Contributing to Hammer: An Introduction

Kyle Beyer

University of Michigan

beykyle@umich.edu

May 12, 2022

Overview

Introduction to Hammer

Hammer is a research framework for radiation transport solvers. It is:

- student driven
- written in object-oriented C++17
- application agnostic
- lightweight
- parallel

Package Manager

Use a package manager, unless you like your life to be hard

- Debian/Ubuntu: use `apt`
- MacOS: install `brew`
- Windows: install `WSL` - then install `apt`

git

`git` is version control software. If you're unfamilar, use this tutorial.

Installation syntax depends on your package manager. With `apt`:

```
> sudo apt-get install git
```

If you insist on installing manually,
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git.

Compiler

A compiler that supports C++17 core language features and library features is necessary. For the common compilers this means:

- `g++`: 7.0
- `clang`: 6.0
- `MSVC`: 19.14
- `AppleClang`: 10

With `apt` it's as easy as:

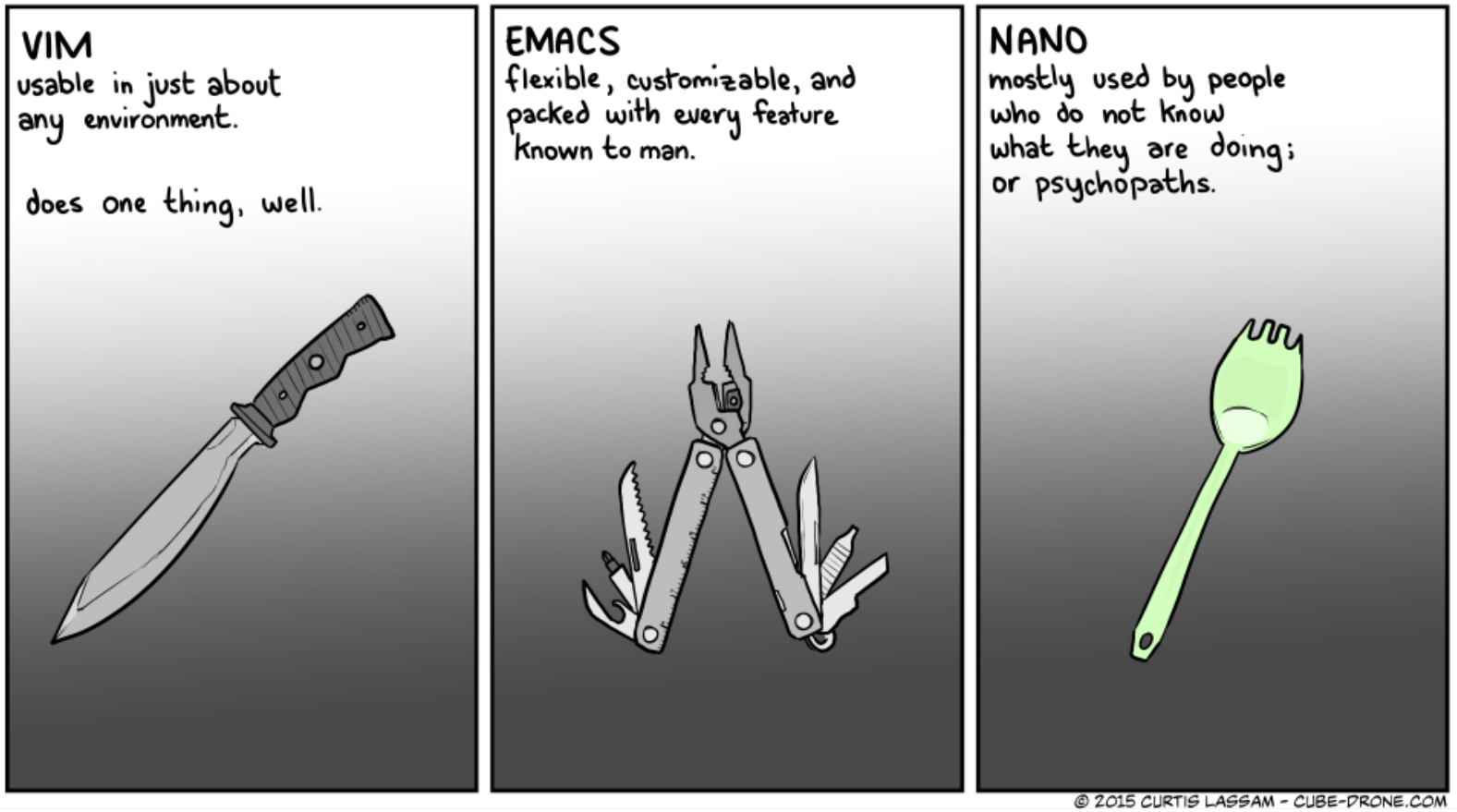
```
> sudo apt-get install g++
```

CMake

We use CMake as a cross platform build system. It can interface with multiple other build systems, like Make, or Ninja.

```
> sudo apt-get install make cmake
```

Text Editor


VIM
usable in just about any environment.
does one thing, well.
EMACS
flexible, customizable, and packed with every feature known to man.
NANO
mostly used by people who do not know what they are doing; or psychopaths.
© 2015 CURTIS LASSAM - CUBE-DRONE.COM

Other tools

- doxygen
- clang-format

```
> sudo apt-get install doxygen clang-format
```

If you have installation or setup trouble, consult the contributing page and the docs, for setup on specific IDEs, or post in the help channel on slack. Most of these tools, as well as several IDEs, are on CAEN windows and RHEL remote desktops, so you can always use those.

Clone the repository

```
> cd
> mkdir demo
> cd demo
> git clone git@github.com:umcpt/mc-hammer-2.git

Cloning into 'mc-hammer-2'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 73802 (delta 5), reused 7 (delta 0), pack-reused 73770
Receiving objects: 100% (73802/73802), 921.70 MiB | 11.72 MiB/s, done
Resolving deltas: 100% (59832/59832), done.

> cd mc-hammer-2
> ls

CMakeLists.txt    data   examples  README.md   src
CONTRIBUTING.md   docs   extern     scripts     test

> ls src

CMakeLists.txt   geometry   monte_carlo   simulation   sn_solver   utility
```

Build the code

```
> mkdir build
> cd build
> cmake ..
> make -j4
> make doc
```

Create a feature branch

A branch is a version of the code. They are 'checked out' from an existing branch, changed, and then can be 'merged' back in. The branch types we use are

- feature-<>
- test-<>
- bugfix-<>
- refactor-<>

New features are documented in Github issues.

```
> git checkout -b feature-lethargy-calculator
```

Implement the feature

In src/utility/lethargy.hpp:

```
namespace util {
/// @brief  Used to calculate the lethargy of an energy
///         against a reference energy
class LethargyCalculator {
private:
  const double ref_energy;

public:
  LethargyCalculator(double ref_energy)
  : ref_energy(ref_energy) {}

  ///@brief  Calculates lethargy given energy, wrt to ref_energy
  double lethargy(double energy);
};
} // namespace util
```

Implement the feature

In `src/utility/lethargy.cpp`:

```
#include "utility/lethargy.hpp"

#include <cmath>

double util::LethargyCalculator::lethargy(double energy) {
  return log(ref_energy / energy);
}
```

Commit changes

```
> git status

On branch feature-lethargy-calculator
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        src/utility/lethargy.cpp
        src/utility/lethargy.hpp

> git add src/utility/lethargy.*
> git status

On branch feature-lethargy-calculator
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        new file:   src/utility/lethargy.cpp
        new file:   src/utility/lethargy.hpp

> git commit -m "Added the LethargyCalculator class"

[feature-lethargy-calculator 31174ddf1] added the LethargyCalculator class
 2 files changed, 20 insertions(+)
 create mode 100644 src/utility/lethargy.cpp
 create mode 100644 src/utility/lethargy.hpp

> git status

On branch feature-lethargy-calculator
nothing to commit, working tree clean
```

Create a test branch

```
> git checkout -b test-lethargy-calculator
```

Implement the tests

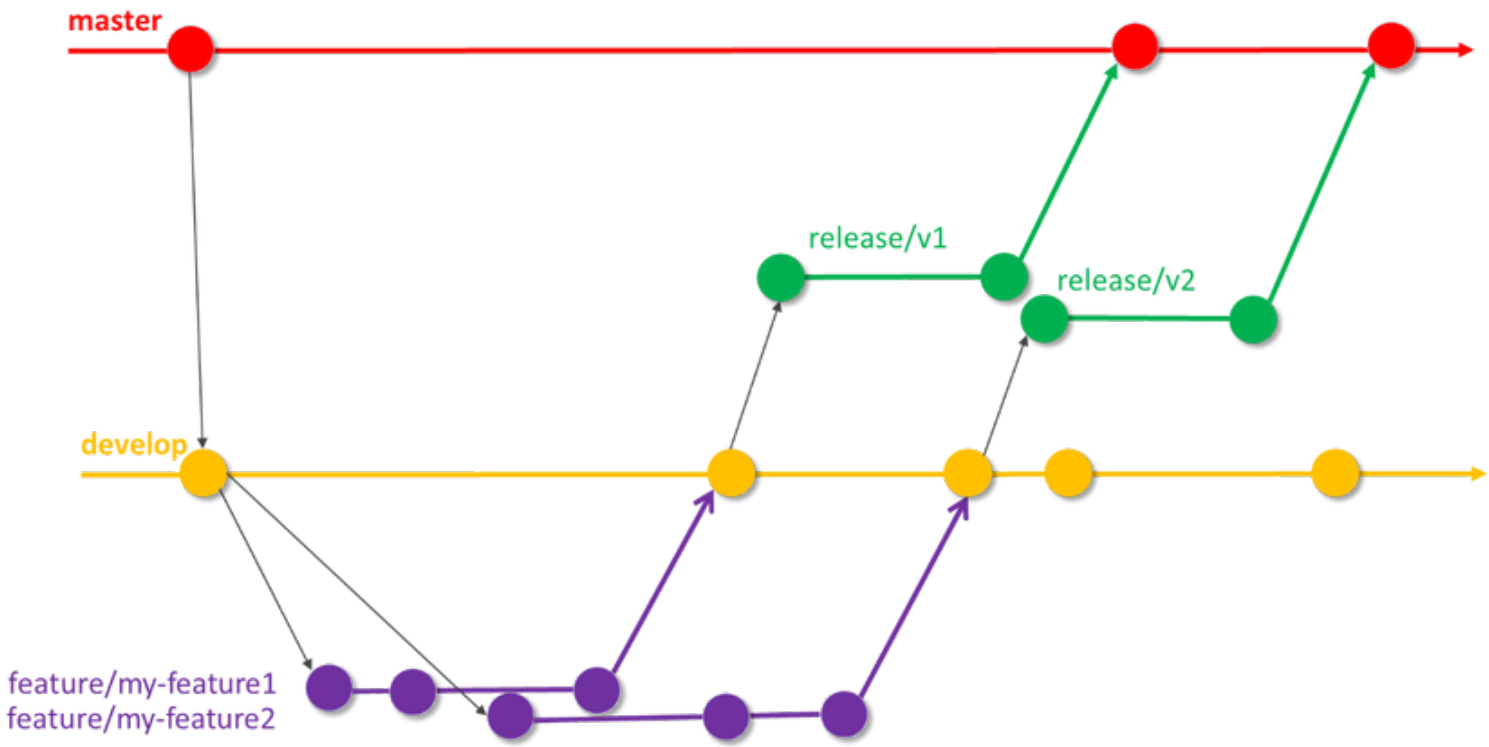
In `test/utility/test_lethargy.cpp`:

```
#include <catch2/catch.hpp>

#include "utility/lethargy.hpp"

TEST_CASE("LethargyCalculator", "[lethargy]") {

  auto myLethargyCalculator = util::LethargyCalculator(1E6);
  REQUIRE( myLethargyCalculator.lethargy(1.0) == Approx(13.81551) )
}
```

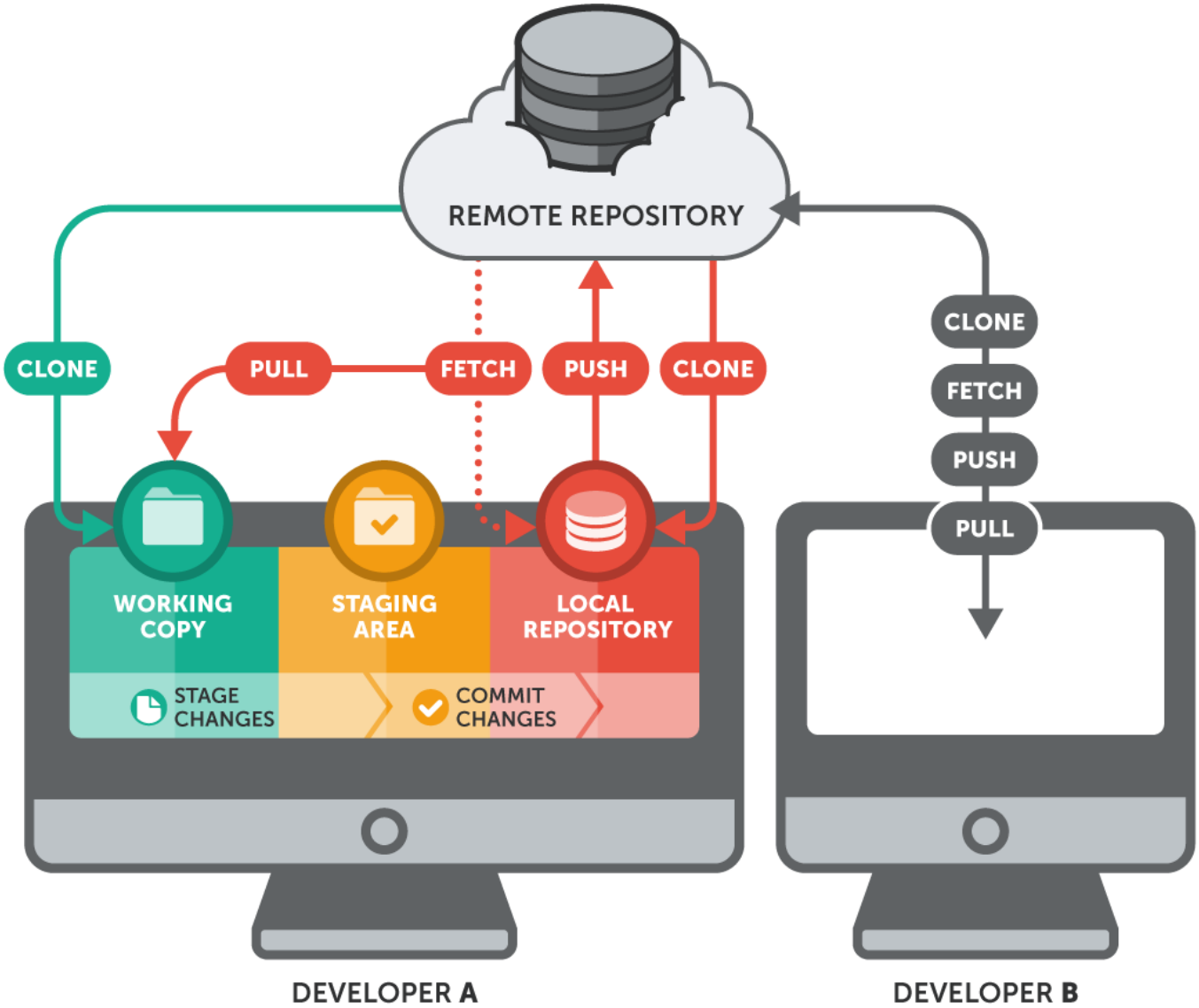
Merge tests into feature branch

```
> git add test/utility/test_lethargy.cpp
> git commit -m "added simple test for LethargyCalculator"

[test-lethargy-calculator 0027fa4fd] added simple test for LethargyCalculator
 1 file changed, 9 insertions(+)
 create mode 100644 test/utility/test_lethargy.cpp

> git checkout feature-lethargy-calculator

Switched to branch 'feature-lethargy-calculator'

> git merge test-lethargy-calculator

Updating 31174ddf1..0027fa4fd
Fast-forward
 test/utility/test_lethargy.cpp | 9 +++++++++
 1 file changed, 9 insertions(+)
 create mode 100644 test/utility/test_lethargy.cpp
```

Branching


master
release/v1
release/v2
develop
feature/my-feature1
feature/my-feature2

Branching


REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
CLONE
FETCH
PUSH
PULL
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
DEVELOPER B

Push to remote branch

This is where the demo stops being interactive!

Now we can push our code, creating a new remote, or upstream, branch with the same name.

```
> git push --set-upstream origin feature-lethargy-calculator
```

Create a pull request

Pull requests are how we merge code into the develop branch.
PRs should summarize:

- what Github issue the PR closes
- the feature/bugfix that was implemented
- the unit testing that was done
- any other changes made to the code

While you're working on your feature, you can create a draft pull-request.
When you feel it is ready, mark it "ready for review".
Make sure you pull from develop early and often!

Code review

Code reviews are an important tool to maintain code quality.

Once you submit a PR, you will likely be asked to make changes, perhaps even several rounds of changes, before you are done with it.

Reviewers should be thorough; checking for code quality, correctness, and readability. Feature developers should respond to comments promptly.

Helpful Links

- setting up an SSH key-pair on github to clone the repo
- contributing page
- xcode setup
- visual studio setup

Current projects

- Constructive solid geometry: cell complements, lattices, etc.
- Photon physics: Incoherent/coherent scatter, photo-electric effect, thick-target Bremstrahlung, pair-production
- Neutron physics: (n,γ), fission, inelastic scatter ENDF laws, thermal scatter on materials, unresolved resonance treatment
- Machine learning to optimize variance reduction