Using CVMFS to distribute LCG Releases

Johannes Heinz
CERN, EP-SFT-SPI

CernVM Workshop 2019
03/06/2019

Introduction to LCG Releases

LCG Releases

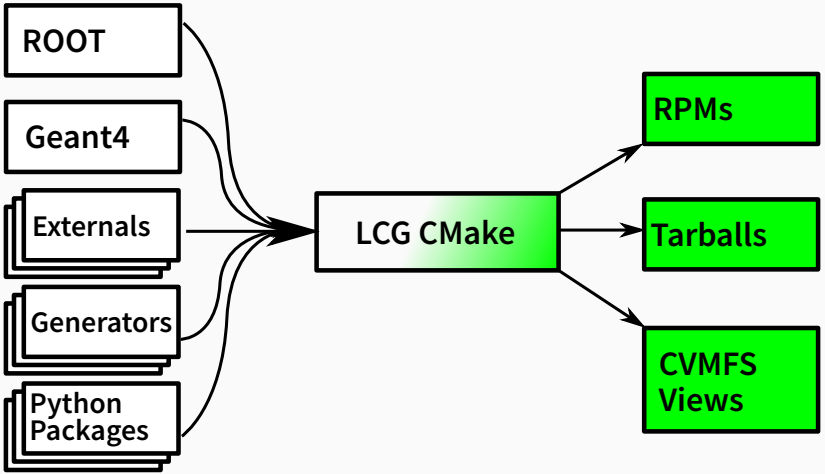
SPI group mandate:

- Service group for experiments
- Solve common tasks only once
- Provide easy access to commonly used programs
- Simplify setup of work environments

Clients:

- ATLAS
- LHCb
- SWAN
- IT-DB-SAS
- BE department

LCG Releases: Schematic



LCG Releases: View

Use the view of the latest release on `lxplus.cern.ch`

```
source /cvmfs/sft.cern.ch/lcg/views/LCG_95a/
       x86_64-centos7-gcc8-opt/setup.sh
```

Look up releases, packages and platforms: LCG Info

LCG Releases: SWAN

Configure Environment

Software stack more...

95a

Platform more...

CentOS 7 (gcc7)

Environment script more...

e.g. $CERNBOX_HOME/MySWAN/myscript.sh

Number of cores more...

2

Memory more...

8 GB

Spark cluster more...

None

LCG Releases: Current Dimensions

- **Architecture:** x86_64
- **Operating system:** Scientific Linux 6, CentOS 7, Ubuntu 16.04 LTS, Ubuntu 18.04 LTS
- **Compiler:**
  - **GCC:** native, 6.2.0, 7.3.0, 8.2.0
  - **Clang:** 7.0.0, 8.0.0
- **Flags:** sse3, avx2, fma, CUDA support
- **Build type:** Debug, Optimized
- **Python version:** 2.7, 3.6
- **ROOT version:** master, 6.18
- **Toolchains:** BE (Beams), Experiments

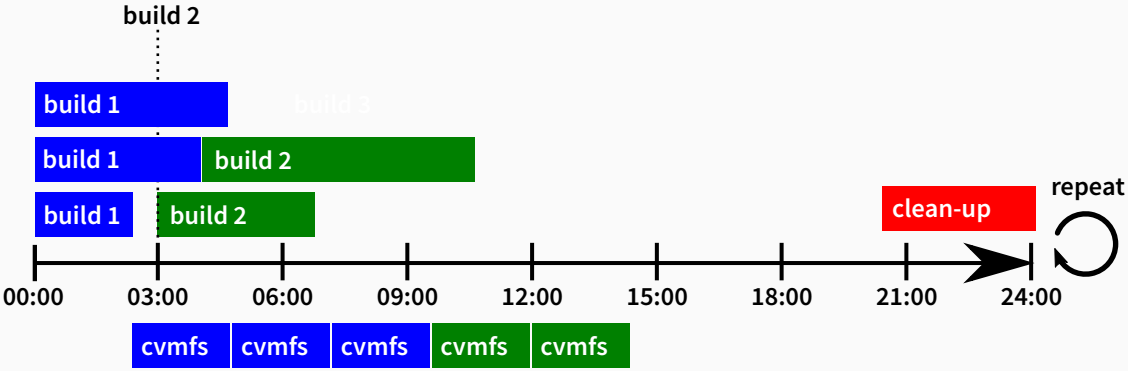
LCG Releases: Future Dimensions

- **Architecture:** x86_64, **aarch64** (ARM), **PowerPC**
- **Operating system:** Scientific Linux 6, CentOS 7, Ubuntu 16.04 LTS, Ubuntu 18.04 LTS, **Fedora ?**
- **Compiler:**
  - **GCC:** native, 6.2.0, 7.3.0, 8.2.0, **9.1.0**
  - **Clang:** 7.0.0, 8.0.0
- **Flags:** sse3, avx2, fma, CUDA support
- **Build type:** Debug, Optimized
- **Python version:** 2.7, 3.6
- **ROOT version:** master, 6.18
- **Toolchains:** BE (Beams), **ATLAS**, **LHCb**, ...

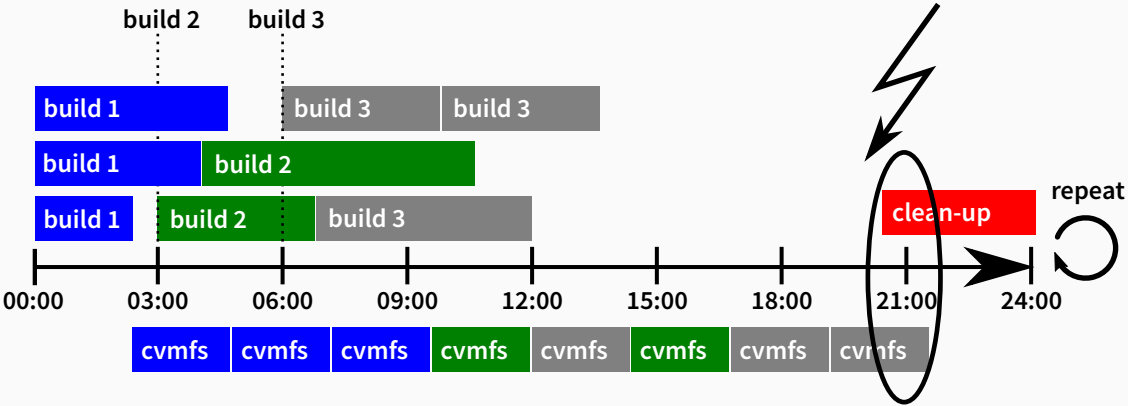
LCG Releases: CVMFS usage

- **Possible combinations per night:** 2 304
- **Actual platforms per night:** 52 (37 published to CVMFS)
- **Packages per Platform:** Typically more than 390
- **Repositories:**
  - `sft.cern.ch` – CVMFS 2.6.0, 67 086 catalogs
  - `sft-nightlies.cern.ch` – CVMFS 2.6.0, 19 299 catalogs
- **Available releases:** From **LCG_71** to **LCG_95a** (plus nightly builds)

LCG CMake: Timeline



LCG CMake: Timeline



New approaches

New approaches to Building software

Current approach:

- Puppetized VM builds for releases, Docker images for nightly builds
- Bind mount CVMFS from host system (CERN CentOS 7)
- Irregular problems while publishing to EOS as staging area
- Active waiting for CVMFS transaction (blocking slots in Jenkins)
- Assumptions about stratum 1 state

New approach:

- Move to a unified approach for all builds $\Rightarrow$ **Kubernetes orchestration**
- Use CSI driver for read access
- Test CERN S3 as staging area
- Asynchronous deployment using `conveyor`
- Use notification about nightly builds to trigger LCG Info update and tests

New approaches to Publication

- Reference use case for new CVMFS features
- Future **HSF** (HEP Software Foundation) repositories:
  - `sw.hsf.org` – Future releases repository
  - `sw-nightlies.hsf.org` – Future nightly repository
  - `test-sw-nightlies.hsf.org` – Experimental repository for testing

Where to limit the lease path?


```
/cvmfs/sft-nightlies.cern.ch/lcg/nightlies/ # base (1)
  dev4/                    # toolchain (5)
  Mon/                     # day (1)
  Python/                  # package (390)
  2.7.15/                  # version (1+)
  x86_64-slc6-gcc8-opt/    # platform (19)
```

$\Rightarrow$ **Take lease at platform level**

Usage of conveyor

- Publish each package separately
- Collect publication IDs

```
id=$(conveyor submit
    --repo test-sw-nightlies.hsf.org
    --lease-path ${PACKAGE_VERSION_PLATFORM}
    --payload "script|
       https://${S3_BUCKET}/${LCG_TRANSACTION_SCRIPT}
                          ?checksum=sha1:${SHA1_HASH}|
       https://${S3_BUCKET}/${$package}.tgz"
    | tail -1
    | jq -r '.job_id')
ids="$ids $id"
```

Usage of conveyor

- Create view with a dependency to each package

```
conveyor submit
  --repo test-sw-nightlies.hsf.org
  --deps "$ids"
  --wait
```

Read from CVMFS within Kubernetes

Resources

Gain read access to CVMFS within Kubernetes using the CSI driver:

- CernVM-FS CSI driver
- CERN IT: Cloud Docs
- Kubernetes documentation: Run a job to completion

At CERN we can take advantage of the pre-configured Kubernetes plugin and the support by IT

Setup Kubernetes with CVMFS on `lxplus-cloud.cern.ch`


```
openstack coe cluster create my-kubernetes-cluster \
    --cluster-template kubernetes-1.13.3-1 \
    --master-flavor m2.medium \
    --node-count 3 --flavor m2.xlarge \
    ...
    --labels cvmfs_csi_enabled=True \
    --labels cvmfs_csi_version=v0.3.0 \
    --labels cvmfs_tag=qa \
    ...
```

Define Kubernetes `job.yaml`


```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-cvmfs-sft
provisioner: csi-cvmfsplugin
parameters:
  repository:  sft.cern.ch
```

Define Kubernetes `job.yaml`


```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-cvmfs-sft-pvc
spec:
  accessModes:
  - ReadOnlyMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-cvmfs-sft
```

Define Kubernetes `job.yaml`


```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: cvmfstest
spec:
  template:
    # (on next slide)
  backoffLimit: 4
```

Define Kubernetes `job.yaml`


```
      containers:
      - name: cvmfs-container
        image: cern/cc7-base
        command: ["ls", "-halt", "/cvmfs/sft.cern.ch/lcg"]
        volumeMounts:
        - mountPath: /cvmfs/sft.cern.ch
          name: sft-pvc
      restartPolicy: Never
      volumes:
      - name: sft-pvc
        persistentVolumeClaim:
          claimName: csi-cvmfs-sft-pvc
          readOnly: true
```

Execute the job and get results:

Send job configuration to Kubernetes:

```
kubectl apply -f ./job.yaml
```

Get pod that runs the job:

```
kubectl describe jobs/cvmfstest
```

Get the output from the job:

```
kubectl logs cvmfstest-mhvng
```

Summary

Challenges:

- Growing number of supported platforms (new compilers, new OSs, ARM support)
- Diversification of experiments' requirements
- Sequential publication to CVMFS too slow
- Build environment consistency hard to control (puppetized VMs vs. Docker images)

Approaches:

- New toolchains for each experiment (ATLAS, LHCb)
- Use multiple release machines with clear defined lease paths
- Use exclusively Docker to define the build environment
- Use Kubernetes to make better use of our build resources