

# GUIA PASO 1

En este paso vamos a obtener 4 valores de las variables secretas que vamos a crear en nuestro Git.

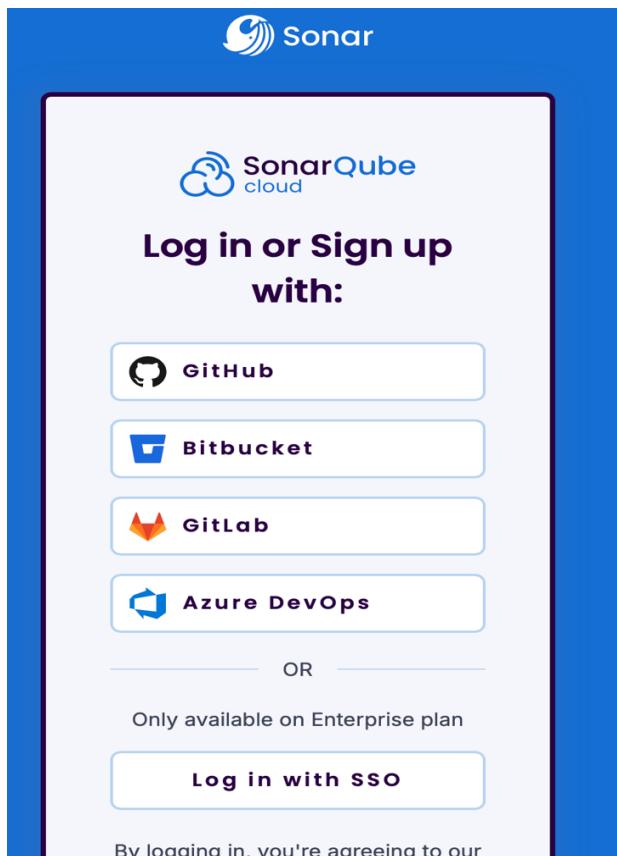
SONAR\_HOST\_URL

SONAR\_ORGANIZATION

SONAR\_PROJECT\_KEY

SONAR\_TOKEN

1. Accedemos al Sonar iniciando sesión desde la cuenta donde vamos a tener nuestro repositorio. Iniciamos sesión con uno de nuestros GIT.



2. Si no tenemos el proyecto creado todavía, tenemos que dar en el + que nos aparece en la esquina superior derecha y elegimos la opción “analyze new project”. Dentro de esa opción elegimos nuestro repositorio que queremos analizar. Aquí vemos ya el valor de nuestra variable “SONAR\_ORGANIZATION”.

## Analyze projects

Select repositories from one of your GitHub organization.

### Organization

davidtome97

[Import another organization](#)

Select all on this page

Search for repositories

 [tfg-cicd-aws-2526](#)

✓ Already imported

 [refactoring-fowler-example](#)

 [Practica3\\_PCTR\\_DependenciasEstados](#)

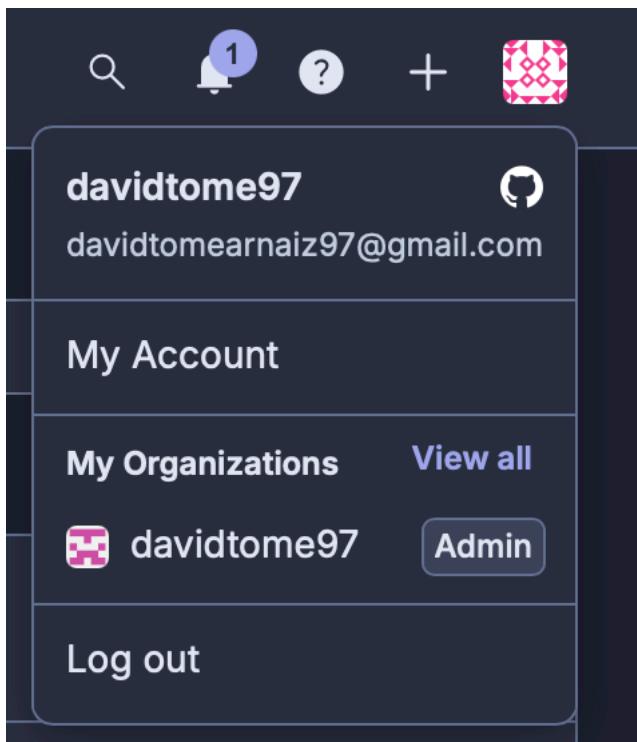
 [ramas](#)

✓ Already imported

 [ActividadRepositorio](#)

 [ejemploArray](#)

3. En la esquina superior derecha, si damos sobre nuestra foto de perfil, podremos ver una opción que pone: “View all”, pulsamos en el y ahí nos aseguramos de ver nuestro valor de la variable “SONAR\_ORGANIZATION”.



4. Esa de ahí abajo sería nuestra organización.

A screenshot of the application's main interface, showing the "Organizations" tab selected in the navigation bar. The page title is "My Account / Organizations". Below the title, the word "Organizations" is centered. A red arrow points to the organization entry "davidtome97-1", which is highlighted with a red border. To the right of this entry are three buttons: "Upgrade", "Delete", and "Leave".

- Una vez que ya tenemos nuestro proyecto y nuestra organización, vamos a ver el valor de nuestra variable “SONAR\_PROJECT\_KEY”. Para ello damos en nuestro nombre del proyecto, en este caso “TFG CI/CD AWS 25/26”.



- Dentro de nuestro proyecto, en la esquina inferior izquierda pulsamos en Administration → Update Key. Se nos abrirá una ventana donde aparece nuestro valor de “SONAR\_PROJECT\_KEY”.

**Update Key**

Edit the key of a project.

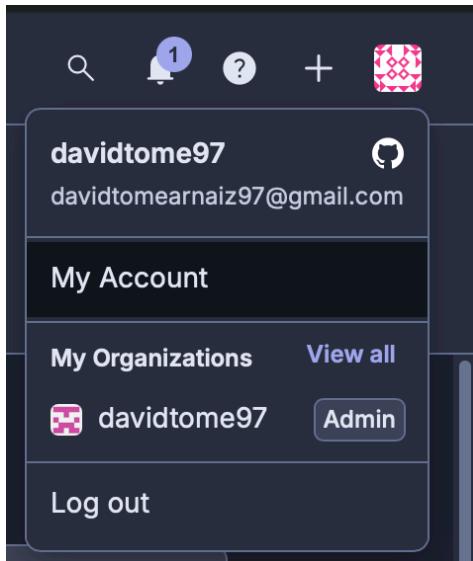
**Project Key** ?

davidtome97\_tfg-cicd-aws-2526

Up to 400 characters. All letters, digits, dash, underscore, period or colon.

**Update**   **Reset**

- Ya solo nos falta crear el token de Sonar. Para ello, desde el inicio de sonar. Si damos a nuestra foto de perfil (parte superior derecha) → My Account



8. Se nos abre una ventana con diferentes pestañas, pulsamos en la de “Security”. Escribimos un nombre en el campo “generate tokens” y damos al botón de generar. Copia ese token que te va a dar ¡¡OJO, ESE TOKEN SOLO LO VAS A VER UNA VEZ!! Si se te olvida, tendrás que generar otro nuevo. Aquí ya tendríamos el valor de nuestra variable “SONAR\_TOKEN”

A screenshot of the SonarCloud Security page. The top navigation bar shows the user profile 'davidtome97' and links for Profile, Security, Notifications, Organizations, Enterprises, and Appearance. The 'Security' tab is active. Below the navigation is a breadcrumb trail 'My Account / Security'. The main section is titled 'Security' and contains a paragraph explaining the purpose of User Tokens. At the bottom, there is a 'Generate Tokens' section with a form to 'Enter Token Name' and a 'Generate Token' button.

9. Por defecto, el valor de nuestra variable “SONAR\_HOST\_URL” es “<https://sonarcloud.io>”

Con esto dariamos como finalizado la obtención de nuestros 4 valores que vamos a crear ahora en Git:

SONAR\_ORGANIZATION (paso 2-3)

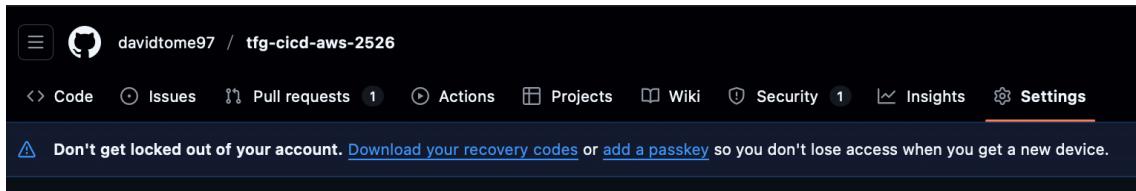
SONAR\_PROJECT\_KEY (paso 5)

SONAR\_TOKEN (paso 7)

SONAR\_HOST\_URL (paso 8)

## ¿Cómo creamos nuestras variables secretas en nuestro github?

1. Iniciamos sesión en nuestro github → accedemos a nuestro repositorio con el que queremos trabajar → settings.



2. En el menú lateral, vamos hasta la opción de security → secrets and variables → actions. Ahí en la pestaña de “secrets”, damos al botón “New repository secret”.

A screenshot of the 'Repository secrets' section within the GitHub repository settings. The left sidebar shows the 'Secrets and variables' option under the 'Actions' category. The main area displays a table of existing secrets, each with a preview icon, name, last updated time, and edit/delete buttons. A green button labeled 'New repository secret' is visible at the top right of the table. The table data is as follows:

Name	Last updated	Actions
AWS_ACCESS_KEY_ID	2 months ago	
AWS_ECR_URL	2 weeks ago	
AWS_REGION	2 months ago	
AWS_SECRET_ACCESS_KEY	2 months ago	

3. Escribimos el nombre de las variables que te he proporcionado y el valor de cada una de ellas que hemos obtenido antes y damos a “add secret”.  
¡¡MUY IMPORTANTE, EL NOMBRE TIENE QUE SER EL MISMO QUE TE HE PROPORCIONADO AL PRINCIPIO DEL MENÚ!!

**Actions secrets / New secret**

---

**Name \***

**Secret \***

**Add secret**

Una vez que creamos todas las variables de este paso hay que hacer un commit para que coja los cambios realizados y no nos falle el siguiente paso:

```
git commit --allow-empty -m "creación de variables"  
git push
```

## ¿Cómo creamos nuestras variables secretas en nuestro gitlab?

1. Iniciamos sesión en nuestro gitlab → projects → nuestro proyecto.

The screenshot shows the GitLab web interface. On the left, there's a dark sidebar titled 'Your work' with various navigation options: Home, Projects (which is selected and highlighted in blue), Groups, Issues, Merge requests (3), To-Do List (3), Milestones, Snippets, Activity, Import history, Workspaces, Environments, Operations, and Security. At the top right, there's a search bar with placeholder text 'Search or go to...'. The main area is titled 'Projects' and shows a summary: Contributed (1), Starred (0), Personal (1), Member (1), and Inactive (0). Below this is a search/filter bar with a dropdown and a placeholder 'Filter or search (3 character minimum)'. At the bottom, it shows the project details: 'davidtome97 / tfg-cicd-aws-2526' (Owner), 'TFG - Automatización CI/CD en AWS con generación automática de workflows', and a commit count of '179 Commits'.

2. En nuestro menú lateral, seleccionamos Settings → CI/CD. Se nos abrirá una nueva ventana.

The screenshot shows the 'Settings' menu for the 'tfg-cicd-aws-2526' project. The 'CI/CD' tab is selected and highlighted in blue. The left sidebar lists other settings tabs: General, Integrations, Webhooks, Access tokens, Repository, Merge requests, CI/CD (selected), Packages and registries, Monitor, and Usage quotas. The main content area displays 'Project information' for 'tfg-cicd-aws-2526' (TFG - Automatización CI/CD en AWS con generación automática de workflows), a commit count of '179 Commits', a 'README' file, and a note that it was 'Created on November 01, 2023'. A 'develop' branch is also visible.

- Desplegamos la opción “Variables” y vemos una opción que pone “add variable”.

A job artifact is an archive of files and directories saved by a job when it finishes.

### Variables

Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more](#).

**Minimum role to use pipeline variables**

Select the minimum role that is allowed to run a new pipeline with pipeline variables. [What are pipeline variables?](#)

- No one allowed Pipeline variables cannot be used.
- Owner
- Maintainer
- Developer

**Access protected resources in merge request pipelines**

Make protected CI/CD variables and runners available in merge request pipelines. Protected resources will only be available in merge request pipelines if both the source and target branches of the merge request are protected. [Learn more](#).

Allow merge request pipelines to access protected variables and runners

**Display manually-defined pipeline variables**

Display all manually-defined variables in the pipeline details page after running a pipeline manually. [Learn more](#).

Display pipeline variables

All manually-defined CI/CD variables and their values are visible to maintainers, which is a security risk if including credentials or other secrets in variables. Do not enable this feature if variables could contain sensitive data. Developers can only view manually-defined variables in their own manual pipelines.

**Project variables**

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

**CI/CD Variables** </> 20 Reveal values Add variable

- Elegimos el tipo que queremos, la visibility que queremos para esta variable, en “key” ponemos el nombre de la variable y en “value” su valor que hemos obtenido anteriormente.
- ¡¡MUY IMPORTANTE, EL NOMBRE TIENE QUE SER EL MISMO QUE TE HE PROPORCIONADO AL PRINCIPIO DEL MENÚ!!**

Type

Variable (default)

Environments

All (default)

Visibility

Visible  
Can be seen in job logs.

Masked  
Masked in job logs but value can be revealed in CI/CD settings. Requires values to meet regular expressions requirements.

Masked and hidden  
Masked in job logs, and can never be revealed in the CI/CD settings after the variable is saved.

Flags

Protect variable  
Export variable to pipelines running on protected branches and protected tags only.

Expand variable reference  
\$ will be treated as the start of a reference to another variable.

Description (optional)

The description of the variable's value or usage.

Key

variable

You can use CI/CD variables with the same name in different places, but the variables might overwrite each other. [What is the order of precedence for variables?](#)

Value

variable\_value

Una vez que creamos todas las variables de este paso hay que hacer un commit para que coja los cambios realizados y no nos falle el siguiente paso:

```
git commit --allow-empty -m "creación de variables CI/CD"
```

```
git push
```

Si nos da error al pasar nuestro proceso de ci/cd, desactivar los análisis automáticos de sonar. Para ello entramos en nuestro proyecto en sonar → en el menú lateral → administration → Analysis Method → y desactivamos la opción “Automatic Analysis”.

The screenshot shows the SonarQube Cloud interface. At the top, there's a navigation bar with the SonarQube logo, 'My Projects', 'My Issues', and 'Explore'. Below this, the user 'davidtome97' is viewing the project 'demo-prueba'. The sidebar on the left lists project details ('Public', 'Main Branch', 'Information', 'Administration') and various settings like 'General Settings', 'Analysis Method' (which is selected and highlighted in blue), 'Quality Profiles', 'Quality Gate', etc. The main content area is titled 'Analysis Method' and contains a sub-section 'Automatic Analysis' with a toggle switch that is currently off. A note below says 'SonarQube Cloud automatically analyzes your default branch and Pull Requests.' To the right, there are sections for 'Set up analysis via other methods' (with options for 'GitHub Actions' and 'CircleCI') and a 'Manually' section with a note about testing.

Volvemos a ejecutar el commit por terminal para ver que realmente pasa bien el sonar:

```
git commit --allow-empty -m "creación de variables CI/CD"
```

```
git push
```

Deberíamos ver algo así:

