

SONARQUBE CPP PROJECT ANALYSIS

Step 1: Install Prerequisites

1. `sudo apt install openjdk-17-jdk -y`
2. `sudo apt install cppcheck`
3. `sudo apt install curl`

Step 2: Install SonarQube community edition

“

```
cd ~/Downloads
```

```
unzip sonarqube-9.9.8.100196.zip
```

```
sudo mv sonarqube-9.9.8.100196 /opt/sonarqube
```

```
sudo chown -R $USER:$USER /opt/sonarqube
```

```
cd /opt/sonarqube/bin/linux-x86-64
```

```
./sonar.sh start
```

”

Step 3: Go to SonarQube server

1. <http://127.0.0.1:9000>
2. Login:
Username- admin
Password- admin
(Change to new password when prompted)
3. Click on manually create project.,
 - A. Give project name [eg., cpp-analysis] and fill other stuffs and click create.
 - B. Click on the newly created project, and locate *locally analyse* section.
Click on that, and give token name [eg., cpp-analysis] set the expiration date to never, and then click on generate token.
 - C. Copy and keep the generated token somewhere safely, [this token will later be used in sonar-project.properties and makefile].

Step 4: create a sample cpp project

1. `mkdir -p ~/cpp-analysis cd ~/cpp-analysis`
2. **main.cpp**
Content:
`#include <iostream>`
`#include <vector>`

```

class Person {
public:
    Person(const std::string& name, int age) : name(name), age(age) {}

    void printInfo() const {
        std::cout << "Name: " << name << ", Age: " << age << std::endl;
    }

private:
    std::string name;
    int age;
};

class Group {
public:
    void addPerson(const std::string& name, int age) {
        Person* p = new Person(name, age); // Memory leak if not deleted
        members.push_back(p);
    }

    void printGroupInfo() const {
        for (int i = 0; i <= members.size(); i++) { // Off-by-one error
            members[i]->printInfo();
        }
    }

    ~Group() {
        for (Person* p : members) {
            delete p; // Properly deleting allocated memory
        }
    }

private:
    std::vector<Person*> members; // Using raw pointers, can cause memory issues
};

int main() {
    Group group;
    group.addPerson("Alice", 30);
    group.addPerson("Bob", 25);

    // Logic error: group.printGroupInfo() not called

```

```
    return 0;
}
```

3. **sonar-project.properties**

Content:

```
# Basic project information
sonar.projectKey=cpp-analysis
sonar.projectName=cpp-analysis
sonar.projectVersion=1.0
sonar.language=cpp
# Source file configuration
sonar.sources=.
# SonarQube server URL
sonar.host.url=http://127.0.0.1:9000
sonar.login=sqp_f488b71485b9e975180f1819d95d5bc84e2eb43d
# External issues report path (will be generated later)
sonar.externalIssuesReportPaths=issues-report.json
```

4. Run Cppcheck to Generate an XML Report:

```
“”
    cppcheck --enable=all --xml main.cpp 2> cppcheck-report.xml
“”
```

Verify the XML Report:

```
“”
cat cppcheck-report.xml
“”
```

5. **convert_cppcheck_to_json.py**

Content:

```
import xml.etree.ElementTree as ET
import json
import sys
if len(sys.argv) != 3:
    print("Usage: python3 convert_cppcheck_to_json.py <input_xml> <output_json>")
    sys.exit(1)
input_file = sys.argv[1]
output_file = sys.argv[2]

errors = []

try:
    tree = ET.parse(input_file)
    root = tree.getroot()
```

```

for error in root.findall('./errors/error'):
    file = error.find('location').get('file') if error.find('location') is
not None else None
    message = error.get('msg')
    severity = error.get('severity').upper()

    errors.append({
        "file": file,
        "message": message,
        "severity": severity
    })

with open(output_file, 'w') as json_file:
    json.dump(errors, json_file, indent=4)

print(f"JSON report saved to {output_file}")
except Exception as e:
    print(f"Error: {e}")

```

6. Convert the Cppcheck XML Report to JSON:

```

"""
python3 convert_cppcheck_to_json.py cppcheck-report.xml issues-report.json
"""

Verify the JSON Report:
"""
cat issues-report.json
"""

```

Step 5: Install SonarScanner CLI

1. Go to this website [SonarScanner](#) and download the version suitable to your system.
2. `cd ~/Downloads`
3. `unzip sonar-scanner-cli-6.2.1.4610-linux-x64.zip`
4. `sudo mv sonar-scanner-6.2.1.4610-linux-x64 /opt/sonar-scanner`
5. `cd ~`
6. `export PATH=$PATH:/opt/sonar-scanner/bin`
7. `echo 'export PATH=$PATH:/opt/sonar-scanner/bin' >> ~/.bashrc`
`source ~/.bashrc`
8. `sonar-scanner -version`

Step 6: Analyzing the cpp-analyse project folder

1. `cd ~/cpp-analyse`

2. sonar-scanner
3. After the successful execution of the above command, go to the sonarqube server and view the updated analysis report changes for the project, this indicates that the sever is working properly.

4. **Makefile**

Content:

CXX = g++

CXXFLAGS = -Wall -Wextra -std=c++17

SOURCES = \$(wildcard *.cpp)

TARGET = example

ISSUES_REPORT = issues-report.json

SONAR_HOST_URL = http://127.0.0.1:9000

SONAR_PROJECT_KEY = cpp-analysis

SONAR_LOGIN = sqp_f488b71485b9e975180f1819d95d5bc84e2eb43d

.PHONY: all sonar fetch-issues clean

all: build sonar fetch-issues

build:

\$(CXX) \$(CXXFLAGS) \$(SOURCES) -o \$(TARGET)

sonar:

@echo "Running SonarScanner..."

@sonar-scanner -Dsonar.projectKey=\$(SONAR_PROJECT_KEY) -Dsonar.sources=. -

Dsonar.host.url=\$(SONAR_HOST_URL) -Dsonar.login=\$(SONAR_LOGIN)

fetch-issues:

@echo "Fetching issues from SonarQube server..."

@curl -u \$(SONAR_LOGIN):

"http://127.0.0.1:9000/api/issues/search?componentKeys=\$(SONAR_PROJECT_KEY)&ps=500" |
jq '.issues[] | {file: .file, message: .message, severity: .severity}'

clean:

rm -f \$(TARGET) \$(ISSUES_REPORT)

5. Now execute this command:

“”

make all

“”

[you will see the issues of the cpp file in the terminal and in the server too!]

OUTPUTS:

1. SonarQube server analysis issue page:

The screenshot shows the SonarQube web interface for a project named 'cpp-analysis'. The browser address bar shows the URL: `127.0.0.1:9000/project/issues?id=cpp-analysis&resolved=false`. The interface includes a sidebar with navigation options like Overview, Issues, Security Hotspots, Measures, Code, and Activity. The 'Issues' tab is active, displaying a list of issues for the file 'main.cpp'. The issues are categorized by type (Code Smell), severity (Info, Major), and scope (Not assigned). The issues are as follows:

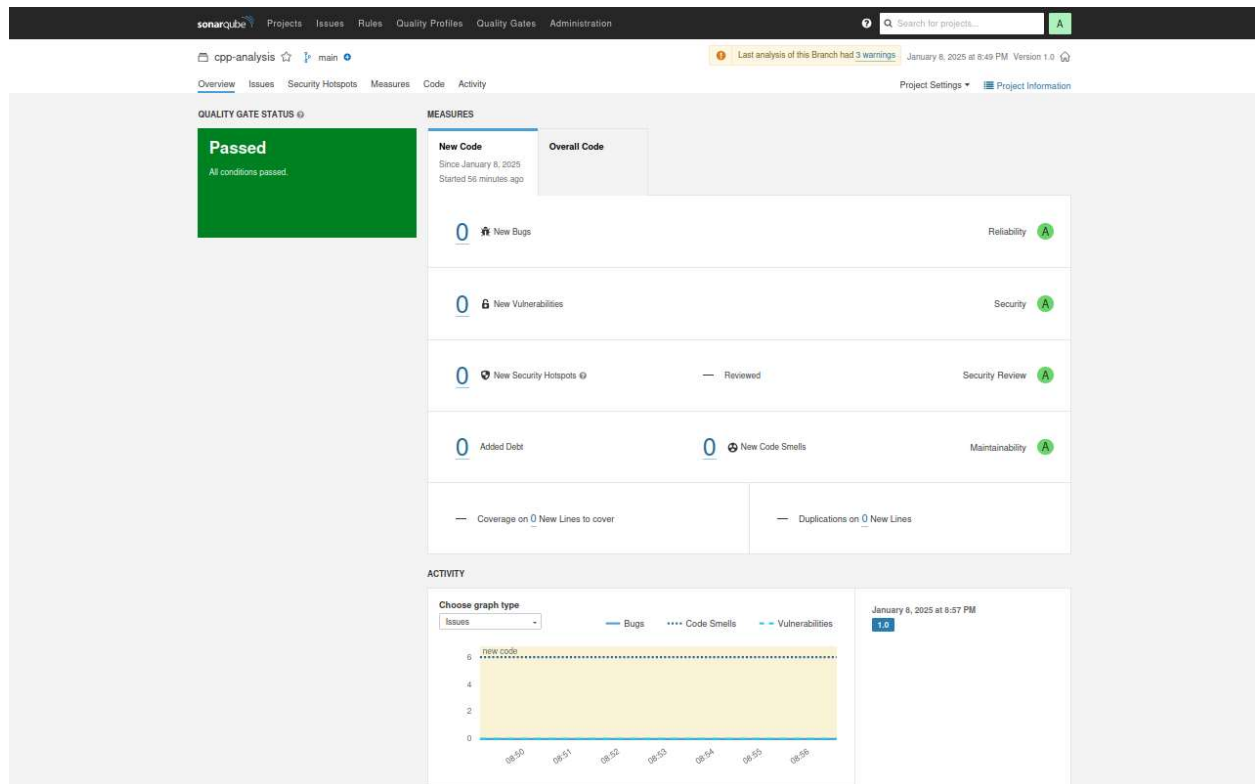
Issue ID	Message	Type	Severity	Scope	Resolution	Status	Effort
L1	Include file: <iostream> not found. Please note: Cppcheck does not need standard library headers to get proper results.	Code Smell	Info	Not assigned	0min effort	Open	0
L2	Include file: <vector> not found. Please note: Cppcheck does not need standard library headers to get proper results.	Code Smell	Info	Not assigned	0min effort	Open	0
L24	The function 'printGroupInfo' is never used.	Code Smell	Info	Not assigned	0min effort	Open	0
L25	Out of bounds access in expression 'members[i]'	Code Smell	Major	Not assigned	0min effort	Open	0
L26	Out of bounds access in expression 'members[i]'	Code Smell	Major	Not assigned	0min effort	Open	0
L26	When <code>i==members.size()</code> , <code>members[i]</code> is out of bounds.	Code Smell	Major	Not assigned	0min effort	Open	0

2. Issues displayed in terminal:

The screenshot shows a terminal window with the command `ananyaa@ananyaa-VirtualBox: ~/cpp-analysis`. The output of the command is a JSON array of issues, each containing a file path, a message, and a severity level. The issues are as follows:

```
{
  "file": null,
  "message": "Include file: <iostream> not found. Please note: Cppcheck does not need standard library headers to get proper results.",
  "severity": "INFO"
},
{
  "file": null,
  "message": "Include file: <vector> not found. Please note: Cppcheck does not need standard library headers to get proper results.",
  "severity": "INFO"
},
{
  "file": null,
  "message": "The function 'printGroupInfo' is never used.",
  "severity": "INFO"
},
{
  "file": null,
  "message": "Out of bounds access in expression 'members[i]'",
  "severity": "MAJOR"
},
{
  "file": null,
  "message": "Out of bounds access in expression 'members[i]'",
  "severity": "MAJOR"
},
{
  "file": null,
  "message": "When i==members.size(), members[i] is out of bounds.",
  "severity": "MAJOR"
}
```

3. SonarQube server overview:



4. Cpp-analysis project folder:

