

# Github Actions

2020.08.03

박원영

# 발표 내용

- Actions
- Docker container action 생성
- JavaScript action 생성
- Metadata
- Exit code
- GitHub Marketplace에 actions 게시
- self-hosted runners

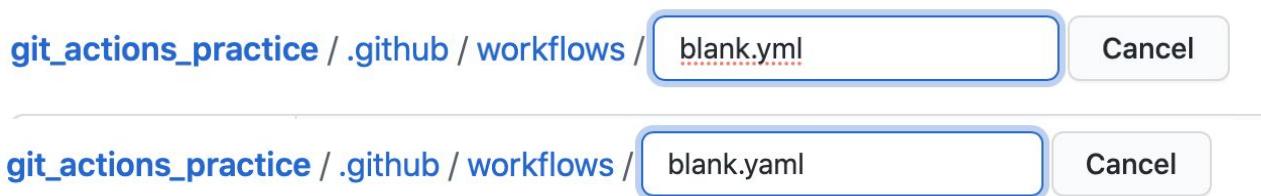
Actions 란?

# Actions

- 개별 작업 => job 생성 & workflow customize
- repository와 상호작용하는 사용자 지정 코드를 이용해서 action 생성 가능
- workflow에서 사용할 고유 actions 작성 / github community에서 구축한 작업을 공유 O
- actions는 docker에서 직접 실행될 수 있음 - actions의 입력값, 출력값, 환경변수 정의 O

# Types of Actions

Docker container/JavaScript actions 구축  
=> metadata 파일(입력값, 출력값, entrypoint)



➤ 이름 지정

# Types of Actions

Docker container actions	JavaScript actions
Linux 운영 체제만, Docker 설치 필요	모든 러너 시스템(Ubuntu, Windows, macOS)와 호환, 러너 시스템에서 직접 실행
느림 (컨테이너 빌드, 검색 시간)	빠름
환경 패키지 by Github actions code => 소비자가 tool/dependency 걱정 X	환경과 action code를 분리 => 코드 단순화

# action 장소 선택

다른 사람들이 사용할 action을 개발 => 자체 repository에 actions 유지

비공개 actions 작성할 경우 저장소 모든 위치 저장 가능

action, workflow, application code를 단일 repository 결합 => .github 폴더에 저장

ex) .github/actions/action-a **and** .github/actions/action-b

# Release management

actions에 업데이트를 배포하는 방법을 제어

(기존 workflow와 호환성 유지하기 위해)

master 브랜치를 참조X => 주 버전을 지정한 후 문제 발생시 구체적 버전을 안내

태그 사용 - 주버전과 부버전 구분

# Release management - target tag

```
steps:  
# Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it  
- uses: actions/checkout@v2
```

➤ 주요 release tag 참조

```
steps:  
# Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it  
- uses: actions/checkout@v2.0.1
```

➤ 특정 부분 release tag 참조

```
steps:  
# Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it  
- uses: actions/checkout@v2-beta
```

➤ release management에 branch 참조

```
steps:  
- uses: actions/javascript-action@172239021f7ba04fe7327647b213799853a9eb89
```

➤ commit의 SHA 사용

# Docker container action 생성

# Github 저장소 생성

The screenshot shows a GitHub repository page. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, the repository name 'Wonyoungpark / hello-world-docker-action' is displayed. The main content area includes tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Under the Code tab, it shows 'master' branch, 1 branch, 0 tags, and a commit from 'Wonyoungpark' labeled 'Initial commit' made 'now'. A file named 'README.md' is listed with the content 'Initial commit' and a timestamp 'now'. At the bottom, there's a section for 'hello-world-docker-action' with a link to 'View raw'.

```
Last login: Sun Aug  2 12:25:41 on console
(base) holly@hollyui-MacBookPro ~ % cd workspace
(base) holly@hollyui-MacBookPro workspace % git clone https://github.com/Wonyoungpark/hello-world-docker-action.git
'hello-world-docker-action'에 복제합니다...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
오브젝트 뮤음 푸는 중: 100% (3/3), 617 bytes | 205.00 KiB/s, 완료.
(base) holly@hollyui-MacBookPro workspace % cd hello-world-docker-action
(base) holly@hollyui-MacBookPro hello-world-docker-action %
```

git repository 생성 후 clone

# Dockerfile 생성 (docker의 시작동작 설명)

The image displays three terminal windows side-by-side, each showing code in a light blue background with white text.

- Top Left Terminal:** Shows the Dockerfile content. It includes the FROM command pointing to alpine:3.10, a COPY command for entrypoint.sh, and an ENTRYPOINT command for the same file.
- Top Right Terminal:** Shows the action.yml configuration file. It defines a service named 'Hello World' with a description of 'Greet someone and record the time'. It has an input 'who-to-greet' and an output 'time'. The runs section specifies using docker with the Dockerfile image and an argument derived from the input.
- Bottom Terminal:** Shows the entrypoint.sh script content, which prints "Hello" followed by the value of \$who.

```
FROM alpine:3.10
COPY entrypoint.sh /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]

name: 'Hello World'
description: 'Greet someone and record the time'
inputs:
  who-to-greet: # id of input
    description: 'Who to greet'
    required: true
    default: 'World'
outputs:
  time: # id of output
    description: 'The time we greeted you'
runs:
  using: 'docker'
  image: 'Dockerfile'
  args:
    - ${inputs.who-to-greet}

who=${1:-world}
echo "Hello" $who

(base) holly@hollyui-MacBookPro hello-world-docker-action % chmod +x entrypoint.sh
```

The screenshot shows the GitHub Actions editor for the file `hello-world-docker-action/.github/workflows/docker-publish.yml`. The code defines a workflow that runs on push events, with a single job named `hello_world_job` that uses the `actions/hello-world-docker-action@master` action to say hello and then echo the current time.

```
1 on: push
2
3 jobs:
4   hello_world_job:
5     runs-on: ubuntu-latest
6     name: A job to say hello
7     steps:
8       - name: Hello world action step
9         id: hello
10        uses: actions/hello-world-docker-action@master
11        with:
12          to-greet: 'Mona the Octocat'
13        # Use the output from the 'hello' step
14        - name: Get the output time
15        run: echo "The time was ${{ steps.hello.outputs.time }}"
16
```

At the bottom of the editor, there is a note: `Use [Control] + Space to trigger autocomplete in most situations.`

The screenshot shows the GitHub Actions results page for the workflow `docker-publish.yml`. The workflow has one result, which is a successful run triggered by a push to the `master` branch. The run was completed 2 minutes ago and has 18s of logs.

**All workflows**

Workflow	Status	Event	Actor
<code>Create docker-publish.yml</code>	Success	Commit 26993b07 pushed by	<code>Wonyoungpark</code>

Seminar GitHub Actions Documentation - GitHub Docs https://s3.us-west-2.amazonaws.com/secu... hello-world-docker-action/action.yml at... Create docker-publish.yml - Wonyoungpar... +

Search or jump to... Pull requests Issues Marketplace Explore

Wonyoungpark / hello-world-docker-action

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Create docker-publish.yml

master · 26993b7 · Re-run jobs

.github/workflows/docker-publish.yml / A job to say hello

succeeded 5 minutes ago in 4s

Search logs

on: push

A job to say hello

Set up job

Build actions/hello-world-docker-action@master

Hello world action step

Run actions/hello-world-docker-action@master

```
1 /usr/bin/docker run --name c201d771590494fa4b4fadfdb60a64d3f01_8203ae --label 87c201 --workdir /github/workspace --rm -e INPUT_WHO-TO-GREET -e HOME -e GITHUB_JOB -e GITHUB_REF -e GITHUB_SHA -e GITHUB_REPOSITORY -e GITHUB_REPOSITORY_OWNER -e GITHUB_RUN_ID -e GITHUB_RUN_NUMBER -e GITHUB_ACTOR -e GITHUB_WORKFLOW -e GITHUB_HEAD_REF -e GITHUB_BASE_REF -e GITHUB_EVENT_NAME -e GITHUB_SERVER_URL -e GITHUB_API_URL -e GITHUB_GRAPHQL_URL -e GITHUB_WORKSPACE -e GITHUB_ACTION -e GITHUB_EVENT_PATH -e RUNNER_OS -e RUNNER_TOOL_CACHE -e RUNNER_TEMP -e RUNNER_WORKSPACE -e ACTIONS_RUNTIME_URL -e ACTIONS_RUNTIME_TOKEN -e ACTIONS_CACHE_URL -e GITHUB_ACTIONS=true -e CI=true -v "/var/run/docker.sock":"/var/run/docker.sock" -v "/home/runner/work/_temp/_github_home":"/github/home" -v "/home/runner/work/_temp/_github_workflow":"/github/workflow" -v "/home/runner/work/hello-world-docker-action/hello-world-docker-action":"/github/workspace" 87c201:d771590494fa4b4fadfdb60a64d3f01 "Mona the Octocat"
```

Hello Mona the Octocat who-to-greet input값 출력

Get the output time

```
1 Run echo "The time was Sun Aug 2 07:53:48 UTC 2020"
4 The time was Sun Aug 2 07:53:48 UTC 2020
```

time-stamp값 출력

Complete job

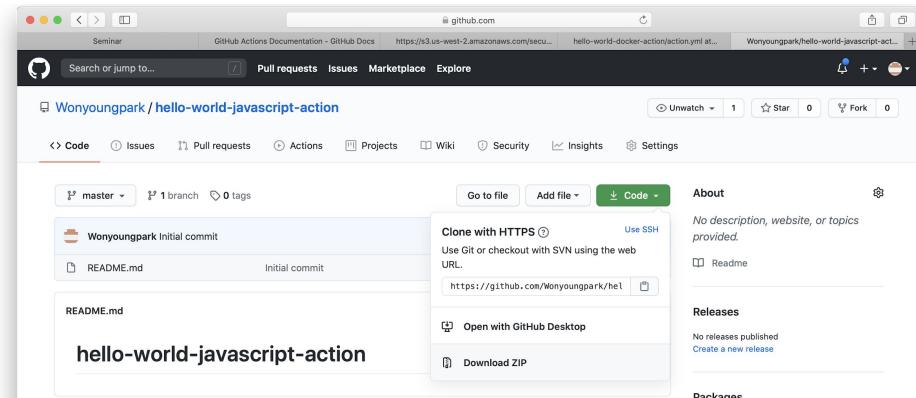
# JavaScript action 생성

# JavaScript action 만들기

Github Actions Toolkit Node.js 모듈 사용 - 개발속도 향상

Node.js 다운로드 : <https://nodejs.org/en/download/current/>

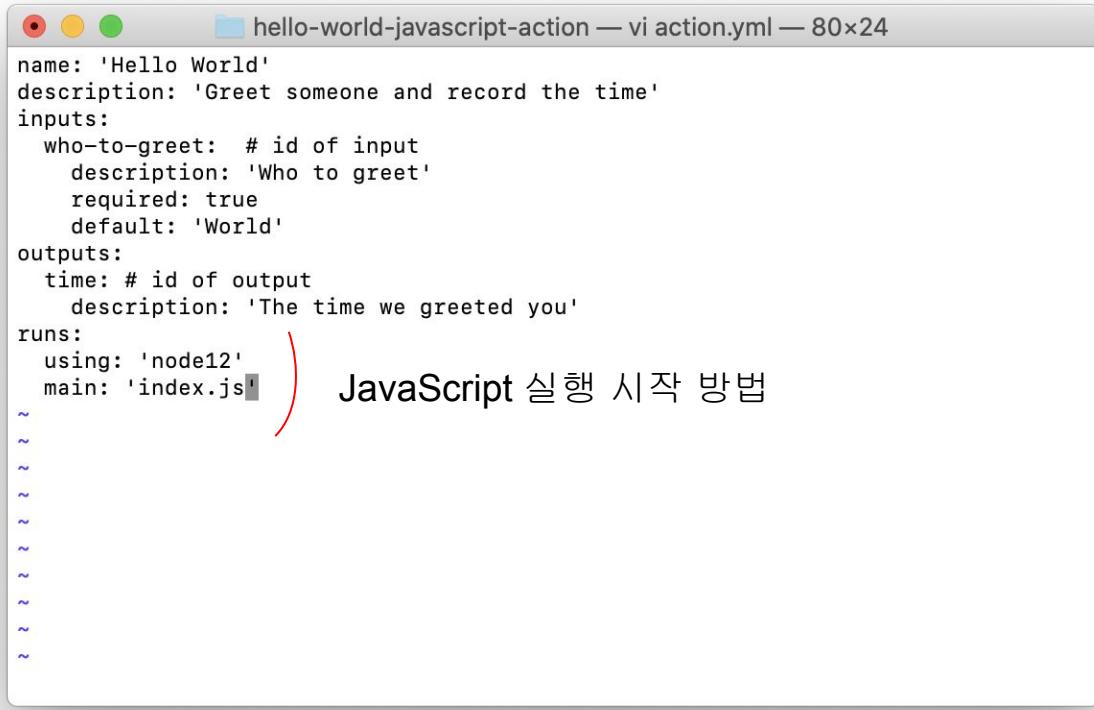
Node.js : Chrome V8 Javascript 엔진으로 빌드된 Javascript 런타임



```
[(base) holly@hollyui-MacBookPro hello-world-javascript-action % npm init -y 디렉토리 초기화]
```

npm : Node.js로 만들어진 모듈을 웹에서 받아서 설치하고 관리해주는 프로그램

# action metadata file 작성



```
hello-world-javascript-action — vi action.yml — 80x24
name: 'Hello World'
description: 'Greet someone and record the time'
inputs:
  who-to-greet: # id of input
    description: 'Who to greet'
    required: true
    default: 'World'
outputs:
  time: # id of output
    description: 'The time we greeted you'
runs:
  using: 'node12'
  main: 'index.js'
```

JavaScript 실행 시작 방법

# actions toolkit package 추가

JavaScript actions를 빠르게 빌드할 수 있는 Node.js package

```
(base) holly@hollyui-MacBookPro hello-world-javascript-action % npm install @actions/core
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN hello-world-javascript-action@1.0.0 No description

+ @actions/core@1.2.4
added 1 package and audited 1 package in 3.064s
found 0 vulnerabilities

(base) holly@hollyui-MacBookPro hello-world-javascript-action % npm install @actions/github
npm WARN hello-world-javascript-action@1.0.0 No description

+ @actions/github@4.0.0
added 20 packages from 53 contributors and audited 21 packages in 18.206s
found 0 vulnerabilities
```

workflow의 명령,  
입출력 변수, 종료  
상태, 디버그 메시지에  
대한 인터페이스 제공

Github actions  
context에 접근 가능

# action code 작성

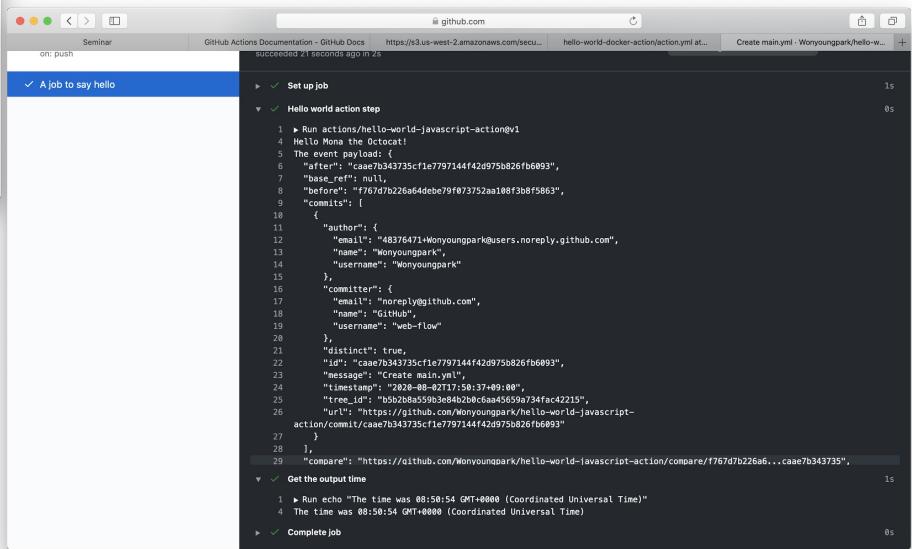
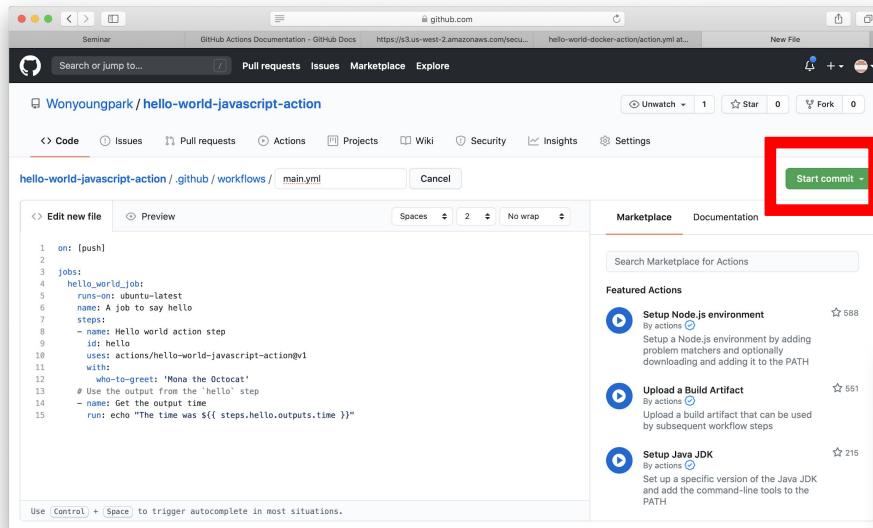


A screenshot of a terminal window titled "hello-world-javascript-action — vi index.js — 80x24". The window displays a block of JavaScript code. The code imports 'actions/core' and 'actions/github' modules, sets up a try-catch block to handle input from the workflow metadata file, logs a greeting message, gets the current time, and logs the JSON webhook payload. It also handles errors by setting the core output to failed. The cursor is positioned inside the catch block.

```
const core = require('@actions/core');
const github = require('@actions/github');

try {
  // `who-to-greet` input defined in action metadata file
  const nameToGreet = core.getInput('who-to-greet');
  console.log(`Hello ${nameToGreet}!`);
  const time = (new Date()).toTimeString();
  core.setOutput("time", time);
  // Get the JSON webhook payload for the event that triggered the workflow
  const payload = JSON.stringify(github.context.payload, undefined, 2)
  console.log(`The event payload: ${payload}`);
} catch (error) {
  core.setFailed(error.message);
}
```

```
git add action.yml index.js node_modules/* package.json package-lock.json
git commit -m "My first action is ready"
git tag -a -m "My first action release" v1
git push --follow-tags
```



# Metadata

# YAML 문법

metadata 파일 = input, output, entrypoint => action.yml / action.yaml

hello-world-javascript-action/.github/workflows/ **action.yml** Cancel

Edit new file Preview Spaces 2

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the action will run. Triggers the workflow on push or pull request
6 # events but only for the master branch
7 on:
8   push:
9     branches: [ master ]
10  pull_request:
11    branches: [ master ]
12
13 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
14 jobs:
15  # This workflow contains a single job called "build"
16  build:
17    # The type of runner that the job will run on
18    runs-on: ubuntu-latest
19
20  # Steps represent a sequence of tasks that will be executed as part of the job
21  steps:
22    # Checks-out your repository under $GITHUB_WORKSPACE so your job can access it
Use [Control] + [Space] to trigger autocomplete in most situations.
```

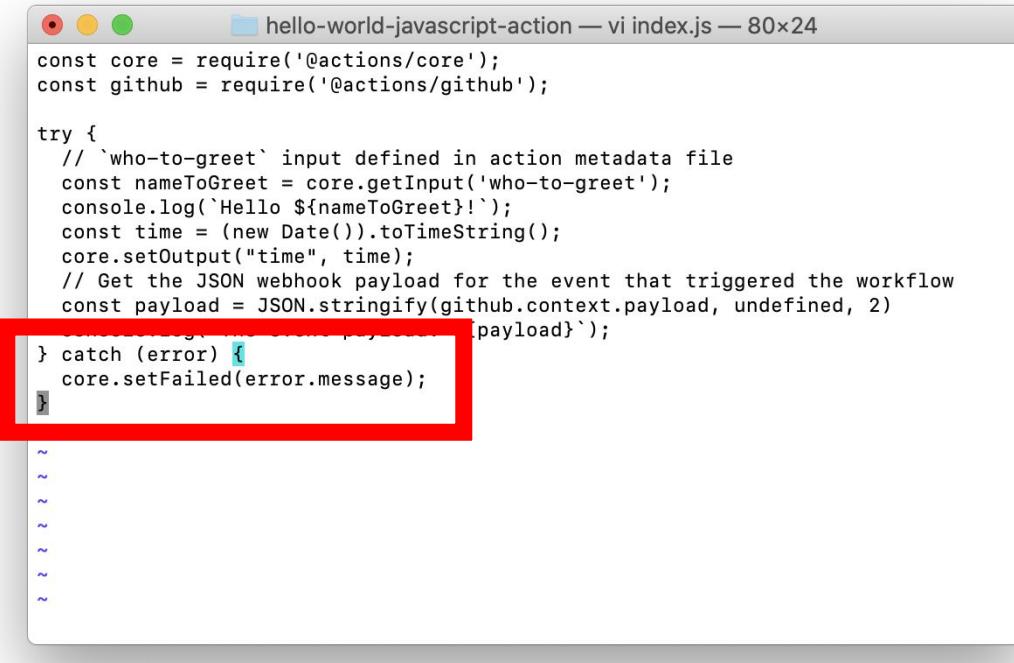
```
name: 'Hello World'
description: 'Greet someone and record the time'
branding:
  color: 'yellow'
  icon: 'code'
inputs:
  who-to-greet: input id
    description: 'Who to greet'
    required: true
    default: 'World'
outputs:
  time: output id
    description: 'The time we greeted you'
runs:
  using: 'node12' 실행
  main: 'index.js' action 코드
```

action metadata 의미와 종류 : <https://docs.github.com/en/actions/creating-actions/metadata-syntax-for-github-actions>

# Exit code

actions의 실행 상태

# Exit Code - JavaScript action



```
hello-world-javascript-action — vi index.js — 80x24
const core = require('@actions/core');
const github = require('@actions/github');

try {
    // `who-to-greet` input defined in action metadata file
    const nameToGreet = core.getInput('who-to-greet');
    console.log(`Hello ${nameToGreet}!`);
    const time = (new Date()).toTimeString();
    core.setOutput("time", time);
    // Get the JSON webhook payload for the event that triggered the workflow
    const payload = JSON.stringify(github.context.payload, undefined, 2)
    core.setSecret("payload", payload);
} catch (error) {
    core.setFailed(error.message);
}
```

~  
~  
~  
~  
~  
~  
~

# Exit Code - Docker container action



A screenshot of a terminal window titled "hello-world-docker-action — vi entrypoint.sh — 80x24". The window shows a portion of a shell script:

```
if index=4 ; then
    echo "Game over!"
    exit 1
fi
```

The cursor is positioned at the start of the "fi" line. Below the code, there are approximately 20 blank lines, each starting with a purple tilde (~).

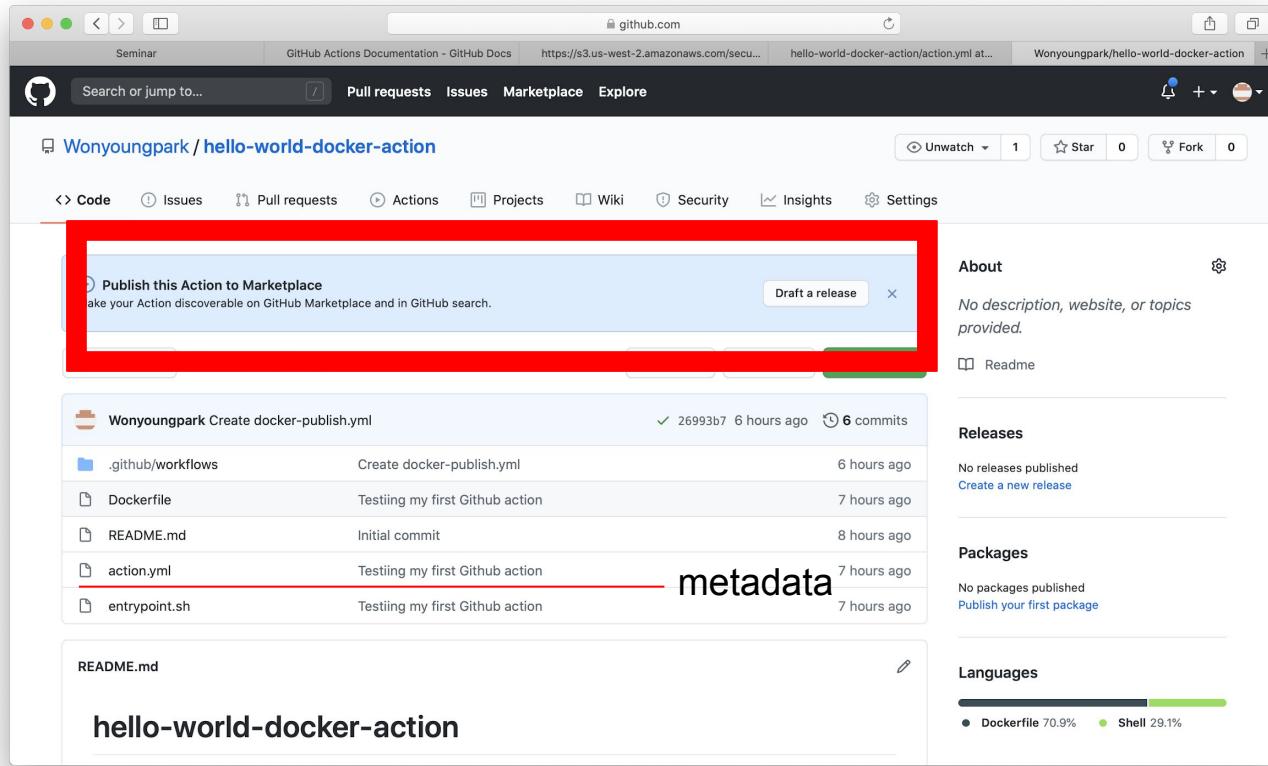
# GitHub Marketplace에 actions 게시

# actions 게시

## Github Marketplace 게시 조건

- Actions는 public repository에 있어야 함.
- 각 repository당 1 action
- Action의 metadata은 repository의 루트 디렉토리에 존재
- Action의 metadata명은 고유

# actions 게시



The screenshot shows a GitHub repository page for `Wonyoungpark/hello-world-docker-action`. A red box highlights a banner at the top left of the repository interface, which reads "Publish this Action to Marketplace" and "make your Action discoverable on GitHub Marketplace and in GitHub search." Below this, there is a list of files and commits. A red arrow points from the word "metadata" in the text below to the `action.yml` file entry in the commit list. The repository has 6 commits, with the most recent being a merge commit by `26993b7` 6 hours ago. Other commits include `Create docker-publish.yml`, `Dockerfile`, `README.md`, and `entrypoint.sh`. The `action.yml` file is described as "Testing my first Github action". The repository also includes a `README.md` file with the text "hello-world-docker-action". On the right side of the page, there are sections for "About", "Releases", "Packages", and "Languages".

Publish this Action to Marketplace  
make your Action discoverable on GitHub Marketplace and in GitHub search.

`Wonyoungpark Create docker-publish.yml` ✓ 26993b7 6 hours ago 6 commits

`.github/workflows` Create docker-publish.yml 6 hours ago

`Dockerfile` Testing my first Github action 7 hours ago

`README.md` Initial commit 8 hours ago

`action.yml` Testing my first Github action 7 hours ago

`entrypoint.sh` Testing my first Github action 7 hours ago

`README.md`

**hello-world-docker-action**

**About**  
No description, website, or topics provided.

**Releases**  
No releases published  
[Create a new release](#)

**Packages**  
No packages published  
[Publish your first package](#)

**Languages**  
● Dockerfile 70.9% ● Shell 29.1%

# actions 게시

The screenshot shows the GitHub Actions release creation interface for the repository `Wonyoungpark/hello-world-docker-action`. A red box highlights the warning message: "Polish this release to the GitHub Marketplace. You must accept the GitHub Marketplace Developer Agreement before publishing an Action." Below this, there are fields for "Tag version" (set to `v2.0.1`) and "Target: master". The "Write" tab is selected, showing a placeholder "Describe this release". On the right, there's a "Tagging suggestions" section with the heading "Semantic versioning".

The screenshot shows the final step of the GitHub Actions release creation process. The release title is `2.0.1`, the target is `master`, and the tag is `Bug fixes to Hello Mona's World`. The "Write" tab is selected, with the text "버전 태그" (Version Tag), "relase" (Release), and "제목" (Title) overlaid. The "Tags" tab is also visible. On the right, there are sections for "Tagging suggestions" and "Semantic versioning". A red box highlights the "Publish release" button at the bottom.

# actions 게시

The screenshot shows the GitHub Actions release action configuration page for a repository named 'hello-world-docker-action'. The 'Releases' tab is selected. A checkbox labeled 'Publish this Action to the GitHub Marketplace' is checked. Below it, there's a note: 'Your action.yml needs changes before it can be published.' Under the 'Name' field, there's an error message: 'Hello World - Name must be unique. Cannot match an existing action, user or organization name.' The 'Description' field contains the text 'Greet someone and record the time'. The 'Icon' and 'Color' fields have placeholder text: 'See list of available icons.' and 'See list of available colors.' respectively. At the bottom, there's a 'README' section.

The screenshot shows the GitHub Actions action configuration page for the 'hello-world-docker-action' repository. The 'Code' tab is selected. A modal window titled 'action.yml' is open, showing the YAML code for the action. The code defines a single step that runs a Docker container to greet someone and record the time.

```
1 name: 'Hello Mona\'s World'
2 description: 'Greet someone and record the time'
3 inputs:
4   who-to-greet: # id of input
5     description: 'Who to greet'
6     required: true
7     default: 'World'
8 outputs:
9   time # id of output
10  description: 'The time we greeted you'
11 run:
12   using: 'docker'
13   image: 'Dockerfile'
14   args:
15     - ${{ inputs.who-to-greet }}
```

# self-hosted runners

# Github의 runner 제공

GitHub-hosted Runner : workflow를 빠르고 간단하게 실행하는 방법

Self-Hosted Runner : workflow를 사용자 정의 환경에서 실행하는 방법

# self-hosted runner repository에 추가

A screenshot of a GitHub repository page for 'Wonyoungpark / hello-world-docker-action'. The page shows various repository details like branches, commits, and actions. The 'Settings' tab is highlighted with a red box.

A screenshot of the 'Actions' tab in the GitHub repository settings. It shows 'Actions permissions' and 'Self-hosted runners' sections. The 'Actions' section is highlighted with a red box. In the 'Self-hosted runners' section, there is a message: 'There are no runners configured for this repository.' and a 'Add Runner' button, which is also highlighted with a red box.

# self-hosted runner repository에 추가

The screenshot shows the GitHub Actions Documentation page for adding a self-hosted runner. The left sidebar has 'Actions' selected. The main content area is titled 'Actions / Add self-hosted runner'. It contains instructions for downloading and configuring the GitHub Actions Runner, including a terminal session showing the download and extraction of the runner package. Below this is a 'Configure' section with a terminal session for setting up the runner and starting the configuration experience. At the bottom, there's a note about using the self-hosted runner in a workflow file.

```
// Create a folder  
$ mkdir actions-runner && cd actions-runner  
// Download the latest runner package  
$ curl -O https://github.com/actions/runner/releases/download/v2.267.1/actions-runner-osx-x64-2.267.1.tar.gz  
// Extract the installer  
$ tar xzf ./actions-runner-osx-x64-2.267.1.tar.gz  
  
// Create the runner and start the configuration experience  
$ ./config.sh --url https://github.com/Wonyoungpark/hello-world-docker-action --token ALRCVF65VXL4UTT7MUDQNC7E3TNI  
// Last step, run it!  
$ ./run.sh  
  
Using your self-hosted runner  
  
# Use this yaml in your workflow file for each job  
runs-on: self-hosted
```

The screenshot shows a terminal window titled 'actions-runner — Runner.Listener ▾ run.sh — 80x24'. It displays the runner registration process. The runner is successfully connected to GitHub and added to the organization. The configuration file is saved, and the runner starts listening for jobs. A red box highlights the final message: '2020-08-02 15:19:04Z: Listening for Jobs'.

```
✓ Connected to GitHub  
  
# Runner Registration  
  
[Enter the name of runner: [press Enter for hollyui-MacBookPro]]  
  
This runner will have the following labels: 'self-hosted', 'macOS', 'X64'  
[Enter any additional labels (ex. label-1,label-2): [press Enter to skip]]  
  
✓ Runner successfully added  
✓ Runner connection is good  
  
# Runner settings  
  
[Enter name of work folder: [press Enter for _work]]  
  
✓ Settings Saved.  
  
[base] hollyui-MacBookPro:~ actions-runner% ./run.sh  
  
✓ Connected to GitHub  
  
2020-08-02 15:19:04Z: Listening for Jobs
```

## Self-hosted runners

The screenshot shows the GitHub Actions 'Self-hosted runners' management interface. It lists a single runner named 'hollyui-MacBookPro' which is currently 'Idle'. There are buttons for 'Runners' and 'Add Runner'.

Runner	Status
hollyui-MacBookPro	Idle

# self-hosted runner를 서비스로 구성하기

머신이 시작될 때 runner를 자동으로 시작

```
actions-runner — zsh — 80x24
[(base) holly@hollyui-MacBookPro actions-runner % ./svc.sh install 서비스 설치
Creating launch runner in /Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist
Creating /Users/holly/Library/Logs/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro
Creating /Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist
Creating runsvcs.sh
Creating .service
svc install complete
[(base) holly@hollyui-MacBookPro actions-runner % ./svc.sh start 서비스 시작
starting actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro
status actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro:
/Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist

Started:
4400 0 actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro

[(base) holly@hollyui-MacBookPro actions-runner % ./svc.sh status 서비스 상태
status actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro]
```

```
actions-runner — zsh — 80x24
[(base) holly@hollyui-MacBookPro actions-runner % ./svc.sh stop 서비스 중지
stopping actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro
o
/Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist: Operation now in progress
status actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro:
/Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist

Stopped

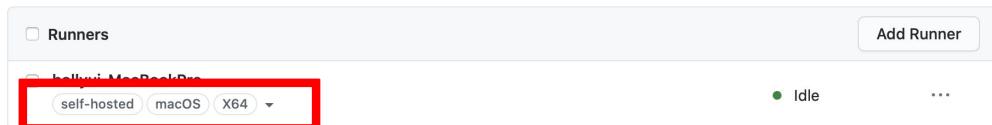
[(base) holly@hollyui-MacBookPro actions-runner % ./svc.sh uninstall 서비스 제거
uninstalling actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro
o
stopping actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro
o
/Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist: Could not find specified service
status actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro:
/Users/holly/Library/LaunchAgents/actions.runner.Wonyoungpark-hello-world-docker-action.hollyui-MacBookPro.plist
```

# self-hosted runner

proxy 서버 => GitHub와 통신하도록 구성

label => self-hosted runner 특성별 구분

## Self-hosted runners



# self-hosted runner 제거

Settings > Actions > Self-hosted runners

## Self-hosted runners

The screenshot shows the GitHub Actions settings interface for managing self-hosted runners. At the top left is a checkbox labeled "Runners". To its right is a large "Add Runner" button. Below this section, a runner is listed with the name "hollyui-MacBookPro". This runner has three status indicators: "self-hosted", "macOS", and "X64". To the right of the runner's name is a status indicator showing a black dot next to the word "Offline". Further to the right is a vertical ellipsis ("...") and a red "Remove" button.

Runners

hollyui-MacBookPro

self-hosted macOS X64 ▾

● Offline

...

Remove

# 참고 자료

Github Actions 공식 문서 : <https://docs.github.com/en/actions/>

Github actions & package repository 사용기  
<https://coffeewhale.com/cicd/github/2019/10/14/github-actions/>

Github Actions 빠르게 시작하기 :  
[https://jonnung.dev/devops/2020/01/31/github\\_action\\_getting\\_started/](https://jonnung.dev/devops/2020/01/31/github_action_getting_started/)

감사합니다.