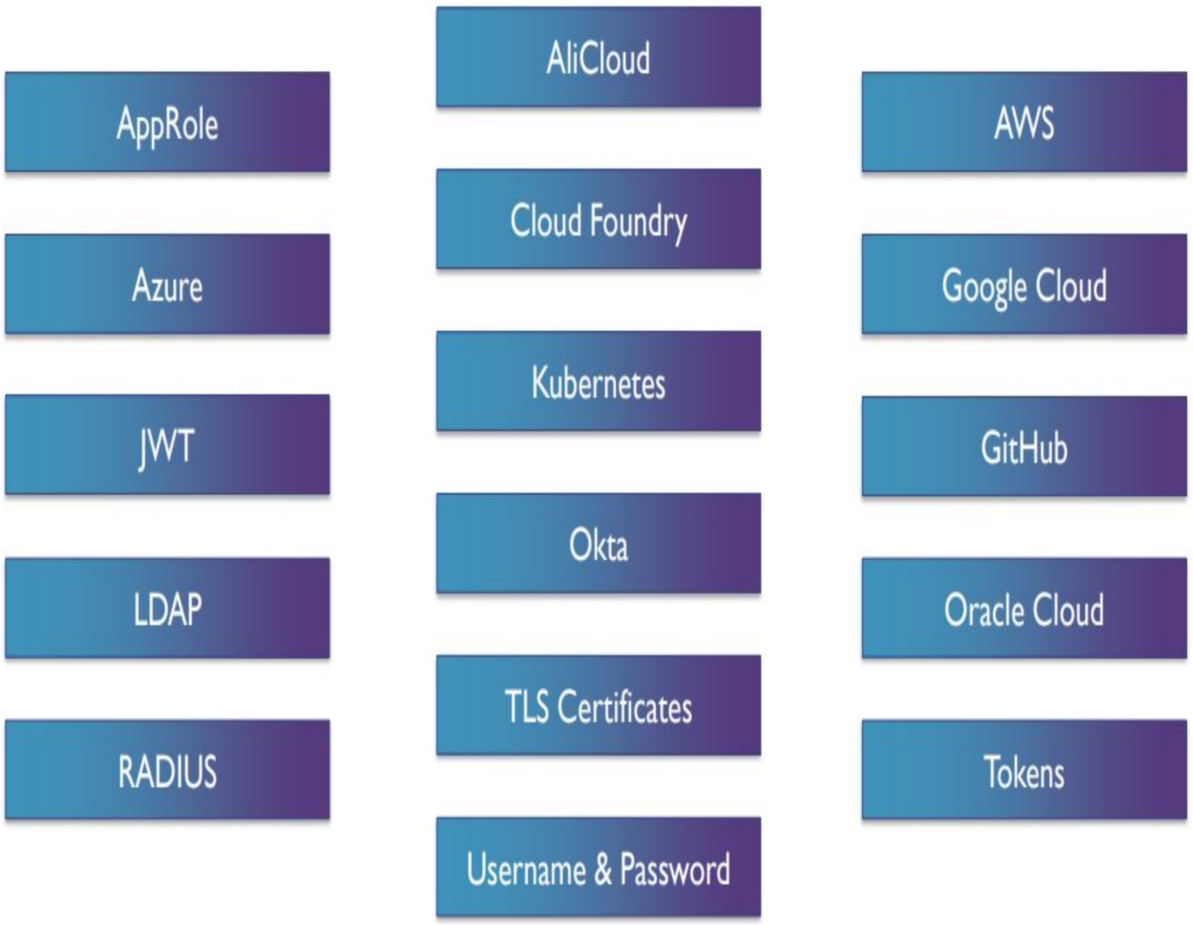
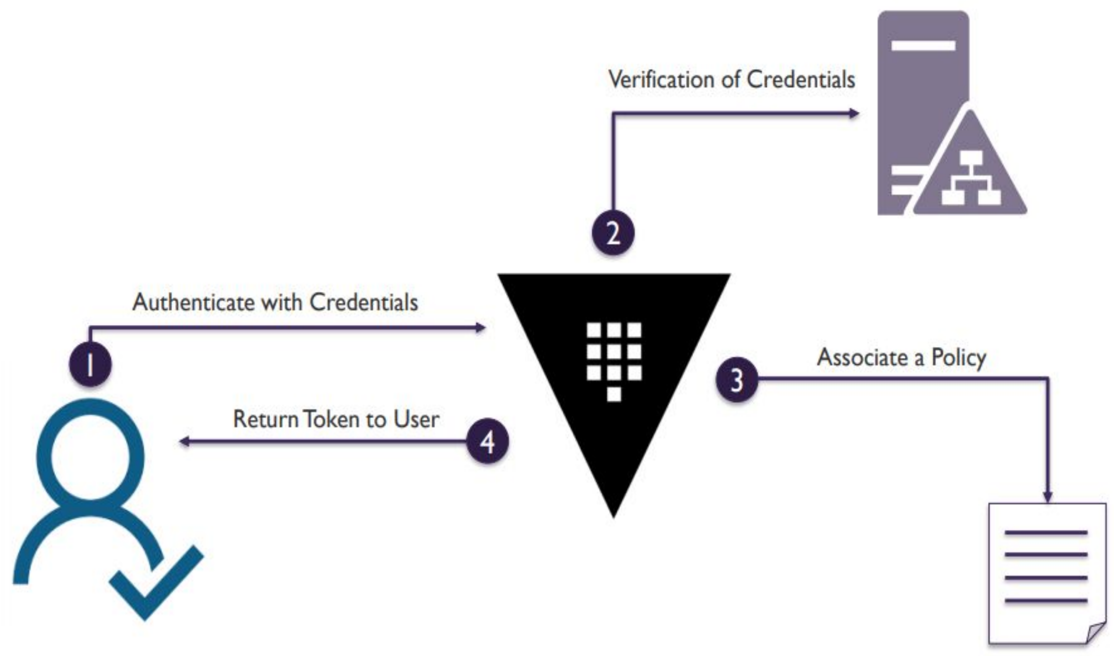
Intro to Auth Methods

- Components that perform authentication to Vault itself.
- Responsible for assigning identity and policies to a user .
- Multiple auth methods can be enabled depending on your use case .
- Auth methods are enabled at a ‘path’ – commonly using the same name as the auth method but not required
- Default authentication method for new install = tokens



Authentication Flow:



Default Auth Method:

- The default auth method is tokens
- Root token is assigned the root policy upon Vault initialization
- Root token should not be used on a day-to-day basis
- After enabling other auth methods, create your own ‘admin’ and delete the root token

```
$ vault token revoke s.uO8rAyMOBg2uHXi9OKu8MtQz
```

Determining the Correct Auth Method:

Different use cases call for different auth methods

Examples of auth methods humans will generally use:

– LDAP

– Userpass

– Okta

Machines will likely use:
– AppRole
– AWS
– Tokens Tokens

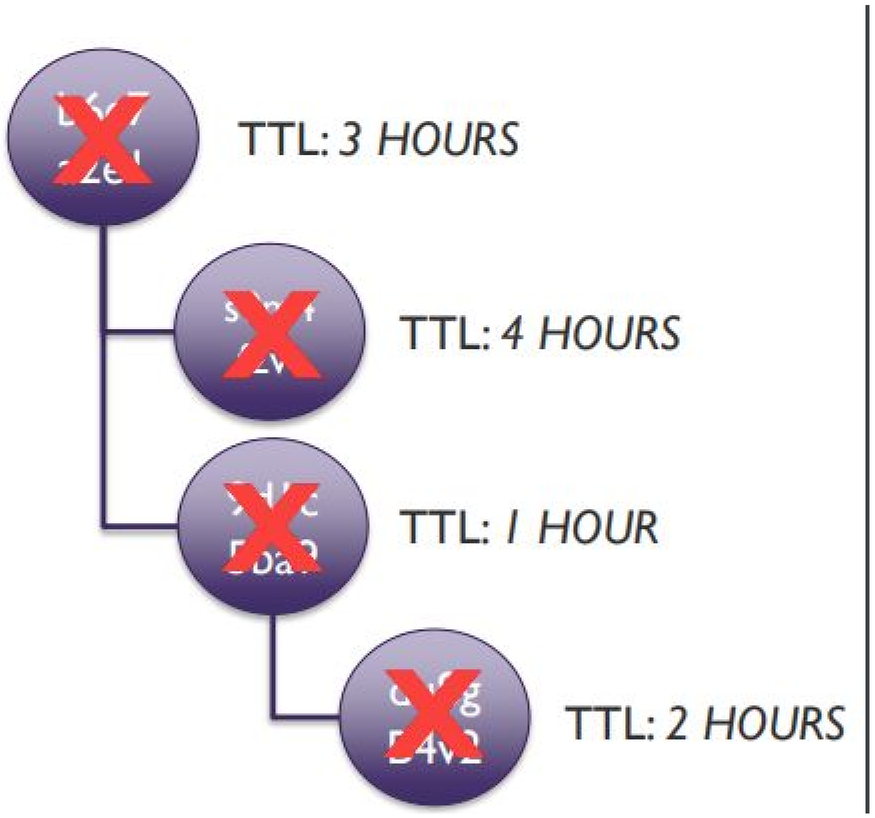
Tokens:

- Tokens are the core method for authentication within Vault
- Tokens can be used directly or generated by another auth method. Vault verifies identity and then generates a unique token to associate with that identity for future requests
- The CLI and UI automatically attach this token, but the API requires this to be manually done
- A token accessor is created alongside the token and serves as reference to the token. The token accessor can be used to:
  – Lookup token capabilities
  – Lookup token properties
  – Revoke the token

- Non-root tokens have an associated TTL (time-to-live)
- After the TTL is expired, the token is revoked
- Tokens can be renewed, if permitted
- Tokens can have a Max TTL, which is a hard limit on the life of the token itself.

Token Hierarchy :

- When a new token is created, it is the child of the creator
- If the parent is revoked or expired, so do all its child tokens
- Parent is almost always a token
- Child can be a token, secret, or auth created by parent



Userpass Auth Method:

- Not seen in the enterprise but frequently used for proof of concepts and demos
- Starts up a local database of users and passwords
- Cannot read usernames and passwords from an external source

AppRole Auth Method:

Ideal auth method for machine to machine communication. Most likely will be used by applications in your environment.

AppRole is like a username & password, but for machines

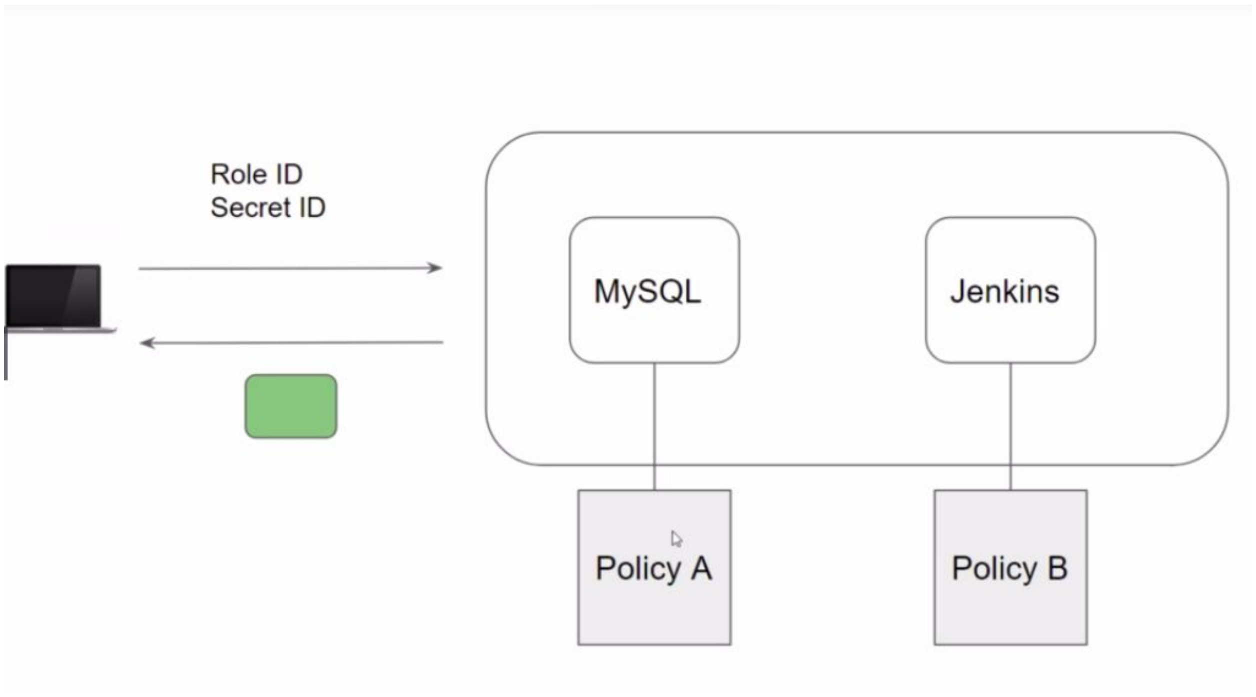
– RoleID – identifiers for our specific role (web, app, etc)

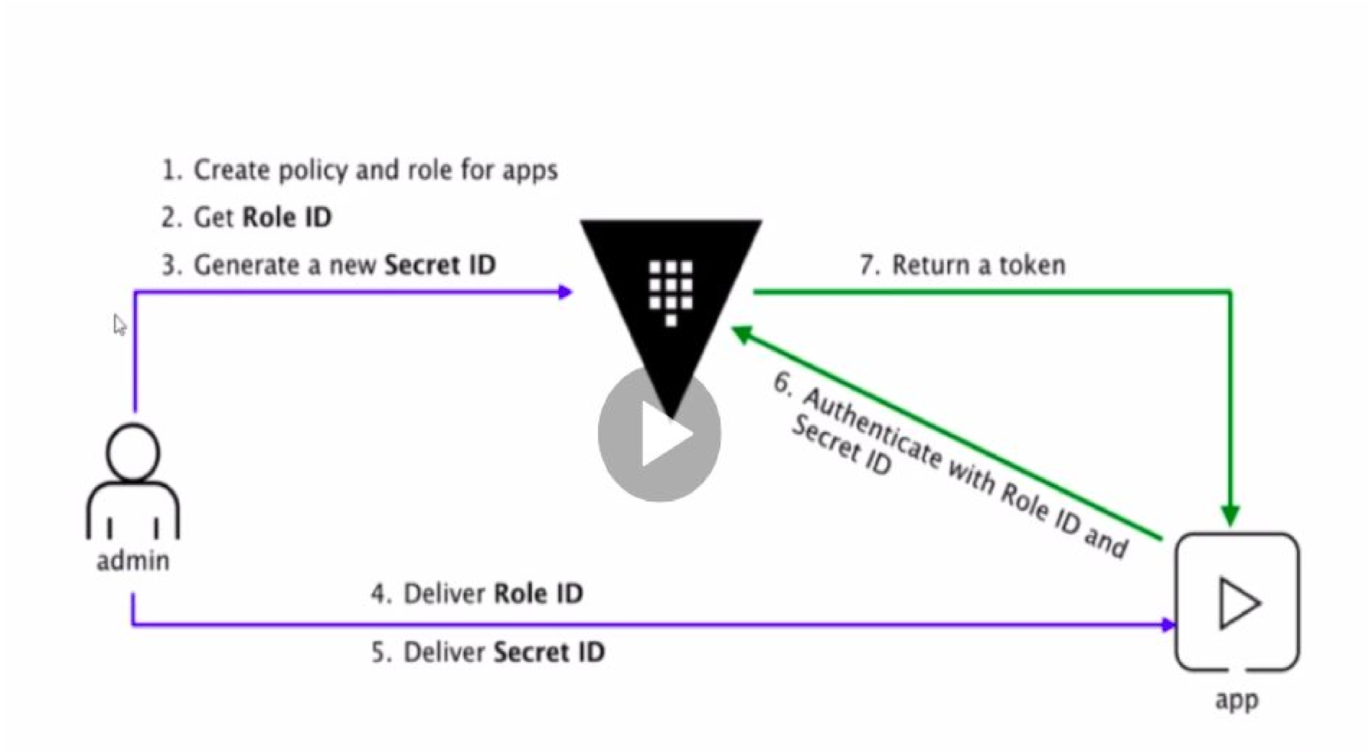
  a) Not considered sensitive
  b) Can be embedded in an AMI, code, Dockerfile, etc

– SecretID – credential required for any login

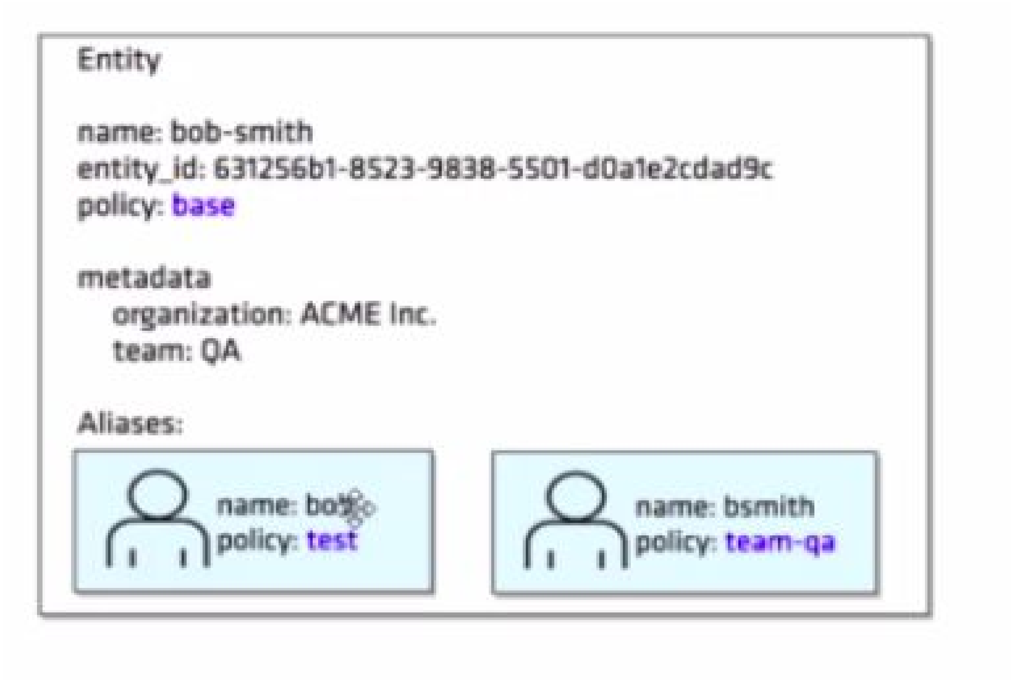
  a) Value is considered sensitive and should be unique to each client
  b) Can be delivered via orchestrator, configuration management

– Put them together: RoleID + SecretID = Vault token




1. Create policy and role for apps
2. Get Role ID
3. Generate a new Secret ID
7. Return a token
6. Authenticate with Role ID and Secret ID
admin
4. Deliver Role ID
5. Deliver Secret ID
app

Entity and Aliases:


Entity

name: bob-smith
entity_id: 631256b1-8523-9838-5501-d0a1e2cdad9c
policy: base

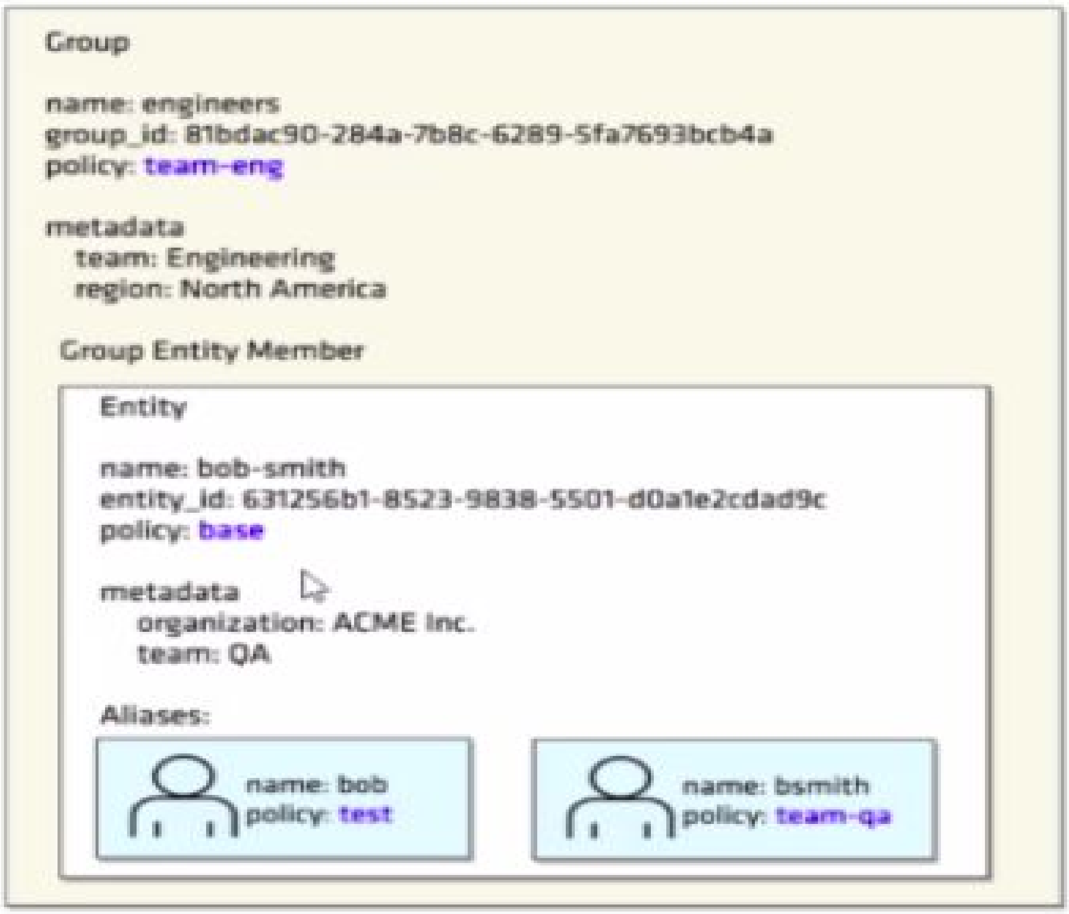
metadata
organization: ACME Inc.
team: QA

Aliases:

name: bob
policy: test

name: bsmith
policy: team-qa

Identity Groups:



Group

name: engineers
group_id: 81bdac90-284a-7b8c-6289-5fa7693bcb4a
policy: team-eng

metadata
  team: Engineering
  region: North America

Group Entity Member

Entity

name: bob-smith
entity_id: 631256b1-8523-9838-5501-d0a1e2cdad9c
policy: base

metadata
  organization: ACME Inc.
  team: QA

Aliases:

name: bob
policy: test

name: bsmith
policy: team-qa