3 ESGI

VERSIONING ET GIT

Détection de vulnérabilités

Durée : 1h30

Matériel autorisé :
L'usage de l'ordinateur et d'un accès à internet est autorisé.

Dès que le sujet est remis, assurez-vous qu'il soit complet.
Le sujet comporte 4 pages numérotées de 1 à 4.

Partie 1 - Questions écrites (30 minutes)

Cet exercice compte pour 10 points.

Remarque :
Le développement et la rédaction de vos réponses sont pris en compte dans la notation.

1. Quelle est l'utilité d'un fichier .gitignore et donnez un exemple de contenu pour celui-ci.
[2 points : 1 pour la définition, 1 pour l'exemple]

Un fichier .gitignore permet d'indiquer à Git quels fichiers ou dossiers doivent être ignorés et ne pas être suivis ou versionnés. cela permet d'exclure des fichiers sensibles, temporaires ou spécifiques à l'environnement local, comme des fichiers de configuration

Exemple :
```
# Ignorer les fichiers log
*.log

# Ignorer les fichiers d'environnement Python
.env
*.env

# Ignorer les fichiers de cache
/.cache/
/.pytest_cache/

# Ignorer les fichiers liés à l'IDE
.vscode/
.idea/

# Ignorer les binaires compilés
*.exe
*.dll
```

2. Expliquez ce qu'est un hook Git et donnez un exemple de cas d'utilisation.
[2 points : 1 pour la définition, 1 pour l'exemple]

Un hook Git est un script qui s'exécute automatiquement à certains moments du cycle de vie de Git, comme lors des commits, des merge, ou des push, ça permet d'automatiser certaines action et d'imposer des règles au sein d'un projet.

Un hook pre-commit peut-être utilisé pour vérifier que le formats de code est respecé et autres avant d'autoriser un commit. Un exemple de script dans .git/hooks/pre-commit pourrait vérifier que le code Python suit la norme (PEP8) avant de permettre le commit.

3. Citez deux bonnes pratiques pour éviter de compromettre des informations sensibles (comme des clés API) dans un projet versionné avec Git.
[2 points : 1 point par bonne pratique]

Utiliser un fichier de configuration ignoré par Git : Placer les informations sensibles dans un fichier de configuration (config.env), ajouté dans le fichier .gitignore pour éviter de le versionner.

Utiliser des variables d'environnement : Configurer des variables d'environnement sur le Système local ou sur le serveur pour stocker les clés API, token et autres informations sensibles, et les référencer dans le code au lieu de les stocker directement.

4. Qu'est-ce qu'une règle de protection de branche et pourquoi l'utiliser dans un projet collaboratif ?
[2 points : 1 pour la définition, 1 pour la justification]

Une règle de protection de branche est une restriction appliquée sur certaines branches d'un projet git afin empêcher les modifications non contrôlées (comme un commit ou un push direct) et pour imposer des processus de validation (comme la revue de code).
Dans un projet collaboratif, ça permet d'assurer la qualité du code, d'éviter les erreurs humaines et de faciliter la collaboration en s'assurant que le code est vérifié par un ou plusieurs collègues avant d'être intégré dans les branches principales, comme main(master pour certains aficionado) ou develop.

5. Comment configurer une règle de protection de branche pour imposer une revue de code avant tout merge sur GitHub ?
[2 points]

Ouvrez les paramètres de votre dépôt sur GitHub.

cliquez sur Branches, sélectionner Add Rule afin d'ajouter une nouvelle règle.

Choisissez la branche à protéger exemple : la branche principale.

Cochez Require pull request reviews before merging pour rendre la revue de code obligatoire avant tout merge.
Sauvegardez pour que cela s'applique