

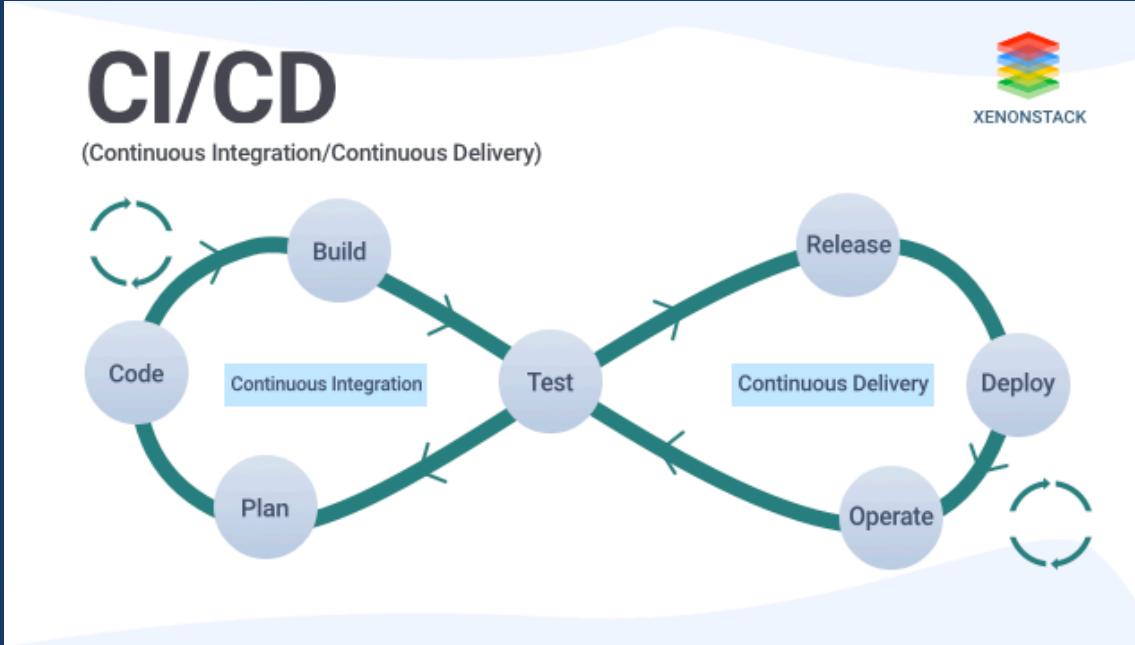
{Secure Systems and Programmable Networks: Tools for the Future}



Università degli Studi di Catania
Dipartimento di Matematica e Informatica
2025



Continuous Integration

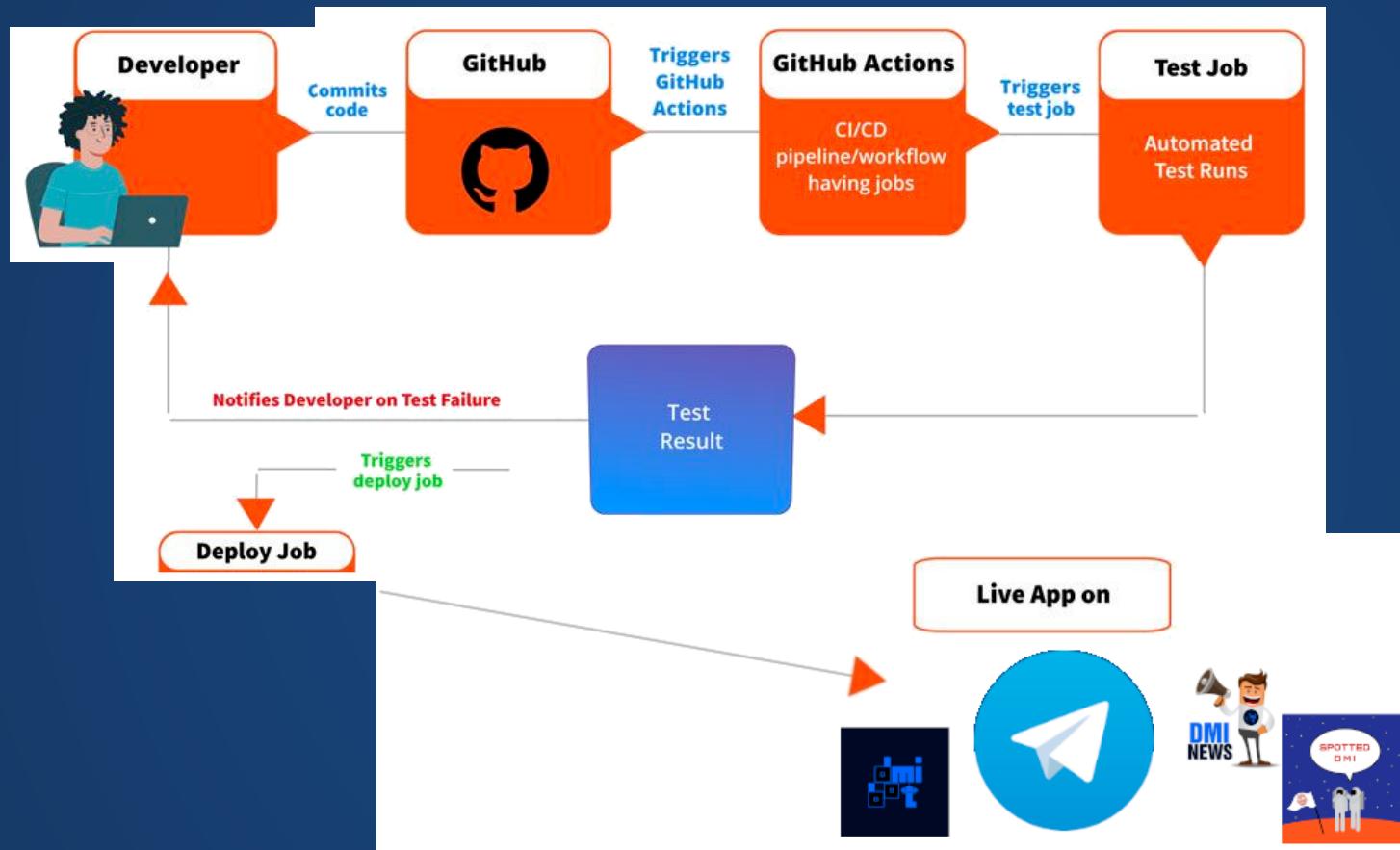


Continuous Delivery

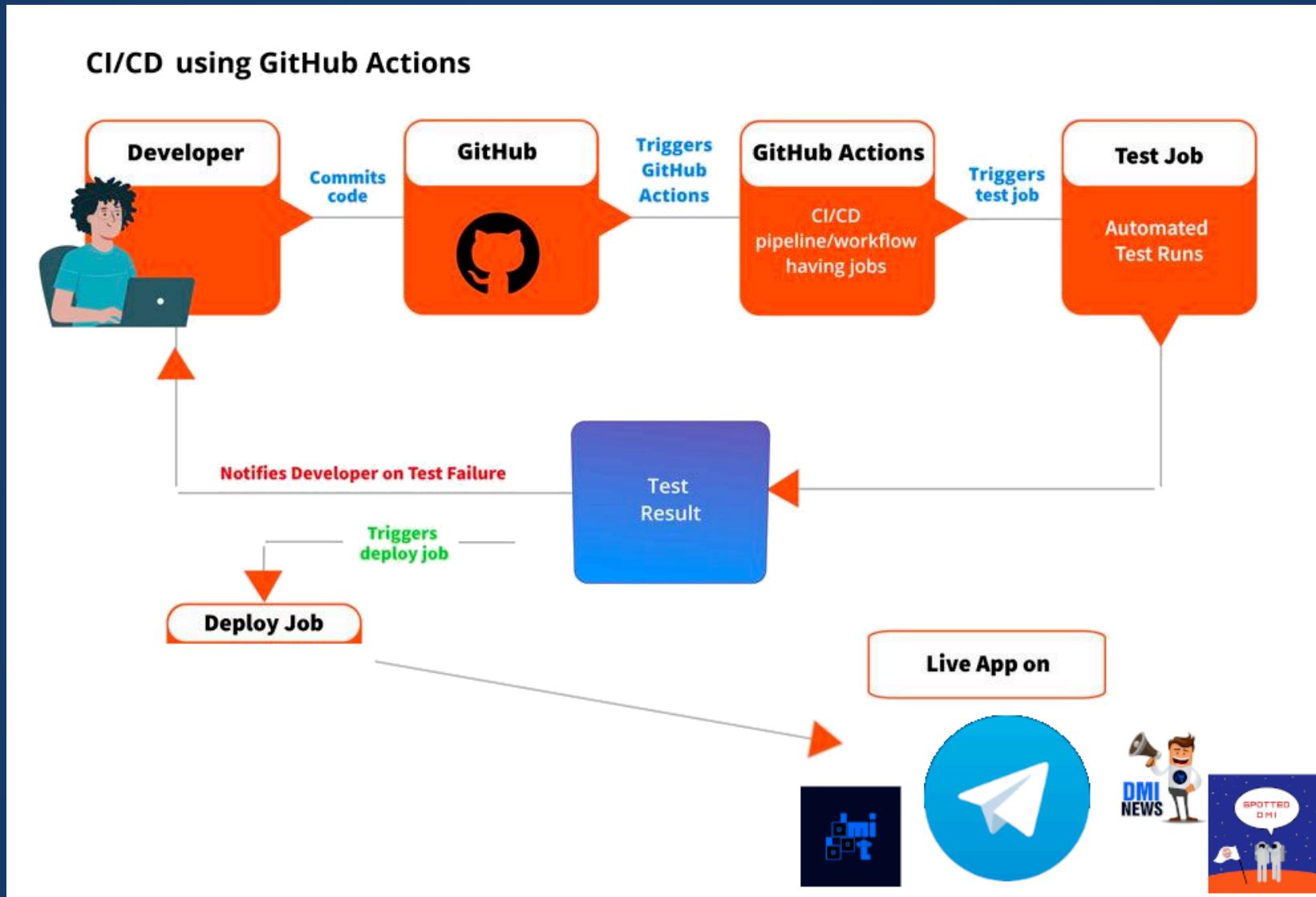
Pipeline



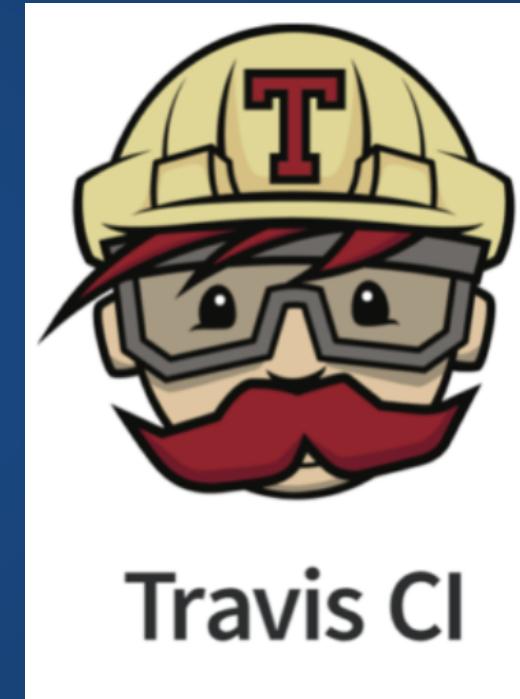
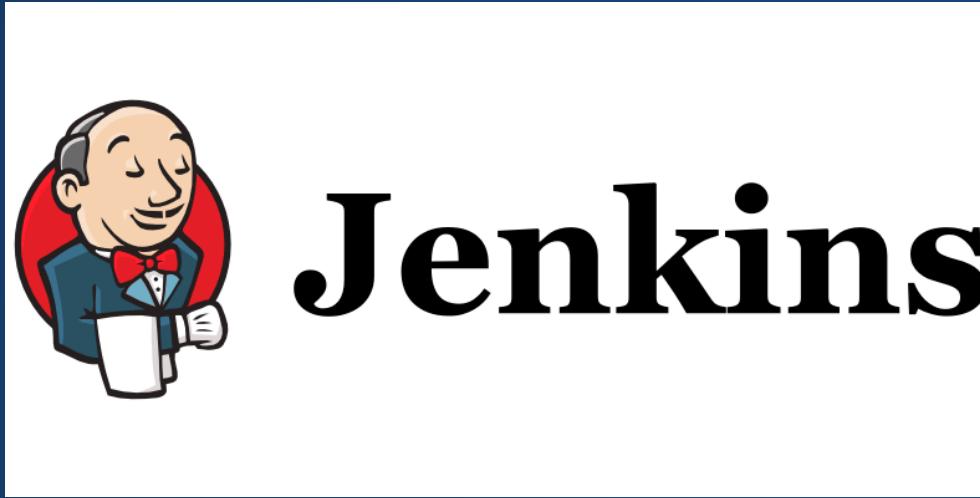
Pipeline -> CI / CD



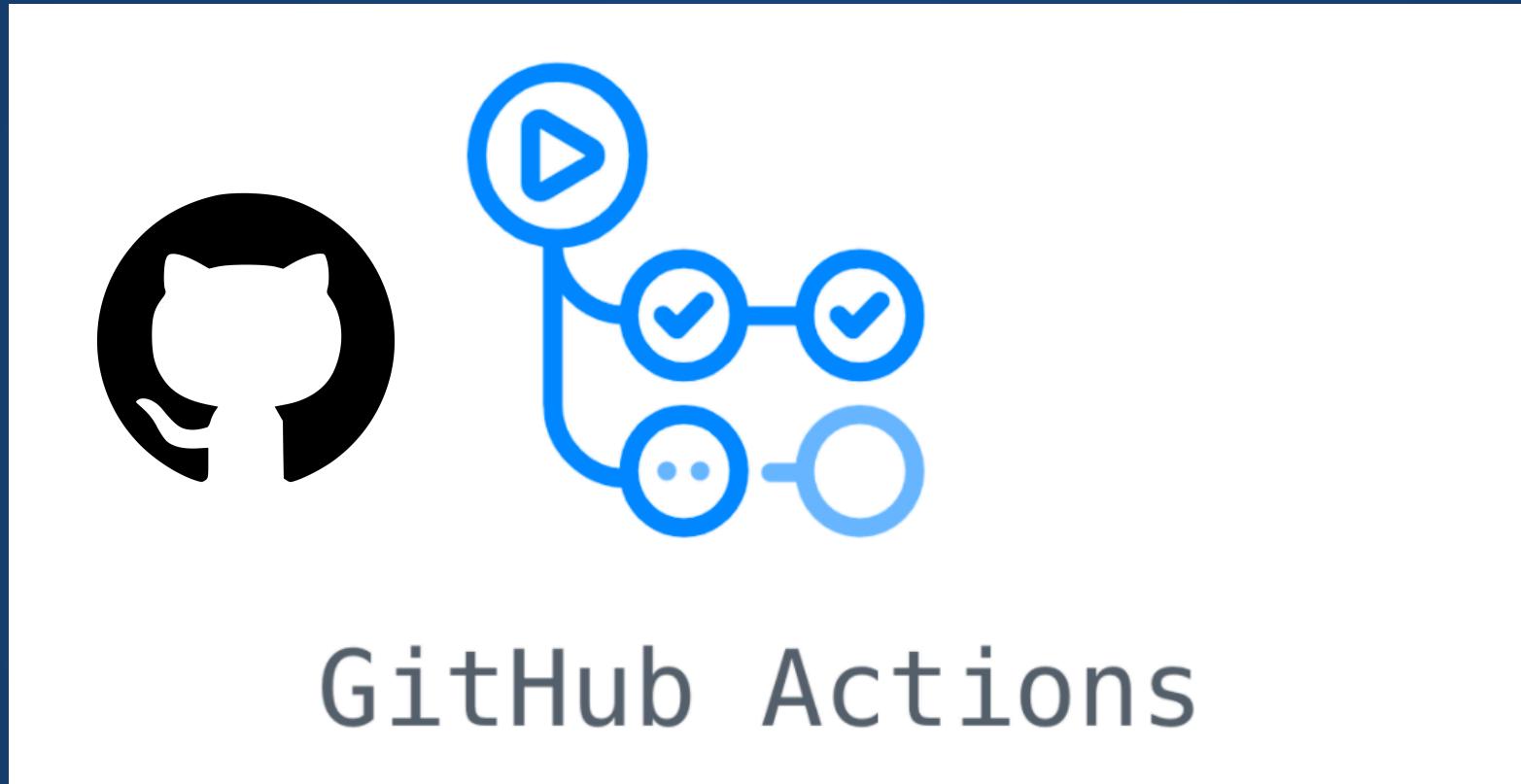
Pipeline -> CI / CD



Pipeline tools



Github Action



2018

Hello World

```
1 name: Hello-World
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10 hello-world-job:
11
12   runs-on: ubuntu-latest
13
14   steps:
15     - name: Hello World
16       run: echo 'Hello World'
```

Hello World (C++)

```
1 name: build-hello-world
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  build-hello-world:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v4
14
15      - name: install g++
16        run: sudo apt install -y g++
17
18      - name: check build
19        run: |
20          g++ hello_world.cpp -o hello_world
21          ./hello_world
22
```

Release

```
1 name: release-hello-world
2
3 on:
4   workflow_dispatch:
5
6 jobs:
7   build-hello-world:
8     permissions: write-all
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v4
12
13      - name: compile and run
14        run: g++ hello_world.cpp -o hello_world_linux
15
16      - name: Create Release
17        env:
18          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
19        run: >-
20          gh release create ${{
21            github.ref_name
22          }} "hello_world_linux"
23          --generate-notes
24          --title "Version ${{
25            github.ref_name
26          }}"
```

Release - cross-platform

```
1 name: release-hello-world
2
3 on:
4   workflow_dispatch:
5
6 jobs:
7   create-release:
8     permissions: write-all
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v4
12      - name: Create Release
13        env:
14          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
15        run: gh release create ${{ github.ref_name }} --generate-notes --title "Version ${{ github.ref_name }}"
16
17 build-hello-world:
18   needs: create-release
19   permissions: write-all
20
21   strategy:
22     matrix:
23       include:
24         - os: ubuntu-latest
25           file_name: hello_world_linux
26         - os: macos-latest
27           file_name: hello_world_mac
28         - os: windows-latest
29           file_name: hello_world_windows.exe
30
31   runs-on: ${{ matrix.os }}
32   name: ${{ matrix.os }}
33
34   steps:
35     - uses: actions/checkout@v4
36     - name: compile
37       run: g++ hello_world.cpp -o ${{ matrix.file_name }}
38
39     - name: Update Release
40       env:
41         GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
42         run: gh release upload ${{ github.ref_name }} "${{ matrix.file_name }}"
```

pipeline checks

```
1 name: build-hello-world
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  build-hello-world:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v4
14
15      - name: install g++
16        run: sudo apt install -y g++
17
18      - name: check build
19        run:
20          - g++ hello_world.cpp -o hello_world
21          - ./hello_world
22
```

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6   cout << "Hello World" << endl;
7
8   return 0;
9 }
```

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5   int x;
6   cout << x << endl;
7
8   cout << "Hello World" << endl;
9
10  return 0;
11 }
```

pipeline checks

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int* p = nullptr;
6
7     cout << *p << endl;
8
9     cout << "Hello World" << endl;
10
11    return 0;
12 }
```

Linter cppcheck

```
1 name: build-hello-world
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  build-hello-world:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v4
14
15      - name: install g++
16        run: sudo apt install -y g++ cppcheck
17
18      - name: run cppcheck
19        run: |
20          cppcheck hello_world.cpp --output-file=report.txt
21          if [ -s report.txt ]; then # if file is not empty
22              cat report.txt
23              exit 1 # let github action fails
24          fi
25
26      - name: check build
27        run: |
28          g++ hello_world.cpp -o hello_world
29          ./hello_world
30
```

Pylint & Pytest

```
1 name: CI
2
3 on:
4   push:
5     branches: [main]
6     paths-ignore:
7       - "README.md"
8       - "docs/**"
9   pull_request:
10    branches: [main]
11    paths-ignore:
12      - "README.md"
13      - "docs/**"
14
15 jobs:
16   test:
17     runs-on: ubuntu-latest
18
19     steps:
20       - uses: actions/checkout@v2
21
22       - name: Set up Python 3.10.0
23         uses: actions/setup-python@v2
24         with:
25           python-version: 3.10.0
26
27       - name: Install dependencies for requirements and testing
28         run: |
29           python -m pip install --upgrade pip
30           if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
31           if [ -f requirements_dev.txt ]; then pip install -r requirements_dev.txt; fi
32
33       - name: Lint with pylint
34         run: pylint src
35
36       - name: Test with pytest
37         run: pytest --cov src tests/ --cov-fail-under=75
```

Reusable workflows

```
1 name: Create and publish a Docker image
2
3 on:
4   workflow_call:
5     inputs:
6       repo_ref: # "author/repository_name" or ${{ github.repository }}
7         required: true
8         type: string
9
10 env:
11   REGISTRY: ghcr.io
12   IMAGE_NAME: ${{ inputs.repo_ref }}
13
14 jobs:
15   build-and-push-image:
16     runs-on: ubuntu-latest
17     permissions:
18       contents: read
19       packages: write
20
21   steps:
22     - name: Checkout repository
23       uses: actions/checkout@v2
24
25     - name: Log in to the Container registry
26       uses: docker/login-action@f054a8b539a109f9f41c372932f1ae047eff08c9
27       with:
28         registry: ${{ env.REGISTRY }}
29         username: ${{ github.actor }}
30         password: ${{ secrets.GITHUB_TOKEN }}
31
32     - name: Extract metadata (tags, labels) for Docker
33       id: meta
34       uses: docker/metadata-action@98669ae865ea3cffbcbaa878cf57c20bbf1c6c38
35       with:
36         images: ${{ env.REGISTRY }}/{{ env.IMAGE_NAME }}
37
38     - name: Build and push Docker image
39       uses: docker/build-push-action@ad44023a93711e3deb337508980b4b5e9bcd5dc
40       with:
41         context: .
42         push: true
43         tags: ${{ steps.meta.outputs.tags }}
44         labels: ${{ steps.meta.outputs.labels }}
```

Reusable workflows

```
1 name: Create and publish a Docker image
2 on:
3   push:
4     branches:
5       - 'main'
6
7 jobs:
8   build:
9     uses: unict-dmi/reusable-workflows/.github/workflows/docker.yml@main
10    with:
11      repo_ref: ${{ github.repository }}
```

<https://github.com/UNICT-DMI/unict-telegram-hub/blob/main/.github/workflows/docker.yaml>

<https://github.com/UNICT-DMI/reusable-workflows/blob/main/.github/workflows/docker.yml>

https://github.com/azerothcore/mod-transmog/blob/master/.github/workflows/core_build.yml

https://github.com/azerothcore/reusable-workflows/blob/main/.github/workflows/core_build_modules.yml

Secrets token

```
1 name: Telegram-Secret-Token
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  hello-world-job:
11
12    runs-on: ubuntu-latest
13
14  steps:
15    - name: Telegram Notify
16      run: >-
17        curl -s --data-urlencode "text=Hello World" "https://api.telegram.org/bot${{ secrets.MY_SECRET_TOKEN }}sendMessage?chat_id=1044000000&text=Hello%20World"
18
```

Secrets token

The screenshot shows the GitHub repository settings page for 'Helias / qd-pipeline'. The 'Settings' tab is highlighted with a red box. On the left, a sidebar lists various repository settings: General, Access, Collaborators, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), Actions, Codespaces, Dependabot, Integrations (GitHub Apps, Email notifications), and Dependabot. The 'Secrets and variables' section is expanded, and its header is also highlighted with a red box. The main content area displays the 'Actions secrets and variables' section, which includes a note about managing reusable configuration data, a description of secrets and variables, and a list of existing secrets. A 'New repository secret' button is visible. A specific secret, 'MY_SECRET_TOKEN', is listed with a lock icon, updated timestamp, and edit/delete buttons.

General

Access

Collaborators

Code and automation

- Branches
- Tags
- Rules
- Actions
- Webhooks
- Codespaces
- Pages

Security

- Code security and analysis
- Deploy keys
- Secrets and variables**

Actions

Codespaces

Dependabot

Integrations

- GitHub Apps
- Email notifications

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Secrets Variables [New repository secret](#)

MY_SECRET_TOKEN	Updated 9 minutes ago		
-----------------	-----------------------	--	--

S Made with Slides.com

Github Pages

The screenshot shows the GitHub Pages settings page for the repository 'test-gh-pages'. The left sidebar contains navigation links for General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), and Integrations (GitHub Apps, Email notifications). The main content area is titled 'GitHub Pages' and states: 'GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.' It displays a message: 'Your site is live at <https://helias.github.io/test-gh-pages/>' and 'Last deployed by Helias 1 minute ago'. A 'Visit site' button and a three-dot menu are shown. Below this, the 'Build and deployment' section shows 'Source' set to 'Deploy from a branch' (main branch) and 'Branch' set to 'main'. It also shows deployment details: 'Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.' At the bottom, there's a 'Save' button. The 'Custom domain' section is present but empty. At the very bottom, there's a checked checkbox for 'Enforce HTTPS' with the note: 'Required for your site because you are using the default domain (helias.github.io)'.

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://helias.github.io/test-gh-pages/>
Last deployed by Helias 1 minute ago

Visit site

Build and deployment

Source

Deploy from a branch

Branch

Your GitHub Pages site is currently being built from the main branch. [Learn more about configuring the publishing source for your site.](#)

main / (root)

Save

Learn how to [add a Jekyll theme](#) to your site.

Your site was last deployed to the [github-pages](#) environment by the [pages build and deployment](#) workflow.
[Learn more about deploying to GitHub Pages using custom workflows](#)

Custom domain

Custom domain

Custom domains allow you to serve your site from a domain other than helias.github.io. [Learn more about configuring custom domains.](#)

Save Remove

Enforce HTTPS

— Required for your site because you are using the default domain (helias.github.io)

example.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int SafeDivide(int a, int b) {
5     cout << "a: " << a << endl;
6     cout << "b: " << b << endl;
7
8     if (b == 0) {
9         return 0; // Return 0 if division by zero
10    }
11
12    return a / b;
13 }
14
15
16 int main() {
17     int a, b;
18
19     cout << "Enter a: ";
20     cin >> a;
21
22     cout << "Enter b: ";
23     cin >> b;
24
25     cout << SafeDivide(a, b) << endl;
26
27     return 0;
28 }
```

Unit Test example

src/math_utils.cpp

```
1 #include "math_utils.h"
2
3 int SafeDivide(int a, int b) {
4     if (b == 0) {
5         return 0; // Return 0 if division by zero
6     }
7
8     return a / b;
9 }
```

src/math_utils.h

```
1 #pragma once
2
3 int SafeDivide(int a, int b);
```

test / math_utils_test.cpp

```
1 #include <gtest/gtest.h>
2 #include "math_utils.h"
3
4 TEST(MathUtilsTest, HandlesZeroDivision) {
5     EXPECT_EQ(SafeDivide(10, 0), 0);
6 }
7
8 TEST(MathUtilsTest, HandlesNormalDivision) {
9     EXPECT_EQ(SafeDivide(10, 2), 5);
10 }
```

Fuzz test example

```
1 #include <cstdint>
2 #include <cstddef>
3 #include "math_utils.h"
4
5 extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
6     if (size < 8) return 0;
7
8     int a = *(reinterpret_cast<const int*>(data));
9     int b = *(reinterpret_cast<const int*>(data + 4));
10
11    SafeDivide(a, b);
12
13    return 0;
14 }
```

Fuzz test example

\$./fuzz math utils

==1453512==ERROR: AddressSanitizer: FPE on unknown address

AddressSanitizer can not provide additional info.

SUMMARY: AddressSanitizer: FPE (build/fuzz math utils+0x1418d4) (BuildId: 0x7f353e000000)

b39d1e6b5479d39aaee3a49ef227df07d8e95b48) in SafeDivide(int, int)

==1453512==ABORTING

MS: 5 CrossOver-InsertRepeatedBytes-ChangeByte-ShuffleBytes-ChangeBinInt-; base unit:

adc83b19e793491b1c6ea0fd8b46cd9f32e592fc

crash-123456789.....

Fuzz test example

```
$ hexdump crash-123456789.....
```

```
00000000 0000 8000 ffff ffff ffff ffff ffff ffff  
00000010 ffff ffff ffff ffff ffff ffff ffff ffff  
00000020 ffff ffff ffff ffff ffff ffff ffff 00ff  
00000030 0000 a300  
00000034
```

HEX signed -> Decimal

80000000 = -2147483648

ffffffffff = -1

```
$ hexdump -v -e "%d, " -e '8/1 "0x%02x, " "\n" ./crash-123456789
```

```
-2147483648, -1, 0x00, 0x00, 0x00, 0x80, 0xff, 0xff, 0xff, 0xff, [...]
```

Fuzz test example

```
1 name: fuzz-test-example
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  fuzz-test:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v4
14
15      - name: install g++
16        run: sudo apt install -y g++
17
18      - name: run build, test and fuzz test
19        run: |
20          mkdir build
21          cd build
22          cmake .. -DCMAKE_CXX_COMPILER=clang++
23          cmake --build . -- -j$(nproc)
24          ./math_utils_test
25          timeout 30 ./fuzz_math_utils || echo "Fuzz test crashed or timed out"
26          if ls crash-* 1> /dev/null 2>&1; then
27            hexdump -v -e '%d, '' -e '8/1 "0x%02x, " "\n"' ./crash-*
28            exit 1 # let github action fails
29        fi
30        echo "Build, test and fuzz test completed successfully"
31
```

Fuzz test example - binary

BIN		DEC	MAX (bit length)
1	$= 2^0 * 1$	= 1	$2^{1-1} = 1$
10	$= 2^1 * 1 + 2^0 * 0$	= 2	
11	$= 2^1 * 1 + 2^0 * 1$	= 3	$2^{2-1} = 3$
100	$= 2^2 * 1 + 2^1 * 0 + 2^0 * 0 = 4$		
111	$= 2^2 * 1 + 2^1 * 1 + 2^0 * 1 = 7$		$2^{3-1} = 7$
1000	$= 2^3 * 1 ...$	= 8	
1111	$= 2^3 * 1 ...$	= 15	$2^{4-1} = 15$

Fuzz test example - binary signed

4 bit

0000 = 0

0001 = 1

...

0111 = 7

1000 = -8

32 bit (int32)

00..00 = 0

00..01 = 1

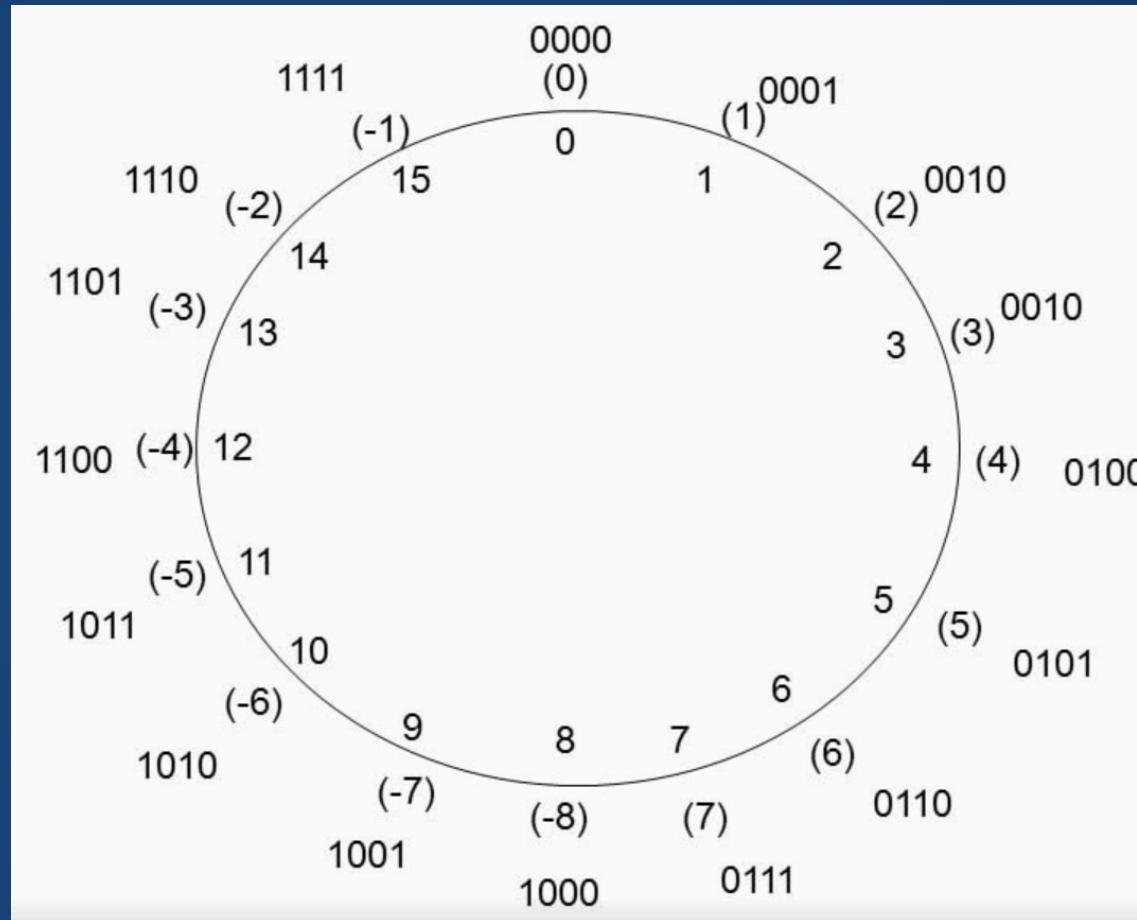
...

01..11 = 2147483647

-8 / -1 = ???? = 8

10..00 = -2147483648

-2147483648 / -1 = ???? = 2147483648



example_workaround.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int SafeDivide(int a, int b) {
5     cout << "a: " << a << endl;
6     cout << "b: " << b << endl;
7
8     if (b == 0) {
9         return 0; // Return 0 if division by zero
10    }
11
12    if (a == -2147483648 && b < 0) {
13        return 2147483647; // Handle overflow case
14    }
15
16    return a / b;
17 }
18
19
20 int main() {
21     int a, b;
22
23     cout << "Enter a: ";
24     cin >> a;
25
26     cout << "Enter b: ";
27     cin >> b;
28
29     cout << SafeDivide(a, b) << endl;
30
31     return 0;
32 }
```