Principles of Scientific Software Development

May 20, 2022

Isabel Restrepo & Paul Stey

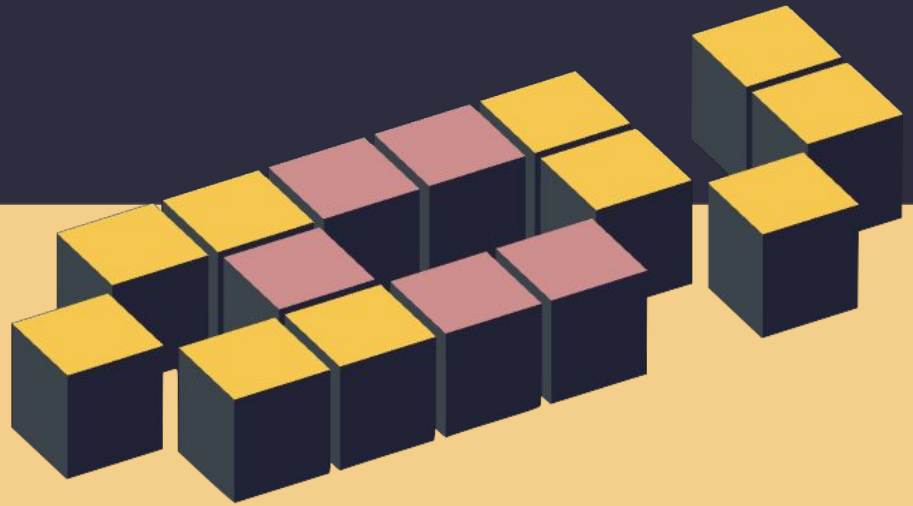
BROWN

BROWN
Center for
Computation &
Visualization

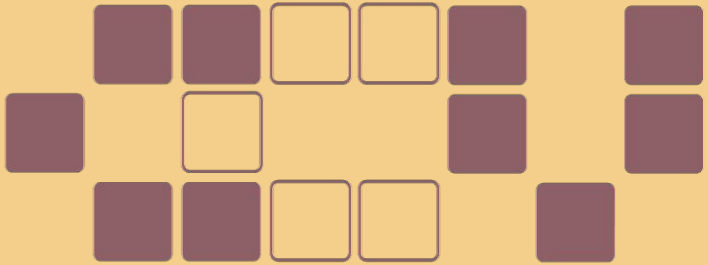
ccv.brown.edu
ccv@brown.edu

Outline

1. About CCV
2. Crisis of Replication
3. Improving reproducibility
4. Principles
    a. Versioned
    b. Easy-to-Use
    c. Rigorous
    d. Packaged



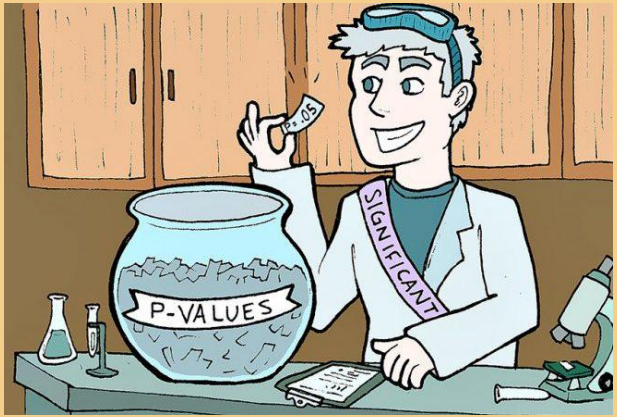
ccv.brown.edu



BROWN
Center for
Computation &
Visualization

Crisis of Replication

1. Ioannadis (2005)
   a. "*Why Most Published Research Findings Are False*"
      i. "*p*-hacking"
      ii. Hypothesizing-after-the-fact
      iii. Small studies with low power
2. Misaligned Incentives in Academia
   a. "*Publish or perish!*"
   b. Priority of "Oh, wow!" findings
   c. Dis-incentives for replication studies



BROWN

BROWN
Center for
Computation &
Visualization

Replication & Reproducibility

1. Reproducibility [in terms of methods] is a necessary component of replication
2. Historically, a very detailed “Methods” section of a paper might have been sufficient
3. That is different now
    a. Computational environments are part of reproducibility

How do we Improve Reproducibility?

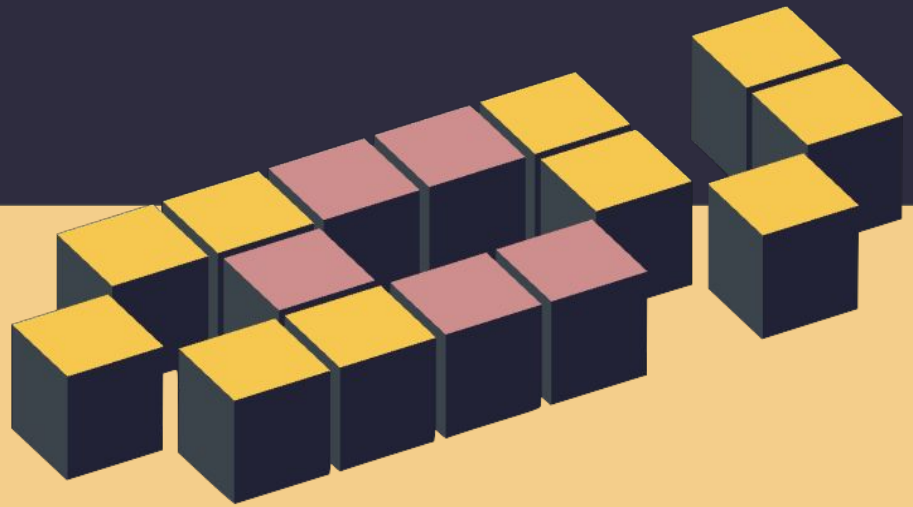
1. Versioned
2. Easy-of-use
    a. Readable, modular, documented
3. Rigor
    a. Testing and CI
4. Packaging code
    a. Cross platform
    b. Containers

BROWN

BROWN
Center for
Computation &
Visualization

Outline

1. About CCV
2. Crisis of Replication
3. Improving reproducibility
4. Principles
    a. Versioned
    b. Easy-to-Use
    c. Rigorous
    d. Packaged

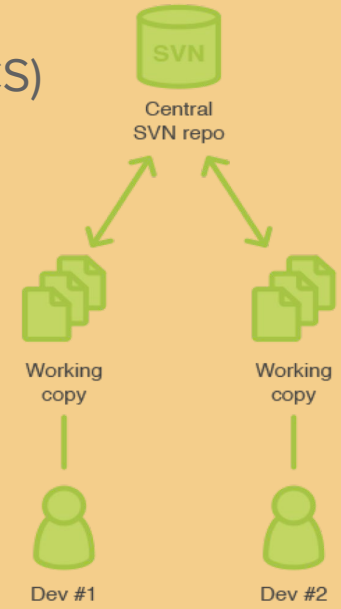


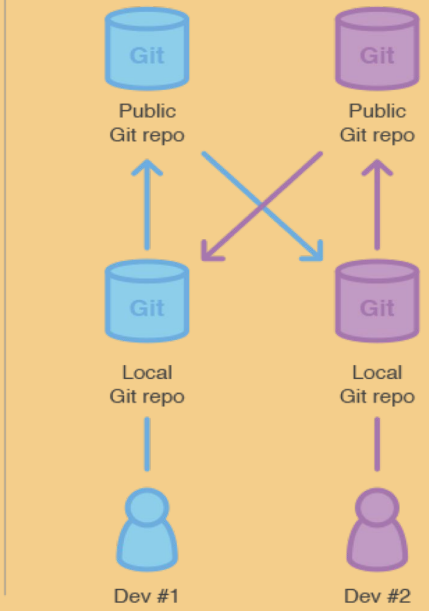
Version Control

1. Any software-related project, independent of size, should be under version control
2. Why?
   a. Tracking changes over time
   b. None of this:
      `boosted_tree_model_v5_final_FINALv2.py`
   c. Protect stable/production code from bugs
   d. Best practice for collaboration

Version Control Systems

1. Examples of Version Control Systems (VCS)
   a. ~~CVS, SVN, Mercurial~~
   b. Git

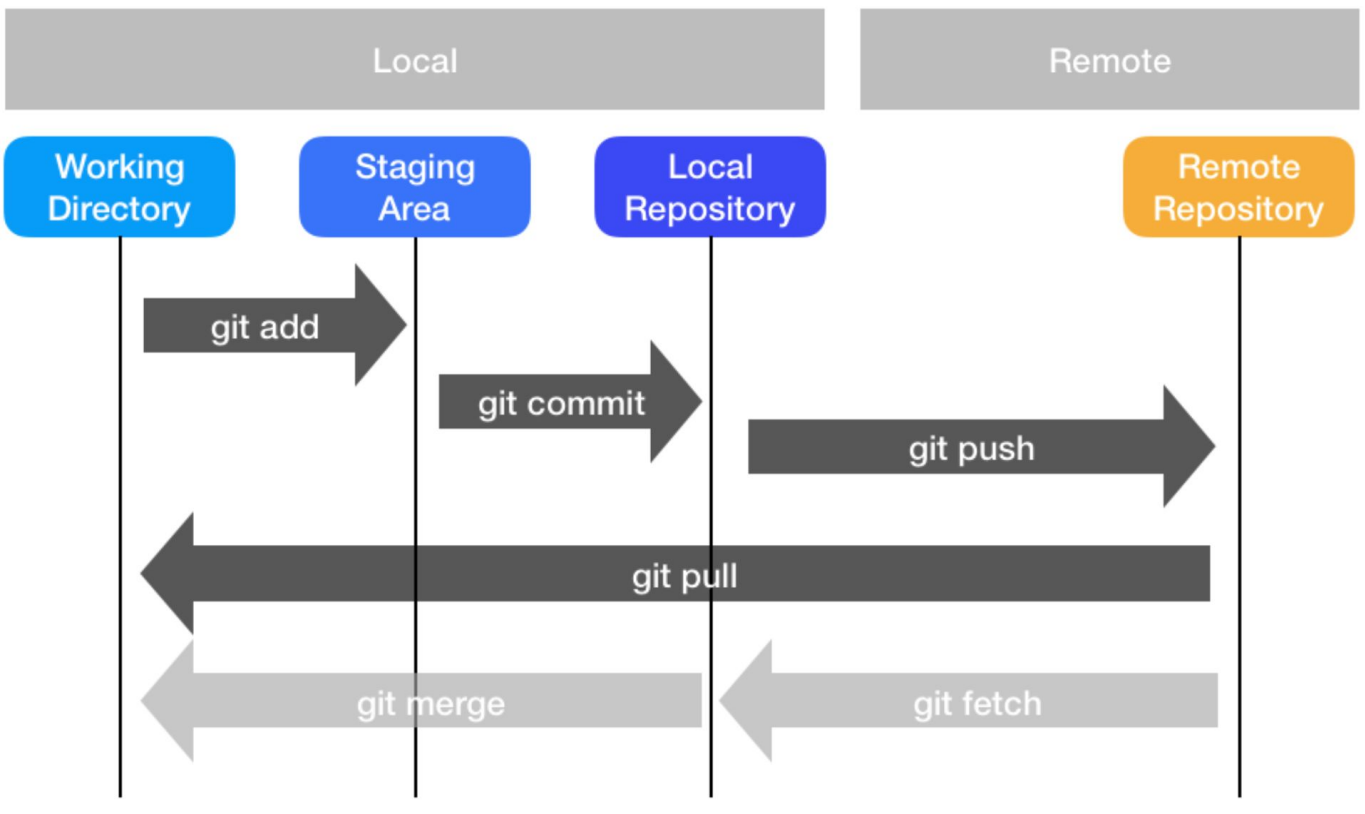




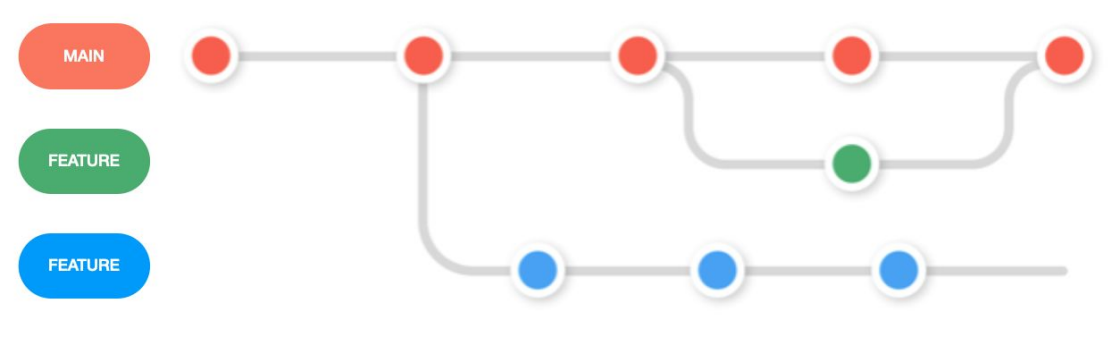
Git Hosting

1. Hosting platforms
    a. Facilitate collaboration
    b. Add functionality
        i. Pull/merge requests
        ii. Automation for testing/deployment (e.g., “Actions”)

Git Basics


Local
Remote
Working Directory
Staging Area
Local Repository
Remote Repository
git add
git commit
git push
git pull
git merge
git fetch

Popular Git Workflows / Branching



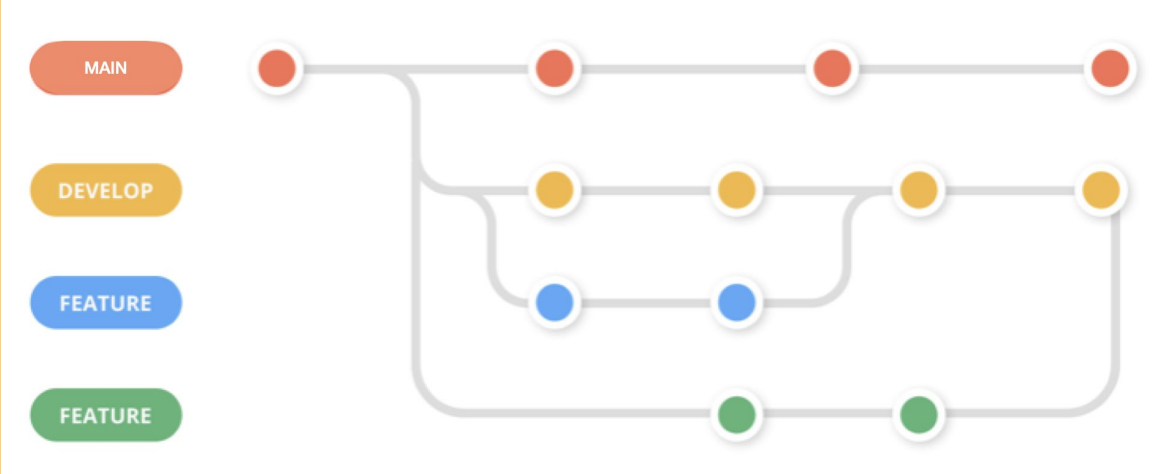
1. Feature Workflow
    a. Create feature/topic branch off the main branch
    b. Main branch is *always stable*

BROWN

BROWN
Center for
Computation &
Visualization

Git Workflows (cont.)

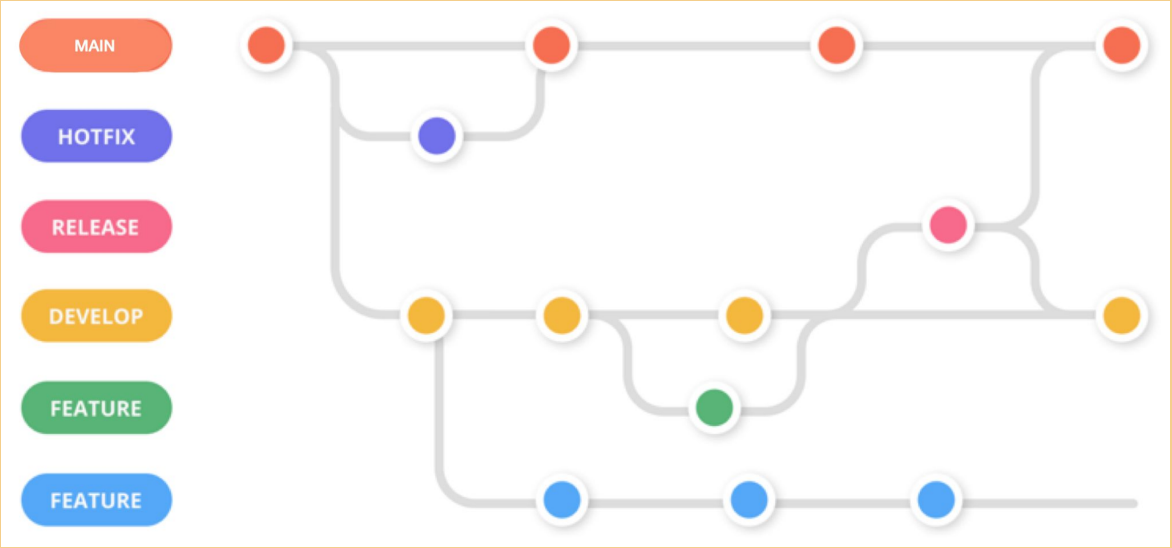
1. Feature Workflow + Develop
   a. Create feature/topic branch off the develop branch
   b. Main branch is production
   c. Useful for staging/production



BROWN

BROWN
Center for
Computation &
Visualization

Git Workflows (cont.)



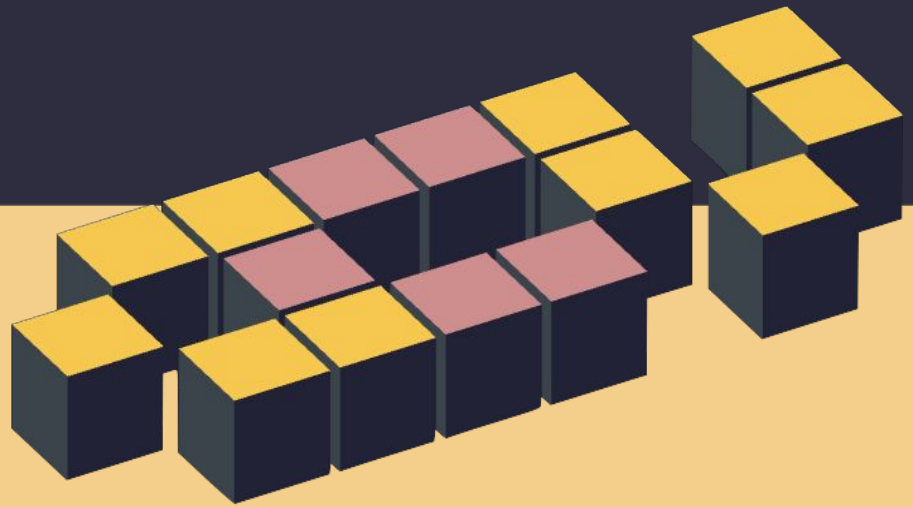
1. GitFlow
   a. Adds “hotfix” and release branches
   b. Popularity has decreased due to complexity

Git Workflows Summary

1. Choose *something*
2. Choose a workflow that satisfies needs of your team
3. Choose a workflow that people will actually use
4. When in doubt, prefer simplicity

Outline

1. About CCV
2. Crisis of Replication
3. Improving reproducibility
4. Principles
    a. Versioned
    b. Easy-to-Use
    c. Rigorous
    d. Packaged



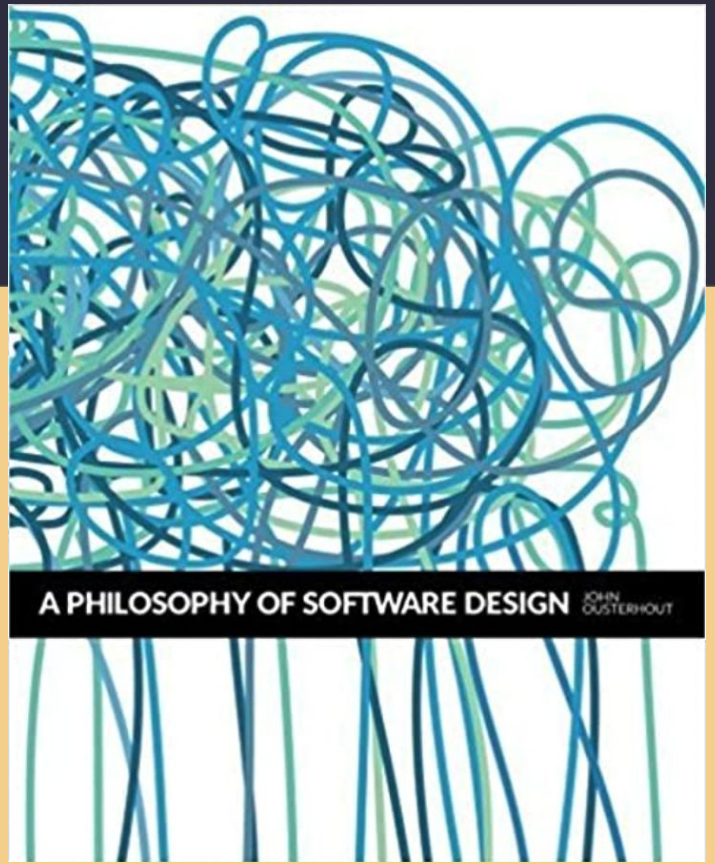
What is Readable Code?

Readable Code

1. Use linter (e.g., lintr, flake8, clippy)
2. Use formatter (e.g., styler, black)
3. Use meaningful variable names
4. Use comments
    a. More "*why*" than "*what*" (D.R.Y.)
5. Simplify expressions
    a. No code golf
6. Remove commented-out code

Modular Code

1. Use small functions
   a. Single-responsibility principle
2. Think of support for multiple types
   a. Function overloading
   b. Generics
3. Develop expertise in language-specific best practices (e.g., imperative, functional, object-oriented)



BROWN

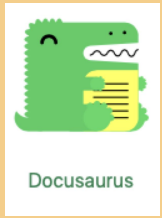
BROWN
Center for
Computation &
Visualization

Documentation

1. README.md
    a. Always, always, always
    b. Basic description of software
    c. Installation
    d. Minimum example
2. For more comprehensive docs:
    a. Use static-site generator
    b. Host it with the repo (e.g., GitHub Pages)

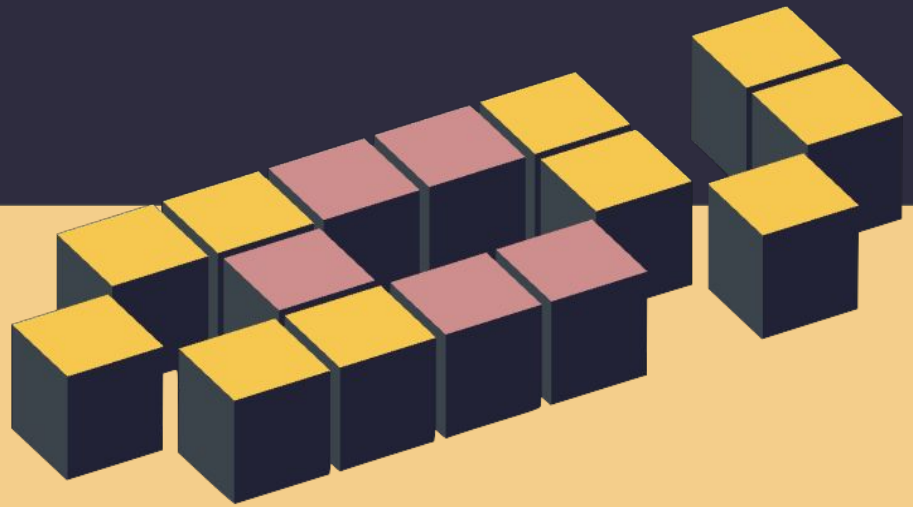
Documentation

1. Static-site generators
    a. From Markdown to beautiful website



Outline

1. About CCV
2. Crisis of Replication
3. Improving reproducibility
4. Principles
    a. Versioned
    b. Easy-to-Use
    c. Rigorous
    d. Packaged



Why write tests?

1. Ariane-5 flight 501 (1996-06-04)
2. Re-used software from Ariane-4
3. Software attempted to convert 64-bit float to 16-bit integer, causing arithmetic overflow

BROWN

BROWN
Center for
Computation &
Visualization

Why write tests?

1. Prevent bugs in software before they can happen
2. Make sure discovered bugs don't re-occur
3. Make updates confidently

Types of Testing

1. Unit Test
    a. Test each function/component
    b. Arrange, Act, Assert pattern

Arrange, Act, Assert

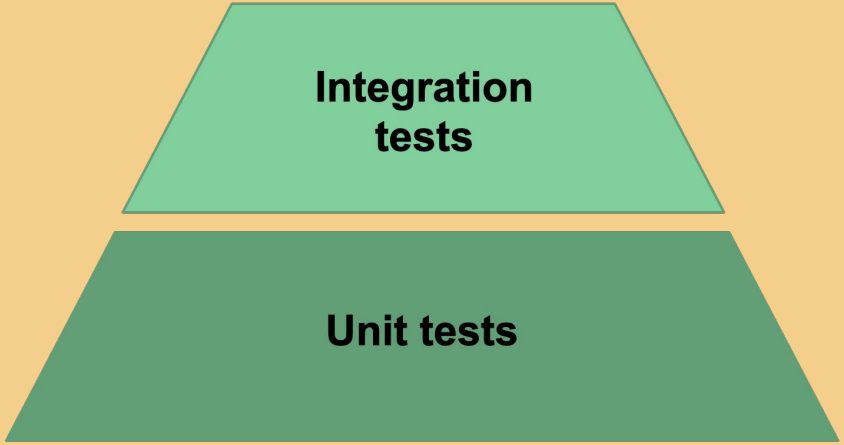
```rust
async fn health_check_works() {
    // Arrange
    let app = spawn_app().await;
    let client = reqwest::Client::new();

    // Act
    let response = client
        .get(&format!("{}/health_check", &app.address))
        .send()
        .await
        .expect("Failed to execute request.");

    // Assert
    assert!(response.status().is_success());
    assert_eq!(Some(0), response.content_length());
}
```

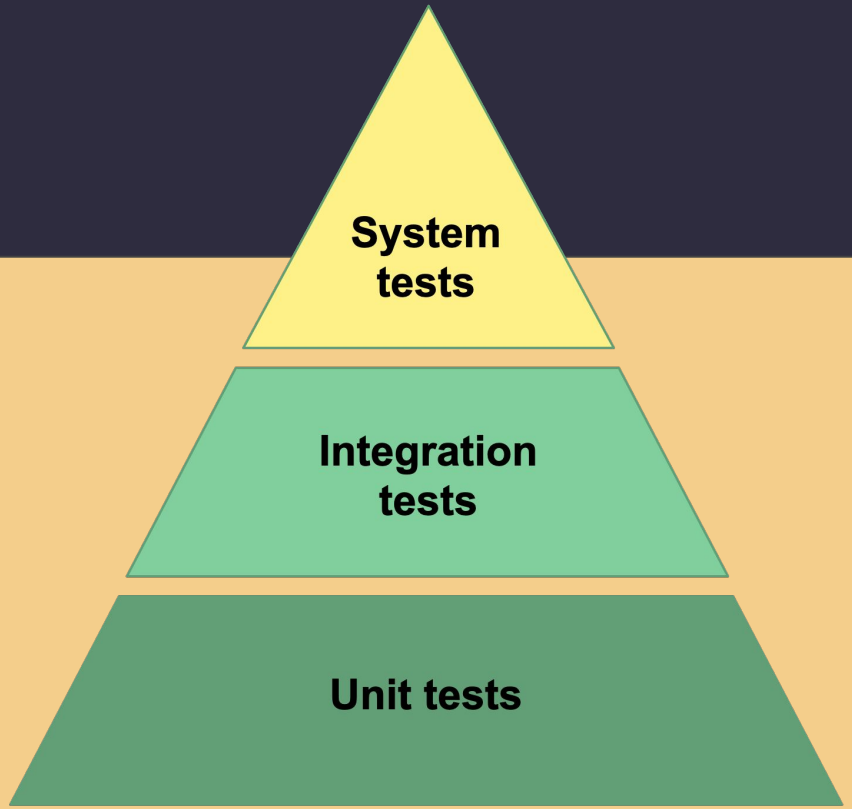
Types of Testing

1. Integration Tests
    a. Tests multiple components and their ability to work together
    b. Often require test data
        i. Faker



Types of Testing

1. System tests
    a. Complete test of system
    b. Sometimes require mock infrastructure (e.g., DB conn)



Continuous Integration (CI)

Local Git Repo

Remote Git Repo

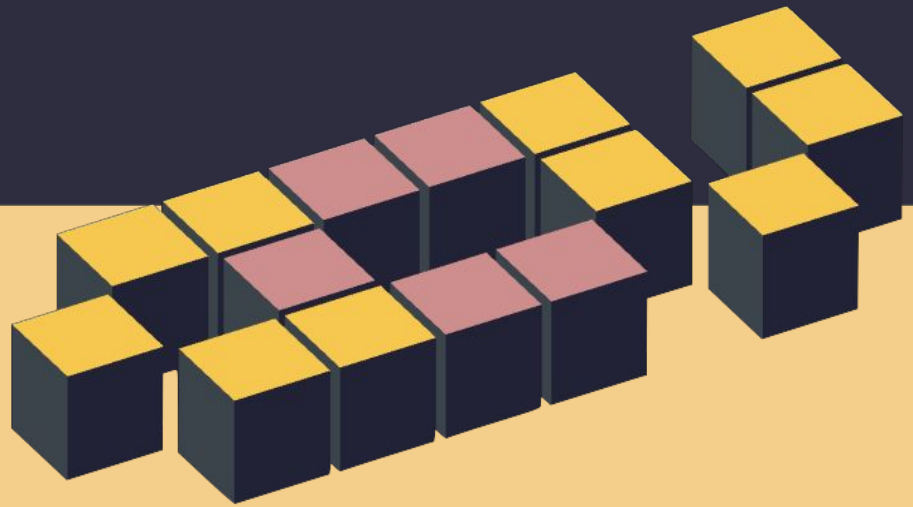
CI Services

git push

web hook

Outline

1. About CCV
2. Crisis of Replication
3. Improving reproducibility
4. Principles
    a. Versioned
    b. Easy-to-Use
    c. Rigorous
    d. Packaged



Packaging Code

1. Writing a package/module/crate
   a. Critical for distribution
   b. Encourages best practices
   c. Greatly simplifies usage

CONDA

Poetry

python™ Package Index

devtools

BROWN

BROWN Center for Computation & Visualization

Reproducible Software

1. Source code
    a. In version control
    b. With release tagged
2. Third-party libraries
    a. Captured in your package manifest
    b. Peg libraries to specific versions
3. Run time environment

What is a Container?

1. Container is a virtualized environments
2. Similar to VMs (i.e., virtual machines)
    a. Containers have less performance overhead
3. Container Software
    a. Docker
    b. Singularity
    c. containerd
    d. Kata

docker

BROWN

BROWN
Center for
Computation &
Visualization

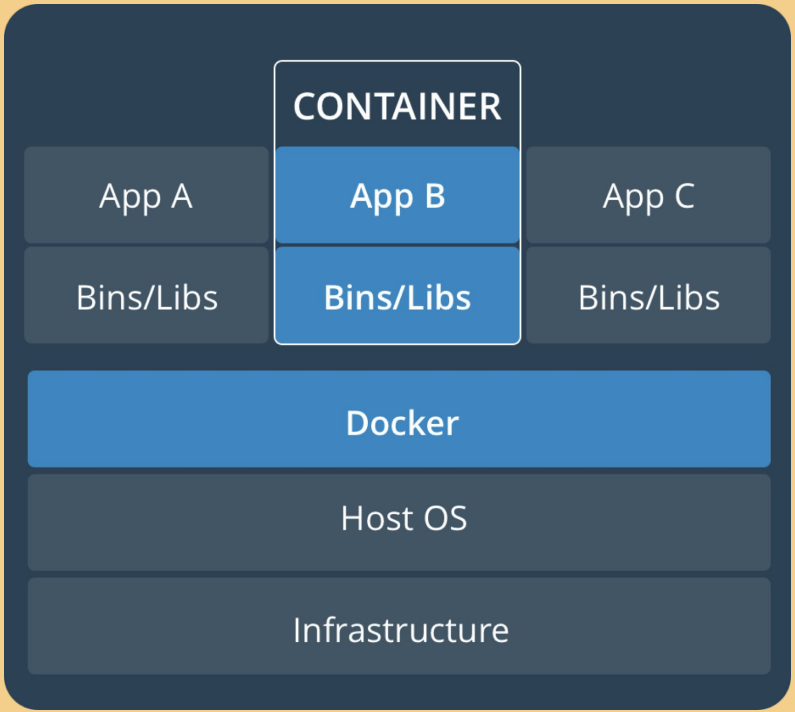
Docker

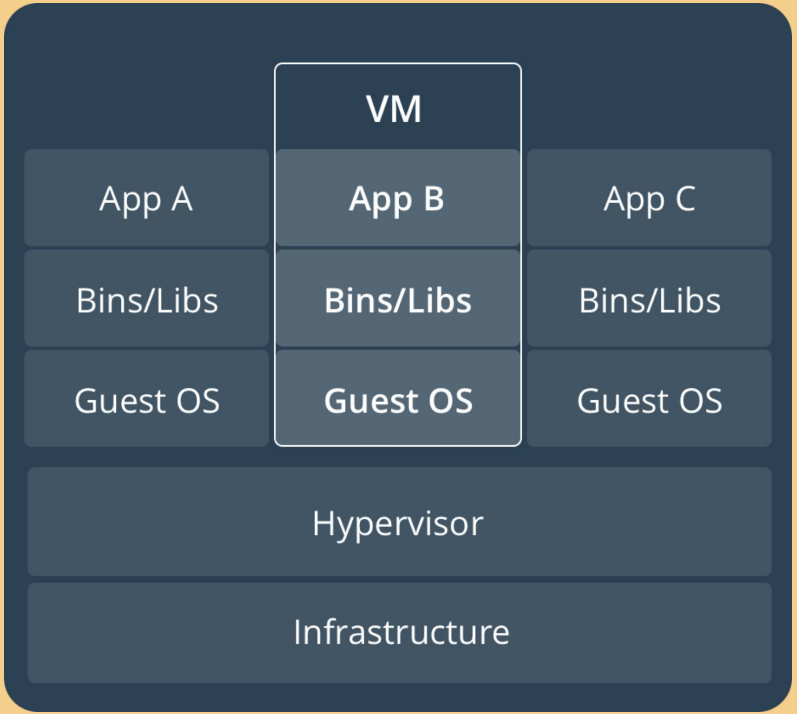
1. Container engine for Windows, macOS, and Linux
2. Easy-to-use
3. Can use host's GPUs
4. Free (as in beer and speech)!!!

BROWN

BROWN
Center for
Computation &
Visualization

Docker vs. VM


CONTAINER
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure


VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure

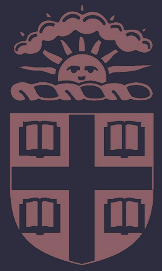
Live Demo

Code available here:

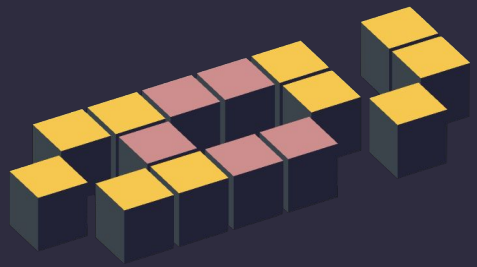
https://github.com/brown-ccv/developing-scientific-software

Any questions?

¯_(ツ)_/¯



BROWN



Thank you!!!

ccv.brown.edu