



# Framework for User Application Software with GitLab and Docker

Maciej Wyżliński

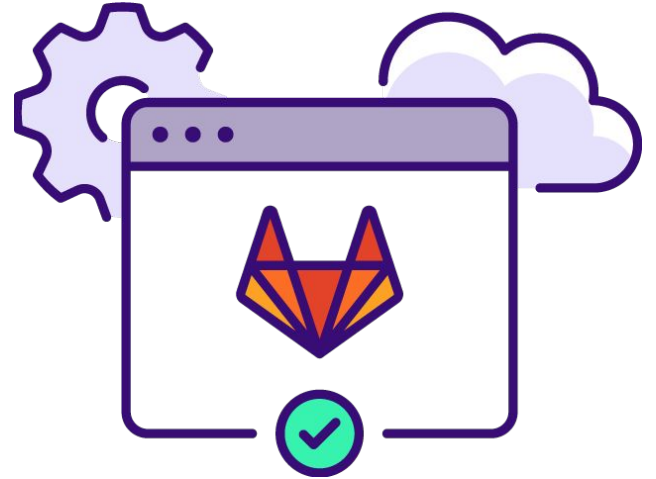


# Outline

- Introduction
- A generic Docker image with CentOS root file system and cross compiler
- Example usage for building ATLAS L1CT software
  - Host configuration
  - Building software
  - Deployment

## Why use Continuous Integration (CI)?

- Automate manual tasks
- Identify breaking changes and bugs
- CI scripts are self-documenting
- Ensure build scripts compatibility with the newest code versions
- It's simple!



# CentOS root file system and cross compiler

The screenshot shows the GitLab interface for the 'centos-roots' repository. At the top, the repository name 'centos-roots' is displayed with a lock icon and 'Project ID: 108452'. To the right, there are buttons for 'Star' (0) and 'Fork' (0). Below this, statistics show '53 Commits', '1 Branch', '0 Tags', '215 KB Files', and '135.6 MB Storage'. A progress bar is visible, and the current branch is 'master'. Navigation buttons include 'History', 'Find file', 'Web IDE', and 'Clone'. A recent commit by 'Maciej Andrzej Wyzlinski' is shown with the message 'Add json and json-devel' and a green checkmark. Below the commit, there are buttons for 'README', 'CI/CD configuration', 'Add LICENSE', 'Add CHANGELOG', and 'Add CONTRIBUTING'. A table lists repository contents with columns for 'Name', 'Last commit', and 'Last update'. The 'README.md' file is highlighted, and its content is partially visible, showing the repository name and a 'pipeline passed' status.

centos-roots  
Project ID: 108452

53 Commits | 1 Branch | 0 Tags | 215 KB Files | 135.6 MB Storage

master | centos-roots / +

History | Find file | Web IDE | Clone

**Add json and json-devel**  
Maciej Andrzej Wyzlinski authored 3 days ago

97a689d7

README | CI/CD configuration | Add LICENSE | Add CHANGELOG | Add CONTRIBUTING

Add Kubernetes cluster

Name	Last commit	Last update
centos	Add json and json-devel	3 days ago
cross	Add cmake to path	2 weeks ago
.gitlab-ci.yml	Update toolchains and variables	3 weeks ago
README.md	Update mkrootfs.py	2 weeks ago

README.md

**centos-roots**

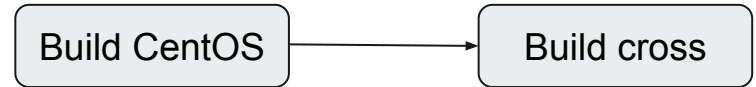
pipeline passed

public <https://gitlab.cern.ch/soc/centos-roots> - starting from CentOS 7 image



# CentOS root file system and cross compiler

- CentOS
  - Download and install qemu
  - Download and cross install a minimal CentOS 7 root file system using dnf
  - Install some additional packages (e.g. python3)
- Cross
  - Download and build cross compiler (gcc 8.3) from source
  - Install a cmake toolchain file for the selected architecture



# Docker images

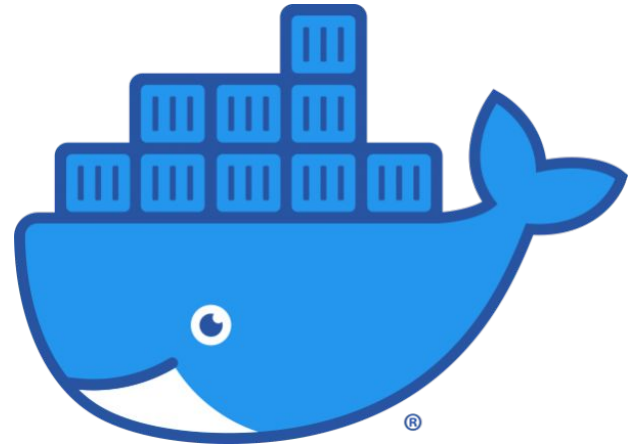
Images to be downloaded from:

- [gitlab-registry.cern.ch/soc/centos-rootfs/cross-aarch64](https://gitlab-registry.cern.ch/soc/centos-rootfs/cross-aarch64)
- [gitlab-registry.cern.ch/soc/centos-rootfs/cross-armv7hl](https://gitlab-registry.cern.ch/soc/centos-rootfs/cross-armv7hl)


They provide:

- A complete CentOS 7 root file system image with basic packages installed
- A script to install or remove packages
- A cmake toolchain file for the chosen architecture
- Environmental variables (eg. CROSS\_CC)

Docker images are ready to be used for user software building!





# Specific application - as example ATLAS MUCTPI software

**M Muctpi Ci**  Project ID: 105452 🔔 ★ Star 0 🍴 Fork 0

🔗 331 Commits 🌿 1 Branch 🏷️ 0 Tags 📁 594 KB Files 💾 583.4 MB Storage

master ▼ muctpi\_ci / + ▼ History Find file Web IDE 📄 Clone ▼

 **Install json-devel** 🚫 f2f11aff   
Maciej Andrzej Wyzlinski authored 3 days ago

📖 README 📁 CI/CD configuration 📄 Add LICENSE 📄 Add CHANGELOG 📄 Add CONTRIBUTING  
📁 Add Kubernetes cluster

Name	Last commit	Last update
📁 centos	Install json-devel	3 days ago
📁 l1ct	Cleanup	2 weeks ago
📁 scripts	Fix sed	2 weeks ago
📁 tdaq	Revert "Don't use package file for aarch64"	3 days ago
🔥 .gitlab-ci.yml	Upload logs as artifacts	1 week ago
🔥 .gitmodules	Fix	1 month ago
📖 README.md	Update readme	2 weeks ago

[https://gitlab.cern.ch/atlas-l1ct/muctpi\\_ci](https://gitlab.cern.ch/atlas-l1ct/muctpi_ci)

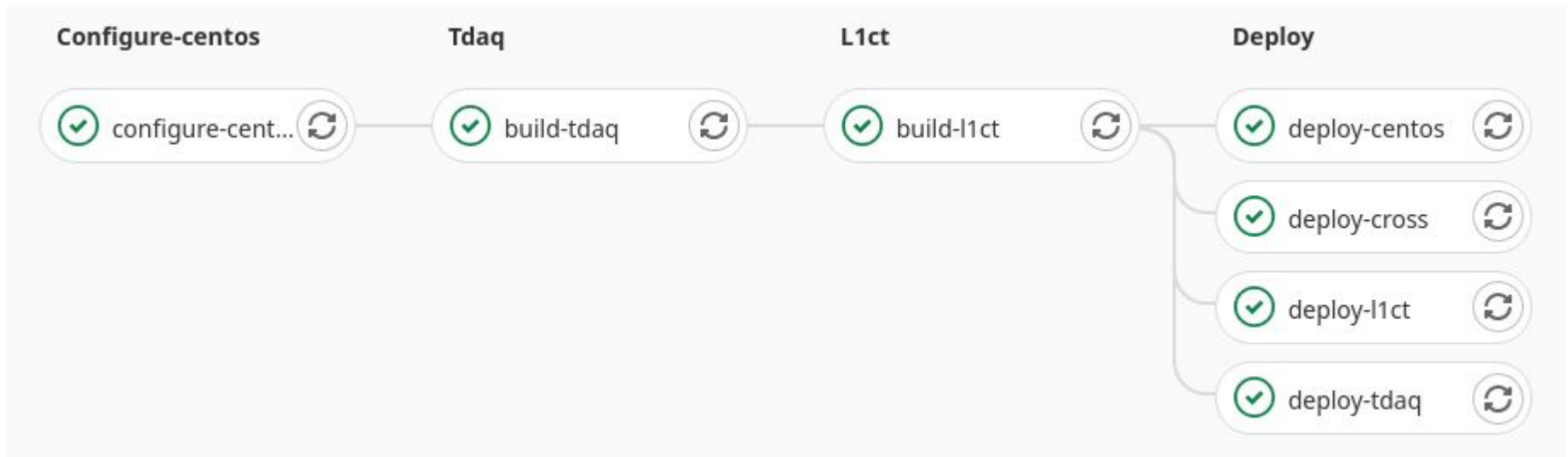


# Goals

- Provide a unified way of installing the following on multiple hosts:
  - CentOS root file system
  - Cross compiler
  - TDAQ and L1CT executables and libraries
- Run nightly builds to check for breaking changes
- Run in an isolated environment (Docker container)
- Builds for both armv7 and aarch64



# Specific application



# Runners

- To create - run command  
“gitlab-runner register”
- Two types of runners
  - Shell runner for building images
  - Docker runner for building software
- For the Docker runner -  
Add “/cvmfs:/cvmfs:ro,shared” to volumes  
list in /etc/gitlab-runner/config.toml

## Runners activated for this project

The screenshot displays three GitLab runners in a list. Each runner entry includes a status indicator (green for active, red for paused), a truncated name, a lock icon, an edit icon, and control buttons. Below the name is the runner's ID and tags.

Runner Name	Status	Runner ID	Tags
m8ua15bb...	Active	pcphese91 #7452	docker, muctpi
vweCX2Xs...	Active	pcphese91 #7451	muctpi, shell
GsUFVWmp...	Paused	pcphl1ct07.cern.ch #7351	docker, muctpi



## Describing a pipeline - example for L1CT job

- image - path to take docker image from
- stage - defines job order
- tags - defines the runner to choose
- script and before\_script - actual commands that are run

<https://docs.gitlab.com/ee/ci/yaml/>

```
build-l1ct:
  image: $SCI_REGISTRY_IMAGE/muctpi-centos-$SMUCTPI_ARCH
  stage: l1ct
  tags:
    - muctpi
    - docker
  dependencies:
    - build-tdaq
  artifacts:
    paths:
      - l1ct/muctpi-cmake/installed/
    expire_in: 3 days
  variables:
    GIT_STRATEGY: clone
  before_script:
    - scripts/checkout_release.sh
  script:
    - cp -r work /
    - ./l1ct/build.sh
```

# CentOS root file system configuration - specific to ATLAS L1CT

```
1 ARG ARCH
2 FROM gitlab-registry.cern.ch/soc/centos-rootfs/cross-$ARCH
3
4 # Build and install i2c-tools
5 RUN git clone git://git.kernel.org/pub/scm/utils/i2c-tools/i2c-tools.git && cd i2c-tools \
6     export CC=$CROSS_CC \
7     export CFLAGS="-O2 -pipe --sysroot=${SYSROOT}" \
8     make -j`nproc` && make install
9
10 # Configure system
11 COPY configure.py /usr/bin
12 RUN configure.py command ldconfig
13 RUN configure.py root-passwd ██████████
14 RUN configure.py useradd mawyzlin -u 134875 -g 1307 --groupname zp --root
15
16 # Install host dependencies
17 RUN yum -v clean expire-cache
18 RUN yum install -y xerces-c xerces-c-devel gcc-c++
19
20 # Build hwcompiler
21 COPY hwcompiler /work/hwcompiler
22 RUN cd /work/hwcompiler && make install
```



## Uploading the Docker image to registry

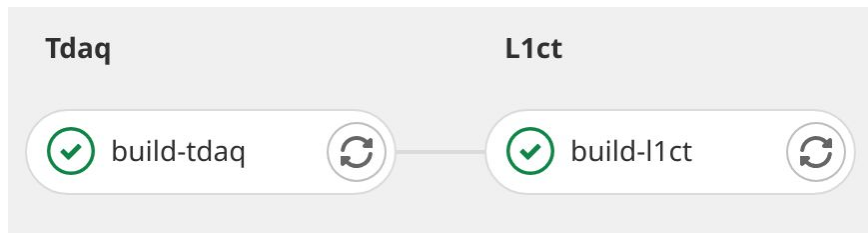
```
configure-centos:
  stage: configure-centos
  tags:
    - muctpi
    - shell
  variables:
    GIT_STRATEGY: clone
    GIT_SUBMODULE_STRATEGY: normal
  script:
    - echo $SCI_REGISTRY_PASSWORD | docker login -u "$SCI_REGISTRY_USER" --password-stdin $SCI_REGISTRY
    - docker build -t $SCI_REGISTRY_IMAGE/muctpi-centos-$SMUCTPI_ARCH centos --build-arg ARCH=$SMUCTPI_ARCH
    - docker push $SCI_REGISTRY_IMAGE/muctpi-centos-$SMUCTPI_ARCH
```



# User Application Software

```
1 export MUCTPI_PYTHON_HOME=${SYSROOT}/usr
2 export MUCTPI_PYTHON_VERSION=3.6m
3 export CMAKE_PREFIX_PATH=/work/tdaq-common/tdaq-common-${TDAQ_VERSION}/cmake_tdaq/cmake:/work
4
5 set -e
6
7 DIRNAME=$(dirname $(readlink -f $0))
8
9 cd $DIRNAME/muctpi-cmake
10
11 mkdir -p build && cd build
12 cmake -D CMAKE_TOOLCHAIN_FILE=$TOOLCHAIN ..
13
14 make -j`nproc`
15 make install
```

## Passing artifacts between jobs



```
artifacts:  
  when: always  
  paths:  
    - work/tdaq  
    - work/tdaq-common  
  expire_in: 3 days
```



# Deployment

- Install to selected hosts
- Rsync for installation
- SSH private/public key authentication
  - Key stored as variable in GitLab repository
- Separate deployment stages
  - When one component fails to install, others can succeed
  - Allow running partial pipelines (Skipping CentOS and cross compiler installation)
- Results - a full installation of:
  - CentOS root file system
  - Cross compiler
  - TDAQ and L1CT software executables and libraries
- `muctpi_setup.sh` script
  - Sets all the paths and environmental variables to allow running software on the target (SoC)
  - Prepares the development environment to recompile software locally on host (PC)



# Pipeline triggers

- By default pipeline runs with every push
- We can set rules to control this behaviour:

```
rules:  
- if : '$CI_PIPELINE_SOURCE == "push"'  
  when: never  
- if : '$CI_PIPELINE_SOURCE == "trigger" || $CI_PIPELINE_SOURCE == "schedule"'  
  when: on_success
```

- Pipeline can be triggered using HTTP API with variables:

```
curl --request POST \  
  --form token=TOKEN \  
  --form ref=master \  
  "https://gitlab.example.com/api/v4/projects/9/trigger/pipeline"
```

Trigger token: dbea











Trigger variables:

Hide values

FULL_PIPELINE	True
HOSTS	pcph1ct10
L1CT_BRANCH	99-00-00
L1CT_VERSION	99-00-00
MUCTPI_ARCH	aarch64
MUCTPI_VERSION	99-00-00
TDAQ_COMMON_VERSION	99-00-00
TDAQ_VERSION	99-00-00



# Nightly builds

Description	Target	Last Pipeline	Next Run	Owner	
Nightly build - armv7hl	Y master	 #2275650	in 10 hours	 Maciej Andrzej Wyzlinski	  
Nightly build - aarch64	Y master	 #2277248	in 10 hours	 Maciej Andrzej Wyzlinski	  



# Summary

- Generic docker images (CentOS + gcc) to base user pipelines on
- Possibility to run builds for different architectures (armv7 and aarch64)
- A full installation on desired hosts, consisting of
  - CentOS root file system
  - Cross compiler
  - TDAQ and L1CT artifacts
- Script for setting up the development environment (PC), or deployment (SoC)
- Nightly builds and API triggers



## Next steps

- Petalinux
  - PMU firmware
  - FSBL
  - U-Boot
  - Device-tree
  - Kernel
- Provide support to select newer gcc (gcc9?) versions



**Questions?**