

Modern Application Development and Deployment

Francisco Borges Aurindo Barros

05/11/2024

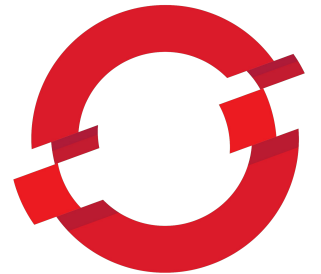
Before we start, a quick presentation.

I'm Francisco Borges Aurindo Barros

**Before we start, a quick
presentation.**

I'm Francisco Borges Aurindo Barros

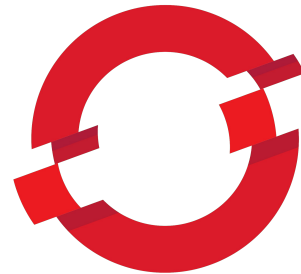
Before we start, a quick presentation.



OPENSIFT

I'm Francisco Borges Aurindo Barros

Before we start, a quick presentation.



OPENSIFT



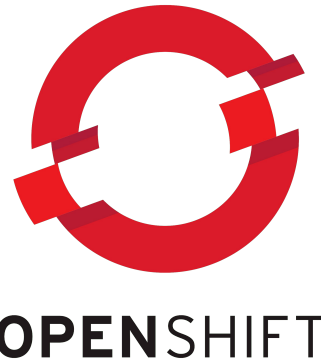
I'm Francisco Borges Aurindo Barros

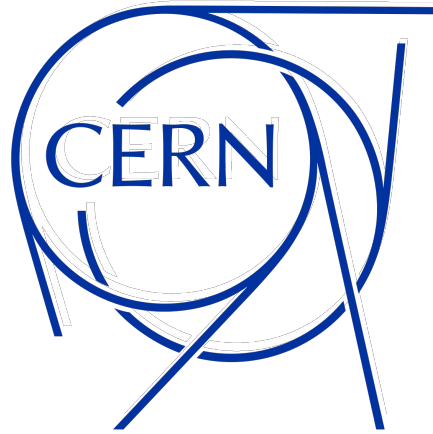
Before we start, a quick presentation.



I'm Francisco Borges Aurindo Barros

Before we start, a quick presentation.

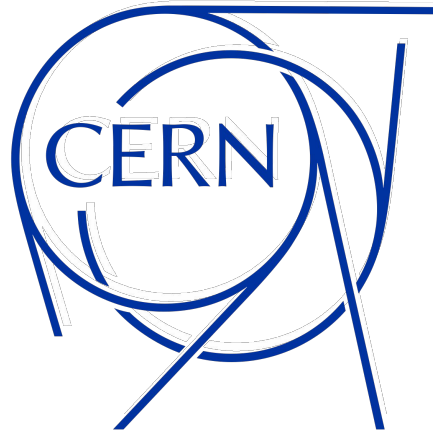




Modern Application Development and Deployment

Francisco Borges Aurindo Barros

05/11/2024



Modern Application Development and Deployment

Francisco Borges Aurindo Barros

05/11/2024

Tools required for today:

- ❏ Git
- ❏ Podman
- ❏ Terminal access
- ❏ Docker
- ❏ Optional
 - ❏ Dive: <https://github.com/wagoodman/dive>

Side Note:

Let's make sure the tools are installed now before we reach the exercises!

Tools required for today:

You can also just use LXPlus*

- ❑ Git
- ❑ Podman
- ❑ Terminal access
- ❑ Docker
- ❑ Optional
 - ❑ Dive: <https://github.com/wagoodman/dive>

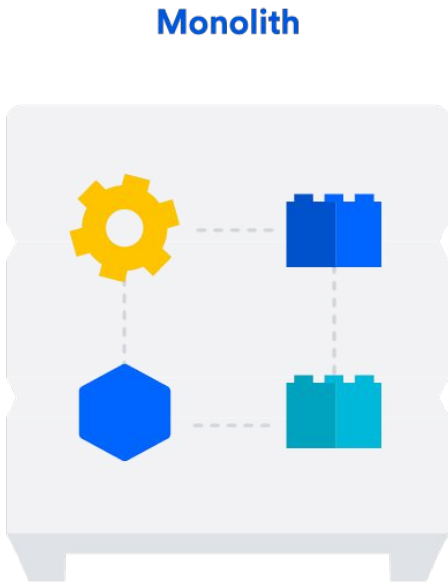
Side Note:

Let's make sure the tools are installed now before we reach the exercises!

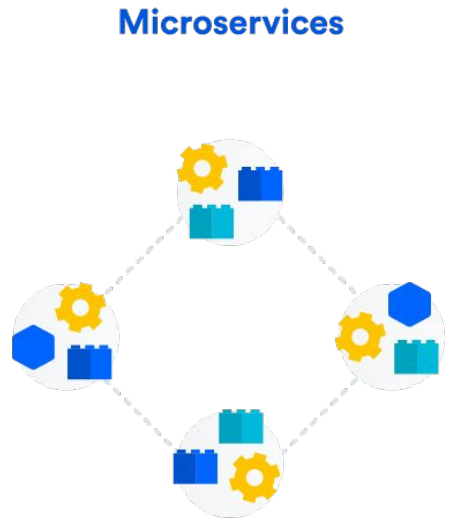
Before that...what deployment models are out there?

Monolithic vs Microservices

Aspect	Monolithic	Microservices
Development	Simpler initially	Complex, but flexible
Scalability	Challenging	Easier, more granular
Deployment	Entire app at once	Independent services
Technology	Limited choices	Diverse stack possible
Maintenance	Simpler for small apps	More complex, but modular

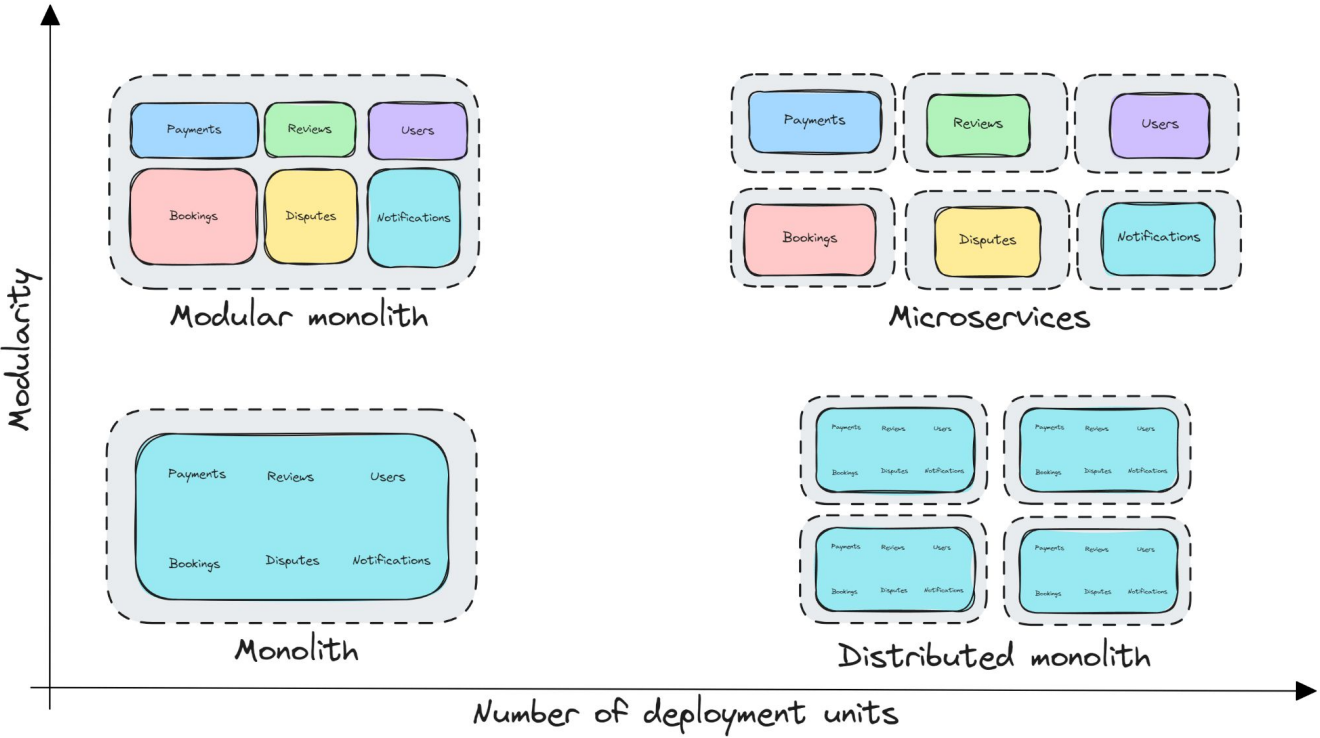


VS



Monolithic vs Microservices, what about in between?

Aspect	Architecture	Deployment
Monolithic	Single, tightly-coupled unit	Single unit deployment
Microservices	Multiple independent services	Independent service deployment
Modular Monolith	Single deployable unit with loosely-coupled modules	Singular unit deployment, with potential for module extraction
Distributed Monolith	Multiple deployable units, tightly-coupled	Multiple unit deployment



What about types of Applications?

Web Servers

Key Functions:

- Content Delivery
- Protocol Handling (HTTP/S)
- Request Management
- Dynamic Content Generation
- Security

Usefulness

- Website Hosting
- Web Application execution
- File Sharing

Universality

- Global Reach
- Cross-platform
(Windows/Linux/macOS)
- Scalability

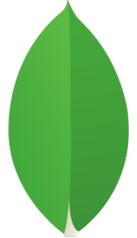


Caddy[®]

NGINX

Database Management Systems (DBMS)

DBMS Type	Description	Examples
Relational (SQL)	Stores data in structured tables with predefined schemas	MySQL, PostgreSQL, Oracle, Microsoft SQL Server
Document-oriented	Stores semi-structured data in documents (usually JSON)	MongoDB, CouchDB, Amazon DocumentDB
Key-Value	Stores data as simple key-value pairs	Redis, Amazon DynamoDB, Riak
Time Series	Optimized for time-stamped or time series data	InfluxDB, TimescaleDB
In-Memory	Stores data primarily in RAM for faster processing	Redis, Memcached



Frameworks and Microservices Frameworks

General frameworks provide a foundation for building entire applications, offering:

- Standardized structure and organization
- Reusable components and libraries
- Common design patterns and best practices

Microservices frameworks are specialized tools for developing distributed systems composed of small, independent services.

- Service discovery and communication mechanisms
- Containerization and orchestration support
- Distributed system management tools



What a classic common application setup looks like?



And many more...

We have WebApps, message-brokers, CI and CD tools, static site generators, etc.



Jenkins



And many more...

We have WebApps, message-brokers, CI and CD tools, static site generators, etc.



Jenkins

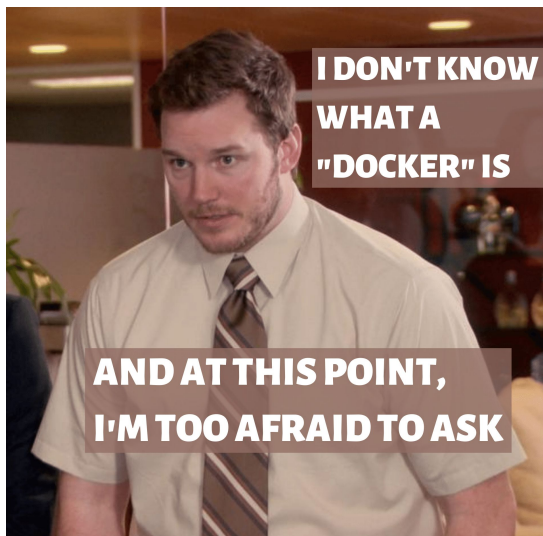
All applications mentioned above can be leveraged by Docker



Docker...



What is docker?





Docker...

~\(\ツ)/~

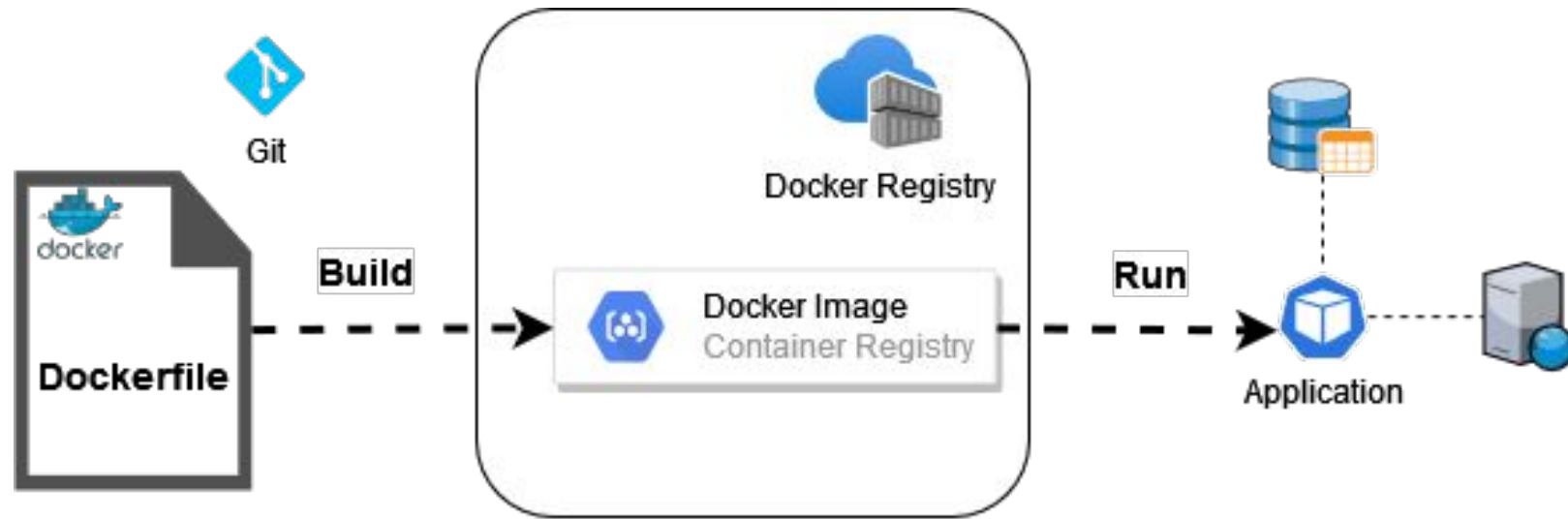
IT WORKS
on my machine

What is docker?

It's a platform to simplify the development, deployment and management of application.

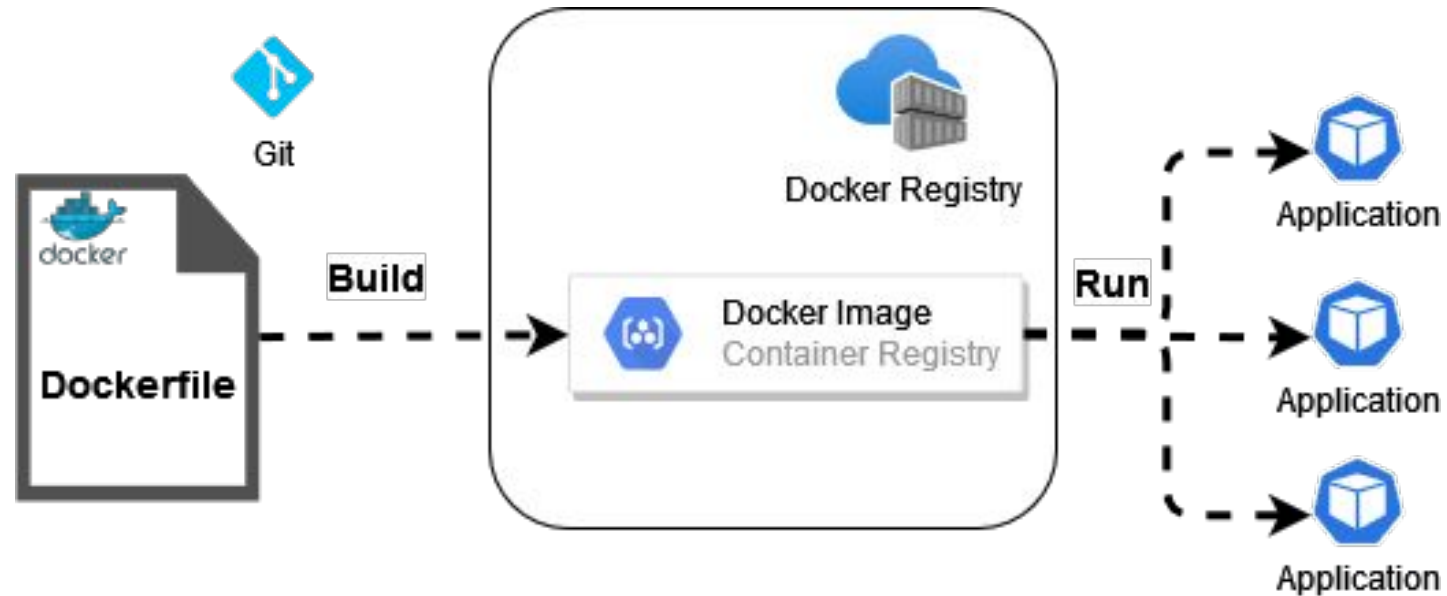
Docker is known for the use of containerization, which allows applications to run in an isolated environment (container).

Key Concepts



- Dockerfile (Instructions to build the Image)
- Images (Read-only templates with application code, libraries and dependencies)
- Containers (Runnable instances of images)
- Docker Engine (Runtime that manages containers)
- Docker Hub (Repository for sharing images)

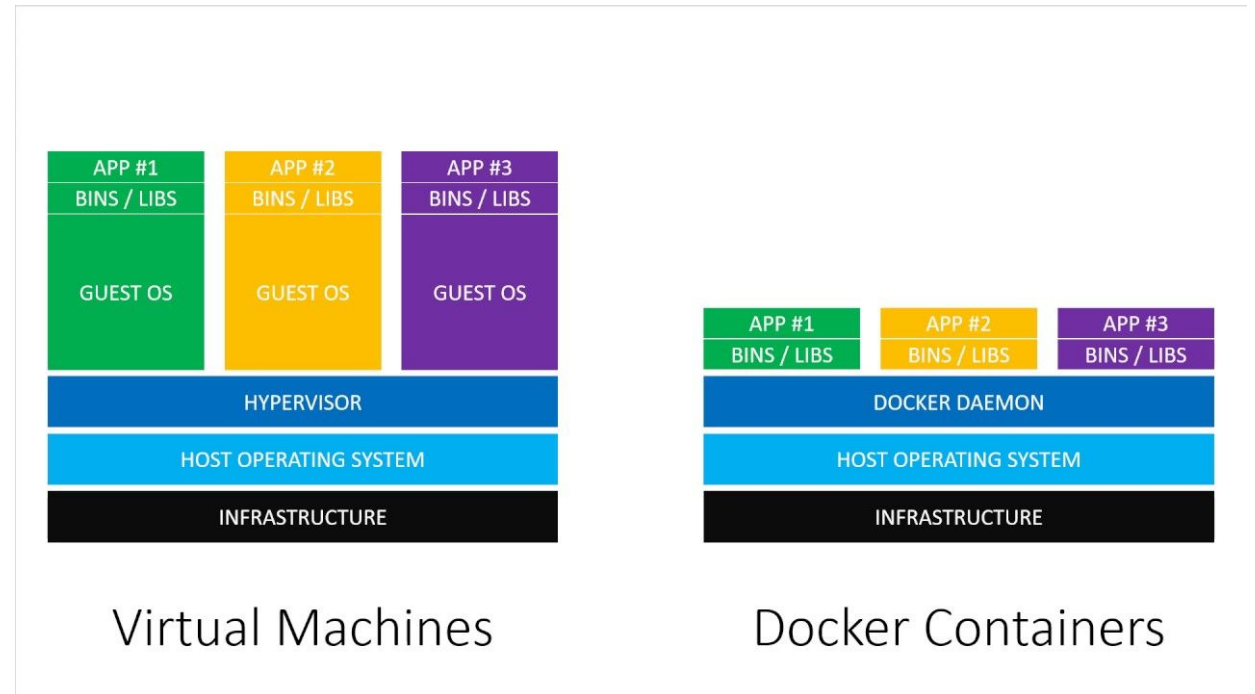
Key Concepts



- Dockerfile (Instructions to build the Image)
- Images (Read-only templates with application code, libraries and dependencies)
- Containers (Runnable instances of images)
- Docker Engine (Runtime that manages containers)
- Docker Hub (Repository for sharing images)

Benefits

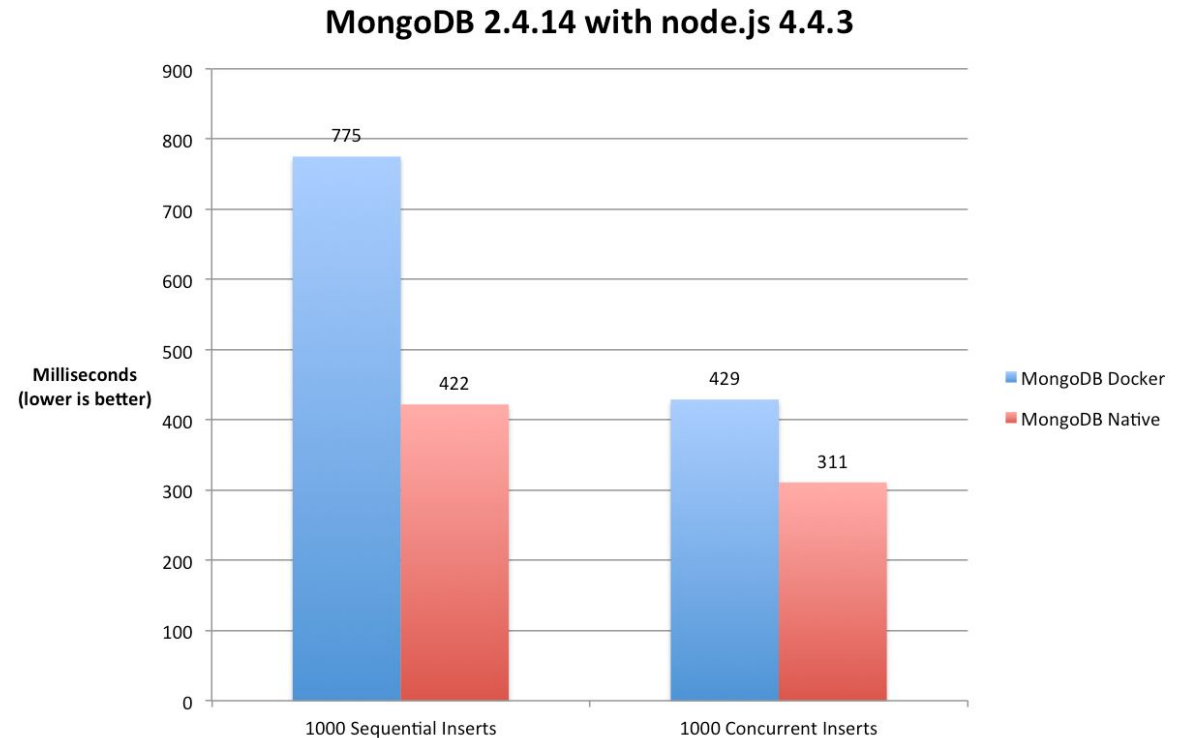
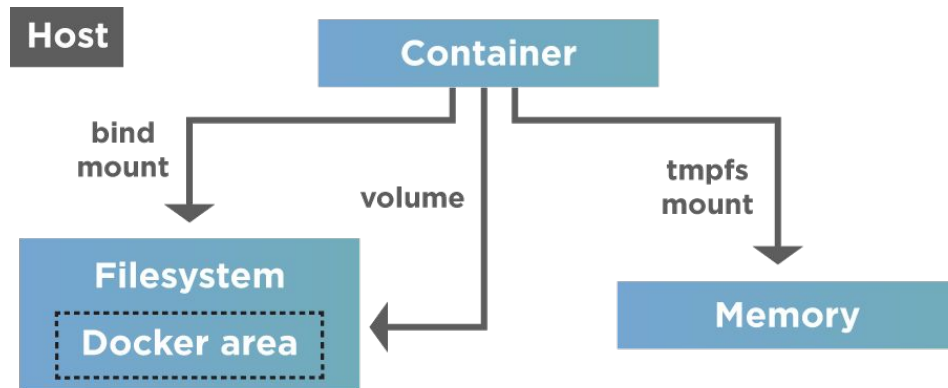
- Portability
- Isolation
- Scalability
- Efficiency
- Testing and CI/CD integration
- Easy collaboration



Cons

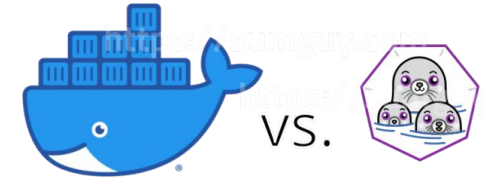
<https://vsupalov.com/docker-latest-tag/>

- Network complexity
- Security concerns (Image vulnerabilities)
- Performance
- Data persistence*



Source: <https://developers.redhat.com/blog/2017/06/14/local-development-setup-for-red-hat-mobile-using-docker>

Docker vs Podman vs ...



Feature	Docker	Podman
Architecture	Daemon-based	Daemonless
Root privileges	Required for daemon	Rootless by default
Image compatibility	OCI Compliant	OCI Compliant
Container runtime	OCI Compliant	OCI Compliant
CLI Compatibility	Standard Docker CLI	Docker-compatible CLI
Kubernetes Integration	Limited native support	Can generate K8s YAMLS

[OCI: Open Container Initiative](#)

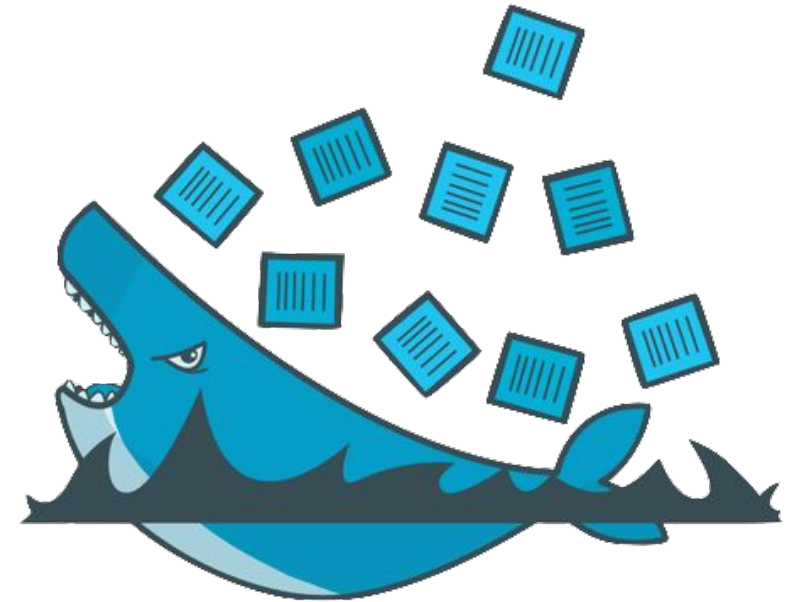
Now let's get into it...

- How can we leverage Docker?

It all starts with Dockerfiles.

Docker publishes best practices:

<https://docs.docker.com/build/building/best-practices/>



Now let's get into it...

- How can we leverage Docker?

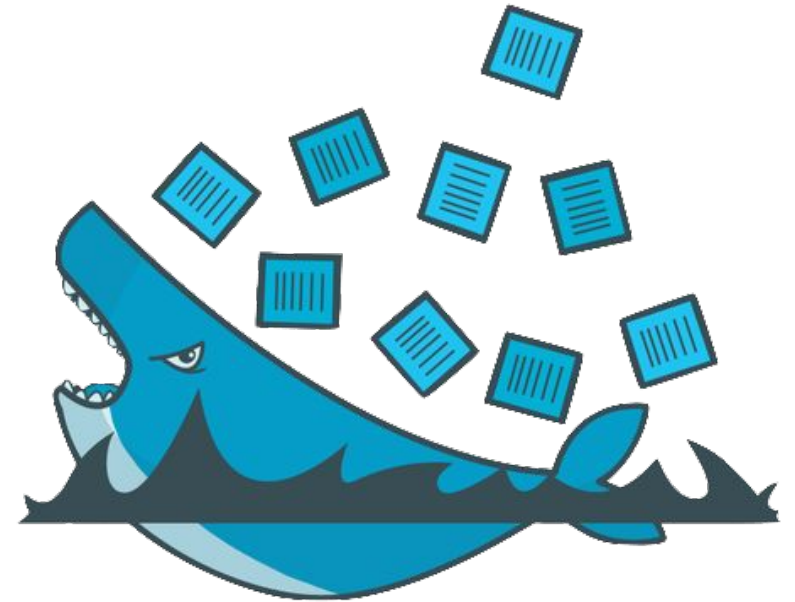
It all starts with Dockerfiles.

Docker publishes best practices:

<https://docs.docker.com/build/building/best-practices/>

Some classic tips when writing Dockerfiles:

- Avoid using *latest* tag
- Use multi-stage
- Docker Ignore should be leveraged
- Run as non-root



Modern Application Development and Deployment

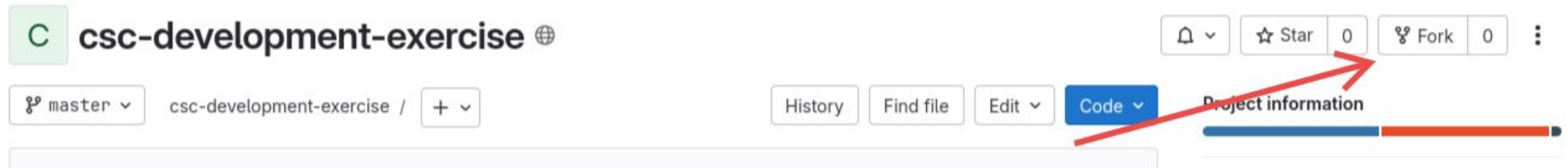
Practical exercise

Let's get the repo !




link: <https://gitlab.cern.ch/fborgesa/csc-development-exercise>



- fork it
- clone it








Let's look into the repo









Name
realworld
 .gitlab-ci.yml
 Dockerfile
 README.md

What do we have?



- Realworld folder (Application)
- .gitlab-ci.yml (what is this?)
- Dockerfile (We know what this is)
- README (standard descriptive file)

Let's look into the repo

Name
 realworld 
 .gitlab-ci.yml
 Dockerfile
 README.md

Name	Last commit	Last update
..		
 realworld	Exercise contents	34 minutes ago
 spec	Exercise contents	34 minutes ago
 templates	Exercise contents	34 minutes ago
 .gitignore	Exercise contents	34 minutes ago
 LICENSE	Exercise contents	34 minutes ago
 README.md	Exercise contents	34 minutes ago
 manage.py	Exercise contents	34 minutes ago
 requirements.txt	Exercise contents	34 minutes ago

Let's look into the repo

Name
realworld
 .gitlab-ci.yml
 Dockerfile
 README.md



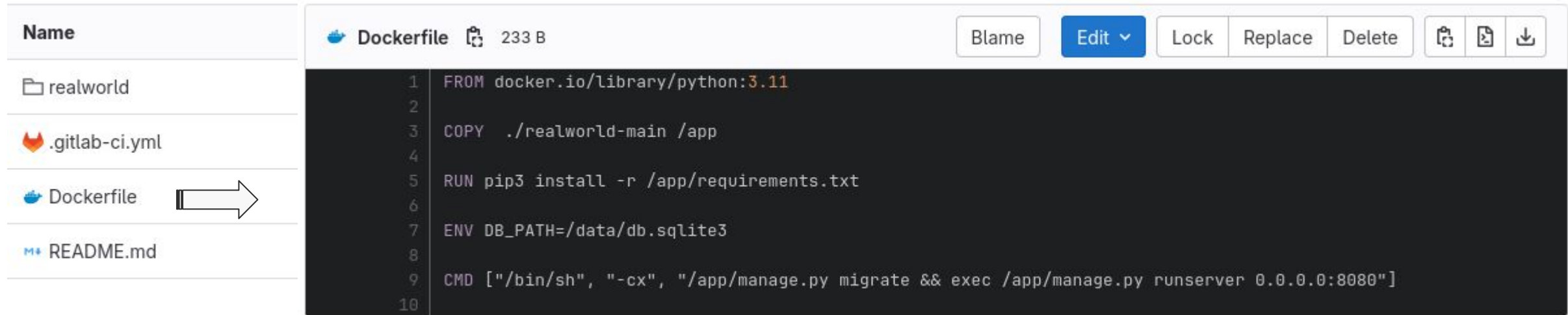
```
.gitlab-ci.yml 615 B
Blame Edit Lock Replace Delete
1 stages:
2   - build
3   - deploy
4
5 'Build container image':
6   stage: build
7   image:
8     name: gitlab-registry.cern.ch/ci-tools/docker-image-builder
9     entrypoint: [""]
10  script:
11    # Prepare Kaniko configuration file, 'CI_*' variables come from Gitlab CI
12    - echo "{\"auths\":{\"$CI_REGISTRY\":{\"username\":\"$CI_REGISTRY_USER\",\"password\":\"$CI_REGISTRY_PASSWORD\"}}}" > /kaniko/.docker/config.j
13    - /kaniko/executor --context $CI_PROJECT_DIR --dockerfile $CI_PROJECT_DIR/Dockerfile --destination ${CI_REGISTRY_IMAGE}:0.1
14    - echo "Successfully built image and pushed to ${CI_REGISTRY_IMAGE}:latest"
15
16
```

Useful documentation: <https://gitlab.docs.cern.ch/docs/>

Upstream documentation: <https://docs.gitlab.com/>

GitLab CI Examples: <https://gitlab.cern.ch/gitlabci-examples>

Let's look into the repo



The screenshot shows a GitLab repository interface. On the left, a file list includes 'realworld', '.gitlab-ci.yml', 'Dockerfile', and 'README.md'. A white arrow points to the 'Dockerfile' entry. The main area displays the content of the Dockerfile, which is 233 bytes. The Dockerfile content is as follows:



```
1 FROM docker.io/library/python:3.11
2
3 COPY ./realworld-main /app
4
5 RUN pip3 install -r /app/requirements.txt
6
7 ENV DB_PATH=/data/db.sqlite3
8
9 CMD ["/bin/sh", "-cx", "/app/manage.py migrate && exec /app/manage.py runserver 0.0.0.0:8080"]
10
```


Useful documentation: <https://gitlab.docs.cern.ch/docs/>

Upstream documentation: <https://docs.docker.com/reference/dockerfile/>


GitLab CI Examples: <https://gitlab.cern.ch/gitlabci-examples>



Let's look into the repo

 **csc-development-exercise** 

 master

[History](#) [Find file](#) [Edit](#) [Code](#)

 **Exercise contents**
Francisco Borges Aurindo Barros authored 4 days ago

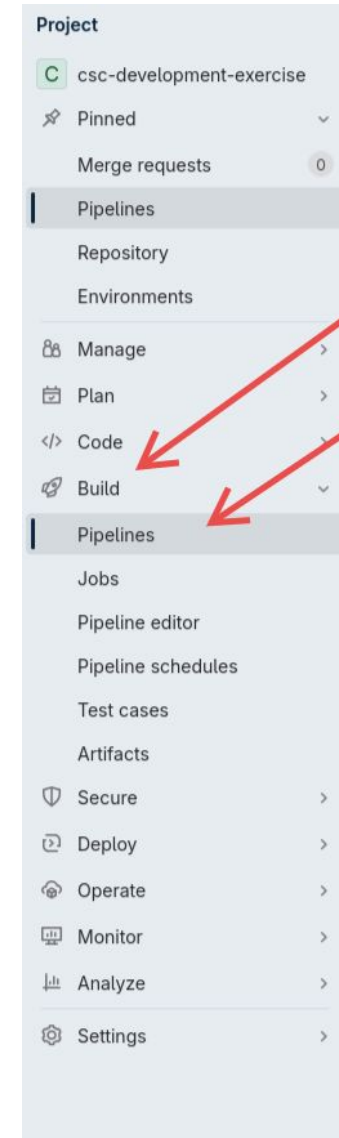
Name	Last commit	Last update
------	-------------	-------------



Fix image building

Ok, pipeline fails ... why?

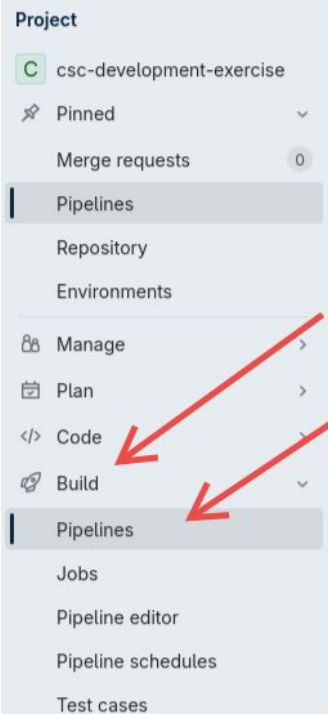
Let's check the logs of the CI



Fix image building

Ok, pipeline fails ... why?

Let's check the logs of the CI



Status	Pipeline	Created by	Stages
Failed 00:00:29 4 days ago	Exercise contents #8389599 master 63c958f1		Failed

Fix image building

Ok, pipeline fails ... why?

Let's check the logs of the CI

Solution:

Change the Dockerfile `COPY` command.

Instead of `realworld-main`, is `realworld`.

```
2  
3 COPY ./realworld-main /app  
4
```



```
2  
3 COPY ./realworld /app  
4
```


Fix image building

```
Successfully installed asgiref-3.8.1 django-4.0.1 django-htmx-1.8.0 django-taggit-2.1.0 django-widget-tweaks-1.4.12 markdown-3.3.6 sqlparse-0.5.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: pip install --upgrade pip
INFO[0027] Taking snapshot of full filesystem...
INFO[0029] ENV DB_PATH=/data/db.sqlite3
INFO[0029] CMD ["/bin/sh", "-cx", "/app/manage.py migrate && exec /app/manage.py runserver 0.0.0.0:8080"]
$ echo "Successfully built image and pushed to ${CI_REGISTRY_IMAGE}:latest"
Successfully built image and pushed to gitlab-registry.cern.ch/fborgesa/csc-development-exercise:latest
Cleaning up project directory and file based variables
Job succeeded
```

Now we do image pulling

Can we pull the image... ?

Both docker and podman CLI can be used

```
docker pull gitlab-registry.cern.ch/<ACCOUNT>/csc-development-exercise:latest
```

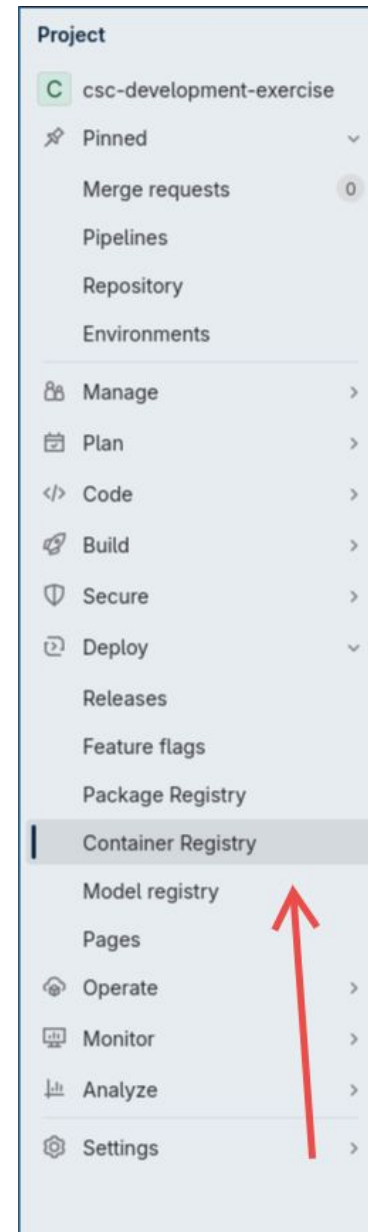
```
podman pull gitlab-registry.cern.ch/<ACCOUNT>/csc-development-exercise:latest
```

Now we do image pulling

Can we pull the image... ?

No? why?

Where can we find our available images?



Now we do image pulling

Project

- csc-development-exercise
- Pinned
- Merge requests 0
- Pipelines
- Repository
- Environments

Can we pu
No? why?
images?

The [next-generation container registry](#) is now available for upgrade and testing on self-managed instances as a Beta feature. This upgraded registry supports online garbage collection, and has significant performance and reliability improvements.

Container Registry

1 Image repository | Cleanup is not scheduled. [Set up cleanup](#) | Container Scanning for Registry: Off

CLI Commands

Filter results [Search] Updated [Sort]

csc-development-exercise 1 tag Published 2 days ago



Settings

Now we do image pulling

Can we pull
No? why?
images?

Project

- csc-development-exercise
- Pinned
- Merge requests 0
- Pipelines
- Repository
- Environments

csc-development-exercise

1 tag Cleanup disabled Created Oct 25, 2024 13:44

Filter results

1 tag

0.1 383.94 MIB

ature. This X

ommands v

Updated v

ed 2 days ago

Settings >

Change the version from latest to a release

Let's go to the '.gitlab-ci.yml' file

```
.gitlab-ci.yml
```

```
-|export DATE=$(date -u +%Y.%m.%dT%H-%M-%SZ);  
  case "$CI_COMMIT_BRANCH" in  
  master) export TAG="RELEASE-${DATE}";;  
  *) export TAG="${CI_COMMIT_BRANCH}-${CI_COMMIT_SHORT_SHA}";;  
  esac  
  ...
```

Change the version from latest to a release

we can make it **simpler** for now

Let's go to the '.gitlab-ci.yml' file

```
.gitlab-ci.yml
```

```
- /kaniko/executor ... --destination ${CI_REGISTRY_IMAGE}:latest
```

Have the image in our local machine

```
podman pull <image>
```

```
podman run -p 8080:8080 -name csc <image>
```



Lets create an account

- Access <sign up>
- Create an account, any example will do.
- Login works!

conduit

[Home](#) [New Article](#) [Sign In](#) [Sign up](#)

Sign up

[Have an account?](#)

Lets restart the application...

- Stop the container.
- Start it again, and try to login, does it work?

conduit

[Home](#) [New Article](#) [Sign In](#) [Sign up](#)

Sign up

[Have an account?](#)

Lets restart the application...

- Stop the container.
- Start it again, and try to login, does it work?

Why?

conduit

[Home](#) [New Article](#) [Sign In](#) [Sign up](#)

Sign up

[Have an account?](#)

Lets mount the DB file

```
podman run -p 8080:8080 \  
  -name csc \  
  --v .db.sqlite3:/app/db.sqlite3:z \  
  <image>
```

Does not work right away?

conduit

[Home](#) [New Article](#) [Sign In](#) [Sign up](#)

Sign up

[Have an account?](#)

Lets mount the DB file

```
podman run -p 8080:8080 \  
  -name csc \  
  --v .db.sqlite3:/app/db.sqlite3:z \  
  <image>
```

Does not work right away?

```
podman stop csc; podman rm csc
```

conduit

[Home](#) [New Article](#) [Sign In](#) [Sign up](#)

Sign up

[Have an account?](#)

Lets create an account, **again...**

- Create an account, **again**.
- Restart the application.
- Login again.

conduit

[Home](#) [New Article](#) [Sign In](#) [Sign up](#)

Sign up

[Have an account?](#)

Bonus: Dive tool

<https://github.com/wagoodman/dive>

dive <image>

```
● Current Layer Contents |
Permission  UID:GID    Size  Filetree
drwxrwxrwx  0:0       94 kB  — app
-rw-rw-rw-  0:0        34 B  |   — .gitignore
-rw-rw-rw-  0:0       1.1 kB  |   — LICENSE
-rw-rw-rw-  0:0       667 B  |   — README.md
-rwxrwxrwx  0:0       665 B  |   — manage.py
drwxrwxrwx  0:0       43 kB  |   — ⊕ realworld
-rw-rw-rw-  0:0       100 B  |   — requirements.txt
drwxrwxrwx  0:0       24 kB  |   — ⊕ spec
drwxrwxrwx  0:0       24 kB  |   — ⊕ templates
-rwxrwxrwx  0:0         0 B  |   — bin → usr/bin
drwxr-xr-x  0:0         0 B  |   — boot
```

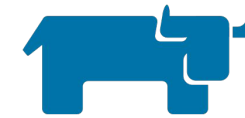
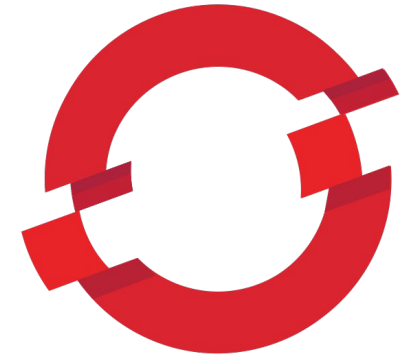
We deployed on our machines, where else can we do that?

Anywhere:

- Any Kubernetes flavoured cluster
- In any remote machine with a Docker runtime

At **CERN**:

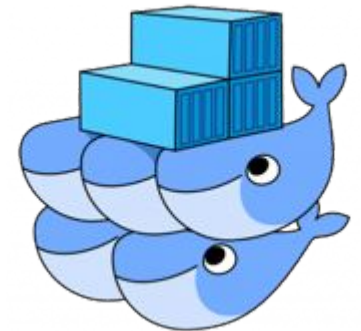
- Kubernetes
 - Docs: <https://kubernetes.docs.cern.ch/>
- Use **PaaS** for small applications
 - Docs: <https://paas.docs.cern.ch/>



RANCHER



K3S





home.cern