Administration Jira Software

Plan de Formation :

Durée suggérée : 3 jours
Public cible : Administrateurs Jira, Responsables IT, Chefs de projet
Pré-requis : Connaissance de base des outils de gestion de projet et des systèmes informatiques

Module 1 : Introduction à Jira Software et son administration

Durée : 2 heures

- Présentation de Jira Software, Jira Core et Jira Service Desk
- Les types de projets dans Jira
- Architecture et composants de Jira
- Rôles et responsabilités d'un administrateur Jira

Exercice :

Exploration de l'interface d'administration de Jira et navigation dans les menus

Administration Jira Software

Module 2 : Installation et Configuration Initiale

Durée : 3 heures

- Exigences d'installation (Windows/Linux)
- Installation de Jira Server
- Configuration de la base de données (PostgreSQL, MySQL, SQL Server, Oracle)
- Licence et gestion des accès

Exercice :

Installation et mise en place d'un environnement de test

Administration Jira Software

Module 3 : Gestion des Utilisateurs et des Groupes

Durée : 3 heures

- Création, modification et suppression des utilisateurs
- Gestion des groupes et rôles de projet
- Intégration avec LDAP et annuaires externes
- Gestion des permissions et accès

Exercice :

Création d'un jeu d'utilisateurs et gestion des accès à un projet

Administration Jira Software

Module 4 : Configuration des Projets et Workflows

Durée : 4 heures

- Création et configuration d’un projet
- Paramétrage des types de tickets et statuts
- Personnalisation des champs et écrans
- Configuration et gestion des workflows
- Automatisation avec les transitions et validations

Exercice :

Création d’un workflow personnalisé avec des étapes et transitions adaptées

Administration Jira Software

Module 5 : Gestion des Permissions et Sécurité

Durée : 2 heures

- Configuration des permissions globales
- Gestion des permissions des projets
- Gestion de la sécurité des tickets
- Sécurisation de l’accès Jira

Exercice :

Configuration d’un projet avec des niveaux d’accès différents

Administration Jira Software

Module 6 : Configuration Avancée et Personnalisation

Durée : 3 heures

- Création et gestion des filtres et tableaux de bord
- Configuration des notifications et email
- Personnalisation de l'interface (bannières, dashboard, gadgets)

Exercice :

Création d'un tableau de bord personnalisé avec des gadgets utiles

Administration Jira Software

Module 7 : Gestion et Maintenance de Jira

Durée : 3 heures

- Sauvegarde et restauration des données
- Mise à jour et migration de Jira
- Supervision et optimisation des performances
- Audit et journalisation

Exercice :

Simuler une migration de données ou une mise à jour de Jira

Administration Jira Software

Module 8 : Intégration et Automatisation

Durée : 4 heures

- Intégration avec Bitbucket, GitHub, Jenkins
- Gestion des Webhooks et API REST Jira
- Automatisation avec les scripts et outils Atlassian (Scriptrunner, Automation for Jira)

Exercice :

Création d'une règle d'automatisation pour simplifier un processus métier

Administration Jira Software

Module 9 : Support et Bonnes Pratiques

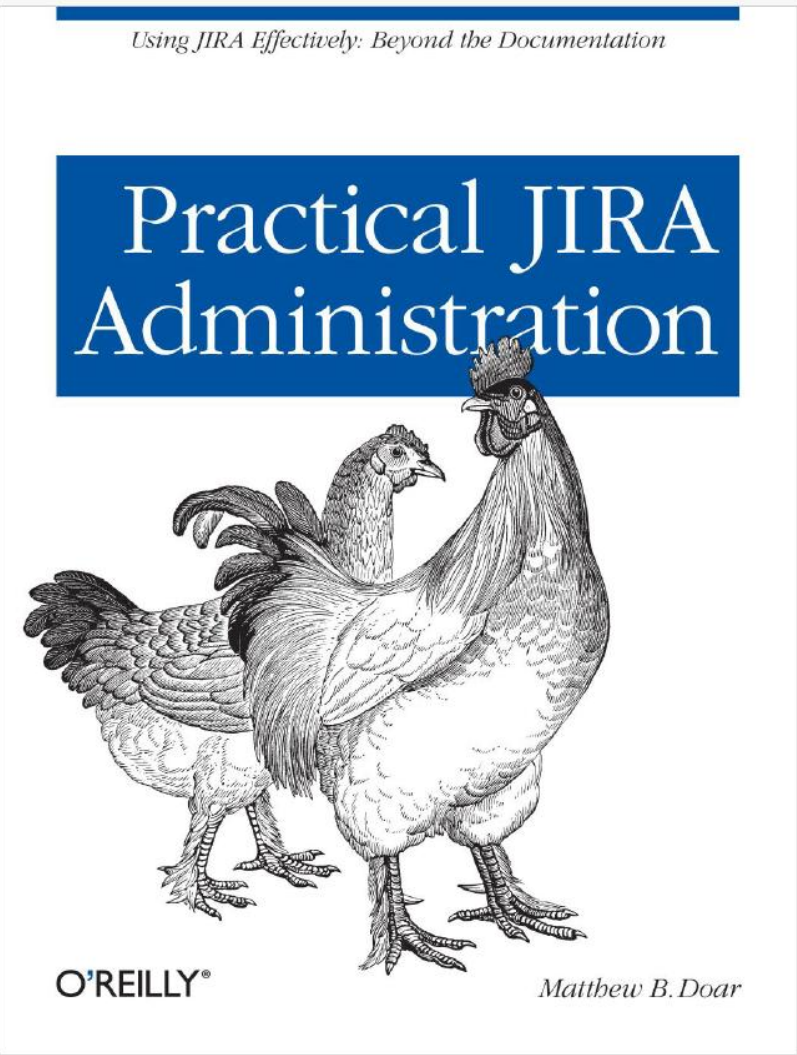
Durée : 2 heures

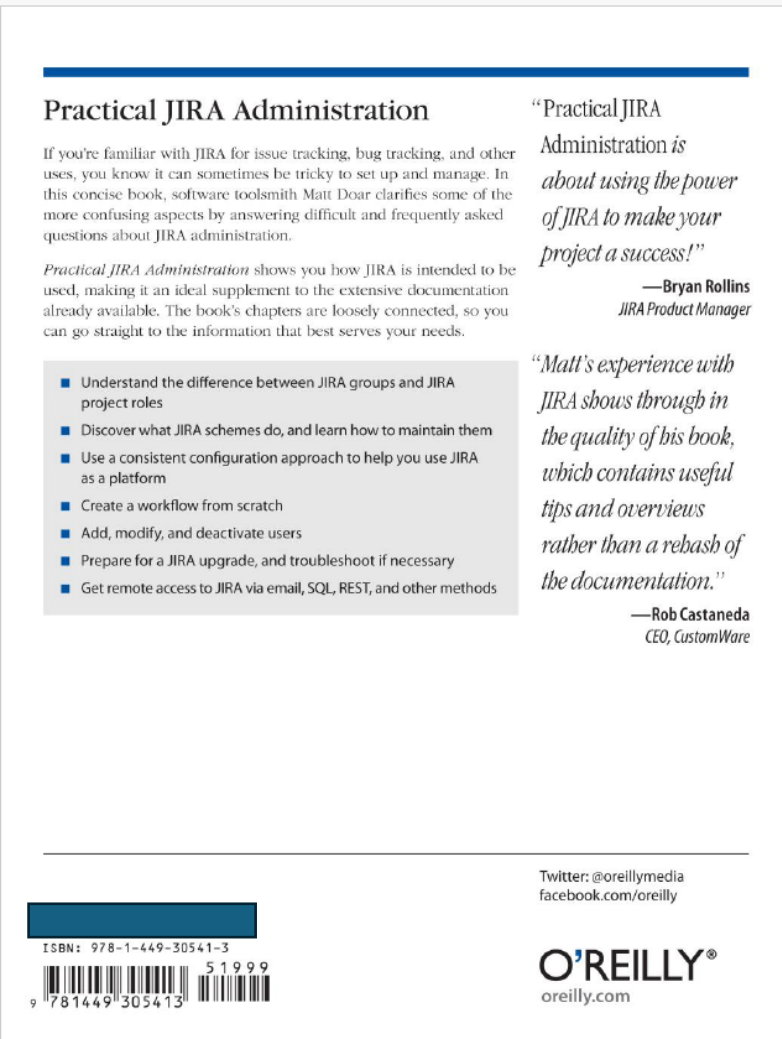
- Dépannage des erreurs courantes
- Bonnes pratiques d’administration Jira
- Ressources et support Atlassian

Exercice :

Étude de cas : Identifier et résoudre des problèmes d’administration courants

Administration Jira Software





Reference : Practical Jira Administration

Module 1 : Introduction à Jira Software et son administration

1. Objectifs du module

Ce premier module vise à fournir une compréhension générale de Jira Software et de ses différentes versions, ainsi que des notions essentielles pour un administrateur Jira.

Durée : 2 heures

2. Introduction à Jira Software, Jira Core et Jira Service Desk

Jira est un outil de gestion de projet et de suivi des tickets développé par Atlassian. Il est principalement utilisé dans les environnements Agile, mais peut être configuré pour divers usages.

3. Les différentes versions de Jira :

- **Jira Core** : Version de base destinée aux entreprises pour la gestion des processus internes.
- **Jira Software** : Version étendue intégrant des outils Agile comme les tableaux Scrum et Kanban.
- **Jira Service Desk** : Version orientée support client, intégrant la gestion des tickets avec des SLA (Service Level Agreements).

Exemple d'utilisation :

- Une équipe de développement Agile utilisera Jira Software pour suivre les sprints.
- Un service informatique utilisera Jira Service Desk pour gérer les demandes des utilisateurs.

2. Les types de projets dans Jira

Un projet dans Jira est un conteneur regroupant les tickets (ou "issues"). Différents types de projets sont disponibles en fonction des besoins.

Principaux types de projets :

- **Projets Business (Jira Core)** : Adaptés aux processus internes (ex : gestion des approbations, suivi des dépenses).
- **Projets Software (Jira Software)** : Incluent des fonctionnalités Agile comme les backlogs, sprints et tableaux Scrum/Kanban.
- **Projets Service Desk (Jira Service Desk)** : Conçus pour gérer les demandes de support avec un portail client.

Différences clés :

- Les **projets Software** offrent des outils Agile.
- Les **projets Service Desk** incluent un système de gestion des demandes et des SLA.
- Les **projets Business** sont plus génériques et adaptés à divers besoins.

3. Architecture et composants de Jira

Jira est une application web reposant sur une architecture client-serveur.

Les principaux composants de Jira :

- **Base de données** : Stocke les tickets, configurations et utilisateurs.
- **Backend** : Partie serveur qui gère la logique de l'application.
- **Interface utilisateur (Frontend)** : Permet aux utilisateurs d'interagir avec Jira via un navigateur web.
- **Plugins et Add-ons** : Extensions permettant d'ajouter des fonctionnalités.
- **API REST** : Permet l'intégration avec d'autres outils (ex : GitHub, Bitbucket, Jenkins).

Comment cela fonctionne ?

1. Un utilisateur crée un ticket via l'interface web.
2. Jira stocke le ticket dans sa base de données.
3. L'administrateur peut configurer des workflows et des automatisations pour traiter le ticket.

4. Rôles et responsabilités d'un administrateur Jira

L'administrateur Jira a un rôle essentiel dans la gestion de la plateforme et l’optimisation des flux de travail.

Les missions principales :

- **Gestion des utilisateurs et permissions** : Création et gestion des comptes, affectation des droits.
- **Configuration des projets** : Paramétrage des types de tickets, workflows, tableaux de bord.
- **Personnalisation et automatisation** : Utilisation des scripts et règles d'automatisation.
- **Maintenance et mises à jour** : Sauvegarde, optimisation et mises à niveau de Jira.
- **Support et formation des utilisateurs** : Assistance et documentation pour les utilisateurs.

Exemple concret :

- Un administrateur doit vérifier que les développeurs ont accès à Jira Software mais pas à Jira Service Desk.
- Il configure les workflows pour automatiser la validation de certaines tâches.

Exercice pratique : Exploration de l'interface d'administration de Jira

Objectif :

Se familiariser avec l'interface de Jira en naviguant dans les différents menus d'administration.

Étapes :

1. **Connexion à Jira** (si possible sur un environnement de test).
2. **Exploration du menu Administration** :
   - Gérer les utilisateurs et groupes
   - Créer un projet test
   - Explorer les workflows et permissions
3. **Personnalisation de l'affichage** : Modification du tableau de bord.
4. **Ajout d'un ticket** et exploration des options disponibles.

Conclusion

Dans ce module, nous avons vu les bases de Jira, ses différentes versions, son architecture et le rôle clé de l'administrateur. Nous avons également exploré l'interface d'administration.

Prochain module : Installation et configuration initiale de Jira.

4. Objectifs du module

Ce module vise à apprendre aux participants comment installer Jira Server, configurer la base de données et gérer les licences et accès. À la fin, les stagiaires seront capables de mettre en place un environnement de test Jira sur leur machine avec des configurations sécurisées.

Durée : 3 heures

1. Exigences d’installation (Windows/Linux)

Avant d’installer Jira Server, il est important de vérifier que le système respecte les prérequis suivants :

Matériel recommandé :

- **Projets légers** (moins de 100 utilisateurs, < 10 000 tickets)
  - 2 CPU, 4 Go de RAM, 10 Go d’espace disque
- **Projets plus importants**
  - 4 CPU, 8 Go de RAM, SSD recommandé

Systèmes d’exploitation supportés :

- **Windows** : Windows Server 2016/2019/2022, Windows 10/11
- **Linux** : Ubuntu, CentOS, Debian, Red Hat

Recommandations de sécurité sur Linux :

- **Configurer les cgroups** pour limiter l'utilisation des ressources par Jira
- **Configurer ulimits** pour éviter les fuites de mémoire et les dépassements de fichiers ouverts
- **Désactiver le swap ou le configurer correctement** pour éviter la dégradation des performances
- **Installer Jira en mode service** pour un démarrage automatique et une meilleure gestion des permissions

Navigateurs compatibles :

- Google Chrome (dernier)
- Mozilla Firefox (dernier)
- Microsoft Edge
- Safari (Mac uniquement)

Prérequis logiciels :

- **Java (JDK 8 ou 11 recommandé)**
- **Base de données** : PostgreSQL, MySQL, SQL Server, Oracle (voir section 3)
- **Ports ouverts** : 8080 (par défaut) pour Jira, 5432 pour PostgreSQL, etc.

2. Installation de Jira Server

2.1 Téléchargement de Jira Server

Aller sur le site Atlassian :
https://www.atlassian.com/software/jira/download

Sélectionner la version **Jira Software Server**.

Télécharger le fichier **.bin** (Linux).

2.2 Installation sur Linux avec recommandations de sécurité

Créer un utilisateur dédié pour Jira (évite d'exécuter Jira en root)

```
sudo adduser --disabled-login --gecos 'JIRA' jira
```

Attribuer les permissions nécessaires

```
sudo usermod -aG sudo jira
```

Configurer les limites ulimits pour Jira (évite les problèmes de fichiers ouverts et mémoire excessive)

```
echo 'jira soft nofile 65536' | sudo tee -a /etc/security/limits.conf
echo 'jira hard nofile 65536' | sudo tee -a /etc/security/limits.conf
echo 'jira soft nproc 4096' | sudo tee -a /etc/security/limits.conf
echo 'jira hard nproc 8192' | sudo tee -a /etc/security/limits.conf
```

Configurer les cgroups pour limiter l'utilisation CPU/RAM de Jira

```
sudo mkdir /sys/fs/cgroup/jira
sudo echo '512M' > /sys/fs/cgroup/jira/memory.limit_in_bytes
sudo echo '1' > /sys/fs/cgroup/jira/cgroup.procs
```

Désactiver le swap pour améliorer les performances

```
sudo swapoff -a
```

Pour rendre cette désactivation permanente, éditer /etc/fstab et commenter les lignes contenant swap.

Installer Jira avec les permissions sécurisées

```
chmod +x atlassian-jira-software-x.x.x-x64.bin
sudo -u jira ./atlassian-jira-software-x.x.x-x64.bin
```

Configurer Jira comme un service systemd pour le démarrage automatique

Créer un fichier /etc/systemd/system/jira.service avec le contenu suivant :

```
[Unit]
Description=JIRA Atlassian Service
After=network.target
[Service]
Type=forking
User=jira
ExecStart=/opt/atlassian/jira/bin/start-jira.sh
ExecStop=/opt/atlassian/jira/bin/stop-jira.sh
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

Activer le service :

```
sudo systemctl daemon-reload
sudo systemctl enable jira
sudo systemctl start jira
```

Vérifier l’état du service :

```
sudo systemctl status jira
```

3. Configuration de la base de données

Exemple :

Configuration PostgreSQL avec recommandations de sécurité

Installer PostgreSQL : sudo apt install postgresql

Créer une base de données pour Jira avec des paramètres de sécurité renforcés :

```
CREATE DATABASE jiradb;
CREATE USER jirauser WITH PASSWORD 'motdepasse_strong!';
ALTER ROLE jirauser SET client_encoding TO 'utf8';
ALTER ROLE jirauser SET default_transaction_isolation TO 'read committed';
ALTER ROLE jirauser SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE jiradb TO jirauser;
```

Activer l'authentification par mots de passe :

Modifier /etc/postgresql/12/main/pg_hba.conf et s'assurer que la ligne suivante est bien configurée :

```
host    jiradb    jirauser    127.0.0.1/32    scram-sha-256
```

Redémarrer PostgreSQL :

```
sudo systemctl restart postgresql
```

Exercice : Installation et mise en place d’un environnement de test sécurisé

Objectif :

Installer Jira Server sur une machine locale en respectant les bonnes pratiques de sécurité.

Étapes :

1. **Créer un utilisateur dédié et configurer les limites système.**
2. **Installer Jira Server et le configurer comme un service.**
3. **Configurer PostgreSQL avec des paramètres sécurisés.**
4. **Vérifier le bon fonctionnement du service Jira.**

Critères de réussite :

- Jira fonctionne correctement et démarre automatiquement.
- La base de données est sécurisée et connectée à Jira.
- Les limites de ressources empêchent les dépassements critiques.

Conclusion

Dans ce module, nous avons vu les prérequis d'installation, les configurations de sécurité avancées et comment rendre Jira plus robuste et performant.

Prochain module : Gestion des utilisateurs et des groupes dans Jira.

5. Objectifs du module

Ce module a pour but d'enseigner aux stagiaires comment gérer les utilisateurs et les groupes dans Jira. À la fin, les participants seront capables d'ajouter des utilisateurs, configurer des groupes et des permissions, et mettre en place une authentification sécurisée.

Durée : 3 heures

6. Introduction à la Gestion des Utilisateurs dans Jira

Jira permet d'administrer les utilisateurs de manière centralisée. Il est possible d'intégrer Jira avec un annuaire LDAP, Active Directory ou de gérer les utilisateurs en interne.

7. Les concepts clés :

- **Utilisateur** : Une personne ayant un compte Jira.
- **Groupe** : Un ensemble d'utilisateurs partageant les mêmes permissions.
- **Rôle de projet** : Un ensemble de permissions assigné à un groupe ou à un utilisateur dans un projet spécifique.

Bonnes pratiques :

- Créer des groupes basés sur les rôles métier (Développeurs, Testeurs, Managers).
- Éviter d'accorder des permissions individuelles, privilégier les groupes.
- Mettre en place une politique de mots de passe robuste.

8. Ajout et Gestion des Utilisateurs

Création d'un utilisateur Jira

1. Se connecter en tant qu'administrateur Jira.
2. Accéder à **Administration > Utilisateurs**.
3. Cliquer sur **Créer un utilisateur**.
4. Remplir les champs requis :
    - Nom d'utilisateur
    - Adresse email
    - Mot de passe temporaire
    - Groupe(s) auquel l'utilisateur appartient
5. Cliquer sur **Créer**.
6. L'utilisateur recevra un email avec ses informations de connexion.

Administration Jira Software

Gestion des utilisateurs

- Modifier un utilisateur : Changer ses groupes, réinitialiser son mot de passe.
- Désactiver un utilisateur : Pour éviter la suppression définitive.
- Supprimer un utilisateur : Action irréversible.

Bonnes pratiques :

- Vérifier que seuls les utilisateurs actifs ont un accès Jira.
- Automatiser la gestion des utilisateurs avec un annuaire LDAP.

Gestion des Groupes et Permissions

Création et gestion des groupes

1. Aller dans **Administration > Groupes**.
2. Cliquer sur **Créer un groupe** et donner un nom pertinent.
3. Ajouter des utilisateurs au groupe.
4. Attribuer des permissions aux groupes via **Schémas de permissions**.

Administration Jira Software

Configuration des permissions

Jira propose des permissions à plusieurs niveaux :

- **Permissions globales** (accès à Jira, administration, gestion des filtres).
- **Permissions de projet** (création de tickets, gestion des workflows).
- **Sécurité des tickets** (limiter l'accès à certains tickets selon les rôles).

Bonnes pratiques :

- Assigner les permissions aux groupes plutôt qu'aux utilisateurs.
- Restreindre les permissions d'administration aux utilisateurs strictement nécessaires.
- Définir des rôles de projet clairs (ex : Scrum Master, Développeur, Testeur).

9. Sécurisation de l'Authentification et de l'Accès

Intégration avec LDAP ou Active Directory

1. Accéder à **Administration > Annuaire des utilisateurs**.
2. Ajouter un **nouveau répertoire LDAP**.
3. Saisir les paramètres de connexion LDAP :
    - Serveur : ldap://ad.example.com
    - Base DN : dc=example,dc=com
    - Filtre utilisateur : (objectClass=user)
4. Tester et enregistrer la configuration.

Configuration de l'authentification SAML et SSO

1. Installer un module SAML/SSO compatible (ex : Atlassian Access).
2. Configurer le fournisseur d'identité (IdP) comme Okta, Azure AD ou Google.
3. Activer l'authentification unique pour les utilisateurs Jira.

Bonnes pratiques :

- Activer l'authentification à deux facteurs (2FA) pour les comptes administrateurs.
- Désactiver les connexions anonymes pour renforcer la sécurité.

Exercice : Création et Gestion d'un Groupe Utilisateur

Objectif : Créer un groupe d'utilisateurs, leur attribuer des permissions et tester la connexion.

Étapes :

1. **Créer un groupe "Développeurs"** dans Jira.
2. **Créer deux utilisateurs** et les ajouter à ce groupe.
3. **Attribuer des permissions** de création et gestion de tickets à ce groupe.
4. **Se connecter avec l'un des utilisateurs** et vérifier les accès.

Critères de réussite :

- Les utilisateurs créés peuvent se connecter.
- Les permissions attribuées sont bien appliquées.
- Un utilisateur en dehors du groupe "Développeurs" ne peut pas accéder aux projets concernés.

Module 4 : Configuration des Projets et Workflows dans Jira

Objectifs du module

Ce module vise à apprendre aux stagiaires comment créer et configurer des projets Jira, définir des workflows adaptés aux besoins des équipes et optimiser la gestion des tickets. À la fin, les participants seront capables de paramétrer des projets et personnaliser des workflows dans Jira.

Durée : 4 heures

1. Introduction aux Projets dans Jira

Un projet dans Jira est un regroupement de tickets et configurations associées. Il définit un cadre pour organiser le travail des équipes.

Types de projets dans Jira :

- **Projets Business (Jira Core)** : Gestion de processus métier sans outils Agile.
- **Projets Software (Jira Software)** : Intègrent des fonctionnalités Agile comme les tableaux Scrum et Kanban.
- **Projets Service Desk (Jira Service Desk)** : Conçus pour la gestion du support client avec des SLA.

Bonnes pratiques :

- Utiliser des **noms de projet clairs et explicites**.
- Configurer des **schémas de permission adaptés** pour éviter les accès non autorisés.
- Définir des **propriétaires de projet** pour assurer une gouvernance claire.

2. Création et Configuration d'un Projet

2.1 Création d'un nouveau projet

1. Aller dans **Administration > Projets**.
2. Cliquer sur **Créer un projet**.
3. Choisir le **type de projet** (Scrum, Kanban, Service Desk, Business).
4. Renseigner les champs requis :
   - Nom du projet
   - Clé du projet (identifiant unique)
   - Modèle de projet
5. Cliquer sur **Créer**.

2.2 Configuration avancée d'un projet

- **Modifier le logo et la description** pour une meilleure identification.
- **Gérer les accès et permissions** pour restreindre les utilisateurs.

- **Personnaliser les champs et écrans** pour adapter les informations affichées.
- **Activer les notifications par email** pour informer les parties prenantes.

Bonnes pratiques :

- **Standardiser les schémas de permissions** pour éviter la complexité.
- **Utiliser des catégories de projet** pour mieux organiser Jira.
- **Archiver les projets obsolètes** pour ne pas surcharger l'instance.

3. Personnalisation des Workflows

Un workflow définit les statuts et transitions d'un ticket.

3.1 Création d'un workflow personnalisé

1. Accéder à **Administration > Workflows**.
2. Cliquer sur **Créer un nouveau workflow**.
3. Ajouter des **statuts personnalisés** (ex : "En validation", "En cours de test").
4. Définir des **transitions** entre statuts (ex : "Valider", "Rejeter").
5. Associer des **règles de transition** (ex : Assigner automatiquement un responsable).
6. Sauvegarder et publier le workflow.

3.2 Utilisation des conditions, validations et post-fonctions

- **Conditions** : Restreignent l'accès à une transition (ex : Seuls les managers peuvent approuver une tâche).
- **Validations** : Vérifient certaines règles avant d'appliquer une transition (ex : Un champ obligatoire doit être rempli).

- **Post-fonctions** : Automatisent des actions après une transition (ex : Envoyer un email de notification).

Bonnes pratiques :

- **Limiter le nombre de statuts** pour garder un workflow clair et lisible.
- **Utiliser des couleurs distinctes** pour différencier les statuts.
- **Tester un workflow sur un projet pilote** avant de le déployer à grande échelle.

4. Sécurisation et Gestion des Permissions de Workflow

Les workflows peuvent être associés à des schémas de permissions pour contrôler qui peut modifier ou faire avancer un ticket.

4.1 Gestion des permissions de transition

1. Accéder à **Administration > Workflows > Édition d'un workflow**.
2. Sélectionner une transition et cliquer sur **Conditions**.
3. Ajouter des restrictions (ex : Seuls les membres du groupe "Développeurs" peuvent passer un ticket en "Résolu").

4.2 Restrictions avancées

- Restreindre la **modification de certains champs** après une transition.
- Empêcher la **réouverture de tickets fermés** sauf par un administrateur.
- Exiger **une approbation managériale** avant de clôturer un ticket.

Bonnes pratiques :

- **Éviter les boucles infinies de transitions** (ex : Un ticket ne peut pas repasser en "Nouveau" une fois en "Terminé").

- **Garder des permissions simples et évolutives**.
- **Assurer la traçabilité** des modifications avec des logs et des notifications.

Exercice : Création d'un Workflow Personnalisé

Objectif :

Créer un workflow adapté à un processus métier spécifique.

Étapes :

1. **Créer un workflow avec 4 statuts** : "À faire", "En cours", "En validation", "Terminé".
2. **Définir les transitions** :
   - "Commencer" (À faire → En cours)
   - "Soumettre pour validation" (En cours → En validation)

- "Valider" (En validation → Terminé)
- "Rejeter" (En validation → En cours)

3. **Configurer une condition** : Seuls les managers peuvent "Valider" un ticket.
4. **Ajouter une post-fonction** : Envoyer un email lors de la validation.
5. **Tester le workflow** sur un projet de test.

Critères de réussite :

- Le workflow fonctionne comme prévu.
- Les permissions et conditions sont bien appliquées.
- L'email de notification est envoyé lors de la validation.

Corrigé :

1. **Accéder à l'administration Jira > Workflows**

2. **Créer un nouveau workflow et ajouter les statuts :**
   - "À faire", "En cours", "En validation", "Terminé"
3. **Configurer les transitions et conditions :**
   - Seuls les managers peuvent effectuer la transition "Valider" vers "Terminé".
   - Ajouter une post-fonction pour envoyer un email lors de la validation.
4. **Associer le workflow au projet de test et vérifier son bon fonctionnement.**
5. **Tester avec un utilisateur manager et un utilisateur standard** pour vérifier les restrictions appliquées.

Conclusion

Dans ce module, nous avons appris à configurer un projet, créer un workflow personnalisé et gérer les permissions associées.

Prochain module : Gestion des Permissions et Sécurité dans Jira.

Module 5 : Gestion des Permissions et Sécurité dans Jira

Objectifs du module

Ce module vise à enseigner aux stagiaires comment configurer les permissions dans Jira pour assurer un contrôle précis des accès et garantir la sécurité des données. À la fin, les participants seront capables de gérer les permissions globales, de projet et d'issue, tout en appliquant les meilleures pratiques de sécurité.

Durée : 3 heures

1. Introduction aux Permissions dans Jira

Jira dispose d'un système de permissions permettant de contrôler qui peut accéder aux projets, créer, modifier ou supprimer des tickets et administrer l'instance.

Types de permissions dans Jira :

- **Permissions globales** : Affectent toute l'instance Jira (ex : administration, création d'utilisateurs).
- **Permissions de projet** : Définissent les actions possibles dans un projet (ex : création de tickets, gestion des sprints).
- **Sécurité des issues** : Restreint l'accès à certains tickets selon des critères spécifiques.

Bonnes pratiques :

- Assigner les permissions aux **groupes** plutôt qu'aux utilisateurs individuels.
- Utiliser **les rôles de projet** pour une gestion flexible des accès.

- Restreindre les permissions d'administration aux utilisateurs strictement nécessaires.

2. Gestion des Permissions Globales

2.1 Configuration des permissions globales

1. Aller dans **Administration > Gestion Globale > Permissions Globales**.
2. Configurer les rôles suivants :
   - **Jira Administrators** : Pleins droits sur l'instance Jira.
   - **Jira System Administrators** : Gère l'infrastructure technique (base de données, intégrations).
   - **Jira Users** : Accès aux fonctionnalités basiques de Jira.
   - **Browse Users** : Permet aux utilisateurs de voir et rechercher d'autres utilisateurs.

Bonnes pratiques :

- Restreindre l'accès à **Jira Administrators** aux seuls administrateurs nécessaires.
- Éviter d'accorder **Jira System Administrators** sauf si indispensable.
- Activer **l'authentification à deux facteurs (2FA)** pour les administrateurs.

3. Gestion des Permissions de Projet

Chaque projet dans Jira peut avoir un schéma de permission spécifique.

3.1 Définition des permissions de projet

1. Accéder à **Administration > Permissions > Schémas de permissions**.

2. Assigner les permissions aux rôles suivants :
    - **Administrateurs de projet** : Gestion complète du projet.
    - **Développeurs** : Gestion et modification des tickets.
    - **Testeurs** : Consultation et validation des tickets.
3. Appliquer un schéma de permission à un projet via **Paramètres du projet > Permissions**.

3.2 Gestion des rôles de projet

- Un **rôle de projet** est un ensemble de permissions pouvant être attribuées à des groupes d'utilisateurs.
- Exemples de rôles courants :
    - **Product Owner** : Peut créer et prioriser les tâches.
    - **Scrum Master** : Peut gérer les sprints et workflows.

- **Développeur** : Peut modifier les tickets mais ne peut pas les supprimer.

Bonnes pratiques :

- Créer des **modèles de permissions réutilisables** pour homogénéiser la gestion des accès.
- Ne donner des droits de suppression qu'aux administrateurs de projet.
- Vérifier régulièrement les permissions attribuées pour éviter l'accumulation d'accès inutiles.

4. Sécurité des Issues et Workflows Sécurisés

4.1 Configuration de la sécurité des issues

1. Aller dans **Administration > Schémas de sécurité des issues**.
2. Créer un schéma et définir des règles d'accès :

- **Visibilité restreinte** : Seuls certains groupes peuvent voir l'issue.
- **Accès basé sur les rôles** : Ex : seuls les testeurs peuvent voir les bugs en validation.

3. Appliquer le schéma de sécurité au projet concerné.

4.2 Restreindre les transitions dans un workflow

- Aller dans **Administration > Workflows**.
- Sélectionner un workflow et modifier une transition.
- Ajouter une **condition** : Ex : "Seuls les administrateurs peuvent clore un ticket".
- Ajouter une **post-fonction** : Ex : "Envoyer un email à l'équipe lorsque le ticket passe en 'Terminé'".

Bonnes pratiques :

- Protéger les **issues sensibles** (ex : tâches liées à la sécurité informatique).
- Éviter que **tous les utilisateurs puissent modifier les statuts des tickets**.
- Activer la **journalisation des actions** pour suivre les modifications sensibles.

Exercice : Configuration d'un Schéma de Permissions Sécurisé

Objectif :

Créer un schéma de permissions et l'appliquer à un projet de test.

Étapes :

1. **Créer un schéma de permissions** avec les rôles suivants :
   - Administrateurs du projet : Pleins droits.

- Développeurs : Création et modification des tickets.
- Testeurs : Accès en lecture seule.

2. **Appliquer ce schéma à un projet existant**.
3. **Créer une issue test et vérifier les restrictions d'accès**.

Critères de réussite :

- Les développeurs ne peuvent pas modifier les permissions du projet.
- Les testeurs ne peuvent pas modifier les issues.
- Les administrateurs du projet ont bien un contrôle total.

Corrigé :

1. **Accéder à Administration > Permissions > Schémas de permissions**.

2. **Créer un nouveau schéma et ajouter les permissions suivantes :**
    - Administrateurs du projet : Pleins droits.
    - Développeurs : Création et modification des tickets.
    - Testeurs : Lecture seule.
3. **Aller dans Administration > Projets et appliquer le schéma au projet de test.**
4. **Créer une issue et tester les permissions avec différents comptes utilisateurs.**
5. **Corriger les éventuelles erreurs de permissions et valider le fonctionnement.**

Conclusion

Dans ce module, nous avons vu comment gérer les permissions globales, de projet et des issues dans Jira tout en appliquant les bonnes pratiques de sécurité.

Prochain module : Configuration avancée et Automatisation dans Jira.

Module 5 : Gestion des Permissions et Sécurité dans Jira

Objectifs du module

Ce module vise à enseigner aux stagiaires comment configurer les permissions dans Jira pour assurer un contrôle précis des accès et garantir la sécurité des données. À la fin, les participants seront capables de gérer les permissions globales, de projet et d'issue, tout en appliquant les meilleures pratiques de sécurité.

Durée : 3 heures

1. Introduction aux Permissions dans Jira

Jira dispose d'un système de permissions permettant de contrôler qui peut accéder aux projets, créer, modifier ou supprimer des tickets et administrer l'instance.

Types de permissions dans Jira :

- **Permissions globales** : Affectent toute l'instance Jira (ex : administration, création d'utilisateurs).
- **Permissions de projet** : Définissent les actions possibles dans un projet (ex : création de tickets, gestion des sprints).
- **Sécurité des issues** : Restreint l'accès à certains tickets selon des critères spécifiques.

Bonnes pratiques :

- Assigner les permissions aux **groupes** plutôt qu'aux utilisateurs individuels.
- Utiliser **les rôles de projet** pour une gestion flexible des accès.
- Restreindre les permissions d'administration aux utilisateurs strictement nécessaires.

2. Gestion des Permissions Globales

2.1 Configuration des permissions globales

1. Aller dans **Administration > Gestion Globale > Permissions Globales**.
2. Configurer les rôles suivants :
   - **Jira Administrators** : Pleins droits sur l'instance Jira.
   - **Jira System Administrators** : Gère l'infrastructure technique (base de données, intégrations).
   - **Jira Users** : Accès aux fonctionnalités basiques de Jira.
   - **Browse Users** : Permet aux utilisateurs de voir et rechercher d'autres utilisateurs.

Bonnes pratiques :

- Restreindre l'accès à **Jira Administrators** aux seuls administrateurs nécessaires.

- Éviter d'accorder **Jira System Administrators** sauf si indispensable.
- Activer **l'authentification à deux facteurs (2FA)** pour les administrateurs.

3. Gestion des Permissions de Projet

Chaque projet dans Jira peut avoir un schéma de permission spécifique.

3.1 Définition des permissions de projet

1. Accéder à **Administration > Permissions > Schémas de permissions**.
2. Assigner les permissions aux rôles suivants :
   - **Administrateurs de projet** : Gestion complète du projet.
   - **Développeurs** : Gestion et modification des tickets.

- **Testeurs** : Consultation et validation des tickets.

3. Appliquer un schéma de permission à un projet via **Paramètres du projet > Permissions**.

3.2 Gestion des rôles de projet

- Un **rôle de projet** est un ensemble de permissions pouvant être attribuées à des groupes d'utilisateurs.
- Exemples de rôles courants :
  - **Product Owner** : Peut créer et prioriser les tâches.
  - **Scrum Master** : Peut gérer les sprints et workflows.
  - **Développeur** : Peut modifier les tickets mais ne peut pas les supprimer.

Bonnes pratiques :

- Créer des **modèles de permissions réutilisables** pour homogénéiser la gestion des accès.

- Ne donner des droits de suppression qu'aux administrateurs de projet.
- Vérifier régulièrement les permissions attribuées pour éviter l'accumulation d'accès inutiles.

4. Sécurité des Issues et Workflows Sécurisés

4.1 Configuration de la sécurité des issues

1. Aller dans **Administration > Schémas de sécurité des issues**.
2. Créer un schéma et définir des règles d'accès :
   - **Visibilité restreinte** : Seuls certains groupes peuvent voir l'issue.
   - **Accès basé sur les rôles** : Ex : seuls les testeurs peuvent voir les bugs en validation.
3. Appliquer le schéma de sécurité au projet concerné.

4.2 Restreindre les transitions dans un workflow

- Aller dans **Administration > Workflows**.
- Sélectionner un workflow et modifier une transition.
- Ajouter une **condition** : Ex : "Seuls les administrateurs peuvent clore un ticket".
- Ajouter une **post-fonction** : Ex : "Envoyer un email à l'équipe lorsque le ticket passe en 'Terminé'".

Bonnes pratiques :

- Protéger les **issues sensibles** (ex : tâches liées à la sécurité informatique).
- Éviter que **tous les utilisateurs puissent modifier les statuts des tickets**.
- Activer la **journalisation des actions** pour suivre les modifications sensibles.

Exercice : Configuration d'un Schéma de Permissions Sécurisé

Objectif :

Créer un schéma de permissions et l'appliquer à un projet de test.

Étapes :

1. **Créer un schéma de permissions** avec les rôles suivants :
   - Administrateurs du projet : Pleins droits.
   - Développeurs : Création et modification des tickets.
   - Testeurs : Accès en lecture seule.
2. **Appliquer ce schéma à un projet existant**.
3. **Créer une issue test et vérifier les restrictions d'accès**.

Critères de réussite :

- Les développeurs ne peuvent pas modifier les permissions du projet.
- Les testeurs ne peuvent pas modifier les issues.
- Les administrateurs du projet ont bien un contrôle total.

Corrigé :

1. **Accéder à Administration > Permissions > Schémas de permissions**.
2. **Créer un nouveau schéma et ajouter les permissions suivantes :**
   - Administrateurs du projet : Pleins droits.
   - Développeurs : Création et modification des tickets.
   - Testeurs : Lecture seule.
3. **Aller dans Administration > Projets et appliquer le schéma au projet de test.**

4. **Créer une issue et tester les permissions avec différents comptes utilisateurs.**
5. **Corriger les éventuelles erreurs de permissions et valider le fonctionnement.**

Conclusion

Dans ce module, nous avons vu comment gérer les permissions globales, de projet et des issues dans Jira tout en appliquant les bonnes pratiques de sécurité.

Prochain module : Configuration avancée et Automatisation dans Jira.

Module 6 : Configuration Avancée et Personnalisation de Jira

Objectifs du module

Ce module vise à approfondir la personnalisation de Jira en configurant les filtres, tableaux de bord, notifications et l'interface utilisateur. À la fin, les participants seront capables de configurer un environnement Jira optimisé et adapté aux besoins spécifiques des équipes.

Durée : 3 heures

1. Création et Gestion des Filtres dans Jira

Les filtres permettent d'afficher des ensembles spécifiques de tickets en fonction de critères définis par l'utilisateur.

1.1 Création d'un filtre personnalisé

1. Aller dans **Rechercher des tickets > Recherche avancée**.

2. Utiliser la syntaxe JQL (Jira Query Language) pour définir des critères.
    - Exemple : Afficher les tickets ouverts assignés à l'utilisateur actuel :

```
assignee = currentUser() AND status != Terminé ORDER BY priority DESC
```

3. Cliquer sur **Enregistrer sous filtre** et donner un nom pertinent.
4. Partager le filtre avec une équipe si nécessaire.

1.2 Automatisation avec les abonnements aux filtres

- Un utilisateur peut s'abonner à un filtre pour recevoir des notifications régulières sur les nouvelles issues correspondant aux critères.
- Configurer un abonnement dans **Gérer les filtres > Abonnements**.

Bonnes pratiques :

- Nommer les filtres de manière explicite.
- Partager les filtres uniquement avec les groupes concernés.
- Utiliser les filtres enregistrés pour alimenter des tableaux de bord dynamiques.

2. Personnalisation des Tableaux de Bord

Un tableau de bord permet de visualiser en un coup d'œil les données Jira pertinentes pour un utilisateur ou une équipe.

2.1 Création d'un tableau de bord

1. Aller dans **Tableaux de bord > Créer un tableau de bord**.
2. Ajouter un titre et une description.
3. Configurer les permissions de visualisation et d'édition.

4. Ajouter des gadgets pertinents (ex : "Liste des tickets assignés", "Graphique des tickets par statut").
5. Sauvegarder et partager avec l'équipe.

Bonnes pratiques :

- Adapter les gadgets au public cible (managers, développeurs, testeurs).
- Ne pas surcharger un tableau de bord avec trop d'informations.
- Utiliser des graphiques pour une meilleure lisibilité.

3. Configuration des Notifications et Emails

Les notifications permettent d'informer les utilisateurs des changements effectués sur les tickets qui les concernent.

3.1 Configuration du schéma de notifications

1. Aller dans **Administration > Systèmes > Notifications**.
2. Sélectionner un schéma de notification et l'associer à un projet.
3. Définir quels événements déclenchent une notification (ex : Création d'un ticket, modification du statut, ajout d'un commentaire).
4. Ajouter des destinataires (utilisateurs spécifiques, groupes, rôles de projet).
5. Tester les notifications avec un ticket de test.

Bonnes pratiques :

- Éviter l'excès de notifications pour ne pas surcharger les emails des utilisateurs.
- Personnaliser le contenu des emails pour plus de clarté.
- Utiliser des notifications Slack ou Teams pour les mises à jour importantes.

4. Personnalisation de l’Interface et de l’Expérience Utilisateur

4.1 Modification des champs et écrans

- Ajouter ou supprimer des champs via **Administration > Champs personnalisés**.
- Configurer les écrans de création et d’édition des tickets via **Schémas d’écrans**.
- Utiliser les champs obligatoires pour garantir une saisie de données cohérente.

4.2 Personnalisation de l’apparence de Jira

- Modifier le logo et les couleurs via **Administration > Apparence**.
- Personnaliser la page d’accueil avec des annonces ou des messages aux utilisateurs.
- Définir des bannières d’avertissement pour informer les utilisateurs de changements majeurs.

Bonnes pratiques :

- Éviter de trop personnaliser pour conserver une interface intuitive.
- Mettre à jour les annonces globales en cas de maintenance ou de mises à jour.
- Tester les changements d'interface avec un groupe restreint avant un déploiement global.

Exercice : Création d'un Tableau de Bord Dynamique

Objectif :

Créer un tableau de bord Jira qui regroupe les tickets en cours, les tickets urgents et un graphique des tendances d'assignation.

Étapes :

1. **Créer un filtre pour afficher les tickets non résolus** :

status != Terminé ORDER BY priority DESC

2. **Créer un tableau de bord Jira** et ajouter les gadgets suivants :
   - "Liste des tickets assignés à moi"
   - "Graphique des tickets par statut"
   - "Tickets en retard ou bloqués"
3. **Enregistrer et partager** le tableau de bord avec l'équipe.
4. **Tester les résultats** en simulant des changements de statut.

Critères de réussite :

- Le tableau de bord affiche bien les données en temps réel.
- Les filtres sont correctement appliqués.
- Les utilisateurs voient uniquement les informations qui les concernent.

Corrigé :

1. **Aller dans Tableaux de bord > Créer un tableau de bord**.
2. **Ajouter des gadgets** et configurer leurs paramètres :
   - **"Liste des tickets assignés"** → Appliquer le filtre "Tickets ouverts".
   - **"Graphique des tickets par statut"** → Configurer pour afficher les tickets par priorité.
   - **"Tickets en retard ou bloqués"** → Filtrer par date d'échéance dépassée.
3. **Tester avec un compte utilisateur standard** pour vérifier la visibilité.
4. **Partager le tableau de bord avec les rôles pertinents**.

Conclusion

Dans ce module, nous avons vu comment personnaliser Jira en utilisant les filtres, les tableaux de bord, les notifications et l'interface utilisateur pour améliorer l'expérience des utilisateurs.

Prochain module : Gestion et Maintenance de Jira.

Module 7 : Gestion et Maintenance de Jira

Objectifs du module

Ce module vise à enseigner aux stagiaires comment assurer la maintenance, la supervision et l'optimisation des performances de Jira. À la fin, les participants seront capables d'effectuer des sauvegardes, des mises à jour et de surveiller l'état de leur instance Jira.

Durée : 3 heures

1. Sauvegarde et Restauration des Données

Les sauvegardes régulières sont essentielles pour éviter la perte de données en cas de panne ou d'erreur humaine.

1.1 Configuration d'une sauvegarde automatique

1. Accéder à **Administration > Système > Sauvegardes**.
2. Activer la sauvegarde automatique.
3. Définir une fréquence (quotidienne, hebdomadaire, mensuelle).
4. Choisir un emplacement sécurisé pour stocker les fichiers.

1.2 Sauvegarde manuelle

- Via l'interface Jira :
    1. Accéder à **Administration > Sauvegarde système**.
    2. Générer un fichier ZIP contenant la base de données et les configurations.

- Via la base de données :

pg_dump -U jirauser -h localhost -F c -b -v -f /backup/jiradb.backup jiradb

1.3 Restauration d'une sauvegarde

- Charger une sauvegarde via l'interface **Administration > Restauration**.
- Restaurer une base de données PostgreSQL :

pg_restore -U jirauser -d jiradb -v /backup/jiradb.backup

Bonnes pratiques :

- Stocker les sauvegardes sur un serveur sécurisé.
- Tester régulièrement la restauration pour s'assurer qu'elle fonctionne.
- Mettre en place un plan de reprise d'activité (PRA) en cas de panne.

2. Mise à Jour et Migration de Jira

2.1 Mise à jour de Jira

1. Vérifier la compatibilité de la nouvelle version sur Atlassian.
2. Effectuer une sauvegarde complète avant la mise à jour.
3. Télécharger la dernière version de Jira.
4. Lancer la mise à jour :

```
sudo ./atlassian-jira-software-x.x.x-x64.bin
```

5. Vérifier l'intégrité du système après l'installation.

2.2 Migration vers un autre serveur

- Exporter les données via **Sauvegarde système**.
- Installer Jira sur le nouveau serveur.
- Importer les données et vérifier l'intégrité des configurations.

Bonnes pratiques :

- Toujours tester une mise à jour sur un environnement de test.
- Lire les notes de version pour anticiper les changements.
- Planifier les mises à jour en dehors des heures de production.

3. Supervision et Optimisation des Performances

3.1 Surveillance des logs et erreurs

- Accéder aux logs via **Administration > Logs système**.
- Vérifier les fichiers de logs :

tail -f /var/atlassian/application-data/jira/log/atlassian-jira.log

- Rechercher les erreurs fréquentes : problèmes de base de données, erreurs de permission.

3.2 Optimisation des performances

- **Augmenter la mémoire allouée à Jira** :
  - Modifier le fichier setenv.sh :
  - export JVM_MINIMUM_MEMORY=2g

export JVM_MAXIMUM_MEMORY=4g

- **Optimiser la base de données PostgreSQL** :

VACUUM ANALYZE;

- **Réduire les temps de réponse** :
  - Désactiver les plugins inutilisés.
  - Activer la mise en cache des requêtes fréquentes.

Bonnes pratiques :

- Automatiser la surveillance avec un outil comme **Prometheus** ou **Grafana**.
- Vérifier les performances avec **Jira Data Center Health Check**.

- Planifier des audits réguliers pour identifier les goulets d'étranglement.

4. Audit et Journalisation des Actions

4.1 Configuration des audits dans Jira

1. Aller dans **Administration > Sécurité > Journaux d'audit**.
2. Activer la journalisation avancée.
3. Définir la durée de rétention des logs.
4. Exporter les journaux pour analyse.

4.2 Surveillance des accès et permissions

- Vérifier les connexions utilisateur pour identifier les accès non autorisés.
- Auditer régulièrement les permissions attribuées.

- Activer l'authentification à deux facteurs (2FA) pour renforcer la sécurité.

Bonnes pratiques :

- Automatiser l'export des logs pour un suivi continu.
- Analyser les tentatives de connexion échouées.
- Restreindre les accès aux données sensibles.

Exercice : Mise en place d'un Plan de Maintenance

Objectif :

Créer un plan de maintenance préventive pour Jira, incluant la sauvegarde, la supervision et l'optimisation des performances.

Étapes :

1. **Configurer une tâche de sauvegarde automatique** avec un script :

```
echo "pg_dump -U jirauser -h localhost -F c -b -v -f
/backup/jiradb.backup jiradb" > /etc/cron.daily/jira_backup
```

2. **Superviser les performances avec un outil de monitoring** (ex : Grafana, Zabbix).
3. **Analyser les logs d'erreur** et corriger un problème détecté.
4. **Planifier une mise à jour test** sur un serveur hors production.

Critères de réussite :

- La sauvegarde est planifiée et testée avec succès.
- Les logs sont analysés et les erreurs identifiées.
- Une simulation de mise à jour a été réalisée sans incident.

Corrigé :

1. **Créer un script de sauvegarde automatique** et le tester.
2. **Installer un outil de monitoring et configurer une alerte pour les erreurs critiques.**
3. **Extraire les logs et identifier une erreur fréquente (ex : requête SQL lente).**
4. **Planifier une mise à jour test et vérifier la compatibilité des plugins.**

Conclusion

Dans ce module, nous avons appris à assurer la maintenance de Jira en mettant en place des sauvegardes, des mises à jour et une supervision efficace pour garantir des performances optimales.

Prochain module : Intégration et Automatisation dans Jira.

Module 8 : Intégration et Automatisation dans Jira

Objectifs du module

Ce module vise à enseigner aux stagiaires comment intégrer Jira avec d'autres outils et automatiser les processus via les API et les règles d'automatisation. À la fin, les participants seront capables de connecter Jira à des outils externes, configurer des webhooks et utiliser des scripts pour optimiser leur gestion des tickets.

Durée : 4 heures

1. Intégration de Jira avec d'autres Outils

1.1 Connexion avec Bitbucket, GitHub et GitLab

Jira peut être intégré aux outils de gestion de code source pour suivre les commits, branches et pull requests directement dans les tickets.

Exemple : Intégration avec Bitbucket

1. Aller dans **Administration > Applications > Intégrations Bitbucket**.
2. Cliquer sur **Ajouter une nouvelle connexion**.
3. Saisir l'URL de Bitbucket et générer un token API.
4. Associer les projets Jira aux dépôts Bitbucket.

Bonnes pratiques :

- Associer les commits aux tickets Jira avec des messages comme :

JIRA-123: Correction du bug d'affichage

- Utiliser des branches liées aux tickets (feature/JIRA-456).
- Automatiser la transition des tickets lors de la fusion des branches.

1.2 Intégration avec Jenkins et CI/CD

1. Installer le plugin **Jira Plugin** sur Jenkins.
2. Ajouter une tâche post-build dans Jenkins pour mettre à jour Jira :

```
curl -X POST -u user:token \
-H "Content-Type: application/json" \
--data '{"transition": {"id": "21"}}' \
https://jira.example.com/rest/api/2/issue/JIRA-789/transitions
```

3. Vérifier que les builds réussis déclenchent une mise à jour automatique du ticket.

Bonnes pratiques :

- Lier les pipelines CI/CD aux statuts des tickets.
- Automatiser la fermeture d'un ticket après un déploiement réussi.

2. Gestion des Webhooks et API Jira

2.1 Création et Utilisation des Webhooks

Les **webhooks** permettent d'envoyer des notifications lorsqu'un événement se produit dans Jira (ex : mise à jour de ticket).

Exemple : Création d'un webhook

1. Aller dans **Administration > Webhooks**.
2. Cliquer sur **Créer un webhook**.
3. Définir un déclencheur (ex : "Mise à jour de ticket").
4. Saisir l'URL du serveur recevant l'événement.

Bonnes pratiques :

- Utiliser des webhooks pour synchroniser Jira avec Slack, Teams ou d'autres outils internes.
- Éviter d'envoyer des notifications sur des changements mineurs.

2.2 Utilisation de l'API REST Jira

L'API REST de Jira permet d'automatiser la gestion des tickets et d'interagir avec d'autres systèmes.

Exemple : Création d'un ticket via API

```
curl -X POST -u user:token \
-H "Content-Type: application/json" \
--data '{
  "fields": {
    "project": {"key": "TEST"},
    "summary": "Problème détecté en production",
    "description": "Détails de l'incident",
    "issuetype": {"name": "Bug"}
  }
}' \
https://jira.example.com/rest/api/2/issue
```

Bonnes pratiques :

- Utiliser les tokens API au lieu des mots de passe.
- Limiter les accès API aux rôles ayant besoin d'automatisation.
- Monitorer les appels API pour éviter une surcharge de requêtes.

3. Automatisation avec Jira Automation

Jira propose une interface pour créer des règles automatisées sans coder.

3.1 Création d'une règle d'automatisation

1. Aller dans **Administration > Automatisation**.
2. Cliquer sur **Créer une nouvelle règle**.
3. Choisir un **déclencheur** (ex : "Un ticket passe en 'En cours'").
4. Ajouter une **condition** (ex : "Si la priorité est Haute").
5. Définir une **action** (ex : "Notifier un utilisateur").
6. Sauvegarder et activer la règle.

Bonnes pratiques :

- Automatiser les relances de tickets en attente.
- Assigner automatiquement un responsable selon le type de ticket.
- Mettre à jour automatiquement les statuts selon les événements.

Exercice : Automatiser la Gestion des Tickets Bloqués

Objectif :

Créer une règle d'automatisation qui détecte les tickets bloqués et notifie automatiquement l'équipe.

Étapes :

1. **Créer un filtre Jira** pour identifier les tickets bloqués :

status = "En cours" AND priority = "Bloquant"

2. **Créer une règle d'automatisation** :

- **Déclencheur** : Quand un ticket est mis à jour.
- **Condition** : Si la priorité est "Bloquant".
- **Action** : Envoyer une notification à Slack et assigner un responsable.

3. **Tester la règle** en modifiant un ticket.

Critères de réussite :

- Un ticket bloqué déclenche bien une notification.
- L’équipe est informée et un responsable est automatiquement assigné.
- Les logs d’automatisation confirment que la règle fonctionne correctement.

Corrigé :

1. **Aller dans Administration > Automatisation > Créer une règle.**
2. **Ajouter un déclencheur** : "Quand un ticket est mis à jour".
3. **Ajouter une condition** : "Si la priorité = Bloquant".
4. **Ajouter une action** : "Notifier le canal Slack #support et assigner à l'utilisateur X".
5. **Sauvegarder et tester avec un ticket de test.**

Conclusion

Dans ce module, nous avons appris à intégrer Jira avec des outils externes, utiliser les webhooks et l'API REST, et automatiser les tâches pour optimiser la gestion des tickets.

Prochain module : Support et Bonnes Pratiques dans Jira.

Module 9 : Support et Bonnes Pratiques dans Jira

Objectifs du module

Ce module vise à fournir aux stagiaires les connaissances nécessaires pour résoudre les problèmes courants dans Jira et appliquer les meilleures pratiques pour assurer une gestion efficace et pérenne de la plateforme. À la fin, les participants seront capables de diagnostiquer et corriger les erreurs, optimiser l'usage de Jira et utiliser les ressources de support Atlassian.

Durée : 2 heures

1. Dépannage des Erreurs Courantes

Les problèmes les plus fréquents dans Jira peuvent être liés aux permissions, aux performances ou à des erreurs de configuration.

1.1 Erreurs liées aux permissions

Problème : Un utilisateur ne peut pas voir ou modifier un ticket.

Solution :

- Vérifier les **permissions de projet** dans **Administration > Permissions**.
- Vérifier si l'utilisateur appartient au bon **groupe ou rôle**.
- Vérifier la **sécurité des issues** (restriction d'accès par niveau de sécurité).

Problème : Un utilisateur ne peut pas exécuter une transition dans un workflow.

Solution :

- Vérifier les **conditions et validations** associées à la transition dans le workflow.
- Modifier les règles si nécessaire via **Administration > Workflows**.

Bonnes pratiques :

Éviter d'accorder des permissions globales aux utilisateurs.

Utiliser des **groupes et rôles de projet** au lieu d'attribuer des permissions individuelles.

1.2 Problèmes de Performance et Optimisation

Problème : Jira devient lent avec un grand nombre de tickets.

Solution :

1. Optimiser la base de données avec PostgreSQL : VACUUM ANALYZE;
2. Augmenter la mémoire allouée à Jira dans setenv.sh :
3. export JVM_MINIMUM_MEMORY=4g
   export JVM_MAXIMUM_MEMORY=8g
4. Désactiver les **plugins inutilisés** via **Administration > Gestion des Apps**.

Problème : Des erreurs fréquentes dans les logs Jira.

Solution :

1. Analyser les logs avec : tail -f /var/atlassian/application-data/jira/log/atlassian-jira.log
2. Rechercher des messages d'erreur spécifiques.
3. Consulter la documentation Atlassian pour des solutions adaptées.

Bonnes pratiques :

- Effectuer des **audits réguliers** des performances.
- **Planifier des maintenances** pour nettoyer les tickets obsolètes.

2. Bonnes Pratiques pour une Gestion Efficace de Jira

2.1 Structurer les Projets et Workflows

- Créer des **modèles de projets standardisés** pour éviter la duplication des configurations.
- Simplifier les workflows pour éviter trop d’étapes inutiles.
- Utiliser des couleurs et conventions claires pour les statuts.

2.2 Gestion des Tickets et Collaboration

- Attribuer les tickets de manière claire et éviter les "tickets orphelins".
- Utiliser **les commentaires et les @mentions** pour améliorer la communication.
- Fermer les tickets résolus pour éviter un backlog surchargé.

2.3 Automatisation et Optimisation des Processus

- Automatiser les **rappels pour les tickets bloqués**.
- Intégrer Jira avec Slack ou Teams pour des mises à jour en temps réel.
- Utiliser **des rapports et tableaux de bord** pour suivre la progression des projets.

Bonnes pratiques :

- Sensibiliser les utilisateurs aux bonnes pratiques Jira.
- Réaliser des **revues périodiques** des configurations et workflows.

3. Ressources et Support Atlassian

3.1 Documentation Officielle

- Support Atlassian
- Documentation Jira
- Atlassian Community

3.2 Utilisation des Outils de Support

- **Créer un ticket de support Atlassian** en cas de bug majeur.
- Utiliser **les logs d'audit** pour suivre les modifications d'administration.
- Participer aux forums pour partager des expériences et solutions.

Bonnes pratiques :

- Encourager l'autoformation via la documentation Atlassian.
- Tester les nouvelles fonctionnalités en environnement de test avant un déploiement global.

Exercice : Résolution d’un Problème Courant

Objectif :

Diagnostiquer et corriger un problème de permissions dans Jira.

Scénario :

Un utilisateur se plaint de ne pas pouvoir modifier un ticket dans un projet auquel il est censé avoir accès.

Étapes :

1. **Vérifier les permissions du projet** dans **Administration > Permissions**.
2. **Vérifier le groupe de l’utilisateur** et son appartenance aux rôles du projet.
3. **Analyser la sécurité des issues** pour s’assurer que l’accès n’est pas restreint.
4. **Modifier les permissions ou ajouter l’utilisateur au bon rôle**.
5. **Tester avec l’utilisateur concerné pour valider la correction.**

Critères de réussite :

- L’utilisateur peut modifier le ticket.
- La correction respecte les bonnes pratiques de gestion des permissions.
- Aucun autre utilisateur non autorisé ne se retrouve avec un accès excessif.

Corrigé :

1. Aller dans Administration > Permissions > Schéma de permissions du projet.
2. Vérifier que le rôle correspondant à l’utilisateur (ex : Développeur) a bien la permission "Modifier les tickets".
3. Ajouter l’utilisateur au bon rôle si nécessaire via Administration > Utilisateurs et groupes.
4. Tester avec un compte utilisateur standard pour s’assurer que seuls les bons utilisateurs ont accès.
5. Sauvegarder les modifications et informer l’utilisateur concerné.

Conclusion

Dans ce module, nous avons appris à diagnostiquer et résoudre les problèmes courants de Jira, optimiser son utilisation avec des bonnes pratiques et exploiter les ressources Atlassian pour un support efficace.

Les stagiaires devraient maintenant être en mesure d'administrer efficacement Jira dans un environnement professionnel.

Jira en master-master (actif-actif)

L'installation d'une configuration **Jira en master-master** (actif-actif) est une tâche avancée qui nécessite l'utilisation de **Jira Data Center**, car **Jira Server** ne prend pas en charge la haute disponibilité avec un clustering natif.

10. Prérequis pour une Installation Master-Master

Matériel Recommandé :

- **Deux serveurs** physiques ou virtuels avec les mêmes spécifications :
  - **CPU** : 4+ vCPU
  - **RAM** : 8 Go minimum (16 Go recommandé)
  - **Stockage** : SSD recommandé
  - **Réseau** : Connexion rapide avec faible latence entre les nœuds
- **Base de données partagée** (PostgreSQL, MySQL, SQL Server ou Oracle)
- **Stockage partagé** (NFS recommandé) pour les fichiers Jira
- **Un Load Balancer** (Nginx, HAProxy ou un équilibreur de charge matériel)

11. Installation de Jira Data Center sur les Deux Nœuds

Étape 1 : Installation de Jira Data Center sur les deux serveurs

Sur chaque serveur Jira, téléchargez et installez Jira Data Center :

```
wget https://www.atlassian.com/software/jira/downloads/binary/atlassian-jira-software-X.X.X.tar.gz
tar -xvzf atlassian-jira-software-X.X.X.tar.gz -C /opt/
cd /opt/atlassian/jira/bin/
```

Étape 2 : Configuration du Répertoire de Données Partagé

Sur les deux serveurs, montez un stockage **NFS** ou **GlusterFS** qui sera utilisé par Jira :

```
sudo mkdir -p /var/atlassian/application-data/jira
sudo mount -t nfs <IP_NFS_SERVER>:/jira /var/atlassian/application-data/jira
```

Ajoutez cette entrée à /etc/fstab pour un montage persistant :

```
<IP_NFS_SERVER>:/jira /var/atlassian/application-data/jira nfs defaults 0 0
```

Étape 3 : Configuration de la Base de Données Partagée

Sur PostgreSQL (serveur unique ou cluster), créez la base de données Jira :

```
CREATE DATABASE jiradb WITH OWNER jirauser ENCODING 'UTF8';
GRANT ALL PRIVILEGES ON DATABASE jiradb TO jirauser;
```

Configurez Jira pour utiliser cette base en éditant /opt/atlassian/jira/dbconfig.xml :

```
<database>
    <url>jdbc:postgresql://<IP_DB_SERVER>:5432/jiradb</url>
    <username>jirauser</username>
    <password>superpassword</password>
    <driver-class>org.postgresql.Driver</driver-class>
</database>
```

12. Configuration du Load Balancer

Utilisez **HAProxy** pour équilibrer la charge entre les deux nœuds Jira :

```
sudo apt install haproxy -y
```

Ajoutez cette configuration à /etc/haproxy/haproxy.cfg :

```
frontend jira_front
    bind *:80
    default_backend jira_backend
backend jira_backend
    balance roundrobin
    server jira1 <IP_JIRA1>:8080 check
    server jira2 <IP_JIRA2>:8080 check
```

Puis redémarrez HAProxy :

```
sudo systemctl restart haproxy
```

13. Activation du Mode Cluster dans Jira

Dans **/opt/atlassian/jira/bin/setenv.sh**, ajoutez :

```
export CLUSTERED=true
export JIRA_SHARED_HOME=/var/atlassian/application-data/jira
```

Puis redémarrez Jira :

```
sudo /opt/atlassian/jira/bin/start-jira.sh
```

14. Vérification et Tests

- Accédez à Jira via le Load Balancer : http://<LOAD_BALANCER_IP>/
- Vérifiez que les deux nœuds sont visibles dans **Administration > System > Clustering**
- Testez un failover en arrêtant un nœud :

```
sudo systemctl stop jira
```

- Vérifiez que l'autre nœud prend le relais sans interruption.

Conclusion

Cette configuration permet à Jira de fonctionner en mode **master-master** (actif-actif) avec une base de données partagée, un stockage commun et un Load Balancer. Elle assure **haute disponibilité** et **scalabilité** pour des équipes ayant besoin de performances optimales.

Besoin d'ajuster la configuration pour un environnement spécifique ?