QA Test Automation

# Context

You have just joined GitGuardian as the first QA for the Internal monitoring product.

This product aims at monitoring the entire codebase (public and private repositories) belonging to a company or to an individual developer and to look for secrets such as API keys, database connection strings or security certificates that are exposed there.

First contracts have been signed, the product has great traction and the tech team is growing. Given the sales prospects and our intrinsic value proposition that deals with very sensitive data, the quality is becoming an essential aspect.

Your first mission is for you and the technical team to be as confident as possible on each product version for the existing core features.

# Guidelines

Our test strategy is based on many tests types that we would like to improve and to create from scratch, based on you knowledge, check one of the two exercice and provide a solution that you can push any VCS (Github, Gitlab, or Bitbucket) to share your repository link for assessment purposes.

Answer the first questions and choose whenever you are more comfortable , you should spend around 3h to get this work.

1. Discover the product and its core features by signing up to the app on [https://dashboard.gitguardian.com](https://dashboard.gitguardian.com/).
2. From the dashboard, could you provide a list of tests that could be performed
3. Provide a use case where you can simulate a non functional test
4. Provide a best practice for automation tests, test plan, how this will manage easily for you

**API test**

1. Go through the api based on the documentation: <https://api.gitguardian.com/docs/>

and following the instruction of usage,provide at least 2 collections of tests using any API Tooling that you are more comfortable to use where you:

Check the health

Perform a scan from your repositories

Get list of incidents

Perform an assign/unassign resolve, add a comments by looping on the incidents list

**Web Automation**

Use the website <https://dashboard.gitguardian.com> and perform the following automation task

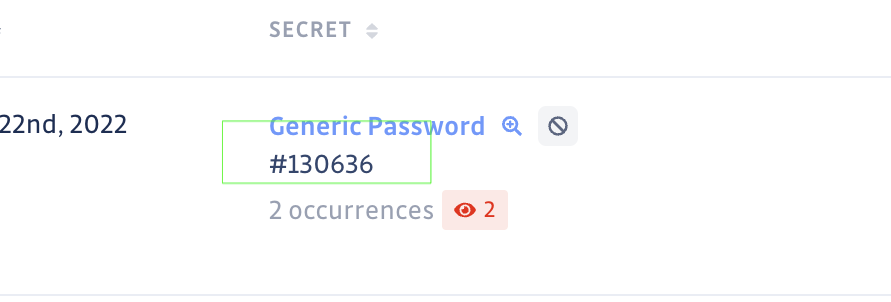
1. Perform the successful login
2. Verify that the incident id for the specific secret incident is the same on the incident details page. Below there are steps to achieve:

**Prerequisites**:

You should have a secret incident in the dashboard

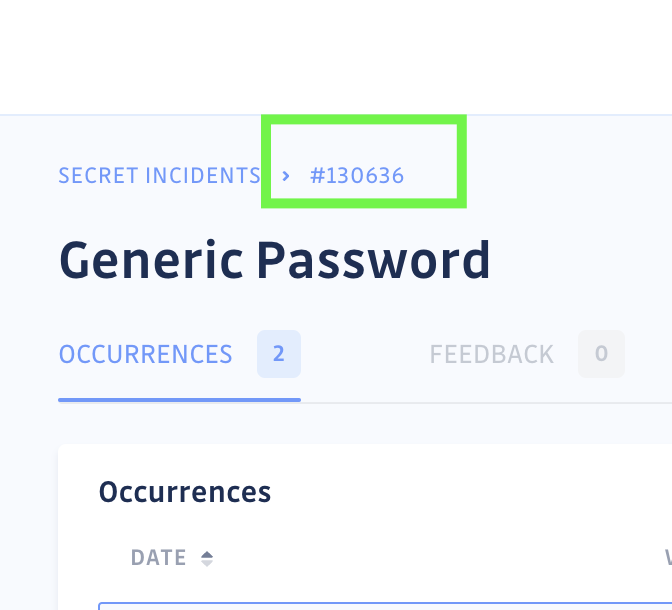
1. Login to the dashboard

2. Select the incident (By clicking on the secret)



3. Verify that incident detail page displaying

4. Verify that the Incident detail page incident id is the same as the Incident page.



Our current stack in E2E Test is based on the Cypress-based automation framework. But feel free to manage the exercise with any automation tool.

You are completely free to present those elements and the decisions you have made in your preferred format. You can do slides for example or use a google sheet / excel as support for your presentation. You can use different resources to better understand what we do (<https://docs.gitguardian.com/>, <https://blog.gitguardian.com/> )

The format of the meeting with the management is a conversation of 45min around your presentation. Written documents must be in English but for french candidates: the conversation will take place in French (at least for the most part ;)).

## Bonus questions

1. Detect the language / framework and paradigm we are using in our product
2. Find a bug :)

# Notes

If you need any clarifications, please send us an email with all your questions. It will not penalize you at all.

# 

# Glossary

Secret: A secret is any of the following: API keys, database connection strings, certificates.

Version control system (VCS): A VCS is a tool that enables the tracking and the management of changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. Eg: GitHub, GitLab, Bitbucket.

Repository: Following [GitHub's definition](https://help.github.com/en/github/getting-started-with-github/github-glossary#repository), a repository is the most basic element of GitHub. They are the easiest to imagine as a project's folder. A repository contains all of the project files (including documentation), and stores each file's revision history. Repositories can have multiple collaborators and can be either public or private.

Commit: A commit is a Git object first and is an individual change to a file (or set of files) within a repository.