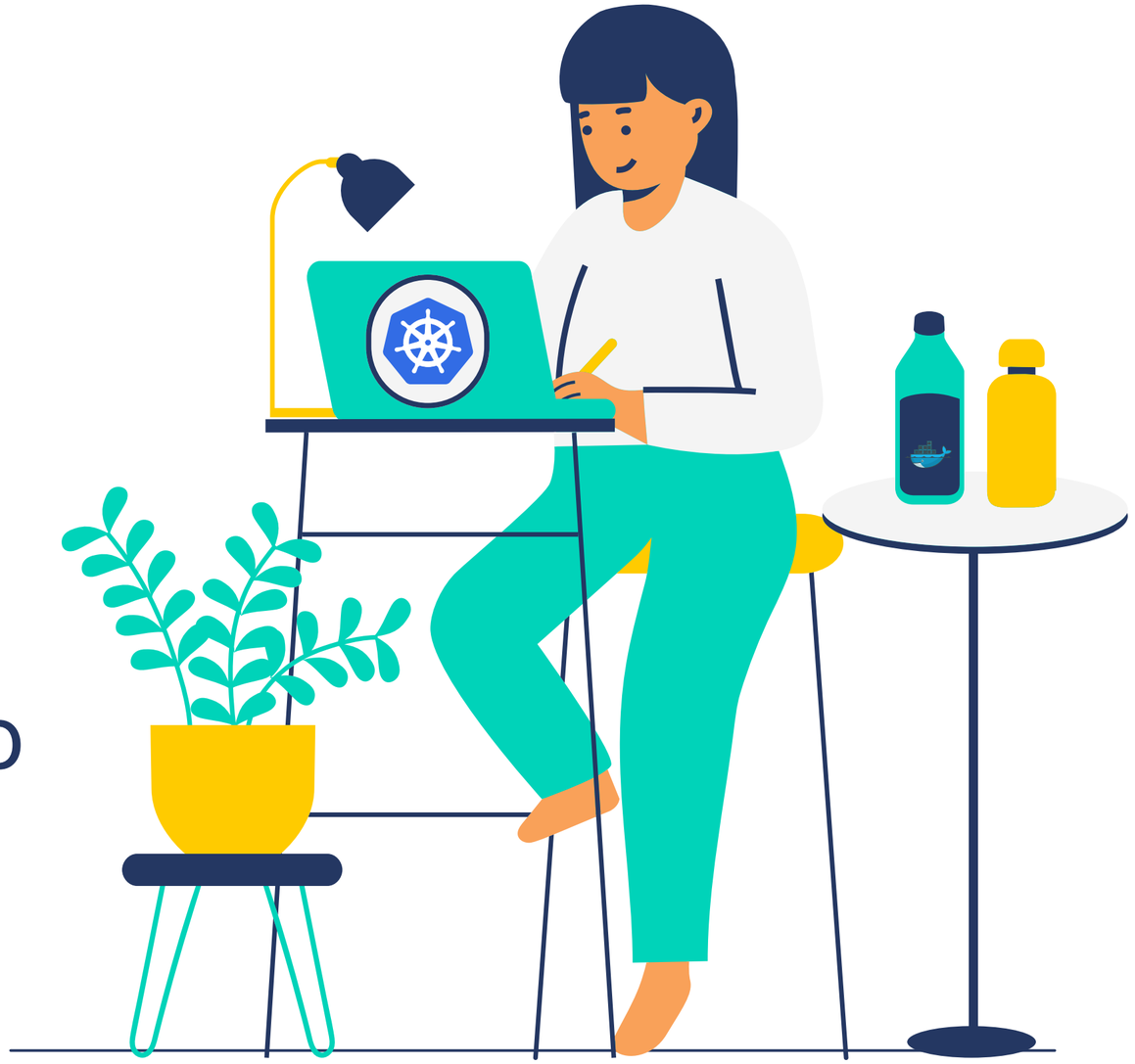
Containers & Kubernetes

A Beginner's Workshop

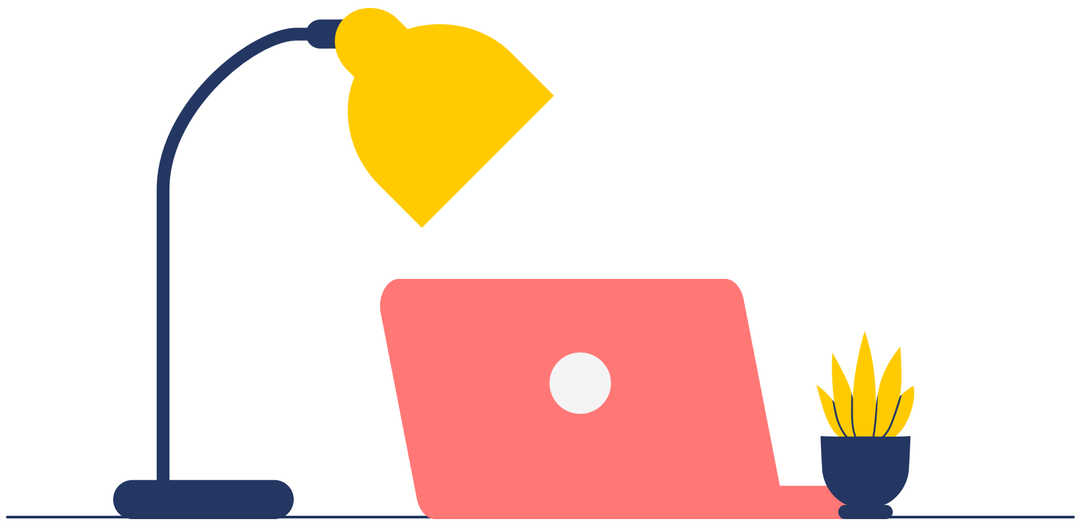
Rigerta Demiri & Katharina Sick

#theNewITGirls | November 17, 2023



About today

Let’s do a step-by-step journey into the world of containers and Kubernetes.



Containers & Docker

What are containers? Why are they cool? How to create & use them?

Kubernetes

What is Kubernetes? Why do so many companies need it? How to get started?

Troubleshooting and Questions

Feel free to ask any time! :)

So, what is Docker?

An open-source tool used to turn your **application** into a **container** that can easily be deployed in any other system.

First things first, what is an Application?

Any piece of **Software** that performs a specific function either for an end user or for another application.

An **Application** is written in a specific **Programming language** and typically has one or more **Dependencies.**

Cool, so what is a Container?

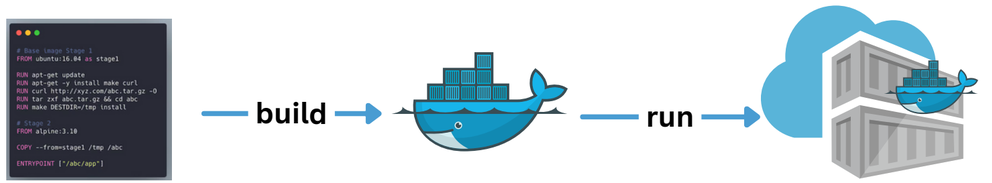
A **self-contained**, **runnable** software application or service.

And how do we do this with Docker?

We write a **Dockerfile**, which is “a recipe” defining how to build a **Docker Image** and when we run the image so we can reach the application, we are in a **Container**

Key Concepts

- **Application**
  - Any piece of software you write, to fulfil a goal for end users or other applications
- **Dependencies**
  - All software libraries an application needs to be able to run successfully
- **Dockerfile**
  - A text file containing a set of instructions
- **Docker image**
  - A read-only blueprint that includes container-creation instructions
  - An **executable** application artifact



- **Docker container**
  - A running instance of a Docker **image** that gets created when the **$ docker run** command is implemented
  - Multiple containers can run from the same Docker image
- **Docker registry**
  - A storage and distribution system for container images (can be **private** or **public**)
  - Docker Hub (hub.docker.com) is the **official public** Docker registry

OK, let's take a step back

Why do we even **need** Docker?
How were applications deployed before it?

Traditional Application Deployment

Developer

Pushes code for 3 applications into their own Git repositories

Web Application 1

Web Application 2

Web Application 3

Application Deployment Process

Artifact

Deployment instructions

Production Server

Install & config all necessary libraries / databases / services for **Appplication 1 directly on the server**

Same for **Application 2**

Same for **Application 3**

User

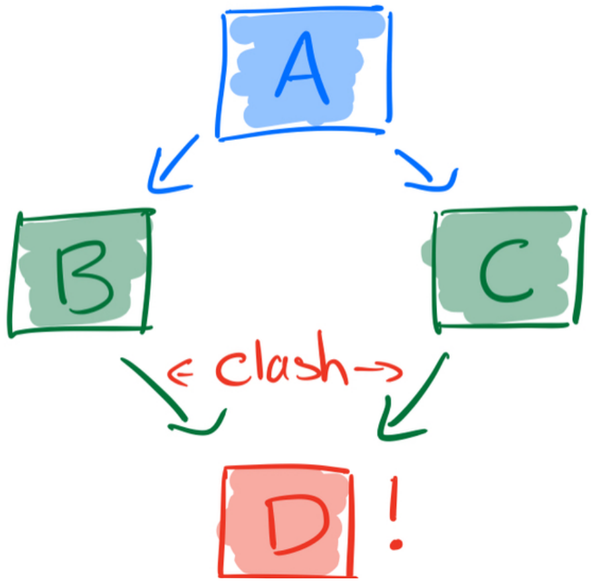
Consumes the web application via a web browser

What is the problem with that?

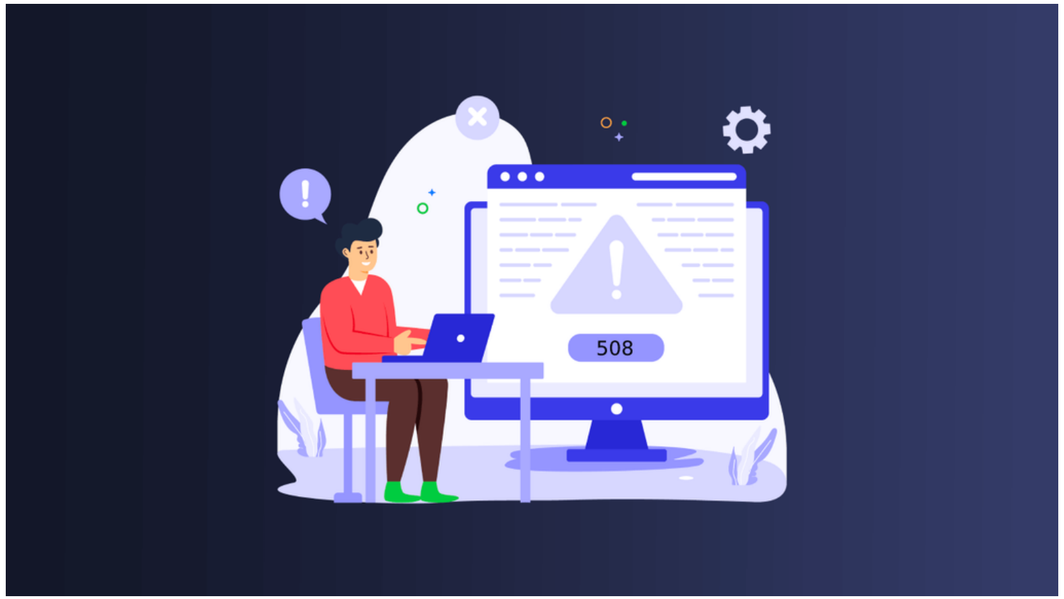
The same production server is usually used for **multiple** applications, for efficiency, **but**:

- Error prone process
- Different applications might need different
  - **software & software versions**
  - **libraries & library versions**

- This causes:
  - dependency issues
  - a painful **deployment process**



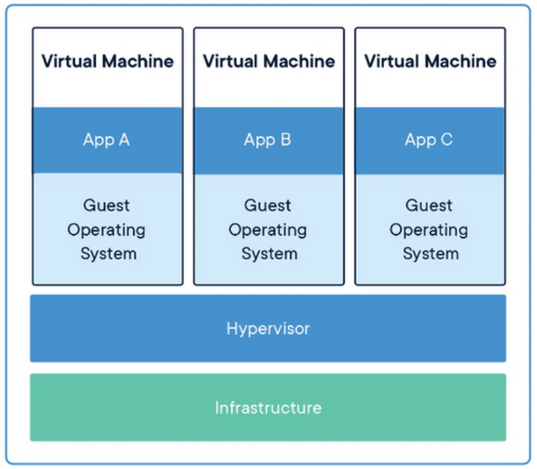
What is the problem with that?

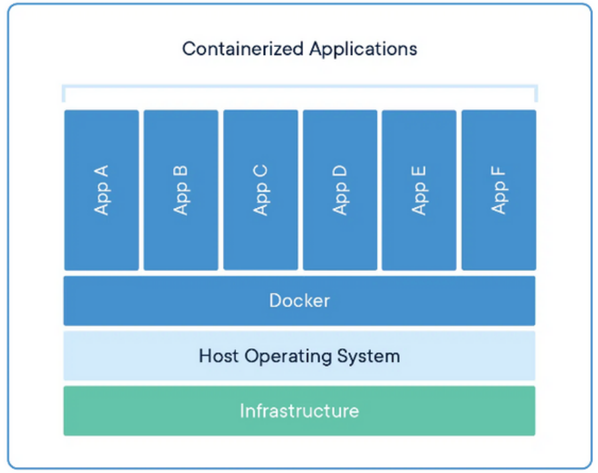


Memory and **processing power** of the production server are **limited**

This is where Kubernetes & Docker play very well together and you will hear all about it in Part 2 of this workshop by Katharina

Enter Docker





VM Virtualization

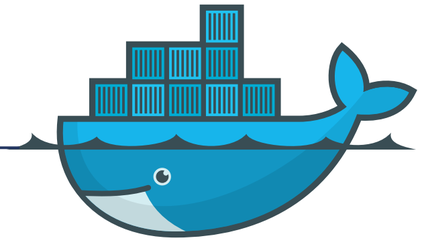
OS Kernel & the application layer of the OS

Docker Virtualization

Only the application layer of the OS

Benefits

- **Isolation**
  - No conflicting with the host system
- **Packaged application and all its dependencies** (code, libraries, system binaries etc)
  - No more “it works on my machine” issues
- **Reliable deployment**
  - A **self-contained** image is used to deploy, making the deployment independent of the OS or other details of the host system
  - Docker runtime is **the only** necessary installation

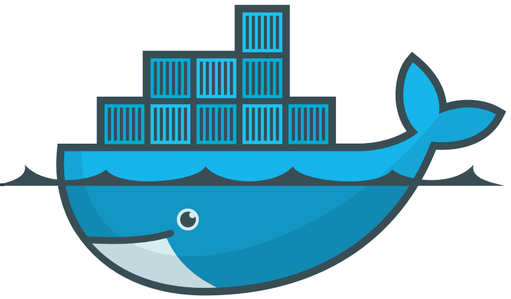


Enter Docker

More benefits

- **Better resource consumption management**
  - Configurable memory & cpu a container can use (docs on resource constraints)

- **Efficiency**
  - A side effect of the lightweight, efficient isolation model of containers
  - **Many** Docker containers can run on a single production server



Demo Time

- Step by step Docker introduction
- Docker Registry
- Use existing Docker Images
- Create your own Docker Image

First steps

- **Docker running locally**
  - Instructions to download and install Docker
  - I run/prefer **Rancher Desktop**

- **An IDE or text editor to use for editing files.**
  - I prefer VSCode - Download link

- **Free Docker Account**
  - You can sign-up for a free Docker account (https://hub.docker.com) and access free unlimited public repositories
  - **Not mandatory**, images are available to unauthenticated users too

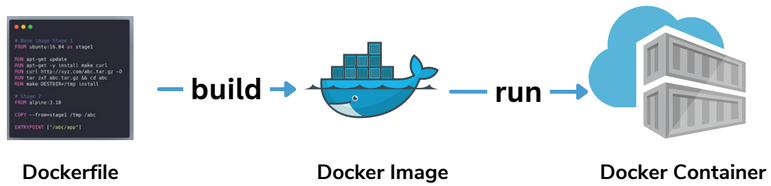
Demo Part 1

- **Where to find Docker Images?**
  - Docker Registries
    - Public (Docker Hub)
    - Private (AWS ECR, Google Container Registry, etc)
- **How to create a Container?**
- **How to access a containerised application**
  - Port Binding
- **How to stop Containers**
- **How to remove Docker Images and Containers**

Demo Part 2

Creating your own Docker Image

- **Creating a Streamlit Application**
- **Running it locally with** streamlit run
- **Building a Docker Image for this Application**



Main Takeaways

- Docker is an open source virtualization technology
- With Docker you can package your application into a single runnable artifact - Docker Image
- Docker Images will run everywhere, independent of the platform you deploy it to

Where to go from here?

Explore the official documentation

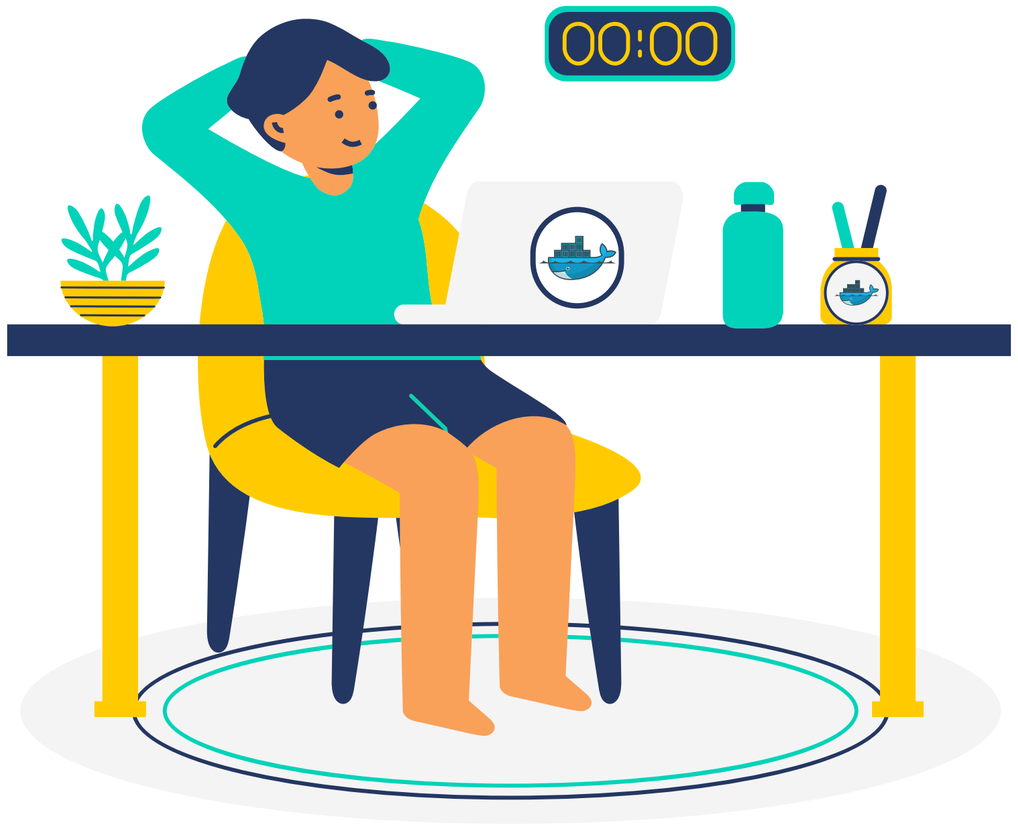
- https://docs.docker.com/get-started/
- https://www.docker.com/resources/what-container/
- https://docker-curriculum.com/#docker-compose
- https://rancherdesktop.io/
- https://docs.streamlit.io/

Watch YouTube videos

- Techworld with Nana

Experiment with side projects

- Build a simple application in any language of your choice and containerise it
- Play around and experiment with Docker
- Have fun!



Thank you!

Happy to take any questions.