Continuous Integration Toolkit

GitHub-first (Minted Edition) with C and C++

November 4, 2025

Contents

How to build. This document uses `minted`, which calls `pygmentize`. Compile with:

```
latexmk -pdf -shell-escape ci_toolkit_minted_ccpp.tex
```

If you cannot enable `-shell-escape`, I can ship a *frozencache* variant with pre-generated styles.

1 Philosophy & Goals

- **Fast feedback:** run on every push/PR; show coverage and lint results in the PR.
- **Quality gates:** enforce coverage thresholds and required checks before merge.
- **Secure by default:** enable CodeQL and Secret Scanning; keep dependencies healthy.
- **Simple to adopt:** copy-paste scaffolds for Node, Python, Java, C, and C++.
- **Maintainable:** reusable workflows, caching, and matrices.

2 Scaffold: Node + Jest

Init project

```
npm init -y
npm i -D jest
mkdir -p src __tests__
```

src/appOperations.js

```
function multiply(a, b) { return a * b; }
function add(a, b) { return a + b; }
function subtract(a, b) { return a - b; }
module.exports = { multiply, add, subtract };
```

__tests__/appOperations.test.js

```
const { multiply, add, subtract } = require('../src/appOperations');

test('multiply 5 x 0 = 0', () => {
  expect(multiply(5, 0)).toBe(0);
});

test('add 5 + 5 = 10', () => {
  expect(add(5, 5)).toBe(10);
});

test('subtract 15 - 5 = 10', () => {
  expect(subtract(15, 5)).toBe(10);
});
```

package.json (scripts)

```
{
  "scripts": {
    "test": "jest --coverage"
  }
}
```

jest.config.js

```
module.exports = {
  testEnvironment: 'node',
  collectCoverage: true,
  coverageThreshold: {
    global: { lines: 80, branches: 80, functions: 80, statements: 80 }
  }
};
```

3 CI Workflow: Tests & Coverage on PRs

.github/workflows/ci.yml

```yaml
name: Unit tests & coverage

on:
  pull_request:
    branches: [ main ]
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    permissions:
      contents: write
      checks: write
      pull-requests: write
    concurrency:
      group: ci-${{ github.ref }}
      cancel-in-progress: true
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 1
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
          cache: 'npm'
      - run: npm ci
      - run: npm test -- --coverage
      - name: Publish Jest coverage to PR
        uses: ArtiomTr/jest-coverage-report-action@v2
        with:
          github-token: ${{ secrets.GITHUB_TOKEN }}
      - name: Upload coverage artifact
        uses: actions/upload-artifact@v4
        with:
          name: coverage-${{ github.sha }}
          path: coverage
          retention-days: 7
```

Why this matters. The coverage thresholds in `jest.config.js` enforce a minimum quality gate. The action posts a coverage summary on the PR so reviewers can spot regressions quickly.

4 Python Variant (pytest)

requirements.txt

```
pytest
pytest-cov
```

app.py

```python
def multiply(a, b):
    return a * b

def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

tests/test_app.py

```python
from app import multiply, add, subtract

def test_multiply():
    assert multiply(5, 0) == 0

def test_add():
    assert add(5, 5) == 10

def test_subtract():
    assert subtract(15, 5) == 10
```

pytest.ini

```
[pytest]
addopts = -q --cov=. --cov-report=term-missing
```

.github/workflows/ci-python.yml

```yaml
name: Pytests

on:
  pull_request:
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: "3.12"
          cache: "pip"
      - run: pip install -r requirements.txt
      - run: pytest
      - name: Upload coverage XML (optional)
        run: pytest --cov=. --cov-report=xml:coverage.xml
      - uses: actions/upload-artifact@v4
        with:
          name: coverage-xml
          path: coverage.xml
```

5 Java Variant (Maven + JaCoCo)

pom.xml (snippets)

```xml
<project>
  <properties>
    <maven.compiler.source>21</maven.compiler.source>
    <maven.compiler.target>21</maven.compiler.target>
  </properties>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.2.5</version>
      </plugin>
      <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.11</version>
        <executions>
          <execution>
            <goals><goal>prepare-agent</goal></goals>
          </execution>
          <execution>
            <id>report</id>
            <phase>test</phase>
            <goals><goal>report</goal></goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

.github/workflows/ci-java.yml

```yaml
name: Maven tests

on:
  pull_request:
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-java@v4
        with:
          distribution: temurin
          java-version: "21"
          cache: maven
      - run: mvn -B -DskipTests=false test
      - uses: actions/upload-artifact@v4
        with:
          name: jacoco
          path: target/site/jacoco
```

6 C Variant (CMake + CTest + gcovr)

Directory layout

```
include/mathops.h
src/mathops.c
tests/test_mathops.c
CMakeLists.txt
```

include/mathops.h

```c
#ifndef MATHOPS_H
#define MATHOPS_H

int add(int a, int b);
int subtract(int a, int b);
int multiply(int a, int b);

#endif
```

src/mathops.c

```c
#include "mathops.h"

int add(int a, int b)      { return a + b; }
int subtract(int a, int b) { return a - b; }
int multiply(int a, int b) { return a * b; }
```

tests/test_mathops.c

```c
#include <stdio.h>
#include "mathops.h"

int main(void) {
    if (multiply(5, 0) != 0) return 1;
    if (add(5, 5) != 10) return 1;
    if (subtract(15, 5) != 10) return 1;
    printf("All C tests passed\n");
    return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.20)
project(ci_c C)

set(CMAKE_C_STANDARD 11)

option(ENABLE_COVERAGE "Enable coverage flags" ON)

if (ENABLE_COVERAGE AND CMAKE_C_COMPILER_ID MATCHES "GNU|Clang")
  add_compile_options(--coverage -O0 -g)
  add_link_options(--coverage)
endif()

add_library(mathops src/mathops.c)
target_include_directories(mathops PUBLIC include)

add_executable(test_math tests/test_mathops.c)
target_link_libraries(test_math PRIVATE mathops)

include(CTest)
add_test(NAME mathops_tests COMMAND test_math)
```

.github/workflows/ci-c.yml (with coverage gate)

```yaml
name: C (CMake + CTest + gcovr)

on:
  pull_request:
  push:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Install tools
        run: |
          sudo apt-get update
          sudo apt-get install -y cmake gcc gcovr lcov
      - name: Configure
        run: cmake -S . -B build -DENABLE_COVERAGE=ON
      - name: Build
        run: cmake --build build --config Debug -- -j2
      - name: Test
        run: ctest --test-dir build --output-on-failure
      - name: Coverage (gcovr, fail under 80%)
        run: |
          gcovr -r . --filter 'src/' --xml -o build/coverage.xml \
                --html --html-details -o build/coverage.html \
                --fail-under-lines 80
      - uses: actions/upload-artifact@v4
        if: always()
        with:
          name: c-coverage
          path: build/coverage.*
```

Notes (C). `gcovr` *fails the job* if line coverage drops below 80% (adjust as needed). Keep optimization off (`-O0`) for accurate coverage.

7 C++ Variant (CMake + GoogleTest + gcovr)

Directory layout

```
include/mathops.hpp
src/mathops.cpp
tests/mathops_test.cpp
CMakeLists.txt
```

include/mathops.hpp

```cpp
#pragma once
namespace mathops {
  inline int add(int a, int b)      { return a + b; }
  inline int subtract(int a, int b) { return a - b; }
  inline int multiply(int a, int b) { return a * b; }
}
```

src/mathops.cpp

```cpp
#include "mathops.hpp"
// All functions are inline in the header for brevity; translation unit kept for
↪  structure.
```

tests/mathops_test.cpp

```cpp
#include <gtest/gtest.h>
#include "mathops.hpp"

TEST(MathOps, Multiply) { EXPECT_EQ(mathops::multiply(5, 0), 0); }
TEST(MathOps, Add)      { EXPECT_EQ(mathops::add(5, 5), 10); }
TEST(MathOps, Subtract) { EXPECT_EQ(mathops::subtract(15, 5), 10); }

int main(int argc, char** argv) {
  ::testing::InitGoogleTest(&argc, argv);
  return RUN_ALL_TESTS();
}
```

CMakeLists.txt (FetchContent + coverage flags)

```
cmake_minimum_required(VERSION 3.20)
project(ci_cpp CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

option(ENABLE_COVERAGE "Enable coverage flags" ON)

if (ENABLE_COVERAGE AND CMAKE_CXX_COMPILER_ID MATCHES "GNU|Clang")
  add_compile_options(--coverage -O0 -g)
  add_link_options(--coverage)
endif()

add_library(mathops_cpp src/mathops.cpp)
target_include_directories(mathops_cpp PUBLIC include)

include(FetchContent)
FetchContent_Declare(
  googletest
  URL https://github.com/google/googletest/archive/refs/tags/v1.14.0.zip
)
FetchContent_MakeAvailable(googletest)

add_executable(mathops_test tests/mathops_test.cpp)
target_link_libraries(mathops_test PRIVATE mathops_cpp GTest::gtest GTest::gtest_main)

include(GoogleTest)
gtest_discover_tests(mathops_test)
```

.github/workflows/ci-cpp.yml (matrix + coverage gate)

```yaml
name: C++ (CMake + GTest + gcovr)

on:
  pull_request:
  push:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        compiler: [gcc, clang]
    steps:
      - uses: actions/checkout@v4
      - name: Install tools
        run: |
          sudo apt-get update
          sudo apt-get install -y cmake g++ clang gcovr lcov
      - name: Select compiler
        run: |
          if [ "${{ matrix.compiler }}" = "clang" ]; then
            export CC=clang
            export CXX=clang++
          else
            export CC=gcc
            export CXX=g++
          fi
          echo "CC=$CC"  >> $GITHUB_ENV
          echo "CXX=$CXX" >> $GITHUB_ENV
      - name: Configure
        run: cmake -S . -B build -DENABLE_COVERAGE=ON
      - name: Build
        run: cmake --build build --config Debug -- -j2
      - name: Test
        run: ctest --test-dir build --output-on-failure
      - name: Coverage (gcovr, fail under 80%)
        run: |
          gcovr -r . --filter 'src/' --xml -o build/coverage.xml \
                --html --html-details -o build/coverage.html \
                --fail-under-lines 80
      - uses: actions/upload-artifact@v4
        if: always()
        with:
          name: cpp-coverage-${{ matrix.compiler }}
          path: build/coverage.*
```

Notes (C++). The matrix validates both `gcc` and `clang`. GoogleTest is pulled with CMake `FetchContent`. Coverage is enforced via `gcovr` with an 80% line gate.

8 Security and Quality Gates (Shared Patterns)

8.1 CodeQL (Code Scanning)

.github/workflows/codeql.yml

```yaml
name: CodeQL

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
  schedule:
    - cron: '0 2 * * 1'

jobs:
  analyze:
    permissions:
      actions: read
      contents: read
      security-events: write
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: github/codeql-action/init@v3
        with:
          languages: javascript, cpp
      - uses: github/codeql-action/autobuild@v3
      - uses: github/codeql-action/analyze@v3
```

8.2 Secret Scanning and Push Protection

.github/secret_scanning.yml (custom patterns & ignores)

```yaml
custom-patterns:
  - name: Internal API token
    pattern: 'sbm_[A-Za-z0-9]{24,}'
    secret-group: internal

paths-ignore:
  - "docs/**"
  - "*.md"
```

8.3 Dependency Hygiene (Dependabot)

.github/dependabot.yml

```yaml
version: 2
updates:
  - package-ecosystem: npm
    directory: "/"
    schedule: { interval: weekly }
    open-pull-requests-limit: 5
  - package-ecosystem: pip
    directory: "/"
    schedule: { interval: weekly }
  - package-ecosystem: maven
    directory: "/"
    schedule: { interval: weekly }
  - package-ecosystem: github-actions
    directory: "/"
    schedule: { interval: monthly }
```

9 Linting & Formatting (Optional but Recommended)

9.1 Node (ESLint)

Install

```
npm i -D eslint
npx eslint --init
```

CI step

```yaml
- run: npm run lint
```

9.2 Python (flake8 + black)

Install

```
pip install flake8 black
```

CI steps

```yaml
- run: flake8 .
- run: black --check .
```

9.3 C/C++ (clang-tidy / clang-format)

Minimal config files

```
# .clang-tidy (example)
Checks: '-*,bugprone-*,performance-*,readability-*'
WarningsAsErrors: 'bugprone-*,performance-*'
```

```
# .clang-format (example)
BasedOnStyle: LLVM
IndentWidth: 2
ColumnLimit: 100
```

CI snippets

```
- name: clang-tidy
  run: clang-tidy **/*.cpp -- -std=c++20

- name: clang-format (diff check)
  run: |
    if ! clang-format --dry-run --Werror $(git ls-files '*.c' '*.cpp' '*.hpp'); then
      echo "Formatting issues found"; exit 1; fi
```

10 Monorepos & Reuse

10.1 Reusable Workflows

.github/workflows/reuse-node.yml

```
name: Reusable Node CI
on:
  workflow_call:
    inputs:
      node-version:
        required: true
        type: string
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: ${{ inputs.node-version }}
          cache: npm
      - run: npm ci && npm test -- --coverage
```

Consumer

```yaml
jobs:
  use-reusable:
    uses: ./.github/workflows/reuse-node.yml
    with:
      node-version: "20"
```

11 Matrices, Caching, and Speed

11.1 Selective runs (paths filters)

```yaml
on:
  pull_request:
    paths:
      - "apps/web/**"
      - ".github/workflows/ci.yml"
```

11.2 General caching tips

- Node: use `setup-node` with `cache: npm`; prefer `npm ci`.
- Python: `setup-python` with `cache: pip`.
- Java: `setup-java` with `cache: maven`.
- C/C++: enable ccache if needed; keep `fetch-depth: 1` for speed.

12 Badges (Optional)

```
![CI](https://github.com/OWNER/REPO/actions/workflows/ci.yml/badge.svg)
[![CodeQL](https://github.com/OWNER/REPO/actions/workflows/codeql.yml/badge.svg)](https://github.com/O
```

13 Troubleshooting (Minted)

- **Missing $ inserted / caption issues:** avoid fragile macros in captions. Use paragraph labels instead.
- **"Missing Pygments output" or "Cannot find style":** ensure `pygmentize` is installed and build with `-shell-escape`. We select the `friendly` style to avoid custom style files.
- **"Missing \end{minted}":** verify each `minted` block is properly closed.
- **Coverage gates for C/C++:** adjust `gcovr` thresholds with `-fail-under-lines` and optionally `-fail-under-branches`.