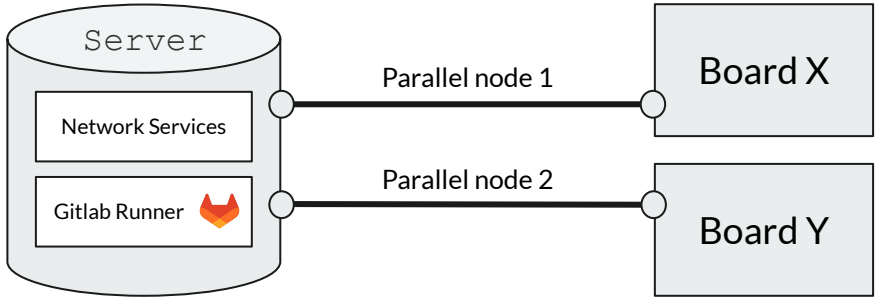
GitLab CI parallel build and testing for Zynq SoC designs

Kareen Arutjunjan
EP-CMD | CMS DAQ group

Acknowledgements:
P. Žejdl & M. Dobson

GitLab



Summary

- Introduction
  - Portfolio
  - The problem
- Gitlab CI
  - The basics of Gitlab CI
- Zynq-BuildSystem CI
  - About Zynq-Buildsystem
- Network-Services
  - Containerized-Network-Services
- Demo
  - Adding a new board to Zynq-Buildsystem CI

Introduction

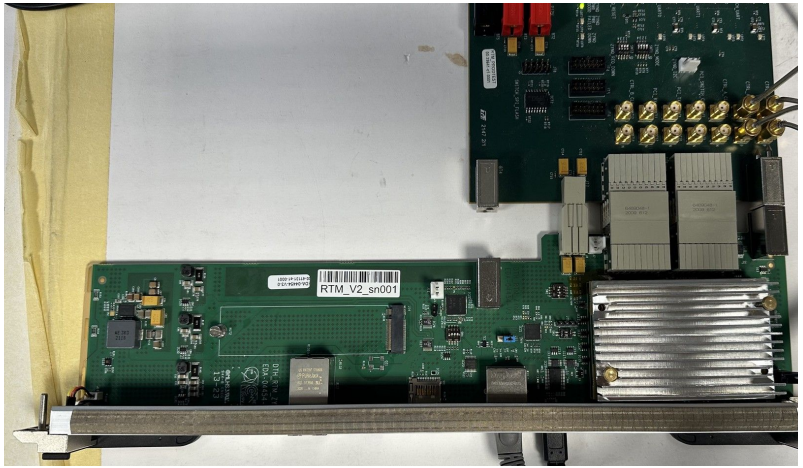
Introduction: Portfolio

The CMS DAQ is currently in the development phase, and therefore needs to support different types of boards at all times.

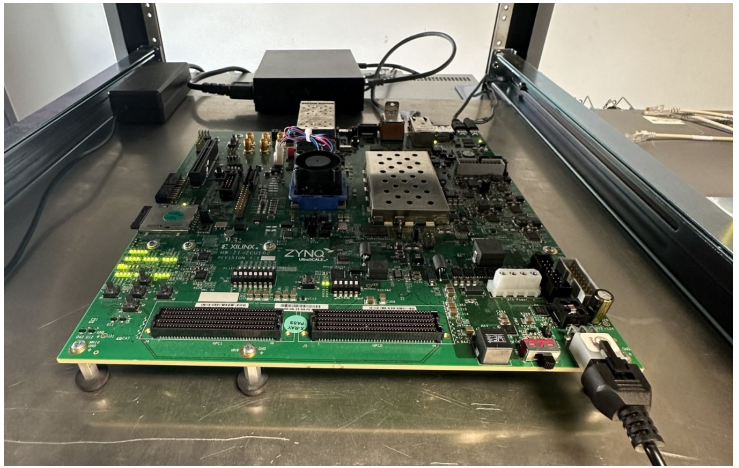
List of boards currently used:

Number	1	2	3	4	5
Board Type	Trenz Mezzanine	ZCU-102	RTM V1	RTM V2	Kria

Introduction: Portfolio



RTMv2 Trenz



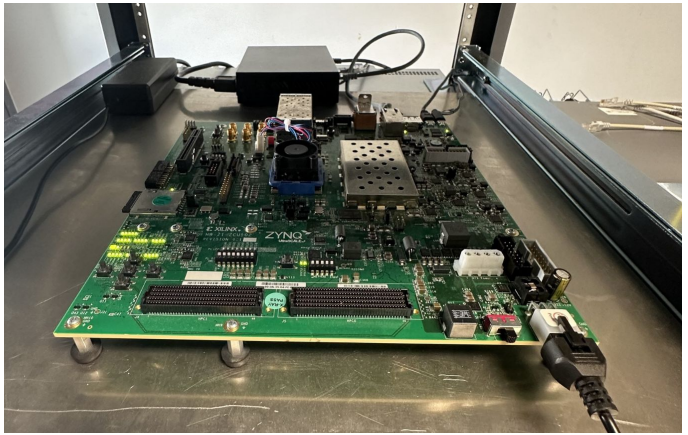
Zynq UltraScale+ MPSoC (ZCU102)

Generating Zynq-Images: The problem

Developing and maintaining embedded systems for Zynq boards presents several challenges:

Efficiency: Manually building firmware images, and root-filesystems is time-consuming.

Testing and Validation: Ensuring that the integrated system works correctly, requires manual interaction, and supervision.



Zynq UltraScale+ MPSoC (ZCU102)

Generating Zynq-Images: A better solution is needed

Gitlab CI

Gitlab CI: What is it?

Gitlab CI - tool that automates the process of building, testing, and deploying software applications.

Gitlab CI Job - a specific task or set of tasks defined in a CI.

Gitlab CI Stage - a logical grouping of one or more jobs within a CI pipeline.

Gitlab Pipeline - sequence of stages. Each stage can include tasks like building code, running tests, etc. Pipelines are defined in a *.gitlab-ci.yml* configuration file.

Gitlab Runner - a process, which is executing jobs, that are defined in CI pipeline.

Gitlab Artifact - an output archive file or directory, which was saved from a job.

Gitlab Parallel Matrix - a process of running a job in parallel with different variable values for each instance of the job.

Gitlab CI: Hello world!

.gitlab-ci.yml

```
stages:
- build # CI Stage

hello-world: # CI Job
 stage: build
 script:
  - echo "Hello world!"
```

CI Pipeline

build

hello-world

CI Job: Output

```
18 $ echo "Hello world!"
19 Hello world!
```

Gitlab CI: Parallel-Matrix - Hello world!

.gitlab-ci.yml

```yaml
stages:
- parallel-build

parallel-job:
 stage: parallel-build
 parallel:
  matrix: # Parallel:Matrix
   - TARGET:
     - "audience"
     - "world"
 script:
  - echo "Hello ${TARGET}!"
```

Gitlab CI: Parallel-Matrix - Conclusion

Enhanced Flexibility

Parallel matrix can be configured to build a project and run tests with variable configuration.

Faster Execution

Parallel matrix enables concurrent execution of multiple jobs, significantly reducing the overall pipeline execution time.

Zynq-BuildSystem CI

Originally designed by Vasileios Amoiridis.

Features added by me (Kareen Arutjunjan):

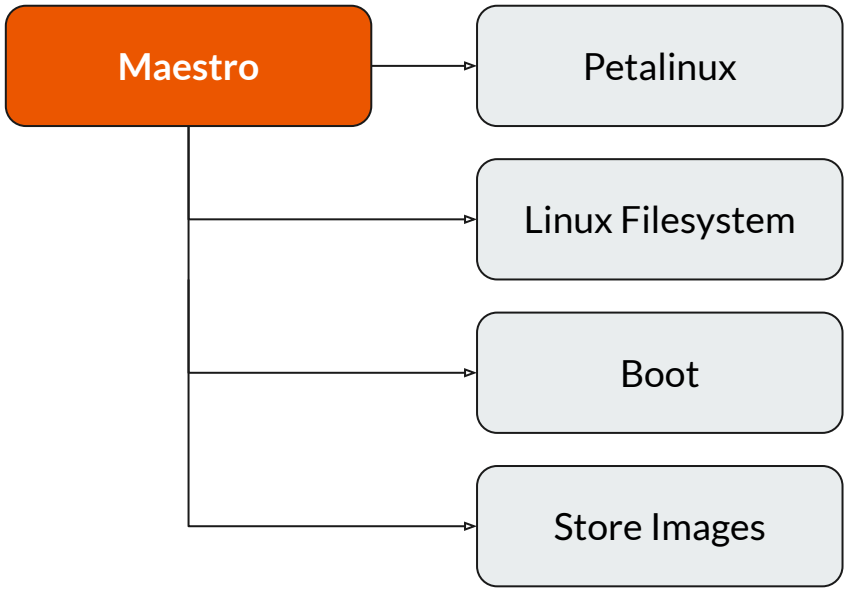
- Multi-board support
- Parallel build

Zynq-BuildSystem CI: Maestro Pipeline

Maestro Pipeline

Acts as an orchestrator to trigger downstream pipelines.

- Sequential Execution
- Dependency Chain

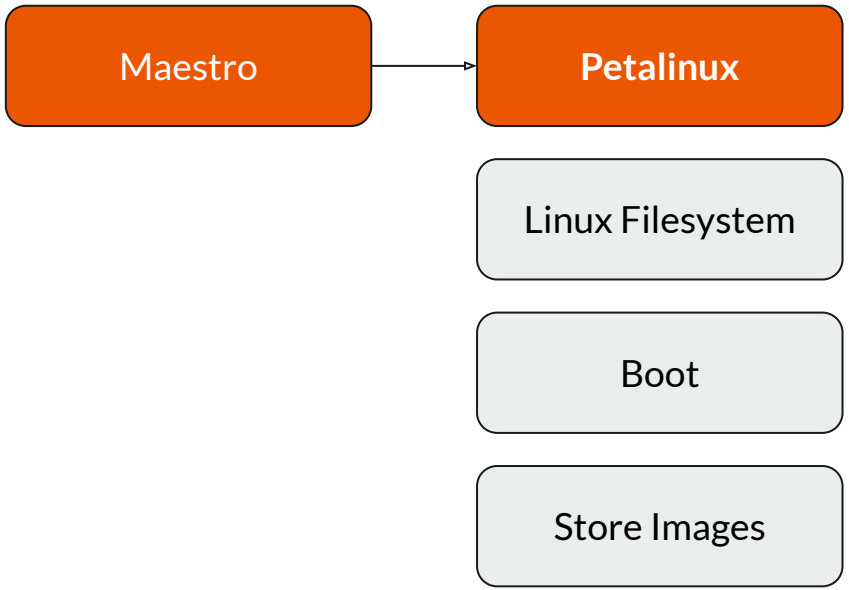


Zynq-BuildSystem CI: Petalinux Pipeline

Petalinux Pipeline

Main responsibility is to Generate Images for Zynq, by using Petalinux framework.

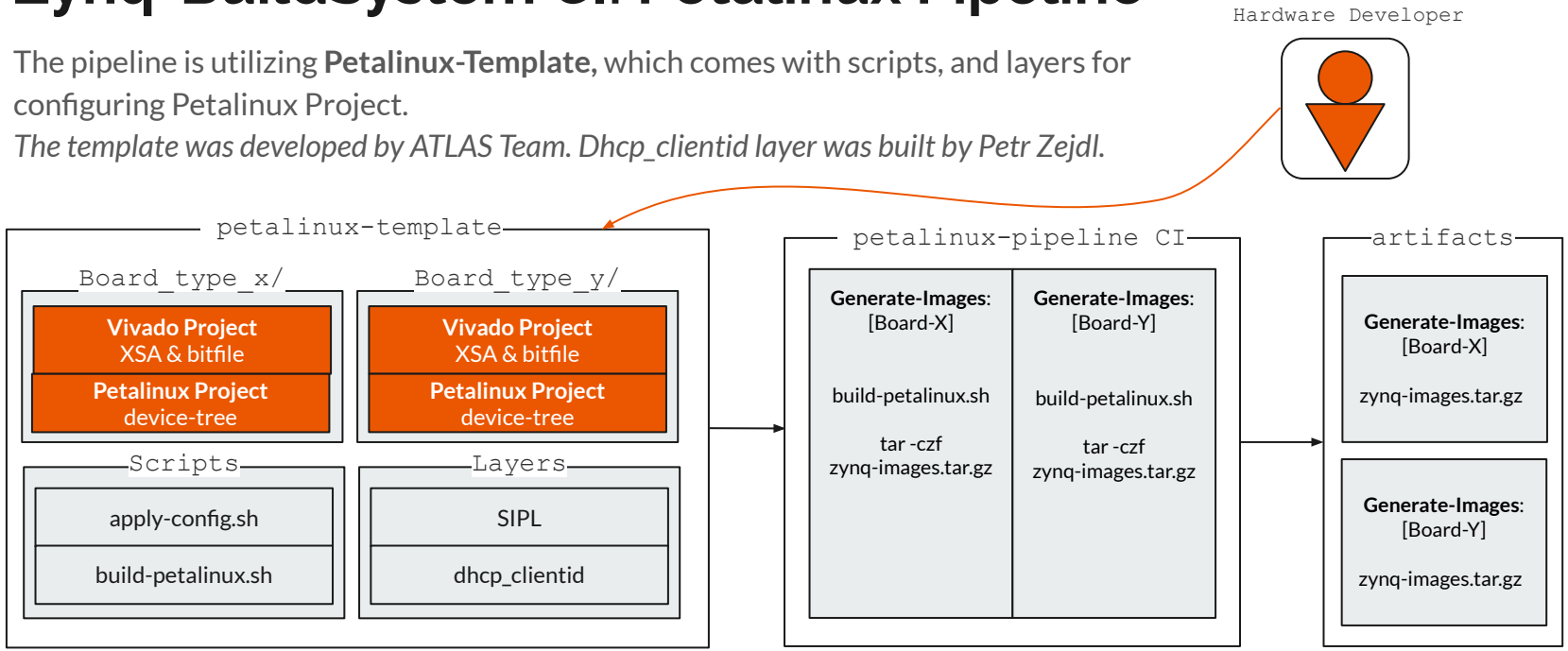
PetaLinux is used for building Linux-based systems for Zynq boards.



Zynq-BuildSystem CI: Petalinux Pipeline

The pipeline is utilizing **Petalinux-Template,** which comes with scripts, and layers for configuring Petalinux Project.

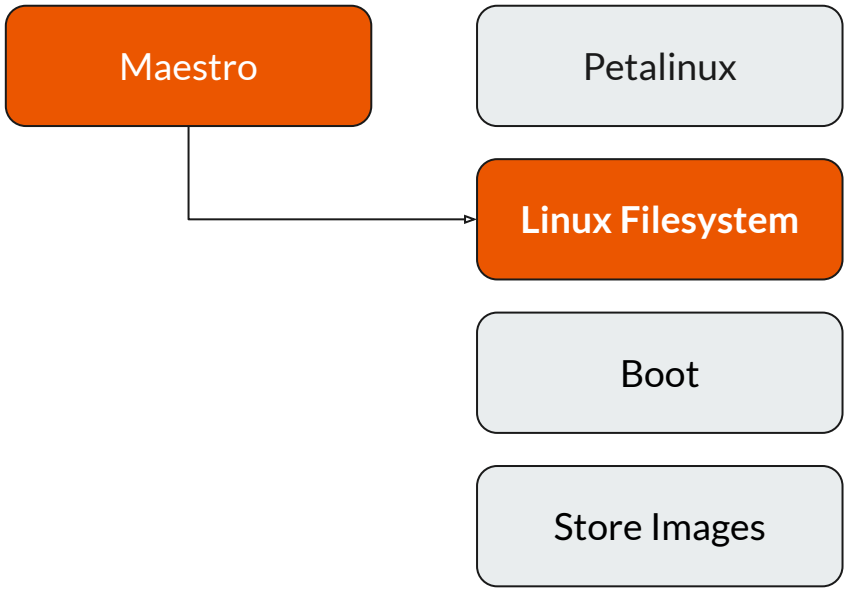
The template was developed by ATLAS Team. Dhcp_clientid layer was built by Petr Zejdl.



Zynq-BuildSystem CI: Linux Filesystem Pipeline

Linux Filesystem

Linux Filesystem Pipeline is responsible for generating sysroot (rootfs) for the Zynq.



Zynq-BuildSystem CI: Linux Filesystem Pipeline

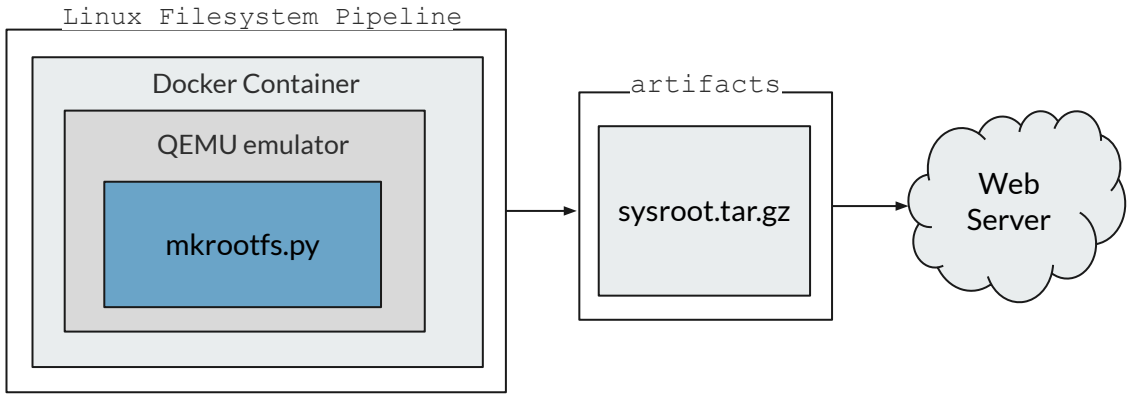
Linux Filesystem Pipeline

A python script runs dnf to install the rootfs.

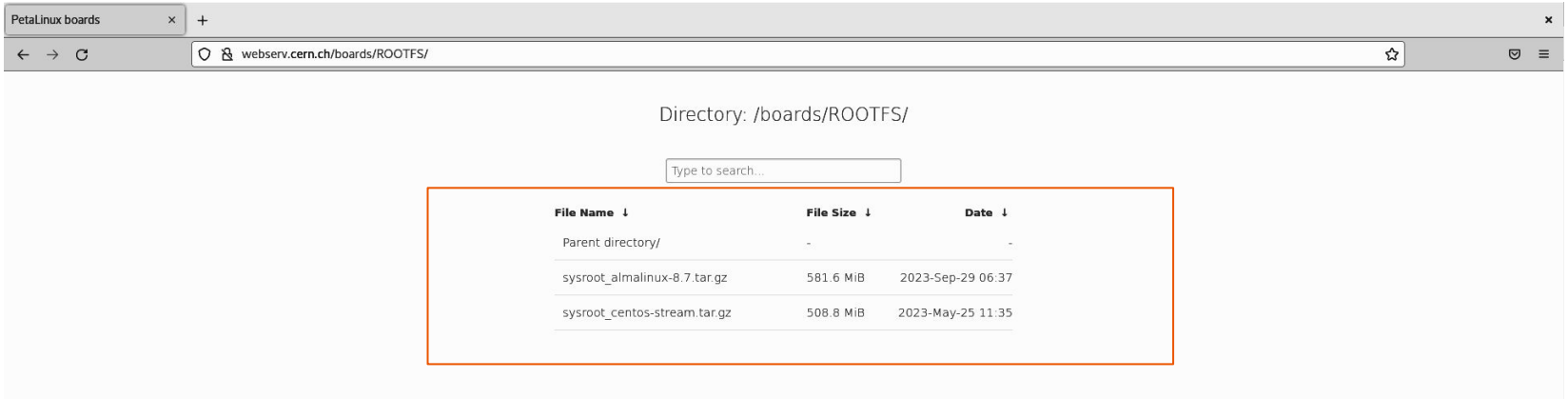
The kernel modules are added later in Boot Pipeline. Meanwhile the generic rootfs is saved on the web-server.

The pipeline currently supports:

- Alma Linux 8
- CentOS Stream



Zynq-BuildSystem CI: Linux Filesystem Pipeline



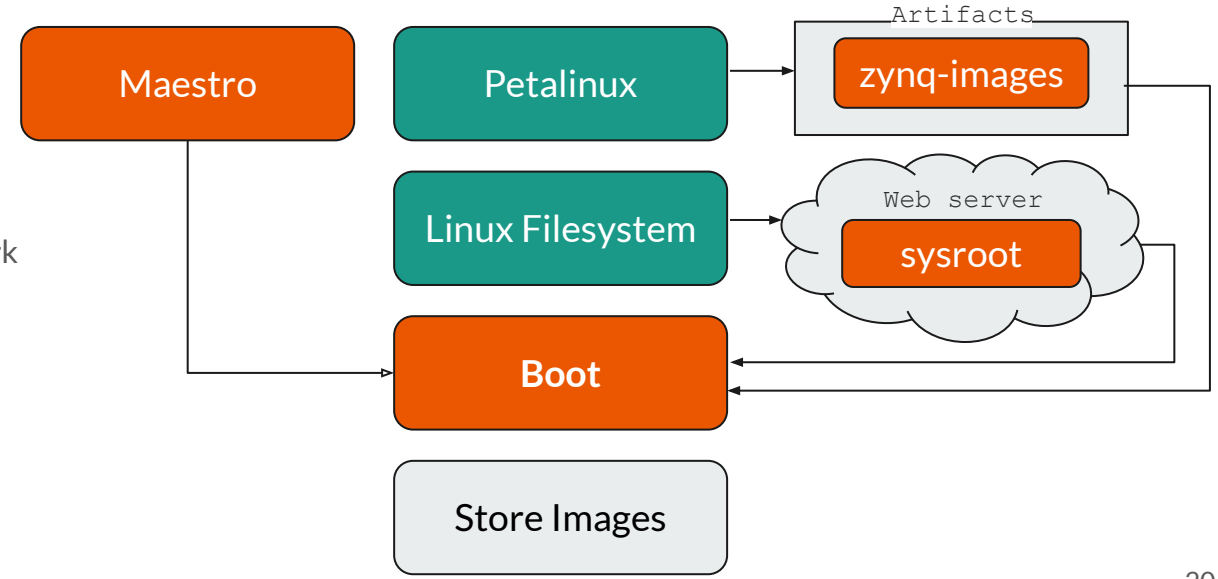
Screenshot of the rootfs directory, on the web-server.

Zynq-BuildSystem CI: Boot Pipeline

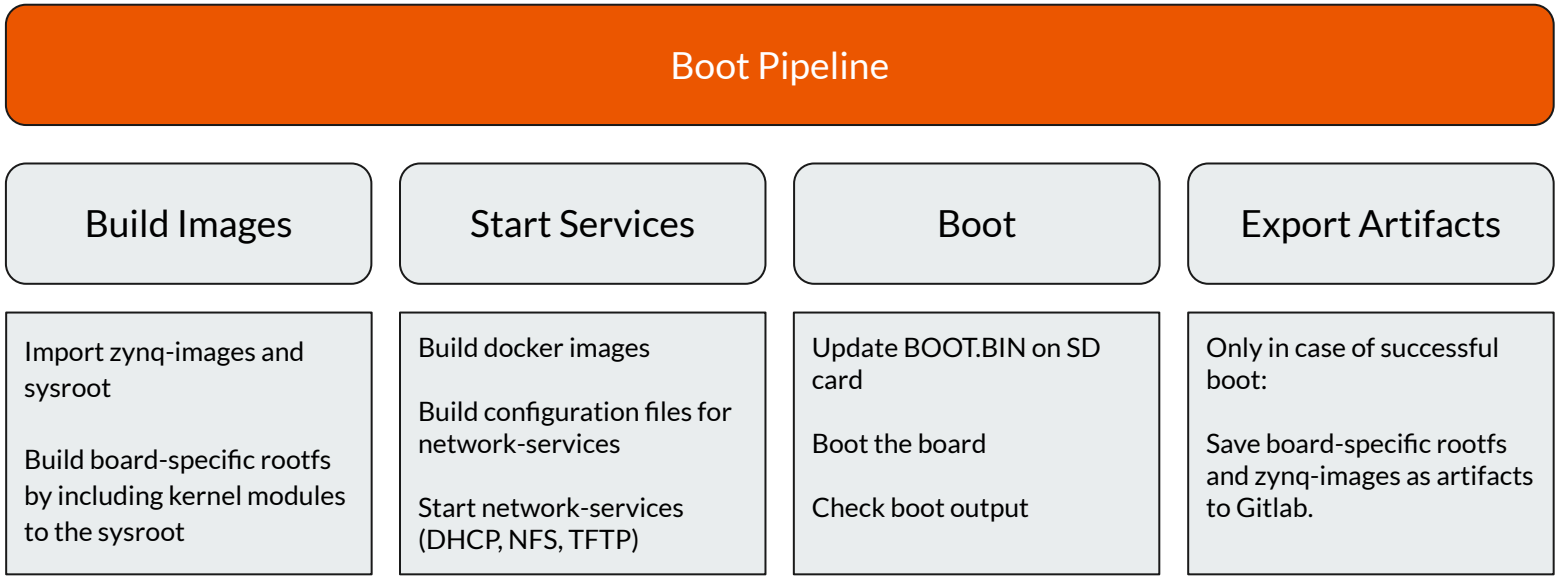
Boot Pipeline

Boot Pipeline is responsible of:

- Building board specific rootfs
- Configuring & running network services
- Testing boot images in action



Zynq-BuildSystem CI: Boot Pipeline Stages



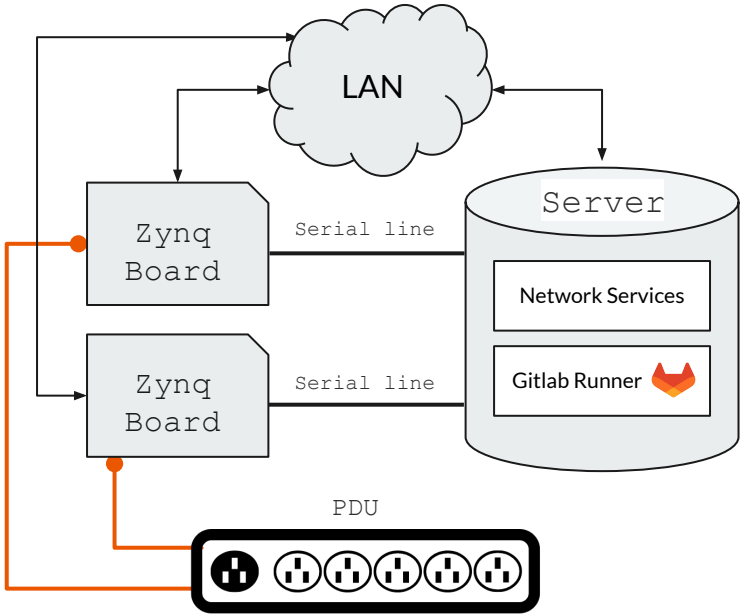
Zynq-BuildSystem CI: Boot Test Setup

Boot Test Setup

Diagram visualizes the current lab environment (it doesn't contain the build machines).

- Board and Server are connected via serial line.
- Board is getting information of network boot from a private network.
- Board's power outlet is connected to the PDU (Power Distribution Unit).
- Server is hosting Gitlab Runner, and network services.

In the future we will switch to IPMC for both power cycling, and serial line connection.

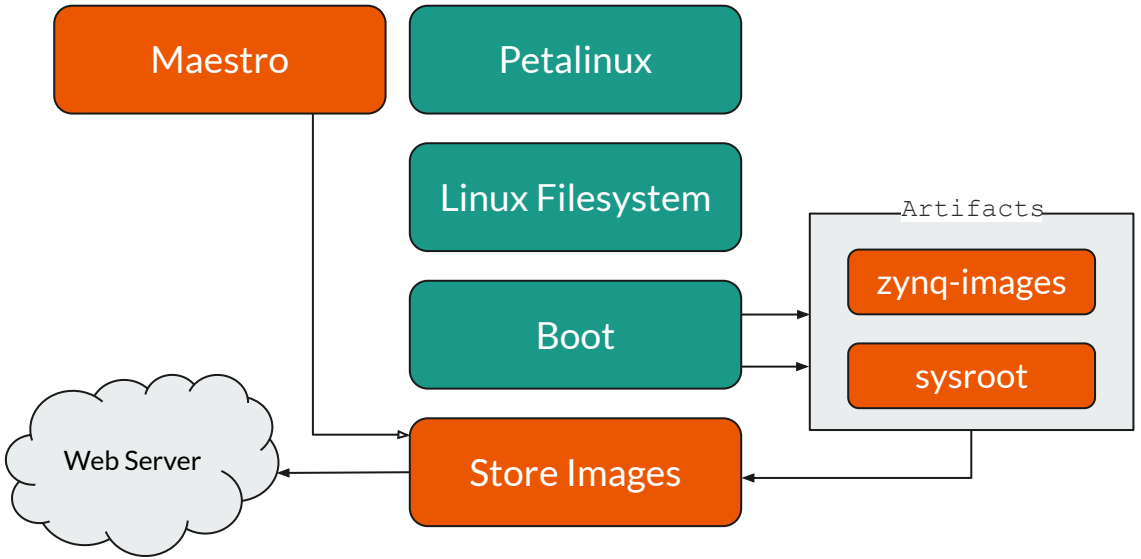


Zynq-BuildSystem CI: Store-Images Pipeline

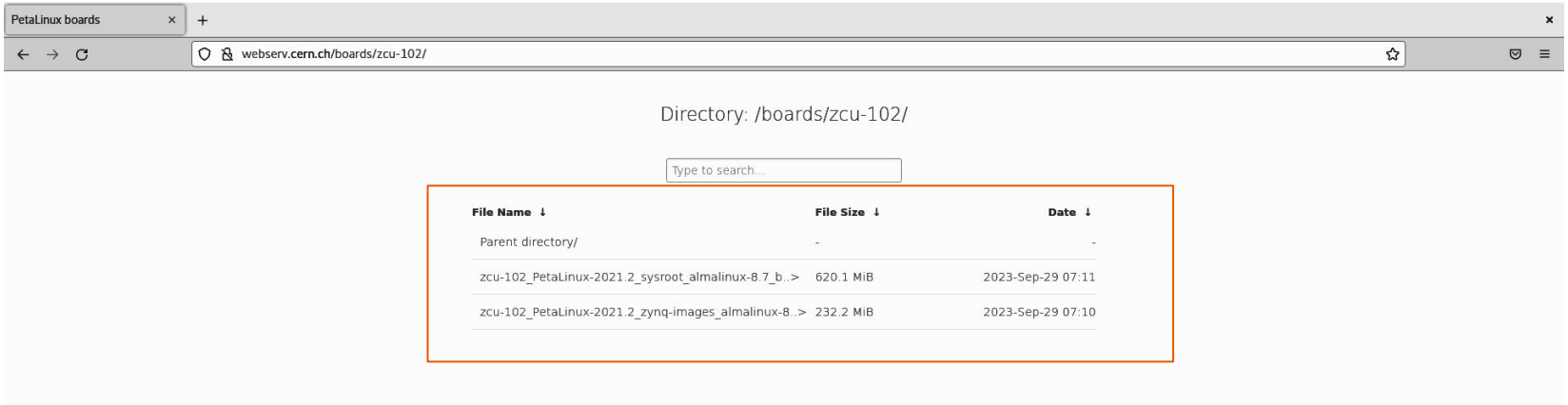
Store-Images Pipeline

Responsible of storing zynq-images, and rootfs on a webserver.

- Download artifacts from Boot pipeline
- Tag each file with a reference to the Petalinux Version, XSA, and sysroot type
- Upload files to a Web Server



Zynq-BuildSystem CI: Store-Images pipeline



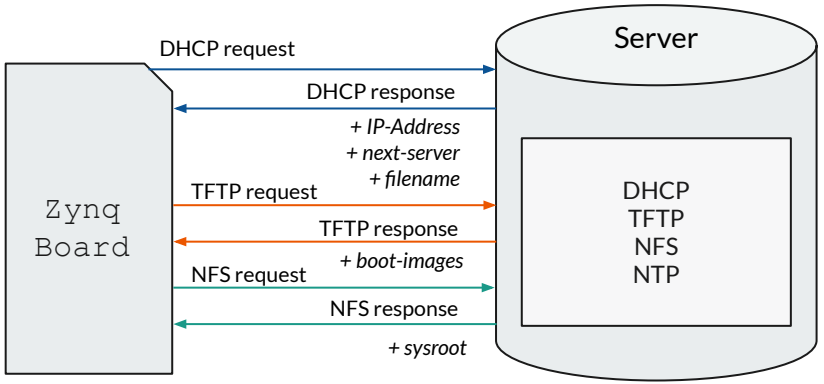
Screenshot of the board images directory, on the web-server.

Network Services

Network Services: What is needed in the lab?

Required services:

- **DHCP** (Dynamic Host Configuration Protocol) for network services. Dnsmasq, used in our case.
- **TFTP** (Trivial File Transfer Protocol) server for providing boot images.
- **NFS** (Network File System) for serving Linux File System.
- **NTP** (Network Time Protocol) for time synchronization. Chronyd, used in our case.

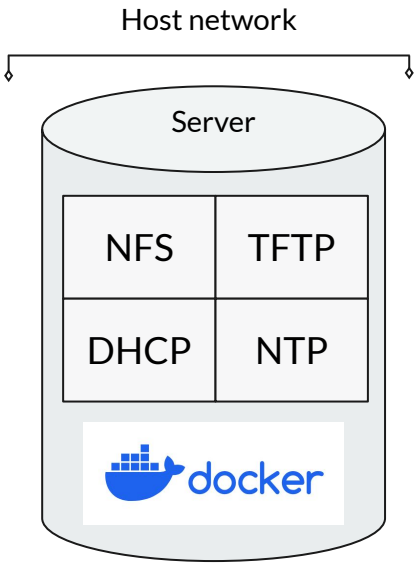


Network Services: Containerization

Why containers?

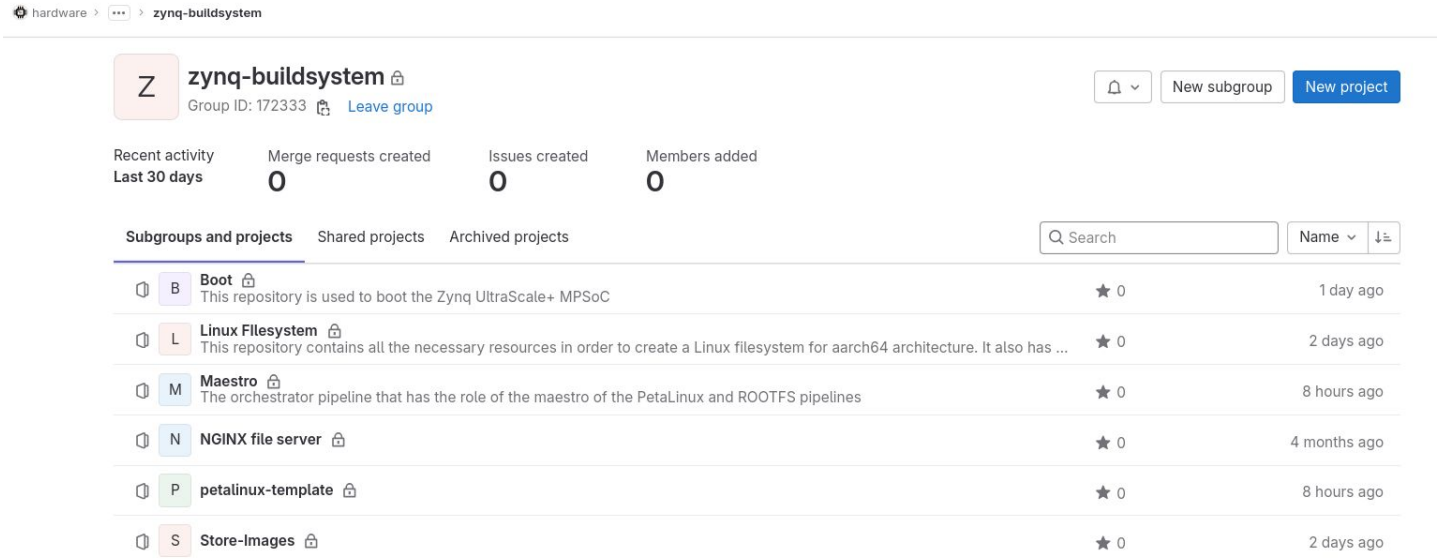
Containerized network services were developed to provide easily deployable services for network boot in the lab environment.

Zynq-buildsystem CI is using these network services. However, they are also available to be used independently.



Demo

Demo: Zynq-Buildsystem Repository

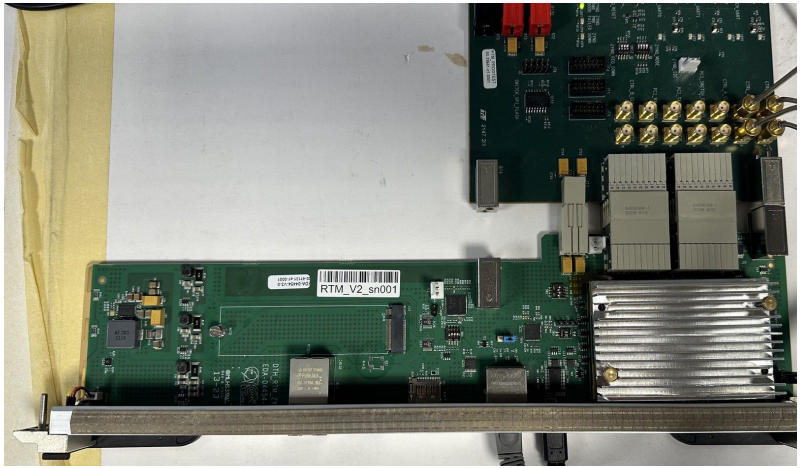


https://gitlab.cern.ch/hardware/zynq/zynq-buildsystem

Demo: Adding a new board to Zynq-Buildsystem CI

Let’s add a new board to the CI:

- board-type: **Trenz-RTM-V2-2021_2**
- boot-volume: **mmcblk1p1**
- testing-host: **ATCA-LAB40-R02-01-02-ctrl-1**

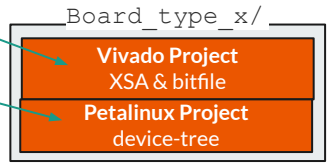


Trenz RTMv2

Demo: Adding a new board to Zynq-Buildsystem CI

In order to add a new board to the Zynq-Buildsystem Pipeline, there are several things which are needed to be done:

- Prepare hardware description file (XSA), and bitfile from board's Vivado project
- Prepare device-tree (system-user.dtsi), from board's Petalinux project



Petalinux Pipeline

1. Create a new folder in `petalinux-template/boards/$BOARD_NAME`
2. Copy the default layers.conf file from `petalinux-template/boards/example/conf/layer.conf` to board directory
3. Symlink boards/$BOARD_NAME/xsa to the xsa file, and boards/$BOARD_NAME/bitfile to the bitfile of the board

Maestro Pipeline

4. Add board type information to: `maestro/parallel-matrix.yml`
5. Add board host information to: `maestro/boot-config.yml`
6. Add board option to the `maestro/variables.yml`

Demo: Check prepared files (Step 1)

Check that the XSA, bitfile, and system-user.dtsi files have been prepared.

```
[karutjun@pccmdstudent2 demo]$ ls -l prepared_files/
total 6548
-rw-r--r--. 1 karutjun zh    8806 Sep 29 19:36 system-user.dtsi
-rw-r--r--. 1 karutjun zh 1847777 Sep 29 19:27 top.04.00.0000.xsa
-rw-r--r--. 1 karutjun zh 4840681 Sep 29 19:27 top.bit
[karutjun@pccmdstudent2 demo]$
```

Demo: Clone Petalinux Pipeline (Step 2)

Clone the Petalinux Pipeline repository:

$ git clone ssh://git@gitlab.cern.ch:7999/hardware/zynq/zynq-buildsystem/petalinux-template.git

```
[karutjun@pccmdstudent2 demo]$ git clone ssh://git@gitlab.cern.ch:7999/hardware/zynq/zynq-buildsystem/petalinux-template.git
Cloning into 'petalinux-template'...
remote: Enumerating objects: 657, done.
remote: Counting objects: 100% (621/621), done.
remote: Compressing objects: 100% (324/324), done.
remote: Total 657 (delta 261), reused 592 (delta 249), pack-reused 36
Receiving objects: 100% (657/657), 33.41 MiB | 79.00 MiB/s, done.
Resolving deltas: 100% (266/266), done.
[karutjun@pccmdstudent2 demo]$ ls -l
total 0
drwxr-xr-x. 5 karutjun zh 154 Sep 29 19:28 petalinux-template
drwxr-xr-x. 2 karutjun zh  47 Sep 29 19:27 prepared_files
[karutjun@pccmdstudent2 demo]$
```

Demo: Add new board type to Petalinux Pipeline (Step 3)

Import prepared files

$ cd petalinux-template
$ mkdir petalinux-template/boards/Trenz-RTM-V2-2021_2
$ cp prepared_files/*
petalinux-template/boards/Trenz-RTM-V2-2021_2/
$ cd petalinux-template/boards/Trenz-RTM-V2-2021_2/

Add new board info

$ mkdir -p recipes-bsp/device-tree/files/
$ mv system-user.dtsi recipes-bsp/device-tree/files/
$ ln -s top.04.00.0000.xsa xsa
$ ln -s top.bit bitfile
$ cp -r ../example/conf/ .
$ git add . && git commit -m “Update for Demo”
$ git push origin master

Expected result:

```
[karutjun@pccmdstudent2 Trenz-RTM-V2-2021_2]$ ls -l
total 6536
lrwxrwxrwx. 1 karutjun zh       7 Sep 29 19:58 bitfile -> top.bit
drwxr-xr-x. 2 karutjun zh      24 Sep 29 20:01 conf
drwxr-xr-x. 3 karutjun zh      25 Sep 29 19:57 recipes-bsp
-rw-r--r--. 1 karutjun zh 1847777 Sep 29 19:51 top.04.00.0000.xsa
-rw-r--r--. 1 karutjun zh 4840681 Sep 29 19:51 top.bit
lrwxrwxrwx. 1 karutjun zh      18 Sep 29 19:59 xsa -> top.04.00.0000.xsa
[karutjun@pccmdstudent2 Trenz-RTM-V2-2021_2]$
```

Demo: Configure parallel-matrix.yml (Step 4)

Add board information to `variables.yml`:

```
    # Zynq-buildsystem CI: Pick a board
    BOARD_TYPE_OPT:
      description: "Pick a single board type from the dropdown list or choose ALL to run the pipeline for
all boards in parallel"
      value: "NO"
      options:
        - "NO"
        - "zcu-102"
        - "Trenz-RTM-V2-2021_2"
        - "ALL"
```

Demo: Configure parallel-matrix.yml (Step 5)

Add board information to `parallel-matrix.yml`:

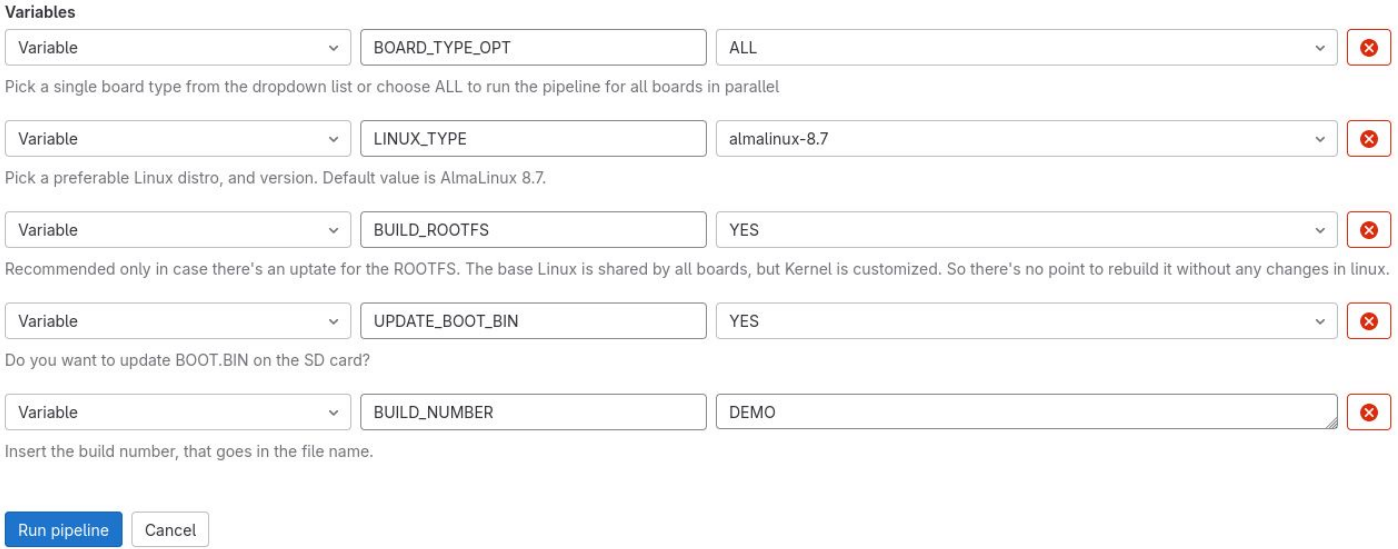
- **BOARD_TYPE: Trenz-RTM-V2-2021_2**
- **XSA:** top.04.00.0000.xsa
- **BOOT_VOLUME:** mmcblk1p1
- **TESTING_HOST**: ATCA-LAB40-R02-01-02-ctrl-1

Demo: Configure parallel-matrix.yml (Step 6)

Add board information to `boot-config.yml`:

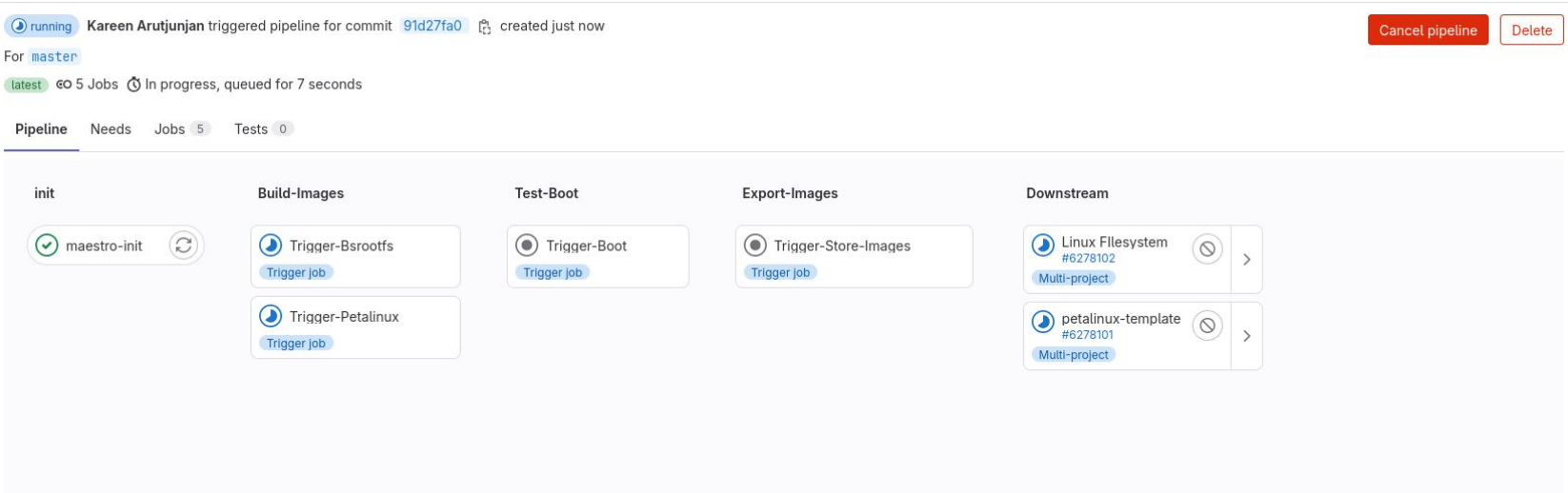
- **hostname: ATCA-LAB40-R02-01-02-ctrl-1**
- **alias:** rtm2-lab40-r02-board05
- **dhcp_client_id:** ff:00:00:00:04:00:02:00:00:31:5a:48:50:4d:2e:33:2d:31:d4:41:54:43:41:2d:34:30:2d:32:42:2d:30:31:2d:52:30:31:2d:31:36:00:00:07:c0:02
- **tty_usb:** ACM0
- **pdu_outlet:** 3
- **ip:** 172.0.0.3

Demo: Trigger Zynq-Buildsystem CI (Step 7)



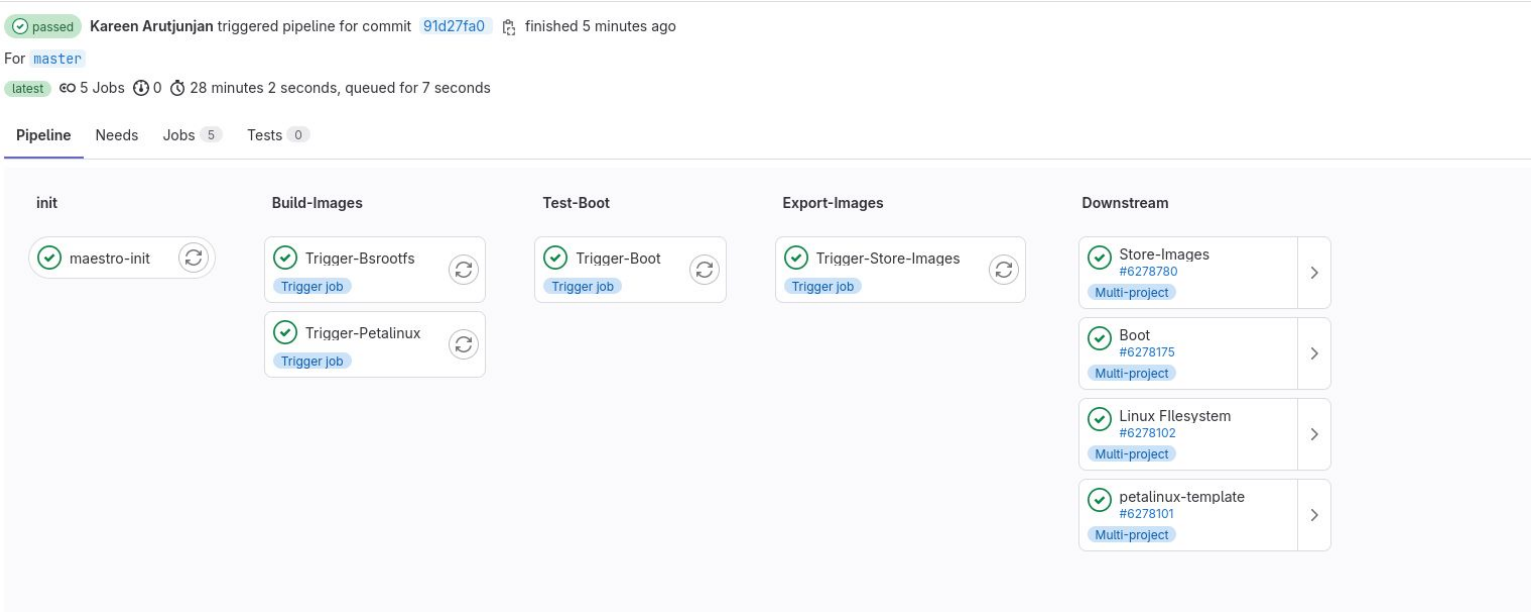
https://gitlab.cern.ch/hardware/zynq/zynq-buildsystem/maestro/-/pipelines/new

Demo: Trigger Zynq-Buildsystem CI (Step 7)

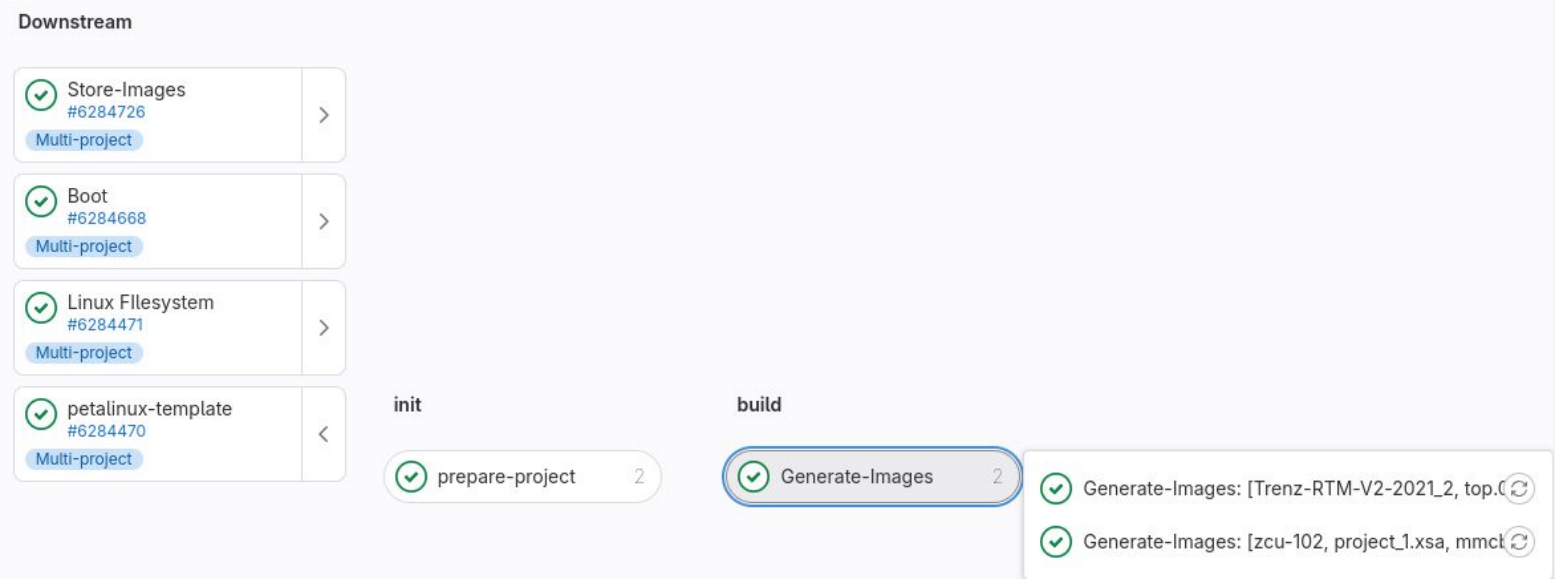


Demo: Trigger Zynq-Buildsystem CI (Step 7)

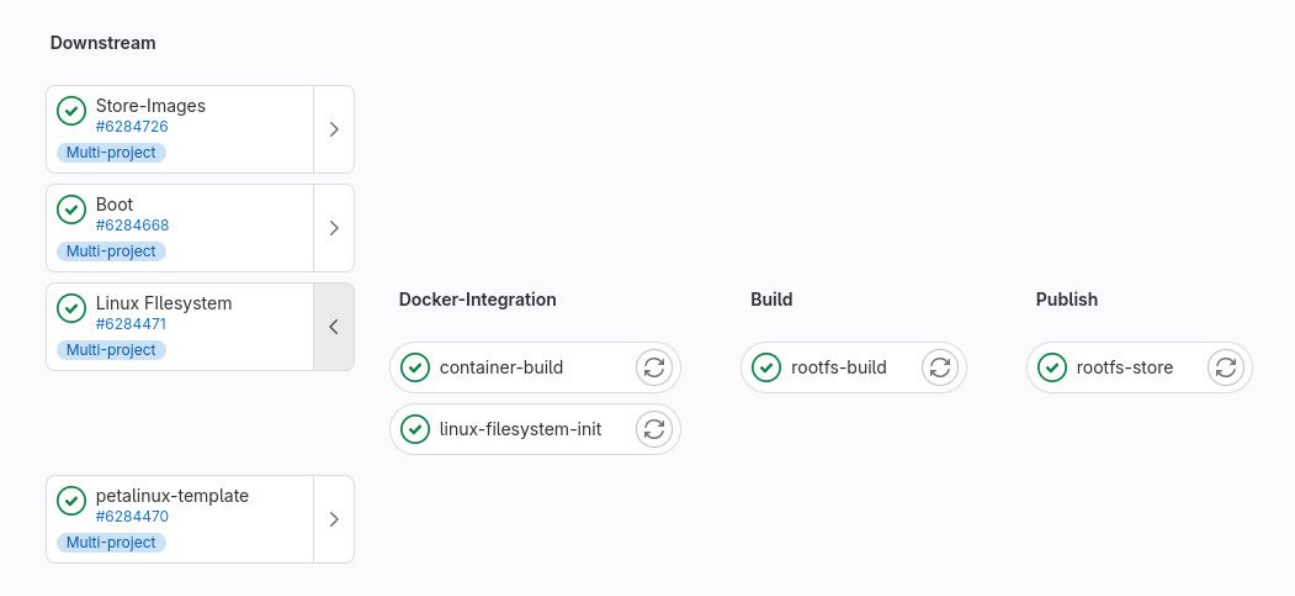
Demo: Check results (Step 8)



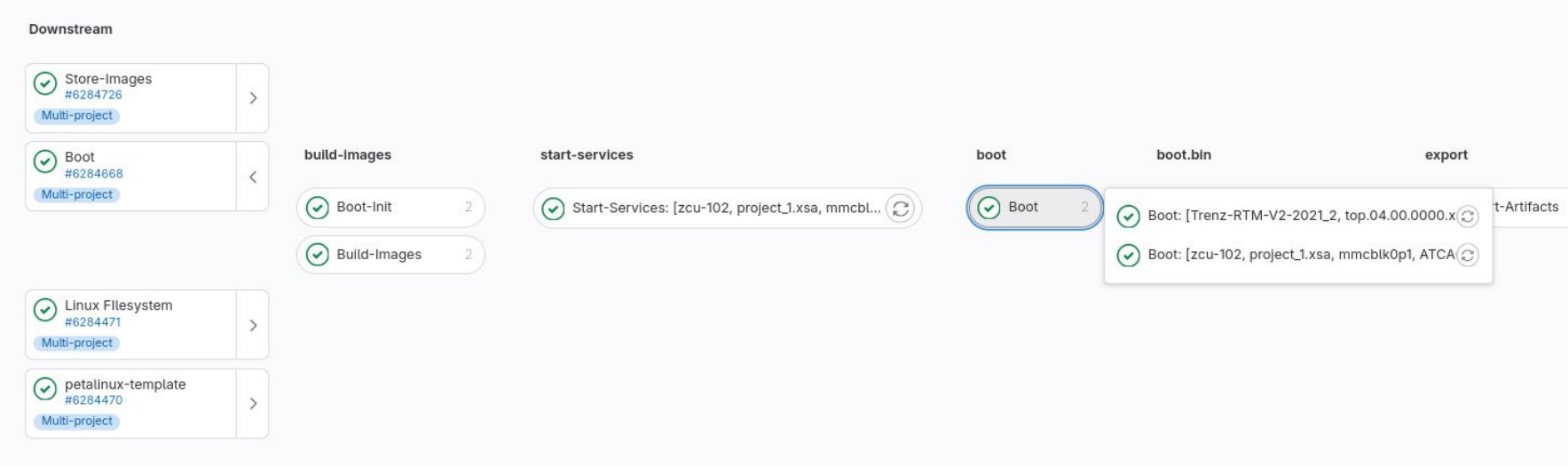
Demo: Check results - Petalinux (Step 9)



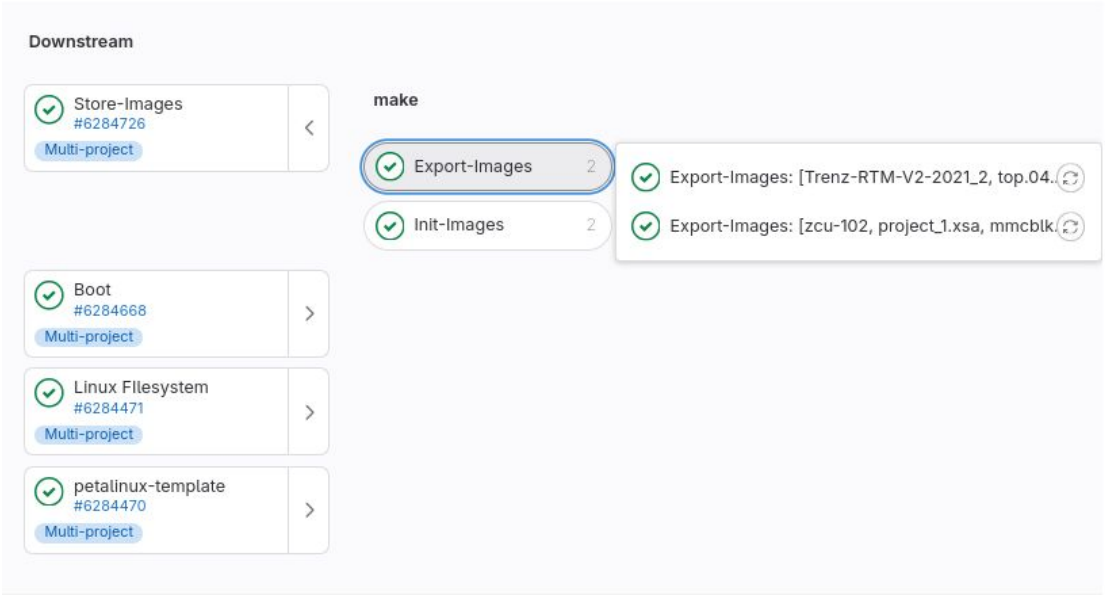
Demo: Check results - Linux Filesystem (Step 10)



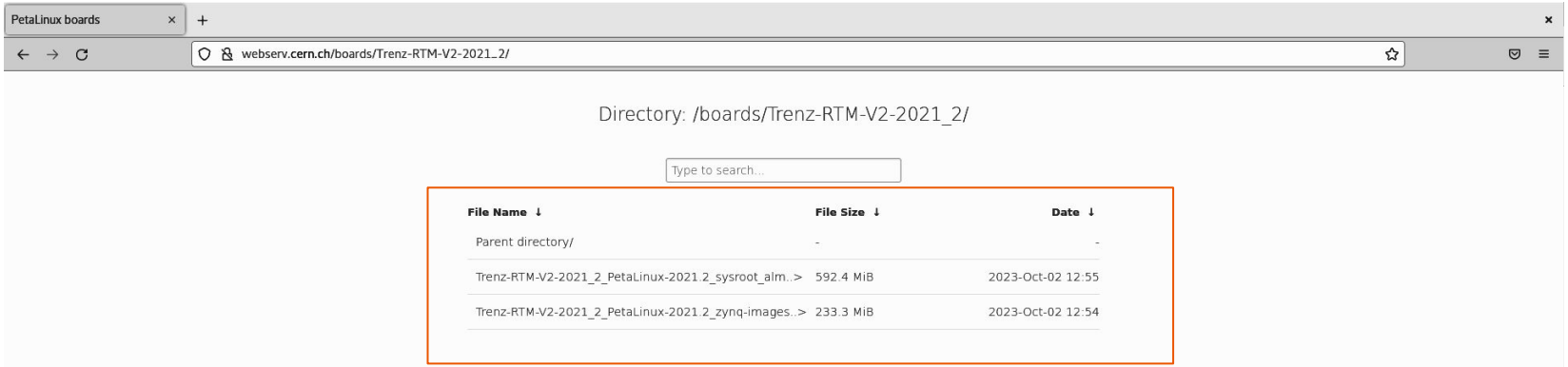
Demo: Check results - Boot (Step 11)



Demo: Check results -Store Images (Step 11)



Demo: Collect images (Step 12)



Screenshot of the zynq-images, on the web-server.

End notes

End notes: Goals accomplished

Goals accomplished

We have successfully built a Pipeline for building and testing software of SoC.

Future work

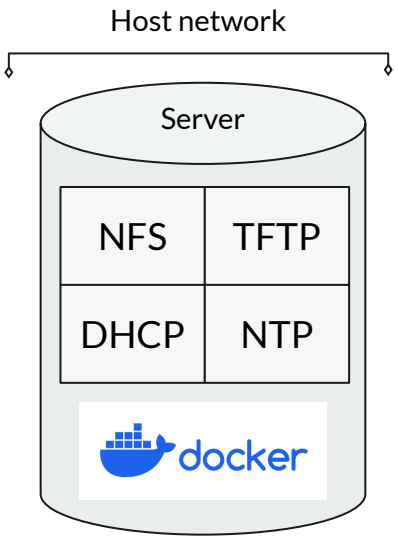
The goal is to extend the project to support deploying software for extended number of boards. For example, fill the crate with 10 boards of same type, and deploy software for all of them with a Pipeline.

Network Services: Public repository

Network Services

The whole network-services repository, along with Dockerfiles, and configuration files are free to use:

- https://gitlab.cern.ch/hardware/network-services/



C'est tout

Questions?