Building Custom GitHub Actions

ANATOMY OF A GITHUB ACTION

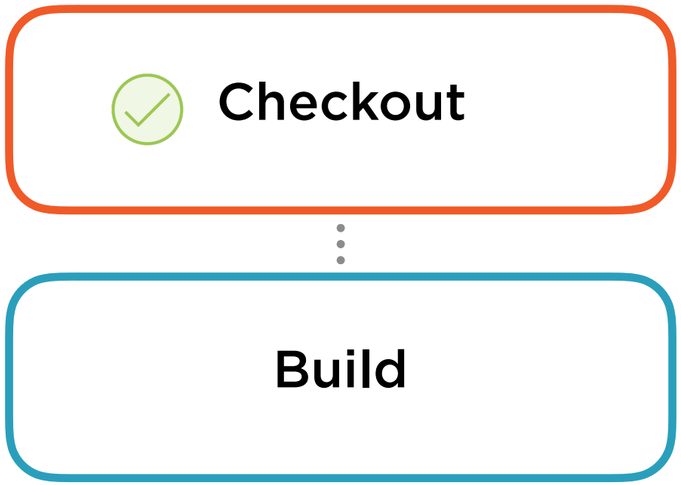
Enrico Campidoglio

@ecampidoglio megakemp.com

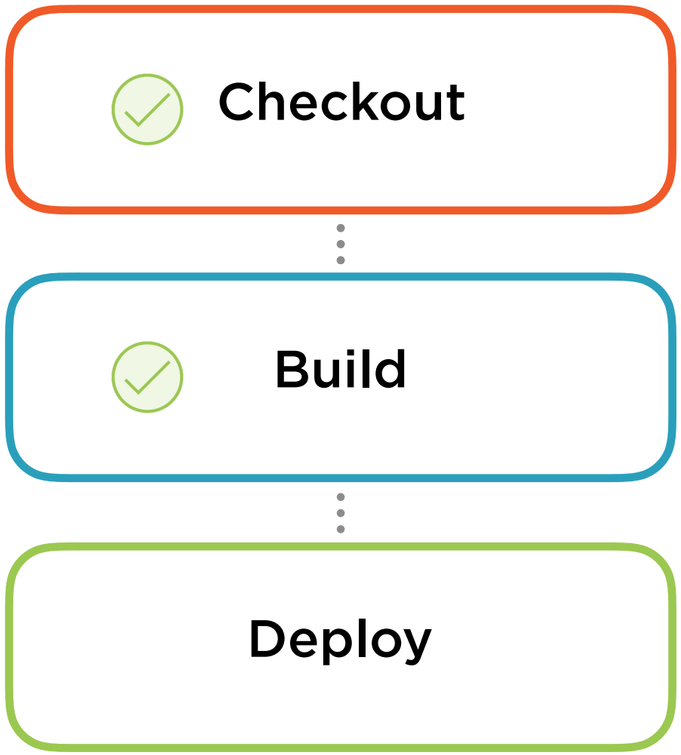
When Should You Build a Custom Action?

```
jobs:
  build_and_deploy:
    steps:
```

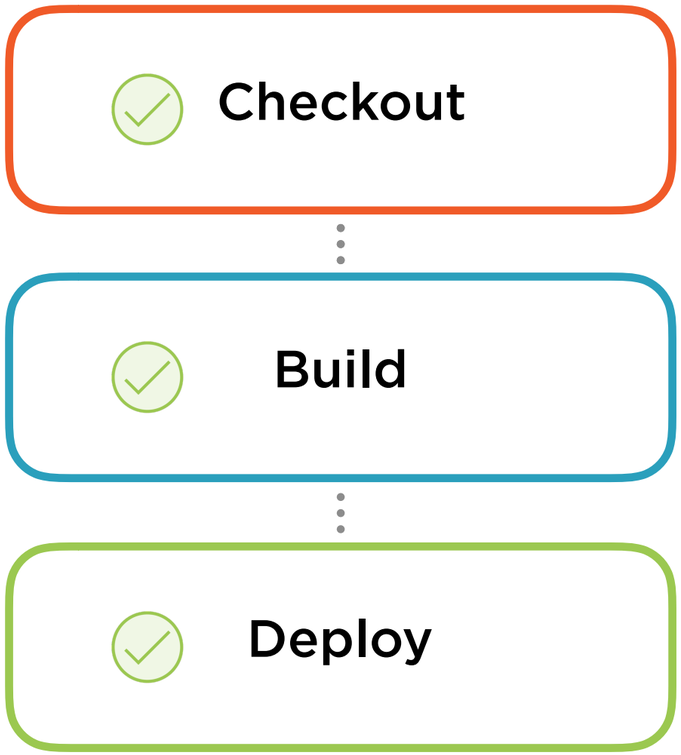
```yaml
jobs:
  build_and_deploy:
    steps:
      - name: Checkout
        uses: actions/checkout@v1
```



```yaml
jobs:
  build_and_deploy:
    steps:
      - name: Checkout
        uses: actions/checkout@v1

      - name: Build
        run: make all
```

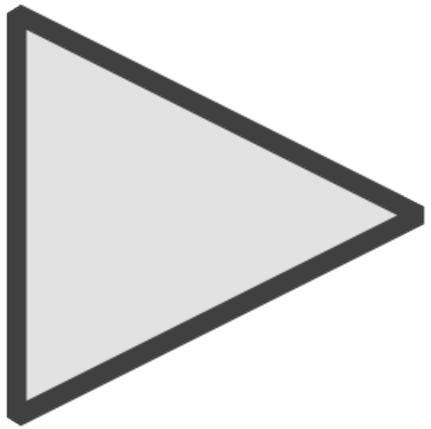


```yaml
jobs:
  build_and_deploy:
    steps:
      - name: Checkout
        uses: actions/checkout@v1

      - name: Build
        run: make all

      - name: Deploy
        uses: actions/deploy@v1
        with:
          environment: Staging
```



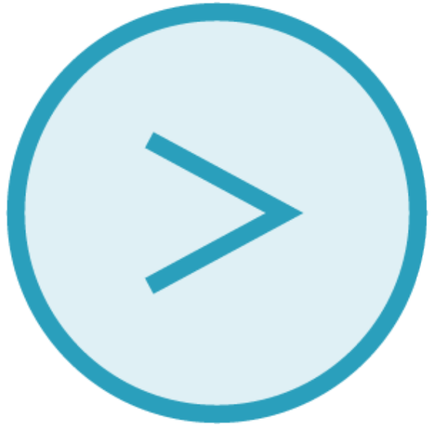
```yaml
jobs:
  build_and_deploy:
    steps:
      - name: Checkout
        uses: actions/checkout@v1

      - name: Build
        run: make all

      - name: Deploy
        uses: actions/deploy@v1
        with:
          environment: Staging
```

Shell Commands vs. Actions



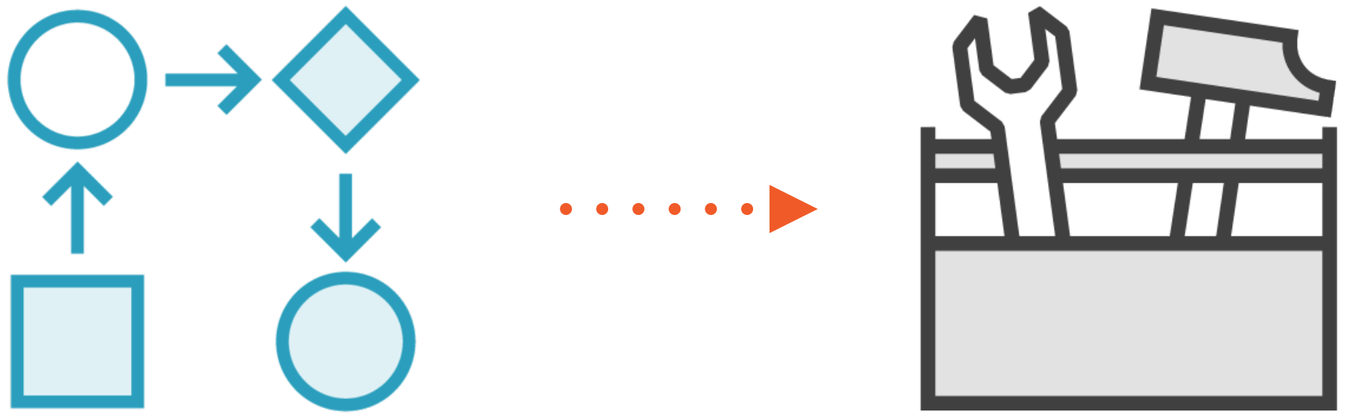
run:

uses:

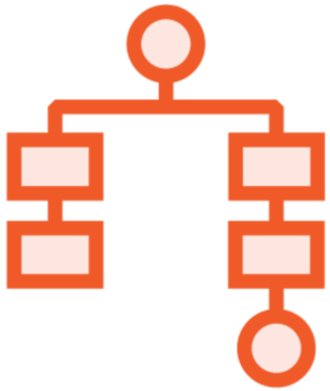


Adapter

Adapter

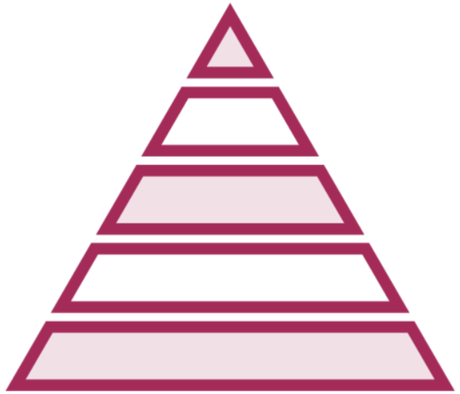


Shell Commands vs. Actions



Control Flow

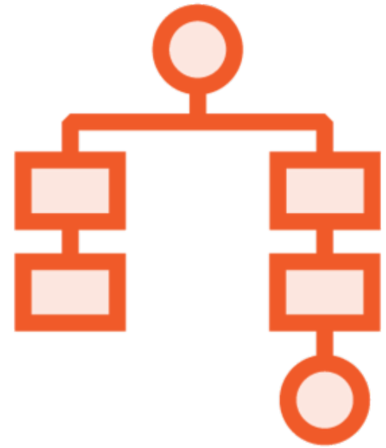
Diagnostic



Complexity

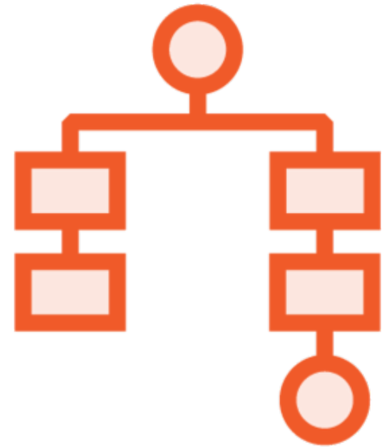
Reuse

```
- name: Build
  run: npm run build
```



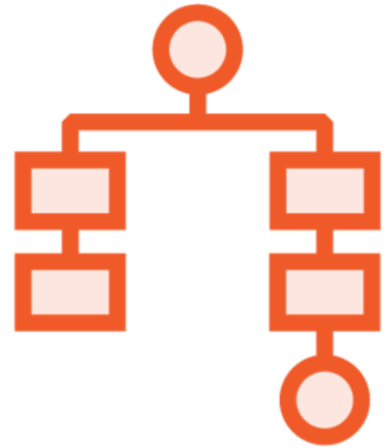
Control Flow

```yaml
- name: Build
  run: |
    npm ci
    npm run build
```



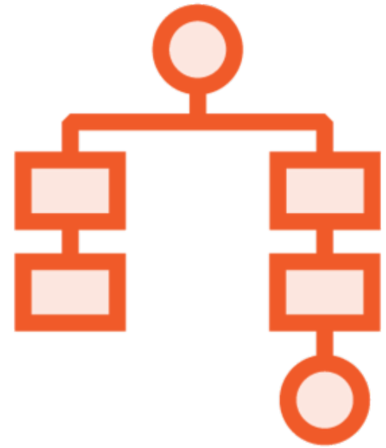
Control Flow

```
- name: Build
  run: |
    npm ci
    npm run build
```



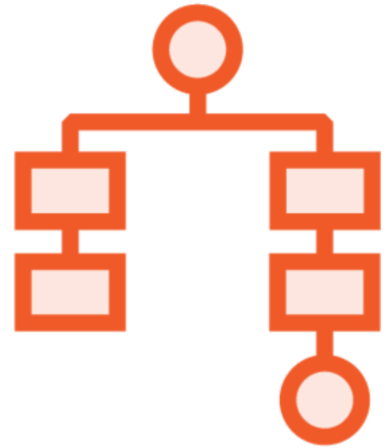
Control Flow

```
- name: Build
  run: |
    npm ci
    npm run build
```



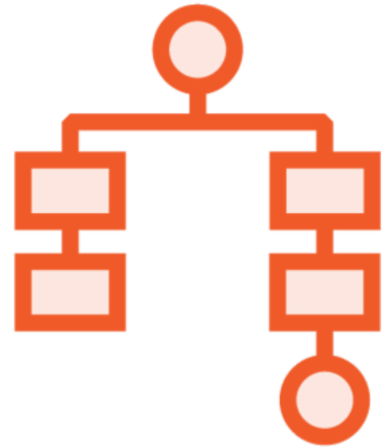
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
```



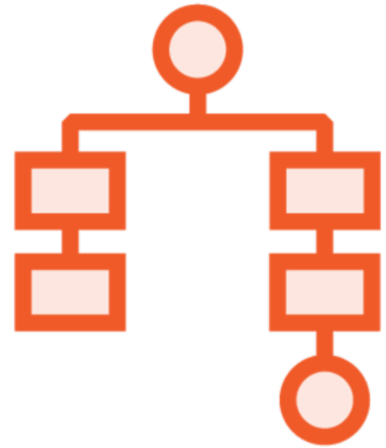
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/project
```



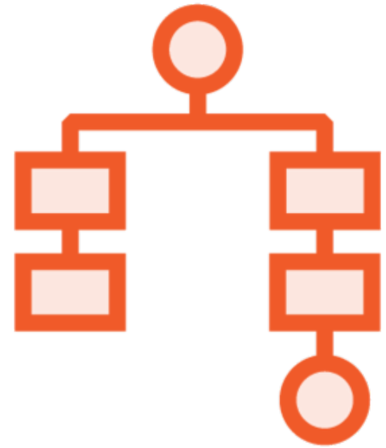
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/project
```



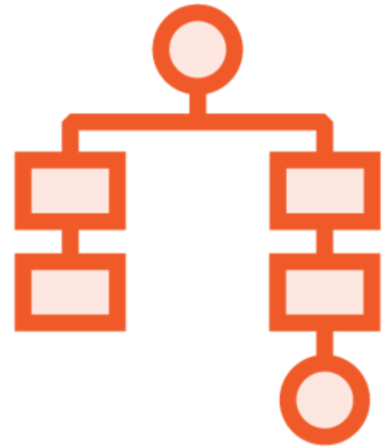
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/project
- name: Build another
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/another
- name: Build yet another
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/yet_another
```



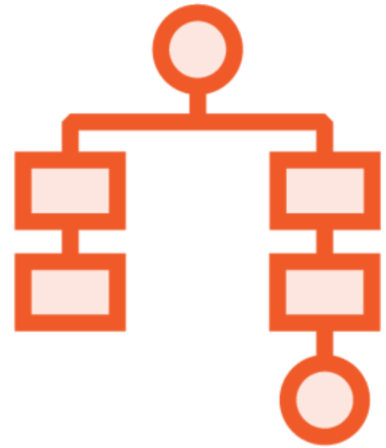
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/project
- name: Build another
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/another
- name: Build yet another
  run: |
    npm ci && npm run build
  working-directory: |
    ./src/yet_another
```



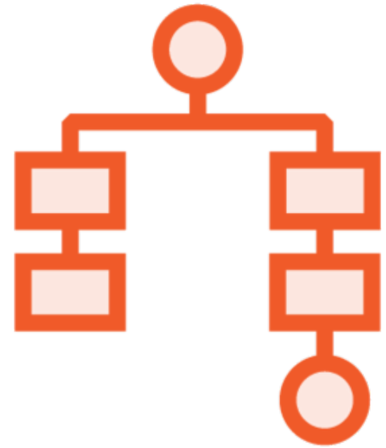
Control Flow

```yaml
- name: Build
  uses: actions/npm-build@v1
  with:
    working-directories:
      - src/project
      - src/another
      - src/yet-another
```



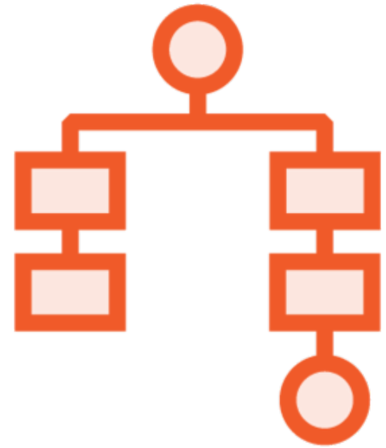
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
```



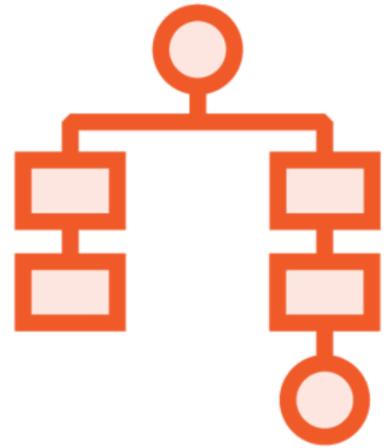
Control Flow

```
- name: Build
  run: |
    npm ci && npm run build
```



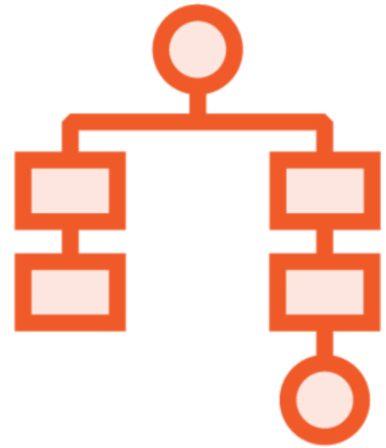
Control Flow

```
jobs:
```



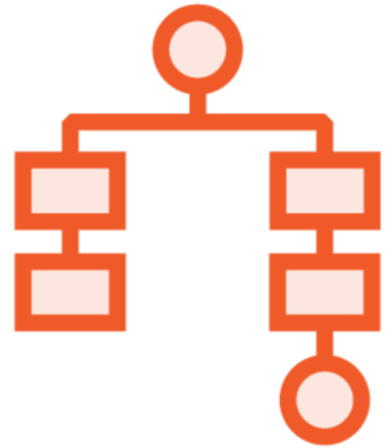
Control Flow

```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
```



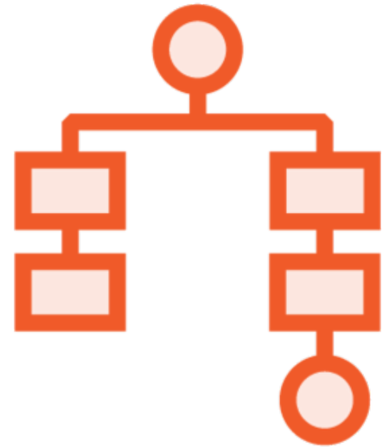
Control Flow

```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
  build:
    steps:
      - name: Build
        run: npm run build
```



Control Flow

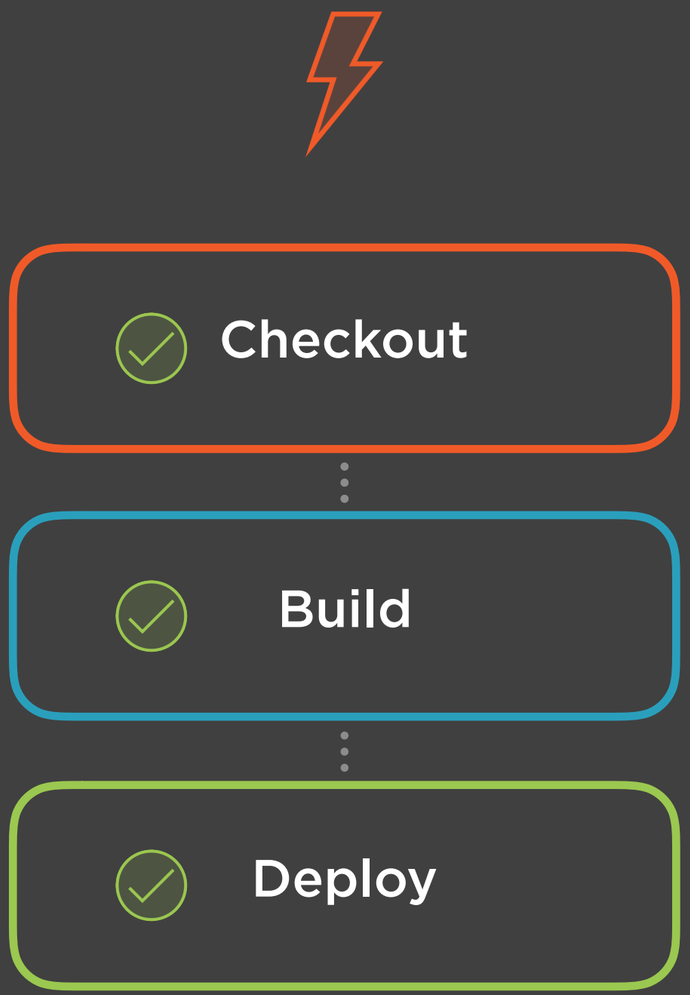
```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
  build:
    needs: install
    steps:
      - name: Build
        run: npm run build
```

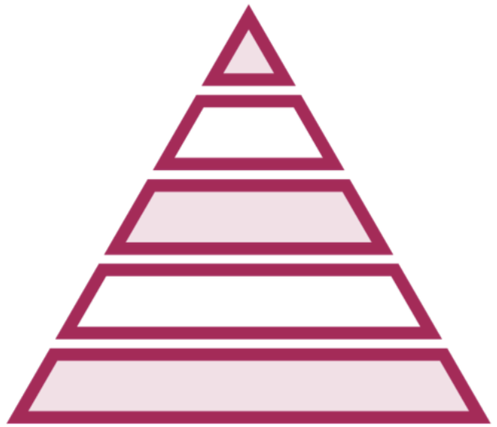


Control Flow

```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
  build:
    needs: install
    steps:
      - name: Build
        run: npm run build
  error:
    steps:
      - name: Handle error
        run: ...
```

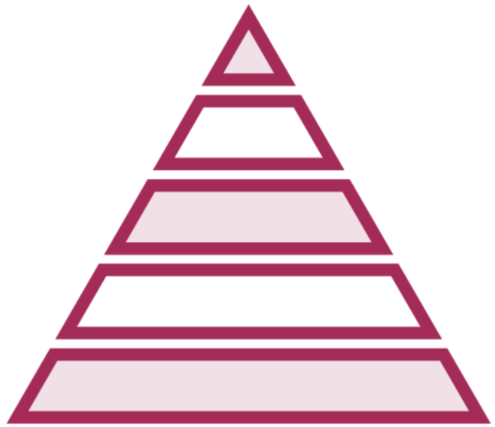
Diagnostic





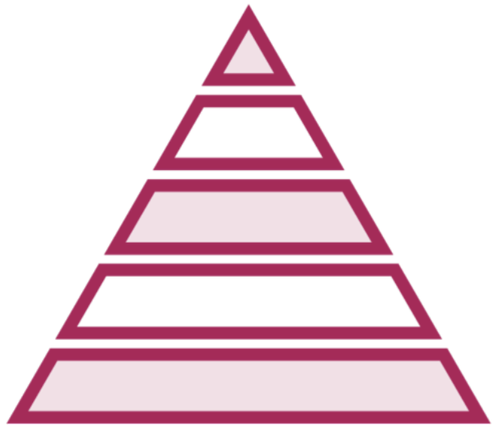
Complexity

```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
  build:
    needs: install
    steps:
      - name: Build
        run: npm run build
```



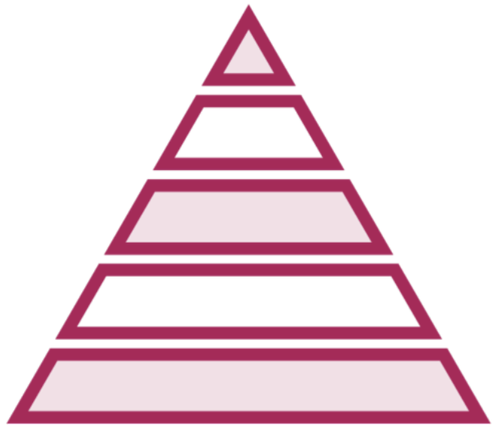
Complexity

```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
        working_directory: ..
  build:
    needs: install
    steps:
      - name: Build
        run: npm run build
        working_directory: ..
```



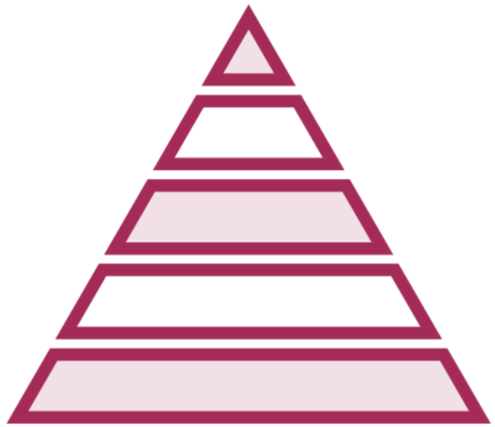
Complexity

```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
        working_directory: ..
  build:
    needs: install
    steps:
      - name: Build
        run: npm run build
        working_directory: ..
```



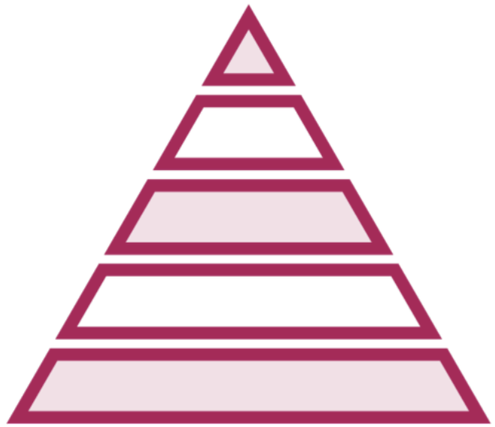
Complexity

```
- name: Deploy
  env:
    USER: ${{ secrets.USER }}
    PWD: ${{ secrets.PWD }}
  run: |
    curl -Ss \
    --fail \
    -X POST \
    -T path/to/app.zip \
    -u $USER:$PWD \
    https://example.com/app \
```



Complexity

```
- name: Deploy
  env:
    USER: ${{ secrets.USER }}
    PWD: ${{ secrets.PWD }}
    VER: |
      ${{ steps.id.outputs }}
  run: |
    curl -Ss \
    --fail \
    -X POST \
    -T path/to/app-$VER.zip \
    -u $USER:$PWD \
    https://example.com/app \
```



Complexity

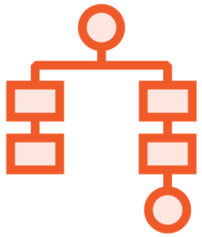
```yaml
jobs:
  install:
    steps:
      - name: Clean install
        run: npm ci
  build:
    needs: install
    steps:
      - name: Build
        run: npm run build
  deploy:
    needs: build
    steps:
      - name: Deploy
        run: npm publish
```

Reuse

```yaml
jobs:
  build_and_deploy:
    steps:
      - name: Build & Deploy
        uses: ./npm-deploy@v1
  build_and_deploy_another:
    steps:
      - name: Build & Deploy
        uses: ./npm-deploy@v1
  and_another:
    steps:
      - name: Build & Deploy
        uses: ./npm-deploy@v1
```

Reuse

When Should You Build a Custom Action?



You need more control flow structures than simple `if` conditions

You need to handle errors to make them easier to diagnose

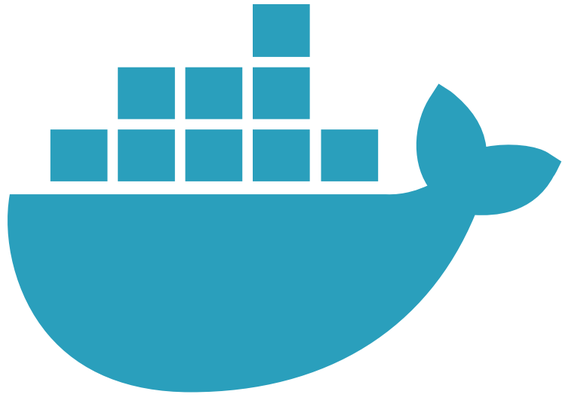
You need to run complex commands or workflows

You need to reuse the same logic

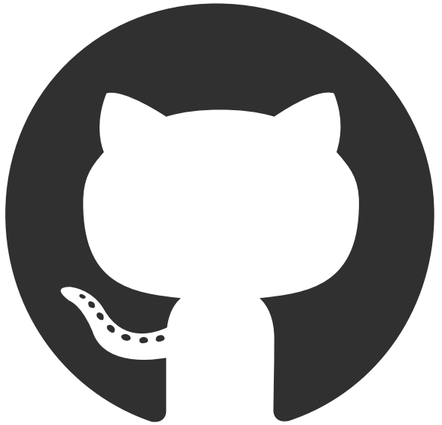
Course Overview

Understanding GitHub Actions

Creating a JavaScript action



Creating a Docker action



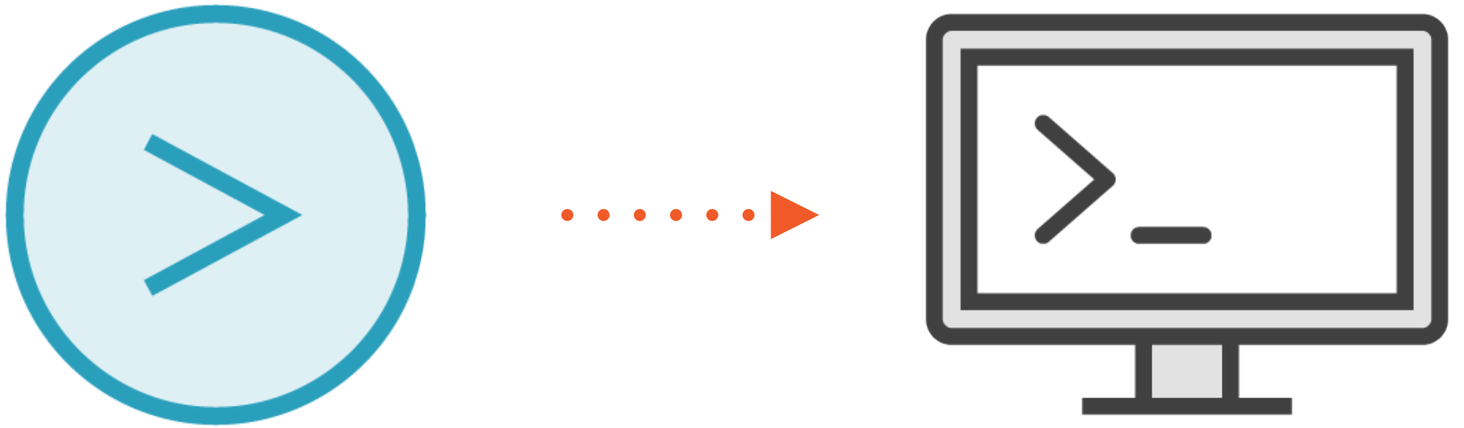
Publishing to the GitHub Marketplace

The Command-line Metaphor

```
$ program param=arg
Running the program...
Here's what I'm doing
I'm done.
$ output=$(program)
$ echo $output
Here's the output
$ echo $?
0 ✅
1 ❌
```

```
- name: Run action
  uses: action@v1
  with:
    param: arg

1 Run action
2 > action@v1 arg

${{ steps.id.outputs.name }}

Here's the output


exit 0 ✅
exit 1 ❌
```

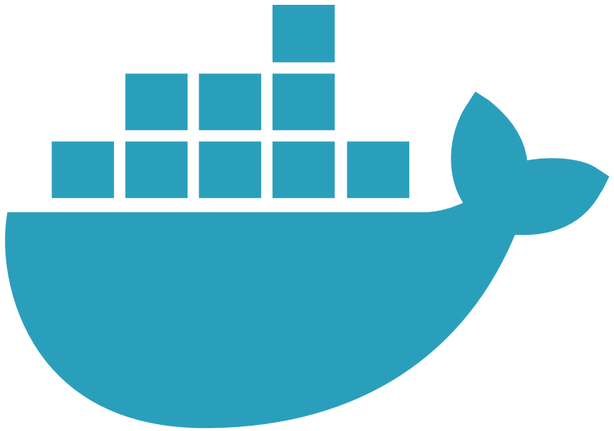
Actions run in
their own **process**



JavaScript and Docker Actions

Two Kinds of Actions

JavaScript Actions



Docker Actions

JavaScript Actions

Fast

They run directly on the host machine

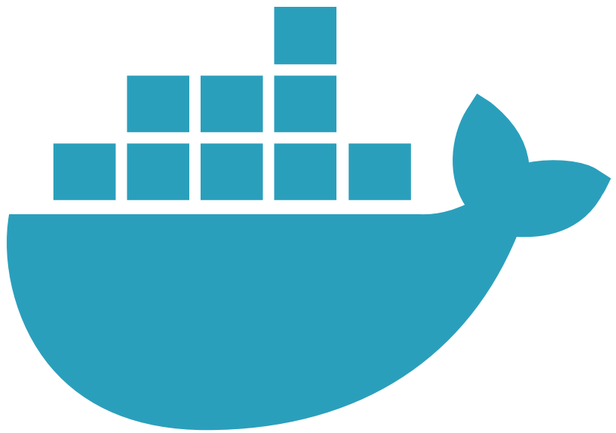
Development Support

Easy to integrate with the GitHub environment

Require Setup

Any external dependencies must be installed

Docker Actions



Flexible

They run virtually any software

Consistent

They come with their own environment

Slower

It takes time to download and start a container

Linux only

They can only run on Linux hosts

The Action Metadata


Metadata
YAML
action.yml

```
name: 'Awesome Action'
```

```
name: 'Awesome Action'
description: 'An action that does awesome things.'
```

```
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
    required: true
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
    required: false
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
    required: false
    default: 'The default argument'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using:
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
  main: 'lib/main.js'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
  main: 'lib/main.js'
```


```yaml
runs:
  using: 'docker'
  image: 'Dockerfile'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
  main: 'lib/main.js'
```


```yaml
runs:
  using: 'docker'
  image: 'docker://image:tag'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
  main: 'lib/main.js'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
  main: 'lib/main.js'
branding:
  icon: 'power'
  color: 'yellow'
```

```yaml
name: 'Awesome Action'
description: 'An action that does awesome things.'
author: 'Enrico Campidoglio'
inputs:
  parameter-name:
    description: 'An input parameter.'
outputs:
  output-value-name:
    description: 'A value returned by the action.'
runs:
  using: 'node12'
  main: 'lib/main.js'
branding:
  icon: 'power'
  color: 'yellow'
```

Communicating with the Host


Build Log

Build Log

```
Current runner version: '2.160.2'
Prepare workflow directory
Prepare all required actions
Download action repository ...
Setting environment variables ...
::set-env name=DEBUG::1
[command]program --param=arg
Running program with 'arg'...
Done
```

Build Log

```
Current runner version: '2.160.2'
Prepare workflow directory
Prepare all required actions
Download action repository ...
Setting environment variables ...
::set-env name=DEBUG::1
[command]program --param=arg
Running program with 'arg'...
Done
```

Build Log

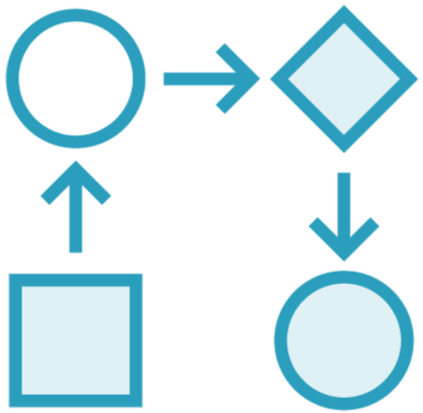
```
Current runner version: '2.160.2'
Prepare workflow directory
Prepare all required actions
Download action repository ...
Setting environment variables ...
::set-env name=DEBUG::1
[command]program --param=arg
Running program with 'arg'...
Done
```

Build Log

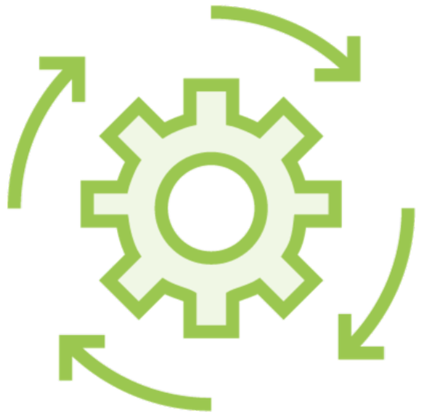
```
Current runner version: '2.160.2'
Prepare workflow directory
Prepare all required actions
Download action repository ...
Setting environment variables ...
env:
  DEBUG: 1
[command]program --param=arg
Running program with 'arg'...
Done
```

Logging Commands

Are instructions for the workflow host that are embedded in the build log.



Workflow



Host


Workflow
Build Log
Host

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

The Logging Command Format

```
::log-command param1=arg,param2=arg::command value
```

Setting an Environment Variable

```
::log-command param1=arg,param2=arg::command value
```

Setting an Environment Variable

```
::set-env param1=arg,param2=arg::command value
```

Setting an Environment Variable

```
::set-env param1=arg::command value
```

Setting an Environment Variable

```
::set-env name=arg::command value
```

Setting an Environment Variable

```
::set-env name=DEBUG::command value
```

Setting an Environment Variable

```
::set-env name=DEBUG::1
```

Setting an Environment Variable

```
::set-env name=DEBUG::1
```

Setting an Output Parameter

```
::set-output name=name_of_param::value
```

Setting an Output Parameter

```
::set-output name=name-of-param::value
```

Setting an Output Parameter

```
::set-output name=name of param::value
```

Adding a Directory to the **PATH**

```
::add-path::/path/to/directory
```

Printing a Log Message

```
::debug::message
```

Printing a Log Message

```
::debug file=name,line=0,col=0::message
```

Printing a Log Message

```
::debug file=name,line=0,col=0::message
```

Printing a Log Message

```
::debug file=name,line=0,col=0::message
```

```
::warning file=name,line=0,col=0::message
```

Printing a Log Message

```
::debug file=name,line=0,col=0::message
```

```
::warning file=name,line=0,col=0::message
```

```
::error file=name,line=0,col=0::message
```

Printing a Masked Message

```
::add-mask::message
```

Printing a Masked Message

```
::add-mask::*******
```

Printing a Masked Message

```
::add-mask::message
```

Printing a Masked Message

```
::add-mask::$VARIABLE
```

Stopping and Starting the Log Commands

```
::stop-comands::token
```

Stopping and Starting the Log Commands

```
⏸ ::stop-comands::token
...
Commands are ignored
...
```

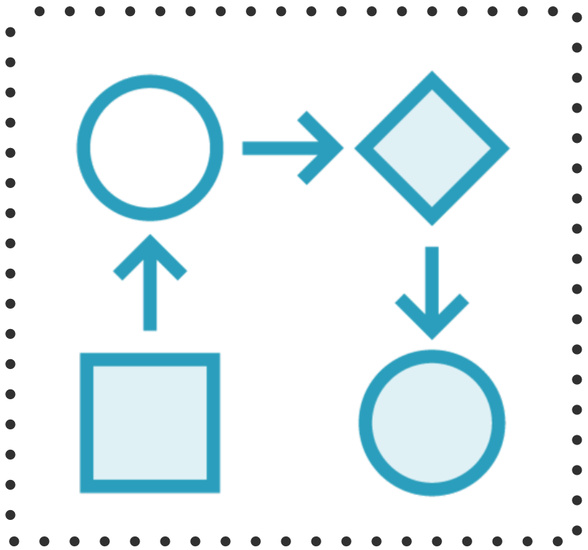
Stopping and Starting the Log Commands

```
::stop-comands::token
...
Commands are ignored
...
```

```
::token::
```

Public vs. Private Actions

GitHub Repository



uses:

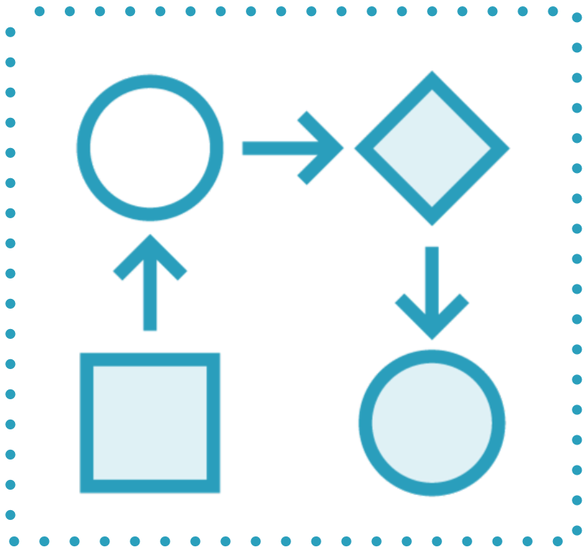
GitHub Repository

action.yml

main.js

Dockerfile

GitHub Repository



uses:

GitHub Repository

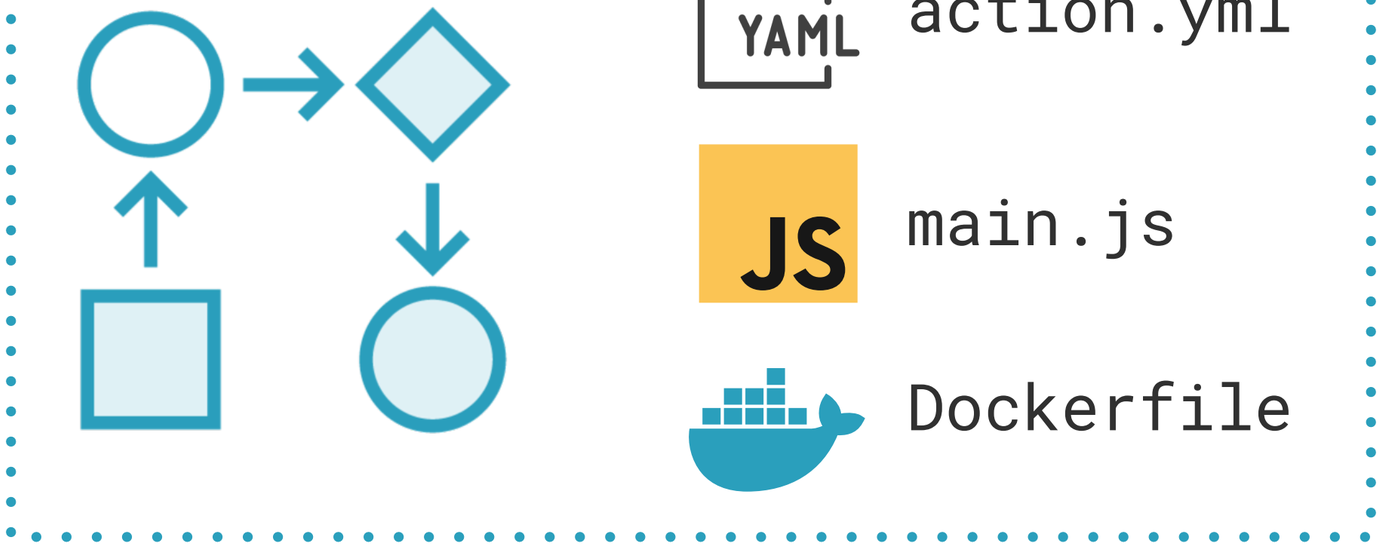
action.yml

main.js

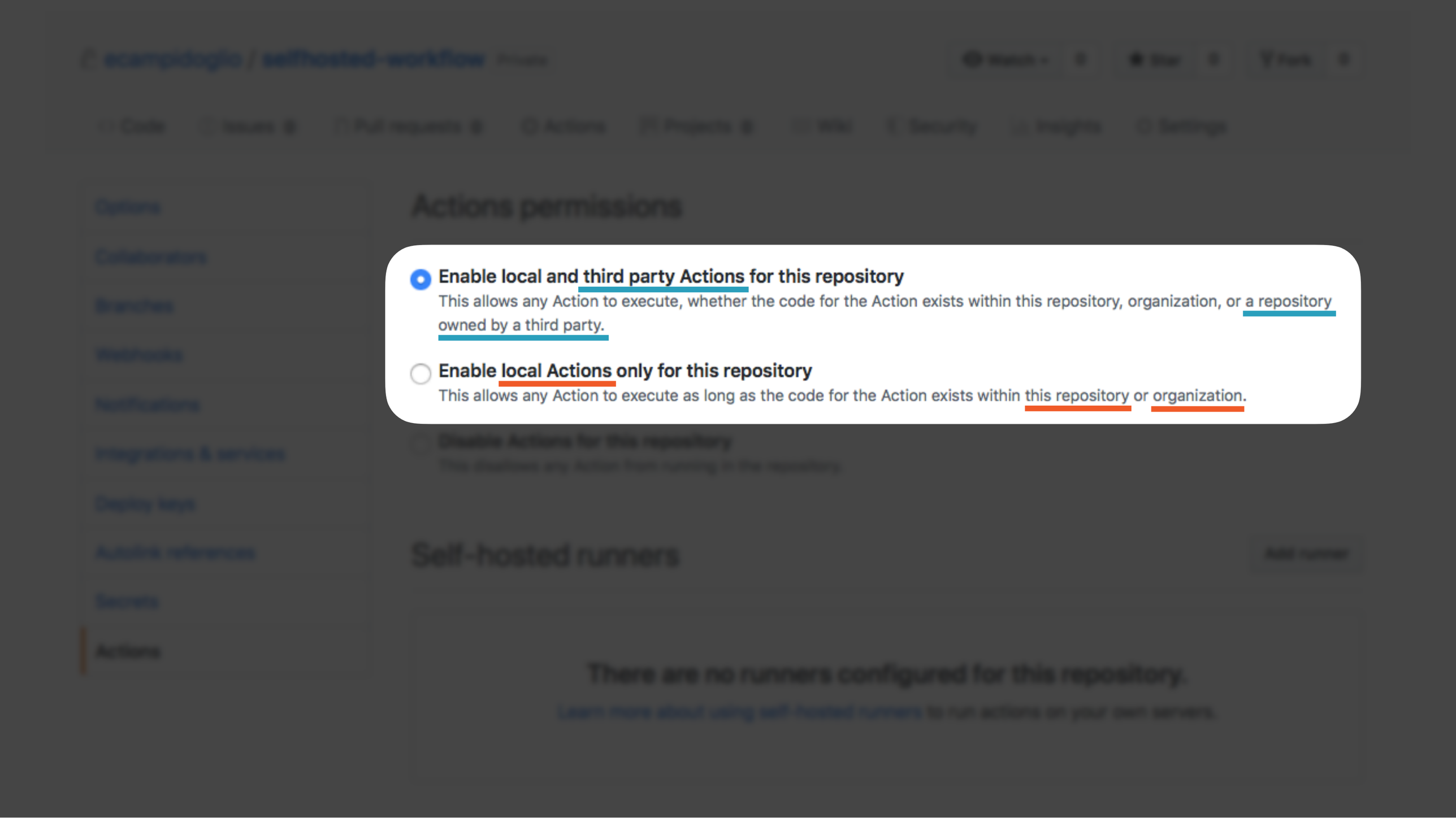
Dockerfile

Public Action

GitHub Repository



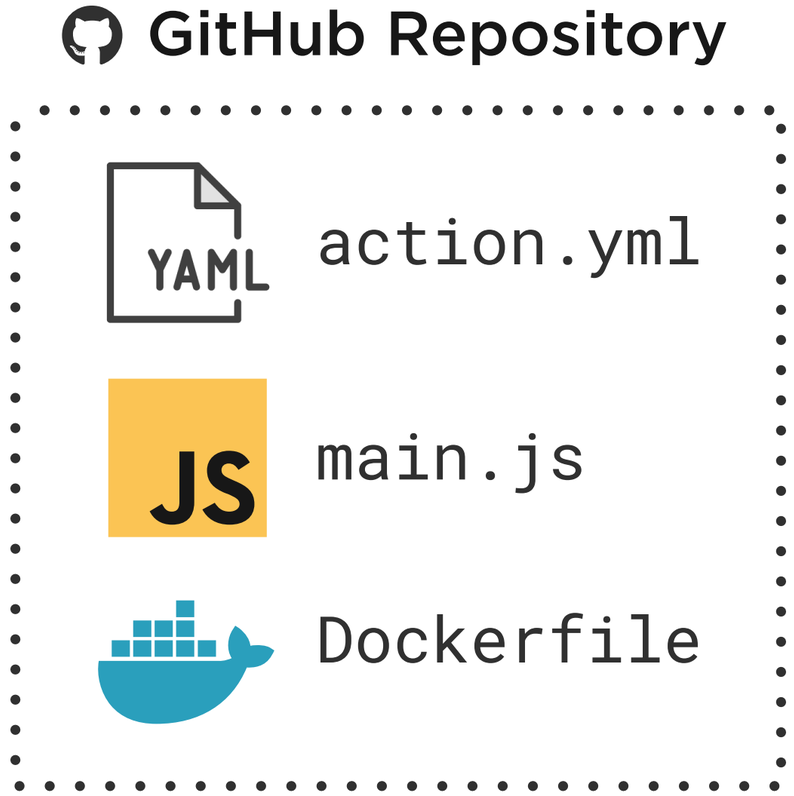
Private Action


Actions permissions
Enable local and third party Actions for this repository
This allows any Action to execute, whether the code for the Action exists within this repository, organization, or a repository owned by a third party.
Enable local Actions only for this repository
This allows any Action to execute as long as the code for the Action exists within this repository or organization.
Self-hosted runners
There are no runners configured for this repository.

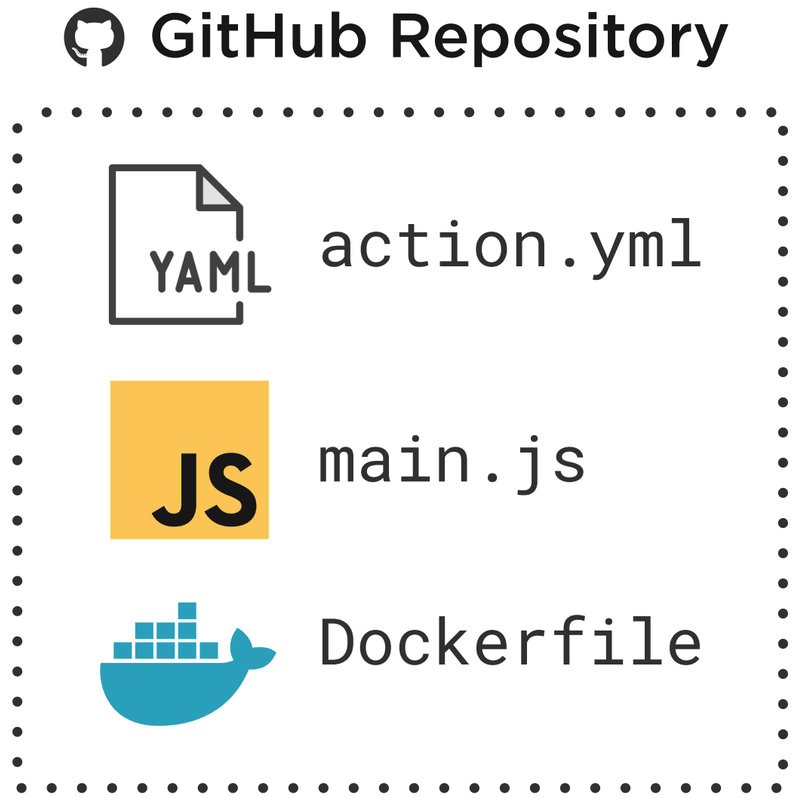
Public and private actions
are **referenced** differently
from a workflow

```
- name: Public action
```

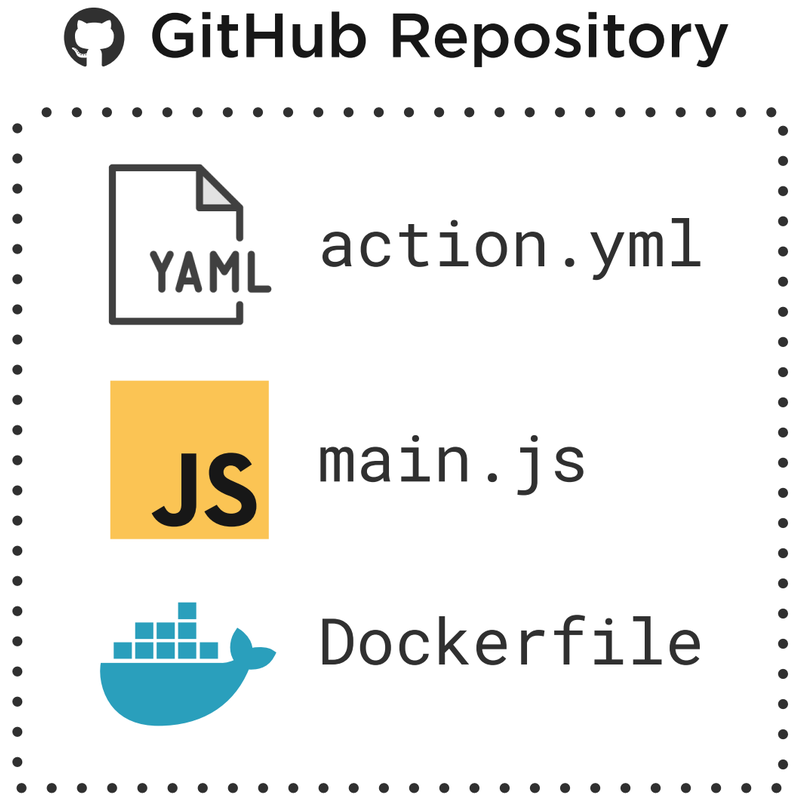
```
- name: Public action
```



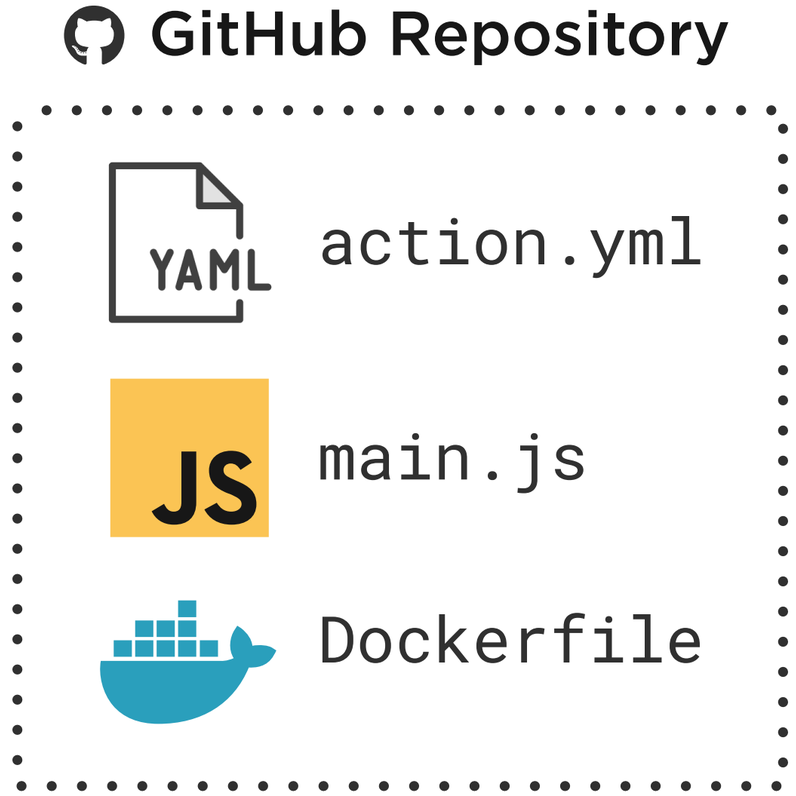
```yaml
- name: Public action
  uses: owner/repo
```



```yaml
- name: Public action
  uses: owner/repo@ref
```

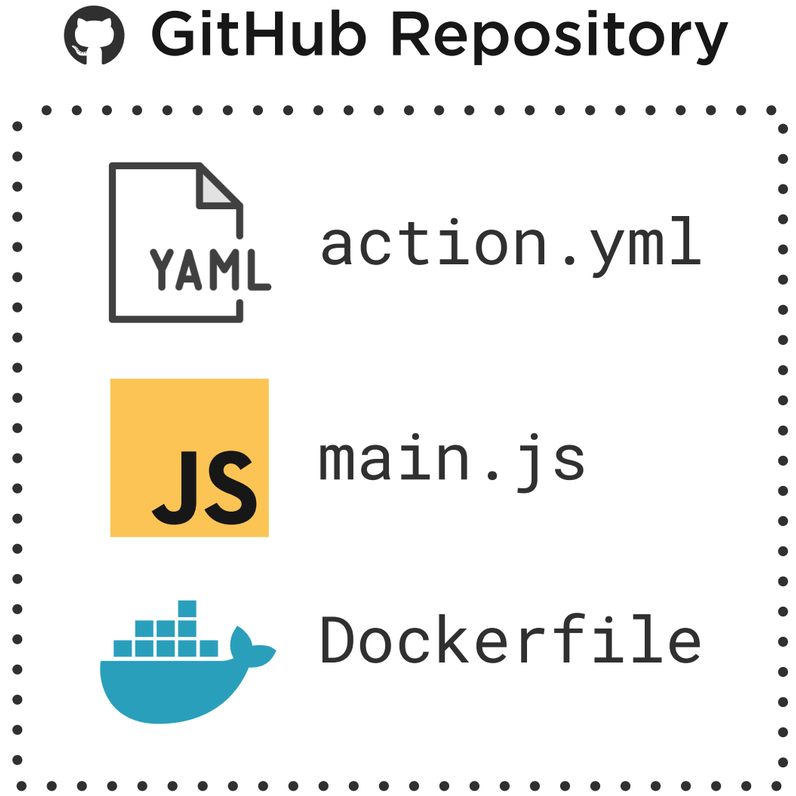


```yaml
- name: Public action
  uses: owner/repo@ref
```



```yaml
- name: Public action
  uses: owner/repo@12az34
```

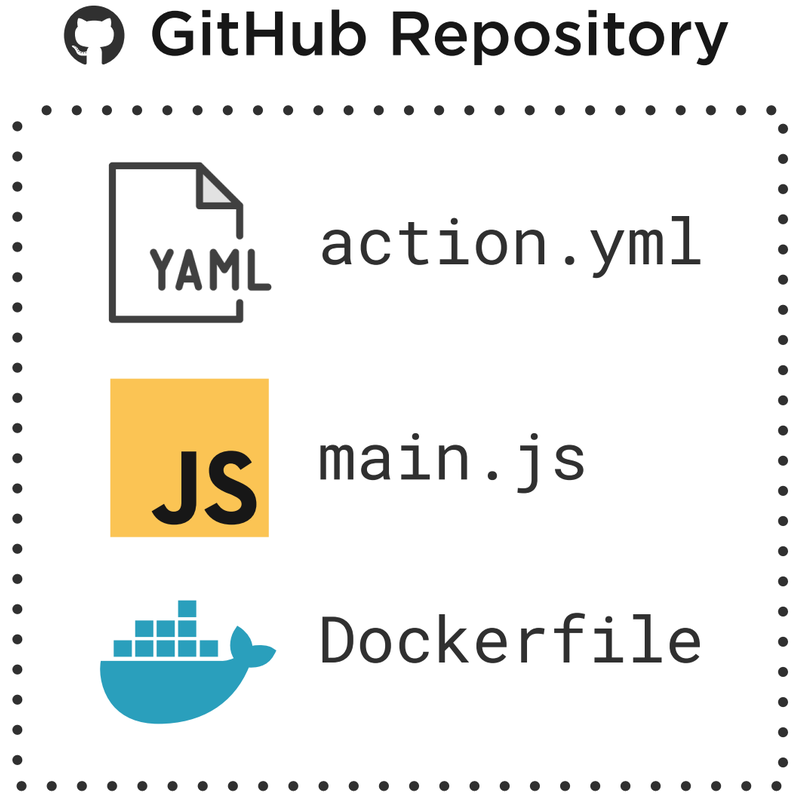
commit SHA



```
- name: Public action
  uses: owner/repo@v1
```

commit SHA

version number



Semantic Versioning 2.0.0

Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Introduction

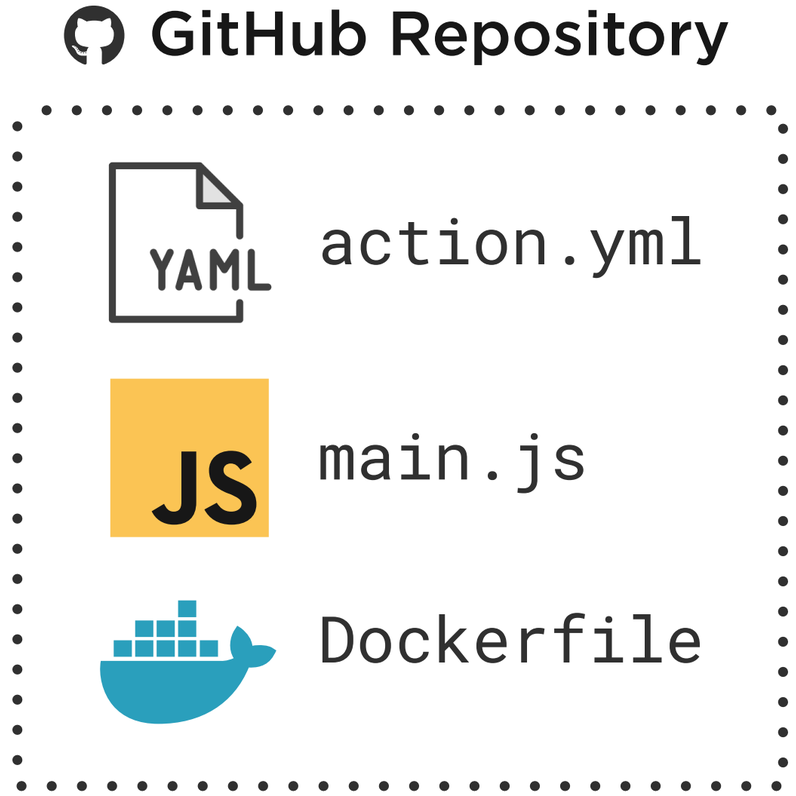
In the world of software management there exists a dreaded place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself, one day, in this pit of despair.

In systems with many dependencies, releasing new package versions can quickly become a nightmare. If the dependency specifications are too tight, you are in danger of version lock (the inability to upgrade a package without having to release new versions of every dependent package). If dependencies are specified too loosely, you will inevitably be bitten by version promiscuity (assuming compatibility with more future versions than is reasonable). Dependency hell is where you are when version lock and/or version promiscuity prevent you from easily and safely moving your project forward.

```
- name: Public action
  uses: owner/repo@v1
```

commit SHA

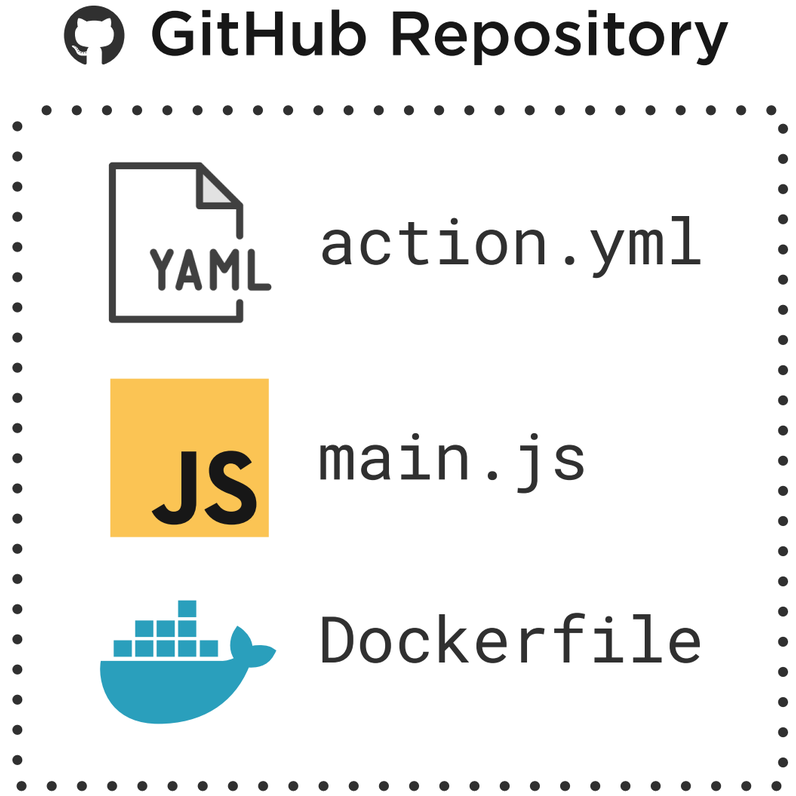
version number



```
- name: Public action
  uses: owner/repo@v1.2
```

commit SHA

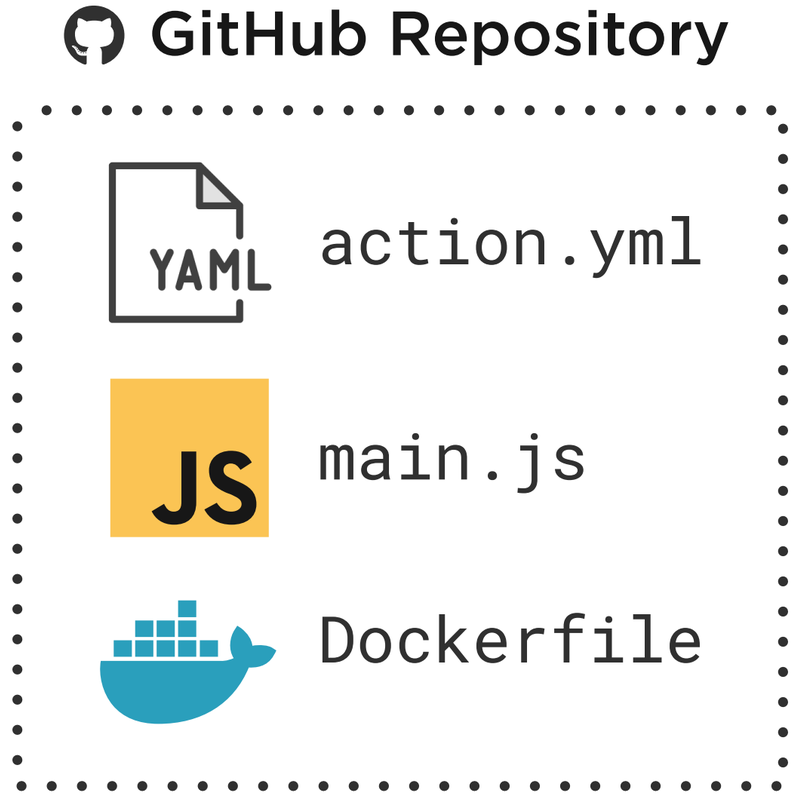
version number



```
- name: Public action
  uses: owner/repo@v1.2.3
```

commit SHA

version number

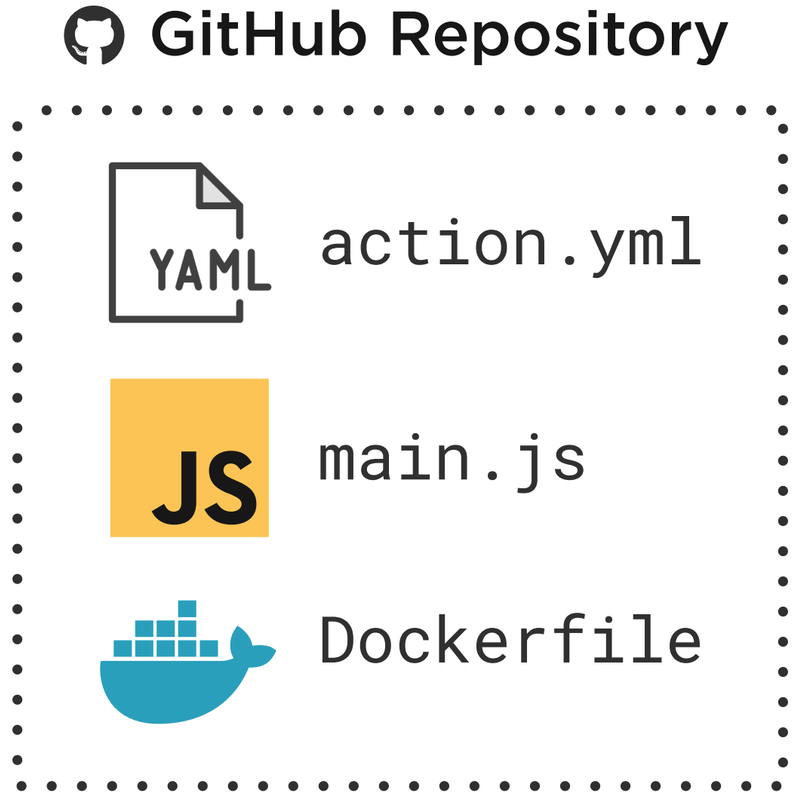


```
- name: Public action
  uses: owner/repo@master
```

commit SHA

version number

branch name

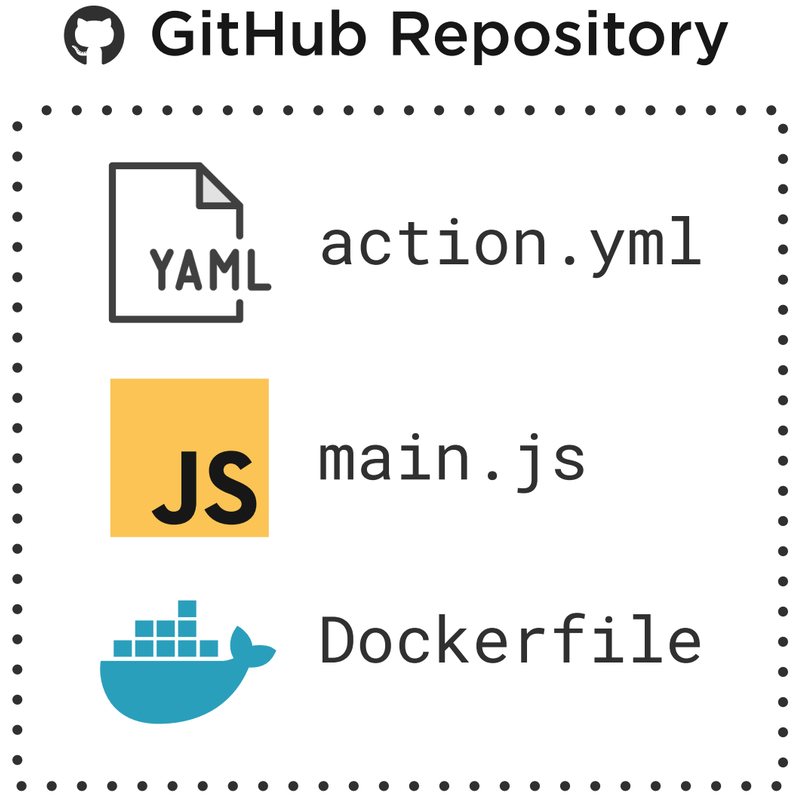


```
- name: Public action
  uses: owner/repo@feature
```

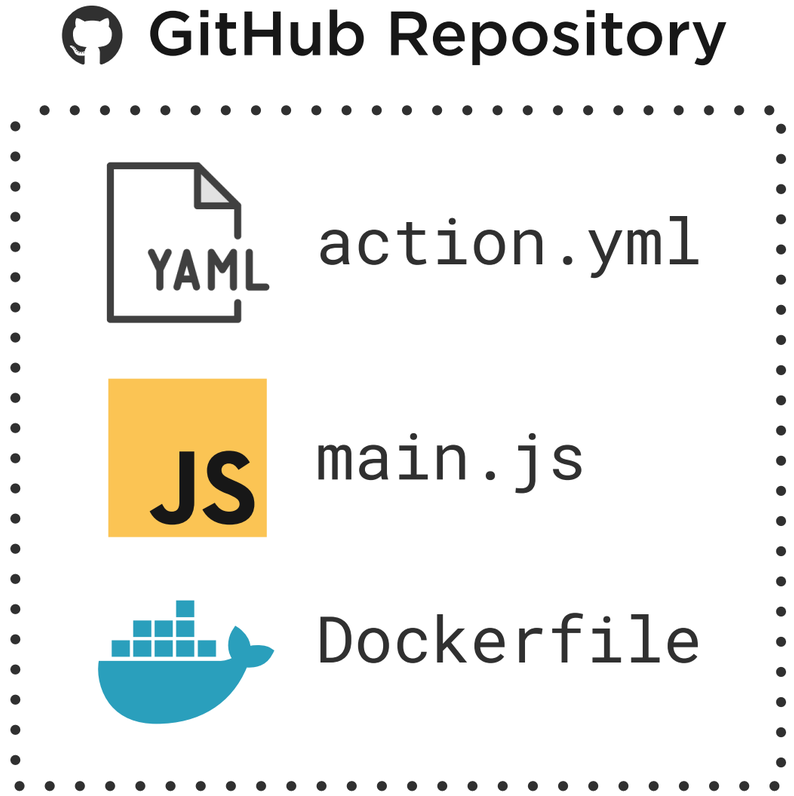
commit SHA

version number

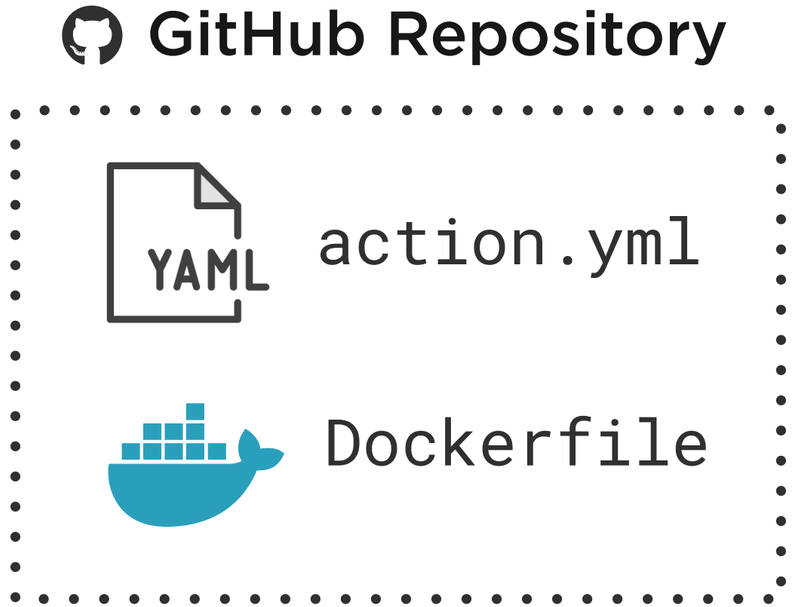
branch name



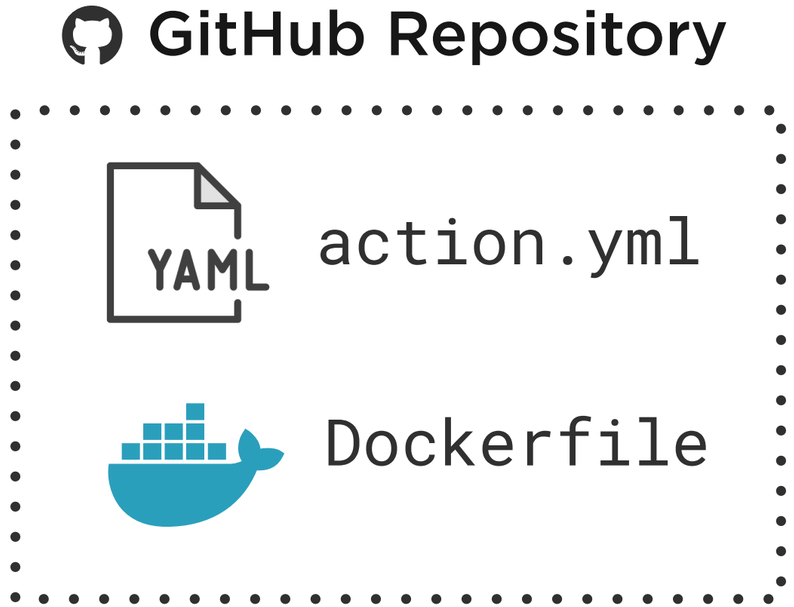
```
- name: Public action
```



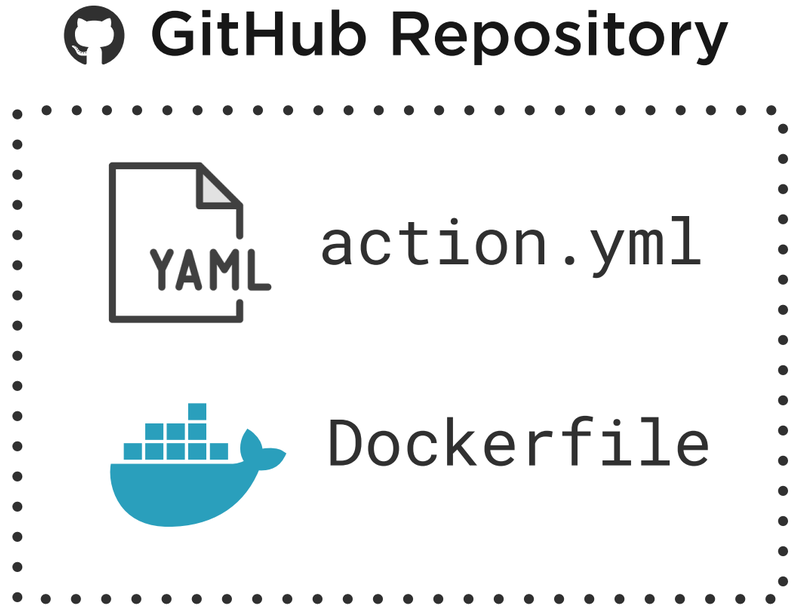
```
- name: Public action
```



```
- name: Public action
```

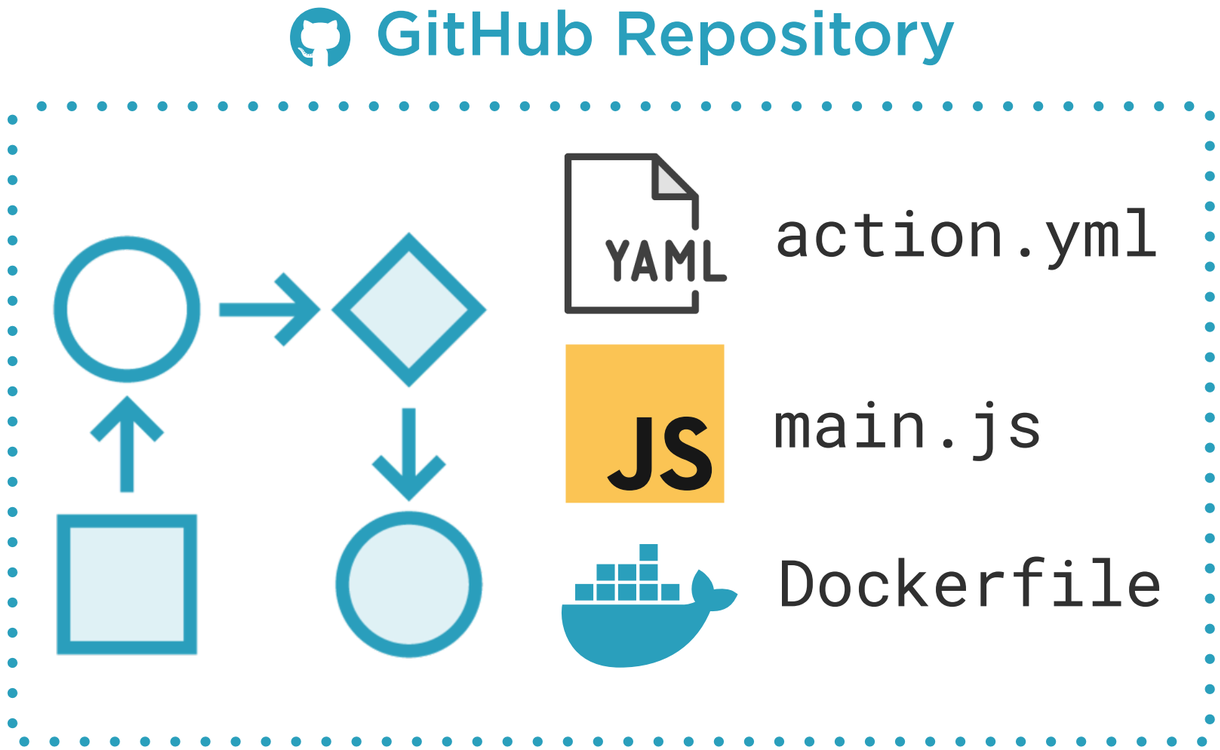


```yaml
- name: Public action
  uses: docker://image:tag
```

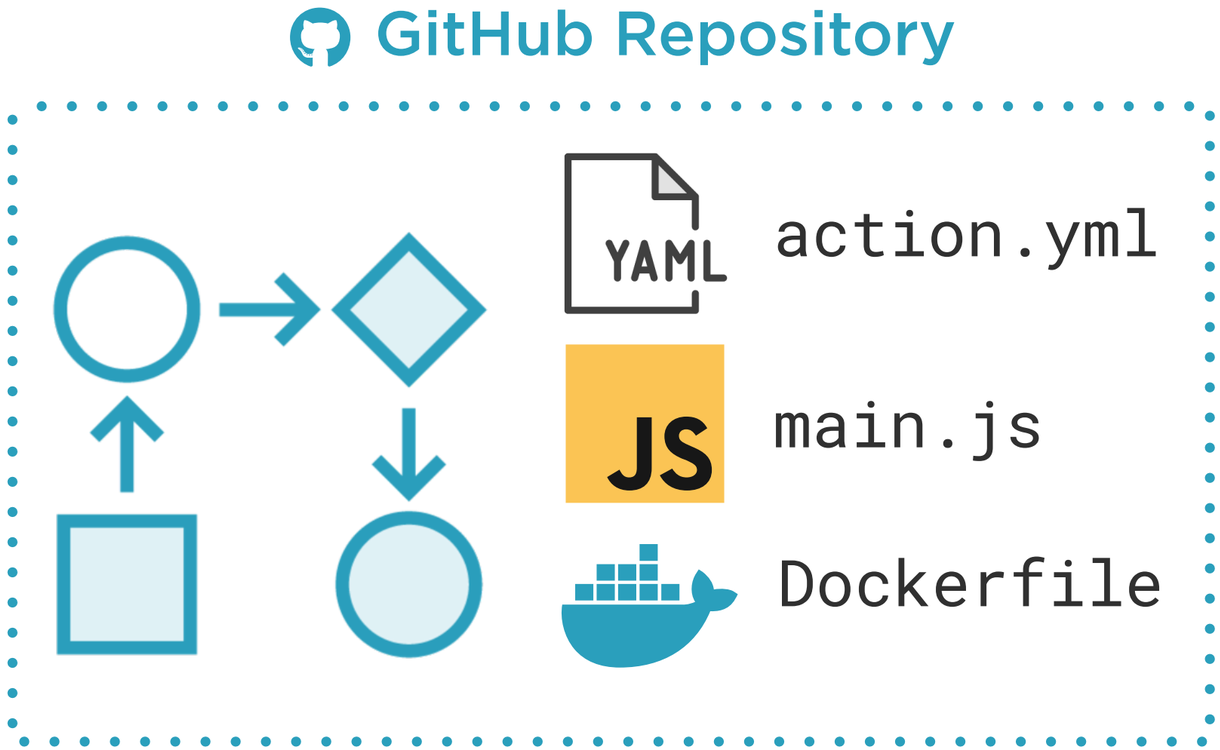


```
- name: Private action
```

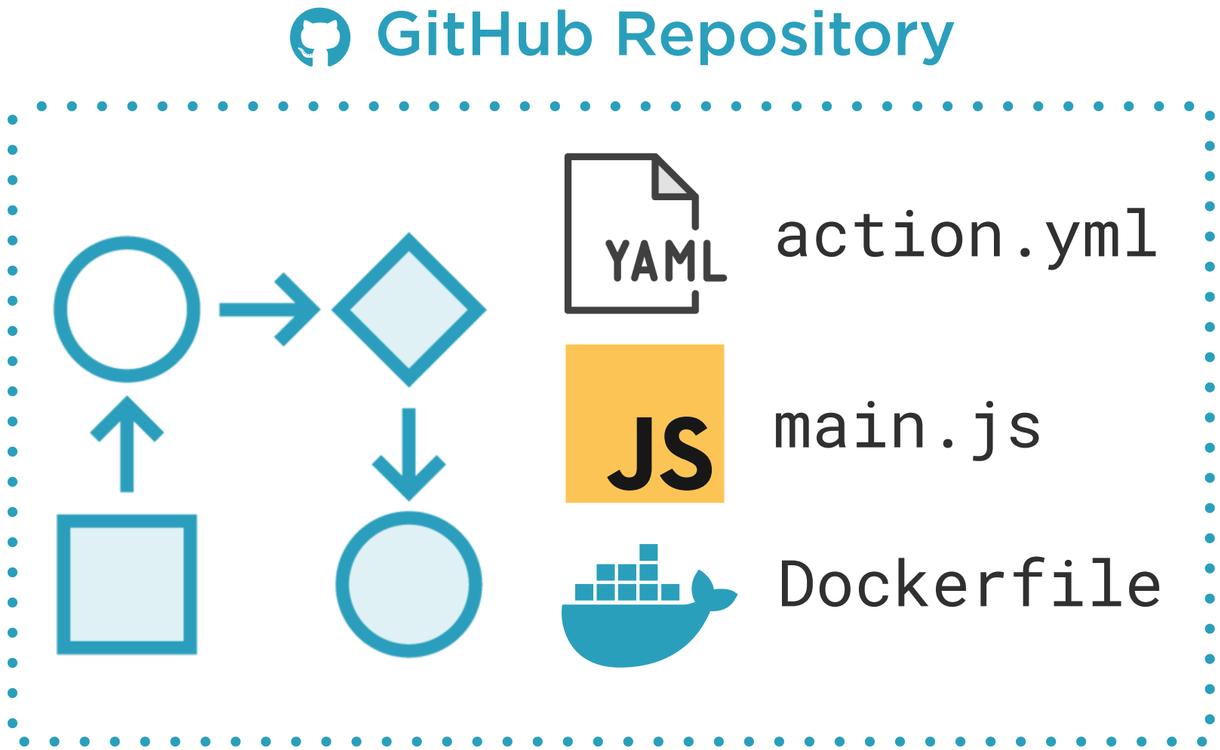
```
- name: Private action
```



```yaml
- name: Private action
  uses: ./path/to/action
```



```yaml
- name: Private action
  uses: ./.github/actions/...
```

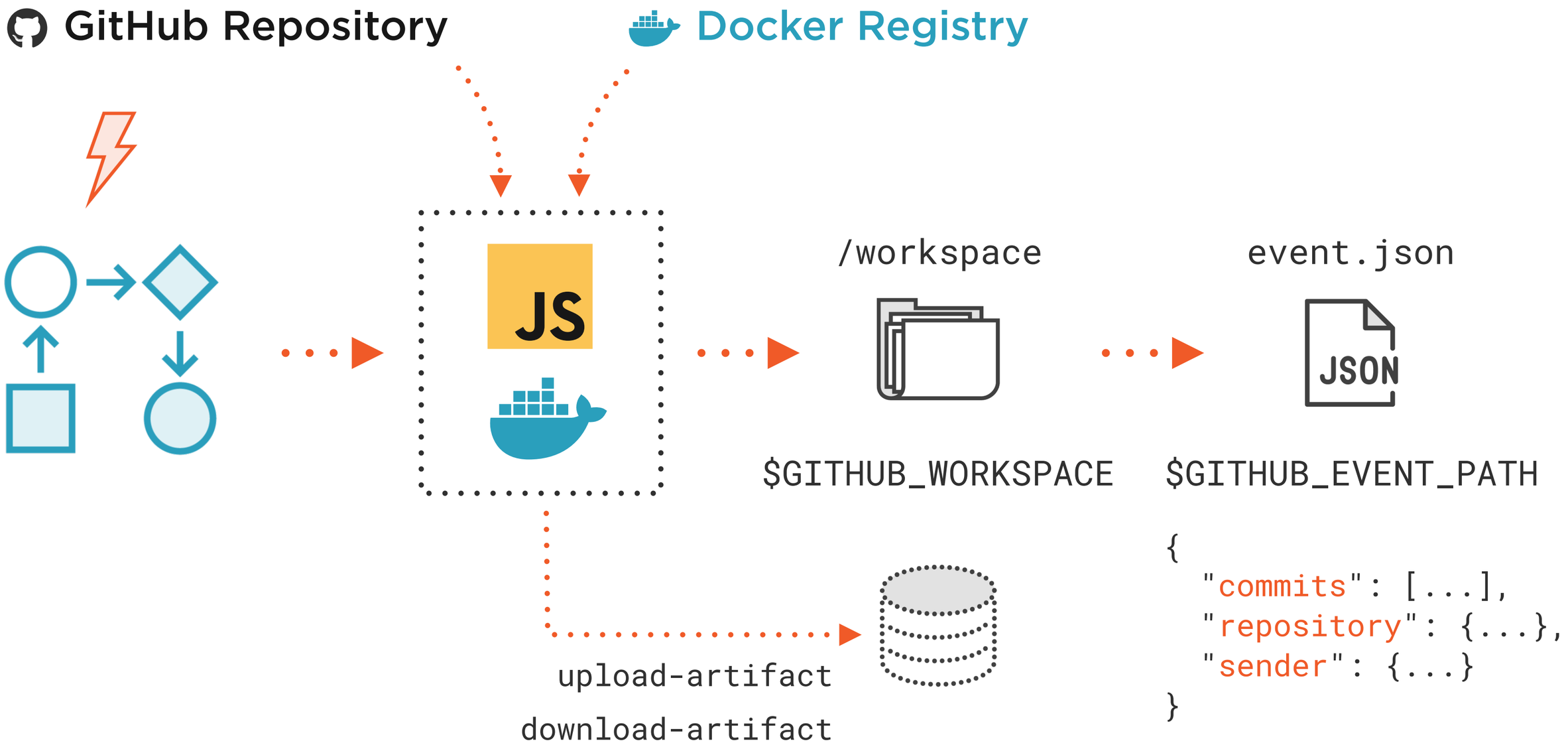


Who would **benefit** from
your custom GitHub action?

Public Action

Private Action

What Happens at Runtime


GitHub Repository
Docker Registry
JS
/workspace
event.json
JSON
$GITHUB_WORKSPACE
$GITHUB_EVENT_PATH
{
"commits": [...],
"repository": {...},
"sender": {...}
}
upload-artifact
download-artifact

Summary

- When to build a custom GitHub Action
- GitHub Actions are like CLI programs
- JavaScript and Docker actions
- Action Metadata
- How actions interact with the host
- Public vs. private actions
- What happens at runtime