SUSE Hack Week 2024 - Day 3

Date:	2024-11-20
modified:	2024-11-20
tags:	SUSE, openSUSE, pagure, HackWeek, AWS, CodePipeline, Coding, OpenSource
description:	Experiences and outcome of my third day at SUSE Hack Week 2024
category:	Code
slug:	suse-hack-week-2024-day-3
Author:	Dominik Wombacher
lang:	en
transid:	suse-hack-week-2024-day-3
Status:	published

I continued where I left off on the second day, I was thinking about my architecture last night. AWS CodePipeline requires a bit of a custom implementation within the AWS Account, for example with AWS Lambda, to handle third party repos that are not GitHub, Gitlab or Bitbucket. Maybe the pagure ci plugin should then a generic webhook style implementation. It defines the payload that is send and the expected answer to update the PR status. But leaves the actual implementation up to the target system. That way, the plugin isn't limited to one single service. I can use it to interact with AWS CodePipeline, or directly AWS CodeBuild, or any other system were I can implement the server side logic myself.

This shifts the scope of my project a little, but the outcome stays the same. I probably need two AWS Lambda functions, one to receive the webhook and trigger the build. Another to recieve status updates from CodeBuild phases and send them back to pagure. The AWS credentials for pagure need permission to trigger the Lambda function for incoming requests. The function has permissions to interact with S3 and CodePipeline / Codebuild. The outgoing Lambda uses the pagure ci token and gets it from a Env Var or AWS Secrets Manager. Every component follows the least privilege principle.

The pagure ci jenkins plugin routes expect a `POST` request but the token is part of the URL. My plugin will use routes like `/ci/webhook/<repo>` instead of `/ci/jenkins/<repo>/<pagure_ci_token>/build-finished`. The token has to be submitted as part of the request. This keeps the token away from plain text logs, it's still a secret and should be treated like that.

With the idea of a generic plugin in mind, Authentication becomes a challenge. Calling an AWS Lambda on its function URL works best with a SIGv4 authentication request. For any other, basic authentication seems the most common and reasonable way. I have to figure out how to handle this properly. I don't want to add more fields to pagure ci and the Web UI. First thing in mind: URI Scheme. `https` means plain request with basic auth based on `ci_username` and `ci_password`, if set. `https+sigv4` = AWS SIGv4 authentication request, uses `ci_username` (Access Key) and `ci_password` (Secret Key).

I'll add the Lambda code and a IaC example (AWS CDK, AWS CloudFormation or OpenTofu/Terraform) to the pagure repo. That way the end-to-end integration becomes a first-class citizen in pagure, not just the ci plugin.

The time I could invest today was pretty limited and focused on further Architecture details. I started with the pagure ci plugin, it is in an early stage but I'm confident that there is more to demonstrate tomorrow.