

Industrialiser Drupal dans un grand groupe

Par Simon VART & Guillaume ROUXELIN





Deux REX pour le prix d'un

2 projets
séparés sur
Drupal 8 / 9

- Simon :
Solocal.com
- Guillaume :
Ooreka Verticales

On travaille
ensemble
pendant
quelques mois

Changement(s)
de mission(s)



1. Acquia matata



Un mot de contexte



- Solutions Digitales pour entreprises locales – PagesJaunes – 280.000 clients – 14M visites
- 2018 : lancement d'une nouvelle offre commerciale orientée e-commerce
- In scope : capacité d'inscription en ligne, acquisition de leads
- Soon : Campagne TV de publicité
- Passerelle pour services tiers (CRM, APIs, landing...)



Solocal.com

- ~1M visites/mois
- Drupal 8
- En cours de refonte

TMA

- Contenu
- Actualités
- Données financières
- Marketing

E-commerce

- Agile
- POC
- Code custom
- APIs

Hébergement

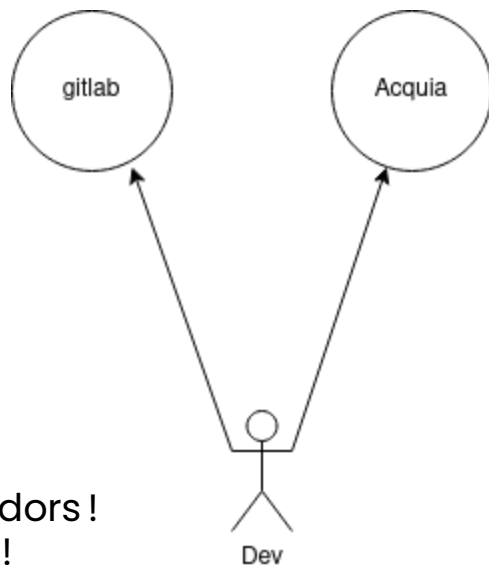
- Acquia
- Acquia is the Premier Drupal Company



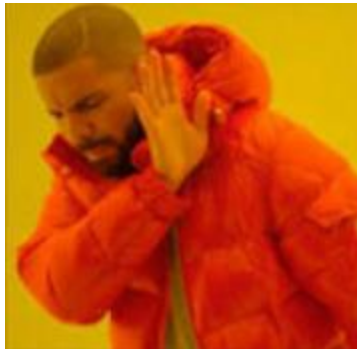
Acquia Cloud Platform



Réception du site



Build local !
Commit vendors !
Double repo !





WHAT DO WE WANT ?



DEPLOY



WHERE ?



ACQUIA



HOW ?



WITH GIT



WHEN DO WE WANT IT ?



ANYTIME

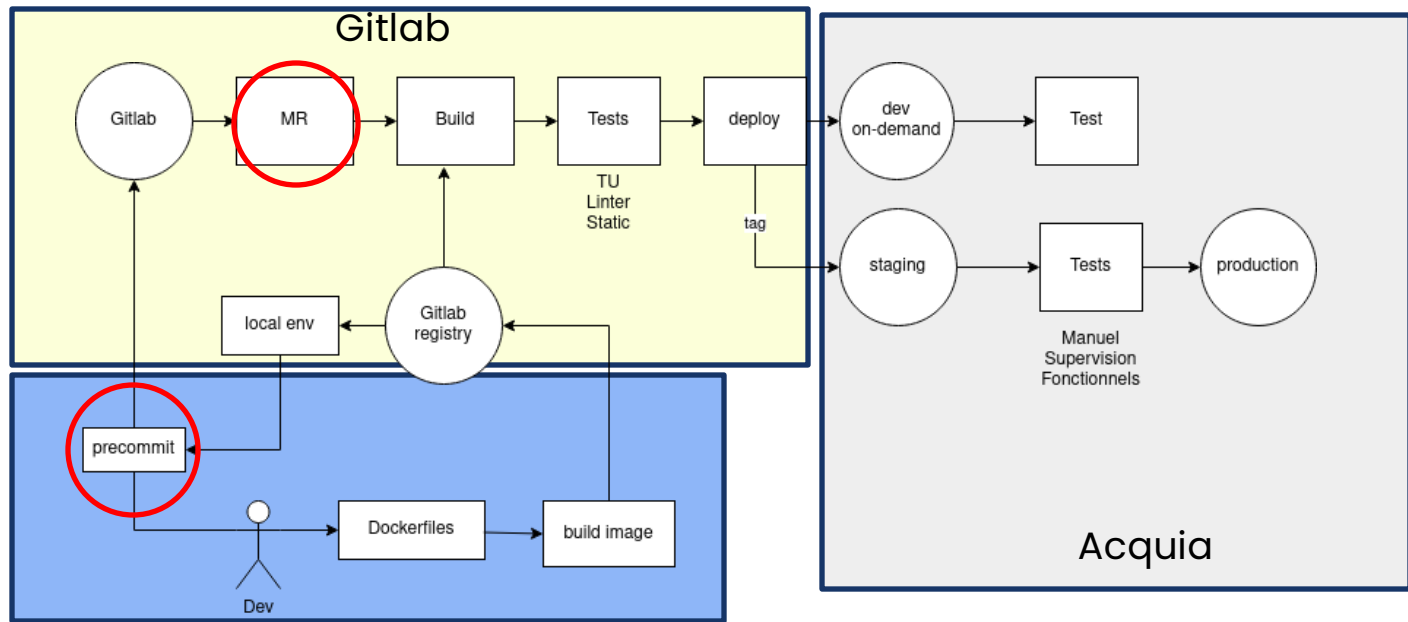


imgflip.com



Solution : Pipeline

défini notre façon de produire et de s'assurer de la qualité de code



- Adhérence applicative aux versions (PHP, MariaDB) de production
- Repository Acquia est un repository des versions applicatives (tags)
- Gitlab : private composer repository pour les modules custom (share)
- Symfony/asset pour la gestion de version (Varnish/cloudflare)
- Tests bloquants (sécurité, linting)
- Tests fonctionnels (behat) avec déploiement on-demand





Key points

ISO PROD

Code is all

All is code

MR centric

Sanctuariser

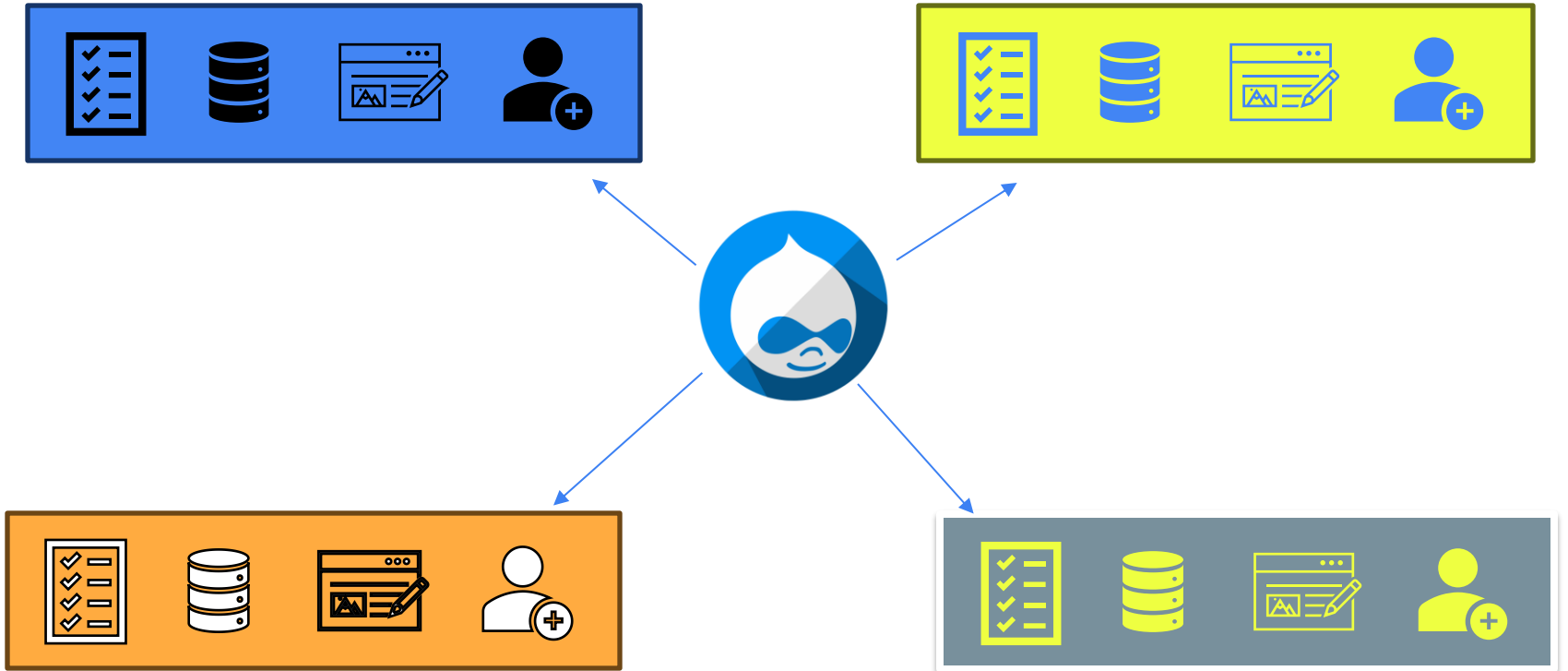
Contraindre



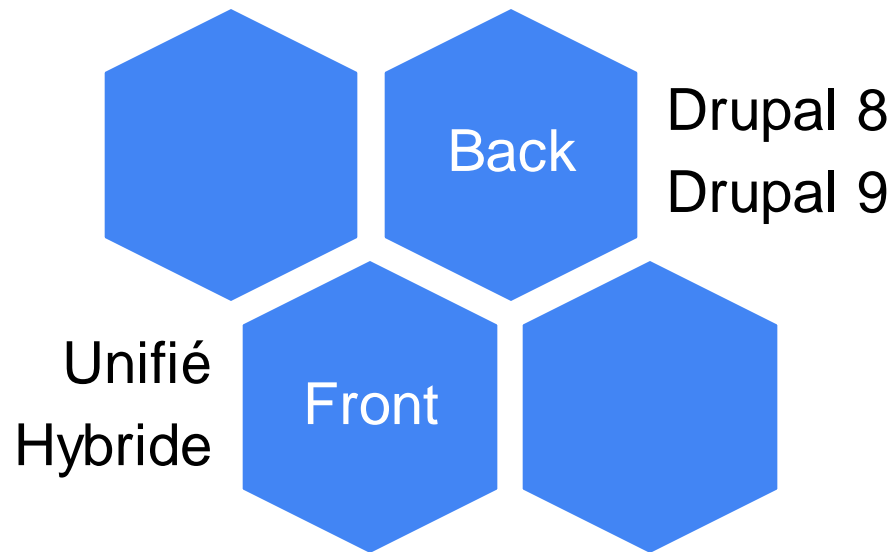
2. Un seul pour les gouverner tous



Multisites



Multisites : Ooreka Verticales





Multisites : Avantages & Inconvénients



Avantages

- Facilite l'arrivée d'un nouveau développeur
- Permet à un développeur de basculer plus vite d'un projet à l'autre
- Facilite l'installation d'un nouveau projet



Inconvénients

- Pensez le développement au global
- Compétences techniques pour créer et configurer les nouveaux sites
- Très forte dépendance entre les sites en cas de mise à jour majeure



Multisites : Outils du quotidien

Docker

- Tous nos projets étaient dockerisés
- Docker-compose + env.

Gulpfile

- Compilation des fichiers SASS

Gitlab CI / CD

- Qualité de code
- Acquia Build and Launch Tools (BLT)

La pratique de l'[intégration continue](#) (CI) consiste à intégrer [automatiquement](#) et régulièrement les modifications de code dans un référentiel de code source partagé. [La distribution et/ou le déploiement continu](#) (CD) désigne quant à eux un processus en deux volets qui englobe l'intégration, les tests et la distribution des modifications apportées au code

<https://www.redhat.com/fr/topics/devops/what-is-ci-cd>



Exemple fichier gitlab-ci.yml pour Drupal 10

```
phpcs:  
  stage: qa  
  extends: .job_template  
  rules:  
    - if: '$CI_SKIP_QA == "1" || $CI_SKIP_QA_PHPCS == "1"'  
      when: never  
    - when: on_success  
  script:  
    - phpcs  
      --standard=${CI_QA_PHPCS_STANDARD}  
      --ignore=${CI_QA_IGNORE}  
      --extensions=${CI_QA_SUFFIX}  
      --report-junit=report-${CI_JOB_NAME}/phpcs_junit.xml  
      ${CI_DIRS_QA_PHPCS}
```

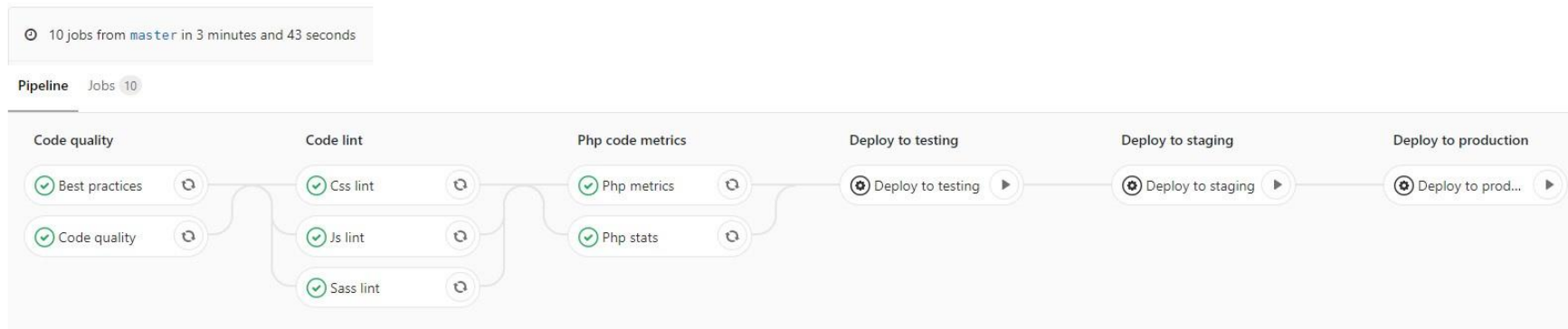
https://gitlab.com/mog33/gitlab-ci-drupal/-/blob/4.x-dev/gitlab-ci/jobs/gitlab-ci.yml?ref_type=heads



Pipeline

Un pipeline dans **GitLab CI/CD** est constitué une série d'étapes qui sont exécutées **automatiquement** chaque fois qu'il y a un changement dans votre référentiel de code source. Ces étapes sont conçues pour **automatiser** diverses tâches, telles que la compilation du code, l'exécution de tests, le déploiement d'applications, et bien plus encore.

<https://blog.stephane-robert.info/docs/pipeline-cicd/gitlab/introduction/>



<https://developpeur-drupal.com/article/drupal-8-gitlab-ci>



Multisites : Zoom sur le makefile

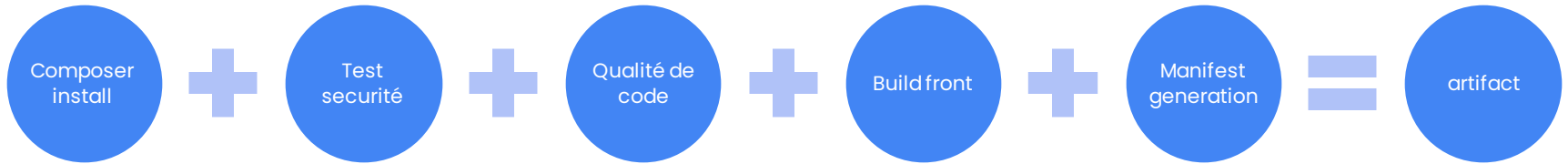
```
## === PHPQA =====  
qa-cs-fixer-dry-run-modules: # Run php-cs-fixer in dry-run mode  
  $(PHPQA_RUN) php-cs-fixer fix ./web/modules/custom --rules=@PSR12 --verbose --dry-run  
.PHONY: qa-cs-fixer-dry-run-modules  
  
qa-cs-fixer-modules: ## Run php-cs-fixer  
  $(PHPQA_RUN) php-cs-fixer fix ./web/modules/custom --rules=@PSR12 --verbose  
.PHONY: qa-cs-fixer-modules  
  
qa-cs-fixer-dry-run-themes: # Run php-cs-fixer in dry-run mode  
  $(PHPQA_RUN) php-cs-fixer fix ./web/themes/custom/vinylis --rules=@PSR12 --verbose --dry-run  
.PHONY: qa-cs-fixer-dry-run-themes  
  
qa-cs-fixer-themes: ## Run php-cs-fixer  
  $(PHPQA_RUN) php-cs-fixer fix ./web/themes/custom/vinylis --rules=@PSR12 --verbose  
.PHONY: qa-cs-fixer-themes  
  
qa-phpstan: ## Run phpstan.  
  $(PHPQA_RUN) phpstan analyse ./web/modules/custom ./web/themes/custom/vinylis --level=3  
.PHONY: qa-phpstan  
  
qa-phpcpd: ## Run phpcpd (copy/paste detector).  
  $(PHPQA_RUN) phpcpd ./web/modules/custom ./web/themes/custom/vinylis  
.PHONY: qa-phpcpd  
  
## ==== DRUPALQA =====  
drupal-cs: #Run phpcs on custom modules  
  $(DRUPALQA_RUN) phpcs web/modules/custom
```



3. En route vers le déploiement continu



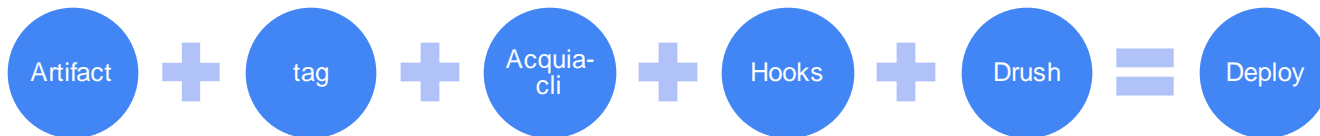
Build is easy



- Problème de sécurité, de linting, de qualité de code : bloquant
- Infrastructure as code
- Chaque commit de MR produit un artifact contenant une version
- Chaque déploiement produit un tag (= artifact)



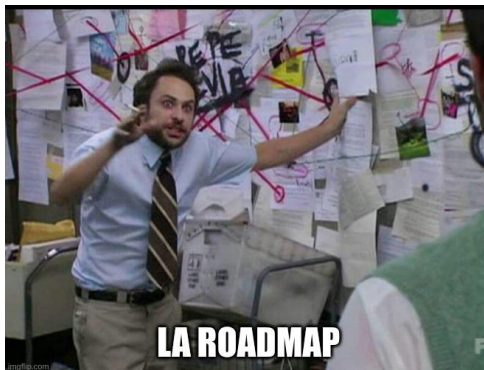
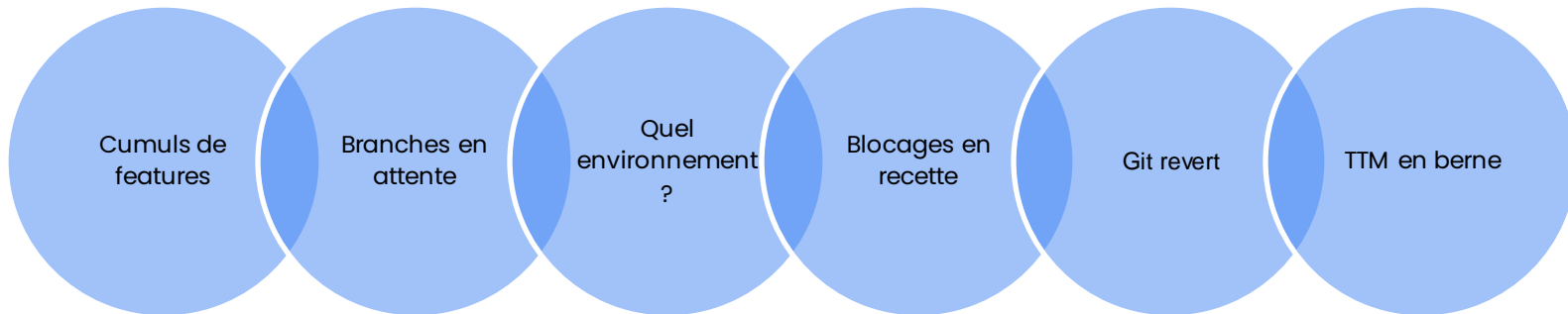
Deploy is easy



- Acquia cli = backups + BDD sync
- Hooks = scripts + drush
- Drush = cr / cim / updb / modules sync
- Modules sync : déclaration des modules dépendant dans services.yml, récupéré à la compilation du conteneur Symfony (!= config split), selon l'environnement
- Rollback = tag précédent (!BDD!)



Pain points





Simplify

Une feature = une MR = un pipeline = une recette = un déploiement = un tag = une version.

La communication : chaque MEP émet un message Teams avec un descriptif de la feature.
Le Kanban JIRA est synchronisé avec les étapes du pipeline.

Capacité de changer de feature à tout moment en fonction des capacités de recette.

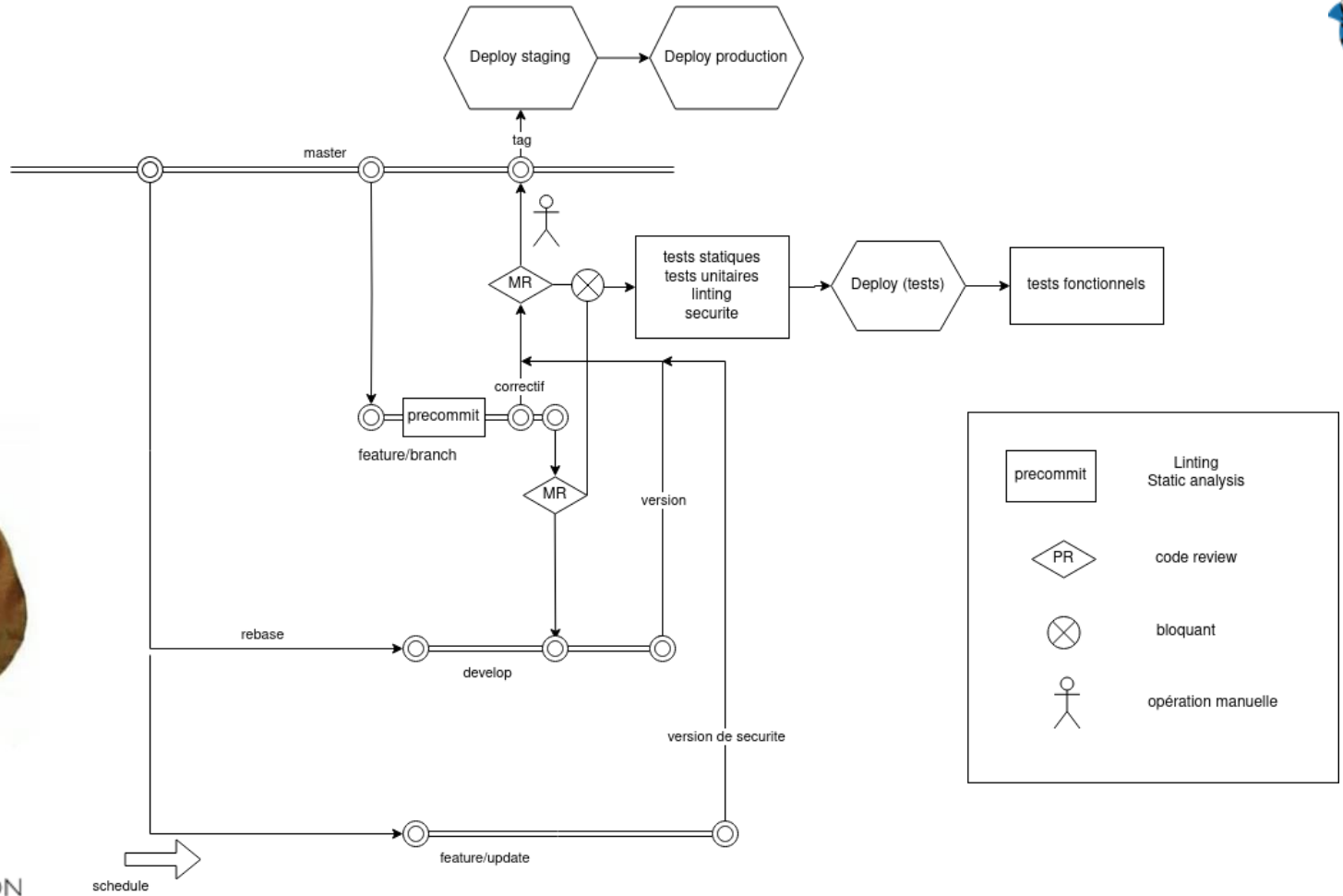
Les développeurs peuvent mettre en production les développements "techniques".

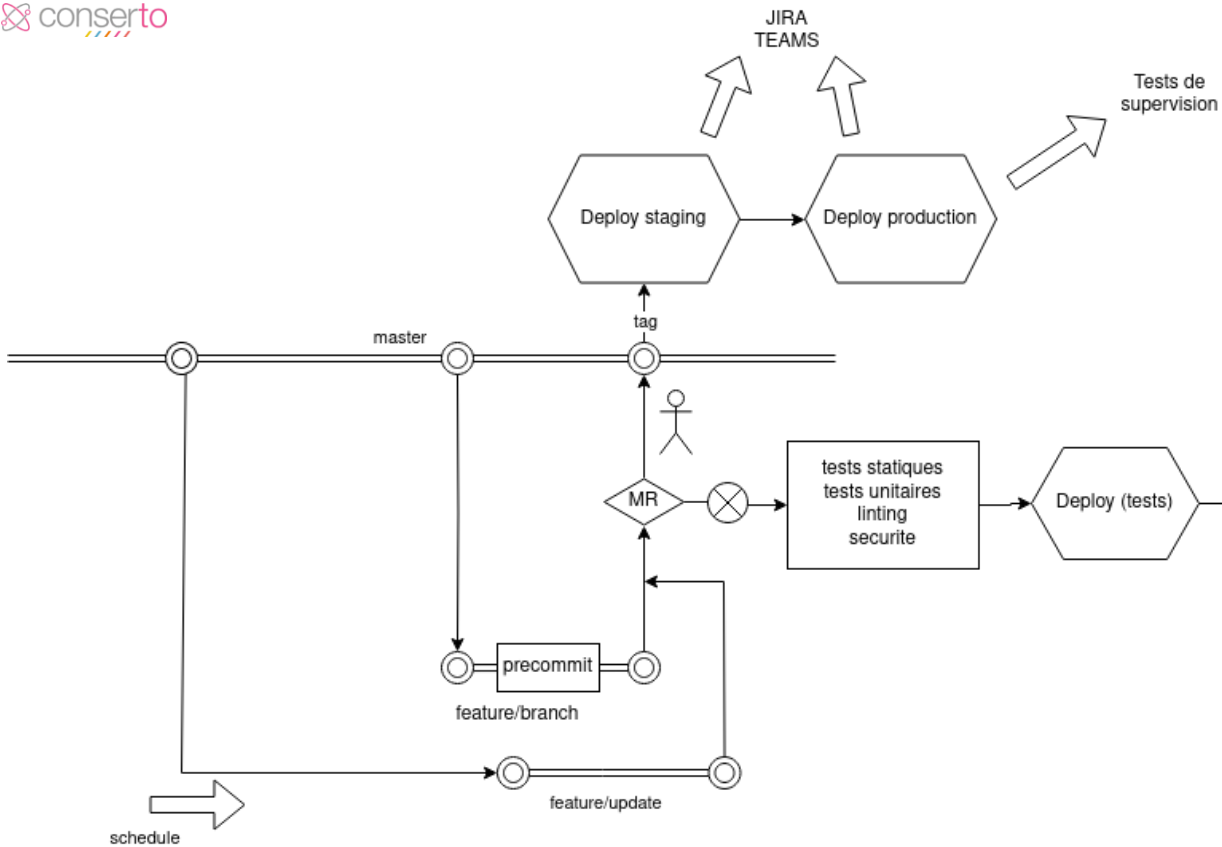
Les (rares) conflits de code sont toujours sur le poste local.

La MEP n'est plus un évènement (parfois plusieurs par jour).

Automatisation du déploiement des mises à jour de sécurité (composer update).

Une particularité : le merge est réalisé APRES le déploiement (sanctuarisation).








DEV : cas spécial
"pour les devs"
PF intégration avec capacité
de reset semi-automatique





Actions manuelles depuis le pipeline



[JIRA ] Correction 

 Open  requested to merge [feature/](#)  into [master](#) 1 week ago

Overview **0** Commits **5** Pipelines **11** Changes **7**

Closes 




 Merge request pipeline #1423443 waiting for manual action 


Merge request pipeline waiting for manual action for [67dbfd35](#) 4 days ago

[development](#)











Deployed to [staging](#) 3 days ago

[production](#)

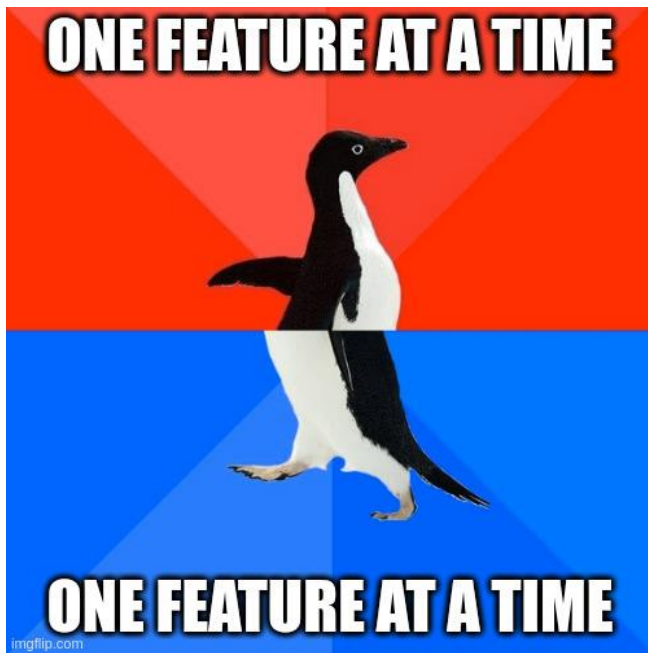
8  [Approve](#) Requires 1 approval from solocal devs approvers.

 Merge blocked: the source branch must be rebased onto the target branch. [Rebase](#)

Stage: staging

-  copy:db:prod:ode3 
-  copy:db:prod:stg 
-  crawl:stg 
-  deploy:acquia:ode3 
-  deploy:acquia:stg 

Side effects

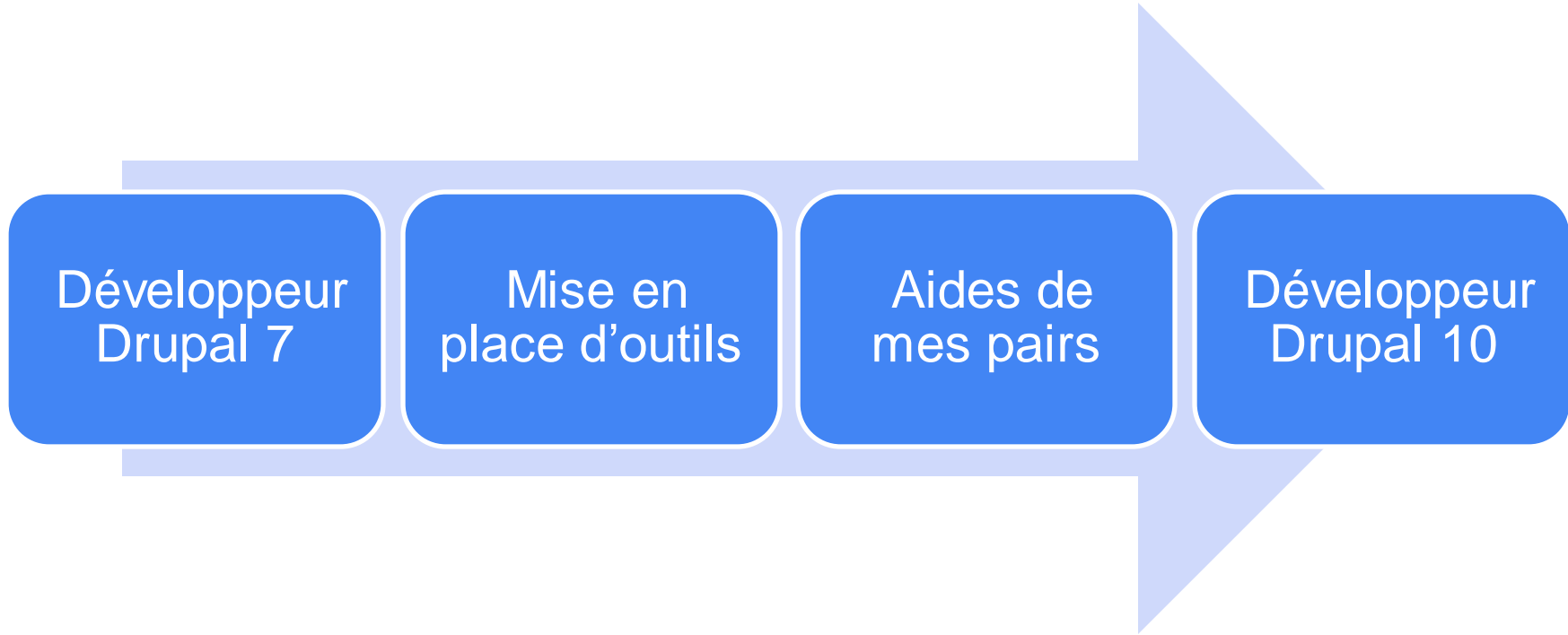




4. Développeur, tu deviendras



Montée en compétences





Montée en compétences : PHPSTORM

IDE pour PHP

Tout en un

PHPStorm ou vsCode



Montée en compétences : Plugins PHPStorm

Drupal

- Complétion de code spécifiques à Drupal
- Coding style

Symfony Support

- Complétion de code

Code with me

- Pair programming en distanciel
- Permet de coder à deux sur le même fichier

PHP Annotations

- Autocomplétion sur les annotations

PHP Inspections

- Analyse de code

PHP Toolbox

- Compléments pour Symfony Support et PHP Annotations





Montée en compétences : Xdebug

Utilisation pour le back

- Pour coder mes fonctions en mode pas à pas
- Pour vérifier l'évolution de ma variable au fur et à mesure des fonctions dans lesquelles elle passe
- Pour vérifier que j'ai récupéré les bonnes variables

Utilisation pour le front

- Utilisation de `drupal_breakpoint()` avec le module `twig_tweak`
- Permet de voir les variables disponibles (custom ou natives) dans un template ce qui peut être difficile parfois avec le système de templating de Drupal



Montée en compétences : Dump vs. xDebug

```
array:1 [▼  
  "#attributes" => array:1 [▶]  
]
```

```
Drupal\Core\Form\FormState {#524 ▼  
  #complete_form: null  
  #build_info: array:4 [▶]  
  #rebuild_info: []  
  #rebuild: false  
  #invalidToken: false  
  #response: null  
  #ignoreDestination: false  
  #redirect: null  
  #no_redirect: null  
  #method: "POST"  
  #requestMethod: "GET"  
  #cache: false  
  #no_cache: null  
  #values: []  
  #cleanValueKeys: array:4 [▶]  
  #input: []  
  #always_process: null  
  #must_validate: null  
  #programmed: false  
  #programmed_bypass_access_check: true  
  #process_input: null  
  #submitted: false  
  #executed: false  
  #triggering_element: null  
  #has_file_element: null  
  #groups: []  
  #storage: []  
  #buttons: []  
  #temporary: []  
  #validation_complete: false  
  #errors: []  
  #limit_validation_errors: null  
  #validate_handlers: []  
  #submit_handlers: []  
}
```

dump

The screenshot shows the xDebug interface with a stack trace on the left and a variable dump on the right. The stack trace includes files like SearchArtistForm.php, FormBuilder.php, and FormController.php. The variable dump shows the following structure:

```
$this = (Drupal\VinylList\ApiSearch\Form\SearchArtistForm)  
$form_state = (Drupal\Core\Form\FormState)  
$form = (array[1])  
  #attributes = (array[1])  
    class = (array[1])  
      0 = "vinyl-list-api-search-search"  
$_COOKIE = (string[1]) ["PHPSTORM"]  
$_SERVER = (array[47])  
$_SESSION = (array[3])  
Constants
```

xDebug



Montée en compétences : Qualité de code

Drupal Best Practices

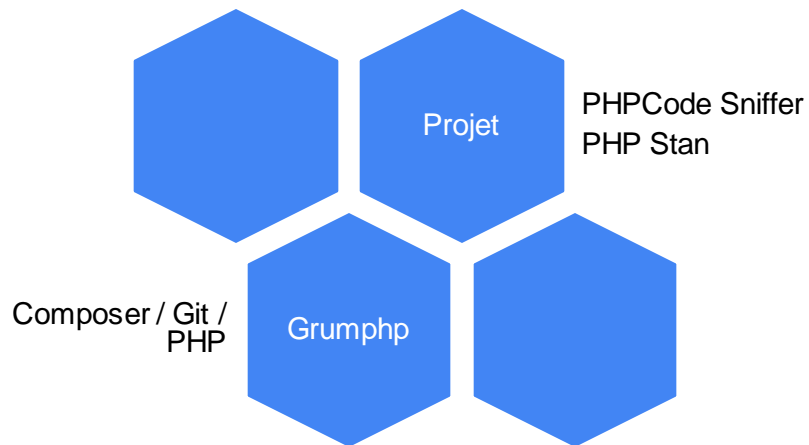
- Coding Standard
- Security

Refactoring

- La règle du boy-scout



Montée en compétences : Pre-commit



```
verweto:grumphp-demo/ (master) $ v [19:39:52]
```



Montée en compétences : Aides en interne

Relecture

- Lors des merge requests

Travail en binôme

- Pair programming
- Aides lors des difficultés
- Réfléchir à la meilleure manière d'aborder un ticket



Montée en compétences : Aides externes

Infos sur Drupal

- Documentation officielle
- Docs sur les APIs
- Drush commands

Communauté Drupal

- Slack
- Meetup Rennes
- Blog

Formations

- Lié à Drupal
- Lié à mon environnement de travail





5. Déploiement de la configuration

En général

- La configuration est intégralement dans la base de code
- Le maintien à jour est fastidieux ("normalement c'est à jour")
- Elle est importée à chaque déploiement sur chaque environnement



Le risque

- Une seule source de vérité : la base de production
- Multiples environnements (config split)
- Multiples contributeurs (changement de conf)
- Gestion des conflits (gérer la conf, merci)
- La livraison :

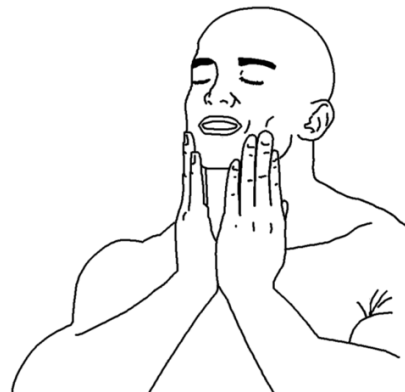




Notre solution

--partial. Allows for partial config imports from the source directory. Only updates and new configs will be processed with this flag (missing configs will not be deleted). No config transformation happens.

- La charge de la synchronisation BDD est reportée sur le développeur
- NB : Il doit exister une façon simple de récupérer la base de production
- Seuls les fichiers de configuration de la feature sont commités
- Test d'import facilité en local (BDD+import)
- Les tickets comportant de la conf sont identifiés (MR lisible)
- Un process automatisé supprime la conf après merge dans master (clean state)



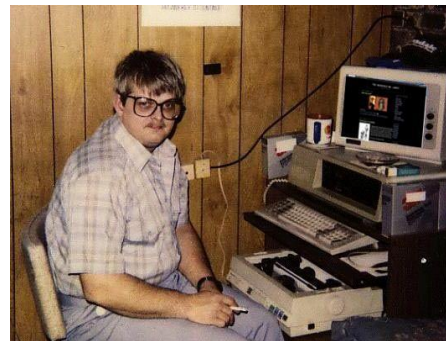


6. Tests & Outils



Words of advice

- Git en ligne de commande (`git add -p`)
- Precommit est un puissant outil de normalisation qui maintient la qualité de code et facilite les relectures et les gestions de conflit (DrupalPractice, PHPStan, PHPMD)
- Tout typer
- Résister à l'inclusion de nouvelles dépendances (modules)
- Interpréter la crainte comme l'indicateur d'un problème à résoudre
- Réviser la documentation d'onboarding à chaque entrant





Behaviour Driven Development

- Méthodologie de développement (et de spécifications)
- Chaque ticket contient les tests d'acceptance
- Le développeur et le testeur partagent cette définition (DOR/DOD)
- La marche subséquente vers l'automatisation n'est pas énorme
- L'essentiel de l'effort est de produire les scénarios de tests. L'outil pour les implémenter est un choix technique.



Tests de supervision

- Tests de supervision embarqués dans le code
- Les tests de supervision font partie du processus de déploiement ordinaire (régressions)
- Modification de code = modification du test par le développeur
- Processus pour déporter les tests vers un outil externe après une MEP réussie
- Maintien de l'observabilité de l'application



Observabilité / Monitoring

- Import des logs (streams) vers une instance ELK
- Normalisation des logs (process/classe/methode/arguments) (Trait)
- Try/catch
- Typage permet de catcher les TypeError
(très bon indicateur de problèmes causés par une MEP)
- Monitoring au sein de l'équipe de développement = adhérence entre les sondes et les indicateurs
- Correctifs de bugs associés à un indicateur (disparition du log)



7. Bilan



SLA >
99.98%

Rapports
dev / métiers
fluidifiés

Agilité dans
la
priorisation

Agilité dans
les livraisons

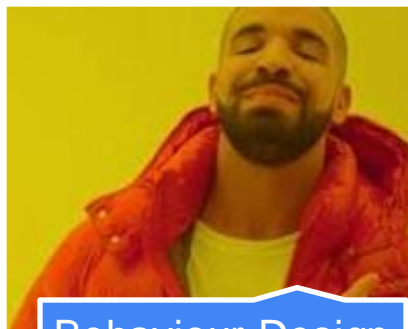
TTM < 1h

Bugs
détectés par
les devs

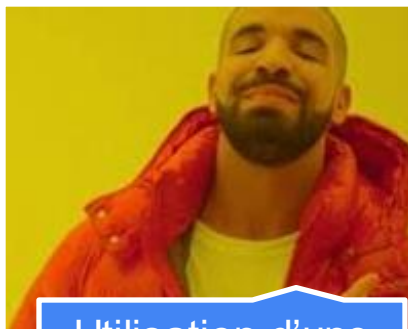




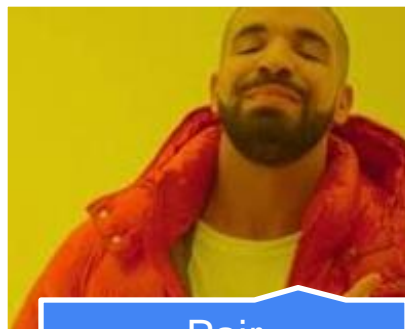
Bilan de l'industrialisation



Behaviour Design
Development



Utilisation d'une
CI / CD



Pair
Programming



BLT



Pour aller plus loin...

Retrouvez ces slides ainsi que des ressources sur le site web de CONCERTO





Merci

*pour votre
écoute*

