# Secret Scanning:

When a repository is enabled for secrets scanning, GitHub scans entire Git history on all branches present in the repository, even if the repository is archived. If any string matches with patterns provided by secret scanning partners, by other service providers, or defined by the organization, are reported as alerts in the "**Security**" tab of repository.

If secret scanning detects a secret, GitHub generates an alert and

* Sends an email alert to the repository administrators and organization owners.
* Sends an email alert to the contributor who committed the secret to the repository, with a link to the related secret scanning alert. The commit author can then view the alert in the repository and resolve the alert.
* Displays an alert in the "Security" tab of the repository

Organization owners can also enable secret scanning for advanced security for all repositories or for all new repositories within an organization.

**Secret scanning patterns:**

GitHub Enterprise Cloud maintains these different sets of secret scanning patterns:

* **Partner patterns.** Used to detect potential secrets in all public repositories. For details, see "[Supported secrets for partner patterns](https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/secret-scanning-patterns#supported-secrets-for-partner-patterns)."
* **Advanced security patterns.** Used to detect potential secrets in repositories with secret scanning enabled.

The GitHub scans for secrets issues by different service provides by leveraging the secret pattern specific to the issues providers. The GitHub supports many popular service providers like Adobe, AWS, Azure, Atlassian, Google and many other.

For complete list: <https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/secret-scanning-patterns#supported-secrets-for-advanced-security>

**Custom secret patterns:**

In addition to the above list of secret patterns an organization can define its own custom patterns for secret scanning.

For more details: <https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/defining-custom-patterns-for-secret-scanning>

* **Push protection patterns.** Used to detect potential secrets in repositories with secret scanning as a push protection enabled. For details, see "[Supported secrets for push protection](https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/secret-scanning-patterns#supported-secrets-for-push-protection)."

**Push Protection:**

**Note:** Secret scanning as a protection push is currently in beta and subject to change. To request access to the beta release, need to contact [GitHub account management team](https://github.com/enterprise/contact).

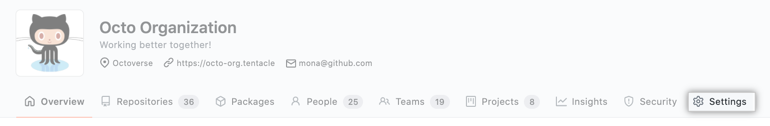
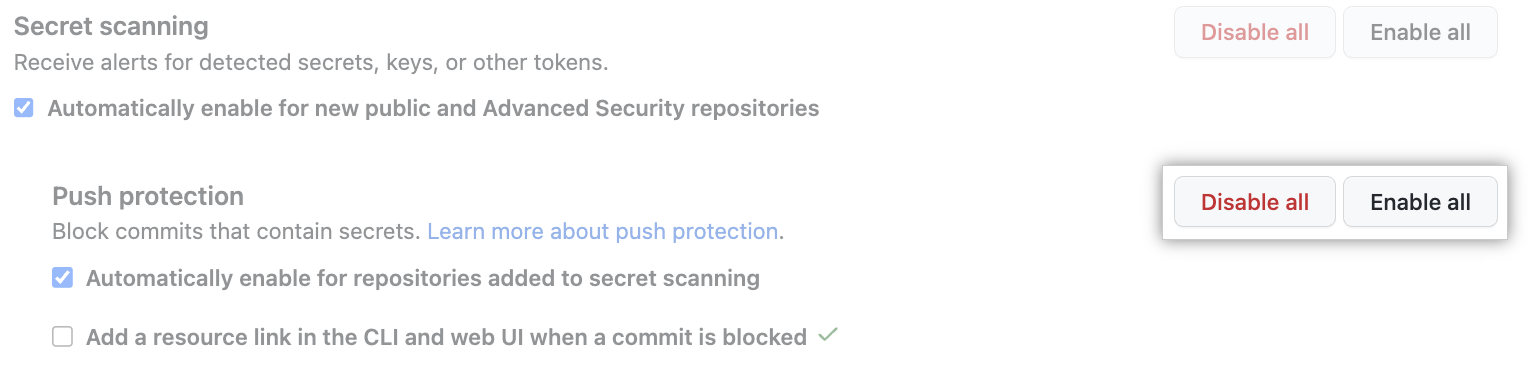
In general, secret scanning for advanced security checks for secrets *after* a push and alerts users to exposed secrets. But on enabling push protection, secret scanning also checks pushes for high-confidence secrets. Secret scanning lists any secrets it detects so the author can review the secrets and remove them or, if needed, allow those secrets to be pushed.

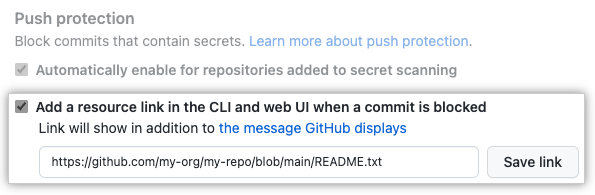
If a contributor bypasses a push protection block for a secret, GitHub:

* generates an alert.
* creates an alert in the "Security" tab of the repository.
* adds the bypass event to the audit log.
* sends an email alert to organization owners, security managers, and repository administrators, with a link to the related secret and the reason why it was allowed.

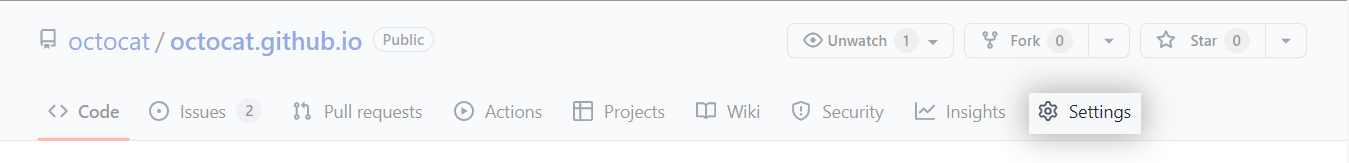
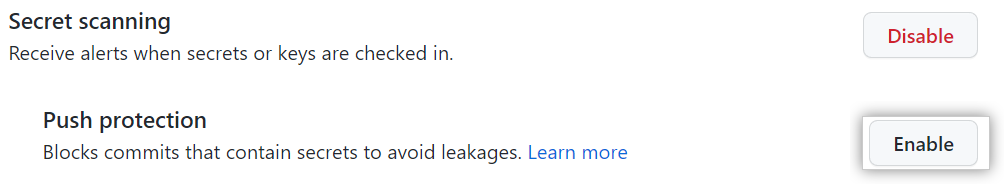
Organization owners, security managers, and repository administrators can enable push protection for secret scanning via the UI and API.

### **Enabling secret scanning as a push protection for an organization**

* On GitHub.com, navigate to the main page of the organization.
* Under your organization name, click Settings.
* 
* In the "Security" section of the sidebar, click Code **security and analysis**.
* Under "Code security and analysis", find "GitHub Advanced Security."
* Under "Secret scanning", under "Push protection", click **Enable all**.
* Optionally, click "Automatically enable for repositories added to secret scanning."
* Optionally, to include a custom link in the message that members will see when they attempt to push a secret, select **Add a resource link in the CLI and web UI when a commit is blocked**, then type a URL, and click **Save link**.
* **Note:** The ability to add resource links to blocked push messages is currently in public beta and subject to change.



### **Enabling secret scanning as a push protection for a repository**

* On GitHub.com, navigate to the main page of the repository.
* Under your repository name, click Settings.
* In the "Security" section of the sidebar, click Code **security and analysis**.
* Under "Code security and analysis", find "GitHub Advanced Security."
* Under "Secret scanning", under "Push protection", click **Enable**.

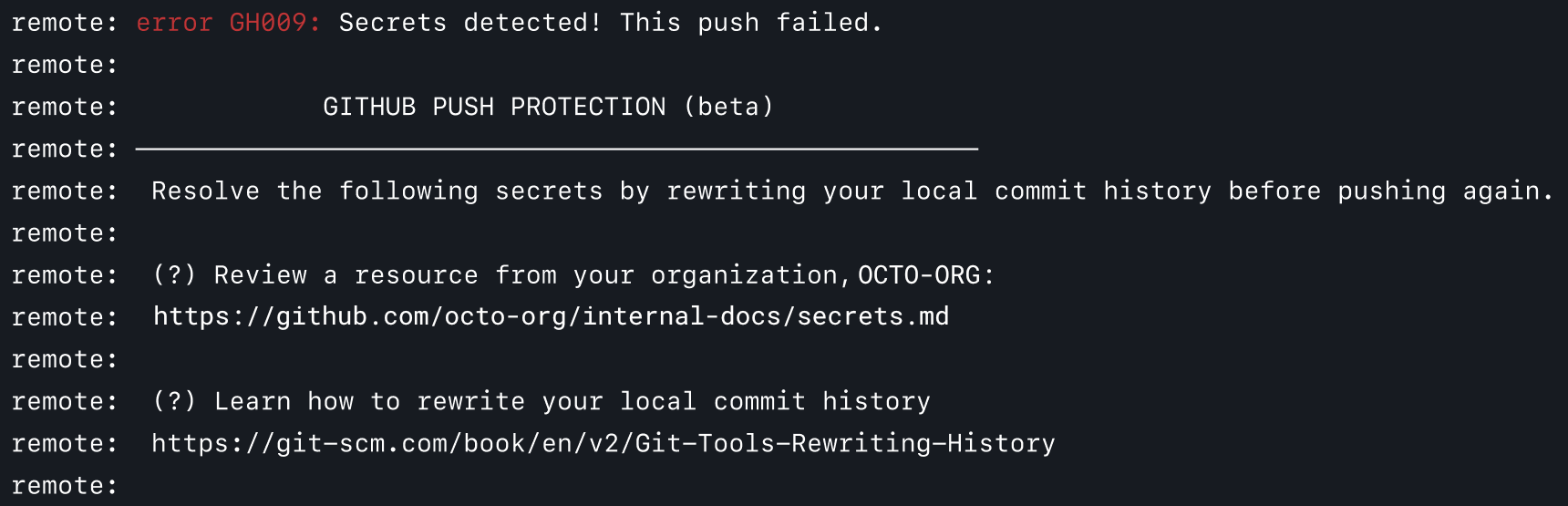
## **Using secret scanning as a push protection from the command line**

When you attempt to push a supported secret to a repository or organization with secret scanning as a push protection enabled, GitHub will block the push. You can remove the secret from your branch or follow a provided URL to allow the push.

Up to five detected secrets will be displayed at a time on the command line. If a particular secret has already been detected in the repository and an alert already exists, GitHub will not block that secret.

Organization admins can provide a custom link that will be displayed when a push is blocked. This custom link can contain organization-specific resources and advice, such as directions on using a recommended secrets vault or who to contact for questions relating to the blocked secret.

**Note:** The ability to add resource links to blocked push messages is currently in public beta and subject to change.



If you confirm a secret is real, you need to remove the secret from your branch, *from all the commits it appears in*, before pushing again. For more information about remediating blocked secrets, see "[Pushing a branch blocked by push protection](https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/pushing-a-branch-blocked-by-push-protection#resolving-a-blocked-push-on-the-command-line)."

If you confirm a secret is real and that you intend to fix it later, you should aim to remediate the secret as soon as possible. For example, you might revoke the secret and remove the secret from the repository's commit history. Real secrets that have been exposed must be revoked to avoid unauthorized access. You might consider first rotating the secret before revoking it. For more information, see "[Removing sensitive data from a repository](https://docs.github.com/en/enterprise-cloud@latest/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository)."

**Notes**:

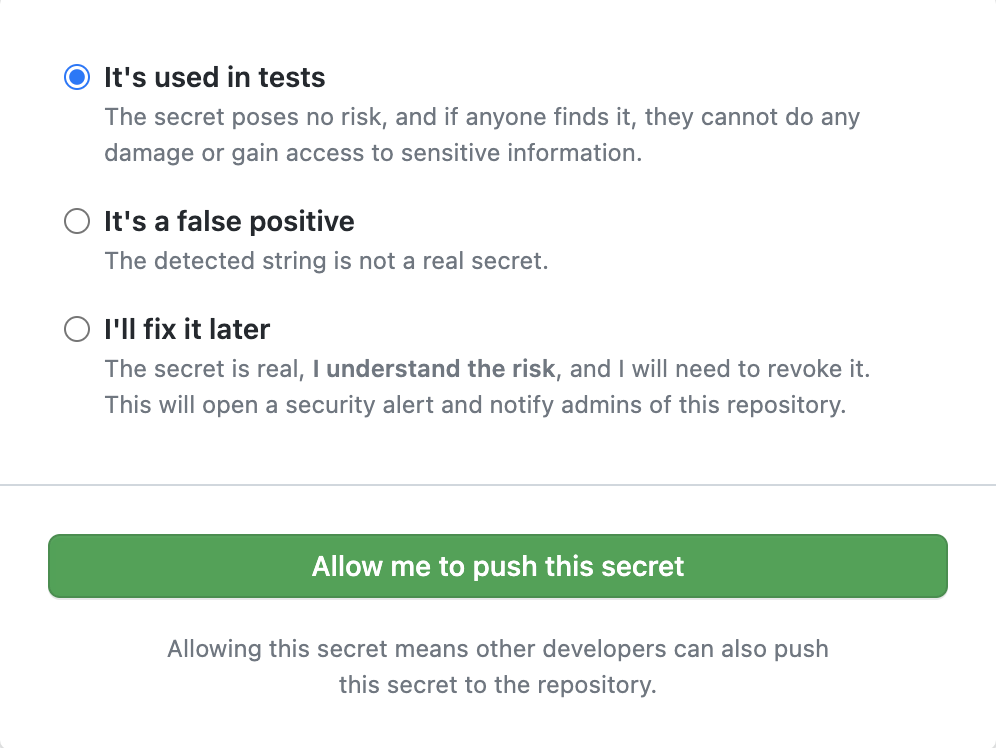
* If your git configuration supports pushes to multiple branches, and not only to the current branch, your push may be blocked due to additional and unintended refs being pushed. For more information, see the [push.default options](https://git-scm.com/docs/git-config#Documentation/git-config.txt-pushdefault) in the Git documentation.
* If secret scanning upon a push times out, GitHub will still scan your commits for secrets after the push.

### **Allowing a blocked secret to be pushed**

If GitHub blocks a secret that you believe is safe to push, you can allow the secret and specify the reason why it should be allowed.

When you allow a secret to be pushed, an alert is created in the "Security" tab. GitHub closes the alert and doesn't send a notification if you specify that the secret is a false positive or used only in tests. If you specify that the secret is real and that you will fix it later, GitHub keeps the security alert open and sends notifications to the author of the commit, as well as to repository administrators. For more information, see "[Managing alerts from secret scanning](https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/managing-alerts-from-secret-scanning)."

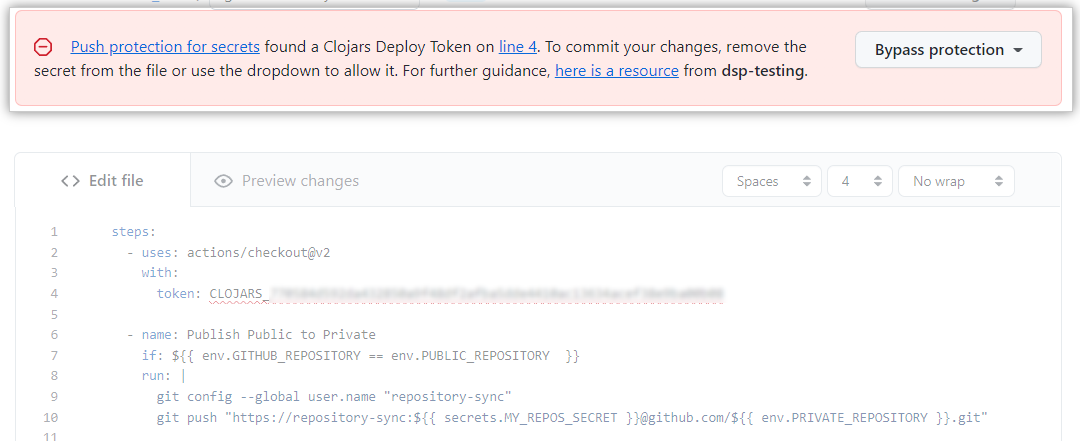
When a contributor bypasses a push protection block for a secret, GitHub also sends an email alert to the organization owners, security managers, and repository administrators who have opted in for email notifications.

* Visit the URL returned by GitHub when your push was blocked.
* Choose the option that best describes why you should be able to push the secret.
  + If the secret is only used in tests and poses no threat, click **It's used in tests**.
  + If the detected string is not a secret, click **It's a false positive**.
  + If the secret is real but you intend to fix it later, click **I'll fix it later**.
* Click **Allow me to push this secret**.
* Reattempt the push on the command line within three hours. If you have not pushed within three hours, you will need to repeat this process.

## **Using secret scanning as a push protection from the web UI**

When you use the web UI to attempt to commit a supported secret to a repository or organization with secret scanning as a push protection enabled, GitHub will block the commit.

You will see a banner at the top of the page with information about the secret's location, and the secret will also be underlined in the file so you can easily find it.



GitHub will only display one detected secret at a time in the web UI. If a particular secret has already been detected in the repository and an alert already exists, GitHub will not block that secret.

Organization admins can provide a custom link that will be displayed when a push is blocked. This custom link can contain resources and advice specific to your organization. For example, the custom link can point to a README file with information about the organization's secret vault, which teams and individuals to escalate questions to, or the organization's approved policy for working with secrets and rewriting commit history.

**Note:** The ability to add resource links to blocked push messages is currently in public beta and subject to change.

You can remove the secret from the file using the web UI. Once you remove the secret, the banner at the top of the page will change and tell you that you can now commit your changes.



### **Bypassing push protection for a secret**

If you confirm a secret is real, you need to remove the secret from your branch, *from all the commits it appears in*, before pushing again. For more information about remediating blocked secrets, see "[Pushing a branch blocked by push protection](https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/pushing-a-branch-blocked-by-push-protection#resolving-a-blocked-push-in-the-web-ui)."

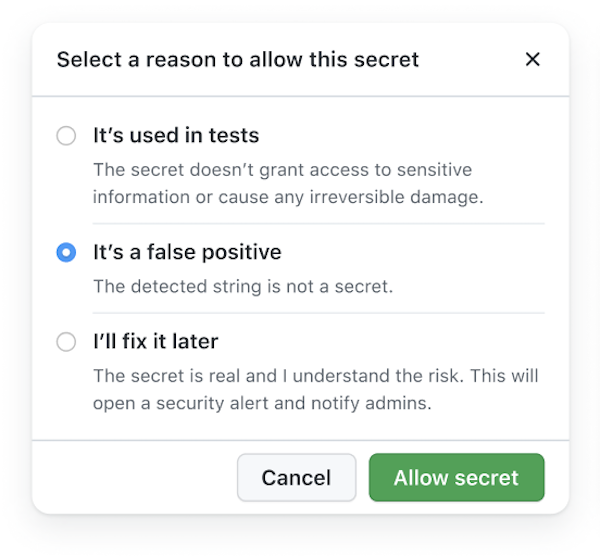
If you confirm a secret is real and that you intend to fix it later, you should aim to remediate the secret as soon as possible. For more information, see "[Removing sensitive data from a repository](https://docs.github.com/en/enterprise-cloud@latest/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository)."

If GitHub blocks a secret that you believe is safe to push, you can allow the secret and specify the reason why it should be allowed.

When you allow a secret to be pushed, an alert is created in the "Security" tab. GitHub closes the alert and doesn't send a notification if you specify that the secret is a false positive or used only in tests. If you specify that the secret is real and that you will fix it later, GitHub keeps the security alert open and sends notifications to the author of the commit, as well as to repository administrators. For more information, see "[Managing alerts from secret scanning](https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/managing-alerts-from-secret-scanning)."

When a contributor bypasses a push protection block for a secret, GitHub also sends an email alert to the organization owners, security managers, and repository administrators who have opted in for email notifications.

If you confirm a secret is real and that you intend to fix it later, you should aim to remediate the secret as soon as possible.

* In the banner that appeared at the top of the page when GitHub blocked your commit, click **Bypass protection**.
* Choose the option that best describes why you should be able to push the secret.
  + If the secret is only used in tests and poses no threat, click **It's used in tests**.
  + If the detected string is not a secret, click **It's a false positive**.
  + If the secret is real but you intend to fix it later, click **I'll fix it later**.
* 
* Click **Allow secret**.

CodeQL (Code Scanning):

CodeQL is the code analysis engine developed by GitHub to automate security checks. CodeQL treats code like database and runs CodeQL queries on the database to identify problems in the database. The query results will be shown as code scanning alerts in the Security tab of the repository.

CodeQL supports both compiled and interpreted languages and can find vulnerabilities and errors in code that's written in the supported languages.

* C/C++
* C#
* Go
* **Java**
* **JavaScript/TypeScript**
* Python
* Ruby

For complete list of supported languages and versions refer <https://codeql.github.com/docs/codeql-overview/supported-languages-and-frameworks/.>

There are two main ways to use CodeQL analysis for code scanning:

* Add the CodeQL workflow to your repository. This uses the [github/codeql-action](https://github.com/github/codeql-action/) to run the CodeQL CLI. For more information, see "[Setting up code scanning for a repository](https://docs.github.com/en/code-security/secure-coding/automatically-scanning-your-code-for-vulnerabilities-and-errors/setting-up-code-scanning-for-a-repository#setting-up-code-scanning-using-actions)."
* Run the CodeQL CLI directly in an external CI system and upload the results to GitHub. For more information, see "[About CodeQL code scanning in your CI system](https://docs.github.com/en/code-security/secure-coding/using-codeql-code-scanning-with-your-existing-ci-system/about-codeql-code-scanning-in-your-ci-system) .

In TFA we are going to use the action github/codeql-action as TFA is using the GitHub itself as orchestrator.