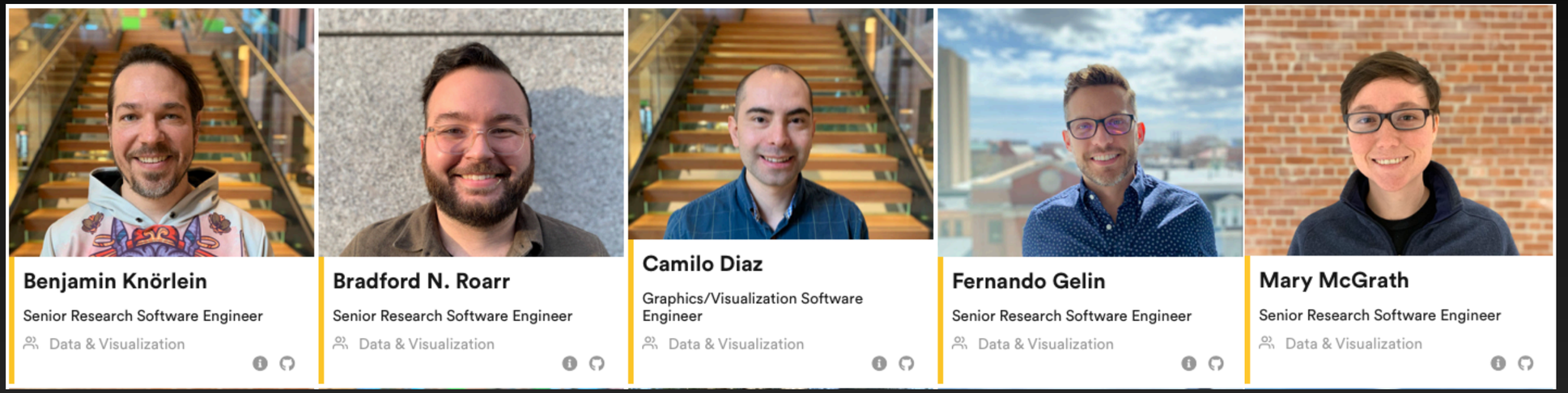
Reproducible Data Science

Data 1030. Brown University

Isabel Restrepo. Lead Data Scientist. CCV Brown University. October 18th 2020

Slide credits: August Guang, Bradford Roarr, Paul Stey and Andrew Leith

My Team



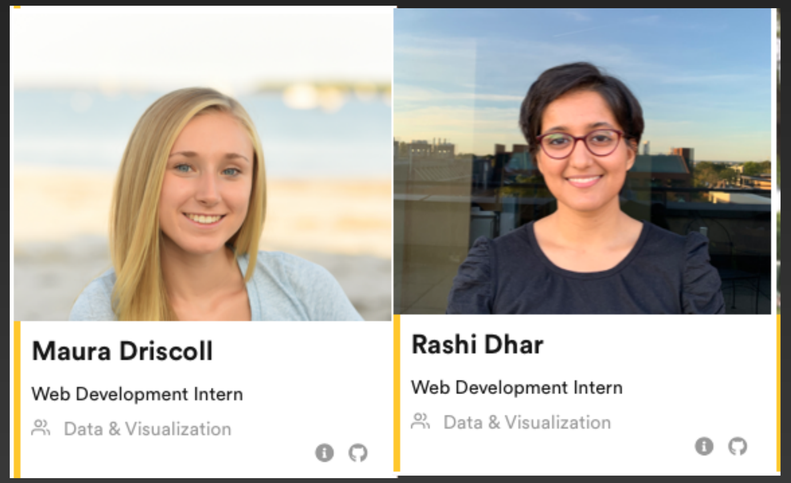
Benjamin Knörlein
Senior Research Software Engineer
Data & Visualization

Bradford N. Roarr
Senior Research Software Engineer
Data & Visualization

Camilo Diaz
Graphics/Visualization Software Engineer
Data & Visualization

Fernando Gelin
Senior Research Software Engineer
Data & Visualization

Mary McGrath
Senior Research Software Engineer
Data & Visualization



Maura Driscoll
Web Development Intern
Data & Visualization

Rashi Dhar
Web Development Intern
Data & Visualization

What is Reproducible Data Science?

Three main components

- Reproducible Software
- Reproducible Data
- Reproducible Workflows

Dilemma

- Scientific method presumes replicability
- Replication studies have historically NOT been viewed as innovative and often rejected as incremental

Contributing factors

- Competition for tenure track positions
- “Publish or perish”
- Accumulating publications is critical, particularly early in career
- Academic works have no plan for long term maintenance

Reproducibility is essential for replication

- “Methods” section in journal articles
- Historically, this might have satisfied many concerns
- Without technical infrastructure, replication cannot be complete

Efforts to do something about it

- “Why Most Published Research Findings Are False”
  - J. Ioannidis (2005) PLoS Medicine
  - Disregard for statistical rigor, (“*P-*hacking”, Test many hypotheses, no Type-I error correction, small sample, low-power studies)
- FAIR guiding principles (Findable, Accessible, Interoperable and Reusable) - NIH
  - Push for papers backed by repositories of software and data.
  - Trends for "live" paper

Reproducible Software

- Version Control
- Ease of use
  - Readable
  - Modular
  - Documentation
- Rigor
  - Testing
  - CI
- Packaging code
  - Cross platform
  - Containerization

Version Control

" A component of software configuration management, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information."

Wikipedia

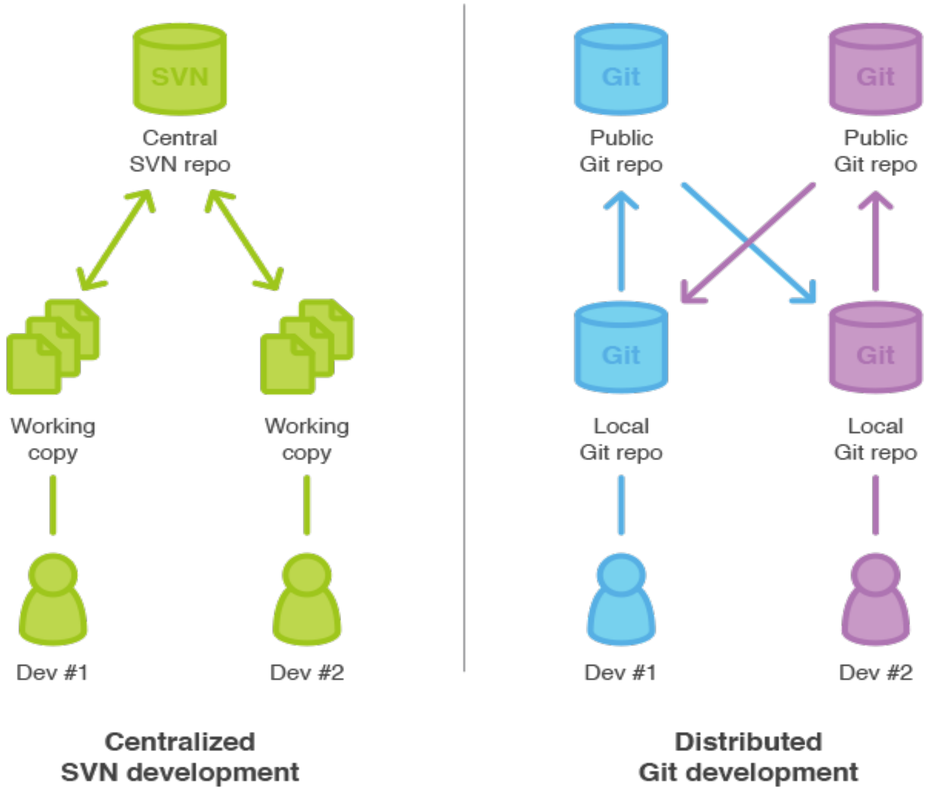
Version Control

Any software related project, independent of size, should be under version control or in a *repository*

WHY?

- Tracking changes over time
- Good collaboration practices
- Protect stable/production code from bugs

- **Technologies**
  - ~~CVS, SVN, Mercurial~~ (History)
  - **GIT**



- **Hosts**

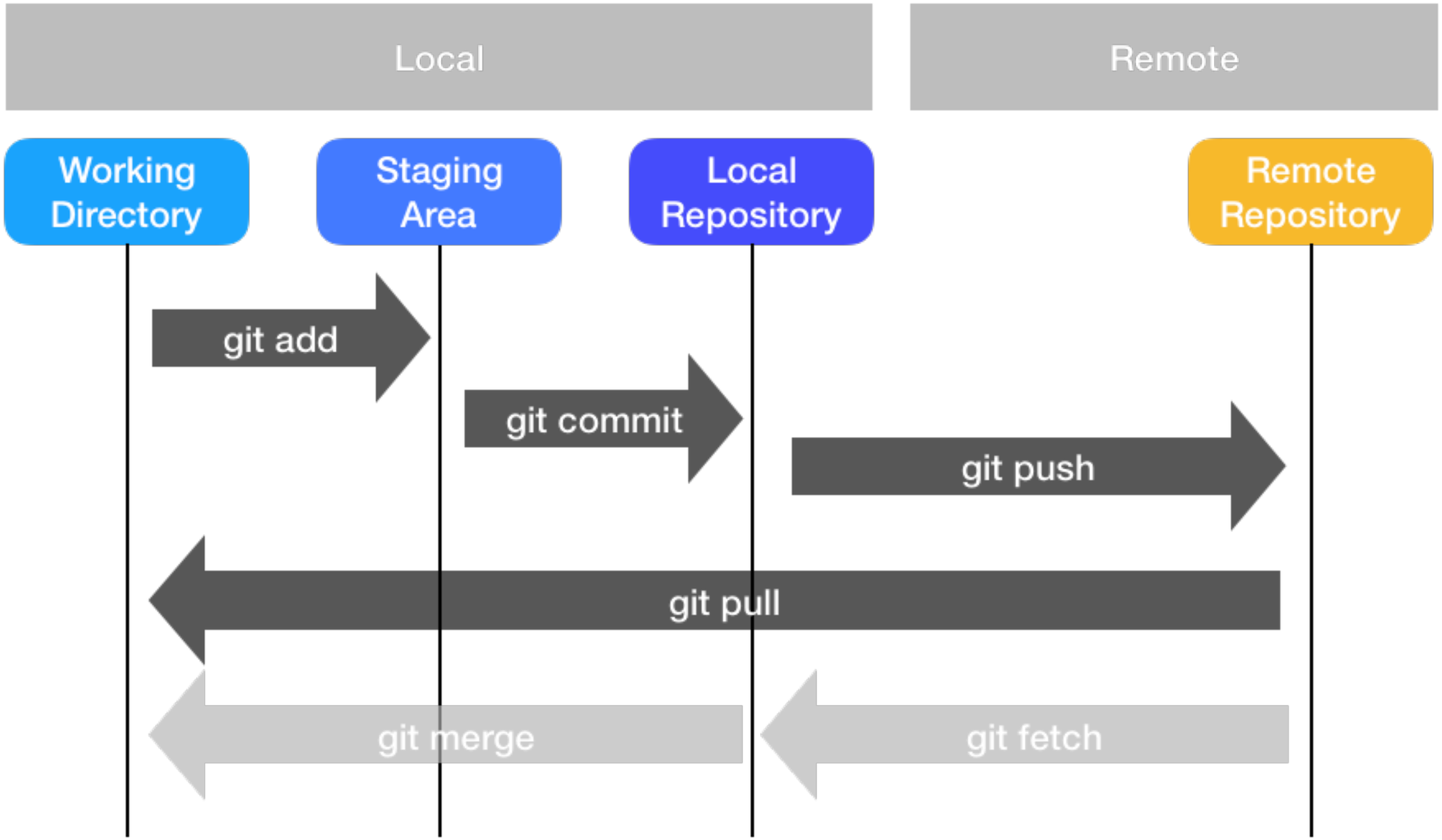
GitHub

GitLab

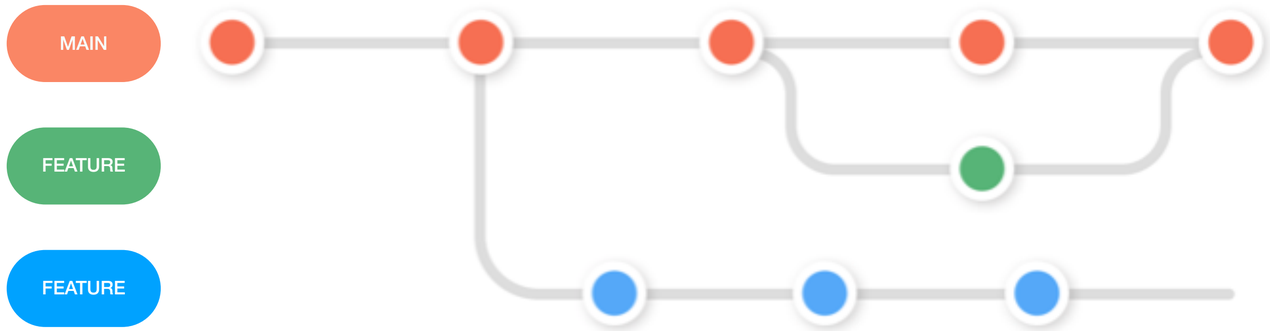
Bitbucket

* https://hoantran-it.blogspot.com/2016/06/git-tutorial-1-git-committing-and.html

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



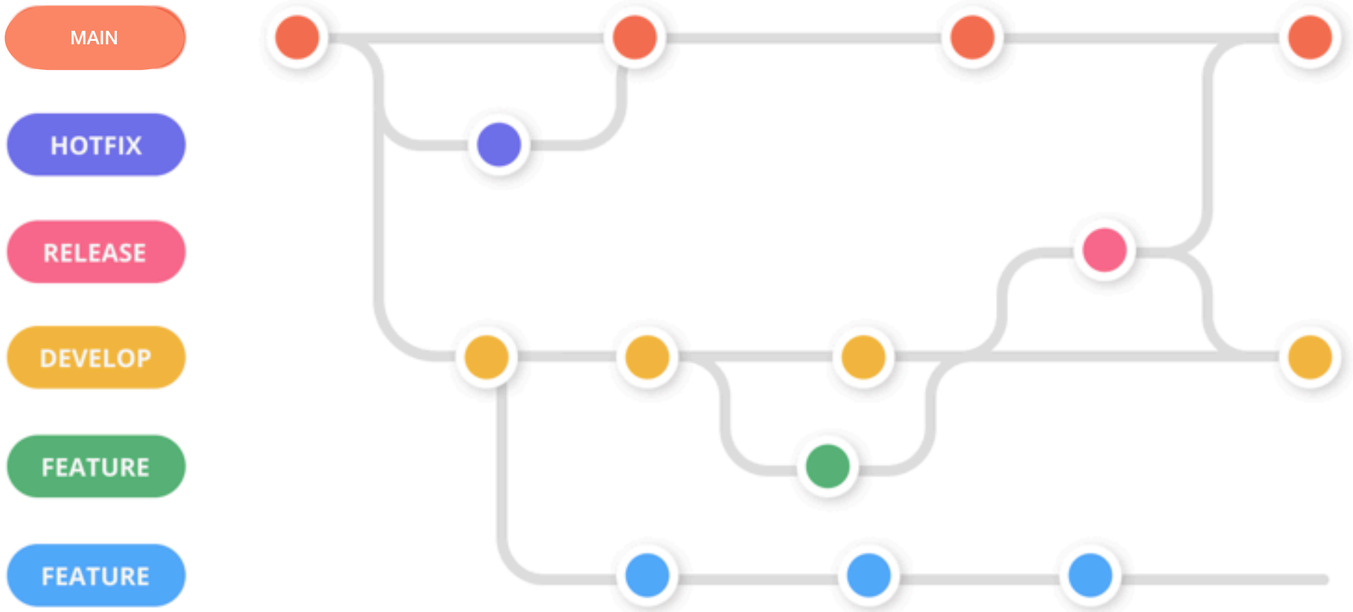
Feature Workflow

- Create feature/topic branches of main branch
- Main branch is always stable

Feature Workflow + Develop

- Create feature/topic branches of develop branch
- Main branch is the production branch
- Useful for staging/production services

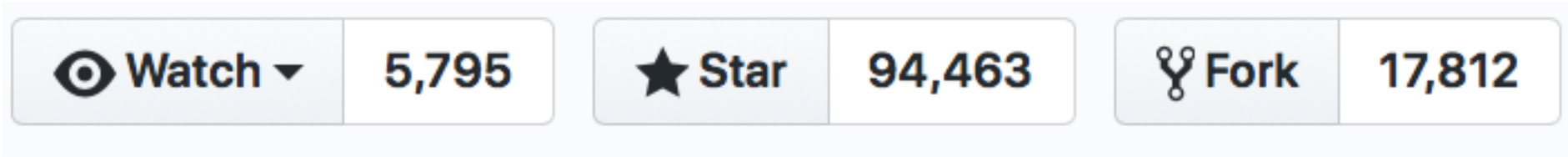
Popular Git Workflows



Gitflow

- Adds hotfix and release branches
- Original workflow
- Popularity has decreased due to complexity

Git Social



- Today Version Control is more than simply tracking changes

- It is social:
  - Accelerates rate, quality and number of people working on Open Source
  - Less wheels reinvented :)
  - A way to build your life CV of code
  - Development of tools for testing, documenting, etc that are better integrated

Reproducible Software

- Version Control
- **Ease of use**
  - Readable
  - Modular
  - Documentation
- Rigor
  - Testing
  - CI
- Packaging code
  - Cross platform
  - Containerization

What is READABLE code?

Use a linter: flake8

Use a formatter: black

Use meaningful names for variables

READABLE code

Use comments

Remove commented out code

Simplify statements

Don’t abuse one liners

Use Types if possible:
mypy

Modular code

Use small functions

Think of support for multiple types

Familiarize yourself with best practices for specific language. Functional vs Object Oriented

Documentation

- ALWAYS include a README.md
  - Basic description
  - Installation
  - Minimum Example

- For more comprehensive documentation use a static-site generator. Host it with repository, i.e. GitHub Pages
  - Track and easy to publish

Documentation

- Basic description
- Installation
- Minimum Example

  ...
- Badges
- Contributing Guidelines

README.md

Documentation	Linux CPU	Linux GPU	Mac OS CPU	Windows CPU	Android
api reference	build passing	build passing	build passing	build failing	build failing Download 1.8.0-rc0

TensorFlow is an open source software library for numerical computation using data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture enables you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow also includes TensorBoard, a data visualization toolkit.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

Keep up to date with release announcements and security updates by subscribing to announce@tensorflow.org.

Documentation

- Basic description
- Installation
- Minimum Example

  ...
- Badges
- Contributing Guidelines

README.md

Installation

See Installing TensorFlow for instructions on how to install our release binaries or how to build from source.

People who are a little more adventurous can also try our nightly binaries:

Nightly pip packages

- We are pleased to announce that TensorFlow now offers nightly pip packages under the tf-nightly and tf-nightly-gpu project on pypi. Simply run `pip install tf-nightly` or `pip install tf-nightly-gpu` in a clean environment to install the nightly TensorFlow build. We support CPU and GPU packages on Linux, Mac, and Windows.

Individual whl files

- Linux CPU-only: Python 2 (build history) / Python 3.4 (build history) / Python 3.5 (build history) / Python 3.6 (build history)
- Linux GPU: Python 2 (build history) / Python 3.4 (build history) / Python 3.5 (build history) / Python 3.6 (build history)
- Mac CPU-only: Python 2 (build history) / Python 3 (build history)
- Windows CPU-only: Python 3.5 64-bit (build history) / Python 3.6 64-bit (build history)
- Windows GPU: Python 3.5 64-bit (build history) / Python 3.6 64-bit (build history)
- Android: demo APK, native libs (build history)

Documentation

- Basic description
- Installation
- Minimum Example

...

- Badges
- Contributing Guidelines

README.md

Try your first TensorFlow program

```
$ python
```

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a + b)
42
>>> sess.close()
```

README.md

Contribution guidelines

If you want to contribute to TensorFlow, be sure to review the contribution guidelines. This project adheres to TensorFlow's code of conduct. By participating, you are expected to uphold this code.

We use GitHub issues for tracking requests and bugs. So please see TensorFlow Discuss for general questions and discussion, and please direct specific questions to Stack Overflow.

The TensorFlow project strives to abide by generally accepted best practices in open-source software development:

Static-site generators

MkDocs

Project documentation with Markdown.

SLATE

API DOCS GENERATOR

Docusaurus

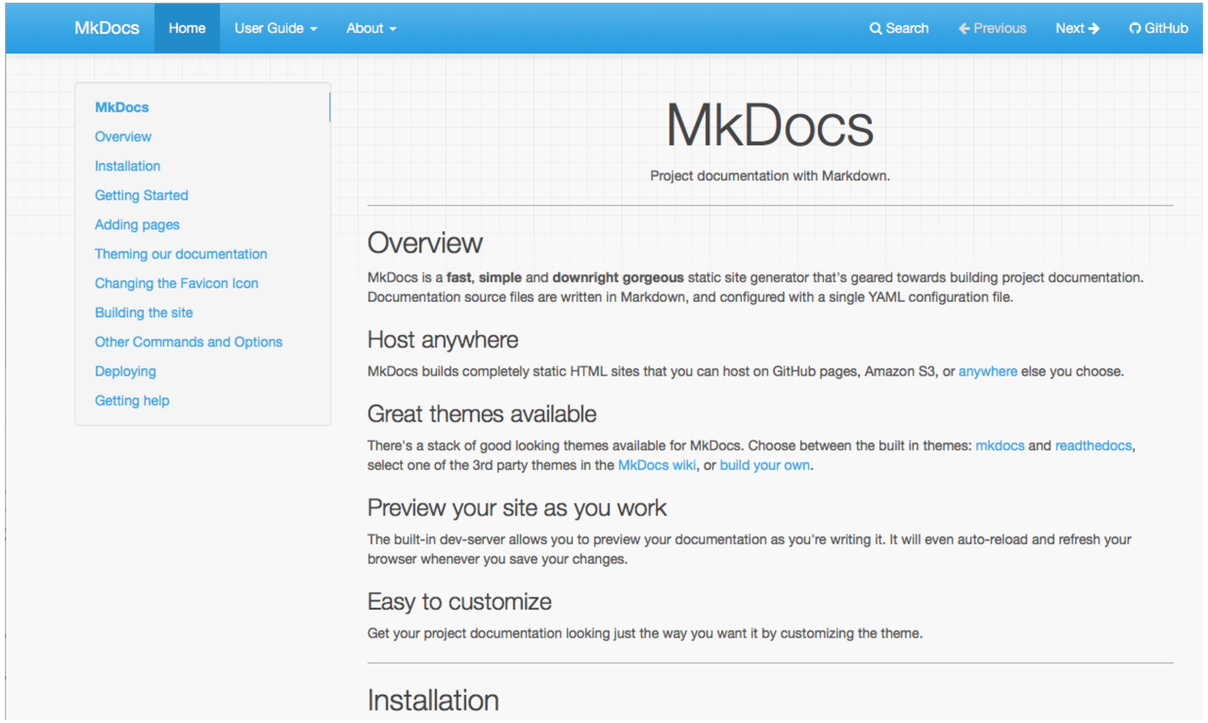
Read *the* Docs

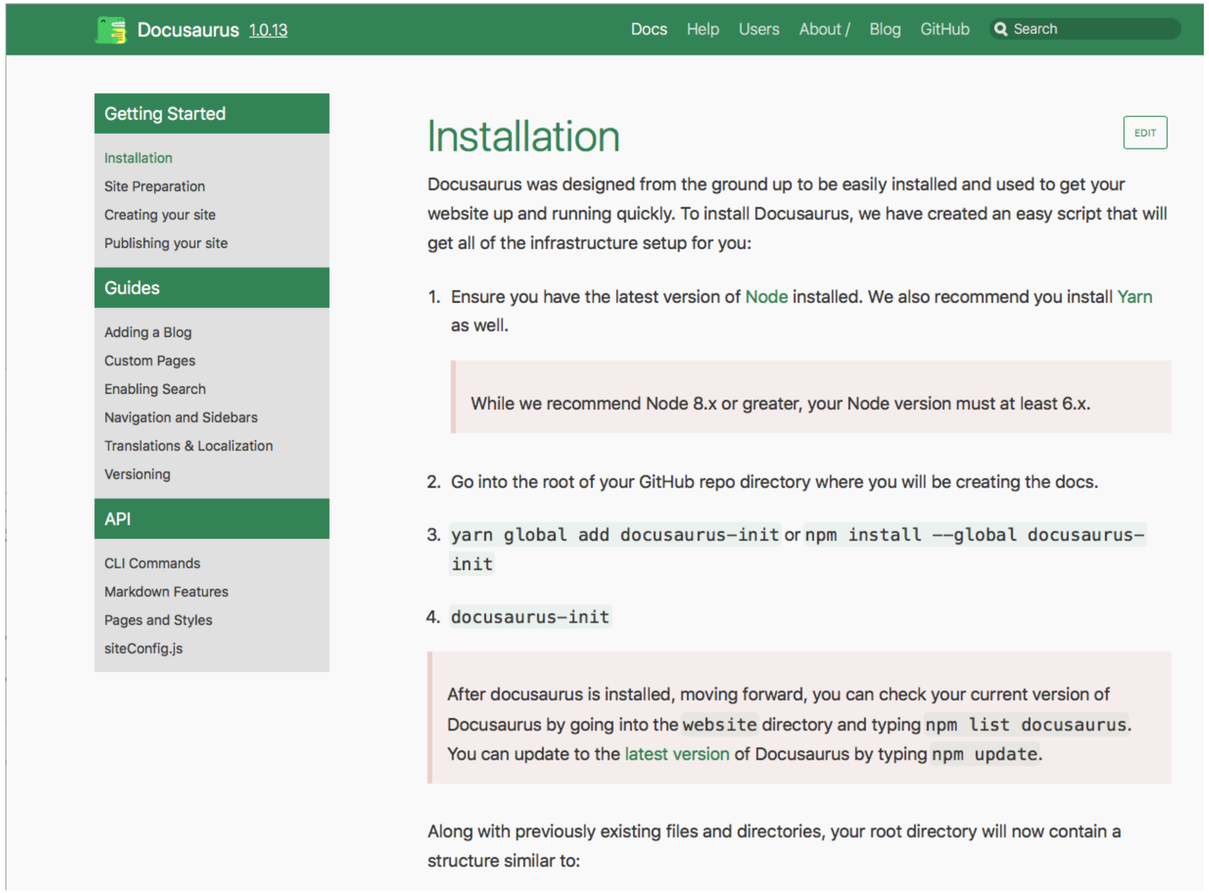
VuePress

https://dev.to/nafis/5-free-static-documentation-generators-you-must-check-out-5ged

Static-site generators

- From Markdown to beautiful websites out of the box





Reproducible Software

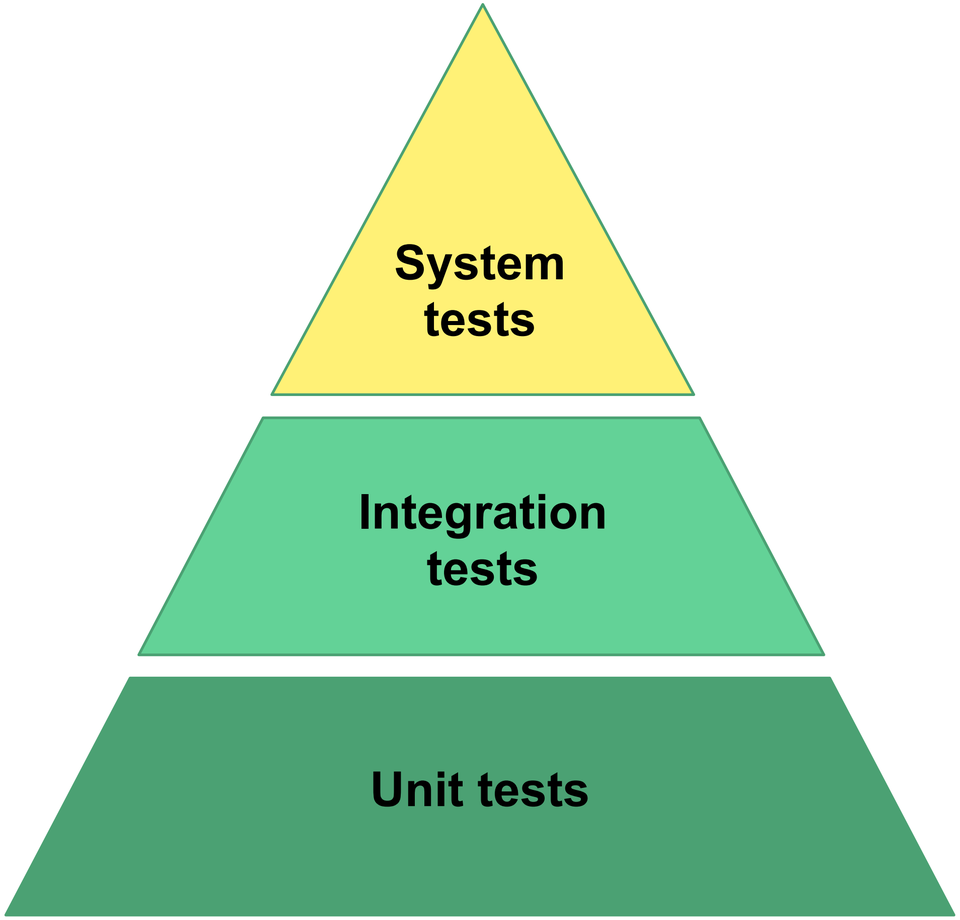
- Version Control
- Ease of use
  - Readable
  - Modular
  - Documentation
- **Rigor**
  - Testing
  - CI
- Packaging code
  - Cross platform
  - Containerization

What can happen when you don’t test?

- Navigation software for Ariane-5 had been untested
- Software converted 64-bit floating-point data into 16-bit unsigned integer values
- Arithmetic overflow

Testing

- Prevent bugs in software before they happen
- Make sure discovered bugs don’t happen again
- Make major updates go more smoothly, i.e. less debugging

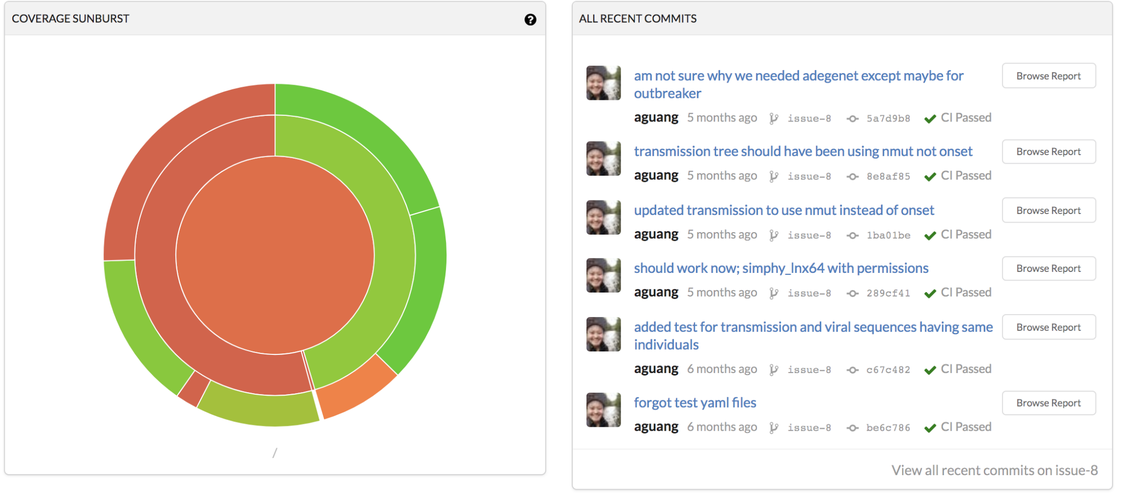


pytest

test that

Testing

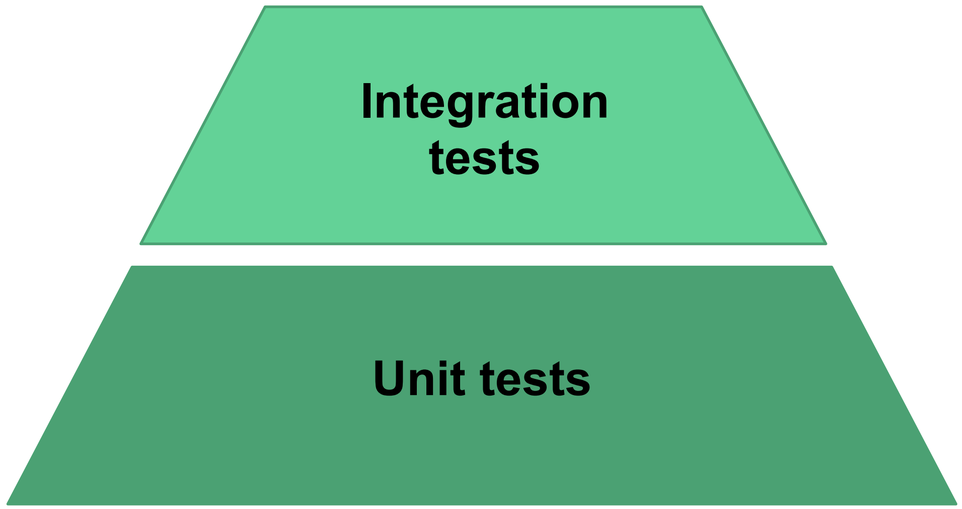
- Unit tests
  - Test each function/component



Files					Coverage
tests	258	247	0	11	95.73%
transmissim	308	165	0	143	53.57%
setup.py	2	0	0	2	0.00%
Project Totals (8 files)	568	412	0	156	72.53%

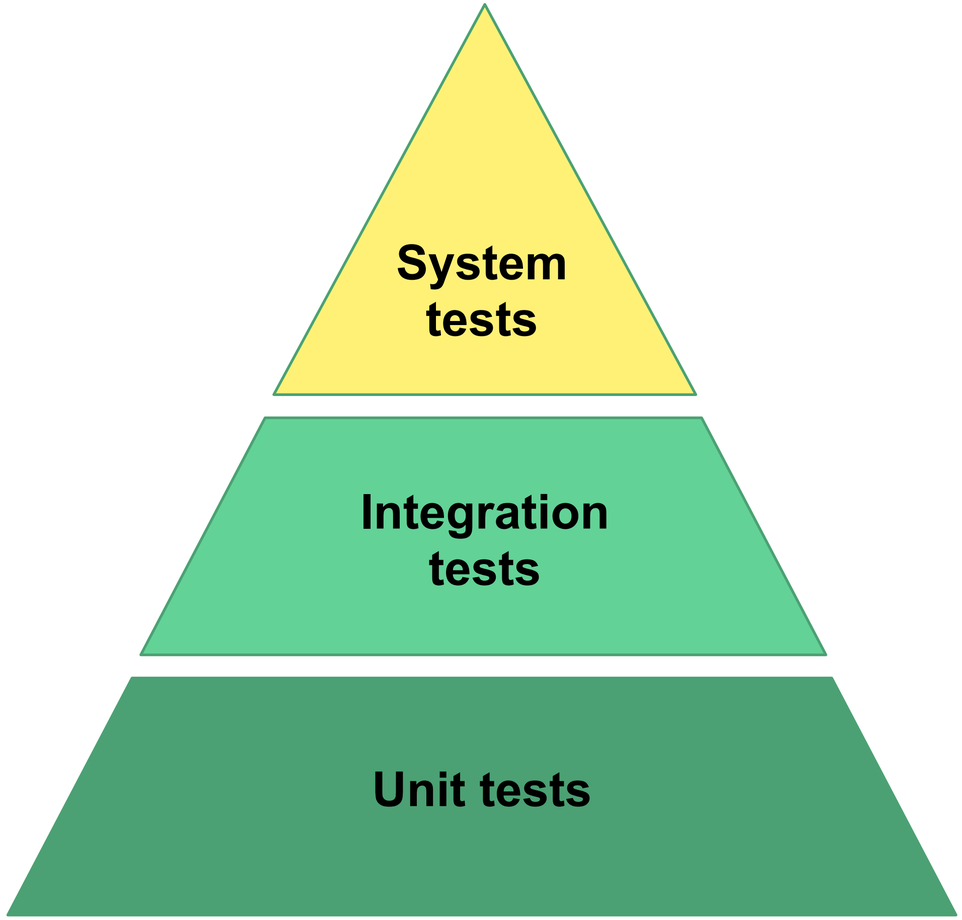
Testing

- Unit tests
  - Test each function/component
- Integration tests
  - Test multiple components together
  - Do they integrate properly with each other?
  - Starts to require test data

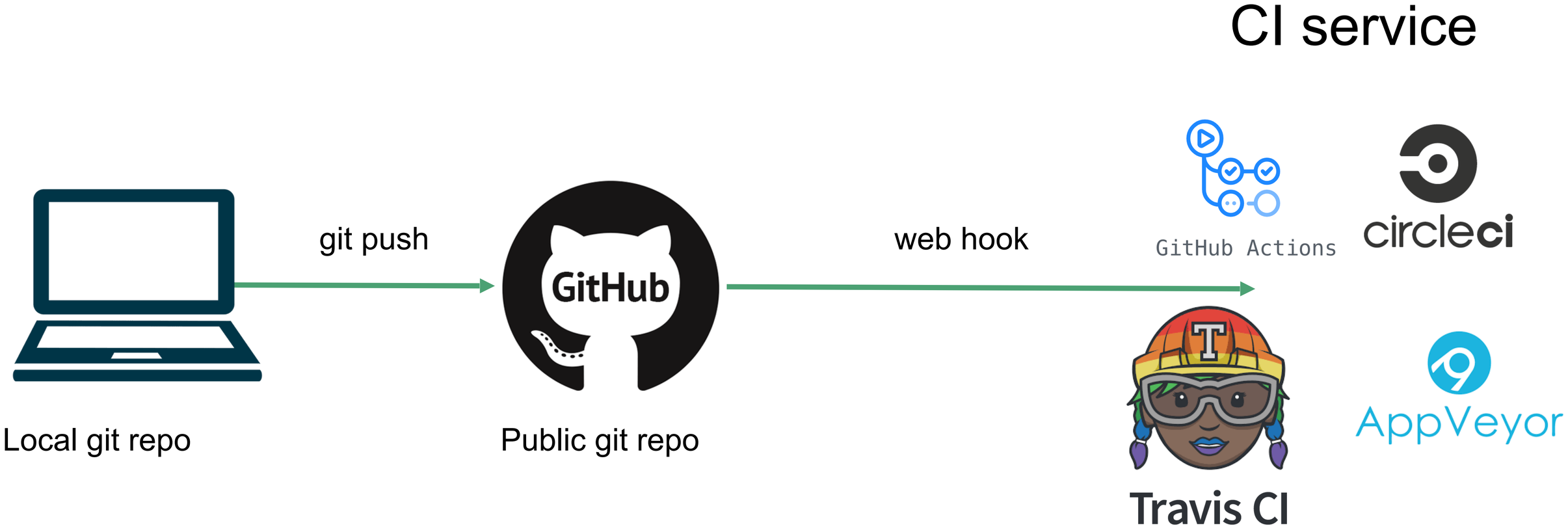


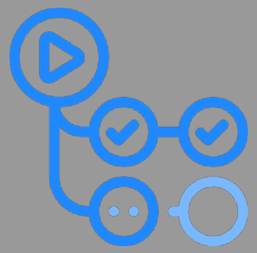
Testing

- Unit tests
  - Test each function/component
- Integration tests
  - Test multiple components together
  - Do they integrate properly with each other?
  - Starts to require test data
- System tests
  - Test entire system
  - Does system work as expected?
- At all levels, should also test that errors are handled correctly!



Continuous Integration

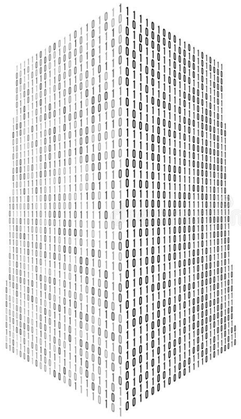




CI services automate all of this

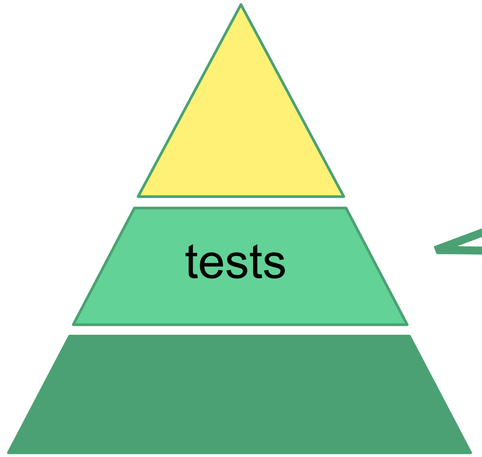
git clone

Virtual environment



CODE BUILD

build passing



build passing

deploy!

Build failing

Build failing

Go back & fix your code :(

Best Practices

- Version Control
- Ease of use
  - Readable
  - Modular
  - Documentation
- Rigor
  - Testing
  - CI
- **Packaging code**
  - Cross platform
  - Containerization

Packages

- Fundamental units of reproducible code
- Includes (ideally):
  - Functions
  - Documentation on how to use them
  - Sample data
  - Unit tests
  - Stable releases
- Often packages on repositories go through a code review that enforces stylistic standards and improves quality

How to package code?

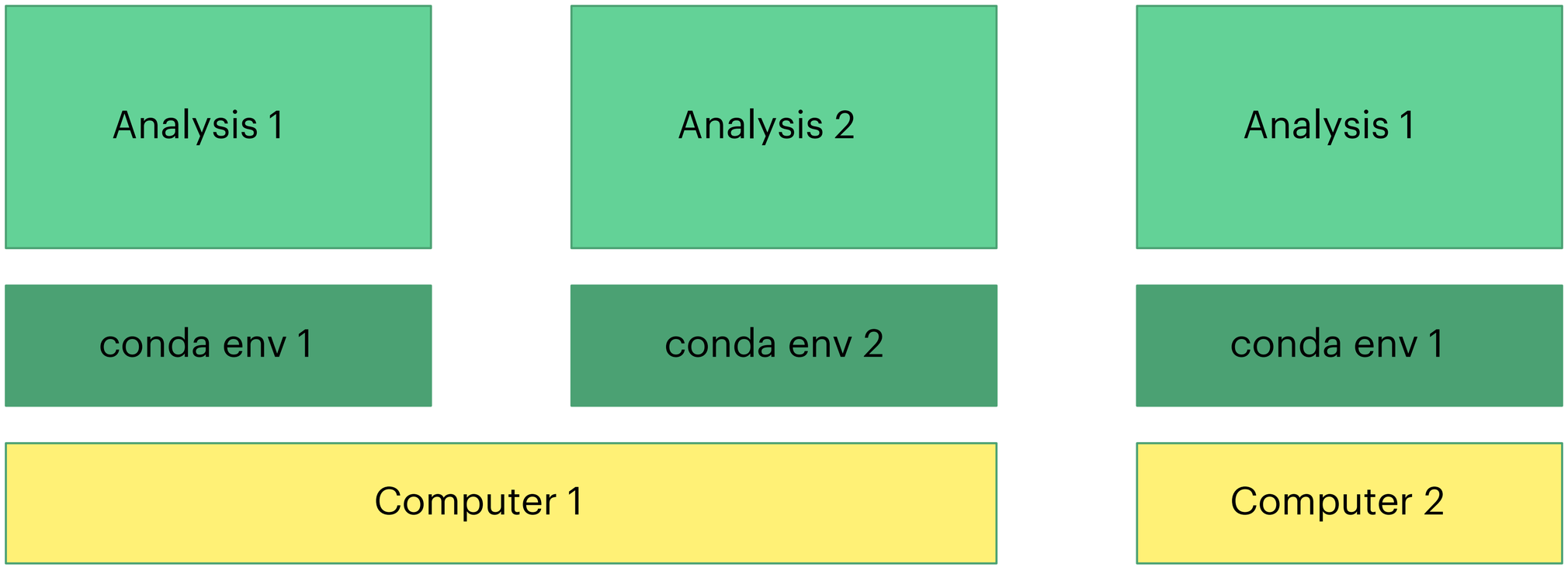
Packaging code

- Local development
  - Virtual environments
  - Dependency management
  - Packaging management

pyenv + Poetry

Welcome to Python's HELL

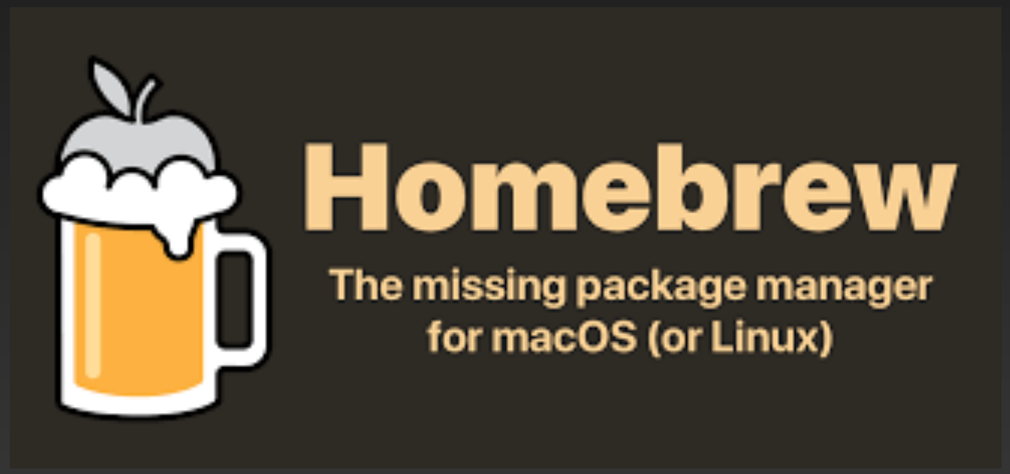
Open-source cross-platform
package & environment manager



Data science’s favorite

- Very popular for data science
- Requires packages to be available in CONDA's registry
- Can install PYPI installs (can create conflicts) `pip install pkg`
- Typically people publishes first to PYPI and then to CONDA
- It's more than a package manager, it can link to different low-level libraries (libc)
- Can be slow (recent improvements and c++ alternatives)

General package manager



pyenv manages you python versions

Installation	`brew install pyenv`
List versions of Python available to install	`pyenv install --list`
Install most recent stable version	`pyenv install 3.8.1`
Activate global environment	`pyenv global 3.8.1`
Listing available versions to the system	`pyenv versions`
Configure a local environment (.python-version)	`pyenv local 3.8.0`

pipX allows you to install Python CLI utilities in their own environment

Installation

```
python -m pip install pipx
```

Poetry handles dependency and virtual-environment-management in a way that’s very intuitive.

Installation

```
pipx install poetry
```

Create a new project

```
poetry new poetry-demo
```

Use virtual environments inside the project's root directory

```
poetry config virtualenvs.in-project true
```

Making sure that Poetry is using Pyenvs python

```
poetry env use $(pyenv which python)
```

Poetry

```
poetry new poetry-demo
```

```
$ poetry add pendulum
```

```
poetry-demo
├── pyproject.toml
├── README.rst
├── poetry_demo
│   └── __init__.py
└── tests
    ├── __init__.py
    └── test_poetry_demo.py
```

```
[tool.poetry]
name = "poetry-demo"
version = "0.1.0"
description = ""
authors = ["Sébastien Eustace <sebastien@eustace.io>"]

[tool.poetry.dependencies]
python = "*"

[tool.poetry.dev-dependencies]
pytest = "^3.4"
```

```
poetry install
```

If there is no poetry.lock file present, Poetry resolves all dependencies listed in your pyproject.toml file and downloads the latest version of their files.

```
$ poetry build

Building poetry (1.0.0)
- Building sdist
- Built poetry-1.0.0.tar.gz

- Building wheel
- Built poetry-1.0.0-py2.py3-none-any.whl
```

Poetry

```
$ poetry publish

Publishing poetry (1.0.0) to PyPI

  - Uploading poetry-1.0.0.tar.gz 100%
  - Uploading poetry-1.0.0-py2.py3-none-any.wh
```

Detour: GitHub Actions Example

https://github.com/python-poetry/poetry

Reproducible Software

- Source code
- Third-Party libraries
- Runtime environment

Source Code

- Captured in version control
- Releases tagged
- Immutable tags

Third-Party Libraries

- Captured in a package manifest file
- Libraries pegged to specific version
- Peg libraries to one version, not a range
- Select libraries that are maintained and have a corporate sponsor

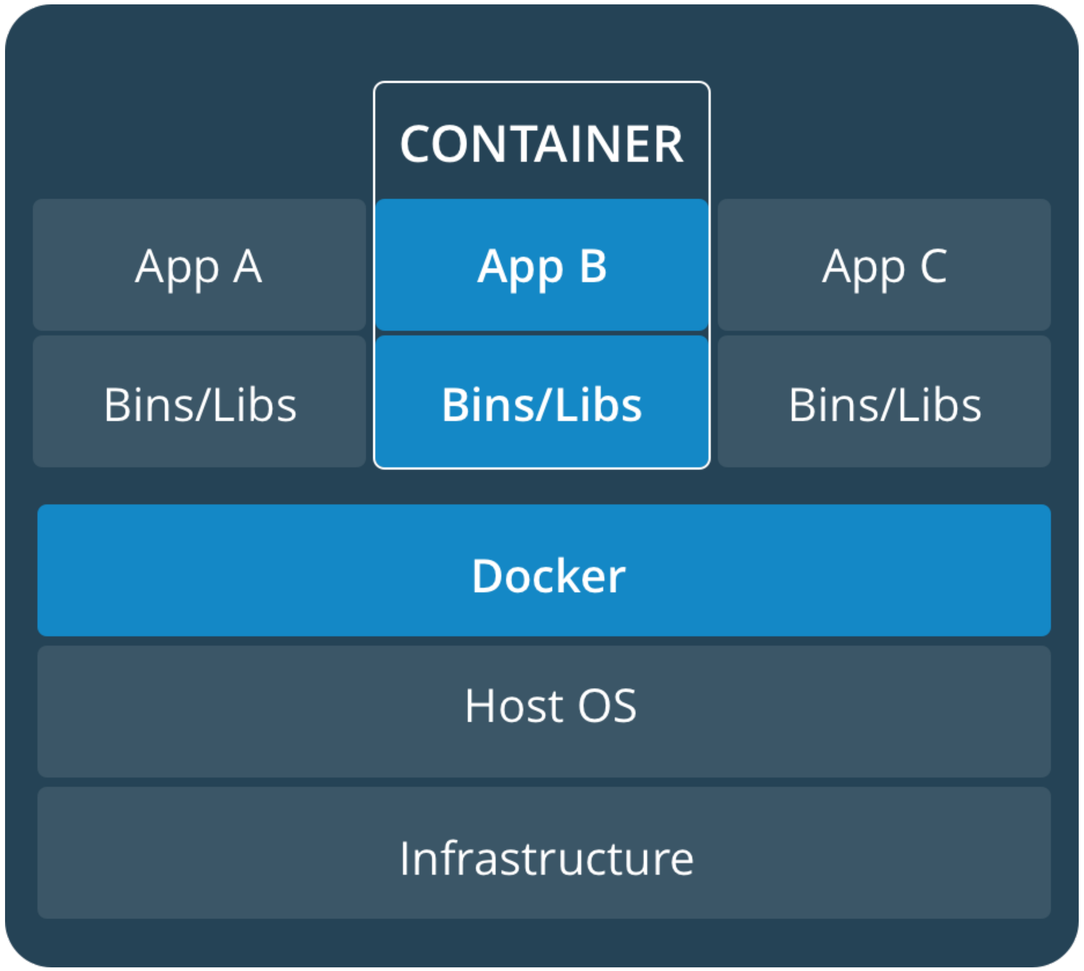
Runtime environment

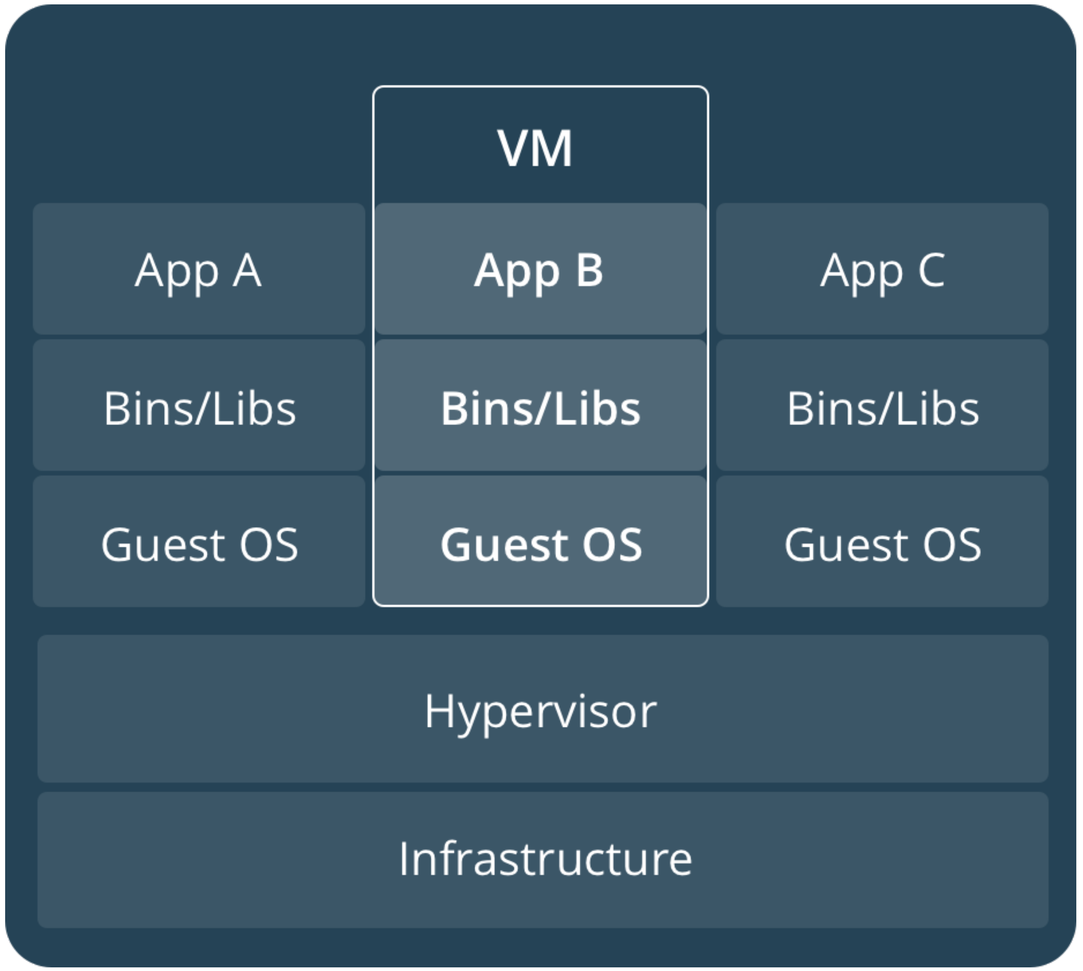
- Capture system software requirements
- Freeze runtime environment
- Package runtime environment for consumption

docker

podman

Docker vs. VM





source: docs.docker.com

Docker - Creating an Image

There are several ways to create a Docker image:

- dockerfile (write code yourself)
- repo2docker (create an image from a github repo)

Once created, images can be shared, either using the central ‘DockerHub’ repository, GitHub or by exporting and sharing a tarball

Dockerfile

```
-FROM python
+FROM python:3.8

 RUN apt-get update && apt-
get install -y \
     libxml2-dev \
```

```
-FROM python:3.8
+FROM python:3.8.0-buster

 RUN apt-get update && apt-
get install -y \
     libxml2-dev \
```

Always specify full version for images

Dockerfile

```
 FROM python:3.8.0-buster

 RUN apt-get update && apt-get
install -y \
-    libxml2-dev \
-    libsqlite3-dev
+    libxml2-dev=2.9.4+dfsg1-7+b3
\
+    libsqlite3-dev=3.27.2-3

 COPY requirements.txt .
 RUN pip install -r
requirements.txt
```

Always specify full version for images

Always specify versions of system software

Dockerfile

```
 WORKDIR /app
 COPY . ./

+ENV RANDOM_SEED=1024
 ENTRYPOINT ["python"]
-CMD ["my-script.py"]
+CMD ["my-script.py", "special",
"arguments", "here"]
```

Always specify full version for images

Always specify versions of system software

Capture arguments and env variables

Gotchas

- Understand the OS your image uses
- Different operating systems have different packages
- Alpine linux uses a different libc (glibc vs musl)

Building Containers

Build from Dockerfile in current directory

```
docker build .
```

Build and give a name

```
docker build -t science
```

Build, name and tag

```
docker build -t science:1.0.7
```

Reproducible Data

- Make it available!
- Keep track of transformations, data-provenance

GIT-LFS or GIT ANNEX allow you to version control your data

Installation

```
brew install git-lfs
```

Set up git-lfs for your user

```
git lfs install
```

Indicate which files git-lfs should track

```
git lfs track "*.psd"
```

Add .gitattributes file

```
git add .gitattributes
```

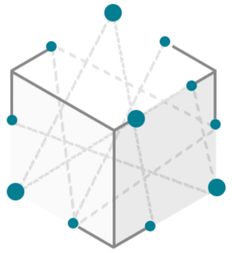
Add, commit and push as usual

```
git add file.psd
git commit -m "Add design file"
git push origin master
```

kaggle

FAIRsharing.org
standards, databases, policies

Leverage domain-specific data repositories



OpenNEURO

The Dataverse® Project

GIN

Modern Research Data Management for Neuroscience

...distributed version control, flavoured for science

Reproducible Workflows

Snakemake

```
# Count words.
rule count_words:
    input:    'books/isles.txt'
    output:   'isles.dat'
    shell:    'python wordcount.py
books/isles.txt isles.dat'
```

- Keep track of all the steps performed in an analysis
- Bash scripts are not an efficient way of storing a workflows

Reproducible Data + Workflow + Parameters

Track data

```
git add train.py
dvc add images.zip
```

Connect code and data by commands

```
dvc run -d images.zip -o images/ unzip
-q images.zip
dvc run -d images/ -d train.py -o
model.p python train.py
```

Make changes and reproduce

```
vi train.py
dvc repro model.p.dvc
```

Share code

```
git add .
git commit –m 'The baseline model'
git push
```

Share data and ML models

```
dvc remote add myremote –d s3://
mybucket/image_cnn
dvc push
```

Questions

Phew! That was a lot!