Workshop Reproducibility for Big Data Science: Git, Conda, Docker - advanced

Lukas Heine, Sameh Khattab

Disclaimer:

Opinions are our own and not the views of our employer.

DO NOT distribute these slides or their content without our consent.

Who am I?

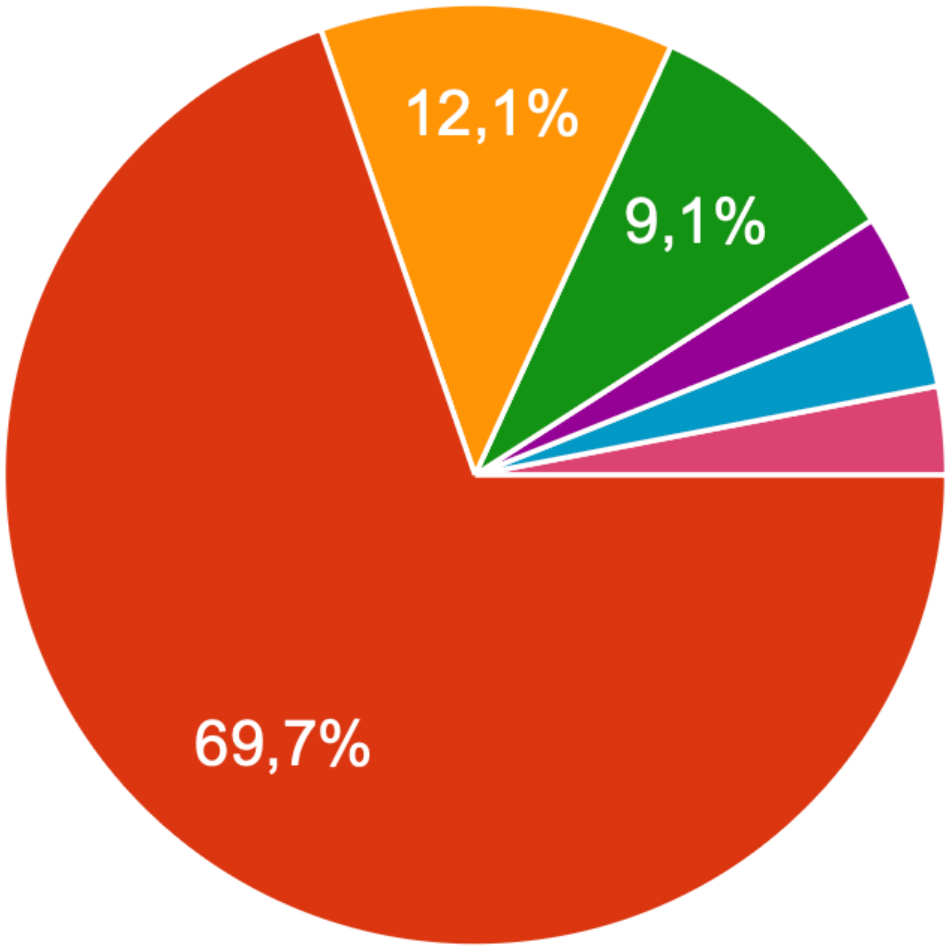
Lukas Heine

- Fraunhofer IMS & IIS, Siemens Healthineers
- Final year PhD Candidate @ IKIM
- https://mml.ikim.nrw

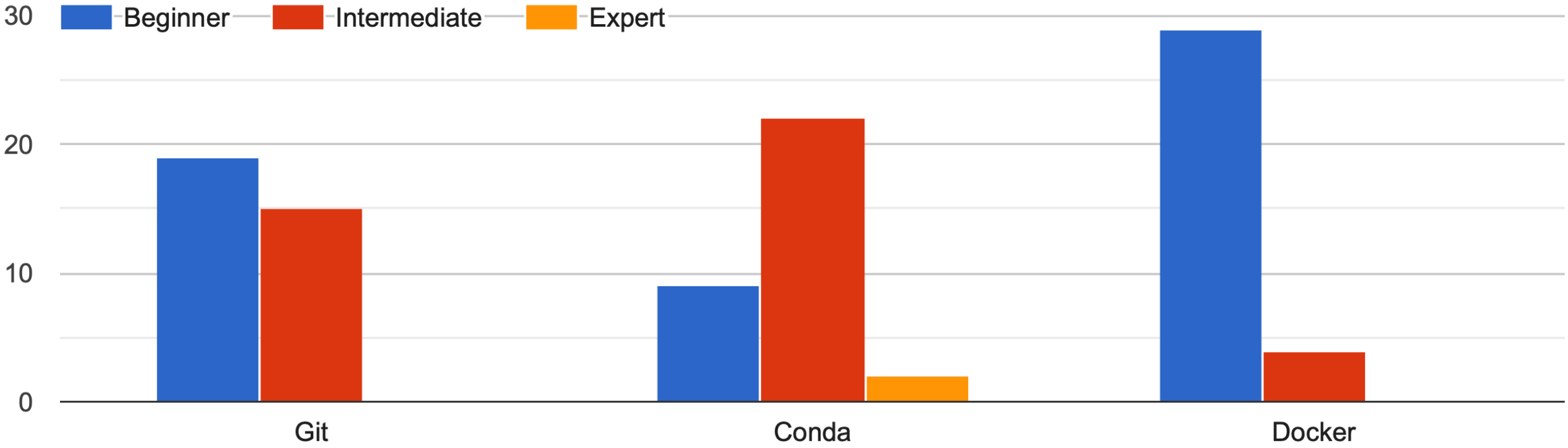
More importantly...

Who are you?

What do you expect from this workshop?



- Medicine
- Natural Science
- Computer Science
- Psychology
- Medical Engineer
- Computational Neuroscience
- Computational Biology


30
20
10
0
Beginner
Intermediate
Expert
Git
Conda
Docker

What struggles do you encounter in working with data and/or code in your daily work? (excerpt)

- *"Dependency management and conflicts. Cluttered global environment. Reproducibility issues"*
- *"When I want to use the code I used before, I can not find them. And everytime I want to install a new software, always take me lot of time, and make my previous software doesn't work."*
- *"docker containers being in the wrong architecture. (on a apple silicone mac) creating easy sharable dashboards for data visualization"*
- *"Managing diligently changes to my code, making meaningful comments to my commits and versioning the repos. Keeping really good notes of my methods, results, and justifications that lead to changes of methodology."*

Which topics would you like to learn about in this workshop? (excerpt)

- *"I am curious about the tools mentioned and see how they are applied."*
- *"CI/CD and testing"*
- *"Git, Docker"*
- *"docker containers like the streamlit we did but maybe also for developing (devpod seems interesting)"*
- *"Things that docker can be used for that are maybe not so common"*

Overview

Topic	Time
Problem settings	? - 10:30
Code versioning & experiment tracking	10:30 - 11:30
Managing environments	11:30 – 12:00
Lunch	12:00 - 13:00
Managing environments II	13:00 – 13:15
Containers, registries, docker compose	13:20 - 14:30
Summary & Hands-on	14:45 – 16:45
Wrap-up	16:45 - 17:00

A note on the style of this workshop

3 Resources:

1. Slides (available in Git: https://github.com/code-lukas/cologne-workshop-advanced)
2. Recording (Part I, Part II (this session) available later)
3. Git repository

⚙️ Slide provides opportunity for interaction

Quiz:

https://iccb-workshop-quiz.streamlit.app

Problem setting I

"*It works on my machine*"

"*Do not touch this piece of code*"

"*This used to work*"

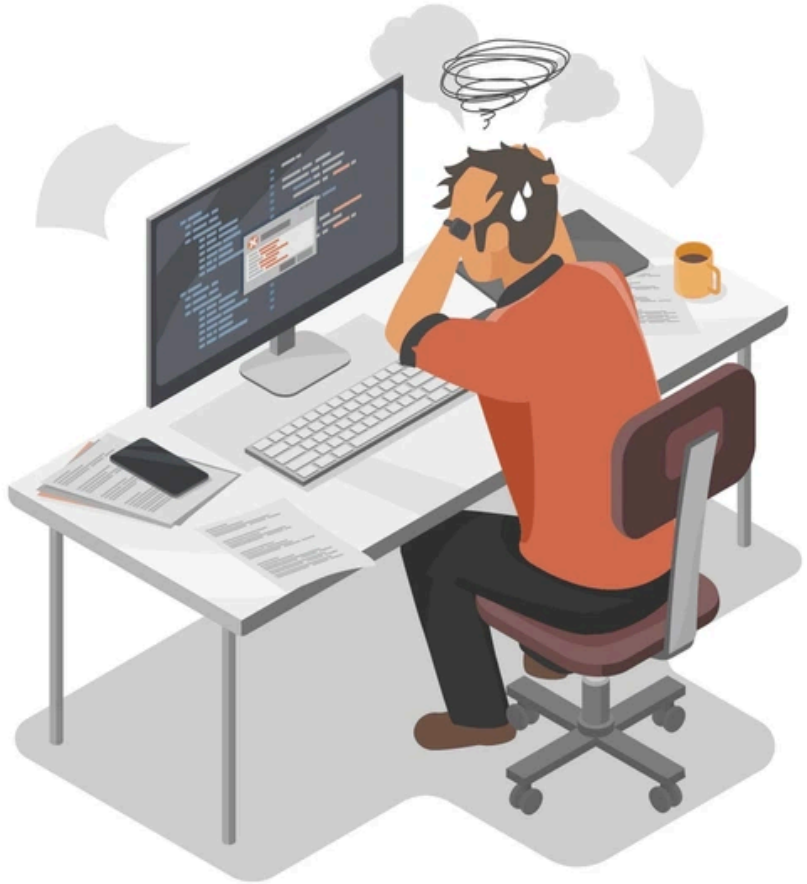
"*I have a backup on a USB drive*"

"*As long as it runs...*"

"*Warnings are not errors*"

"*It's just me who looks at this code anyways*"

"*Deployment is something that the IT people can look at*"



Problem setting II

Tools such as Git, Conda and Docker are designed to make your life as a researcher, software developer and scientist easier. This benefit comes at the cost of added complexity and overhead in project planning. Your objective is to understand what role each tool plays, what they **should** be used for and also what they **shouldn't** be used for.

Problem setting III

No tools in the world will save you from yourself. If you design a complex, undocumented, hard-to-maintain system, be aware of the risks this entails.

Hard truths from research & industry

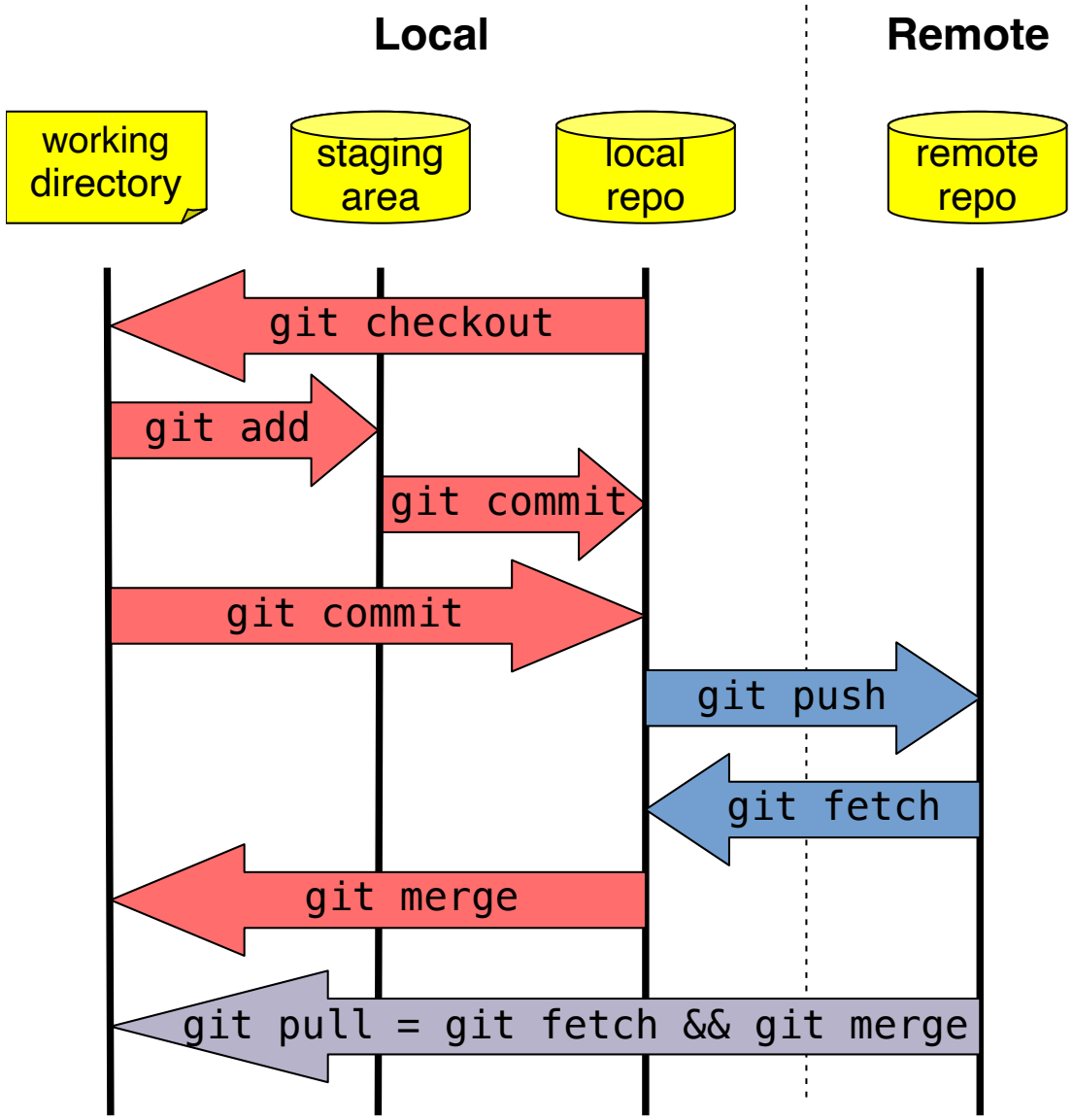
- **If there is no Git, it does not exist**: Few people - if any - in our domain will bother reimplementing your method. Some venues will require a well-documented repository. Some frameworks are popular because they are easy to use and have good documentation.
- **Your public Git profile is like a modern business card**: Showcasing interesting, well-coded and well-documented projects eliminates potential discussions about your skillset.
- No other technology is as widely used as Git
- Git commits serve as a proof of work and contribution.

Git I

- Code versioning system, knowledge base, issue tracking, project management
- Work is organized in a **repository**
- A repository consists of one or more **branches** (default: **main** (master))
- Contributions are tracked in **commits**
- **Local** git repositories & **Remote** repositories

Git II

- Repositories are either **public** or **private**
- Code can be accessed using a **fork** (independent copy) or a **clone** (linked copy that keeps a reference to the target repository)
- **HTTPS** or **SSH** can be used to access repositories



Git III : Rebasing vs. merging ⚙️

- If you have strong opinions on this, have at it!

Pushing an existing folder

```
cd existing_folder
git init --initial-branch=main
git remote add origin git@your.git.here:your_namespace/repo_name.git
git add .
git commit -m "chore: Initial commit"
```

Pushing an existing git

```
cd existing_repo
git remote rename origin old-origin
git remote add origin git@your.git.here:your_namespace/repo_name.git
```

Try it out! ⚙️

Fork the following repository: https://github.com/code-lukas/cologne-workshop-advanced

Linking a pull request to an issue

Read more

Gitfalls

- Working on the same file at the same time in the same branch
- Accidentally committing large files
- Liberal use of `--force`
- Pushes to the `main` branch
- Not writing tests
- Not using branches
- Not pulling frequently
- No project structure
- No `README.md`

Regressions

- How do find the commit where a given feature last worked?
- `git checkout`?
- `git bisect`!

Binary search for the bad apple

```
git bisect start
git bisect bad # Current version is bad
git bisect good COMMIT_SHA # COMMIT_SHA is known to be good
```

Project structures

- The folder structure of your project largely depends on your domain and toolstack
- *Keep seperate thing seperate* (e.g. preprocessing and analysis)
- *Store configurations in a seperate folder, parse configs from this folder*
- *For configs, use readable formats like YAML or TOML*
- *Never type numbers* (admittedly controversial, but there is some truth to this)
- *Keep logs of your experiments, do not just write to stdout*
- *"Modern" experiment tracking with Weights & Biases*

Writing a helpful README ⚙️

- What do you consider good examples of READMEs?
- What do you expect to find?
- What do you find difficult when writing a README?

Completeness vs. brevity

1. Title and summary
2. Installation
3. Usage
4. Project structure (optional)
5. Testing
6. Contributing
7. License
8. Paper? Citation, acknowledgements

Optional

- API documentation
- Roadmap or changelog
- FAQ / Troubleshooting
- Screenshots or demo GIFs

Issue templates ⚙️

- Store in `.github/ISSUE_TEMPLATES`
- Different templates for different purposes, e.g. bug reports, regressions, feature requests, etc.
- Low effort templates attract low effort issues

Git for project / research management

Closed Bug #6

abustany opened on Jul 8

Describe the bug
It looks like the code assumes and case-insensitive filesystem

To Reproduce
Detailed steps to reproduce the behavior:

- Have a nifti filename called `aNiftiFilename.nii.gz`
- Run `mede-deidentify --verbose -i ~/aNiftiFilename.nii.gz -o . --deface`

Expected behavior
Things work

Error logs
I get an error

```
Traceback (most recent call last):
  File "/opt/pytorch/lib/python3.12/site-packages/torch/utils/data/_utils/worker.py", line 349, in _worker_loo
    data = fetcher.fetch(index)  # type: ignore[possibly-undefined]
           ^^^^^^^^^^^^^^^^^^^^
  File "/opt/pytorch/lib/python3.12/site-packages/torch/utils/data/_utils/fetch.py", line 52, in fetch
    data = [self.dataset[idx] for idx in possibly_batched_index]
            ~~~~~~~~~~~~^^^^^
  File "/home/ec2-user/medical_image_deidentification/mede/dataset.py", line 354, in __getitem__
    nifti_img = nib.load(file_path)
                ^^^^^^^^^^^^^^^^^^^
  File "/opt/pytorch/lib/python3.12/site-packages/nibabel/loadsave.py", line 103, in load
    raise FileNotFoundError(f"No such file or no access: '{filename}'")
FileNotFoundError: No such file or no access: '/home/ec2-user/aniftifilename.nii.gz'
```

Additional context

Assignees
code-lukas

Labels
bug

Type
Bug

Projects
No projects

Milestone
No milestone

Relationships
None yet

Development
Code with agent mode
fix: addresses issue with upper-case filenames
TIO-IKIM/medical_image_deidentification

Notifications
Customize

32

Remponator assigned code-lukas on Jul 9

code-lukas linked a pull request that will close this issue fix: addresses issue with upper-case filenames #6 on Jul 9

code-lukas mentioned this on Jul 9

fix: addresses issue with upper-case filenames #6

code-lukas closed this as completed in #6 on Jul 9

code-lukas on Jul 9 — Member

Thanks for bringing this up, I tested this with the same filename as your example - it should work properly now. Feel free to reopen this if you encounter any issues.

abustany on Jul 9 — Author

Thanks for the quick fix!

Excursion: Code styles & linting ⚙

Example

Miscellaneous ⚙️

- `git config --global alias.{ALIAS_NAME} '{COMMAND}'`
- `git reset --hard @{u}`
- `git log --date=short --pretty=format:%ad | sort | uniq -c | sort -r | head`
- `git log --author="author_name"`
- The `checkout` operation used to be severely overloaded (branch creation, switch branch, get a certain version of a file...)
- Easier: use `switch` and `restore`
- `git stash`

Frequently occurring files (hopefully) ⚙

- `README.md`
- `requirements.txt`
- `environment.yml`
- `Dockerfile`
- `docker-compose.yml`
- `.gitlab-ci.yml`
- `poetry.lock`, `Pipfile.lock`, `uv.lock`
- `pyproject.toml`
- `setup.py`
- `.gitignore`

.gitignore ⚙

Files and directories listed in this file **will not be considered in commits**. Include build files, large files or config files that contain sensitive information.

Keep your repositories small & safe, put everything in there that does not need to be shared.

Menial work

- Some tasks are always executed sequentially, e.g. make changes to a project, rebuild the project's docker image, push the image to a registry, deploy a container
- ... this gets repetitive really fast, for complex pipelines you might forget steps.

Solution: CI/CD

- **CI/CD**: Continuous Integration / Continuous Delivery
- Automate reocurring tasks using GitHub actions / GitLab pipelines, etc.
- Example use cases: application deployments, report generation

CI/CD prerequisite: runners

"Self-managed runners are GitLab Runner instances that you install, configure, and manage in your own infrastructure. You can install and register self-managed runners on all GitLab installations." - GitLab Docs

If your host system has an existing docker installation, you can use the offical gitlab runner image. This also means that you can have local, private runner on your device that handles tasks.

Let's try this! ⚙️

```
docker run -d --name gitlab-runner --restart unless-stopped
  -v /Users/lh/Desktop/gitlab-runner/config:/etc/gitlab-runner \
  -v /var/run/docker.sock:/var/run/docker.sock \
  gitlab/gitlab-runner:latest
```

- If you are interested in trying this yourself and need assistance, let's talk during the hands-on!

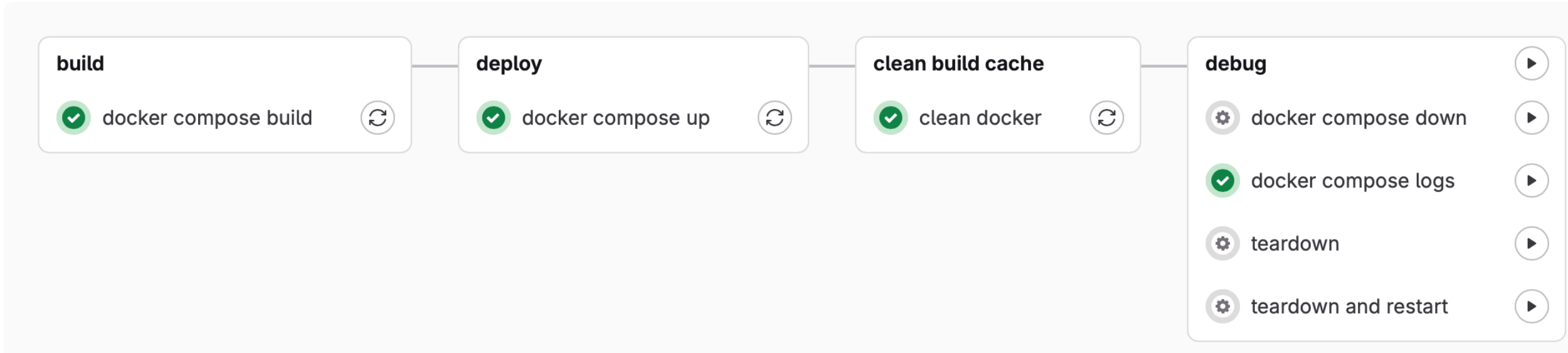
ci: more rigorous clean up, some ci fixes

Blocked **Lukas Heine** created pipeline for commit `dc468cbd`

For `master`

latest 8 jobs

Pipeline Jobs 8 Tests 0



Other ⚙

- Publicly available Git repositories are invaluable learning resources. Go to a famous repository and read their README, look at their project structure, read through their files, see how they code. What do they do differently? Why do they do it differently? What do you agree with? What are just opinions?
- Do not reinvent the wheel (unless you do it for learning purposes). GitHub is full of great projects that you can take inspiration from.
- Honorable mention: https://github.com/codecrafters-io/build-your-own-x

Questions?

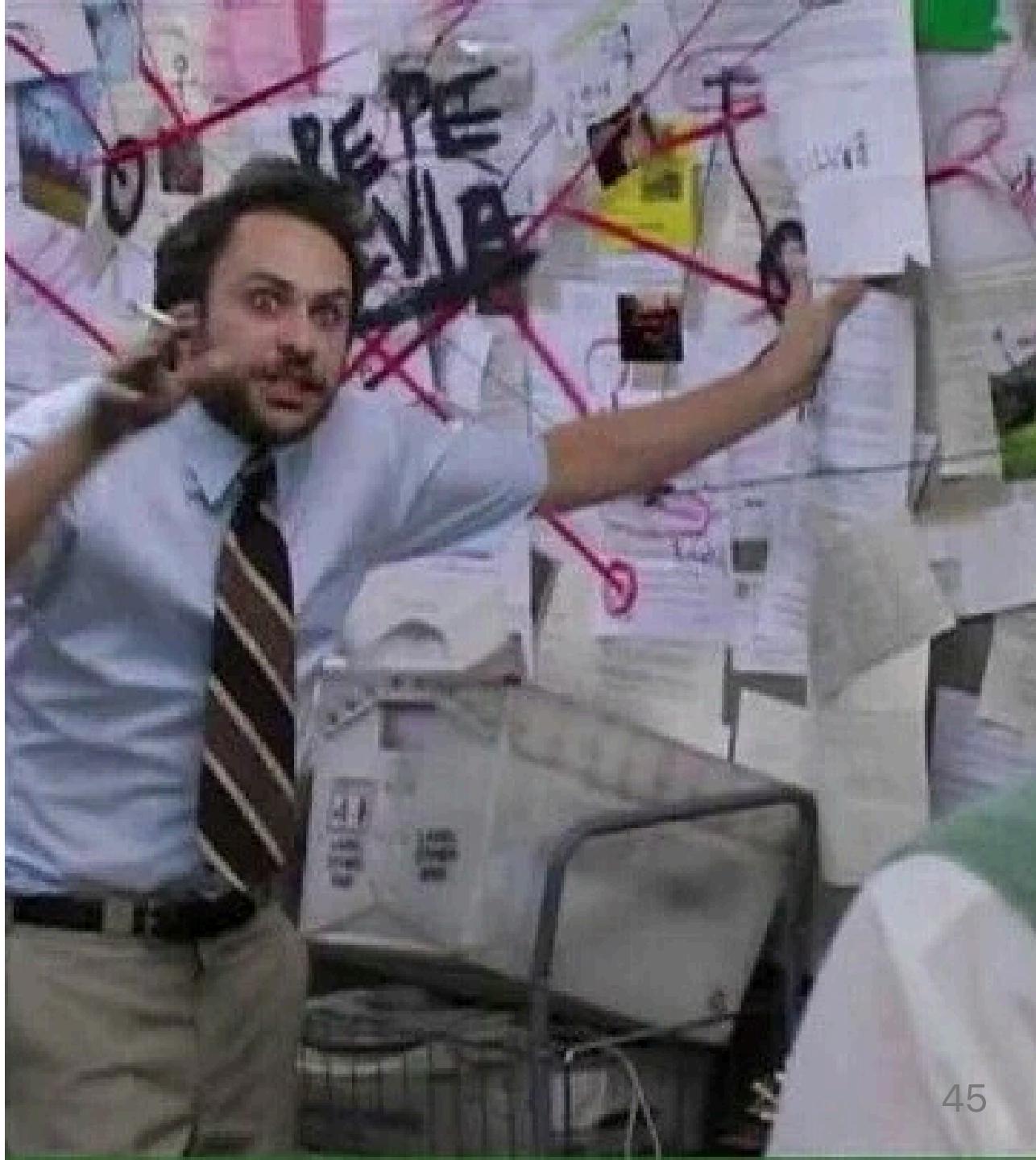
Git cheat sheet

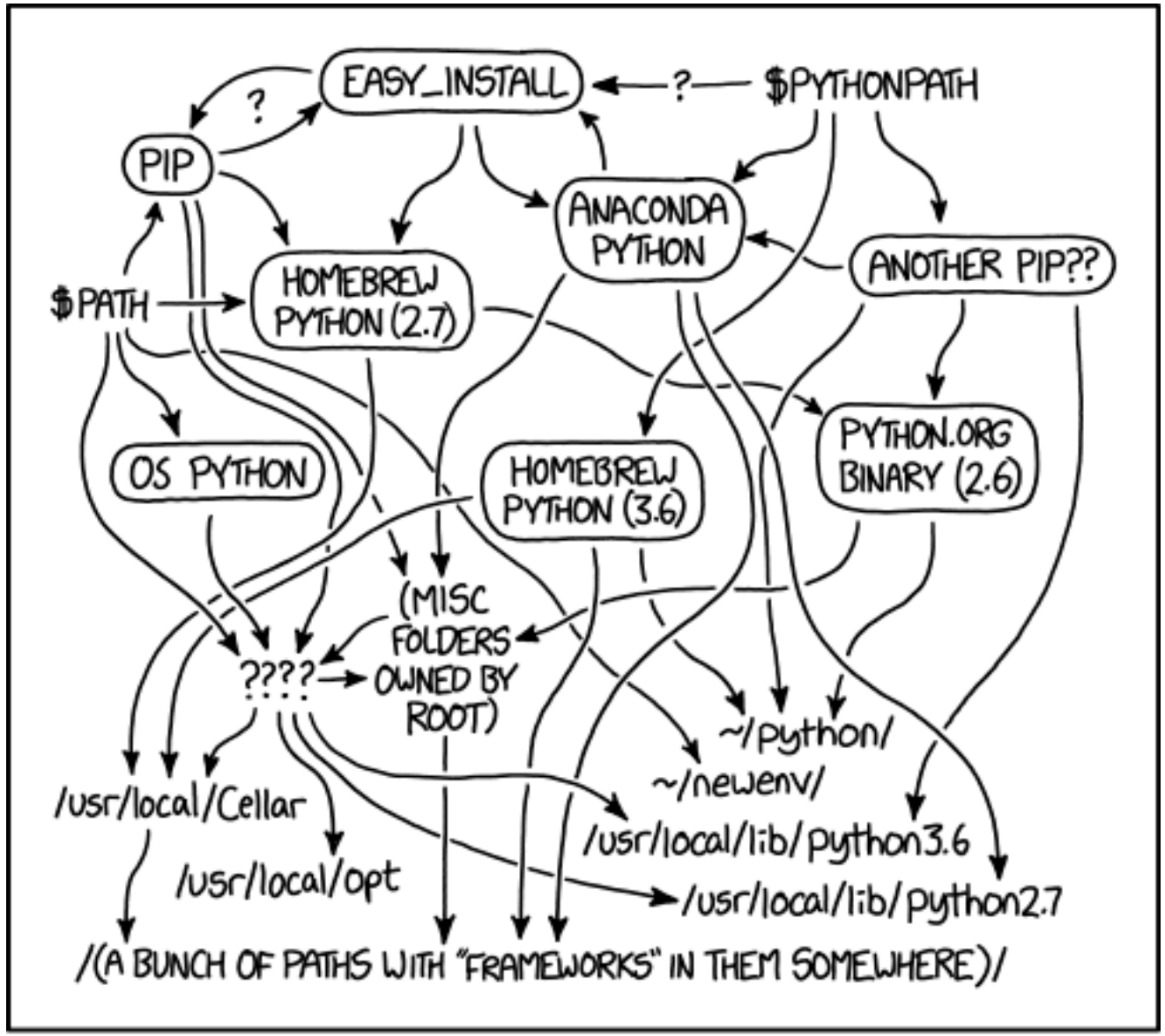
So You Think You Know Git?

Oh My Git!

Python packaging tools

- `Pip`
- `Poetry`
- `PDM`
- `Hatch`
- `Rye`
- `uv`
- `Conda`, `Mamba`
- `Pyflow`
- Not quite as bad as javascript frameworks, not great either

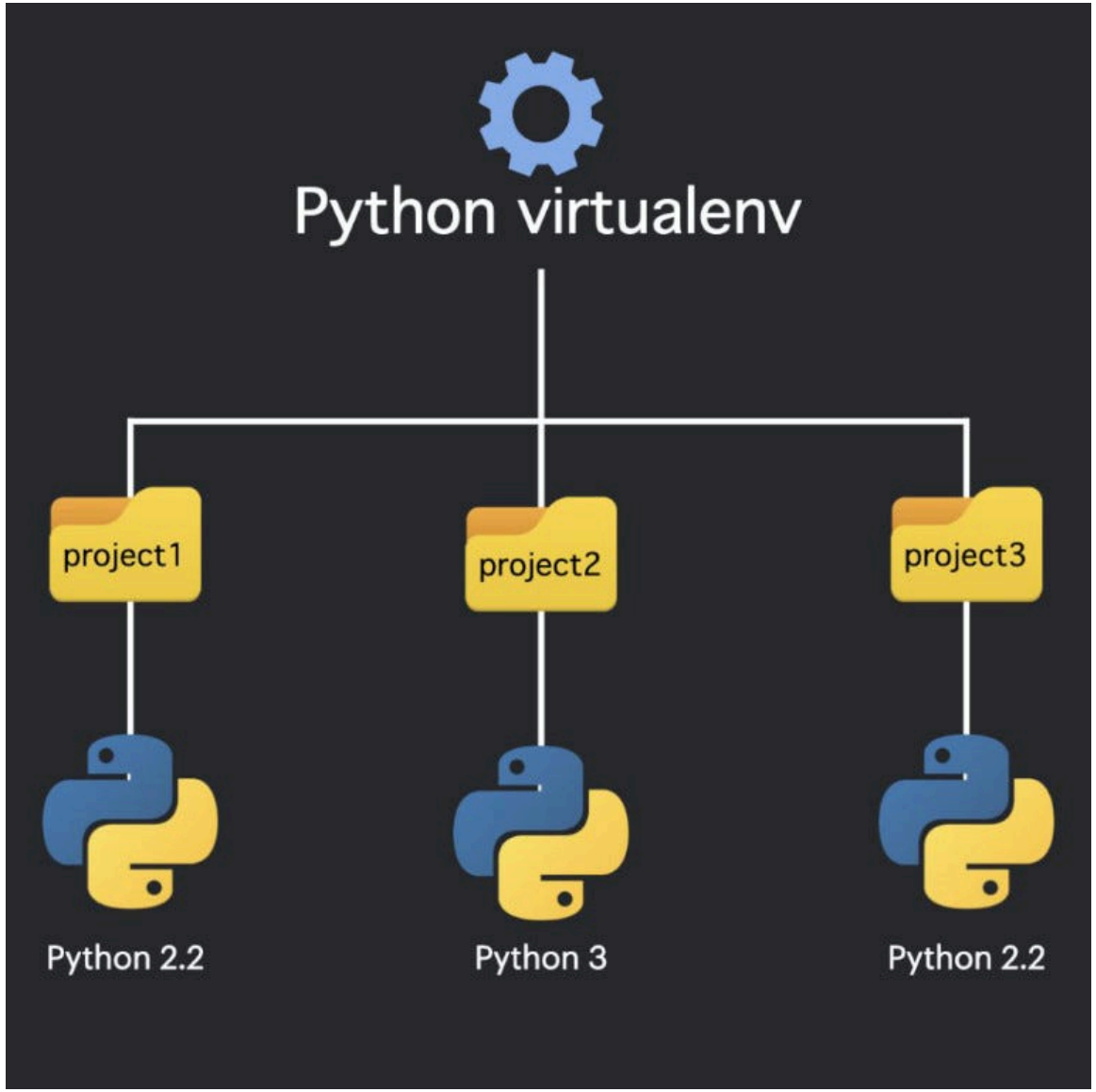




MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Virtual environments

- Problem setting: You work in multiple projects that each have their own respective requirements, packages, dependencies and programming languages.
  How do you keep track?



Conda - package & environment manager

- Conda manages both different evironments **and** serves as a package manager
- **Why is it popular?**
  Conda is able to resolve and bundle complex dependecies from non-python sources
- **What are the downsides?**
  Conda's default dependency resolver is *slow*. Faster than installing dependencies manually, but slow enough to grind your gears. People were so frustrated that they reimplemented `conda` in C++: Mamba

Conda Licensing Dispute (2024 - 2025)

Background

- **Anaconda, Inc.** introduced new licensing and access terms for Conda repositories.
- Commercial use of `repo.anaconda.com` now requires **paid enterprise subscriptions**.
- Authentication barriers disrupted automated and CI/CD workflows.

Key points of dispute

- **License shift:** From permissive open access → commercial restrictions.
- **Community backlash:**
  - Broke reproducibility in open research & education.
  - Sparked debate over “open source” vs. “free to use.”
- **Ecosystem response:**
  - Growth of **Conda-Forge** and **Mamba** and as open alternatives.
  - Push for fully FOSS dependency resolution and hosting.

Current status (late 2025)

- Anaconda keeps commercial licensing for its channels.
- **Conda-Forge** and **Mamba-Forge** act as de-facto open community hubs.
- Ongoing debate: *sustainability vs. openness* in package infrastructure.

Building your own package ⚙️

- Example `pyproject.toml`
- Full overview

Conda - getting started ⚙️

Creating an environment with a specific python version

- `conda create -n myenv python=3.12`
  You can install packages using `conda install ...`
- If you are on Apple Silicon you can alternatively use:
  `conda create --platform osx-64 --name python-x64 python=3.12`
  to produce an env native to x86_64

Conda - a *fictional* scenario

You have a conference deadline approaching, you work all night to make it.
The submission server is acting up. In a desperate attempt, you hunch over your mouse, send a prayer and spam the submit button. It finally goes through. You go to sleep.

A few months later, you get reviews - mostly positive, but reviewer 2 wants you to conduct "a few more experiments". You open your project, rerun your code and ... alas, it doesn't work!

Where did you go wrong?

Conda - good practices

1. Do not reuse envs for multiple projects that have different requirements.
2. After submissions or stable versions run `conda env export > environment.yml`
3. If you work on multiple systems, this may not be enough `conda list --explicit > spec-file.txt`

   Limitation: operating system platform needs to be identical

   Solution: use docker

```
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: osx-64
@EXPLICIT
https://repo.anaconda.com/pkgs/free/osx-64/mkl-11.3.3-0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/numpy-1.11.1-py35_0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/openssl-1.0.2h-1.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/pip-8.1.2-py35_0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/python-3.5.2-0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/readline-6.2-2.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/setuptools-25.1.6-py35_0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/sqlite-3.13.0-0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/tk-8.5.18-0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/wheel-0.29.0-py35_0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/xz-5.2.2-0.tar.bz2
https://repo.anaconda.com/pkgs/free/osx-64/zlib-1.2.8-3.tar.bz2
```

Good practices continued ⚙️

- After a submission or similar has been made, tag the current commit
  `git tag mysubmission <COMMIT_ID>`
- Personal preference: keep a `submission` branch

```
mkdir -p ./submission/
conda env export > ./submission/submission_environment.yml
conda list --explicit > ./submission/submission_spec_file.txt
git tag SUBMISSION $(git rev-parse HEAD)
```

What if I didn't .. ? ⚙️

- `conda list --revisions` and `conda install --revision <REVISION>` are your friends
- **Keep in mind that we are only talking about your code thus far!**
- But what about data?

When do I commit, when do I backup my environment, when do I build a container?

- Depends on how much (or little) you trust yourself
- Two extremes: **Never** (please don't do this) *or* each commit triggers a CI/CD worfklow that builds a new Docker container, tags it with the `CI_COMMIT_SHORT_SHA` and pushes it to a registry

Excursion: (not) seeding ⚙️

```
~ ➤ python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:19) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.random.random((3, 3))
array([[0.97629484, 0.75438467, 0.97059105],
       [0.68049933, 0.51270379, 0.42925281],
       [0.00193277, 0.25561221, 0.54453228]])
>>>
[5]  + 77795 suspended  python3
~ ➤ python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:19) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> np.random.random((3, 3))
array([[0.29347713, 0.13773941, 0.1256977 ],
       [0.65878117, 0.11126275, 0.2233628 ],
       [0.76567172, 0.69839965, 0.06252917]])
```

Excursion: seeding ⚙

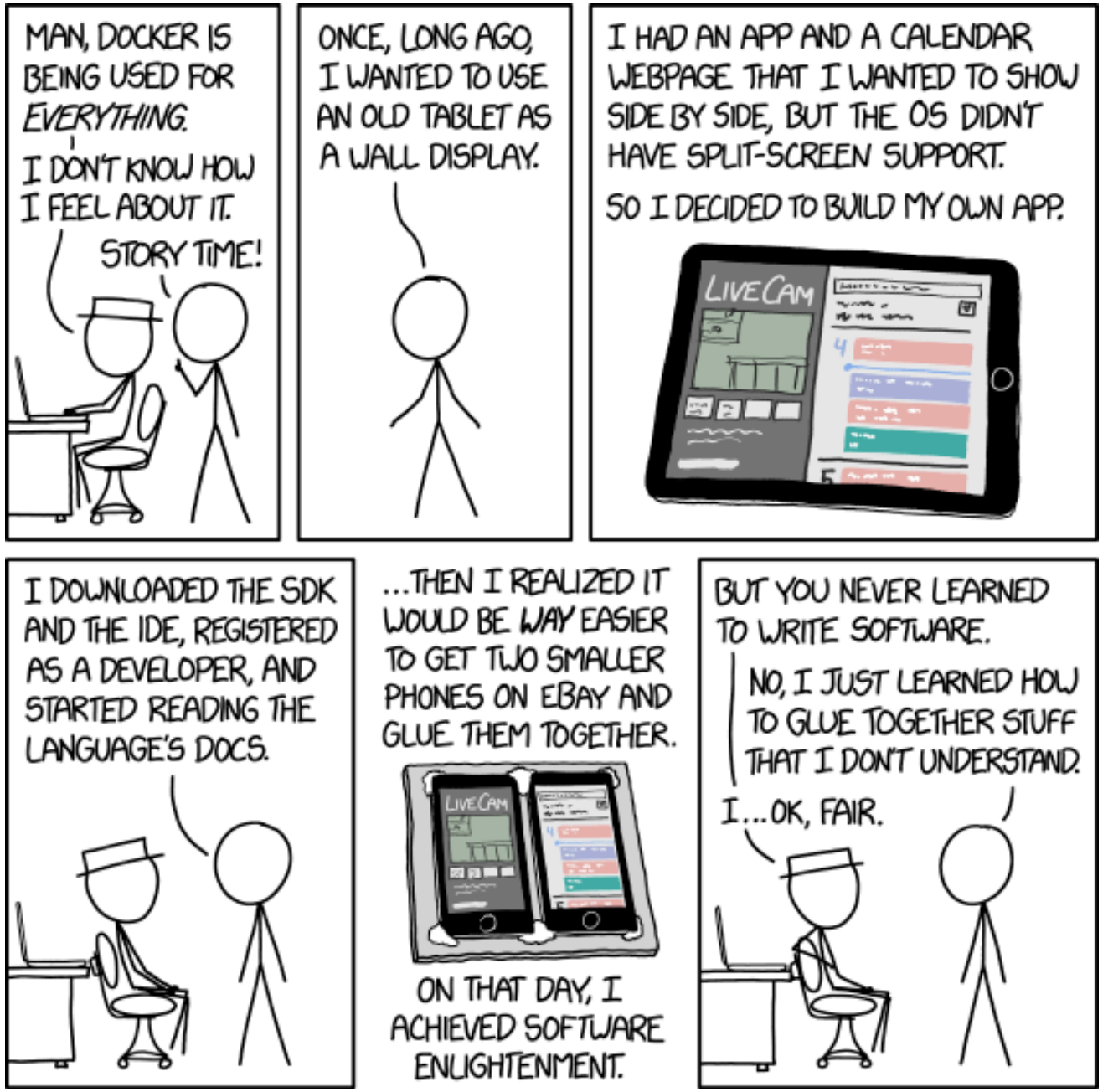
```
~ ➤ python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:19) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> rng = np.random.default_rng(seed=42)
>>> arr1 = rng.random((3, 3))
>>> arr1
array([[0.77395605, 0.43887844, 0.85859792],
       [0.69736803, 0.09417735, 0.97562235],
       [0.7611397 , 0.78606431, 0.12811363]])
>>>
[3]  + 77446 suspended  python3
~ ➤ python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:19) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> rng = np.random.default_rng(seed=42)
>>> arr1 = rng.random((3, 3))
>>> arr1
array([[0.77395605, 0.43887844, 0.85859792],
       [0.69736803, 0.09417735, 0.97562235],
       [0.7611397 , 0.78606431, 0.12811363]])
```

Questions?

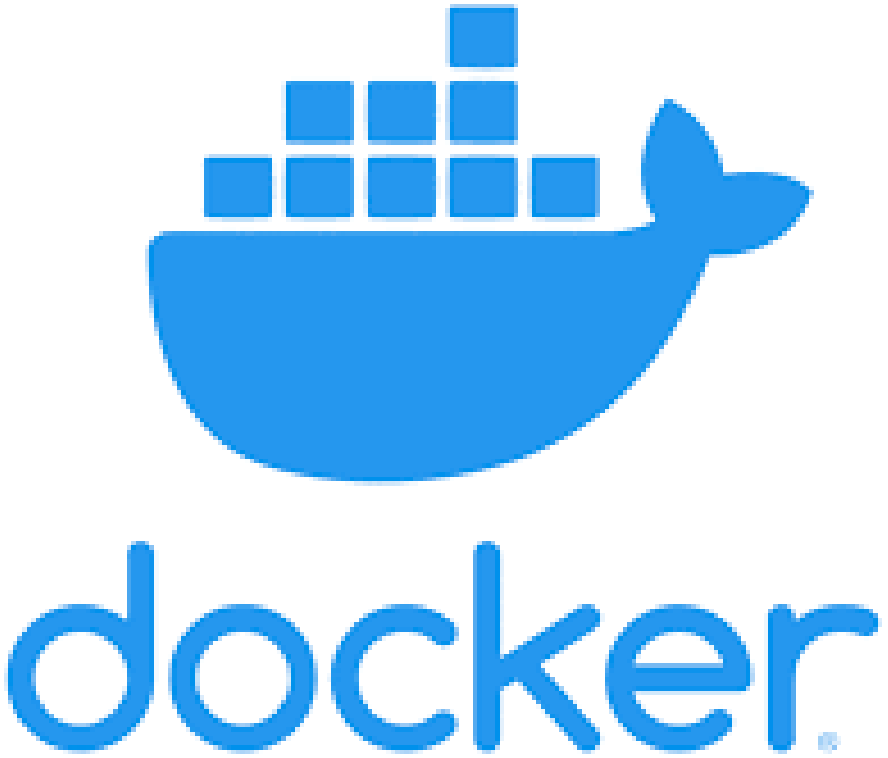
Getting started

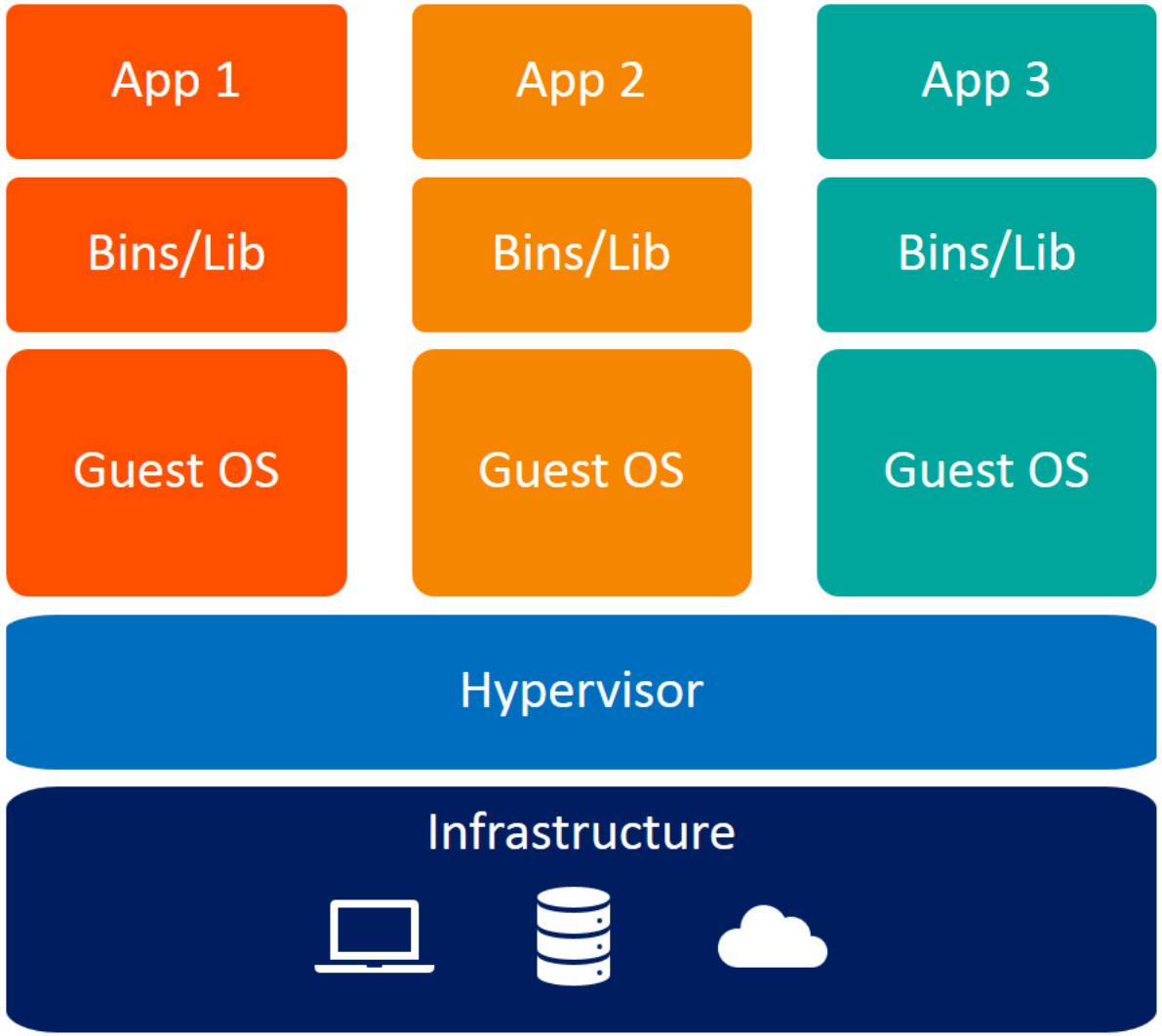
Conda cheat sheet

All conda commands

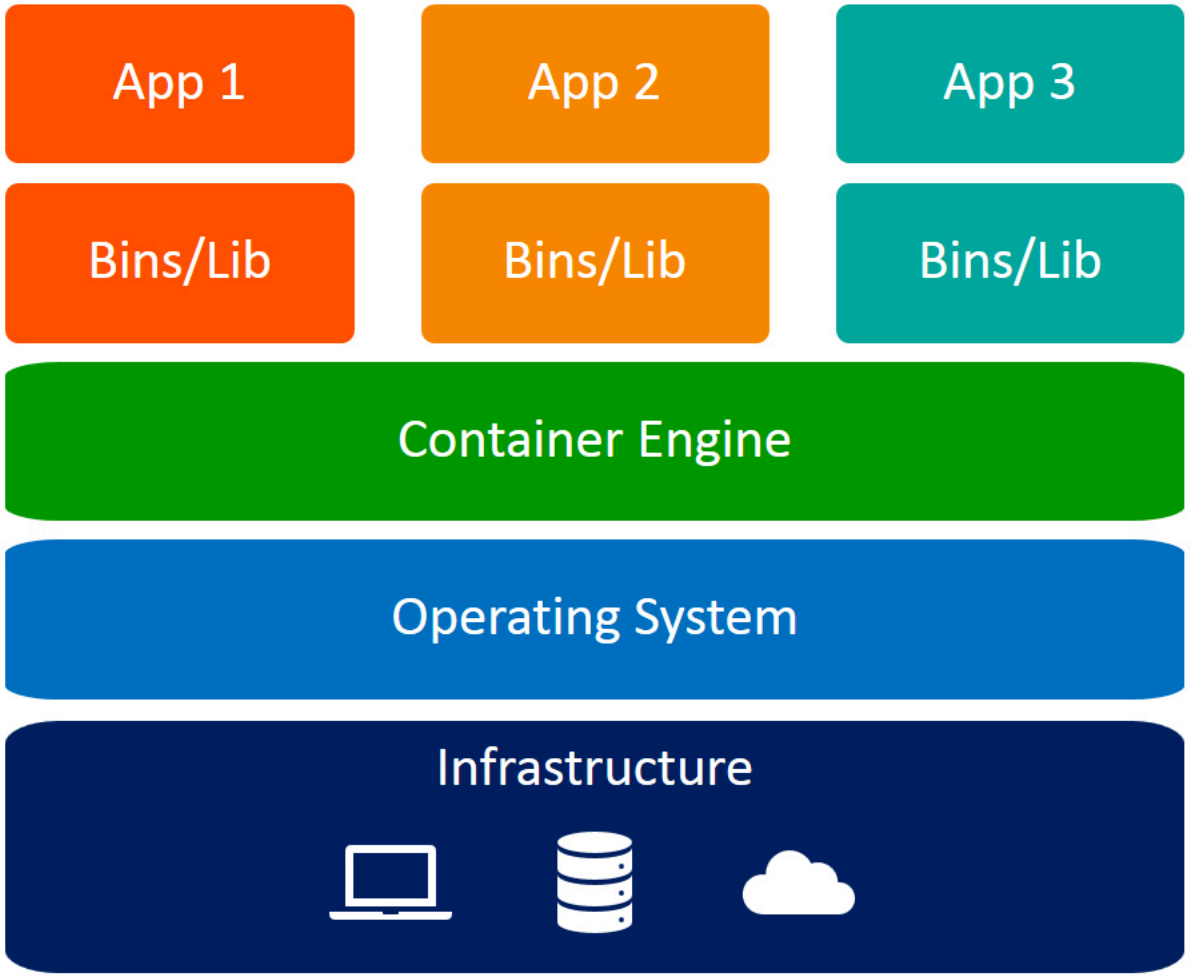

MAN, DOCKER IS BEING USED FOR EVERYTHING.
I DON'T KNOW HOW I FEEL ABOUT IT.
STORY TIME!
ONCE, LONG AGO, I WANTED TO USE AN OLD TABLET AS A WALL DISPLAY.
I HAD AN APP AND A CALENDAR WEBPAGE THAT I WANTED TO SHOW SIDE BY SIDE, BUT THE OS DIDN'T HAVE SPLIT-SCREEN SUPPORT.
SO I DECIDED TO BUILD MY OWN APP.
LIVECAM
I DOWNLOADED THE SDK AND THE IDE, REGISTERED AS A DEVELOPER, AND STARTED READING THE LANGUAGE'S DOCS.
...THEN I REALIZED IT WOULD BE WAY EASIER TO GET TWO SMALLER PHONES ON EBAY AND GLUE THEM TOGETHER.
LIVECAM
ON THAT DAY, I ACHIEVED SOFTWARE ENLIGHTENMENT.
BUT YOU NEVER LEARNED TO WRITE SOFTWARE.
NO, I JUST LEARNED HOW TO GLUE TOGETHER STUFF THAT I DON'T UNDERSTAND.
I... OK, FAIR.

Docker





Virtual Machines



Containers

Benefits of containerization

- Containers allow users to bring all dependencies (configs, packages and other dependencies) to a workflow
- Due to their isolated nature, they have several beneficial implications for security, reproducibility and maintainability – they also simplify deployments greatly
- They avoid invasive and time-consuming local installations

Terminology

Container: 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. What is a container?

Image: A container image is a standardized package that includes all of the files, binaries, libraries, and configurations to run a container. What is an image?

Registry: An image registry is a centralized location for storing and sharing your container images. It can be either public or private. Docker Hub is a public registry that anyone can use and is the default registry. What is a registry?

Let's try it out! ⚙️

```
docker run --rm -it ubuntu sh
```

Congratulations, you are now running a fully fledged ubuntu on your host machine!

Worfklow

`Dockerfile` -> `docker build . –t myimage` -> `docker run myimage`

Basic commands ⚙️

- `docker ps` : list all running containers
- `docker ps -a` : list all containers, including stopped ones
- `docker image ls` : list images
- `docker volume ls` : list volumes
- `docker pull <registry_name>/<image_name>:<tag>` : pull an image from a given registry
- `docker login` : login to Docker Hub
- `docker logs <container_name or container_id>` : get logs from a specific container
- `docker exec -it <container_name or container_id> <command>` : execute a command in a running container

Housekeeping ⚙

- Docker stores build steps in layers to optimize build times and efficiency.
- `docker container rm <container_name or container_id>`
- `docker image rm <image_name or image_id>`
- `docker volume rm <volume_name or volume_id>`
- `docker container prune`, `-af`
- `docker image prune`, `-af`
- `docker volume prune`, `-af`
- `docker buildx prune`, `-af`
- `docker system prune`, `-af`

Docker, Conda and Git - disclaimer: avoid conda here

```
FROM continuumio/miniconda3:23.10.0-1

RUN apt-get update && apt-get install -y \
    build-essential \
    python3-dev \
    python3-pip \
    python3-setuptools \
    python3-wheel \
    gcc \
    g++ \
    && rm -rf /var/lib/apt/lists/*

RUN conda install -y -q python=3.10
RUN conda install -y -q conda-libmamba-solver
RUN conda install -y -q conda-forge::openslide=4.0.0
RUN conda config --set solver classic

WORKDIR /app
COPY requirements.txt /app
RUN pip install -r requirements.txt

COPY . /app
CMD ["uvicorn", "slide_provider:slide_endpoint", "--host", "0.0.0.0", "--port", "3306", "--reload"]
```

```yaml
variables:
  TAG_ID: $CI_COMMIT_SHORT_SHA
stages:
  - build
  - deploy
  - clean build cache
  - debug

docker compose build:
  stage: build
  script:
    - docker compose build --no-cache

docker compose up:
  stage: deploy
  script: docker compose up -d --force-recreate --remove-orphans

clean docker:
  stage: clean build cache
  script:
    - docker system prune -af

docker compose logs:
  stage: debug
  rules:
    - when: manual
  script: docker compose logs
```

One step at a time

- **How do I get data into my container?**
  Application files should be copied into the container inside `Dockerfile` using the `COPY` keyword. For development purposes or access to files on the host system make use of `volumes`.

Example:

```
volumes:
    - "/directory/on/host/:/directory/in/container"
```

This definition can be extended using flags such as read-only (ro).

```
volumes:
    - "/directory/on/host/:/directory/in/container:ro"
```

How do I set environment variables inside the container?

- `docker run -e SOME_VAR='some_value'`
-
```
environment:
  SOME_VAR: 'some_value'
```

I do not want to rebuild my containers all the time when i make changes to my code

- Use a seperate docker compose file to mount files at runtime:

```
services:
  slide-provider:
    volumes:
      - "./digitalpathology:/Volumes/digitalpathology"
  storage-viewer:
    volumes:
      - "./digitalpathology:/Volumes/digitalpathology"
```

- Chain your compose files:

  `docker compose -f docker-compose.yml -f docker-compose-develop.yml up`

I cannot see my app in my browser

- Remember the port mapping for your service

```
nginx:
  build: ./Viewer
  restart: unless-stopped
  ports:
    - "8080:80"   # HOST:CONTAINER
  depends_on:
    - slide-provider
    - storage-viewer
  networks:
    - base_network
```

I do not want to copy certain files

- Similar to a `.gitignore`, you can create a `.dockerignore` that will ignore files specified therein

Reminders

- Keep an up-to-date list of requirements for your project!
  `conda env export -f environment.yml` and `pip freeze > requirements.txt` are your friends
- Not sure which version is the latest? https://pypi.org/
- Know which runtimes and tools you require (Python, R, ...)
- Set up your Docker container / development container when your project starts

Docker alternatives

- Podman (no daemon, child processes)
- Apptainer (root installation, non-root execution)
- Many more

Excursion: Devcontainers ⚙️

Open the Git repository in VSCode -> Reopen in container.

Cool Docker features ⚙️

- Have you heard of LLMs?

```
docker desktop enable model-runner
docker desktop enable model-runner --tcp 12434
docker model pull ai/smollm2:360M-Q4_K_M
docker model run ai/smollm2:360M-Q4_K_M "Give me a fact about orcas."
```

Search settings

- General
- Resources
- Docker Engine
- Builders
- AI
- Kubernetes
- Software updates
- Extensions
- Beta features
- Notifications
- Advanced

AI

Settings for beta AI features can be found here.

Docker Model Runner

Enable Docker Model Runner Give feedback

Enable GPU-accelerated inference engines on /var/run/docker.sock and model-runner.docker.internal:8
Note: Inference engine support may take a few minutes to initialize.

Enable host-side TCP support

Port

12434

default: 12434

CORS Allowed Origins

Custom

Add origin +

Usage in scripts ⚙️

```
from langchain_openai import ChatOpenAI

llm = ChatOpenAI(
    base_url="http://localhost:12434/engines/v1",
    model="ai/smollm2:360M-Q4_K_M",
    api_key="none"
)

response = llm.invoke("Give me a fact about orcas.")

print(response.content)
```

`curl` alternative

```
curl http://localhost:12434/engines/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer none" \
  -d '{
    "model": "ai/smollm2:360M-Q4_K_M",
    "messages": [
      {"role": "user", "content": "Give me a fact about orcas."}
    ]
  }'
```

If it runs locally you can ...

- use it on sensitive information
- **use it together with your local runner**
- **use it in local scripts**
- **use it in other workflows** such as n8n

Questions?

Docker cheat sheet

No amount of theory can replace hands-on knowledge...

Hands-on

Prerequisites ⚙️

- Streamlit: Streamlit is the greatest crutch to exist for developers that **really** want to have a nice frontend, but **really don't** want to learn javascript.
- Streamlit docs: https://docs.streamlit.io
- Molecule icon generator
- MedShapeNet
- NYC Uber ridesharing

Exercise ⚙

0. (Optional) Form groups if you wish
1. Find a dataset (recommendations on the next slide)
2. Set up a Git repository for your project
3. Analyse the data, build a small visualization app in a docker container and push it to Docker Hub
4. Present one quick fact from your data that you find interesting and worth sharing

Dataset inspiration ⚙

- https://github.com/awesomedata/awesome-public-datasets
- Stanford open policing
- https://github.com/evangambit/JsonOfCounties
- https://www.dolthub.com/discover
- https://www.springboard.com/blog/data-science/15-fun-datasets-to-analyze/
- https://www.kaggle.com/datasets

Other exercises ⚙️

- n8n workflow orchestration with locally deployed LLM
- Docker compose: Extend the streamlit app by writing and retrieving information from a locally deployed database container (e.g. Postgres)

Practical session ⚙

Thank you for attending!