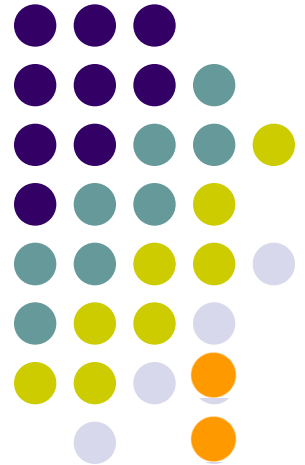


Arquitetura de Software

Multiprocessamento

Interrupção, Virtualização e Containerização

José Motta Lopes
josemotta@bampli.com



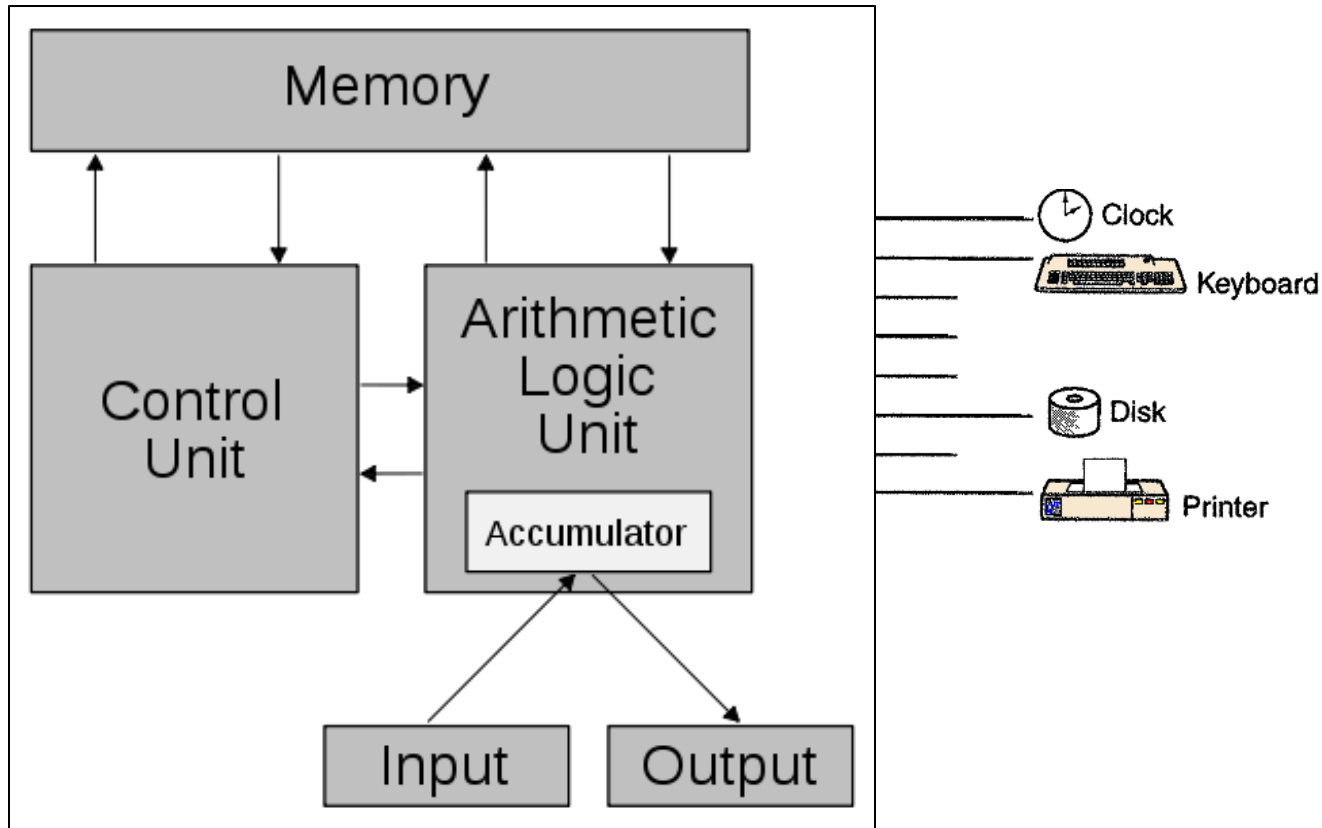


Agenda

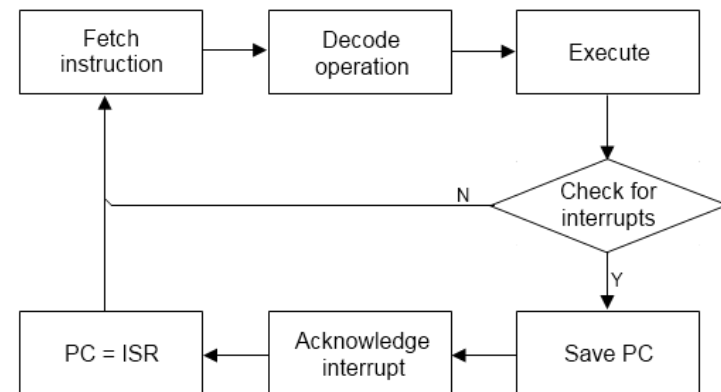
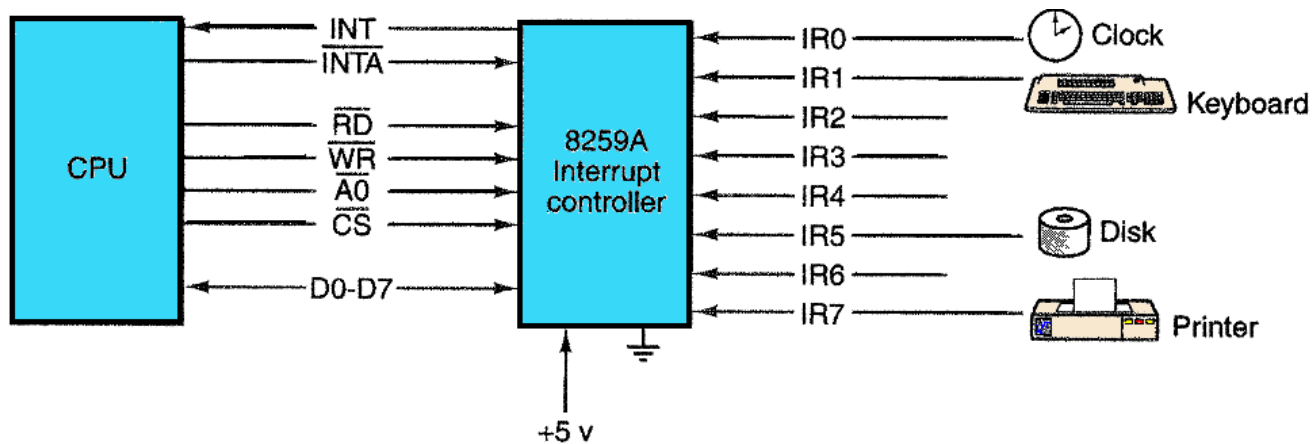
- CPU von Neumann
- Interrupção
- Multiprocessamento
- Virtualização
- Containerização
- Docker
- Arquitetura Docker
- Vantagens Docker
- Dockerfile
- Demo Docker-Python
- Projeto IoT: Estação do Clima
- Workflow P&D Software
- Estação Remota
- Estação Usuário



CPU von Neumann



Interrupção

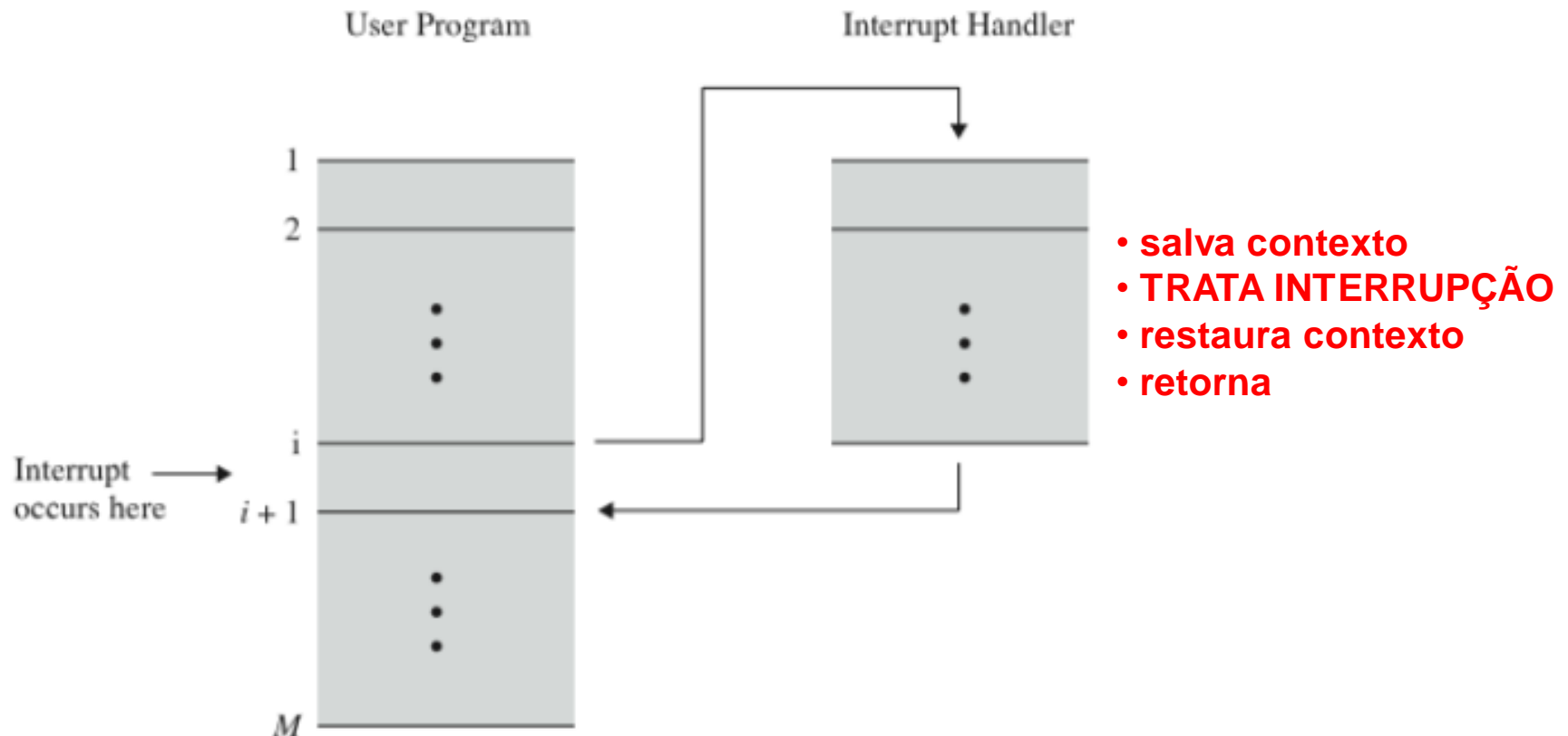




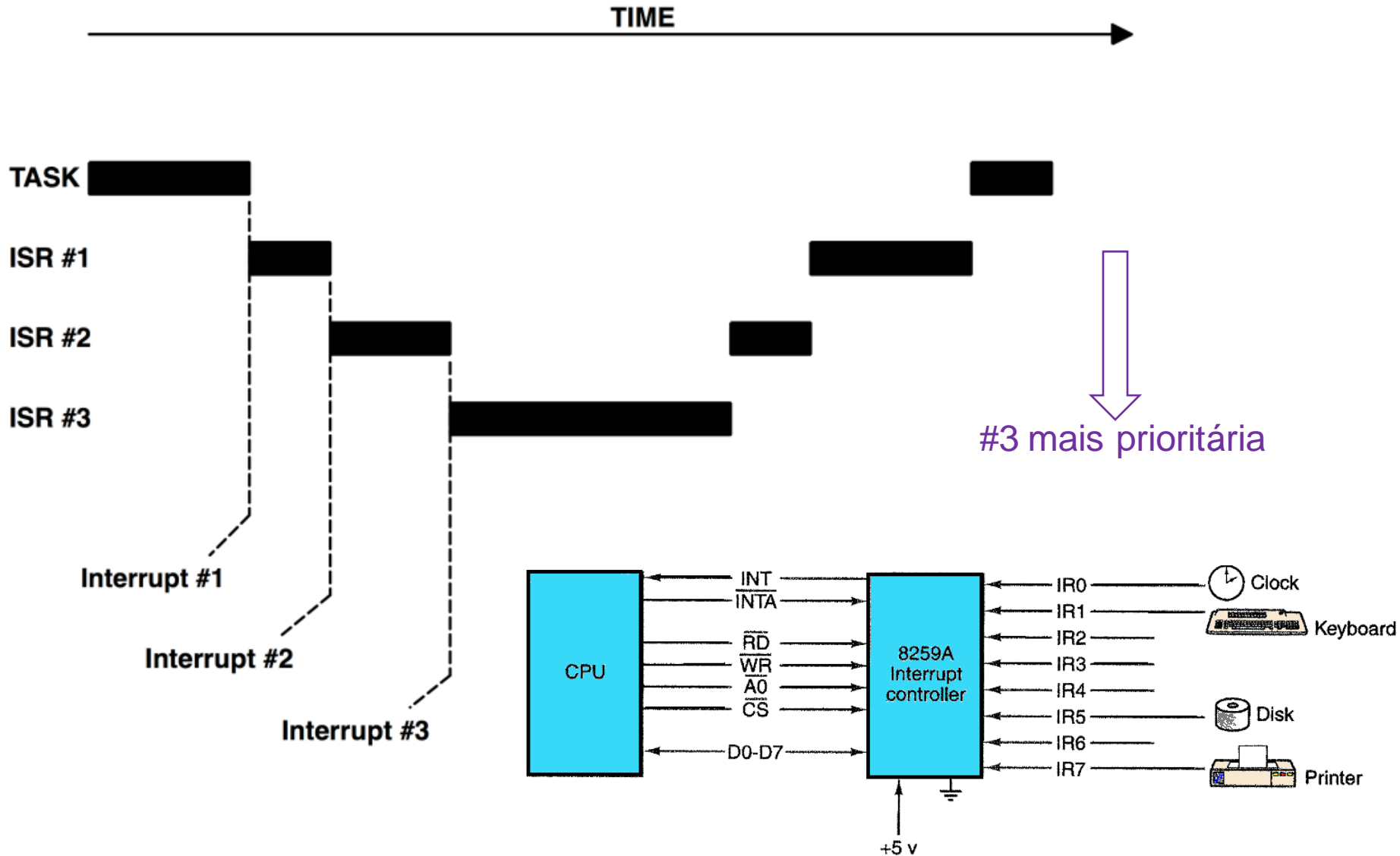
Rotina tratamento interrupção

Fontes de interrupção:

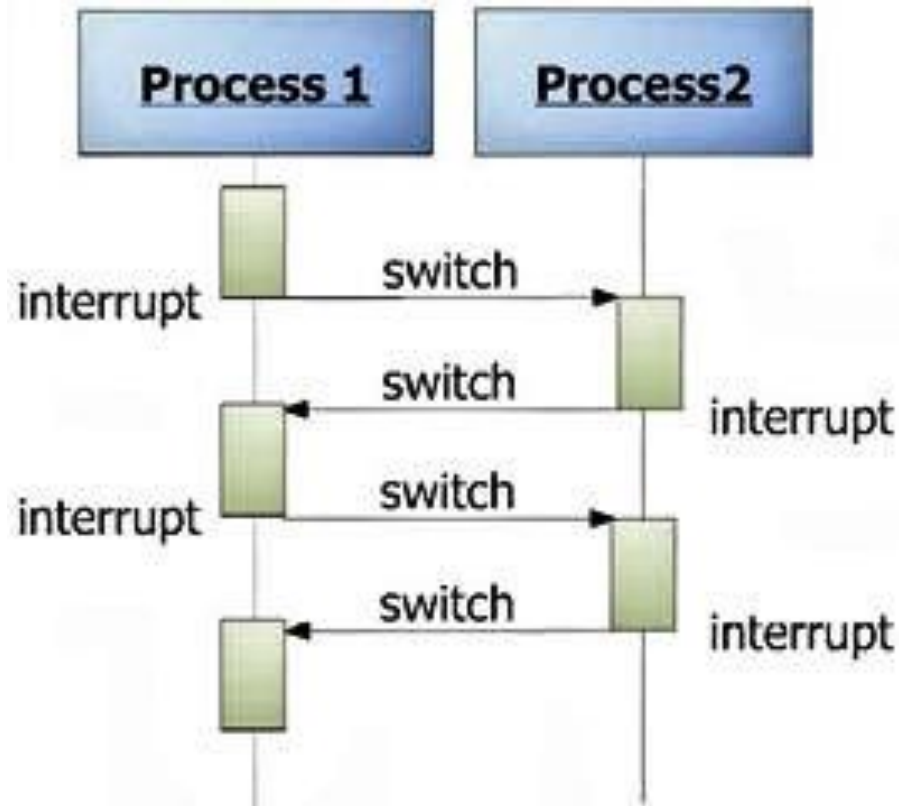
- Hardware
- Processador
- Software



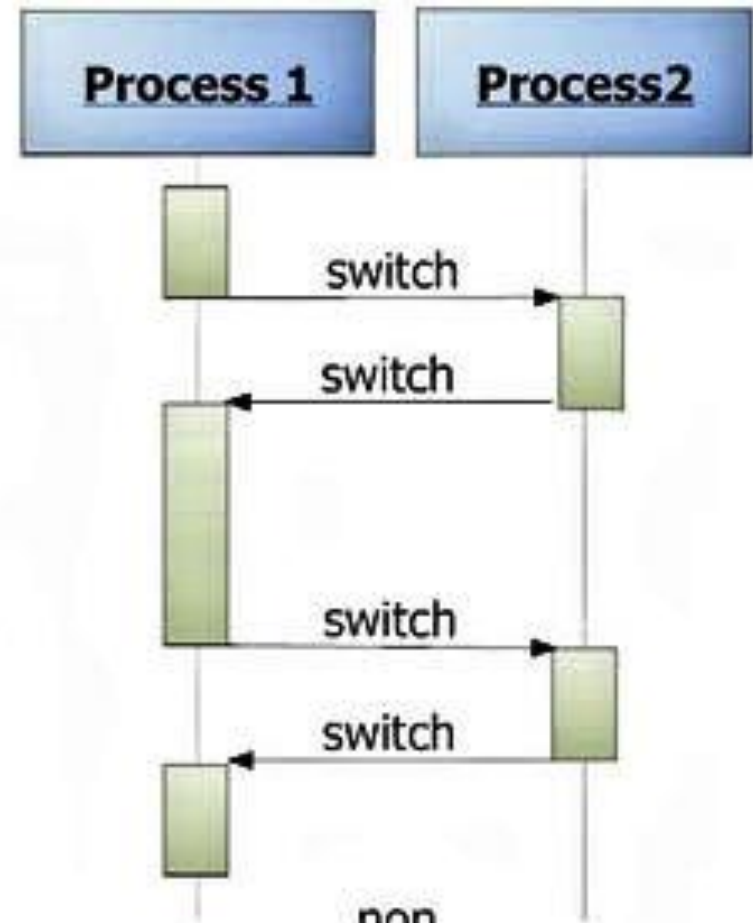
Interrupção



Multiprocessamento

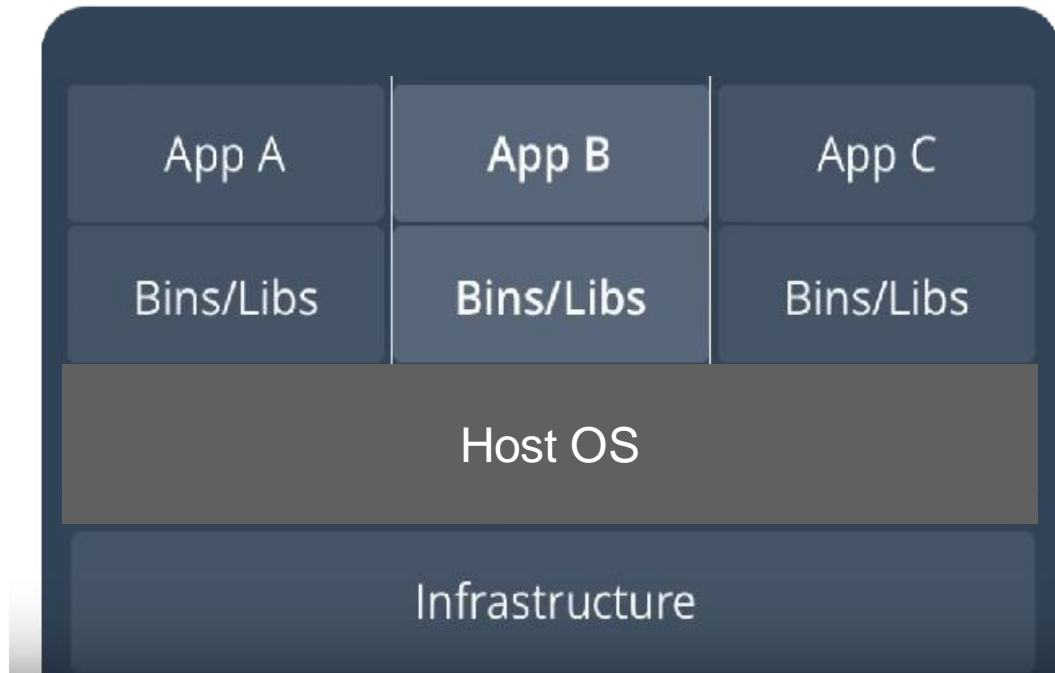


pre-emptive

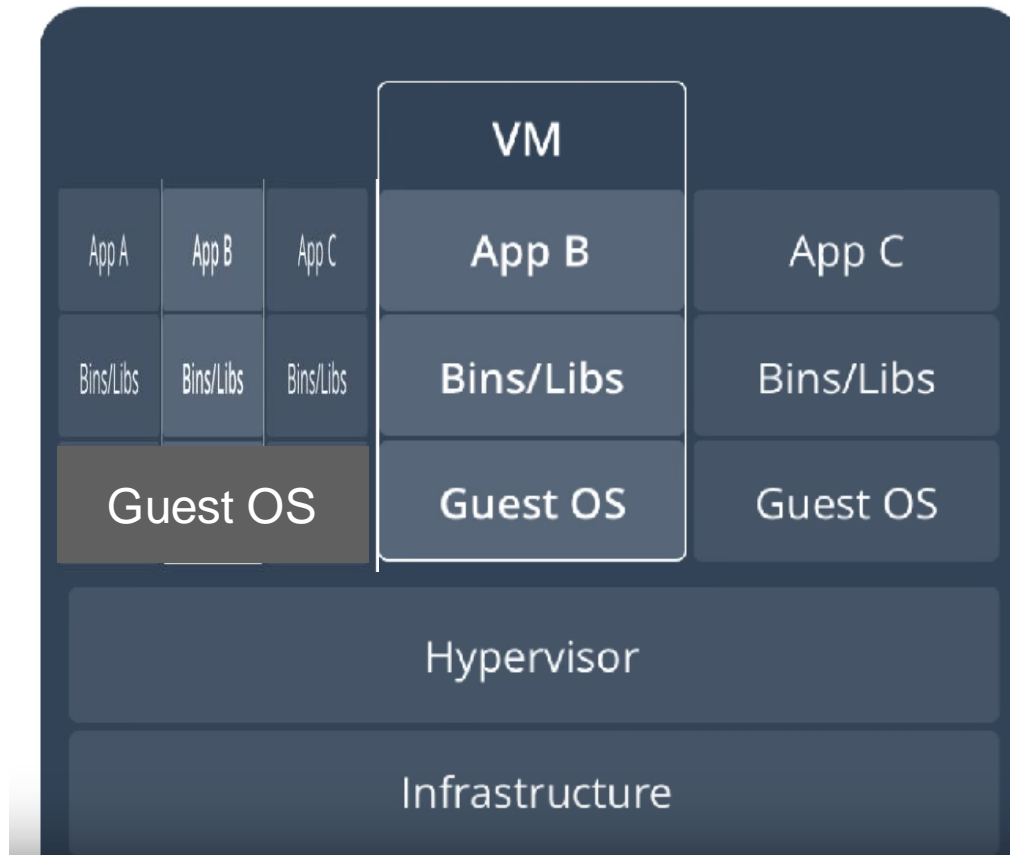


non
pre-emptive

Multiprocessamento



Virtualização



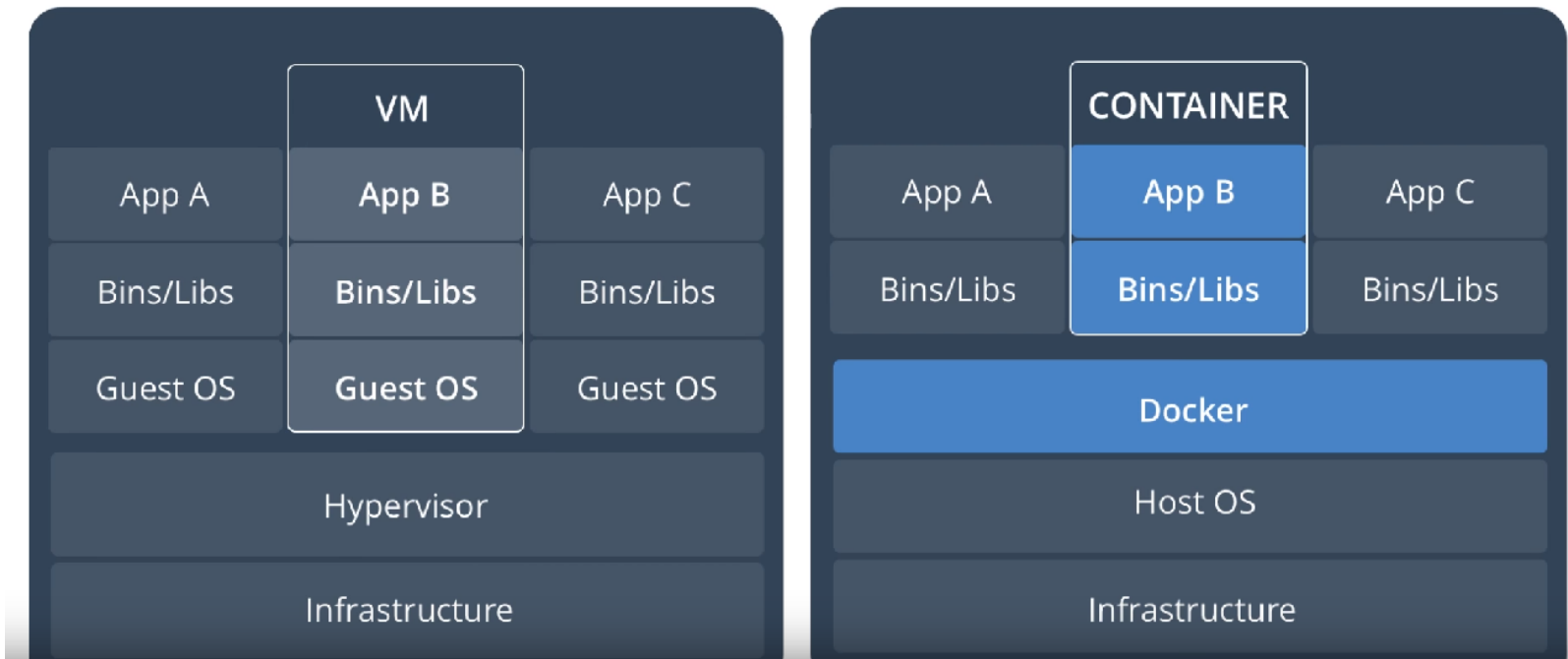
Containerização



Container x VM



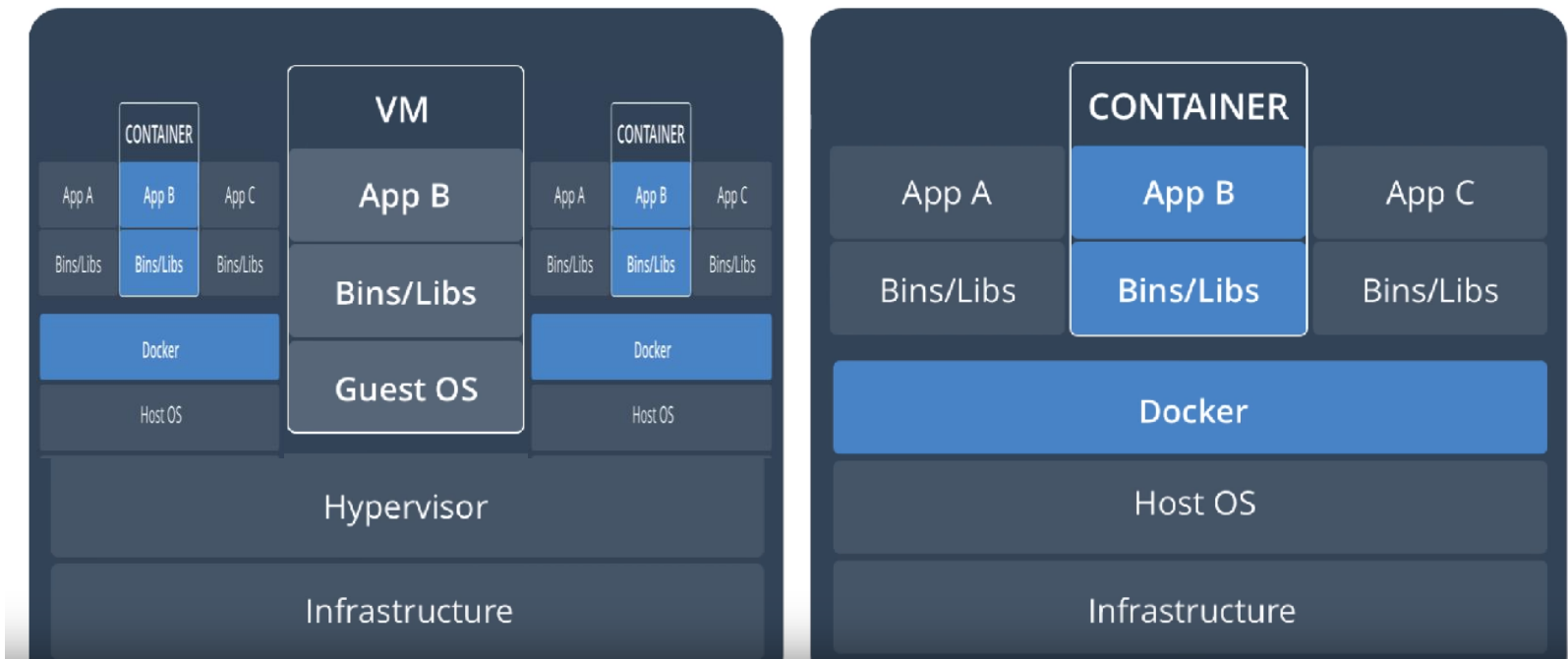
Docker vs VM



Containers + VMs



Docker + VM

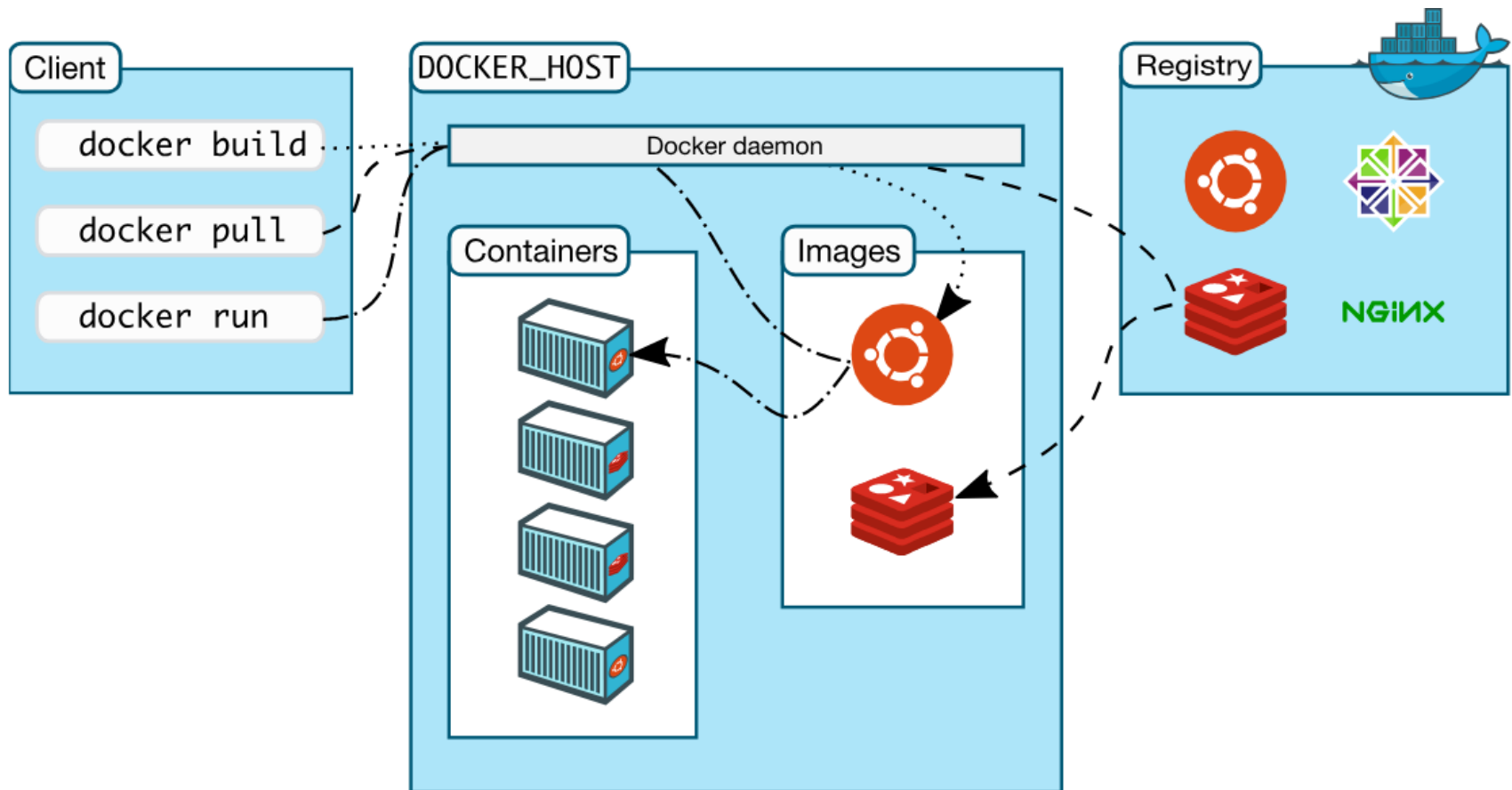


Docker



- **Plataforma para desenvolvedores e sysadmins**
- **Desenvolvimento, deploy e execução de aplicações**
- **Utiliza containers Linux para o deploy de aplicações**
- **Flexibilidade: qualquer aplicação pode ser containerizada**
- **Leve: deploy de atualizações e upgrades *on-the-fly***
- **Portabilidade: build local, deploy to cloud, run anywhere**
- **Escalável: distribuição automática de réplicas de containers**

Arquitetura Docker





Vantagens Docker

- Agiliza o ciclo de vida do desenvolvimento
- Ambiente padrão utilizando containers
- Desenvolvedores escrevem código localmente e compartilham
- Transferência de aplicações para ambiente de testes
- Re-deploy de upgrades e atualizações para ambiente de testes
- Ambiente de produção utilizando mesmas imagens
- Compatibilidade com ambientes diversos misturados
 - Laptop do desenvolvedor
 - Máquinas físicas e virtuais em datacenters
 - Provedores da nuvem
- Alternativa competitiva para potencializar as máquinas virtuais
- Eficiência em pequenos e médios deployments
- Ambientes Linux, Windows e MacOS
- Plataformas VMWare, Amazon AWS, Microsoft Azure



Dockerfile

- Define o que será incluído no container
- Documento contendo comandos para montar a imagem
- Docker monta imagens automaticamente a partir do Dockerfile

Exemplos de comandos típicos:

- `docker info`
- `docker build .`
- `docker run -it -p 80:80 nginx`

- `docker push usuario/imagem:latest`
- `docker pull usuario/imagem:latest`
- `docker run usuario/imagem:latest`



Dockerfile

Hello World com Python:

- `docker build -t bamplifier/docker-python-test .`
- `docker run -p 4000:80 bamplifier/docker-python-test`
- `docker push bamplifier/docker-python-test`
- `docker pull bamplifier/docker-python-test`

```
C:\ MINGW64:/c/Users/jo/Desktop/docker-python
jo@CANOAS22 MINGW64 ~/Desktop/docker-python
$ ls
app.py  Dockerfile  requirements.txt
jo@CANOAS22 MINGW64 ~/Desktop/docker-python
$ _
```



Dockerfile

```
jo@CANOAS22 MINGW64 ~/Desktop/docker-python
$ cat dockerfile
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
jo@CANOAS22 MINGW64 ~/Desktop/docker-python
$
```

App.py



```
$ cat app.py
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
          "<b>Hostname:</b> {hostname}<br/>" \
          "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)

jo@CANOAS22 MINGW64 ~/Desktop/docker-python
$
```

Demo Build

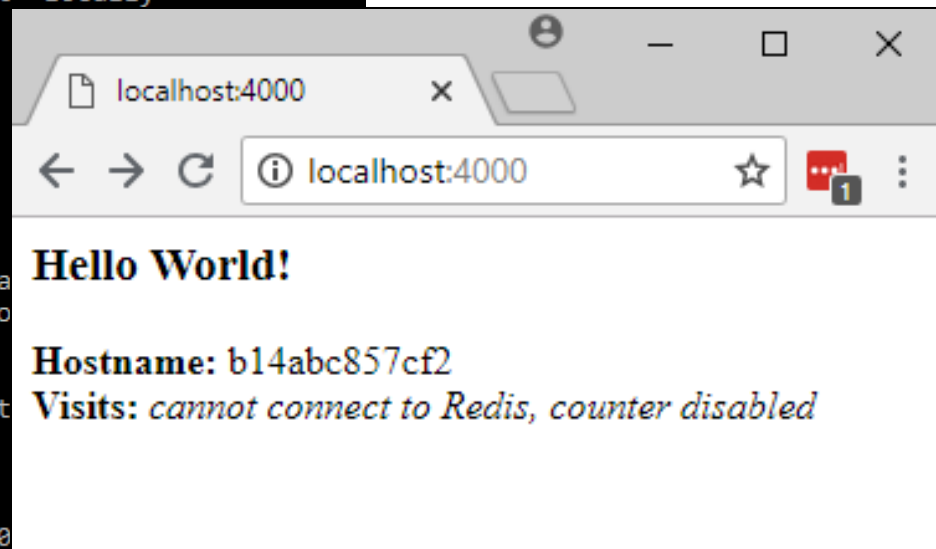


```
$ docker build .
Sending build context to Docker daemon 5.12kB
Step 1/7 : FROM python:2.7-slim
2.7-slim: Pulling from library/python
683abb4ea60: Already exists
8af590abc616: Already exists
f6cf6c5b7946: Already exists
e61a0272532e: Already exists
Digest: sha256:8ccd3a1e1836e2d1d1abda0c7f56b726fe70e2f4148190ce38456021c5ffc744
Status: Downloaded newer image for python:2.7-slim
--> 02ca219cf841
Step 2/7 : WORKDIR /app
Removing intermediate container 1b7090cd5e6e
--> d7d5a1318761
Step 3/7 : ADD . /app
--> e7a0d7e210ad
Step 4/7 : RUN pip install --trusted-host pypi.python.org -r requirements.txt
--> Running in 8690b6d6a8f8
Collecting Flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/7f/e7/08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb4b/Flask-1.0.2-py2.py3-none-any.whl (91kB)
Collecting Redis (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/3b/f6/7a76333cf0b9251ecf49efff635015171843d9b977e4ffcf59f9c4428052/redis-2.10.6-py2.py3-none-any.whl (64kB)
Collecting itsdangerous>=0.24 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/dc/b4/a60bcdba945c00f6d608d8975131ab3f25b22f2bcfe1dab221165194b2d4/itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2>=2.10 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/7f/ff/ae64bacdfc95f27a016a7bed8e8686763ba4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting Werkzeug>=0.14 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
Collecting click>=5.1 (from Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/34/c1/8806f99713ddb993c5366c362b2f908f18269f8d792aff1abfd700775a77/click-6.7-py2.py3-none-any.whl (71kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10->Flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/4d/de/32d741db316d8fdb7680822dd37001ef7a448255de9699ab4bfcdbf4172b/MarkupSafe-1.0.tar.gz
Building wheels for collected packages: itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for itsdangerous: started
  Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/2c/4a/61/5599631c1554768c6290b08c02c72d7317910374ca602ff1e5
  Running setup.py bdist_wheel for MarkupSafe: started
  Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/33/56/20/ebe49a5c612fffe1c5a632146b16596f9e64676768661e4e46
Successfully built itsdangerous MarkupSafe
Installing collected packages: itsdangerous, MarkupSafe, Jinja2, Werkzeug, click, Flask, Redis
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.0 Redis-2.10.6 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 8690b6d6a8f8
--> ffd86045138
Step 5/7 : EXPOSE 80
--> Running in f9b1f54276d1
Removing intermediate container f9b1f54276d1
--> be10482adc5f
Step 6/7 : ENV NAME World
--> Running in e12809f3f0ee
Removing intermediate container e12809f3f0ee
--> e16a11f5b811
Step 7/7 : CMD ["python", "app.py"]
--> Running in 0ecf26397201
Removing intermediate container 0ecf26397201
--> 436b23ebf863
Successfully built 436b23ebf863
```

Running



```
$ docker run -p 4000:80 bamplifier/docker-python-test
Unable to find image 'bamplifier/docker-python-test:latest' locally
latest: Pulling from bamplifier/docker-python-test
683abbb4ea60: Already exists
8af590abc616: Already exists
f6cf6c5b7946: Already exists
e61a0272532e: Already exists
6134331c2d29: Already exists
b5569e5b6380: Already exists
7dcaf1b1b3b2: Already exists
Digest: sha256:5fc2627e0d94acb7f2e9af72e8f51822946e5c6deca
Status: Downloaded newer image for bamplifier/docker-python-test
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
172.17.0.1 - - [11/Jul/2018 14:52:05] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [11/Jul/2018 14:52:06] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [11/Jul/2018 14:54:00] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [11/Jul/2018 14:54:07] "GET / HTTP/1.1" 200 -
```



```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
b14abc857cf2	bamplifier/docker-python-test	"python app.py"	38 minutes ago	Up 38 minutes	0.0.0.0:4000->80/tcp

Demo Docker-Python

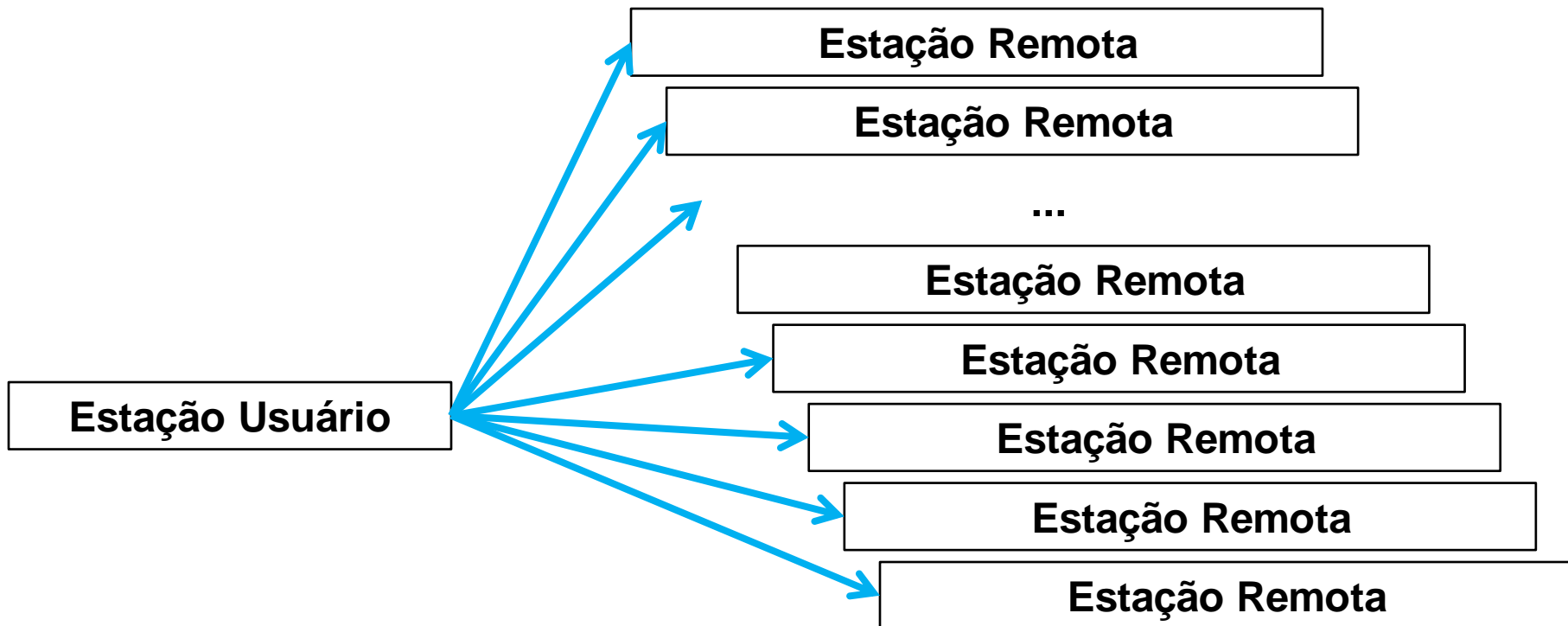


- Instalar Docker
- Demonstração

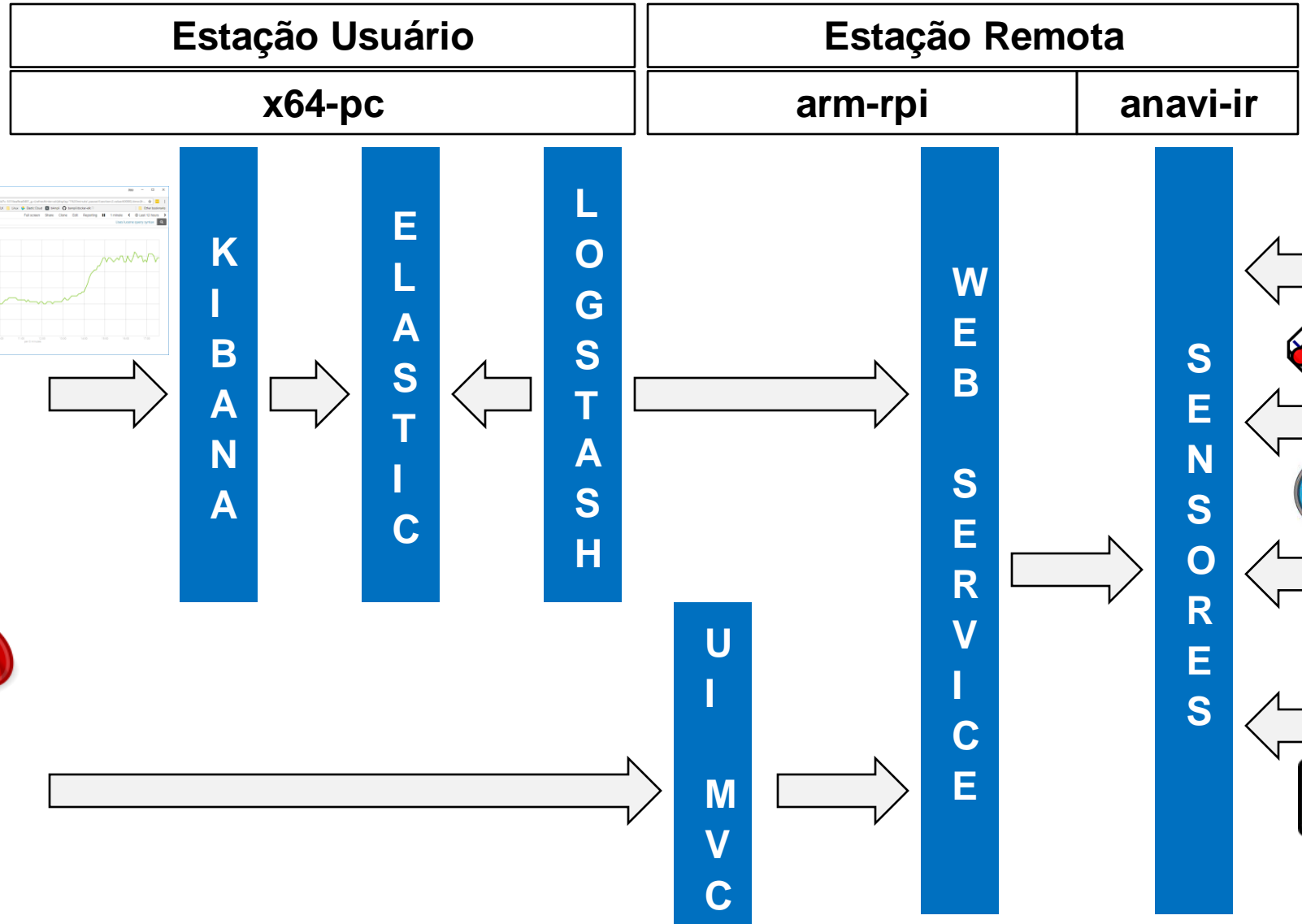


Projeto IoT: Estação do Clima

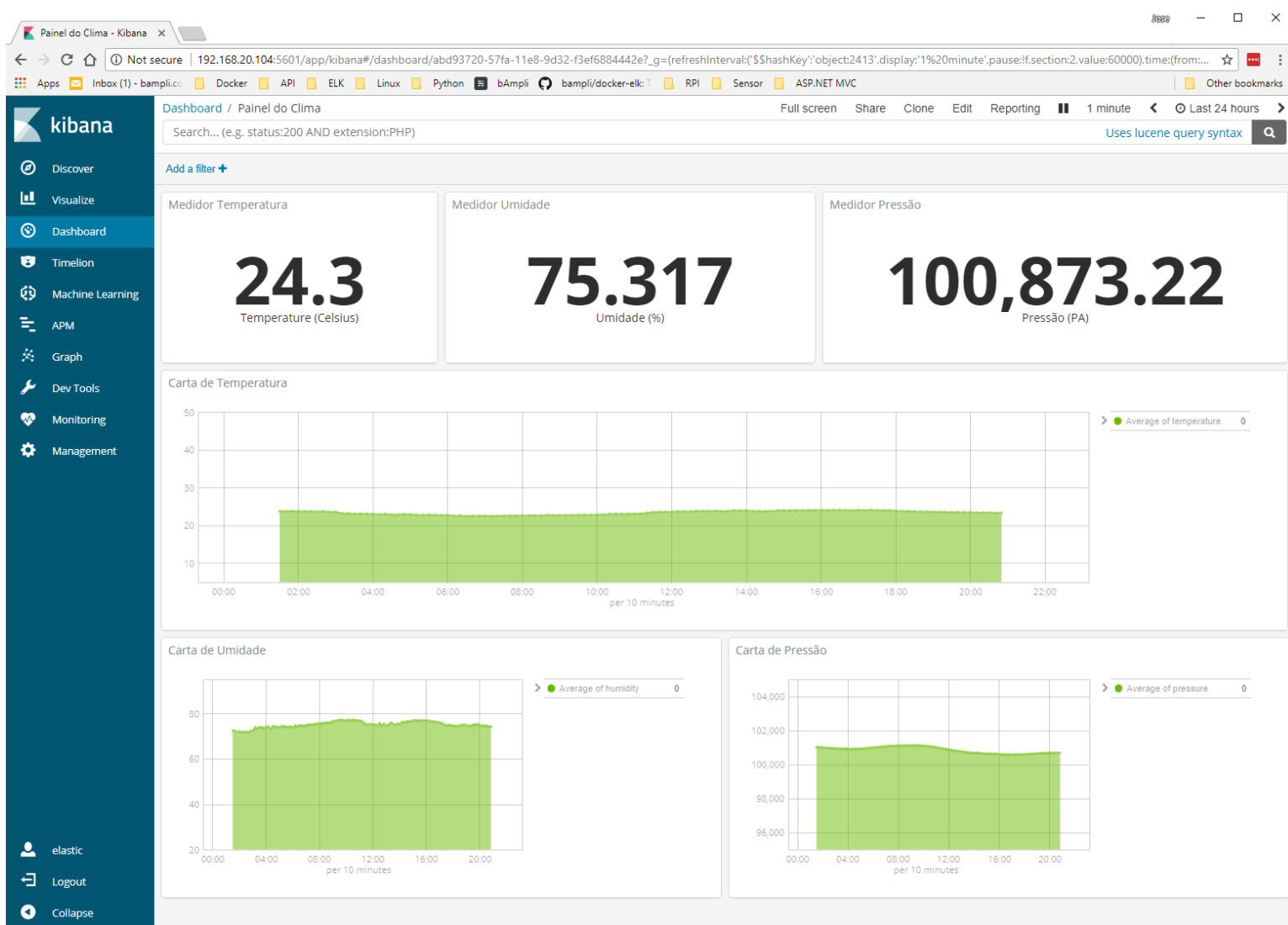
- Coleta de dados em tempo real por Estações Remotas
- Sensores de temperatura, pressão e umidade
- Painel do Clima na Estação do Usuário faz acompanhamento



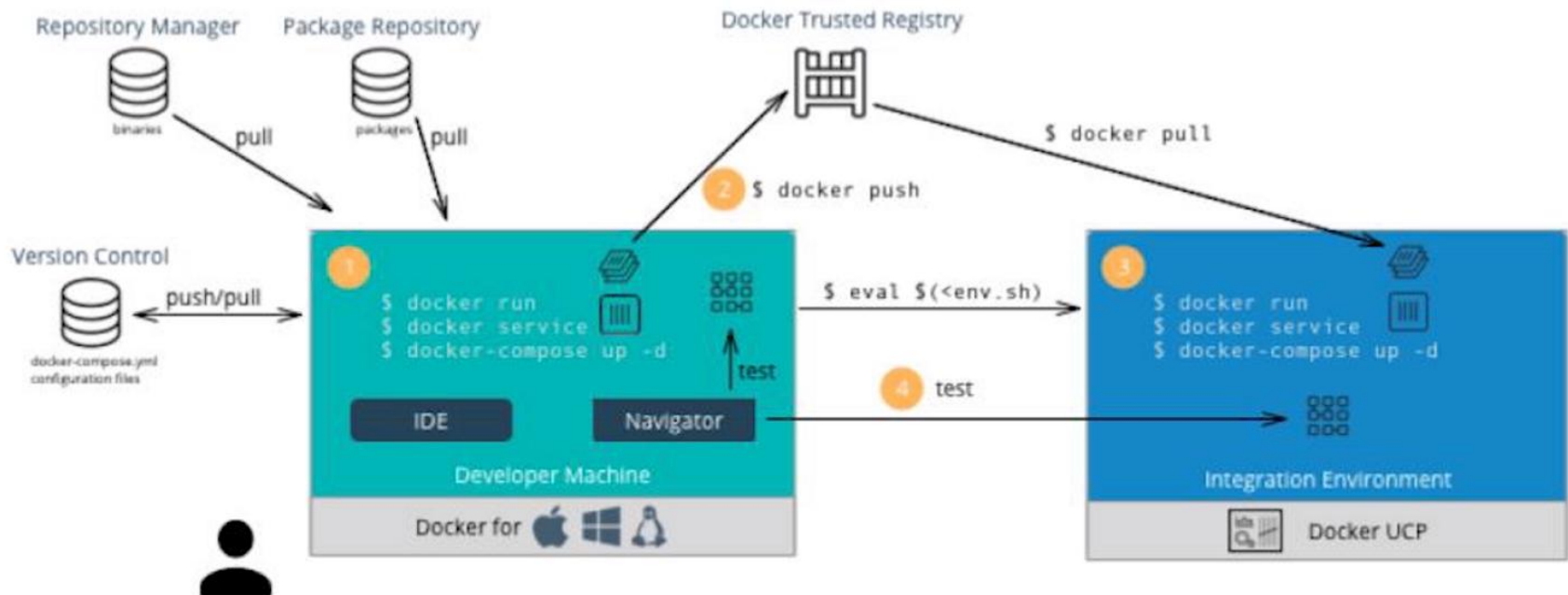
Projeto IoT: Estação do Clima



Painel do Clima



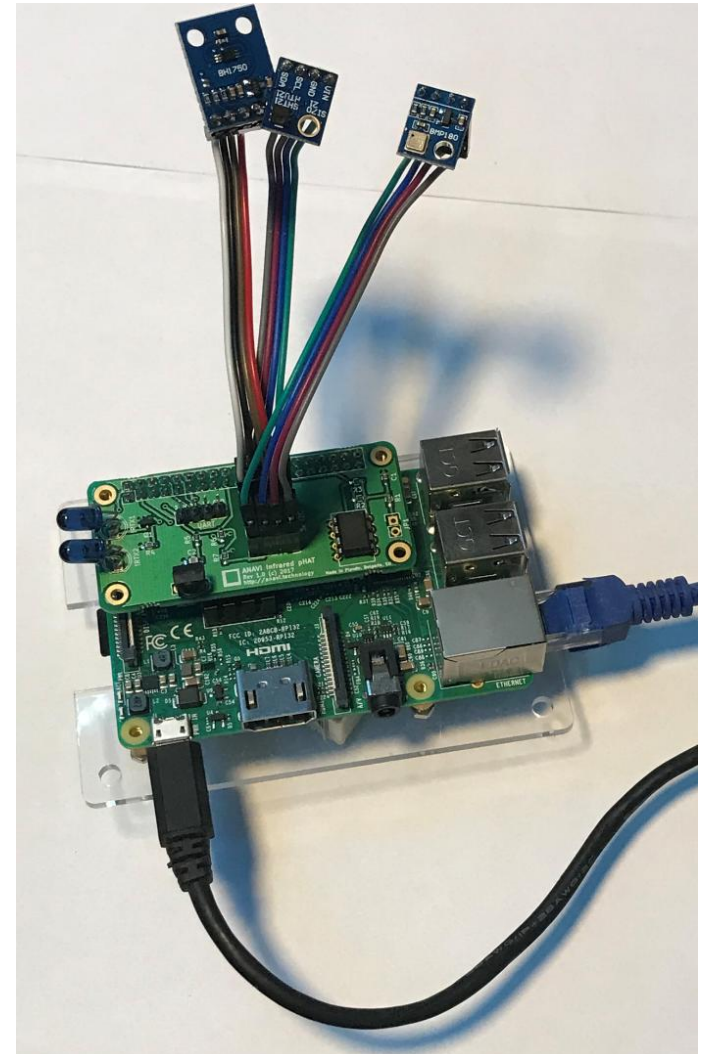
Workflow P&D Software





Estação Remota

- **Raspberry Pi 3 Model B**
 - CPU Quad Core 64-bit 1.2 GHz
 - 1 GB RAM
 - Micro SD 15 GB
 - Wireless Lan / Ethernet 100 Mbps
 - GPIO 40-pinos
- **ANAVI Infrared pHAT**
 - 2 x transmissores IR
 - Receptor IR photo sensor
 - Sensor pressão barométrica
 - Sensor temperatura e umidade
 - Sensor intensidade de luz





Estação Remota

A partir do [IoT Starter Pi Thing](#):

- API First Design
- SwaggerHub gera código de **web service**
- ASP.NET MVC gera código da **interface usuário**
- Visual Studio 2017 Community IDE integra projetos
- Build realizado em PC x64 veloz cria imagem Docker
- Push imagem para DockerHub registry
- Deploy na Raspberry Pi fazendo pull imagem
- Docker-Compose orquestra microservices & rede



Estação Usuário

- **ELK Stack**
 - Logstash rastreia informações
 - Elastic armazena informações
 - Kibana gera dashboard com gráficos
- **Docker for Windows com CentOS Linux**
- **Orquestração Docker-Compose**