Event Driven Closed Pull Requests On GitHub

Wednesday, 25 August 2021

Prepared by:
Kseniya Stadnik
AWS ProServe Associate Consultant

James Jacob
AWS ProServe DevOps Consultant

TABLE OF CONTENTS

1. Introduction

Business Requirement

AWS CodeBuild does not natively support webhook triggers on GitHub PULL_REQUEST_CLOSED events. Currently, webhook filter groups can be used to specify which GitHub webhook events trigger a build. These events include PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_REOPENED, and PULL_REQUEST_MERGED. A common use-case for intercepting closed pull request events is to perform cleanup actions in a CI/CD driven workflow.

The focus of this document is the design and implementation of a custom architecture to support intercepting GitHub PULL_REQUEST_CLOSED events to perform complex actions using AWS services based on this event.

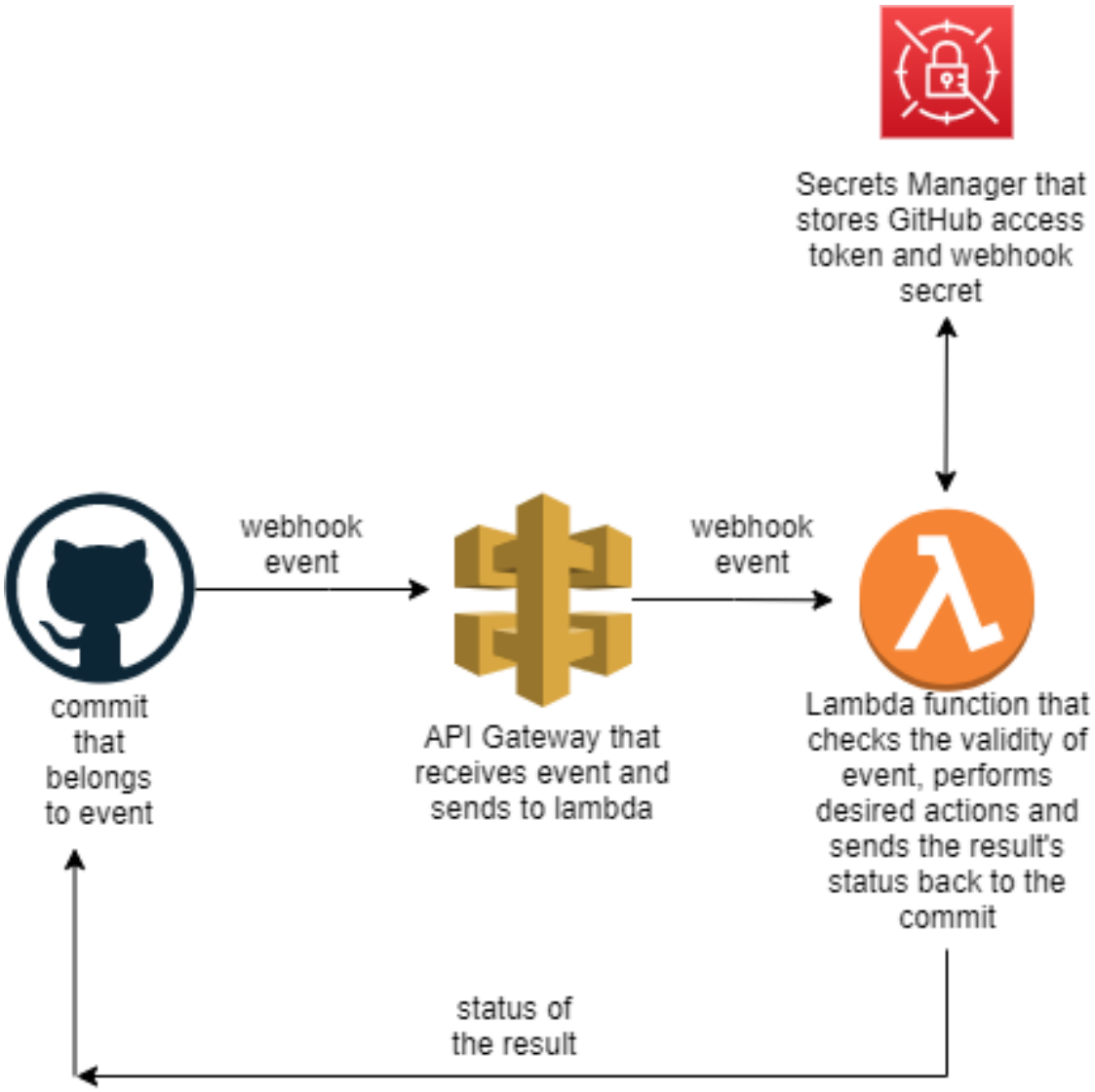
AWS Services used

- AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

- AWS API Gateway makes it easy to create, publish, maintain, monitor, and secure APIs at any scale.

- AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources.

2. Solution Architecture



This solution requires any GitHub Pull Request Event to be initiated by the user. Once the event is triggered, a HTTP POST payload will be sent to the webhook's configured URL, which is an API Gateway endpoint URL.

The API Gateway then sends the event to the Lambda function that checks for event validity and security, performs required actions and returns the status of the build back to the commit that triggered the event.

3. IMPLEMENTATION

Create AWS resources

During this step AWS API Gateway, Secrets Manager Secret, Lambda will be created.

The cloud infrastructure is defined in code and provisioned through AWS CloudFormation using AWS CDK (instructions to install aws-cdk).

To deploy required AWS Resources, one needs to go inside the cdk folder and do the following:

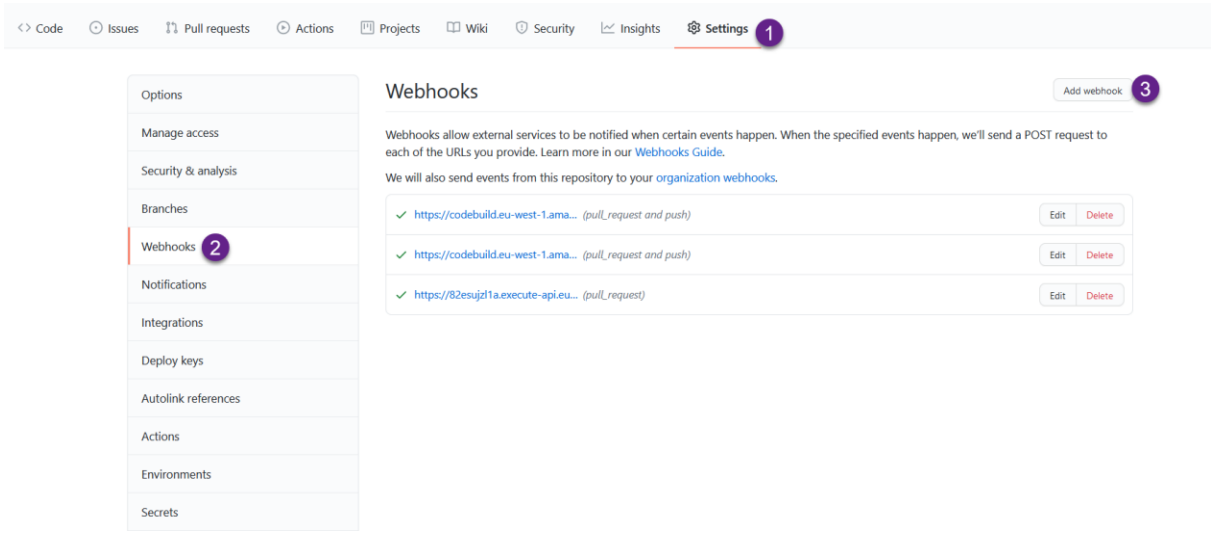
1. Change parameters value inside cdk.context.json file.
2. Run the command below to deploy resources:

```
cdk deploy closed-pr-event-stack
```

As a result, the required infrastructure components needed for the provided solution will be created.

Create GitHub webhook

Once the AWS resources are created, webhook can be created from GitHub:
1. Navigate to the desired repository
2. Go to Settings -> Webhooks
3. Click Add webhook



4. In Payload URL field enter API Gateway endpoint URL received from cdk output;
5. Select ‘application/json’ as a Content type;
6. Get the secret value from a Secret Manager that was created during the deployment of AWS services and paste it to webhook secret.
7. Tick ‘Pull requests’ in a selection of individual events.
8. Click ‘Add webhook’

As a result, GitHub webhook will be created and will send any pull request events, including needed ‘close’ event, to an AWS API endpoint.

4. Solution Logic

GitHub actions sometimes can be considered as an alternative of the automation in GitHub. If you are working with private repositories and you need to perform some actions in repository A based on the events happened in repository B, in order for GitHub actions to work properly during the communication, GitHub actions need to used GitHub personal access token.

GitHub token can be stored as Organization/Repository secret in GitHub. Very often Business does not want to store sensitive information in this way. Additionally, in case of Repository secret anyone who has access to GitHub Settings can view that secret. Very often you don’t want to open a secret to every user of that repository and you want to restrict access to the secret in a similar way of how AWS IAM Roles limit access to resources.

Unfortunately, currently there is no integration between AWS Roles and GitHub. Therefore, the solution was to use AWS Secrets Manager and store the GitHub personal access token there. Then using AWS IAM Roles restrict access to this secret as required.

Since GitHub actions were not chosen as a service that can perform actions based on different GitHub events because of lack of required security level and since CodeBuild can only be triggered on a limit GitHub events, AWS Lambda has been chosen as a service that can perform desired actions when it receives a WebHook event.

Next step was to find a way of how to send webhook events to lambda function in a secure way. As a result, AWS API Gateway with resource policy has been chosen as an intermediary between Lambda and GitHub.

Once Lambda function is configured to receive payloads from API Gateway, it'll receive any payload sent to the configured API Gateway endpoint. For security reasons, it’s better to limit requests to those coming from GitHub. One of the way of doing this is to set up a secret token and validate the information in the Lambda function itself. This secret token is called WebHook Secret and the value for this secret will be stored in a Secrets Manager with the same reasons as for GitHub Personal Access Token plus the ability for Lambda function to retrieve it during the security check process.

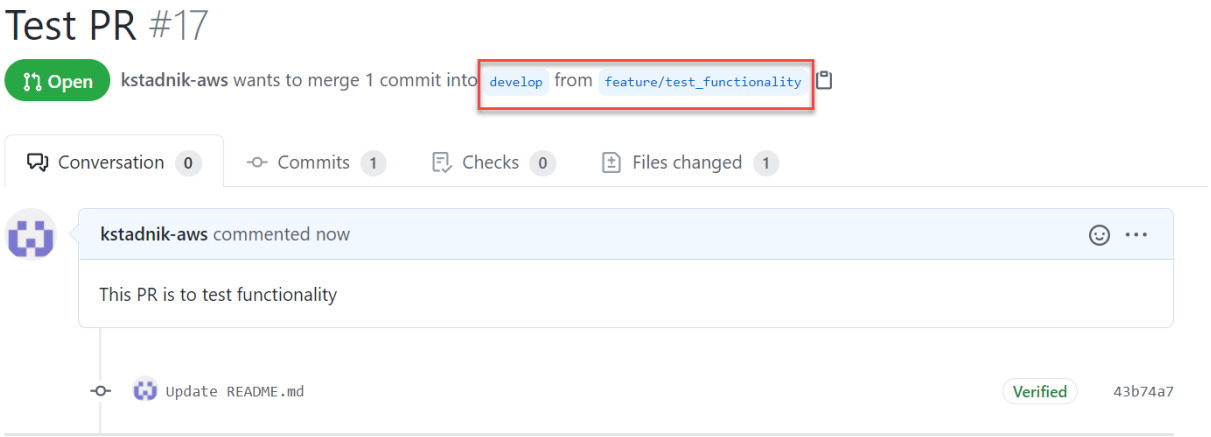
When WebHook Secret is set, GitHub uses it to create a hash signature with each payload. This hash signature is included with the headers of each request as X-Hub-Signature-256. The intention is to calculate a hash using WebHook secret value stored in Secrets Manager, and ensure that the result matches the hash from GitHub. GitHub uses an HMAC hex digest to compute the hash.

As a result, when Lambda function receives a payload, it will first check the incoming event for security (line 44-58) and in case if event is secure proceed, otherwise send a message 'Unauthorized request'.

5. DEMONSTRATION

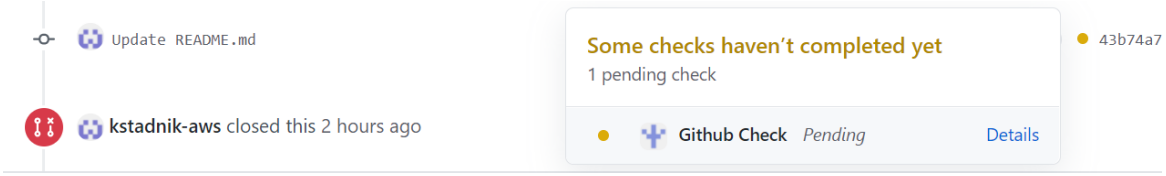
As a demonstration that proposed solution works correctly, a dummy Pull Request will be opened to a develop branch and then closed.

1. Open a dummy Pull Request:

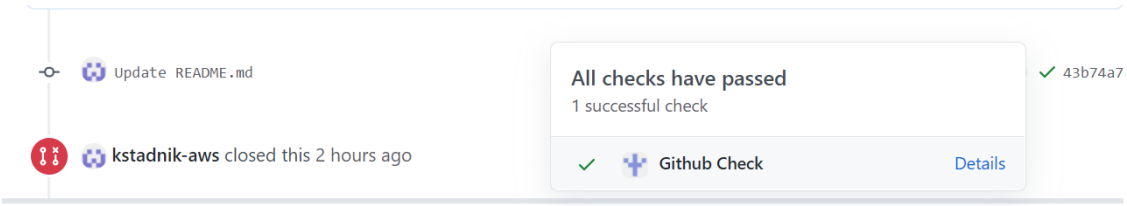


2. Close a dummy Pull Request:

After PR is closed, the status of the commit will be in 'pending' status



When Lambda function finishes its actions, if successful, the status of the commit will be 'success'. In case if Lambda's execution is failed, the status of the commit will be 'failure'



Details stores the link to the Lambda's execution logs.

Next steps

The provided solution can be generalized further for any webhook events, not only CLOSED_PULL_REQUEST. When Lambda function receives an event, it can detect the type of the event (e.g. pull_request, push and etc.) and based on that type performs needed actions. Therefore, by changing code in Lambda function, more customization can be achieved.