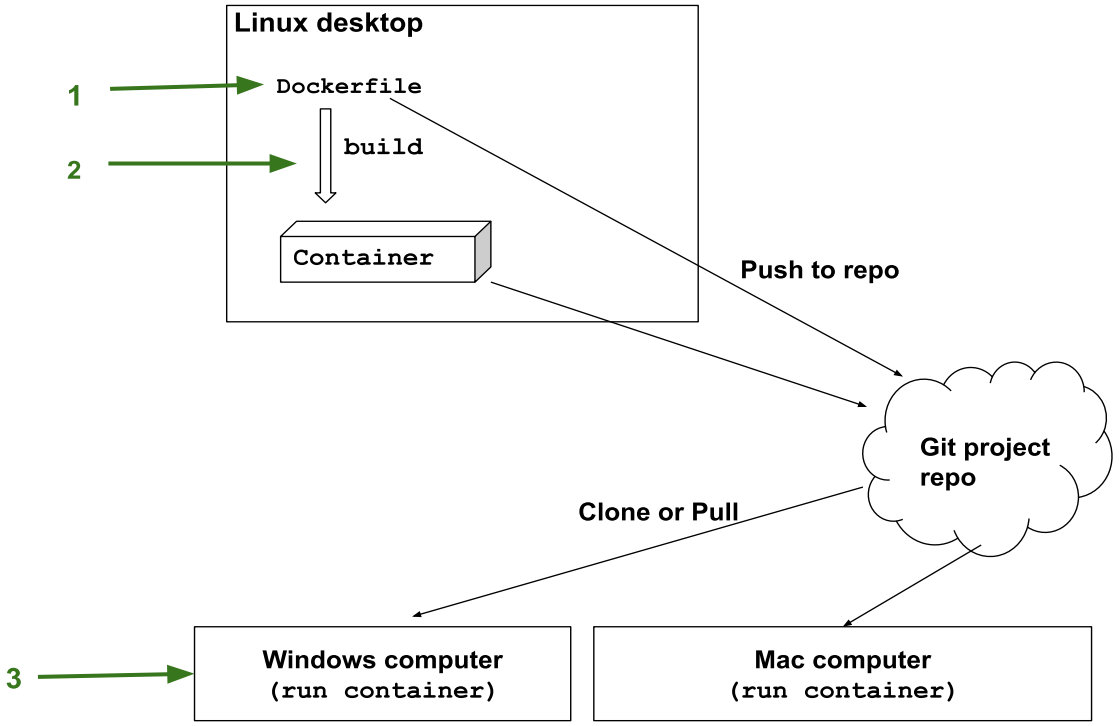
Ensure your code works for anyone

- "But it works for me!?!"
- Often, code fails to work for external users due to environmental issues
  - Mismatch in language version
  - Mismatch in versions of dependencies / missing dependencies
  - Packages not available on their operating system, different OS
- What if you could "give" your environment to users?
  - Containers - "A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another." - www.docker.com

Containers

- Container providers – docker, podman, LXD
- Docker is the most popular, is available on most OSes like linux, MacOS, windows. Podman is very similar to docker and is recommended on newer versions of linux
- Gitlab/Github allow you to store your containers so no need to build them everytime.
- Containers facilitate scalability, portability, uniformity

Containers



Dockerfile

```
FROM python:3.12
WORKDIR /usr/local/app

# Install the application dependencies
COPY requirements.txt ./
RUN pip install -r requirements.txt

# Copy in the source code
COPY src ./src
RUN ls -ltr
EXPOSE 6000

# Setup a seismo user so the container
doesn't run as the root user
RUN useradd seismo
USER seismo

CMD ["uvicorn", "app.main:app", "--
host", "0.0.0.0", "--port", "8080"]
```

Running research code at SCSN

- Code developed during research is included in seismic network operations
- Operations run 24/7, essential that code is well tested, reliable and configurable.
- Meet Advanced National Seismic System standards w.r.t latency, quality and availability
- Meet ShakeAlert (earthquake early warning) requirements
- Ensure new software complies with the existing system
- Multiple testing stages (test, staging, shadow, primary)
- Infrastructure as code to ensure consistency of deployment and configuration
- Logging, audit trail and monitoring
- Authentication, authorization and security

Useful links

- Git basics / tutorial - https://www.atlassian.com/git
- Git cheatsheet - https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- Docker - https://www.docker.com/
- Podman - https://podman.io/

Thank you. Questions?