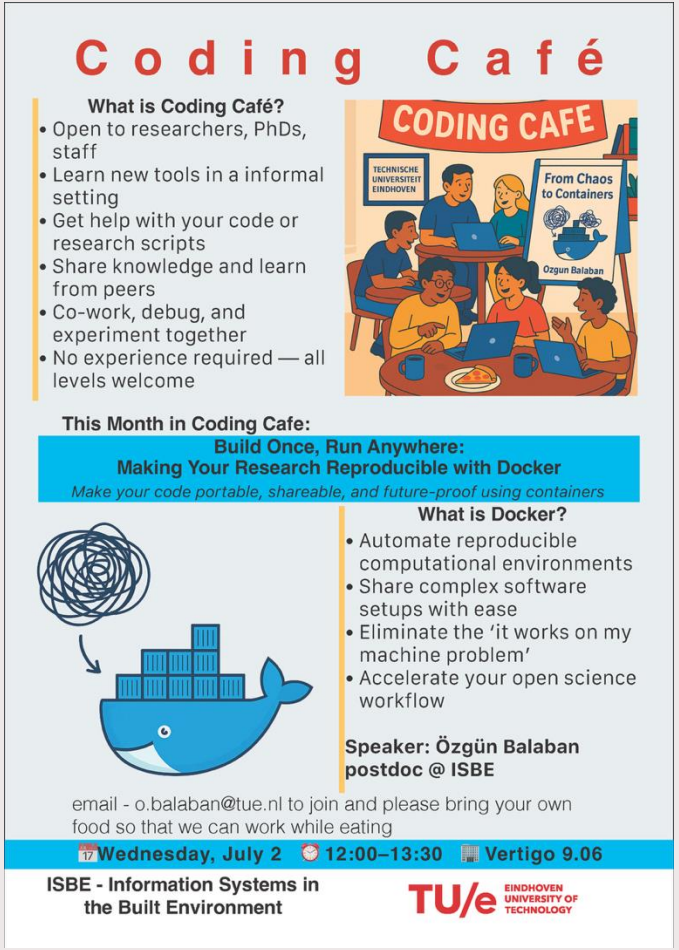
TU/e
ISBE
CODING CAFÉ

Coding Café – 2 July 2025

Speaker: Özgün Balaban



What is a Coding Café?

Originating in places like Netherlands eScience Center, Programming Cafés are low-threshold, peer-driven sessions where researchers:

- 🧑💻 Present a short demo of a programming method or tool
- 🛠️ Ask or offer help with code or workflows
- Create networking between people working on same issues / software
- Co-work informally, with coffee or pizza
- Focus on community over curriculum

What is reproducibility and why is it important?

Intellectual Property

Horror Stories

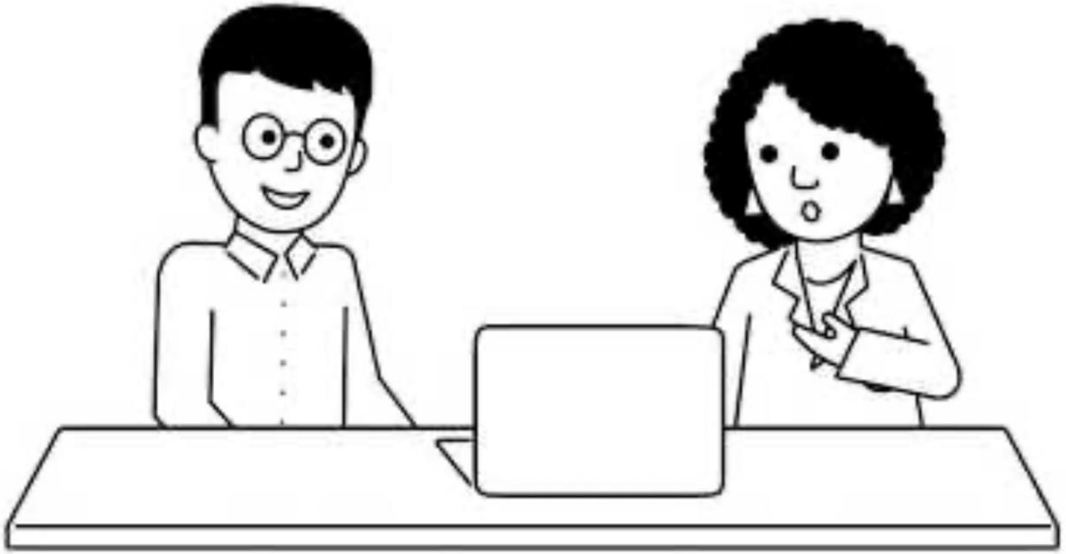
Software Dependency

- ❌ Software not available for your OS (Mac, Windows, Linux)
- 🔄 Endless install loops: each dependency needs another
- 🎲 Only some versions work together correctly
- 🤷‍♀️ You're unsure what version you're even using
- 🤝 Same software, different results on different systems
- 📦 Local install works, but fails elsewhere (colleague’s machine, server, etc.)
- 📉 Others can't install your tool/package
- 🧪 You can't reproduce an old project because its environment is gone

- Dependecy Hell

ref: https://carpentries-incubator.github.io/docker-introduction

How can software containers help your research?



https://www.youtube.com/watch?v=HelrQnm3v4g&t=18s

⚠️ When Installation Fails, Research Suffers

🧪 **Common Problems**

- ❌ You can’t use a tool because it's not available or installable
- 🤷‍♀️ You can’t reproduce results — you're unsure what tools you actually used
- 🔒 You can’t access newer resources — software setup isn't portable
- 🚫 Others can’t validate or build on your work — **it only runs on your machine**

🚀 The Good News: **Containers to the Rescue**

What is a Container?

Motivation

You want to install research software **without breaking your system** or needing a new computer.

Imagine:

You could spin up a **separate filesystem & OS** *inside* your current computer

It stays isolated from your main system

It includes **everything** the software needs to run

What is a Container?

Definition (Docker)

"A container is a standard unit of software that packages up code and all its dependencies so the application runs reliably from one computing environment to another."
— Docker

🚢 Analogy: Like a Shipping Container

What is a Container?

Before containers: cargo = **mixed, fragile, chaotic**
With containers: **everything is grouped, portable, predictable**
Software containers let you “ship” a complete working setup to **any computer** with Docker installed — and **it just works**

Key Benefits

- Reproducibility
- Portability
- Isolation
- Consistency

Docker in a Nutshell

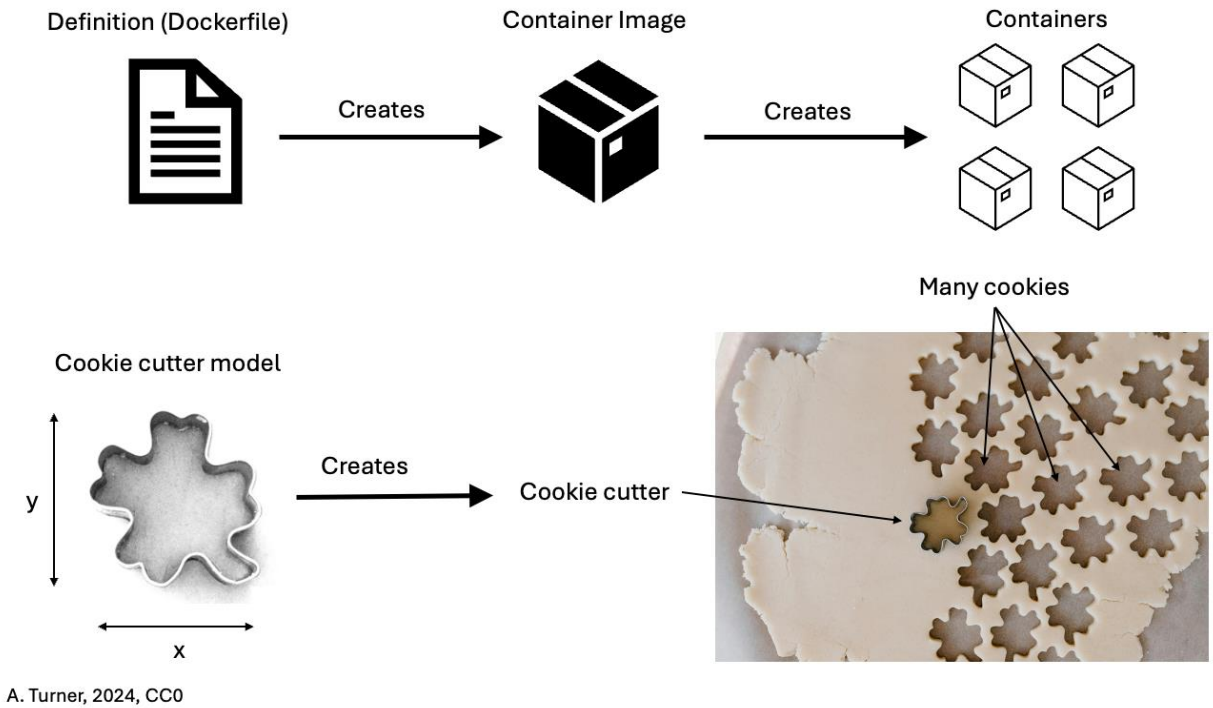
Docker is a tool to **build and run containers**.

- It's not the only container tool, but it's the most widely used.
- We use Docker in this workshop to create **reproducible, portable environments**.

What is a Container Image?

- A **container** is a running environment — like a temporary mini-system
- A **container image** is the **recipe or template** that defines:
  - What software is inside
  - What dependencies are needed
  - How to run it

Analogy: Cookie Cutter



A. Turner, 2024, CC0

Analogy: Cookie Cutter?

- **Image** = the **cookie cutter** (blueprint)
- **Container** = the **cookie** (a running instance)
- Use the same cutter (image) to make multiple identical cookies (containers)
- Containers come and go; the image stays the same

Key Concepts

- **Docker** runs containers from images
- **Images** define the environment
- **Containers** are active, disposable environments

Wrap up

🧪 **The Problem**

- Scientific software is hard to install and reproduce because of:
- Complex dependency chains
- OS differences
- Lack of documentation
- Unreliable re-installs

Wrap up

The Solution: Containers

- A container is a self-contained, portable filesystem.
  Here's what that gives you:
- Key Benefits of Using Containers in Research
- Documentation
  → A clear, repeatable record of tools and dependencies
- Portability
  → Runs on any system with Docker (Mac, Windows, Linux)
- Reproducibility
  → Same software, same environment — everywhere
- Configurability
  → Allocate CPU/memory as needed for clusters or small machines

Wrap up

🚀 **What's Next?**

In this workshop, you'll learn how to:

- Run containers from existing images
- Create and share your own container images

Docker

https://labs.play-with-docker.com – online playground

Local installation

Docker commands -- version

```
$ docker --version
```

Docker commands -- list

```
$ docker container ls
```

Docker commands -- help

$ docker –help

$ docker container --help

Downloading Docker images

```
$ docker image ls   # list all the images

$ docker image pull hello-world
```

Dockerhub

Running the hello-world container

```
$ docker container run hello-world

$ docker container run alpine

$ docker container run alpine cat /etc/os-release

$ docker container run alpine echo 'Hello World'
```

Running containers interactively

```
$ docker container run -it alpine sh
```

Try some commands:

```
ls
whoami
```

Running containers - ubuntu

```
$ docker container run ubuntu apt-get --help
```

Removing Images

```
$ docker image ls

$ docker image rm <imageid>

$ docker image rm hello-world
```

What containers are running?

$ docker container ls

$ docker container ls –all

$ docker container prune !!! Be careful removes all stopped containers

What is a Dockerfile?

Text file with instructions to build an image

```
FROM python:3.9
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

What is a Dockerfile?

```
FROM python:3.12-bullseye
ENV PYTHONUNBUFFERED=1
RUN mkdir /code
WORKDIR /code
RUN pip install poetry
COPY pyproject.toml poetry.lock ./
RUN poetry install --no-root
COPY . .
EXPOSE 8000
ENTRYPOINT ["poetry", "run", "python", "manage.py", "runserver", "0.0.0.0:8000"]
```

Simple Dockerfile example

```
# Use official Python image as base
FROM python:3.10-slim
# Set working directory in container
WORKDIR /app
# Copy script into container
COPY app.py .
# Set the default command to run
CMD ["python", "app.py"]
```

```
docker build -t hello-docker .
docker run hello-docker
```

Check this example from github repo!

Using Volumes

```
docker run -v $(pwd):/app my-python-app
```

Host directory is mounted into container
Useful for saving outputs or code reuse

Using Volumes - example

```
FROM python:3.10-slim
WORKDIR /app
COPY writer.py .
CMD ["python", "writer.py”

docker build -t volume-demo .
docker run -v $(pwd)/data:/data volume-demo
```

Check this example from github repo!

Networking example

Docker Networks

By default, containers are isolated

Use docker network create to define networks

```
docker run -p 8888:8888 -e JUPYTER_TOKEN='mypassword' jupyter/base-notebook
```

This creates a jupyter environment that is accessible from the browser

http://localhost:8888/?token=mypassword

For more information - documentation

- Docker Overview
- Dockerfile Reference
- Volumes in Docker

Thank you

Contact:

- Özgün Balaban
  Postdoctoral Researcher at ISBE
  o.balaban@tue.nl