

OAuth2 HTTP file transfers at IceCube

HTCondor Week 2022
David Schultz



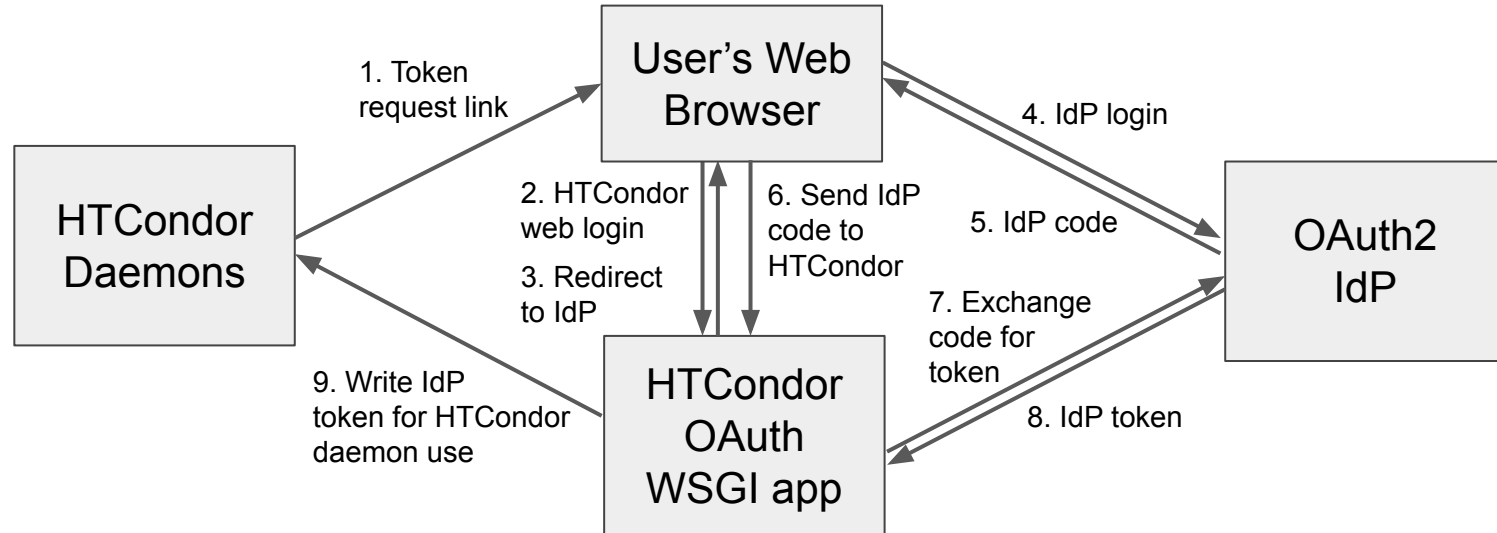
Overview

- OAuth2 token-based file transfer mechanism
 - » Admin configuration
 - » User job submission
- Nginx support for OAuth2
- IceCube's experience

OAuth2 principles

Getting a token

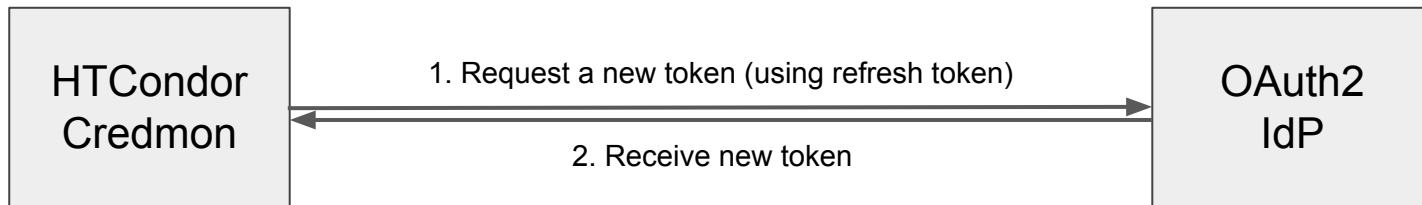
- HTCondor needs an initial token from an OAuth2 identity provider (IdP)
- This is web-based, using a redirect to the actual login page of the IdP



OAuth2 principles

Refreshing a token

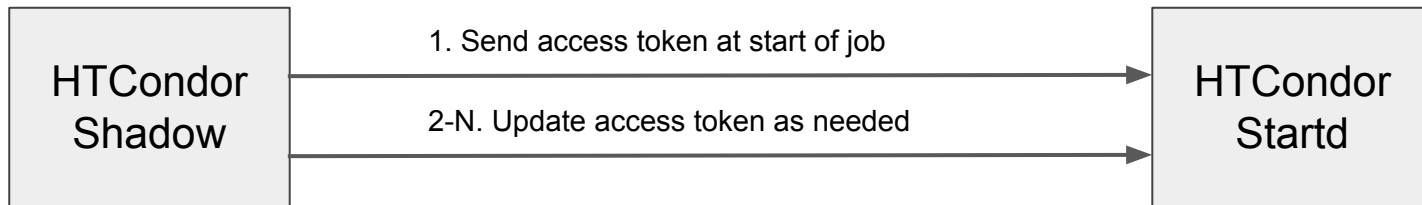
- HTCondor holds a refresh token - lifetime can be extended
- Must be exchanged for a short-lived access token to do transfers
- Avg lifetime of tokens:
 - Refresh - days to months
 - Access - minutes to hours



OAuth2 principles

Sending a token with a job

- In order to use OAuth2 file transfers, the access token needs to be sent with the job to the startd
 - This happens during job startup, and also throughout the lifetime of the job
 - An access token may expire before a job is finished, so it continually gets updated
- Note: the refresh token is never sent to the job / startd



OAuth2 admin configuration

Initial setup - [HTCondor docs](#)

- Install the `condor_credmon_oauth` daemon, Apache, and the WSGI application
 - The `condor-credmon-oauth` rpm has most of this
 - Get the example WSGI app with

```
rpm -ql condor-credmon-oauth | grep "condor_credmon_oauth\.conf"
```

and place it in the apache web directory

- Register HTCondor as a client in your IdP and add to HTCondor config:

```
use feature: oauth
MYIDP_CLIENT_ID = htcondor_client
MYIDP_CLIENT_SECRET_FILE = /etc/condor/.secrets/myidp_client_secret
MYIDP_AUTHORIZATION_URL = https://myidp.edu/auth
MYIDP_TOKEN_URL = https://myidp.edu/token
MYIDP_RETURN_URL_SUFFIX = /return/myidp # the lowercase version of MYIDP
```

OAuth2 as a user

1. Write submit file using OAuth2

- Tell HTCondor which IdP you are using (must match what the admin configured)
- (optional) specify additional scopes and “handle” name for the token
- Use the token in https file transfers
 - Syntax is <IdP>+https://url or <IdP>.<handle>+https://url
 - Use *transfer_output_remaps* to send output to a url

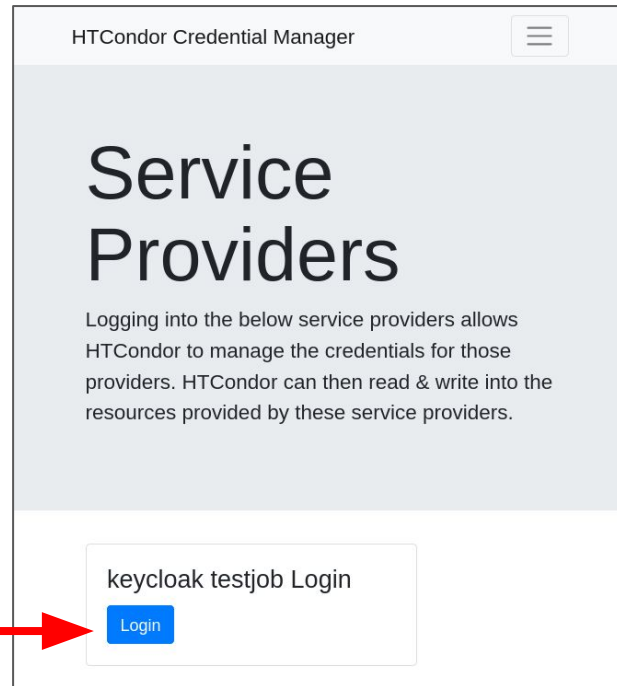
```
# example submit syntax
use_oauth_services = myidp
myidp_oauth_permissions_testjob = write # extra scopes, and defines “service” name
transfer_input_files = myidp.testjob+https://my-storage-server.edu/data/infile.dat
transfer_output_files = outfile.dat
transfer_output_remaps = "outfile.dat = myidp.testjob+https://my-storage-server.edu/data/outfile.dat"
```

OAuth2 as a user

2. Get your token

- HTCondor needs an initial token from your identity provider (IdP)
- This is web-based, using a redirect to the actual login page of the IdP
- After a token is acquired, jobs may be submitted

```
dschultz@condor:~$ condor_submit test.sub
Submitting job(s)
Hello, dschultz.
Please visit:
http://localhost:22280/key/5b2dfca80ec4b5ebce55c40b114c40ab
572903171e37efba065de29a2789999e
```



Using Nginx as an HTTP server

Nginx is a great HTTP server that powers 1/3 of the web

- Free version does not support OAuth2 directly
- Instead, you make a sub-request to a server that can do OAuth2
 - This can be a simple python script, which understands your token format
 - Can also evaluate policy, though note it is not good to do complex logic
 - This needs to be fast, and handle many requests per second
 - Example:
https://github.com/WIPACrepo/keycloak-http-auth/blob/main/keycloak_http_auth/server.py
 - Only validates the token and gets posix uid/gid from the token
- The next step is to make Nginx read/write as the correct user
 - Nginx (with a plugin) supports embedded Lua scripts, and here is one to call setsuid and setfsuid
https://github.com/WIPACrepo/keycloak-http-auth/blob/main/nginx_default.conf#L53

IceCube's experience

IceCube has Keycloak for an IdP

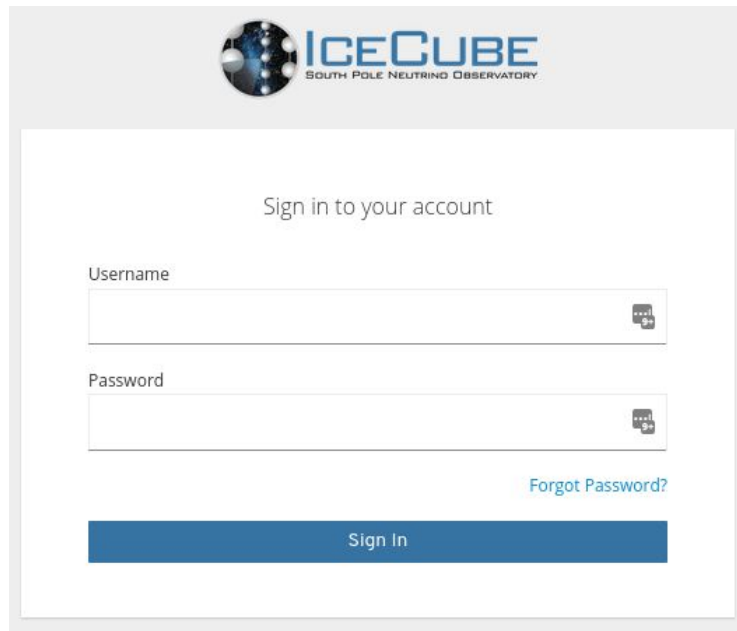
- Recently deployed as a single-sign-on solution, source of truth for user management
- Supports OAuth2

A long history with gridftp

- Have used x509 proxies for data transfer for many years
- Production works mostly well, but users dislike or are confused by x509

Transitioning to HTTP-based transfers

- A long testing phase
- Proof-of-concept took several months to iron out wrinkles



The screenshot shows the login interface for the IceCube South Pole Neutrino Observatory. At the top, there is a logo consisting of a globe with blue dots and the text "ICECUBE" in large blue letters, with "SOUTH POLE NEUTRINO OBSERVATORY" in smaller blue letters below it. Below the logo, the text "Sign in to your account" is centered. There are two input fields: "Username" and "Password". Each field has a small icon on the right side that looks like a key or a lock. Below the password field, there is a link that says "Forgot Password?". At the bottom, there is a blue button with the text "Sign In" in white.

IceCube's experience

Some bumps along the way

- Getting Nginx configured correctly
 - Needed a specific build with the correct plugins to support Lua
 - Ubuntu packages are the easy way, instead of using the Nginx container
 - How to bake in common config and allow overriding only necessary details, like port numbers
 - Now have CI tests and a Docker container with the working formula
- HTCondor curl plugin did not support OAuth “handles”
 - Needed a fix to change from `<IdP>_<handle>` to `<IdP>.<handle>` to make it a valid url (fixed in 9.0.12)
- Reusing tokens in subsequent job submissions is prone to failure
 - Sometimes need to clear stored tokens before resubmitting
 - Advising to use “handles” for each submission to avoid this

IceCube's experience

Deploying Nginx HTTP servers

- Using Kubernetes
 - Nginx + auth sub-request as two containers in a pod
 - Auto-scales with load

Plans for the future

- Deploying for users soon
 - With current syntax for now
 - May try to hide some of the details via job transforms
- Production still has some work
 - How to transfer refresh tokens from production app to HTCondor?
 - Some ideas we need to test

Questions?