Service Orchestration

BigData from testing to production

PhD. Ronal Muresano

Course Roadmap

Docker Swarm

Orchestration in a real private cluster (Deploying and Creating services)

Kubernetes

Production Orchestration. Automating deployment, scaling, and management of containerized applications

Orchestration for developing

Docker and Docker-Compose.

Big Data Challenges

An overview applied to the SMEs. Orchestration Philosophy

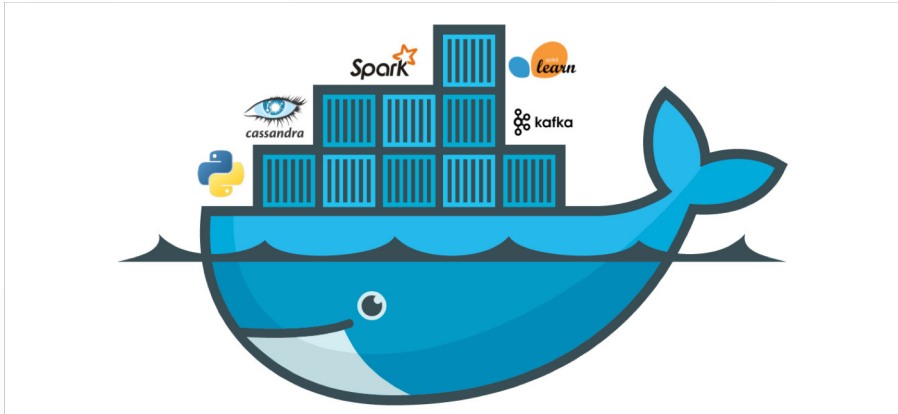
Cloud Infraestructure

Google Cloud and AWS deployments

Docker

It’s a software platform which allows to packages application with its dependencies with the objective of deploying them easily.

These packages are called containers.

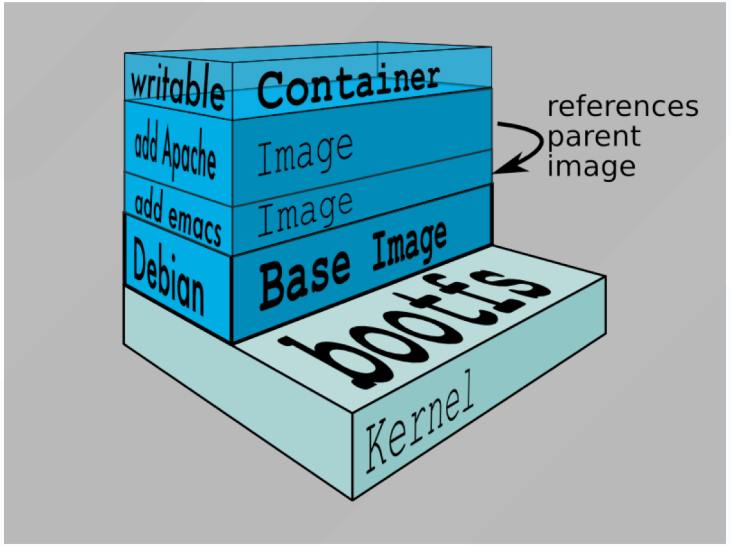


- Portability of your compute environment
- Self-sufficient
- Light-Weight
- Flexible and scalable
- Strengthen your engineering steps
- Reproducibility (As Data Scientists)

Source: https://towardsdatascience.com/how-docker-can-help-you-become-a-more-effective-data-scientist-7fc048ef91d5

Docker image

A docker image is a self-contained package in which is included all the necessary tools, code, runtime, libraries, environment variables, config files, dependencies, etc.



- An image is conformed by a set of layers which are Superimposition one to another.
- IMPORTANT: each instruction is a layer, in this sense, we can minimize the image size if some of the layer are grouped.
- A new layer is created when an image is modified.
- From an image, it can be created N containers that execute the image's content

To inspect the docker image layers: docker history <name of the image>

Docker Advantages

Standardization and Productivity

Ensure consistency across multiple development and release cycles, standardizing your environment.

CI Efficiency

Enables you to build a container image and use that same image across every step of the deployment process

Compatibility and Maintainability

Eliminate the “it works on my machine” problem once and for all. One of the benefits that the entire team will appreciate is parity.

Simplicity and Faster Configurations

One of the key benefits of Docker is the way it simplifies matters

Rapid Deployment

Docker manages to reduce deployment to seconds.

Continuous Deployment and Testing

Ensures consistent environments from development to production

Multi-Cloud Platforms

One of Docker’s greatest benefits is portability.

Isolation

Docker ensures your applications and resources are isolated and segregated.

Security

From a security point of view, it ensures that applications that are running on containers are completely segregated and isolated from each other

Scalable

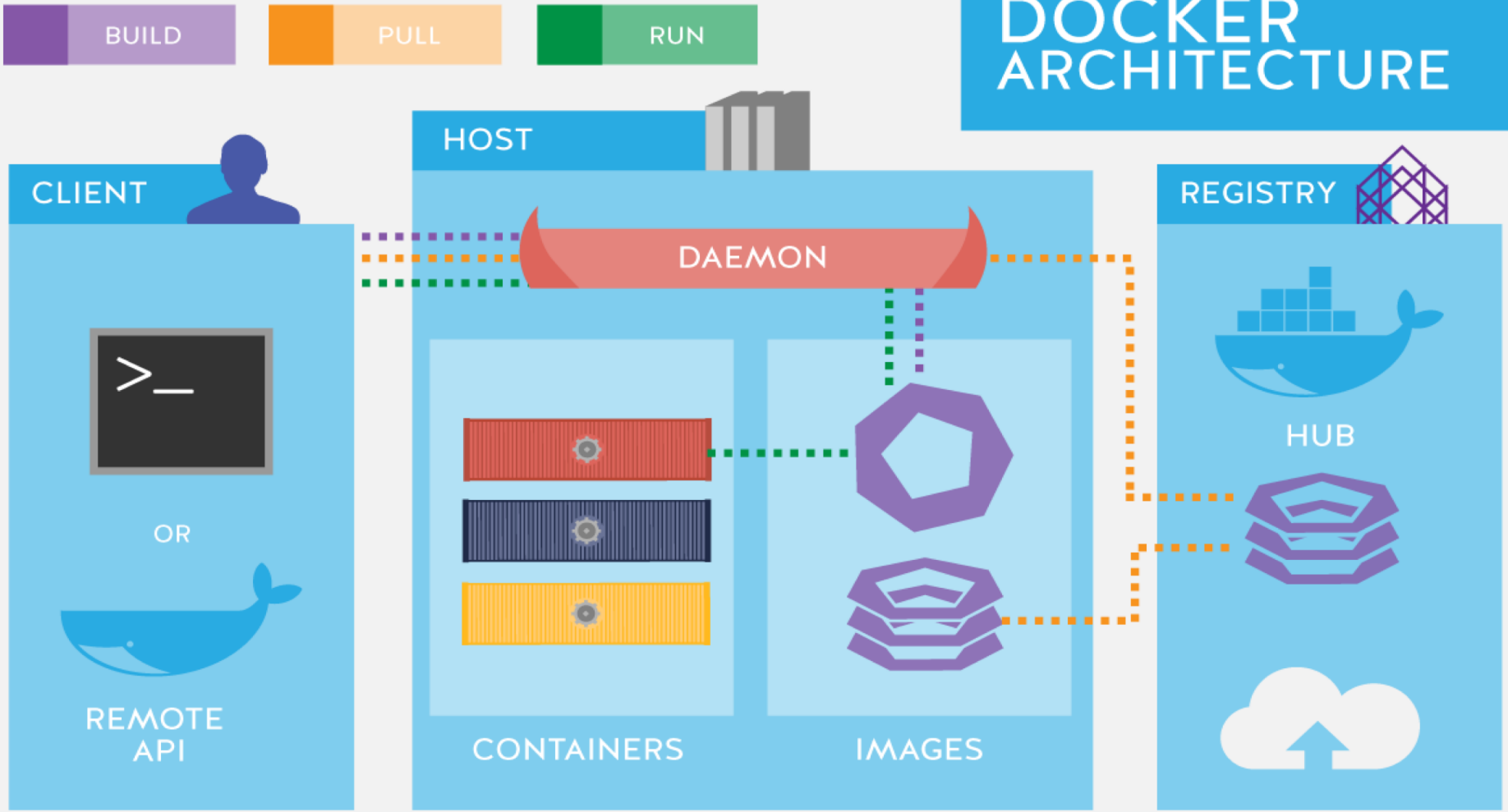
It can increase and automatically distribute the container replication

Docker Architecture

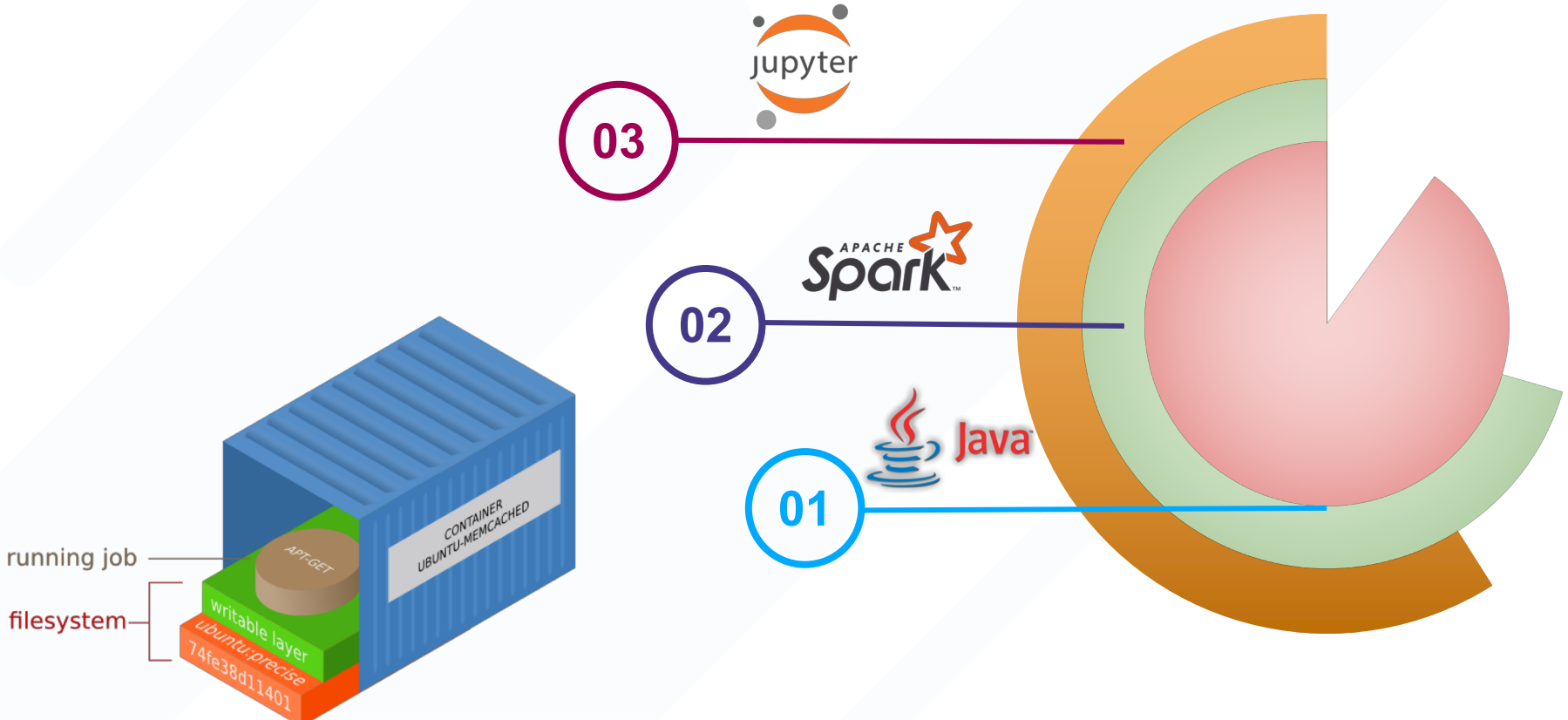
It uses a client-server architecture, where the client communicates with the docker daemon in order to execute, build and distribute the containers.

The communication between the client and the daemon is through an API-Rest. They use sockets UNIX, if they are running in the same system or TCP when a client talk with a remote daemon.

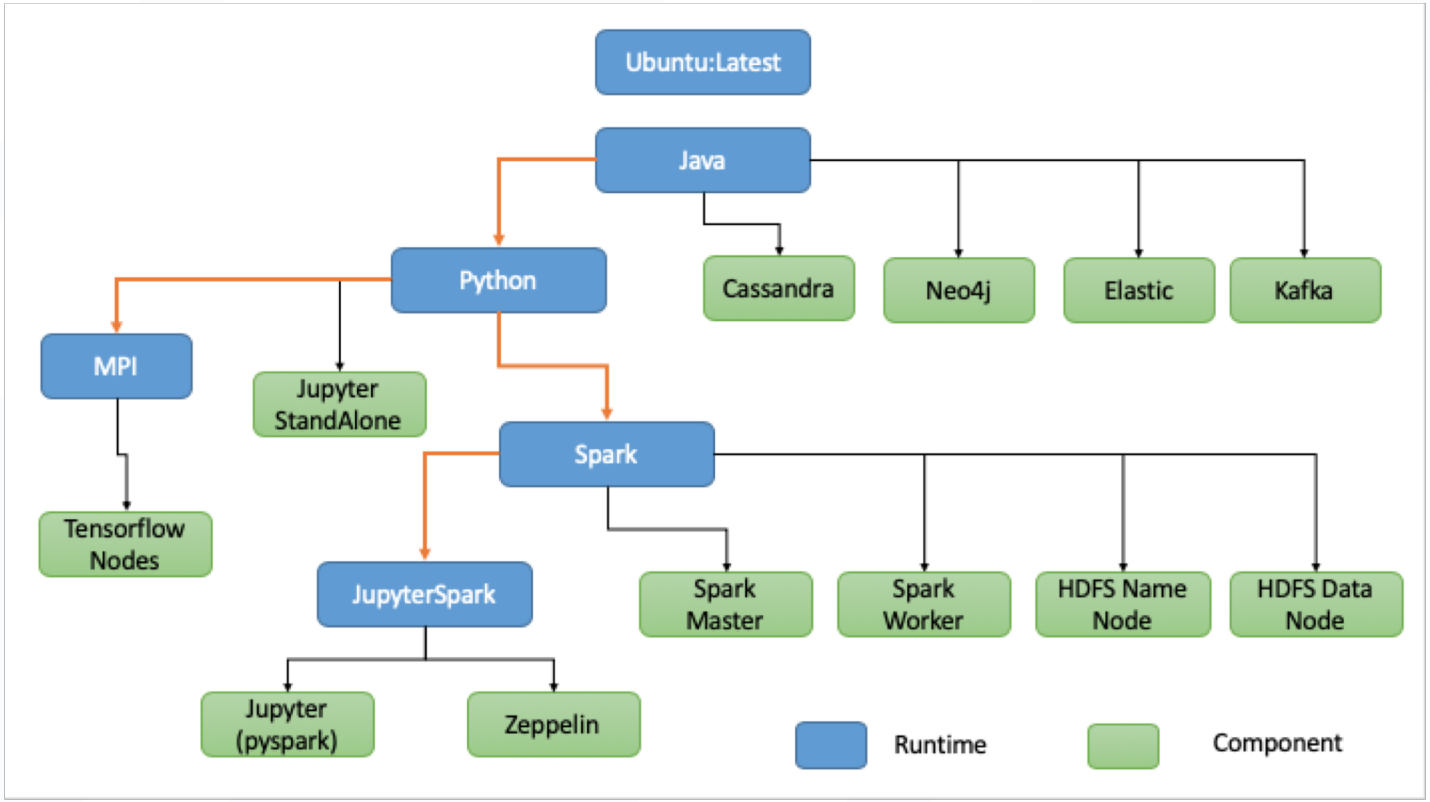
DOCKER ARCHITECTURE



Docker image (Runtime for BDaaS)



Docker image (Runtime for BDaaS)



Docker image (Runtime for BDaaS)

```
rmuresano@rmuresano-pc:~/git$ docker images
REPOSITORY                                TAG                 IMAGE ID            CREATED             SIZE
rmuresano/bdaasjupyter                    0_0_1               2ac679c04de1        3 weeks ago         2.73GB
rmuresano/bdaasspark                      0_0_1               f301f6a0a6a5        3 weeks ago         1.54GB
rmuresano/bdaaspython                     0_0_1               7947fff07802        3 weeks ago         1.02GB
rmuresano/bdaasjava                       0_0_1               20ec23cf25bf        3 weeks ago         588MB
radiatus.iti.es/runtime/mpi               0_0_25              80958b89f09b        4 weeks ago         3.83GB
radiatus.iti.es/runtime/jupyterspark      0_0_25              fe8e0c03074a        4 weeks ago         3.11GB
radiatus.iti.es/runtime/spark             0_0_25              affb73b17000        4 weeks ago         1.45GB
radiatus.iti.es/runtime/hadoop            0_0_25              8ab082177c47        4 weeks ago         1.28GB
radiatus.iti.es/runtime/python            0_0_25              7a35cb9205c2        4 weeks ago         998MB
radiatus.iti.es/runtime/mysql             0_0_25              a63fadab2f60        4 weeks ago         939MB
radiatus.iti.es/runtime/basic             0_0_25              d7e8b05df2d2        4 weeks ago         423MB
ubuntu                                    latest              7698f282e524        7 weeks ago         69.9MB
nginx                                     latest              53f3fd8007f7        2 months ago        109MB
node                                      lts-slim            8bfde27ed7db        2 months ago        143MB
saasdk/local-stamp                        latest              4dec561fdf0f        4 months ago        848MB
dockersamples/visualizer                  latest              f6411ebd974c        6 months ago        166MB
saasdk/forge.domains                      latest              7dc75f09bb01        7 months ago        206MB
registry                                  2                   2e2f252f3c88        9 months ago        33.3MB
saasdk/eslap.cloud_runtime_native_dev     2_0_0               383c9709f1ea        17 months ago       525MB
saasdk/eslap.cloud_runtime_native         2_0_0               71cea2ae9bbc        17 months ago       216MB
saasdk/eslap.cloud_elk                    1_0_0               d6c67b531589        2 years ago         594MB
eslap.cloud/elk                           1_0_0               d6c67b531589        2 years ago         594MB
rmuresano@rmuresano-pc:~/git$ ++
```

Docker image (Runtime for BDaaS)

Example 1:

```
rmuresano@rmuresano-pc:~/git$ docker run --rm -ti debian
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
6f2f362378c5: Pull complete
Digest: sha256:118cf8f3557e1ea766c02f36f05f6ac3e63628427ea8965fb861be904ec35a6f
Status: Downloaded newer image for debian:latest
root@c056b43ddffa:/#
```

```
root@c056b43ddffa:/# cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
root@c056b43ddffa:/#
```

Docker image (Runtime for BDaaS)

```
debian                                  latest              e1de74e67cc7        3 weeks ago         101MB
```

A complete documentation can be found:
https://docs.docker.com/engine/reference/commandline/cli/

Docker management

Docker run: Executes an order to run a new container.

Docker exec: Invokes an order in a running container.

Docker ps: Lists containers

Docker rm: Deletes container with an specific tag

Docker rmi: Deletes images. Sometimes must be forced.

Docker images: List Images (For us Runtimes)

Docker build: Build an image from a DockerFile

Docker Registry management

Docker tag: It control and manage the image tagging
Docker login: Identify the in the Docker registry
Docker pull and push: Push and Pull an image to/from the registry
Docker search: Search an image in the Docker registry

Docker port: List the associated ports to an specific container
Docker network: Manages the network interconnection between containers
Docker volume: Shows the volume information
Docker inspect: Returns the low level information about a docker container.

Docker image (Runtime for BDaaS)

```
rmuresano@rmuresano-pc:~/git$ docker search elasticsearch
NAME                                   DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
elasticsearch                          Elasticsearch is a powerful open source se...   3736                [OK]
nshou/elasticsearch-kibana             Elasticsearch-7.1.1 Kibana-7.1.1                102                                     [OK]
itzg/elasticsearch                     Provides an easily configurable Elasticsea...   67                                      [OK]
mobz/elasticsearch-head                elasticsearch-head front-end and standalon...   46
elastichq/elasticsearch-hq             Official Docker image for ElasticHQ: Elast...   31                                      [OK]
lmenezes/elasticsearch-kopf            elasticsearch kopf                              18                                      [OK]
bitnami/elasticsearch                  Bitnami Docker Image for Elasticsearch          18                                      [OK]
elastic/elasticsearch                  The Elasticsearch Docker image maintained ...   17
barnybug/elasticsearch                 Latest Elasticsearch 1.7.2 and previous re...   17                                      [OK]
taskrabbit/elasticsearch-dump          Import and export tools for elasticsearch       16                                      [OK]
esystemstech/elasticsearch             Debian based Elasticsearch packing for Lif...   15
monsantoco/elasticsearch               ElasticSearch Docker image                      11                                      [OK]
justwatch/elasticsearch_exporter       Elasticsearch stats exporter for Prometheus     9
mesoscloud/elasticsearch               [UNMAINTAINED] Elasticsearch                    9                                       [OK]
blacktop/elasticsearch                 Alpine Linux based Elasticsearch Docker Image   8                                       [OK]
centerforopenscience/elasticsearch     Elasticsearch                                   4                                       [OK]
barchart/elasticsearch-aws             Elasticsearch AWS node                          3
phenompeople/elasticsearch             Elasticsearch is a powerful open source se...   1                                       [OK]
bitnami/elasticsearch-exporter         Bitnami Elasticsearch Exporter Docker Image     1                                       [OK]
jetstack/elasticsearch-pet             An elasticsearch image for kubernetes PetSets   1                                       [OK]
18fgsa/elasticsearch-ha                Built from https://github.com/18F/kubernet...   0
18fgsa/elasticsearch                   Built from https://github.com/docker-libra...   0
wreulicke/elasticsearch                elasticsearch                                   0                                       [OK]
backplane/elasticsearch-curator        Elasticsearch Curator (https://github.com/...   0
axway/elasticsearch-docker-beat        "Beat" extension to read logs of container...   0                                       [OK]
```

Docker Management (Runtime for BDaaS)

Example 2:

Deployment Information

```
rmuresano@rmuresano-pc:~/git$ docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS
          NAMES
972e8f4c42fc        rmuresano/bdaasjupyter:0_0_4         "/bin/sh -c ./star..."   6 seconds ago       Up 5 seconds        0.0.0.0:8900->8900/tcp
          jupyter
2732b95e72f7        rmuresano/bdaassparkworker:0_0_4     "/bin/sh -c ./star..."   6 seconds ago       Up 5 seconds        0.0.0.0:8081->8081/tcp
          jupyterspark_sparkworker_1
ad0f7269fd06        rmuresano/bdaassparkmaster:0_0_4     "/bin/sh -c ./star..."   6 seconds ago       Up 6 seconds        0.0.0.0:7077->7077/tcp, 0.0.0.0:8080->8080/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:50070->
50070/tcp   sparkmaster
```

Port

```
rmuresano@rmuresano-pc:~/git$ docker port 972e8f4c42fc
8900/tcp -> 0.0.0.0:8900
```

Volume Information

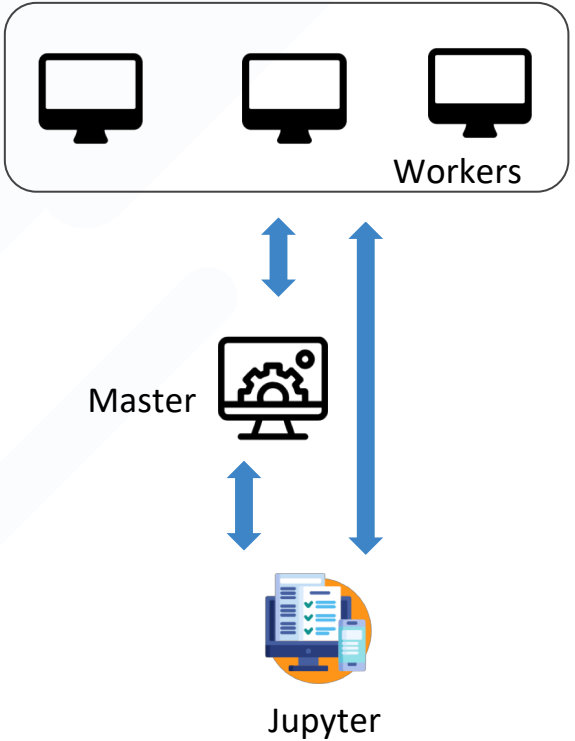
```
rmuresano@rmuresano-pc:~/git$ docker volume ls
DRIVER              VOLUME NAME
local               jupyterspark_hdfs-name-node
local               jupyterspark_jupyter-vol
```

```
rmuresano@rmuresano-pc:~/git$ docker volume inspect jupyterspark_jupyter-vol
[
    {
        "CreatedAt": "2019-06-05T15:32:58+02:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/jupyterspark_jupyter-vol/_data",
        "Name": "jupyterspark_jupyter-vol",
        "Options": {},
        "Scope": "local"
    }
]
```

Docker Management (Runtime for BDaaS)

```
rmuresano@rmuresano-pc:~$ docker network inspect 9b0d03c3bfaf
[
    {
        "Name": "jupyterspark_default",
        "Id": "9b0d03c3bfaf58593965866abdcd0827c172f56e536facc2bab4d68a30b36982",
        "Created": "2019-06-05T14:39:05.36512168+02:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "2732b95e72f7b96afbbb848576b7d325daee2205394ae87af515989f6bc53b92": {
                "Name": "jupyterspark_sparkworker_1",
                "EndpointID": "1fc590e352834f4325fb0a947628ec4cbadce45abe361eb58ff62a95eb540eb2",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "972e8f4c42fc34f0e386ebe00d9f71950203bde1be9b6c49f51ebc695871eca1": {
                "Name": "jupyter",
                "EndpointID": "c2201cfbfb198be67296c6232aef29b6e6d0ef7aafd67121b12b98e9123d345b",
                "MacAddress": "02:42:ac:12:00:04",
                "IPv4Address": "172.18.0.4/16",
                "IPv6Address": ""
            },
            "ad0f7269fd06e203ed96de519881ef8eae9bc1c1654cf8124068c7b71daf2295": {
                "Name": "sparkmaster",
                "EndpointID": "2645656d33b647b347965e85403e0b530054347b55ea9127d4a10826d3750415",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "default",
            "com.docker.compose.project": "jupyterspark"
        }
    }
]
```

Container network specification



Docker Runtime building for BDaaS (DockerFile)

- To generate an image, we have to use the instruction docker build.
- The Dockerfile is a text document which contains a set of instructions in order to assemble an image.
- The context is the set of files that are in an specific route (local directory) o an URL (git repository reference)
- The context is processed in a recursive manner. This means if a directory is included all sub-directory are also included.

Docker Runtime building for BDaaS (DockerFile)

- Like a Makefile (shell script with keywords)
- Extends from a Base Image
- Results in a new Docker Image
- Imperative, not Declarative
- Docker file lists the steps needed to build an images
- docker build is used to run a Docker file
- It can define default command for docker run, ports to expose, etc.

Creating a Runtime from a DockerFile (1/2)

Example 3:

Using the following invocation the docker client create a new tagged image with the name defined. For example, rmuresano/<name-of-the-image> and the version 0_0_1.

It is done passing the working directory as the current context and using the DockerFile.

```
rmuresano@rmuresano-pc:~/git/bdaasmicroservices/docker/runtimes/helloExample$ docker build -t my_docker_flask:latest .
Sending build context to Docker daemon   5.12kB
Step 1/6 : FROM python
 ---> 34a518642c76
Step 2/6 : COPY . /app
 ---> cc32ab3aca88
Step 3/6 : WORKDIR /app
 ---> cfc472c244b1
Removing intermediate container d9bb2d8d7e92
Step 4/6 : RUN pip install -r requirements.txt
 ---> Running in c8960b77caa7
Collecting flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/c3/31/6904ac846fc65a7fa6cac8b4ddc392ce96ca08ee67b0f97854e9575bbb26/Flask-1.1.0-py2.py3-none-any.whl (94kB)
Collecting flask_restful (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/17/44/6e490150ee443ca81d5f88b61bb4bbb133d44d75b0b716ebe92489508da4/Flask_RESTful-0.3.7-py2.py3-none-any.whl
Collecting Werkzeug>=0.15 (from flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/9f/57/92a497e38161ce40606c27a86759c6b92dd34fcdb33f64171ec559257c02/Werkzeug-0.15.4-py2.py3-none-any.whl (327kB)
Collecting click>=5.1 (from flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl (81kB)
Collecting Jinja2>=2.10.1 (from flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/1d/e7/fd8b501e7a6dfe492a433deb7b9d833d39ca74916fa8bc63dd1a4947a671/Jinja2-2.10.1-py2.py3-none-any.whl (124kB)
Collecting itsdangerous>=0.24 (from flask->-r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting pytz (from flask_restful->-r requirements.txt (line 2))
```

Creating a Runtime from a DockerFile (2/2)

Dockerfile

```python
# app.py - a minimal flask api using flask_restful
from flask import Flask
from flask_restful import Resource, Api
app = Flask(__name__)
api = Api(app)
You, 5 minutes ago | 1 author (You)
class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}


api.add_resource(HelloWorld, '/')
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')      You, 5 minut
```

Dockerfile

```
You, a few seconds ago | 1 author (You)
# Dockerfile - this is a comment. Delete me if you want.
FROM python
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENTRYPOINT ["python"]
CMD ["app.py"]
```

Docker run

```
rmuresano@rmuresano-pc:~/git/bdaasmicroservices/docker/runtimes/helloExample$ docker run --name ronal -d -p 5000:5000 my_docker_flask:latest
eb03c5f48fc4a676dfbc94ed98c30f2593feee515449b11de09dc82c100db534
rmuresano@rmuresano-pc:~/git/bdaasmicroservices/docker/runtimes/helloExample$ docker ps
CONTAINER ID        IMAGE                  COMMAND             CREATED             STATUS              PORTS                    NAMES
eb03c5f48fc4        my_docker_flask:latest   "python app.py"     3 seconds ago       Up 2 seconds        0.0.0.0:5000->5000/tcp   ronal
```

DockerFile Instruction

FROM: Normally it is the first instruction in the Dockerfile, and it identifies from which image we will start to build the image.

```
1  FROM <image> [AS <name>]
2  FROM <image>[:<tag>] [AS <name>]
3  FROM <image>[@<id-image>] [AS <name>]
```

ENV: It is used to declare variables which will be assigned in the building process. It can be used in the following instructions: ADD, COPY, ENV, EXPOSE, FROM, LABEL, USER, VOLUME, WORDIR, etc.

```
1  ENV VAR VALUE
2  ENV VAR=VALUE [VAR2=VALUE]
```

ARGS: available during the build of a Docker image (RUN etc), not after the image is created and containers are started from it.

```
1 ARG VERSION=latest
2 FROM busybox:$VERSION
3 ARG VERSION
4 RUN echo $VERSION > image_version
```

DockerFile Instruction

RUN: It executes one or more instruction in the container build. It creates a new layer in case that execution is successfully and this layer will be used by the next Dockerfile instruction.

```
1  FROM ubuntu
2  RUN instruction
3  RUN ["exec file", "param1", "param2", ...]
4  RUN apt-get -y update && apt-get install -y bzip2 build-essential libssl1.0.0 libpam0g openssl \
5      && apt-get install -y autotools-dev libssl1.0-dev libpam0g-dev zlib1g-dev debhelper dh-autoreconf \
6      && apt-get install -y openssh-server sshpass whois net-tools bc  nano && \
7      rm -rf /var/lib/apt/lists/*
```

EXPOSE: It is used to inform in which port the container will listen while it is executing. By default, it is TCP, but it can be defined the TCP o UDP

```
9   EXPOSE 8888
10  EXPOSE 8088/UDP
```

DockerFile Instruction

ENTRYPOINT: It is used to set up a container as an executable. The instruction indicates which program must be started when the container is started. In case that this instruction appears many times it will be considered the last defined in the Dockerfile.

```
ENTRYPOINT ./start-up-config.sh param1 param2
```

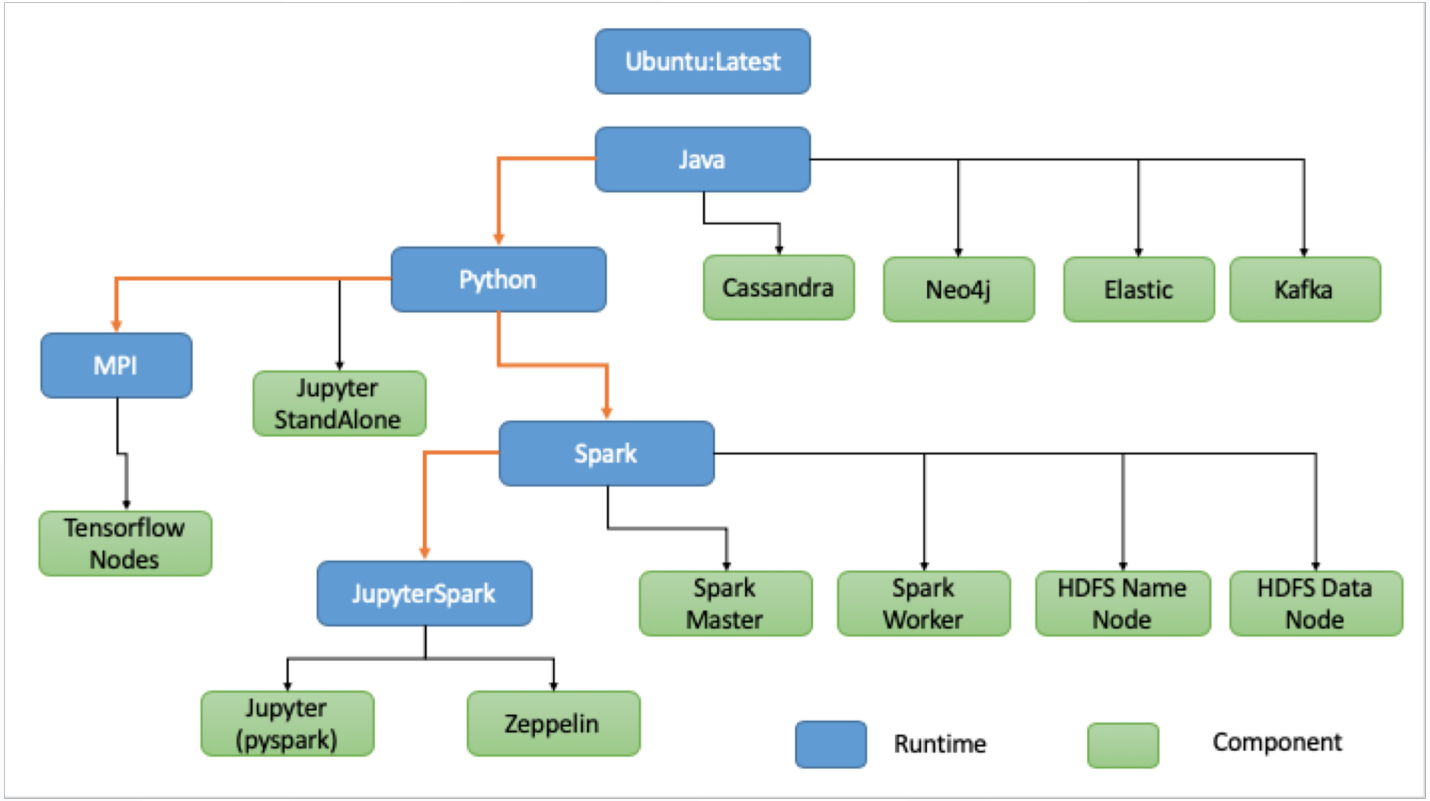
USER: This instruction establishes the user name (UID) and the group (GID) which must be used when the container processes the following instruction RUN, CMD, ENTRYPOINT.

```
RUN cd /src; npm install

RUN useradd -c 'Node.js user' -m -d /home/node -s /bin/bash node
RUN chown -R node.node /src
USER node
ENV HOME /home/node

EXPOSE  8080
CMD ["node", "/src/index.js"]
```

DockerFile Runtime Image Tree



DockerFile Creating the first image (Java)

```
1   FROM ubuntu:latest
2   WORKDIR /home/legacy
3   COPY packages/ ./
4
5   RUN apt-get -y update
6   RUN apt-get install -y bzip2
7   RUN apt-get install -y build-essential
8   RUN apt-get install -y libssl1.0.0
9   RUN apt-get install -y libpam0g
10  RUN apt-get install -y openssl
11  RUN apt-get install -y autotools-dev libssl1.0-dev libpam0g-dev \
12      zlib1g-dev debhelper dh-autoreconf
13  RUN apt-get -y update && \
14      apt-get install -y openssh-server sshpass whois net-tools bc  nano && \
15      rm -rf /var/lib/apt/lists/*
16
17  WORKDIR /home/legacy/shellinabox
18  RUN mkdir -p shellinaboxBin && autoreconf -i  \
19      && ./configure --prefix=/home/legacy/shellinabox/shellinaboxBin/ \
20      && make && make install
21        You, a few seconds ago • Uncommitted changes
22  ENV PATH="$PATH:/home/legacy/jre/bin"
23
24  WORKDIR /home/legacy
25
```

DockerFile Creating the first image (Java)

Example 4: Optimizing the image

[Important] To use a tool for exploring a docker image, layer contents, and discovering ways to shrink your Docker image size.

In our case, we will use Docker Dive: https://github.com/wagoodman/dive

Installation Procedure:

Ubuntu/Debian

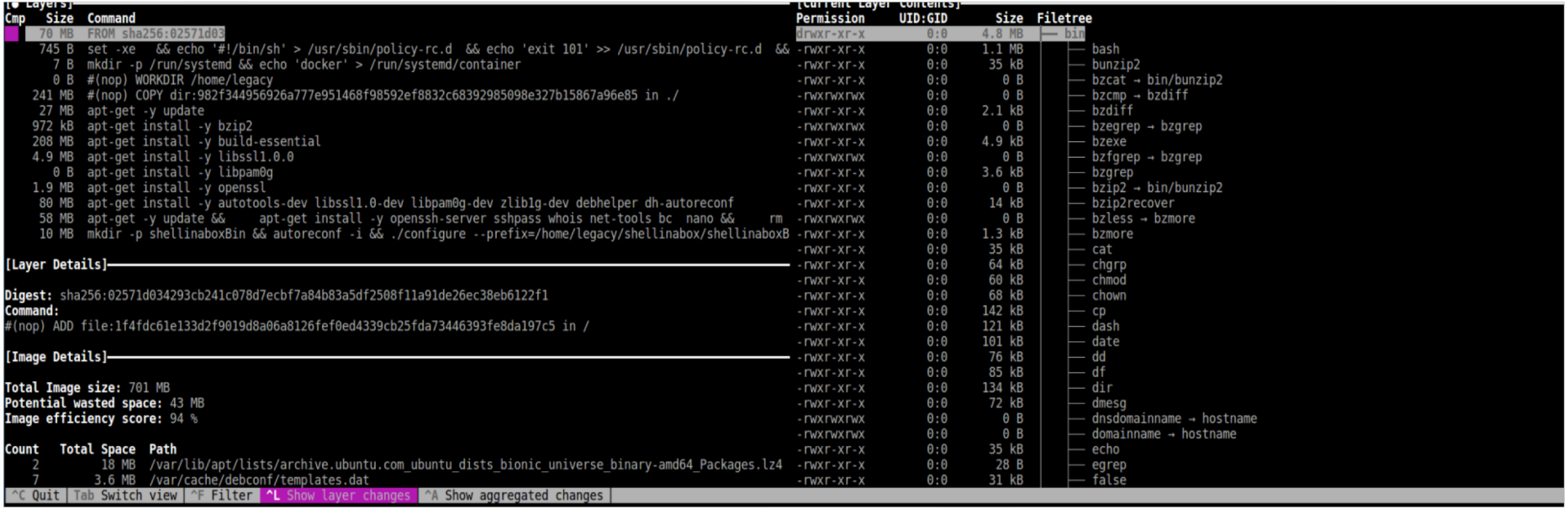
```
wget https://github.com/wagoodman/dive/releases/download/v0.7.2/dive_0.7.2_linux_amd64.deb
sudo apt install ./dive_0.7.2_linux_amd64.deb
```

Mac

```
brew tap wagoodman/dive
brew install dive
```

DockerFile Creating the first image (Java)

Analyzing the image with dive tool



DockerFile Creating the first image (Java)

Example 4: Optimizing the image

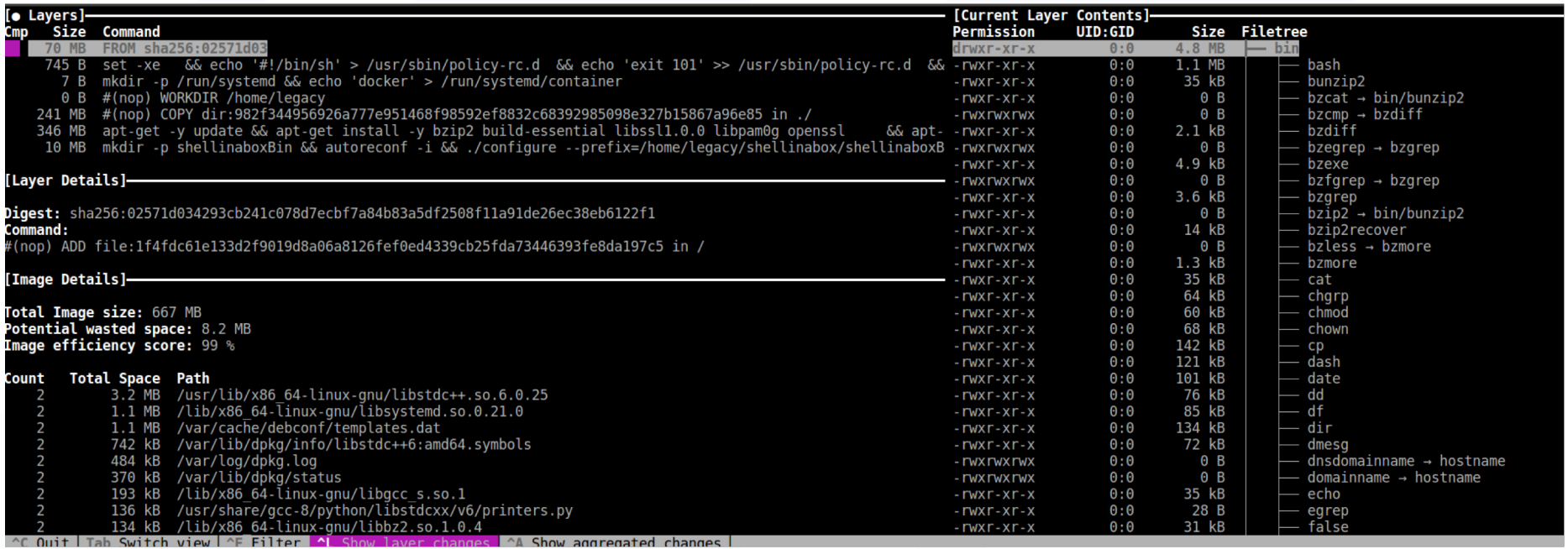
```
Muresano, a day ago | 1 author (Muresano)
1  FROM ubuntu:latest
2  WORKDIR /home/legacy
3  COPY packages/ ./
4  RUN apt-get -y update && apt-get install -y bzip2 build-essential libssl1.0.0 libpam0g openssl \
5      && apt-get install -y autotools-dev libssl1.0-dev libpam0g-dev zlib1g-dev debhelper dh-autoreconf \
6      && apt-get install -y openssh-server sshpass whois net-tools bc  nano && \
7      rm -rf /var/lib/apt/lists/*
8
9  WORKDIR /home/legacy/shellinabox
10 RUN mkdir -p shellinaboxBin && autoreconf -i && ./configure --prefix=/home/legacy/shellinabox/shellinaboxBin/ && make && make install
11 ENV PATH="$PATH:/home/legacy/jre/bin"
12
```

Improvement in size around 4.85%

```
rmuresano@rmuresano-pc:~/git/bdaasmicroservices/docker/runtimes/java-optimization-1$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
rmuresano/bdaasjava         0_0_3               edfb976e9539        52 seconds ago      667MB
rmuresano/bdaasjava         0_0_2               01c63bc738a0        8 minutes ago       701MB
```

DockerFile Creating the first image (Java)

Analyzing the optimized image with dive tool



DockerFile Creating the first image (Java)

To avoid using different Dockerfiles, it was add the functionality to define diverse steps in the same file. Each step start with the instruction from.

Once a step is built, we can copy files from other layers using the **COPY** command and the parameter flag --from

Example:

Improvement in size around 54 %


```
Muresano, a day ago | 1 author (Muresano)
1  FROM ubuntu:latest as builder
2  WORKDIR /home/legacy
3  COPY packages/ ./
4  RUN apt-get -y up
5      && apt-get in
6      && apt-get in
7      rm -rf /var/l
8  WORKDIR /home/leg
9  RUN mkdir -p shel
10
11 FROM ubuntu:lates
12 WORKDIR /home/leg
13 COPY --from=build
14 COPY --from=build
15 ENV PATH="$PATH:/
16
17
```

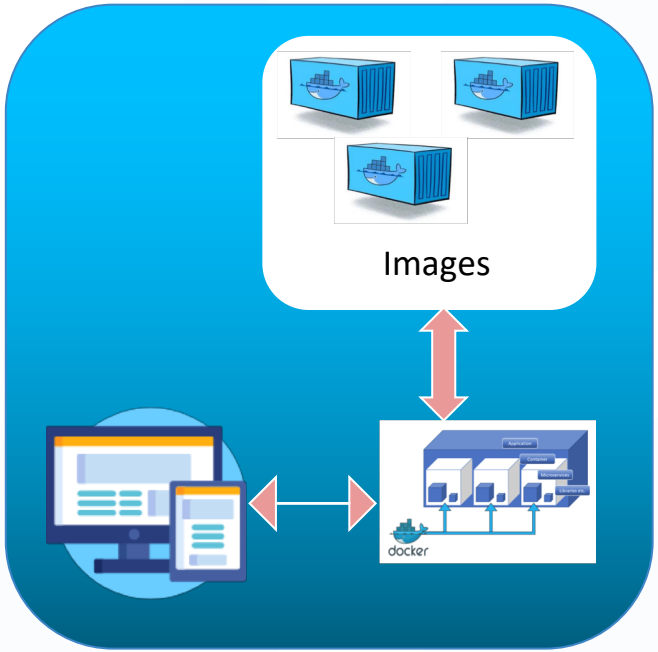
```
MacBook-Pro-de-Ronal:radiatus3 rmuresano$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
rmuresano/bdaascassandra            0_0_4               23685696b08f        23 hours ago
rmuresano/bdaasneo4j                0_0_4               ed8b92379679        25 hours ago
rmuresano/bdaassparkworker          0_0_4               d1ff45a0f6ae        29 hours ago
rmuresano/bdaassparkmaster          0_0_4               16f68f5371c0        29 hours ago
rmuresano/bdaasjupyter              0_0_4               5d359d0e0935        29 hours ago
rmuresano/bdaasspark                0_0_4               3cfdece2be29        30 hours ago        1.24GB
rmuresano/bdaaspython               0_0_4               151df2f5b132        30 hours ago        720MB
rmuresano/bdaasjava                 0_0_4               fa78e638247f        30 hours ago        321MB
rmuresano/bdaasjava                 0_0_3               cf4cb6d87e93        30 hours ago        667MB
rmuresano/bdaasjava                 0_0_2               87ffd9b93379        30 hours ago        701MB
ubuntu                              latest              7698f282e524        7 weeks ago         69.9MB
jupyter/datascience-notebook        latest              029fd3e52059        8 weeks ago         5.49GB
```

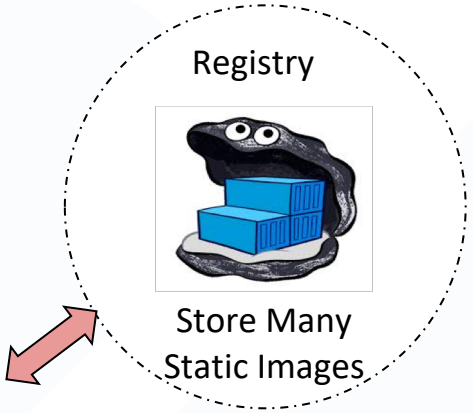
DockerFile Creating the Second image (Python)

Exercise 1: Creating the first image. Time Approximate (25~30 Min)

Steps:
1: Download the miniconda packages https://docs.conda.io/en/latest/miniconda.html
2: Create an image from the optimized image rmuresano/bdaasjava:0_0_4
3: Copy the installer to the image
4: Install the
4: Update the conda and pip packages
5: Remove all additional packages
6: Establish and ENV variable with the miniconda/bin Path.

Docker Registry (Runtime for BDaaS)





On-premise

- Docker Trusted Registry

Public Clouds

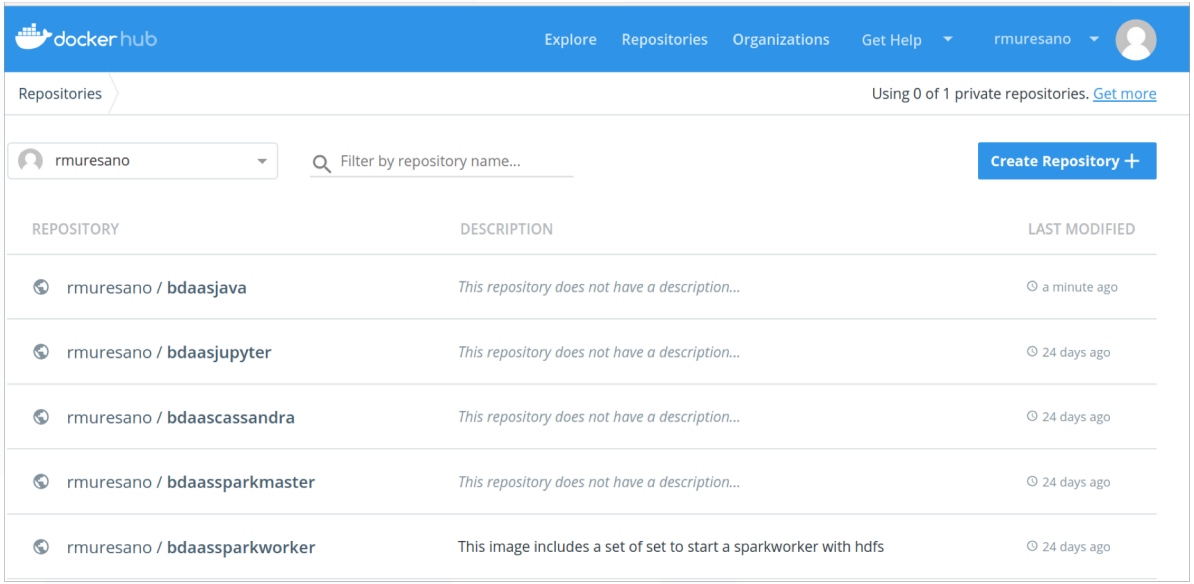
- Docker Hub
- Docker Trusted on Cloud
- Azure container Registry
- AWS container Registry (ECS -ECR)
- Google Container Registry.
- Other Cloud

Docker Hub Registry (Creating an account)

Exercise 2: https://hub.docker.com/ Time 5~10 minutes

Push image to docker Hub

```
rmuresano@rmuresano-pc:~/git/bdaasmicroservices/docker/runtimes/java-optimization-2$ docker push rmuresano/bdaasjava:0_0_4
The push refers to a repository [docker.io/rmuresano/bdaasjava]
7f63c72ccd1b: Pushed
07da8bc77580: Pushing [==================================================>]  239.3MB
cd9a495985a2: Pushed
8d267010480f: Pushed
270f934787ed: Pushed
02571d034293: Pushed
```



Deploy a registry server

1) Use a command like the following to start the registry container:

```
$ docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

2) Pull the ubuntu:16.04 image from Docker Hub.

```
$ docker pull ubuntu:16.04
```

3) Tag the image as localhost:5000/my-ubuntu

```
$ docker tag ubuntu:16.04 localhost:5000/my-ubuntu
```

4) Tag the image as localhost:5000/my-ubuntu

```
$ docker push localhost:5000/my-ubuntu
```

Compile Components and Runtimes tree

Exercise 3: 15~20 minutes to create all environment

Objective: Learn the procedure to compile and create big data components.

Steps:

Runtime

1) Git clone git@github.com:mureron/bdaasmicroservices.git bdaas.
2) Enter in the docker/runtime/packages directory.
3) Run the script ./DownloadFiles.sh (wait until all software and dependencies are downloaded.
4) Exit the docker/runtime/packages directory and enter the docker/runtime
5) Execute the script build-runtime.sh. (Take a coffee)

Components

1) Enter in the folder docker/components
2) Execute the script ./compileComponents.sh
3) Now we have all the runtimes and components done :)

Course Roadmap

Docker Swarm

Orchestration in a real private cluster (Deploying and Creating services)

Orchestration for developing

Docker and Docker-Compose.

Kubernetes

Production Orchestration. Automating deployment, scaling, and management of containerized applications

Big Data Challenges

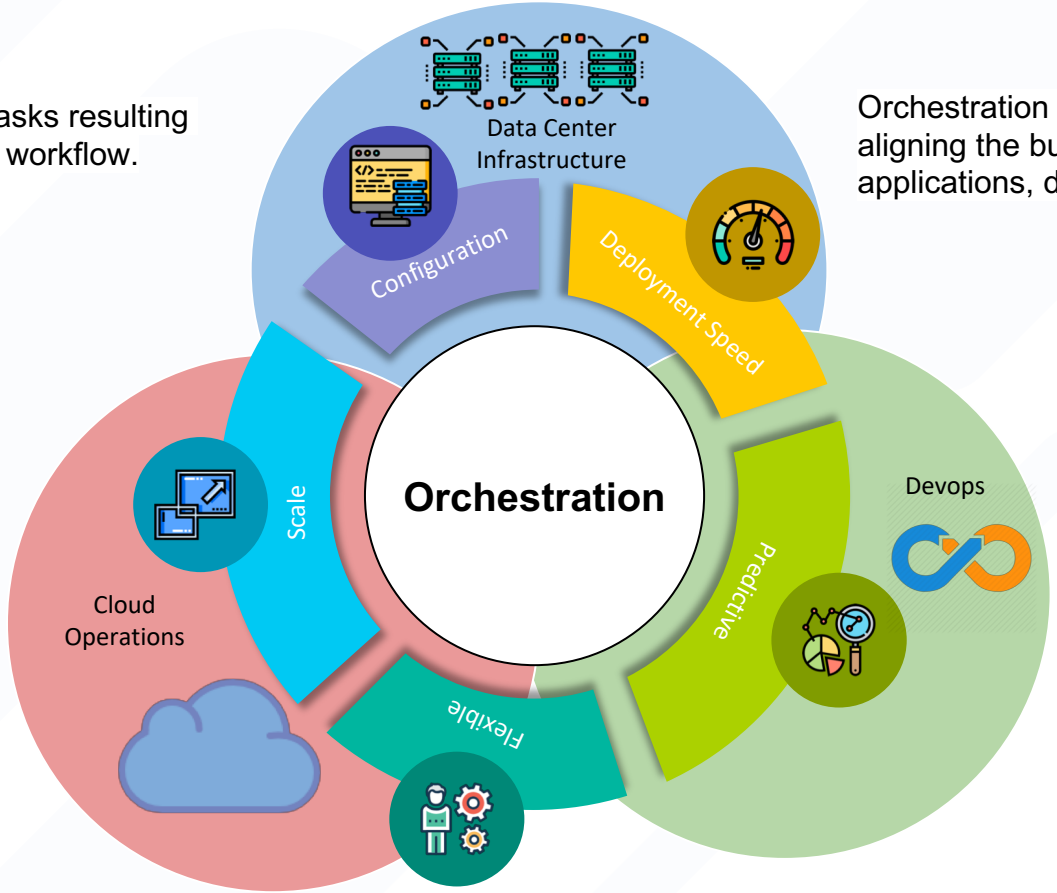
An overview applied to the SMEs. Orchestration Philosophy

Cloud Infraestructure

Google Cloud and AWS deployments

Service Orchestration

Coordination of automated tasks resulting in a consolidated process or workflow.



Orchestration in this sense is about aligning the business request with the applications, data, and infrastructure.

Docker-Compose (Multi-container Services)

Typical applications have multiple components. For example, Web App.

Docker Compose is a tool for defining and running multi-container Docker application.

with a single command, you create and start all the services from your configuration.

Compose works in all environments: production, staging, development, testing, as well as CI workflows

Using Compose is basically a three-step process:

1. Define your app's environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run docker-compose up and Compose starts and runs your entire app.

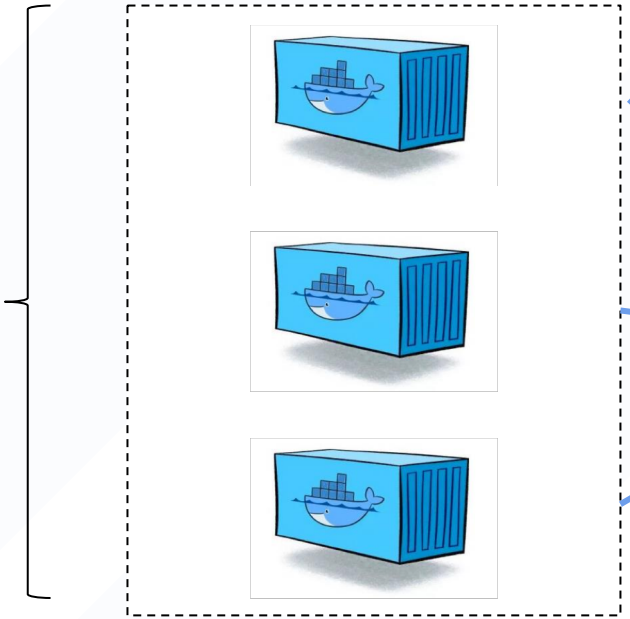
source:https://docs.docker.com/compose/overview/

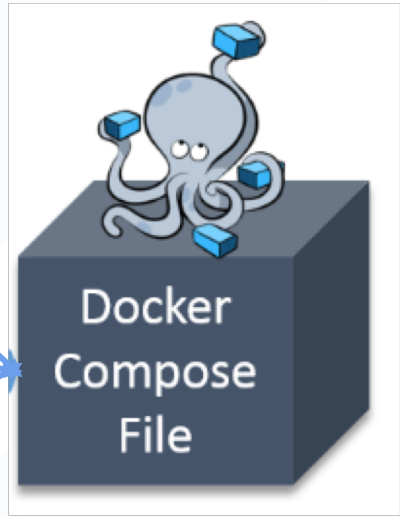
Docker-Compose (Multi-container Services)

A first Step from the development to a production environment.

The main idea is to help you to define and build application stacks.

Containers





Docker Compose configuration file allows to define: environment variables, networks or volumes

Docker-Compose (Multi-container Services)

docker-compose.yaml ▸ {} volumes ▸ jupyter-vol

```yaml
version: '3'
services:
  jupyter:
    image: rmuresano/bdaasjupyter:0_0_4
    hostname: jupyter
    container_name: jupyter
    environment:
      - STANDALONE=YES
      - JUPYTERPORT=8900
      - HDFS=NO
    ports:
      - "8900:8900" # WEB interface JupyterLab
    volumes:
      - jupyter-vol:/data/jupyter/

volumes:
  jupyter-vol:
```

- The cluster specification is done in a YAML file, normally called docker-compose.yml

- It is started using

```
docker-compose up
```

- The service can be scaled using the option

```
docker-compose up --scale SERVICE=NUM
```

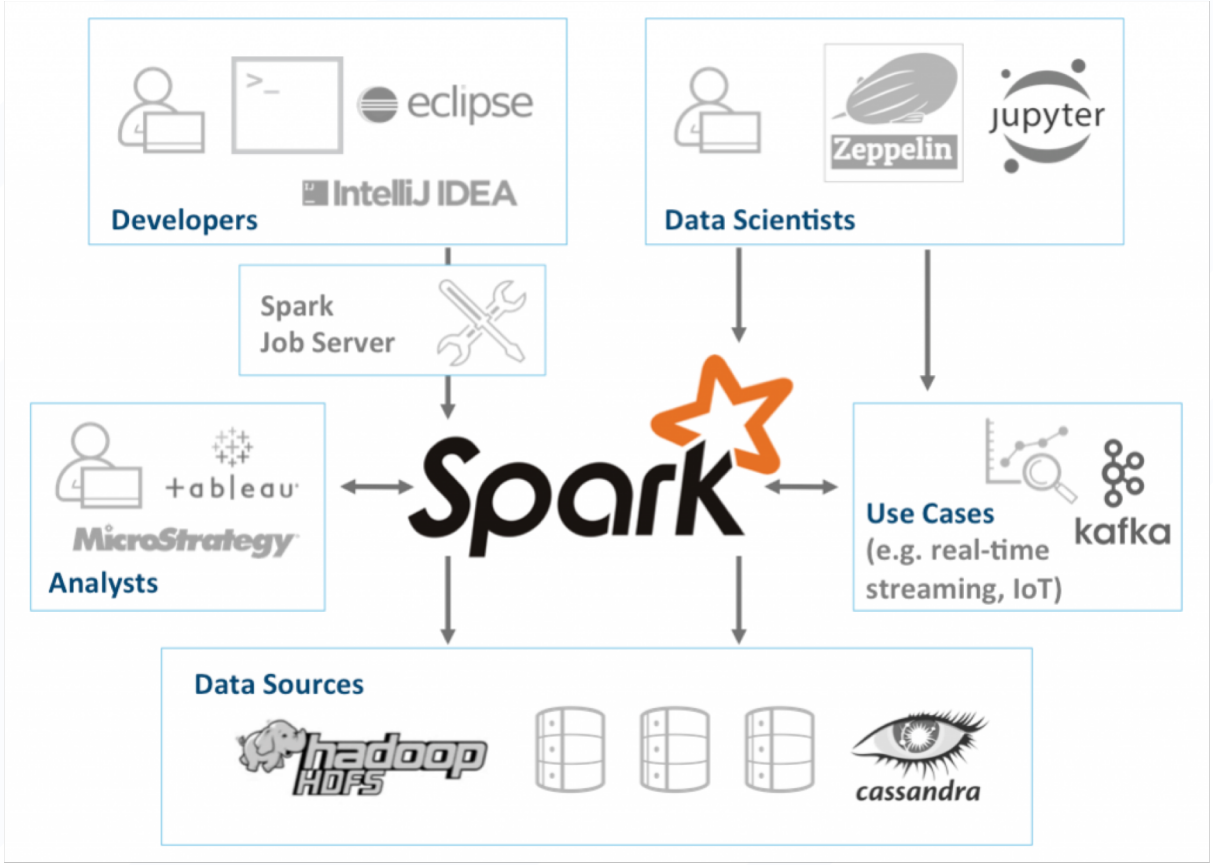
Demo

Running Docker-Compose (1/5)

Example 5: Start a batch service for Big Data (Jupyter-Spark)

```
MacBook-Pro-de-Ronal:jupyterspark rmuresano$ docker-compose up -d
Creating sparkmaster ... done
Creating jupyter                    ... done
Creating jupyterspark_sparkworker_1 ... done
MacBook-Pro-de-Ronal:jupyterspark rmuresano$ docker-compose ps
           Name                         Command               State                                        Ports
-------------------------------------------------------------------------------------------------------------------------------------------
jupyter                       /bin/sh -c ./startup-config.sh   Up      0.0.0.0:8900->8900/tcp
jupyterspark_sparkworker_1    /bin/sh -c ./startup-config.sh   Up      0.0.0.0:8087->8081/tcp
sparkmaster                   /bin/sh -c ./startup-config.sh   Up      0.0.0.0:50070->50070/tcp, 0.0.0.0:7077->7077/tcp,
                                                                       0.0.0.0:8080->8080/tcp, 0.0.0.0:9000->9000/tcp
```

Running Docker-Compose (2/5)

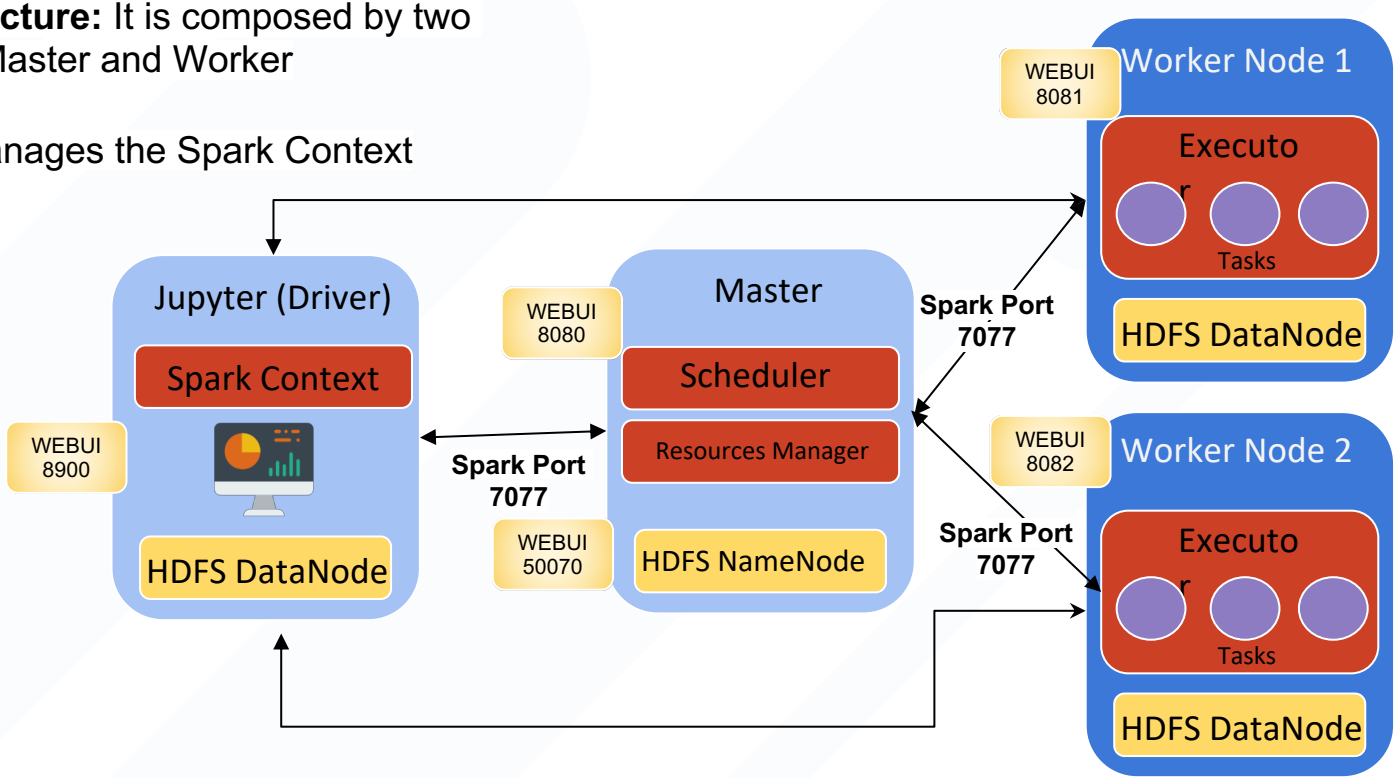


Running Docker-Compose (3/5)

Analyzing the architecture, we have to split into two parts (Jupyter, Spark and HDFS))

Spark architecture: It is composed by two components Master and Worker

Jupyter: It manages the Spark Context



Running Docker-Compose (4/5)

Jupyter

Environment:

STANDALONE=NO
JUPYTER PORT=8900
SPARK_MASTER_HOSTNAME=sparkmaster
SPARK_MASTER_PORT=7077
HDFS_NAMENODE_HOSTNAME=sparkmaster
HDFS_NAMENODE_METADATA_PORT=9000
HDFS_REPLICATION_FACTOR=1
HDFS_DATANODE_WEBUI_PORT=50075
SPARK_WORKER_WEBUI_PORT=8081
HDFS_NAMENODE_WEBUI_PORT=50070
HDFS=YES

Ports to Expose:
8900:8900

Volume:
jupyter-vol:/data/jupyter

Depends on:
Spark Master

Spark Master

Environment:

SPARK_MASTER_PORT=7077
SPARK_MASTER_WEBUI=8080
HDFS_CLUSTER_NAME=HDFS_TEST
HDFS_NAMENODE_METADATA_PORT=9000
HDFS_REPLICATION_FACTOR=1
HDFS_NAMENODE_WEBUI_PORT=50070
HDFS=YES

Ports to Expose:
8080:8080
7077:7077
9000:9000
50070:50070

Volume:
hdfs-name-node:/hdfs/namenode

Spark Worker

Environment:

SPARK_MASTER_HOSTNAME=sparkmaster
SPARK_MASTER_PORT=7077
HDFS_NAMENODE_HOSTNAME=sparkmaster
HDFS_NAMENODE_METADATA_PORT=9000
HDFS_REPLICATION_FACTOR=1
HDFS_DATANODE_WEBUI_PORT=50075
SPARK_WORKER_WEBUI_PORT=8081
HDFS_NAMENODE_WEBUI_PORT=50070
HDFS=YES

Ports to Expose:
8081-8090:8081

Volume:
hdfs-data-node:/hdfs/datanode

Depends on:
Spark Master

Running Docker-Compose (5/5)

Spark Master

```
# Set up NameNode and Spark Master
  sparkmaster:
    image: rmuresano/bdaassparkmaster:0_0_4
    hostname: sparkmaster
    container_name: sparkmaster
    environment:
      - SPARK_MASTER_PORT=7077
      - SPARK_MASTER_WEBUI_PORT=8080
      - HDFS_NAMENODE_METADATA_PORT=9000
      - HDFS_CLUSTER_NAME=HDFS_TEST
      - HDFS_NAMENODE_WEBUI_PORT=50070
      - HDFS_REPLICATION_FACTOR=1
      - HDFS=YES
    ports:
      - "8080:8080" #Web Interface Master
      - "7077:7077" #Master Port
      - "9000:9000" # HDFS PORT
      - "50070:50070" # WEB NameNode Int
    volumes:
      - hdfs-name-node:/hdfsdata/nameNode/
```

Spark Worker

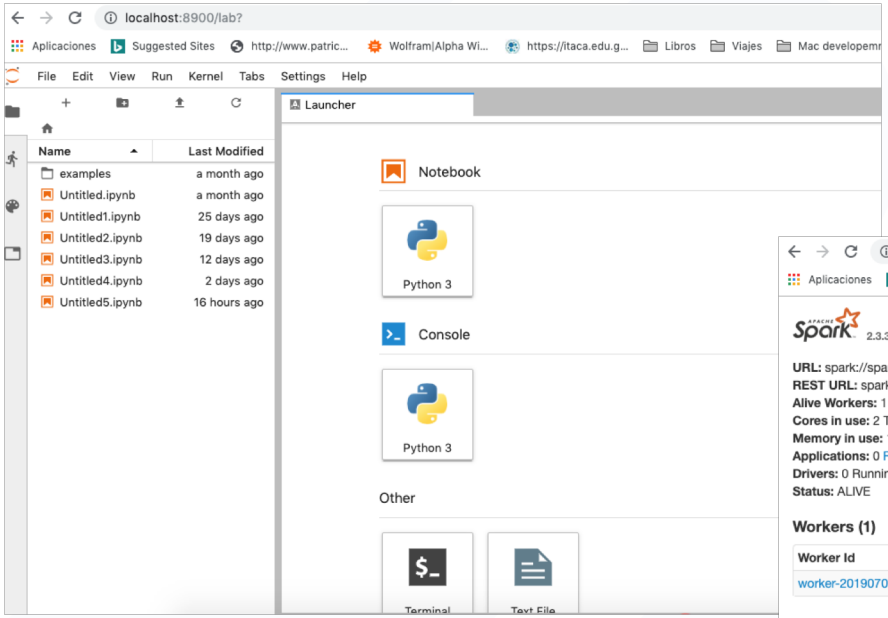
```
# Setup the Worker Node 1.
  sparkworker:
    image: rmuresano/bdaassparkworker:0_0_4
    hostname: sparkworker
    depends_on:
      - sparkmaster
    links:
      - sparkmaster
    environment:
      - SPARK_MASTER_HOSTNAME=sparkmaster
      - SPARK_MASTER_PORT=7077
      - HDFS_NAMENODE_HOSTNAME=sparkmaster
      - HDFS_NAMENODE_METADATA_PORT=9000
      - HDFS_REPLICATION_FACTOR=1
      - HDFS_DATANODE_WEBUI_PORT=50075
      - SPARK_WORKER_WEBUI_PORT=8081
      - HDFS_NAMENODE_WEBUI_PORT=50070
      - HDFS=YES
    ports:
      - "8081-8090:8081"
```

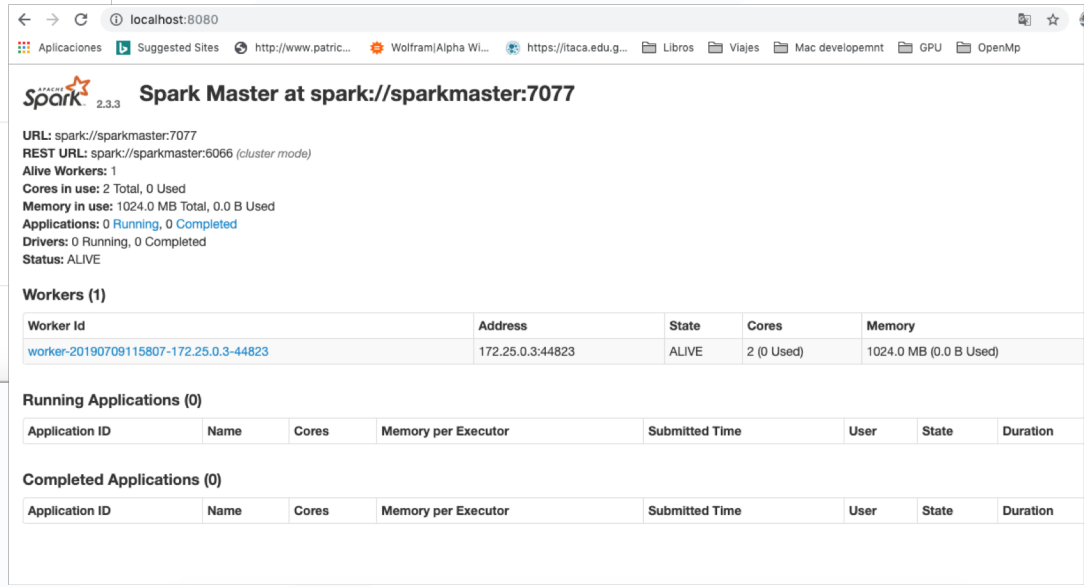
Jupyter

```
  jupyter:
    image: rmuresano/bdaasjupyter:0_0_4
    hostname: jupyter
    container_name: jupyter
    depends_on:
      - sparkmaster
    links:
      - sparkmaster
    environment:
      - STANDALONE=NO
      - JUPYTERPORT=8900
      - HDFS=YES
      - HDFS_NAMENODE_HOSTNAME=sparkmaster
      - HDFS_DATANODE_WEBUI_PORT=50075
      - HDFS_NAMENODE_METADATA_PORT=9000
      - HDFS_REPLICATION_FACTOR=1
      - SPARK_MASTER_HOSTNAME=sparkmaster
      - SPARK_MASTER_PORT=7077
    ports:
      - "8900:8900" # WEB interface JupyterLab
    volumes:
      - jupyter-vol:/data/jupyter/
```

Run the example using: Docker-compose up --scale sparkworker=2

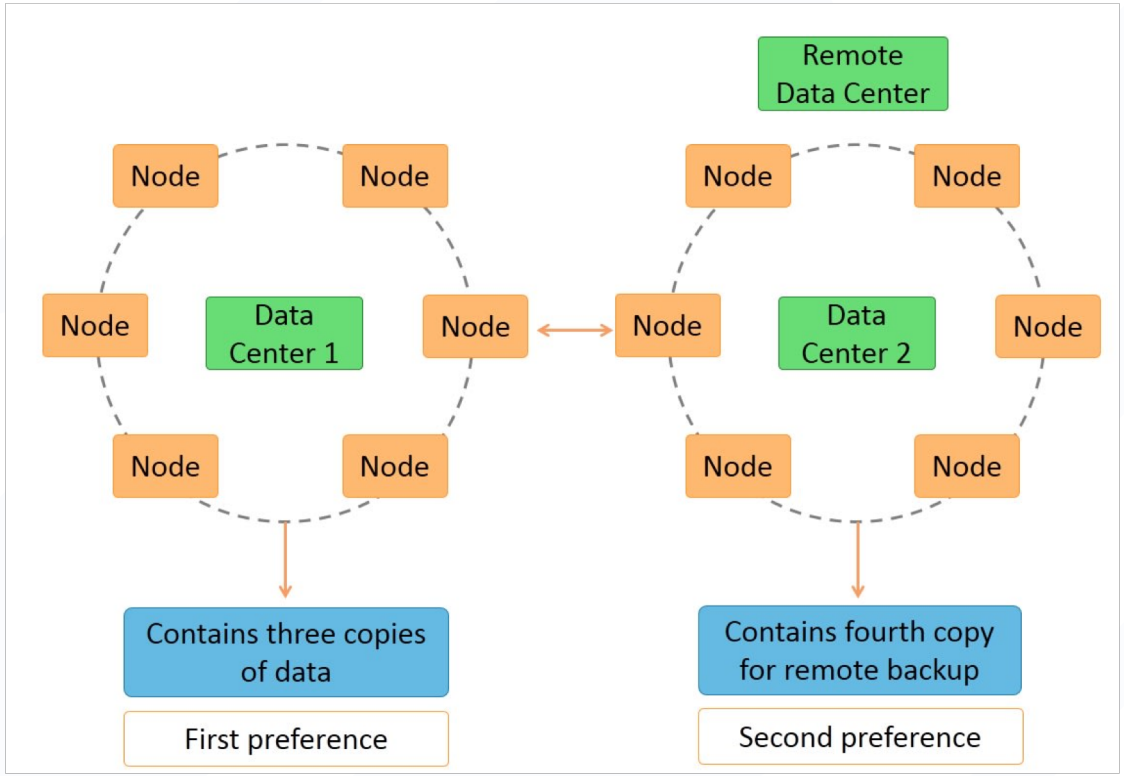
Running Docker-Compose (5/5)





Running another example

Exercise 4: Time 20 minutes (How to create a compose for Jupyter-Cassandra Cluster)




```
cassandra-seed:
  image: rmuresano/bdaascassandra:0_0_4
  environment:
    - CLUSTER_NAME=seminario
    - STORAGE_PORT=7000
    - NATIVE_TRANSPORT_PORT=9042
    - RPC_PORT=9160
    - CASSANDRA_SEEDS=172.28.1.2,172.28.1.3,172.28.1.4
  volumes:
    - cassandra-volume-seed:/data/cassandra
  networks:
    cassandra:
      ipv4_address: 172.28.1.2
```

Big Data as a Service: A new approach for the SMEs

Muchas gracias por su atención

Ronal Muresano
rmuresano@gmail.com
rmuresano@iti.es