Sonarcloud

create new folder in vs

create python file

eg

hello.py

```
def greet(name):
    return f"Hello, {name}!"


print(greet("World"))
```

-> git init

-> git add hello.py

-> git commit -m "Initial commit"

create new repo on GitHub

copy url

-> git branch -M main

-> git remote add origin <>

-> git push -u origin main

login on sonar cloud

click on + button, analyse new project. link the git repo

Generate a Token:

Go to SonarCloud.

In the top right corner, click on your user profile and choose My Account.

Navigate to Security on the left menu.

Scroll down to the Tokens section and click Generate Token.

Copy the generated token

go to project -> information -> copy project key and organization key

go to vs->

->mkdir .github\workflows

create new file - sonarcloud.yml

```
name: Build
on:
 push:
  branches:
   - main
 pull_request:
  types: [opened, synchronize, reopened]
jobs:
 sonarcloud:
  name: SonarQube Cloud
  runs-on: ubuntu-latest
  steps:
   - uses: actions/checkout@v4
    with:
     fetch-depth: 0  # Shallow clones should be disabled for a better relevancy of analysis
   - name: SonarQube Cloud Scan
    uses: SonarSource/sonarcloud-github-action@master
    env:
     SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```

go to git repo (sonarcloud)

settings->

secrets and variables-> actions -> new repo secret -> name =SONAR_TOKEN - paste copied token

in vs code-> new file -> sonar-project.properties

```
# Project identification

sonar.projectKey=<> (project key)

sonar.organization=<> (proj organization)

sonar.host.url=https://sonarcloud.io

sonar.login=<> (secret)


# Project details

sonar.projectName=Python SonarCloud Demo

sonar.projectVersion=1.0


# Source file settings

sonar.sources=.

sonar.language=py
```

Go to sonarcloud project -> administration -> analysis method -> toggle automatic analysis off

-> git add .

->git commit -m "Add SonarCloud configuration and workflow2

->git push origin main

-> git status

go to GitHub and check actions tab -> workflow should be running

go to sonarcloud dashboard and check quality gate.

to get all parameters, try diff code:

```
import os


# 1. Hardcoded credentials and printing sensitive data
def connect_to_db():
    username = "root"  # Hardcoded credentials
    password = "12345"  # Hardcoded password
    print(f"Connecting to database with username: {username} and password: {password}")  #
Exposing credentials
    # Simulating a bad connection (e.g., missing actual connection logic)


# 2. SQL Injection vulnerability and concatenation of user input
def unsafe_query(user_input):
    query = "SELECT * FROM users WHERE name = '" + user_input + "';"  # Vulnerable to SQL
Injection
    print("Executing query: " + query)  # Directly logging the potentially dangerous query
    # Executing without sanitizing or using prepared statements


# 3. Path traversal vulnerability without validation
def read_file(file_name):
    if "../" in file_name:  # Extremely naive attempt at sanitization
        print("Trying to block path traversal, but failing.")
    with open(file_name, 'r') as file:  # Potential path traversal issue
        data = file.read()
        print("File content: " + data)  # Printing file content unsafely


# 4. Use of insecure hashing algorithm and poor error handling
def hash_password(password):
    import hashlib
    try:
        hashed = hashlib.md5(password.encode()).hexdigest()  # Using MD5 (insecure)
        print("MD5 hash of password: " + hashed)
    except:
```

```
        print("Something went wrong while hashing, but I'm not telling you what!")


# 5. Environment variables logging (security issue)
def log_environment():
    print("Listing all environment variables for fun and insecurity:")
    for key, value in os.environ.items():
        print(f"{key}: {value}")


# 6. Unused imports and inefficient code
import random
for i in range(1000000):  # Inefficient loop doing nothing
    pass


if __name__ == "__main__":
    connect_to_db()
    unsafe_query("'; DROP TABLE users; --")
    read_file("/etc/passwd")
    hash_password("very_secure_password")
    log_environment()
```