Synk setup - Sast

Setting up the Snyk free version with GitHub Actions involves a few key steps: creating a Snyk account, generating an API token, storing the token securely in GitHub, and configuring your GitHub Actions workflow to run the Snyk scan.

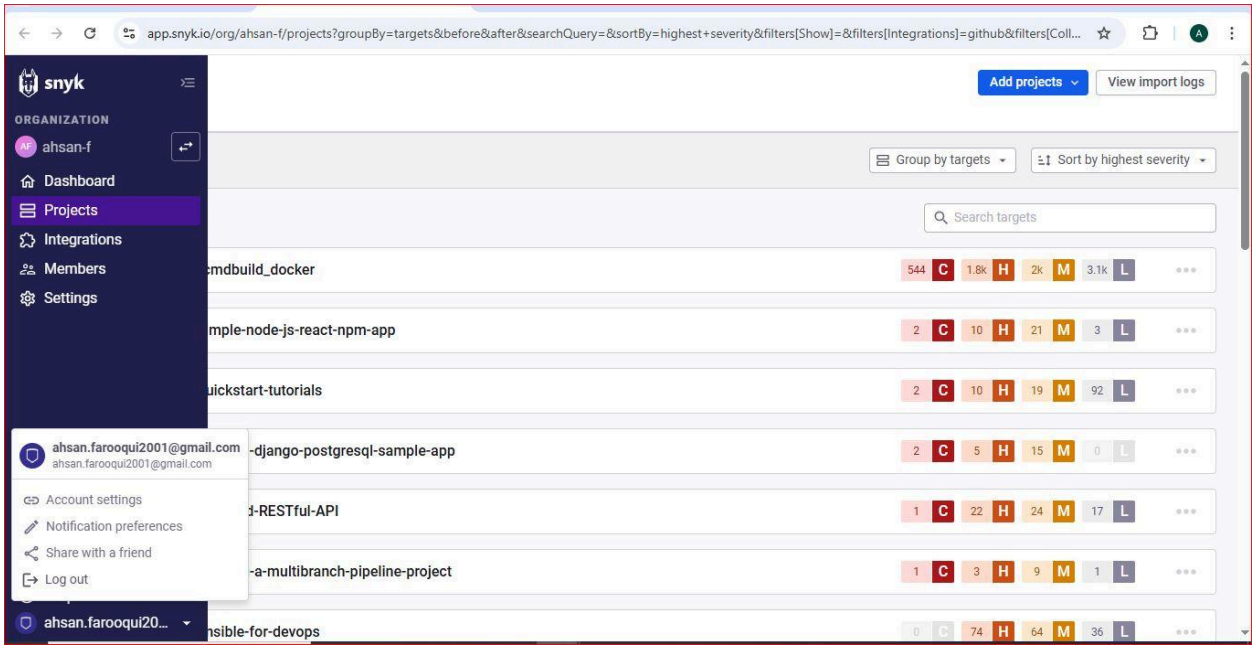
1. Create a Snyk Account and Get Your API Token

 * Sign up for Snyk: Go to the Snyk website and create a free account. You can typically sign up using your GitHub account.

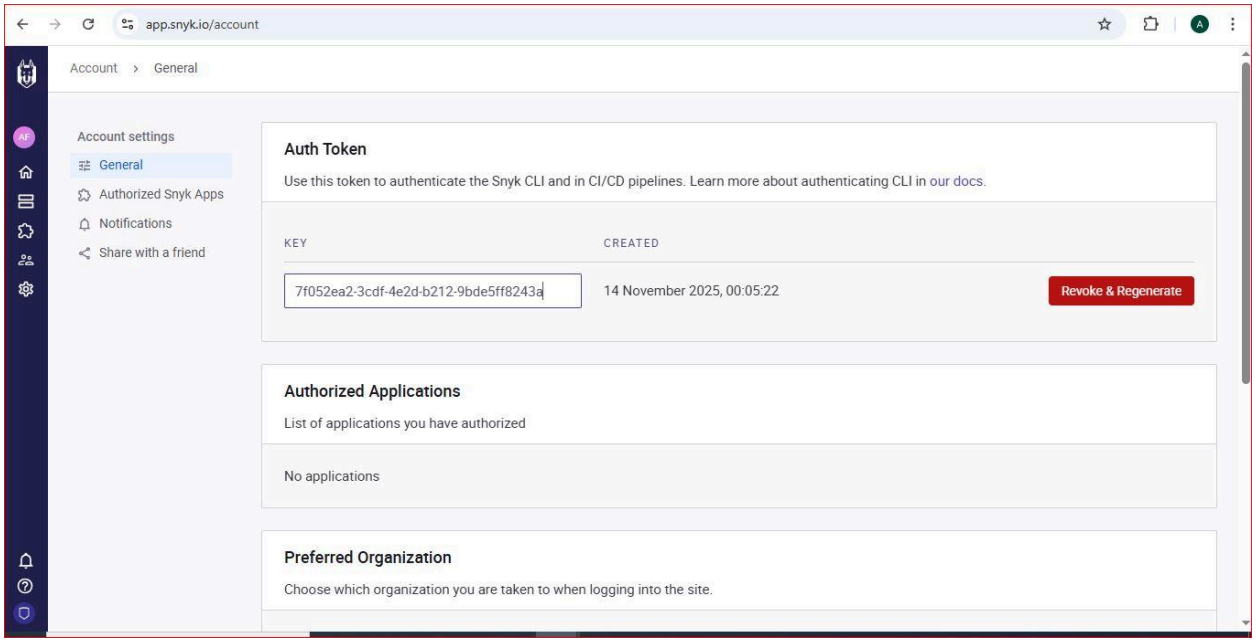
 * Generate/Retrieve API Token: Once logged in to Snyk, navigate to your Account Settings (usually under your profile icon in the top left or bottom left).

 * Go to the General tab and locate the API Token.

 * Copy the token. This token is essential for authenticating Snyk in your GitHub workflow.



Account Setting



Copy API key if already created or click Revoke & Regenerate to create new one and copy and stored it for using it with Github actions secrets.

2. Store the Snyk Token as a GitHub Secret

For security, you must store the Snyk API token as a GitHub Secret in your repository.

* In your GitHub repository, go to Settings.
* In the left sidebar, navigate to Security \rightarrow Secrets and variables \rightarrow Actions.
* Click New repository secret.
* Set the Name as SNYK_TOKEN (using this exact name is a common practice and often required by the Snyk Actions).
* In the Secret field, paste the Snyk API token you copied in Step 1.
* Click Add secret.

3. Configure the GitHub Actions Workflow

You'll create a new GitHub Actions workflow file (a YAML file) that uses one of the official Snyk Actions. This file should be placed in the .github/workflows/ directory of your repository.

Below is an example of a basic workflow using the snyk/actions/node@master action for a Node.js project. Snyk provides similar actions for other languages (e.g., maven, gradle, golang, etc.) and for container or infrastructure-as-code scanning.

Example Workflow (.github/workflows/snyk-scan.yml)

```
name: Snyk Security Scan

on:
  push:
    branches: [ main, master ]
  pull_request:
    branches: [ main, master ]
```

```
jobs:
  security:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout Repository
      uses: actions/checkout@v4

    # (Optional) Setup your language environment if Snyk Action doesn't handle it
    # For Node.js, you might need this to install dependencies
    - name: Use Node.js
      uses: actions/setup-node@v4
      with:
        node-version: '20'
        cache: 'npm' # Use appropriate package manager cache

    - name: Install Dependencies
      run: npm install # Adjust based on your package manager (e.g., yarn install, pip install)

    - name: Run Snyk to check for vulnerabilities
      # Use the action that corresponds to your project type (e.g., /node@master for Node.js)
      uses: snyk/actions/node@master
      env:
        # Use the secret you created in Step 2 for authentication
        SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
      with:
        # 'test' scans for vulnerabilities and fails the build if issues are found
        command: test
        # Optionally add arguments, e.g., to report results to GitHub Code Scanning
        args: --sarif-file-output=snyk.sarif --file=package-lock.json

    - name: Upload Snyk scan results to GitHub Code Scanning
      if: always() # Ensures this step runs even if the snyk test fails
      uses: github/codeql-action/upload-sarif@v3
      with:
        sarif_file: snyk.sarif
```

4. Commit and Test
 * Commit the new .github/workflows/snyk-scan.yml file to your repository.
 * The workflow will automatically trigger based on the events you specified (e.g., a push or pull_request to main/master).
 * Go to the Actions tab in your GitHub repository to monitor the run.

If the scan is successful, you'll see the results in the job log, and if you include the SARIF upload steps, you'll also see results in the Security tab under Code scanning alerts. The Snyk

free plan supports scanning public repositories and a limited number of tests for private repositories.