McGill | Information Technology Services

How McGill University manages, tests and deploys 1,000 Drupal websites with Ansible and Gitlab CI

DrupalCon Portland 2024

7 May | 16:10 PM

Connect.
Learn.
Innovate.

About me

Thomas Fline

Web Developer Analyst

McGill University, Montreal, Canada

thomas.fline@mcgill.ca

https://drupal.org/u/fengtan

https://github.com/fengtan

https://linkedin.com/in/thomasfline

https://www.mcgill.ca/it

McGill University

One of the largest universities in Canada:

- 12 faculties
- 1,200 programs
- 10,000 courses
- 40,000 students
- 12,000 staff

- 1,000 Drupal websites
- 1,500 site managers
- 9 million page views/month
- 500,000 pages (published nodes)
- 30,000 blocks

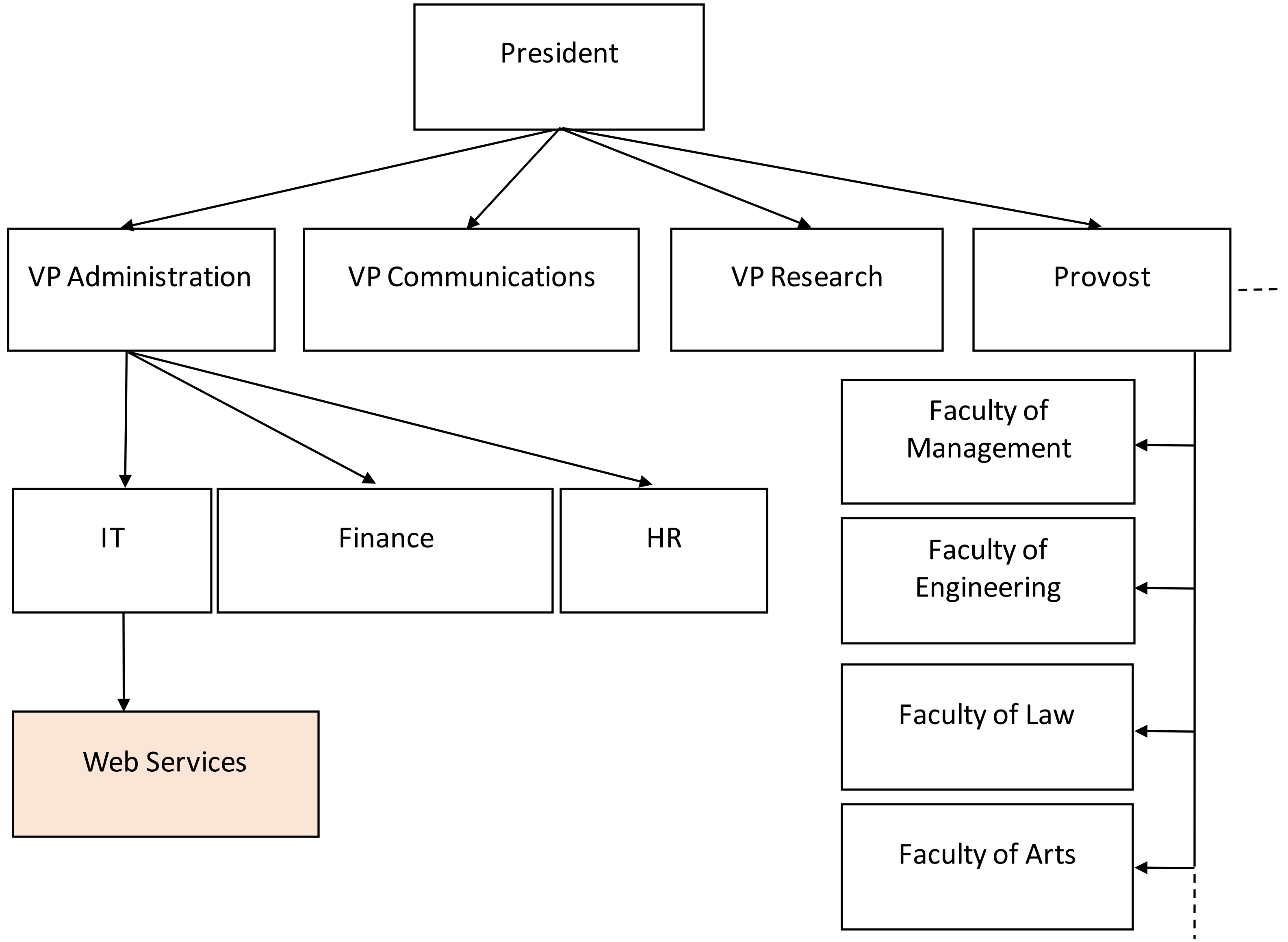
McGill University Web Services

Part of central IT services.

Offer uniform, Drupal-based websites to departments of the University.

10 team members including:

- Manager
- Backend developers
- Frontend developers
- Support analysts



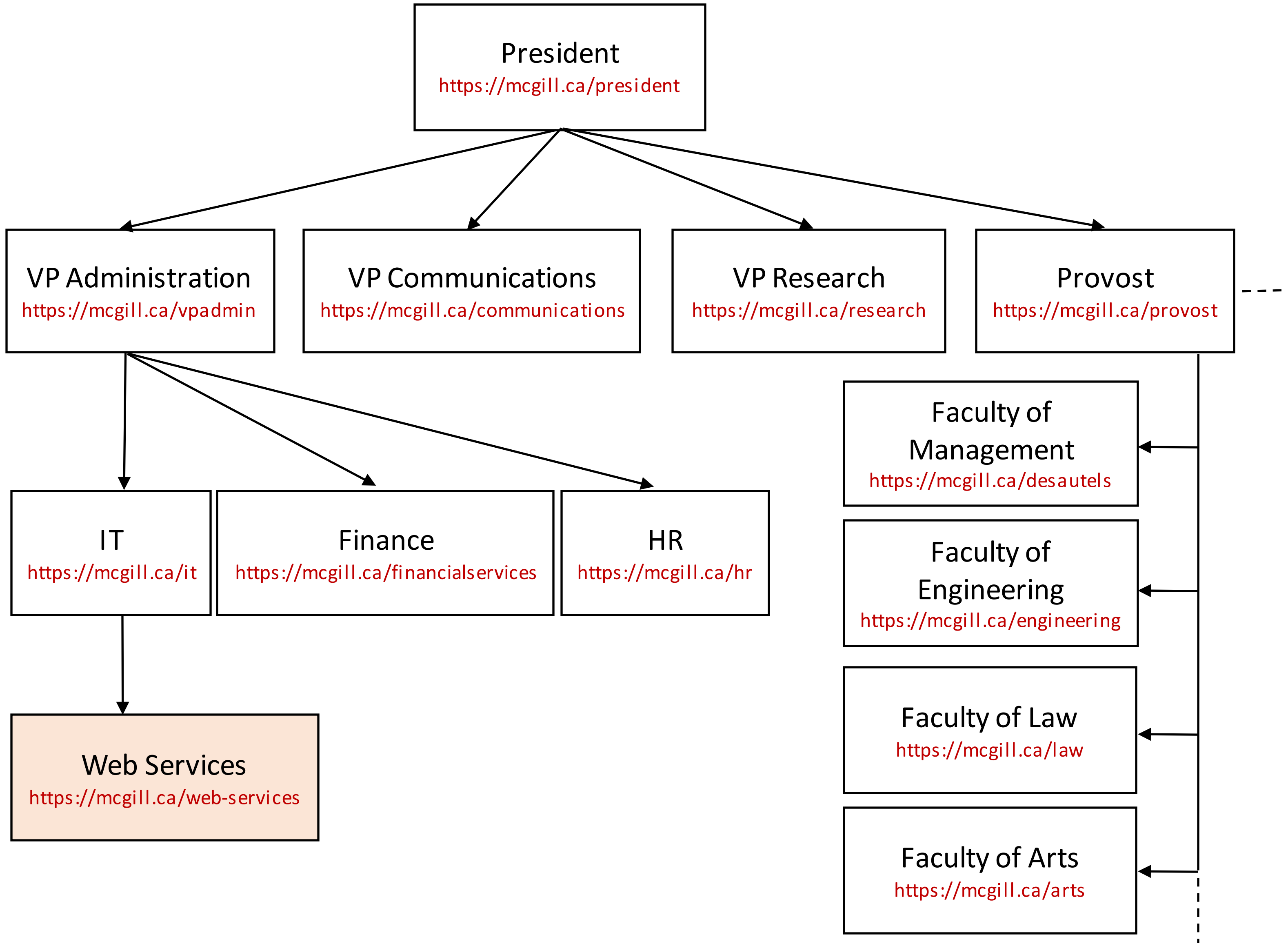
McGill University Web Services

Part of central IT services.

Offer uniform, Drupal-based websites to departments of the University.

10 team members including:

- Manager
- Backend developers
- Frontend developers
- Support analysts


President
https://mcgill.ca/president
VP Administration
https://mcgill.ca/vpadmin
VP Communications
https://mcgill.ca/communications
VP Research
https://mcgill.ca/research
Provost
https://mcgill.ca/provost
IT
https://mcgill.ca/it
Finance
https://mcgill.ca/financialservices
HR
https://mcgill.ca/hr
Web Services
https://mcgill.ca/web-services
Faculty of Management
https://mcgill.ca/desautels
Faculty of Engineering
https://mcgill.ca/engineering
Faculty of Law
https://mcgill.ca/law
Faculty of Arts
https://mcgill.ca/arts

Website examples

Faculties

- https://www.mcgill.ca/arts
- https://www.mcgill.ca/music
- https://www.mcgill.ca/science

Administration

- https://www.mcgill.ca/hr
- https://www.mcgill.ca/it

Student life

- https://www.mcgill.ca/campus-life
- https://www.mcgill.ca/studentservices

Departments of study

- https://www.mcgill.ca/surgery
- https://www.mcgill.ca/philosophy
- https://www.mcgill.ca/geography

Course catalog and admissions

- https://www.mcgill.ca/study
- https://www.mcgill.ca/admissions
- https://www.mcgill.ca/exams

Services

- https://www.mcgill.ca/directory
- https://www.mcgill.ca/search

Museums

- https://www.mcgill.ca/redpath
- https://www.mcgill.ca/medicalmuseum

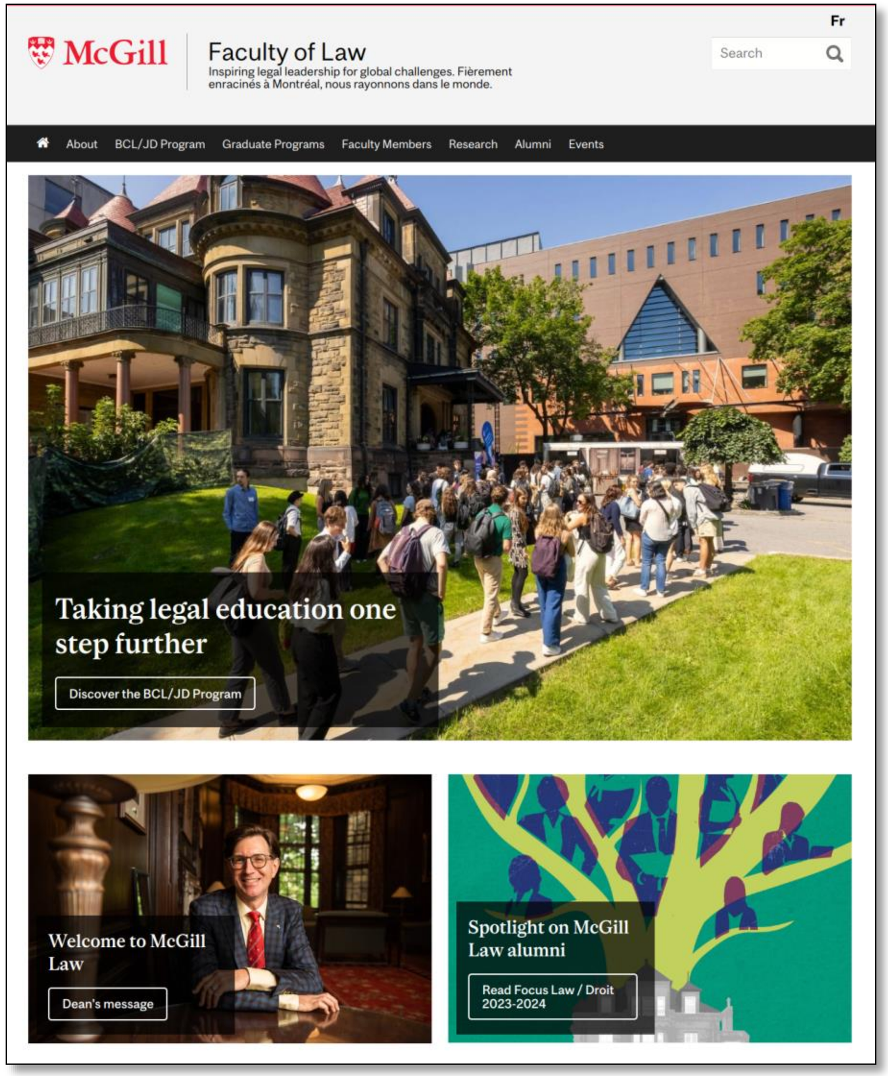
External relations

- https://www.mcgill.ca/newsroom

Research labs, conferences, etc...

Website examples

https://www.mcgill.ca/law



Faculty of Law

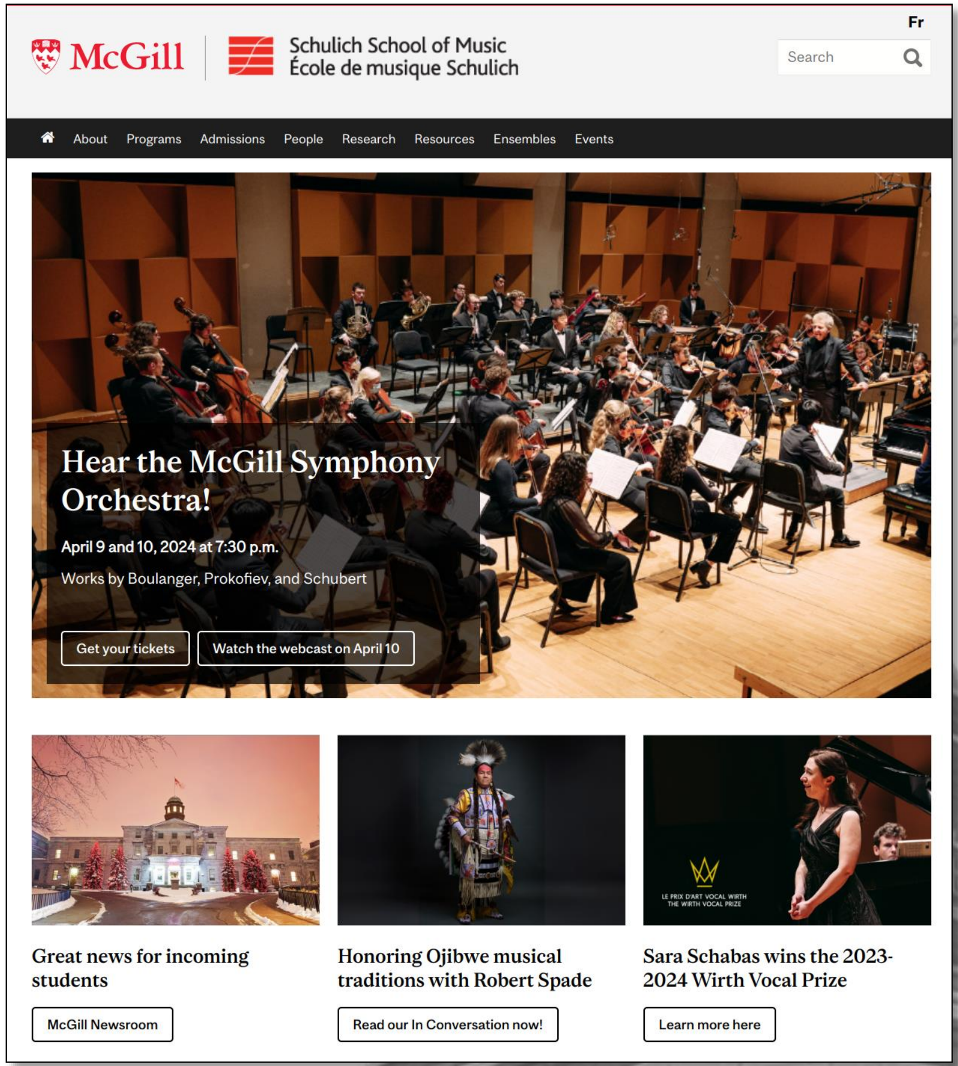
Inspiring legal leadership for global challenges. Fièrement enracinés à Montréal, nous rayonnons dans le monde.

Taking legal education one step further

Welcome to McGill Law

Spotlight on McGill Law alumni

https://www.mcgill.ca/music



Schulich School of Music
École de musique Schulich

Hear the McGill Symphony Orchestra!

April 9 and 10, 2024 at 7:30 p.m.

Works by Boulanger, Prokofiev, and Schubert

Great news for incoming students

Honoring Ojibwe musical traditions with Robert Spade

Sara Schabas wins the 2023-2024 Wirth Vocal Prize

Agenda

1. Deployment: infrastructure
2. Deployment: application
3. Site Management
4. Development
5. Tests & Quality

Deployment: infrastructure

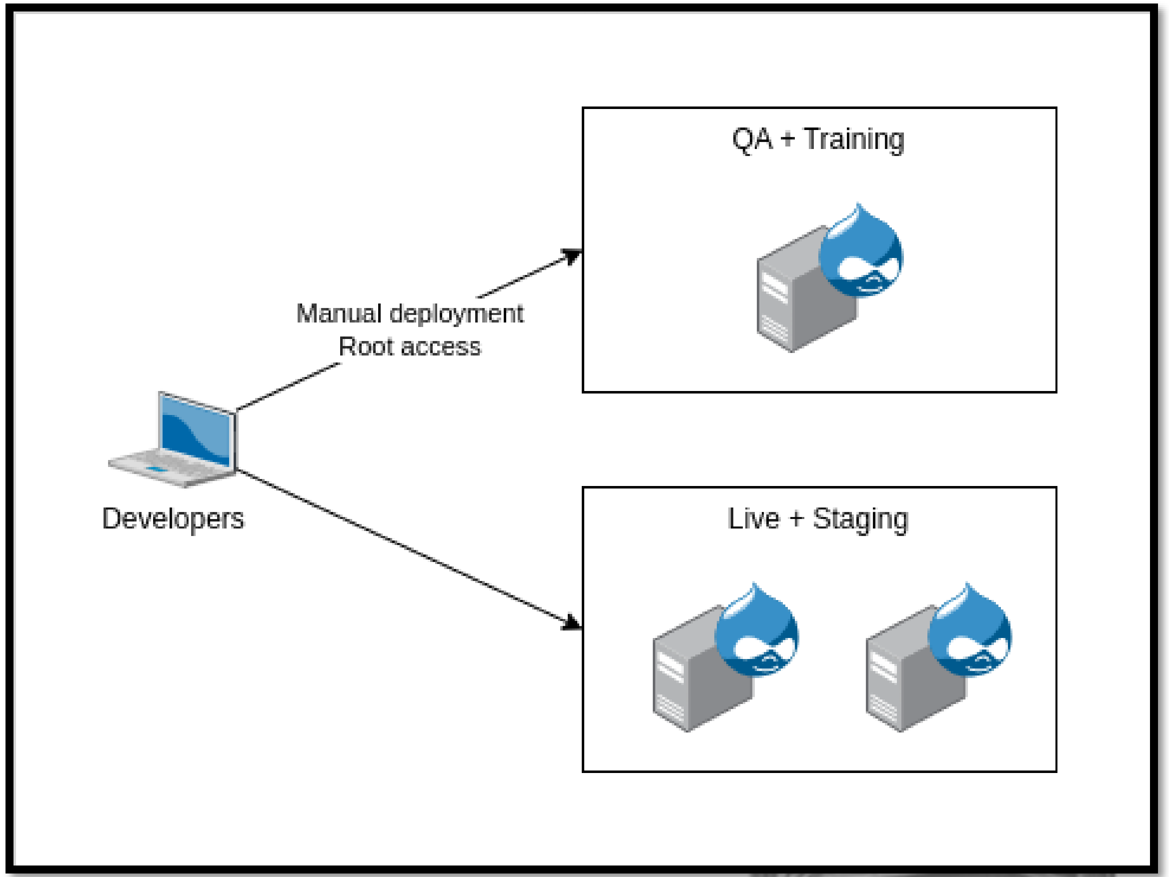
Manual deployment

Deploying the infrastructure was:

- Manual
- With privilege escalation (sudo root)
- On machines hosting multiple environments

Problems:

- No guarantee that environments are alike
- No guarantee that all servers in a given environment are alike
- Inappropriate permissions
- Lack of transparency
- No trace of changes

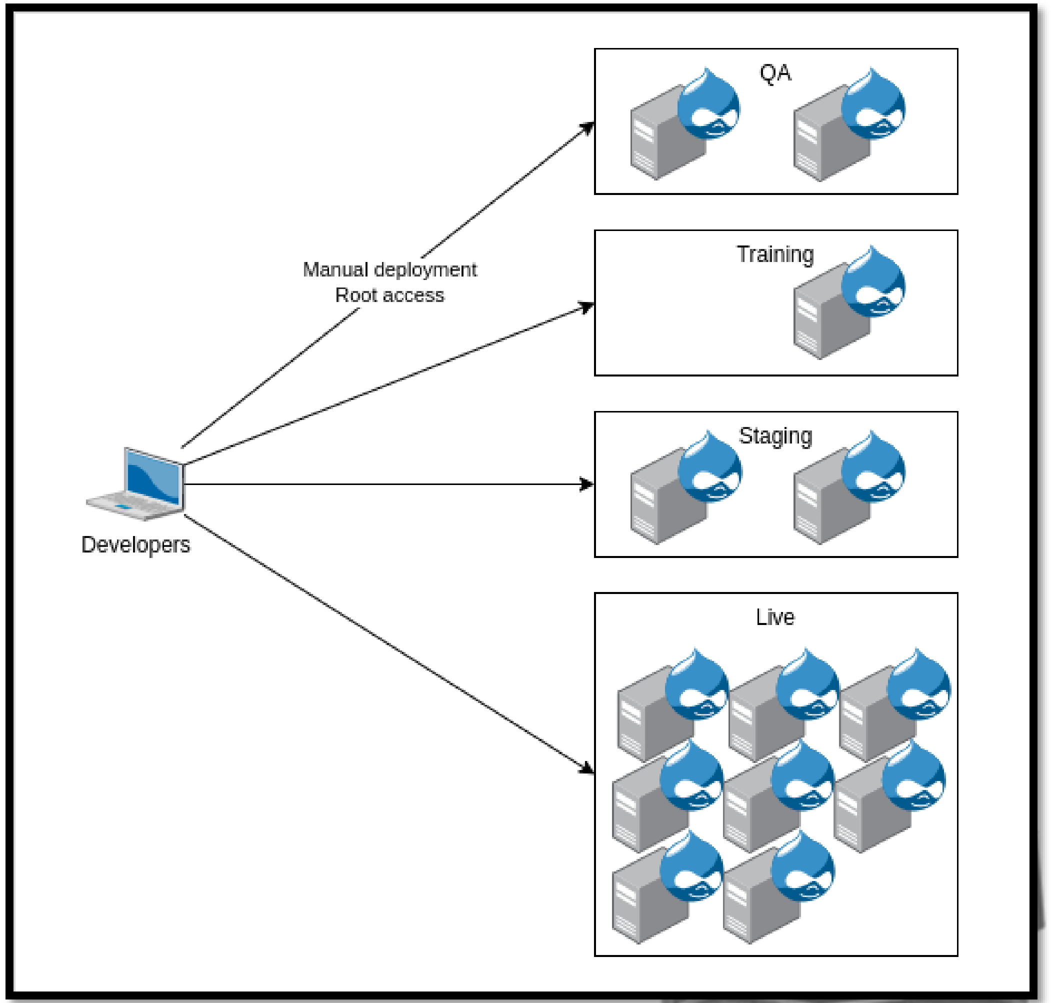


Segregated environments

Switched to VMs and segregated environments.

Made things worse: we ended up having to configure 13 machines manually.

Need for automation.



Ansible

Agentless configuration management system.

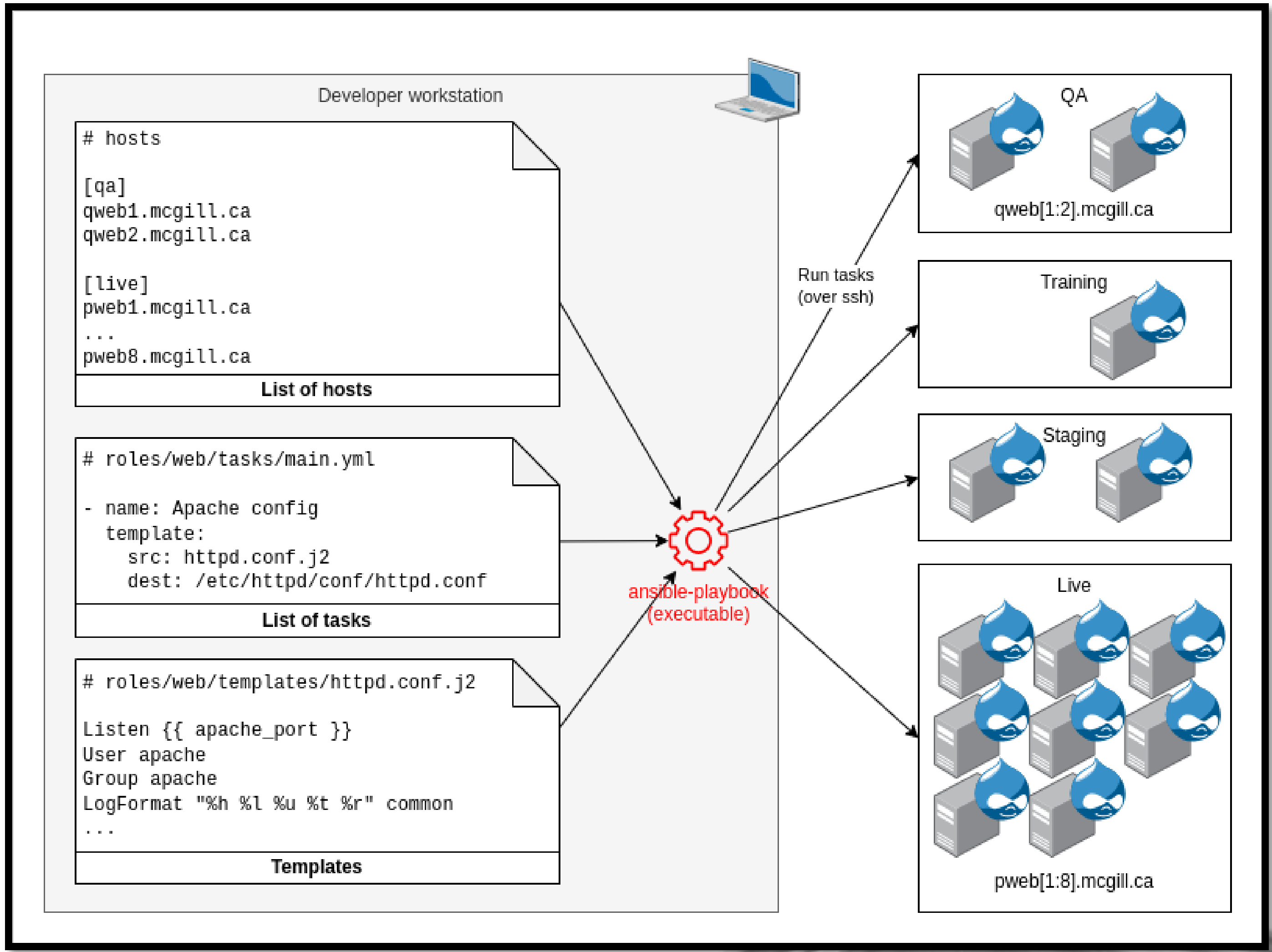
Runs an *Ansible playbook* (list of tasks) on remote hosts via ssh.

Guarantees:

- Consistent state of all environments
- Consistent state of all machines in a given environment

Problems:

- Execution environment specific to each developer
- No visibility on when, why, who updated the servers



```
# hosts

[qa]
qweb1.mcgill.ca
qweb2.mcgill.ca

[live]
pweb1.mcgill.ca
...
pweb8.mcgill.ca
```

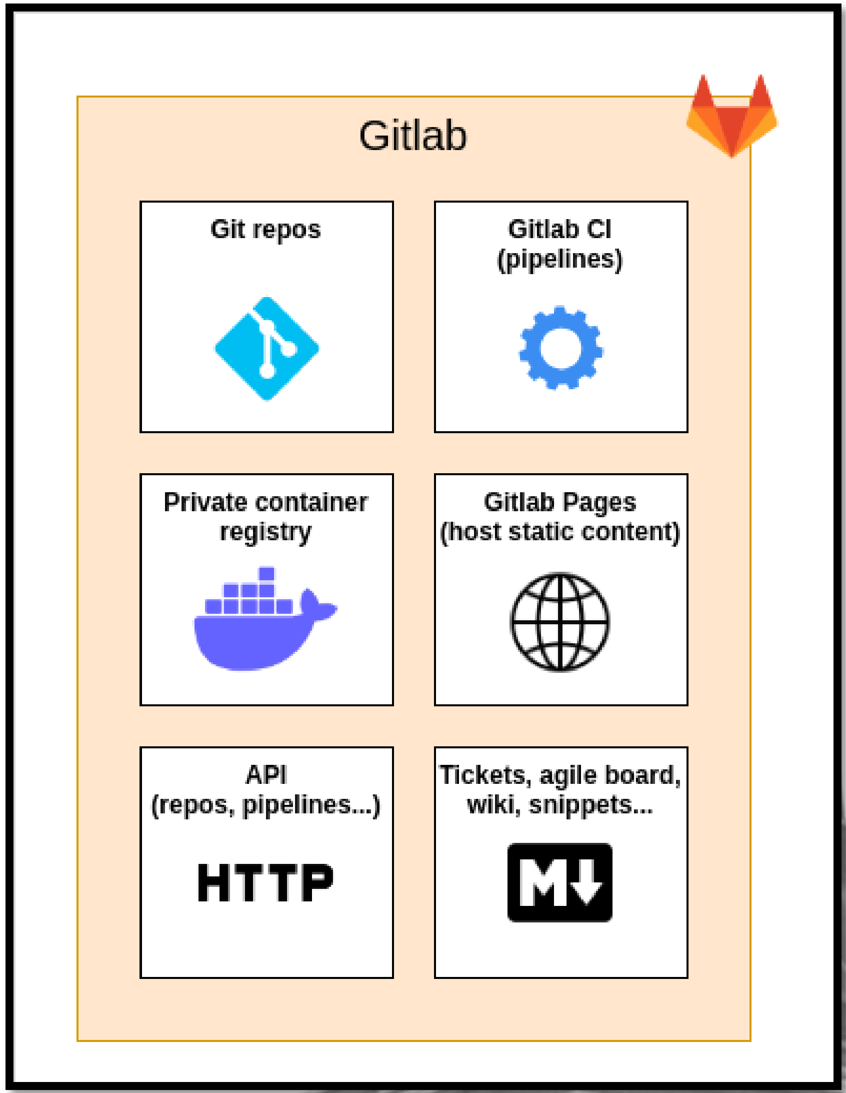
```
# roles/web/tasks/main.yml

- name: Apache config
  template:
    src: httpd.conf.j2
    dest: /etc/httpd/conf/httpd.conf
```

```
# roles/web/templates/httpd.conf.j2

Listen {{ apache_port }}
User apache
Group apache
LogFormat "%h %l %u %t %r" common
...
```

Gitlab

Git repo, but also:

- Continuous Integration (CI) system
- Private container registry
- Hosting solution for static content (Gitlab Pages)
- Comprehensive API for interacting with Git repos, CI pipelines, container registry, etc
- Ticketing system & agile board
- Wiki, code snippets
- Etc.



Gitlab
Git repos
Gitlab CI (pipelines)
Private container registry
Gitlab Pages (host static content)
API (repos, pipelines...)
HTTP
Tickets, agile board, wiki, snippets...

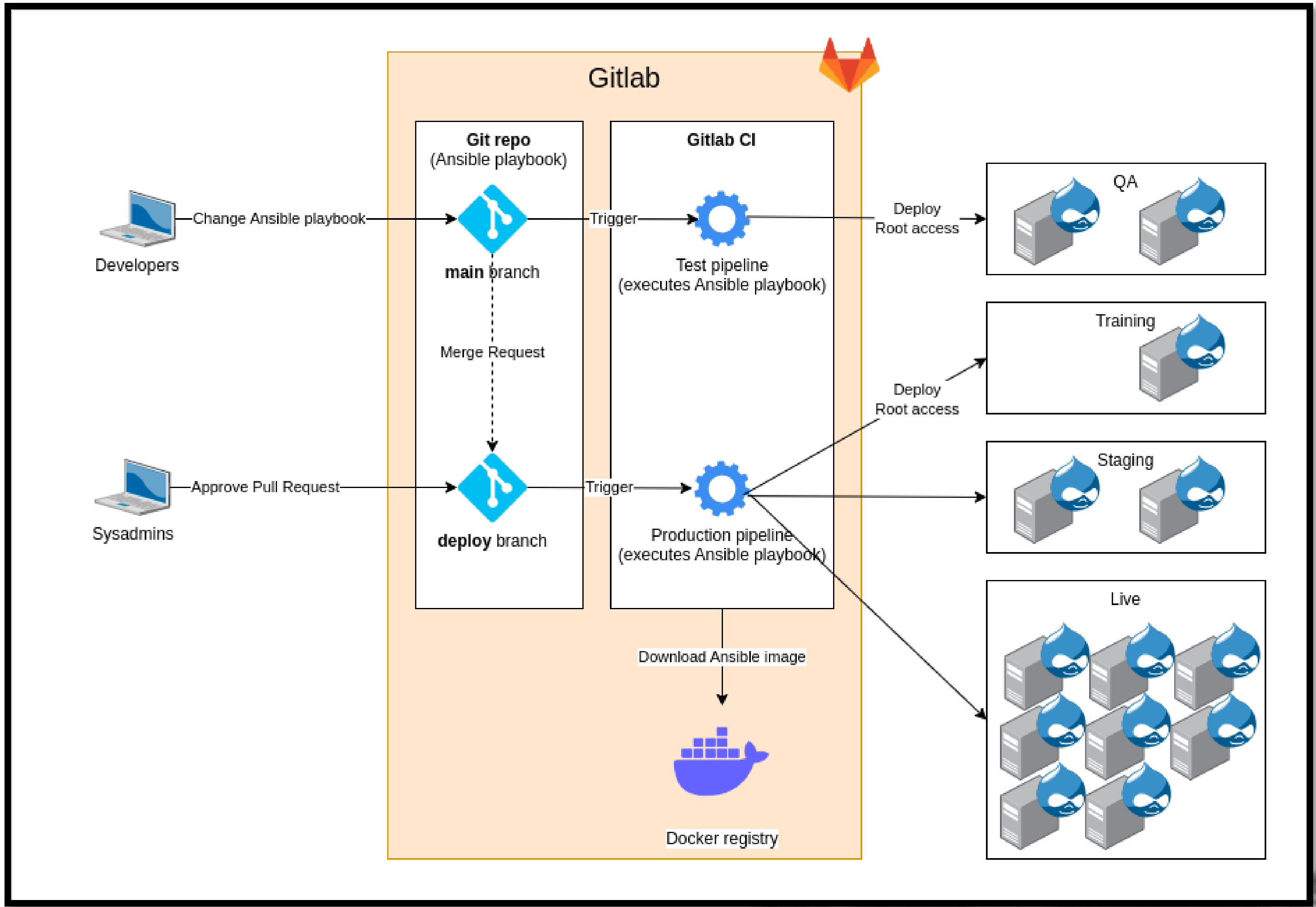
Automated deployment

Deploying the infrastructure is now:

- Automated
- No direct escalation of privileges (root access)
- On environment-specific VMs

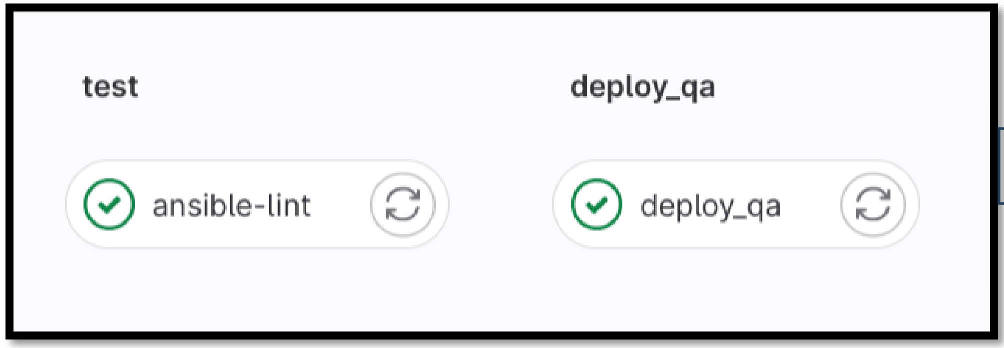
Guarantees:

- Environments are alike
- Servers in a given environment are alike
- Consistent execution environment
- Visibility with other developers and sysadmins
- All changes are peer-reviewed
- All changes are traced

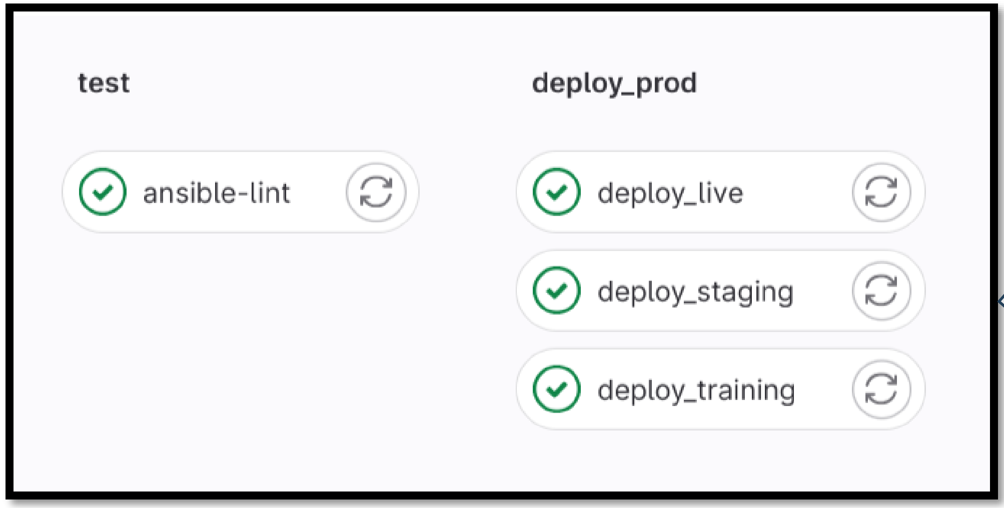


Gitlab CI pipelines (infrastructure)

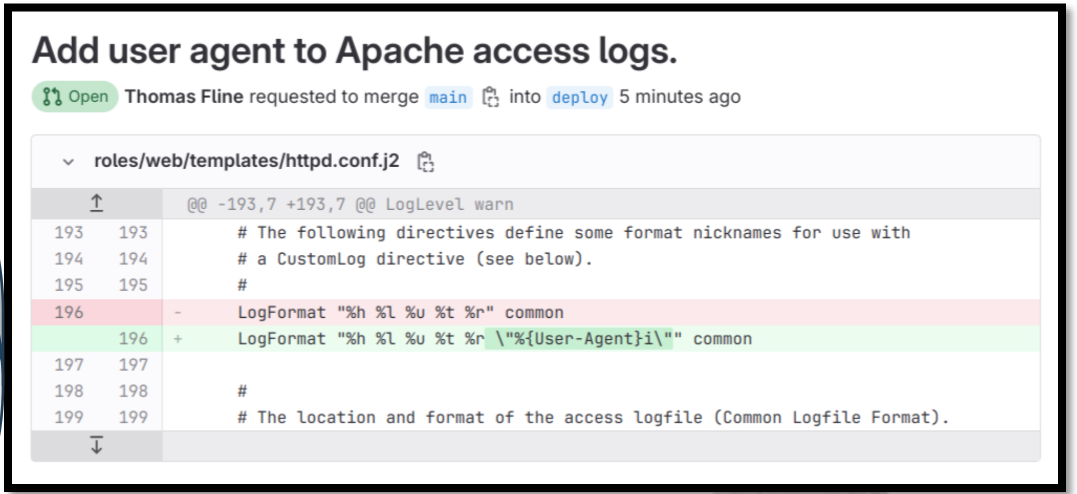
Test pipeline (**main** branch):



Production pipeline (**deploy** branch):



Merge **main** to **deploy**:



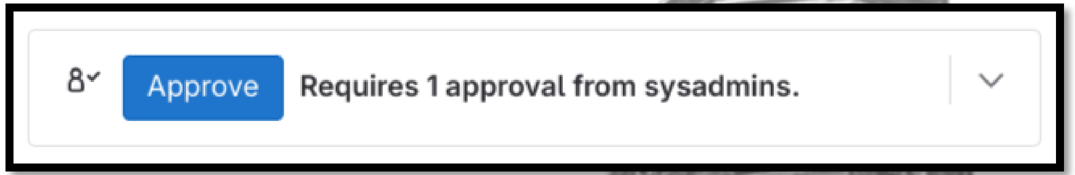
Add user agent to Apache access logs.

Open **Thomas Fline** requested to merge `main` into `deploy` 5 minutes ago

roles/web/templates/httpd.conf.j2

```
@@ -193,7 +193,7 @@ LogLevel warn
     # The following directives define some format nicknames for use with
     # a CustomLog directive (see below).
     #
-    LogFormat "%h %l %u %t %r" common
+    LogFormat "%h %l %u %t %r \"%{User-Agent}i\"" common

     #
     # The location and format of the access logfile (Common Logfile Format).
```

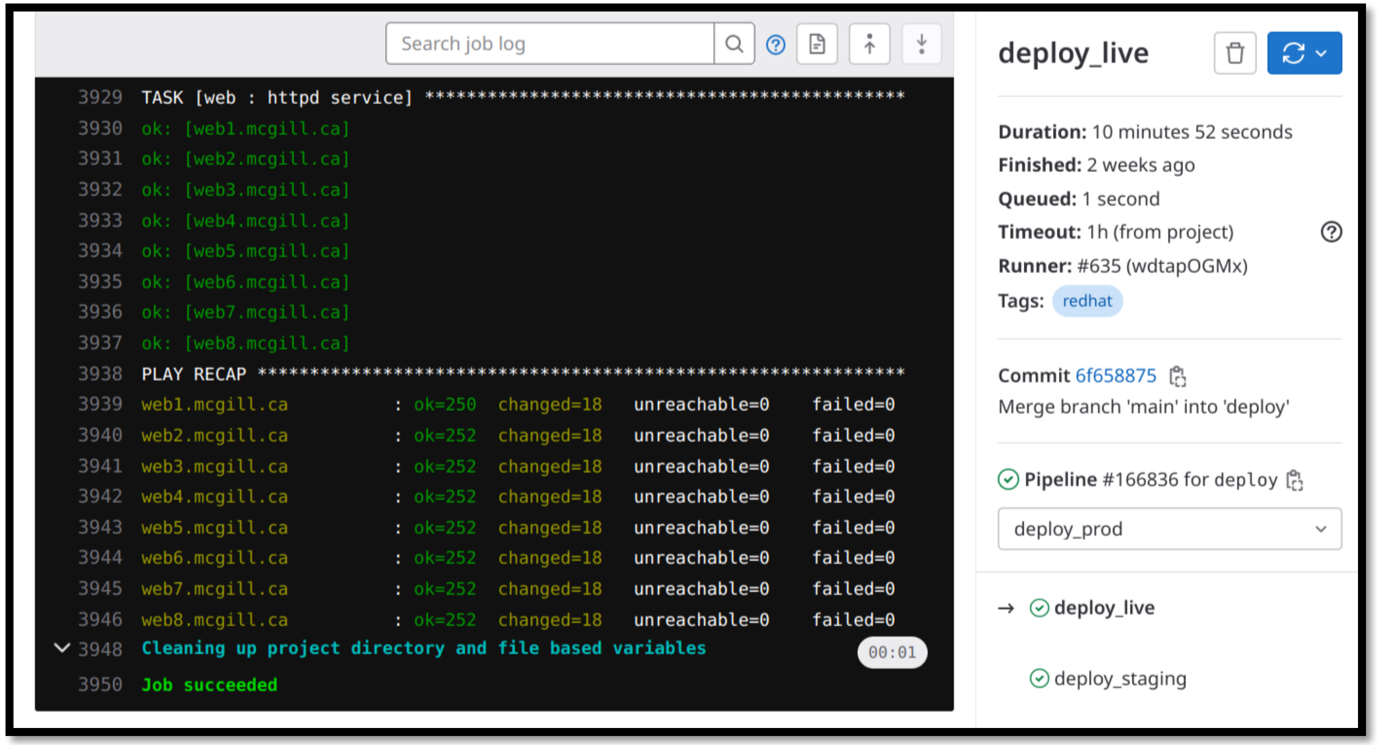
Get approval from sysadmins:



Requires 1 approval from sysadmins.

Gitlab CI jobs

Logs of all changes (output the Ansible executions):



```
TASK [web : httpd service] ************************************************
ok: [web1.mcgill.ca]
ok: [web2.mcgill.ca]
ok: [web3.mcgill.ca]
ok: [web4.mcgill.ca]
ok: [web5.mcgill.ca]
ok: [web6.mcgill.ca]
ok: [web7.mcgill.ca]
ok: [web8.mcgill.ca]
PLAY RECAP ****************************************************************
web1.mcgill.ca             : ok=250  changed=18   unreachable=0    failed=0
web2.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
web3.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
web4.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
web5.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
web6.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
web7.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
web8.mcgill.ca             : ok=252  changed=18   unreachable=0    failed=0
Cleaning up project directory and file based variables                00:01
Job succeeded
```

deploy_live

Duration: 10 minutes 52 seconds
Finished: 2 weeks ago
Queued: 1 second
Timeout: 1h (from project)
Runner: #635 (wdtapOGMx)
Tags: redhat

Commit 6f658875
Merge branch 'main' into 'deploy'

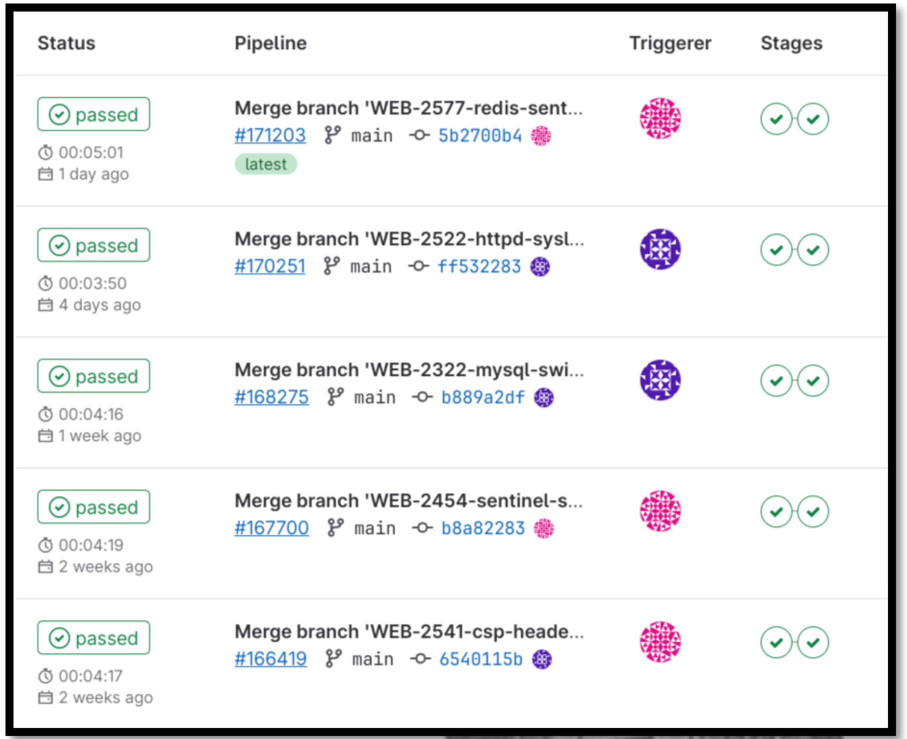
Pipeline #166836 for deploy

deploy_prod

→ **deploy_live**

deploy_staging

History of all changes:



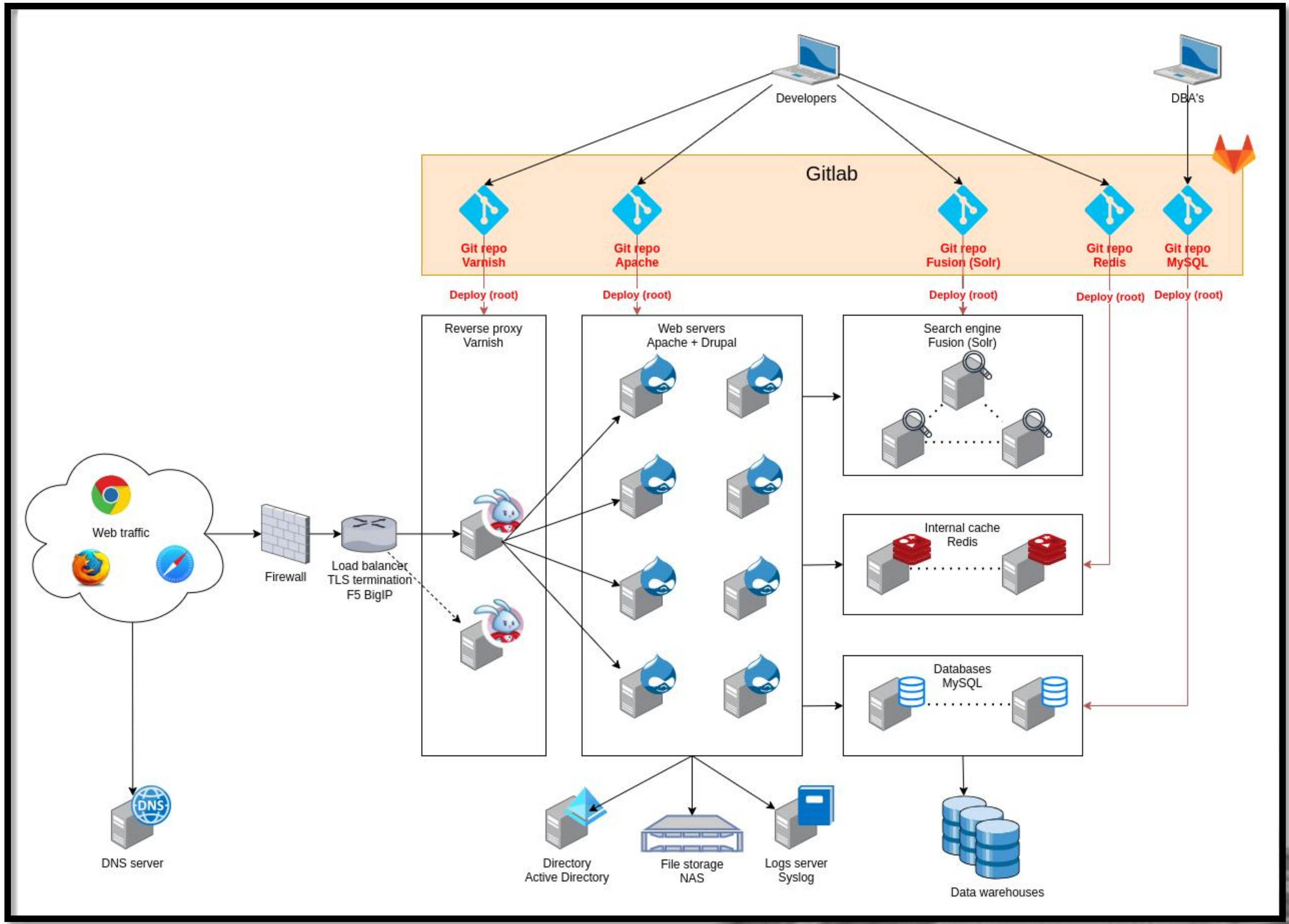
Status	Pipeline	Triggerer	Stages
passed 00:05:01 1 day ago	Merge branch 'WEB-2577-redis-sent... #171203 main 5b2700b4 latest		✓ ✓
passed 00:03:50 4 days ago	Merge branch 'WEB-2522-httpd-sysl... #170251 main ff532283		✓ ✓
passed 00:04:16 1 week ago	Merge branch 'WEB-2322-mysql-swi... #168275 main b889a2df		✓ ✓
passed 00:04:19 2 weeks ago	Merge branch 'WEB-2454-sentinel-s... #167700 main b8a82283		✓ ✓
passed 00:04:17 2 weeks ago	Merge branch 'WEB-2541-csp-heade... #166419 main 6540115b		✓ ✓

Automated deployment everywhere

We now use Ansible to configure all our servers.

Our infrastructure can be discovered just by reading the Ansible playbook ("Infrastructure as Code").

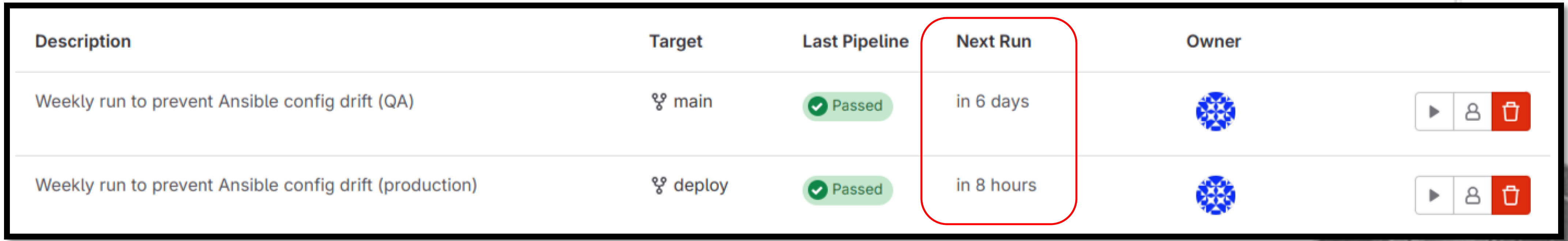
We can discover how servers managed by other teams are configured (e.g. MySQL servers managed by the DBA's).



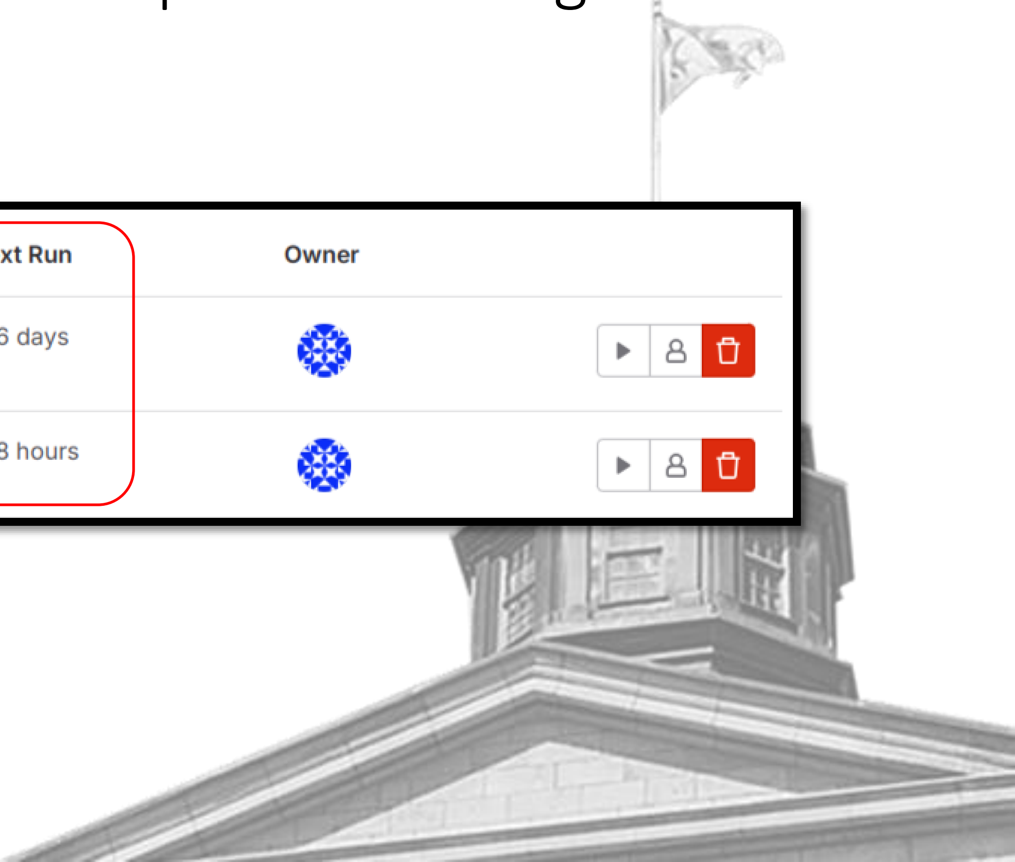
Scheduled pipelines

Automatically run CI pipelines on a weekly basis, on all environments.

Reconciles servers with the Ansible playbook to prevent config drift.



Description	Target	Last Pipeline	Next Run	Owner
Weekly run to prevent Ansible config drift (QA)	main	Passed	in 6 days	
Weekly run to prevent Ansible config drift (production)	deploy	Passed	in 8 hours	



Deployment: application

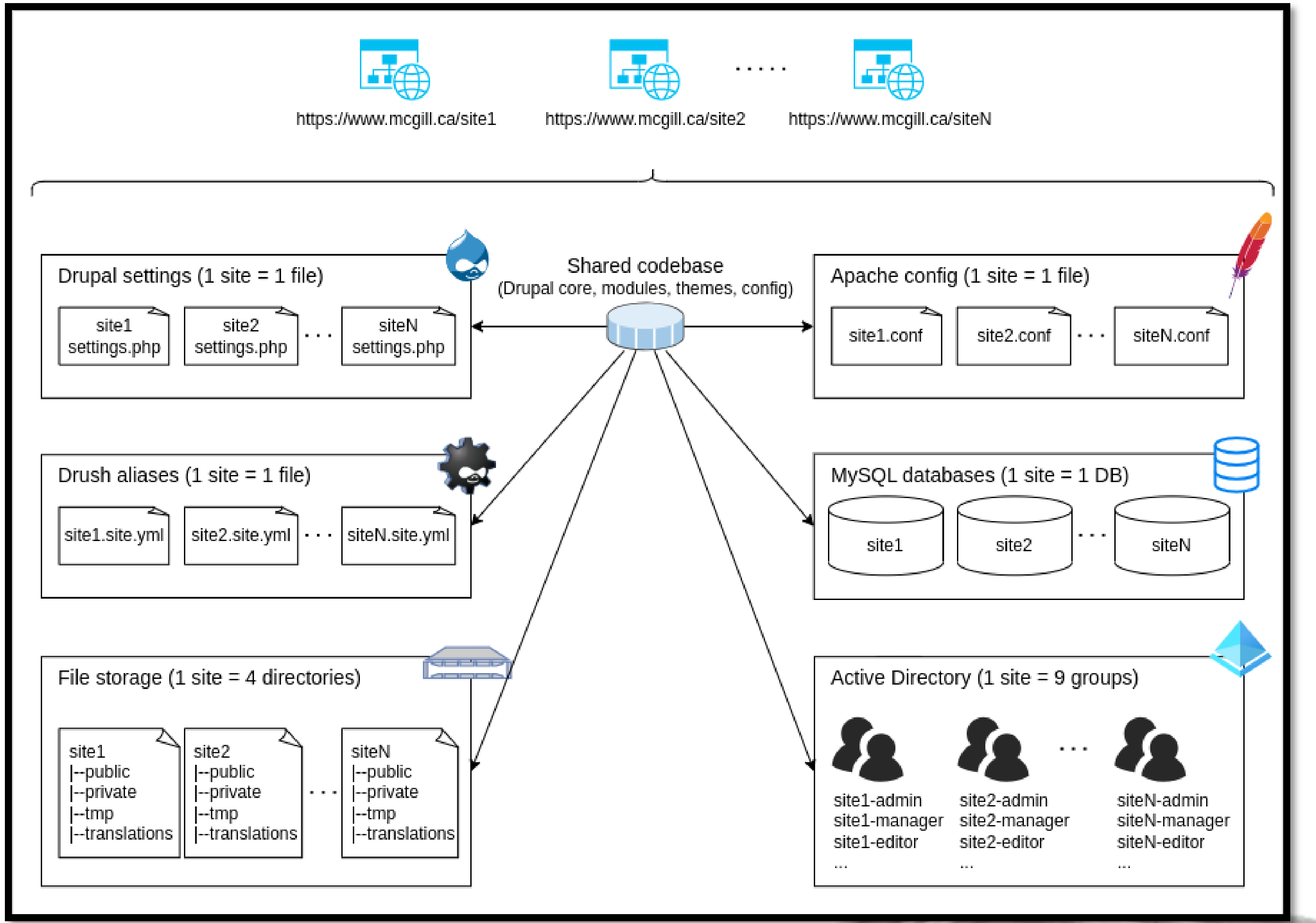
Multisite Drupal installation

1 website =

- 1 Apache config file
- 1 Drupal settings file (settings.php)
- 1 Drush alias file
- 1 MySQL database
- 4 directories (user files)
- 9 Active Directory groups

Codebase shared between all websites:

- Drupal core and its dependencies
- Drupal modules (custom & contrib)
- Drupal themes
- Drupal config



https://www.mcgill.ca/site1
https://www.mcgill.ca/site2
.....
https://www.mcgill.ca/siteN
Drupal settings (1 site = 1 file)
site1 settings.php
site2 settings.php
siteN settings.php
Shared codebase
(Drupal core, modules, themes, config)
Apache config (1 site = 1 file)
site1.conf
site2.conf
siteN.conf
Drush aliases (1 site = 1 file)
site1.site.yml
site2.site.yml
siteN.site.yml
MySQL databases (1 site = 1 DB)
site1
site2
siteN
File storage (1 site = 4 directories)
site1 |--public |--private |--tmp |--translations
site2 |--public |--private |--tmp |--translations
siteN |--public |--private |--tmp |--translations
Active Directory (1 site = 9 groups)
site1-admin site1-manager site1-editor ...
site2-admin site2-manager site2-editor ...
siteN-admin siteN-manager siteN-editor ...

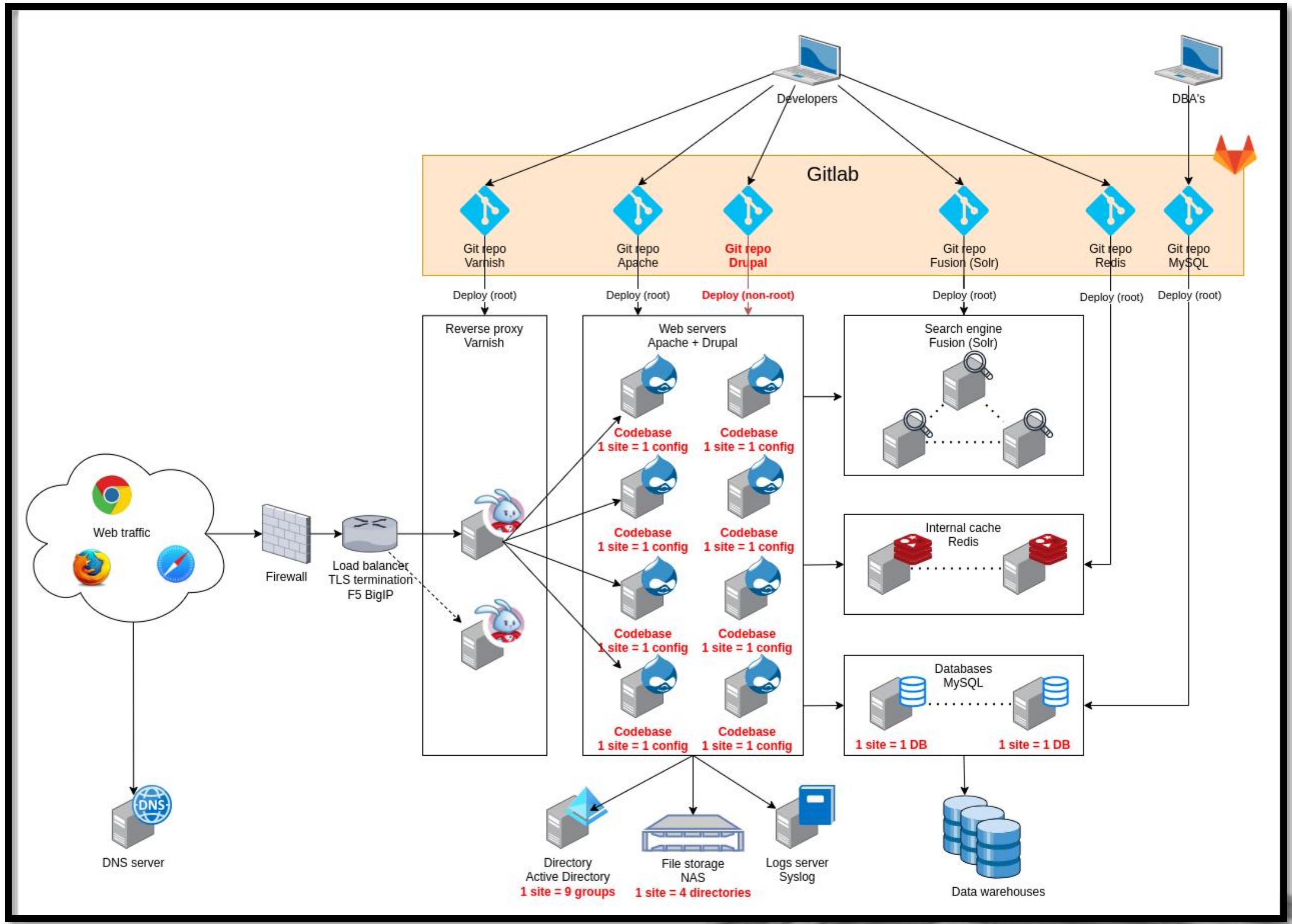
Application deployment

Same principle as for infrastructure.

CI pipeline executes Ansible which deploys the code and updates settings.php, Apache config, etc.

Differences compared to infrastructure playbooks:

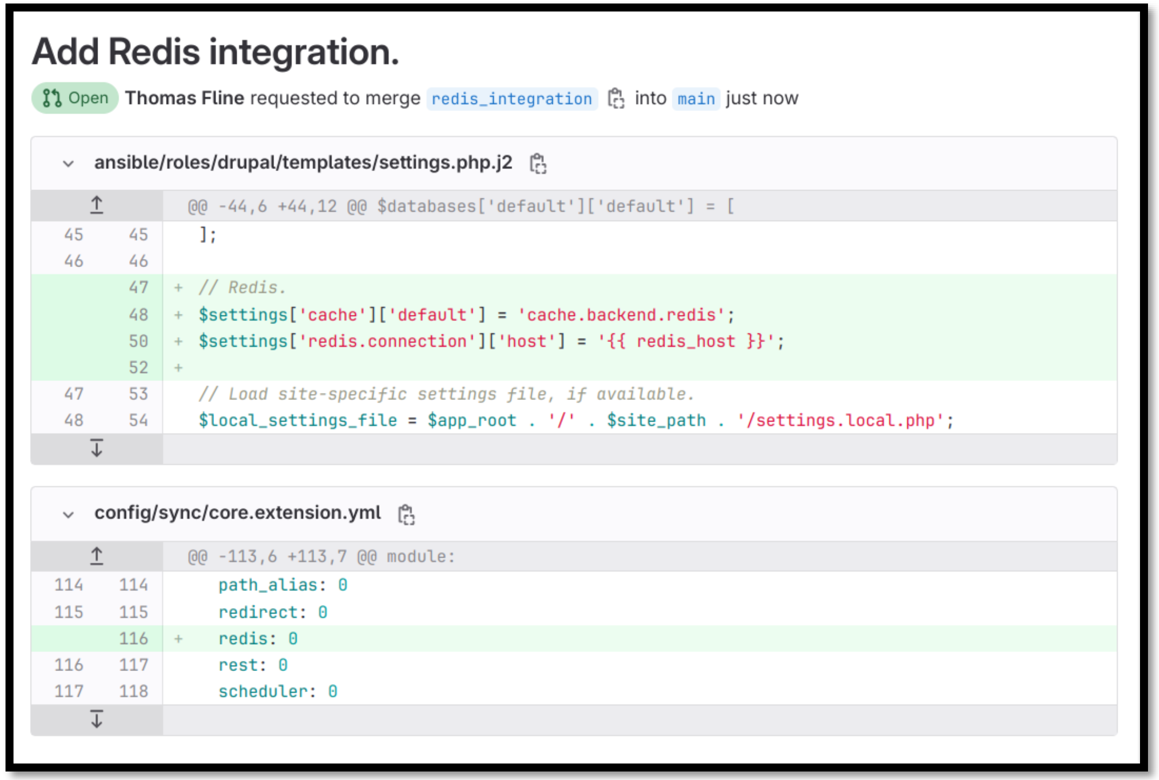
- No privilege escalation
- No need to request approval of Merge Requests from sysadmins
- Playbook stored in same Git repo as our Drupal codebase



Git repo structure

Ansible playbook is part of our Drupal Git repo (outside of the web root), along with modules, themes, config, etc.

Allows to deploy *all* changes in a single Merge Request, for instance when adding a new module and updating settings.php accordingly.



Add Redis integration.

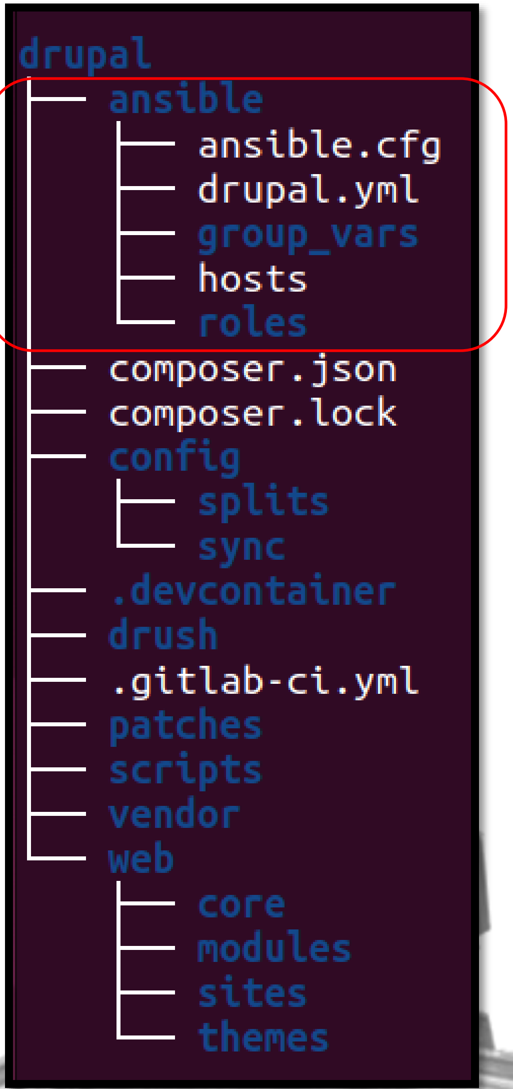
Open Thomas Fline requested to merge `redis_integration` into `main` just now

ansible/roles/drupal/templates/settings.php.j2

```
@@ -44,6 +44,12 @@ $databases['default']['default'] = [
 ];

+// Redis.
+$settings['cache']['default'] = 'cache.backend.redis';
+$settings['redis.connection']['host'] = '{{ redis_host }}';
+
 // Load site-specific settings file, if available.
 $local_settings_file = $app_root . '/' . $site_path . '/settings.local.php';
```

config/sync/core.extension.yml

```
@@ -113,6 +113,7 @@ module:
   path_alias: 0
   redirect: 0
+  redis: 0
   rest: 0
   scheduler: 0
```

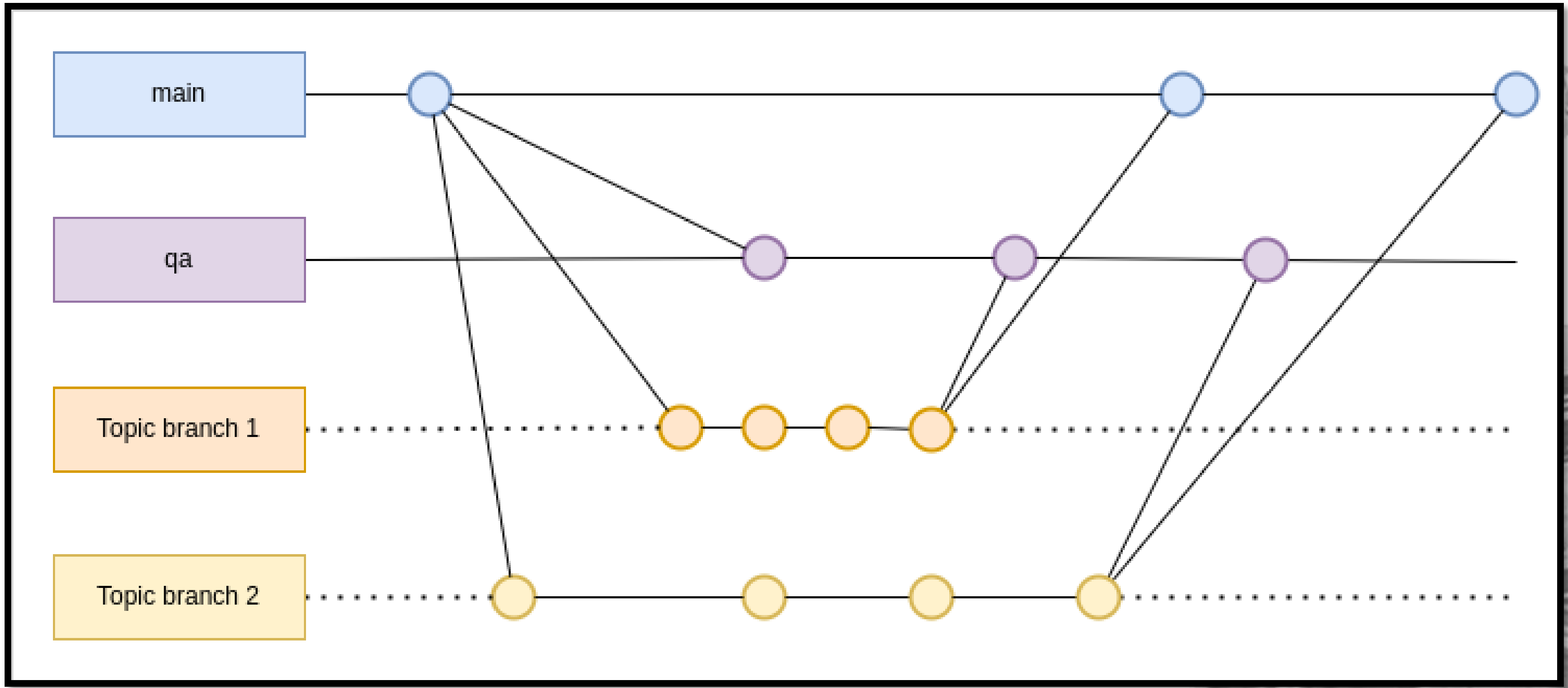


```
drupal
├── ansible
│   ├── ansible.cfg
│   ├── drupal.yml
│   ├── group_vars
│   ├── hosts
│   └── roles
├── composer.json
├── composer.lock
├── config
│   ├── splits
│   └── sync
├── .devcontainer
├── drush
├── .gitlab-ci.yml
├── patches
├── scripts
├── vendor
└── web
    ├── core
    ├── modules
    ├── sites
    └── themes
```

Git branching model

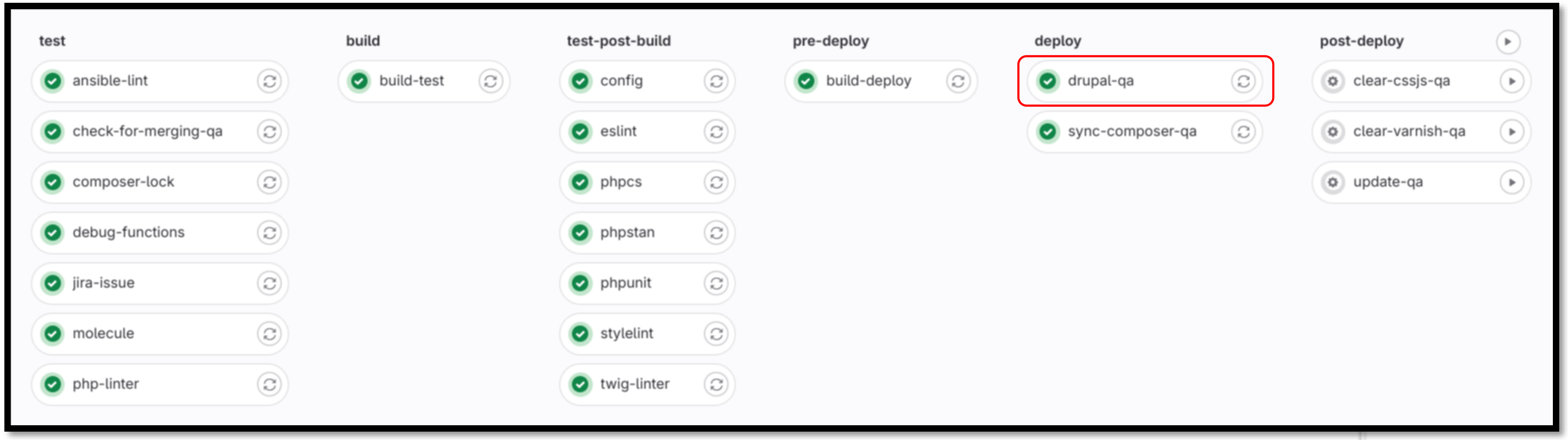
Each topic branch is merged to **qa** and then **main** independently.

Allows to deploy exactly one feature at a time.



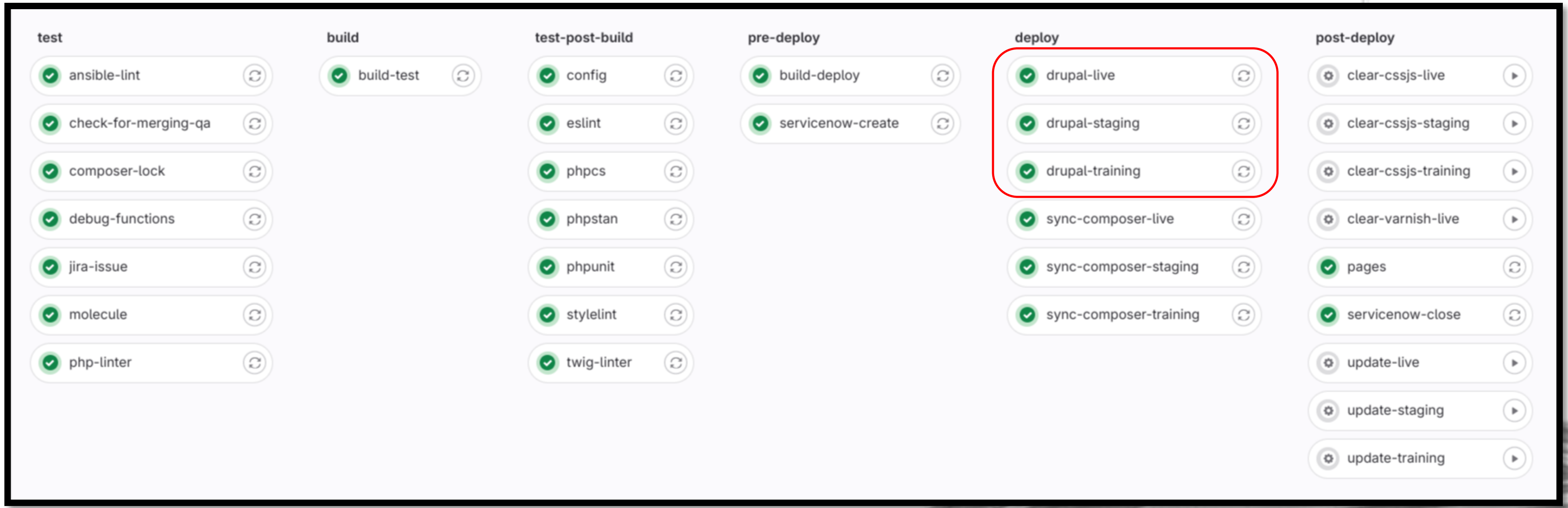
Gitlab CI pipelines (application)

Test pipeline
(**qa** branch)



test	build	test-post-build	pre-deploy	deploy	post-deploy
ansible-lint	build-test	config	build-deploy	drupal-qa	clear-cssjs-qa
check-for-merging-qa		eslint		sync-composer-qa	clear-varnish-qa
composer-lock		phpcs			update-qa
debug-functions		phpstan			
jira-issue		phpunit			
molecule		stylelint			
php-linter		twig-linter			

Production pipeline
(**main** branch)



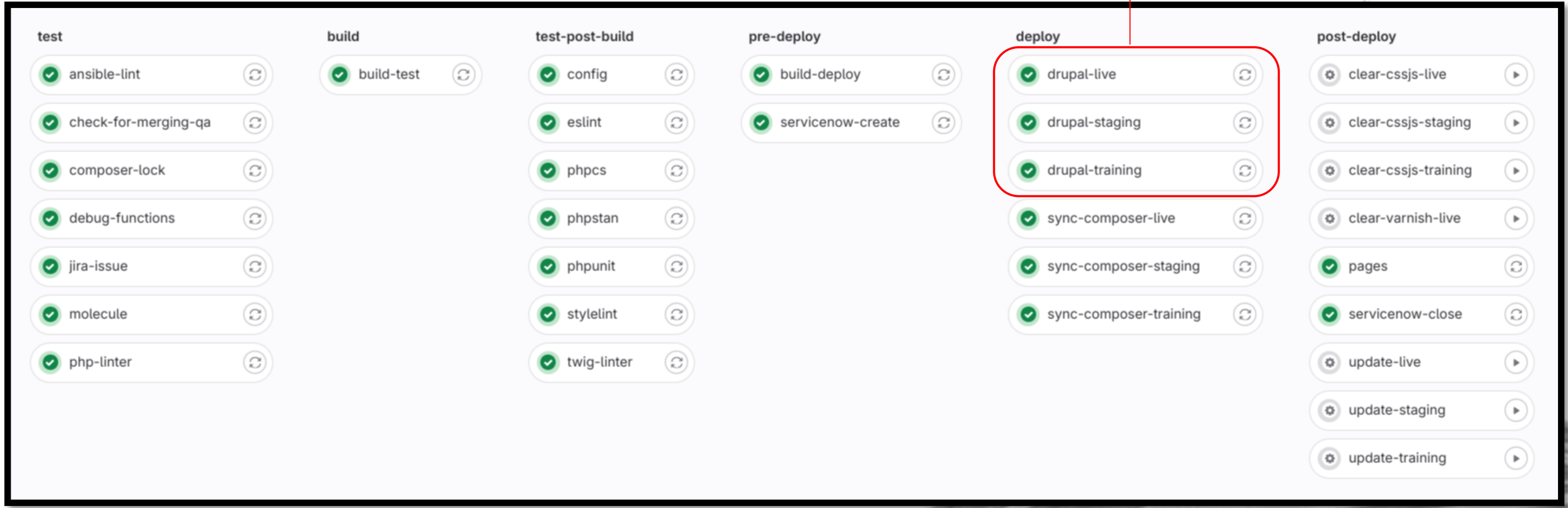
test	build	test-post-build	pre-deploy	deploy	post-deploy
ansible-lint	build-test	config	build-deploy	drupal-live	clear-cssjs-live
check-for-merging-qa		eslint	servicenow-create	drupal-staging	clear-cssjs-staging
composer-lock		phpcs		drupal-training	clear-cssjs-training
debug-functions		phpstan		sync-composer-live	clear-varnish-live
jira-issue		phpunit		sync-composer-staging	pages
molecule		stylelint		sync-composer-training	servicenow-close
php-linter		twig-linter			update-live
					update-staging
					update-training

Gitlab CI job: deployment

Run Ansible tasks on prod environments (deploys updates to Drupal codebase, settings.php files, etc).

```yaml
# .gitlab-ci.yml
drupal-live:
  image: registry.mcgill.ca/cicd/ansible:2.14
  script: ansible-playbook drupal.yml --limit live
```

Production pipeline (**main** branch)



test	build	test-post-build	pre-deploy	deploy	post-deploy
ansible-lint	build-test	config	build-deploy	drupal-live	clear-cssjs-live
check-for-merging-qa		eslint	servicenow-create	drupal-staging	clear-cssjs-staging
composer-lock		phpcs		drupal-training	clear-cssjs-training
debug-functions		phpstan		sync-composer-live	clear-varnish-live
jira-issue		phpunit		sync-composer-staging	pages
molecule		stylelint		sync-composer-training	servicenow-close
php-linter		twig-linter			update-live
					update-staging
					update-training

Gitlab CI job: post-deployment

Update all sites

```
# .gitlab-ci.yml
update-live:
  image: registry.mcgill.ca/cicd/ansible:2.14
  script: ansible live[0] -m command --args \
  "{{ drupal_root }}/scripts/drush.php deploy"
```

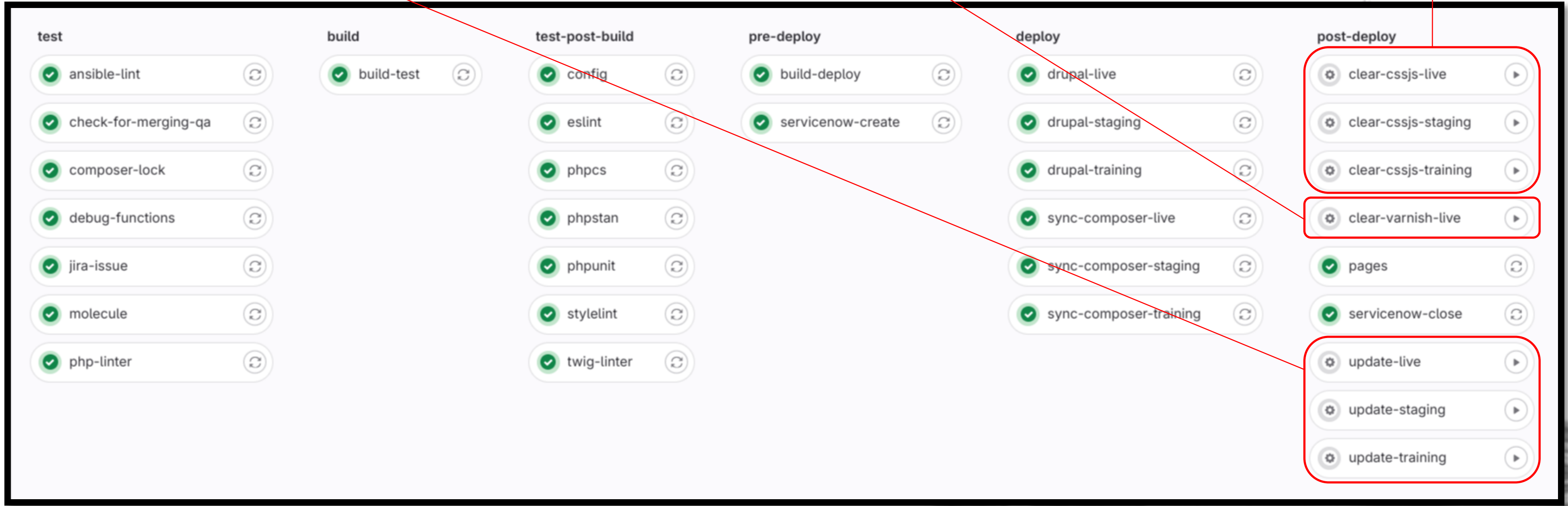
Clear Varnish on all sites

```
# .gitlab-ci.yml
clear-varnish-live:
  image: registry.mcgill.ca/cicd/ansible:2.14
  script: ansible live[0] -m command --args \
  "{{ drupal_root }}/scripts/drush.php cache-rebuild-external"
```

Clear CSS & JS caches on all sites

```
# .gitlab-ci.yml
clear-cssjs-live:
  image: registry.mcgill.ca/cicd/ansible:2.14
  script: ansible live[0] -m command --args \
  "{{ drupal_root }}/scripts/drush.php cc css-js"
```

Production pipeline (**main** branch)



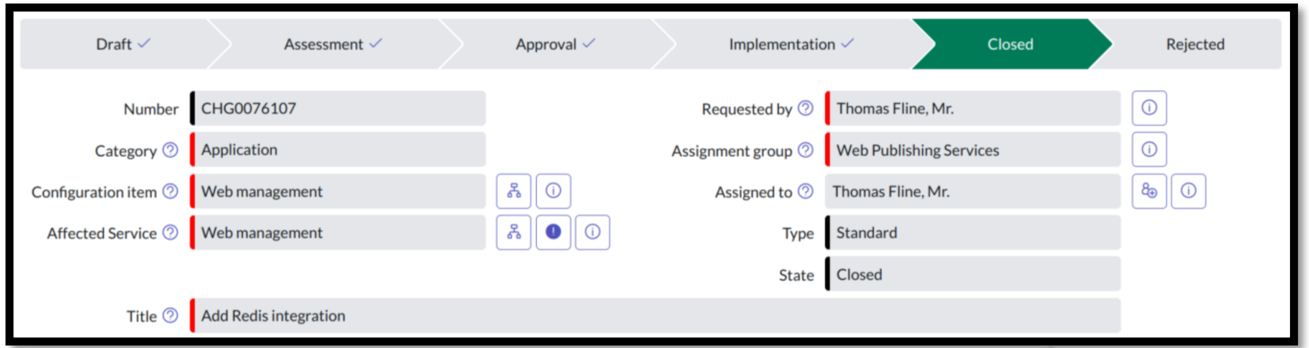
test	build	test-post-build	pre-deploy	deploy	post-deploy
✓ ansible-lint	✓ build-test	✓ config	✓ build-deploy	✓ drupal-live	clear-cssjs-live
✓ check-for-merging-qa		✓ eslint	✓ servicenow-create	✓ drupal-staging	clear-cssjs-staging
✓ composer-lock		✓ phpcs		✓ drupal-training	clear-cssjs-training
✓ debug-functions		✓ phpstan		✓ sync-composer-live	clear-varnish-live
✓ jira-issue		✓ phpunit		✓ sync-composer-staging	✓ pages
✓ molecule		✓ stylelint		✓ sync-composer-training	✓ servicenow-close
✓ php-linter		✓ twig-linter			update-live
					update-staging
					update-training

Change Management

All changes to production must be disclosed in a central registry (ServiceNow), common to all McGill IT.

We use Gitlab pipelines to create and close Change Requests using the ServiceNow API, and post messages in chatroom.

Reduces bureaucracy.



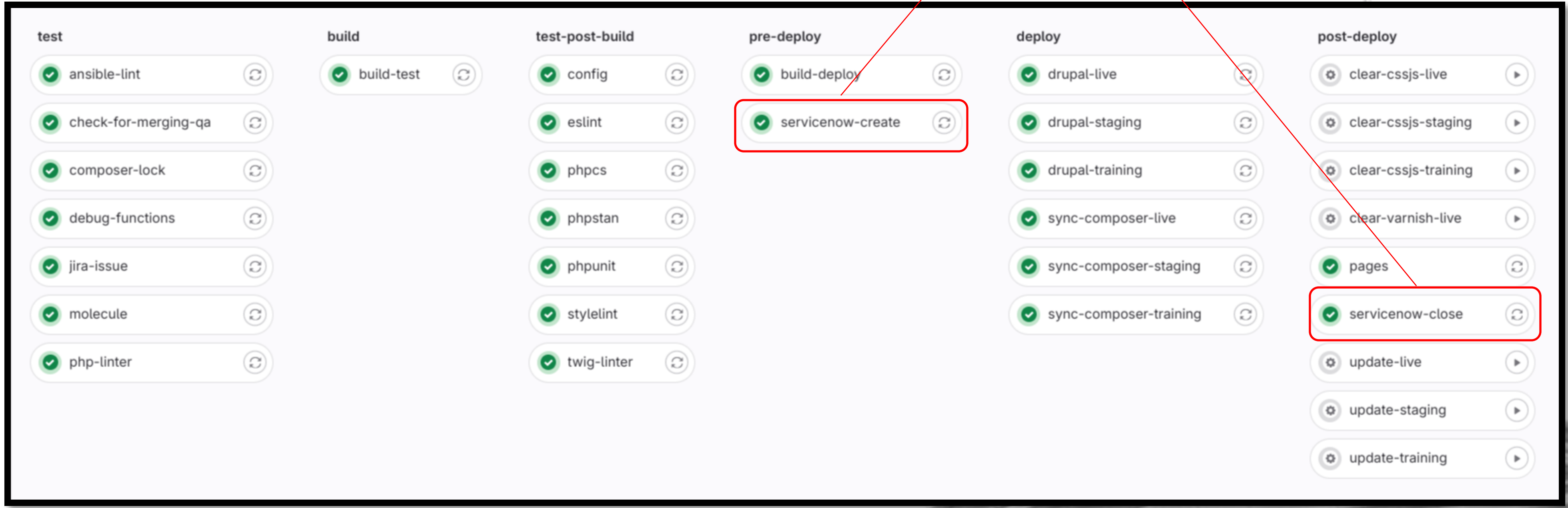
Draft ✓ | Assessment ✓ | Approval ✓ | Implementation ✓ | Closed | Rejected

Number	CHG0076107	Requested by	Thomas Fline, Mr.
Category	Application	Assignment group	Web Publishing Services
Configuration item	Web management	Assigned to	Thomas Fline, Mr.
Affected Service	Web management	Type	Standard
		State	Closed
Title	Add Redis integration		

Create Change Request

Close Change Request

Production pipeline (main branch)



test	build	test-post-build	pre-deploy	deploy	post-deploy
ansible-lint	build-test	config	build-deploy	drupal-live	clear-cssjs-live
check-for-merging-qa		eslint	servicenow-create	drupal-staging	clear-cssjs-staging
composer-lock		phpcs		drupal-training	clear-cssjs-training
debug-functions		phpstan		sync-composer-live	clear-varnish-live
jira-issue		phpunit		sync-composer-staging	pages
molecule		stylelint		sync-composer-training	servicenow-close
php-linter		twig-linter			update-live
					update-staging
					update-training

Ansible & Drupal updates

Renovate bot.

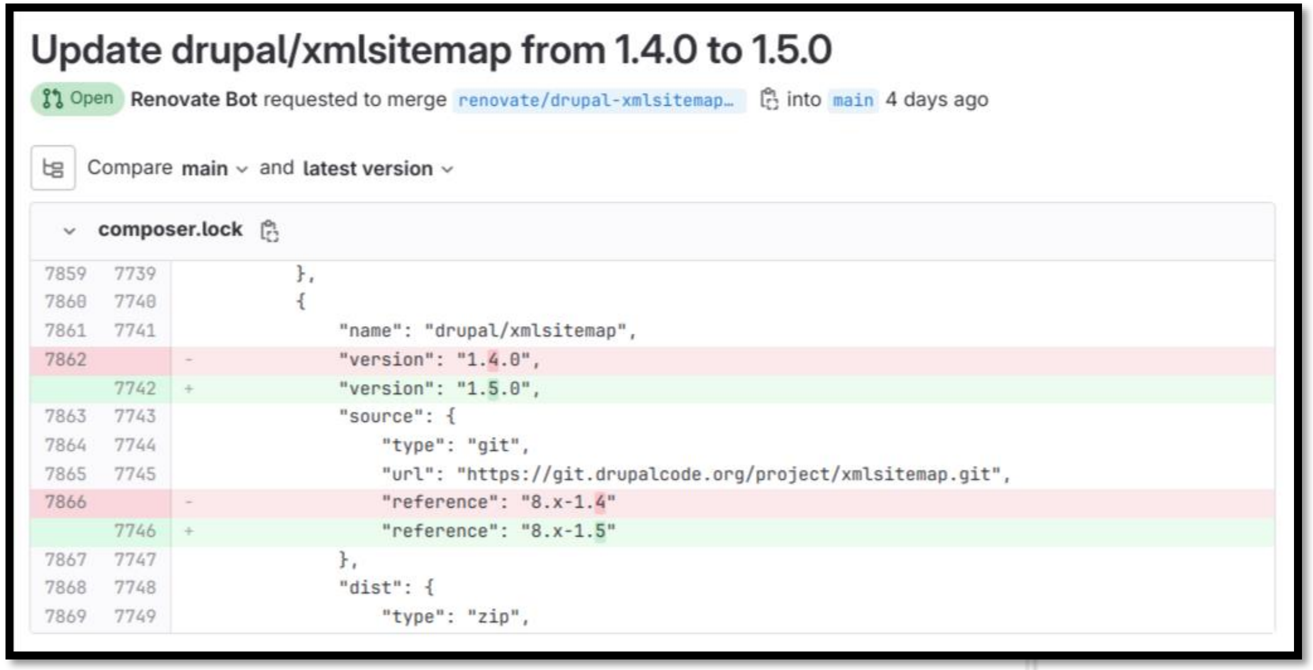
Automatically creates a Merge Request when a dependency has a new version available.

Supports:

- Docker images (e.g. Ansible)
- Composer packages (e.g. Drupal modules)
- NodeJS packages
- Etc.

Highly configurable.

https://github.com/renovatebot/renovate



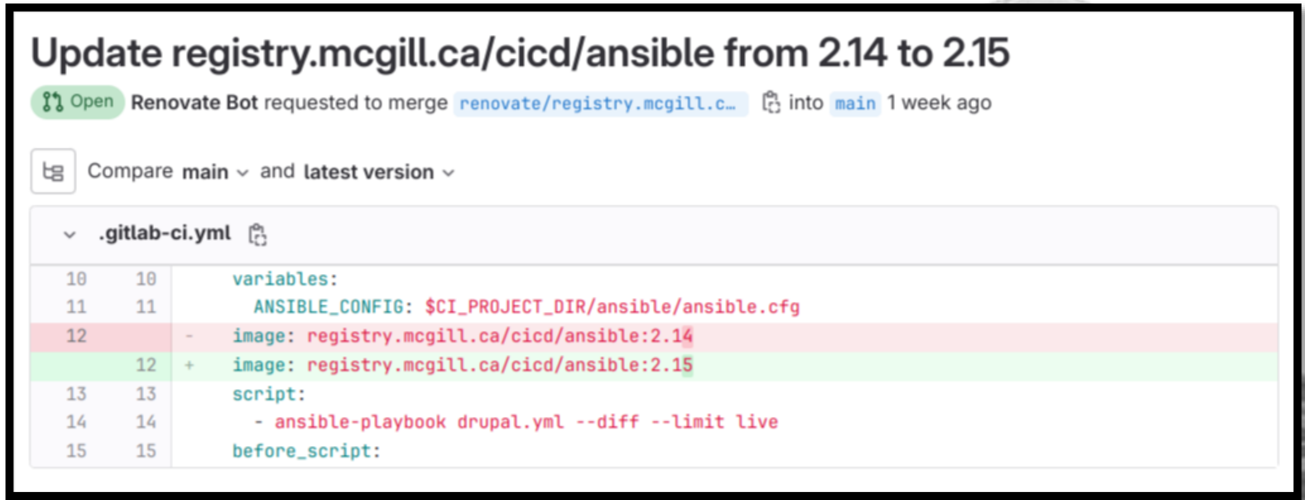
Update drupal/xmlsitemap from 1.4.0 to 1.5.0

Open Renovate Bot requested to merge renovate/drupal-xmlsitemap… into main 4 days ago

Compare main and latest version

composer.lock

```
        },
        {
            "name": "drupal/xmlsitemap",
-           "version": "1.4.0",
+           "version": "1.5.0",
            "source": {
                "type": "git",
                "url": "https://git.drupalcode.org/project/xmlsitemap.git",
-               "reference": "8.x-1.4"
+               "reference": "8.x-1.5"
            },
            "dist": {
                "type": "zip",
```



Update registry.mcgill.ca/cicd/ansible from 2.14 to 2.15

Open Renovate Bot requested to merge renovate/registry.mcgill.c… into main 1 week ago

Compare main and latest version

.gitlab-ci.yml

```
  variables:
    ANSIBLE_CONFIG: $CI_PROJECT_DIR/ansible/ansible.cfg
- image: registry.mcgill.ca/cicd/ansible:2.14
+ image: registry.mcgill.ca/cicd/ansible:2.15
  script:
    - ansible-playbook drupal.yml --diff --limit live
  before_script:
```

Site Management

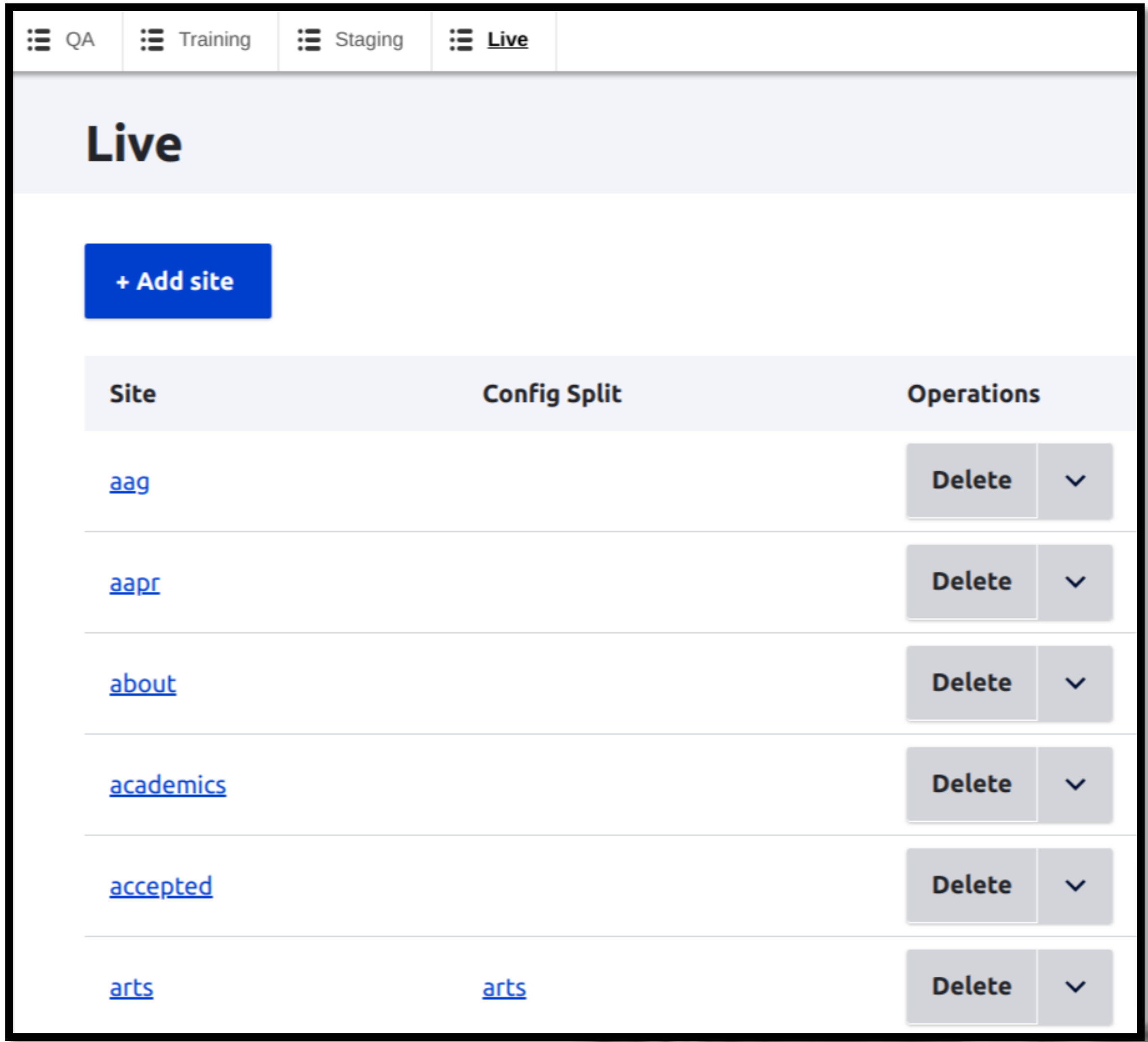
Self-service tool for managing websites

Internal tool for site operations:

- Create website
- Delete website
- Edit website (e.g. change its config split)
- Rename website
- Copy website across environments

Each site may be assigned a *config split* if it requires extra functionality (e.g. a specific module or user role).

https://www.drupal.org/project/config_split



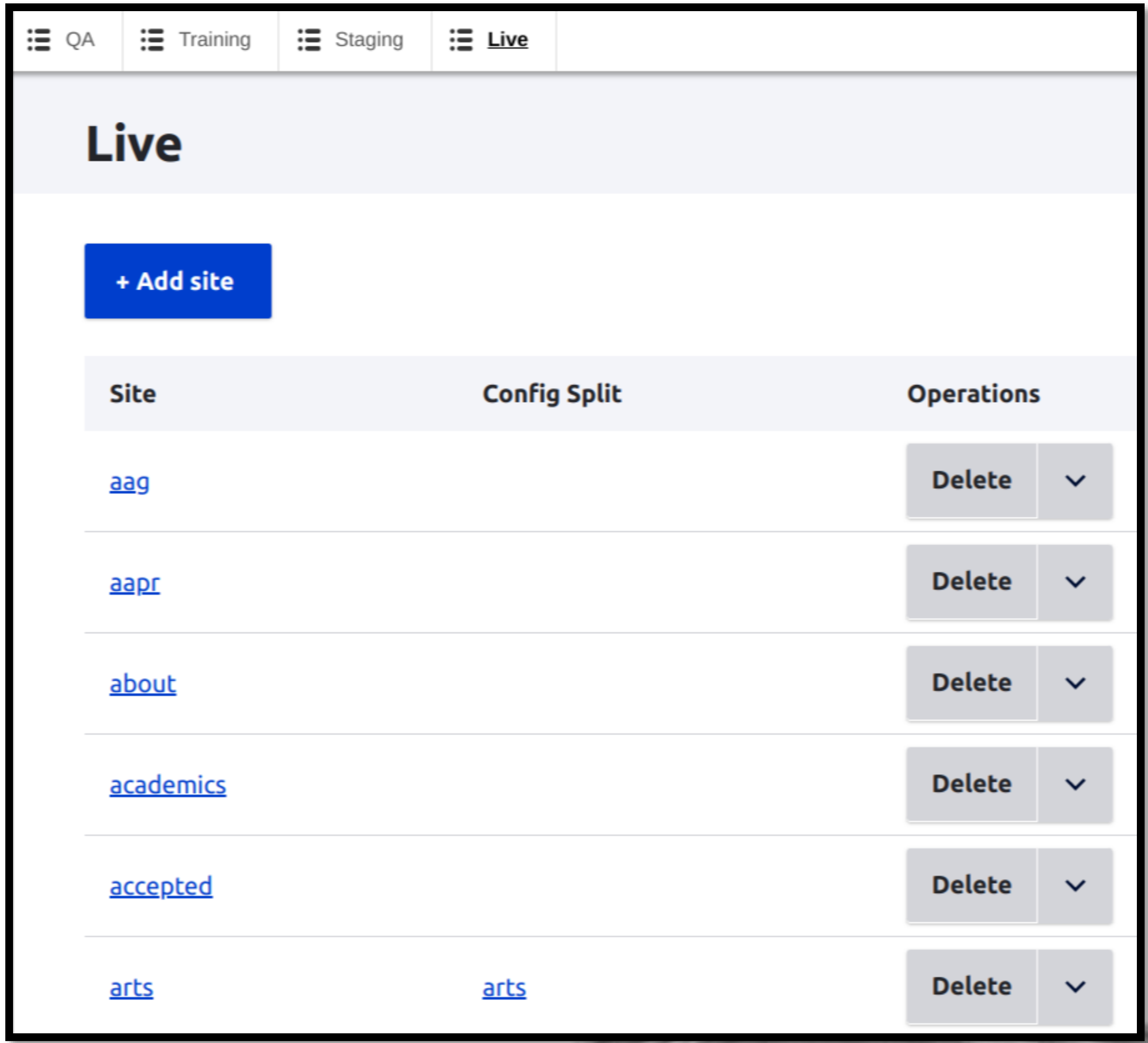
Site	Config Split	Operations
aag		Delete
aapr		Delete
about		Delete
academics		Delete
accepted		Delete
arts	arts	Delete

Self-service tool for managing websites

Sites are listed in an Ansible variable (authoritative source of truth) stored in a YAML file as part of our Git repo.

Internal tool maintains this variable (i.e. controls updates to the YAML file) and triggers CI pipelines via the Gitlab API.

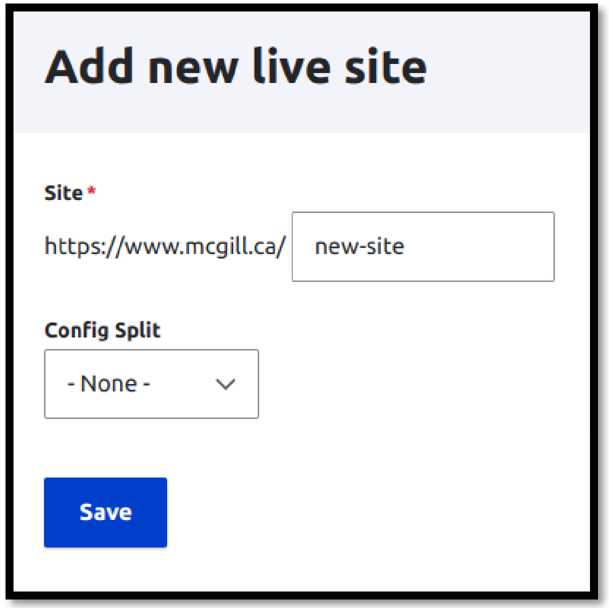
```
# ansible/group_vars/live/sites
drupal_sites:
  aag: { }
  aapr: { }
  about: { }
  academics: { }
  accepted: { }
  arts:
    config_split: arts
  asap: { }
  ...
```



Live

Site	Config Split	Operations
aag		Delete
aapr		Delete
about		Delete
academics		Delete
accepted		Delete
arts	arts	Delete

Site creation



Add new live site

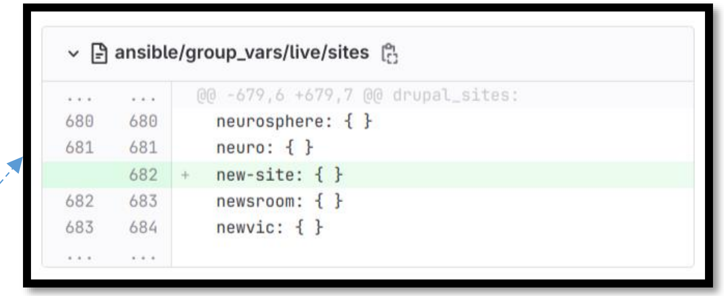
Site *

https://www.mcgill.ca/ new-site

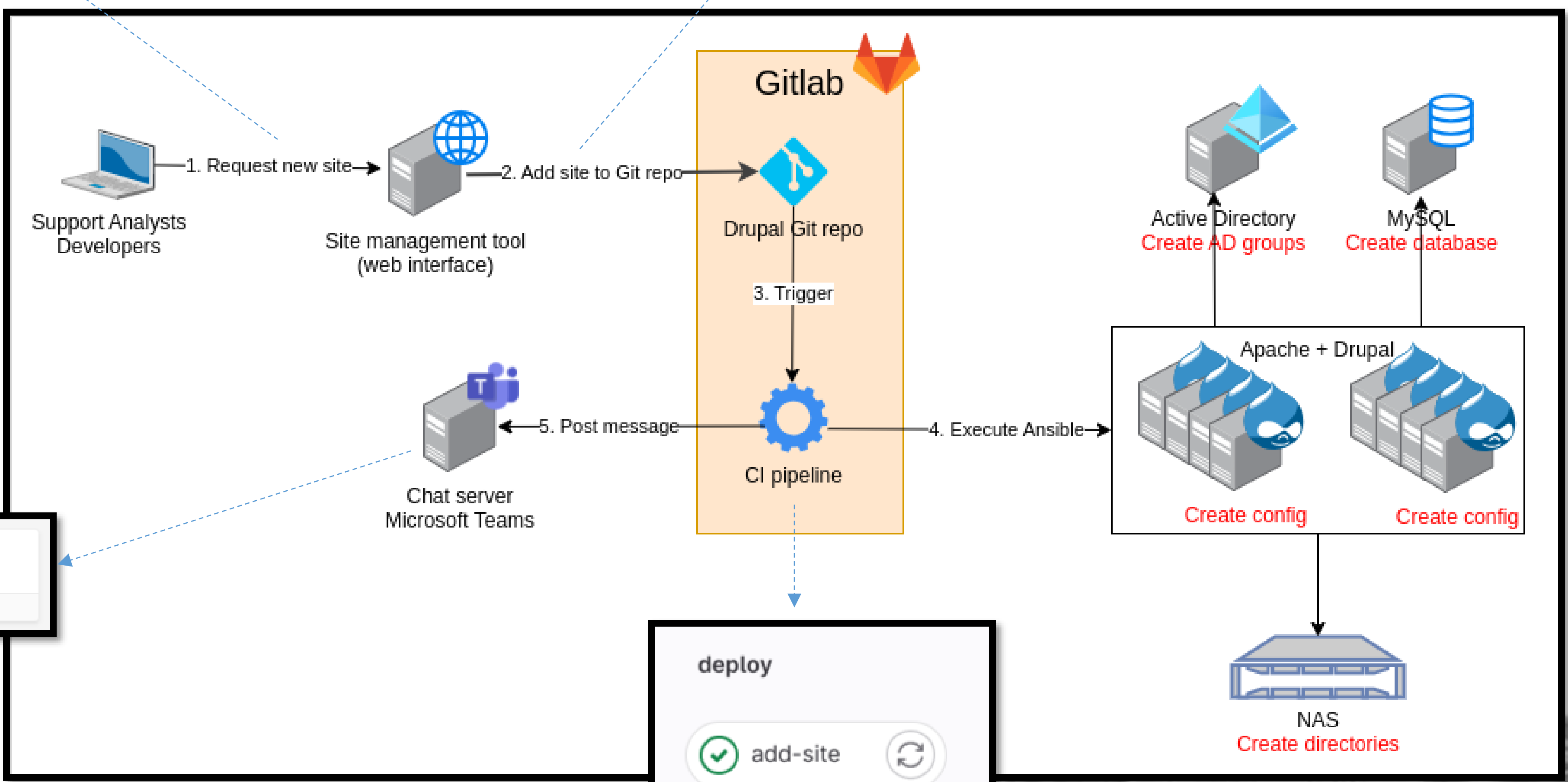
Config Split

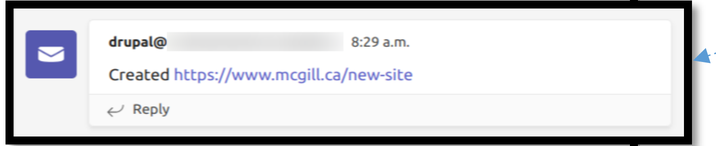
- None -

Save



```
ansible/group_vars/live/sites

...   ...     @@ -679,6 +679,7 @@ drupal_sites:
680   680       neurosphere: { }
681   681       neuro: { }
      682   +   new-site: { }
682   683       newsroom: { }
683   684       newvic: { }
...   ...
```

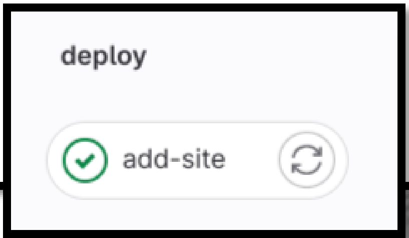




drupal@[redacted] 8:29 a.m.

Created https://www.mcgill.ca/new-site

Reply



deploy

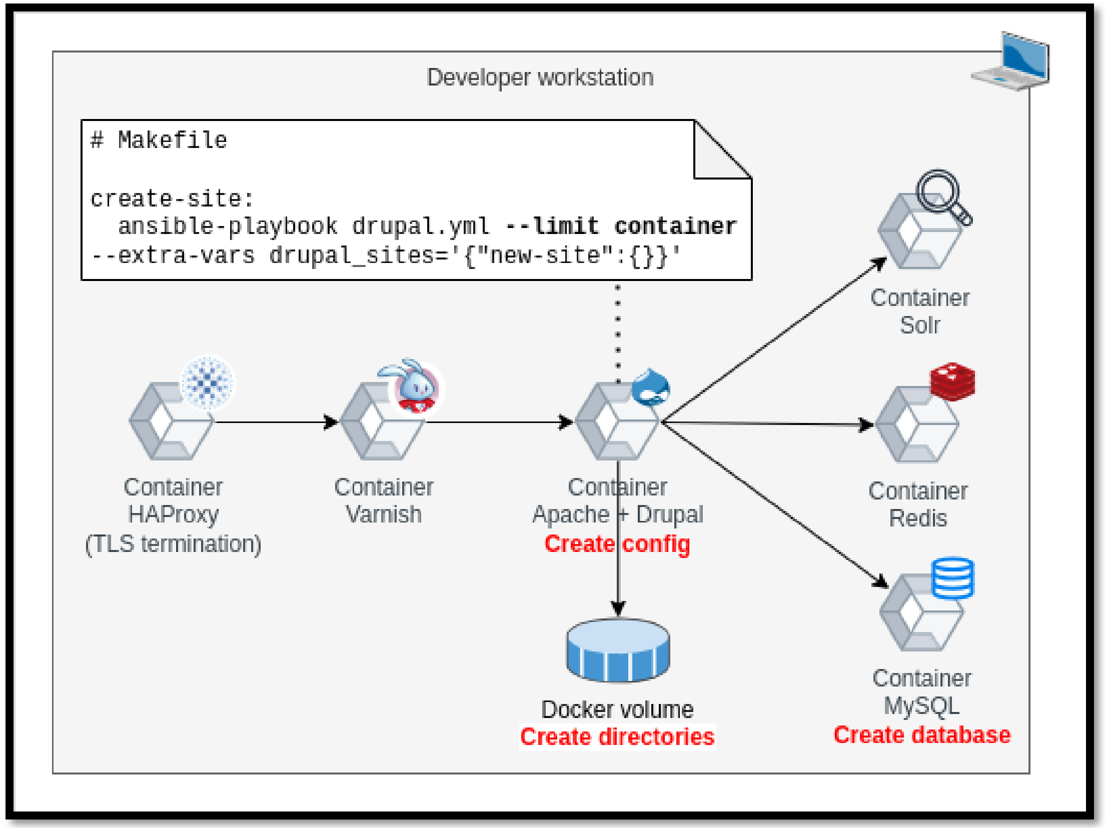
✓ add-site

Development

Ansible in a container

Development environment based on Docker-Compose that mimics production.

Ansible tasks are re-used: websites are created locally the same way they are created on production.



```
# Makefile

create-site:
  ansible-playbook drupal.yml --limit container
--extra-vars drupal_sites='{"new-site":{}}'
```

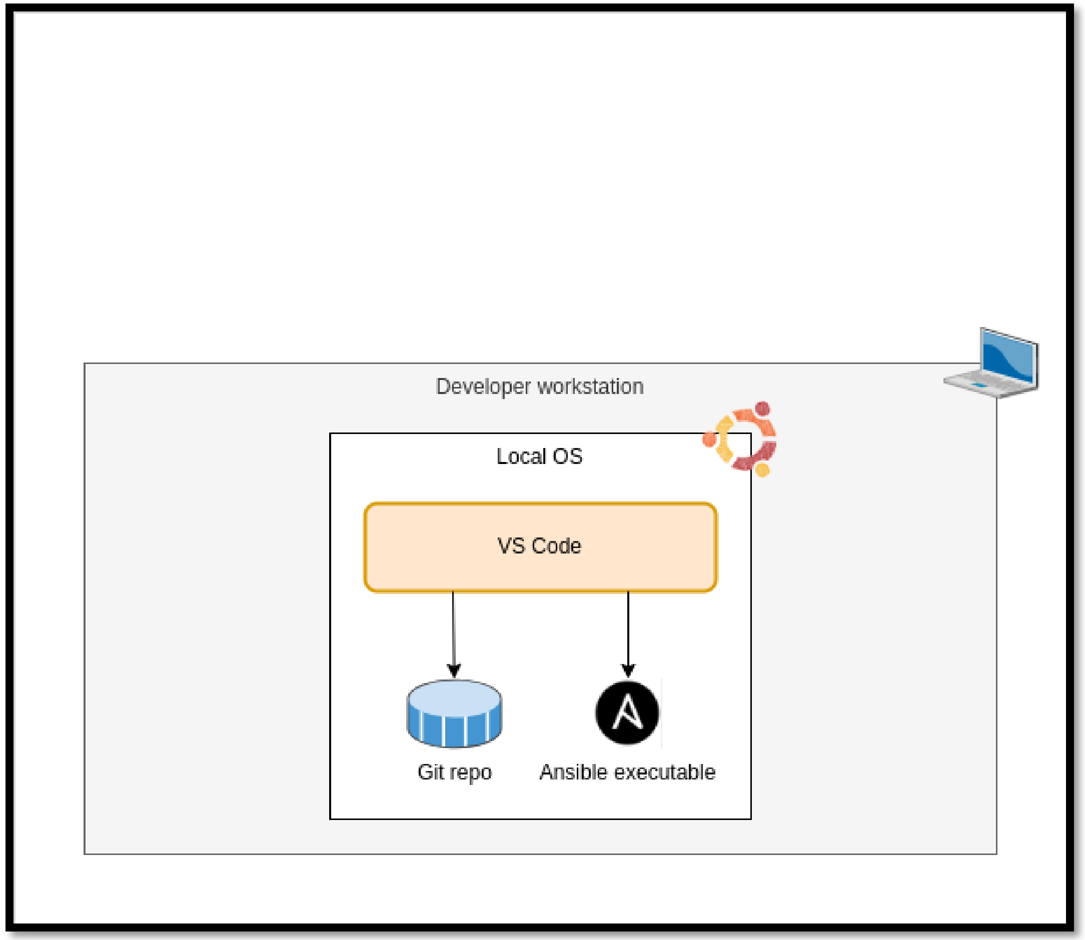
```
# ansible/hosts

[container]
localhost ansible_connection="local"

[qa]
qweb1.mcgill.ca
qweb2.mcgill.ca
...
```

Ansible runs:

- With a local connection (instead of SSH)
- On localhost (i.e. the container) instead of remote servers

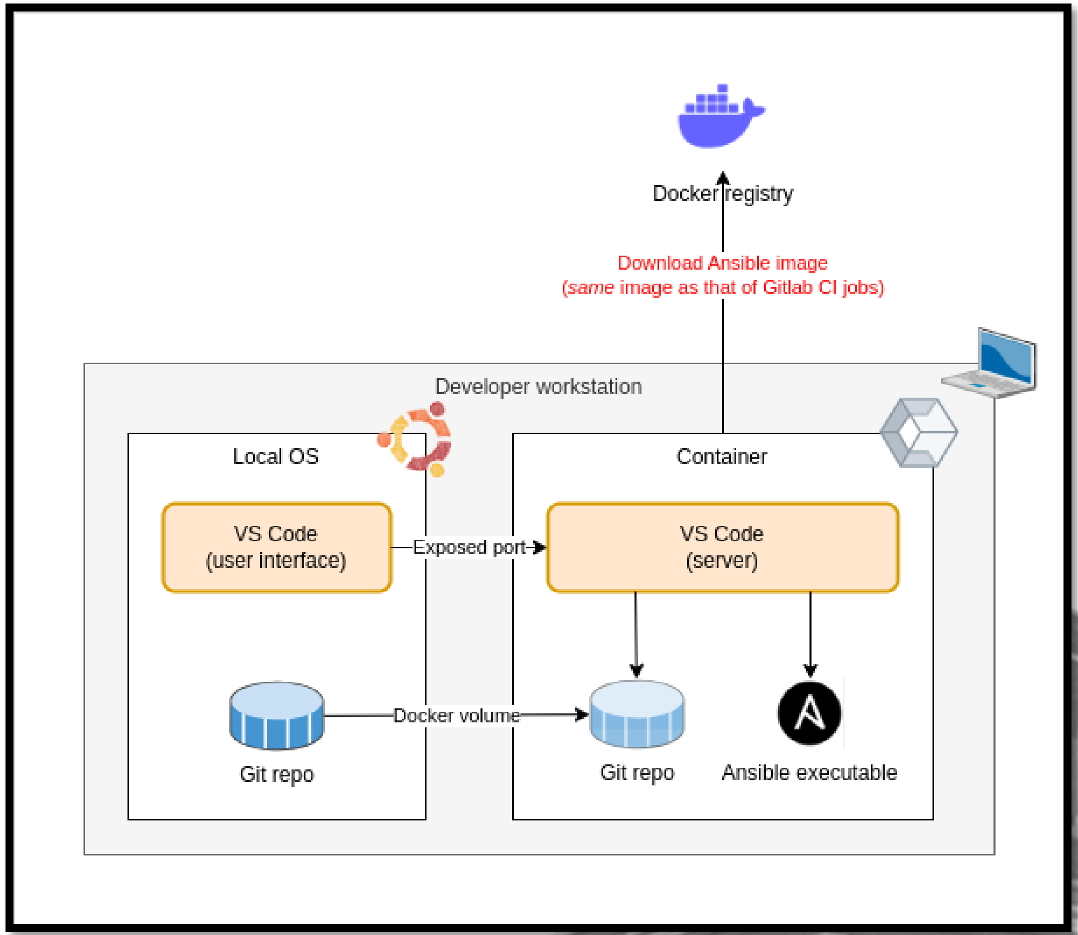
Gitlab pipelines: VS Code "Dev Containers"

Regular VS Code



VS Code with "Dev Containers" extension

https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers

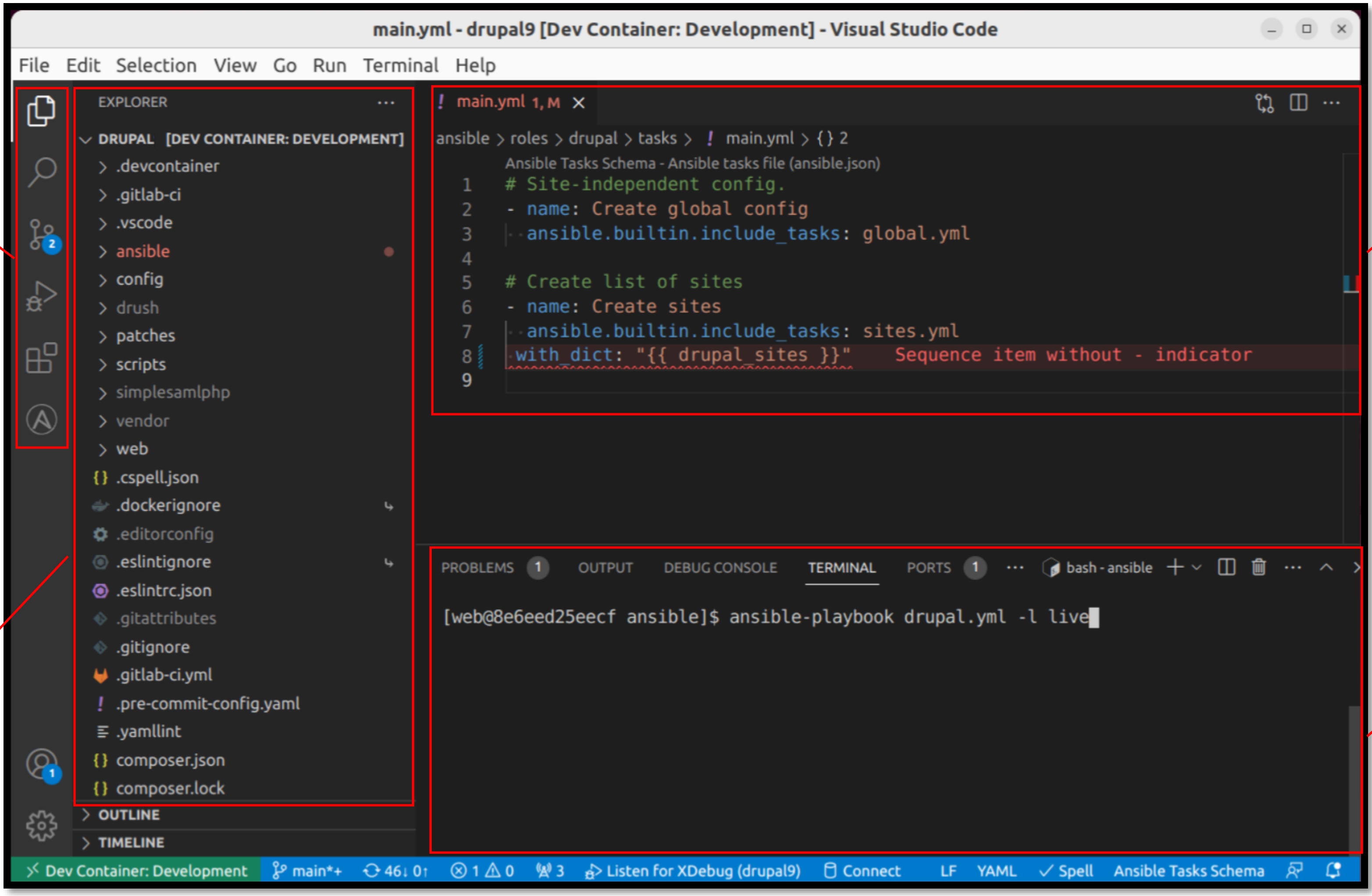


Gitlab pipelines: VS Code "Dev Containers"

VS Code runs within the *same* Docker image as Gitlab CI jobs (same Ansible version, OS version, etc).
Allows to reproduce the execution environment of the Gitlab CI jobs.

Extensions installed *inside* the container

Edit files *inside* the container


main.yml - drupal9 [Dev Container: Development] - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
DRUPAL [DEV CONTAINER: DEVELOPMENT]
.devcontainer
.gitlab-ci
.vscode
ansible
config
drush
patches
scripts
simplesamlphp
vendor
web
.cspell.json
.dockerignore
.editorconfig
.eslintignore
.eslintrc.json
.gitattributes
.gitignore
.gitlab-ci.yml
.pre-commit-config.yaml
.yamllint
composer.json
composer.lock
OUTLINE
TIMELINE
main.yml 1, M
ansible > roles > drupal > tasks > main.yml > {} 2
Ansible Tasks Schema - Ansible tasks file (ansible.json)
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
bash - ansible
Dev Container: Development
main*+
46↓ 0↑
1 0
3
Listen for XDebug (drupal9)
Connect
LF
YAML
Spell
Ansible Tasks Schema

```yaml
# Site-independent config.
- name: Create global config
  ansible.builtin.include_tasks: global.yml

# Create list of sites
- name: Create sites
  ansible.builtin.include_tasks: sites.yml
 with_dict: "{{ drupal_sites }}"    Sequence item without - indicator
```

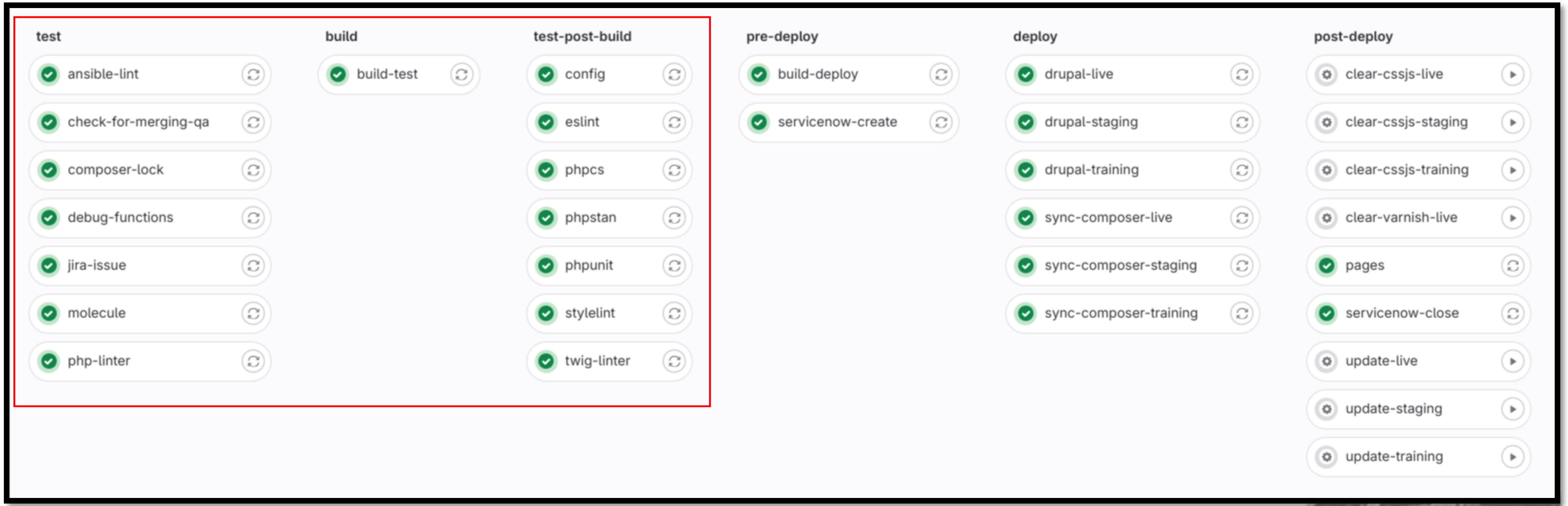
```
[web@8e6eed25eecf ansible]$ ansible-playbook drupal.yml -l live
```

Explorer shows files *within* the container

Shell opens *within* the container

Tests & Quality

Tests in Gitlab CI pipeline



test	build	test-post-build	pre-deploy	deploy	post-deploy
ansible-lint	build-test	config	build-deploy	drupal-live	clear-cssjs-live
check-for-merging-qa		eslint	servicenow-create	drupal-staging	clear-cssjs-staging
composer-lock		phpcs		drupal-training	clear-cssjs-training
debug-functions		phpstan		sync-composer-live	clear-varnish-live
jira-issue		phpunit		sync-composer-staging	pages
molecule		stylelint		sync-composer-training	servicenow-close
php-linter		twig-linter			update-live
					update-staging
					update-training

Ansible Lint

Automated code review for Ansible.

Allows to detect:

- YAML formatting errors
- Risky file permissions
- Obsolete Ansible modules
- Etc.

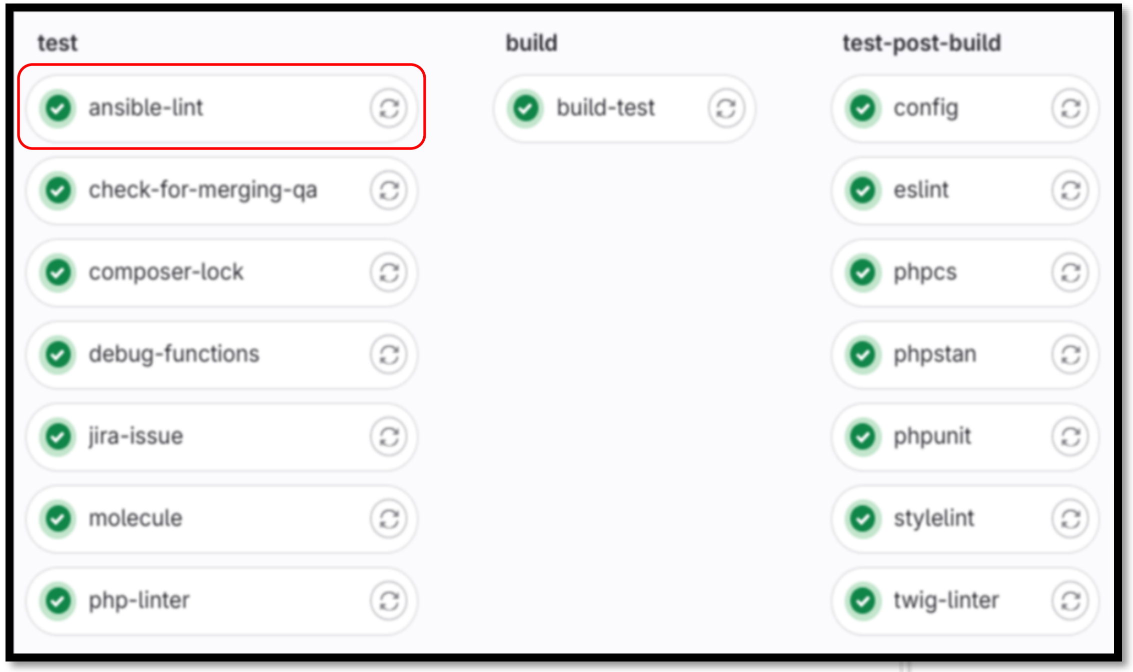
We run it in:

1. VS Code (code edits)
2. Git pre-commit hooks (code commits)
3. Gitlab CI pipelines (code push)

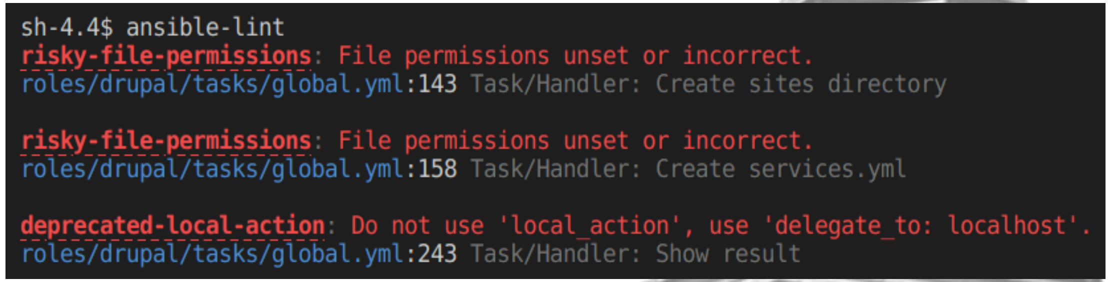
https://github.com/ansible/ansible-lint

https://marketplace.visualstudio.com/items?itemName=redhat.ansible

https://pre-commit.com/hooks.html



test	build	test-post-build
✓ ansible-lint	✓ build-test	✓ config
✓ check-for-merging-qa		✓ eslint
✓ composer-lock		✓ phpcs
✓ debug-functions		✓ phpstan
✓ jira-issue		✓ phpunit
✓ molecule		✓ stylelint
✓ php-linter		✓ twig-linter



```
sh-4.4$ ansible-lint
risky-file-permissions: File permissions unset or incorrect.
roles/drupal/tasks/global.yml:143 Task/Handler: Create sites directory

risky-file-permissions: File permissions unset or incorrect.
roles/drupal/tasks/global.yml:158 Task/Handler: Create services.yml

deprecated-local-action: Do not use 'local_action', use 'delegate_to: localhost'.
roles/drupal/tasks/global.yml:243 Task/Handler: Show result
```

Molecule

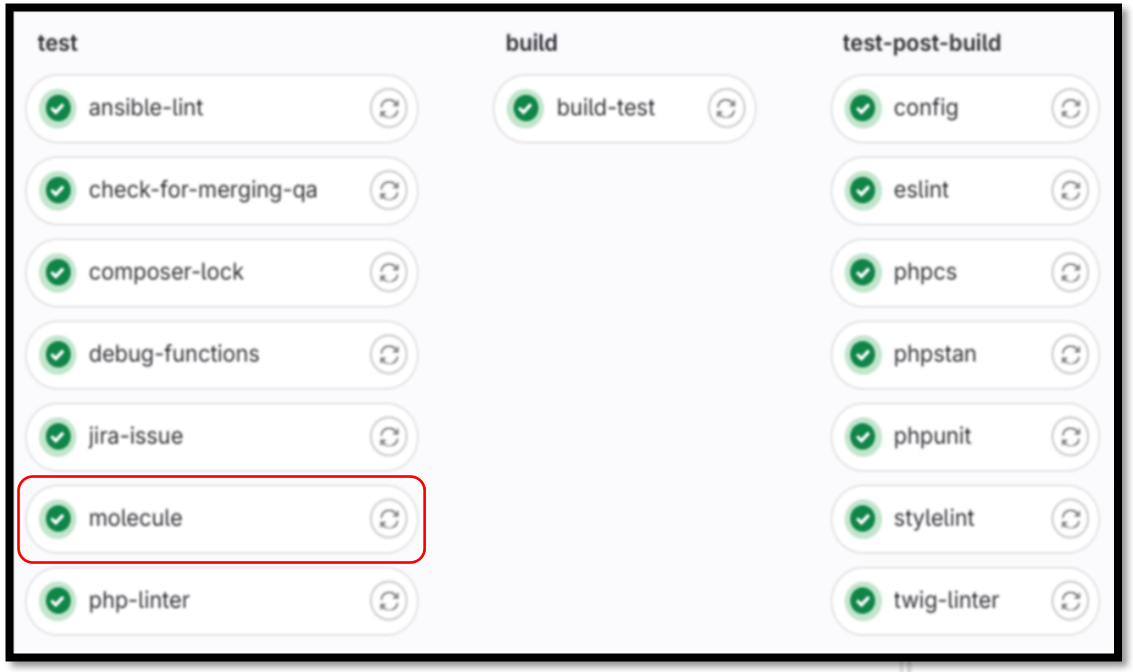
Automated tests for Ansible.

Runs a sequence of tasks:

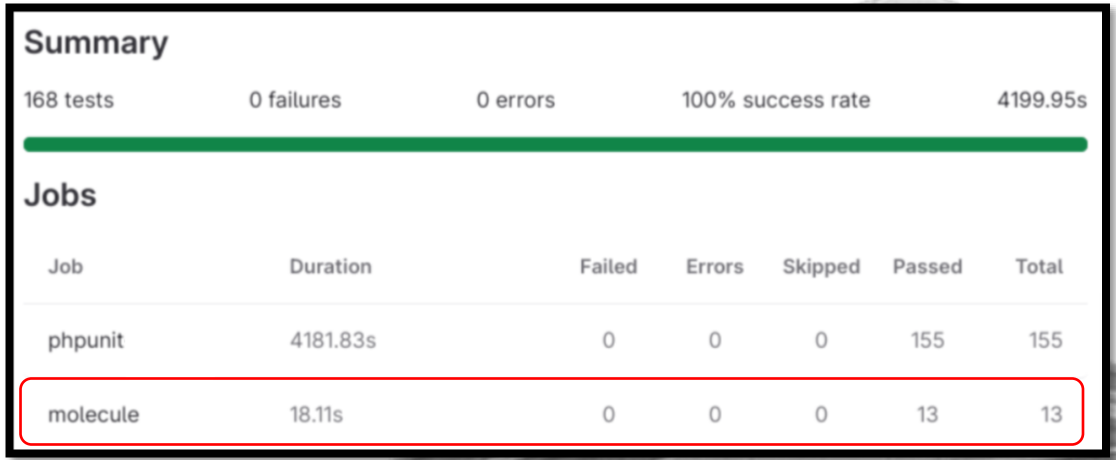
1. "Prepare": initialize the test environment
2. "Converge": execute Ansible tasks
3. "Verify": test what the Ansible tasks did
4. "Clean up": clean up the test environment

Summary of results exported to Gitlab and can be viewed directly in the web interface (Merge Request), along with other automated tests (PHPUnit).

https://github.com/ansible-community/molecule



test	build	test-post-build
✓ ansible-lint	✓ build-test	✓ config
✓ check-for-merging-qa		✓ eslint
✓ composer-lock		✓ phpcs
✓ debug-functions		✓ phpstan
✓ jira-issue		✓ phpunit
✓ molecule		✓ stylelint
✓ php-linter		✓ twig-linter



Summary

168 tests 0 failures 0 errors 100% success rate 4199.95s

Jobs

Job	Duration	Failed	Errors	Skipped	Passed	Total
phpunit	4181.83s	0	0	0	155	155
molecule	18.11s	0	0	0	13	13

PHP static analysis

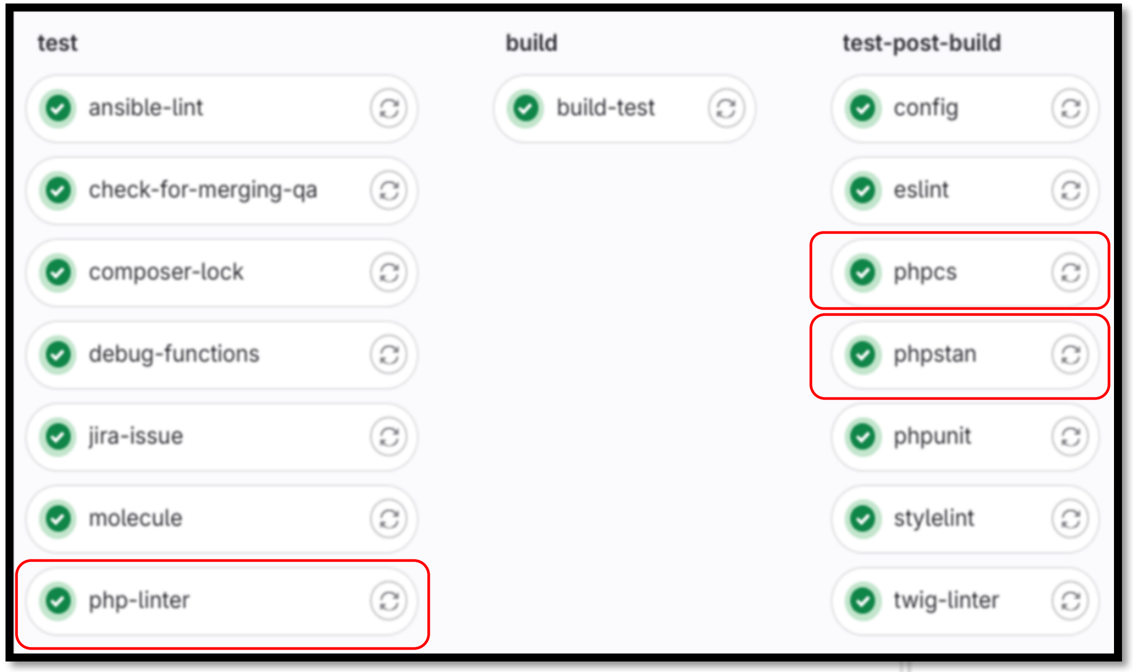
Tools to statically analyze the code:

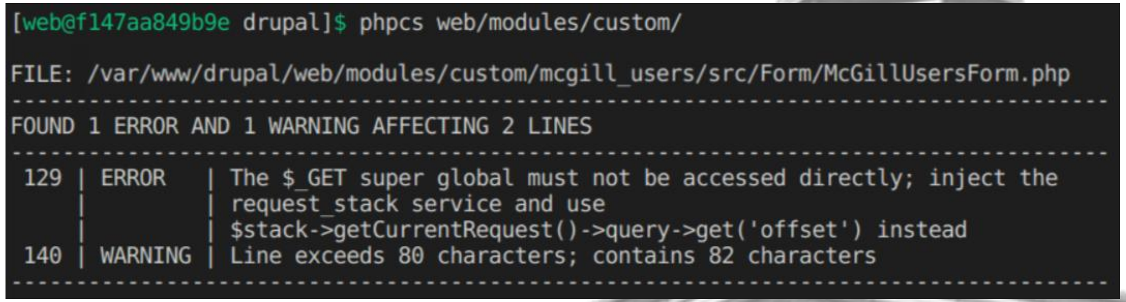
- PHP Linter (built into PHP)
- PHP Code Sniffer https://github.com/PHPCSStandards/PHP_CodeSniffer/
    - Drupal
    - DrupalPractice
    - PHPCompatibility
- PHPStan https://phpstan.org/

Allows to detect:

- PHP syntax errors
- Debug functions e.g. kint(), debug_backtrace()
- Compliance with Drupal coding standards
- Dead code, undefined variables, methods, classes
- Drupal best practices e.g. dependency injection, translation, sanitization, security

Install via the **drupal/core-dev** composer package so we run the same version as Drupal core.





```
[web@f147aa849b9e drupal]$ phpcs web/modules/custom/

FILE: /var/www/drupal/web/modules/custom/mcgill_users/src/Form/McGillUsersForm.php
------------------------------------------------------------------------------------
FOUND 1 ERROR AND 1 WARNING AFFECTING 2 LINES
------------------------------------------------------------------------------------
 129 | ERROR   | The $_GET super global must not be accessed directly; inject the
     |         | request_stack service and use
     |         | $stack->getCurrentRequest()->query->get('offset') instead
 140 | WARNING | Line exceeds 80 characters; contains 82 characters
------------------------------------------------------------------------------------
```

Composer

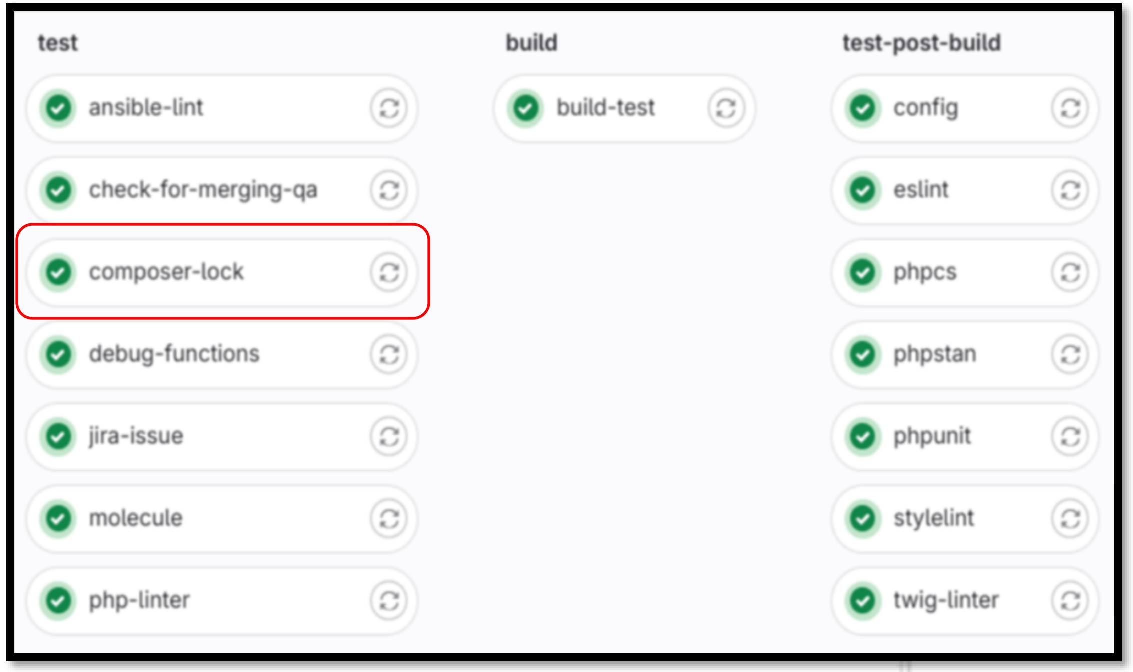
Validates composer.json and composer.lock.

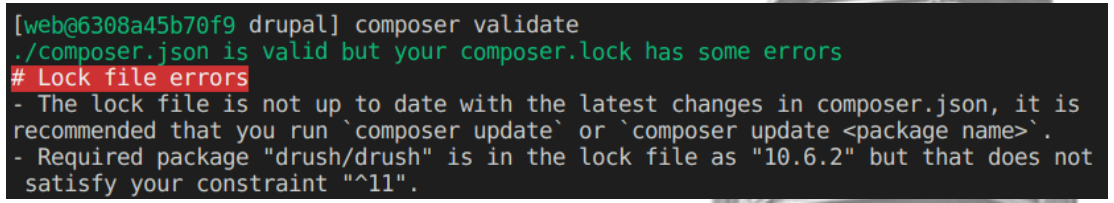
Allows to detect:

- JSON formatting errors
- Missing packages in composer.lock
- Unsatisfied version constraints in composer.lock
- Contents of composer.lock out of date with composer.json (metadata, list of repos, dependencies, constraints, patches...)

Errors typically introduced when:

- Merging composer.json or composer.lock
- Manually changing these files





```
[web@6308a45b70f9 drupal] composer validate
./composer.json is valid but your composer.lock has some errors
# Lock file errors
- The lock file is not up to date with the latest changes in composer.json, it is
recommended that you run `composer update` or `composer update <package name>`.
- Required package "drush/drush" is in the lock file as "10.6.2" but that does not
 satisfy your constraint "^11".
```

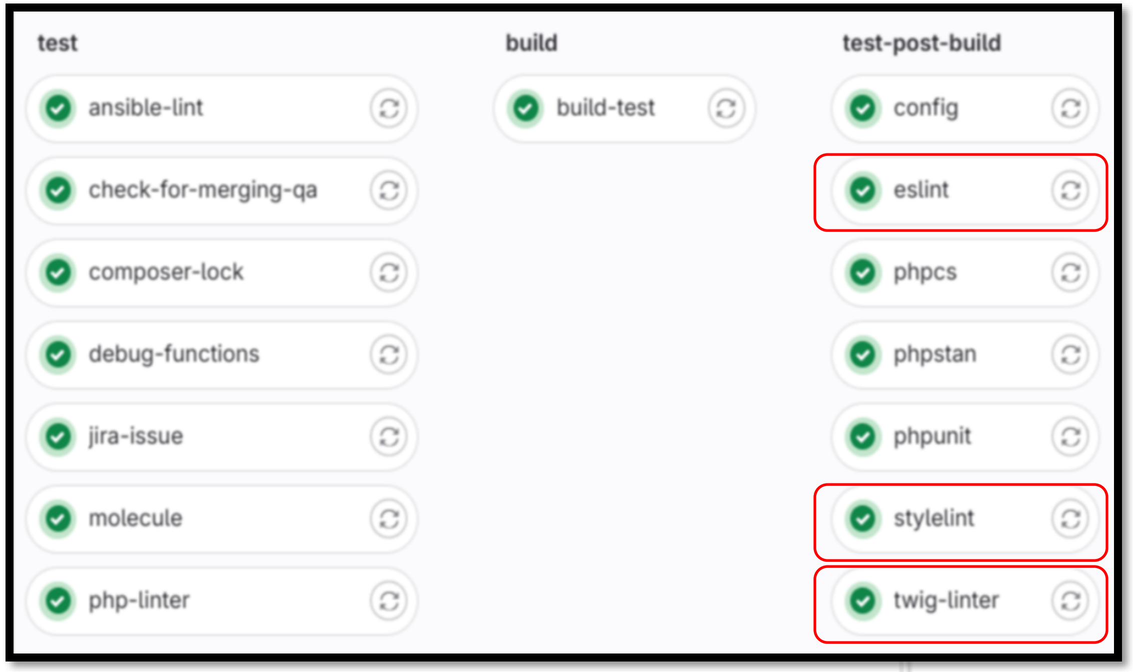
Frontend static analysis

Eslint & Stylelint

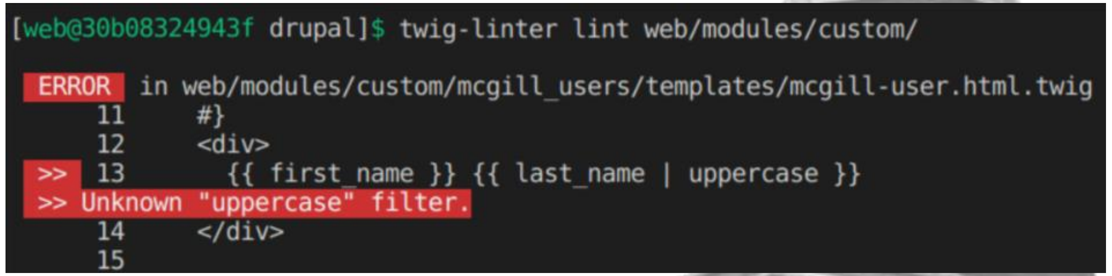
- JS and CSS linters
- Ensure custom JavaScript/CSS have a valid syntax and are compatible with Drupal coding standards
- Re-use config provided by Drupal core (.eslintrc.json and .stylelintrc.json)

Twig-linter

- Ensure custom Twig templates have a valid syntax and use valid filters
- https://github.com/sserbin/twig-linter



test	build	test-post-build
✓ ansible-lint	✓ build-test	✓ config
✓ check-for-merging-qa		✓ eslint
✓ composer-lock		✓ phpcs
✓ debug-functions		✓ phpstan
✓ jira-issue		✓ phpunit
✓ molecule		✓ stylelint
✓ php-linter		✓ twig-linter



```
[web@30b08324943f drupal]$ twig-linter lint web/modules/custom/

 ERROR  in web/modules/custom/mcgill_users/templates/mcgill-user.html.twig
      11      #}
      12      <div>
 >>   13        {{ first_name }} {{ last_name | uppercase }}
 >> Unknown "uppercase" filter.
      14      </div>
      15
```

Drupal config

Ensure Drupal exported config is internally consistent.

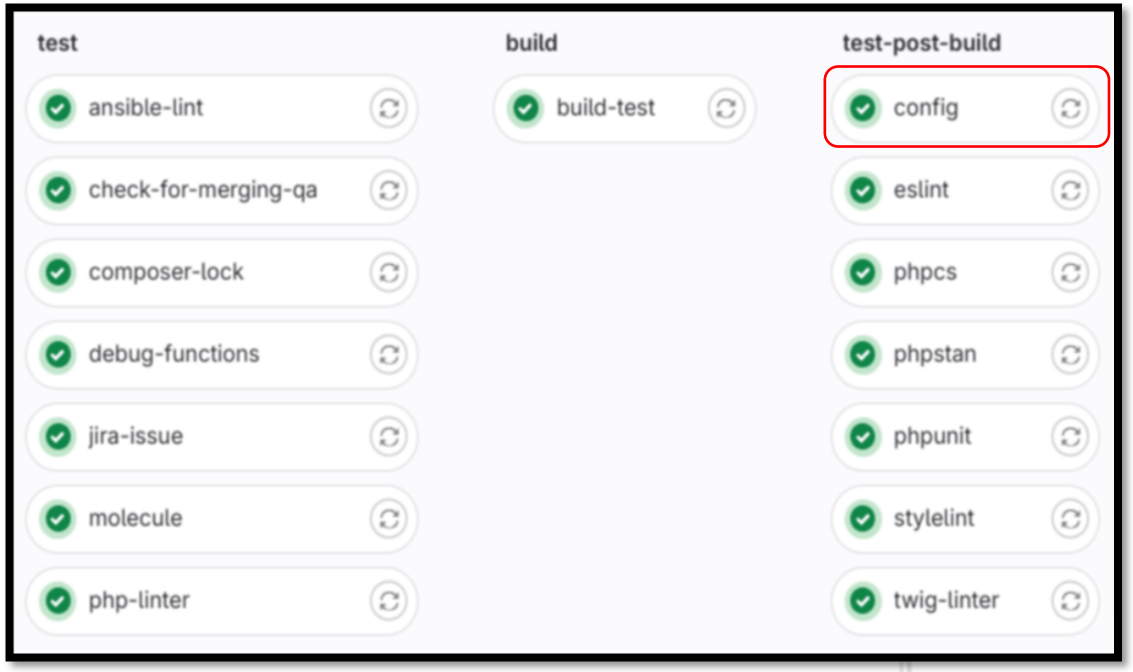
How to test:

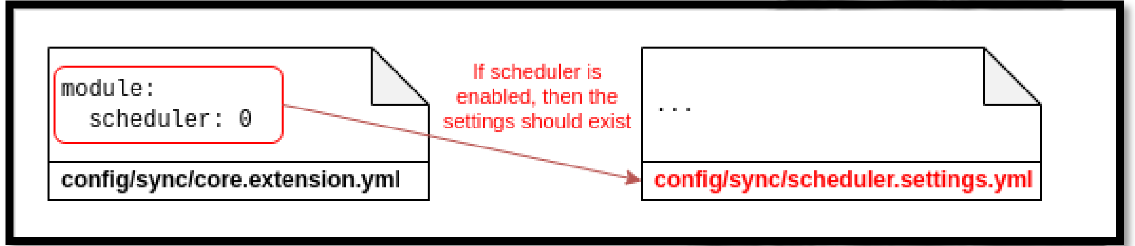
1. Install fresh site from config export
   drush site:install --existing-config
2. Check if active config matches exported config
   drush config:status
3. If a discrepancy is found, then fail

Example:

Module "scheduler" includes configuration settings.

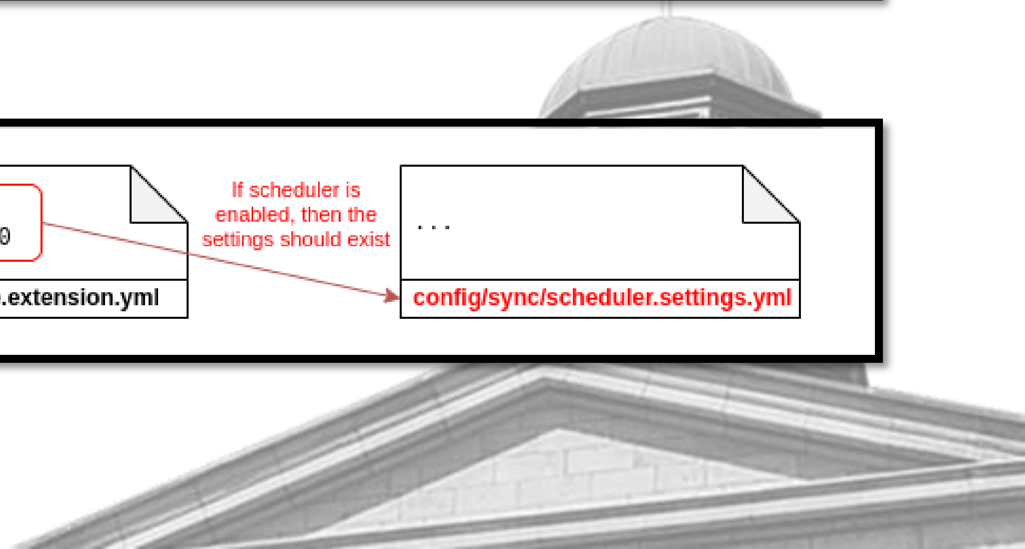
If that module is listed as enabled in the exported config, then the associated configuration should also exist as a YAML file.





```
module:
  scheduler: 0
```

config/sync/core.extension.yml
If scheduler is enabled, then the settings should exist
...
config/sync/scheduler.settings.yml

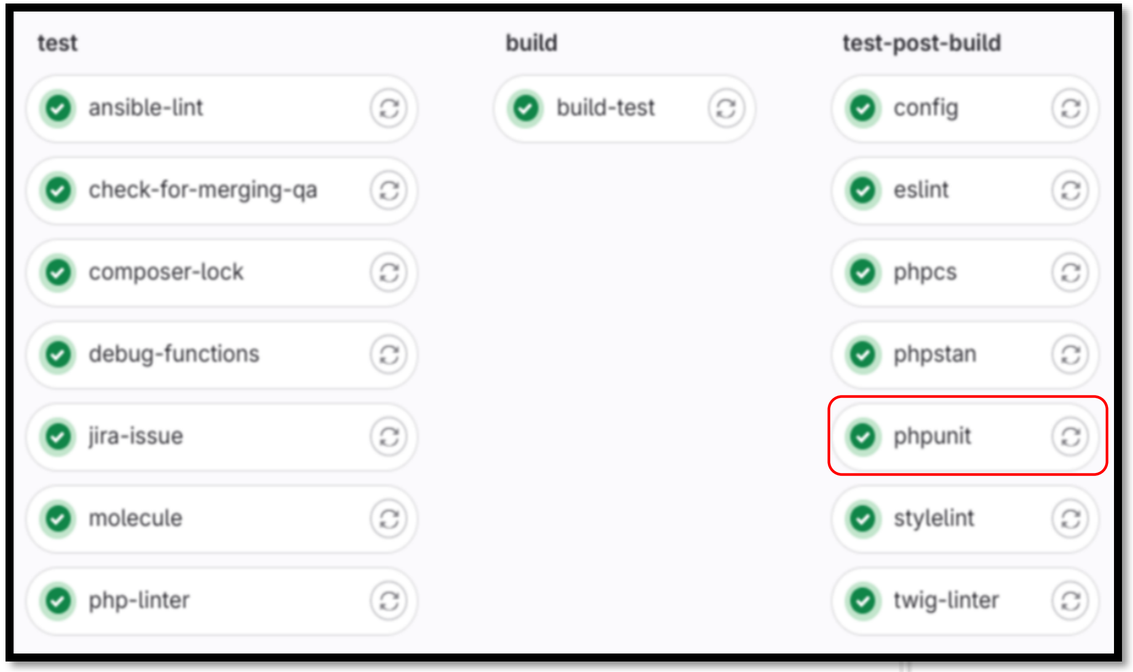


PHPUnit

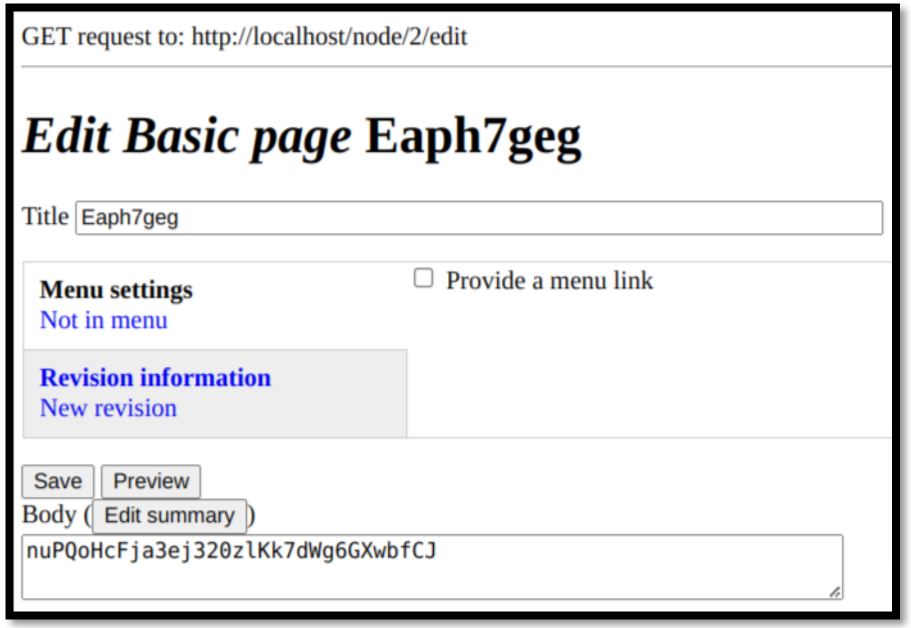
Automated tests for Drupal.

Summary of results exported to Gitlab and can be viewed directly in the web interface (Merge Request).

HTML output of Functional tests exported as a Gitlab artifact and published as a static site on Gitlab Pages.



test	build	test-post-build
✓ ansible-lint	✓ build-test	✓ config
✓ check-for-merging-qa		✓ eslint
✓ composer-lock		✓ phpcs
✓ debug-functions		✓ phpstan
✓ jira-issue		✓ phpunit
✓ molecule		✓ stylelint
✓ php-linter		✓ twig-linter



GET request to: http://localhost/node/2/edit

Edit Basic page Eaph7geg

Title Eaph7geg

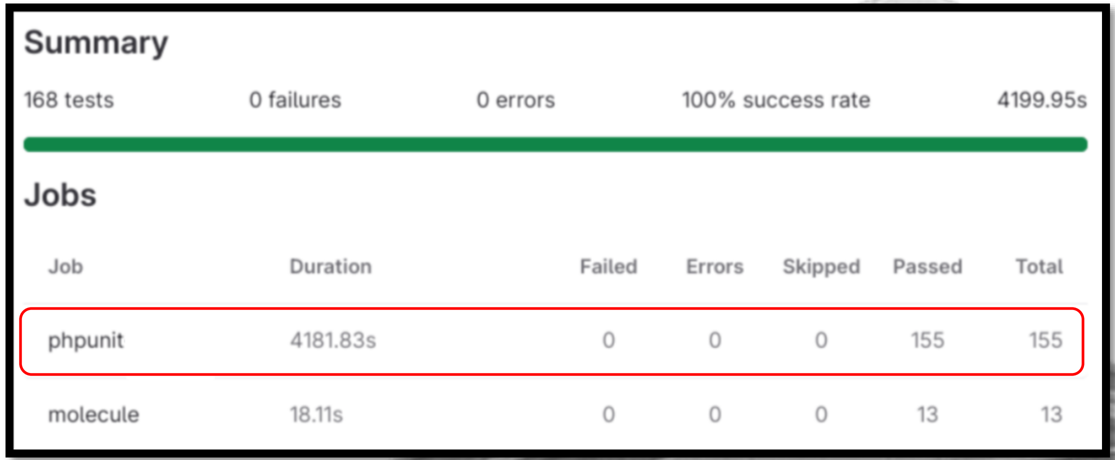
Menu settings
Not in menu

☐ Provide a menu link

Revision information
New revision

Body (Edit summary)

```
nuPQoHcFja3ej320zlKk7dWg6GXwbfCJ
```



Summary

168 tests 0 failures 0 errors 100% success rate 4199.95s

Jobs

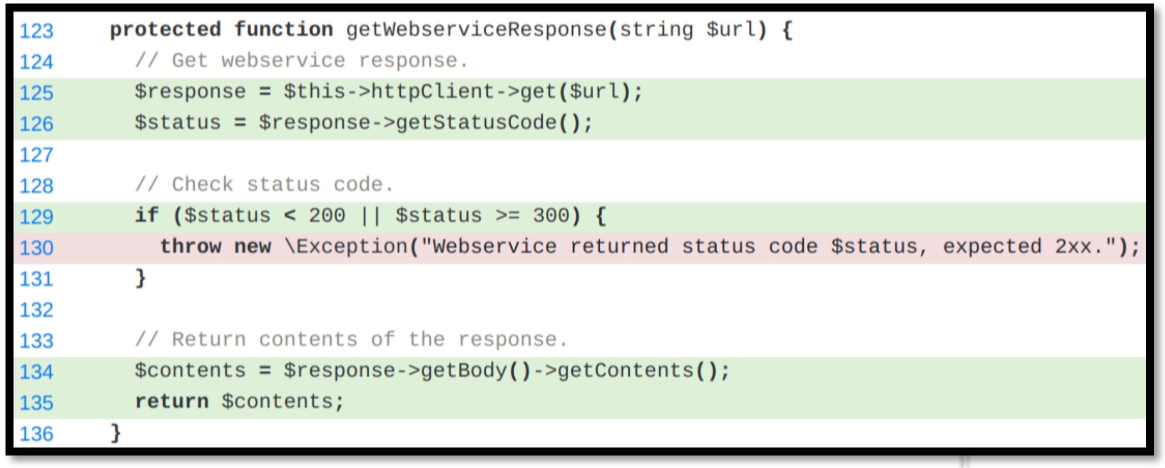
Job	Duration	Failed	Errors	Skipped	Passed	Total
phpunit	4181.83s	0	0	0	155	155
molecule	18.11s	0	0	0	13	13

Code coverage report (PHPUnit)

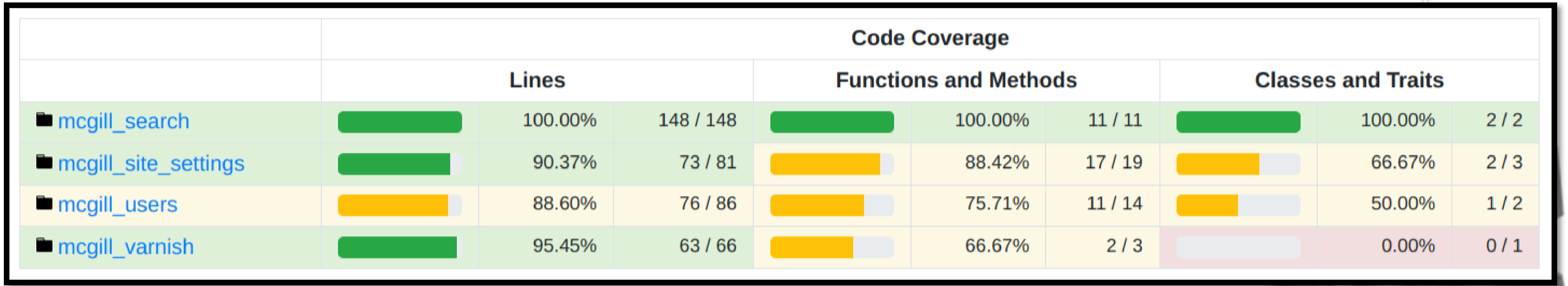
Shows which parts of the code are executed when PHPUnit tests are run.

Gotcha: accounts for Unit and Kernel tests, but not Functional tests.

Report generated with Xdebug and published as a static site on Gitlab Pages.



```php
  protected function getWebserviceResponse(string $url) {
    // Get webservice response.
    $response = $this->httpClient->get($url);
    $status = $response->getStatusCode();

    // Check status code.
    if ($status < 200 || $status >= 300) {
      throw new \Exception("Webservice returned status code $status, expected 2xx.");
    }

    // Return contents of the response.
    $contents = $response->getBody()->getContents();
    return $contents;
  }
```



	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
mcgill_search	🟩	100.00%	148 / 148	🟩	100.00%	11 / 11	🟩	100.00%	2 / 2
mcgill_site_settings	🟩	90.37%	73 / 81	🟨	88.42%	17 / 19	🟨	66.67%	2 / 3
mcgill_users	🟨	88.60%	76 / 86	🟨	75.71%	11 / 14	🟨	50.00%	1 / 2
mcgill_varnish	🟩	95.45%	63 / 66	🟨	66.67%	2 / 3		0.00%	0 / 1

Upcoming challenges

✓ Preview changes before deployment (Ansible's "check" mode)

Automate synchronization of secrets with our password manager

Run phpunit in concurrent mode

Increase automated tests coverage

Deploy to the Cloud/Kubernetes

Thanks

Download these slides:

https://github.com/fengtan/fengtan

Thomas Fline

thomas.fline@mcgill.ca

https://drupal.org/u/fengtan

https://github.com/fengtan

https://linkedin.com/in/thomasfline

https://www.mcgill.ca/it