Cheat sheet

WiFi automation with Ansible and SD-WAN Meraki Cheat Sheet

Cisco Meraki is one of Cisco's enterprise-cloud managed networking solutions, and is popular for managing access points, security appliances, L2 and L3 switches, and more. This cheat sheet shows how to link up Automated NetOps through a GitHub event trigger with Ansible Automation Platform. The procedure shown here lets you manage the Meraki-controlled devices using a modern "infrastructure as code" model.

The example in this cheat sheet configures GitHub to send notifications about events through a Webhook. Typical events in GitHub include configuration changes and pushes to the repository. The Webhook sends an HTTP POST request to Ansible to trigger an Ansible automation job.

If you don't already administer a network with a Meraki account, you can follow the steps in this cheat sheet by signing up for the Cisco Devnet Sandbox Lab for Meraki Small Business, which allows you an eight-hour reservation (session) by default. After the reservation is ready, the sandbox lab gives you administrative access to configure wireless connections and networking for your specific settings.

The cheat sheet goes through the following steps:

1. Reserve a Cisco DevNet sandbox if you do not have your own Meraki environment (optional).
2. Configure access through the Cisco Meraki dashboard, to allow provisioning from Ansible.
3. Configure a GitHub repository with your wireless settings.
4. Configure the Ansible organization, project, credentials, and execution environment.
5. Enable a GitHub trigger to invoke Ansible.
6. Test the system by creating an automatic trigger.

The automated process of handling an event passes through the following components in order:

1. GitHub repo (which sends the trigger through a Webhook)
2. Ansible
3. Meraki Dashboard Controller API
4. Network device (access point)

Reserving a Cisco DevNet sandbox (optional)

For this cheat sheet, we will use a Meraki Small Business sandbox (Figure 1), because it allows some specific provisioning tasks.

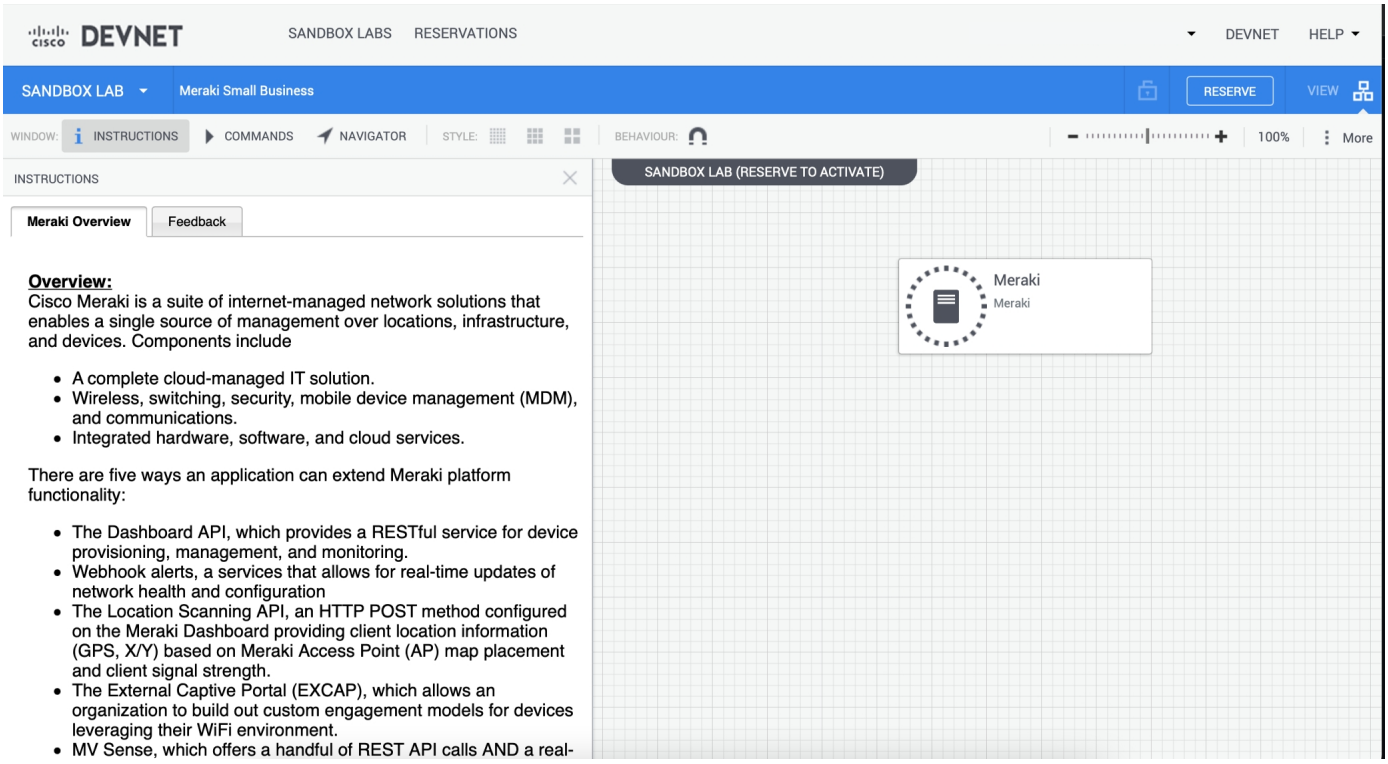

Figure 1: The Meraki Small Business sandbox in the Cisco DevNet web interface.

Once you log in with your Cisco DevNet credentials, click on the **Reserve** button (Figure 2). After you reconfirm the reservation, the sandbox triggers a setup job.

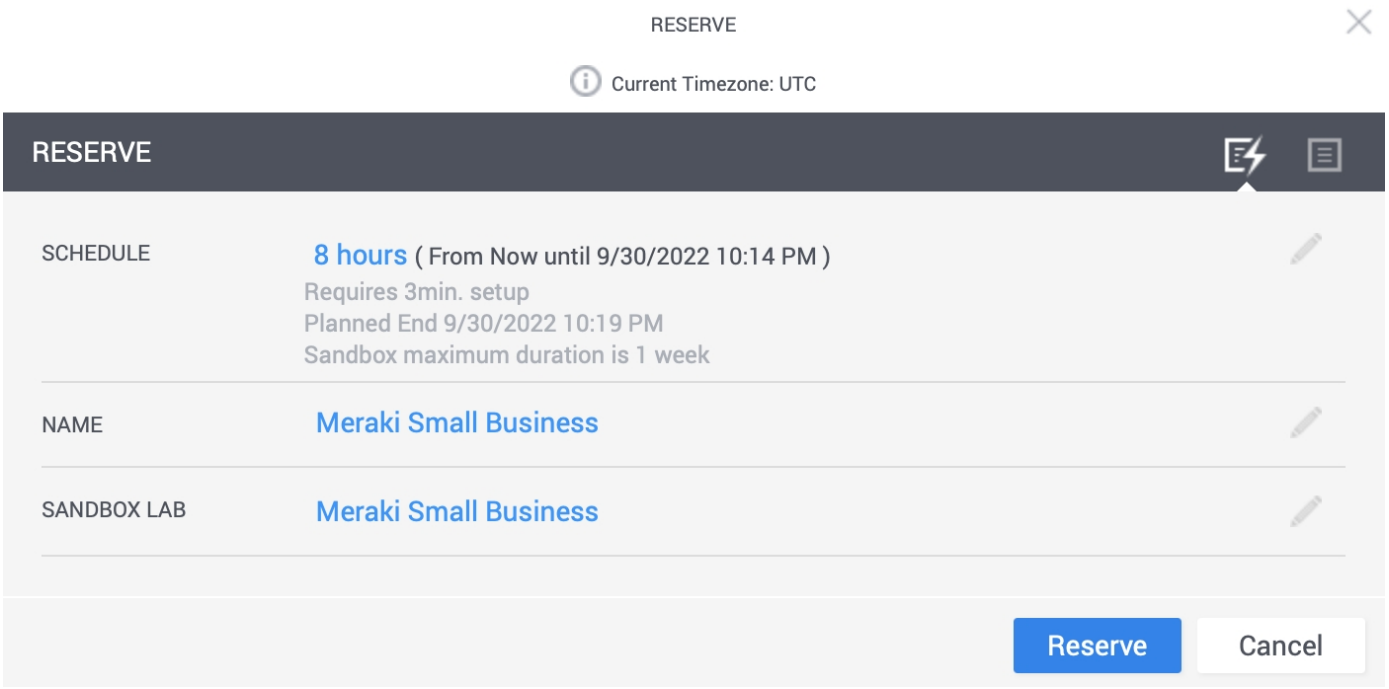

Figure 2: Reserving the Meraki Small Business sandbox.

Wait about five minutes to get a confirmation mail with the credentials you will use to log into the Cisco Meraki dashboard. You might get the username and will be asked to reset your password.

When you connect to the Meraki dashboard, a anonymized network is created with a name based on your email address. In my case, the network name is `DNSMB4-dxxxxxlgmail.com`. This network grants you write access, and should be used throughout this entire demo.

Configuring access through the Cisco Meraki dashboard

In order to grant Ansible access to manage events on your network, you need to give it an API key generated by Meraki. The following subsections explain how to accomplish this and other useful tasks on the dashboard.

Finding your network

The user interface (UI) shows all organizations you're in. The one you're using for this exercise is named **DevNet Sandbox**. For your organization, select **Summary** and scroll down to **Networks** to view all the networks available for your organization. In Figure 3, I have reached the **Networks** page and have entered a few letters into the filter to restrict the networks shown.

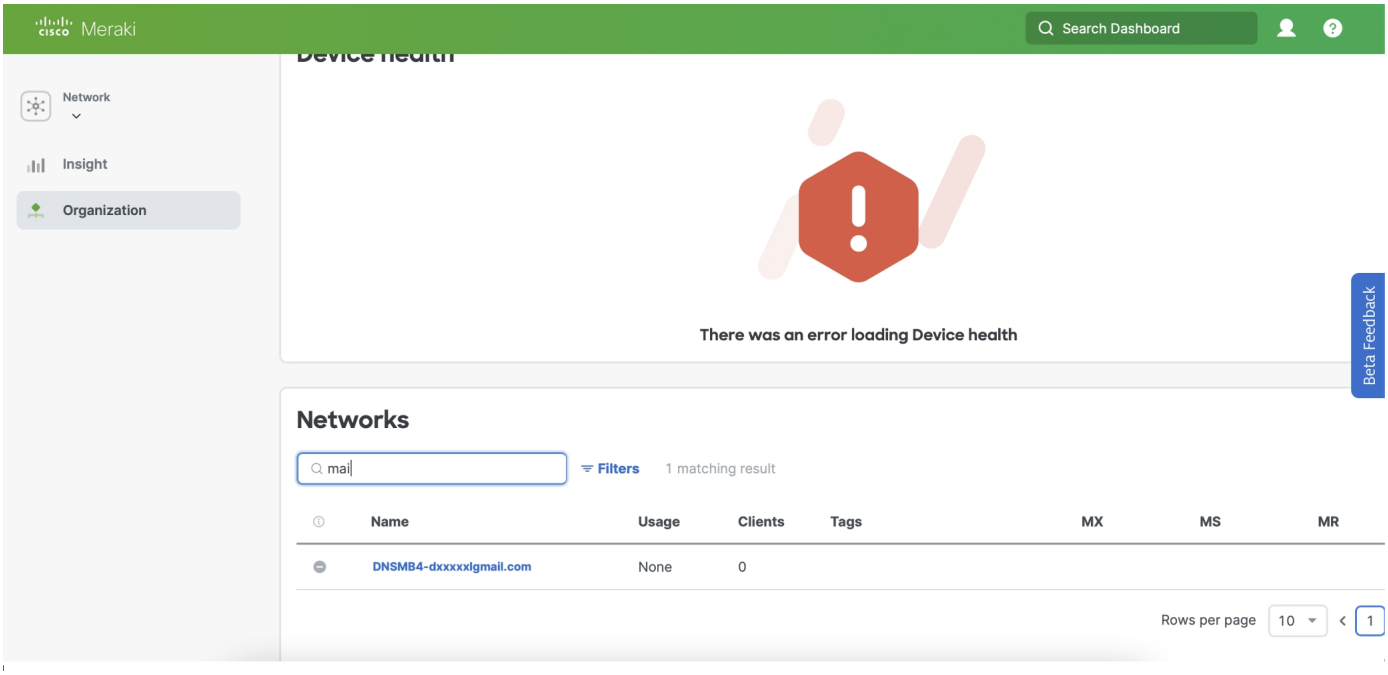

Figure 3: Filter the list of available networks.

Once you find your assigned network, click on it to view its statistics (Figure 4).

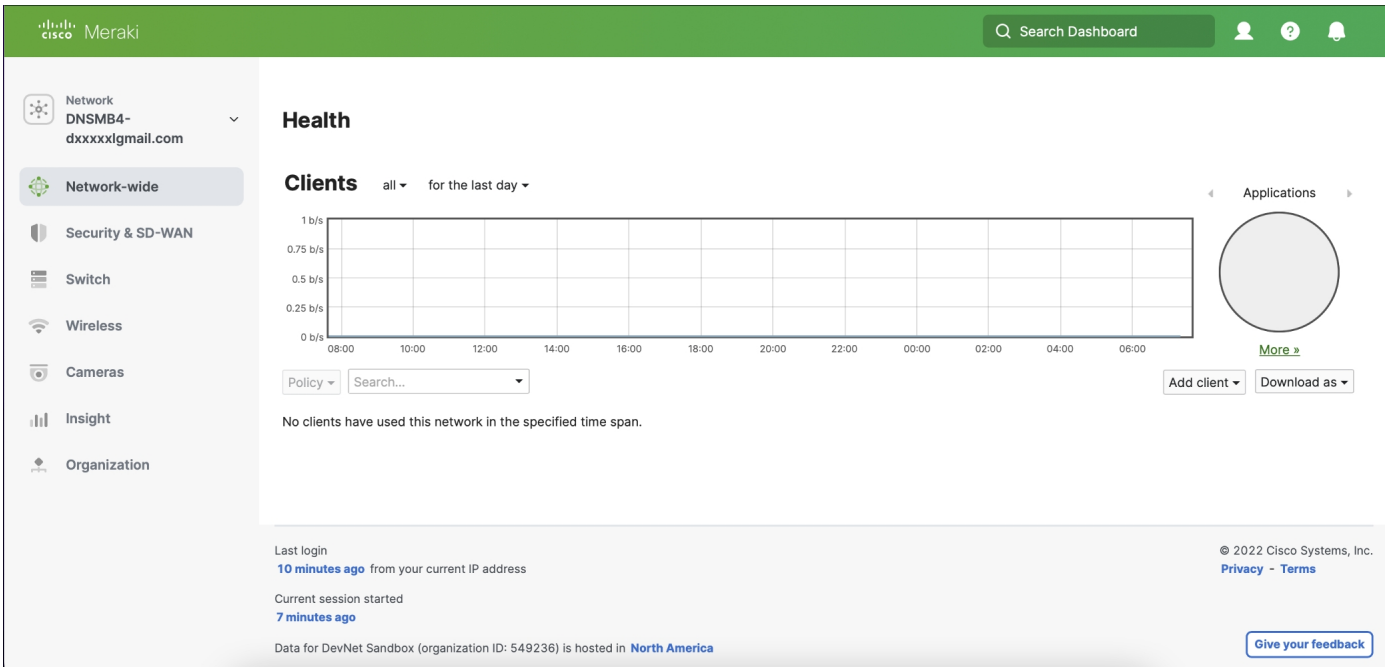

Figure 4: View the network statistics.

Getting access to settings

From the **Organization** page, choose **Configure→Settings** to bring up the page in Figure 5. Near the end of this page you can find the **Dashboard API access** setting.

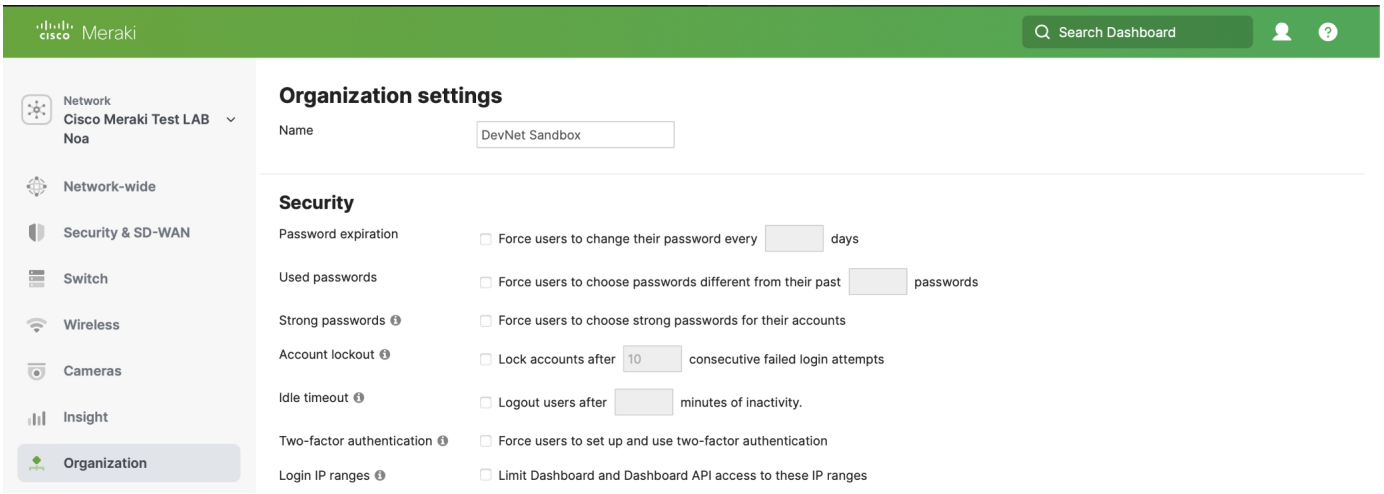

Figure 5: Navigate to the Organization configuration page to locate the Dashboard API access setting.

Getting access to the Meraki Dashboard API

Under the **Dashboard API access** heading, click **Enable access to the Cisco Meraki Dashboard API** (Figure 6).

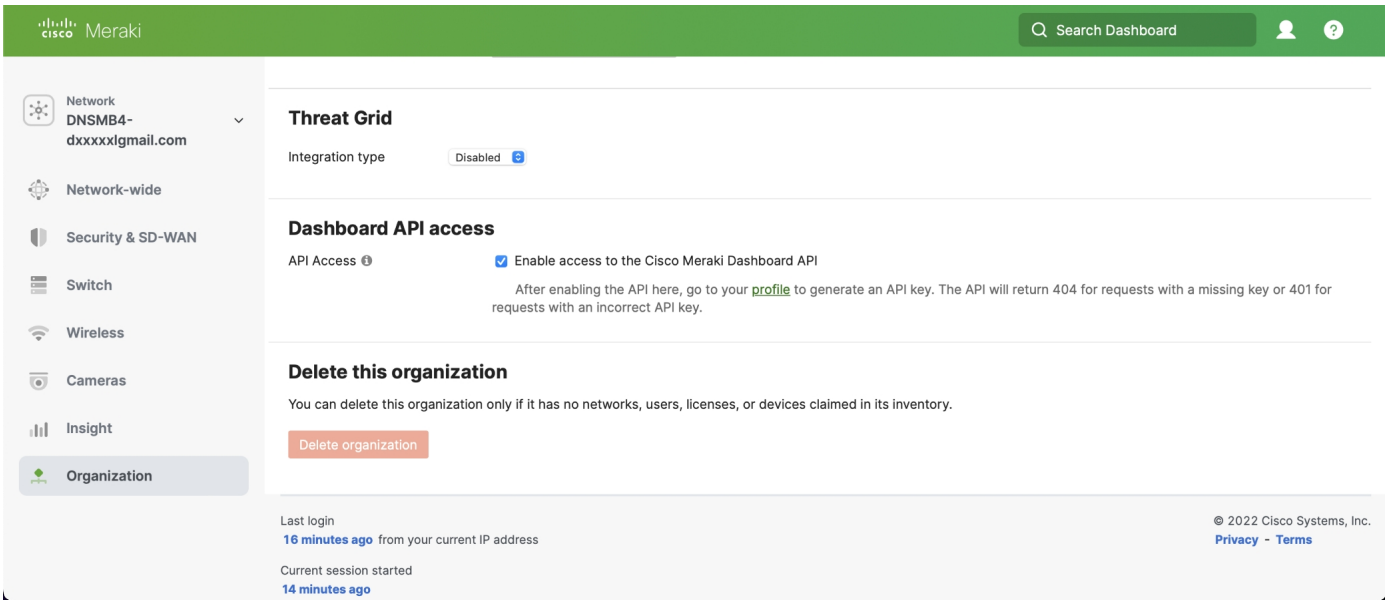

Figure 6: Select the checkbox to enable API access.

Generating the API key

There are two ways to reach this configuration setting:

- Click the Profile link from Organization→Configure→Dashboard API access and go to API access.
- Select the user icon from the top right corner. From there, select **My profile** followed by **API access**. Once on this page, click **Generate new API key** (Figure 7).

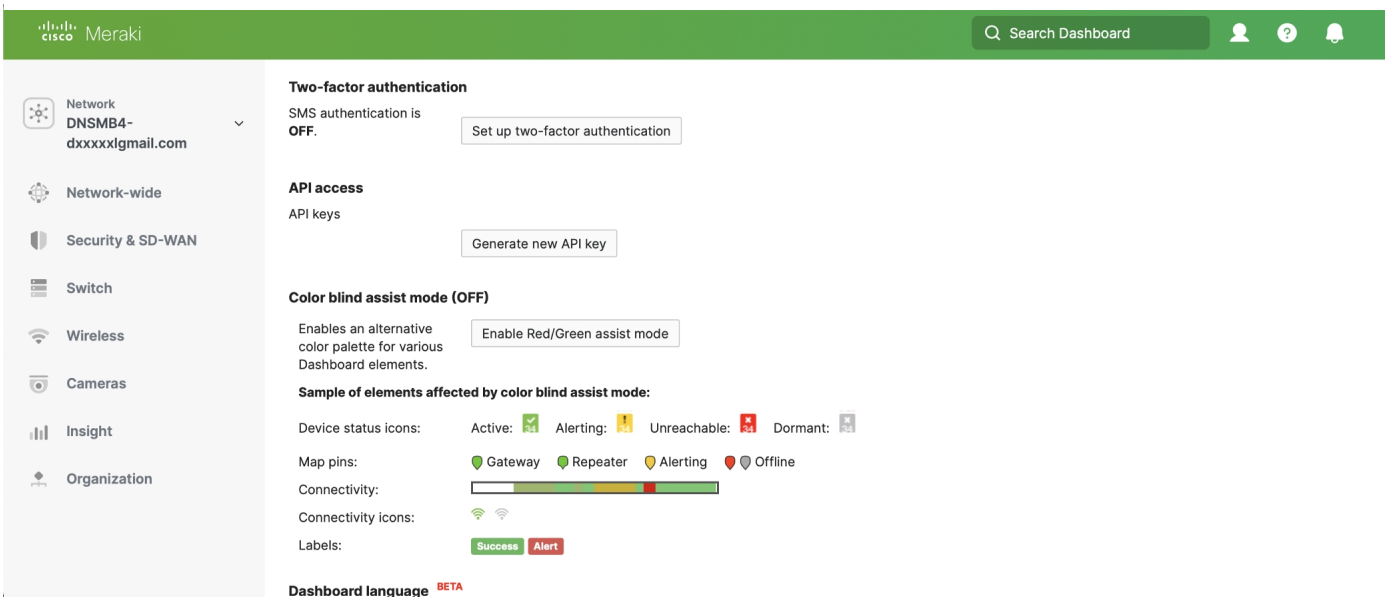

Figure 7: Click the button to generate a new API key.

Make sure you save the API key, because it will be available for copying only once. When you finish, click **Done**.

After you generate at least one API key, you can generate additional ones on the same page, or revoke a key (Figure 8). As a general practice, make sure to revoke API keys if you are not using them.

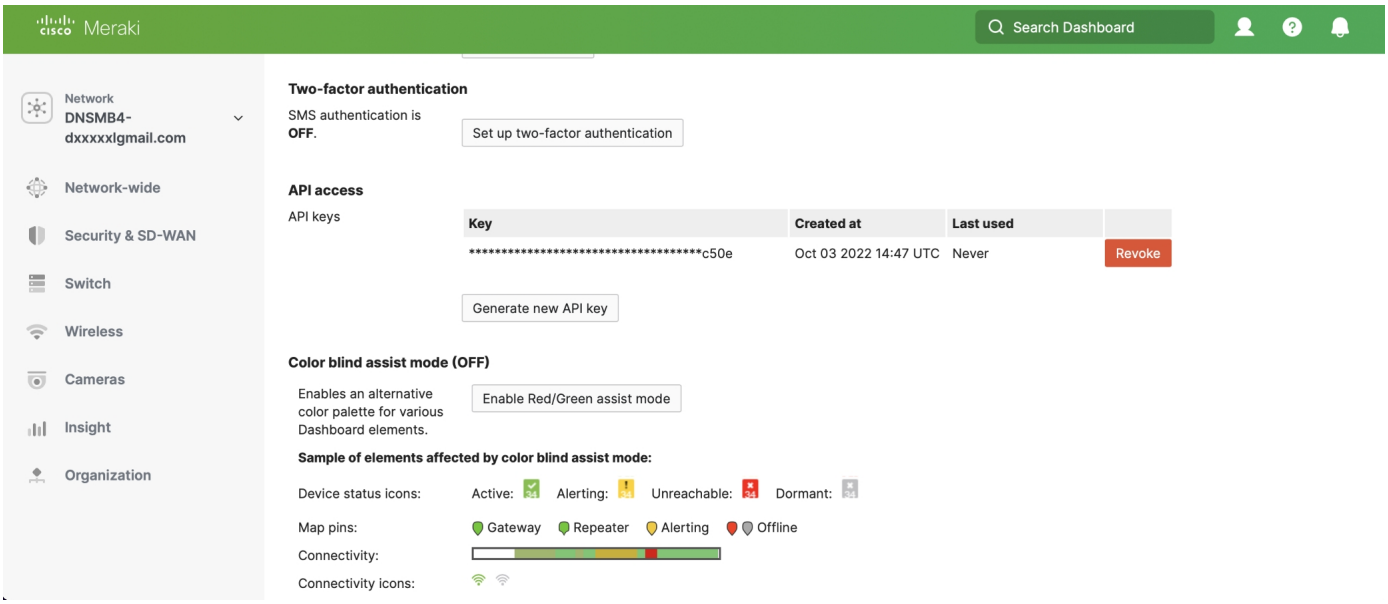

Figure 8: You can generate additional API keys or revoke unused ones.

Configuring wireless network SSIDs

If you go to **Wireless→Configure→SSIDs** you will see that, as part of enabling the sandbox, SSID 1 has already been configured. For instance, in my environment I have one SSID named "DNSMB4 - wireless WiFi." You can manually disable the network, rename it, or edit access controls.

After every change, make sure to click on the **Save Changes** button to preserve the changes, or press **cancel** to discard them.

Configuring a GitHub repository with your wireless settings

The examples in this cheat sheet use this GitHub repository of network tests. Log in to GitHub with your credentials and clone the repository to replicate the integration in this cheat sheet.

Edit your version of the `network_vars.yml` to change the first two properties, giving them the following values:

```
Org_Name: DevNet Sandbox
Net_Name: <Your assigned Network Name from Meraki dashboard>
```

In my repository, these properties look like:

```
Org_Name: DevNet Sandbox
Net_Name: DNSMB4-dxxxxxlgmail.com
```

Also edit the SSIDs in `network_vars.yml`. For my environment, I edited SSIDs 2 and 3 so that the final playbook looked like this:

```
Org_Name: DevNet Sandbox
Net_Name: DNSMB4-dxxxxxlgmail.com
SSID:
  - name: MyCompany_customers
    number: 2
    enabled: no
    auth_mode: psk
    encryption_mode: wpa
    psk: yourcustomerwifipass
  - name: MyCompany_employees
    number: 3
    enabled: yes
    auth_mode: psk
    encryption_mode: wpa
    psk: youremployeeswifipass
```

Configuring the Ansible organization, project, credentials, and execution environment

Ansible needs to run in a project within Meraki. The following subsections show the steps that set up everything Ansible needs.

Configuring your organization

From the **Administration** page, choose **Execution Environments** and set up your execution environment, including the Meraki collection (Figure 9).

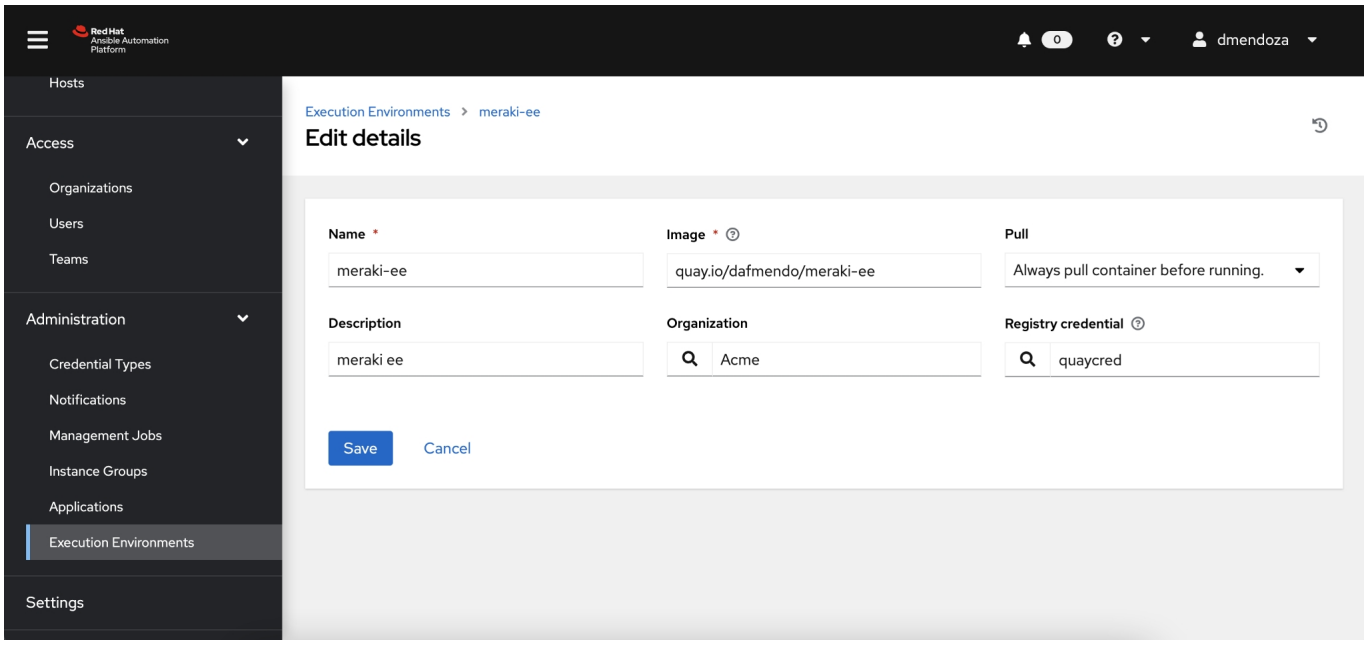

Figure 9: Fill in the fields to configure your execution environment.

In my case, I had previously created a Meraki execution environment following the steps described in this blog and the requirements.yml file had the following:

```
collections:
-ansible.units
-cisco.meraki
```

Once I had the execution environment ready, I uploaded it into Quay to reuse for all my demos using Cisco Meraki sandboxes. From the Access page, choose Organization (Figure 10).

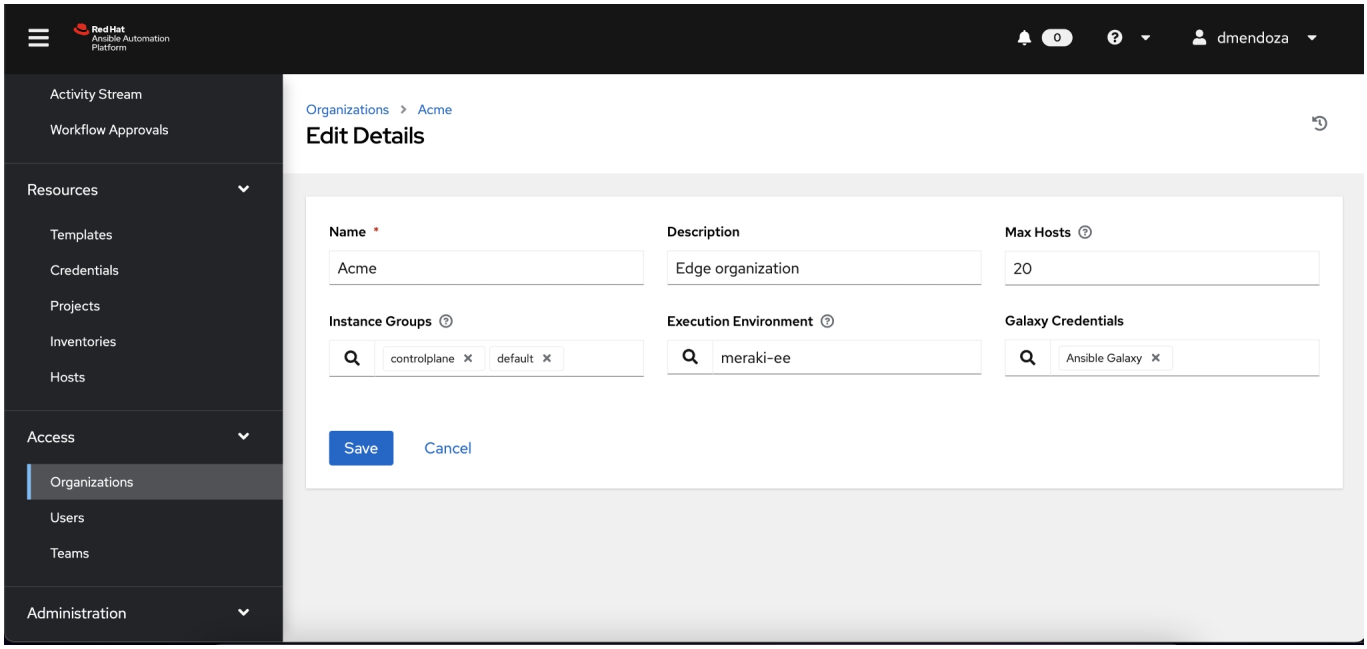

Figure 10: Navigate to the Organization configuration page.

Source Control credentials

If you are using a private repository, you need a Source Control credential similar to the one shown in Figure 11.

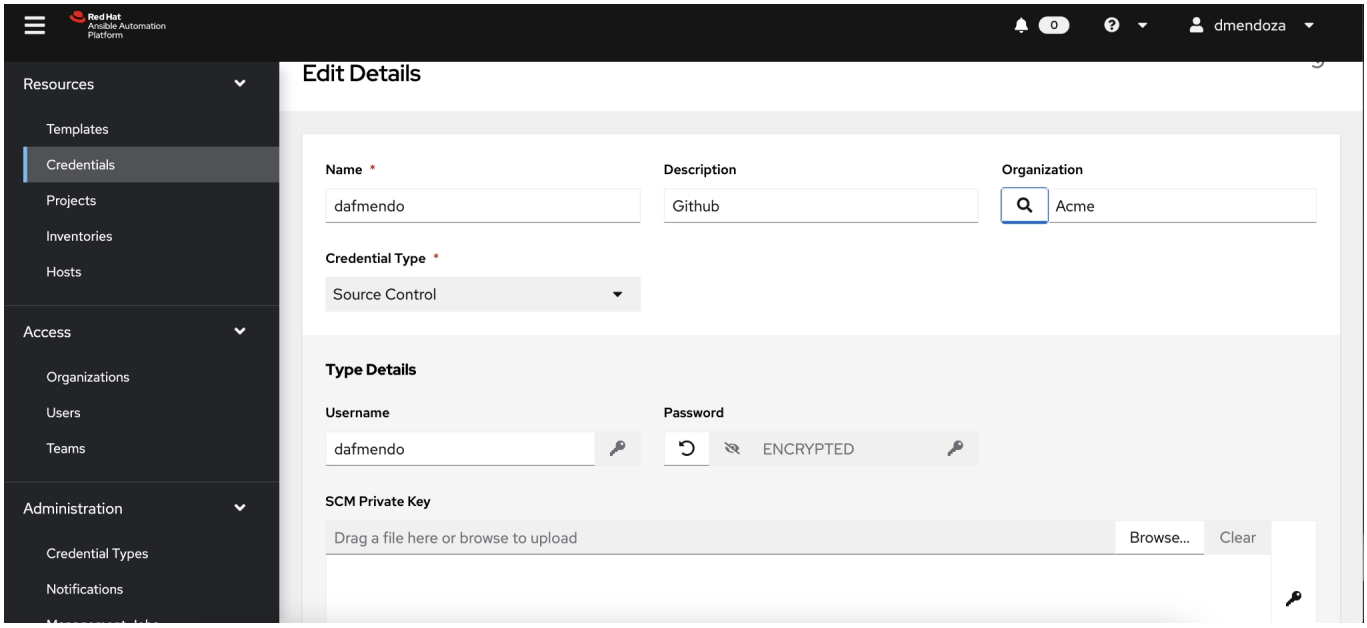

Figure 11: Set up a Source Control credential for a private repository.

Creating a Credential Type

To create the Meraki API Key credential, create a new Credential Type by visiting the **Administration** page and choosing **Credential Types**. You can then add a configuration for the credentials as shown in Figure 12.

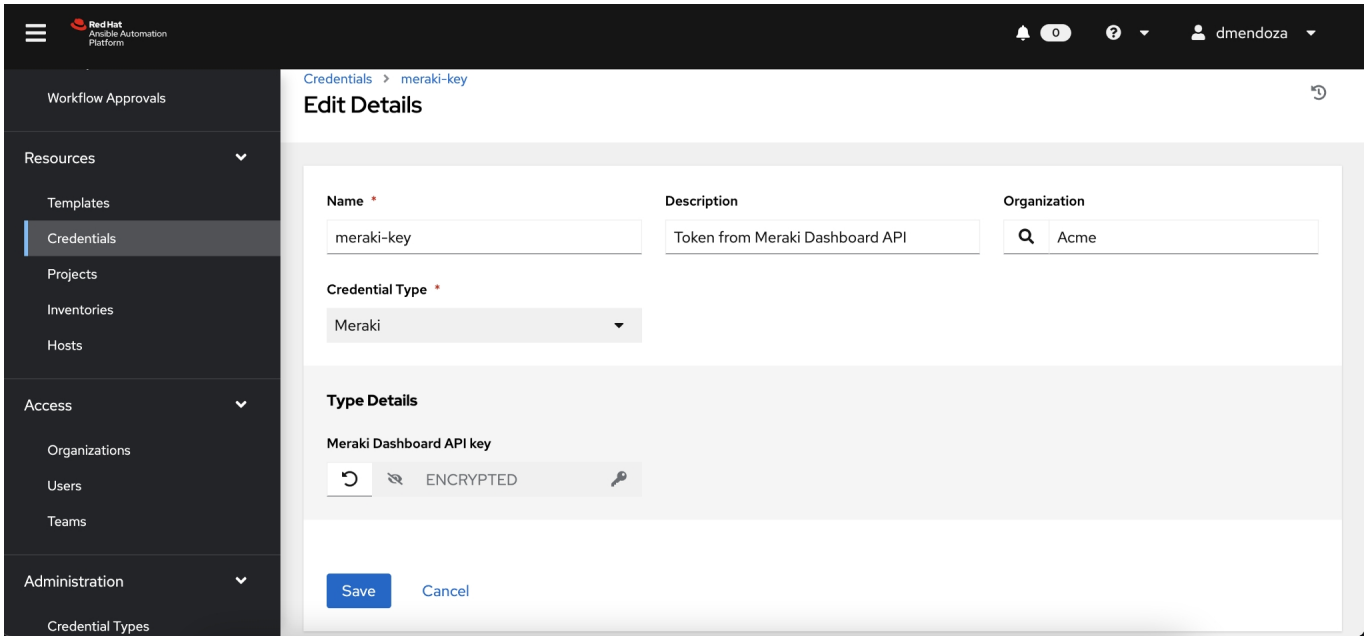

Figure 12: Create a new Credential Type to add a configuration for the Meraki API key credentials.

Configuring the API key

After creating the Meraki Credential Type, configure the API key that you created and copied from the Cisco Meraki Dashboard. From the **Resources** page choose **Credentials** and then **Add**. The details are shown in Figure 13.

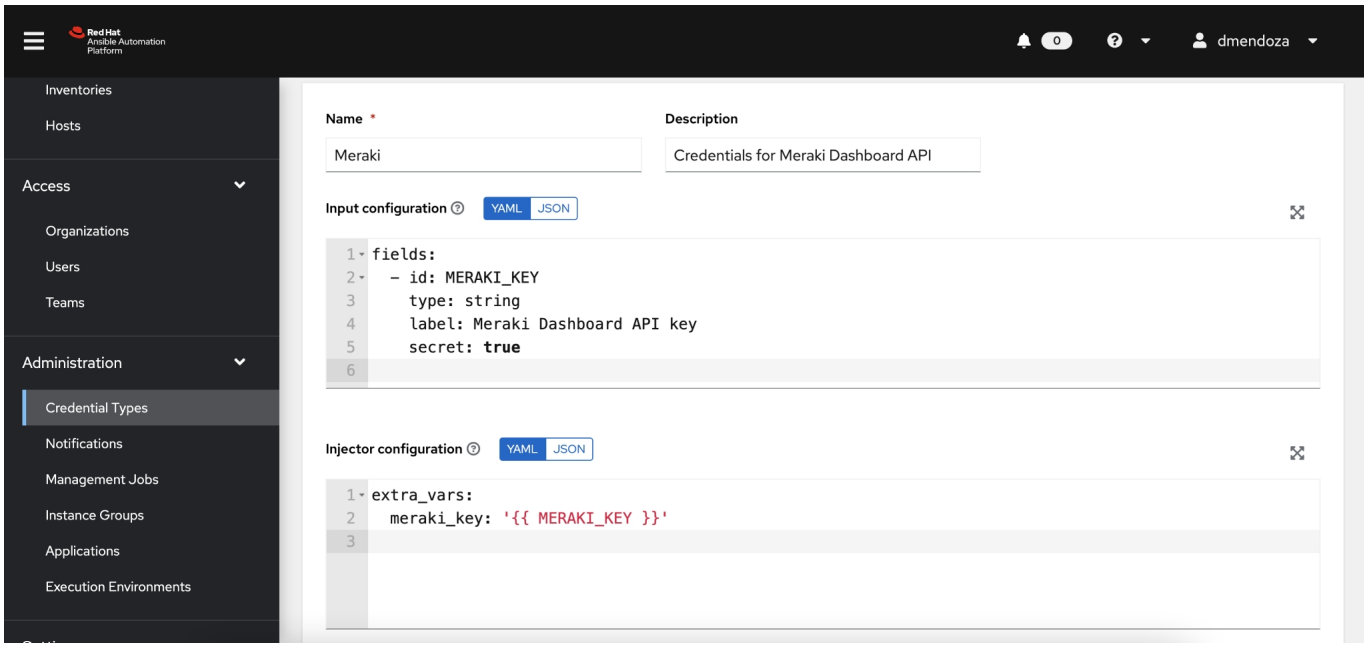

Figure 13: Configure the Meraki API key credentials as shown

Creating a project

Create a project by visiting **Resources** and choosing **Projects**. Figure 14 shows typical details for a project. In the **Source Control URL** field, this project specifies the GitHub repository you created previously. In my case, I named the repository `https://github.com/dafmendo/gitops_ansible-sdwan.`

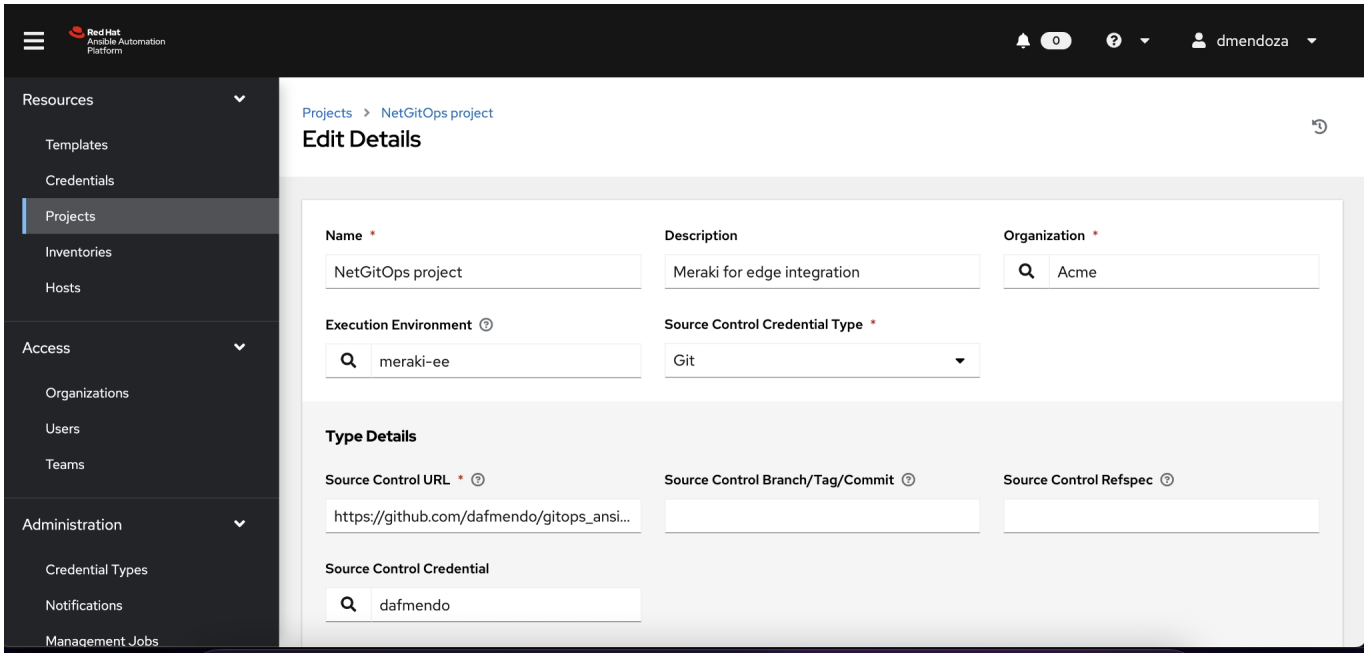

Figure 14: Details for a sample project.

If you select the **Update revision on launch** option (Figure 15), Ansible will automatically sync the project with the latest version of the GitHub repository and import the latest changes.

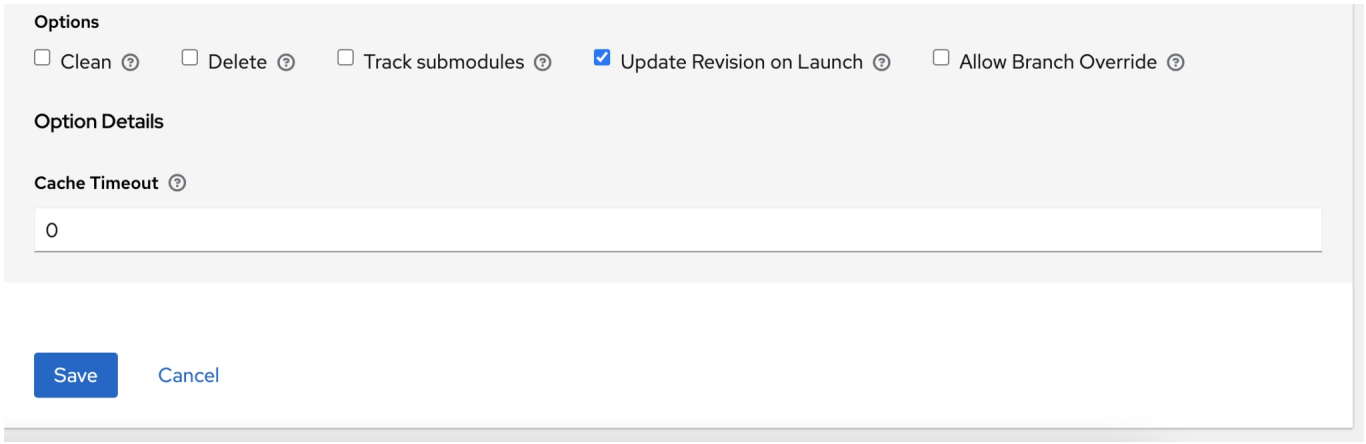

Figure 15: Select the Update Revision on Launch option to enable automatically syncing.

Enabling a GitHub trigger to invoke Ansible

Having set up Meraki, GitHub, and Ansible, you can create a Webhook trigger now in GitHub to invoke an Ansible job, through the steps in the following subsections.

Creating a personal access token

Create a personal access token (PAT) to allow the Webhook integration, by visiting **GitHub Repo → Settings → Developer** (Figure 16).

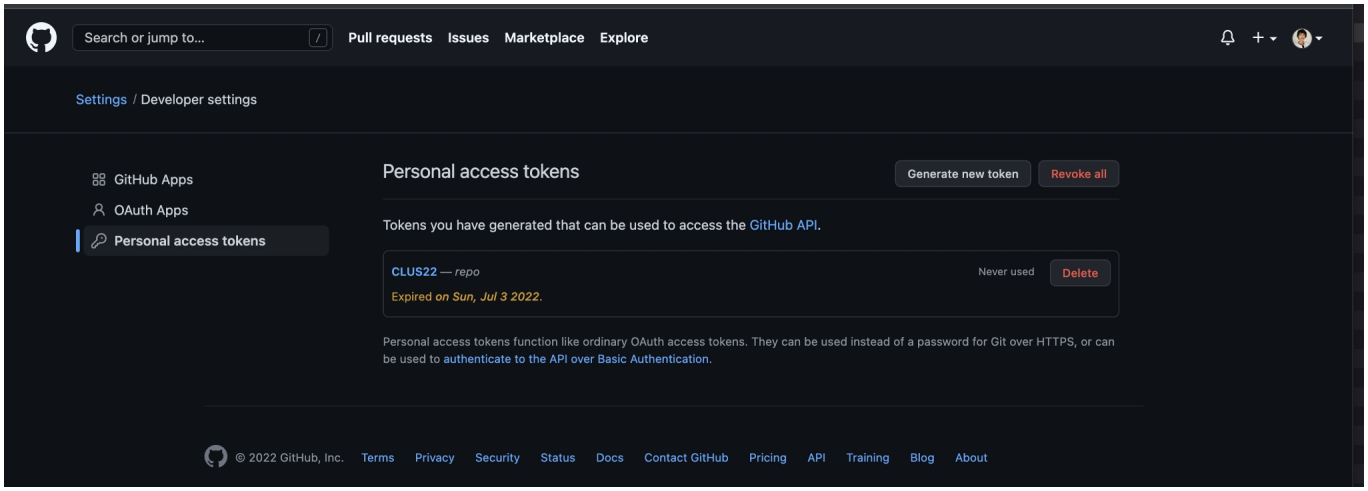

Figure 16: Create a personal access token in GitHub.

I am enabling the PAT to track all changes in the repository (Figure 17), but GitHub allows a deeper granularity of choices.

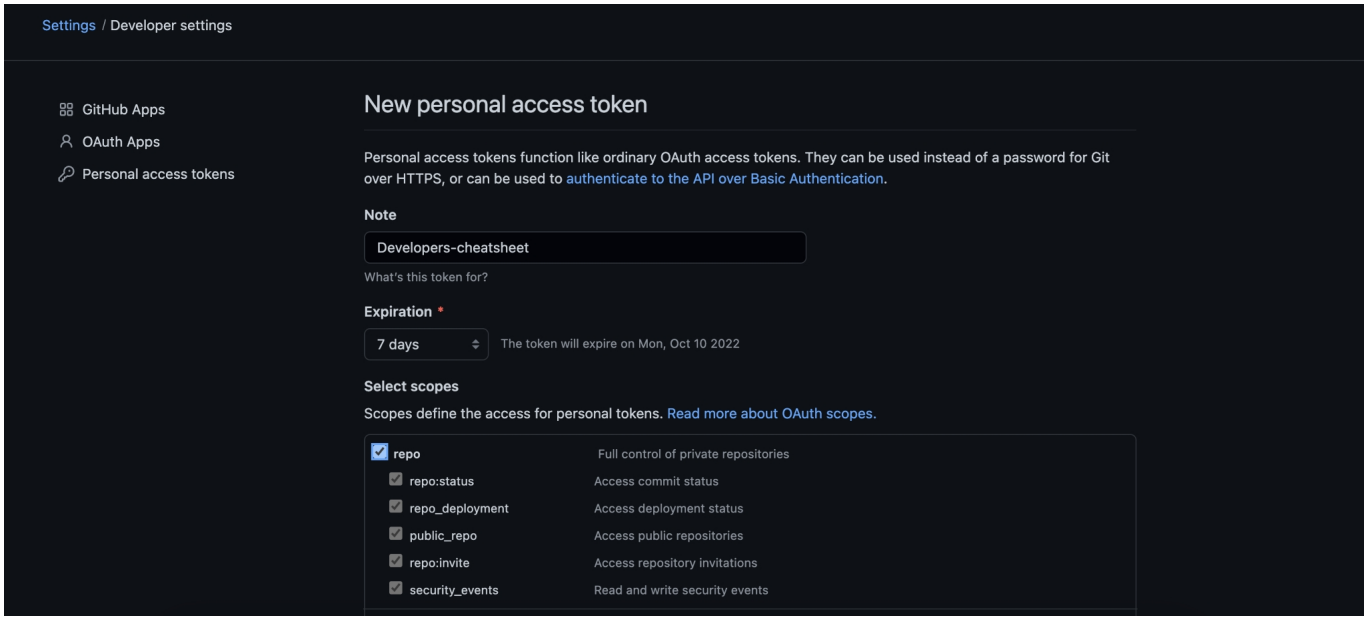

Figure 17: Defining the scope for tracking changes in the repository.

Once the PAT is created, make sure to copy it and save it, because you will not be able to view it in cleartext once you leave the page.

Create credentials for the PAT

To create the GitHub PAT credentials in Ansible, click the **Add** button from the **Resources→Credentials** page. Figure 18 shows typical credential settings.

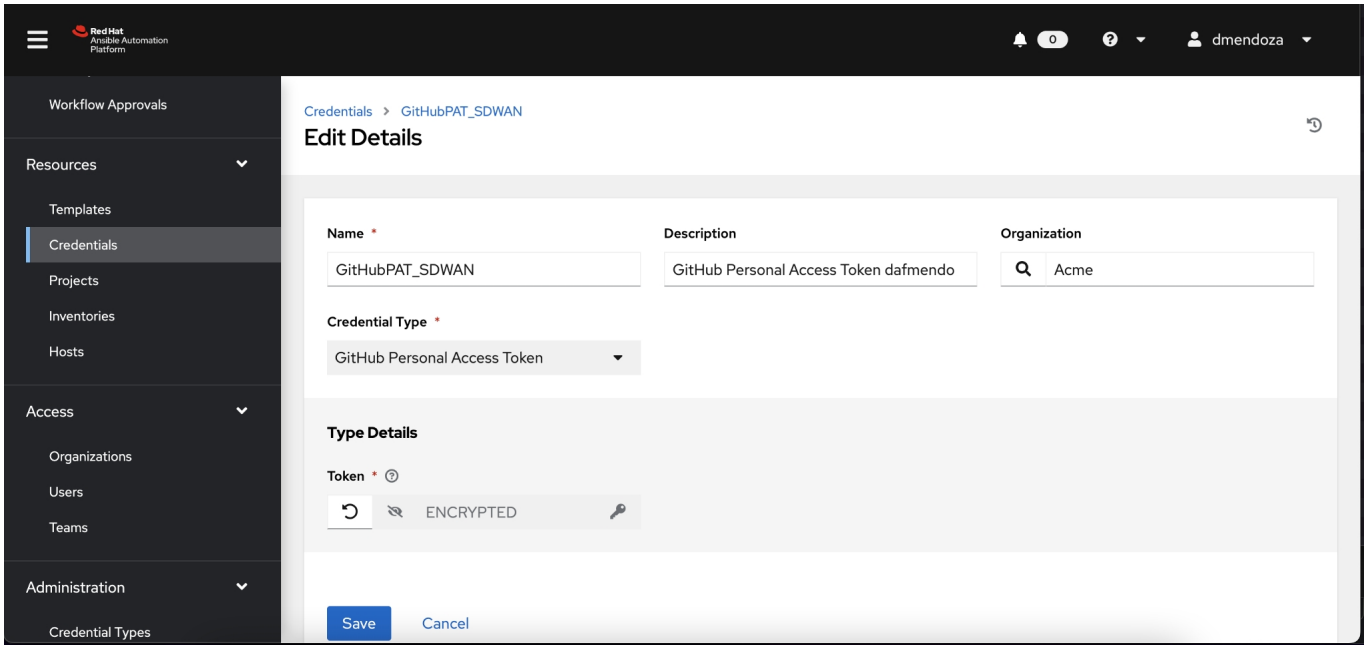

Figure 18: Credential settings for a sample project.

Creating a template that uses a Webhook

Create a template in Ansible by visiting **Resources→Templates** and selecting **Add job template** from the Add pulldown button (Figure 19).

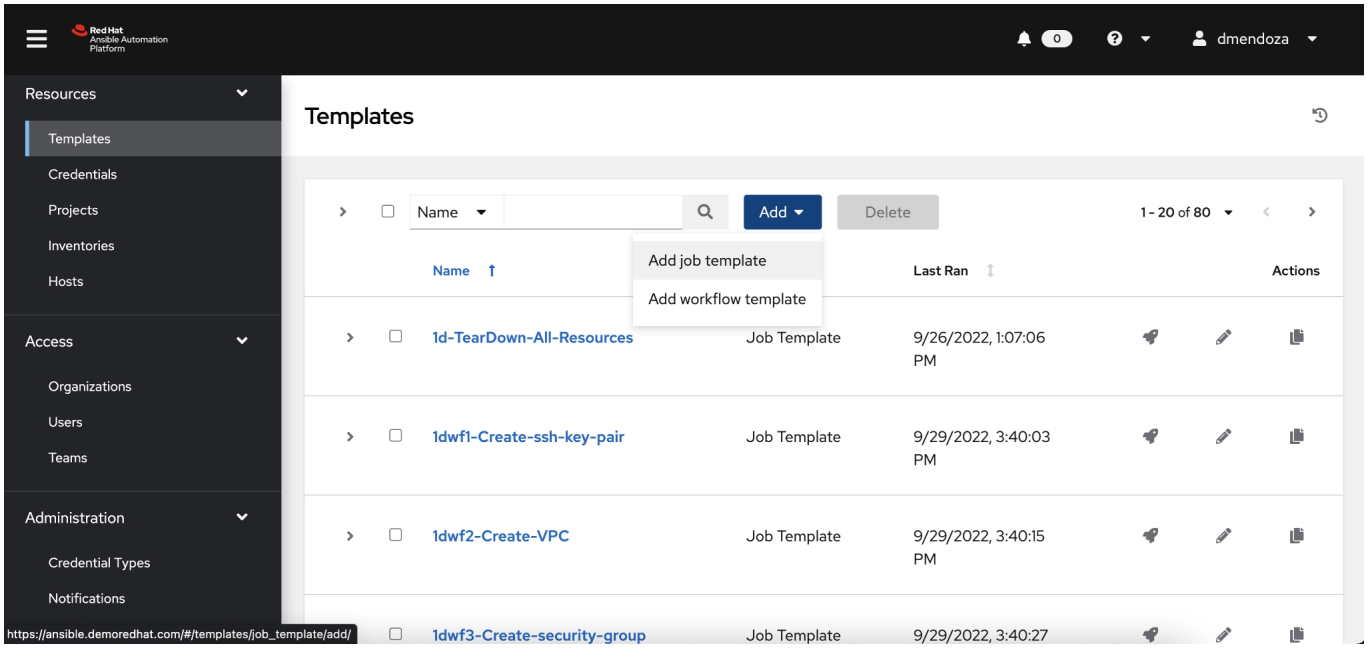

Figure 19: Create a new template in the Red Hat Ansible Automation Platform interface.

Fill out the settings to use the "Configure SSID" playbook as shown in Figure 20.

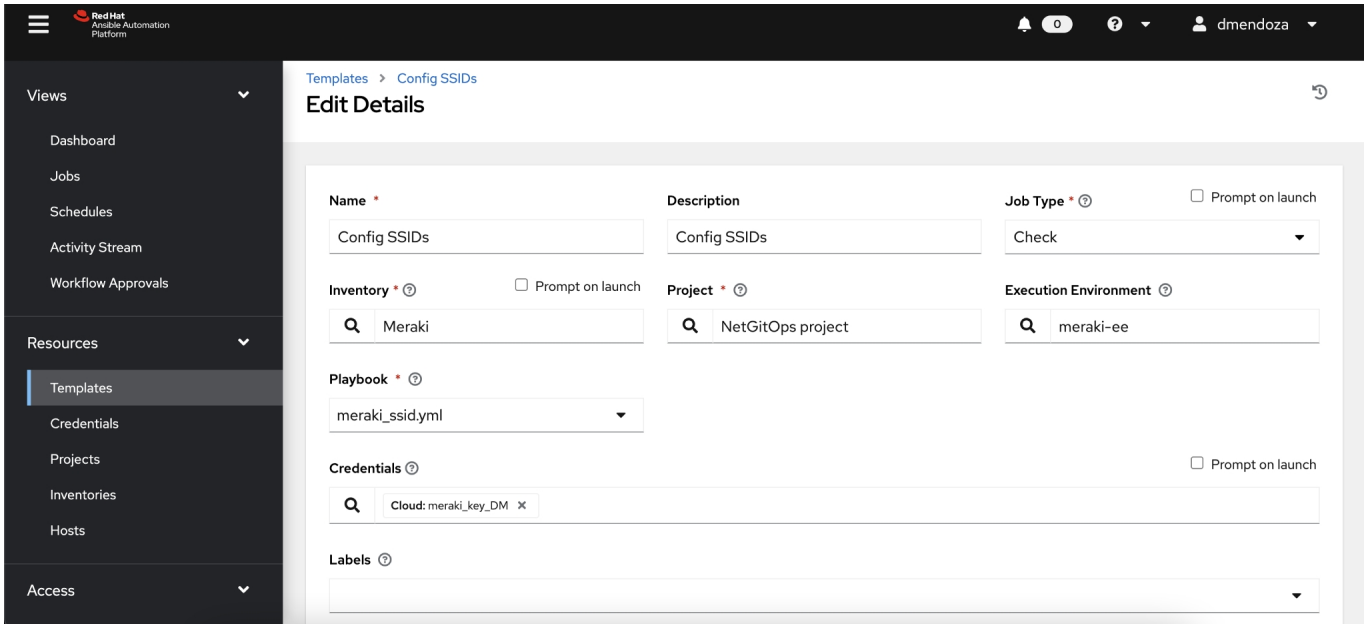

Figure 20: Configure use of the "Config SSIDs" playbook.

Before you save your settings, enable a Webhook to execute the job template by checking the **Enable Webhook** checkbox. Choose **GitHub** as the Webhook service as shown in Figure 21. A Webhook URL will be created.

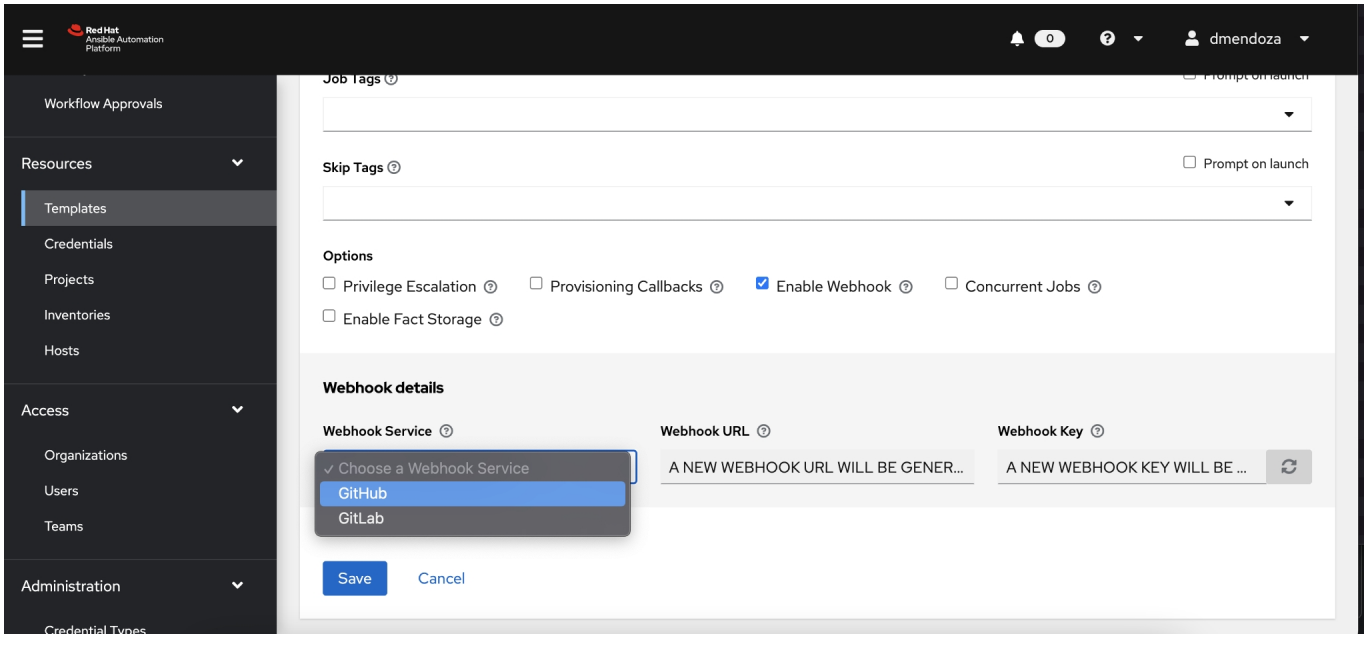

Figure 21: Enable the GitHub Webhook service.

Copy the URL of the Webhook, because you will use that URL later to configure the Webhook from GitHub.

Further down on the same page, configure the **Webhook Credential** (Figure 22).

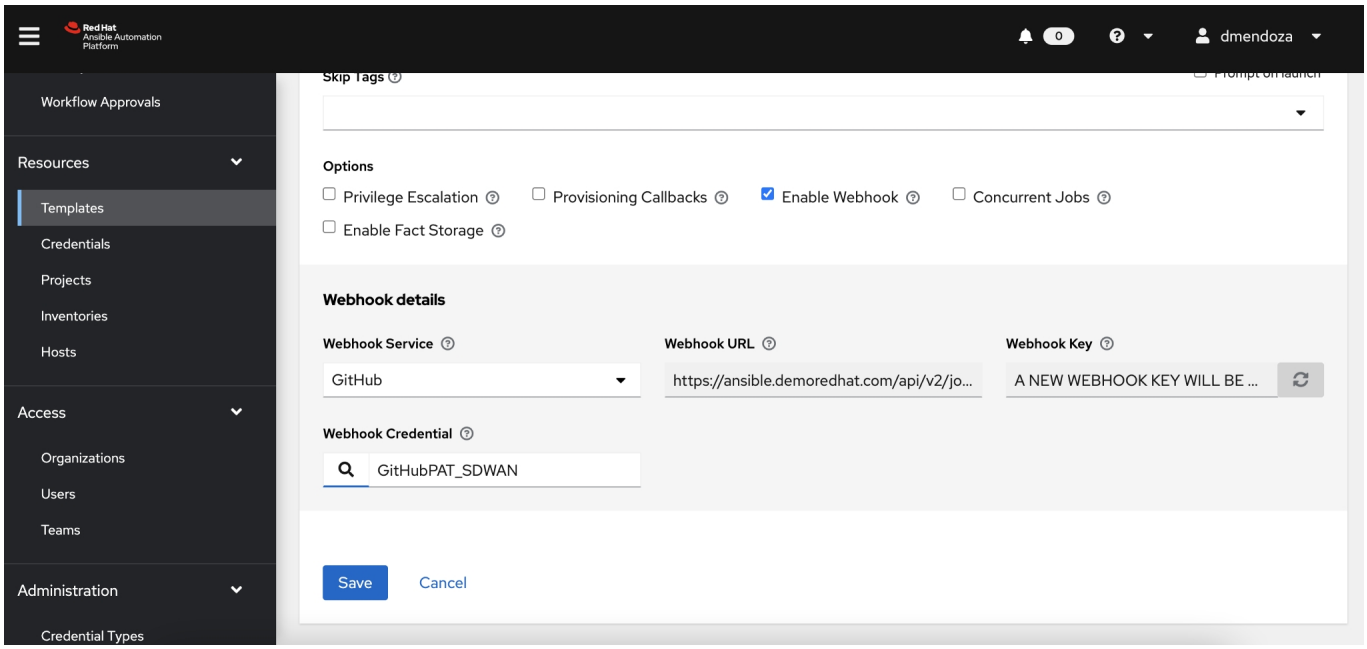

Figure 22: Configure the Webhook credential.

A Webhook key is automatically generated by Ansible once you press **Save** in the job template. Figure 23 shows the details of the template, including the Webhook key.

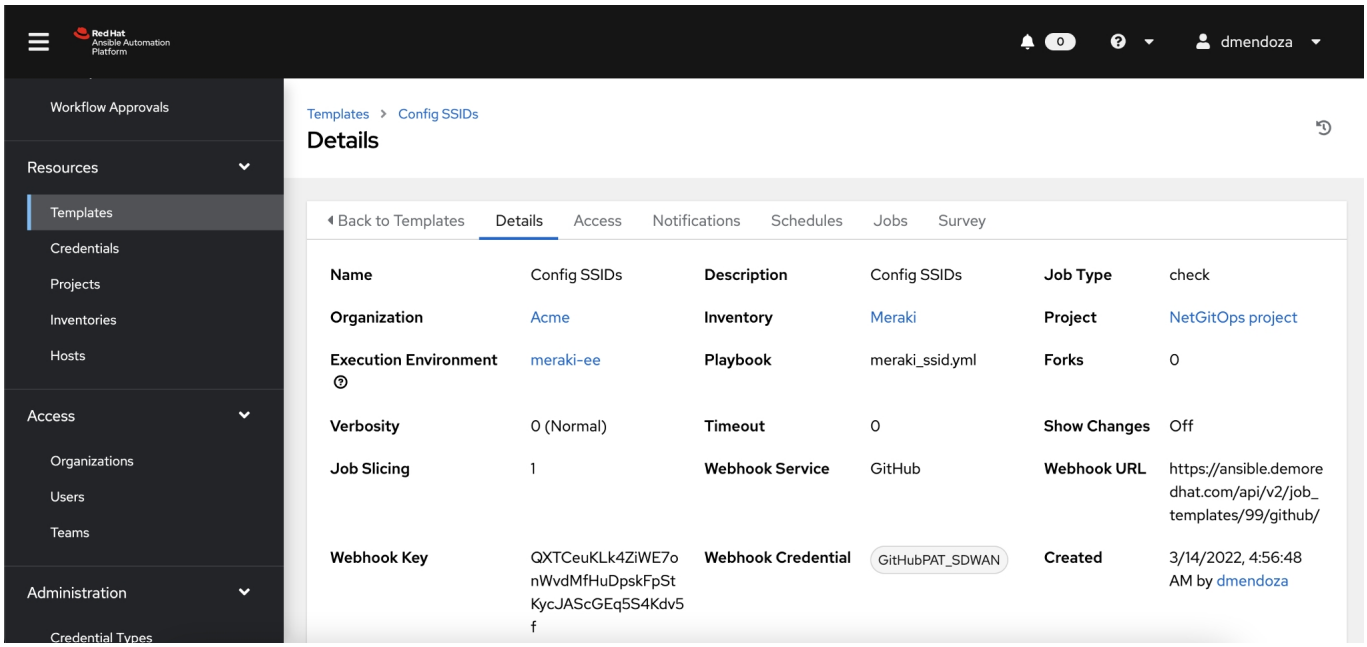

Figure 23: The job template details with the Webhook key.

Creating a Webhook in GitHub

Log back into GitHub and choose the repository you are using for this example (Figure 24). You are going to enable the Webhook from the repository, not at the global user level where you configured the PAT credentials.

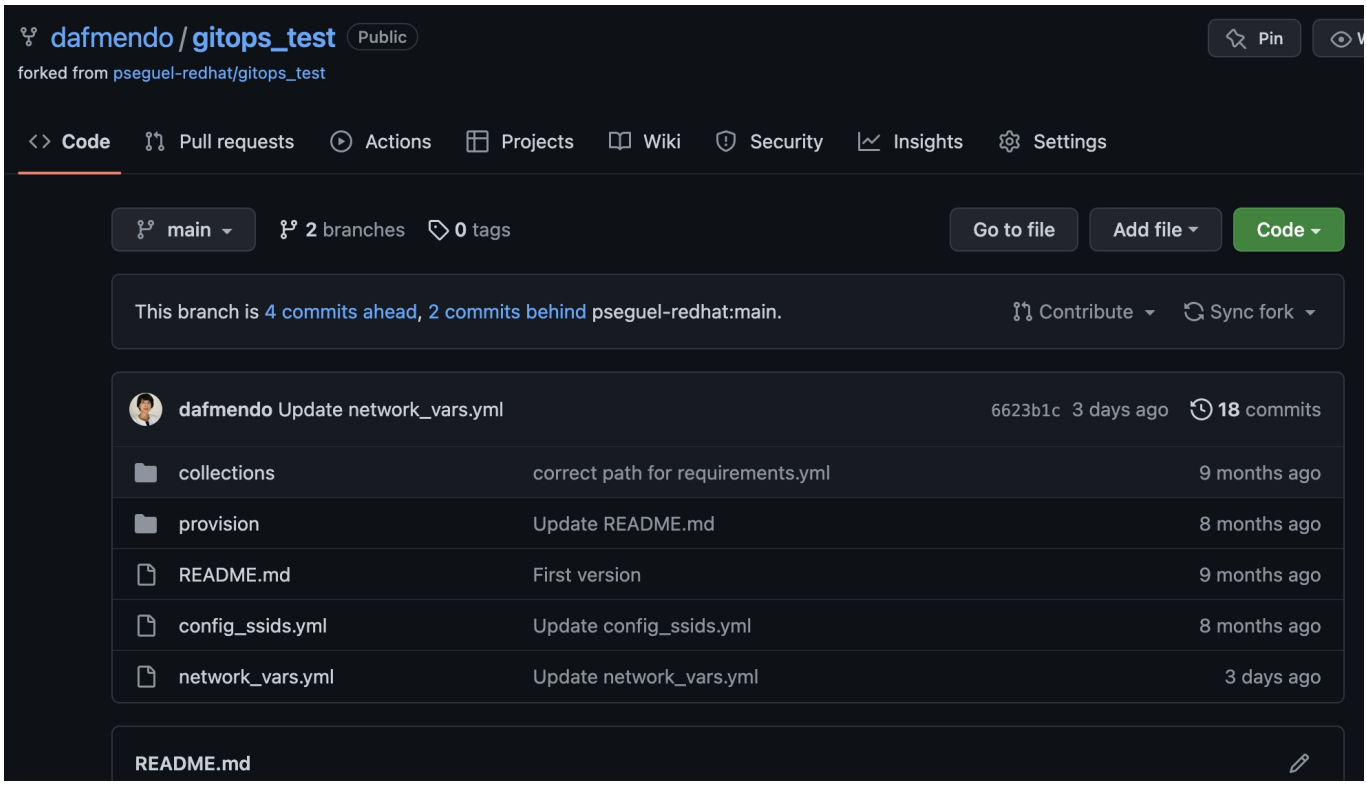

Figure 24: Enable the webhook from the GitHub repository.

Choose **Settings→Code and automation→Webhooks** (Figure 25) and click **Add webhook**.

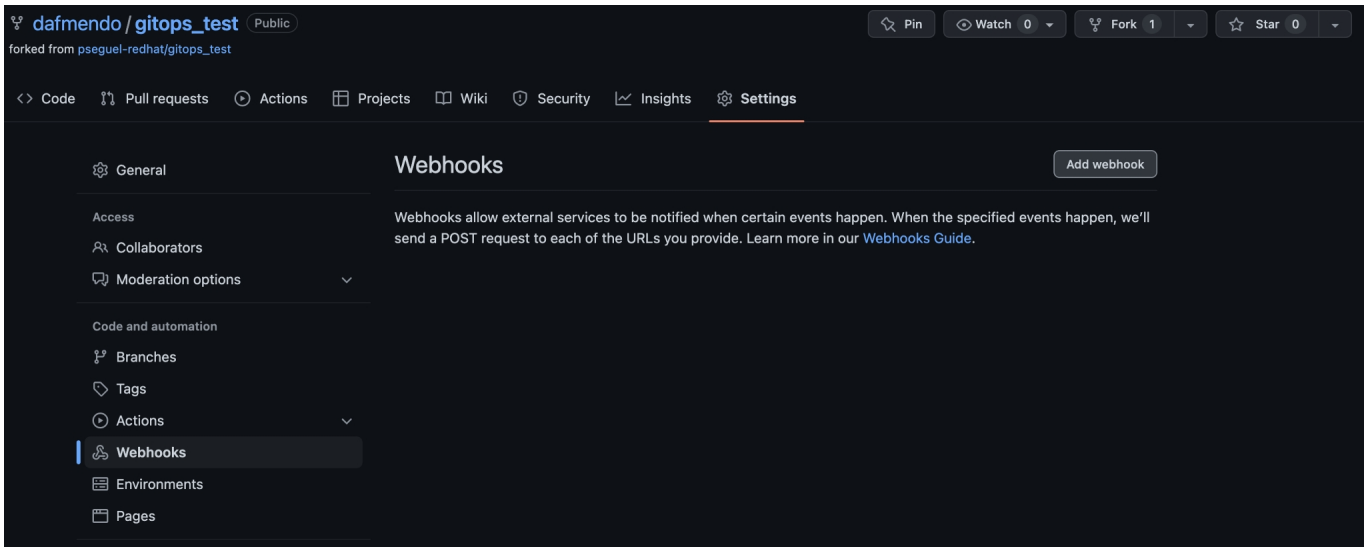

Figure 25: Add the webhook.

Complete the **Payload URL** and **Secret** fields with the information generated by Ansible after you created your job template (Figure 26). The **Content Type** should be `application/json`. For the sake of simplicity I have disabled SSL verification, but SSL verification is strongly recommended for production environments due to security concerns.

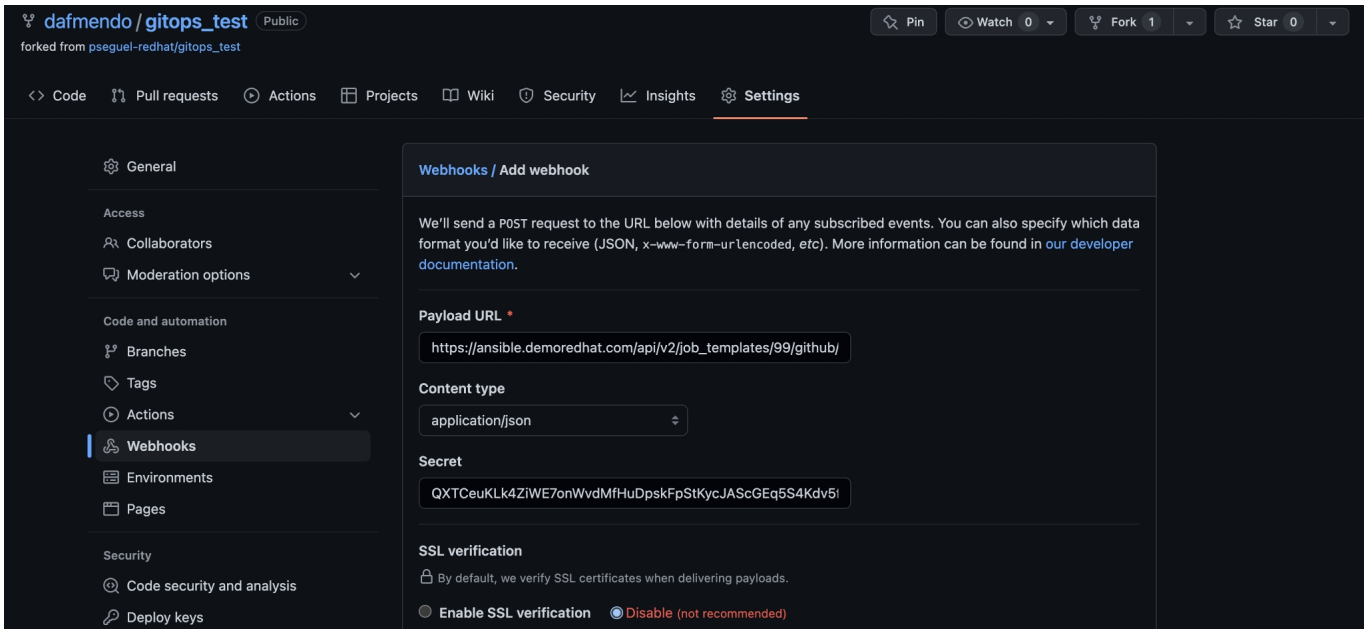

Figure 26: Fill in the Payload URL and Secret fields.

To trigger this Webhook, enable only push events (Figure 27).

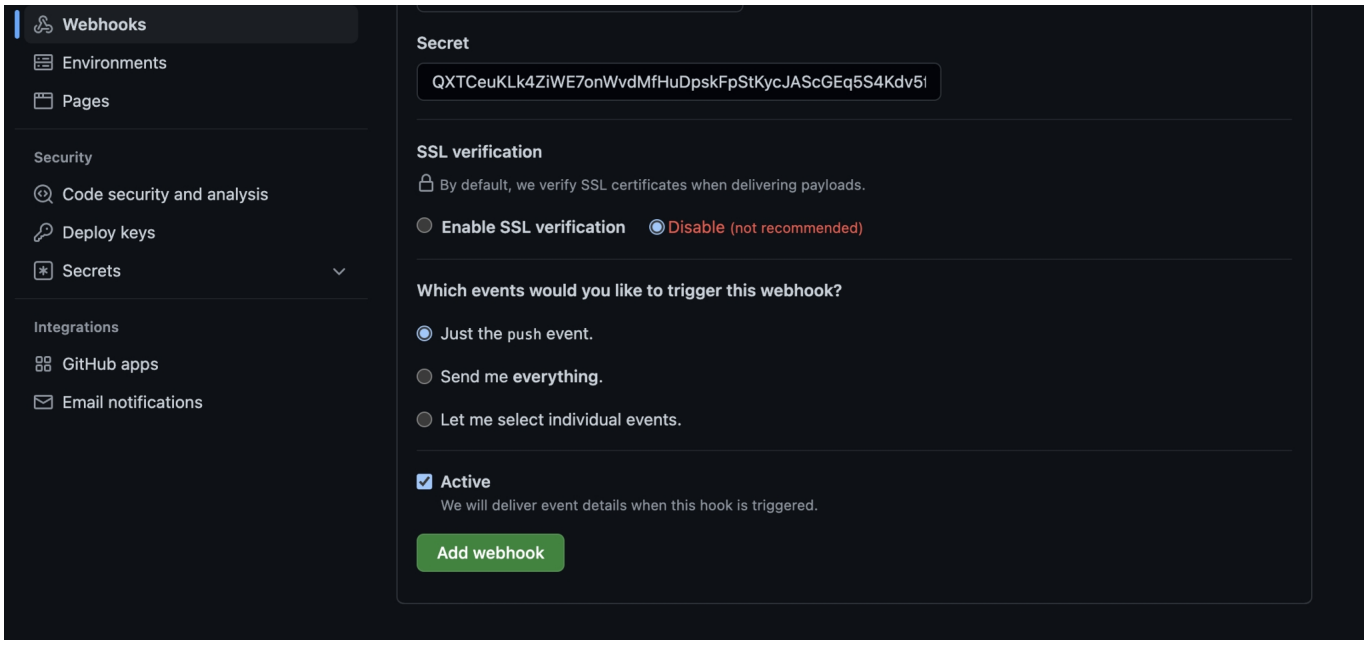

Figure 27: Enable only push events to trigger the webhook.

Depending on your security settings, the Webhook addition might require two-step approval.

Testing the system by creating an automatic trigger

Now that the integration is ready, you can trigger a change from the GitHub repository simply by editing the `network_vars.yml` playbook and pushing your change to the repository. You can then verify that GitHub delivered the event to Ansible by checking **Recent Deliveries** under **Webhooks→Manage webhooks** (Figure 28).

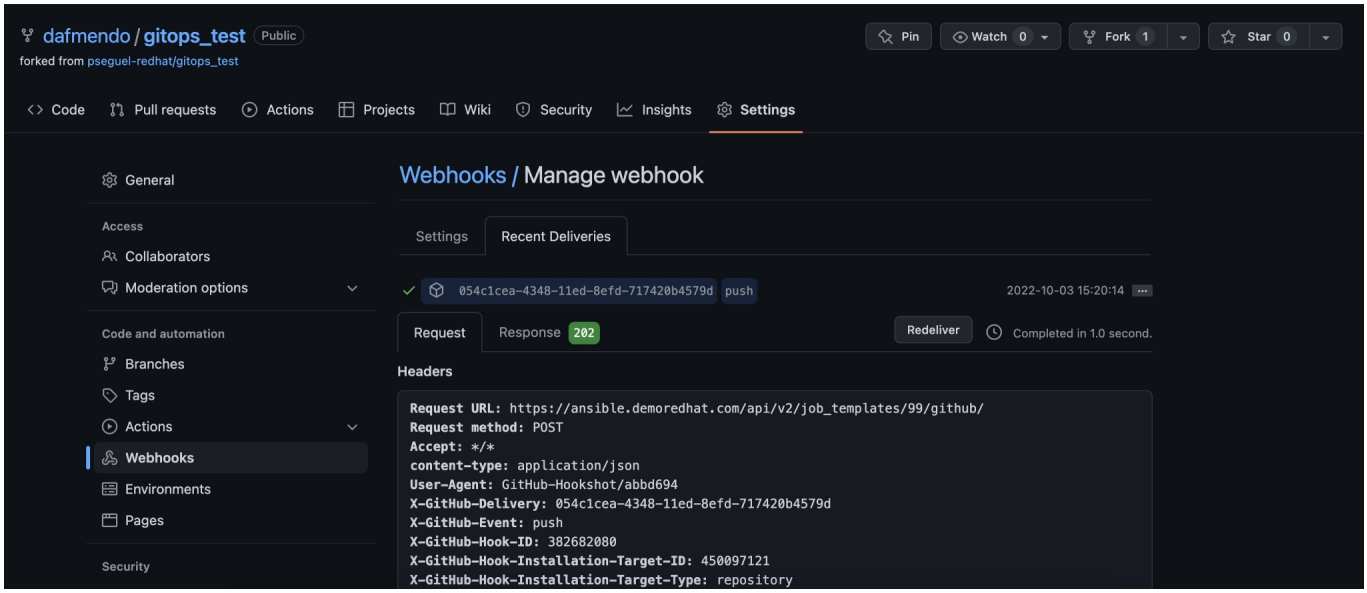

Figure 28: Verify the event delivery in GitHub.

In the Ansible dashboard, you can select **Views→Jobs** to observe the job execution.