ESUP-Days #21

Apereo Europe 2016

Feb. 1st-2nd, 2016
Paris

L'utilisation et l'engagement de l'Université du Québec à Montréal envers le logiciel libre

Emmanuel Vigne
Université du Québec à Montréal

L'UQAM en bref

- Université publique francophone créée en 1969
- 6 facultés et une école
  - École des sciences de la gestion
  - Faculté des arts
  - Facultés de communications
  - Faculté de science politique et de droit
  - Faculté des sciences
  - Faculté des sciences de l'éducation
  - Faculté des sciences humaines
- 43 000 étudiants à l'automne 2015
- 1158 professeurs, 28 maitres de langues, 2303 chargés de cours
- 135 cadres pour 2000 employés

Le Vice-rectorat aux systèmes d'information

- Vice-rectorat créé en juin 2013
- Regroupe depuis juillet 2015 les services informatiques, le service de l'audiovisuel et le service des bibliothèques avec environ 400 employés
- Changement de paradigme en cours
  - Passage d'un niveau de maturité de 1.5 vers 3.0 (CMMI)
  - Architecture d'ensemble réalisée
  - Approche de partage et réalisation sous license libre
  - Orienté étudiant et mobile
  - Changement de culture demandant du temps

Utilisation et promotion de logiciels libres

- Réduire nos dépendances envers les éditeurs de logiciel
- S'éviter les culs de sac technologiques et les mises à jour sous contrainte
  - Ex Windows xp
  - Ex MySql et MariaDB
- Remettre à la communauté tous les développements effectués avec des fonds publics
- Approche similaire adoptée pour le « libre accès »

Promotion de l'utilisation de solutions libres

Logiciels sur les postes de travail des utilisateurs

Promotion de l'utilisation de Libre Office

- 2 types de licences Microsoft
  - "Campus" (location de type « all you can eat », montant fixe selon le nombre d'étudiants et d'employés)
  - "Select" (Achat des produits à la pièce, facturation mensuelle selon les rapports d'installation)
- Licence "Select" gérée centralement par le VRSI
  - Refacturation aux unités des frais de licence
- Séminaires de formation en bureautique
  - Très faible coûts pour les étudiants

Promotion de l'utilisation de Libre Office (2)

- Rédaction en cours d'une politique sur les formats d'échange de documents en interne et de production de documents externes (Ex: Gouvernement britannique)
- Libre Office systématiquement installé sur tous les postes
- Séminaires informatiques dédiés à Libre Office
  - Transition de Ms-Office vers Libre Office (sans frais)
  - Séminaires spécialisés à faible coûts
  - Diminution de l'offre des séminaires Ms-Office
- Questions systématiques lors de demandes d'installation de Ms-Office
  - Installation uniquement dans les cas nécessaires

Autres cas de promotion de solutions libres

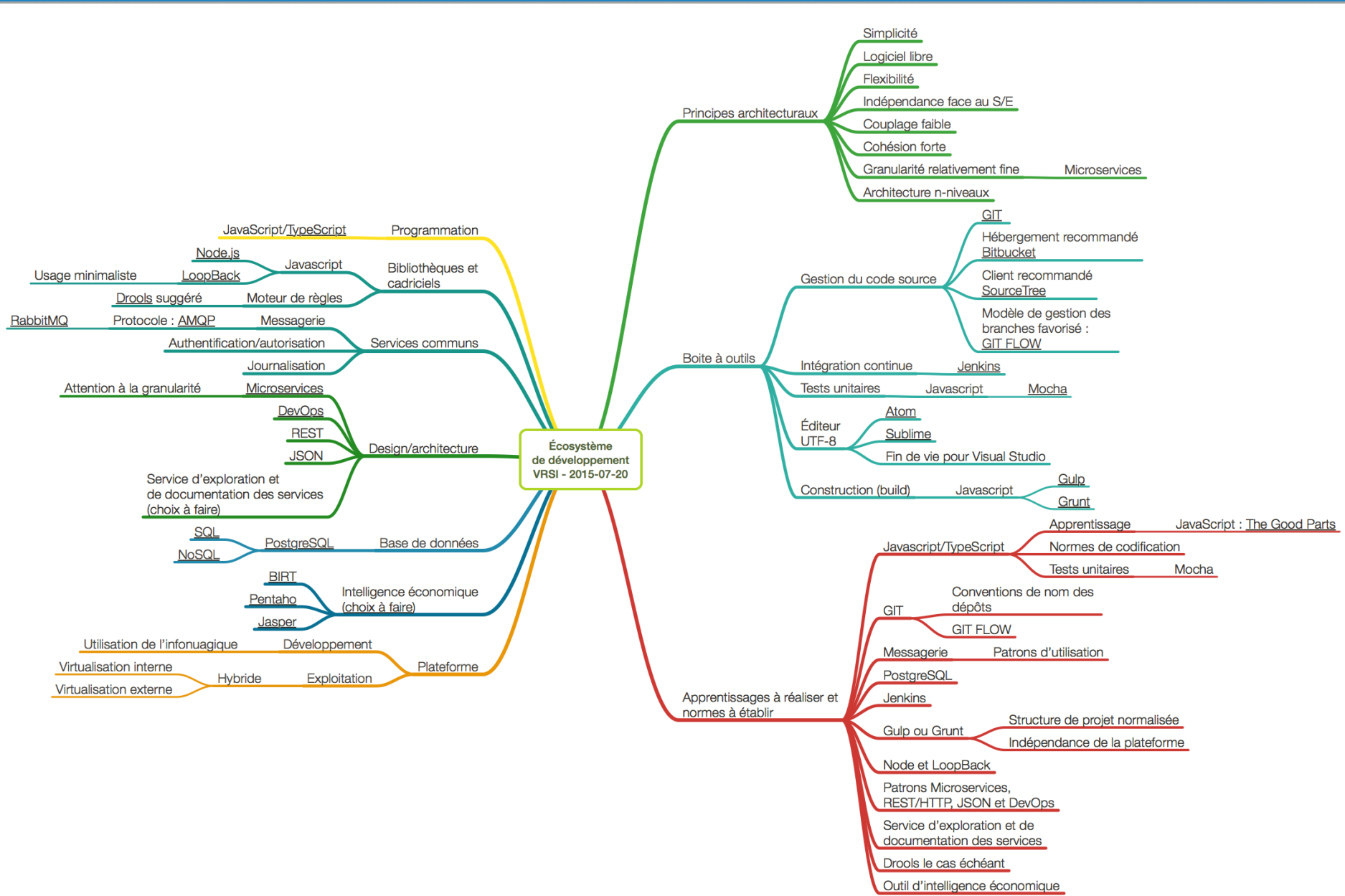
- PSPP et R , stratégie similaire avec formation et recommandations systématiques
- Travail continu avec les chercheurs et les professeurs pour trouver des solutions libres équivalentes aux solutions actuellement utilisées et/ou enseignées
  - Ex Gimp vs Photoshop, Inkscape vs Illustrator

Écosystème de développement

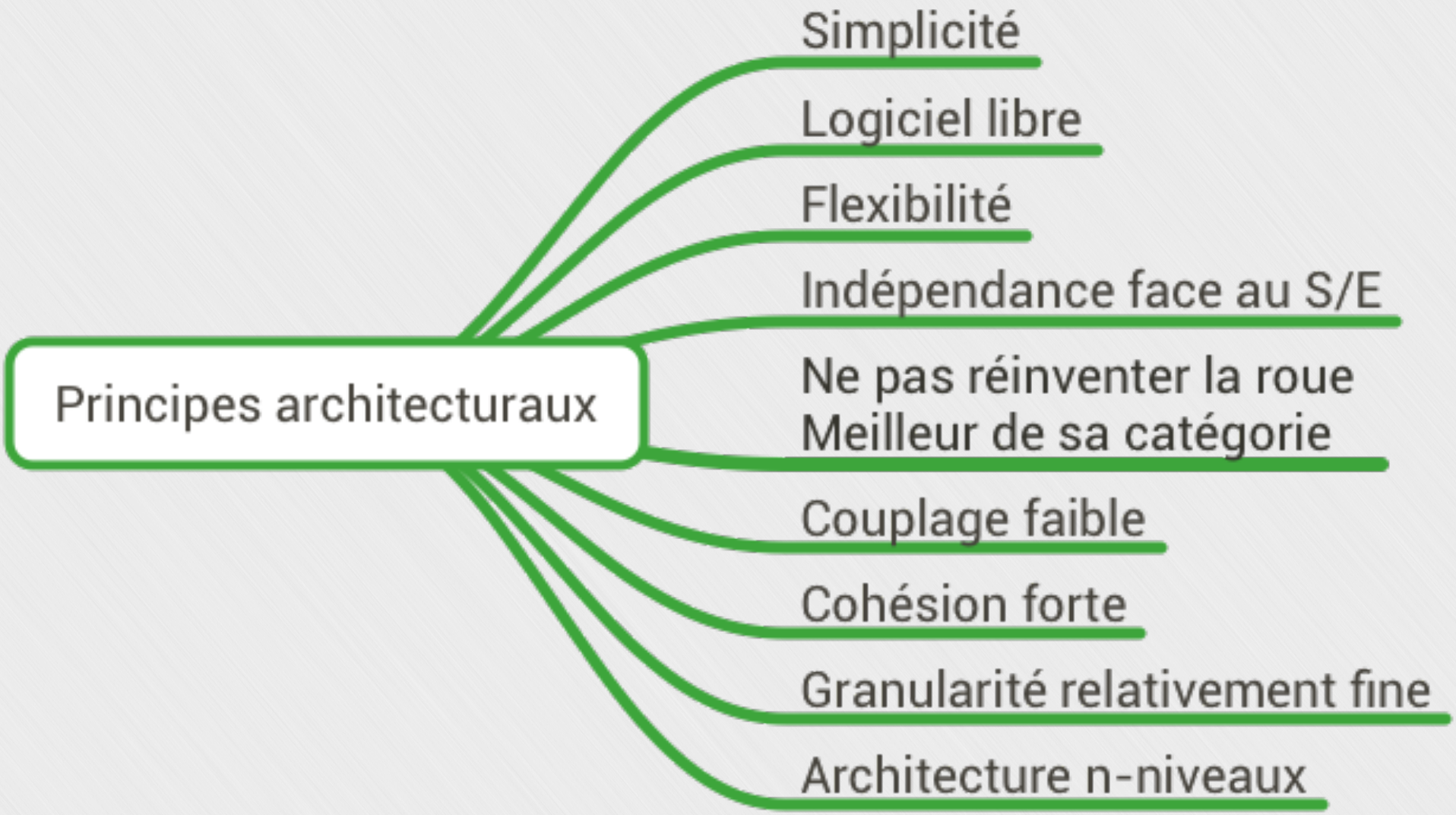
e

Environnements de développement , d'intégration et de déploiement

Écosystème du VRSI

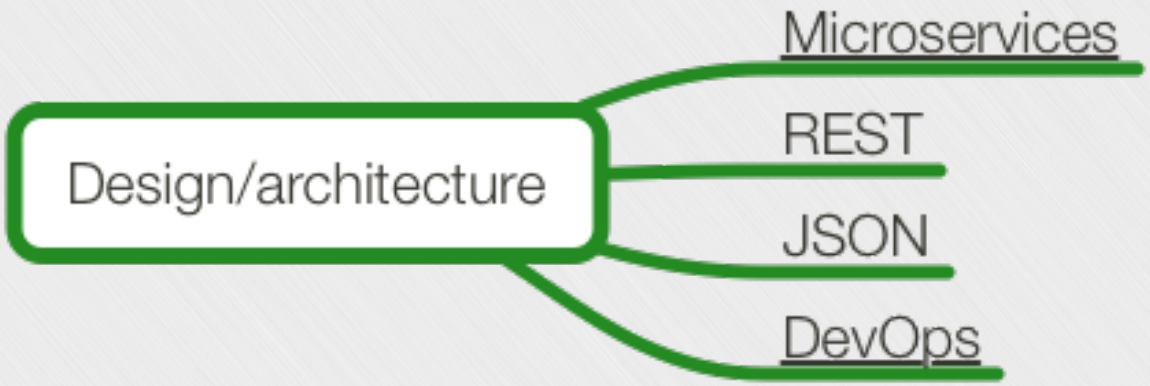


Principes architecturaux

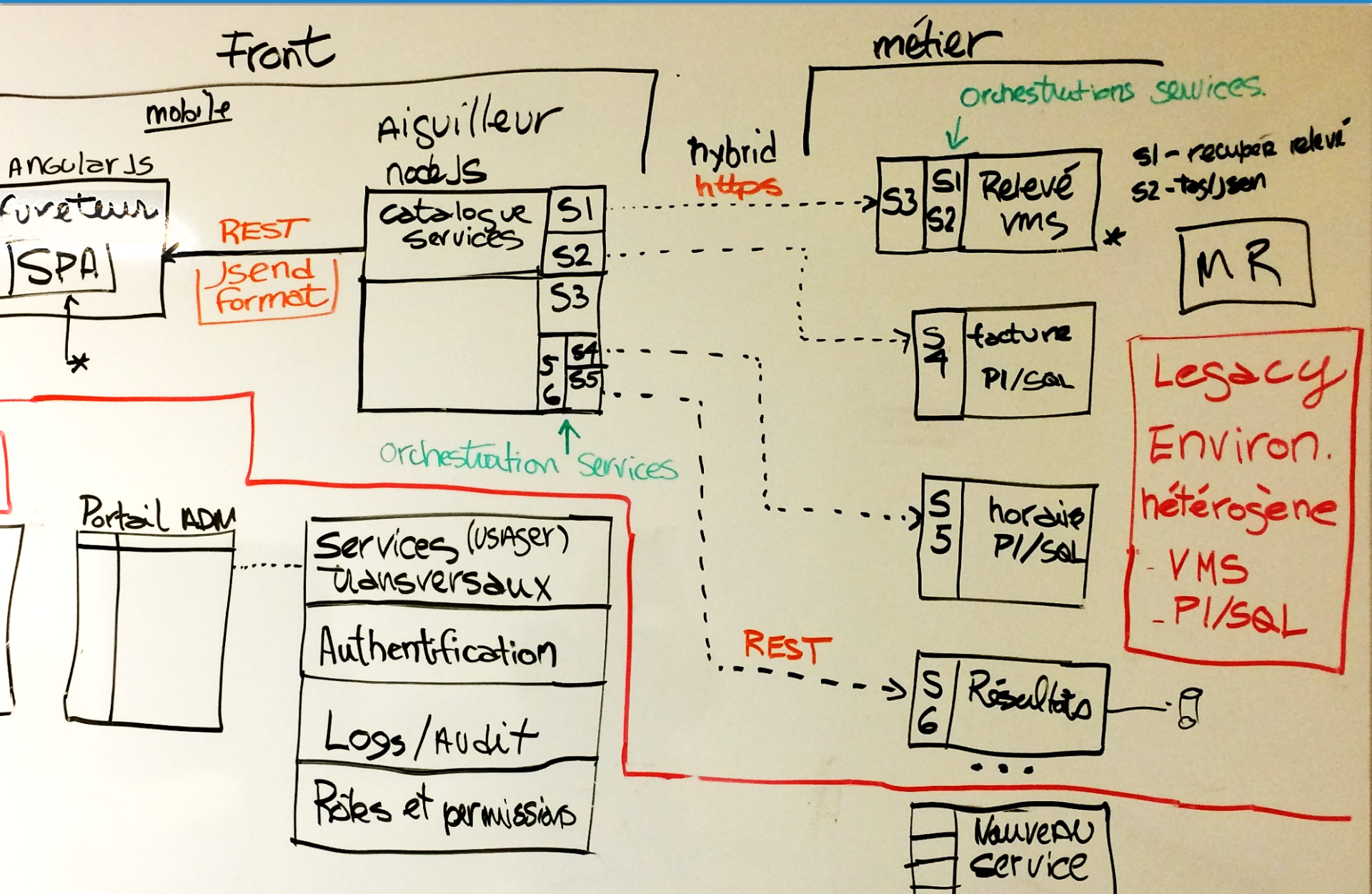


Choix architecturaux

- Application monopage (SPA)
- Microservices
- Orchestration (au niveau de l'aiguilleur)
- Catalogue de services simple
- Amalgame de services (patrimoniaux et nouveaux)
- JavaScript et JSON de bout en bout



Choix architecturaux



Flux de travail. Du 'commit' au déploiement

Écosystème de développement *libreDA*
Flux de travail (basé sur GitFlow)

Dev local

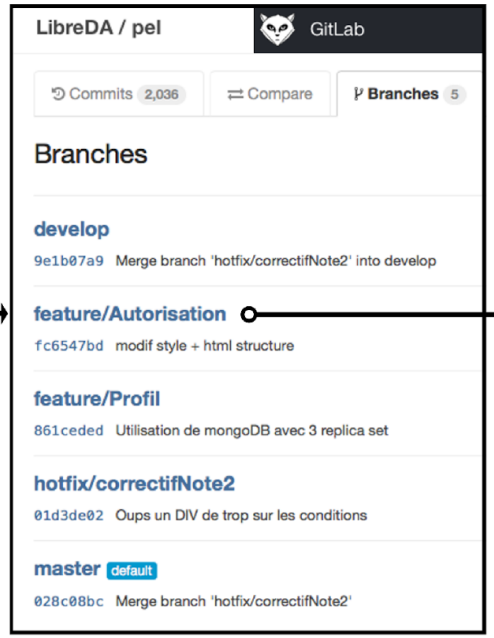
Linux

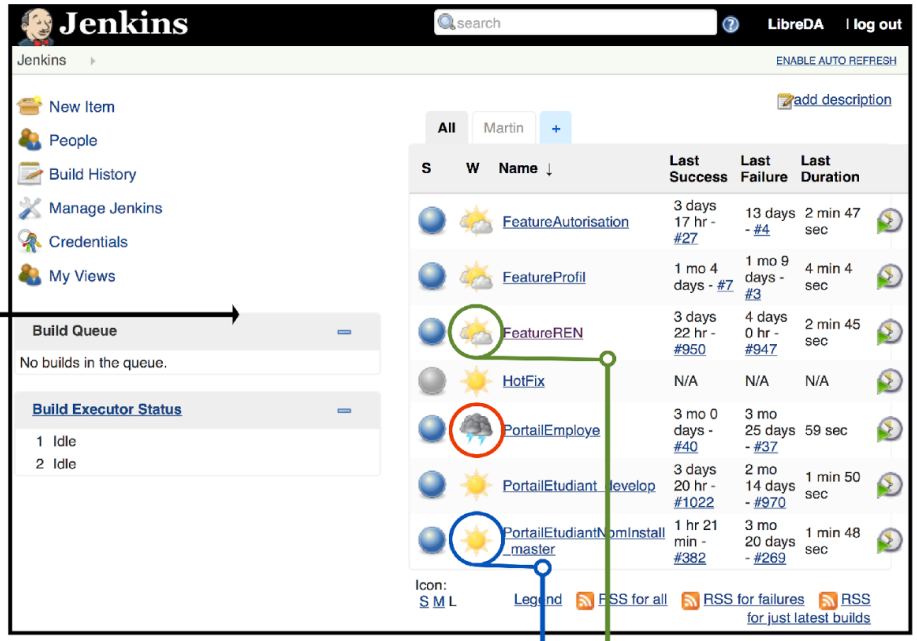
OSX

Windows

YO GRUNT BOWER

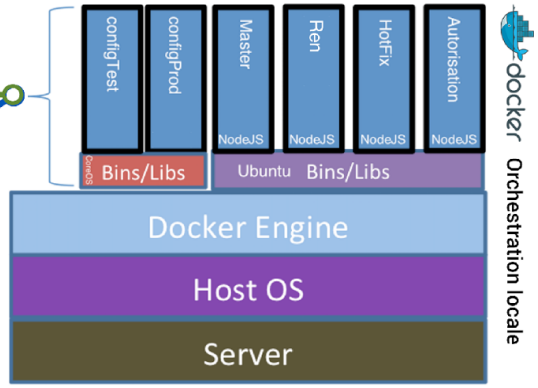
- Architecture applicative;
- Automatismes de développement;
  - maj,
  - tests,
  - linter,
  - couverture,
  - etc.;
- Gestionnaires des ressources et des dépendances.



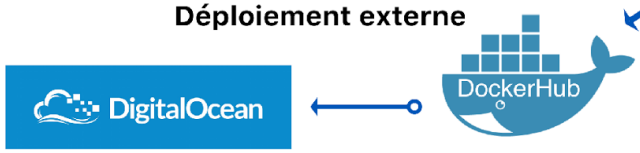


Serveurs de tests UX et fonctionnels

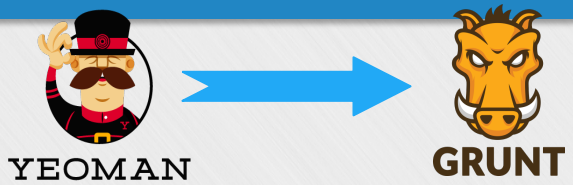
Conteneurs



Déploiement externe



Développement et construction avec Grunt



Gruntfile.js (>1000 lignes)

Développer/Débuguer

```
grunt.registerTask('serve',
  function(target) {
  if (target === 'debug') {
    return grunt.task.run([
      'clean:server',
       ...
    ]);
  }else {
    grunt.task.run([
      'clean:server',
      'injector:sass',
      'concurrent:server',
      'injector',
      'wiredep',
      'autoprefixer',
      'replace:configClient',
      'express:dev',
      'wait',
      'open',
      'watch'
      ]);
  }
});
```

Tests

```
grunt.registerTask('test',
function(target) {
  if (target === 'server') {
    return grunt.task.run([
      'clean:reportserver',
      'mochaTest:xunit',
      'replace:patchMochaXunit',
      'cover:server',
      'jshint:server'
    ]);
  } else if (target === 'client') {
    return grunt.task.run([
      ...
      'injector:sass',
      'concurrent:test',
      'injector',
      'autoprefixer',
      'karma',
      'cover:client',
      'jshint:client'
    ]);
  } else if (target === 'e2e') {
    return grunt.task.run([
      'clean',
      ...
      'protractor'
    ]);
  } else grunt.task.run([
    'test:server',
    'test:client'
  ]);
});
```

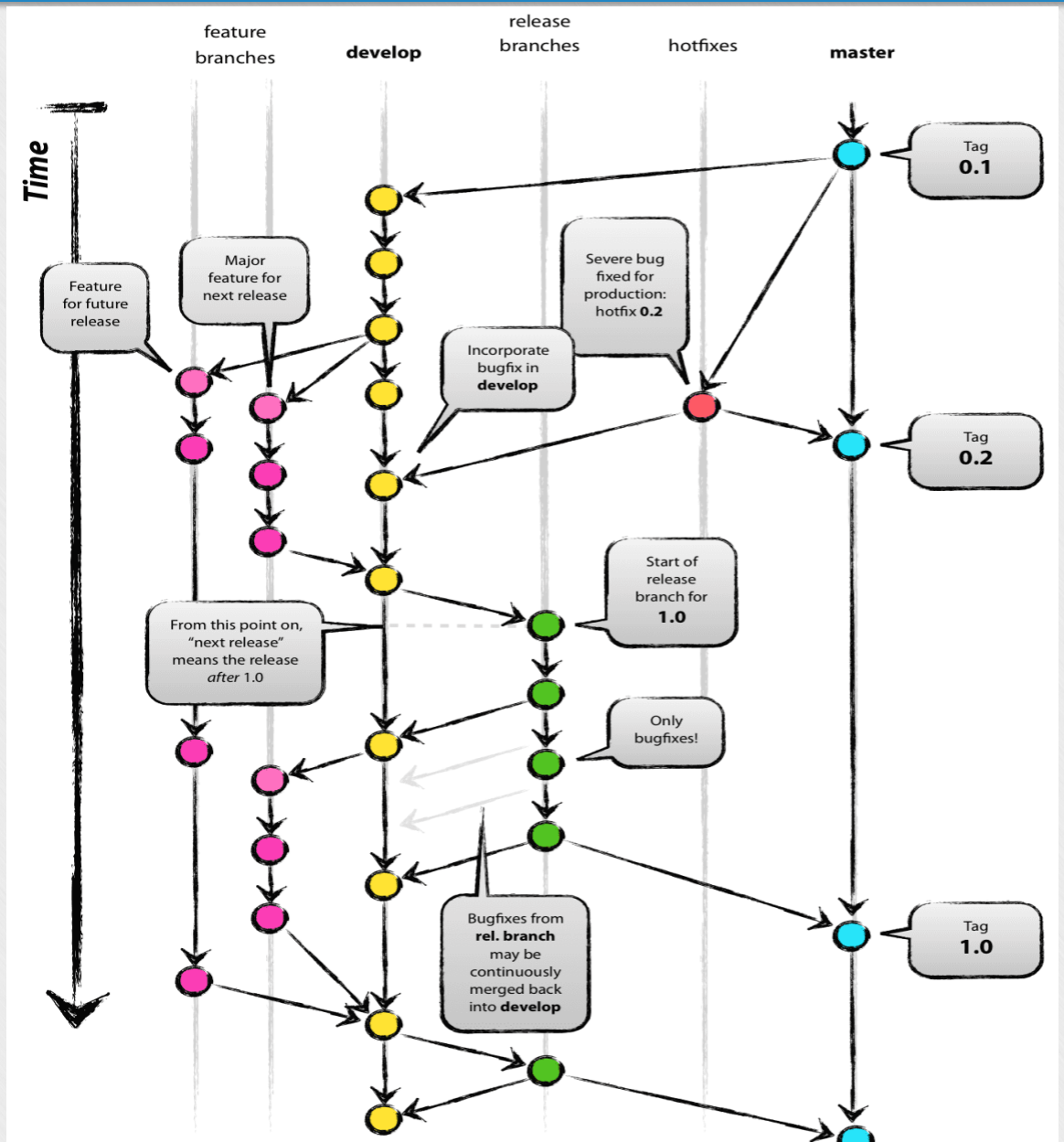
Gestion des versions avec GIT

- Distribué
- Travail hors ligne
- Infonuagique GitLab, GitHub, BitBucket …
- Accessible du fureteur

Utilisation de Git Flow

- Un « Branching Model » qui a fait ses preuves
- Supporte la complexité si nécessaire
- Impose une structure et un flux de travail strict
- Existence de scripts pour simplifier son utilisation
  - SourceTree, HubFlow (GitFlow For GitHub)
  - SmartGit, …
- Associé à notre architecture de commit - déploiement

Git Flow



Intégration avec Jenkins

- Intégration des développements locaux
- Reconstruction complète de l'application après chaque commit
- Effectue les tests
  - Rapports de tests et de couverture
- Permet de savoir qui a fait quoi et quand
- Automatisation du déploiement seulement si tout est OK
- Reflète notre modèle Gitflow
- Paramétrage et validation des configurations de déploiement

Déploiement

- Dépôt de l'application par Jenkins sur un serveur d'orchestration de déploiement
- Construction d'un conteneur applicatif (Docker) à partir de cette application
- Démarrage du conteneur avec l'application (automatique pour les tests, manuel pour la production)
- Testeurs ont accès à Jenkins pour vérifier si les dernières modifications sont en place
- Entre 2 et 3 minutes pour déploiement à jour suite au commit

Vidéo Jenkins

Flux de travail. Du 'commit' au déploiement

Écosystème de développement *libreDA*
Flux de travail (basé sur GitFlow)

Dev local

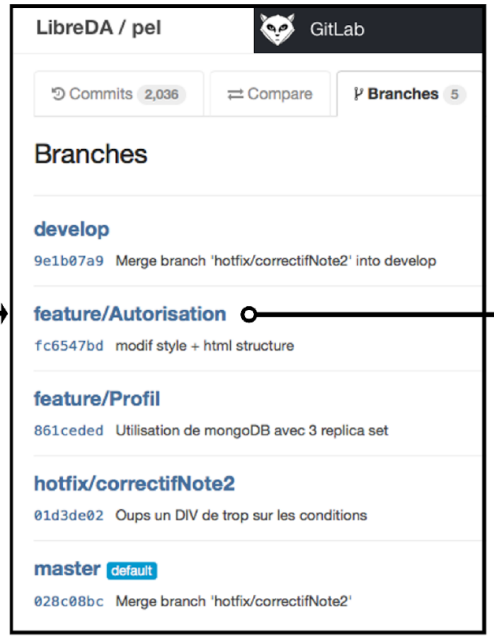
Linux

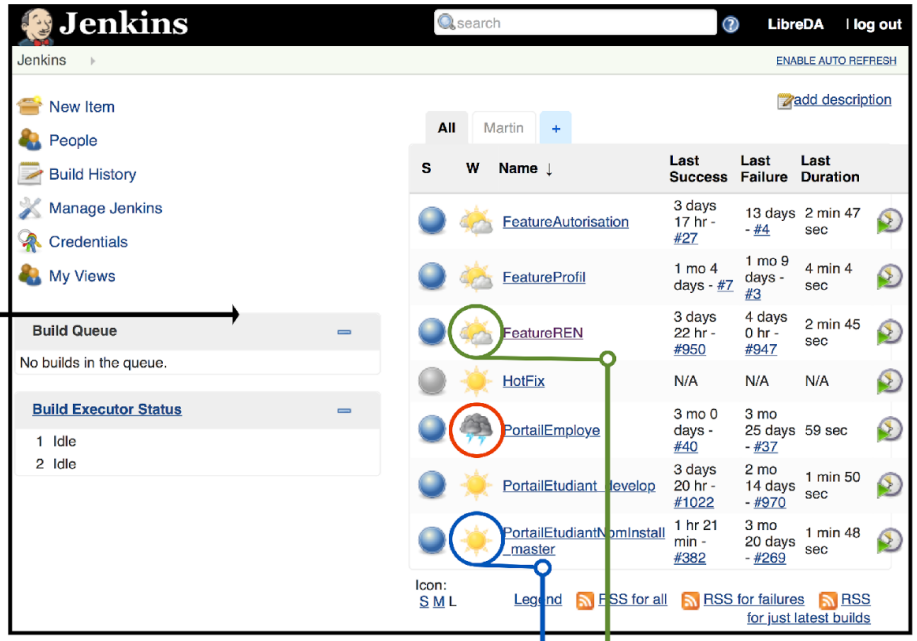
OSX

Windows

YO GRUNT BOWER

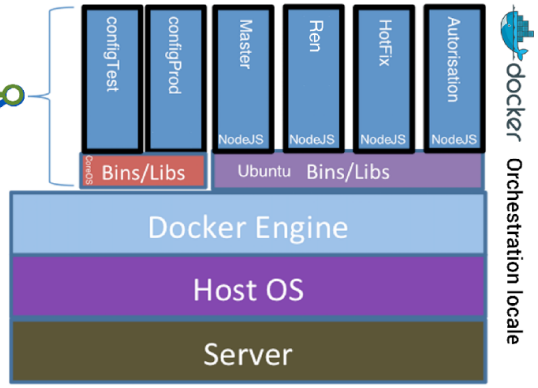
- Architecture applicative;
- Automatismes de développement;
  - maj,
  - tests,
  - linter,
  - couverture,
  - etc.;
- Gestionnaires des ressources et des dépendances.



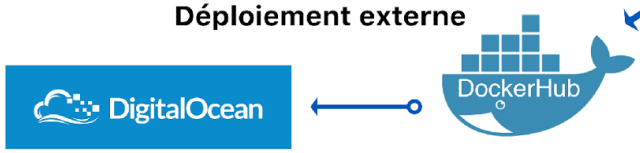


Serveurs de tests UX et fonctionnels

Conteneurs



Déploiement externe



Environnement utilisé pour nouveau portail étudiant

- Projet témoin du projet LibreDA, soit le renouvellement du système d'information et de gestion des études de l'UQAM
- Élaboration et mise en place de nouvelles pratiques de développement
- Améliorer l'expérience étudiante et centraliser l'accès aux services
- Former une équipe à ces nouvelles façons de faire

Vidéo du produit portail