REST APIs — The Silver Bullet in Pipeline Automation

As we know, there is no silver bullet in IT. Crowning REST APIs as the silver bullet in pipeline automation is only meant to emphasize the great potential REST APIs can bring for the whole automation landscape in the CI/CD pipelines. How so? Let's explore it through a few use cases.

Use Case 1: Automating SonarQube Project Creation in the CI Pipeline

Adding SonarQube scan in the GitHub Actions CI pipeline is easy, we can simply call the following action sonarsource/sonarqube-scan-action:

```
- name: SonarQube Scan
  uses: sonarsource/sonarqube-scan-action@427bad70165a74f9d7133d48ea31b482a53ee9fa
  env:
    SONAR_TOKEN: ${{ secrets.SONARQUBE_TOKEN }}
    SONAR_HOST_URL: ${{ secrets.SONARQUBE_HOST }}
```

However, automation for SonarQube scan should not stop at this step. This step assumes your project has already been manually set up in SonarQube. Let's push the automation yardstick a bit further — to include the project creation in SonarQube in your GitHub Actions CI workflow as well. This is where SonarQube REST APIs shine.

Let's think a bit and gather the requirement. We want our CI workflow to achieve the following:

- Query SonarQube by triggering the SonarQube API call to see if our project already exists.
- If it already exists, scan by calling maven sonar plugin or the SonarScan action mentioned above.
- If the project doesn't exist already, create the project in SonarQube first by triggering the SonarQube API call, then proceed to scan.

Let's explore the SonarQube documentation on projects related to APIs. Here are the two APIs we are interested in:

- GET api/projects/search
- POST api/projects/create

Putting them into our GitHub Actions CI workflow, we now have the following:

- Line 12 calls the search API by passing in the PROJECT_KEY as request parameter. It then filters the search response by calling jq -r '.components[].key', which extracts the key element from the components list. The result of this is assigned to PROJECTS.
- Line 14 checks if PROJECT_KEY passed in exists in the filtered project result from the search API call. If it exists already, line 15 merely prints that the project already exists, and we can proceed to scan.
- Line 17–20 in the else block triggers a POST create API call to create the new project in SonarQube.
- Line 24–27 runs sonar scan for the project.

Also to mention, the conditional statement on line 2 is based on the value passed in from the input parameter sonar-scan-flag. This can be defined in a reusable workflow. The calling workflow can then pass in this flag as true and the PROJECT_KEY to enable the workflow to create projects in SonarQube and run a Sonar scan.

Surprise your SonarQube admin with automating Sonar scan for your project through your CI workflow. No more manual project creation is needed in SonarQube. Power of automation in the pipeline through REST APIs.

Use Case 2: Automating Dynamic Application Security Testing (DAST) With Acunetix in CD Pipeline

Acunetix is an automated web application security testing tool that audits your web applications by checking for vulnerabilities like SQL Injection, Cross-site scripting, and other exploitable vulnerabilities. Acunetix scans any website or web application accessible via a web browser and uses the HTTP/HTTPS protocol.

Let’s explore Acunetix REST API to incorporate Acunetix DAST scan into our CD pipeline. Thanks to the great instructions provided by Acunetix, adding an Acunetix scan as an additional step in our CD workflow is as simple as adding these lines below:

```
- name: Trigger Acunetix Scan
  run: |
    curl -k -i  --request POST \
          --url "${{ secrets.ACUNETIX_URL }}" \
          --header "X-Auth: ${{ secrets.ACUNETIX_APIKEY }}" \
          --header "content-type: application/json" \
          --data '{"profile_id":"11111111-1111-1111-1111-111111111111",
"schedule":{"disable":false,"start_date":null,"time_sensitive":false}, "user_authorized_to_scan":"yes",
"target_id":"${{ inputs.target-id }}"}'
```

If your GitHub secrets are correctly configured, when triggering this workflow, you will notice the workflow result returns HTTP status code 201, indicating Acunetix scan for the specified target id has been successfully kicked off from the pipeline.

Notice this does not mean your scan result is successful. It only means your scan has been successfully kicked off. Acunetix scan sometimes can last hours, depending on the application, so it’s not feasible for it to keep our workflow hanging. Instead, Acunetix has an email notification feature that can be enabled for your project so the scan result can be emailed to you upon scan completion.

Use Case 3: Managing GitHub Packages With REST APIs in the Release Pipeline

Assume you use GitHub Packages as an artifact registry for your Java maven projects. You may run into scenarios where you need to manually delete certain package versions, such as when your maven release workflow unexpectedly fails due to reasons such as GitHub throwing 500 internal server errors during peak load. Rolling back a maven release is tricky; you can try running `mvn release:rollback`, but I have noticed this rollback command doesn’t always do a thorough job. Sometimes manual verification and/or manual deletion of the package version is still required.

Thankfully GitHub Packages offers a set of REST APIs. You can now call some of those APIs to automate maven release rollback in case your maven release or maven release rollback fails.

The sample code snippet below carries out the following actions:

- Runs `mvn help:evaluate` to discern the project version from its pom file, see line 3.
- Calls GitHub Packages API to get all the versions for a particular artifact (`<PACKAGE_NAME>`), see lines 8–12.
- Find out the number of different versions of that package in response to the REST API call above. See line 16.
- Parse the response to find the package version id matching the package version. See line 20.
- If the package only has one version, delete the package by calling the delete package REST API. See lines 23–30.
- Otherwise, if the package version was found and multiple versions for that package were found, delete the version which matches the pom version by calling the delete package version REST API. See lines 31–38.

This is an example of how you can use GitHub Packages REST APIs fluently in GitHub Actions workflows to manage the packages if needed.

Many other use cases for REST APIs can do the heavy lifting in your pipeline automation. Take a closer look at your pipelines, especially in the actions/steps where your pipeline interacts with third-party tools/frameworks, which are great candidates for further automation. Explore the REST APIs offered by those tools/frameworks, and you will have some aha moments on taking your automation in your pipelines to the next level.

Pipeline Security

With calling REST APIs in GitHub Actions workflows, you may wonder how we ensure pipeline security. Glad you asked. This is where you can rely on StepSecurity's Harden-Runner. Harden-Runner guards your pipelines by whitelisting the outbound endpoints for each workflow step. With Harden-Runner hard at work, you can have peace of mind knowing that your pipelines are free of supply chain attacks.

I have explored Harden-Runner in my previous blogs before. Feel free to check out my blogs on Harden-Runner:

- A First Look at Harden-Runner: The Must-Have GitHub Action To Prevent Supply Chain Attacks
- DevOps Self-Service Centric Pipeline Security and Guardrails

Summary

Automation opportunity in the pipelines is boundless. GitHub Actions offers the flexibility and feasibility to incorporate many actions to automate many tasks. However, when you need more ready-made action in the marketplace or want to push your automation to the next level, look to REST APIs offered by your specific task vendors.

More often than not, your vendors already offer a set of APIs to help automate tasks that are usually performed manually. Always ask yourself: is there more room for automation in the pipelines? Explore the great potential those REST APIs can bring to your pipelines to automate your day-to-day tasks.

Happy automating! Happy coding!

References

Web API — SonarQube
https://www.acunetix.com/support/docs/introduction/
https://www.acunetix.com/blog/web-security-zone/acunetix-rest-api/
https://www.acunetix.com/support/docs/wvs/integrating-acunetix-with-github-for-ci-cd/
https://docs.github.com/en/rest/packages?apiVersion=2022-11-28
https://www.stepsecurity.io/
https://www.stepsecurity.io/products/harden-runner