SECRETS MANAGER > INTEGRATIONS

Ansible

View in the help center:
https://bitwarden.com/help/ansible-integration/

Ansible

Bitwarden offers an integration with Ansible to retrieve secrets from Secrets Manager and inject them into your Ansible playbook. The lookup plugin will inject retrieved secrets as masked environment variables inside an Ansible playbook. To setup the collection:

Requirements

- We recommend installing Python packages in a Python virtual environment.
- Current version of Ansible installed on your system.
- Bitwarden Secrets Manager with an active machine account.

Prior to setting up the Ansible collection, we recommend that you also open Secrets Manager to access your access token and any secrets you wish to include in the setup.

Install the Bitwarden Ansible collection

The following guide is a setup example for the Bitwarden collection using a Linux machine.

1. Install the Bitwarden SDK:

   Bash

   ```bash
   pip install bitwarden-sdk
   ```

2. Install bitwarden.secrets collection:

   Bash

   ```bash
   ansible-galaxy collection install bitwarden.secrets
   ```

Now that the Ansible collection has been installed, we can begin calling Bitwarden secrets from an Ansible playbook with `bitwarden.secrets.lookup`. The following section will include examples to demonstrate this process.

> ⓘ **Note**
>
> macOS users may need to set the following environment variable in shell in order to avoid Ansible issues upstream.
>
> - `export OBJC_DISABLE_INITIALIZE_FORK_SAFETY=YES`

Fetch Bitwarden secrets

To fetch secrets from Secrets Manager in your playbook, there are two methods:

Save access token as environment variable.

Using the Secrets Manager, we can securely set our access token as an environment variable in the shell and use the playbook to retrieve the secret. To authenticate the access token:

1. In the shell, run the following command to set your access token environment variable:

```Bash
export BWS_ACCESS_TOKEN=<ACCESS_TOKEN_VALUE>
```

2. Now that the environment variable has been set, we can use the lookup plugin to populate variables in our playbook. For example:

```Bash
  vars:
        database_password: "{{ lookup('bitwarden.secrets.lookup', '<SECRET_ID>') }}"
```

> ⓘ **Note**
>
> By setting `BWS_ACCESS_TOKEN` as an environment variable, the access token can be referenced without including the raw access token value in the playbook.

Supply access token in playbook

The Secrets Manager access token can also be referenced within the playbook itself. This method would not require you to use the `BWS_ACCESS_TOKEN` environment variable in your shell, however, the access token value will be stored in the playbook itself.

1. Access tokens may be included in the playbook with the following example:

```Bash
vars:
    password_with_a_different_access_token: "{{ lookup('bitwarden.secrets.lookup', '<SECRET_ID_V
ALUE>',
access_token='<ACCESS_TOKEN_VALUE>') }}"
```

Using this method, multiple access tokens may be referenced in a single playbook.

Retrieve secret from different server

Bitwarden self-hosted users can retrieve secrets from their Bitwarden server by including the `base_url`, `api_url` and `identity_url`:

```bash
vars:
    secret_from_other_server: "{{ lookup('bitwarden.secrets.lookup', '<SECRET_ID>', base_url='https://bitwarden.example.com' ) }}"
    secret_advanced: >-
      {{ lookup('bitwarden.secrets.lookup', '<SECRET_ID>',
        api_url='https://bitwarden.example.com/api',
        identity_url='https://bitwarden.example.com/identity' ) }}
```

Example playbook

The following is an example of a playbook file with several configuration options.

```bash
---
- name: Using secrets from Bitwarden

  vars:
    bws_access_token: "{{ lookup('env', 'CUSTOM_ACCESS_TOKEN_VAR') }}"
    state_file_dir: "{{ '~/.config/bitwarden-sm' | expanduser }}"
    secret_id: "9165d7a8-2c22-476e-8add-b0d50162c5cc"

    secret: "{{ lookup('bitwarden.secrets.lookup', secret_id) }}"
    secret_with_field: "{{ lookup('bitwarden.secrets.lookup', secret_id, field='note' ) }}"
    secret_with_access_token: "{{ lookup('bitwarden.secrets.lookup', secret_id, access_token=bws_access_token ) }}"
    secret_with_state_file: "{{ lookup('bitwarden.secrets.lookup', secret_id, state_file_dir=state_file_dir ) }}"

  tasks:
    - name: Use the secret in a task
      include_tasks: tasks/add_db_user.yml # reference the secrets with "{{ secret }}", "{{ secret_with_field }}", etc.
```

ⓘ **Note**

In the example above the CUSTOM_ACCESS_TOKEN_VAR demonstrates that you may include multiple, different access tokens. These do not have to be hard carded and can be supplied securely to your playbook.

Variable	Additional information
bws_access_token	Lookup access token env variable.
state_file_dir	A directory where your authentication state can be cached.
secret_id	ID of the secret you wish to lookup.
secret	Lookup a secret value and store it as a variable named "secret".
secret_with_field	Lookup a secret with additional field output. In this example, the lookup will return the secret's 'note' value.
secret_with_access_token	Lookup a secret with the access token value included in the request.
secret_with_state_file	Lookup a secret with the pre configured state file included in the request.

Additional requests and fields

In addition to the secret_id, several fields can be included in the bitwarden.secrets.lookup. A The following JSON object includes all of the fields that can be referenced in the playbook lookup:

Bash

```
{
  "id": "be8e0ad8-d545-4017-a55a-b02f014d4158",
  "organizationId": "10e8cbfa-7bd2-4361-bd6f-b02e013f9c41",
  "projectId": "e325ea69-a3ab-4dff-836f-b02e013fe530",
  "key": "SES_KEY",
  "value": "0.982492bc-7f37-4475-9e60",
  "note": "",
  "creationDate": "2023-06-28T20:13:20.643567Z",
  "revisionDate": "2023-06-28T20:13:20.643567Z"
}
```

To retrieve additional fields such as `"note"`, the following command can be added to the playbook:

Bash

```
  vars:
        database_password: "{{ lookup('bitwarden.secrets.lookup', '0037ed90-efbb-4d59-a798-b103012487a0', field='note') }}"
```