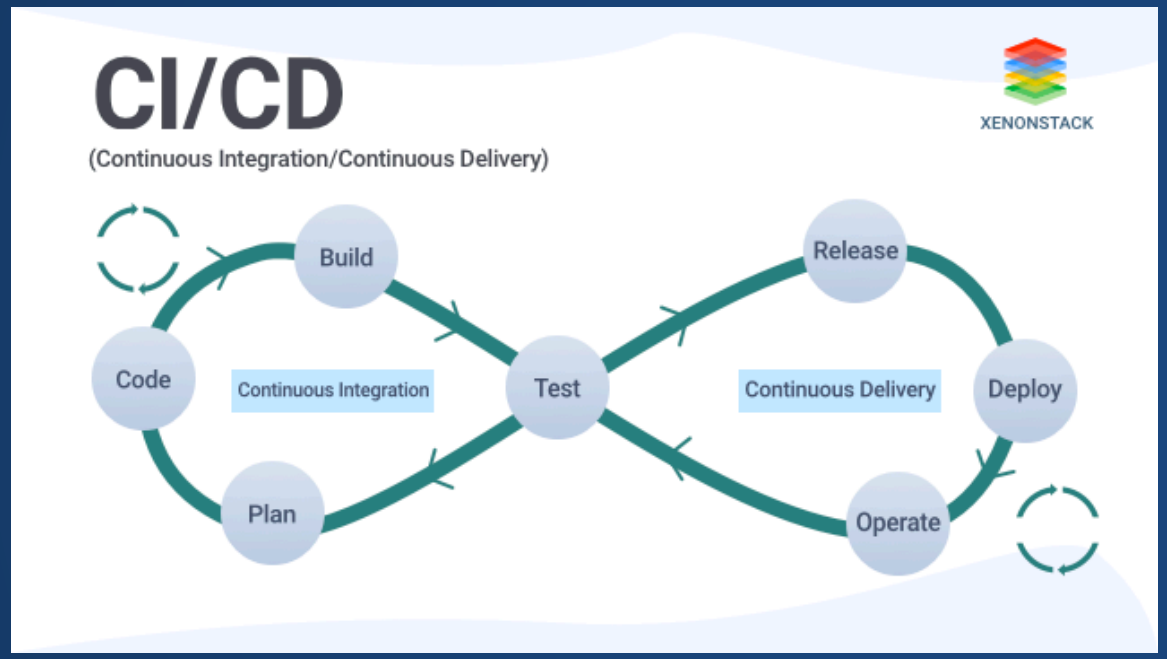
{Secure Systems and Programmable Networks: Tools for the Future}

Università degli Studi di Catania
Dipartimento di Matematica e Informatica
2025

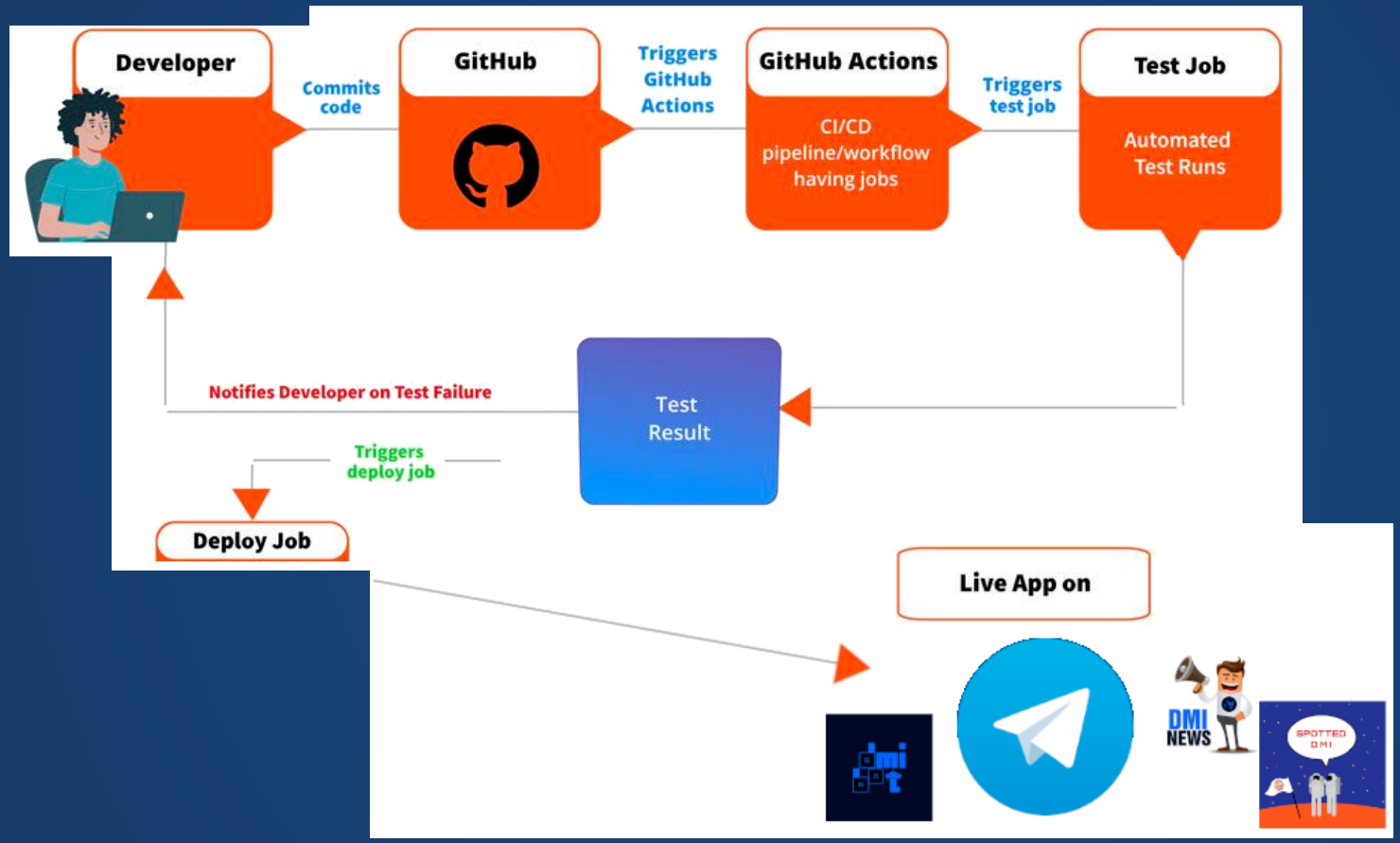
Continuous Integration



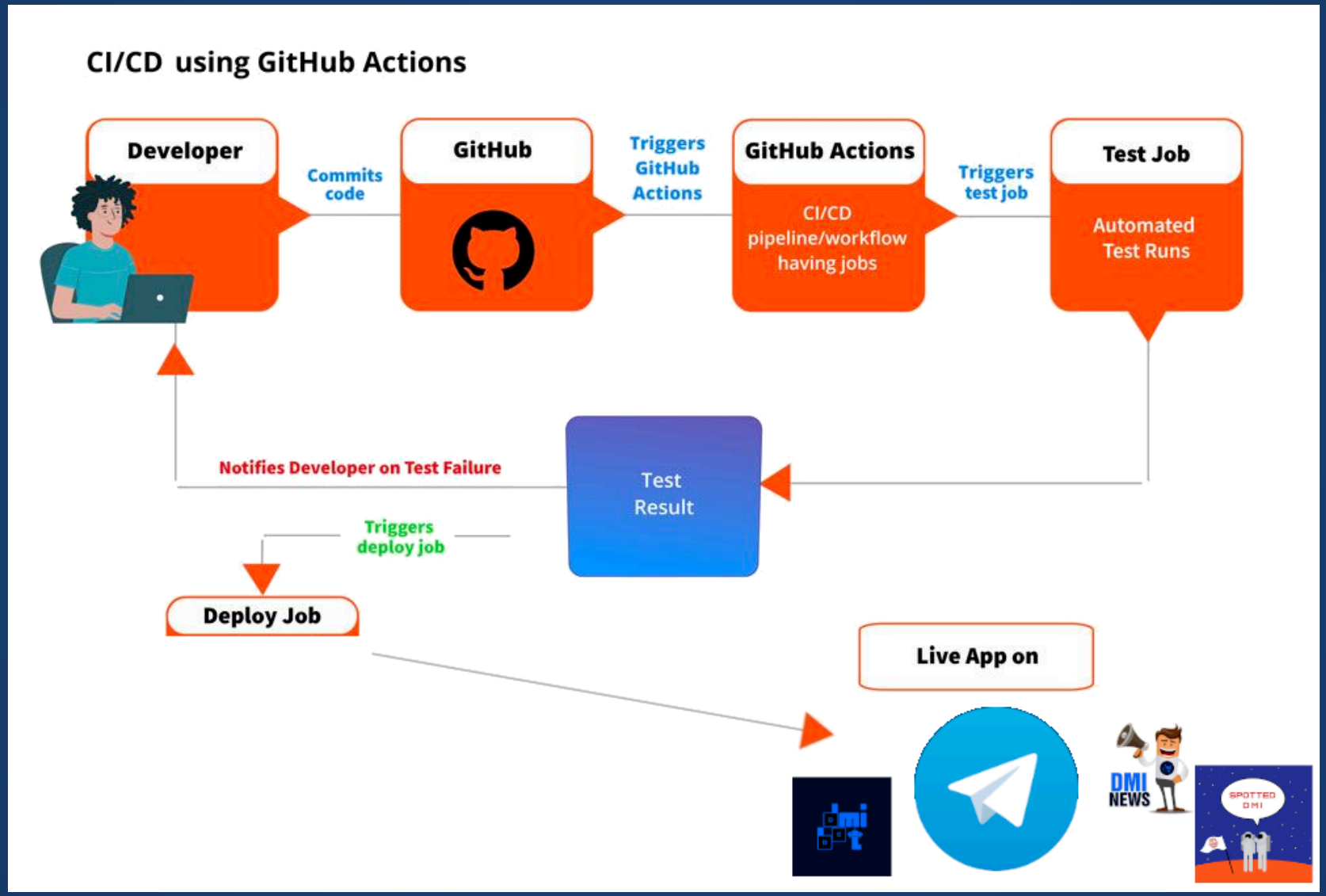
Continuous Delivery

Pipeline

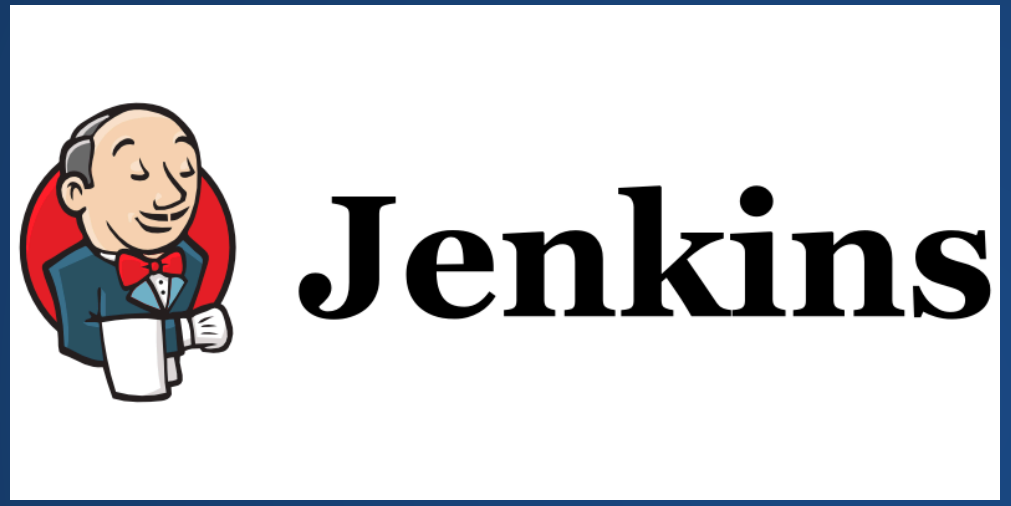
Pipeline -> CI / CD

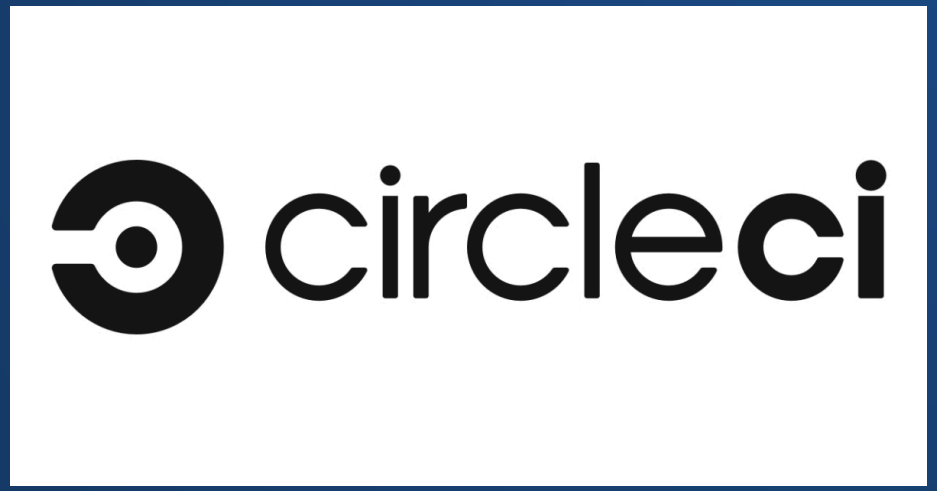


Pipeline -> CI / CD

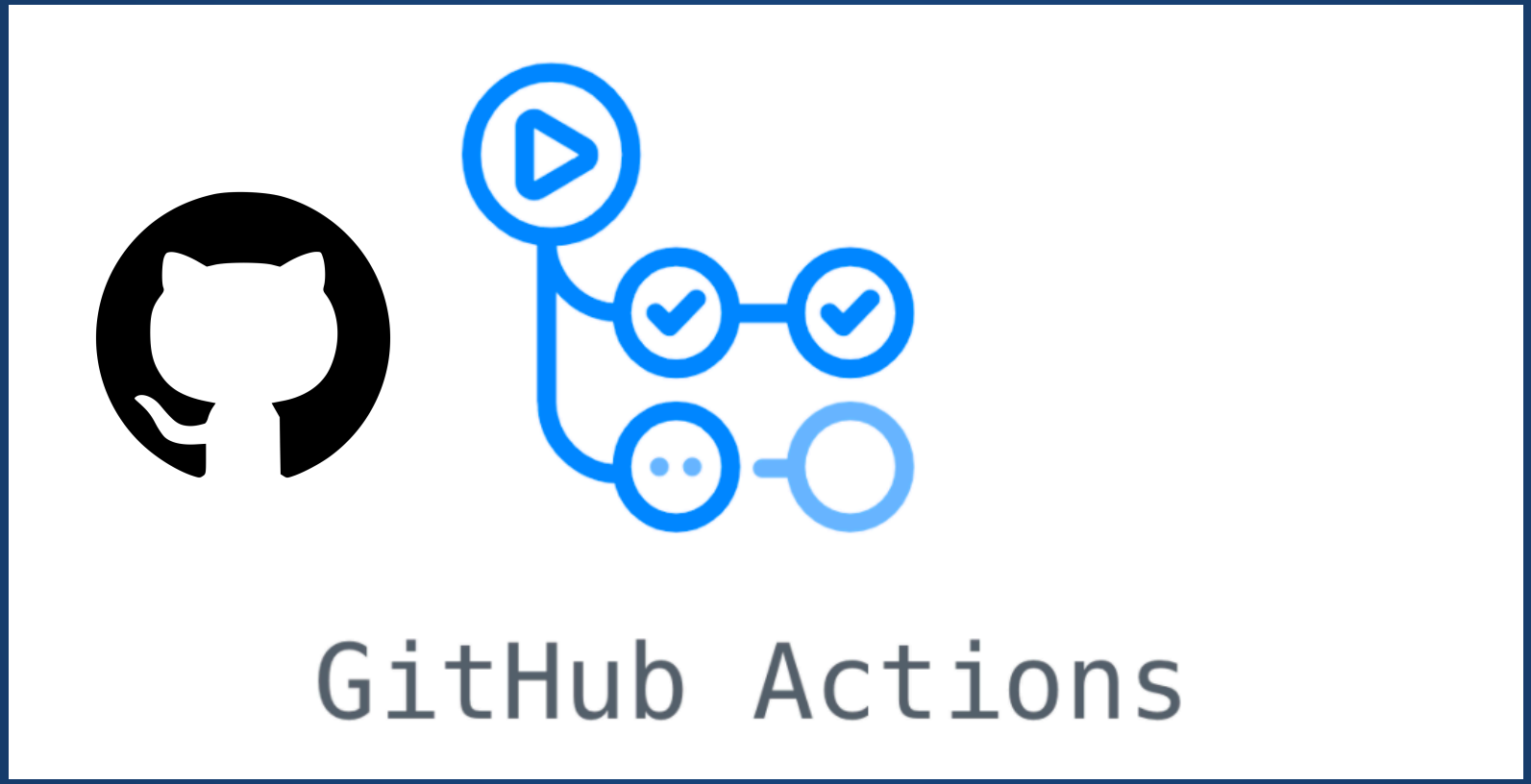


Pipeline tools





Github Action



2018

Hello World

```yaml
name: Hello-World

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

jobs:
  hello-world-job:

    runs-on: ubuntu-latest

    steps:
      - name: Hello World
        run: echo 'Hello World'
```

Hello World (C++)

```yaml
name: build-hello-world

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

jobs:
  build-hello-world:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: install g++
        run: sudo apt install -y g++

      - name: check build
        run: |
          g++ hello_world.cpp -o hello_world
          ./hello_world

```

Release

```yaml
name: release-hello-world

on:
  workflow_dispatch:

jobs:
  build-hello-world:
    permissions: write-all
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: compile and run
        run: g++ hello_world.cpp -o hello_world_linux

      - name: Create Release
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: >-
          gh release create ${{ github.ref_name }}
          "hello_world_linux"
          --generate-notes
          --title "Version ${{ github.ref_name }}"
```

Release - cross-platform

```yaml
name: release-hello-world

on:
  workflow_dispatch:

jobs:
  create-release:
    permissions: write-all
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Create Release
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: gh release create ${{ github.ref_name }} --generate-notes --title "Version ${{ github.ref_name }}"

  build-hello-world:
    needs: create-release
    permissions: write-all

    strategy:
      matrix:
        include:
          - os: ubuntu-latest
            file_name: hello_world_linux
          - os: macos-latest
            file_name: hello_world_mac
          - os: windows-latest
            file_name: hello_world_windows.exe

    runs-on: ${{ matrix.os }}
    name: ${{ matrix.os }}

    steps:
      - uses: actions/checkout@v4
      - name: compile
        run: g++ hello_world.cpp -o ${{ matrix.file_name }}

      - name: Update Release
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: gh release upload ${{ github.ref_name }} "${{ matrix.file_name}}"
```

pipeline checks

```yaml
name: build-hello-world

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

jobs:
  build-hello-world:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: install g++
        run: sudo apt install -y g++

      - name: check build
        run: |
          g++ hello_world.cpp -o hello_world
          ./hello_world

```

```cpp
#include <iostream>
using namespace std;

int main() {

  cout << "Hello World" << endl;

  return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main() {
  int x;
  cout << x << endl;

  cout << "Hello World" << endl;

  return 0;
}
```

pipeline checks

```cpp
#include <iostream>
using namespace std;

int main() {
  int* p = nullptr;

  cout << *p << endl;

  cout << "Hello World" << endl;

  return 0;
}
```

Linter cppcheck

```yaml
name: build-hello-world

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

jobs:
  build-hello-world:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: install g++
        run: sudo apt install -y g++ cppcheck

      - name: run cppcheck
        run: |
          cppcheck hello_world.cpp --output-file=report.txt
          if [ -s report.txt ]; then # if file is not empty
            cat report.txt
            exit 1 # let github action fails
          fi

      - name: check build
        run: |
          g++ hello_world.cpp -o hello_world
          ./hello_world

```

Pylint & Pytest

```yaml
name: CI

on:
  push:
    branches: [main]
    paths-ignore:
      - "README.md"
      - "docs/**"
  pull_request:
    branches: [main]
    paths-ignore:
      - "README.md"
      - "docs/**"

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Set up Python 3.10.0
        uses: actions/setup-python@v2
        with:
          python-version: 3.10.0

      - name: Install dependencies for requirements and testing
        run: |
          python -m pip install --upgrade pip
          if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
          if [ -f requirements_dev.txt ]; then pip install -r requirements_dev.txt; fi

      - name: Lint with pylint
        run: pylint src

      - name: Test with pytest
        run: pytest --cov src tests/ --cov-fail-under=75
```

Reusable workflows

```yaml
name: Create and publish a Docker image

on:
  workflow_call:
    inputs:
      repo_ref: # "author/repository_name" or ${{ github.repository }}
        required: true
        type: string

env:
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ inputs.repo_ref }}

jobs:
  build-and-push-image:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Log in to the Container registry
        uses: docker/login-action@f054a8b539a109f9f41c372932f1ae047eff08c9
        with:
          registry: ${{ env.REGISTRY }}
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Extract metadata (tags, labels) for Docker
        id: meta
        uses: docker/metadata-action@98669ae865ea3cffbcbaa878cf57c20bbf1c6c38
        with:
          images: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}

      - name: Build and push Docker image
        uses: docker/build-push-action@ad44023a93711e3deb337508980b4b5e9bcdc5dc
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

Reusable workflows

```yaml
name: Create and publish a Docker image
on:
  push:
    branches:
      - 'main'

jobs:
  build:
    uses: unict-dmi/reusable-workflows/.github/workflows/docker.yml@main
    with:
      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

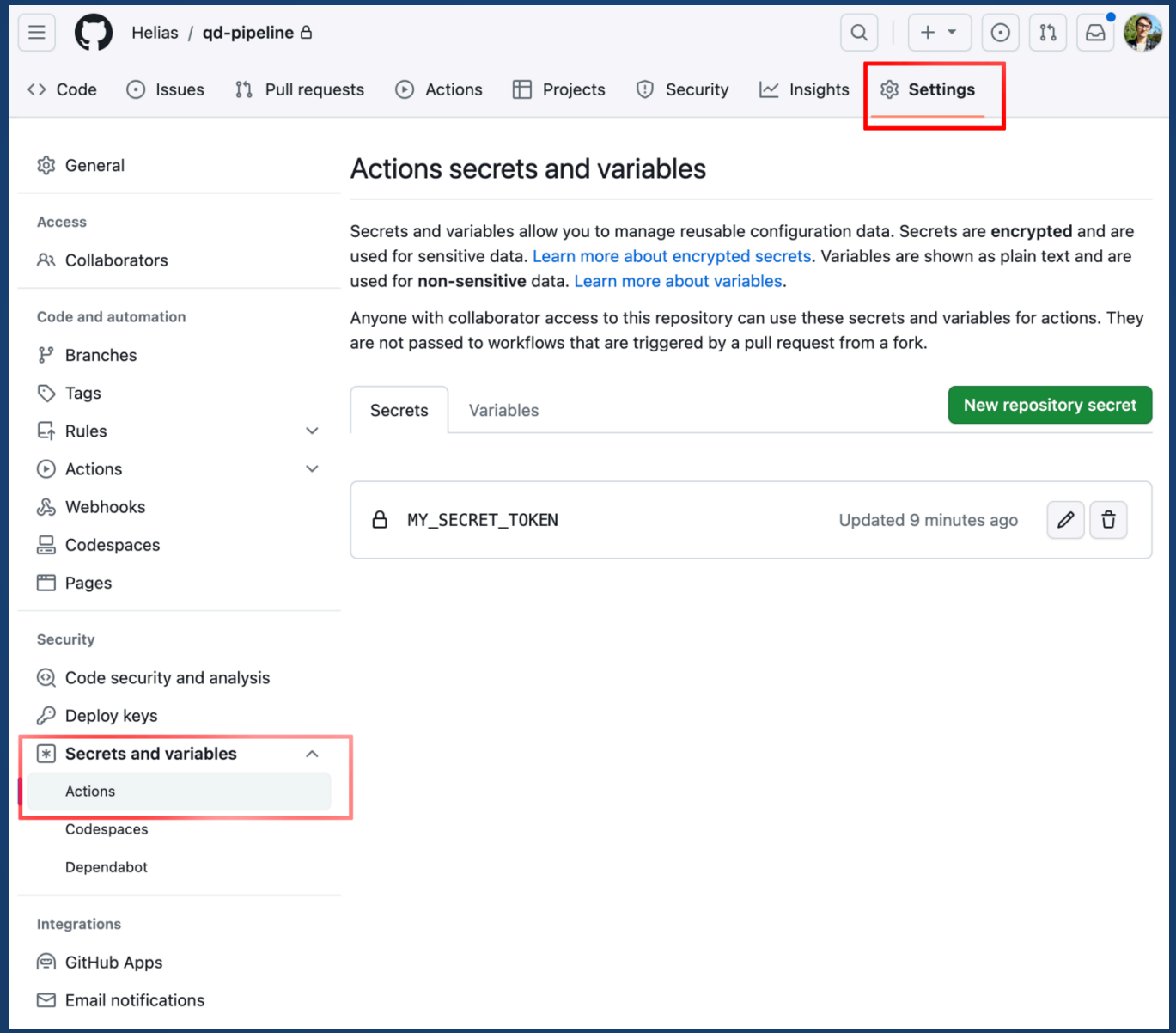
https://github.com/azerothcore/mod-transmog/blob/master/.github/workflows/core_build.yml

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

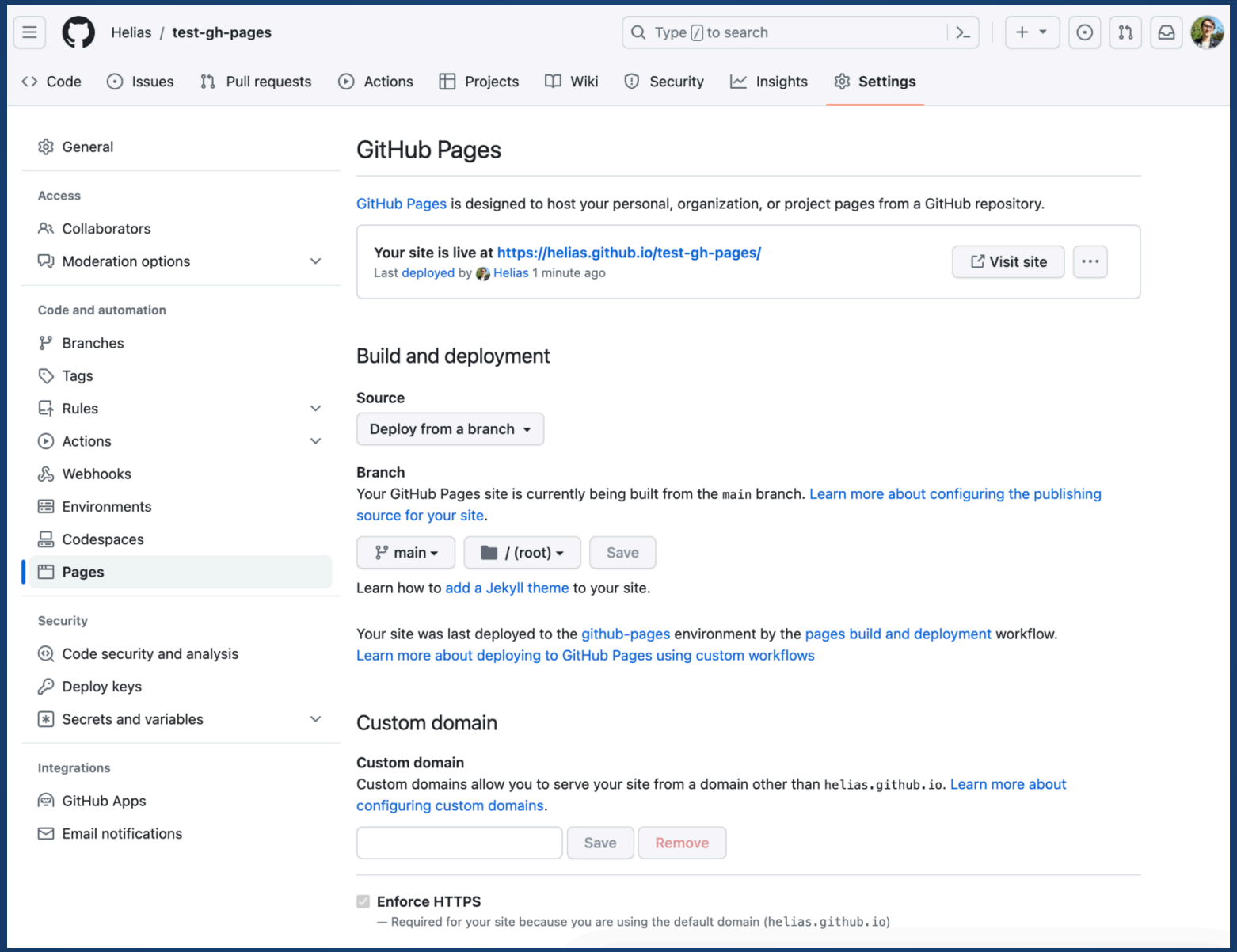
Secrets token

```yaml
name: Telegram-Secret-Token

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

jobs:
  hello-world-job:

    runs-on: ubuntu-latest

    steps:
      - name: Telegram Notify
        run: >-
          curl -s --data-urlencode "text=Hello World ✅"
          "https://api.telegram.org/bot${{ secrets.MY_SECRET_TOKEN }}/sendMessage?chat_id=10440(
```

Secrets token



Github Pages



example.cpp

```cpp
#include <iostream>
using namespace std;

int SafeDivide(int a, int b) {
    cout << "a: " << a << endl;
    cout << "b: " << b << endl;

    if (b == 0) {
        return 0; // Return 0 if division by zero
    }

    return a / b;
}


int main() {
    int a, b;

    cout << "Enter a: ";
    cin >> a;

    cout << "Enter b: ";
    cin >> b;

    cout << SafeDivide(a, b) << endl;

    return 0;
}
```

Unit Test example

src/math_utils.cpp

```cpp
#include "math_utils.h"

int SafeDivide(int a, int b) {
    if (b == 0) {
        return 0; // Return 0 if division by zero
    }

    return a / b;
}
```

src/math_utils.h

```cpp
#pragma once

int SafeDivide(int a, int b);
```

test / math_utils_test.cpp

```cpp
#include <gtest/gtest.h>
#include "math_utils.h"

TEST(MathUtilsTest, HandlesZeroDivision) {
    EXPECT_EQ(SafeDivide(10, 0), 0);
}

TEST(MathUtilsTest, HandlesNormalDivision) {
    EXPECT_EQ(SafeDivide(10, 2), 5);
}
```

Fuzz test example

```cpp
#include <cstdint>
#include <cstddef>
#include "math_utils.h"

extern "C" int LLVMFuzzerTestOneInput(const uint8_t* data, size_t size) {
    if (size < 8) return 0;

    int a = *(reinterpret_cast<const int*>(data));
    int b = *(reinterpret_cast<const int*>(data + 4));

    SafeDivide(a, b);

    return 0;
}
```

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: b39d1e6b5479d39aaee3a49ef227df07d8e95b48) in SafeDivide(int, int)

==1453512==ABORTING

MS: 5 CrossOver-InsertRepeatedBytes-ChangeByte-ShuffleBytes-ChangeBinInt-; base unit: adc83b19e793491b1c6ea0fd8b46cd9f32e592fc

0x0,0x0,0x0,0x80,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xca,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xf

crash-123456789.....

Fuzz test example

$ hexdump crash-123456789.....

```
0000000 0000 8000 ffff ffff ffff ffff ffff ffff
0000010 ffff ffff ffff ffff ffff ffff ffff ffff
0000020 ffff ffff ffff ffff ffff ffff ffff 00ff
0000030 0000 a300
0000034
```

HEX signed -> Decimal

80000000 = -2147483648

ffffffff = -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

```yaml
name: fuzz-test-example

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

jobs:
  fuzz-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: install g++
        run: sudo apt install -y g++

      - name: run build, test and fuzz test
        run: |
          mkdir build
          cd build
          cmake .. -DCMAKE_CXX_COMPILER=clang++
          cmake --build . -- -j$(nproc)
          ./math_utils_test
          timeout 30 ./fuzz_math_utils || echo "Fuzz test crashed or timed out"
          if ls crash-* 1> /dev/null 2>&1; then
            hexdump -v -e '"%d, "' -e '8/1 "0x%02x, " "\n"' ./crash-*
            exit 1 # let github action fails
          fi
          echo "Build, test and fuzz test completed successfully"

```

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

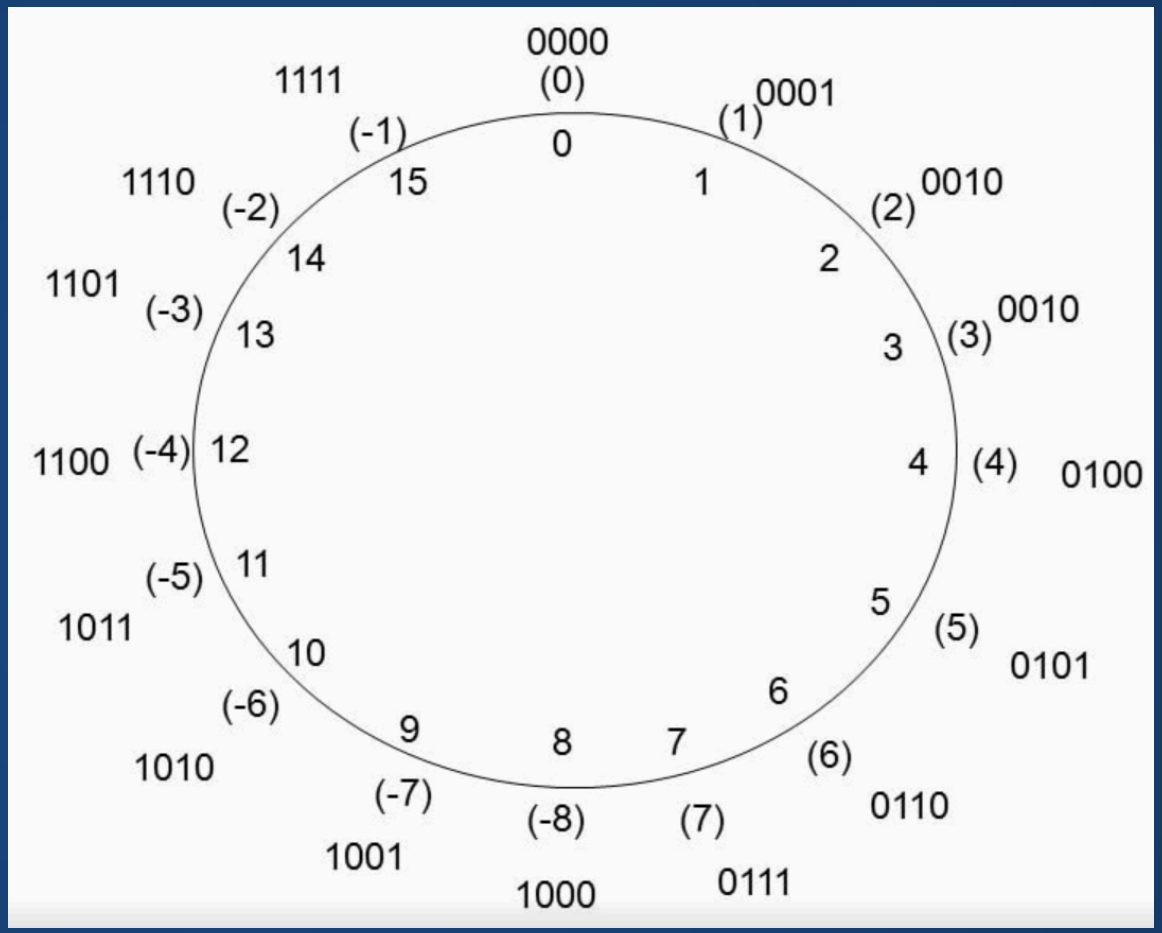
0000 = 0

0001 = 1

...

0111 = 7

1000 = -8



32 bit (int32)

00..00 = 0

00..01 = 1

...

01..11 = 2147483647

10..00 = -2147483648

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

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

example_workaround.cpp

```cpp
#include <iostream>
using namespace std;

int SafeDivide(int a, int b) {
    cout << "a: " << a << endl;
    cout << "b: " << b << endl;

    if (b == 0) {
        return 0; // Return 0 if division by zero
    }

    if (a == -2147483648 && b < 0) {
        return 2147483647; // Handle overflow case
    }

    return a / b;
}


int main() {
    int a, b;

    cout << "Enter a: ";
    cin >> a;

    cout << "Enter b: ";
    cin >> b;

    cout << SafeDivide(a, b) << endl;

    return 0;
}
```