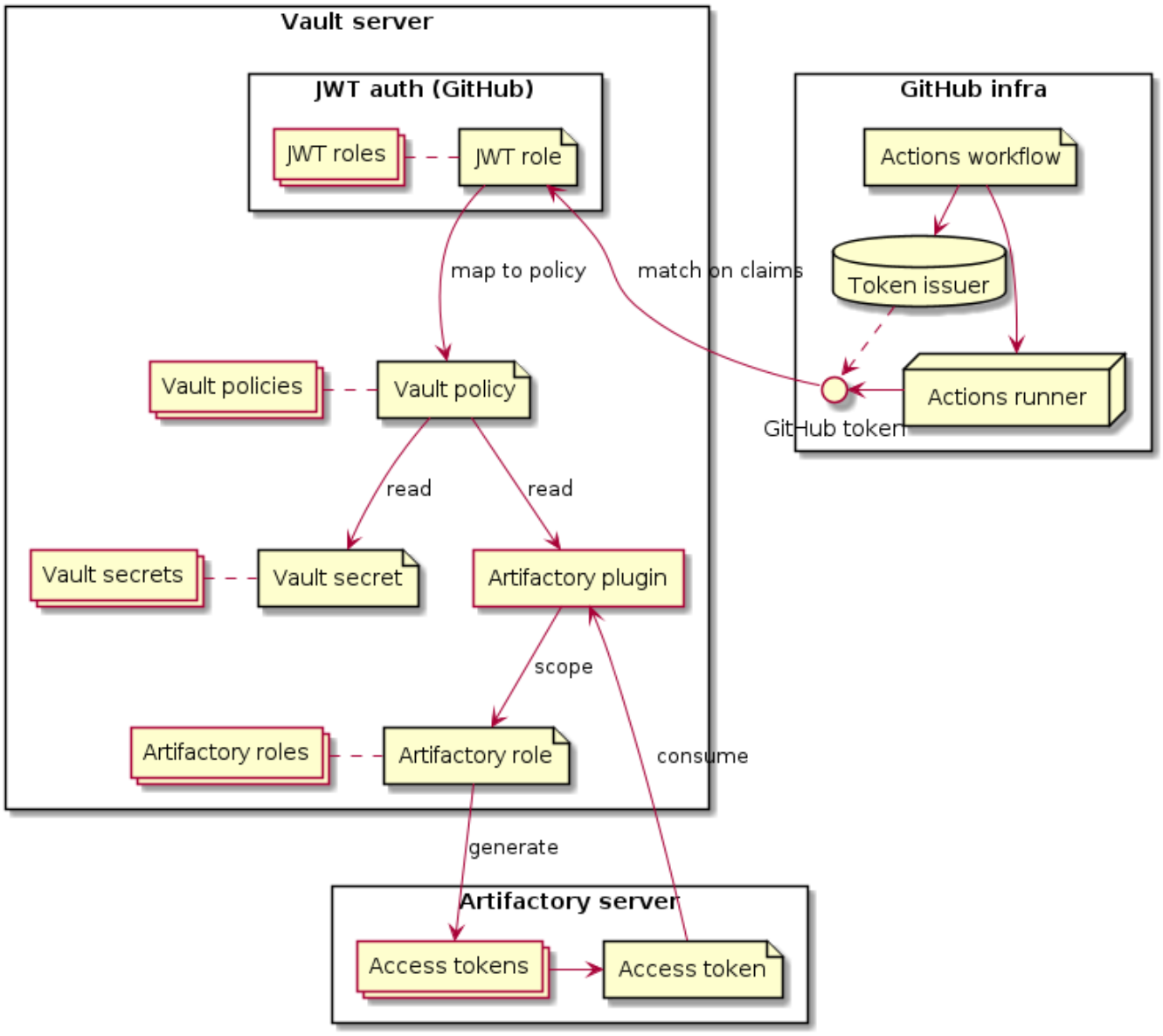
HOW-TO: Setup Vault Artifactory secrets plugin

Background

- Setup the Vault Artifactory secrets plugin from scratch. The main docs are not that helpful when trouble happens.
- I have Vault and GitHub JWT integration working well. My example repo is here: https://github.com/tenzin-io/test-vault
  - Prerequisite: HOW-TO: Setup GitHub actions to access Vault secrets

Interaction of GitHub and Vault in accessing Artifactory Vault plugin secrets



Procedure

Prerequisites

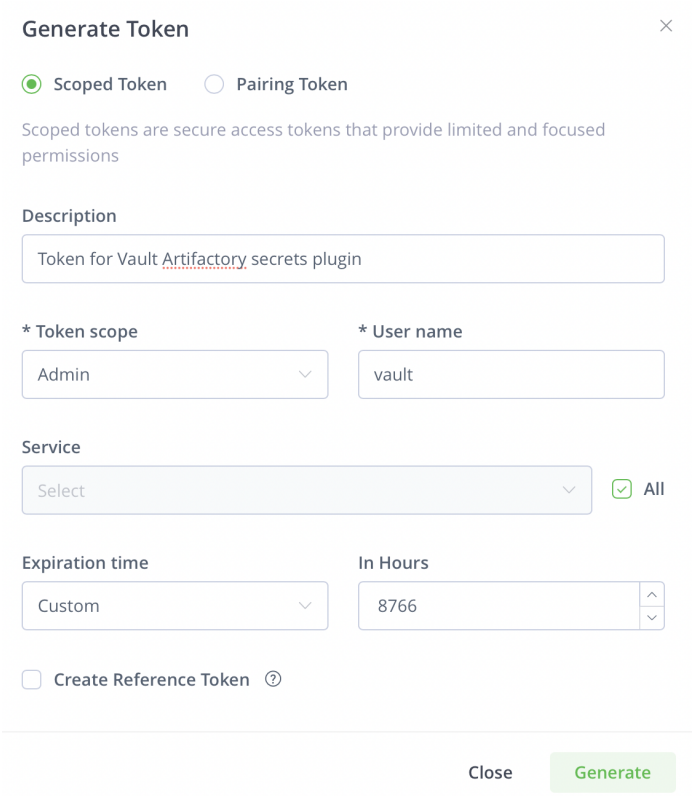
The Vault server configuration needs to be prepared and the plugin downloaded.

	Steps

1	Open the Vault server configuration file in an editor.
	
	Example vault-server.hcl
	
	`ui = true`
	
	`listener "tcp" {`
	`  address = ":8200"`
	`  tls_disable = false`
	`  tls_cert_file = "/certs.d/vault.crt"`
	`  tls_key_file = "/certs.d/vault.key"`
	`  tls_client_ca_file = "/certs.d/rootCA.pem"`
	`}`
	
	`storage "file" {`
	`  path = "/vault/file"`
	`}`
	
	`plugin_directory = "/vault/plugins"`
	`disable_mlock = true`
	
	`# for debugging plugin and other problems`
	`log_level = "trace"`

2	The following should be included in the configuration.
	
	`plugin_directory = "/vault/plugins"`
	`disable_mlock = true`
	
	`disable_mlock` is not safe, as this means that memory could be swapped to disk.

3	Download the Artifactory plugin: • https://github.com/jfrog/artifactory-secrets-plugin/releases
4	Move the binary to the `/vault/plugins` folder. 1. Ensure that the plugins directory is owned by user vault and group vault. 2. Ensure that the binary `artifactory` has only user vault read/write/execute permissions. More details on this plugin file permission is found here.
5	Record the SHA256 sum of the binary, this will be needed in the plugin registration step. `sha256sum /vault/plugins/artifactory`
6	Reload Vault after updating the configuration.

7	Login to the Artifactory server as an administrator and generate an Access Token for Vault plugin use. Navigate to Administration User Management Access Tokens Generate Token. 
8	Record the access token value.

Setting up Vault server with Artifactory plugin

The setup of the Artifactory plugin for Vault will have to be done completely with the Vault CLI.

	Steps

1	Log in to Vault using the Vault CLI, the user should have access to modifying plugins and policies.
	`vault login`
	`vault login -method=<ldap, userpass, github, etc>`

2 Register the Artifactory plugin into Vault.

```
vault write sys/plugins/catalog/secret/artifactory \
    sha_256="$(sha256sum /vault/plugins/artifactory | cut -d " " -f 1)" \
    command="artifactory"
        args="-tls-skip-verify=true"
```

Do **not** use the new `vault plugin register` method to register the plugin. The secrets engine fails to initialize the plugin properly.

The flags passed into the `args` field can be found by using `-h` on the plugin binary.

artifactory -h

```
# ./vault_plugins/artifactory -h
Usage of vault plugin settings:
  -ca-cert string

  -ca-path string

  -client-cert string

  -client-key string

  -tls-skip-verify

2022-09-04T11:01:11.267-0400 [ERROR] could not parse flags: error="flag: help requested"
```

3 Read the contents of the plugin path.

```
vault read sys/plugins/catalog/secret/artifactory
```

Example output

```
# vault read sys/plugins/catalog/secret/artifactory
Key        Value
---        -----
args       [-tls-skip-verify=true]
builtin    false
command    artifactory
name       artifactory
sha256     0ca7f4ad38be7fd790ae39aed96dc1961eeccf0f991632c82ffbd544658162eb
```

My setup needs the `-tls-skip-verify` due to the way I use DNS with self-signed certs, Lets Encrypt certs and Cloudflare certs.

4

Write the Artifactory plugin configuration. This will allow the plugin to authenticate to the correct Artifactory server.

```
vault write artifactory/config/admin \
        url=https://repo.tenzin.io/artifactory \
        access_token=2ZXIiOiIyIiw.....joiUlMyNTYiLCJ
```

> ⚠ The `access_token` should be high powered, as this parent token will be creating child access tokens for the roles.
>
> The roles can then be used to limit the permission scope for generated child access tokens.

5

Create the Artifactory role which will be used to generate Artifactory access tokens.

I've applied a scope to limit the permission of the generated tokens from this role.

```
vault write artifactory/roles/github-actions \
        username="github-actions" \
        scope="applied-permissions/groups:publish" \
        default_ttl=4h \
        max_ttl=8h
```

More details on the scope field can be found here.

6

Retrieve the Artifactory access token. The role name from earlier step corresponds to the token name.

```
vault read artifactory/token/github-actions
```

Example output

```
# vault read artifactory/token/github-actions
Key                Value
---                -----
lease_id           artifactory/token/github-actions/Ro5I9EeVJQXb3o7zYtllzp6Y
lease_duration     5m
lease_renewable    true
access_token       HRahY6VfGLbOGsvg4pZah13CifvDFDo.....Cv2pRPbsxueM5tTaow7sGXCQ
role               github-actions
scope              applied-permissions/groups:publish
token_id           a6be0ec9-efbe-43c1-bef7-2dc09709eaf8
```

7

Add a new Vault policy to allow reading from the Artifactory token path.

```
vault policy write read_artifactory_token - <<EOF
path "artifactory/token/github-actions"
{
 capabilities = ["read"]
}
EOF
```

8	Update the GitHub JWT auth role for test-vault to include the new policy `read_artifactory_token` created in the prior step.
	
	test-vault.json
	
	`{`
	`  "role_type": "jwt",`
	`  "policies": ["read_secret_mysecret", "read_artifactory_token" ],`
	`  "bound_audiences": "https://github.com/tenzin-io",`
	`  "user_claim": "repository",`
	`  "verbose_oidc_logging": true,`
	`  "bound_claims_type": "string",`
	`  "bound_claims":{`
	`  "repository": "tenzin-io/test-vault"`
	`  }`
	`}`
	
	`vault write auth/jwt/role/test-vault - < test-vault.json`

Update GitHub Actions workflow to retrieve the Artifactory access token

	Steps

1	Update the workflow YAML manifest and include the hashicorp/vault-action step.
	
	`- name: Import Secrets`
	`  uses: hashicorp/vault-action@v2.4.0`
	`  id: secrets`
	`  with:`
	`  url: https://vault.tenzin.io`
	`  method: jwt`
	`  role: test-vault`
	`  secrets: |`
	`  secret/data/mysecret secret_one | SECRET_ONE ;`
	`  /artifactory/token/github-actions access_token | JF_GITHUB_ACTIONS_TOKEN ;`

2	⚠ When accessing secrets from other secret engines, the absolute path is needed. More details on accessing other secrets here.
3	Dispatch the Actions workflow. Verified that my example repo is able to access the Artifactory access token: https://github.com/tenzin-io/test-vault

Appendix

Troubleshooting notes

Plugin debugging

The option `log_level=trace` is very helpful in the Vault server configuration file to debug plugin startup issues.

```
log_level = "trace"

disable_mlock = true
```

The option `disable_mlock` was needed for the Artifactory plugin to start, **even though** the Vault server info said mlock was supported and enabled.

Artifactory plugin failure, error message

```
[ERROR] plugin shutting down: error="cannot allocate memory"
```