CERN

Fully Automated: Updates on the Continuous Integration for supported Linux distributions at CERN

HEPiX Spring 2023

2023-03-28

Ben Morrice

On behalf of CERN IT

IT-CD

Agenda

- Introduction
- Recap / Overview of work presented from HEPiX Autumn 2021 [0]
- Evolution since
- Useful for others?
- Upstream automation!?

[0] https://indico.cern.ch/event/1078853/contributions/4576274/

Introduction

- This talk is focused on image generation
  - Docker images
  - OpenStack cloud images (Virtual Machine **and** bare-metal)

- This talk will not touch on CERN's daily/weekly release cycle for packages
  - If you are interested in this, please see the talk presented at HEPiX Autumn 2021 ***"Navigating the Stream: automating CentOS releases at CERN"*** [0]

[0] https://indico.cern.ch/event/1078853/contributions/4576273/

Background

- Linux Support has a weekly rotaer that takes care of running manual scripts for package updates and **ad-hoc** image building
- Image testing is performed on a **manual**, **irregular** timeline
- Tasks are **repetitive, error-prone** and are required to be done for each of our supported distributions

Not scalable: a recipe for trouble

- Testing prior to release is performed manually
- Difficult to catch issues, often resulting in our user community informing us of problems via support tickets
- Additional distributions come and require to be supported, increasing the workload

Solution?

- Moved away from manual scripts to CI/CD tasks (automated scripts!)
- Added automated testing, both upstream and CERN specific
- We adapted our CI so it is as much distribution independent as possible to ease adding additional distributions in the future

Tooling

- Koji: https://docs.pagure.org/koji/
    - RPM and image building system
- Openstack Nova (VMs), Glance (cloud images), Ironic (bare metal provisioning)
- GitLab, GitLab runners, GitLab CI/CD capabilities
- One or two Python and Bash scripts to glue everything together

Testing

All of our automation is based on the same sets of tests:

- https://gitlab.cern.ch/linuxsupport/testing/centos_functional_tests
    - Mirrored from upstream
- https://gitlab.cern.ch/linuxsupport/testing/cern_centos_functional_tests
    - Tests specific to CERN, though we welcome contributions 😄
    - So far AFS, NTP, IPv6, Kerberos and partitioning tests among others
- https://gitlab.cern.ch/ai-config-team/cern_puppet_centos_functional_tests
    - Tests only relevant for Puppet-managed machines
    - Cloud-init bootstrapping, Puppet runs, IPv6, Auth setup among others

Image life cycle: Docker images

Docker images are created via gitlab-ci pipelines

- AlmaLinux 9 (x86_64 and aarch64) [NEW as of January 2023]
- AlmaLinux 8 (x86_64 and aarch64) [NEW as of January 2023]
- CentOS Stream 9 (x86_64 and aarch64)
- CentOS Stream 8 (x86_64 and aarch64)
- CERN CentOS 7 (x86_64)

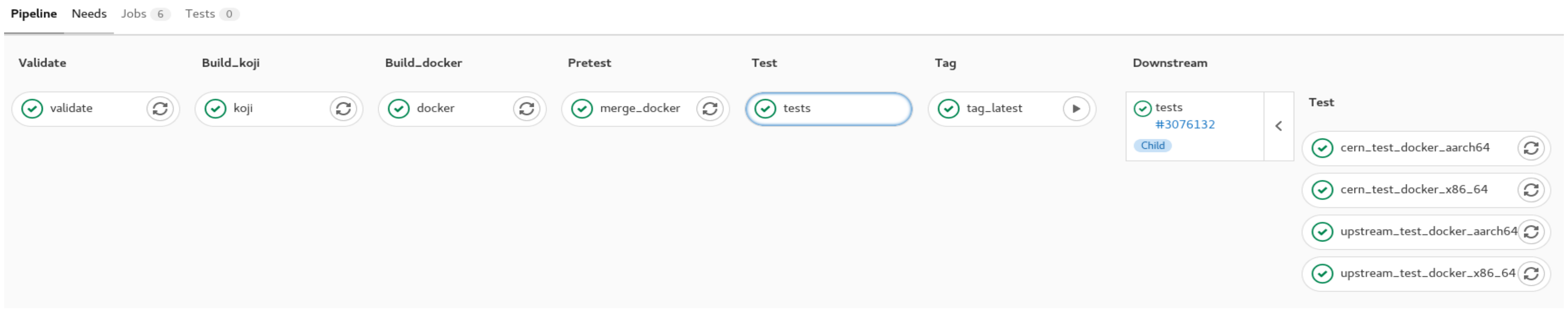
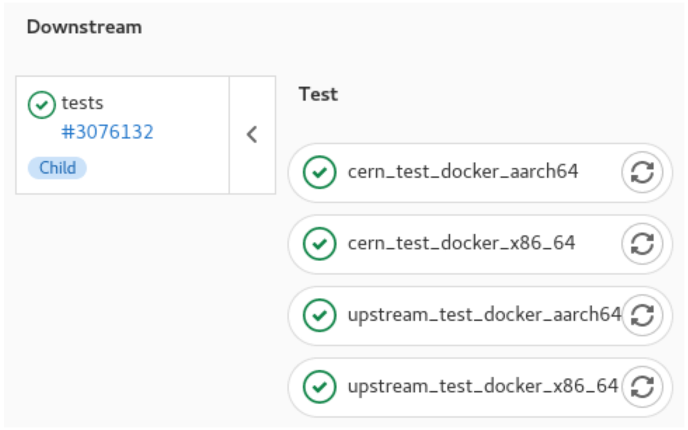


Image life cycle: Docker images

Once a month:

- Validate the Kickstart file to use
- We build each image using Koji image building capabilities and create Docker images
- We then take the resulting tarball and push it to our internal registry as a test image
- We run then our set of tests, if they pass
    - The image is auto-promoted to the *:latest* tag
    - pushed to [our Dockerhub organisation]()


Downstream

tests
#3076132
Child

Test

cern_test_docker_aarch64
cern_test_docker_x86_64
upstream_test_docker_aarch64
upstream_test_docker_x86_64

Image life cycle: OpenStack cloud images

Once a month:

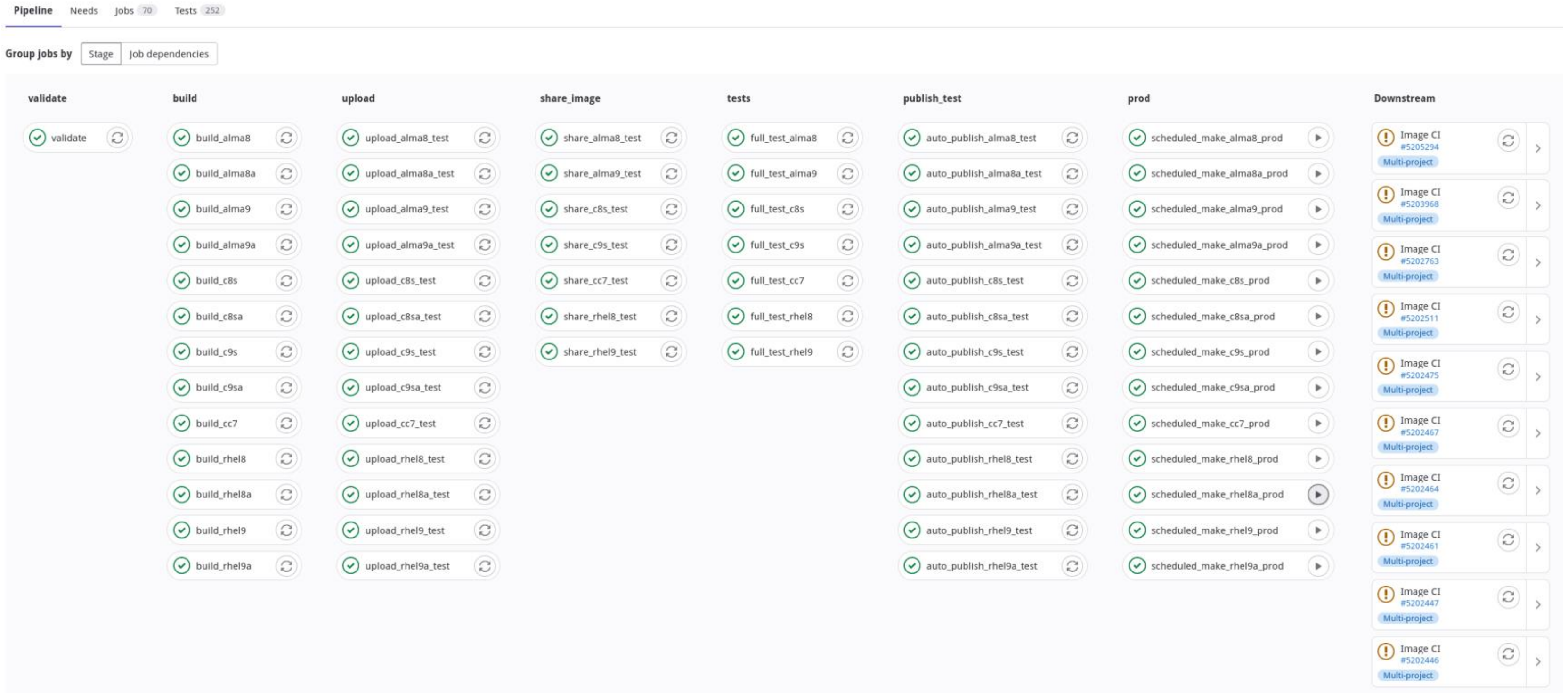
- gitlab-ci scheduled tasks run and make use of our Koji build system to build disk images
- Each images is built using a Kickstart file

The current selection of built images:

- AlmaLinux 9 (x86_64 and aarch64) [NEW as of January 2023]
- AlmaLinux 8 (x86_64 and aarch64) [NEW as of January 2023]
- Red Hat Enterprise Linux 9 (x86_64 and aarch64) [NEW as of January 2023]
- Red Hat Enterprise Linux 8 (x86_64 and aarch64) [NEW as of January 2023]
- Red Hat Enterprise Linux 7 (x86_64) [NEW as of January 2023]
- CentOS Stream 9 (x86_64 and aarch64)
- CentOS Stream 8 (x86_64 and aarch64)
- CERN CentOS 7 (x86_64)

- We have both x86_64 and aarch64 Koji build nodes
- Our images are compatible with both BIOS and UEFI boot modes

Image life cycle: OpenStack cloud images

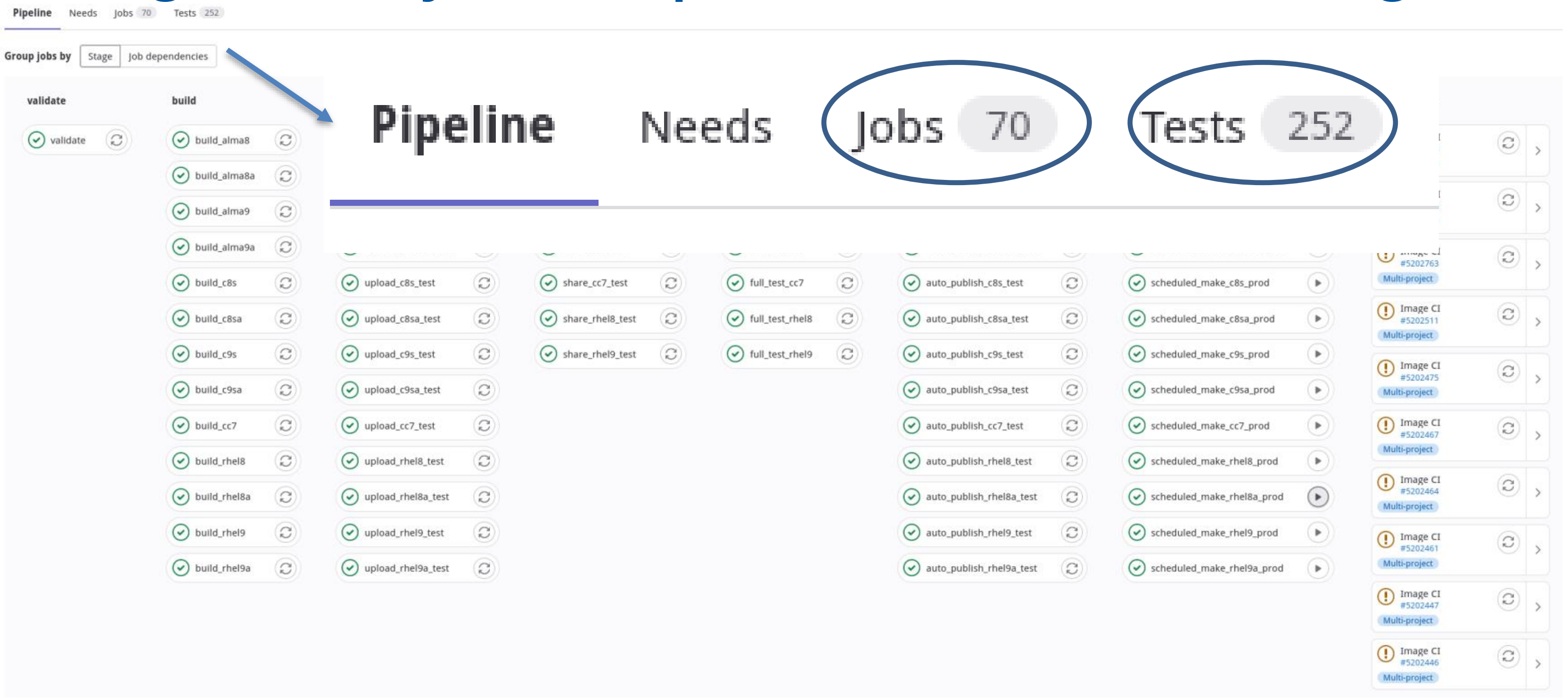


Pipeline | Needs | Jobs 70 | Tests 252

Group jobs by: Stage | Job dependencies

validate	build	upload	share_image	tests	publish_test	prod	Downstream
✓ validate	✓ build_alma8	✓ upload_alma8_test	✓ share_alma8_test	✓ full_test_alma8	✓ auto_publish_alma8_test	✓ scheduled_make_alma8_prod	! Image CI #5205294 Multi-project
	✓ build_alma8a	✓ upload_alma8a_test	✓ share_alma9_test	✓ full_test_alma9	✓ auto_publish_alma8a_test	✓ scheduled_make_alma8a_prod	! Image CI #5203968 Multi-project
	✓ build_alma9	✓ upload_alma9_test	✓ share_c8s_test	✓ full_test_c8s	✓ auto_publish_alma9_test	✓ scheduled_make_alma9_prod	! Image CI #5202763 Multi-project
	✓ build_alma9a	✓ upload_alma9a_test	✓ share_c9s_test	✓ full_test_c9s	✓ auto_publish_alma9a_test	✓ scheduled_make_alma9a_prod	! Image CI #5202511 Multi-project
	✓ build_c8s	✓ upload_c8s_test	✓ share_cc7_test	✓ full_test_cc7	✓ auto_publish_c8s_test	✓ scheduled_make_c8s_prod	! Image CI #5202475 Multi-project
	✓ build_c8sa	✓ upload_c8sa_test	✓ share_rhel8_test	✓ full_test_rhel8	✓ auto_publish_c8sa_test	✓ scheduled_make_c8sa_prod	! Image CI #5202467 Multi-project
	✓ build_c9s	✓ upload_c9s_test	✓ share_rhel9_test	✓ full_test_rhel9	✓ auto_publish_c9s_test	✓ scheduled_make_c9s_prod	! Image CI #5202464 Multi-project
	✓ build_c9sa	✓ upload_c9sa_test			✓ auto_publish_c9sa_test	✓ scheduled_make_c9sa_prod	! Image CI #5202461 Multi-project
	✓ build_cc7	✓ upload_cc7_test			✓ auto_publish_cc7_test	✓ scheduled_make_cc7_prod	! Image CI #5202447 Multi-project
	✓ build_rhel8	✓ upload_rhel8_test			✓ auto_publish_rhel8_test	✓ scheduled_make_rhel8_prod	! Image CI #5202446 Multi-project
	✓ build_rhel8a	✓ upload_rhel8a_test			✓ auto_publish_rhel8a_test	✓ scheduled_make_rhel8a_prod	
	✓ build_rhel9	✓ upload_rhel9_test			✓ auto_publish_rhel9_test	✓ scheduled_make_rhel9_prod	
	✓ build_rhel9a	✓ upload_rhel9a_test			✓ auto_publish_rhel9a_test	✓ scheduled_make_rhel9a_prod	

Image life cycle: OpenStack cloud images



validate
- validate

build
- build_alma8
- build_alma8a
- build_alma9
- build_alma9a
- build_c8s
- build_c8sa
- build_c9s
- build_c9sa
- build_cc7
- build_rhel8
- build_rhel8a
- build_rhel9
- build_rhel9a

- upload_c8s_test
- upload_c8sa_test
- upload_c9s_test
- upload_c9sa_test
- upload_cc7_test
- upload_rhel8_test
- upload_rhel8a_test
- upload_rhel9_test
- upload_rhel9a_test

- share_cc7_test
- share_rhel8_test
- share_rhel9_test

- full_test_cc7
- full_test_rhel8
- full_test_rhel9

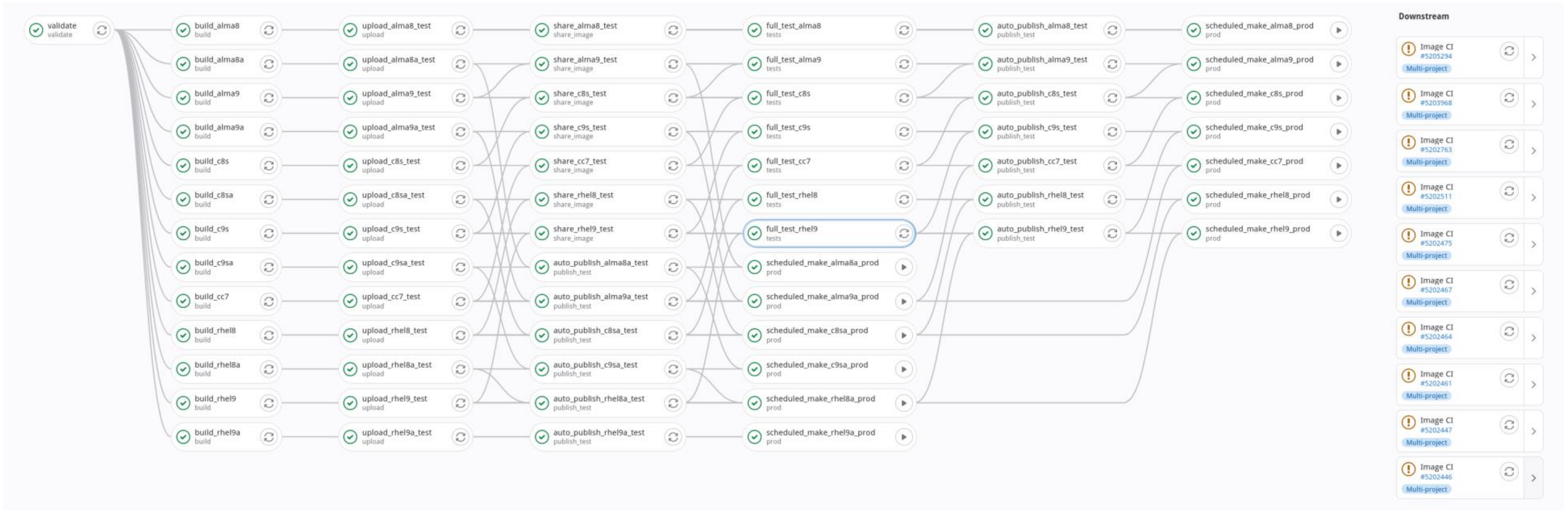
- auto_publish_c8s_test
- auto_publish_c8sa_test
- auto_publish_c9s_test
- auto_publish_c9sa_test
- auto_publish_cc7_test
- auto_publish_rhel8_test
- auto_publish_rhel8a_test
- auto_publish_rhel9_test
- auto_publish_rhel9a_test

- scheduled_make_c8s_prod
- scheduled_make_c8sa_prod
- scheduled_make_c9s_prod
- scheduled_make_c9sa_prod
- scheduled_make_cc7_prod
- scheduled_make_rhel8_prod
- scheduled_make_rhel8a_prod
- scheduled_make_rhel9_prod
- scheduled_make_rhel9a_prod

- Image CI #5202763 Multi-project
- Image CI #5202511 Multi-project
- Image CI #5202475 Multi-project
- Image CI #5202467 Multi-project
- Image CI #5202464 Multi-project
- Image CI #5202461 Multi-project
- Image CI #5202447 Multi-project
- Image CI #5202446 Multi-project

Image life cycle: OpenStack cloud images

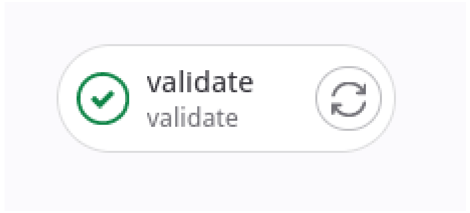
Got dependencies?
Basically, each distribution is built independently – but from a single pipeline



Part 0: Preparation steps

- We validate the Kickstart files

```
dnf install -y pykickstart
ksvalidator ...
```



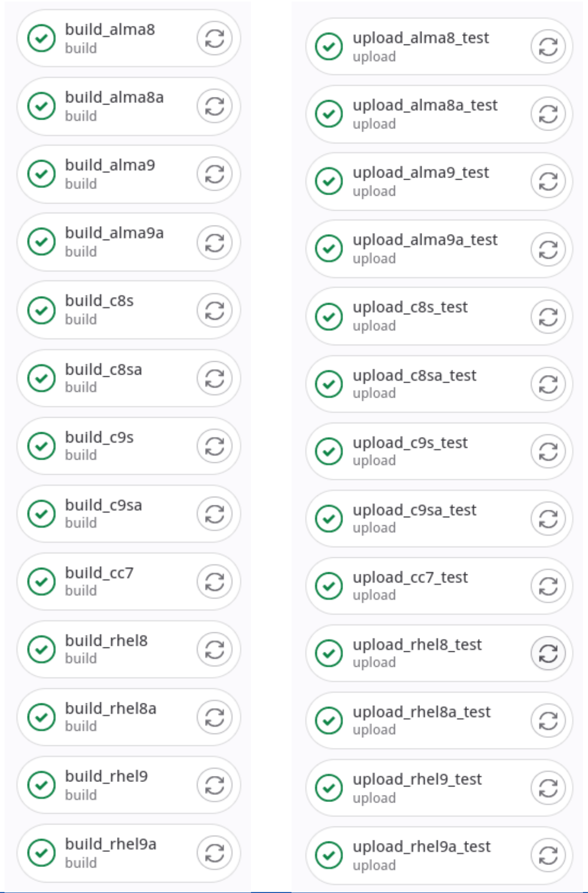
validate
validate

Part 1: Building

- For each distribution we build via Koji using our internal installation trees, i.e. our local mirror

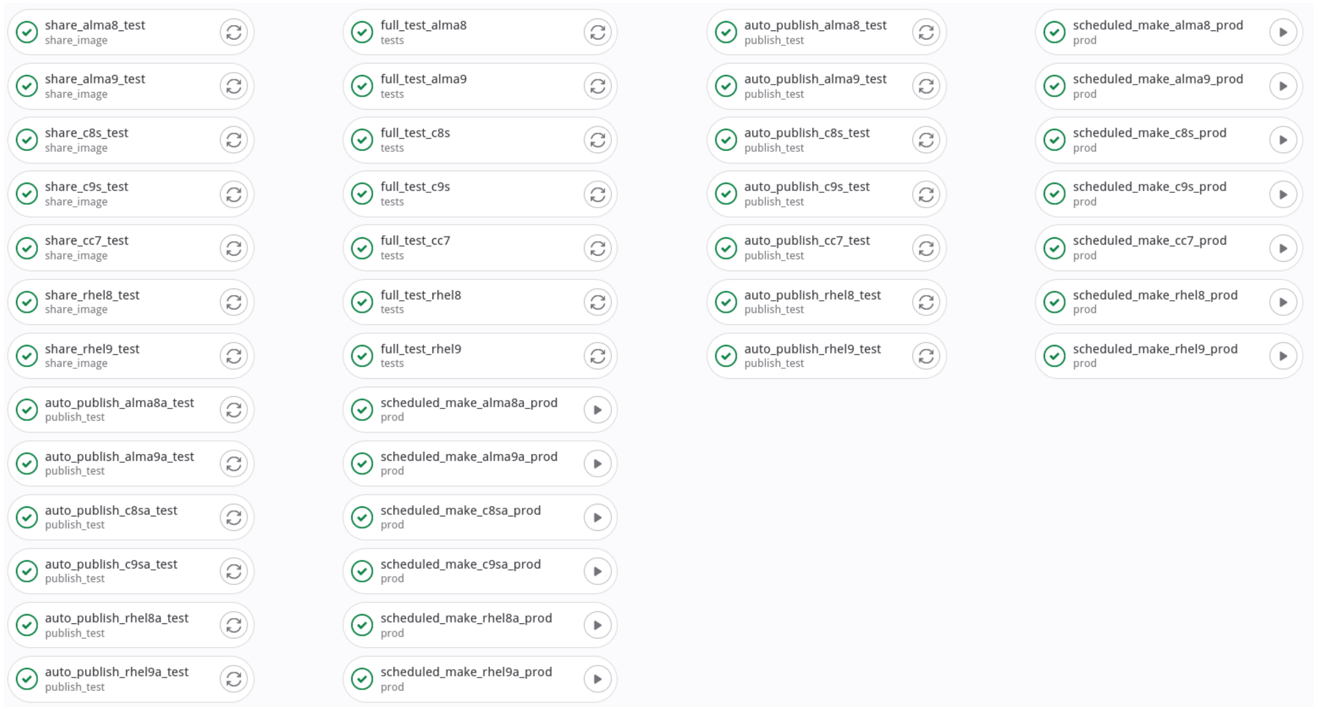
```
koji image-build --wait alma8-cloud 20230309 alma8-
image-8x
http://linuxsoft.cern.ch/cern/alma/8/BaseOS/$arch/os/
x86_64 --
ksurl=git+ssh://git@gitlab.cern.ch:7999/linuxsupport/koj
i-image-build#79397e32 --kickstart=alma8-cloud.ks …
```

- Once built, we upload it to our testing Openstack projects where we will create the test machines


build_alma8
build
build_alma8a
build
build_alma9
build
build_alma9a
build
build_c8s
build
build_c8sa
build
build_c9s
build
build_c9sa
build
build_cc7
build
build_rhel8
build
build_rhel8a
build
build_rhel9
build
build_rhel9a
build

upload_alma8_test
upload
upload_alma8a_test
upload
upload_alma9_test
upload
upload_alma9a_test
upload
upload_c8s_test
upload
upload_c8sa_test
upload
upload_c9s_test
upload
upload_c9sa_test
upload
upload_cc7_test
upload
upload_rhel8_test
upload
upload_rhel8a_test
upload
upload_rhel9_test
upload
upload_rhel9a_test
upload

Part 2: Testing and sharing



share_alma8_test
share_image

share_alma9_test
share_image

share_c8s_test
share_image

share_c9s_test
share_image

share_cc7_test
share_image

share_rhel8_test
share_image

share_rhel9_test
share_image

auto_publish_alma8a_test
publish_test

auto_publish_alma9a_test
publish_test

auto_publish_c8sa_test
publish_test

auto_publish_c9sa_test
publish_test

auto_publish_rhel8a_test
publish_test

auto_publish_rhel9a_test
publish_test

full_test_alma8
tests

full_test_alma9
tests

full_test_c8s
tests

full_test_c9s
tests

full_test_cc7
tests

full_test_rhel8
tests

full_test_rhel9
tests

scheduled_make_alma8a_prod
prod

scheduled_make_alma9a_prod
prod

scheduled_make_c8sa_prod
prod

scheduled_make_c9sa_prod
prod

scheduled_make_rhel8a_prod
prod

scheduled_make_rhel9a_prod
prod

auto_publish_alma8_test
publish_test

auto_publish_alma9_test
publish_test

auto_publish_c8s_test
publish_test

auto_publish_c9s_test
publish_test

auto_publish_cc7_test
publish_test

auto_publish_rhel8_test
publish_test

auto_publish_rhel9_test
publish_test

scheduled_make_alma8_prod
prod

scheduled_make_alma9_prod
prod

scheduled_make_c8s_prod
prod

scheduled_make_c9s_prod
prod

scheduled_make_cc7_prod
prod

scheduled_make_rhel8_prod
prod

scheduled_make_rhel9_prod
prod

Once built and uploaded to Openstack we run our tests (explained in next slide)
If everything goes well, we publish them for everyone as TEST images
We then manually promote to production

Part 2.1: Testing

- Each image triggers jobs for our cloud use cases
- We use extensively Gitlab CI Multiproject pipelines, Pipeline triggers and CI templates

- image-ci is responsible for storing **all** the CI magic ❤️

- Our image-ci can also be manually triggered, or integrated with other tools to assist with testing of CERN image bootstrap processes

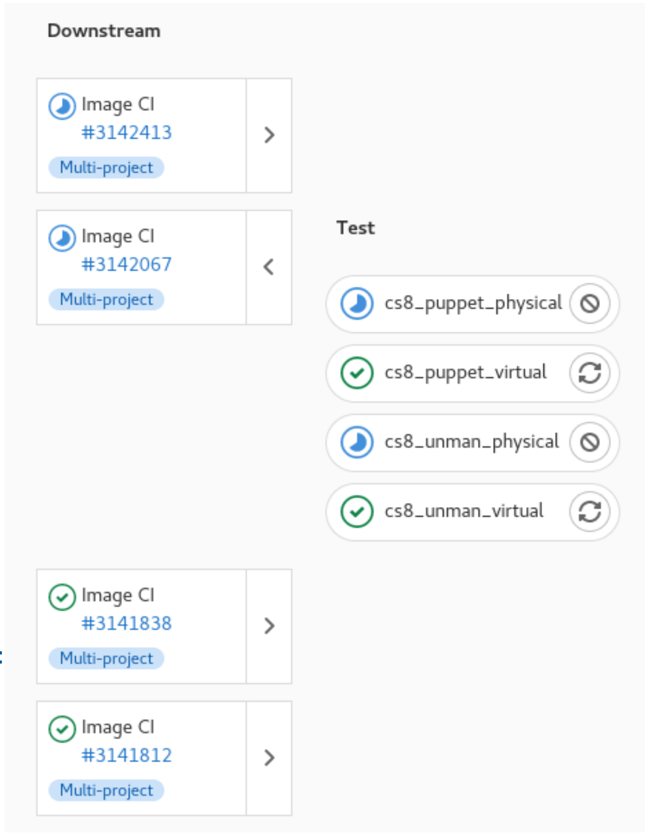


image-ci: What is it?

- Simply put, it's a Gitlab-CI workflow for testing OpenStack images

- Provides stages to
  - Boot OpenStack virtual machines (with logic to get the 'latest' version)
  - Waits until VM is up, and connects via ssh and executes test suites
  - Log files are retained for future analysis
- Supports
  - Puppet managed machines as well as unmanaged
  - Virtual machines and physical machines
  - All distributions that CERN currently maintain

image-ci: Scheduled triggers?

- Runs every 2 hours for puppet managed VMs
- Runs every 12 hours puppet managed physical servers (Ironic)

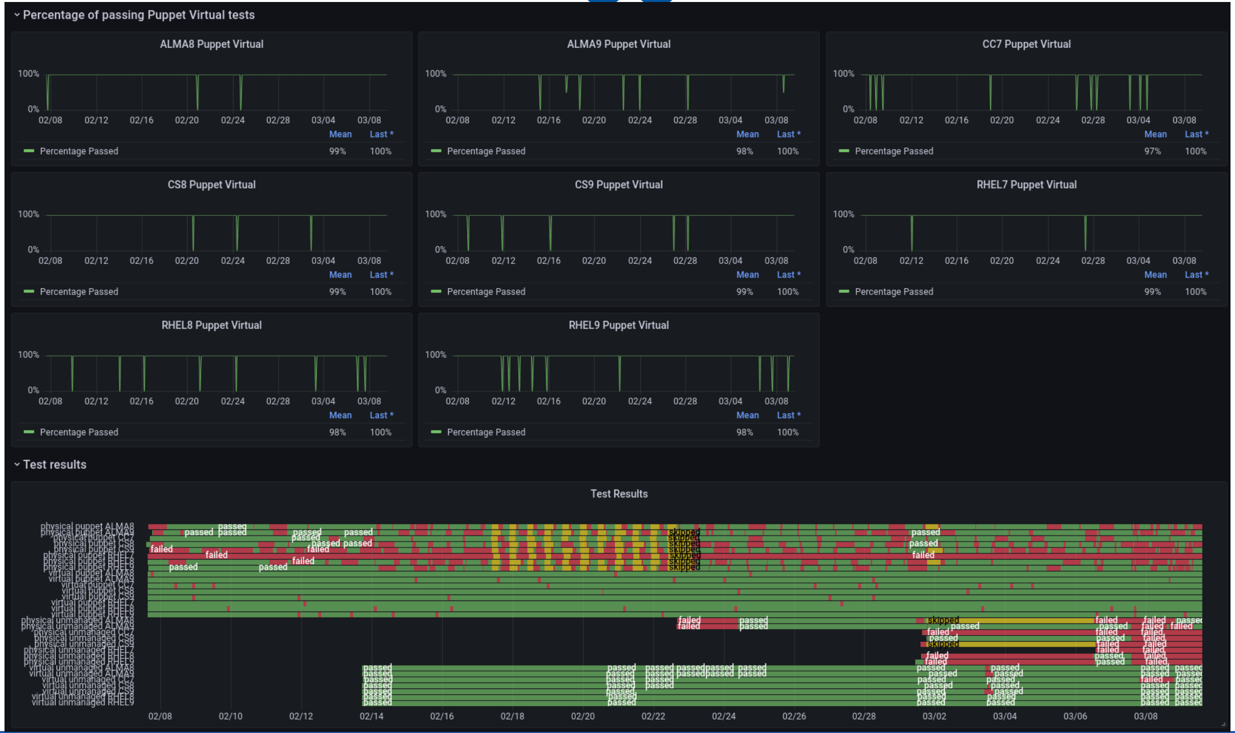
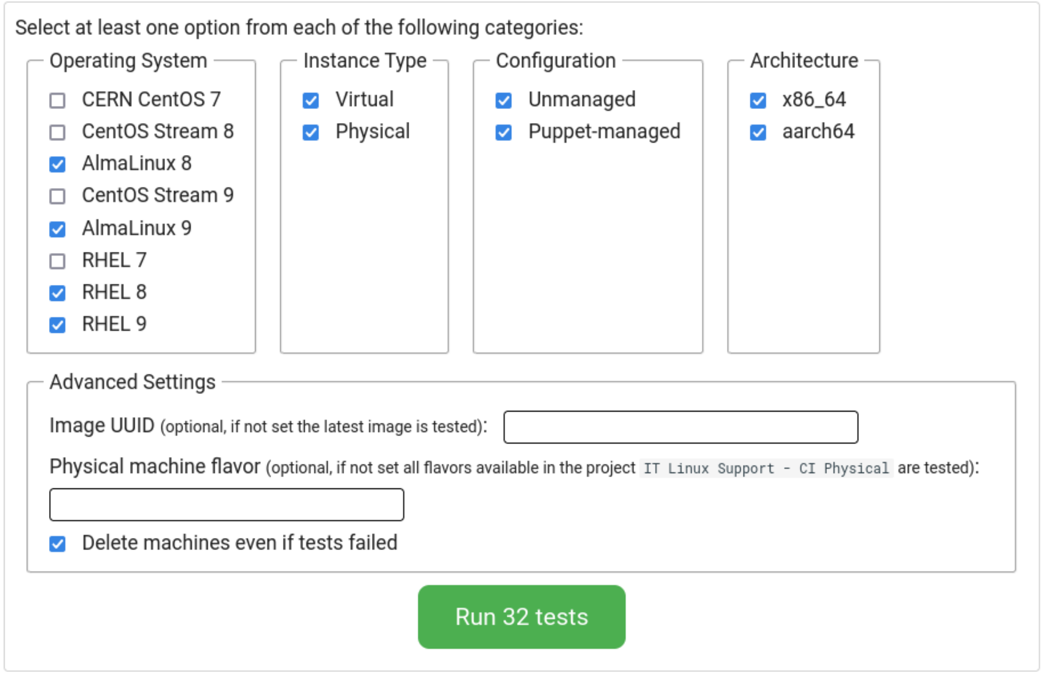


image-ci: Manual triggers?

- Manual triggers?
  - No problems!

Note: aarch64 'support' is there, but is dependent on actual aarch64 hardware being available :)



Select at least one option from each of the following categories:

Operating System
- [ ] CERN CentOS 7
- [ ] CentOS Stream 8
- [x] AlmaLinux 8
- [ ] CentOS Stream 9
- [x] AlmaLinux 9
- [ ] RHEL 7
- [x] RHEL 8
- [x] RHEL 9

Instance Type
- [x] Virtual
- [x] Physical

Configuration
- [x] Unmanaged
- [x] Puppet-managed

Architecture
- [x] x86_64
- [x] aarch64

Advanced Settings

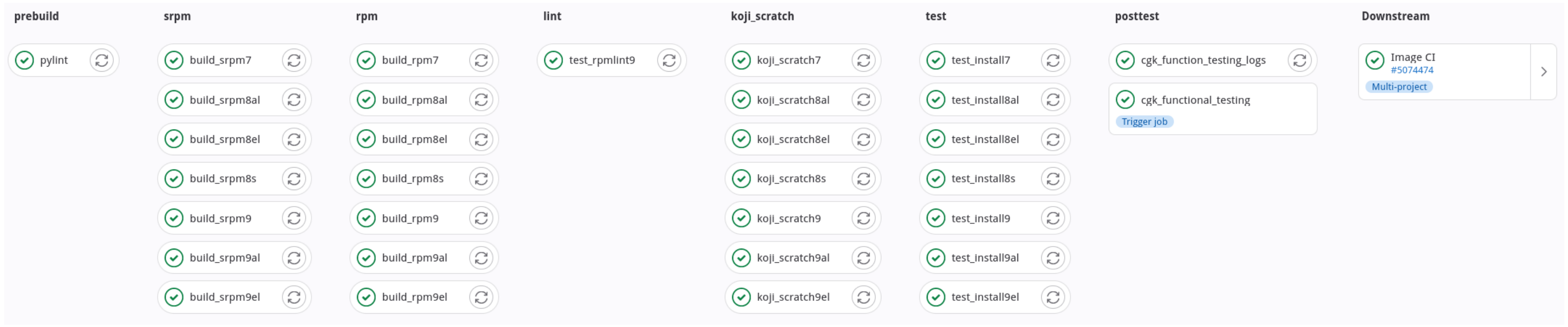
Image UUID (optional, if not set the latest image is tested):

Physical machine flavor (optional, if not set all flavors available in the project `IT Linux Support - CI Physical` are tested):

- [x] Delete machines even if tests failed

image-ci: Integration with other projects

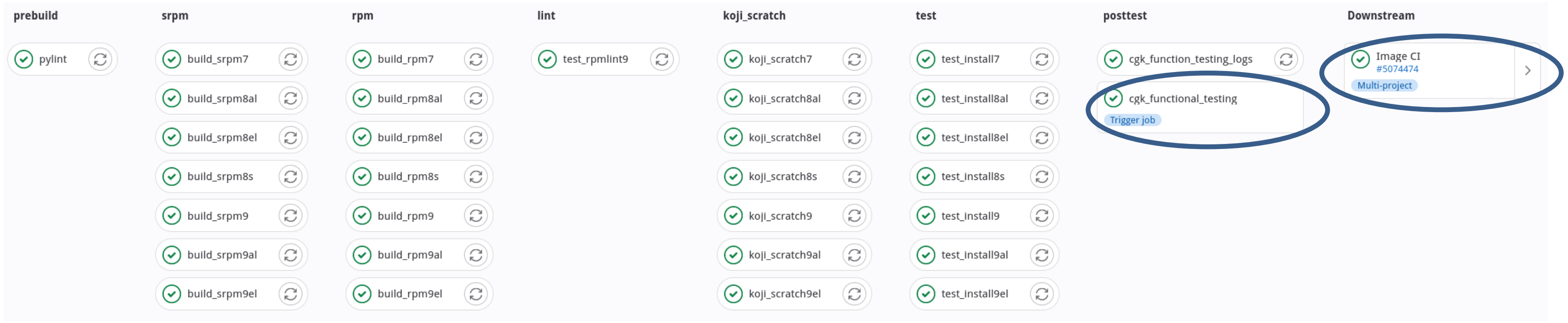
- image-ci can be even be used by other groups
- Recently cern-get-keytab was ported from perl to python
  - cern-get-keytab is often instantiated early on in a system bootstrap (cloud-init userdata)
  - How can we ensure that all functionality remains the same as the perl port?



prebuild	srpm	rpm	lint	koji_scratch	test	posttest	Downstream
✓ pylint	✓ build_srpm7	✓ build_rpm7	✓ test_rpmlint9	✓ koji_scratch7	✓ test_install7	✓ cgk_function_testing_logs	✓ Image CI #5074474 Multi-project
	✓ build_srpm8al	✓ build_rpm8al		✓ koji_scratch8al	✓ test_install8al	✓ cgk_functional_testing Trigger job	
	✓ build_srpm8el	✓ build_rpm8el		✓ koji_scratch8el	✓ test_install8el		
	✓ build_srpm8s	✓ build_rpm8s		✓ koji_scratch8s	✓ test_install8s		
	✓ build_srpm9	✓ build_rpm9		✓ koji_scratch9	✓ test_install9		
	✓ build_srpm9al	✓ build_rpm9al		✓ koji_scratch9al	✓ test_install9al		
	✓ build_srpm9el	✓ build_rpm9el		✓ koji_scratch9el	✓ test_install9el		

image-ci: Integration with other projects

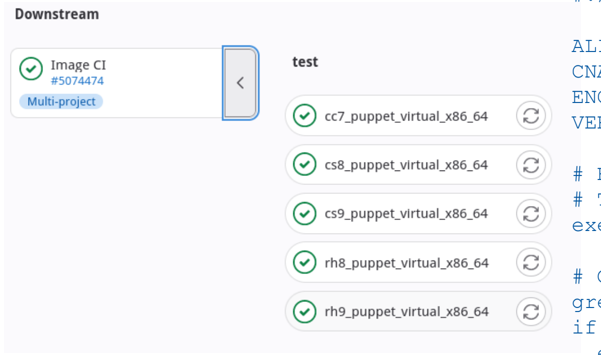
- image-ci can be even be used by other groups
- Recently cern-get-keytab was ported from perl to python
  - cern-get-keytab is often instantiated early on in a system bootstrap (cloud-init userdata)
  - How can we ensure that all functionality remains the same as the perl port?



prebuild	srpm	rpm	lint	koji_scratch	test	posttest	Downstream
✓ pylint	✓ build_srpm7	✓ build_rpm7	✓ test_rpmlint9	✓ koji_scratch7	✓ test_install7	✓ cgk_function_testing_logs	✓ Image CI #5074474 Multi-project
	✓ build_srpm8al	✓ build_rpm8al		✓ koji_scratch8al	✓ test_install8al	✓ cgk_functional_testing Trigger job	
	✓ build_srpm8el	✓ build_rpm8el		✓ koji_scratch8el	✓ test_install8el		
	✓ build_srpm8s	✓ build_rpm8s		✓ koji_scratch8s	✓ test_install8s		
	✓ build_srpm9	✓ build_rpm9		✓ koji_scratch9	✓ test_install9		
	✓ build_srpm9al	✓ build_rpm9al		✓ koji_scratch9al	✓ test_install9al		
	✓ build_srpm9el	✓ build_rpm9el		✓ koji_scratch9el	✓ test_install9el		

image-ci: Integration with other projects

- image-ci can be even be used by other groups
- Recently cern-get-keytab was ported from perl to python
  - cern-get-keytab is often instantiated early on in a system bootstrap (cloud-init userdata)
  - How can we ensure that all functionality remains the same as the perl port?


Downstream
Image CI
#5074474
Multi-project
test
cc7_puppet_virtual_x86_64
cs8_puppet_virtual_x86_64
cs9_puppet_virtual_x86_64
rh8_puppet_virtual_x86_64
rh9_puppet_virtual_x86_64

```
#!/bin/bash

ALIAS=cgkcitesting.cern.ch
CNAME=cgkcitesting-cname.cern.ch
ENCRYPTION_CIPHER=AES256_CTS_HMAC_SHA1
VERBOSE=--verbose

# Ensure logging of the script goes to /var/log/messages
# This way it's retrieved as an artifact on job success
exec 1> >(logger -s -t $(basename $0)) 2>&1

# Only attempt stuff if the imageci has already connected via SSH
grep "Authorized to root, krb5 principal imageci@CERN.CH" /var/log/secure &>/dev/null
if [ $? -ne 0 ]; then
  echo "imageci has not connected via ssh yet, exiting early"
  exit 1
fi
echo "imageci has connected via ssh! proceeding ..."
```

Benefits from adopting automation

- We can add new images to our pipelines without much effort
    - As we did recently with for; *Red Hat Enterprise Linux 9, Red Hat Enterprise Linux 8, AlmaLinux 9, AlmaLinux 8* with the recent **change of recommendation** for **Linux** at CERN [0]
- We can also add support for other architectures, such as aarch64 😄
    - There is an increasing interest and deployment of aarch64 at CERN!
    - Limited ARM hardware exists today, and more is on order. aarch64 VMs running on aarch64 CPUs for sure beats `qemu-system-aarch64` emulation
- Other teams involved in the bootstrapping and configuration of machines can test their changes using our image-ci project (or tooling such as the `cern-get-keytab` example)
- With the extensive use of GitLab CI templates we can change all the images at once
- Control image releases with a single button
- Track history of past image builds

[0] https://linux.web.cern.ch/#fermilabcern-recommendation-for-linux-distribution

Could this help **$YOU** ?

- Would image-ci, our OpenStack or docker images be useful for you or your site
- Can we help?
- Get in touch linux.team@cern.ch !

Even more tests?

- What if we could have some of the testing we do internally, done before a package is even released?🤔

- CERN has had initial discussions with the Red Hat QA team to integrate some of our 'downstream' testing, into Red Hat's QA (upstream) testing
  - CERN would do less, the community would benefit more
  - However, this hasn't be done yet as the team is/was busy with the CS8->RHEL8 workflow changes
    - Perhaps someone else from the HEP community could add pressure / push for this?

www.cern.ch