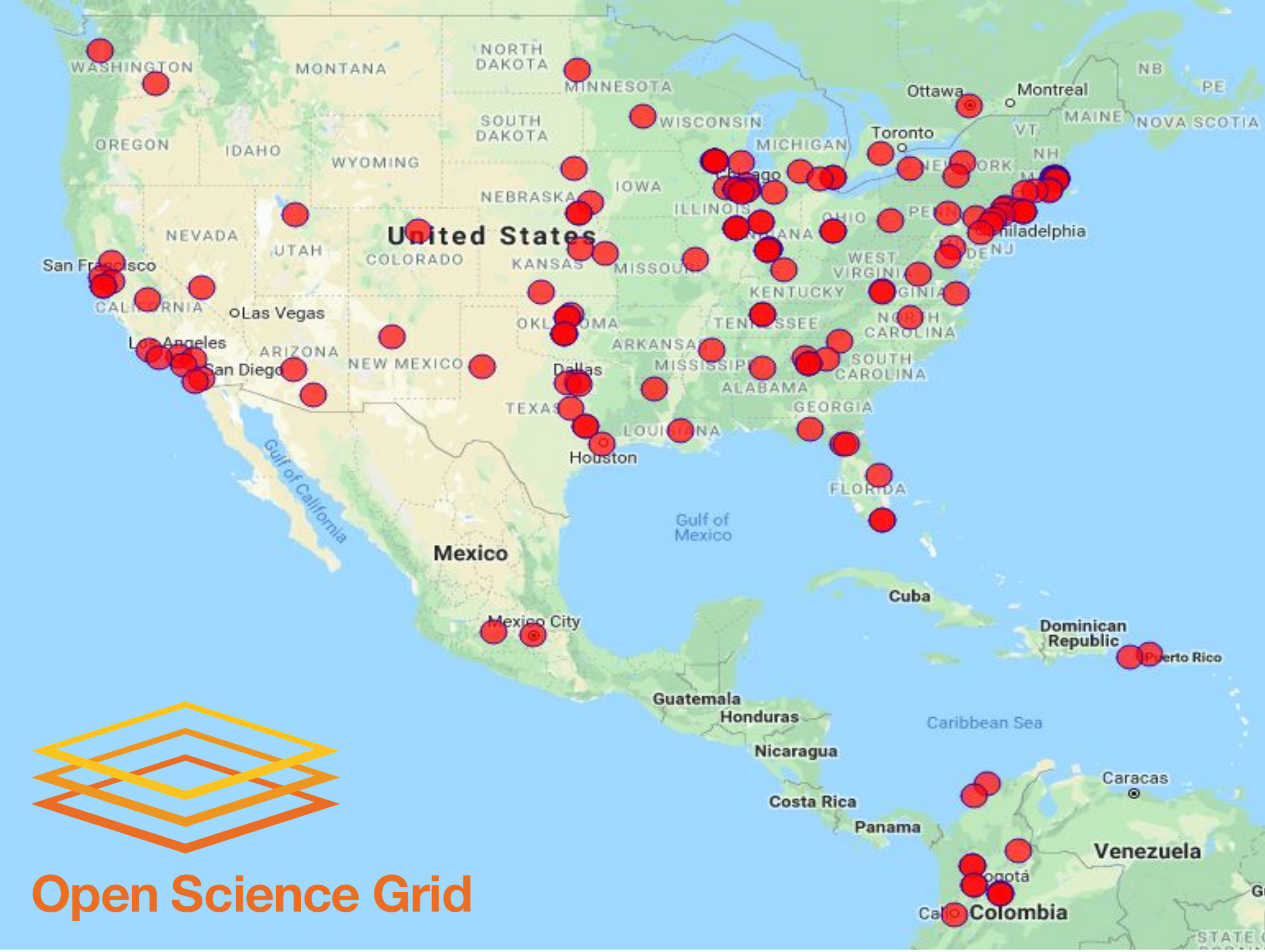
Containers in Distributed Computing

Mats Rynge, USC Information Sciences Institute

What is the
Open Science Grid?

In the last 24 Hours	
255,000	Jobs
4,482,000	CPU Hours
3,271,000	Transfers
774	TB Transfers
In the last 30 Days	
9,011,000	Jobs
135,559,000	CPU Hours
115,894,000	Transfers
28,650	TB Transfers
In the last 12 Months	
103,134,000	Jobs
1,676,117,000	CPU Hours
1,989,970,000	Transfers
288,000	TB Transfers



A consortium of researchers and institutions who *share* compute and data resources for ***distributed* high-throughput computing (dHTC)**

Is it OSG-able?

Open Science Grid

Per-Job Resources	**Ideal Jobs!** (up to 10,000 cores, per user!)	**Still Very Advantageous!**	**Probably not…**
cores (GPUs)	**1** (1; non-specific)	**<16** (1; specific GPU type)	**>16 (or MPI)** (multiple)
Walltime *(per job)*	**<10 hrs*** *or checkpointable	**<20 hrs*** *or checkpointable	**>20 hrs**
RAM *(per job)*	**<few GB**	**<10 GB**	**>10 GB**
Input *(per job)*	**<1 GB**	**<10 GB**	**>10 GB**
Output *(per job)*	**<1 GB**	**<10 GB**	**>10 GB**
Software	*'portable' (pre-compiled binaries, transferable, containerizable, etc.)*	*most other than* □□□	*licensed software; non-Linux*

Distributed Computing

- Resources are geographically distributed
  - But so is administration - local administrators, approaches and policies!

- Managing your software stack
  - You have N clusters to deploy on
  - All with smaller differences (for example - different set of base libraries/version/…)

- Container instance life cycle is usually short (ephemeral - life time of the job), and usually have no services

- Unique challenge: Managing container image distribution

Container Motivations

Consistent environment (default images) - If a user does not specify a specific image, a default one is used by the job. The image contains a decent base line of software, and because the same image is used across all the sites, the user sees a more consistent environment than if the job landed in the environments provided by the individual sites.

Custom software environment (user defined images) - Users can create and use their custom images, which is useful when having very specific software requirements or software stacks which can be tricky to bring with a job. For example: Python or R modules with dependencies, TensorFlow

Enables special environment such as GPUs - Special software environments to go hand in hand with the special hardware.

Process isolation - Sandboxes the job environment so that a job can not peek at other jobs.

File isolation - Sandboxes the job file system, so that a job can not peek at other jobs' data.

Container Lifecycle (Hint: ephemeral)

Each and every job is encapsulated in a separate container instance

Container instance dies when the job finishes

An incredible amount of container image reuse, as workloads generally use one or a small number of images for a large number of jobs

- 1st ton-scale experiment
- 3.2t of LXe, 2t in TPC
- All systems commissioned since Fall 2016
- Calibration and science data taking now ongoing

Experiment located at the Laboratori Nazionali del Gran Sasso (LNGS), Italy

Columbia · RPI · Nikhef · Muenster · Stockholm · Mainz · MPIK, Heidelberg · Freiburg · Zurich

Chicago

UCLA

UCSD

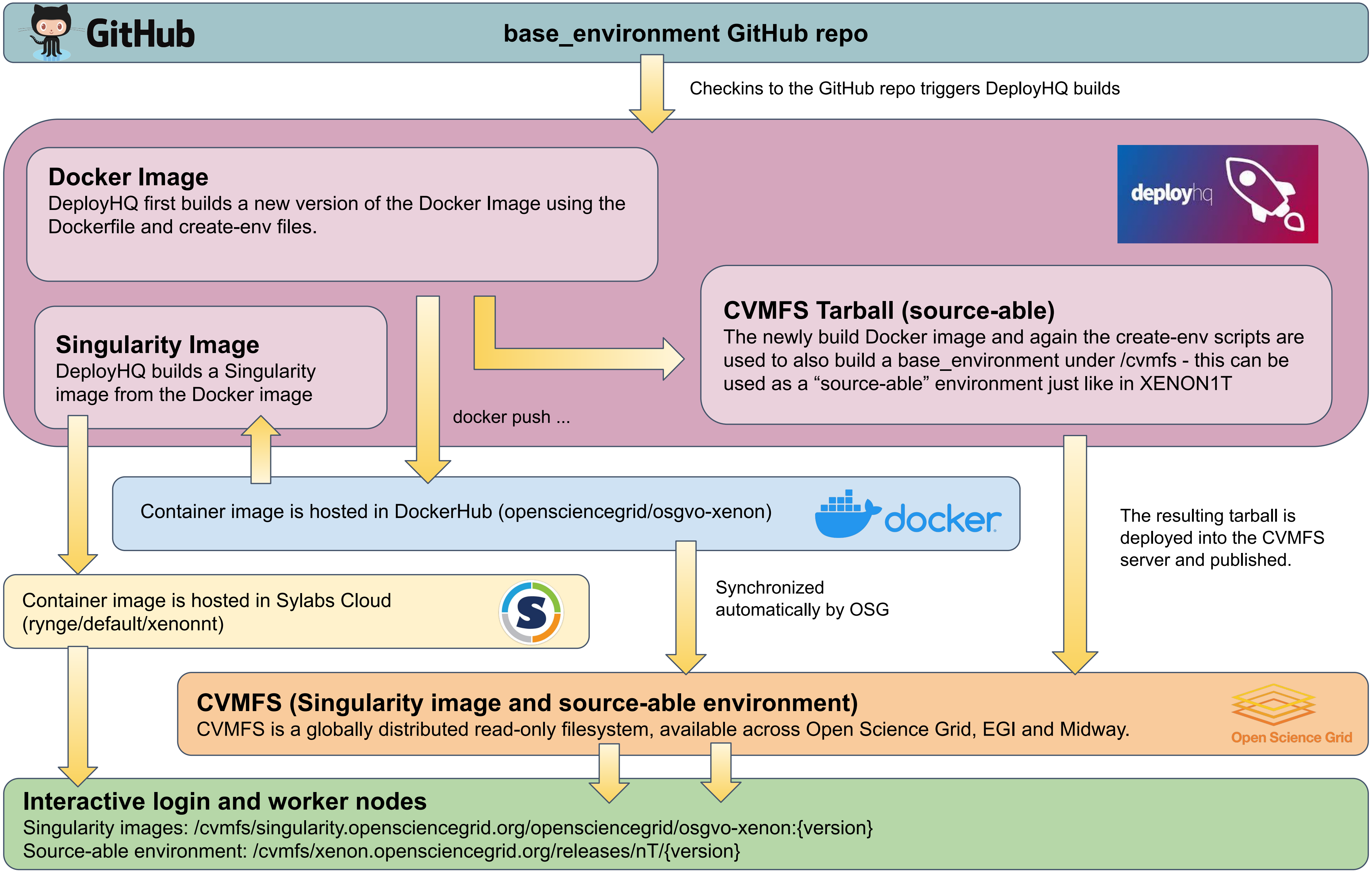
Rice

Tokyo

Nagoya

Kobe

Purdue · Coimbra · Subatech · LPNHE · LAL · Bologna · LNGS · Torino · Napoli · Weizmann · NYUAD


GitHub
base_environment GitHub repo
Checkins to the GitHub repo triggers DeployHQ builds
Docker Image
DeployHQ first builds a new version of the Docker Image using the Dockerfile and create-env files.
deployhq
CVMFS Tarball (source-able)
The newly build Docker image and again the create-env scripts are used to also build a base_environment under /cvmfs - this can be used as a "source-able" environment just like in XENON1T
Singularity Image
DeployHQ builds a Singularity image from the Docker image
docker push ...
Container image is hosted in DockerHub (opensciencegrid/osgvo-xenon)
docker
The resulting tarball is deployed into the CVMFS server and published.
Container image is hosted in Sylabs Cloud (rynge/default/xenonnt)
Synchronized automatically by OSG
CVMFS (Singularity image and source-able environment)
CVMFS is a globally distributed read-only filesystem, available across Open Science Grid, EGI and Midway.
Open Science Grid
Interactive login and worker nodes
Singularity images: /cvmfs/singularity.opensciencegrid.org/opensciencegrid/osgvo-xenon:{version}
Source-able environment: /cvmfs/xenon.opensciencegrid.org/releases/nT/{version}

Be careful with CPU capabilities (AVX for example) if you want to share your containers

Codes are often auto detecting CPU capabilities at runtime

Can cause problems if you build on a “new” machine and wants to run it on an “old” one

Don’t rely on “latest” or unversioned software installs in your Dockerfile - not reproducible!

```
docker pull tensorflow/tensorflow:latest
docker pull tensorflow/tensorflow:2.1.0

conda install tensorflow
conda install tensorflow==2.1.0
```

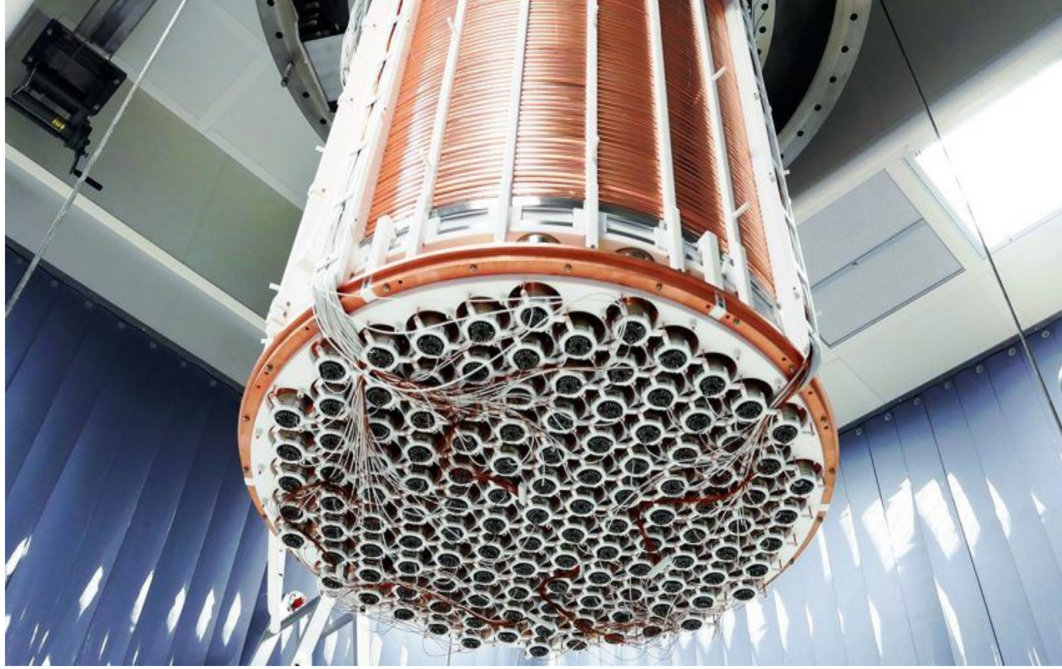
XENONnT - Dark Matter Search

Two workflows: Monte Carlo simulations, and the main processing pipeline.

Workflows execute across Open Science Grid (OSG) and European Grid Infrastructure (EGI)

Rucio for data management

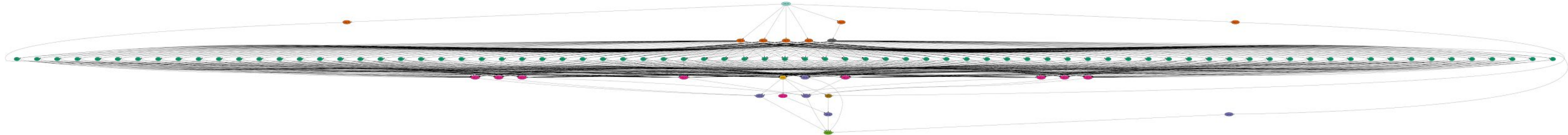
MongoDB instance to track science runs and data products.



```
------------------------------------------------------------------------------
Type           Succeeded Failed  Incomplete  Total     Retries   Total+Retries
Tasks          4000      0       0           4000      267       4267
Jobs           4484      0       0           4484      267       4751
Sub-Workflows  0         0       0           0         0         0
------------------------------------------------------------------------------

Workflow wall time                                       : 5 hrs, 2 mins
Cumulative job wall time                                 : 136 days, 9 hrs
Cumulative job wall time as seen from submit side        : 141 days, 16 hrs
Cumulative job badput wall time                          : 1 day, 2 hrs
Cumulative job badput wall time as seen from submit side : 4 days, 20 hrs
```

Main processing pipeline is being developed for XENONnT - data taking will start March 2020. Workflow in development:



Pegasus Workflow Management System

USC Viterbi
School of Engineering
Information Sciences Institute

https://pegasus.isi.edu

Why Pegasus?

Automates complex, multi-stage processing pipelines

Enables parallel, **distributed computations**

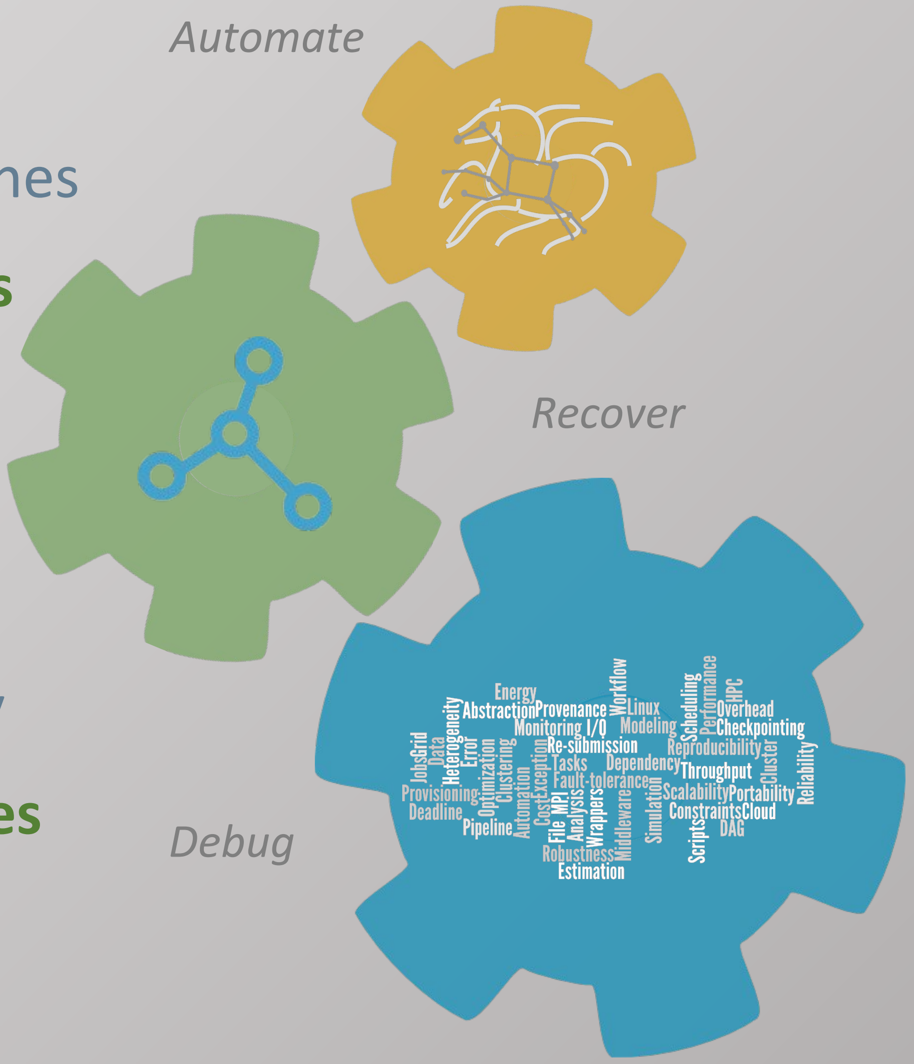
Automatically executes data transfers

Reusable, aids **reproducibility**

Records how data was produced (**provenance**)

Handles **failures** with to provide reliability

Keeps track of data and **files**



HTCondor High Throughput Computing

NSF funded project since 2001, with close collaboration with HTCondor team

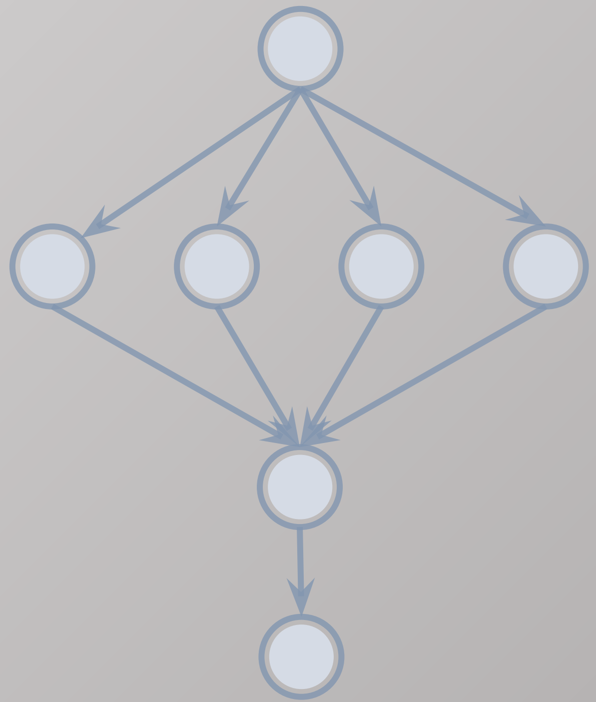
Key Pegasus Concepts

Pegasus WMS == Pegasus planner (mapper) + DAGMan workflow engine + HTCondor scheduler/broker

- Pegasus maps workflows to infrastructure
- DAGMan manages dependencies and reliability
- HTCondor is used as a broker to interface with different schedulers

Workflows are DAGs

- Nodes: jobs, edges: dependencies
- No while loops, no conditional branches
- Jobs are standalone executables



Planning occurs ahead of execution

Planning converts an abstract workflow into a concrete, executable workflow

- Planner is like a compiler

DAX

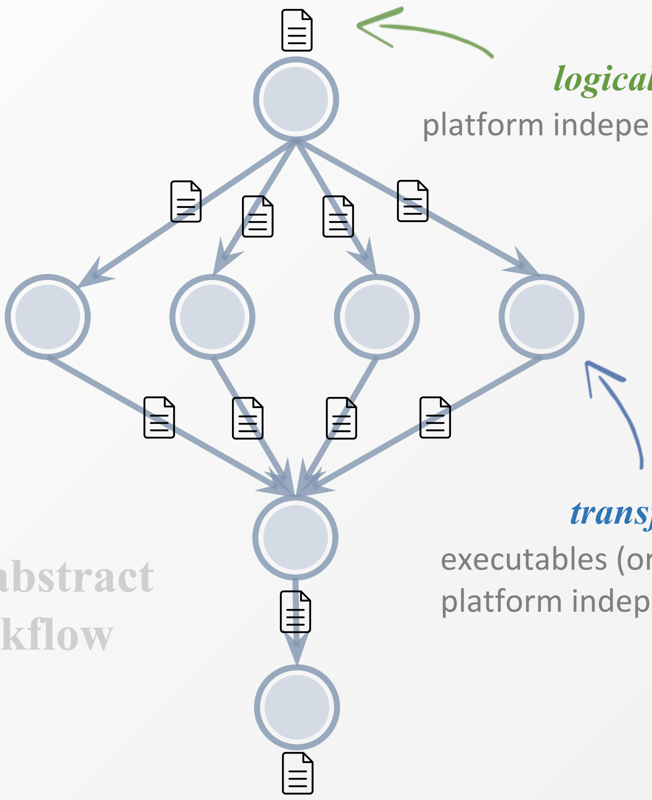
DAG in XML

DAG

directed-acyclic graphs

Portable Description

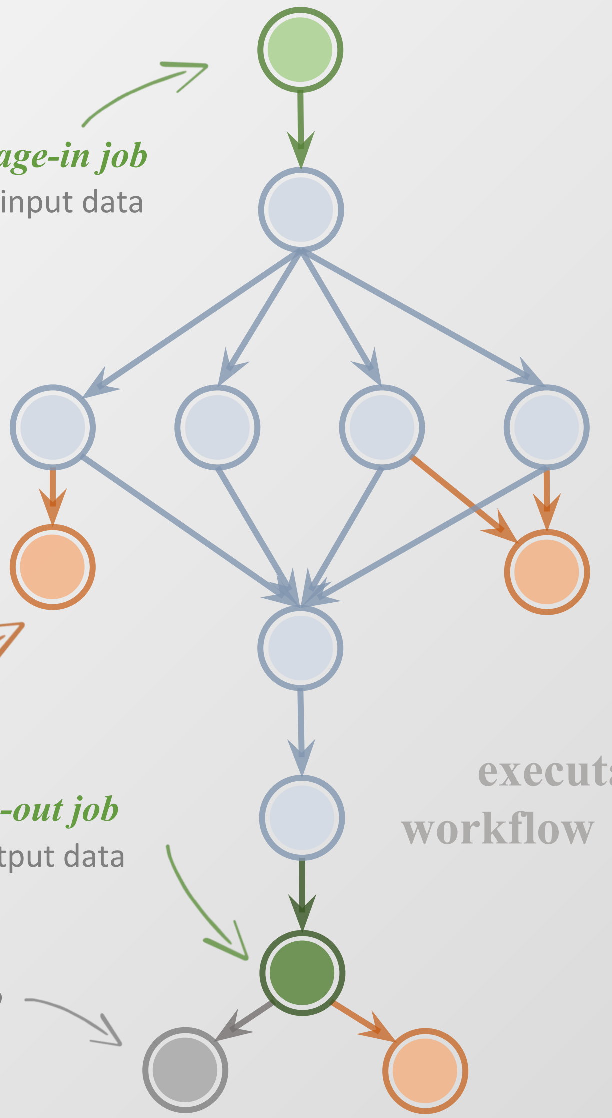
Users do not worry about
low level execution details



logical filename (LFN)
platform independent (abstraction)

transformation
executables (or programs)
platform independent

abstract
workflow



stage-in job
Transfers the workflow input data

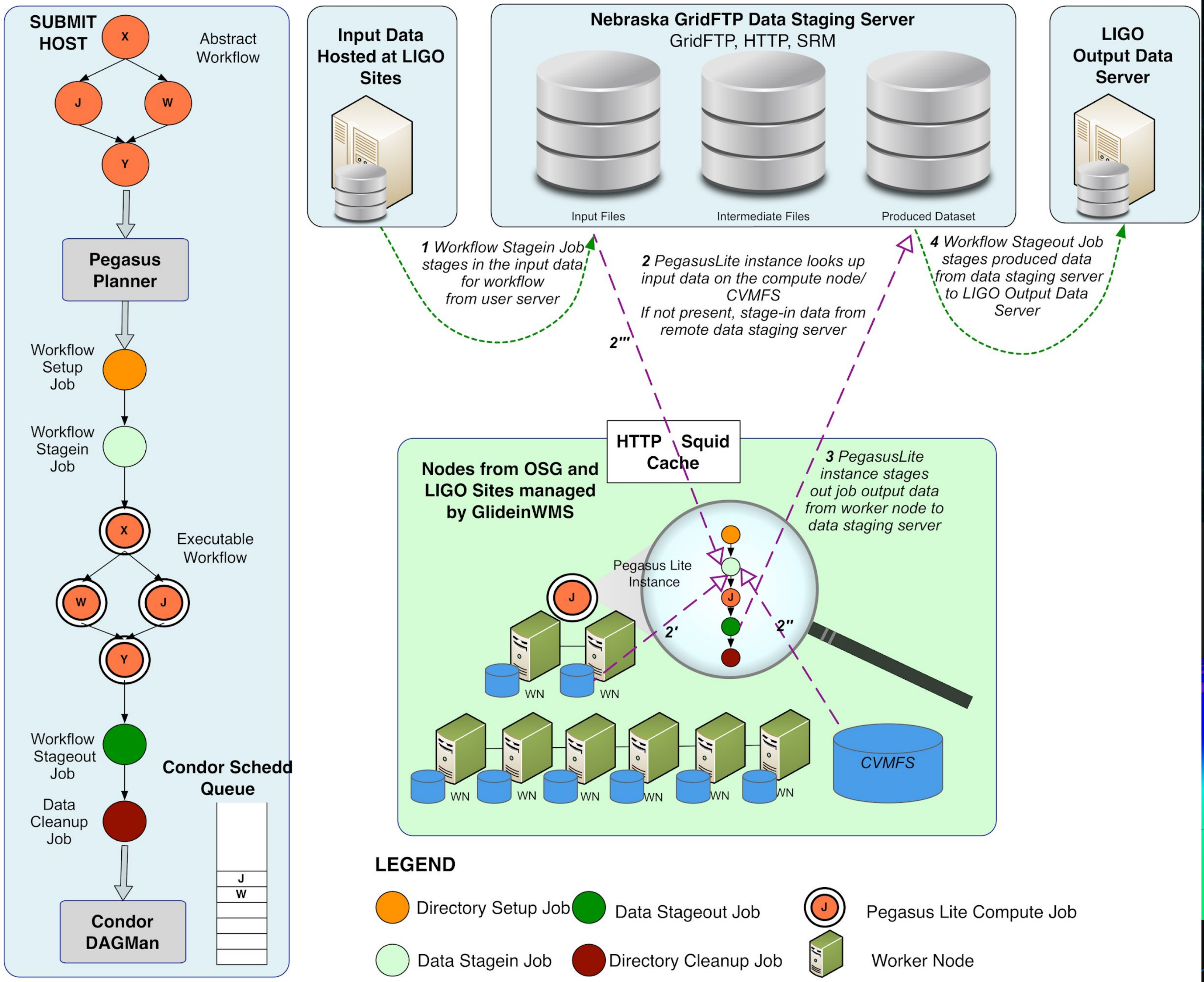
cleanup job
Removes unused data

stage-out job
Transfers the workflow output data

registration job
Registers the workflow output data

executable
workflow

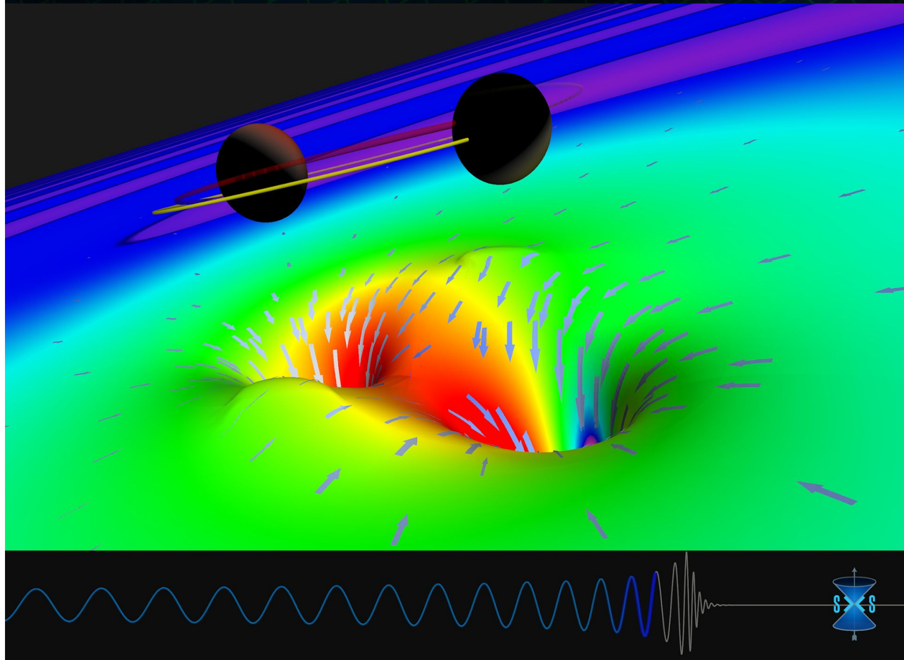
Data Flow for LIGO Pegasus Workflows in OSG

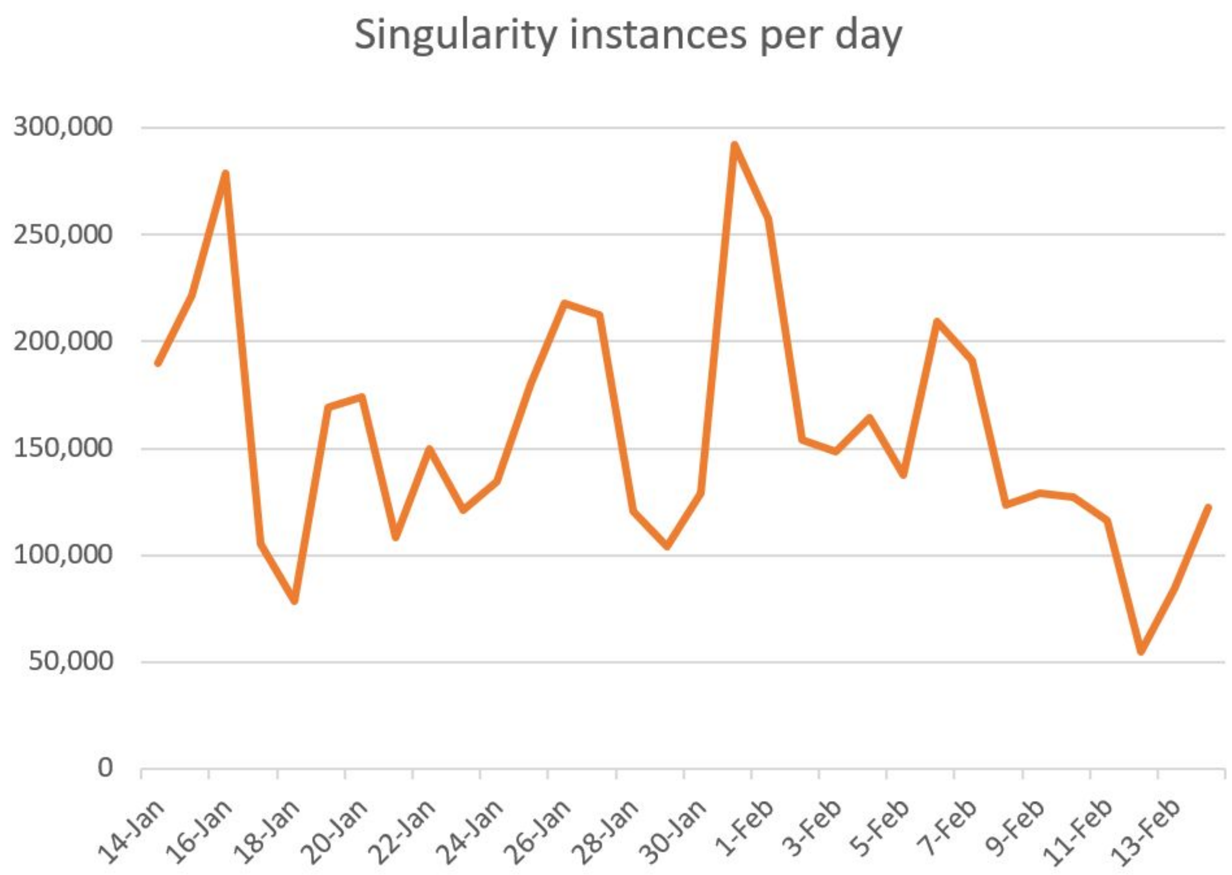


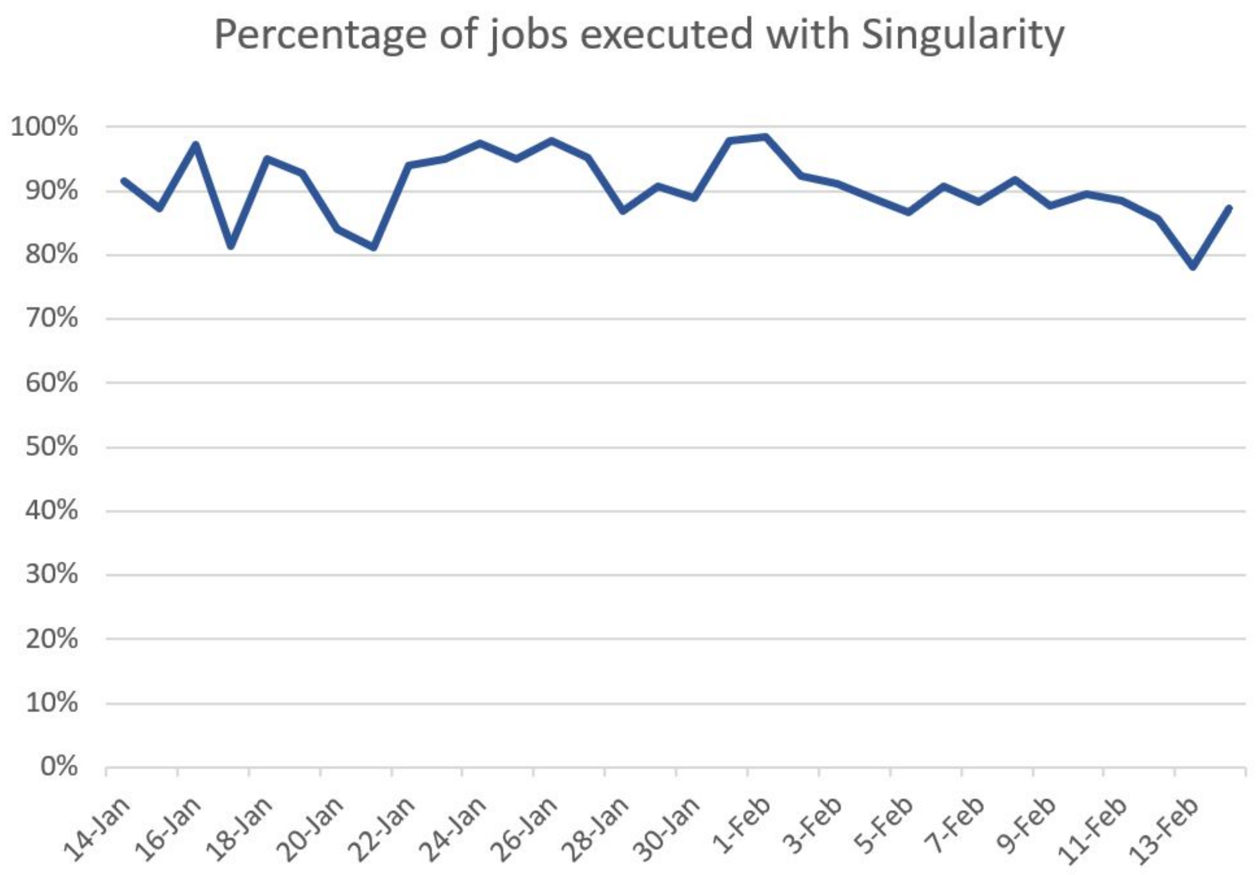
Advanced LIGO – Laser Interferometer Gravitational Wave Observatory

60,000 compute tasks
Input Data: 5000 files (10GB total)
Output Data: 60,000 files (60GB total)

Executed on LIGO Data Grid, EGI, Open Science Grid and XSEDE







300,000 containers x 5 GB (average size of container) =

1.5 PB / day

We need an efficient way to distribute containers!

CVMFS - CERN Virtual Machine File System

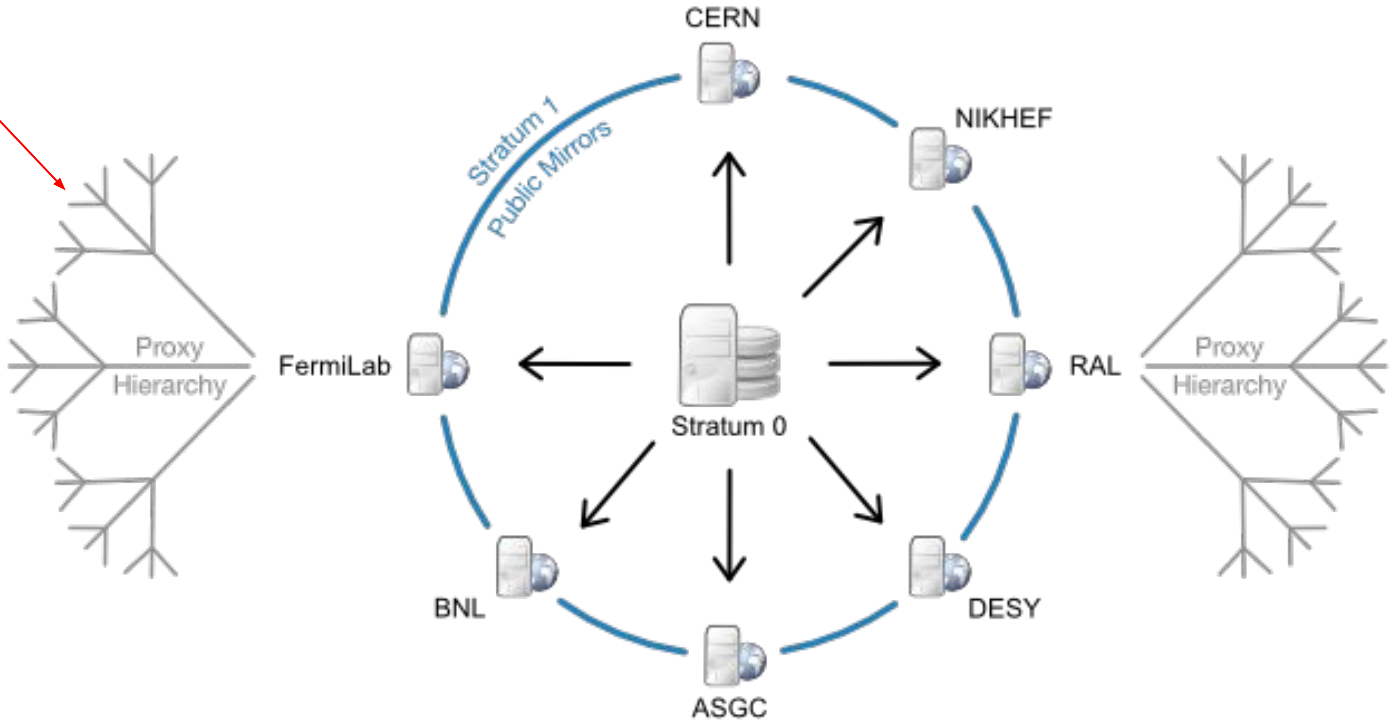
"The CernVM File System provides a scalable, reliable and low-maintenance software distribution service. It was developed to assist High Energy Physics (HEP) collaborations to deploy software on the worldwide-distributed computing infrastructure used to run data processing applications. CernVM-FS is implemented as a POSIX read-only file system in user space (a FUSE module). Files and directories are hosted on standard web servers and mounted in the universal namespace /cvmfs."

Your job is here!

Used for software and data!

Heavily cached, read-only

Available across OSG, EGI, some XSEDE resources



CVMS Repositories

/cvmfs/

ams.cern.ch

atlas.cern.ch

cms.cern.ch

connect.opensciencegrid.org

gwosc.osgstorage.org

icecube.opensciencegrid.org

ligo-containers.opensciencegrid.org <- large project with their own containers

nexo.opensciencegrid.org

oasis.opensciencegrid.org <- “modules” software

singularity.opensciencegrid.org <- general containers (next few slide)

snoplus.egi.eu

spt.opensciencegrid.org <- project from talk yesterday (South Pole Telescope)

stash.osgstorage.org <- ~1PB of user published data

veritas.opensciencegrid.org

xenon.opensciencegrid.org

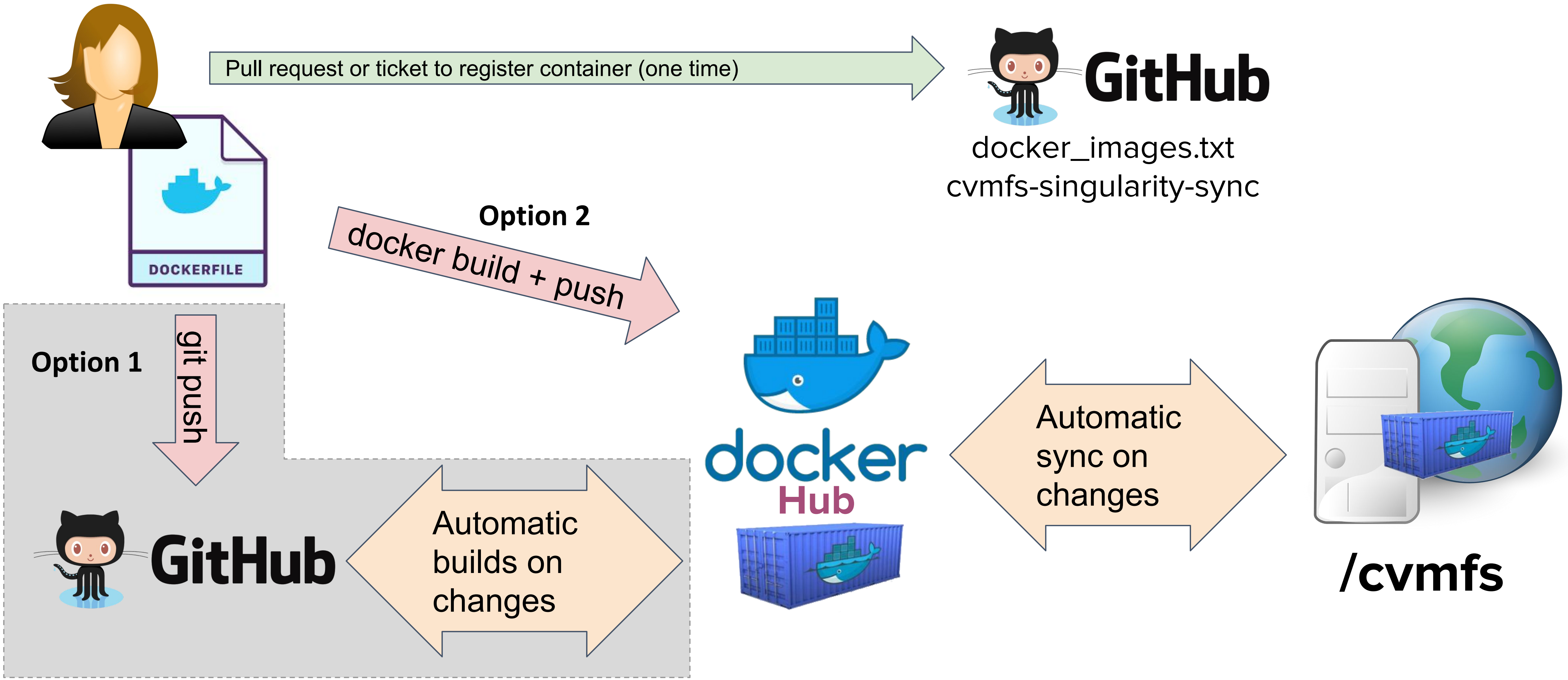
cvmfs-singularity-sync

- Containers are defined using Docker
  - Public Docker Hub

- … and executed with Singularity
  - No direct access to the Singularity command line - that is controlled by the infrastructure

- https://github.com/opensciencegrid/cvmfs-singularity-sync (next slide)

User-defined Container Publishing



Extracted Images

OSG stores container images on CVMFS in extracted form. That is, we take the Docker image layers or the Singularity img/simg files and export them onto CVMFS. For example, ls on one of the containers looks similar to ls / on any Linux machine:

```
$ ls /cvmfs/singularity.opensciencegrid.org/opensciencegrid/osgvo-el7:latest/
cvmfs  host-libs  proc  sys  anaconda-post.log     lib64
dev    media      root  tmp  bin                   sbin
etc    mnt        run   usr  image-build-info.txt  singularity
home   opt        srv   var  lib
```

Result: Most container instances only use **a small part** of the container image **(50-150 MB)** and that part is **cached** in CVMFS!

Thank you!