Learning Objectives

Learners will be able to…

- **Explain secrets usage and importance**
- **Use secret on repo or global level**
- **Explain Vault advantages**

info

Make Sure You Know

Learners should be familiar with GitHub and its usage.

Limitations

Learners should have an active GitHub repository to explore activities.

Secrets

In CI/CD, private information such as access credentials to private GitHub repositories, user passwords for repositories, encryption keys and certificates, and other sensitive data are often used and must be stored securely.

There are several ways to store private information and keep the infrastructure secure during the CI/CD processes:

- **Repository-level secrets:** Some version control systems like GitHub provide encrypted variables that can be used to store private information at the repository level. This allows developers to share these secrets with only a few people and restrict access to those with the necessary privileges.
- **Special “Vaults”:** Vault by HashiCorp is an open source tool that provides secure storage for secrets such as passwords, tokens, certificates, and encryption keys. It also offers extra features like dynamic secrets generation, lease-based operations and revocation of secrets.
- **Infrastructure-level secrets:** At the infrastructure level, services like Kubernetes and AWS Secrets Manager make it possible to store private CI/CD information in a safe way. This lets developers store, access, and manage CI/CD secrets within their own infrastructure, making sure that only authorized people have access to them.

Secrets add an extra layer of security to CI/CD processes, making sure that only people who are allowed to see private information can do so.

Checkpoint

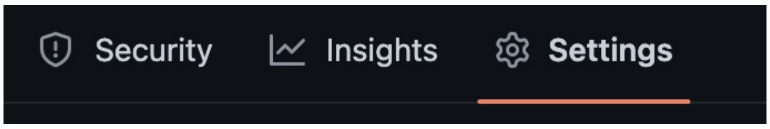
GitHub Secrets

GitHub allows users to store secrets on an organizational or repository level using **GitHub Secrets**.

Say we have a CI/CD job that needs to upload assets to our server and needs specific credentials to do so. We can put these credentials as a string-to-string mapping in **GitHub Secrets** and use the CI/CD job to access them.

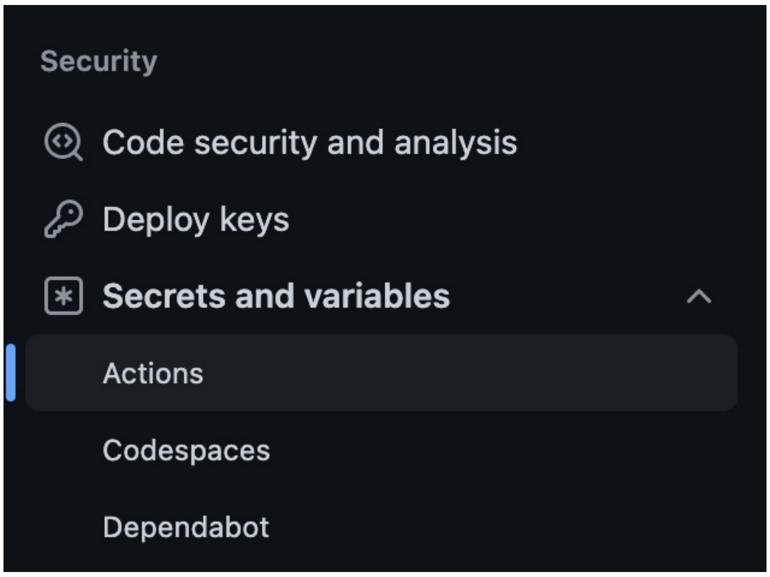
Setting Up Secrets

To set up a GitHub Secret, you will need to go to your repository on GitHub and click on the `Settings` tab.



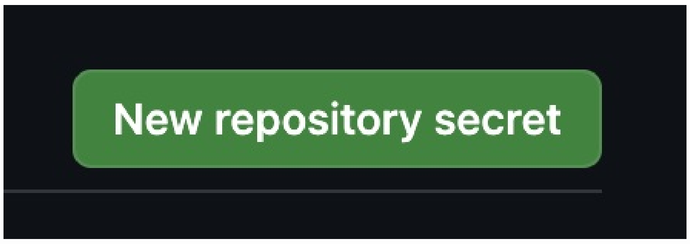
The settings tab of a github repository

From there, choose `Secrets and Variables` from the `Security` section of menu on the left and click on `Actions`.



The actions tab in the Secrets and Variables section of GitHub repository settings.

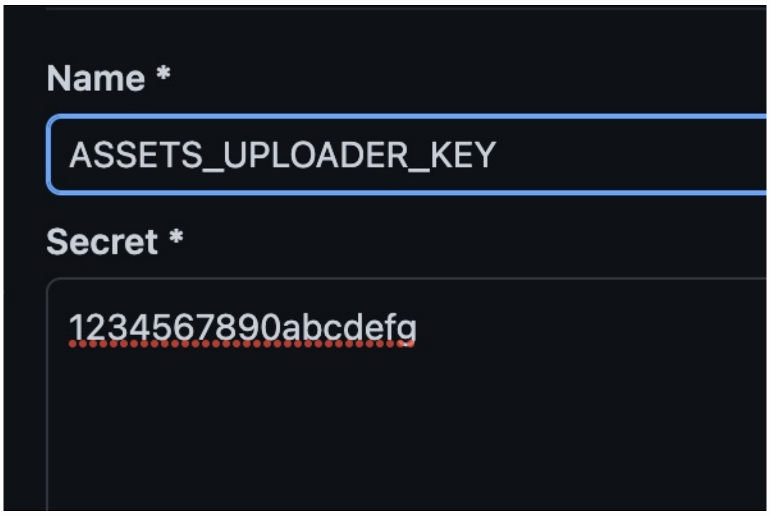
On the following screen, click the `New Repository Secret` button.



A green new repository secret button.

You’ll then be asked to give your secret a name and enter its value.

- This name will be your **GitHub Token** that represents the sensitive information placed in the value.

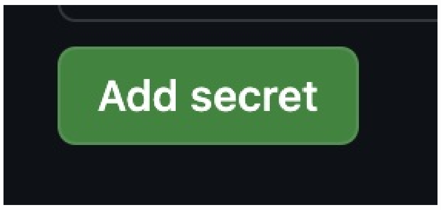


ASSETS_UPLOADER_KEY

1234567890abcdefg

GitHub name and secret definition form.

Click the `Add Secret` button once you’ve put in the necessary information.



A green add

Your secret is now safely stored and is ready to be used in **GitHub Actions**.

Take a look at the example code snippet below.

```
     - name: build
        run: |
          ./build.sh "${{ secrets.ASSETS_UPLOADER_KEY }}" "${{
secrets.ASSETS_UPLOADER_SECRET }}"
        timeout-minutes: 15
```

In the above job, we are passing the uploader’s key and secret from GitHub secrets as parameters to our `build.sh` script.

GitHub Secrets can also be used as **environment variables** in our job definition.

Let’s look at an example CI/CD job that needs access to two secrets: an `ASSETS_UPLOADER_SECRET` and an `ASSETS_UPLOADER_KEY`.

In this case, the job would be defined as follows:

```
     - name: build
        run: |
          ./build.sh
        timeout-minutes: 15
        env:
        ASSETS_UPLOADER_SECRET: "${{
secrets.ASSETS_UPLOADER_SECRET }}"
        ASSETS_UPLOADER_KEY: "${{ secrets.ASSETS_UPLOADER_KEY
}}"
```

With the `secrets` object, we can access all of the secrets that are stored in our GitHub repository. To use a secret, we only need to put its name as a key on the `secrets` object.

Now, we can automatically fetch and set the value of that specific secret into our environment variables through the CI/CD job.

By using **GitHub secrets** as environment variables in CI/CD jobs, we can make sure all of our sensitive data is stored safely and that only those who need it can access it.

Checkpoint

Vaults

Vaults is another type of storage and management system for your passwords, keys, and other secrets. Vaults also offer a secure, reliable way to store our secrets.

Role-based access control is built into vault secrets so that only the right people can see the right information at the right time.

One of the best things about vault secrets is that they are **ephemeral**. This means that passwords made by the vault disappear after a certain amount of time (usually 20 minutes). This makes security much safer because credentials that have been stolen can no longer be used in bad ways before they become invalid.

Let’s imagine some staff members need read/write access to a MongoDB database. We can generate limited-privilege 20-minute passwords from the main password vault instead of giving them the main account and password. This protects sensitive data and grants access only when needed.

If an employee’s PC is breached or they leave your company, vault-generated passwords expire and access is revoked, so you don’t need to regenerate and replace all passwords.

The same can be applied for managing infrastructure or deploying/uploading resources to the cloud provider, a temporary set of AWS Secret and key can be generated for the pipeline operation.

Vaults can also be used to securely store static secrets, such as tokens and access keys, as GitHub secrets actions. This helps ensure that the QA team, for example, only has access to what is necessary for testing but not production environments.

Take a look at the following example that uses HashiCorp vault.

```
        - name: Import Secrets
          uses: hashicorp/vault-action@v2
          id: secrets
          with:
            url: https://vault.example.com/
            role: content
            method: jwt
            secrets: |
                kv/data/content/codio-ed-dev client_id;
                kv/data/content/codio-ed-dev secret_id;
        - name: Publish
          uses: codio/codio-assignment-publish-action@master
          with:
            client-id: ${{ steps.secrets.outputs.client_id }}
            secret-id: ${{ steps.secrets.outputs.secret_id }}
            course-name: <Course Name>
            assignment-name: <Assignment Name>
            dir: ./
            changelog: ${{ github.event.head_commit.message }}
```

Vault secrets can be used by the services themselves or as part of GitHub actions. Another benefit of using vaults is auditing. All requests are logged with the vault, ensuring that any suspicious activity or attempts at unauthorized access are recorded and visible. CI/CD pipelines can then interact with the vault in order to secure and retrieve secrets during operations.

Checkpoint