

## Troubleshooting Tips for GitHub Actions Workflows

A compilation of GitHub Actions workflows errors and solutions from my own work experience



Photo by author

While working with GitHub Actions workflows building and deploying microservices into AWS, I have run into some errors/exceptions and dived deep into finding their solutions. I am listing a collection of such error scenarios and their solutions, hoping to be of some help to those who are working with GitHub Actions.

The detailed CI/CD workflows referenced in the troubleshooting details below are documented in my story “[A Deep Dive into GitHub Actions’ Reusable Workflows](#)”.

## Error #1: CI workflow failed at “Scan ECR Image with Trivy vulnerability scanner” step

If you use [Trivy](#) to scan your docker image and if your CI workflow failed at the ECR image scan step with the following error (see screenshot below), which is caused by a vulnerability issue in the docker base image. In this case, base image `amazoncorretto:17.0.2-alpine3.15` was used for Spring Boot microservices (no new image with the fix has been published as of this writing), you will need to provide the workaround to update the packages in your base image in your `Dockerfile`.

Specifically, add line `RUN apk update && apk upgrade -U -a` under the `FROM` line in your `Dockerfile`. With this one-line fix, your CI should be able to pass that ECR image scan step. By the way, I highly recommend Trivy for image scan, it not only scans your app's dependencies to check vulnerabilities, it also scans your base image. Add Trivy scan to your CI workflow if you haven't already.

```
31 2022-12-21T03:18:44.374Z INFO Detected OS: alpine
32 2022-12-21T03:18:44.374Z INFO Detecting Alpine vulnerabilities...
33 2022-12-21T03:18:44.376Z INFO Number of language-specific files: 1
34 2022-12-21T03:18:44.376Z INFO Detecting jar vulnerabilities...
35 2022-12-21T03:18:44.383Z INFO Table result includes only package filenames. Use '--format json' option to get the full path to the package file.
36
37 ***/*: (alpine 3.15.4)
38 =====
39 Total: 1 (HIGH: 0, CRITICAL: 1)
40
41
42 | Library | Vulnerability | Severity | Installed Version | Fixed Version | Title |
43 |-----|-----|-----|-----|-----|-----|
44 | zlib | CVE-2022-37434 | CRITICAL | 1.2.12-r0 | 1.2.12-r2 | zlib: heap-based buffer over-read and overflow in inflate() |
45 | | | | | | in inflate.c via a... |
46 | | | | | | https://avd.aquasec.com/nvd/cve-2022-37434 |
47 |-----|-----|-----|-----|-----|-----|
```

## Error #2: release workflow failed with “500 Internal Server Error”

During the course of testing the maven release workflow, I run into “500 Internal Server Error” and pointing to GitHub Packages not behaving as expected. I contacted GitHub tech support regarding this error, and here is their response:

*“It looks like this was a transient error, which possibly happened during a time of high load”*. They recommended the following workaround to allow retry when such an error occurs. If by any chance you still run into this error even after 3 retries, the best way to clean up the release process is to manually bump up the main’s pom version to the next development snapshot version, and re-release the new version.

```
env:
  MAVEN_OPTS: -Dhttp.keepAlive=false -Dmaven.wagon.http.pool=false
  -Dmaven.wagon.http.retryHandler.class=standard -
  Dmaven.wagon.http.retryHandler.count=3
```

### **Error #3: “Couldn’t retrieve verification key from your identity provider, please reference AssumeRoleWithWebIdentity documentation for requirements”**

If you run into the above error at step “Configure AWS credentials”, it indicates a misconfiguration for OIDC. Follow the four steps below to properly configure OIDC in AWS.

#### **Step 1: Add the Identity Provider (if it doesn’t exist already)**

1. Open the IAM console.
2. In the navigation pane, choose “Identity providers”, and then choose “Add provider”.
3. For “Configure provider”, choose “OpenID Connect”.

4. For “Provider URL”, type `https://token.actions.githubusercontent.com`
5. For “Audience”, type `sts.amazonaws.com` since we are using the [official action](#).
6. Choose “Add provider”.

## Step 2: Add Role and Trust Policy

Add a new role and assign policies as needed for your new application. The policies in this step could vary depending on the specific application you are planning to deploy. If there is an existing role you would like to use, be sure it’s configured correctly by verifying the following steps.

Once the role is added, ensure its trust policy is defined like the following, `Federated` line should specify the `ARN` for the identity provider we just created above (sample snippet below for reference only, ensure you change the `ARN` for the identity provider according to your account on line 7). The `aud` section (line 12) should always have value `sts.amazonaws.com`, as defined in the Identity Provider section above.

## Step 3: Add Inline Policy for ECR Access

For microservices, it’s a common step to build and push the docker image to ECR. The GitHub action we use, `aws-actions/amazon-ecr-login@v1`, to log into ECR requires the push/pull permissions to ECR private repo. The built-in policies for ECR didn’t work for push/pull of images from/to private repo, unfortunately, so I had to

create an inline policy for ECR access. Add this inline policy to our role in addition to the existing policies we already selected for that role:

#### Step 4: GitHub Actions Workflow Configuration

Create a new GitHub Environment secret, with key `ROLE_TO_ASSUME`, and value as that role's `ARN`.

Before the jobs are defined, we need to specify the permission in our GitHub Actions workflow, need to add in particular, `id-token: write` permission in the caller workflow, which allows the JWT to be requested from GitHub's OIDC provider. We won't be able to request the OIDC JWT token if the `permissions` setting for `id-token` is set to `read` or `none`.

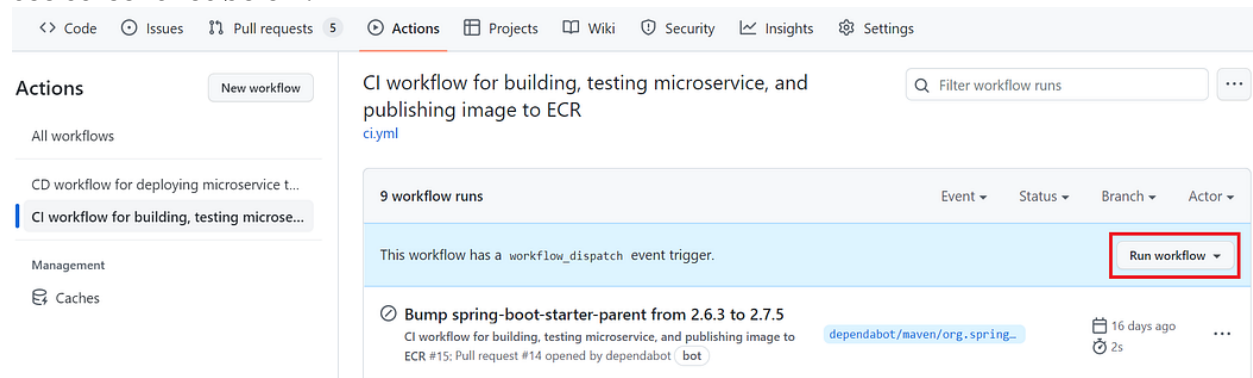
#### Error #4: “NoCredentialProviders: no valid providers in chain”

If you happen to run into this error in your workflow execution, it usually indicates GitHub Actions cannot find the secrets defined. Check to see if you specified `environment` in your workflow. If you are working on a brand new workflow which interacts with AWS resources, be sure to add this line (see below) above the steps defined in your workflow job.

```
environment: ${ { inputs.env || 'dev' } }
```

**Error #5: missing manual trigger even when you have `workflow_dispatch` trigger in your workflow**

If you have `workflow_dispatch` trigger defined in your workflow, but for some reason you don't see the "Run workflow" button for your workflow under Actions tab in your repository, check to see if the branch you are in is the default branch of your repository. You can merge your feature branch into your default branch, revise the trigger to point to your feature branch, once code is merged, you will see the latest code and workflow file changes reflected. You should now see the "Run workflow" button under the Actions tab, see screenshot below.



## Error #6: GitHub PR page displays "This branch was successfully deployed" even when you run a CI workflow

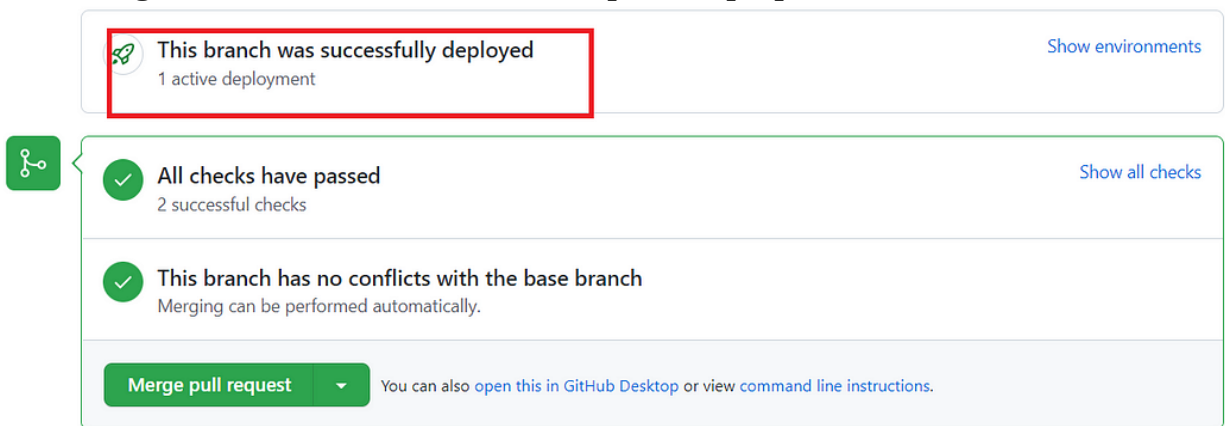
Provided you have the PR trigger in your workflow, when a new PR is raised, it automatically triggers the CI workflow. No actual deployment is done to your AWS environment. You wonder why on the PR page GitHub displays this line "This branch was successfully deployed" (see screenshot below). I contacted GitHub tech support, and here is their response:

*"This is happening because the workflow has a job that is referencing the **dev** environment. When you reference an environment in a workflow job, GitHub implicitly creates a deployment to that environment. Deployment in this case refers to the [deployments concept](#) in GitHub rather than to the external service provider.*

*At the moment, there is no way to change or remove that message from the pull request timeline if the job in the workflow is referencing an environment, sorry.*

*I agree that the message does seem confusing and we've received similar feedback from other users about this. I have added this ticket to the discussion. I am not able to give you a timeline for when changes to it will be made but this will be documented on our [Changelog](#)."*

In our sample CI workflow, we need to reference the secrets defined in `dev` GitHub environment to perform actions such as configure AWS credentials and push image to ECR. With CI workflow referencing the `dev` environment, it is the very reason why GitHub displays that message. Nothing we can do on our end. For now, just be aware of this message, and wait for GitHub to come up with a proper fix.



The screenshot shows a GitHub pull request status bar. At the top, a light blue box contains a green rocket icon, the text "This branch was successfully deployed" with "1 active deployment" below it, and a "Show environments" link. Below this is a green box with a branch icon on the left. It contains two green checkmark icons: "All checks have passed" with "2 successful checks" and "Show all checks" link, and "This branch has no conflicts with the base branch" with "Merging can be performed automatically." At the bottom of the green box is a green "Merge pull request" button with a dropdown arrow, and a link to "open this in GitHub Desktop" or view "command line instructions."

## **Error #7: “Workflow does not exist or does not have a workflow\_dispatch trigger in this tag”**

This error happens when you try to deploy a released tag to any environment through manual trigger by picking the specific tag. The reason for this error is that your workflow file must have changed since your tag was created by maven release. This happens when



you tweak/refine your workflows in the beginning phase of your adoption of GitHub Actions. So if your workflow has changed since your last maven release, the best way to fix it is to just release a new version and deploy the new tag. That works!

### **Error #8: Workflow failed at step “Build, tag, and push image to AWS ECR”**

If you see an error in your GitHub Actions log under the step “Build, tag, and push image to AWS ECR”, complaining about retrying multiple times trying to push image to ECR, eventually returning exit code 1. It’s most likely because your app either doesn’t have an ECR repository configured in the ECR registry, or your IAM Role for your app doesn’t have the proper permission configured in your ECR repository. Double-check both!

### **Error #9: “Input required and not supplied: aws-region”**

This error occurs in “Configure AWS Credentials” step. Even though you have your IAM Role and AWS\_REGION properly added in GitHub secrets, you still run into this error, why? A few reasons:

1). Ensure your OIDC permission is properly set in your GitHub Actions workflow. Be sure to check you have the following permission defined properly before calling the reusable workflow containing "Configure AWS Credentials" or any other AWS actions.

```
permissions:
  id-token: write # need this for OIDC
  contents: read
```

2). Ensure you have “Configure AWS Credentials” step defined before any step that manipulates environment variables. It appears to be a potential bug in AWS action



“Configure AWS Credentials”, whenever a github environment variable is defined and passed into `$GITHUB_ENV`, “Configure AWS Credentials” step somehow pulls that environment variable and passes it to its action instead of the IAM role and AWS region values from secrets. So suggest to always put “Configure AWS Credentials” step before any step that deals with `$GITHUB_ENV`.

### **Error #10: “no such file or directory” when deploying artifacts**

For many of the microservices deploying images to ECS, reusable workflows have nailed down the details for deployment already, but for microservices which deploy artifacts other than docker image, such as deploying jar file to a Lambda function, this error “no such file or directory” may occur when the path to the artifact is not defined correctly, especially in a monorepo scenario, or a nested project structure inside an existing legacy repository. Be sure to specify the full file path to the artifact whenever the artifact path is required.

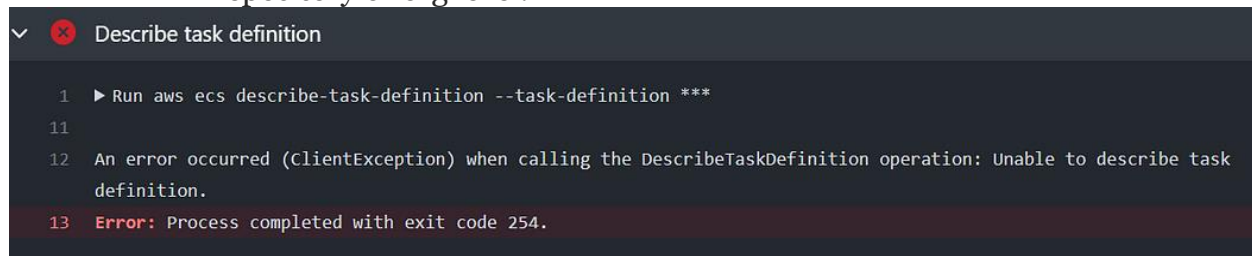
### **Error #11: release workflow fails with artifact already exists error**

Snapshot version can be released multiple times, but the release version can only be released once. If a developer tries to release the same version after that version has already been released (the only possible reason for this could be the developer forgot to merge the latest code from the `release` branch into the `main` after the `release`), and the release workflow will throw artifact “already exists” error. The only way to avoid such errors is to ensure the release branch is merged back into the `main` once a version has been successfully released to GitHub Packages, which bumps up the version in `main` to the next snapshot.

## Error #12: An error occurred (ClientException) when calling the DescribeTaskDefinition operation: Unable to describe task definition

If by any chance your workflow fails at `describe-task-definition` step, with the error seen in the screenshot below, you need to check two things:

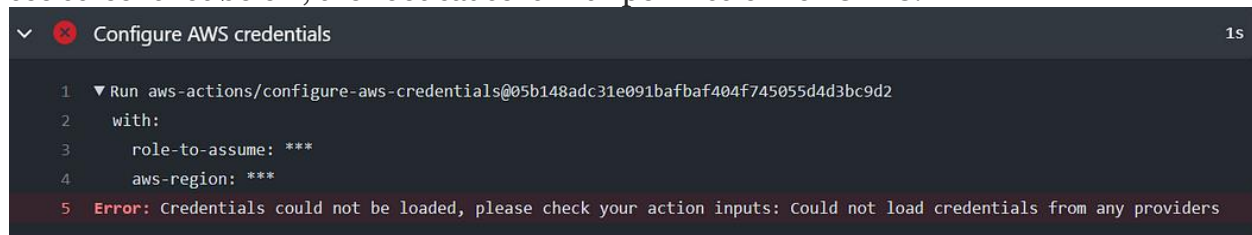
1. Ensure there is no typo in your secret configuration for `ECS_TASK_DEFINITION`.
2. If you run this workflow in an environment/account which is located in a different region than the default `AWS_REGION` defined at your repo's secret, you need to define `AWS_REGION` and `AWS_DEFAULT_REGION` in your environment secrets to overwrite the values for those secrets at the repository or org level.



```
1  ▶ Run aws ecs describe-task-definition --task-definition ***
11
12  An error occurred (ClientException) when calling the DescribeTaskDefinition operation: Unable to describe task
    definition.
13  Error: Process completed with exit code 254.
```

## Error #13: Error “Credentials could not be loaded, please check your action inputs: Could not load credentials from any providers”

If your workflow fails at the `configure-aws-credentials` step with the above error, see screenshot below, the root cause is with permission for OIDC.



```
1  ▼ Run aws-actions/configure-aws-credentials@05b148adc31e091bafbf404f745055d4d3bc9d2
2    with:
3      role-to-assume: ***
4      aws-region: ***
5  Error: Credentials could not be loaded, please check your action inputs: Could not load credentials from any providers
```

You need to add permission for OIDC `id-token: write`. If your workflow is calling a reusable workflow, permissions are set at the caller workflow, NOT in the reusable workflow, so ensure your caller workflow has the following permissions properly set (line 5–6) BEFORE you call the reusable workflow, sample snippet below:

#### **Error 14: “npm ERR! code E400”**

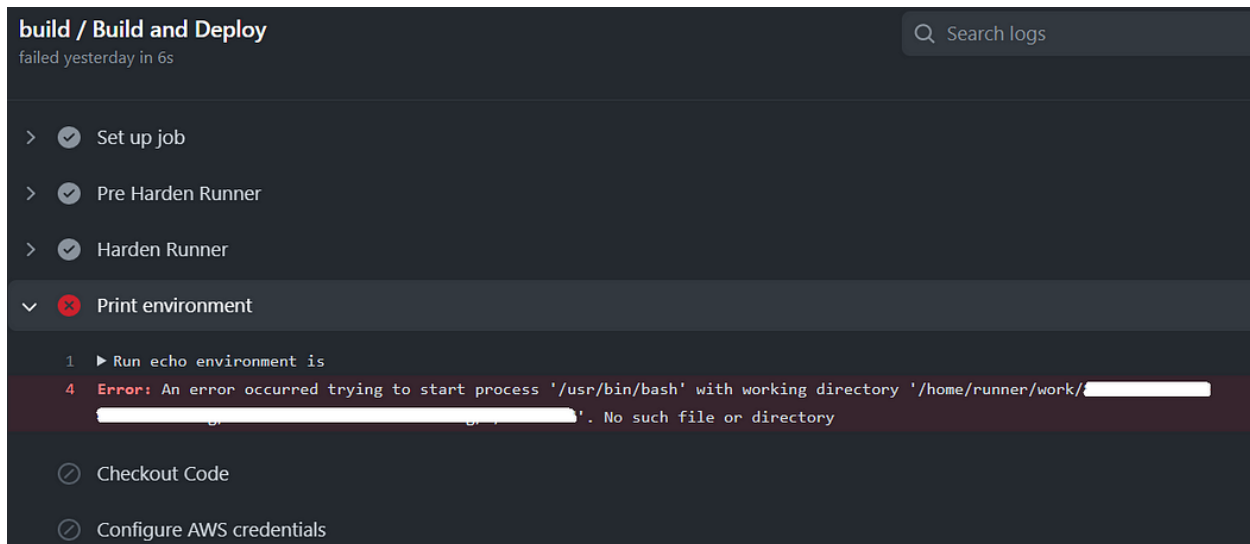
When you try to publish an npm package to GitHub Packages, if you run into this error, with details such as the following:

```
"253npm ERR! 400 Bad Request -
PUT https://npm.pkg.github.com/@<repo-name>- failed to stream
package from json: unhandled input: Cannot publish over existing
version"
```

It indicates you cannot publish the same version package to GitHub Packages as that same version already exists. So try bumping up the version and retry.

#### **Error 15: An error occurred trying to start process ‘usr/bin/bash’ with working directory**

This error appears to be tricky, especially when all working-directory and paths are defined properly for the workflow. The fix is actually really simple. You need to always put the checkout code action (`actions/checkout`) right after the harden runner step (`step-security/harden-runner`). Once the sequence is switched, this error disappears!



## Error 16: “Connection reset by peer”

If you run into intermittent error “**Connection reset by peer**” during either `terraform plan` or `terraform apply` step, sample screenshot below, be aware that this is a known issue with HashiCorp’s `terraform-provider-aws`, root cause appears to be with AWS API rate limits: [Intermittent connection reset by peer error when Terraform apply · Issue #23614 · hashicorp/terraform-provider-aws \(github.com\)](https://github.com/hashicorp/terraform-provider-aws/issues/23614).

An enhancement to retry in such an error scenario has been submitted here, [Add retry handling when a request's connection is reset by a peer · Issue #10715 · hashicorp/terraform-provider-aws \(github.com\)](https://github.com/hashicorp/terraform-provider-aws/issues/10715).

As of this writing, that enhancement is still in open status. We hope this enhancement gets implemented soon by HashiCorp team. Meanwhile, if you encounter a such error during your workflow run, please retry your workflow, the error should usually go away.

```

1  ▼ Run # convert repo and env to lower case and pass to Terraform as env variables
2  # convert repo and env to lower case and pass to Terraform as env variables
3  export DEPLOY_REPO=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs | sha256sum | cut -d ' ' -f 1)
4  export DEPLOY_ENV=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs | sha256sum | cut -d ' ' -f 1)
5  export AWS_DEFAULT_REGION=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs | sha256sum | cut -d ' ' -f 1)
6  terraform plan -input=false -var-file=.tfvars -no-color
7  shell: /usr/bin/bash -e {0}
8  env:
9    DEPLOY_REPO: [REDACTED]
10   DEPLOY_ENV: dev
11   AWS_DEFAULT_REGION: ***
12   AWS_REGION: ***
13   AWS_ACCESS_KEY_ID: ***
14   AWS_SECRET_ACCESS_KEY: ***
15   AWS_SESSION_TOKEN: ***
16
17  Error: error loading state: RequestError: send request failed
18  caused by: Get "https://[REDACTED].***.amazonaws.com/[REDACTED]": read tcp [REDACTED]:443: read: connection reset by peer
19
20  Error: Process completed with exit code 1.

```

### Error 17: Version ### not found in the setup step

This error was uncovered in the “Set up Python” step for one of our Python apps. See the screenshot below. It appears to be odd because the previous runs for that same workflow had no issue with setting up Python. The root cause of this particular issue was Ubuntu/Python version compatibility.

If your GitHub-hosted workflow runners run on `ubuntu-latest` version, which could mean varied Ubuntu version numbers depending on what the latest Ubuntu version is. For example, the following workflow's Ubuntu version was 22.04, which does not offer a python version 3.8.8 for that action as indicated in the link in that screenshot below the error message, while the workflow runs prior to that had Ubuntu version 20.04, which does offer a python version 3.8.8 for that particular action.

A classic example of version incompatibility. The fix is to change in the workflow file to point to the compatible Ubuntu version from `runs-on: ubuntu-latest` to `runs-on: ubuntu-20.04`. It would require much more effort and testing from the development team to upgrade Python version of their app, which should still be an option for the team to consider in the longer term. The best practice is to revisit this version manifest link in a period of time to see if the latest Ubuntu version does support your particular Python version and upgrade the Ubuntu version accordingly. Or better yet, upgrade the Python version of your app when the resource and time are ready.

```
✓ ✖ Set up Python

1 ▼ Run actions/setup-python@13ae5bb136fac2878aff31522b9efb785519f984
2   with:
3     python-version: 3.8.8
4     check-latest: false
5     token: ***
6     update-environment: true
7   Version 3.8.8 was not found in the local cache
8   Error: Version 3.8.8 with arch x64 not found
9   The list of all available versions can be found here: https://raw.githubusercontent.com/actions/python-versions/main/versions-manifest.json
```

To find out which version of Ubuntu your workflow runner runs on, you can expand “Set up job” step in your workflow run result page, expand “Operating System” to capture the actual Ubuntu version.

## python-ci / Build and test

failed 5 hours ago in 2m 12s

### Set up job

```
1 Current runner version: '2.299.1'
2 ▼ Operating System
3   Ubuntu
4   22.04.1
5   LTS
6 ▶ Runner Image
11 ▶ Runner Image Provisioner
13 ▶ GITHUB_TOKEN Permissions
```

We explored some of the errors I encountered during working with GitHub Actions workflows. I hope this list helps save some troubleshooting time for those who work with GitHub Actions workflows.

Happy Coding!