Davide 'CoderDave' Bennegrio for GitHub
Posted on 4 mars 2021

The GITHUB_TOKEN in GitHub Actions: How it Works, Change Permissions, Customizations

#devops #github #ci/cd #codenewbie

Today I'm gonna tell you everything about the **GITHUB_TOKEN** in GitHub Actions. You will learn what it is, **how it works**, how to **customize** its behavior, and how to limit or **change its permissions**.

Video

As usual, if you are a **visual learner**, or simply prefer to watch and listen instead of reading, here you have **the video with the whole explanation and demo**, which to be fair is much ***more complete*** than this post.

Link to the video: https://youtu.be/EK07KPEJnjY

If you rather prefer reading, well... let's just continue :)

What is GITHUB_TOKEN

Let's start with what the GITHUB_TOKEN is in GitHub Actions and how it works.

The GITHUB_TOKEN is a **special access token** that you can use to authenticate on behalf of GitHub Actions. GitHub **automatically creates** a GITHUB_TOKEN secret for you to use in your workflow, and you can use it to authenticate in a workflow run.

The way this works is that when you enable GitHub Actions in a repository, **GitHub installs a GitHub App** on that. The GITHUB_TOKEN secret is basically a GitHub App installation access token.

Before each job begins, GitHub fetches an installation access token for the job from that GitHub App. Since the App has access to a single repo, the **token's permissions are limited to the repository** that contains your workflow. And to make it even more secure, the token expires when the job is finished.

Hope the mechanism is now clearer. Let's quickly see how to use a GITHUB_TOKEN

Use GitHub Token

There are 2 ways to use the token: from *secrets* and from the *context*.

```
- uses: actions/labeler@v2
  with:
    repo-token: ${{ secrets.GITHUB_TOKEN }}
```

In this first example we use the `secrets.GITHUB_TOKEN` to consume it. As mentioned, the secret is automatically generated so you can just use it straight away.

```
- name: Create a Release
  id: create_release
  uses: actions/create-release@v1
  env:
    GITHUB_TOKEN: ${{ github.token }}
```

Here instead we use the GitHub context, which contains the token. Note that the two are equivalent.

Personal Access Token vs GITHUB_TOKEN

If you are thinking _"why should I use the GITHUB_TOKEN instead of my normal PAT?"_, remember that a Personal Access Token is always available, so if someone is able to steal that PAT they can potentially do some harm.

The GITHUB_TOKEN instead expires just right after the job is over. So even if someone is able to steal it (which is *almost impossible*), they basically can't do anything wrong.

Default Permissions

By Default, the GITHUB_TOKEN has a quite comprehensive list of permissions assigned to it:

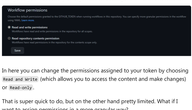
Scope	Default access (permissive)	Default access (restricted)	Maximum access by fork repo
actions	read/write	none	read
checks	read/write	none	read
contents	read/write	read	read
deployments	read/write	none	read
issues	read/write	none	read
metadata	read	none	read
packages	read/write	none	read
pull-requests	read/write	none	read
repository-projects	read/write	none	read
security-events	read/write	none	read
statuses	read/write	none	read

This table shows the permissions granted to the GITHUB_TOKEN by default. Good thing is that people with admin permissions to an enterprise, organization, or repository can set the default **permissions to be either permissive or restricted**.

The Permissions UI

So, let's see how we can ***change the permissions*** of the GITHUB_TOKEN to make it even more secure.

Just go to your repository or organization ***Settings***, then click on ***Actions***.



In here you can change the permissions assigned to your token by choosing `Read and Write` (which allows you to access the content and make changes) or `Read-only`.

That is super quick to do, but on the other hand pretty limited. What if I want to assign permissions in a more granular way?

Granular permissions via YAML

You can use the `permissions` key in the **YAML workflow** file to modify permissions for the GITHUB_TOKEN for an entire workflow or for individual jobs.

```
permissions:
  contents: write
  pull-requests: write
  issues: read
  packages: none
```

And you can use all the permissions that are listed in the table above. Additionally, as you can see below, it supports *intellisense* if you do it in the GitHub interface directly:

> When the permissions key is used, all unspecified permissions are set to no access, with the exception of the metadata scope, which always gets read access.

You can personalize the token permissions either at Job level, or at whole workflow level (or actually both):

```
{...}
permissions:
  contents: write
  pull-requests: write

jobs:
  job1:
    runs-on: ubuntu-latest

    steps:
      {...}

  job2:
    runs-on: ubuntu-latest
    permissions:
      issues: write
    steps:
      {...}
```

Conclusions

Hope you have now a better understanding about the GITHUB_TOKEN, what it does and how we can set its permissions properly. Let me know in the comment section below if you have any other questions about it.

Also, check out this video where I talk about creating Personal Access Tokens in GitHub.

Like, share and follow me for more content:

- YouTube
- Buy me a coffee
- Patreon
- CoderDave.io Website
- Merch
- Facebook page
- GitHub
- Twitter
- LinkedIn
- Podcast

Discussion (2)

Subscribe

Add to the discussion

dney · Sep 24 '21 • Edited on Sep 24

I am using Github-actions with codebuild but each time the CodeBuild stage starts to run,I am getting Error message: The Security token included in the request is invalid.please can you help with how to fix it?

```
Run aws-actions/aws-codebuild-run-build@v1
with:
project-name: CodeBuild
buildspec-override: dev-emv/buildspec.yml
env-vars-for-codebuild: TF_INPUT,
AWS_ACCESS_KEY_ID,
AWS_SECRET_ACCESS_KEY,
AWS_REGION,
ROLE_TO_ASSUME,
ROLE_DURATION_SECONDS,

env:
tf_version: latest
TG_version: latest
AWS_DEFAULT_REGION: us-east-2
AWS_REGION: us-east-1
AWS_ACCESS_KEY_ID: ***
AWS_SECRET_ACCESS_KEY: ***
AWS_SESSION_TOKEN: ***
TF_INPUT: false
ROLE_TO_ASSUME: ***
ROLE_DURATION_SECONDS: 3600
**STARTING CODEBUILD**
Error: The security token included in the request is invalid
**CODEBUILD COMPLETE**
```

1 like · Reply

Davide 'CoderDave' Bennegrio · Sep 24 '21

It means there is something wrong with your credentials. Either the accessKeyId or secretAccessKey (or both) are wrong. You can try validating your credentials using the AWS cli using the STS get caller identity call before using them in your code.

1 like · Reply

Code of Conduct · Report abuse

Read next

- **When Does Inflation Hit Cloud Prices and What Can You Do About It?** — CAST AI · Apr 15
- **50 Github Repositories for a developer** — Dhanush N · Apr 3
- **DevOps: How to Host a Simple Static Website on AWS S3** — Adam Przewoźny · Apr 16
- **How GitHub is Improving Developer Experience** — Santosh Yadav · Apr 14