FreeFEM CI/CD and CMake support

FreeFEM days 2025 - 2025/12/18

Simon Legrand - Inria Paris

What is CI/CD?

CI: Process that frequently checks the workability of a source code
CD: Process that frequently delivers new versions of a software

What is CI/CD?

CI: Process that frequently checks the workability of a source code
CD: Process that frequently delivers new versions of a software

Why ?

- Changes don't break existing code
- Offer packages ready to use

What is CI/CD?

CI: Process that frequently checks the workability of a source code
CD: Process that frequently delivers new versions of a software

Why ?

- Changes don't break existing code
- Offer packages ready to use

How ?

- Frequently build the application and run tests
- Package it for a variety of OS/configurations

What is CI/CD?

CI: Process that frequently checks the workability of a source code
CD: Process that frequently delivers new versions of a software

Why ?

- Changes don't break existing code
- Offer packages ready to use

How ?

- Frequently build the application and run tests
- Package it for a variety of OS/configurations

Common tools

- Jenkins
- Gitlab CI
- Github Actions
- ... many more

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

One workflow file per configuration type:

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

One workflow file per configuration type:

- No MPI:
  - Minimal (no additional dependencies)
  - Sequential (`–enable-download`)

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

One workflow file per configuration type:

- No MPI:
  - Minimal (no additional dependencies)
  - Sequential (`–enable-download`)
- With MPI and dependencies (through PETSc)
  - Full OpenMPI (external)
  - Full MPICH (from PETSc)
  - Full MS-MPI (external)

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

One workflow file per configuration type:

- No MPI:
  - Minimal (no additional dependencies)
  - Sequential (`–enable-download`)
- With MPI and dependencies (through PETSc)
  - Full OpenMPI (external)
  - Full MPICH (from PETSc)
  - Full MS-MPI (external)

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

One workflow file per configuration type:

- No MPI:
  - Minimal (no additional dependencies)
  - Sequential (`–enable-download`)
- With MPI and dependencies (through PETSc)
  - Full OpenMPI (external)
  - Full MPICH (from PETSc)
  - Full MS-MPI (external)

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

Based on Github Actions.

Workflows located in `.github/workflows/`, triggered by:

- **push** on develop/master
- **pull requests** on develop.

One workflow file per configuration type:

- No MPI:
  - Minimal (no additional dependencies)
  - Sequential (`–enable-download`)
- With MPI and dependencies (through PETSc)
  - Full OpenMPI (external)
  - Full MPICH (from PETSc)
  - Full MS-MPI (external)

minimal.yml

```yaml
linux:
  strategy:
    matrix:
      version: [22.04, 24.04]
      cfg:
      - {opts: --enable-debug}
      - {opts: --enable-optim --enable-generic}
...
  runs-on: ubuntu-${{ matrix.version }}
  steps:
    - name: Configure
      run: |
        autoreconf -i
        ./configure --without-mpi ${{ matrix.cfg.opts }}
    - name: Build
      run: make -j 4
    - name: Check
      run: |
        make check -i
        ./etc/actions/failed_tests_logs.sh
```

FreeFEM CI

		Minimal	Sequential	Full OpenMPI	Full MPICH	Full MS-MPI
Linux	22.04	✅	✅	✅	✅	N/A
	24.04	✅	✅	✅	✅	N/A
MacOS	14	✅	✅	×(broken OpenMPI in brew)	✅	N/A
	15	✅	✅	×(broken OpenMPI in brew)	✅	N/A
Windows	Server 2022	✅	✅	N/A	N/A	✅

FreeFEM CD

FreeFEM CD

Triggered by `v*.*` tags

FreeFEM CD

Triggered by `v*.*` tags

Currently:

- DEB packaging automated (Full MPICH) in PR
- EXE and DMG manually uploaded

FreeFEM CD

Triggered by `v*.*` tags

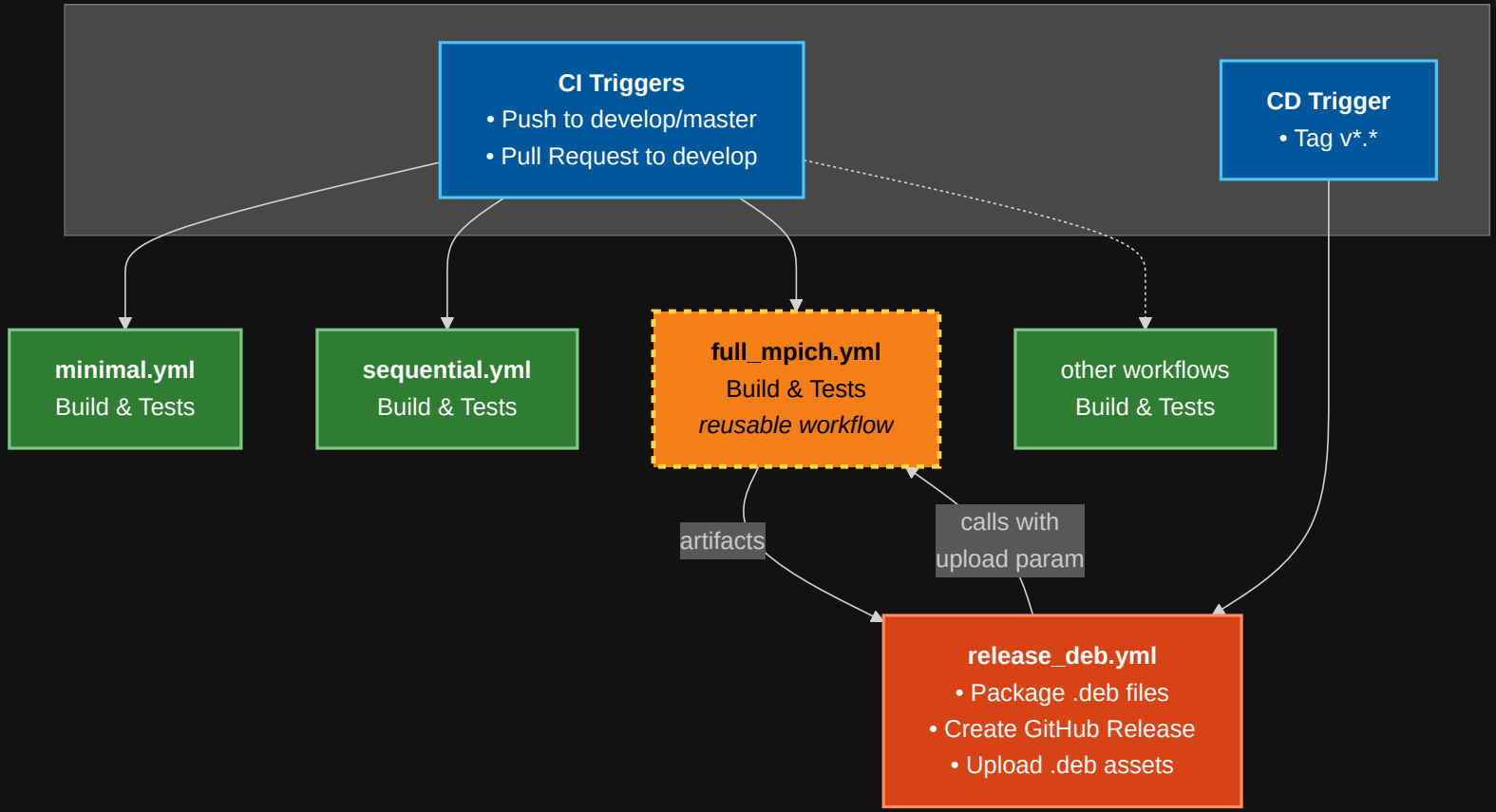
Currently:

- DEB packaging automated (Full MPICH) in PR
- EXE and DMG manually uploaded

Goal is full automation BUT:

- Signing and permissions problems on MacOS
- I didn’t look at EXE yet.

FreeFEM CI/CD workflow graph



CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

- `find_package()` with config files
- `FetchContent` for downloading dependencies

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

- `find_package()` with config files
- `FetchContent` for downloading dependencies

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

- `find_package()` with config files
- `FetchContent` for downloading dependencies

Performance

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

- `find_package()` with config files
- `FetchContent` for downloading dependencies

Performance

- Fast configuration (compiled tool vs shell scripts)

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

- `find_package()` with config files
- `FetchContent` for downloading dependencies

Performance

- Fast configuration (compiled tool vs shell scripts)

Becoming Standard

CMake support

I like autotools… as a user. As a developer, I don’t like macros.

Why CMake ?

Modern Workflow

- Native IDE support (VS Code, XCode, Visual Studio, …)
- Out-of-source builds by default

All-in-One

- Build + Test (CTest) + Package (CPack) + Install

Dependency Management

- `find_package()` with config files
- `FetchContent` for downloading dependencies

Performance

- Fast configuration (compiled tool vs shell scripts)

Becoming Standard

- Used by: LLVM, Qt, KDE, OpenCV, …

CMake support

Also a refactoring opportunity.

CMake support

Also a refactoring opportunity.

Users

No more

```
FreeFem++
FreeFem++-nw
FreeFem++-mpi
```

A single entry point

```
FreeFem++
```

linked with available dependencies.

CMake support

Also a refactoring opportunity.

Users

No more

```
FreeFem++
FreeFem++-nw
FreeFem++-mpi
```

A single entry point

```
FreeFem++
```

linked with available dependencies.

Developers

- PETSc as a single dependency
- Code deduplication (plugins seq/mpi)

Cmake support

Currently a work in progress

Working backbone:

```
configure → compile → package
                    \
                     \ → install
```

- No MPI
- No optional external dependency
- Packaging only in DEB and DMG (CLI)

Thank you for you attention

My fork on GitHub