

# Token Authentication Background

Mine Altunay, Dave Dykstra  
October 6, 2021  
FIFE Token Task Force Meeting

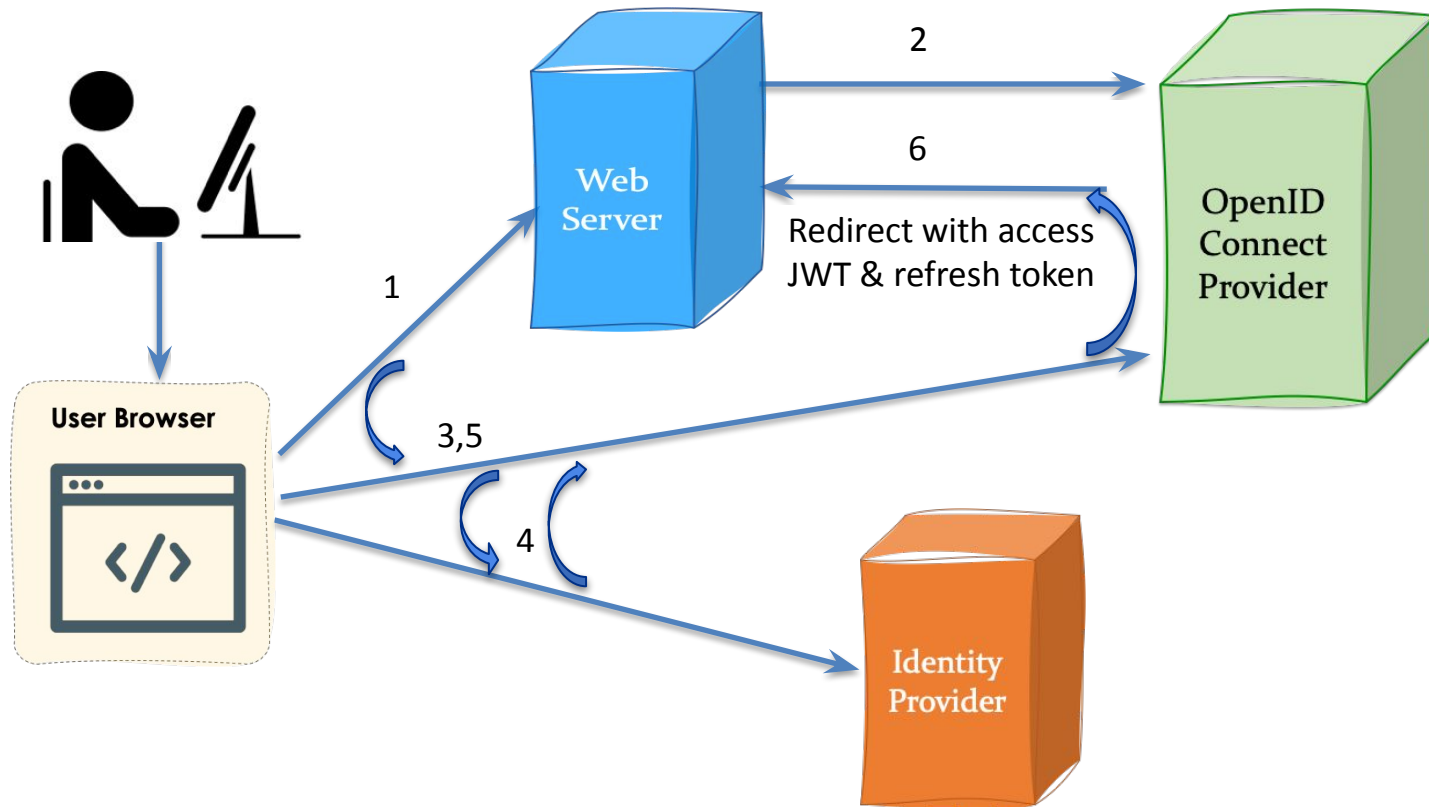
# Why Switch to Tokens?

- The primary reason to switch to tokens is that X.509 proxies were never used outside of the grid community
  - They were invented by Globus, and Globus has abandoned support for the libraries. OSG and a few others have taken up support in the “Grid Community Toolkit” but OSG is dropping their support too
  - X.509 user certificates depended on support at the SSL/TLS layer that is only rarely used
- OAuth2/OpenID Connect (OIDC) JSON Web Tokens (JWTs) are in very wide-spread use, and they are more secure because they enable much more fine grained control
  - There are a lot of existing tools that we can use with them, although we also often need some customization
  - They’re much easier to use because they are sent at a higher layer, e.g. http Bearer header
  - Fine grained control does make them more complicated to use
- Note: X.509 host certificates are not going away, and they are an essential component to securely verifying JWTs over https

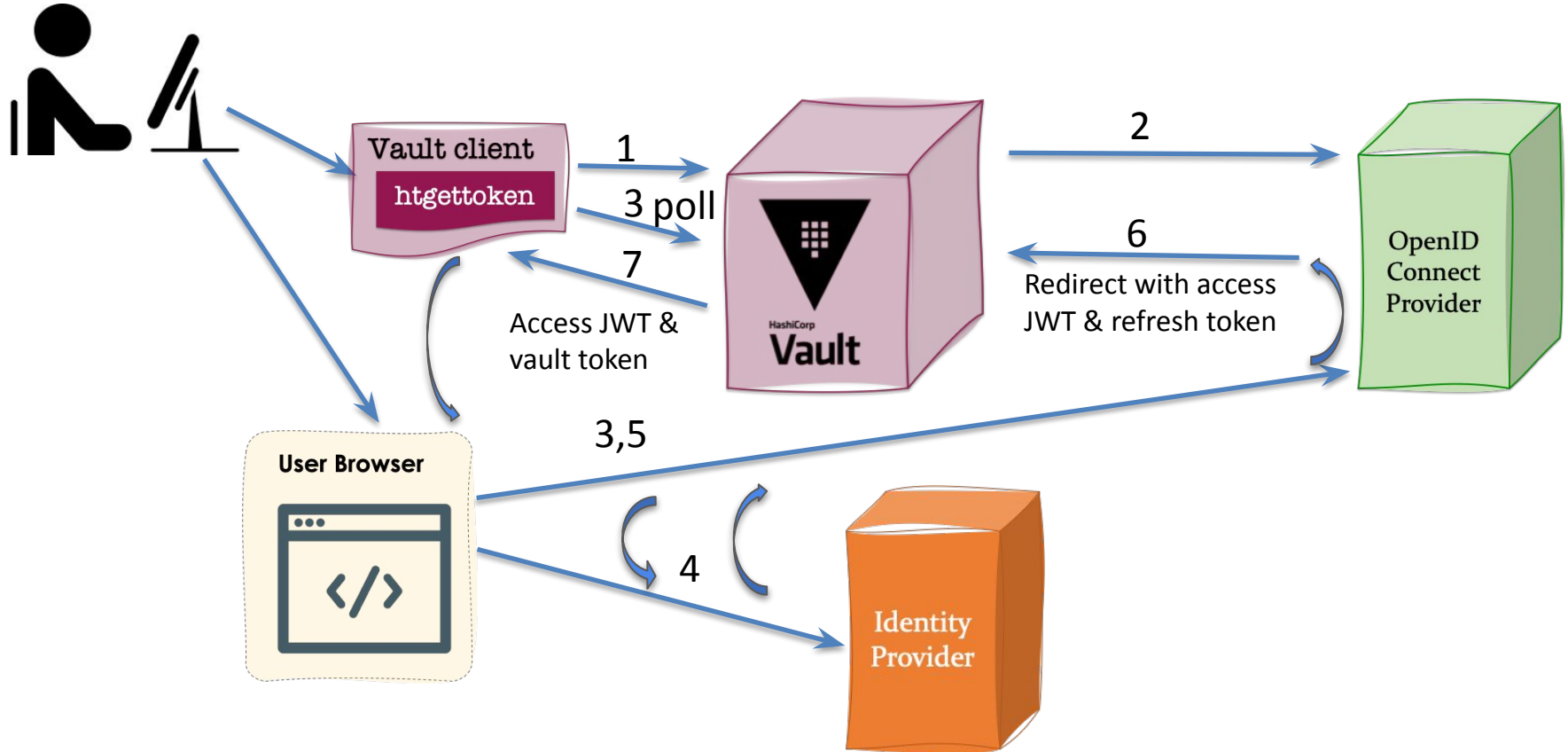
# Why use Vault?

- OAuth2/OIDC is designed for use in web browsers & web servers, but many of our tools are based on command line
- After initial web browser approval, OAuth2 usually uses a “refresh token” that can be indefinitely renewed, so it has high security value and needs protection
- The available command line tool oidc-agent (from the science research community, at KIT) is not user friendly enough at protecting refresh token
- We needed a server-based solution, analogous to MyProxy in our old architecture
- Hashicorp Vault is a very popular open source generic secret store server, that already supports OIDC and Kerberos
  - Needed some slight additions, submitted as PRs
  - Very flexible plugin architecture, REST/JSON API, and flexible access policies
  - Issues its own tokens for very flexible access to particular paths in its filesystem-like space
  - Needed configurator (htvault-config), and a new command line client (htgettoken) to control the flows

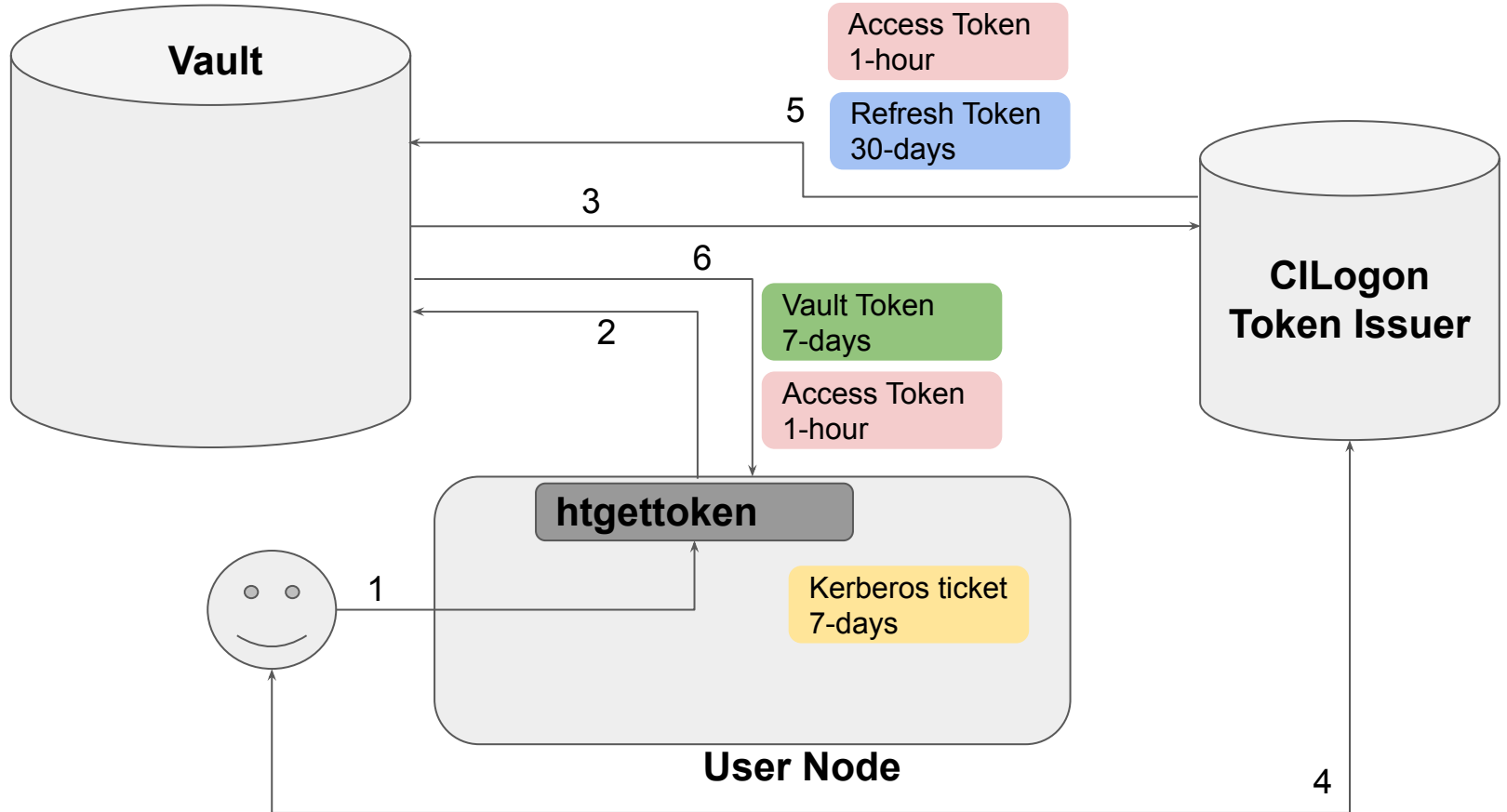
# Normal federated OIDC flow



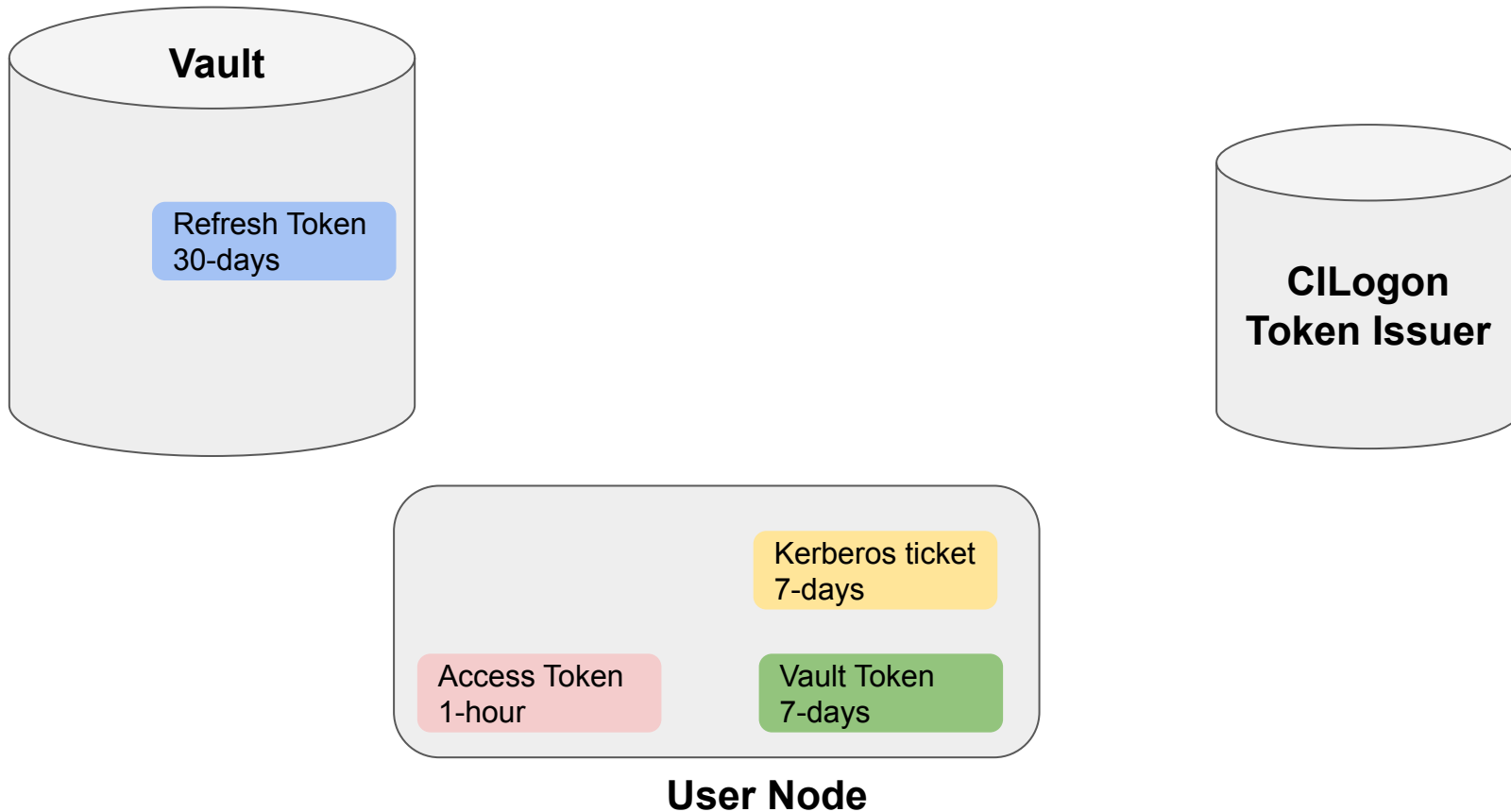
# htgettoken with Vault initial OIDC flow



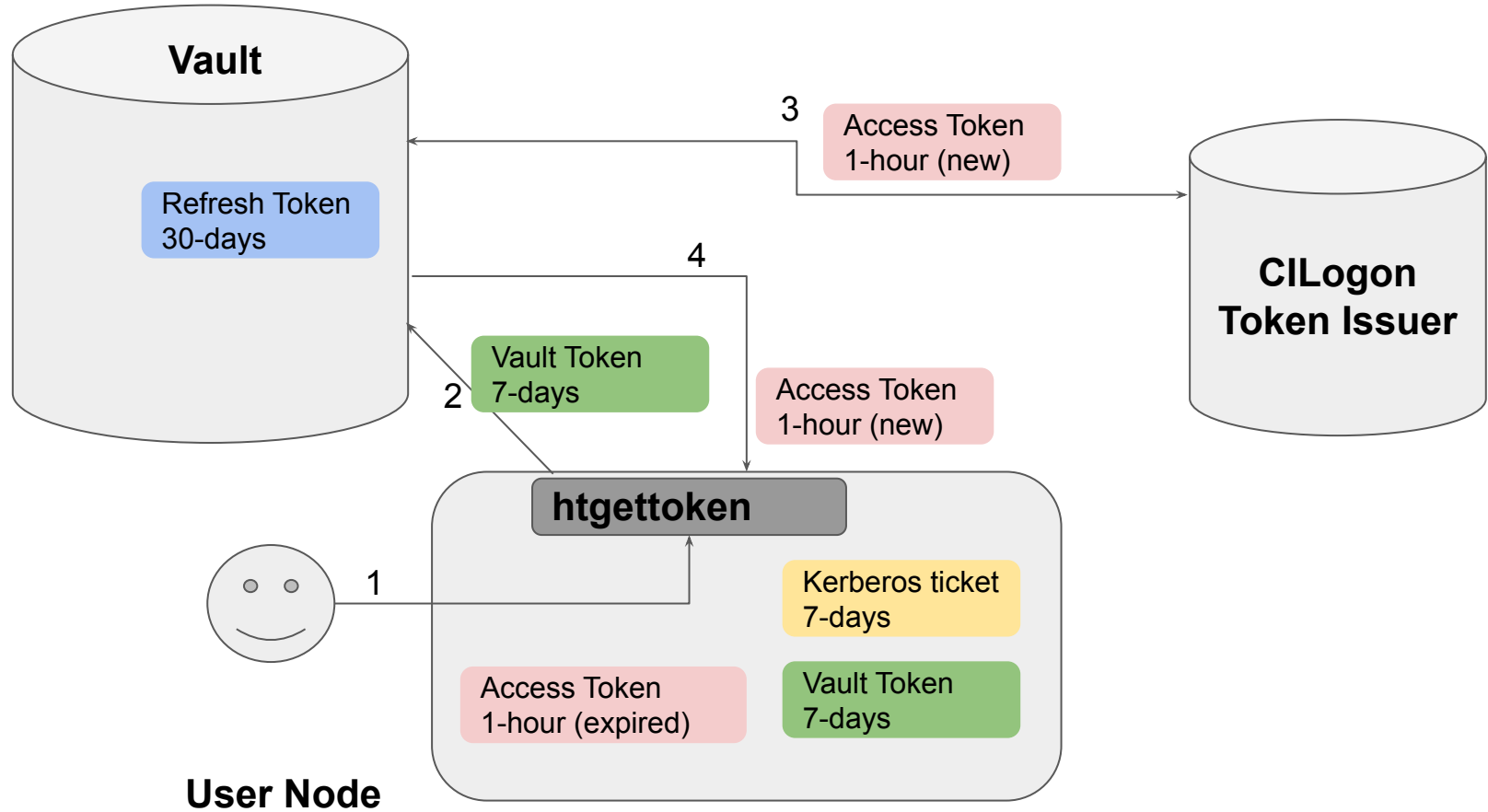
# Initial Authentication (Case 0)



# Credential Lifetimes and Storage Overview

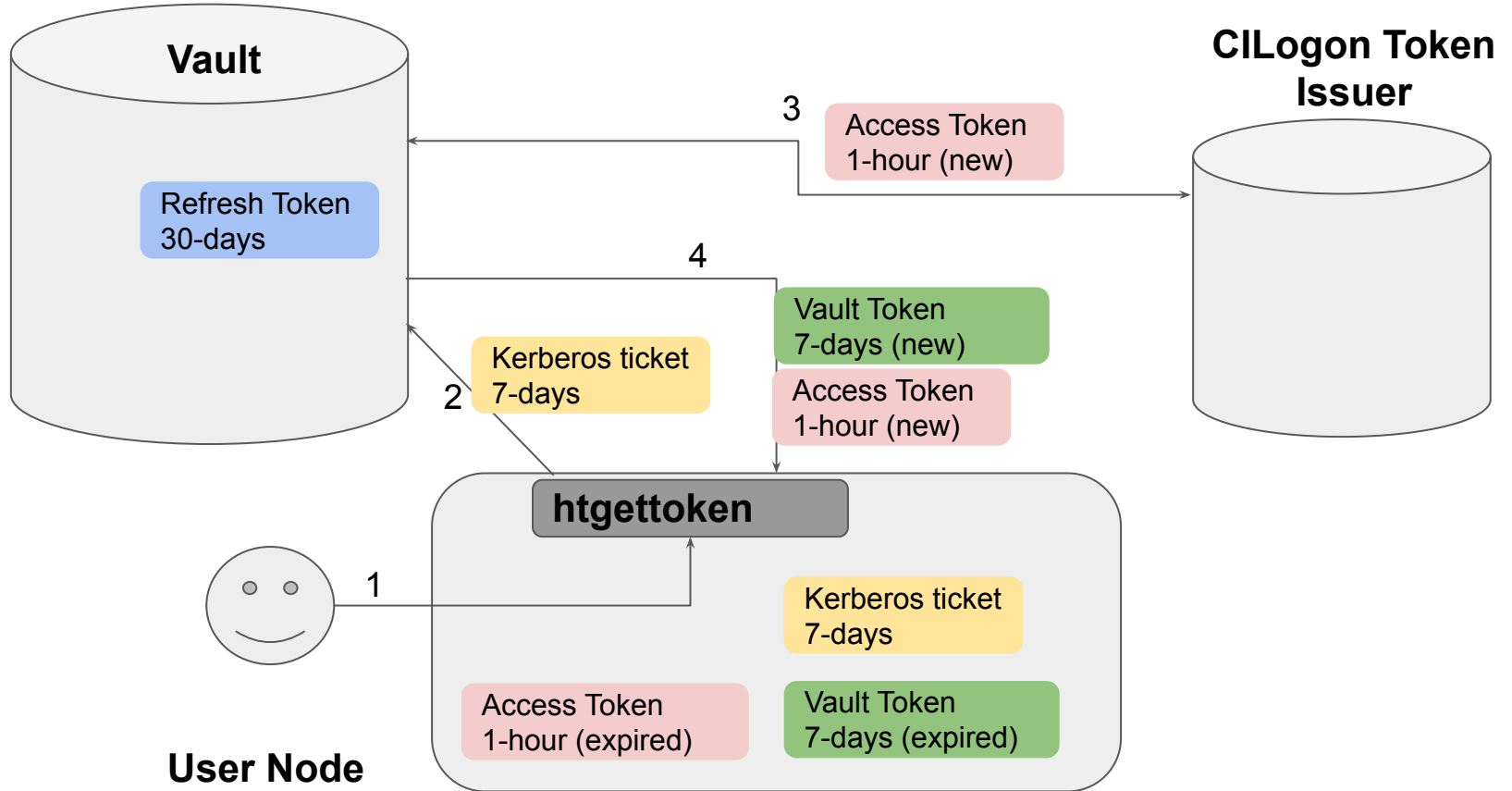


# Authentication with Vault Token (within 7 days, Access Token Expired, Case 1)





# Authentication with Kerberos (Vault token and Access token both expired, Case 2)



## **Authentication after Refresh Token Expires (A month of No Activity, Case 3)**

- As long as a user keeps using htgettoken, it will keep renewing its Access Token, Vault Token and the Refresh Token before expiring.
- If a user has no activity for a month, the Refresh Token expires.
- After the Refresh Token expires, the user must authenticate just like the initial authentication Case 0.

# htgettoken files

- htgettoken normally uses 3 files:
  1. A “credkey” which is an index for the credential, a portion of the path in the vault “filesystem” where the refresh token is stored.
    - Comes from the token issuer
    - Normally for fermilab it simply matches your user id
    - Stored in home directory when a new refresh token is issued, can be shared across client machines, per issuer and per role
  2. A vault token, stored in /tmp/vt\_u\$(id -u) by default
    - Not in home directory because it is sensitive for security, but not in per-session space so it can be reused across login sessions
  3. A bearer, or access token, stored in \${XDG\_RUNTIME\_DIR:-/tmp}/bt\_u\$(id -u) by default
    - Specified by WLCG Bearer Token Discovery standard
    - Defaults to \$XDG\_RUNTIME\_DIR, managed by systemd, which goes away after user completely logs out of a machine (but shared between multiple logins at once)

# Example with htgettoken -v

```
$ env|grep HTG
HTGETTOKENOPTS=--web-open-command=xdg-open
$ htgettoken -v -a fermicloud543.fnal.gov -i dune
Attempting OIDC authentication with https://fermicloud543.fnal.gov:8200
```

Complete the authentication via web browser at:

```
https://cilogon.org/device/?user_code=ZJL-CP9-KZG
```

Running 'xdg-open' on the URL

Waiting for response in web browser

Storing vault token in /tmp/vt\_u3382

Saving credkey to /nashome/d/dwd/.config/htgettoken/credkey-dune-default: dwd

Saving refresh token to https://fermicloud543.fnal.gov:8200

at path secret/oauth-dune/creds/dwd:default

Getting bearer token from https://fermicloud543.fnal.gov:8200

at path secret/oauth-dune/creds/dwd:default

Storing bearer token in /run/user/3382/bt\_u3382

# Example decode

```
$ httokencode
{
  "wlcg.ver": "1.0",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "sub": "dwd@fnal.gov",
  "nbf": 1633465577,
  "scope": "storage.create:/dune/scratch/users/dwd compute.create compute.read
compute.cancel compute.modify storage.read:/dune",
  "iss": "https://cilogon.org/dune",
  "exp": 1633469182,
  "iat": 1633465582,
  "wlcg.groups": [
    "/dune"
  ],
  "jti":
  "https://cilogon.org/oauth2/4a5c03b2a93e4e118b27cb23c1e68a17?type=accessToken&
ts=1633465582430&version=v2.0&lifetime=3600000"
}
```

# CILogon as our token issuer

- We have arranged with CILogon to be our token issuer
- FERRY has been updated to store data about our users in an LDAP server that CILogon hosts, and CILogon uses that information to issue tokens
  - Lists which users are allowed with which VOs and Roles
  - FERRY defines “capabilitysets” to indicate which scopes to include for each VO and Role
  - Vault is configured with corresponding VOs and Roles and just asks for the right capabilityset, and CILogon issues the token
- JWTs are verified by looking up well-known url under the “iss” claim, e.g.
  - <https://cilogon.org/fermilab/.well-known/openid-configuration>
- Under there is a lot of information about the issuer including public signing keys

# Support for “robot” (unattended) operation

- htgettoken supports use of robot kerberos credentials to get new vault tokens
  - Robot kerberos credentials are long lived
  - Principals are in the form “user/purpose/machine.name”
    - “user” can also be a group login, for example “dunepro”
    - In fact, we have configured all our shared roles by default to store refresh tokens in vault under the group name, but that can be overridden by FERRY
  - User (or authorized user for a group) does OIDC authentication once but specifies `htgettoken --credkey` option matching Kerberos principal to store refresh token in subpath under the user’s Vault secrets path
    - The same `htgettoken` command can be used with robot Kerberos credentials
    - This gets used instead of the `credkey` file

# Managed Token Service

- Working with long-lived kerberos keytabs can be tricky, and they are high value from a security perspective since they don't expire
- For that reason, we are planning a FIFE Managed Token Service analogous to the FIFE managed proxy service
- Will push access tokens to experiment machines, and possibly vault tokens, and will push vault tokens to HTCondor



# HTCondor integration

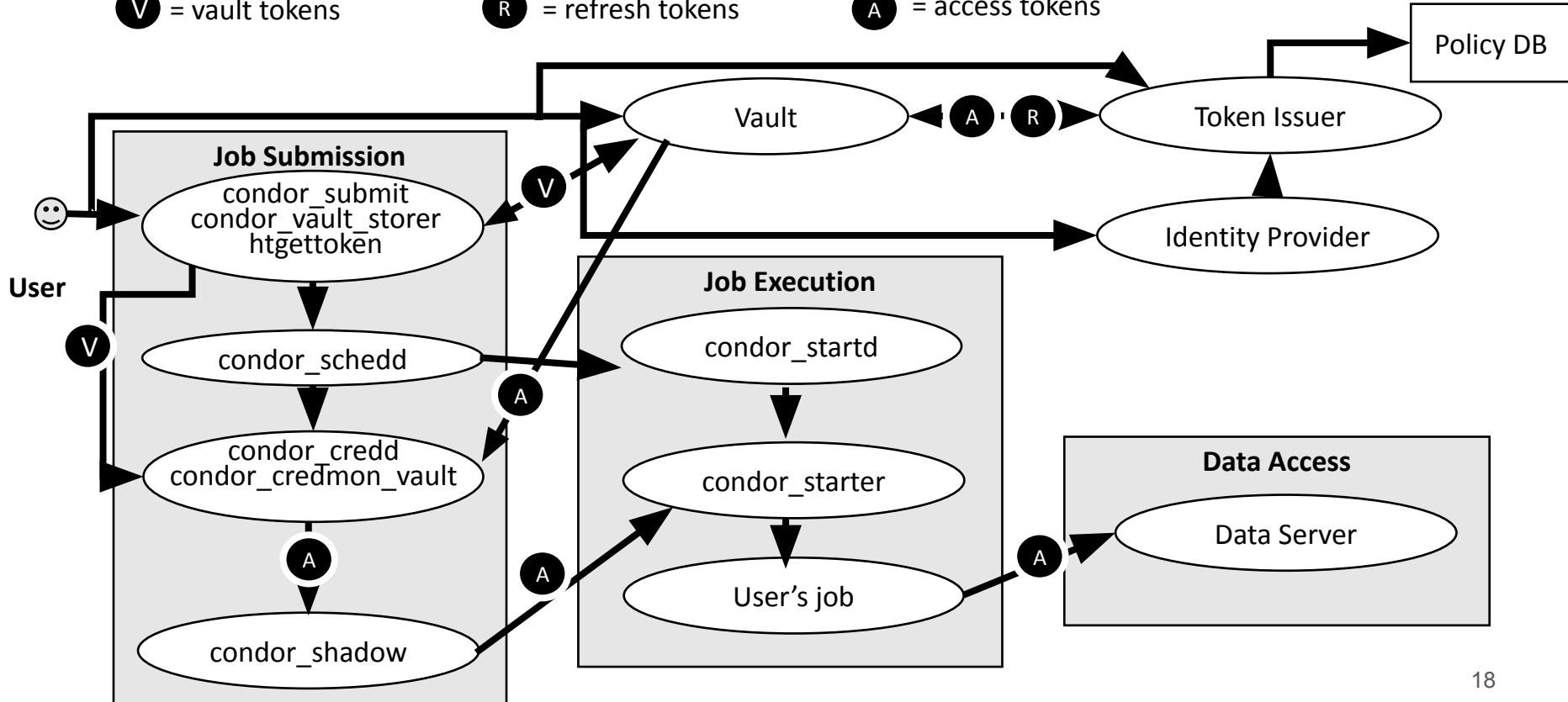
- htgettoken and Vault have been integrated into HTCondor
  - condor\_submit can be configured to automatically invoke htgettoken as needed and store a vault token in condor\_credd
    - Vault token used by condor\_credmon\_vault to get new short-lived access tokens pushed to jobs
    - Vault token is extra long, 4 weeks, in order to work with jobs that are queued for a long time
      - Corresponds to time of proxies stored in MyProxy
  - Submit file specifies issuer, optional role, and optionally can choose reduced audience and/or scopes
    - May obtain more than one token for a job
    - Based on previous implementation of OAuth2 credential support
  - Vault token is stored with an extension indicating the VO & role, so can keep a variety on same machine
  - Available in HTCondor's builds of 9.0.6+ and 9.1.5+
    - Also available in all OSG builds of htcondor-9.0+

# Token flow with HTCondor and Vault

**V** = vault tokens

**R** = refresh tokens

**A** = access tokens



# HTCondor configuration

- System admin:
  - Install condor-credmon-vault rpm and set for example:  
`SEC_CREDENTIAL_GETTOKEN_OPTS = -a fermicloud543.fnal.gov`
- User submit file for example:  
`use_oauth_services = dune`  
`dune_oauth_permissions = storage.read:/ #optional`  
`dune_oauth_resource = https://eos.cern.ch #optional`
- Service names may include role, such as `cms_production`
- Handles may appended to store multiple variations for each service:  
`dune_oauth_permissions_readonly = storage.read:/`  
`dune_oauth_permissions_write = storage.write:/`
- All tokens end up in `$_CONDOR_CREDS`

# Links

- WLCG Authorization Working Group client tools investigation report
  - <https://github.com/WLCG-AuthZ-WG/client-tools>
- Bearer token discovery:
  - <https://github.com/WLCG-AuthZ-WG/bearer-token-discovery>
- WLCG JWT profile
  - <https://github.com/WLCG-AuthZ-WG/common-jwt-profile>
- Vault & plugins
  - <https://www.vaultproject.io/>
  - <https://github.com/hashicorp/vault-plugin-auth-jwt>
  - <https://github.com/puppetlabs/vault-plugin-secrets-oidc>
- htvault-config: <https://github.com/fermitools/htvault-config>
- htgettoken: <https://github.com/fermitools/htgettoken>
- Htcondor with vault docs: <https://htcondor-vault.readthedocs.io>
- oidc-agent: <https://indigo-dc.gitbook.io/oidc-agent/>