Down with the Downtime

Agile Moodle Updates with Docker and GitLab CI

Niels Gandraß

Niels.Gandrass@haw-hamburg.de
https://gandrass.de

Annual Conference of the Moodle an Hochschulen e.V.

Rosenheim Technical University of Applied Sciences

2026-03-03

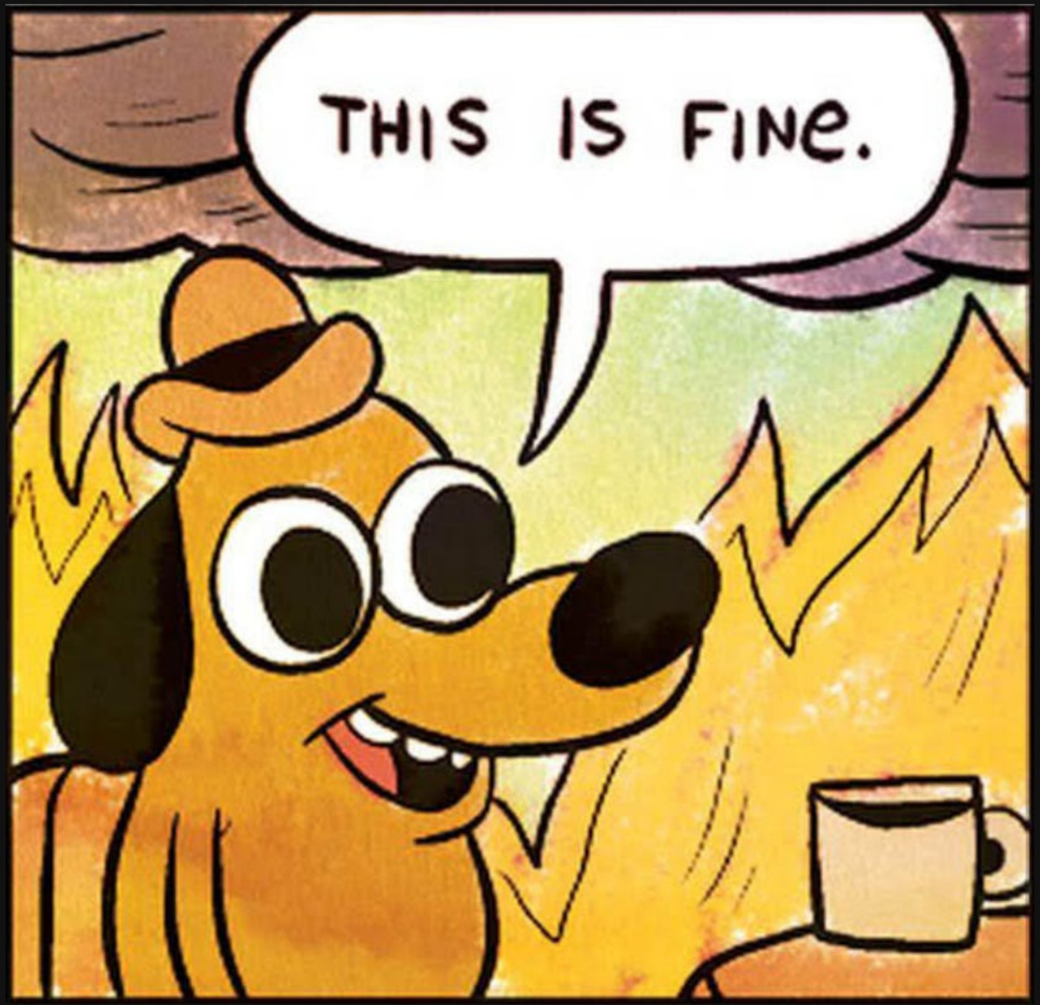
Table of Contents

A World Before Containers

How we were doing it back in the days and why we are doing it differently now

Roughly 10 years ago ...

The platform was a small project and not yet widely used by university members



The *"How not to do it"* List

1. Moodle source code was deployed to the server only once at the beginning
2. Everyone just pushed via SFTP to the server
3. Plugin updates were manually uploaded to the respective plugin folders
4. Moodle core updates where simply pushed on top of everything
   - Just hit *"override existing files"* — It'll be fine ... 🔥
   - Deleted or renamed files remained on the server indefinitely
5. Each service (web server, database, ...) was installed and updated manually
6. All applications had to share the same PHP and DB versions
7. No proper change management
8. Updates required long downtimes due to manual actions and testing

Moving in the right direction

The platform grew but we were still far from todays usage numbers

Nonetheless we needed to address many of the previously discussed problems

- Migrated Moodle core and plugin source code to git repository
- Introduction of Git Flow for change and release management
- SFTP upload replaced by git checkout
- Using multiple VMs to separate production and test systems

Remaining challenges

We now had multiple growing Moodle deployments, creating a lot of maintenance work for us

1. Updates still require a lot of manual interaction
2. Long downtimes must be scheduled and quick changes were impossible
3. Data and software versions on the servers may be different
4. No reliable way of testing code changes locally
5. Applying tested and staged changes to production was still cumbersome
6. Various security implications
   - For more details on securing your Moodle platforms see:
     Niels Gandraß, Philipp Kropp, Alexander Bias:
     Securing Moodle — A brief introduction to web application security and low-hanging fruits anyone should harvest

We are from Hamburg …
so there is an obvious solution!

Disclaimer

We do not have the time to show you each and every custom shell script or Dockerfile we created, but we will walk you through **the whole architecture** so that you **know where to start** when you want to do this yourself!

Clean and Flexible Moodle Deployments with Docker

Moving everything into containers for ease of use and flexibility with re-use in mind

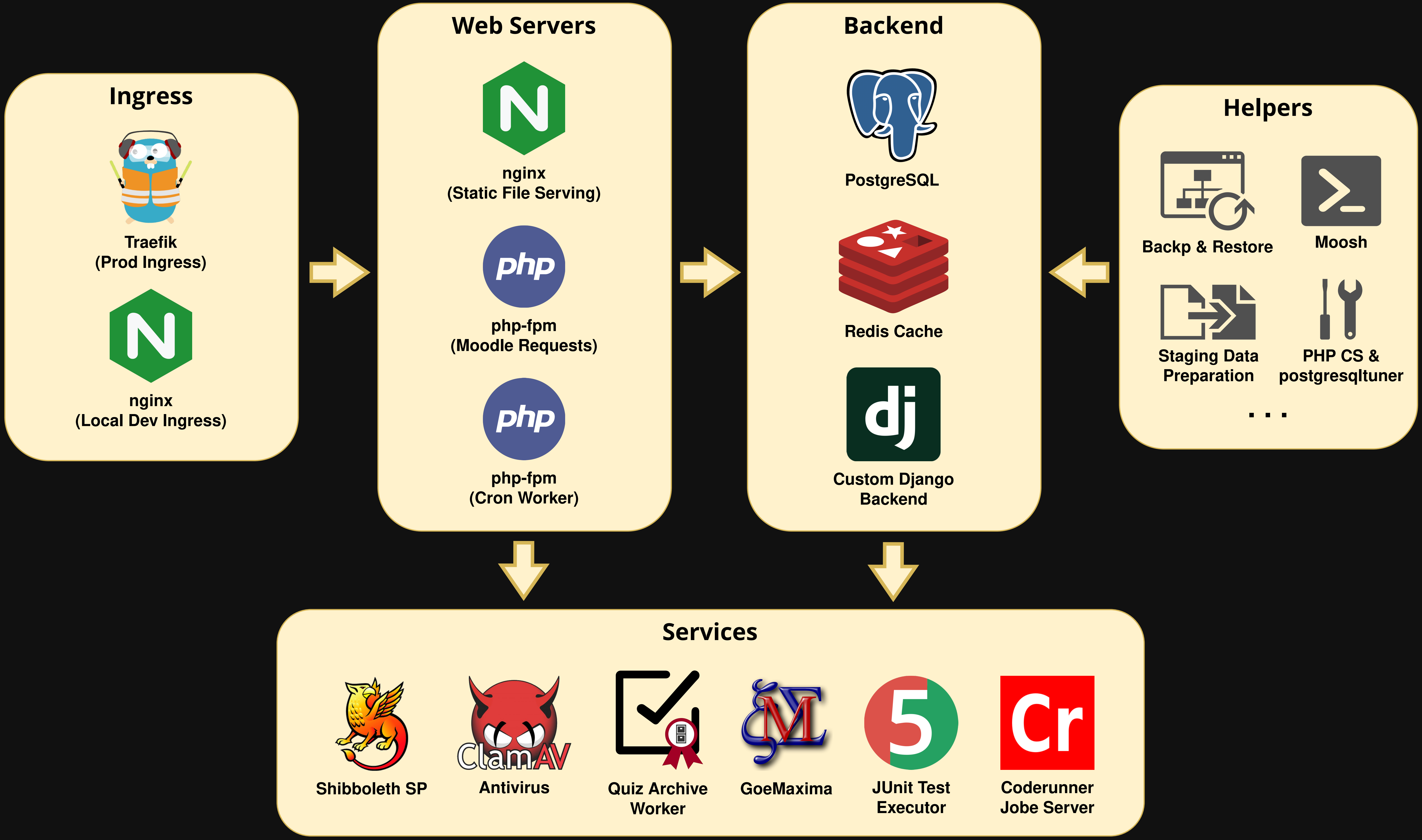
Moving everything into containers

Moodle, plugin, and software updates were still manual and this needed to change

1. Creation of Dockerfiles for all our existing services
2. Transferring the whole component management and software config to code
3. Migration of all local development environments to new Docker stack
4. Creation of custom scripts for backup and restore
5. App optimizations for container deployment
6. Replacement of existing manual VM-based deployments with Docker stacks

This resulted in the following application stack ...

Architecture of our Moodle stack



Managing Moodle and plugin versions

- Moodle core and all plugin versions are managed inside an `.env` file
- The full Moodle source tree is automatically generated during the Docker build stage
- Combination of Moodle and plugins is manifested in versioned Docker images

```
# vvvvv MOODLE_405_STABLE, release 4.5.9 vvvvv
MOODLE_GIT_HEAD=6b4d33a98f43ac19f1a37952467c8fbc722d0bdb

MOODLE_PLUGIN_QTYPE_STACK_GIT_HEAD=tags/v4.11.1
MOODLE_PLUGIN_QUIZ_ARCHIVER_GIT_HEAD=tags/v3.2.0
MOODLE_PLUGIN_THEME_BOOSTUNION_GIT_HEAD=tags/v4.5-r32
MOODLE_PLUGIN_USERAUTODELETE_GIT_HEAD=tags/v1.5.0

GOEMAXIMA_STACK_VERSION=2026010500
GOEMAXIMA_VERSION=1.2.0
QUIZ_ARCHIVE_WORKER_VERSION=3.3.9
```

Excerpt of an example mymoodle.env file

Managing software versions

- Software versions are defined in the same `.env` file
- Effortlessly switching between different PHP / DB / … versions
- Additional tools, e.g., XDebug, can be included in local dev builds

```
# Selection of software versions
PHP_FPM_VERSION=8.3
POSTGRES_VERSION=18-alpine
CLAMAV_VERSION=latest

# Service-specific settings
BACKUP_COMPRESS=1
BACKUP_CPU_THREADS=8
MOODLE_ALLOW_USER_CREATION_ON_RESTORE=false
MOODLE_DATABASE_NAME=myawesomemoodle
```

Excerpt of an example mymoodle.env file

Tying everything together with Docker Compose

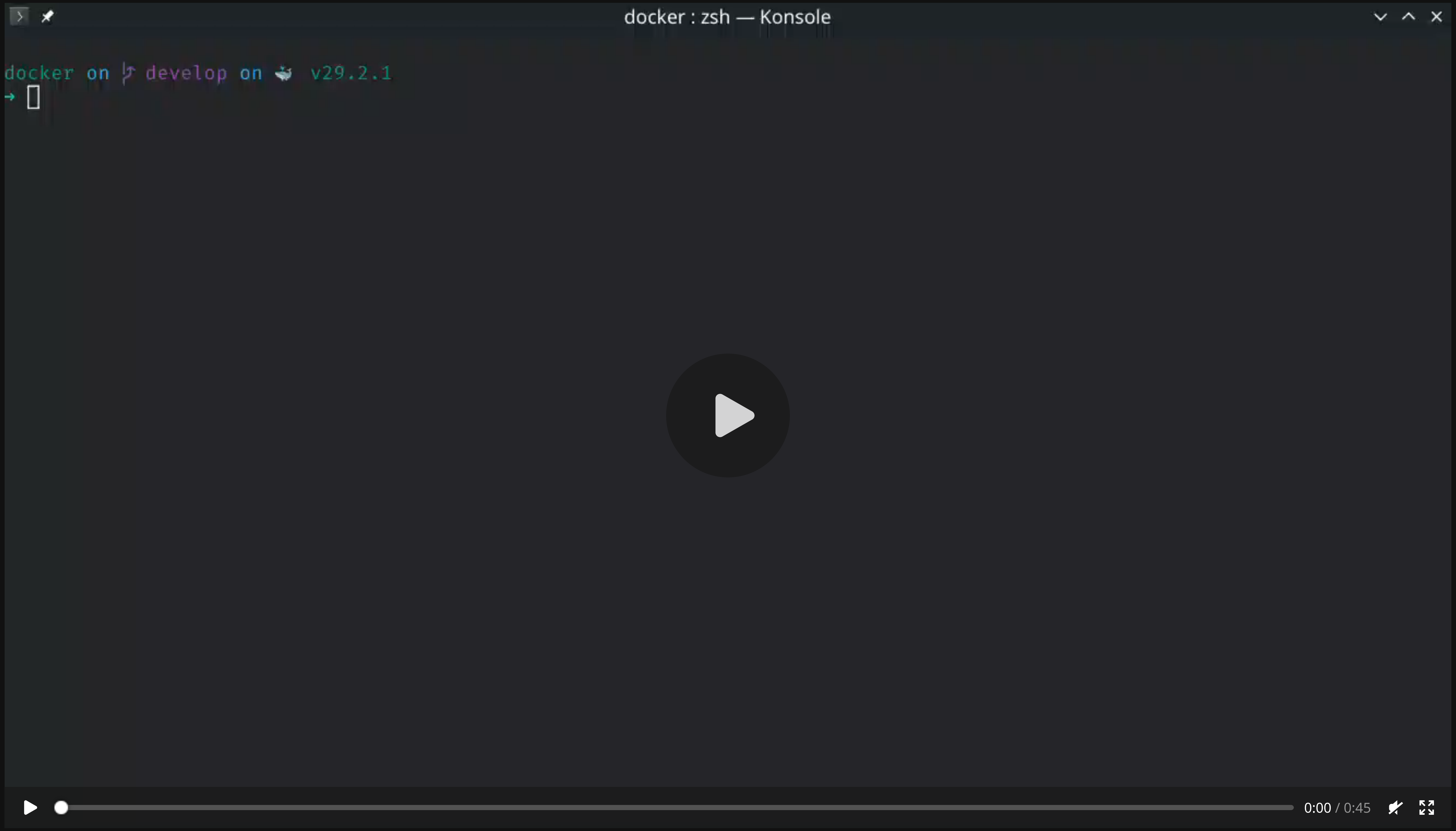
- All available services are defined inside a `docker-compose.yml`
- Single services can selectively be (de-)activated using compose profiles
- A deployment can be created using a single `docker compose up` command
  - In production we use additional helper scripts to perform sanity checks before touching a Moodle stack though

```
# Selection of services to run
COMPOSE_PROFILES=moodle,database,redis,shibboleth-sp
```

Excerpt of an example mymoodle.env file

Running a small Moodle stack

This example is used for local development of the Quiz Archiver plugin



```
docker on  develop on  v29.2.1
→ 
```

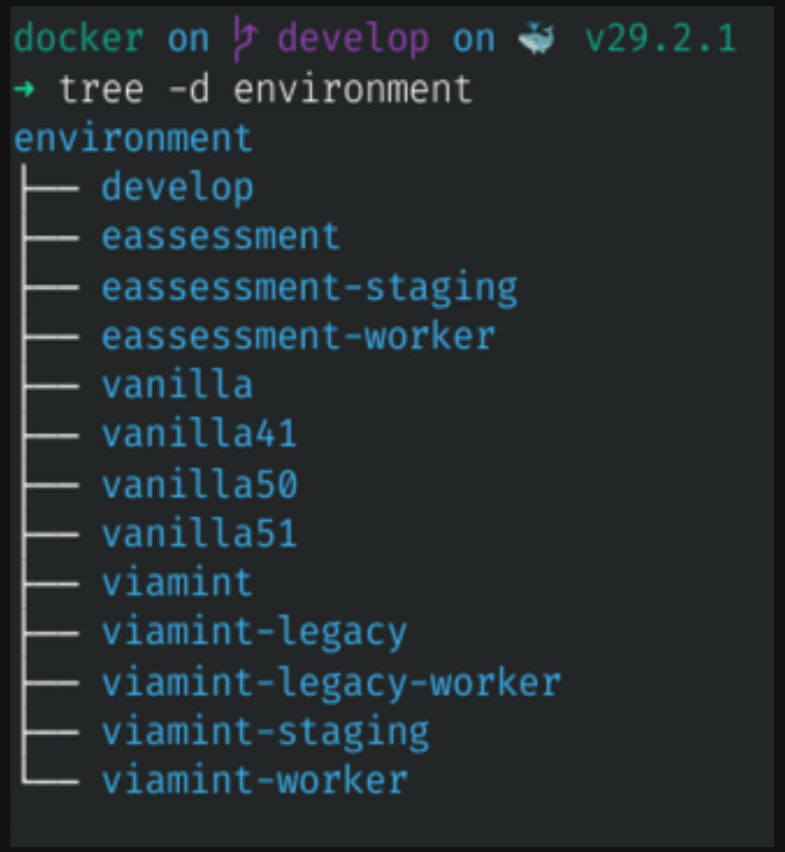
Running Multiple Moodle Platforms

Effortlessly create fully-compatible production, staging, test, and development deployments

Separation into Moodle environments

Using the same Docker stack for multiple different Moodle deployments

- Common `docker-compose.yml` and `.env` files can be extended and customized for every environment
  - Independent Moodle, plugin, and software versions
  - Further environment specific overrides
- Final config is generated from selected environment
- Only differences need to be modeled
- All high-level interfaces (e.g., backup) remain the same
- Envorinment configs are *"diffable"* (text based) and versioned inside a git repository



```
docker on ᚴ develop on 🐳 v29.2.1
➜ tree -d environment
environment
├── develop
├── eassessment
├── eassessment-staging
├── eassessment-worker
├── vanilla
├── vanilla41
├── vanilla50
├── vanilla51
├── viamint
├── viamint-legacy
├── viamint-legacy-worker
├── viamint-staging
└── viamint-worker
```

Example of multiple environments

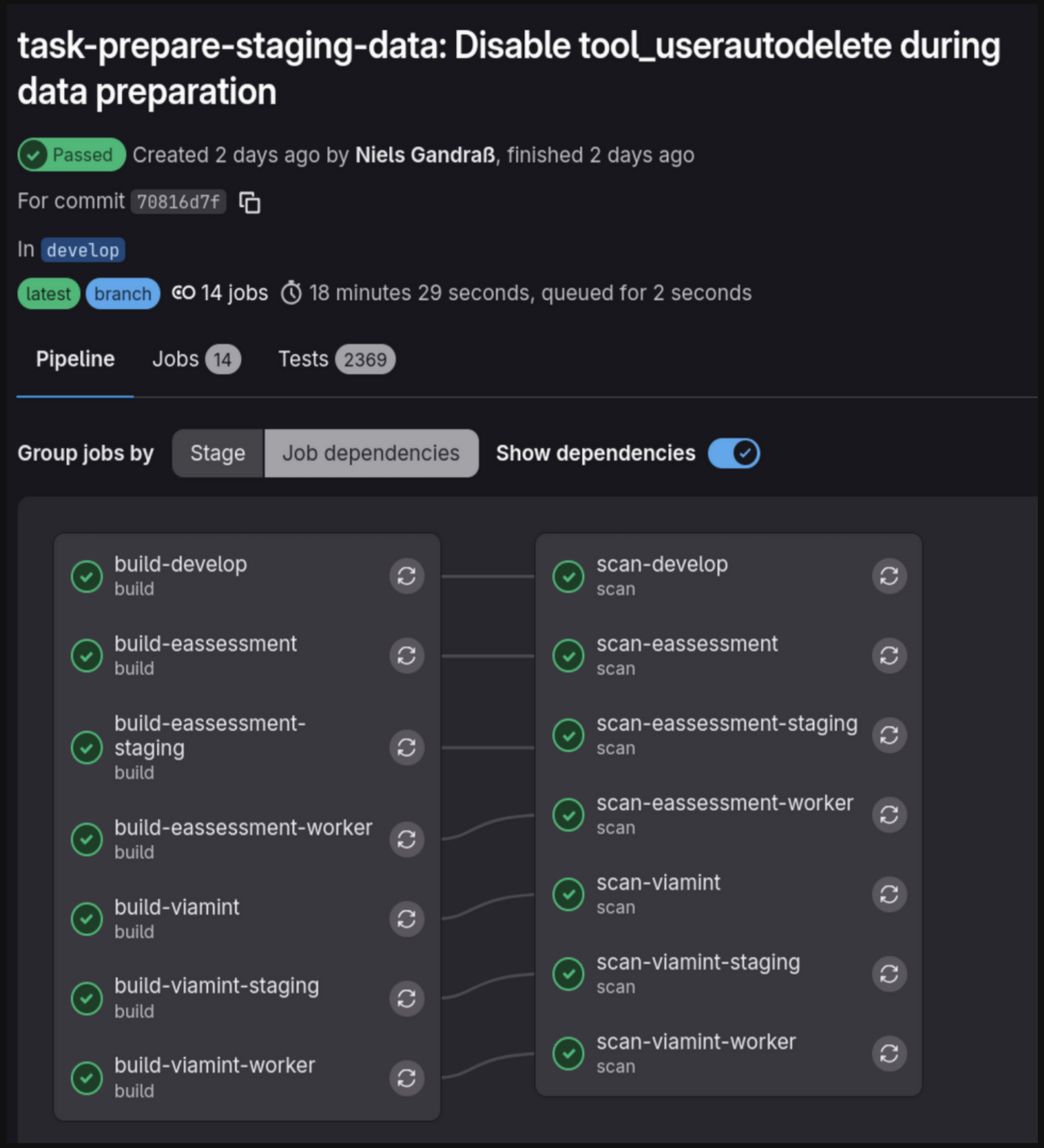
Automating Builds

Using GitLab CI to always have the latest container images at hand

Getting started ...

Finally letting the server do the work for us!

1. Create a `.gitlab-ci.yml` for your Docker build
   - Use DinD for job / build separation
   - Selectively use shared caches for speedup
2. Add additional test stages as desired
3. Tag images appropriately
   - Unique version tags for releases
   - Permanent tags for selected branches
   - Short-lived tags for merge requests
4. Push images to Docker registry
5. Deploy 🚀



task-prepare-staging-data: Disable tool_userautodelete during data preparation

Passed Created 2 days ago by **Niels Gandraß**, finished 2 days ago

For commit `70816d7f`

In `develop`

latest branch 14 jobs 18 minutes 29 seconds, queued for 2 seconds

build-develop build — scan-develop scan

build-eassessment build — scan-eassessment scan

build-eassessment-staging build — scan-eassessment-staging scan

build-eassessment-worker build — scan-eassessment-worker scan

build-viamint build — scan-viamint scan

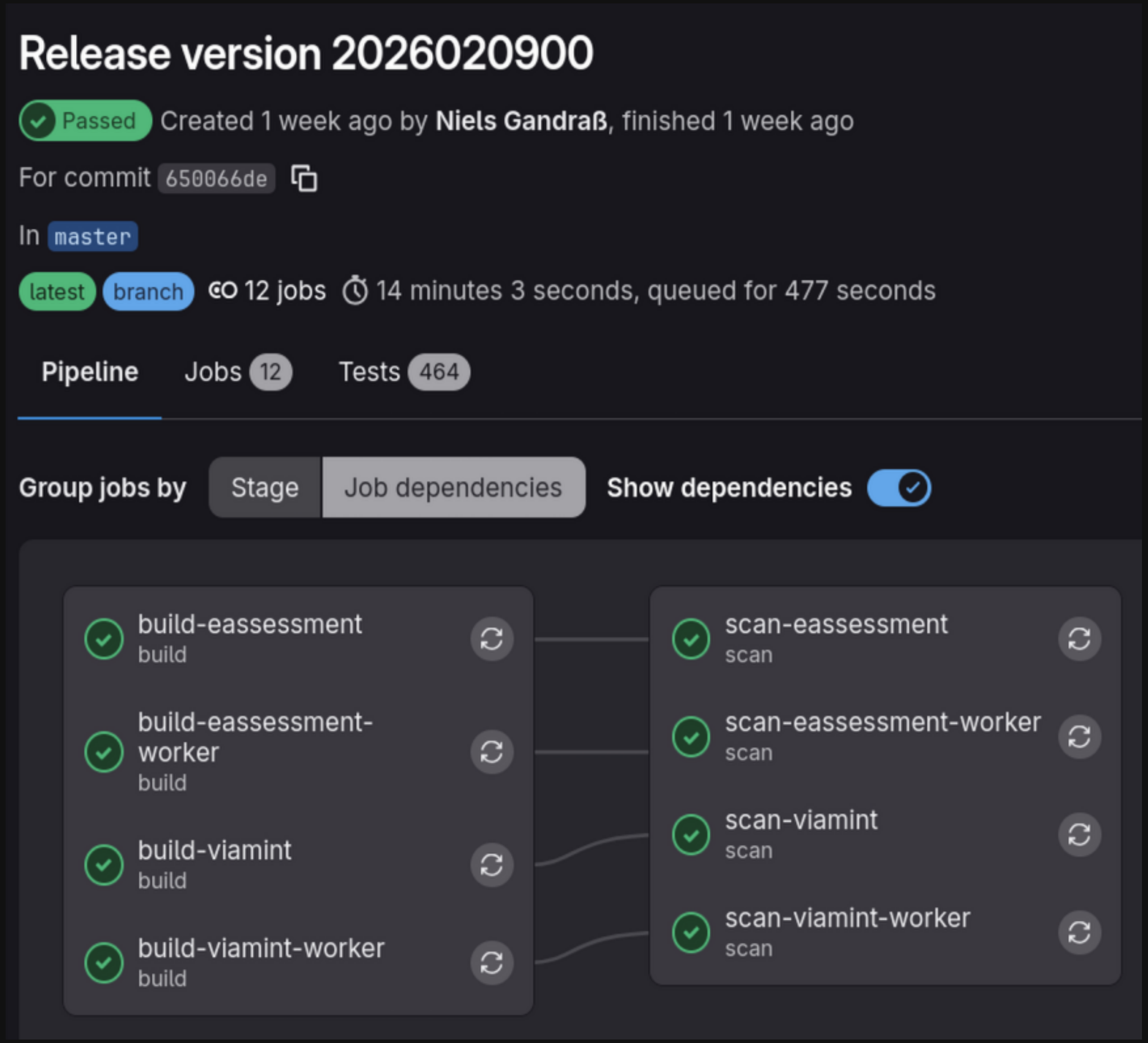
build-viamint-staging build — scan-viamint-staging scan

build-viamint-worker build — scan-viamint-worker scan

Example of a full build

Builds are selective

Only images that faced changes are built for speedup and resource conservation



Release version 2026020900

Passed Created 1 week ago by **Niels Gandraß**, finished 1 week ago

For commit `650066de`

In `master`

latest branch 12 jobs 14 minutes 3 seconds, queued for 477 seconds

Pipeline Jobs 12 Tests 464

Group jobs by Stage Job dependencies Show dependencies

build-eassessment
build

build-eassessment-worker
build

build-viamint
build

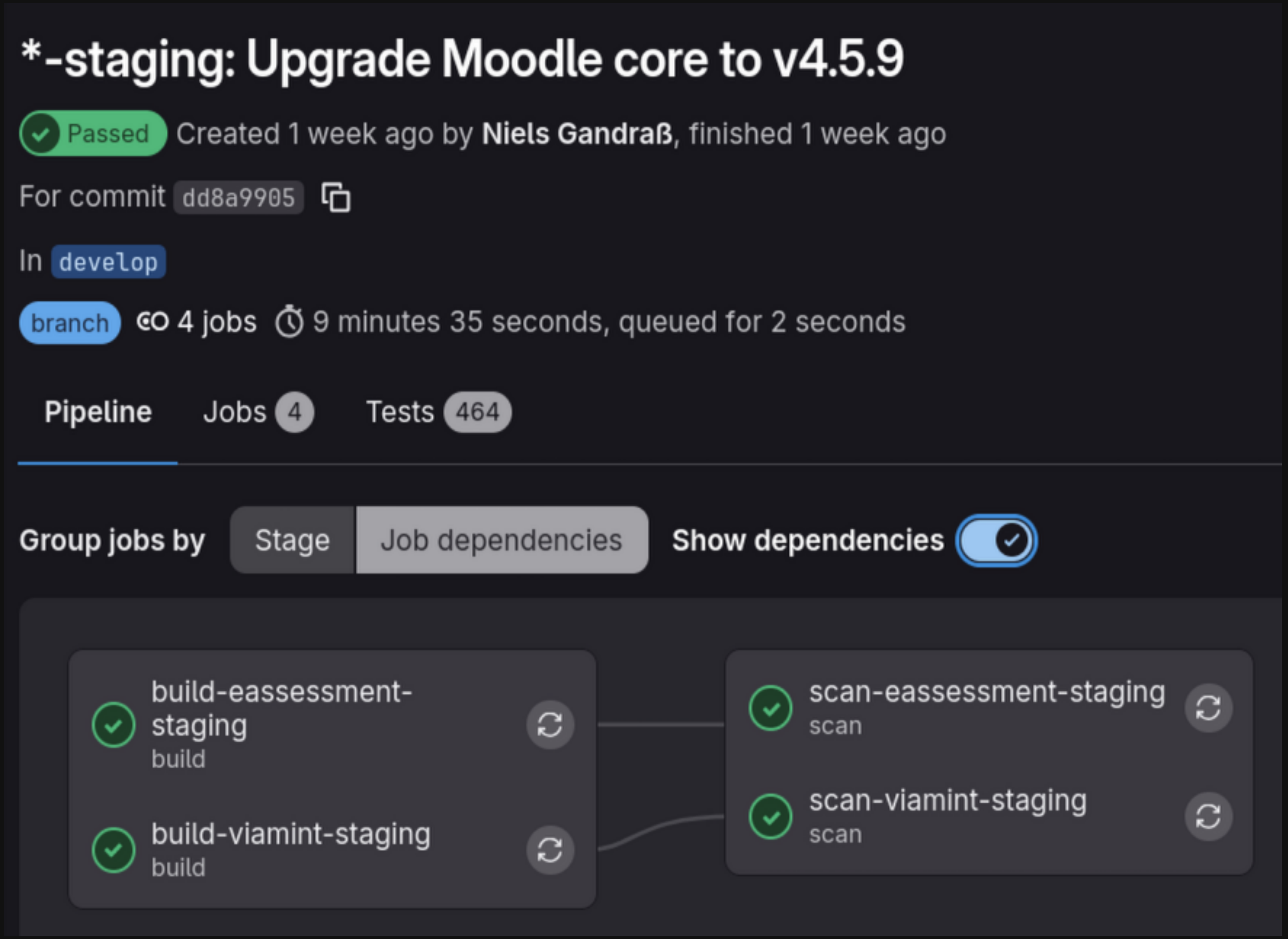
build-viamint-worker
build

scan-eassessment
scan

scan-eassessment-worker
scan

scan-viamint
scan

scan-viamint-worker
scan



***-staging: Upgrade Moodle core to v4.5.9**

Passed Created 1 week ago by **Niels Gandraß**, finished 1 week ago

For commit `dd8a9905`

In `develop`

branch 4 jobs 9 minutes 35 seconds, queued for 2 seconds

Pipeline Jobs 4 Tests 464

Group jobs by Stage Job dependencies Show dependencies

build-eassessment-staging
build

build-viamint-staging
build

scan-eassessment-staging
scan

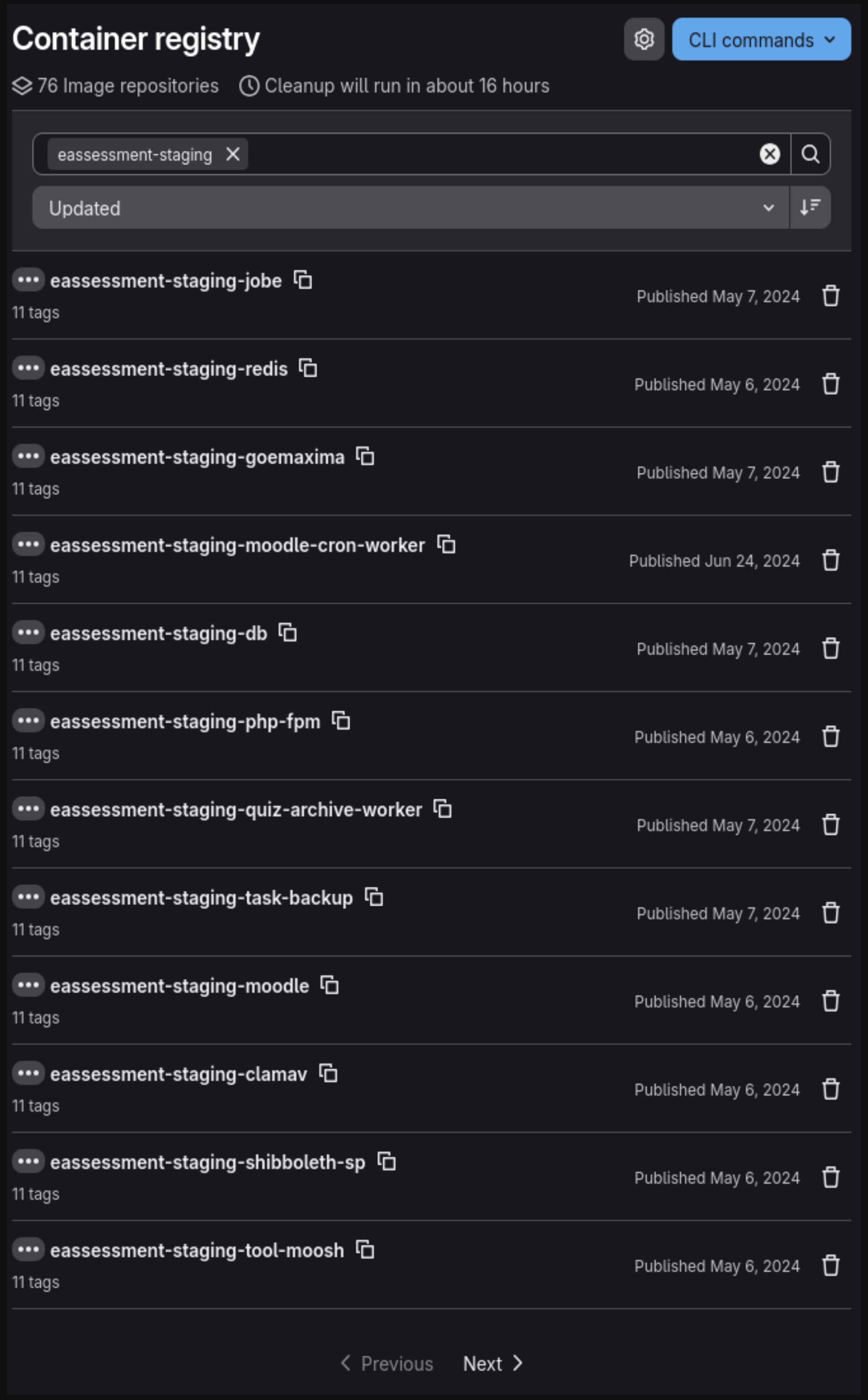
scan-viamint-staging
scan

Pre-built images via the GitLab registry

- Deployment-ready images are available via the GitLab container registry
- Specific builds can be pulled by developers
- Images are automatically scanned for known vulnerabilities and SBOMs can be generated
- Old images are automatically deleted according to a configured cleanup policy
- Image source can be customized (e.g., for local development or CI environments)

```
# Special environmental configurations for GitLab CI/CD
DOCKER_REGISTRY_PATH=${CI_REGISTRY_IMAGE}
DOCKER_UPSTREAM_REGISTRY_PREFIX=${CI_DEPENDENCY_PROXY_GROUP_IMAGE_PREFIX}/
```

Excerpt of gitlab-ci.env



Container registry

76 Image repositories · Cleanup will run in about 16 hours

eassessment-staging

Updated

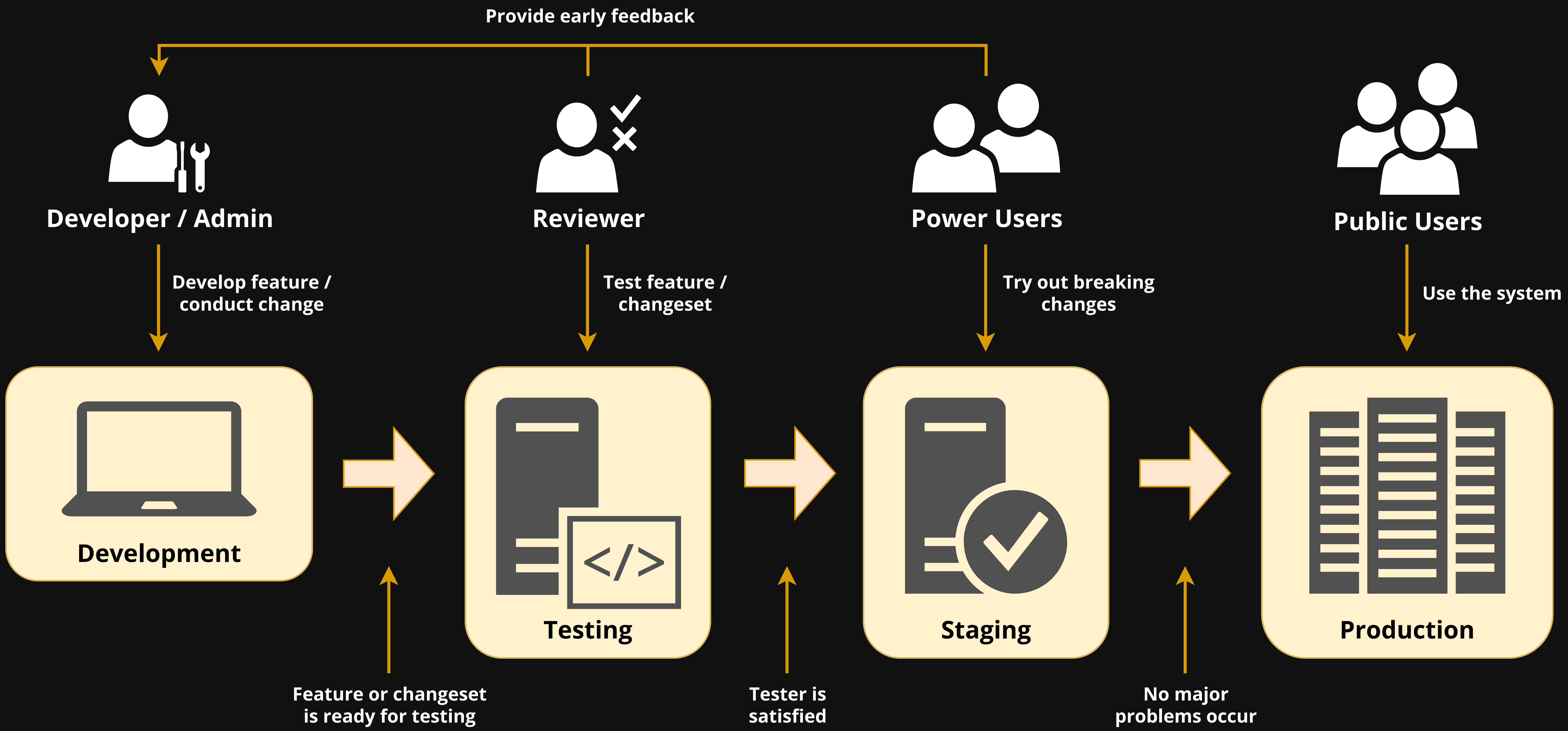
eassessment-staging-jobe	11 tags	Published May 7, 2024
eassessment-staging-redis	11 tags	Published May 6, 2024
eassessment-staging-goemaxima	11 tags	Published May 7, 2024
eassessment-staging-moodle-cron-worker	11 tags	Published Jun 24, 2024
eassessment-staging-db	11 tags	Published May 7, 2024
eassessment-staging-php-fpm	11 tags	Published May 6, 2024
eassessment-staging-quiz-archive-worker	11 tags	Published May 7, 2024
eassessment-staging-task-backup	11 tags	Published May 7, 2024
eassessment-staging-moodle	11 tags	Published May 6, 2024
eassessment-staging-clamav	11 tags	Published May 6, 2024
eassessment-staging-shibboleth-sp	11 tags	Published May 6, 2024
eassessment-staging-tool-moosh	11 tags	Published May 6, 2024

Docker images in GitLab registry

Establishing a Release Process

Versioning your deployments, getting the code out there, and migrating Moodle

A full multi-stage release process



Getting code onto the staging systems

1. Tag a release for the component you'd like to change *(if required)*

```
$ git commit -m "My plugin release!"
$ git tag v3.2.0
$ git push --tags
```

2. Change the version inside the desired `.env` file

```
MOODLE_PLUGIN_BLOCK_SYNTAXHELP_GIT_HEAD=tags/v2.3.2
-MOODLE_PLUGIN_MOD_QUIZ_ARCHIVER_GIT_HEAD=tags/v3.1.4
+MOODLE_PLUGIN_MOD_QUIZ_ARCHIVER_GIT_HEAD=tags/v3.2.0
MOODLE_PLUGIN_THEME_EASSESSMENT_GIT_HEAD=tags/v1.1.5

-# vvvvv MOODLE_405_STABLE, release 4.5.8+ vvvvv
-MOODLE_CORE_GIT_HEAD=23ad73ee5677a0d0bf1039182a4e87868d4ccdf5
+# vvvvv MOODLE_405_STABLE, release 4.5.9 vvvvv
+MOODLE_CORE_GIT_HEAD=6b4d33a98f43ac19f1a37952467c8fbc722d0bdb
```

3. Commit and push the changes. Then let CI do the heavy lifting 😎
4. Pull new images and replace existing containers with new ones
5. Be happy ❤️

Applying staged changes to production builds

1. Apply all staged changes to the production `.env` file (or "copy & paste")

```
$ vim -d eassessment-staging.env eassessment.env
$ git commit -m "eassessment: Apply staged changes"
```

2. Create and push a new production release

```
$ git commit -m "Release version 2026030200"
$ git tag 2026030200
$ git push --tags
```

Yep, that's all!

Your production build is now at the exact state of your staging build

Rolling out

1. Pull the latest release of images from the registry in advance

```
$ ./pull.sh myenvironment
```

Downtime

2. Replace existing containers with new ones

```
$ ./update.sh myenvironment
```

3. Let Moodle finish the database migration *(if any)*

That's it 🚀

In our case, step 2 & 3 usually complete in under 60 seconds!

Summary

- Platform updates usually take no longer than 60 seconds
    - Longer downtimes only required for complex updates / changes
- Maintenance work was drastically reduced
- Error-rate of production upgrades went down to near zero
- Established a proper change management for code and software
- Security and resilience of our infrastructure increased
- Power users are happy to be heard early in the process
- Overall product cycles became way more agile
- We had fun doing it ...
- ... and we have fun working with it ;)

Resources and Q&A Session

That was a lot of information in a short amount of time …

Resources

Further information to dig deeper
Click on a link to open

- Docker Containers for Moodle Developers (moodlehq/moodle-docker)
- Tutorial: Create and run your first GitLab CI/CD pipeline
- Using Docker to build Docker images within GitLab CI/CD
- Administrating Moodle via command line
- Getting started with Docker Compose
- Niels Gandraß, Philipp Kropp, Alexander Bias: Securing Moodle — A brief introduction to web application security and low-hanging fruits anyone should harvest

Q&A Session

Now it's time for your questions!

You can find all slides of this talk at
https://gandrass.de