PYTHON PARA AUTOMAÇÃO EM DEVOPS
PROF.: Leandro Lessa

Atividade

Desenvolvimento de uma Ferramenta de Backup Automatizado com Modularização, Testes Unitários e Execução em Container Docker

Objetivos de Ensino

Desenvolver uma aplicação Python que realize backups automáticos de arquivos em um diretório especificado, simulando cenários reais de DevOps. A aplicação deve ser modularizada para promover reutilização de código, incluir geração de logs para rastreamento de operações, testes unitários com pytest para garantir a qualidade, e ser executável em um container Docker para portabilidade e isolamento.

Requisitos Específicos

Backups de Arquivos

- A aplicação deve copiar arquivos de um diretório de origem para um diretório de destino, criando backups dos arquivos.
    - Bônus: Criação de versionamento com timestemp no nome ou pasta(ex.: arquivo_backup_20231216_120000.ext | PASTA 20251218).
- Simulação de falhas: Incluir opções para testar cenários de erro, como diretórios inexistentes.

Modularização dos Módulos em Python

- Dividir o código em módulos separados:
  - backup.py: Funções principais para realizar o backup.
  - logger.py: Funções para geração de logs.
  - main.py: Script principal que integra os módulos e executa o backup.
  - Outros módulos conforme implementação pessoal.
- Usar boas práticas: Funções puras, tratamento de exceções e documentação com docstrings.

Testes Unitários Utilizando Pytest

- Criar um pacote tests/ com arquivos de teste (ex.: test_backup.py, test_logger.py).
- Cobrir cenários como: backup bem-sucedido, falha em diretórios inexistentes, verificação de logs gerados.
- Desenvolver testes de acordo com as funções desenvolvidas.

Geração de Logs dos Arquivos Realizados Backups

- Registrar em um arquivo de log (ex.: backup.log) detalhes como: arquivos copiados, timestamps, erros ocorridos e status de sucesso/falha.
- Níveis de log: INFO para operações normais, ERROR para falhas.

Execução do Projeto em um Container do Docker

- Criar um Dockerfile que instale dependências (Python, pytest) e copie o código para o container.
- O container deve executar o main.py
    - **Bônus:** Criar entrypoint, com opções para passar argumentos (ex.: diretórios de origem e destino via linha de comando).
    - https://docs.python.org/3/library/sys.html (sys.argv)

Entregáveis

- Código fonte completo (módulos Python, testes, Dockerfile).
- Relatório explicando a arquitetura, como executar o projeto e resultados dos testes.
- Demonstração: Documento passo a passo com os scripts utilizados para execução do container e os prints dos resultados obtidos.

Critérios de Avaliação

- Funcionalidade: Backups corretos e logs gerados (40%).
- Modularização e Qualidade de Código: Separação de responsabilidades e docstrings (20%).
- Testes: Aplicação de testes unitários (20%).
- Docker: Container funcional e portável (20%).