Fundamentos de los datos

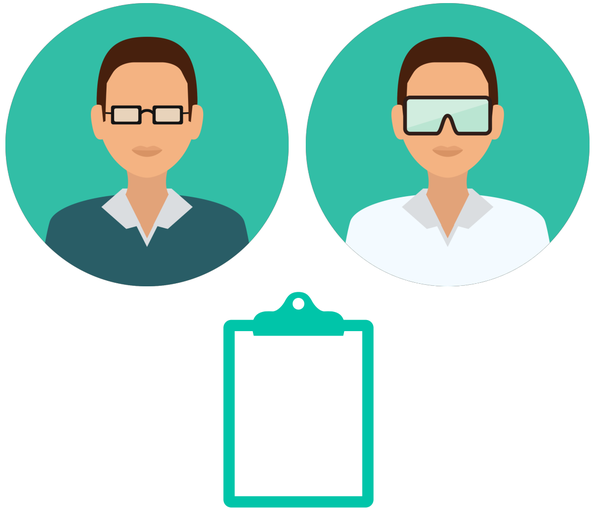
Towards reproducibility of computational environments for Scientific Experiments using Container-based virtualization

Maximiliano Osorio, Carlos Buil-Aranda, Hernán Vargas
UTFSM, Chile
Instituto Milenio de Investigación sobre los Fundamentos de los Datos, Chile

RESEARCHERS

Inspired by O. Corcho

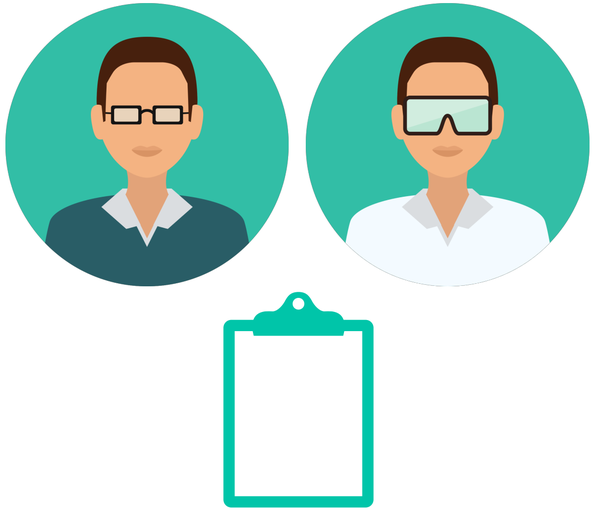
HYPOTHESIS



RESEARCHERS

Inspired by O. Corcho

HYPOTHESIS



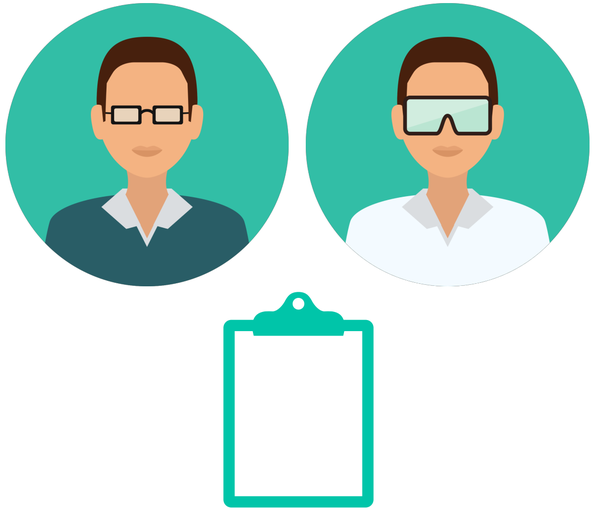
RESEARCHERS

EXPERIMENTS

RESULTS

Inspired by O. Corcho

HYPOTHESIS



REVIEWERS AND RESEARCHERS

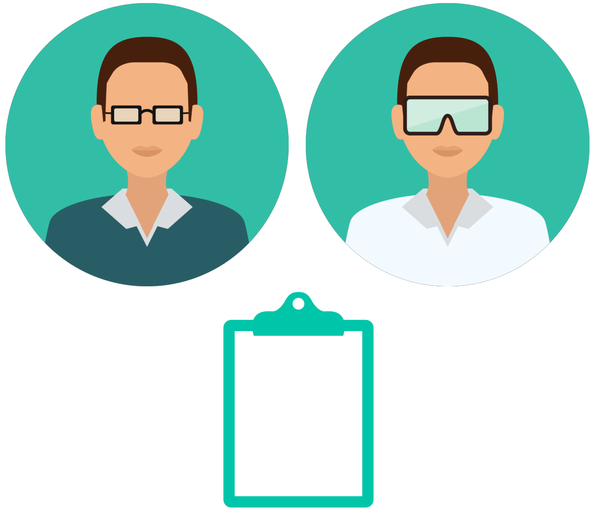
RESEARCHERS

EXPERIMENTS

RESULTS

Inspired by O. Corcho

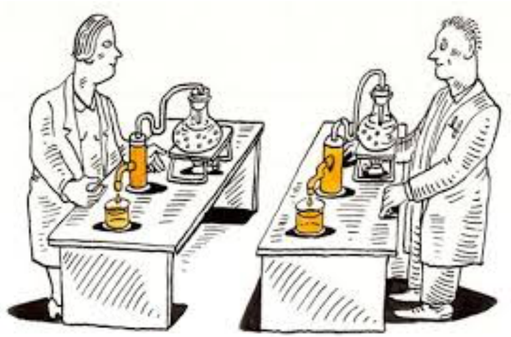
HYPOTHESIS



RESEARCHERS

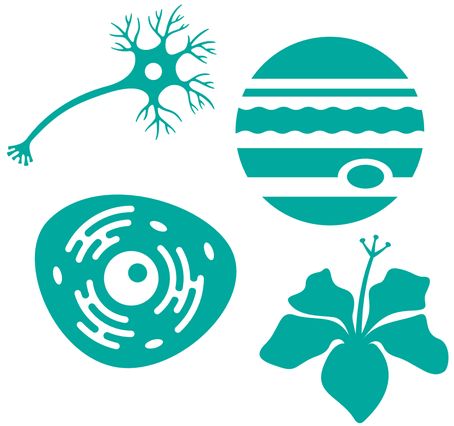
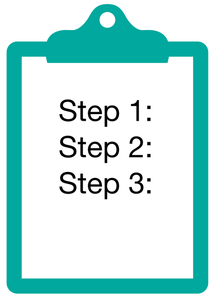
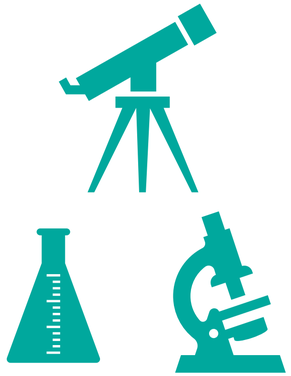
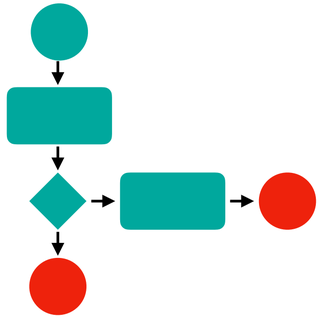
REVIEWERS AND RESEARCHERS

REPRODUCIBILITY



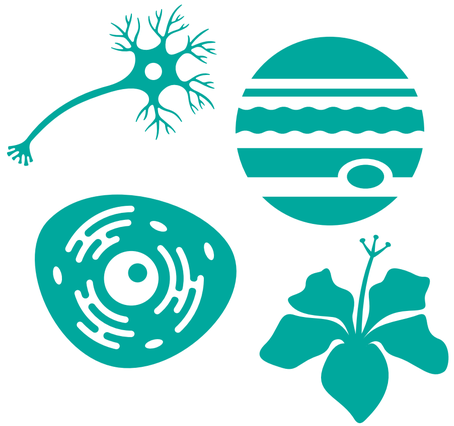
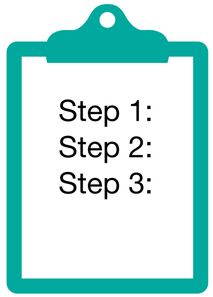
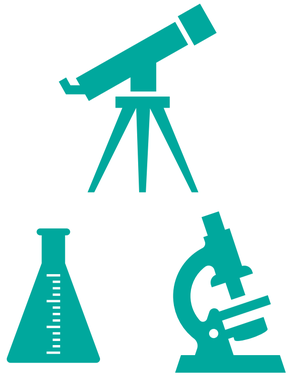
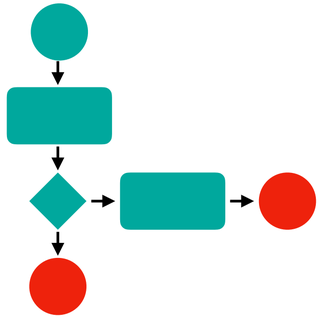
Inspired by O. Corcho

RESOURCES

	DATA	METHOD	EQUIPMENT
IN VIVO/ VITRO			
IN SILICO			

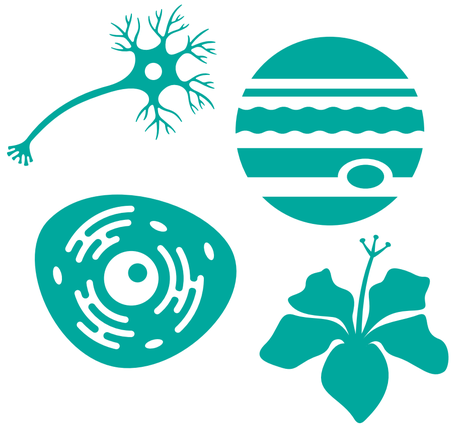
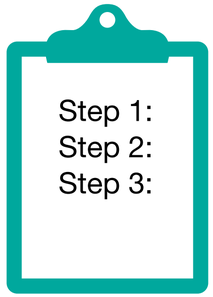
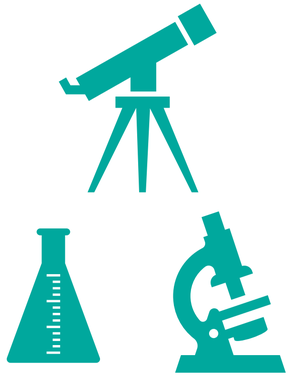
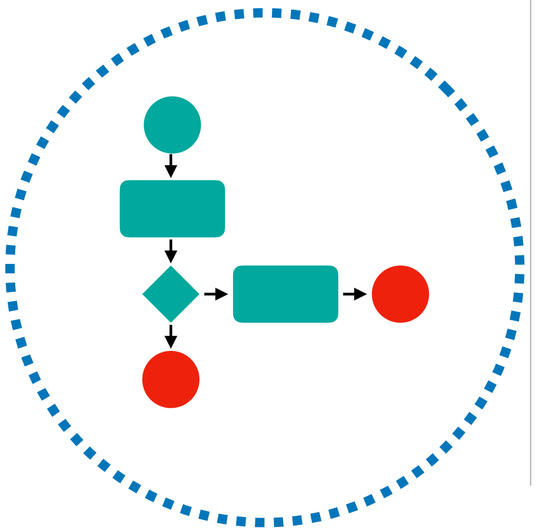
Inspired by O. Corcho

RESOURCES

	DATA	METHOD	EQUIPMENT
IN VIVO/ VITRO			
IN SILICO			

Inspired by O. Corcho

RESOURCES

	DATA	METHOD	EQUIPMENT
IN VIVO/ VITRO			
IN SILICO			

Inspired by O. Corcho

RESOURCES

	DATA	METHOD	EQUIPMENT
IN VIVO/ VITRO			
IN SILICO			

Inspired by O. Corcho

CONSERVATION

In order to reproduce or replicate any digital artifact we need to properly handle its conservation. (King, 1995).

CONSERVATION

To achieve conservation one needs to guarantee that sufficient information exists with which to:

- Understand
- Evaluate
- Build

… without any additional information from the author (King, 1995).

TO RUN THE EXPERIMENT,

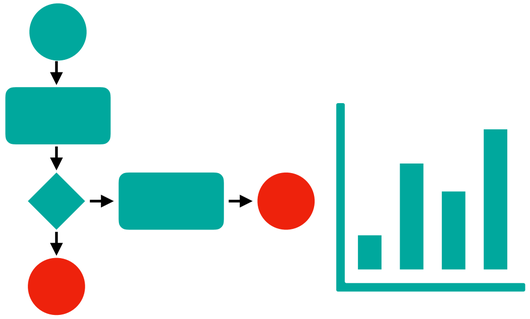
you must have a similar computational environment.

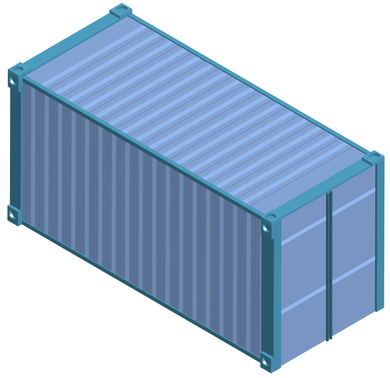
CONSERVATION TYPES

PHYSICAL
CONSERVATION

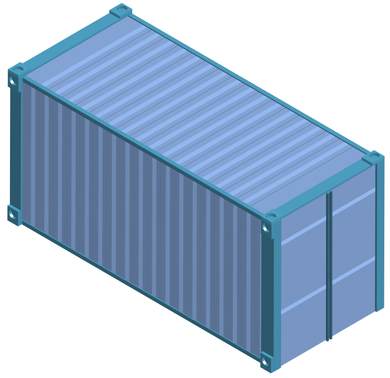
LOGICAL
CONSERVATION

PHYSICAL CONSERVATION

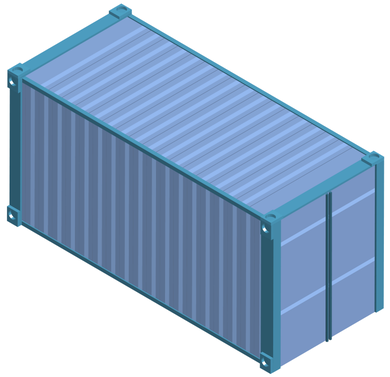




PHYSICAL CONSERVATION



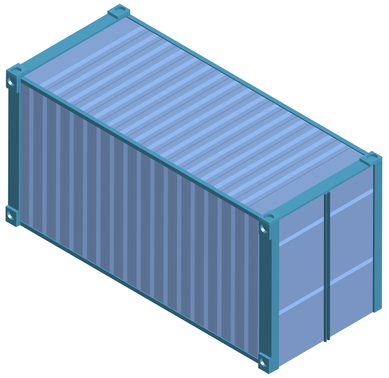
PHYSICAL CONSERVATION



PHYSICAL CONSERVATION

?

LOGICAL CONSERVATION



TODAY: ISSUES

01 Annotation processes are semi-automated, This leaves a lot of work to the scientists.

02 Logical conservation must be clean. Some annotation process require to install new components.

03 Physical conservation is nice but it is avoided because the high storage demand of VM images.

NEEDS

01 Logical and physical conservation are required

02 Describe the software components and building steps of the environment

03 Allow to the scientists share the environment and run it

INSPIRATION

- (Santana et. al 2017) define semantic vocabularies that describes the execution environment of scientific workflows, so as to conserve it.

- They define a process for documenting the workflow application and its related management system, as well as their dependencies
  These tools are not automated.

- They reproduce a workflow execution in different Cloud platforms using Virtual Machines

- They doesn’t address physical conservation

OUR WORK

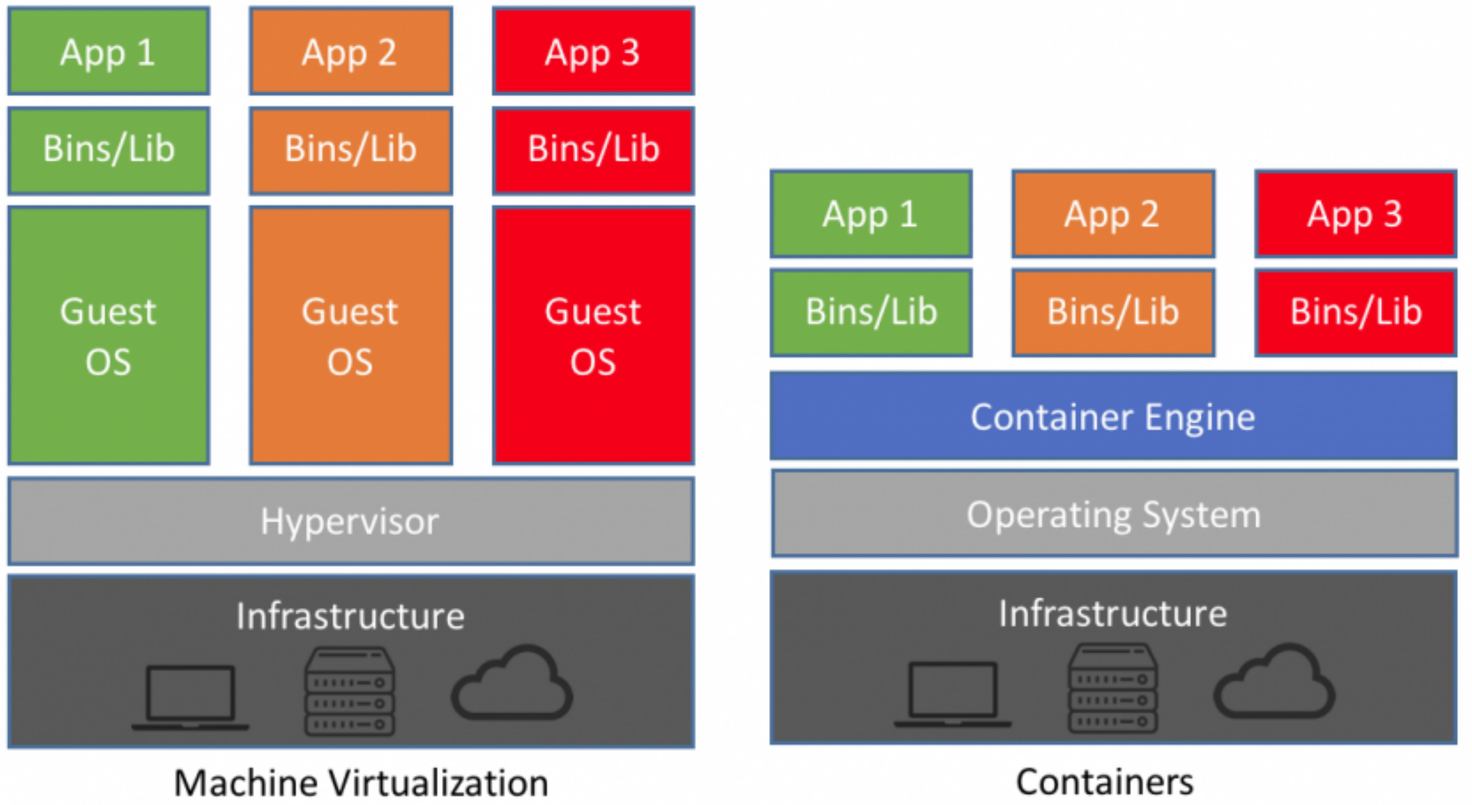
A **container** is used to **encapsulate** a **software component and the dependencies**.

DOCKERFILE

```
FROM ubuntu:16.04
LABEL maintainer="Craig Citro <craigcitro@google.com>"

# Pick up some TF dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
        build-essential \
        curl \
        libfreetype6-dev \
        libhdf5-serial-dev \
       libzmq3-dev \
        pkg-config \
        python \
        python-dev \
        rsync \
        software-properties-common \
       && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

RUN pip --no-cache-dir install \
        Pillow \
        h5py \
        ipykernel \
        jupyter \
        keras_applications==1.0.5 \
        keras_preprocessing==1.0.3 \
        matplotlib \
        numpy
```

DOCKER CONTAINERS

- Lightweight: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies.

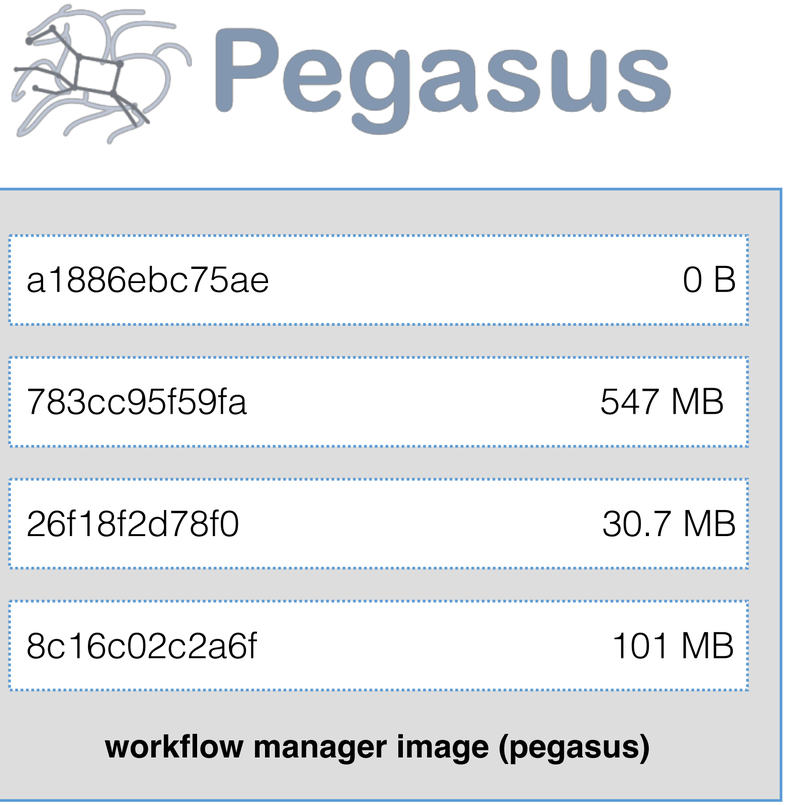


Source: docker.com

LAYERS

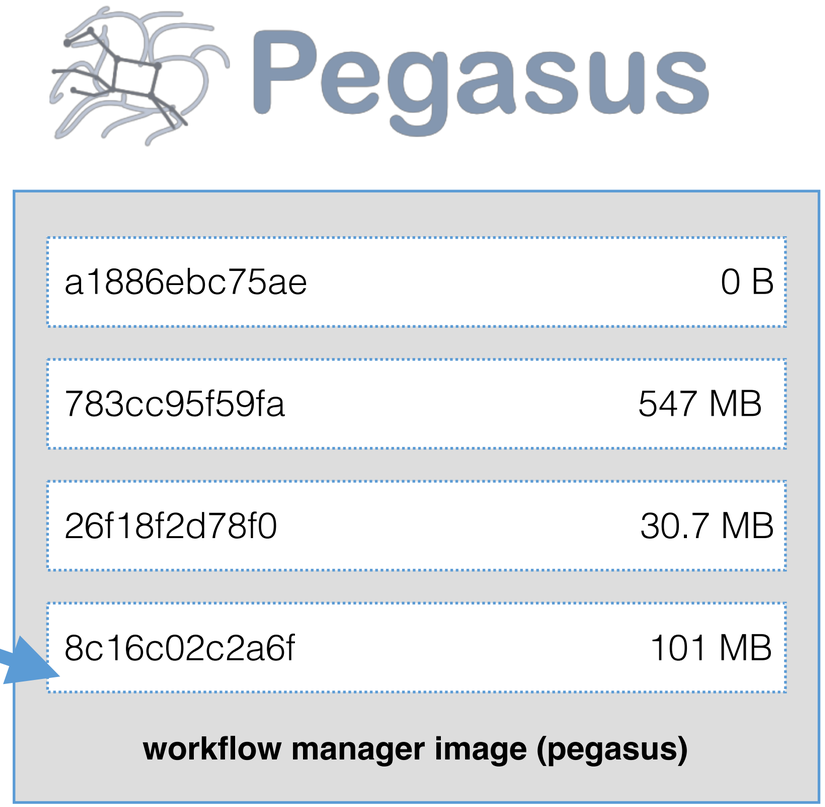
- Each Docker image references a list of read-only layers that represent filesystem differences.

```
FROM ubuntu:16.04

# Pick up some TF dependencies
RUN apt-get update && apt-get install -y --
no-install-recommends \
        wget \
        gnupg
RUN DEBIAN_FRONTEND=noninteractive apt-get
install -y --no-install-recommends pegasus
condor
```



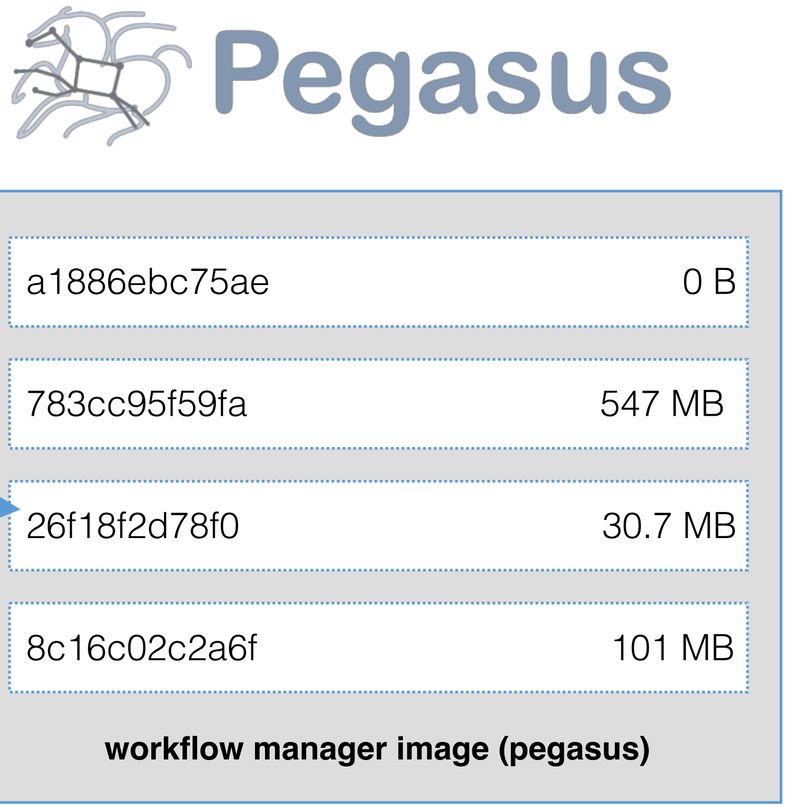
LAYERS

```
FROM ubuntu:16.04

# Pick up some TF dependencies
RUN apt-get update && apt-get install -y --
no-install-recommends \
        wget \
        gnupg
RUN DEBIAN_FRONTEND=noninteractive apt-get
install -y --no-install-recommends pegasus
condor
```



LAYERS

```
FROM ubuntu:16.04

# Pick up some TF dependencies
RUN apt-get update && apt-get install -y --
no-install-recommends \
        wget \
        gnupg
RUN DEBIAN_FRONTEND=noninteractive apt-get
install -y --no-install-recommends pegasus
condor
```



LAYERS

```
FROM ubuntu:16.04

# Pick up some TF dependencies
RUN apt-get update && apt-get install -y --
no-install-recommends \
        wget \
        gnupg
RUN DEBIAN_FRONTEND=noninteractive apt-get
install -y --no-install-recommends pegasus
condor
```

Pegasus

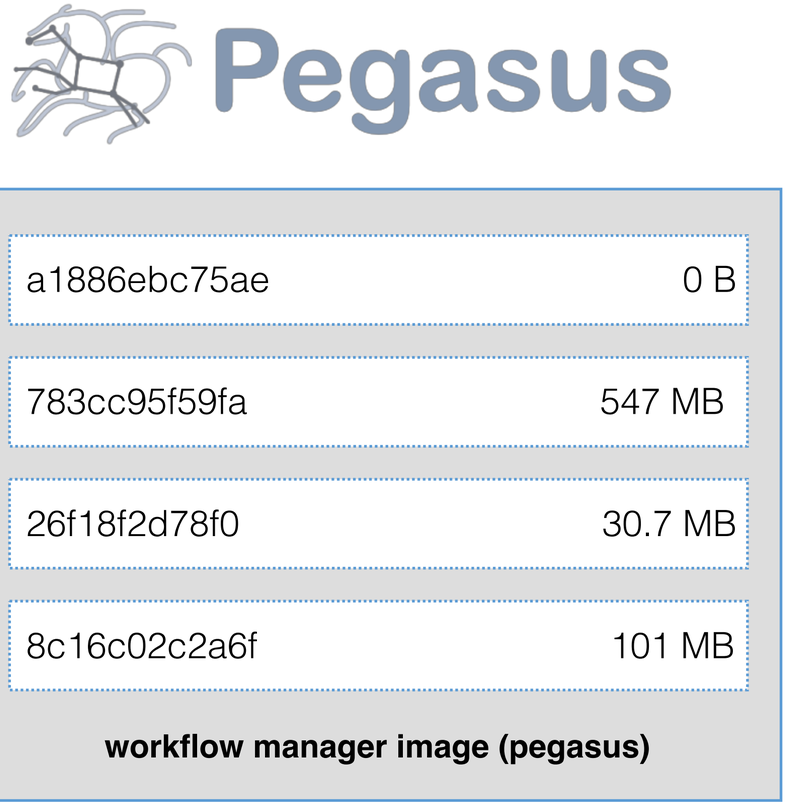
a1886ebc75ae	0 B
783cc95f59fa	547 MB
26f18f2d78f0	30.7 MB
8c16c02c2a6f	101 MB

workflow manager image (pegasus)

A NEW WORKFLOW

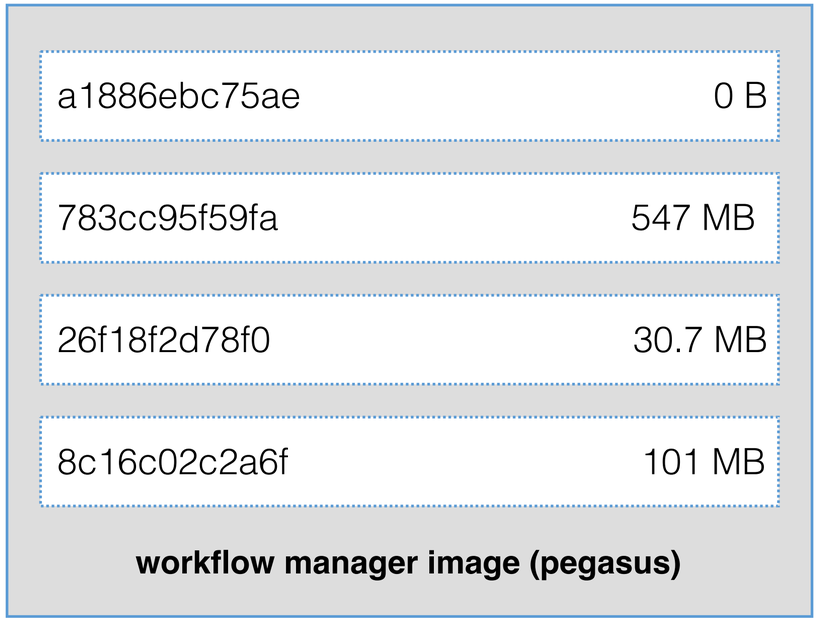
To create a new workflow, we need to import the pegasus image.
Next, we can install the new software

```
FROM pegasus
RUN pip install numpy
```

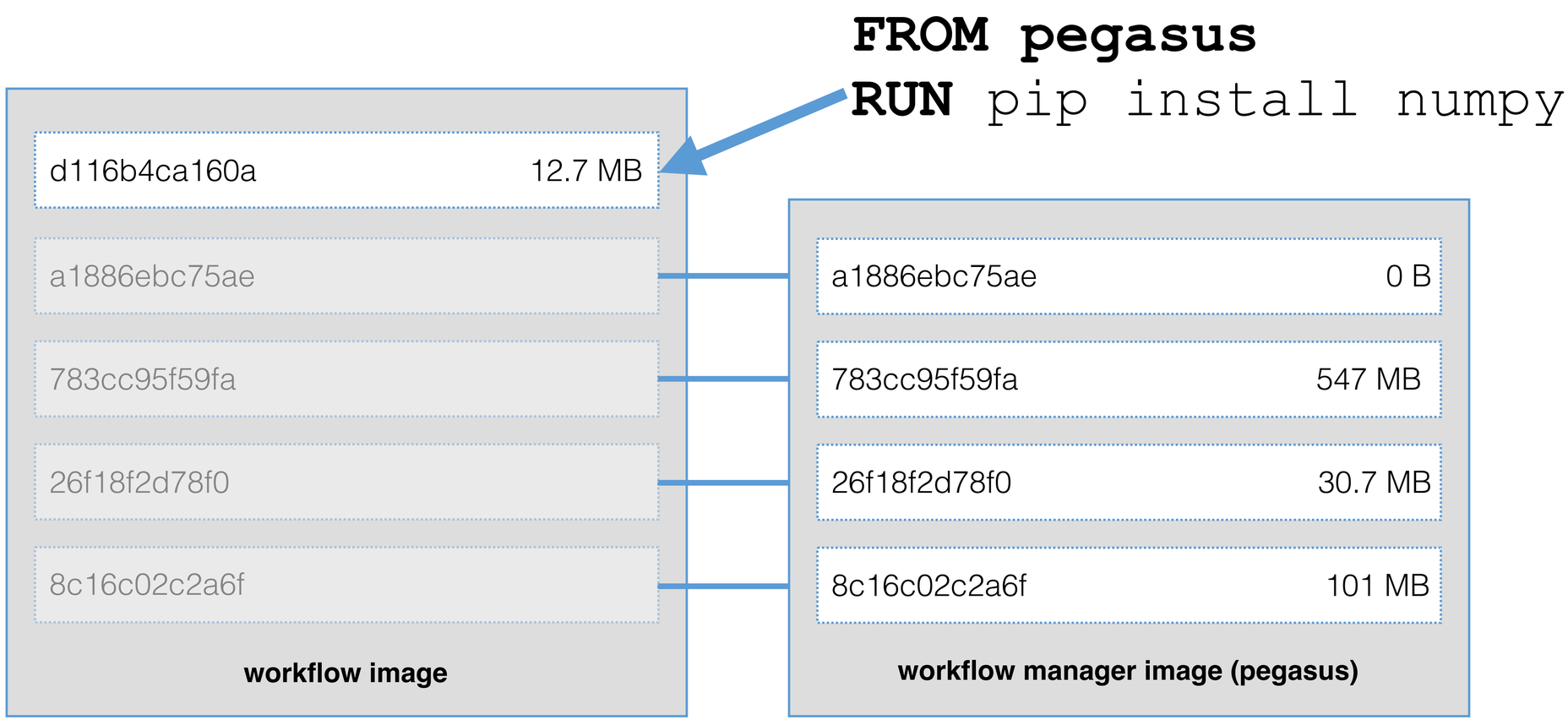


A NEW WORKFLOW

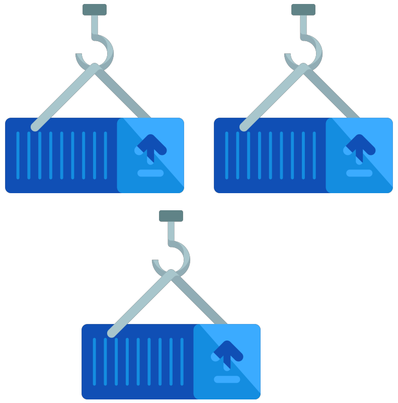
```
FROM pegasus
RUN pip install numpy
```



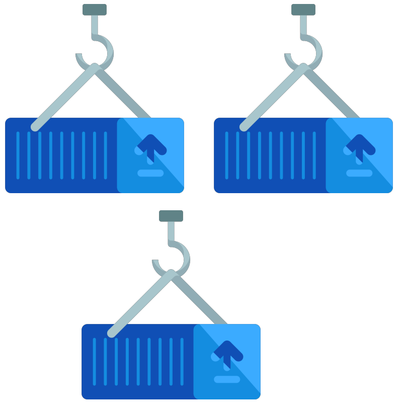
LAYERS


FROM pegasus
RUN pip install numpy
d116b4ca160a 12.7 MB
a1886ebc75ae
783cc95f59fa
26f18f2d78f0
8c16c02c2a6f
workflow image
a1886ebc75ae 0 B
783cc95f59fa 547 MB
26f18f2d78f0 30.7 MB
8c16c02c2a6f 101 MB
workflow manager image (pegasus)

OUR WORK

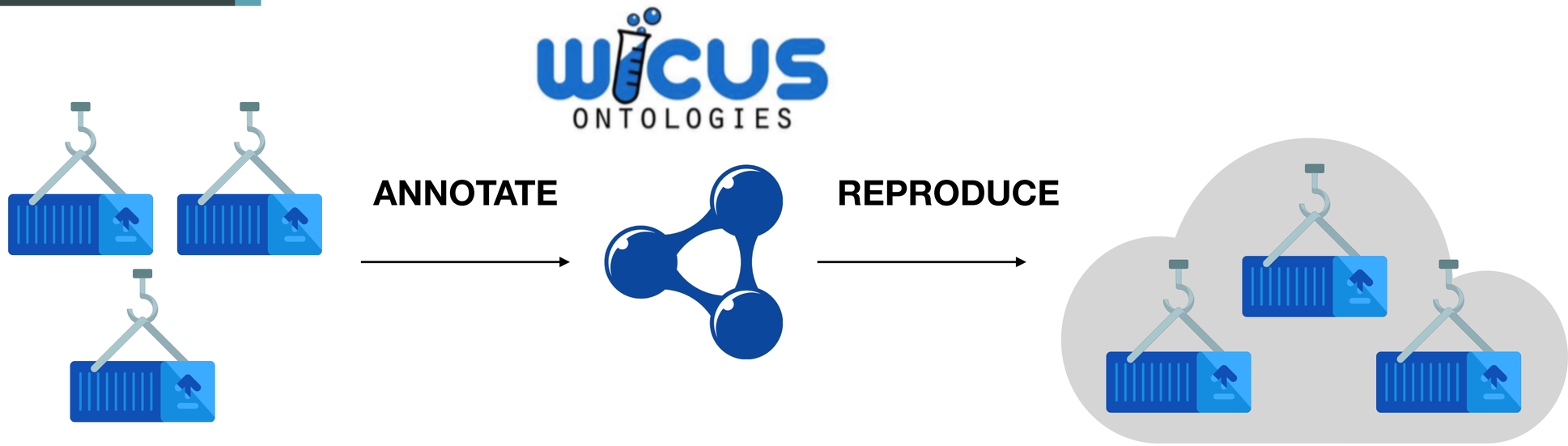


LOGICAL CONSERVATION

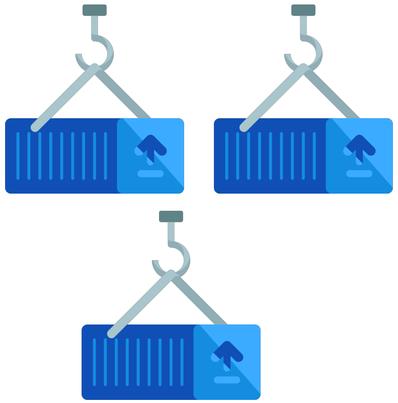


ANNOTATE

LOGICAL CONSERVATION



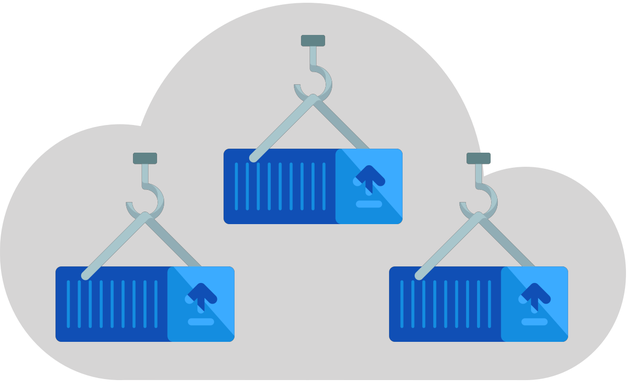
PHYSICAL CONSERVATION



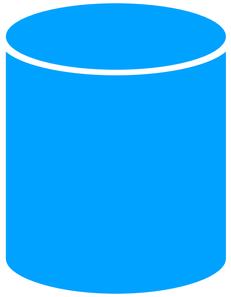
WICUS ONTOLOGIES

ANNOTATE

REPRODUCE



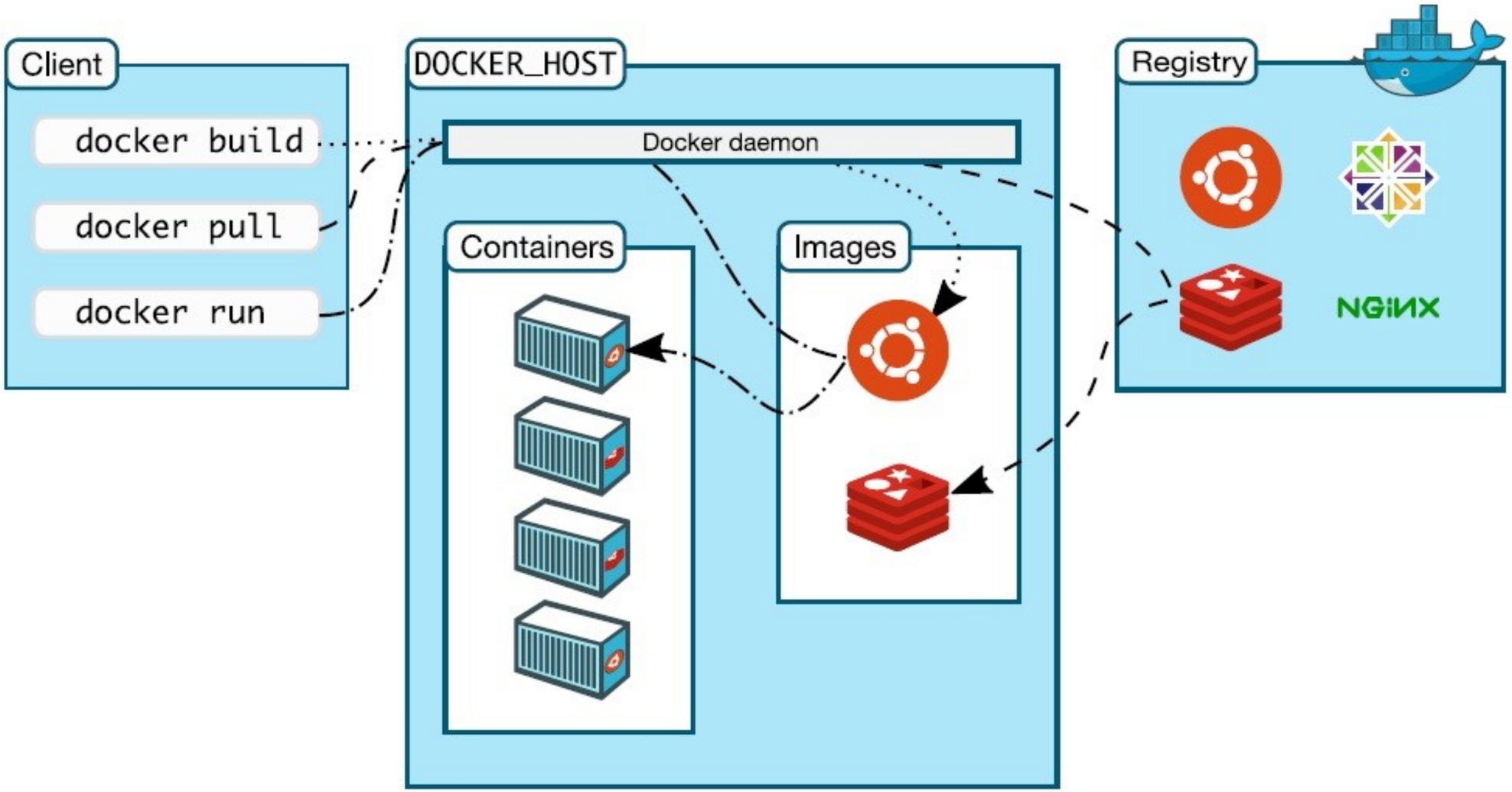
STORE



DOCKERHUB

```
FROM ubuntu:16.04
LABEL maintainer="Craig Citro <craigcitro@google.com>"

# Pick up some TF dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
        build-essential \
        curl \
        libfreetype6-dev \
        libhdf5-serial-dev \
       libzmq3-dev \
        pkg-config \
        python \
        python-dev \
        rsync \
        software-properties-common \
       && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

RUN pip --no-cache-dir install \
        Pillow \
        h5py \
        ipykernel \
        jupyter \
        keras_applications==1.0.5 \
        keras_preprocessing==1.0.3 \
        matplotlib \
        numpy
```

PHYSICAL CONSERVATION



LOGICAL CONSERVATION

```
apt-get install -y --no-install-recommends \
        build-essential \
        curl \
        libfreetype6-dev \
        libhdf5-serial-dev \
        libpng12-dev \
        libzmq3-dev \
        pkg-config \
        python \
        python-dev \
```

LOGICAL CONSERVATION

```
apt-get install -y --no-install-recommends \
        build-essential \
        curl \
        libfreetype6-dev \
        libhdf5-serial-dev \
        libpng12-dev \
        libzmq3-dev \
        pkg-config \
        python \
        python-dev \
```

You have 10 installed packages, right?

ISSUES

- This command installs 184 packages (we got this information from our approach).
- There is no information of the versions or dependencies.

ANNOTATIONS

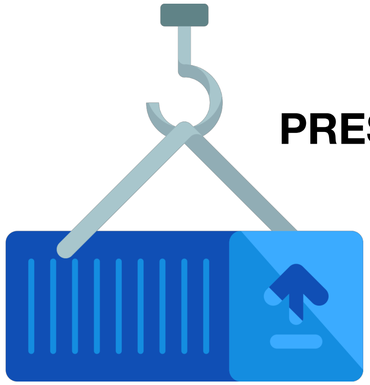
- We use and extend Clair (by CoreOS).
- Clair describe the software components (name, version) installed by:
  - apt (Debian, Ubuntu)
  - yum (RedHat, Centos, Oracle)
  - apk (Alpine)
- We extend Clair to describe other package managers: conda (any Linux)

BUILDING STEPS

- We store the Dockerfile.
- If not there is Dockerfile, we extract the building steps from the manifest of the image.

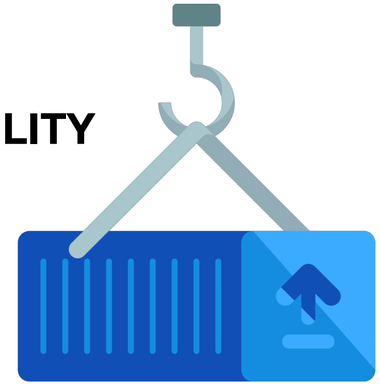

Pegasus

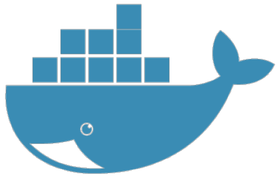
PHYSICAL CONSERVATION



PRESERVATION

REPLICABILITY

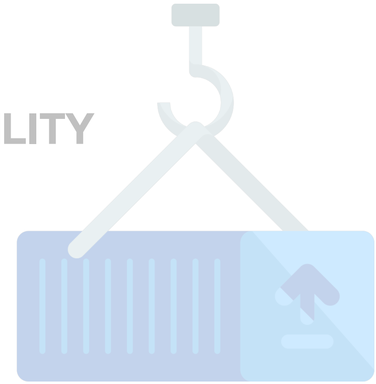




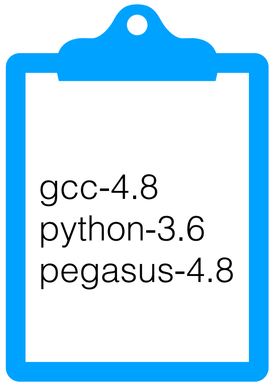
Pegasus

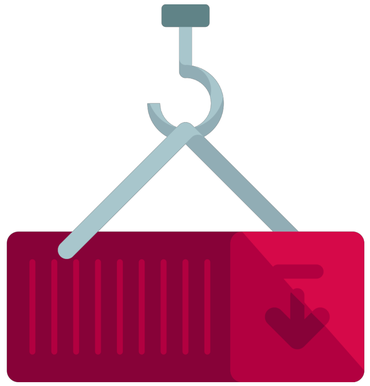
PHYSICAL CONSERVATION

PRESERVATION → REPLICABILITY



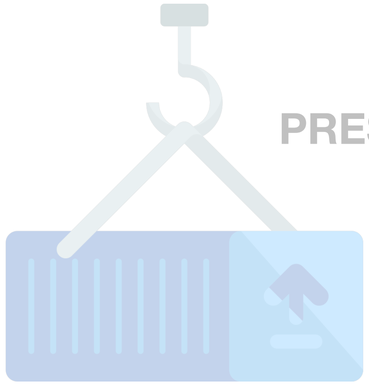
LOGICAL CONSERVATION

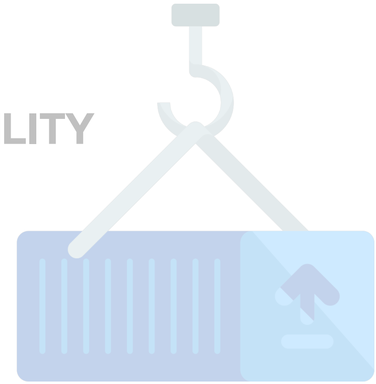




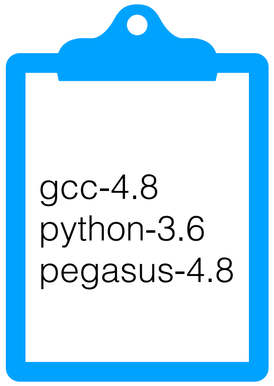
PHYSICAL CONSERVATION

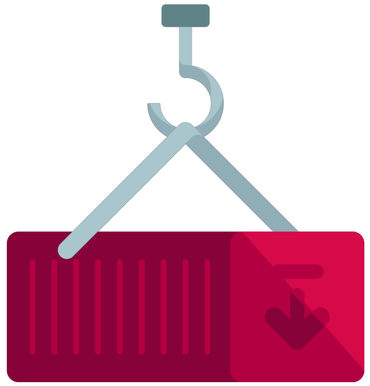
PRESERVATION → REPLICABILITY





LOGICAL CONSERVATION

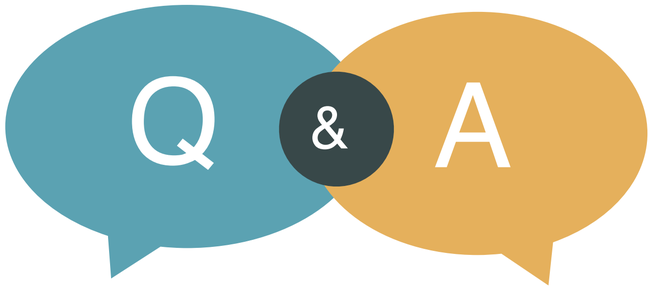




CONSERVATION → ALGORITHM → REPRODUCIBILITY

RESULTS

- A method to conduct logical and physical conservation of the computational environment of an experiment using Docker containers
- We achieve logical conservation without spending the considerable amount of effort annotating the software components and their dependencies
- We rely the physical conservation on DockerHub and their lightweight Docker images.



WE’LL BE ANSWERING QUESTIONS NOW