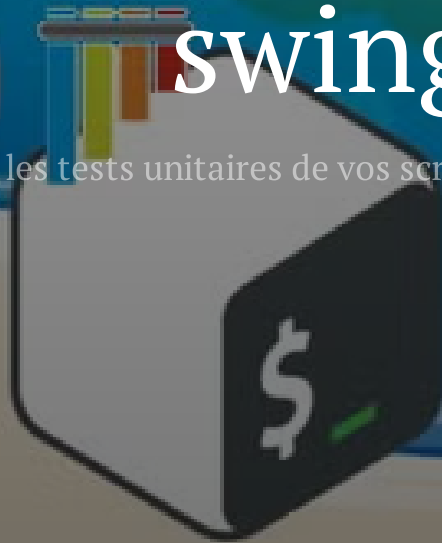


Django Rennes art, Bashville swing

Écrivez les tests unitaires de vos scripts bash avec pytest








Sommaire

1. Présentation du projet ecoCode
2. Pytest et le plugin pytest-shell-utilities
3. Prérequis à ajouter au script
4. Démo script toolbox.sh
5. Installation de l'environnement
6. Présentation des tests unitaires du script toolbox.sh
7. Explication des méthodes utilisés pour les tests
8. Intégration d'un workflow github action pour exécuter les tests
9. Conclusion

Présentation du projet ecoCode

Réduire l'impact environnemental et social des solutions numériques.

-  Green code initiative
-  Plugins SonarQube
-  Projet ecoCode-java
-  docker compose et Dockerfile
-  Toolbox

Pytest et le plugin pytest-shell-utilities

Objectif : utiliser pytest pour lancer les tests unitaires d'un script bash.

2 plugins de disponibles sur PyPi :

- pytest-shell
- **pytest-shell-utilities**

Fonctionnalités attendues :

- Tester le code retour
- Récupérer la sortie standard
- Récupérer la sortie erreur

Utilisation du module **subprocess** pour exécuter la commande et récupérer le résultat.

Prérequis à ajouter au script

toolbox.sh

```
function main() {
    ARGS=() TEST=0
    # ...
    if [[ $TEST -gt 0 ]]; then
        execute_unit_test
        return $?
    fi
    # ...
    return 0
}

function execute_unit_test() {
    if [[ -z "${ARGS[0]}" ]]; then
        error "No function to execute" && return 1
    fi
    # If a function is passed as the first argument, we check that it exists
    if ! [[ $(type -t "${ARGS[0]}") == function ]]; then
        error "Function with name ${ARGS[0]} does not exist" && return 2
    fi
    # Initialize fixtures
    [[ $FIXTURE -eq 1 ]] && ECOCODE_DOCKER_ENV="test_docker_env"
    # execute function
    "${ARGS[@]}"
    return $?
}
```

Démo script toolbox.sh

Exécuter la fonction sans l'option **-test**

```
./toolbox.sh info "test"
Usage toolbox.sh [OPTIONS] COMMAND
# ...
echo $?
0
```

Exécuter la fonction avec l'option **-test**

```
./toolbox.sh info "test" --test
test
echo $?
0
```

Exécuter une fonction non définie avec l'option **-test**

```
./toolbox.sh test1 --test
Function with name test1 does not exist
echo $?
2
```

Installation de l'environnement

Méthodes d'installation.

Sans Docker :

- python 3.12
- poetry
- maven
- java

Avec Docker :

- toolbox.Dockerfile
- utils.sh

Présentation des tests unitaires du script toolbox.sh

tests/test_toolbox.py

Prérequis et import des modules

```
import inspect
import os
import pytest

current_dir: str = os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe()))))
if os.environ['HOME'] == "/app":
    current_dir = "/app/tests"
project_path: str = os.path.abspath(f"{current_dir}/..")
script: str = os.path.abspath(f"{current_dir}/../toolbox.sh")
```

Exemple de test unitaire

```
def test_function_not_exist(shell):
    ret = shell.run(script, "test_function", "--test")
    assert ret.stderr.rstrip() == "Function with name test_function does not exist"
    assert ret.returncode == 2
```


Explication des méthodes utilisés pour les tests

Récupérer le code retour du script bash

```
ret.returncode
```

Vérifier la sortie standard

```
ret.stdout  
ret.stdout.rstrip()  
ret.stdout.splitlines()[0]
```

Vérifier la sortie erreur

```
ret.stderr
```

Intégration d'un workflow github action pour exécuter les tests

`.github/workflows/bash_tests.yml`

- Installation de Python
- Installation de Poetry
- Installation des dépendances
- Exécution des tests

Remarque : gestion du cache pour la prochaine exécution de la pipeline dans github

Conclusion

Simplicité de mise en place du plugin, avec Poetry notamment.

Nécessite une petite adaptation du script shell pour pouvoir exécuter une fonction unitairement.

Plusieurs méthodes de disponibles avec le plugin pytest-shell-utilities pour récupérer :

- le code retour du script bash
- la sortie standard
- la sortie erreur

Tests unitaires facilement intégrables dans un workflow github action.