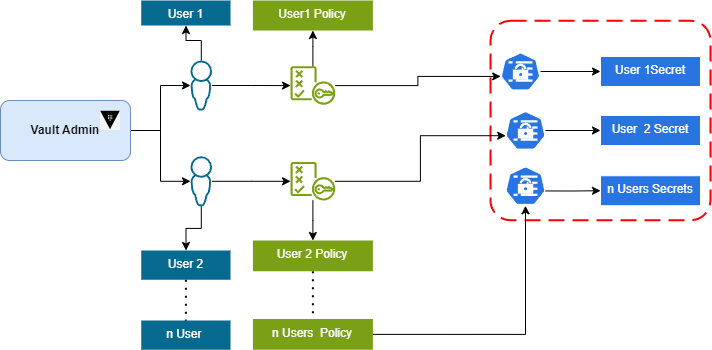
**GCP Easy Dashboard**

**Vault Admin Guide: -**

**Architecture for vault setup:**

****

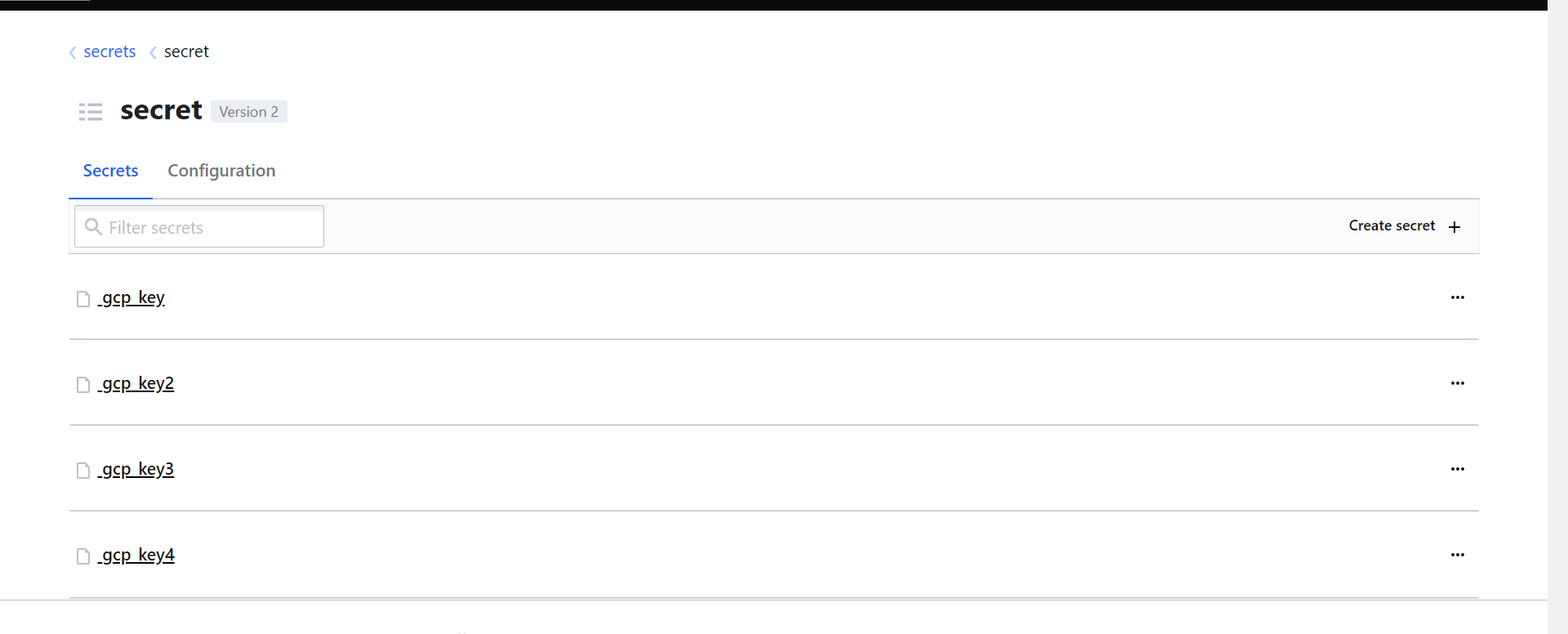
**On boarding a user to the dashboard: -**

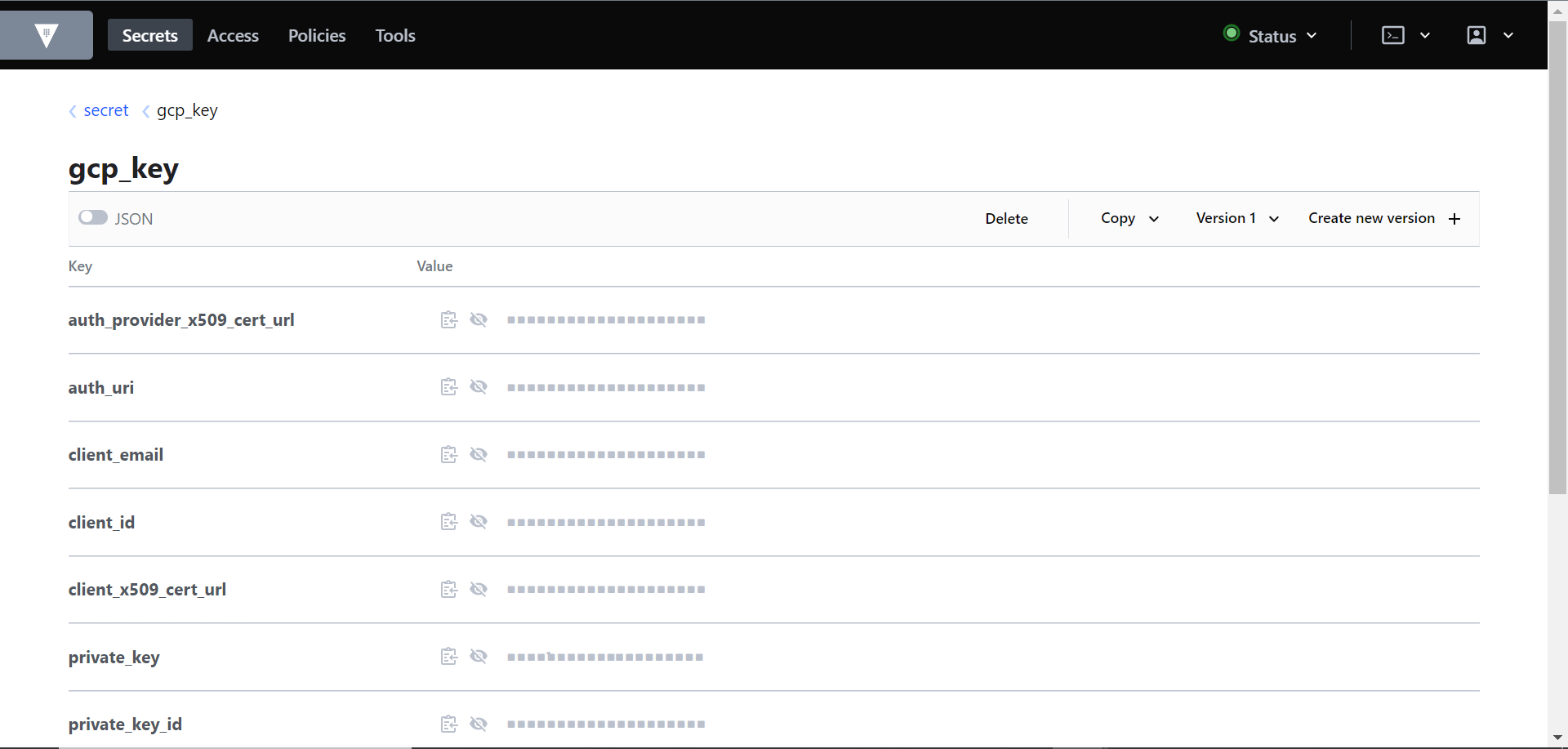
**Step 1:** User who wants to get access to the dashboard needs to provide their gcp service account to the vault Admin.

**Step 2:** Vault admin will take user service account file and create a secret in the vault with it, using a vault secret path, and admin will create authentication method using password and username further admin creates a policy to bind the username and password with respective secret of that user.

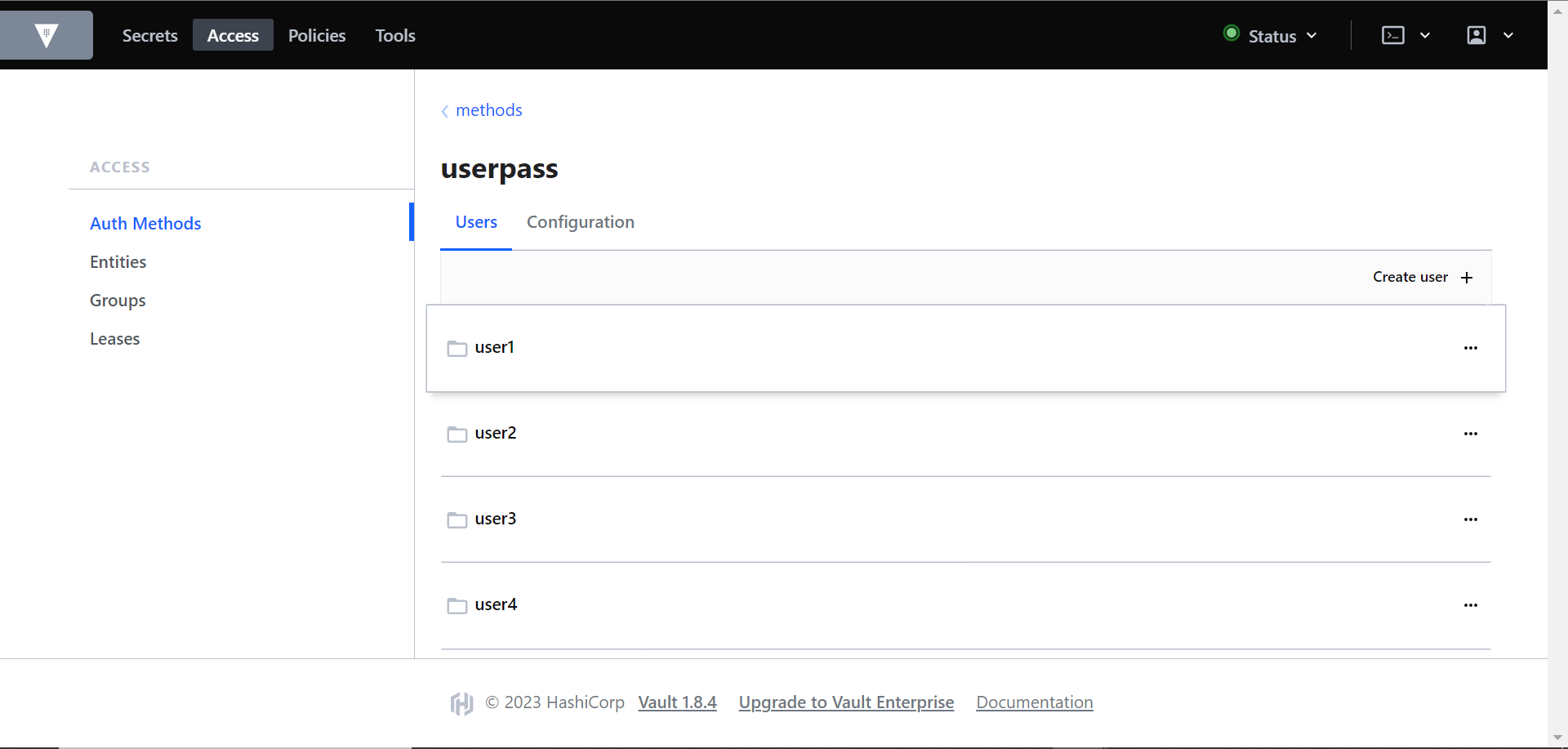
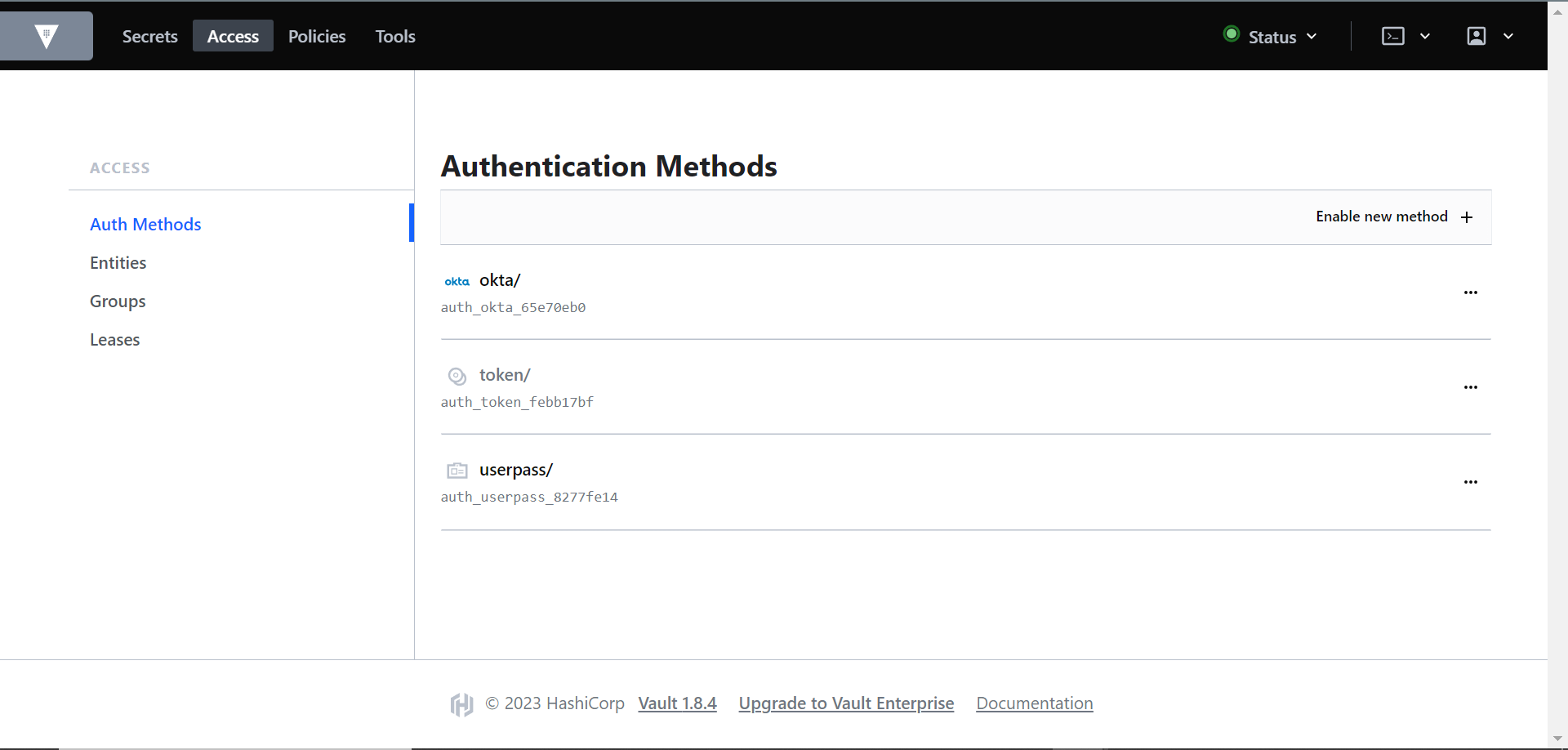
**Step 3:** Admin will send the vault username, vault password and secret path to the requested user **.**

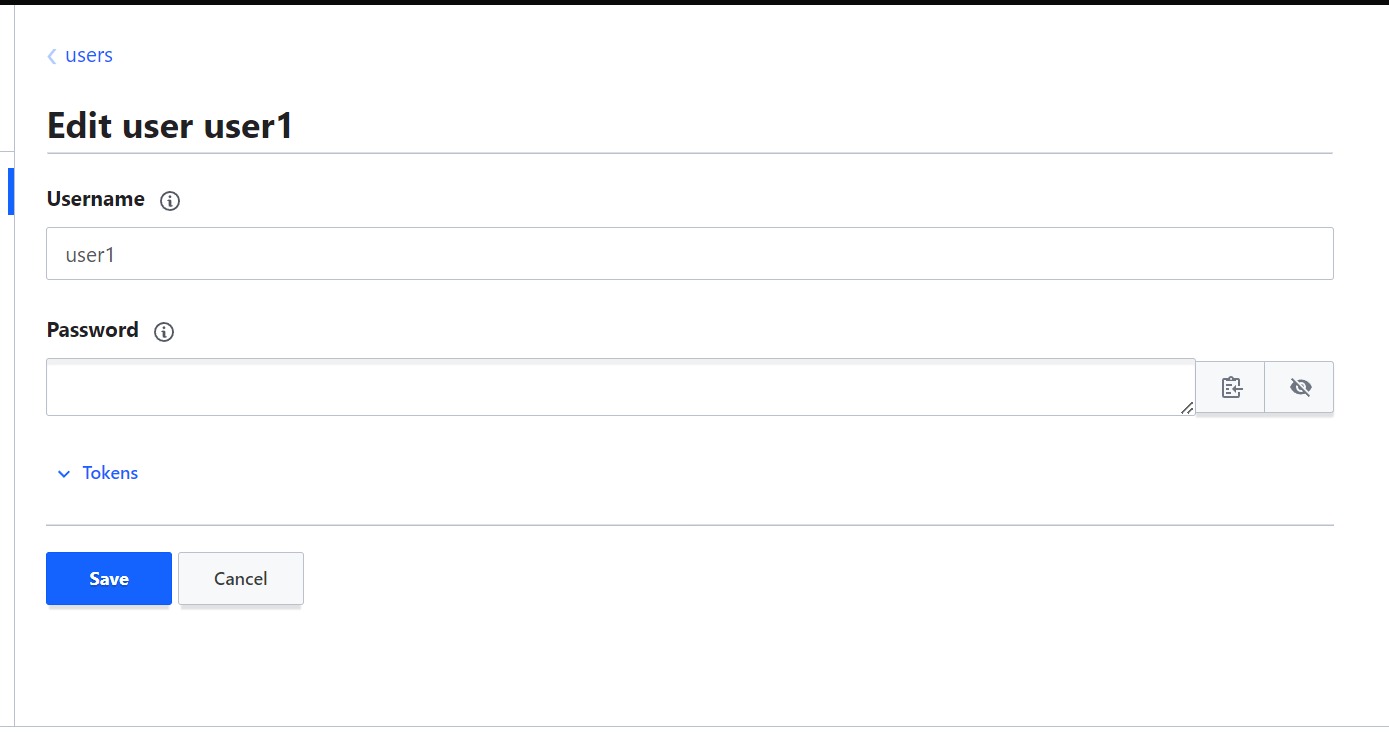
**On boarding user details:**

**Step 1: -** create secret with service account json(format).

****

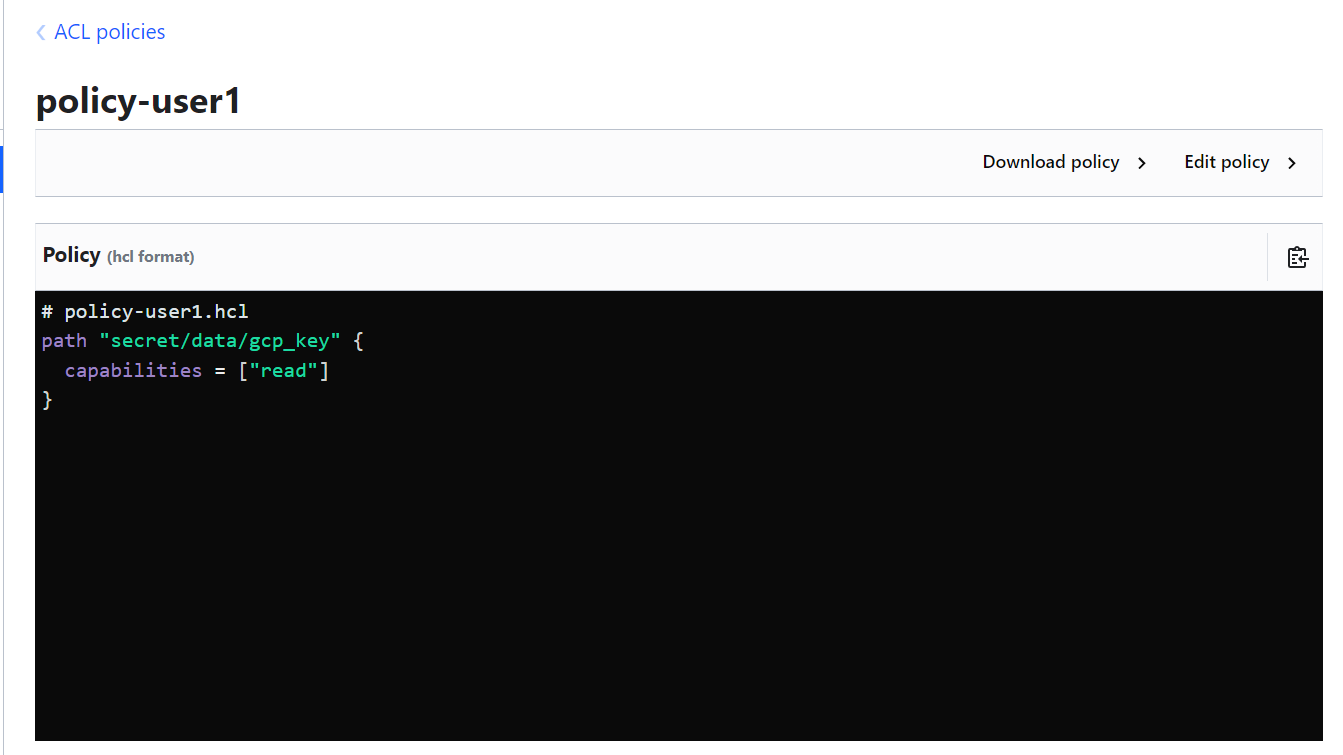
**Step 2:-** Create authentication method for username and password then create user details**.**

****

****

**Step 3:** Create the policy for the user to restrict their access only to their secret.

****

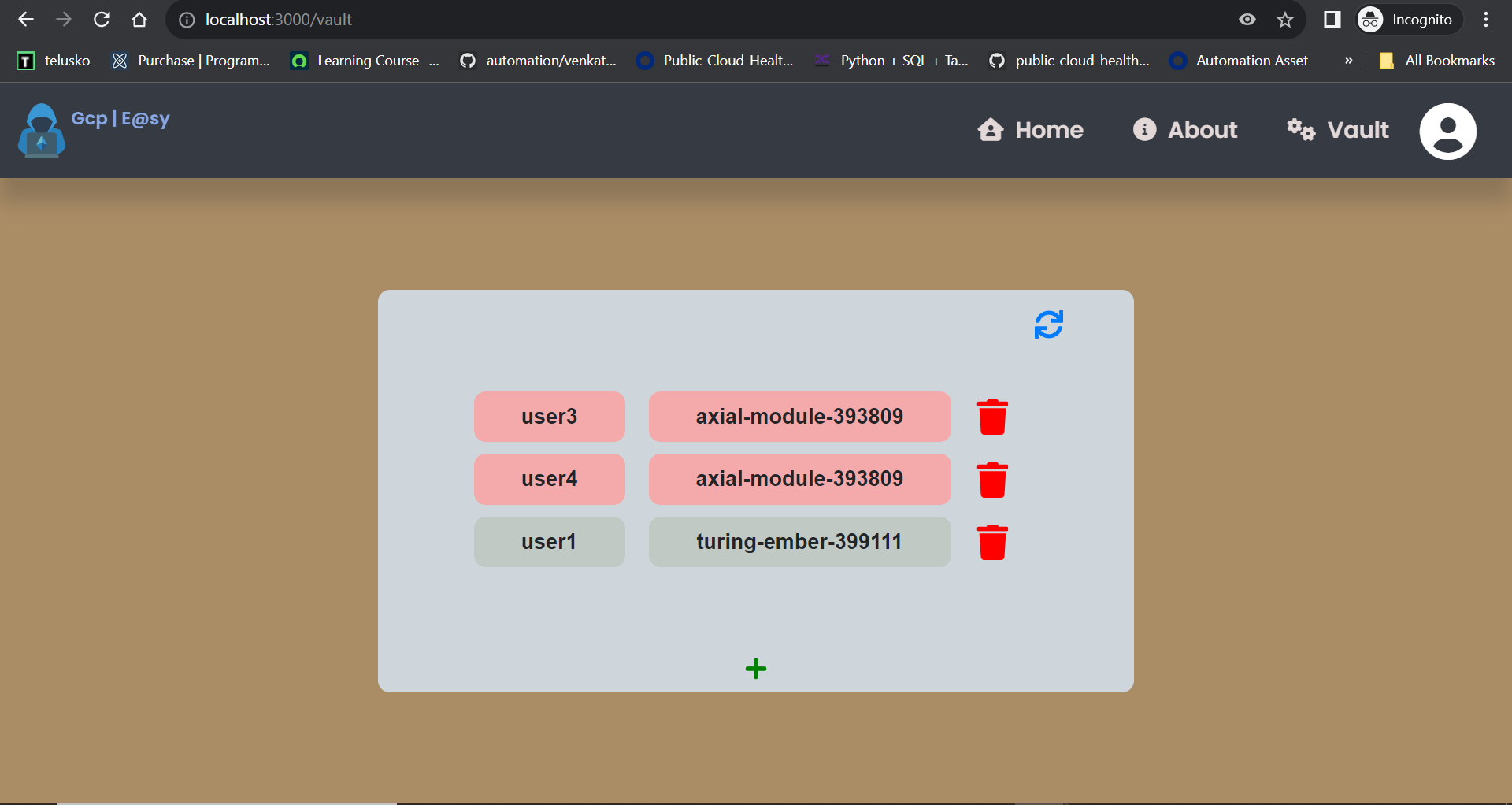
****

**vault write auth/userpass/users/<username> password="<userpassword>" policies="<user policy>"**

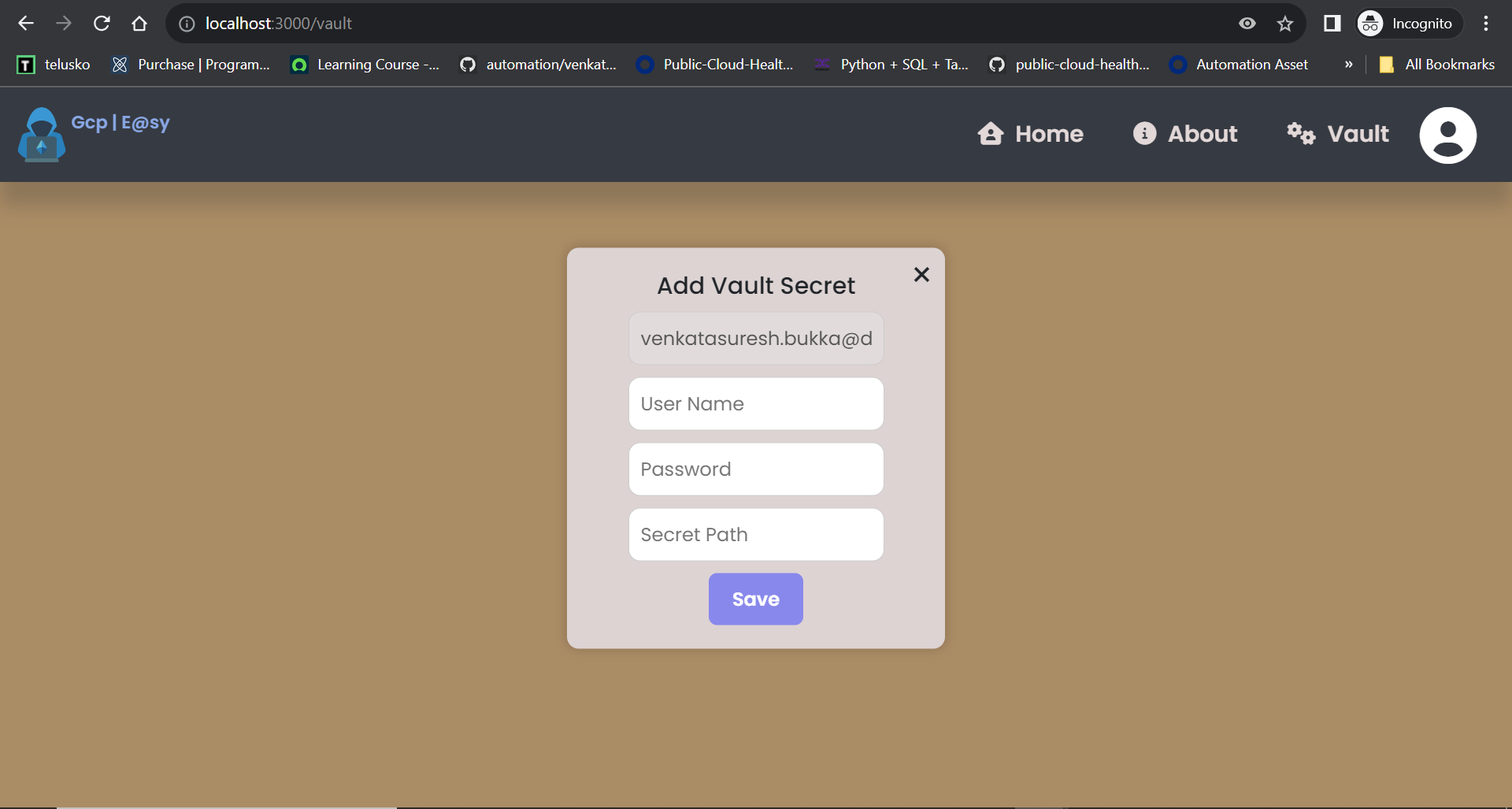
After creating username, password and policy for the requested user then execute the above command to bind the user and user policy, make sure to replace them with actual username, password, and policy for the user.

**Users Guide:**

* After user got their username, password and secret path from vault admin user can now login to the dashboard via dxc azure ad+okta authentication and the should navigate to vault page to get started.

****

* Users now can add their vault credentials in the dashboard which will be stored in a private DB the details are vault username, vault password and secret path.

****

* They can add the details using the above form and in the backend code it will have check if the username , password and secret key are valid from vault or not if they matched then it will store them in db and fetches the respective project id which is in the secret in vault, then upon successful save user will their project name and vault user in the page, like wise they can add other vault details which will contain different projects.

**Note**: The db will only store user dxc email, vault username, vault password and secret path, users can even delete their vault details any time they want.

**Dev list**

* disks not attached to a vm
* monitoring && alerts
* change the state of the compute engine.
* delete un-wanted or un-used resources!
* provide recommendations.
* delete service accounts not used.

**CREATE** **TABLE** `gcp\_dashboard\_app\_vaultdetails` (

`id` **BIGINT**(19) **NOT** **NULL** **AUTO\_INCREMENT**,

`vault\_username` **VARCHAR**(255) **NOT** **NULL** **COLLATE** 'latin1\_swedish\_ci',

`vault\_password` **VARCHAR**(255) **NOT** **NULL** **COLLATE** 'latin1\_swedish\_ci',

`secret\_path` **VARCHAR**(255) **NOT** **NULL** **COLLATE** 'latin1\_swedish\_ci',

`user\_id` **BIGINT**(19) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`) **USING** **BTREE**,

**INDEX** `gcp\_dashboard\_app\_va\_user\_id\_8cc0108d\_fk\_gcp\_dashb` (`user\_id`) **USING** **BTREE**,

**CONSTRAINT** `gcp\_dashboard\_app\_va\_user\_id\_8cc0108d\_fk\_gcp\_dashb` **FOREIGN** **KEY** (`user\_id`) **REFERENCES** `gcp\_dashboard\_app\_user` (`id`) **ON** **UPDATE** **NO** **ACTION** **ON** **DELETE** **NO** **ACTION**

)

**COLLATE**='latin1\_swedish\_ci'

**ENGINE**=**InnoDB**

**AUTO\_INCREMENT**=50

;

**CREATE** **TABLE** `gcp\_dashboard\_app\_user` (

`id` **BIGINT**(19) **NOT** **NULL** **AUTO\_INCREMENT**,

`email` **VARCHAR**(254) **NOT** **NULL** **COLLATE** 'latin1\_swedish\_ci',

**PRIMARY** **KEY** (`id`) **USING** **BTREE**,

**UNIQUE** **INDEX** `email` (`email`) **USING** **BTREE**

)

**COLLATE**='latin1\_swedish\_ci'

**ENGINE**=**InnoDB**

**AUTO\_INCREMENT**=8

;