Benchmark tooling for common models and operations

Google Summer of Code 2023

NUMFOCUS FluxML Project Proposal

skyleaworlder

skyleaworlder@outlook.com

Project Introduction

FluxML is a deep learning stack built in 100% Julia code, which aims to provide high-speed and light-weight abstraction of GPU, auto-differentiation and deep learning model composites.

FluxML adheres well to Julia's best-practice and design patterns, By splitting the different layers of the deep learning framework into different repositories, FluxML has benefits on code reuse across the Julia ML ecosystem. However, the lack of "mono-repo" architecture has led to the increased complexity in benchmark system design, presenting some difficulties for FluxML’s developers.

This proposal will introduce a specific benchmarking system to adapt to the current requirements of the FluxML community. The system includes GitHub Actions workflows that are triggered by specific events, a FluxML-specific benchmarking tool and a repository for benchmarking code.

Related Technologies Introduction

1. BenchmarkTools.jl / PkgBenchmark.jl / BenchmarkCI.jl

BenchmarkTools.jl is a Julia package to make running benchmarks easier. “BenchmarkGroup” and “@benchmark” are widely used in various projects.

PkgBenchmark.jl is based on BenchmarkTools.jl, which provides "benchmarkpkg" to run benchmarks and also provides various useful structures to store benchmark results such as "BenchmarkResult" and "BenchmarkJudgement". PkgBenchmark.jl uses "Module" as input to run benchmarks. However, it requires the module to provide a "SUITE" variable defined in "benchmark/benchmarks.jl". [17]

BenchmarkCI.jl is listed in the idea list, which is based on PkgBenchmark.jl and can be easily called in GitHub Actions or some other CI platforms with less code and less configuration. It is easier to use due to its higher level of encapsulation, but inherits all the shortcomings of PkgBenchmark.jl. [12]

2. GitHub Actions

GitHub Actions is a DevOps platform embedded in GitHub, which helps developers to automate software workflows, now with world-class CI/CD. Build, test, and deploy code right from GitHub, and make code reviews, branch management, and issue triaging work the way users want. [1]

In GitHub Actions, the most powerful part of Actions is GitHub Marketplace, which shares a lot of "actions" developed by Github users or Github staff. FluxML now uses "setup-julia", "julia-runtest", "cache" and several other small but useful actions maintained by the julia-actions organization to define several workflows to run tests and get coverage reports.

3. GitHub REST API and GitHub Webhook

The most relevant APIs for this project are the workflow and issue parts. The former allows developers to trigger the pipeline directly, while the latter allows us to re-comment in the original PR thread.

GitHub Webhook greatly promotes the operability of GitHub itself, especially GitHub Actions, and it’s widely used by users to extend the functionality of workflows.

4. BuildKite (extra)

BuildKite is another DevOps platform, which enables convenient connection to GitHub repositories or organizations with the permissions of the maintainers. As for a GitHub project, BuildKite provides a "pipeline-upload" plugin to fetch .buildkite/pipeline.yml from GitHub.

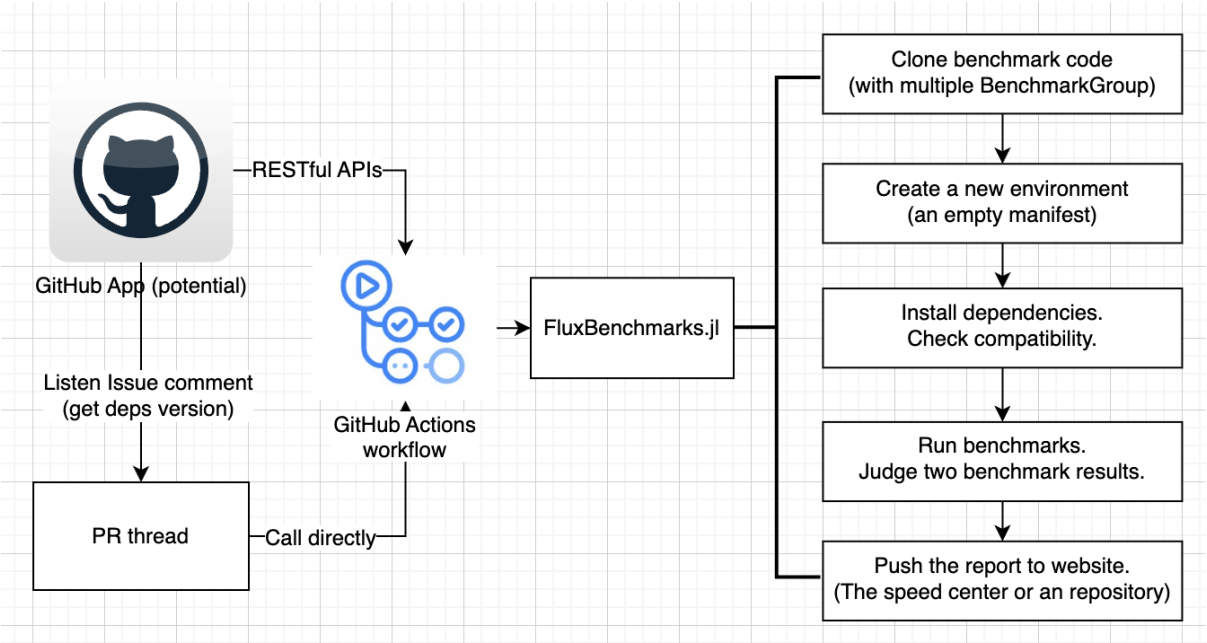
Buildkite never accesses users' code, and doesn’t maintain any agents, so users need to install and run agents on their own infrastructure. [2] Julia projects that require GPU resources to test and verify, now almost always use agents provided by the JuliaGPU community, including FluxML. [3,4].

5. BuildKite REST APIs and BuildKite Webhook (extra)

BuildKite provides user-friendly REST APIs and Webhook. With “API Access Token” of the relevant organization, a new "pipeline build" can be created easily. BuildKite also sends HTTP requests automatically after executing specific actions. BuildKite presents several demos on GitHub to show how to use webhooks. [5]

Implementation Details

As for the details of the whole project, I'd like to present the architecture diagram.

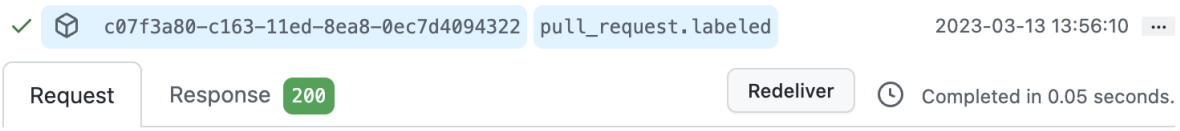


According to the diagram, I'll introduce my implementation design in four parts: "workflow trigger", "FluxBenchmarks.jl details", "benchmark design" and "potential wrappers".

Workflow trigger

In GitHub Actions, developers need to declare the events and activities that trigger the workflows. To avoid running out of GitHub Actions quota, the following trigger events or some specific activity types can be taken into consideration.[6]

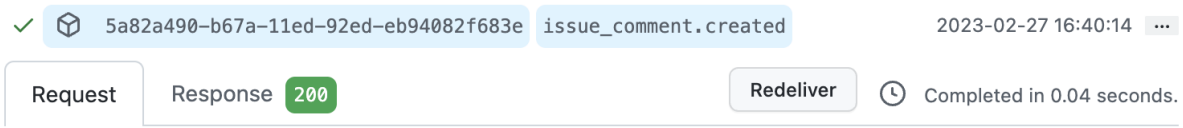
- **pull_requesr.labeled**: "pull_request" can be a good event. We can use the "labeled" activity type. The ideal corresponding solution is to attach on the pull request thread a brand-new label. (e.g. "type/benchmark")



```
"label": {
  "id": 5184047234,
  "node_id": "LA_kwDOJA8ggc8AAAABNP5Igg",
  "url": "https://api.github.com/repos/Eternal-Night-Archer/BenchmarkTrigger/labels/enhancem
  "name": "enhancement",
  "color": "a2eeef",
  "default": true,
  "description": "New feature or request"
},
```


The “label” object is included in the webhook request of “**pull_request.labeled**”. In the workflow definition, we can easily use “${{ github.event.label.name == ‘type/benchmark’ }}” as the “if-condition”. [6,7] You can check this in 1. Workflow definition yaml.

- **issue_comment**: “issue_comment” can also be a good choice. We can declare that only specified maintainers can cause the workflow run to execute to a specific location to run benchmarks. As for the activity types, both “created” and “edited” are good choices.



```
"comment": {
  "url": "https://api.github.com/repos/Eternal-Night-Archer/BenchmarkTrigger/issues/comme
  "html_url": "https://github.com/Eternal-Night-Archer/BenchmarkTrigger/pull/1#issuecomme
  "issue_url": "https://api.github.com/repos/Eternal-Night-Archer/BenchmarkTrigger/issues/
  "id": 1445917917,
  "node_id": "IC_kwDOJA8ggc5WLvTd",
  "user": {
    "login": "skyleaworlder",
    "id": 45269589,
    "node_id": "MDQ6VXNlcjQ1MjY5NTg5",
    "avatar_url": "https://avatars.githubusercontent.com/u/45269589?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/skyleaworlder",
    "html_url": "https://github.com/skyleaworlder",
    "followers_url": "https://api.github.com/users/skyleaworlder/followers",
    "following_url": "https://api.github.com/users/skyleaworlder/following{/other_user}",
    "gists_url": "https://api.github.com/users/skyleaworlder/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/skyleaworlder/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/skyleaworlder/subscriptions",
    "organizations_url": "https://api.github.com/users/skyleaworlder/orgs",
    "repos_url": "https://api.github.com/users/skyleaworlder/repos",
    "events_url": "https://api.github.com/users/skyleaworlder/events{/privacy}",
    "received_events_url": "https://api.github.com/users/skyleaworlder/received_events",
    "type": "User",
    "site_admin": false
  },
  "created_at": "2023-02-27T08:40:13Z",
  "updated_at": "2023-02-27T08:40:13Z",
  "author_association": "CONTRIBUTOR",
  "body": "Hi! This is a comment by GitHub Actions!",
```


The “**comment**” object contains the “body” field. In the workflow definition, we can use “${{ startWith(github.event.comment.body, ‘xxx’) }}” and “${{ github.actor == ‘xxx’ }}” to distinguish runs. [8]

FluxBenchmarks.jl details

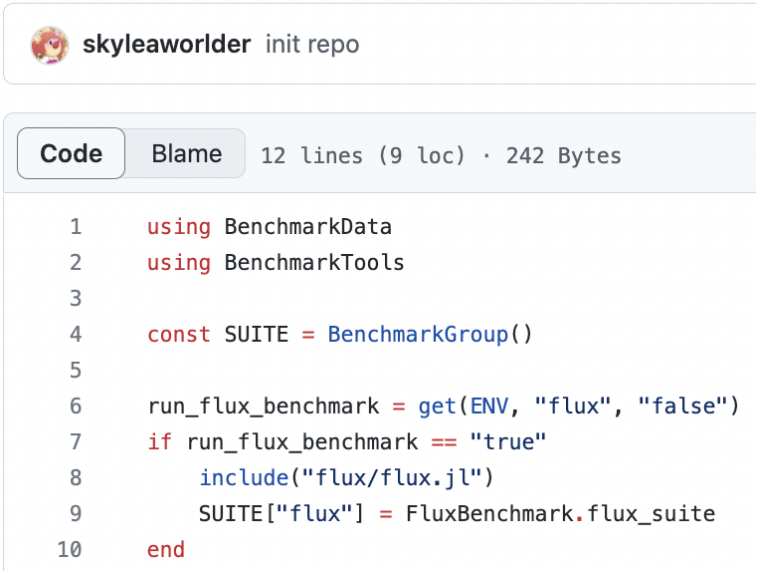
1. Clone benchmark code

I plan to **create a dedicated repository to store benchmark code (called “FluxBenchmarkCode.jl”)**, because of the following facts:

- As one of the references provided for candidates in the idea list, BenchmarkCI.jl uses “git checkout” to check out the code from git history, which means that **it cannot run benchmarks** on those commits **where you haven’t added the “benchmark” folder and provided a “SUITE”**. [12] If we adopt this way, all previous versions of the FluxML stack will not be able to run benchmarks.

- If we choose to add a “benchmark” folder into each essential repository, we cannot do integrated tests easily.

BenchmarkData / benchmark / **benchmarks.jl**



skyleaworlder init repo

Code Blame 12 lines (9 loc) · 242 Bytes

```
1   using BenchmarkData
2   using BenchmarkTools
3
4   const SUITE = BenchmarkGroup()
5
6   run_flux_benchmark = get(ENV, "flux", "false")
7   if run_flux_benchmark == "true"
8       include("flux/flux.jl")
9       SUITE["flux"] = FluxBenchmark.flux_suite
10  end
```

(There are various ways to organize benchmark code. I recommend a specific approach: including all the BenchmarkGroup definitions in a benchmark code repository. Developers can use environment variables or other methods to select the specific parts they want to run.)

On the basis of previous discussion, here I plan to refer to BenchmarkCI.jl, and run the “git” command directly to clone code.

2. Create a new environment

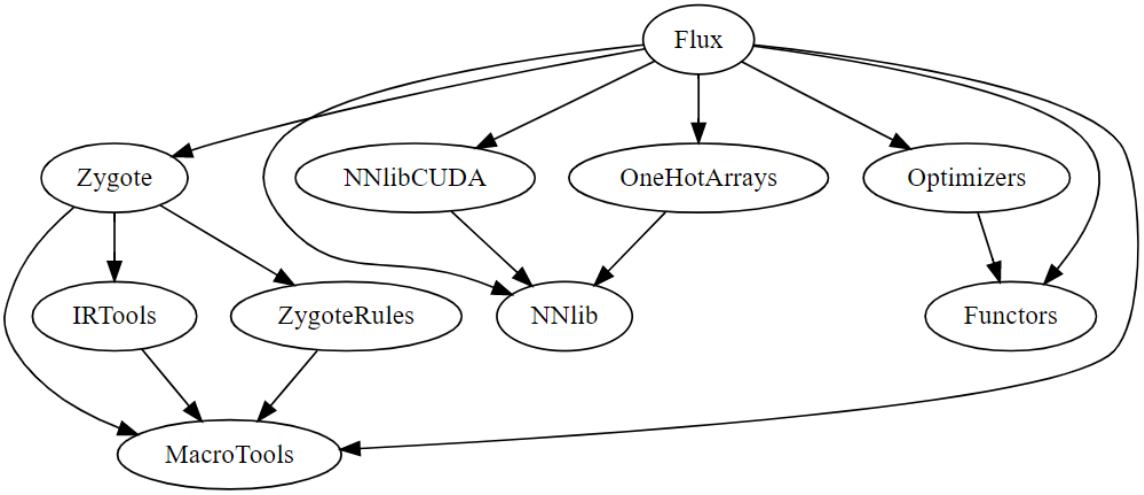
Pkg.jl provides the “activate” command. We can define a “Project.toml” file in FluxBenchmarkCode.jl.

Both “Pkg.activate” and “cd” should be called to create a new environment.

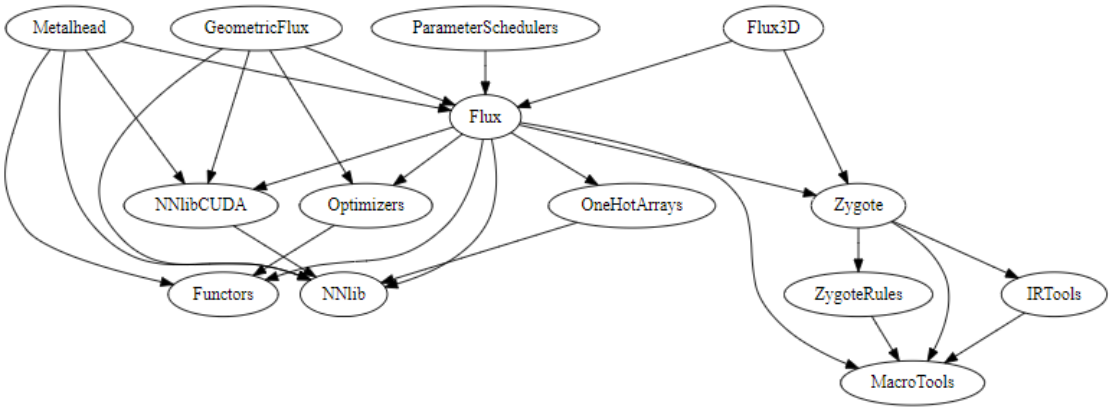
```
Pkg.activate("../BenchmarkData/")
cd("../BenchmarkData/")
```

3. Install dependencies

The following two pictures shows the current dependency graph of the FluxML stack. [13]



If we view “Flux.jl” as the origin, we can get the above graph.



If we consider “Metalhead.jl” and other “applications”, we can get the above graph. But in short, we can know that the FluxML stack is not too sophisticated to manage.

In Pkg.jl, it provides the “add” command to install dependencies. But there is a detail that **conflict / exception only occurs if developers “add” dependencies from top down**. If you “add” dependencies from down up, Pkg.jl will help to update out-of-date dependencies, instead of throwing an exception.

Therefore, FluxBenchmarks.jl must maintain the relationship of dependencies, and install the dependencies layer by layer.

4. Run benchmarks

In order to judge two benchmark results, it's required to install two different sets of dependencies. "BenchmarkTools.jl" and "PkgBenchmark.jl" provide various useful functionalities like "BenchmarkGroup" and "BenchmarkResult". We can reimplement or extend them to satisfy our own requirements.

You can see my idea in this piece of [code (2. FluxBenchmarks.jl skeleton)](#).

5. Push the report to website

For the storage of the reports, I have two solutions to consider:

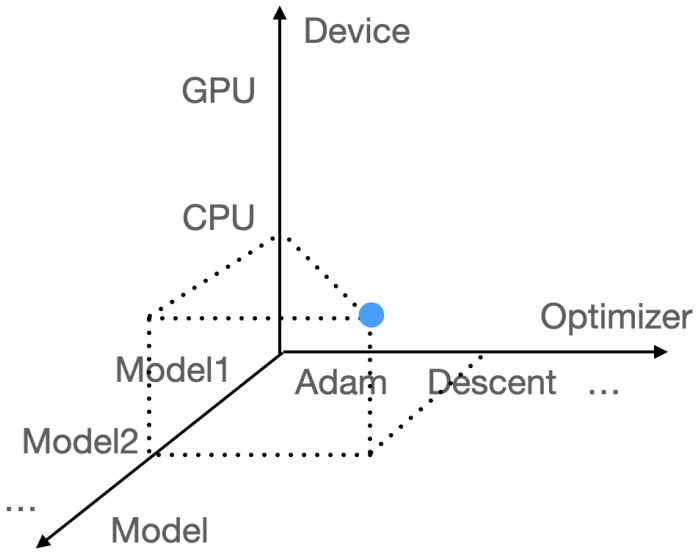
- "JuliaGPU speed center" and "TaylorDiff" are listed in the idea list. I refer to the way **"FluxBench.jl"** and CUDA.jl upload benchmark results. [14,15]

```
HTTP.post("$(ENV["CODESPEED_SERVER"])/result/add/json/",
          ["Content-Type" => "application/x-www-form-urlencoded"],
          HTTP.URIs.escapeuri(Dict("json" => JSON.json(flat_results))))
```

- We can **create a new branch named "result" in "FluxBenchmarkCode.jl"**. We can add an action in the benchmark workflow to ensure the results are uploaded when the benchmarks are completed.

Benchmark design

According to the idea list and current discussions, the contents of the Benchmark code repository can be organized based on 4 topics. Among them, "Operators" will be separately taken into consideration. In addition, the benchmarks of a model are also related to 3 other topics: "Training model and inference (complete end-to-end examples)", "Optimizers (AD)" and "Device". The benchmarks require a comprehensive evaluation of the utilization of these 4 topics.



1. Operators

I plan to initialize a group of basic layers (e.g. Conv, MeanPool, Dropout, BatchNorm, etc.) with different sizes and connect them together respectively.

```
for ch in [8, 16, 32, 64]
    input = rand(Float64, 64, 64, ch, 64)
    model = Conv((3,3), ch => ch)

    # ignore some code
    # maybe like model(input)
end
```

2. Training model and inference

I plan to use some small open datasets (e.g. MNIST and CIFAR-10) to train and test models in Metalhead.jl or defined by Flux.jl.

3. Optimizers

There’re various kinds of optimizers in Optimizer.jl. The benchmarks of different optimizers are also very important. I plan to provide different optimizers to meet customization needs when training the network.

4. Device

Since the GPU resources of the JuliaLang projects are primarily deployed on BuildKite, this task may be considered as an additional one. I plan to provide a device option that can be easily utilized on BuildKite in the future.

```
if on_gpu
    model = gpu(model)
    x = gpu(x)
end
```

Potential wrapper or advanced functions

1. GitHub PR thread comment

To improve user experience, I think it's necessary to send a comment to the PR thread that triggers the benchmark pipeline after the benchmarks are completed. We can use an action used to create a comment in a given PR thread. [17] **You can see my idea by looking at the example here (5. Use the github-action to comment).**

2. GitHub App (extra)

Using GitHub Actions Expressions, we can indeed extract our needed runs, but Expressions sometimes execute unexpectedly due to the bugs. Actually, building a GitHub Bot / GitHub App can be a good choice.

Probot is a kind of framework to help developers to build their customized GitHub Bots. I have built a simple Probot example and made it use GitHub RESTful APIs successfully. **You can see my idea in this available example (3. Probot example).**

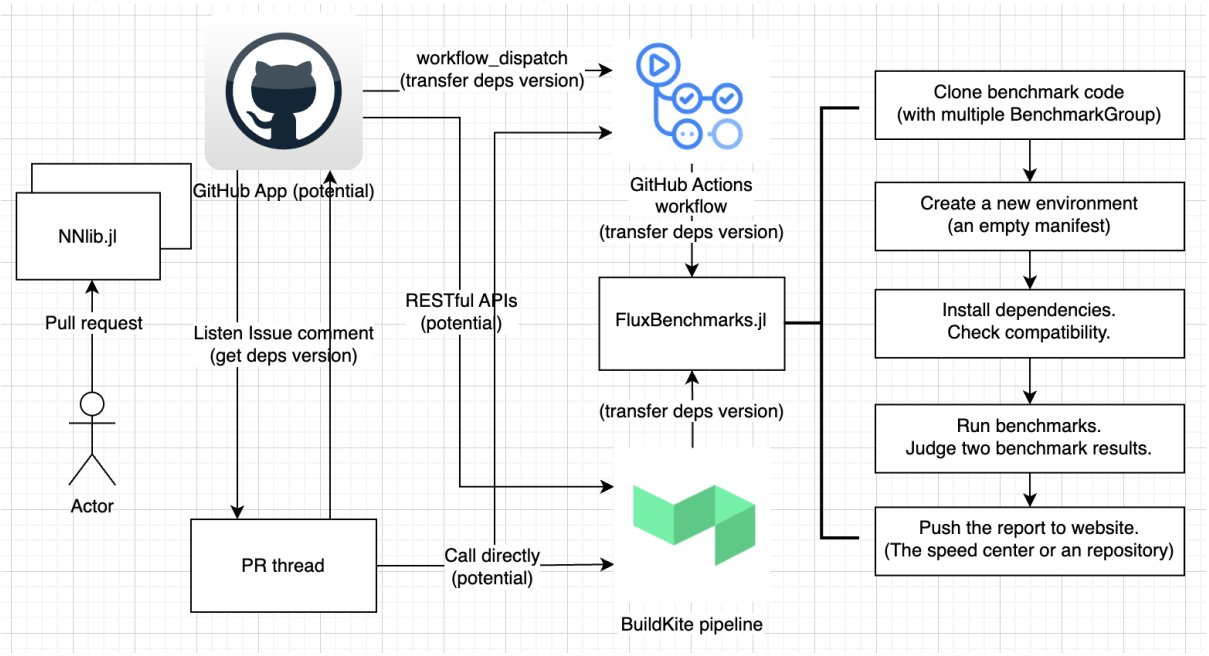
Of course, **building a bot is not a necessary way**. Here I just show I'm able to use Probot to polish this project and promote user experience. But now I think it can be an **optional target**.

3. BuildKite (extra)

During my preparation, I read the API docs of BuildKite platform and became familiar with the BuildKite platform by using my PC as a buildkite agent to run CI pipeline builds.

Now **I have a [repository](#)**, [15] which can provide a bit of abstraction when calling BuildKite APIs to some extent.

Based on the current discussions (due to the need to communicate with the JuliaGPU community), I plan to include this feature as an extra task. But I think I can easily integrate this feature into FluxBenchmarks.jl. The whole architecture would be like:



Code Snippets or Existing Demos

1. Workflow definition yaml

```yaml
name: Benchmark

on:
  pull_request:
    types: [ labeled ]

jobs:
  benchmark:
      if: ${{ github.event.label.name == 'type/benchmark' }}
      runs-on: ubuntu-latest

      steps:
      - uses: actions/checkout@v3
      - name: "ignore the following code"
        run: echo "ignore the following code"
```

2. FluxBenchmarks.jl skeleton (MOST IMPORTANT)

The following code includes cloning benchmark code, creating a new environment, installing dependencies and running benchmarks. The only problem I think is that it’s too simple and many exceptions and unexpected behaviors are not considered.

```
mutable struct Dependency
    name::Union{Nothing,String}
    rev::Union{Nothing,String}
    version::Union{Nothing,String}
    url::Union{Nothing,String}
end

function Dependency(;name::Union{Nothing,String} = nothing,
                     rev::Union{Nothing,String} = nothing,
                     version::Union{Nothing,String} = nothing,
                     url::Union{Nothing,String} = nothing)
    !isnothing(url) || !isnothing(rev) ||
        !isnothing(version) || throw(error("illegal input of Dependency"))
    Dependency(name, rev, version, url)
end

function dependency_to_packagespec(dep::Dependency)
    if !isnothing(dep.rev)
        PackageSpec(name = dep.name; version = dep.version)
    elseif !isnothing(dep.version)
        PackageSpec(name = dep.name; rev = dep.rev)
    elseif !isnothing(dep.url)
        PackageSpec(url = dep.url)
    else
        PackageSpec()
    end
end
```

Dependency, a much simpler “PackageSpec”, is used to dovetail with our requirements for dependencies installation. (Here I don’t want it to have so many fields like PackageSpec.)

```
function setup(deps::Vector{Dependency})
    if !isdir("../BenchmarkData/")
        run(`git clone https://github.com/Eternal-Night-Archer/BenchmarkData ../BenchmarkData`)
    end

    Pkg.activate("../BenchmarkData/")
    cd("../BenchmarkData/")

    for dep in deps
        Pkg.add(dependency_to_packagespec(dep))
    end
end

function teardown()
    Pkg.rm(all_pkgs = true)
    pwd = ENV["PWD"]
    Pkg.activate(pwd)
    cd(pwd)
end
```

- Function “setup” is to initialize the benchmark running conditions, including cloning the code repository, creating a new environment and adding dependencies.
- Function “teardown” is to destroy the conditions, including removing dependencies and switching back.

The “setup” and “teardown” can be used together and reused multiple times:

```
first_deps = [
    Dependency(name = "BenchmarkTools", version = "1.3.2"),
    Dependency(url = "https://github.com/skyleaworlder/Flux.jl#metrics-proposal"),
    Dependency(name = "NNlib", version = "0.8.12")
]

second_deps = [
    Dependency(name = "BenchmarkTools", version = "1.3.2"),
    Dependency(name = "Flux", version = "0.13.14"),
    Dependency(name = "NNlib", version = "0.8.19")
]

function main()
    setup(first_deps)
    import BenchmarkData
    import PkgBenchmark
    PkgBenchmark.benchmarkpkg(BenchmarkData)
    teardown()

    setup(second_deps)
    import BenchmarkData
    import PkgBenchmark
    PkgBenchmark.benchmarkpkg(BenchmarkData)
    teardown()
end
```

In REPL:

```
julia> setup(first_deps)
  Activating project at `~/FluxML/BenchmarkData`
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manifest.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Project.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manifest.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Project.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg/src/Registry/registry_instance.jl:328
   Resolving package versions...
    Updating `~/FluxML/BenchmarkData/Project.toml`
  [6e4b80f9] + BenchmarkTools v1.3.2
    Updating `~/FluxML/BenchmarkData/Manifest.toml`
  [6e4b80f9] + BenchmarkTools v1.3.2
  [682c06a0] + JSON v0.21.3
  [69de0a69] + Parsers v2.5.8
  [21216c6a] + Preferences v1.3.0
  [66db9d55] + SnoopPrecompile v1.0.3
  [56f22d72] + Artifacts

julia> import BenchmarkData

julia> import PkgBenchmark

julia> res1 = PkgBenchmark.benchmarkpkg(BenchmarkData)
PkgBenchmark: Running benchmarks...
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manifest.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Project.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg/src/Registry/registry_instance.jl:328
SHELL --> /bin/bash
CAML_LD_LIBRARY_PATH --> /home/lijg/.opam/4.14.0+flambda/lib/stublibs:/home/lijg/.opam/4.14.0+flambda/lib/ocaml/stublibs:/home/lijg/.opam/4.14.0+flambda/lib/ocaml

done (took 8.0/141///1 seconds)
Benchmarking 100%|                                                                                | Time: 0:00:10
Benchmarkresults:
    Package: /home/lijg/FluxML/BenchmarkData
    Date: 15 Mar 2023 - 07:25
    Package commit: dirty
    Julia commit: 0434de
    BenchmarkGroup:
        1-element BenchmarkTools.BenchmarkGroup:
          tags: []
          "flux" => 2-element BenchmarkTools.BenchmarkGroup:
                  tags: []
                  "maxpool" => 2-element BenchmarkTools.BenchmarkGroup:
                          tags: []
                          "normal" => Trial(17.698 μs)
                          "global" => Trial(35.661 μs)
                  "meanpool" => 2-element BenchmarkTools.BenchmarkGroup:
                          tags: []
                          "normal" => Trial(17.891 μs)
                          "global" => Trial(35.691 μs)
```

```
julia> teardown()
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
    Updating `~/FluxML/BenchmarkData/Project.toml`
  [6e4b80f9] - BenchmarkTools v1.3.2
  [587475ba] - Flux v0.13.12 `https://github.com/skyleaworlder/Flux.jl#metrics-proposal#master`
  [872c559c] - NNlib v0.8.19
    Updating `~/FluxML/BenchmarkData/Manifest.toml`
  [621f4979] - AbstractFFTs v1.3.1
  [7d9f7c33] - Accessors v0.1.28
  [79e6a3ab] - Adapt v3.6.1
  [dce04be8] - ArgCheck v2.3.0

julia> setup(second_deps)
  Activating project at `~/FluxML/BenchmarkData`
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
   Resolving package versions...
    Updating `~/FluxML/BenchmarkData/Project.toml`
  [6e4b80f9] + BenchmarkTools v1.3.2
    Updating `~/FluxML/BenchmarkData/Manifest.toml`
  [6e4b80f9] + BenchmarkTools v1.3.2
  [682c06a0] + JSON v0.21.3
  [69de0a69] + Parsers v2.5.8
  [21216c6a] + Preferences v1.3.0
  [66db9d55] + SnoopPrecompile v1.0.3

julia> Pkg.status()
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
Project BenchmarkData v0.1.0
Status `~/FluxML/BenchmarkData/Project.toml`
  [6e4b80f9] BenchmarkTools v1.3.2
  [587475ba] Flux v0.13.14
  [872c559c] NNlib v0.8.19
```

```
julia> res2 = PkgBenchmark.benchmarkpkg(BenchmarkData)
PkgBenchmark: Running benchmarks...
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Manife
st.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
┌ Warning: Expected key "git-tree-sha1" to exist in registry TOML file at "/home/lijg/.julia/registries/Projec
t.toml"
└ @ Pkg.Registry /cache/build/default-amdci5-6/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Pkg
/src/Registry/registry_instance.jl:328
SHELL --> /bin/bash
CAML_LD_LIBRARY_PATH --> /home/lijg/.opam/4.14.0+flambda/lib/stublibs:/home/lijg/.opam/4.14.0+flambda/lib/ocam
l/stublibs:/home/lijg/.opam/4.14.0+flambda/lib/ocaml
OCAML_TOPLEVEL_PATH --> /home/lijg/.opam/4.14.0+flambda/lib/toplevel

 done (took 8.091793512 seconds)
 Benchmarking 100%|                                                                                | Time: 0:00:10
 Benchmarkresults:
     Package: /home/lijg/FluxML/BenchmarkData
     Date: 15 Mar 2023 - 07:28
     Package commit: dirty
     Julia commit: 0434de
     BenchmarkGroup:
         1-element BenchmarkTools.BenchmarkGroup:
           tags: []
           "flux" => 2-element BenchmarkTools.BenchmarkGroup:
                   tags: []
                   "maxpool" => 2-element BenchmarkTools.BenchmarkGroup:
                           tags: []
                           "normal" => Trial(17.502 μs)
                           "global" => Trial(35.669 μs)
                   "meanpool" => 2-element BenchmarkTools.BenchmarkGroup:
                           tags: []
                           "normal" => Trial(17.842 μs)
                           "global" => Trial(35.699 μs)


julia> judgement = PkgBenchmark.judge(res1, res2)
Benchmarkjudgement (target / baseline):
    Package: /home/lijg/FluxML/BenchmarkData
    Dates: 15 Mar 2023 - 7:25 / 15 Mar 2023 - 7:28
    Package commits: dirty / dirty
    Julia commits: 0434de / 0434de


julia> PkgBenchmark.export_markdown(
export_markdown(io::IO, results::PkgBenchmark.BenchmarkResults) in PkgBenchmark at /home/lijg/.julia/packages/
PkgBenchmark/nuPay/src/benchmarkresults.jl:107
export_markdown(io::IO, judgement::PkgBenchmark.BenchmarkJudgement; export_invariants) in PkgBenchmark at /hom
e/lijg/.julia/packages/PkgBenchmark/nuPay/src/benchmarkjudgement.jl:62
export_markdown(file::String, results::PkgBenchmark.BenchmarkResults) in PkgBenchmark at /home/lijg/.julia/pac
kages/PkgBenchmark/nuPay/src/benchmarkresults.jl:101
export_markdown(file::String, results::PkgBenchmark.BenchmarkJudgement; kwargs...) in PkgBenchmark at /home/li
jg/.julia/packages/PkgBenchmark/nuPay/src/benchmarkjudgement.jl:56
julia> PkgBenchmark.export_markdown("report.md", judgement)

shell> cat report.md
# Benchmark Report for */home/lijg/FluxML/BenchmarkData*

## Job Properties
* Time of benchmarks:
    - Target: 15 Mar 2023 - 07:25
    - Baseline: 15 Mar 2023 - 07:28
```

Benchmark Group List

Here's a list of all the benchmark groups executed by this job:

- ["flux", "maxpool"]
- ["flux", "meanpool"]

Julia versioninfo

Target

```
Julia Version 1.8.3
Commit 0434deb161e (2022-11-14 20:14 UTC)
Platform Info:
  OS: Linux (x86_64-linux-gnu)
      Ubuntu 22.04.1 LTS
  uname: Linux 5.15.0-47-generic #51-Ubuntu SMP Thu Aug 11 07:51:15 UTC 2022 x86_64 x86_64
  CPU: Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz:
                speed         user         nice          sys         idle          irq
       #1-64   895 MHz   21772178 s      29707 s    2626266 s  9702074883 s          0 s
  Memory: 125.51469421386719 GB (113390.671875 MB free)
  Uptime: 1.520040358e7 sec
  Load Avg:  5.17  3.15  1.57
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-13.0.1 (ORCJIT, cascadelake)
  Threads: 1 on 64 virtual cores
```

Baseline

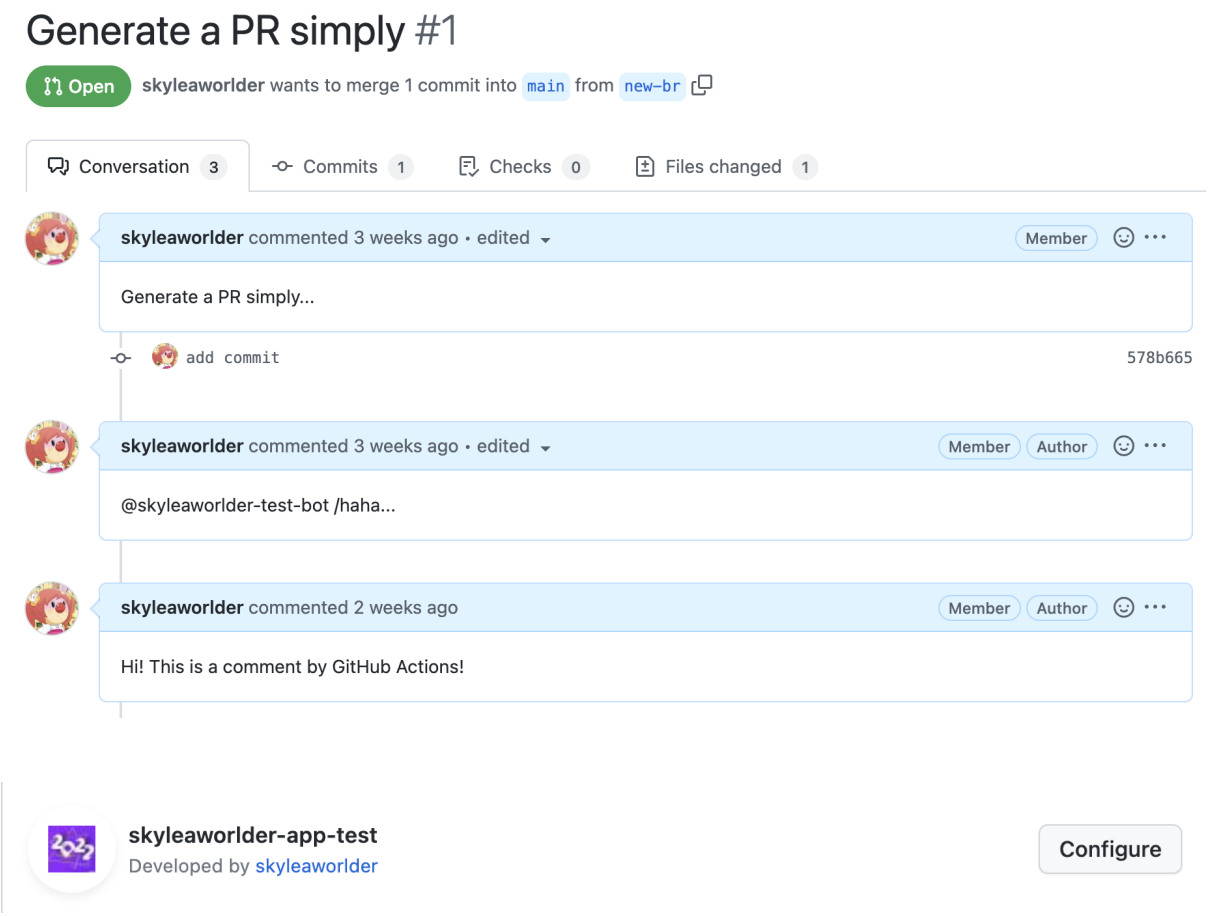
```
Julia Version 1.8.3
Commit 0434deb161e (2022-11-14 20:14 UTC)
Platform Info:
  OS: Linux (x86_64-linux-gnu)
      Ubuntu 22.04.1 LTS
  uname: Linux 5.15.0-47-generic #51-Ubuntu SMP Thu Aug 11 07:51:15 UTC 2022 x86_64 x86_64
  CPU: Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz:
                speed         user         nice          sys         idle          irq
       #1-64  2100 MHz   21774719 s      29707 s    2627366 s  9702169439 s          0 s
  Memory: 125.51469421386719 GB (114829.14453125 MB free)
  Uptime: 1.520055712e7 sec
  Load Avg:  2.33  2.66  1.62
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-13.0.1 (ORCJIT, cascadelake)
  Threads: 1 on 64 virtual cores
```

3. Probot example

```
export = (app: Probot) => {
  app.on(["issue_comment.edited"], async (context) => {
    app.log.info("issue_comment edited");
    // "includes" can be replaced with other functions.
    if (context.payload.comment.body.includes("@skyleaworlder-test-bot /")) {
      app.log.info(`inputs: repository: ${context.payload.repository.full_name}`)
    }
  });
}
```

You can check this example in three repositories:

- A bot built with Probot (code) (https://github.com/Eternal-Night-Archer/trigger-bot). [9]
- A PR comment triggers “issue_comment.edited” (the repository to which my GitHub App / GitHub Bot listens) (https://github.com/Eternal-Night-Archer/BenchmarkTrigger/pull/1#issuecomment-1439580741). [10]
- A workflow triggered by the GitHub Bot / GitHub App automatically (https://github.com/Eternal-Night-Archer/BenchmarkData/actions/runs/4280428143/attempts/1). [11]



skyleaworlder-app-test

Installed 3 weeks ago · Developed by skyleaworlder · App settings · https://github.com/apps/skyleaworlder-app-test

A Probot app

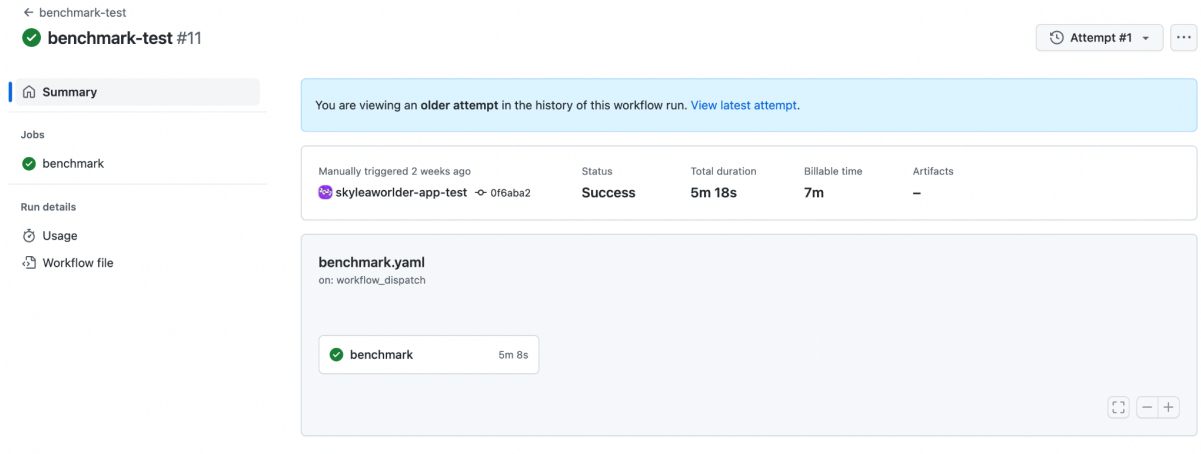
Permissions

- ✓ **Read** access to metadata
- ✓ **Read** and **write** access to actions, actions variables, issues, pull requests, repository hooks, and workflows

Subscribe to events

Based on the permissions you've selected, what events would you like to subscribe to?

- ☐ **Meta**
  When this App is deleted and the associated hook is removed.
- ☐ **Security advisory**
  Security advisory published, updated, or withdrawn.
- ☑ **Issue comment**
  Issue comment created, edited, or deleted.
- ☑ **Issues**
  Issue opened, edited, deleted, transferred, pinned, unpinned, closed, reopened, assigned, unassigned, labeled, unlabeled, milestoned, demilestoned, locked, or unlocked.
- ☐ **Label**
  Label created, edited or deleted.
- ☐ **Milestone**
  Milestone created, closed, opened, edited, or deleted.
- ☐ **Merge queue entry**
  Merge Queue entry added
- ☐ **Public**
  Repository changes from private to public.
- ☑ **Pull request**
  Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.
- ☐ **Pull request review**
  Pull request review submitted, edited, or dismissed.
- ☐ **Pull request review comment**
  Pull request diff comment created, edited, or deleted.
- ☐ **Pull request review thread**
  A pull request review thread was resolved or unresolved.
- ☑ **Repository**
  Repository created, deleted, archived, unarchived, publicized, privatized, edited, renamed, or transferred.
- ☐ **Star**
  A star is created or deleted from a repository.
- ☐ **Watch**
  User stars a repository.
- ☑ **Workflow job**
  Workflow job queued, waiting, in progress, or completed on a repository.
- ☑ **Workflow run**
  Workflow run requested or completed on a repository.



4. BuildKite.jl (Incomplete | WIP)

https://github.com/skyleaworlder/BuildKite.jl

README.md

BuildKite.jl

Yet another API implementation of BuildKite REST-ful APIs, but in julia. (simply wraping APIs with HTTP.jl and JSON.jl) So BuildKite.jl tries to provide some functions, which adheres to BuildKite APIs (Version 2) | BuildKite Documentation.

(~~Actually, the main purpose of this repository for me is only to read API docs, own a understanding of some specific concepts, use my PC as a buildkite agent to run CI pipeline builds and test functions in this repository to make sure they could and would work, which got myself much more familiar with BuildKite platform.~~)

Credit

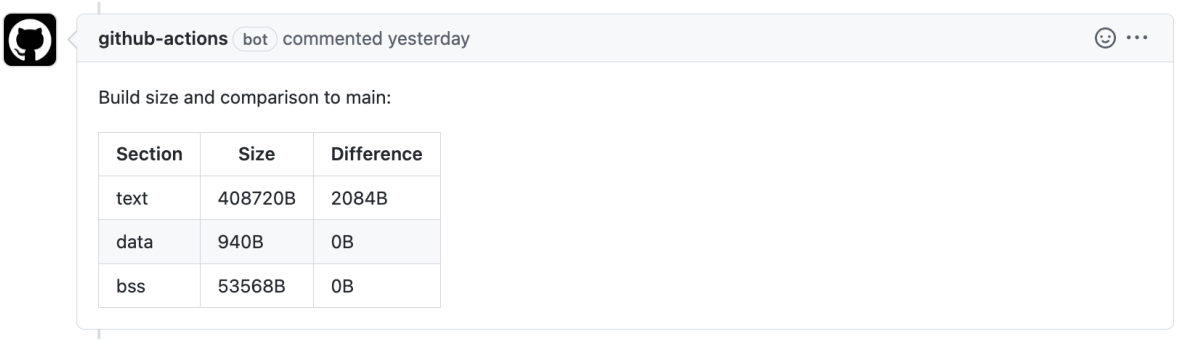
- GitHub.jl: refer to HTTP action wrapper implementation; really learn a lot meta-programming and design ideas from `@ghdef` although I didn't use these thoughts here.
- HTTP.jl: refer to client functionalities.

5. Use the github-action account to comment

I've experienced this action in https://github.com/Eternal-Night-Archer/BenchmarkData/actions/runs/4280428143/workflow.

NOTICE: since I use PAT (Personal Access Token) instead of "{{ GITHUB_TOKEN }}", it seems that I posted a comment by myself in https://github.com/Eternal-Night-Archer/BenchmarkTrigger/pull/1#issuecomment-1445917917. I pass my PAT only to enable this action with permissions (issue comment) to the other repositories. This comment is automatically generated by the "create-or-update-comment" action. If I don't pass my PAT ("create-or-update-comment" action use GITHUB_TOKEN by default), the comment will be posted using the "github-action" account like this:

```
- name: Create or update comment
  uses: peter-evans/create-or-update-comment@5adcb0bb0f9fb3f95ef05400558bdb3f329ee808
  with:
    comment-id: ${{ steps.find-comment.outputs.comment-id }}
    issue-number: ${{ github.event.pull_request.number }}
    body-file: comment
```



github-actions bot commented yesterday

Build size and comparison to main:

Section	Size	Difference
text	408720B	2084B
data	940B	0B
bss	53568B	0B

Workplan

1. Community Bonding Period (May 4th - May 28th)

- Research the history of FluxML and its original infrastructure to get a deeper understanding of the project.
- Discuss the implementation of the project and expected goals with Mentors more precisely and deeply.
- Spend more time and effort on documentation.
- Figure out one feasible solution to fulfill the expectations of the project.
- Acquire the necessary permissions to access project related repositories.
- Continue to attend FluxML-Community-Call-Minutes.

2. Code Phase 1 (May 29th - Jul 10th)

Here I only list my current work plan, the specific arrangements can be changed according to the actual situation at the time.

Week	Tasks	Targets
Week 1-2 (May 29th - Jun 11th)	• Initialize the benchmark code repository. • Discuss and determine the workflow definitions and the calling convention of FluxBenchmarks.jl.	• A benchmark code repository. • A workflow draft. • The calling convention of FluxBenchmarks.jl.
Week 3-4 (Jun 12th - Jun 25th)	• Write benchmarks. • Develop functionality of benchmark code loading. • Develop functionality of dependency installation.	• FluxBenchmarks.jl supports code download and dependency installation.
Week 5-6 (Jun 25th - Jul 10th)	• Write benchmarks. • Develop functionality of running benchmark groups and judgement.	• FluxBenchmarks.jl supports benchmark running and judgement.

3. Code Phase 2 (Jul 14th - Aug 21st)

Here I only list my current work plan, the specific arrangements can be changed according to the actual situation at the time.

Week	Tasks	Targets
Week 7-8 (Jul 14th - Jul 27th)	● Write benchmarks. ● Polish code and export judgement reports into widely applicable format.	● FluxBenchmarks.jl supports reports uploading. ● Available workflows.
Week 9-10 (Jul 28th - Aug 10th)	● Display benchmark results. Re-comment original PR thread. ● Detecting workflows / pipelines completed the event correctly.	● Extra negotiable enhancements. (Multiple ways to display benchmark results)
Week 11 (Aug 11st - Aug 21st)	● Use BuildKite webhook to listen to the build.completed event and utilize action with comment ability.	● Extra negotiable enhancements. (BuildKite support)

4. The Last Week (Aug 21st - Aug 28th)

In the last week, I hope I will have already done all of the coding tasks, but there is a possibility of some unexpected cases that would make a significant impact. To be safe, I plan to put the **maximum** focus on the project this week. (**maximum** possibly means “70 hours this week”)

- Fix bugs and process unexpected behaviors.
- Add the inadequate functionalities.

- Make up the documents.

Major Challenge Foreseen

- Even if I've worked with Pkg.jl for a while, I still don't think I'm a Pkg.jl expert. I think there would be some problems for me to deal with when using Pkg.jl.
- Designing a good benchmark tool for a specific scene is still a challenge:
    - Error process should be considered comprehensively.
    - Tradeoffs between technology and design are everywhere.

Contingency

Because I only introduced some code snippets in this proposal, there is still a possibility of infeasibility for this project. Here I would like to present corresponding potential solutions for each challenge mentioned in the above topic.

- Problems about GitHub Actions: Now I'm familiar with GitHub API Docs. This project would have a strong connection with "Issue comments" and "Workflows" events, which are documented in Workflows docs.
- Problems about Pkg.jl: I think I can solve the problem by consulting the documents or testing in REPL.
- Another implementation: I'm familiar with GitHub "workflow dispatch", which is a potential substitute.
- Problems about WebHook: In the current foreseeable development process, WebHook can greatly improve the user-friendliness of applications on GitHub. I would read the following documents: GitHub WebHooks | WebHook Events and Payloads and BuildKite WebHook | Build Events.
- Problems about BuildKite (extra): Now I'm also familiar with the BuildKite API to some extent. This project would have a connection with "Build", which is well-documented in Builds API.

CV / Related Experience

- GitHub username: skyleaworlder
- Education:
  - Tongji Univ. Dept. CS, B.S. Eng. (2018-2022)
    - I obtained **Outstanding Shanghai Graduates** in 2022. (Top 0.3% national-wide)
    - I got a **National Scholarship** in 2020. (Top 0.2% national-wide)
  - Fudan Univ. SE Lab (2022-2025 expected)
    - TA of Software Quality Assurance and Testing.
- Internship:
  - Bytedance Inc. **DevOps Engineer**
    - I **have a deep understanding of the pipeline concept** in DevOps platform because I exactly developed the **pipeline** and template in DevOps platform.
  - Honor Device Co., Ltd. **Solution Engineer**
    - I developed multiple useful tools (especially git) to support system developers to handle difficult scenes.
  - Apache/shenyu **GLCC Mentee | Contributor**
    - GLCC is also an activity about OSS.
    - I proposed several new **GitHub Actions workflows** to publish docker images and documentations.
      - https://github.com/apache/shenyu/pull/3652
      - https://github.com/apache/shenyu/pull/3685
      - https://github.com/apache/shenyu/pull/3722
      - https://github.com/apache/shenyu/pull/3989
      - https://github.com/apache/shenyu-website/pull/619
      - https://github.com/apache/shenyu-website/pull/620
- Leadership:
  - TJ-CSCCG **Founder & Owner**
    - I led a team to provide LaTeX templates, and share experience of study, interview, examination and so on.

Language and Other Skill Stack

- Julia (I don't know how much I have mastered, but certainly enough)
- GitHub Actions (Expert to some extent)
- Git (Expert to some extent: My GitHub Profile)
- Yaml (Intermediate)
- TypeScript / Node.js
- Actually, I can use various programming languages, (You can check skyleaworlder's repositories)
    - About language: from Lisp (FP) to Ruby (OOP totally),
    - About typing:
        - Java (static & strong typing)
        - C++ (static & weak typing)
        - Python (dynamic & strong typing)
        - JavaScript (dynamic & weak typing).
    - About design: from Go (interface-oriented) to Scala (multi-paradigm) and so on.

Why Me & Why This Project?

For me, both of these seem to be the same.

- I am proficient in using git and GitHub.
- I believe I understand this project better than the other candidates. I proposed my first PR to FluxML in Dec. 2022 and I have a brief understanding of:
    - Most of the code in FluxML, especially Flux.jl and NNlib.jl.
    - Workflow of all repositories in FluxML.
- At present, I'm a graduate student in the Software Engineering Lab of Fudan University. Although I've only been coding for five years, I view **"building systems for developers with well-designed architecture"** as my own mission. Briefly, it's okay to develop a tool for ordinary people; but developing a tool for developers will bring much more sense of accomplishment for me.
- At first, I only had the idea of learning Flux to finish my school assignment and contributed some code casually. However, after accidentally seeing the meeting notes of January 15th. and attending the meeting at the end of that month, I surely

had the intention to apply for this project, because this project closely aligns with my previous experience.
- I would be extremely happy to work for FluxML because it helps build Neural Networks for Julia developers. Also, in my PR process, I experienced that the FluxML maintainers are very knowledgeable from hardware to software development. I believe I can learn a lot in completing this project.

Prior Contribution to FluxML

- [Draft] Accuracy function: https://github.com/FluxML/Flux.jl/pull/2181
- [Merge] Reduce workflow warnings and other 10 similar PRs across FluxML community about GitHub Actions: https://github.com/FluxML/Flux.jl/pull/2173
- [Merge] Reduce lppool conversion: https://github.com/FluxML/NNlib.jl/pull/467
- [Open] Add LPNormPool to Flux.jl: https://github.com/FluxML/Flux.jl/pull/2166
- [Merge] Add lpnormpool to NNlib.jl: https://github.com/FluxML/NNlib.jl/pull/447
- [Merge] Update workflow to generate coverage for specific entries: https://github.com/FluxML/Flux.jl/pull/2136

GSoC Information

- This is my first time to apply for the GSoC project. I have a little experience with events like GSoC. (I took part in GLCC as a mentee of apache/shenyu from July to September last year) But at the same time, I have to admit that I'm still an open-source novice, because I have no long-term experience with a large community till now.
- I only focus on preparing for this project and haven't applied to any other organizations or projects concurrently.

Availability

I don’t have other time-consuming jobs during my GSoC phase. I'll devote more than 28 hours per week. And I'm not in the habit of going out during weekends or spending lots of time on games and short-video platforms. (Yes, there are no TikTok, Ins, Twitter, Bilibili, games and any other apps on my smartphone.)

If time is still not enough, I think that time is like a sponge, there will always be a squeeze.

Future Plan

If my proposal were accepted, my first priority would be to develop the necessary features. On this basis, I think it would be ideal to address the extra parts mentioned above in the development process. If time is not enough, I can continue to complete some well-discussed extra parts afterwards.

As this project is aimed at developers, there will definitely be some points that can be optimized. I'm also willing to update my project to meet user requirements.

Reference

1. GitHub Actions: https://github.com/features/actions
2. BuildKite Getting Started: https://buildkite.com/docs/tutorials/getting-started
3. Discourse JuliaGPU CI Moving To BuildKite: https://discourse.julialang.org/t/juliagpu-ci-moving-to-buildkite/49925
4. BuildKite Julialang Flux.jl: https://buildkite.com/julialang/flux-dot-jl
5. LIFX BuildKite Build Light Node: https://github.com/buildkite/lifx-buildkite-build-light-node/blob/master/index.js
6. GitHub Actions | Using workflow | Events that trigger workflows: https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows
7. StackOverflow | Run Github Actions when pull requests have a specific label: https://stackoverflow.com/questions/62325286/run-github-actions-when-pull-requests-have-a-specific-label
8. GitHub Actions | Learn GitHub Actions | Expressions | startswith: https://docs.github.com/en/actions/learn-github-actions/expressions#startswith
9. skyleaworlder | trigger-bot: https://github.com/Eternal-Night-Archer/trigger-bot
10. skyleaworlder | BenchmarkTrigger | Pull Request | Generate a simple reply: https://github.com/Eternal-Night-Archer/BenchmarkTrigger/pull/1#issuecomment-1439580741
11. skyleaworlder | BenchmarkData | Actions | benchmark-test: https://github.com/Eternal-Night-Archer/BenchmarkData/actions/runs/4280428143/attempts/1
12. tkf | BenchmarkCI.jl: https://github.com/tkf/BenchmarkCI.jl/blob/master/src
13. FluxML: https://github.com/FluxML
14. FluxML | FluxBench.jl: https://github.com/FluxML/FluxBench.jl/blob/master/src/FluxBench.jl
15. JuliaGPU | CUDA.jl: https://github.com/JuliaGPU/CUDA.jl/blob/master/perf/runbenchmarks.jl
16. Marketplace | create-or-update-comment: https://github.com/marketplace/actions/create-or-update-comment
17. JuliaCI | PkgBenchmark.jl: https://github.com/JuliaCI/PkgBenchmark.jl/blob/master/src/runbenchmark.jl#L269