

A Tale of

Dependency Hell

DISCLAIMER

ALL CHARACTERS AND EVENTS DEPICTED HERE
ARE **COMPLETELY FICTIONAL**. ANY SIMILARITY
WITH REAL PEOPLE IS **PURELY COINCIDENTAL**

Dependency Hell



Dependency Hell

bora comparar as
versões

pandog - 1.0.1
ggblob - 12.3.1
uuGCNA - 2.1.0
enblicher - 3.1.9
mcapply - 0.0.1



Helder

AHÁ!
pandog - 1.0.1
ggblob - 12.3.1
uuGCNA - 2.1.0
***enblicher* - 3.2.1**
mcapply - 0.0.1



Pedro

Dependency Hell

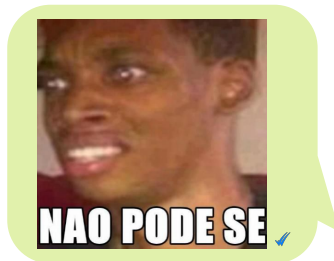


Helder

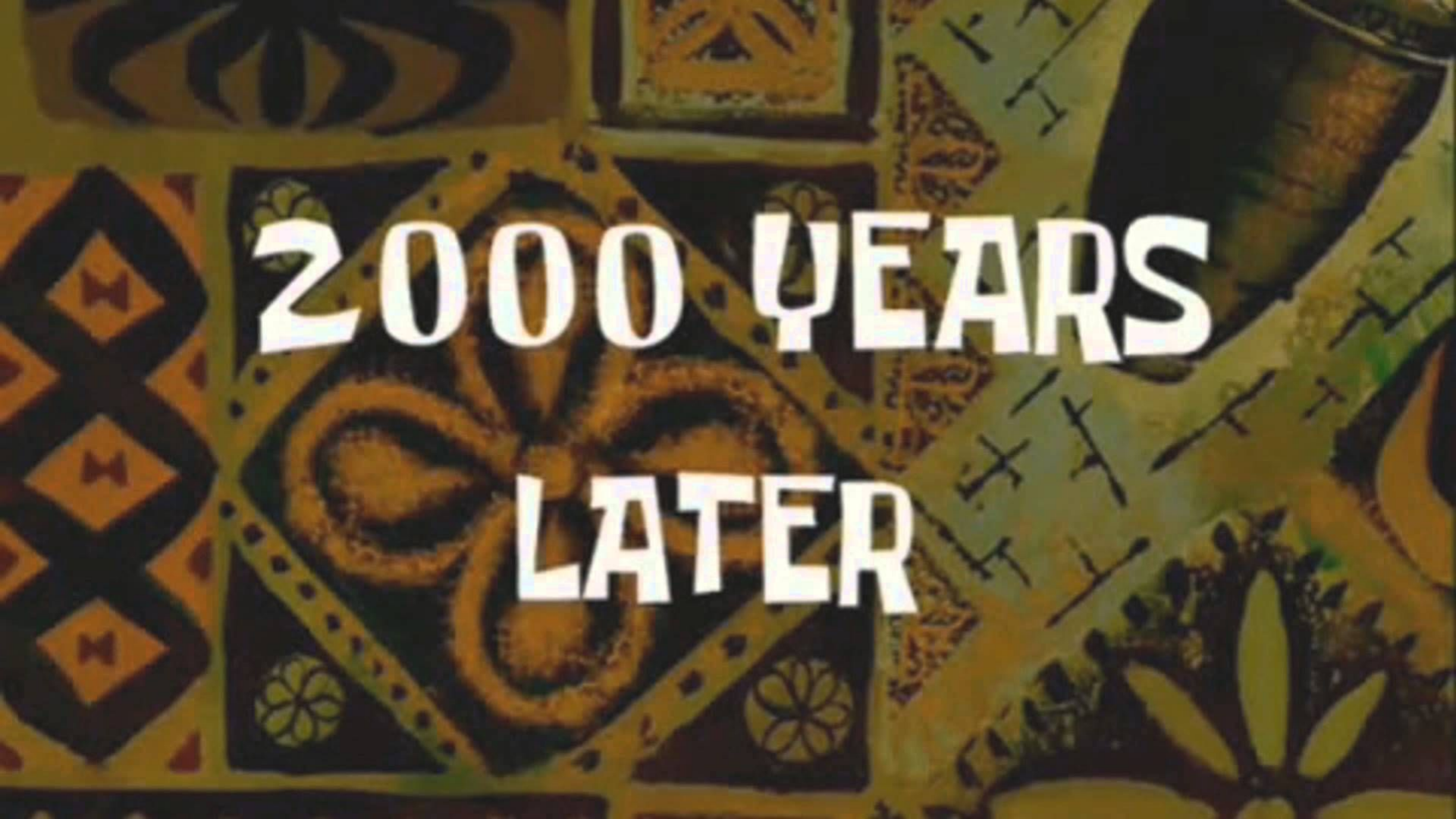
enblicher - 3.2.1
Agora vai mulek!

...

não foi 😭



Pedro



**2000 YEARS
LATER**

Dependency Hell

Qual é a versão do seu Linux? ✓

What?! Eu to no Windows!

Só pode ser isso!
Tem que atualizar pra nova versão *L.1.N.00.X* do Windows ✓



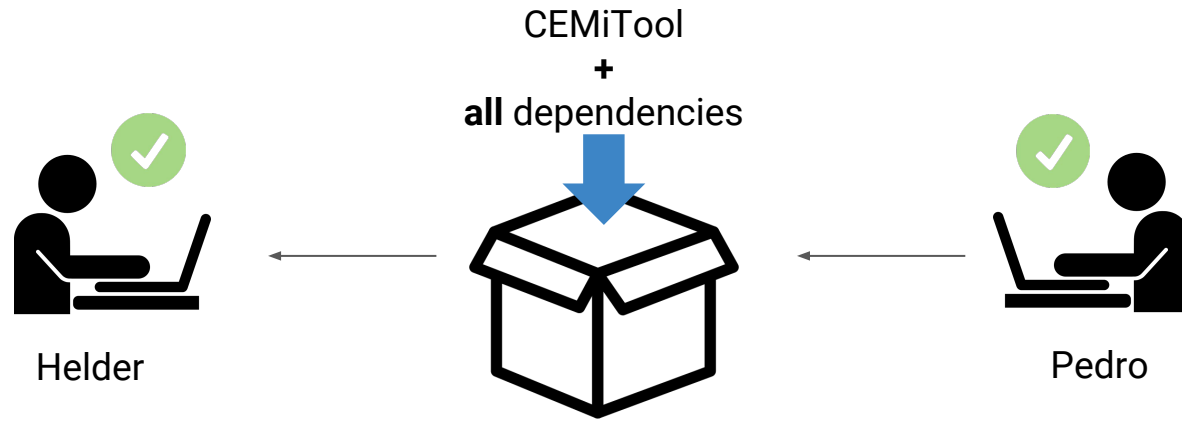
Helder



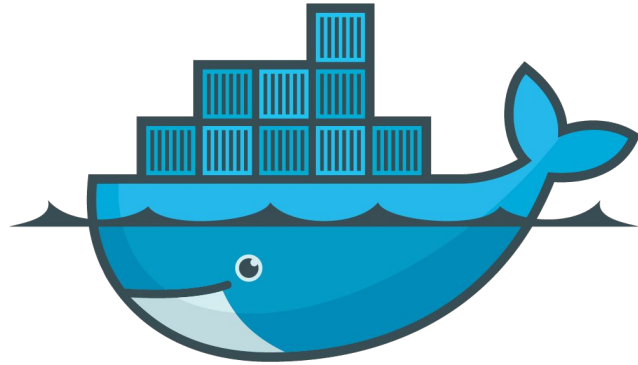
Pedro



What if...



Meet



docker

What is **docker** ?



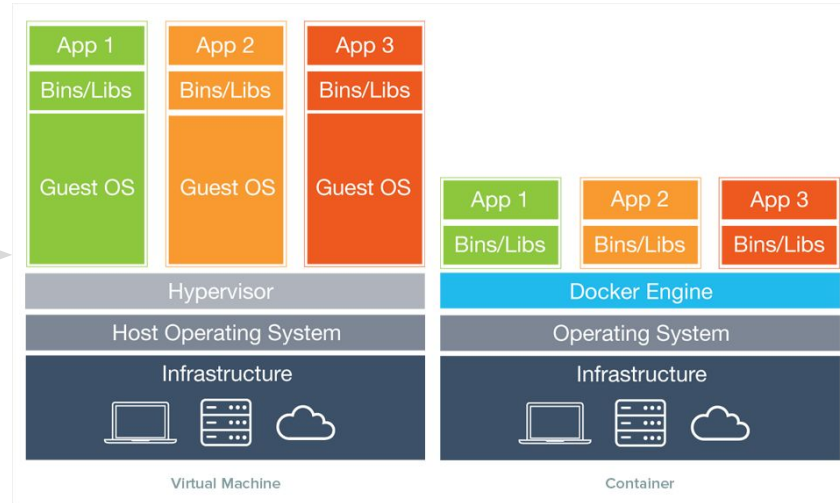
Container System

What are containers?



Operating-system-level
Virtualization Method

VMs x Containers



Core Concepts



Images



Docker Hub



Containers

Images

“A container image is a lightweight, stand-alone, executable package of a piece of software that includes **everything** needed to run it: **code, runtime, system tools, system libraries** and **settings**”



Images



Dockerfile

Base Image



```
FROM ubuntu
```

Dependencies



```
RUN apt-get install update && \  
    apt-get install -y git r-base && \  
    git clone https://github.com/csbl-usp/CEMiTool.git . && \  
    /usr/bin/R CMD INSTALL CEMiTool
```

Images



Build your CEMiTool image

```
$ > docker build -f Dockerfile -t csbl-usp/CEMiTool
```

Dockerfile name

Output Image name

Docker Hub



Similar to GitHub



Stores Docker images

Docker Hub pushing images



Create an user: csbl-usp



Create an open repository: csbl-usp/CEMiTool



Login to Docker Hub

```
$ > docker login
```



Push your CEMiTool image

```
$ > docker push csbl-usp/CEMiTool
```




Docker Hub pulling images



Pull the CEMiTool image from Docker Hub

```
$ > docker pull csbl-usp/CEMiTool
```

Containers

-  Running processes using Images = Container
-  Ephemeral: created and destroyed every time
-  All dependencies are already there

Containers



Let's run a container using CEMiTool Image

```
$ > docker run -it csbl-usp/CEMiTool /usr/bin/R
```

Interactive shell

Image name

Command

Containers



Inside the container

- only R being executed
- exiting R process kills container
- everything is deleted



```
> library(CEMiTool)
```

```
> # perform your analysis
```

```
> quit() # kill R, kill container, kill everybody
```

Containers



Analysis inside containers

How to **input data inside** containers?

How to avoid **results** being **deleted**?

Volumes!

Containers volumes



Share directories between host and container

```
$ > docker run -it -v $PWD:/tmp/work csbl-usp/CEMiTool /usr/bin/R
```

Volume creation
hostdir : containerdir

```
> library(CEMiTool)
```

```
> exprs <- read.table("/tmp/work/expression.csv") #input data available
```

```
> write.table("/tmp/work/results.csv") #save results inside /tmp/work
```

```
> quit() #container gets killed but data is persisted
```


Containers



Running CEMiTool with only one command

```
$ > docker run -v $PWD:/tmp/work csbl-usp/CEMiTool \  
CEMiTool expression.tsv --output=cemitool_analysis \  
--sample-annot=annotation.tsv \  
--gene-column=genes \  
--sample-column=samples
```



Results will be created inside ***\$PWD/cemitool_analysis***

Containers orchestration



Single Host:

- Docker Compose



Multi Host:

- Docker Swarm Mode
- Kubernetes
- Mesos



