

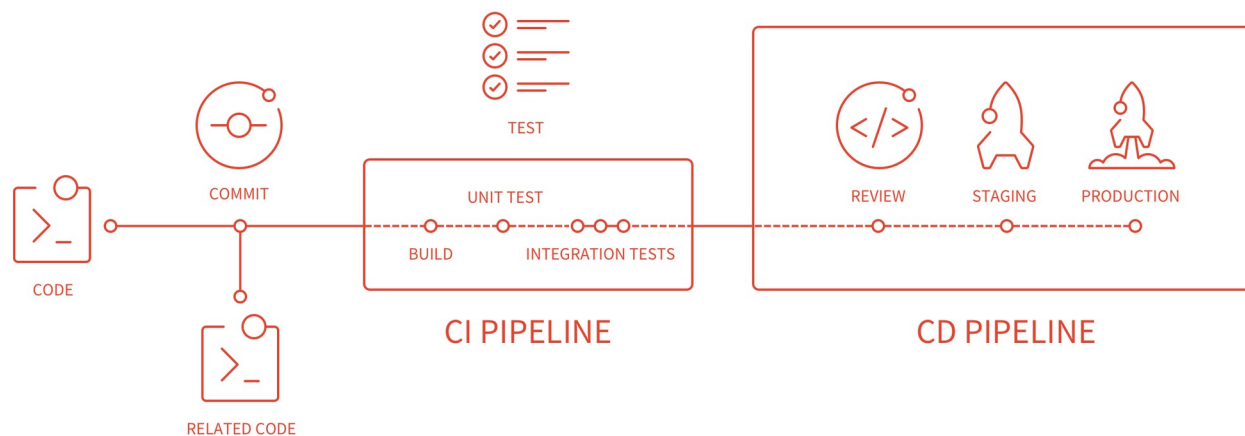
Common Docker images and Gitlab runners for FPGA development

Adrian Byszuk / SY-EPC-CCE
23 November 2022

- Gitlab CI basics
- Docker images of FPGA EDA tools
- Gitlab runners
- ATS-IT proposal for common CERN-wide Gitlab CI infrastructure for FPGA development
- Questions?

Gitlab CI/CD basics

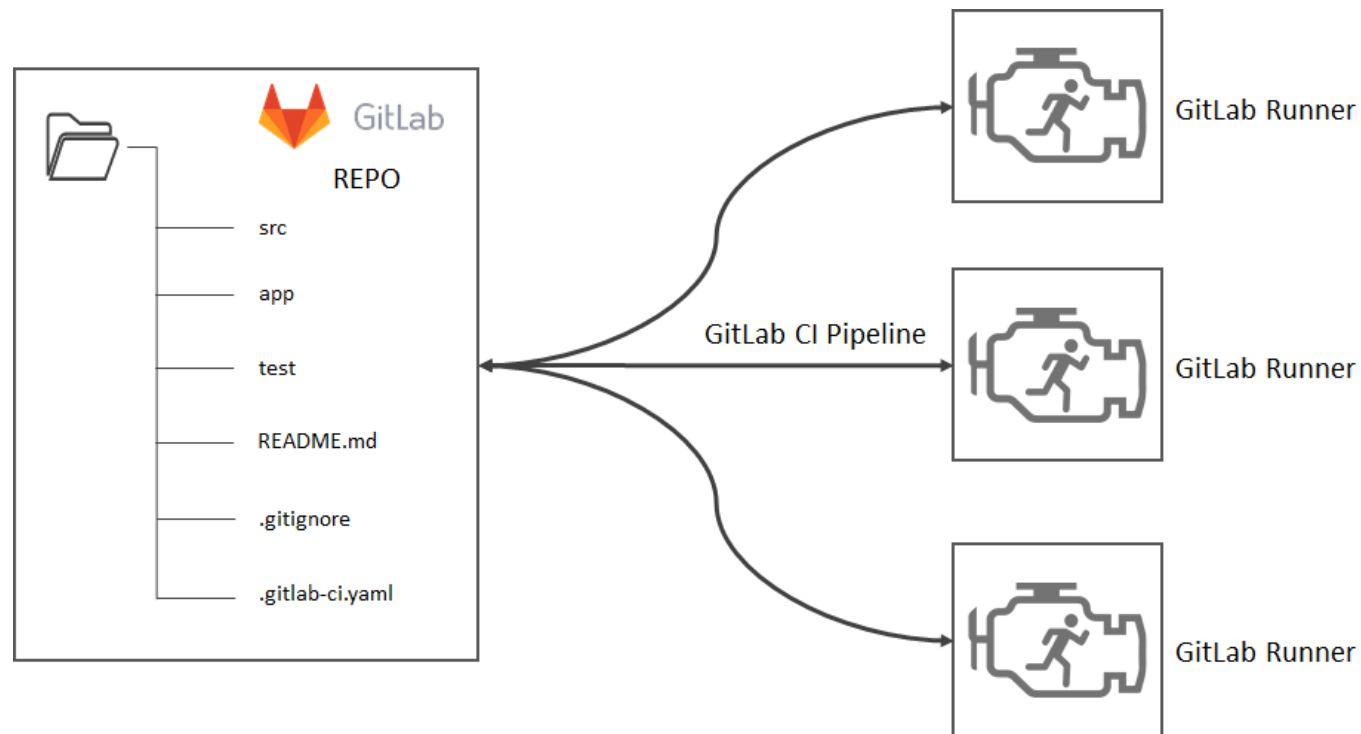
Gitlab CI flow



- When developing software/firmware developers make & push changes often, even multiple times a day
- Each change can introduce a bug
- Ideally you should run test suite and build after each change
- Gitlab CI does exactly that
- „**Pipeline**” is started after each „git push”
- **Pipeline** can execute many **jobs**: tests, build project, generate documentation (Gitlab pages)

Gitlab CI elements

- Two basic elements: **runner** and **Docker image***
- **Gitlab runner** is a computer (physical, VM) that will execute CI **jobs** from a **pipeline**
- **Runner** must be registered to a Gitlab **project**. There can be multiple runners
- **Runner** will start a **Docker container** from the **image** and execute **job** within it
- (**Docker images** aren't exactly needed for runners, but are recommended)



Docker images of FPGA EDA tools

Dockerfiles for HDL EDA software

pipeline skipped

- Dockerfiles for HDL EDA software
- Usage instructions
 - Pulling an image
 - Using images in CI
 - Using images for interactive work
 - Additional image information
 - ISE
 - Vivado/Vitis
 - Xilinx post-release patches
 - Petalinux
 - Petalinux limitations
 - Modelsim
 - QuestaSim
 - Aldec Riviera-PRO
 - GHDL
 - Troubleshooting
 - Cannot connect to the docker daemon
 - Cannot pull the image
 - No space left on device
 - Warning: setting ADDR_NO_RANDOMIZE failed - Operation not permitted
 - Error: cannot connect to X server :0
 - AFS access
- How does the build work?

We've prepared a set of Docker images for EDA software used in FPGA development:

https://gitlab.cern.ch/cce/docker_build/

- Available software: ISE, Vivado/Vitis, Petalinux, Modelsim, Questa, Riviera-PRO, GHDL
- Multiple tool versions are supported
- Intel Quartus support on a TODO list
- Extensive README.md as a documentation
- Can be used for CI or in interactive mode (including GUI)
- Most images based on CC7/C8/CS8 so they can access AFS
- Prebuilt images can be downloaded from container registry:
https://gitlab.cern.ch/cce/docker_build/container_registry

- ISE version: 14.7
- Vivado versions: 2018.1, 2018.3, 2019.1, 2019.2, 2021.1, 2021.2, 2022.1 (Vitis included starting from v2021.1)
- Petalinux versions: 2018.1, 2019.2, 2021.1, 2021.2, 2022.1
- ...more versions available on demand
- Full install – all FPGA families are supported
- Frequently used additional software included: git, python3, nano, sudo, etc
- Uses CERN license server by default, but user can easily override

```
1 .template_vivado:  
2   stage: build  
3   image: gitlab-registry.cern.ch/cce/docker_build/vivado:2019.2
```

```
# Main CERN Xilinx licence server  
ENV XILINXD_LICENSE_FILE=2112@licenxilinx  
ENV XILINX_PATH=/opt/Xilinx/Vivado/$VERSION  
ENV VITIS_PATH=/opt/Xilinx/Vitis/$VERSION
```

```
before_script:  
| - export XILINXD_LICENSE_FILE=port@server-name  
script:  
| - make
```


Simulators

- Modelsim: 10.7a
- Questa: 2022.1
- Aldec Riviera-PRO: 2021.04
- GHDL: GCC backend, built from fairly fresh master snapshot
- All images include prebuilt Xilinx libraries: UNISIM, UNIFAST etc.
- Libraries available under `/opt`, pointed by relevant environment variable
- Frequently used additional software included: git, python3, nano, sudo
- Uses CERN license server by default, but user can easily override

```
ENV PATH="$PATH:/opt/questasim/linux_x86_64"
ENV MGLS_LICENSE_FILE=1717@lxlicen01:1717@lxlicen02:1717@lxlicen03
ENV LM_LICENSE_FILE=$MGLS_LICENSE_FILE

ENV QSIM_ISE_LIBS="/opt/qsim_ise_libs"
ENV QSIM_VIVADO_LIBS="/opt/qsim_vivado_libs"
```

```
ENV PATH="$PATH:/opt/modelsim/modeltech/linux_x86_64"
ENV MODELTECH="/opt/modelsim/modeltech"
ENV MGLS_LICENSE_FILE=1717@lxlicen01:1717@lxlicen02:1717@lxlicen03
ENV LM_LICENSE_FILE=$MGLS_LICENSE_FILE

ENV MSIM_ISE_LIBS="/opt/msim_ise_libs"
ENV MSIM_VIVADO_LIBS="/opt/msim_vivado_libs"
```

```
ENV ALDEC_LICENSE_FILE=27009@licenaldec
# Enable wait for license feature and define time interval
ENV LICENSE_QUEUE=60
ENV RIVIERAPRO_PATH=/opt/aldec/rivierapro

ENV RIVIERAPRO_ISE_LIBS="/opt/rivierapro_ise_libs"
ENV RIVIERAPRO_VIVADO_LIBS="/opt/rivierapro_vivado_libs"
```

```
# GHDL has separate libraries for different VHDL standards because it doesn't
# support mixing them
ENV GHDL_ISE_LIBS=/usr/local/lib/ghdl/vendors/ise
ENV GHDL_VIVADO_LIBS=/usr/local/lib/ghdl/vendors/vivado
ENV GHDL_ISE_LIBS_93=/usr/local/lib/ghdl/vendors/ise-93
ENV GHDL_VIVADO_LIBS_93=/usr/local/lib/ghdl/vendors/vivado-93
```

How to use it?

Gitlab CI:

- Set the *image* variable in your *gitlab-ci.yml*

```
image: gitlab-registry.cern.ch/cce/docker_build/vivado:2019.2
```

- Some images (*cough*... Vivado ...*cough*) have 80+ GB and take a long time to pull – your CI job may timeout
- Advice – pull all images on your runner during setup and add periodic pull to your *crontab*
- For simulators other than GHDL – please reduce the number of simulation jobs to save on valuable license seats

Interactive mode:

- Login to CERN Docker registry (first time only)

```
docker login gitlab-registry.cern.ch
```

- Pull selected image

```
adrian@pcte247806:~/Downloads$ docker pull gitlab-registry.cern.ch/cce/docker_build/petalinux:2019.2
2019.2: Pulling from cce/docker_build/petalinux
56af636389be: Already exists
e0364be80d32: Pull complete
7b8ed4ee4f44: Downloading [=====] 177.3MB/337.8MB
12b7a8ad4309: Downloading [=>] 213MB/9.791GB
```

- Start the container with „*docker run*”
- README contains an example script used to start the container. Depending on your use case, the default parameter set may not meet your expectations

Feedback is very welcome!

Gitlab runners

Private and public Gitlab runners




Setting up a (**private**) Gitlab runner may be a challenge for teams without Linux administration skills:

- Install the host OS, install the runner, configure executor mode, register with access token etc.
- Constant burden of maintenance

Gitlab server may also provide **shared** runners:

- Set-up and maintained by server administrators (IT team)
- Available to everyone
- Used routinely by software programmers
- Not suitable for FPGA development because of huge Docker images
- ...but why can't we have nice things?

Available specific runners

#6084 (i1sr9SFL)    Remove runner

nucte25535

nucte25535

Available shared runners: 214

#28184 (A9Rzi57L)
privileged-runner-849dcf7888-gnljw
docker-machine docker-privileged

#28338 (J96kBR7K)
default-runner-6cdc677fd4-6srlk
cvmfs docker no-runner-9

#28313 (AbTb4uec)
dockerbuild-runner-7674b56476-nkvbz
docker-image-build no-runner-9

#28352 (gQybaYQu)
default-runner-6cdc677fd4-vdvd7
cvmfs docker no-runner-9

#28312 (jS-FP3Ce)
default-runner-6cdc677fd4-4pjaf
cvmfs docker no-runner-9

ATS-IT proposal for common Gitlab infrastructure
for FPGA development

Shared Gitlab infrastructure provided by IT?



There is an ongoing work on a joint ATS-IT proposal about common Gitlab CI infrastructure for FPGA development!

- **This is still a work-in-progress that's not formally accepted**
- Includes both Gitlab runners and Docker images
- This would bring support level for FPGA workflow on par with software development
- Docker images presented here donated as a basis for official IT images
- IT would prepare a set of shared Gitlab runners, configured for FPGA workflow
- Many technical and organisational challenges to be solved



Many questions...

There are still many challenges and questions that need to be answered:

- VM configuration (many projects need 64+ GB RAM and a lot of disk space, which is unusual)
- Estimate number of potential users
- Impact on available EDA licenses
- Huge size of Docker images creates problems for IT infrastructure
- Cost for organisation
- ...many more

