GitHub Actions

Context

- Workflows
    - Automated procedure
    - Made of one or more jobs
    - Can be **scheduled** or **event-triggered**

- Events
    - Triggers a workflow from a specific activity
    - Can be trigged on a **schedule**, **manual events**, & **web-hook events.**

- Jobs
    - Set of **steps** that execute on the same **runner**
    - By *default* will run in **parallel** when multiple jobs are detected
    - Can be configured **sequentially (non-parallel)** that are dependent other jobs

- Steps
    - **Individual task** that can run commands inside a job
    - Executes on the same runner
    - Allows actions in the **same** job to **share data** with each other

- Actions
    - **Standalone** commands that are combined in steps to create a job
    - Can create your own actions
    - Attach **community** created actions by the GitHub community
        - Included as a step

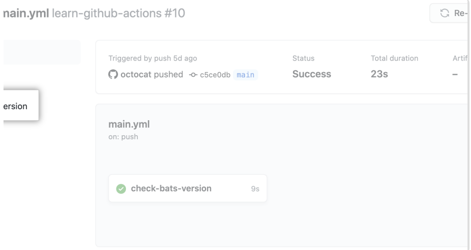
- Runners
    - Server that runs the **GitHub Actions Runner Application**
    - Can use a **runner** provided by **GitHub** or **self hosted**
    - **Listens** for **jobs**
        - Runs one job at a time
        - Reports progress, logs, and results back

Reference:

Introduction to GitHub Actions

GitHub Actions help you automate tasks within your software development life cycle. GitHub Actions are event-driven, meaning that you can run a series of commands after a

https://docs.github.com/en/actions/learn-github-actions/introduction-to-github-actions



Workflows

Example Workflow

```
name: automated-code-check
on: [push]
jobs:
  lint-codebase:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm install
      - run: npm run lint
      # sudo code
      # On success, do nothing
      # On error, job fail
  prettify-codebase:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm install
      - run: npm run prettify
      # sudo code
      # On success, push updated code
      # On fail, do nothing
```

Workflow Breakdown

```
name: automated-code-check
```

Optional, name that will appear in the **Actions tab** of the GitHub repo.

```
on: [push]
```

Specified **event** that will trigger the workflow file.

Reference:

Workflow syntax for GitHub Actions

Workflow files use YAML syntax, and must have either a .yml or .yaml file extension. If you're new to YAML and want to learn more, see " Learn YAML in five minutes." You must store workflow files in the .github/workflows directory of your repository. The name of your workflow.

https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions#onpushpull_requestpaths

```
jobs:
  ...
```

Groups **all** jobs ran under **automated-code-check**.

```
lint-codebase:
  ...
prettify-codebase:
  ...
```

Defines **steps** under specified job name(s).

```
runs-on: ubuntu-latest
  ...
```

Configures a **job** to run on a specified **environment** on a **runner.**

Reference:

Workflow syntax for GitHub Actions

Workflow files use YAML syntax, and must have either a .yml or .yaml file extension. If you're new to YAML and want to learn more, see " Learn YAML in five minutes." You must store workflow files in the .github/workflows directory of your repository. The name of your workflow.

https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions#jobsjob_idruns-on

```
...
  - uses: actions/checkout@v2
```

▼ Keyword **uses** tells job to **retrieve**, a **community** made **action.**

This *example* provided **retrieves** a **community action** to help in running your actions **against** your **code**.

```
# Single line execution
runs: npm install

# Multi line execution
runs: |
  npm install
  npm lint
```

runs Executes **single line** or **multiple line** commands within each **runner**.

Environment Variables

The wiki ***strongly* recommends** the use of environment variables to access the filesystem instead of hardcoded file paths. Set environment variables are available to all **runner environments**.

A list of all **default** environment can be referenced here…

Reference:

Environment variables
GitHub sets default environment variables for each GitHub Actions workflow run. You can also set custom environment variables in your workflow file. GitHub Actions is available with GitHub Free, GitHub Pro, GitHub Free for organizations, GitHub Team, GitHub Enterprise Cloud, GitHub Enterprise Server,
https://docs.github.com/en/actions/reference/environment-variables#default-environment-variables

Setting Custom Environment Variables

```
run: "host: $ENV_HOST,port: $ENV_PORT" > credentials.txt
env:
  ENV_HOST: hostname
  ENV_PORT: 3306
```

Environment Variables are **available** at **every step** in a workflow and can be *secretly* & *securely* used to import values from GitHub.

- If *ENV* variable is **referenced** in runner (inside instance), use `$ENV_VAR`
- Otherwise *ENV* must be referenced (before job is sent to runner) from context using `env.ENV_VARIABLE`

Example *ENV* variable assignment using the **context** vs **runner operating system**

```
jobs:
  start_of_week_job:
    runs-on: ubuntu-latest
    env:
      FIRST_DAY_OF_WEEK: Mon
    steps:
      - name: "It's a brand new week!"
        if: env.FIRST_DAY_OF_WEEK == 'Mon'
        run: echo "Happy $FIRST_DAY_OF_WEEK, it's a brand new week!"
        env:
          FIRST_DAY_OF_WEEK: env.FIRST_DAY_OF_WEEK
```

Reference:

Environment variables

GitHub sets default environment variables for each GitHub Actions workflow run. You can also set custom environment variables in your workflow file. GitHub Actions is available with GitHub Free, GitHub Pro, GitHub Free for organizations, GitHub Team, GitHub Enterprise Cloud, GitHub Enterprise Server,

https://docs.github.com/en/actions/reference/environment-variables

Conditional Jobs

```
...
  # If PROD set to true, run steps
  if: env.ENV_PROD
  run: "host: $ENV_HOST,port: $ENV_PORT" > credentials.txt
  env:
    ENV_HOST: hostname
    ENV_PORT: 3306
    ENV_PROD: true
```

Conditionals are used when you want your **job** to only run if a **specific** *criteria/condition* is met.

Change/Run on specific Working Directory

```
...
  - name: Clean temp directory
    run: rm -rf *
    working-directory: ./temp
```

`working-directory` can **execute** command(s) in a specific directory.

Matrix's

Matrix's runs **multiple** jobs with a given set of **steps**

```
runs-on: ${{ matrix.os }}
strategy:
  matrix:
    os: [ubuntu-16.04, ubuntu-18.04]
steps:
  - name: Check Ubuntu version
    run: lsb_release -a
```

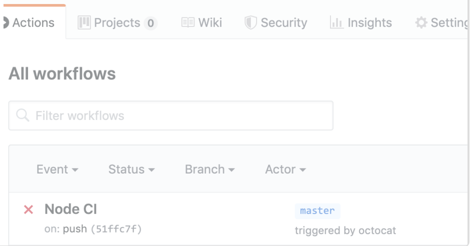
This example given provides two jobs that will be run using the provided **matrix** to execute (*check version command*) on both **Ubuntu 16.04** & **Ubuntu 18.04** *operating system(s)*.

Reference:

Managing complex workflows

This guide shows you how to use the advanced features of GitHub Actions, with secret management, dependent jobs, caching, build matrices, environments, and labels. This article

https://docs.github.com/en/actions/learn-github-actions/managing-complex-workflows#using-a-build-matrix



Running Scripts

```
...
  - name: Run build script
    run: ./.github/scripts/build.sh
    shell: bash
```

Scripts can be **stored** inside your **repository** in which you can provide the *path* & *shell* type to run the script as an action.

Workflow Artifacts

Workflow Artifacts allow you to share data between jobs & allow storage of data after a workflow has been completed.

Example Artifacts:

- Logs
- Test results
- Compressed files

Artifacts properties

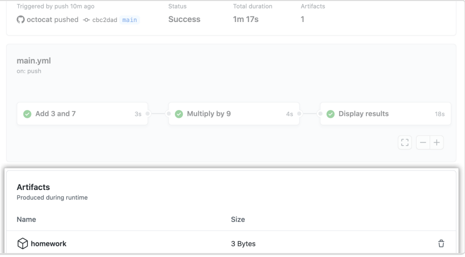
- Free on public or self-hosted runners
    - Private repos limited amount of free minutes/storage
- Default **90 day** retention period
- Once an artifact is deleted it cannot be restored

Reference:

Storing workflow data as artifacts

Artifacts allow you to persist data after a job has completed, and share that data with another job in the same workflow. An artifact is a file or collection of files produced during a workflow

https://docs.github.com/en/actions/guides/storing-workflow-data-as-artifacts#uploading-build-and-test-artifacts



Triggered by push 10m ago
octocat pushed cbc2dad main
Status: Success
Total duration: 1m 17s
Artifacts: 1

main.yml
on: push

Add 3 and 7 3s
Multiply by 9 4s
Display results 18s

Artifacts
Produced during runtime

Name	Size
homework	3 Bytes

Uploading Artifacts

```
...
  steps:
    - shell: bash
    run: echo "Test" > test.log
    - name: Upload test log file as artifact
    uses: actions/upload-artifact@v2
    with:
      name: test-name-of-artifact
      path: test.log
```

In this example we have a `test.log` file which we want to **upload** as an **artifact** called `test-name-of-artifact`.

Modifying Artifact Retention Period

```
...
  steps:
    - shell: bash
    run: echo "Test" > test.log
    - name: Upload test log file as artifact
    uses: actions/upload-artifact@v2
    with:
     name: test-name-of-artifact
     path: test.log
     retention-days: 3
```

Using the same example provided before, we can **customize** the retention period with `retention-days` with the following.

```
...
 retention-days: 3
```

Retention is specified by days in which the value cannot exceed the **retention limit** set by the repo, org, or enterprise.

Downloading Artifacts

```
...
 steps:
  - name: Download test.log artifact
  - uses: actions/download-artifact@v2
  with:
   name: test-name-of-artifact
```

In this we example we download the `test-name-of-artifact` we uploaded in the previous section. You can only **download** artifacts that were uploaded during the **same** workflow run.

```
...
 - uses: actions/download-artifact@v2
```

With this we call a **community** made action to **download** any artifacts needed for this workflow run.

```
...
  with:
    name: test-name-of-artifact
```

▼ Then we will specify a specific artifact we want to download and use in this workflow.

If no `name` is specified, all artifacts will be downloaded in a workflow run.

Basic Attributes/Properties & Characteristics

Attributes/Properties & Associated Characteristics

Aa Name	☑ Min Req.	≡ Properties
name	☐	appears on actions tab as named
on	☑	event driven or scheduled
jobs	☑	groups all jobs under workflow file
runs-on	☑	configures enviornment for a **runner**
- env	☐	used to specify customized **env variables**
steps	☑	groups all steps for each job
- uses	☐	retrieves a community action
- run	☑	execution property
shell	☐	executes a run process in the specified shell environment
with	☐	**dependent** attribute used to associate properties

Name	Min Req.	Properties
path	□	runs specified step within a target directory. can be used along side **!** to negate looking through specified workspace
strategy	□	groups build options
matrix	□	provides build options to run a job in specified group of conditions
if	□	adds conditional criteria before running a step