Technical Studentship overview and

Docker containers setup for the LCG Project

Javier Cervantes Villanueva
Supervisor: Patricia Mendez Lorenzo

Technical Student
Universidad de Murcia (ES)

About me

- BS in Computer Science
- MS in Big Data and Machine Learning *(in progress)*
- Technical Student in the LCG Project *(Sept 2016)*

My work in SFT

- New LCG stack using Python3
- LCG Operations
- Spack
- Containerisation of LCG builds
  - From build nodes to Docker containers
  - Monitoring infrastructure with Grafana and Graphite
- LCGdocs, documentation webpage

New LCG stack using Python3

- Included a new HEP toolchain*: *dev3python3*
- Cloned from dev3 with the Python version as the only difference:

  2.7.3 → 3.5.2

- Several patches needed (further details):
  - Long list of *non-Python3-compliant* packages
- Publishing Python3 LCG releases since LCG_88 (March 2017)
- Currently used in production by SWAN

* A toolchain defines a stack of *packages+versions* to install with LCGCMake
Full list: https://gitlab.cern.ch/sft/lcgcmake/blob/master/cmake/toolchain

LCG Operations – Main contributions

- Actively helped with the following tasks:
  - Installing new packages
  - Updating package versions
  - Including new compilers
  - Managing the build infrastructure (nodes, jenkins, …)
  - Taking care of regular nightlies and releases

- Refactoring of LCG jenkins scripts
  - Modular approach *(easier to add new installation processes)*
  - Homogenous incremental way for both nightlies and releases *(faster installations)*

LCG Operations – Main contributions

- Implemented a Python Library (releasepy) for managing nightlies/releases in CVMFS:
  - Check release installation
  - Remove packages, platforms or releases
  - Check RPM consistency

- Compiler wrapper to force use of compiler flags in new architectures *(-avx2)*
  - Using CMake templates

Spack - Reminder

- Package manager tool from SuperComputing
- One of the most suitable candidate for a common packaging tool for the HEP community
- Active community

More info: Spack – Package manager tool
HSF Meetings

Spack – Proof of principle

- Full nightly builds running with Spack instead of LCGCMake
- Integration with our Jenkins and CDash instances
- Customized recipes for ROOT and Geant4 projects
- Contributions to upstream project on Github
  - Adding support for more than 25 new packages
  - Most of them Python and R modules
- Creation of our own CERN Spack repo
  - Containing mostly MC Generator packages
  - Used and recognized by HSF groups actively working on Spack

Containerisation of the LCG builds

Containerisation - Why Docker containers?

- Current build nodes scenario in numbers:

8 CentOS7

3 CC7

2 SLC6 Physical machines

17 SLC6

7 Ubuntu

- Puppet configuration tool:
  - Extremely useful
  - Not enough to ensure exactly the same environment for every new build

Containerisation – Benefits

- Environment isolation
  - Fixed container image, same build environment
- All Docker hosts run the same OS *(CentOS7)*
  - Only one platform to manage with Puppet
  - Still possible to build for the same platforms *(CentOS7, SLC6, Ubuntu)*
- More flexibility defining different environments
  - New platforms just need a new Docker image
  - No need of new Hardware / Quota / VM
  - Docker images can be *forked* to apply changes on existing ones

Containerisation – Limitations

- LCG Build processes may take up to 100GB
  - Difficult to manage containers running such processes
- Reproducibility is complicated
  - All of the data inside a container disappears once the container shuts down
- Basic monitoring
  - Docker stats command offer very basic information about containers
- No images for MacOS

Containerisation

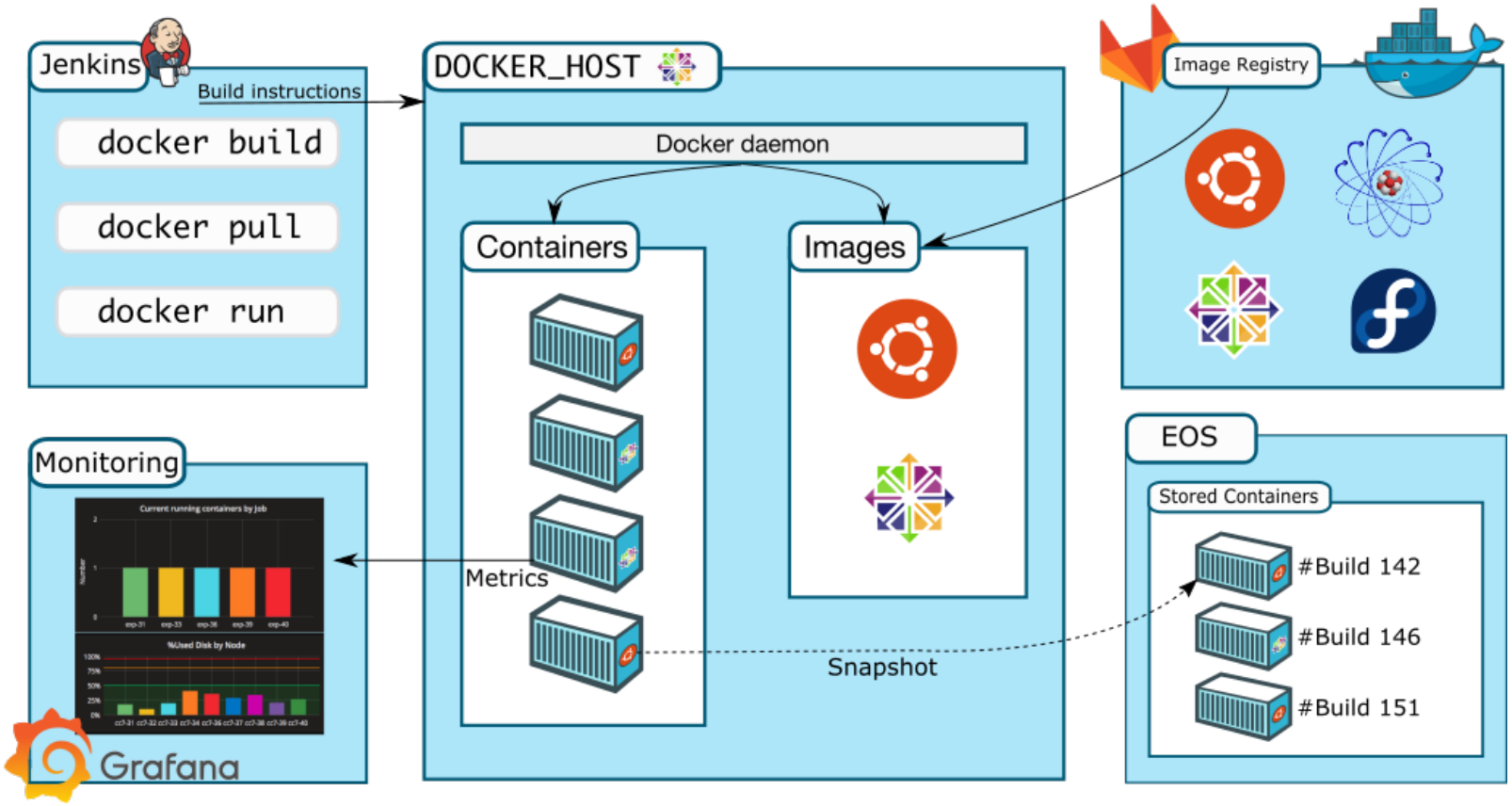
- Migration from build nodes to Docker containers:
  - Nightly builds – Production
  - Releases – Prepared for production

Configuration Matrix		native	gcc62binutils	gcc7binutils
lcg_docker_cc7	Release	●	●	●
	Debug	●	●	●
lcg_docker_slc6	Release	●	●	●
	Debug	●	●	●
lcg_docker_ubuntu16	Release	●	●	●
	Debug	●	●	●

Image: Build #143 lcg_ext_dev4_archdocker

From Build nodes to Containers

OVERVIEW



From Build nodes to Containers

INFRASTRUCTURE

- Bunch of 10 new powerful CentOS7 added
- Fully dedicated to act just like Docker hosts
  - Free resources for the non-LCG builds (ROOT, Geant, …)
  - Easier to manage

From Build nodes to Containers

CONFIGURATION

- New Puppet configuration for Docker hosts:
  - Docker software and configuration
  - Ccache (directory and configuration)
  - CVMFS
  - Statistical scripts
- Existing Jenkins job configurations (defined variables, shell script actions, triggered jobs, …) wrapped into scripts
  - These scripts are injected inside the container
  - Result: Same builds as for build nodes

From Build nodes to Containers

IMAGES

- Container images hosted in a Gitlab Repository registry
  - Private alternative to Dockerhub
- Dockefiles hosted in the same repo
- Images based on Puppet configuration used for build nodes
  - Same base packages
  - Ccache configuration + Code to export/recover to/from EOS
- Available images:
  - Ubuntu16
  - SLC6
  - CentOS7
  - Fedora25

```
$ docker run -it gitlab-registry.cern.ch/sft/docker:cc7
```

From Build nodes to Containers

WORKFLOW

1. Jenkins Server -> Connect to any available Docker host
2. Launch container with code to execute

From Build nodes to Containers

WORKFLOW

1. Jenkins Server -> Connect to any available Docker host
2. Launch container with code to execute

```
$ docker run -e WORKSPACE=$WORKSPACE
           -e ENVIRONMENT_VARIABLES
           -u sftnight
           --name $NAME
           --hostname $HOSTNAME-docker
           --cpus=$DOCKER_CPUS
           -v BIND_MOUNTS_FROM_HOST
         gitlab-registry.cern.ch/sft/docker:$DOCKER_IMAGE
         bash -c "/lcgjenkins/runall-docker.sh $BUILDTYPE $COMPILER $LCG_VERSION"
```

From Build nodes to Containers

WORKFLOW

1. Jenkins Server -> Connect to any available Docker host
2. Launch container with code to execute
   - Trigger other jobs once finished
3. Export container to EOS (`tar` format)
4. Clean up node
5. To recover a container stored in EOS, from *ANY* docker host:

```
$ ./docker_scripts/restore_build.sh $NAME
```

Containerisation – Optimisations applied

IMPROVED BUILD TIME

- Context
  - Building same packages every night
  - (Usually) Few changes from one day to another
- **Solution**: CCache
  - The more we build the same the more CCache helps
  - Different builds from different jobs share compiling information
  - CCache directory persists on the Docker host

Containerisation – Optimisations applied

REDUCE RUNNING OUT OF SPACE PROBLEMS

- Context
  - Problem of full disk nodes causing failed builds
- Solution
  - LCG *latest* released every week
  - Clean up build artifacts and containers once finished.
  - Daily cronjob to ensure removal of possible remaining containers
  - Monthly cronjob to ensure on removal of docker images

Containerisation – Optimisations applied

KEEP REPRODUCIBILITY OF BUILDS

- Context
  - Containers are terminated when the build process finishes.
  - Once terminated, there is no way to manually test the build to find out problems

- Solution
  - Export containers before removal

Containerisation – Optimisations applied

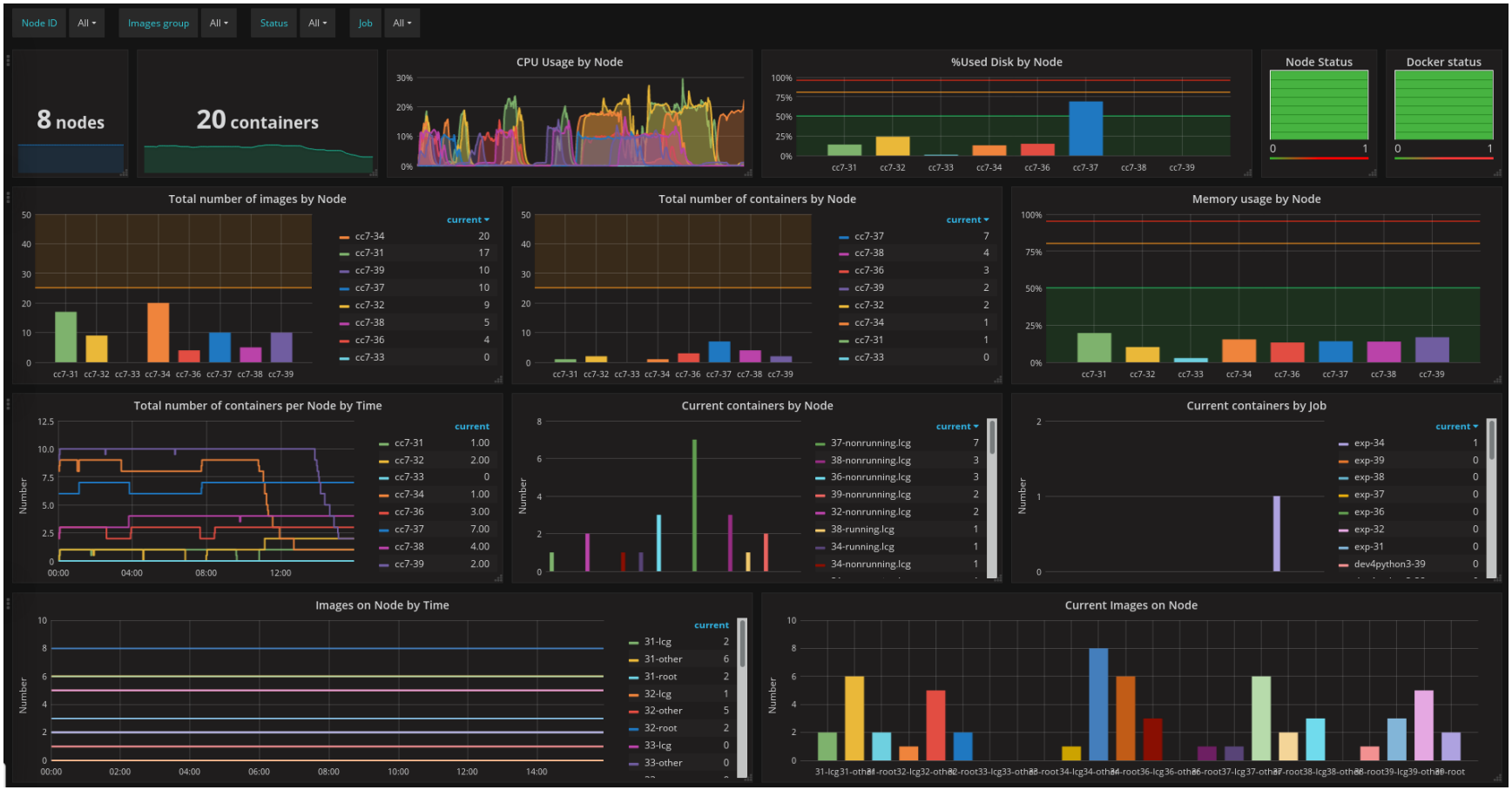
REDUCE EXPORT PROCESS LATENCY

- Context
  - Exporting large containers images may take hours
    - Docker host blocked during this time
    - Bottleneck in jenkins
- Solution
  - Reduce size of containers → LCG latest releases
  - Export only containers most likely to be recovered: Only failed builds

Containerisation – To further optimise

- Integration of Graphdriver plugin (WIP)
  - Although there is no benefit when uploading, download time may be drastically reduced
- Find out best directory policy for CCache
  - Same CCache directory for every build
  - Different CCache directories (for dev3, for dev4, …)

Containerisation - Monitoring with Grafana & Graphite



Live demo

LCGDocs, documentation website

- Main goals:
  - Gather documentation from different sources (Evernote, notebooks, mails, …)
  - Facilitate the job for newcomers

- Based on Gitbook
- EOS hosted (/eos/project/l/lcgdocs)

LCGDocs, documentation website

Type to search

Published with GitBook

Adding hosts to the jenkins infrastructure

Docker hosts

1. Follow the steps in Create a new virtual machine (docker host) to create a new OpenStack virtual machine.
2. Don't forget to create the new volume and attach it.
3. Add a ssh key to access using `sftnight` credentials.
    - From any existing node, use `scp` to copy it over the same path in the new vm
4. Add it as a new Jenkins node:
    - Go to Jenkins nodes
    - Click on *New node*
    - Add a node name
    - Copy it from an existing one, i.e. `lcgapp-centos7-x86-64-31`
    - Modify the argument in the *Launch command* to the correct hostname
    - Set up the correct bunch of labels

http://lcgdocs.web.cern.ch/lcgdocs/

Thank you

- Big thanks to Patricia Mendez and Pere Mato
- Thanks to everyone else on EP-SFT

What's next?

- Fellowship in EP-SFT (1st November)
- Working for the FCC experiment
  - Building infrastructure
  - Supervisor: Benedikt

- *Really excited to continue being part of this team* 😁