HTCondor Security: Philosophy and Administration Changes

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

FEARLESS SCIENCE

“Forget what you know about HTCondor security. We changed it”

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

FEARLESS SCIENCE

Establishing a secure pool

Traditionally, there's been no "easy button" to setup strong security on pools.

- Very easy to setup poor security.
- Very hard to find good advice on strong security.
- No tools provided by HTCondor to setup strong authentication.
  - Is the answer "Google for how to create a new CA with OpenSSL"?

In 2019, we spent significant blood, sweat, and tears providing a new authentication method and new tooling to setup your pool.

Where's my easy button?

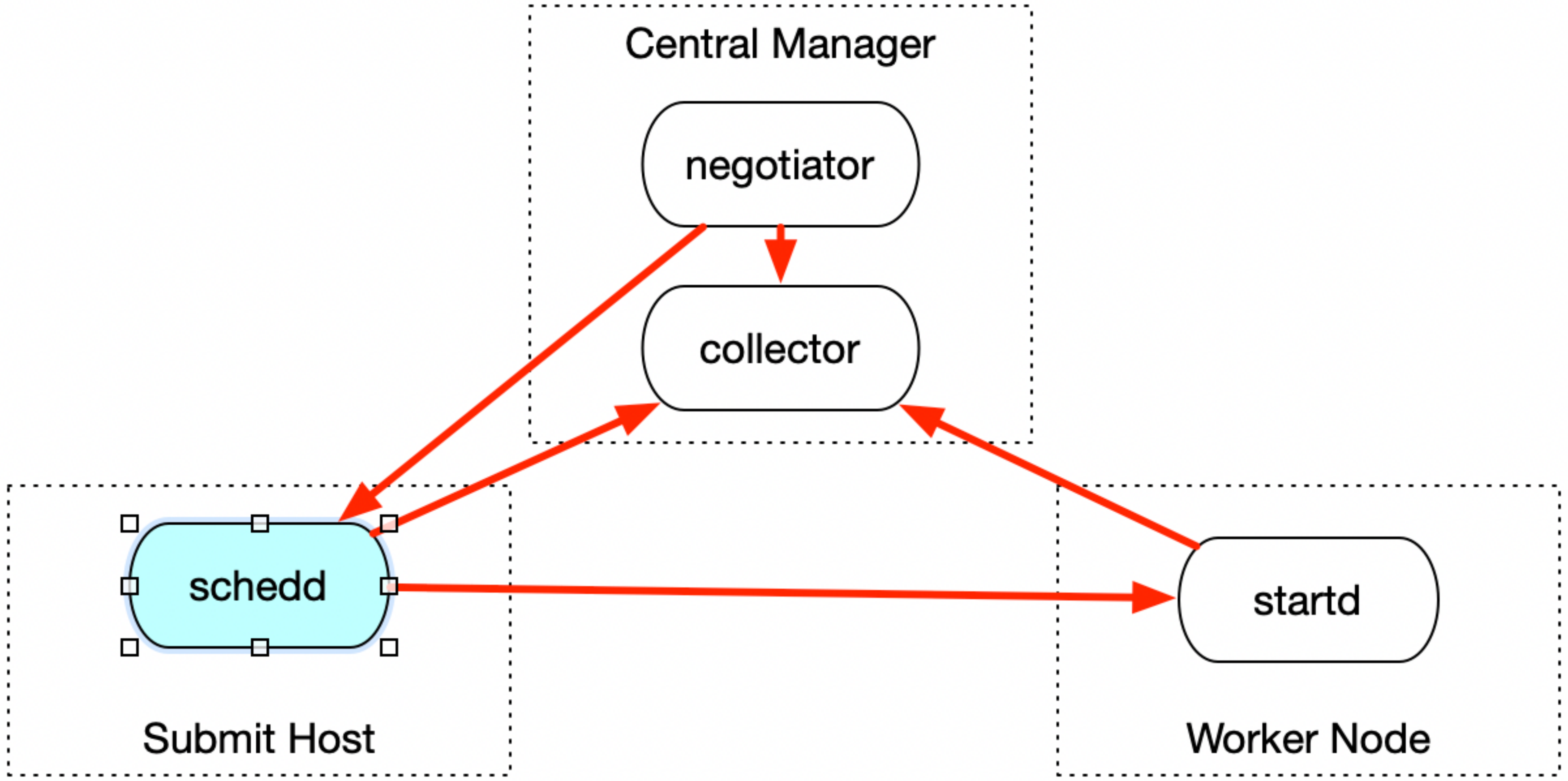
Actual product if anyone is interested…

Classic HTCondor Daemon Security

For invoking a remote command:

- The server and client would negotiate an **authentication** method to establish identities.
  - Example methods: GSI, PASSWORD, SSL.
- Once an **identity** was established, HTCondor would determine if the requested command was **authorized**.
  - Can user foo@example.com perform actions that require DAEMON-level authorization?!

Each pair of daemons that want to talk had to perform this dance!



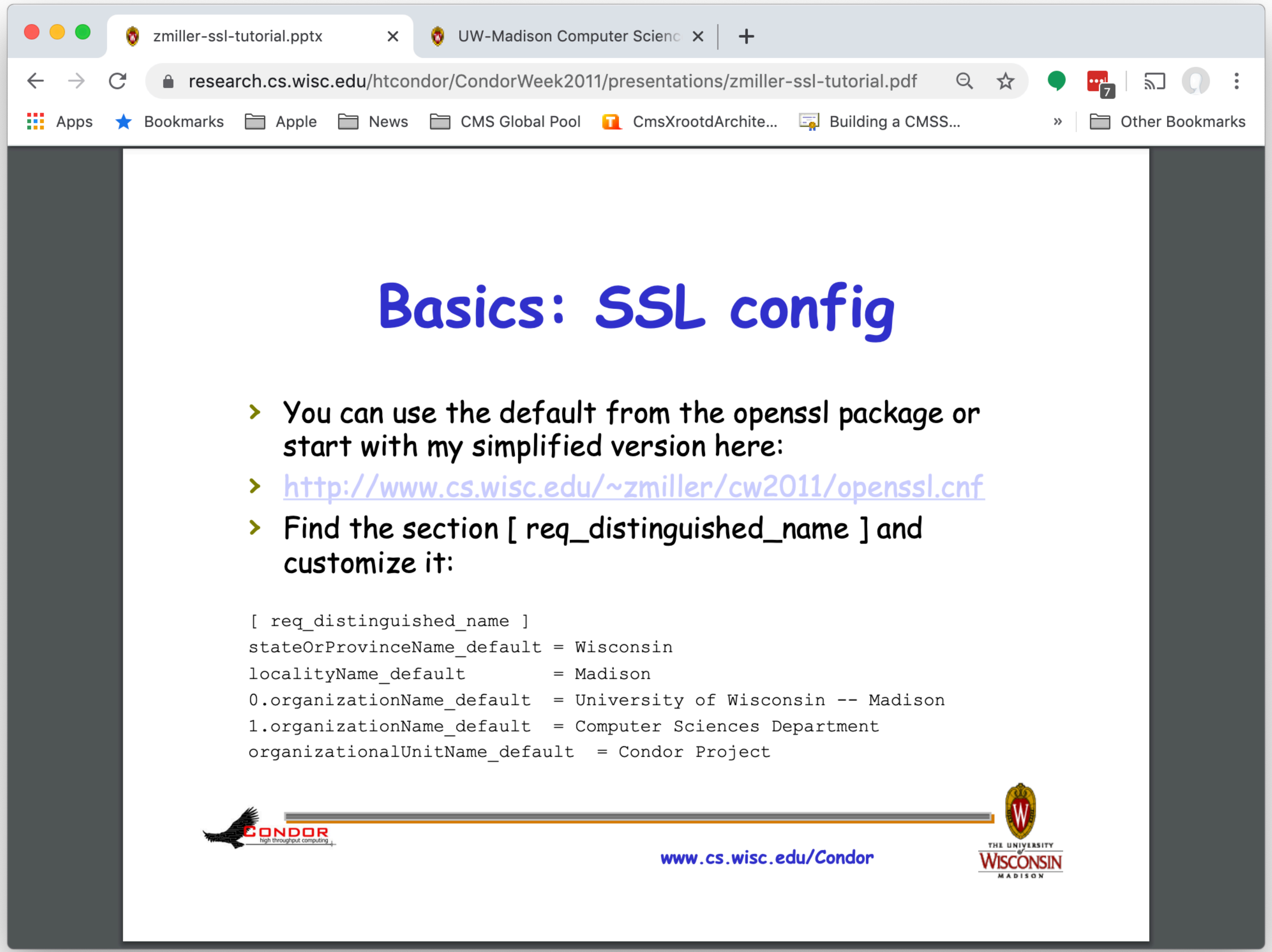
Nobody trusts nobody. All authentication is established from scratch.

How do we setup SSL security for HTCondor?

Step 0. Figure out this is a thing you want to do!

... oodles of old presentations to go through. Which one is right?

Setting up SSL Security

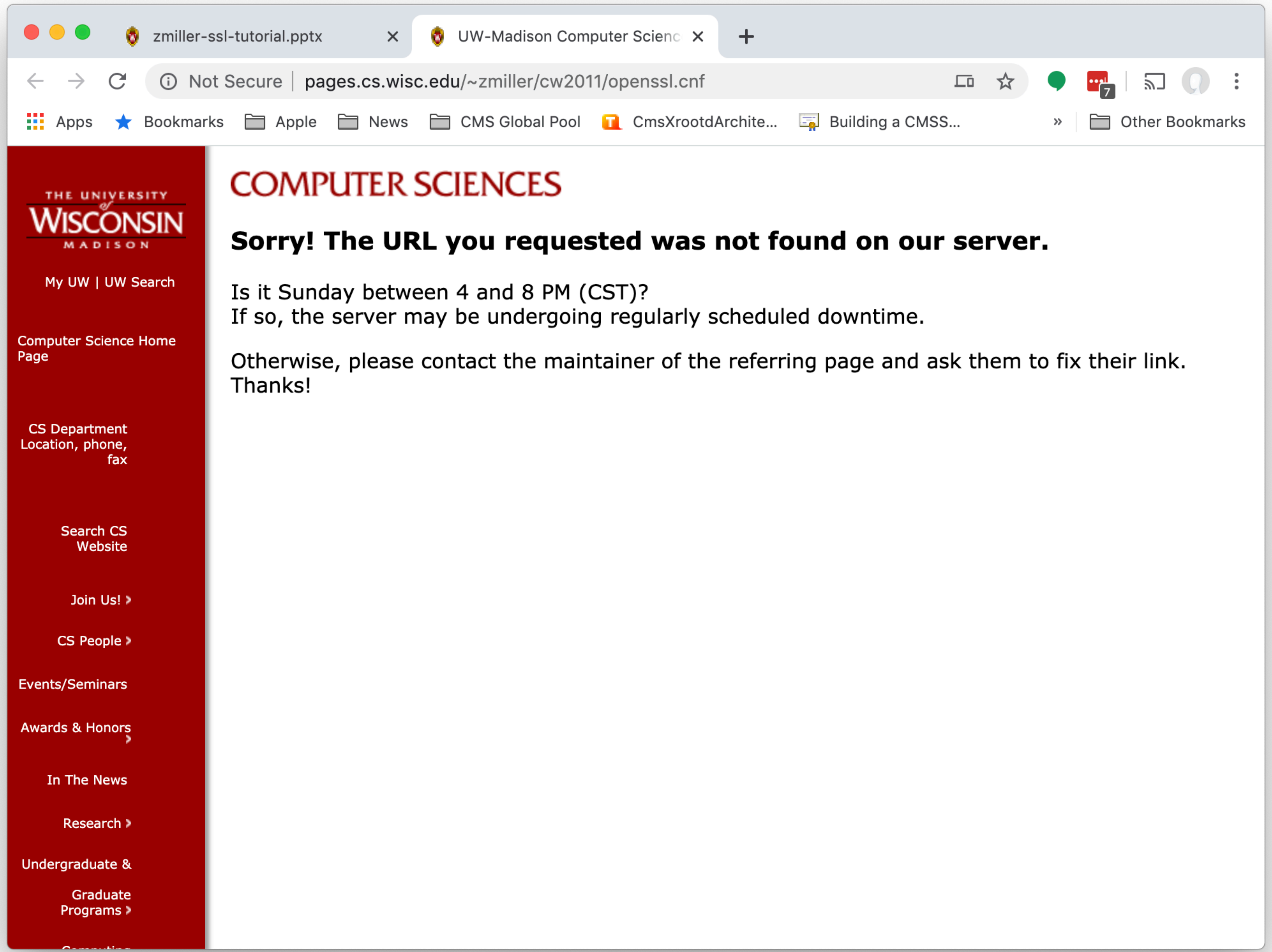

zmiller-ssl-tutorial.pptx
UW-Madison Computer Scienc
research.cs.wisc.edu/htcondor/CondorWeek2011/presentations/zmiller-ssl-tutorial.pdf
Apps
Bookmarks
Apple
News
CMS Global Pool
CmsXrootdArchite...
Building a CMSS...
Other Bookmarks

Basics: SSL config

- You can use the default from the openssl package or start with my simplified version here:
- http://www.cs.wisc.edu/~zmiller/cw2011/openssl.cnf
- Find the section [req_distinguished_name] and customize it:

```
[ req_distinguished_name ]
stateOrProvinceName_default = Wisconsin
localityName_default        = Madison
0.organizationName_default  = University of Wisconsin -- Madison
1.organizationName_default  = Computer Sciences Department
organizationalUnitName_default  = Condor Project
```

www.cs.wisc.edu/Condor


zmiller-ssl-tutorial.pptx
UW-Madison Computer Scienc
Not Secure
pages.cs.wisc.edu/~zmiller/cw2011/openssl.cnf
Apps
Bookmarks
Apple
News
CMS Global Pool
CmsXrootdArchite...
Building a CMSS...
Other Bookmarks
THE UNIVERSITY of WISCONSIN MADISON
My UW | UW Search
Computer Science Home Page
CS Department Location, phone, fax
Search CS Website
Join Us!
CS People
Events/Seminars
Awards & Honors
In The News
Research
Undergraduate & Graduate Programs

COMPUTER SCIENCES

Sorry! The URL you requested was not found on our server.

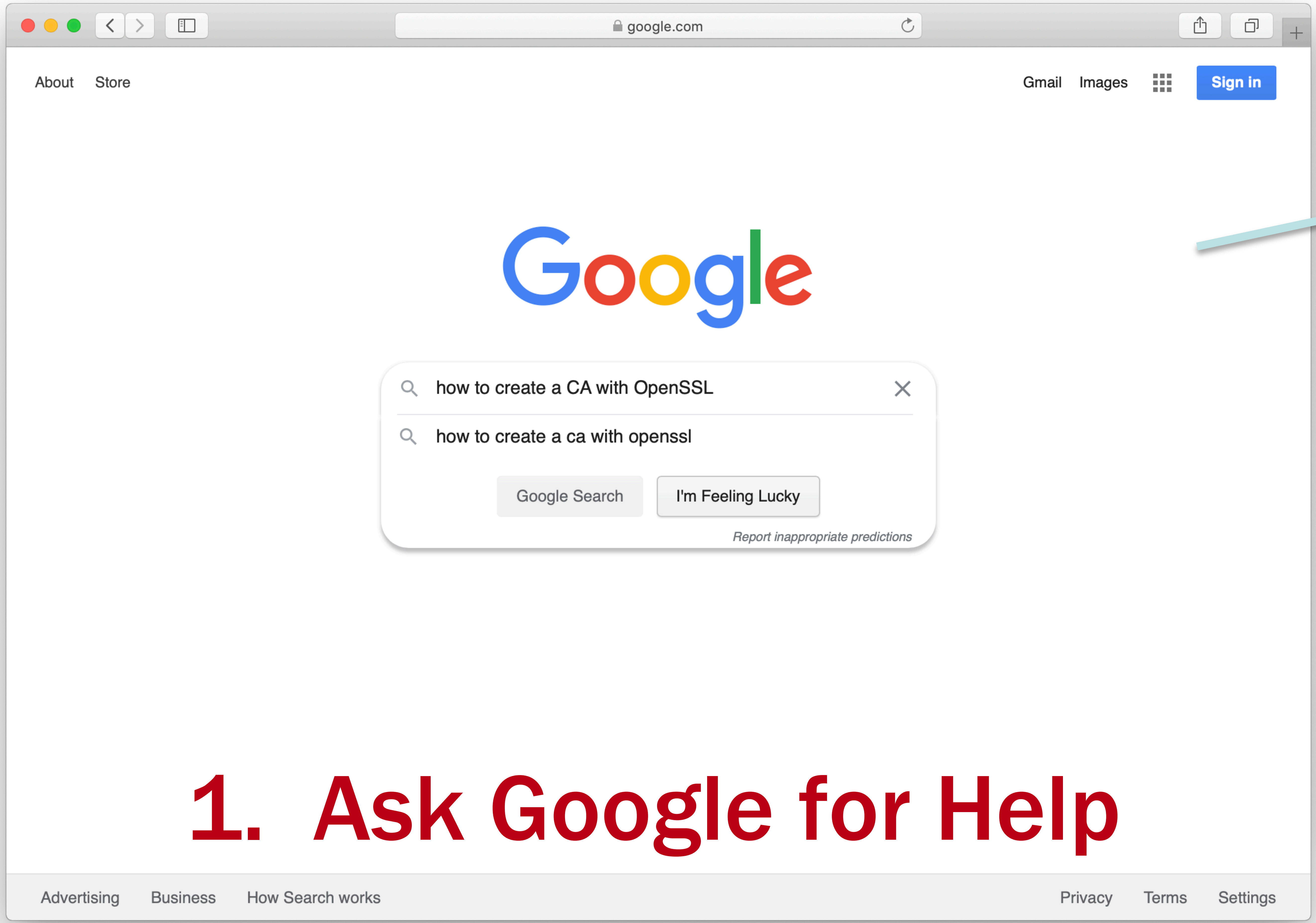
Is it Sunday between 4 and 8 PM (CST)?
If so, the server may be undergoing regularly scheduled downtime.

Otherwise, please contact the maintainer of the referring page and ask them to fix their link.
Thanks!

1a. Find Zach Miller's HTCondor Week 2011 talk.

1b. Whoops

Setting up SSL Security


google.com

About Store Gmail Images Sign in

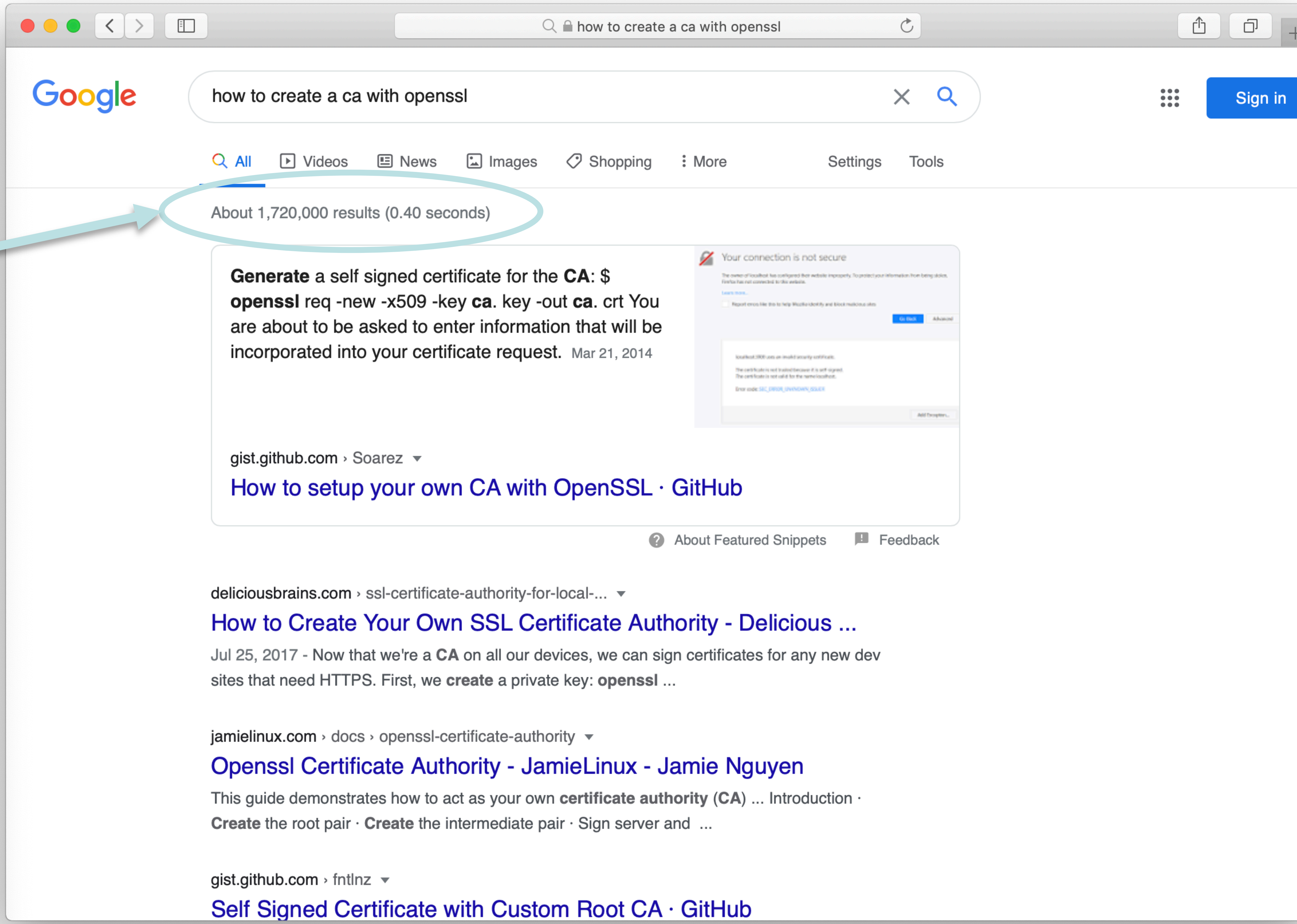
how to create a CA with OpenSSL

how to create a ca with openssl

Google Search I'm Feeling Lucky

Report inappropriate predictions

Advertising Business How Search works Privacy Terms Settings


how to create a ca with openssl

All Videos News Images Shopping More Settings Tools

About 1,720,000 results (0.40 seconds)

Generate a self signed certificate for the **CA**: $ **openssl** req -new -x509 -key **ca**. key -out **ca**. crt You are about to be asked to enter information that will be incorporated into your certificate request. Mar 21, 2014

gist.github.com › Soarez

How to setup your own CA with OpenSSL · GitHub

About Featured Snippets Feedback

deliciousbrains.com › ssl-certificate-authority-for-local-...

How to Create Your Own SSL Certificate Authority - Delicious ...

Jul 25, 2017 - Now that we're a **CA** on all our devices, we can sign certificates for any new dev sites that need HTTPS. First, we **create** a private key: **openssl** ...

jamielinux.com › docs › openssl-certificate-authority

Openssl Certificate Authority - JamieLinux - Jamie Nguyen

This guide demonstrates how to act as your own **certificate authority** (**CA**) ... Introduction · **Create** the root pair · **Create** the intermediate pair · Sign server and ...

gist.github.com › fntlnz

Self Signed Certificate with Custom Root CA · GitHub

1. Ask Google for Help

2. Distribute CA across the cluster

```
# SSL settings
AUTH_SSL_SERVER_CERTFILE=/etc/grid-security/condor/hostcert.pem
AUTH_SSL_SERVER_KEYFILE=/etc/grid-security/condor/hostkey.pem
AUTH_SSL_SERVER_CADIR=/etc/grid-security/certificates
AUTH_SSL_CLIENT_CADIR=/etc/grid-security/certificates
```

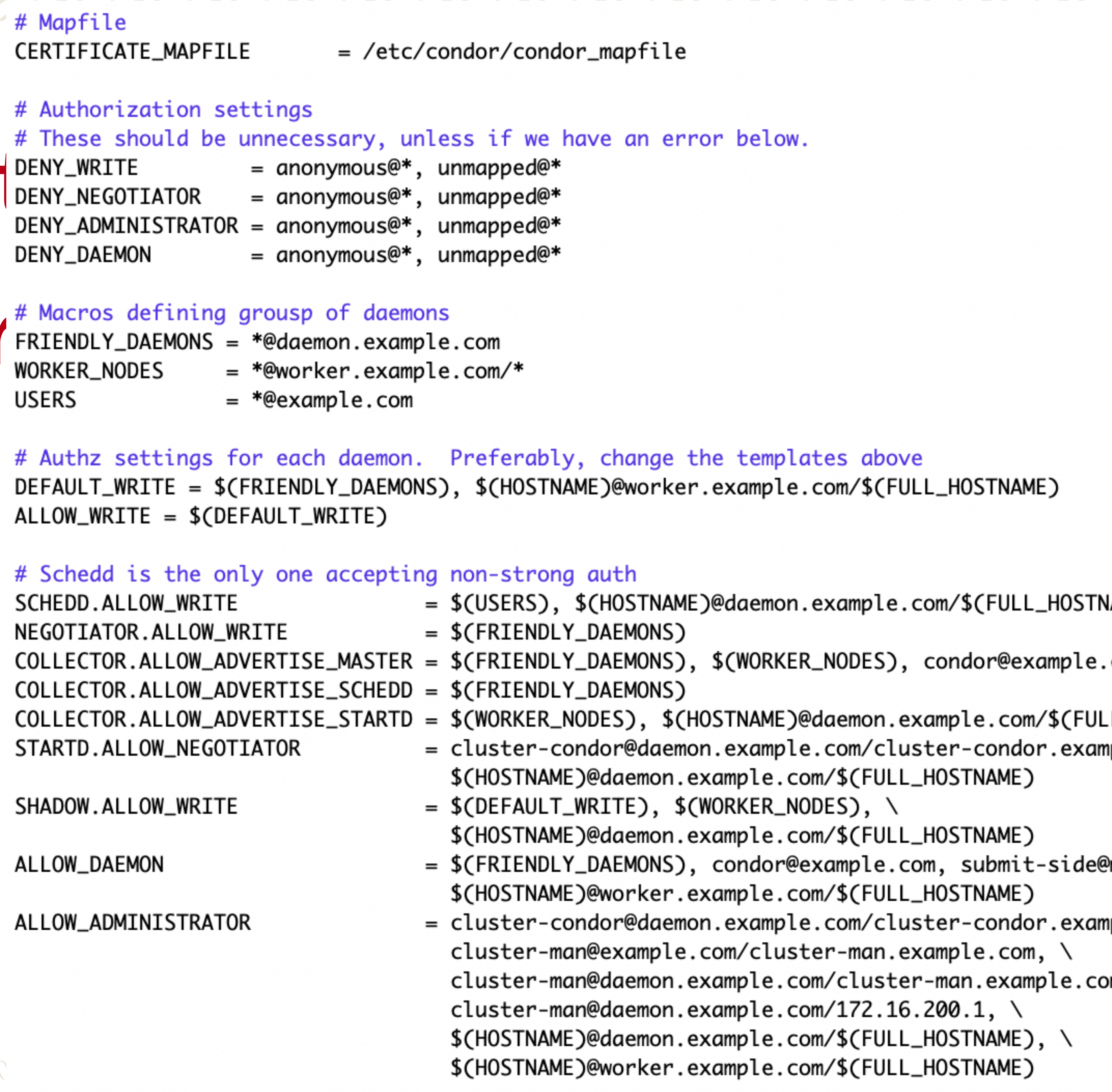
... and add some modest configs

... and distribute host certificates everywhere

```
# Authentication settings
SEC_DEFAULT_AUTHENTICATION                   = REQUIRED
SEC_READ_AUTHENTICATION                      = OPTIONAL
SEC_CLIENT_AUTHENTICATION                    = OPTIONAL
SEC_DEFAULT_AUTHENTICATION_METHODS           = GSI
SCHEDD.SEC_WRITE_AUTHENTICATION_METHODS      = FS,GSI
SCHEDD.SEC_DAEMON_AUTHENTICATION_METHODS     = FS,GSI
SEC_CLIENT_AUTHENTICATION_METHODS            = FS,GSI

# Privacy settings
SEC_DEFAULT_ENCRYPTION = OPTIONAL
SEC_DEFAULT_INTEGRITY  = REQUIRED
SEC_READ_INTEGRITY     = OPTIONAL
SEC_CLIENT_INTEGRITY   = OPTIONAL
SEC_READ_ENCRYPTION    = OPTIONAL
SEC_CLIENT_ENCRYPTION  = OPTIONAL
```

Setting up SSL Security

2. Distribute CA across the clust
3. Configure schedd -> collector
4. startd -> collector auth
5. negotiator -> schedd
6. schedd -> startd

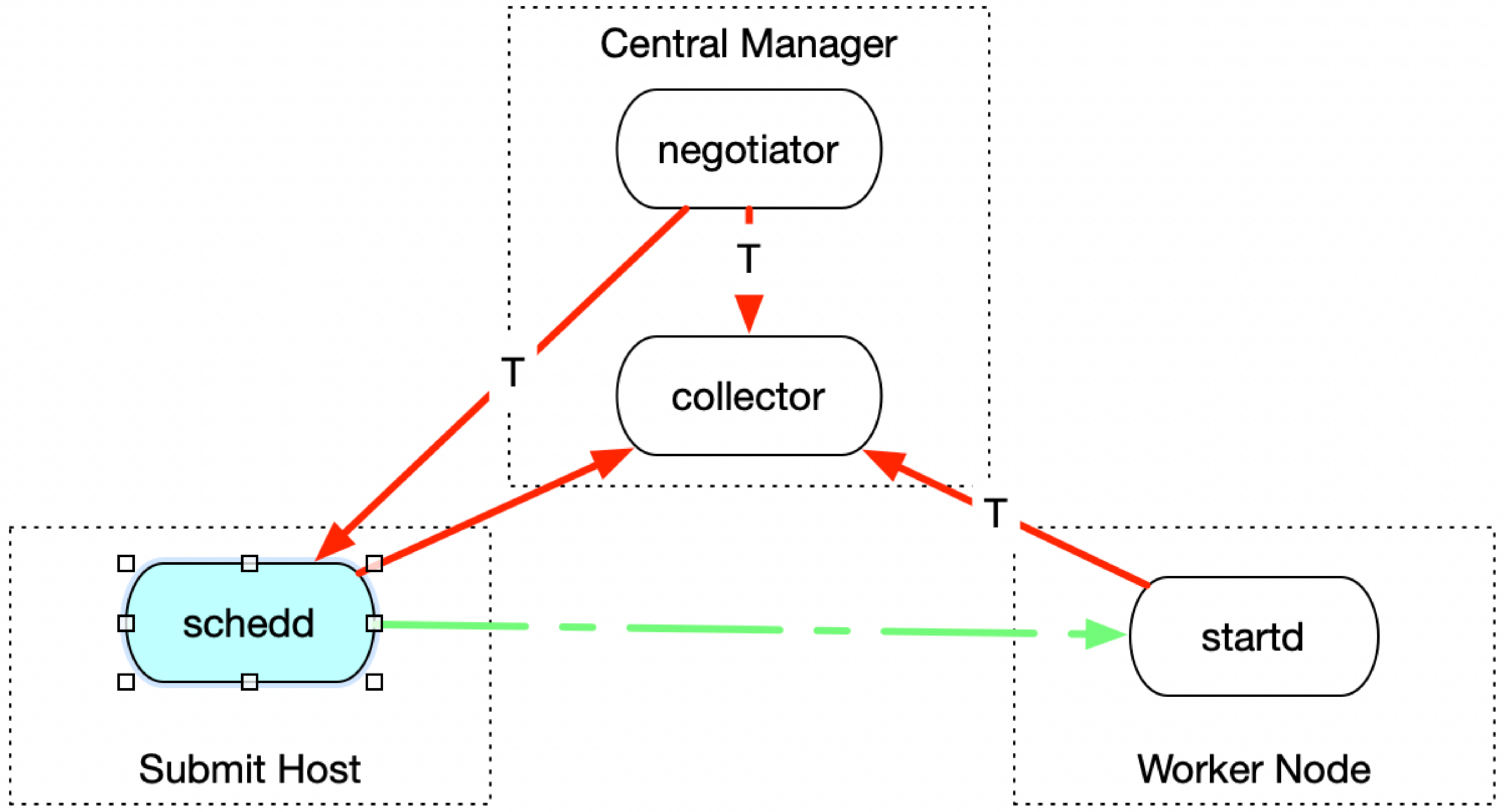


```
# Mapfile
CERTIFICATE_MAPFILE       = /etc/condor/condor_mapfile

# Authorization settings
# These should be unnecessary, unless if we have an error below.
DENY_WRITE          = anonymous@*, unmapped@*
DENY_NEGOTIATOR     = anonymous@*, unmapped@*
DENY_ADMINISTRATOR  = anonymous@*, unmapped@*
DENY_DAEMON         = anonymous@*, unmapped@*

# Macros defining grousp of daemons
FRIENDLY_DAEMONS = *@daemon.example.com
WORKER_NODES     = *@worker.example.com/*
USERS            = *@example.com

# Authz settings for each daemon.  Preferably, change the templates above
DEFAULT_WRITE = $(FRIENDLY_DAEMONS), $(HOSTNAME)@worker.example.com/$(FULL_HOSTNAME)
ALLOW_WRITE = $(DEFAULT_WRITE)

# Schedd is the only one accepting non-strong auth
SCHEDD.ALLOW_WRITE                 = $(USERS), $(HOSTNAME)@daemon.example.com/$(FULL_HOSTN
NEGOTIATOR.ALLOW_WRITE             = $(FRIENDLY_DAEMONS)
COLLECTOR.ALLOW_ADVERTISE_MASTER   = $(FRIENDLY_DAEMONS), $(WORKER_NODES), condor@example.
COLLECTOR.ALLOW_ADVERTISE_SCHEDD   = $(FRIENDLY_DAEMONS)
COLLECTOR.ALLOW_ADVERTISE_STARTD   = $(WORKER_NODES), $(HOSTNAME)@daemon.example.com/$(FUL
STARTD.ALLOW_NEGOTIATOR            = cluster-condor@daemon.example.com/cluster-condor.exam
                                     $(HOSTNAME)@daemon.example.com/$(FULL_HOSTNAME)
SHADOW.ALLOW_WRITE                 = $(DEFAULT_WRITE), $(WORKER_NODES), \
                                     $(HOSTNAME)@daemon.example.com/$(FULL_HOSTNAME)
ALLOW_DAEMON                       = $(FRIENDLY_DAEMONS), condor@example.com, submit-side@
                                     $(HOSTNAME)@worker.example.com/$(FULL_HOSTNAME)
ALLOW_ADMINISTRATOR                = cluster-condor@daemon.example.com/cluster-condor.exam
                                     cluster-man@example.com/cluster-man.example.com, \
                                     cluster-man@daemon.example.com/cluster-man.example.co
                                     cluster-man@daemon.example.com/172.16.200.1, \
                                     $(HOSTNAME)@daemon.example.com/$(FULL_HOSTNAME), \
                                     $(HOSTNAME)@worker.example.com/$(FULL_HOSTNAME)
```

Setting up SSL Security

7. … Or just give up and use host security?

“Match Password Authentication”



The startd trusts any negotiator trusted by its collector!

+

The negotiator trusts any schedd in the collector.

=

<u>The startd trusts any schedd in its collector.</u>

For a few years, HTCondor has had “match password” security. In this case,

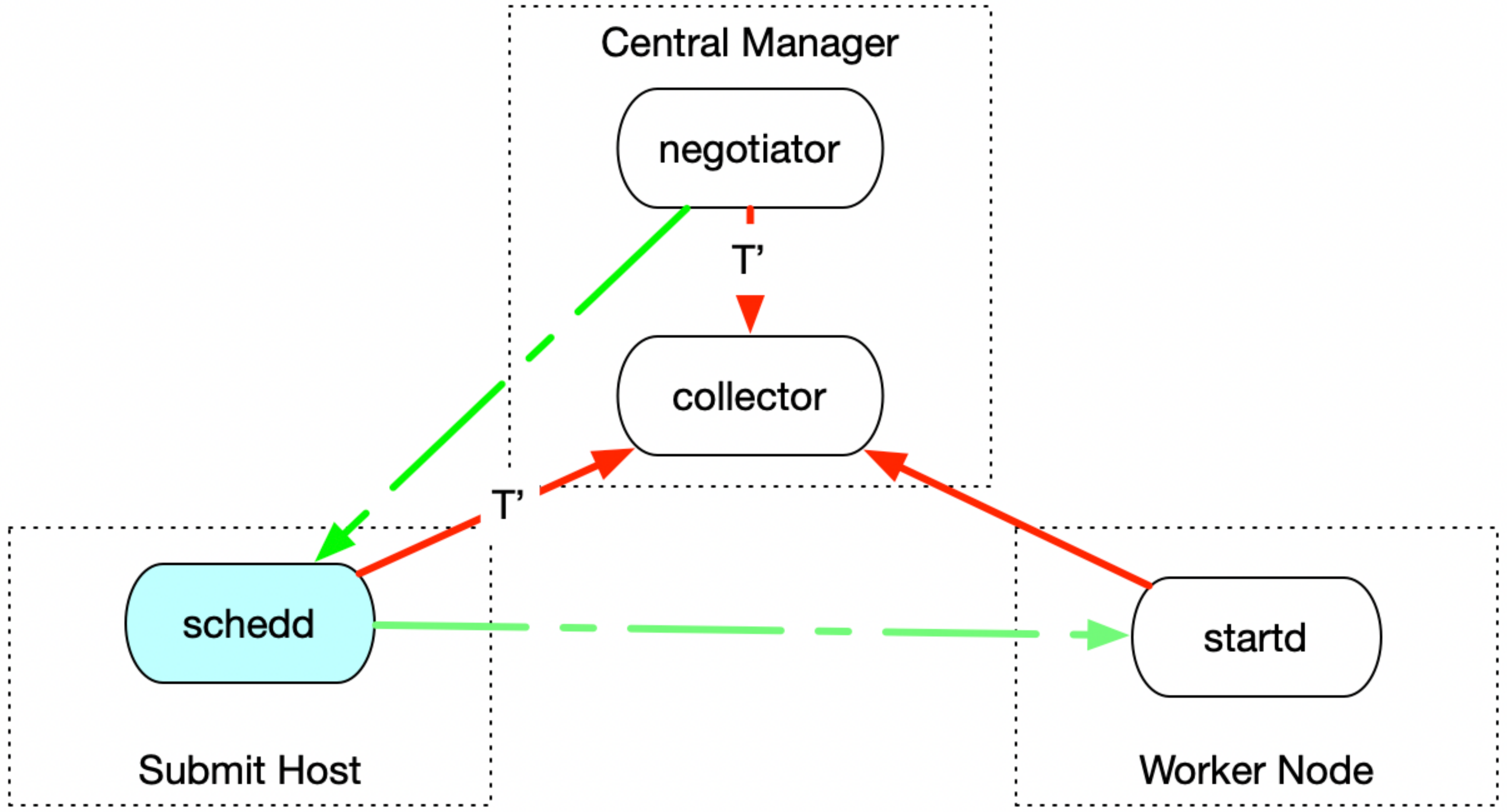
- The startd generates a capability, <u>T</u>, and sends it to the collector in its ClassAd.
  - **Anyone with <u>T</u> is allowed to start jobs on the startd.**
- The negotiator gets <u>T</u> from the collector because the collector trusts the negotiator.
- The schedd gets <u>T</u> as part of the ‘match’ created by the negotiator.
  - Hence the name “match password”.

Extending Trust in the Collector

Starting in 8.9.x, the schedd also generates a token, T', and sends it in its ClassAd.

- The schedd trusts the collector only gives T' to trustworthy negotiators.
- Any client with T' is allowed to be a negotiator for the schedd.

Note the only "full authentication" arrows are to the collector.



TRUST THE COLLECTOR

In 8.9, all trust is established is through the collector.

- Instead of needing credentials between any two daemons, only credentials to authenticate with the collector are needed.
  - We implicitly trust anyone the collector hands our security sessions to.
- Think of the collector as establishing a trust domain.
  - Trust domain -> set of daemons run by the same administrator.

If we trust the collector, why not allow it to issue credentials?

Introducing:

The IDTOKEN

eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.eyJpYXQiOjE1ODk1NjYwOTEsImlzcyI6I
mNvbGxlY3Rvci5leGFtcGxlLmNvbSIsImp0aSI6ImQyODI1YjNhYTkyNzcyYWQ3ZmJi
NmNmMDNmZmI0ZmU2Iiwic3ViIjoiYnJpYW4uYm9ja2VsbWFuQGNvbGxlY3Rvci5le
GFtcGxlLmNvbSJ9.z8LUtjmqL_bqXTtUpC0-nXGflBfW3zI0JuB43S9MOGE

Many of the ideas were extensions of approaches developed as part of SciTokens (NSF #1738962)

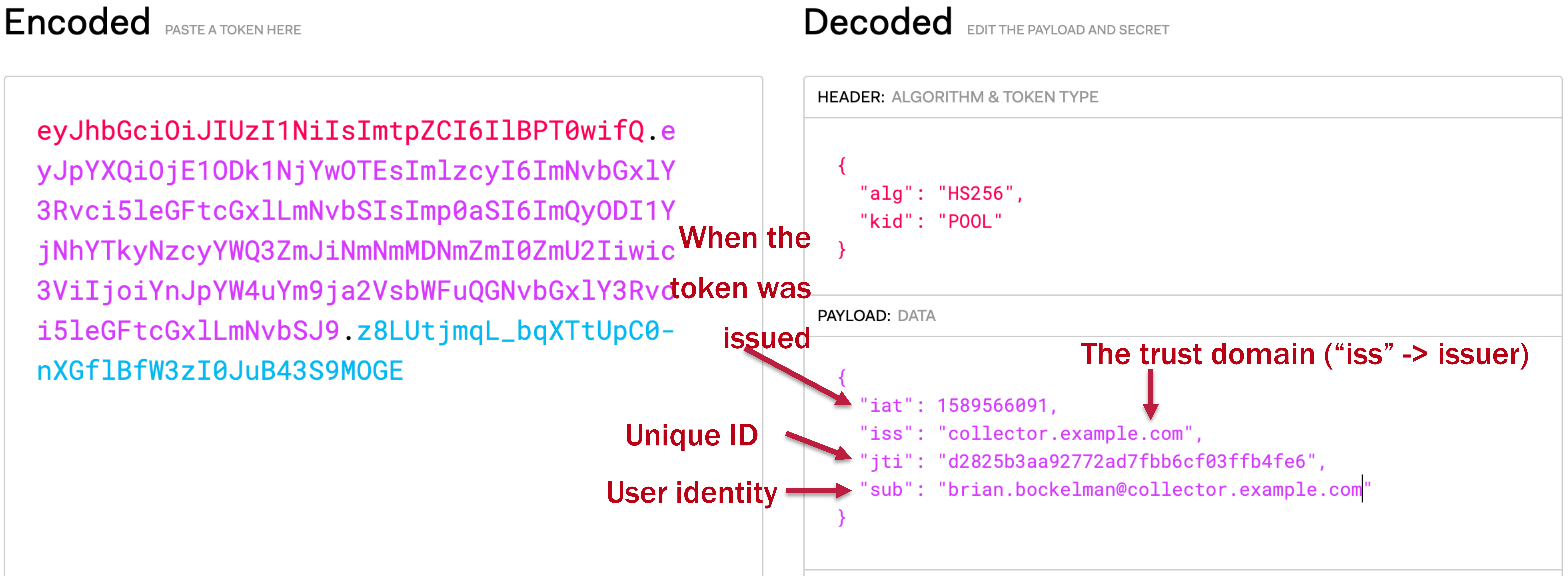
IDTOKENS

An IDTOKEN is a bearer token that can be used to authenticate an identity:

- An **IDTOKEN is signed** (often by the collector) – the signature can be validated by a daemon with the master password.
- Any given token is valid within a single trust domain.
  - Multiple master passwords can be used within the same trust domain.
- The <u>IDTOKEN embeds an identity</u>. The HTCondor authorization system can operate on this identity.
- The IDTOKEN may contain restrictions:
  - On when the token is valid ("expires next week")
  - What the token can be used for ("useful only for READ permission").

Any client can have multiple IDTOKENS – useful for authenticating with servers in different trust domains!

Big secret: IDTOKENS are JWTs - Basic Example



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.e
yJpYXQiOjE1ODk1NjYwOTEsImlzcyI6ImNvbGxlY
3Rvci5leGFtcGxlLmNvbSIsImp0aSI6ImQyODI1Y
jNhYTkyNzcyYWQ3ZmJiNmNmMDNmZmI0ZmU2Iiwic
3ViIjoiYnJpYW4uYm9ja2VsbWFuQGNvbGxlY3Rvd
i5leGFtcGxlLmNvbSJ9.z8LUtjmqL_bqXTtUpC0-
nXGflBfW3zI0JuB43S9MOGE
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "kid": "POOL"
}
```

PAYLOAD: DATA

```
{
  "iat": 1589566091,
  "iss": "collector.example.com",
  "jti": "d2825b3aa92772ad7fbb6cf03ffb4fe6",
  "sub": "brian.bockelman@collector.example.com"
}
```

(output from jwt.io)

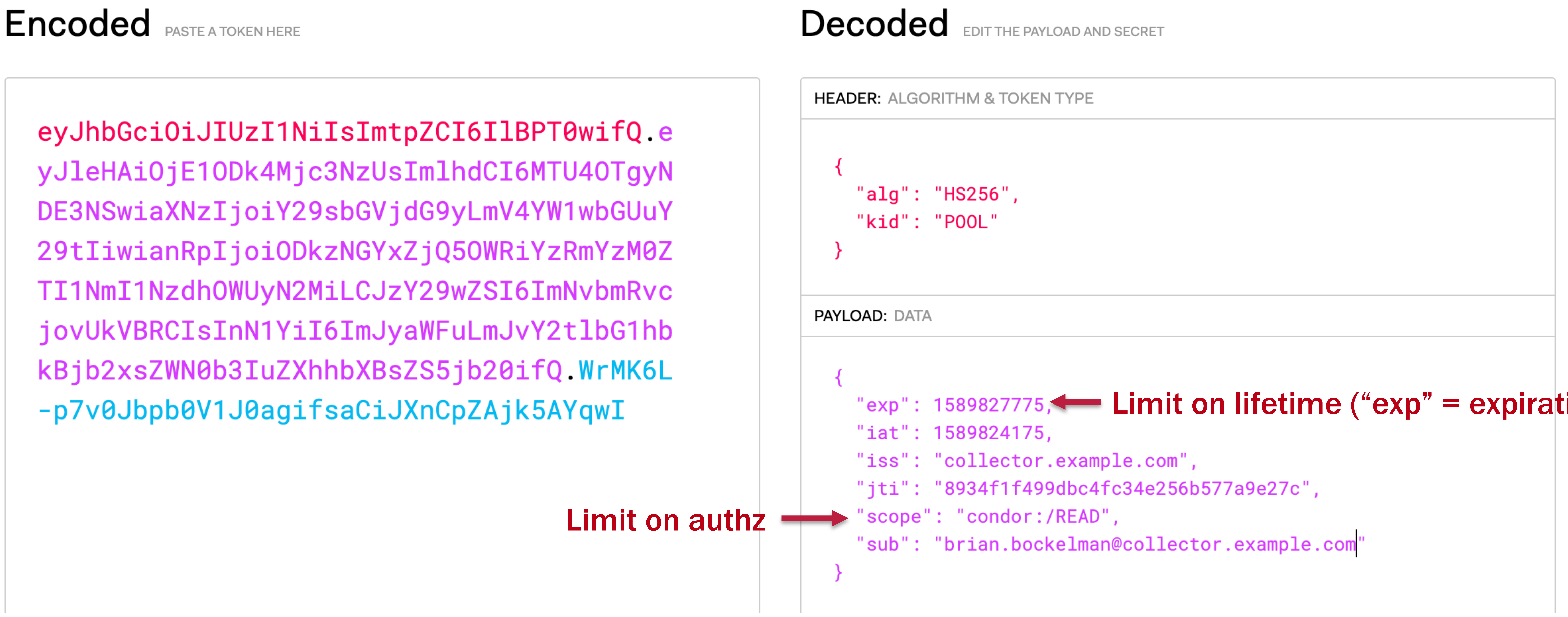
IDTOKENS have IDs

The IDTOKEN contains an identity within HTCondor.

There's no "mapfile" as in SSL/GSI as there's no external identity to map.

You *do* have to authorize an identity to perform an action (ALLOW_* options)

Big secret: IDTOKENS are JWTs –Complex Example



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.e
yJleHAiOjE1ODk4Mjc3NzUsImlhdCI6MTU4OTgyN
DE3NSwiaXNzIjoiY29sbGVjdG9yLmV4YW1wbGUuY
29tIiwianRpIjoiODkzNGYxZjQ5OWRiYzRmYzM0Z
TI1NmI1NzdhOWUyN2MiLCJzY29wZSI6ImNvbmRvc
jovUkVBRCIsInN1YiI6ImJyaWFuLmJvY2tlbG1hb
kBjb2xsZWN0b3IuZXhhbXBsZS5jb20ifQ.WrMK6L
-p7v0Jbpb0V1J0agifsaCiJXnCpZAjk5AYqwI
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "kid": "POOL"
}
```

PAYLOAD: DATA

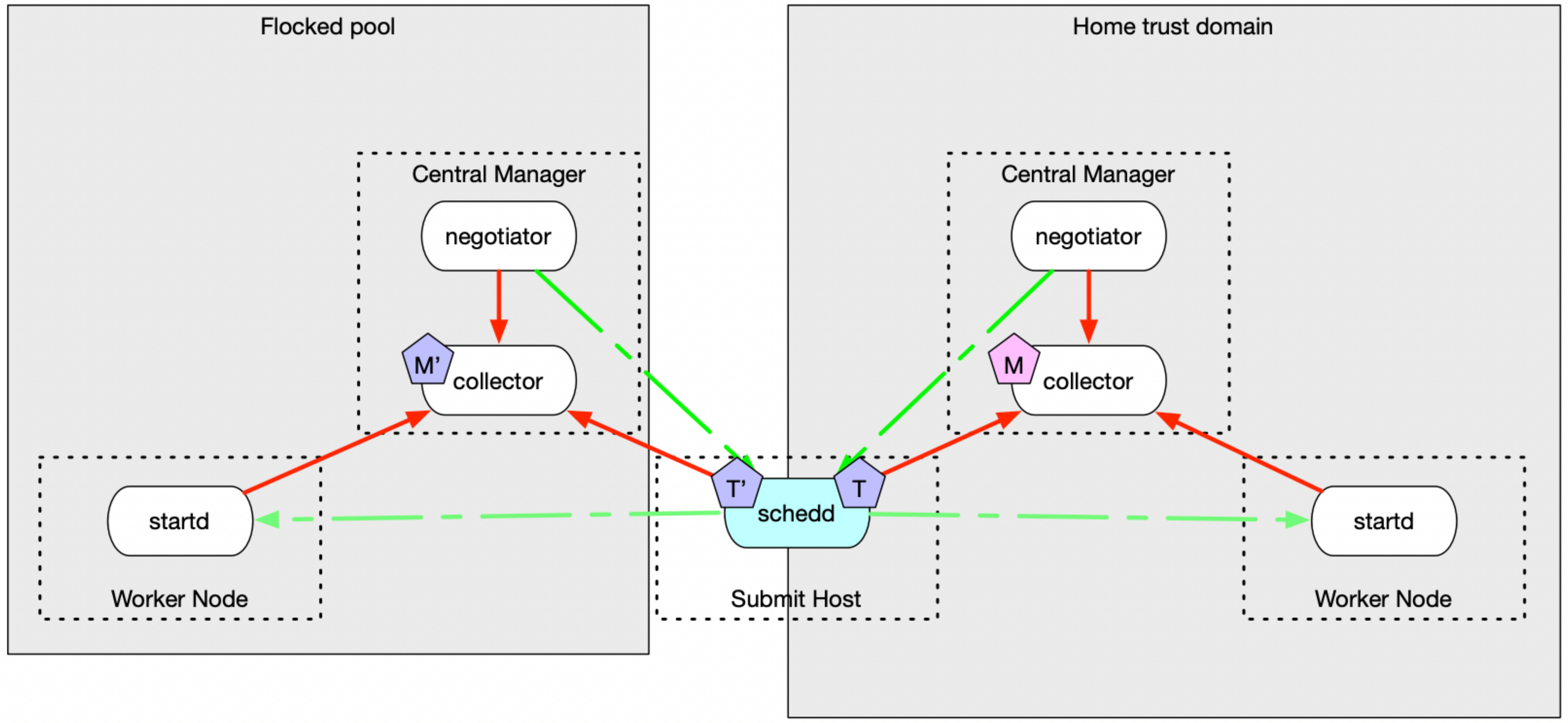
```
{
  "exp": 1589827775,
  "iat": 1589824175,
  "iss": "collector.example.com",
  "jti": "8934f1f499dbc4fc34e256b577a9e27c",
  "scope": "condor:/READ",
  "sub": "brian.bockelman@collector.example.com"
}
```

Limit on lifetime ("exp" = expiration)

Limit on authz

(output from jwt.io)

Trust domains, tokens, and passwords – Oh my!



To see trust domains, tokens, and passwords in action, consider the case of schedd flocking:

- Each pool is a different **trust domain** – the two pool administrators are distinct!
  - Accordingly, each collector has a separate **master password**.
- Each token is in one trust domain (and signed by a different password) so the schedd needs two tokens – **one for each pool**!

Where to find my token information

Each token is a file in a directory:

- ~/.condor/tokens.d/ (users)
- /etc/condor/tokens.d/ (condor or root)
- Overridden by SEC_TOKEN_DIRECTORY

Each master password is also a file in the directory:

- /etc/condor/passwords.d/
- Overridden by SEC_PASSWORD_DIRECTORY

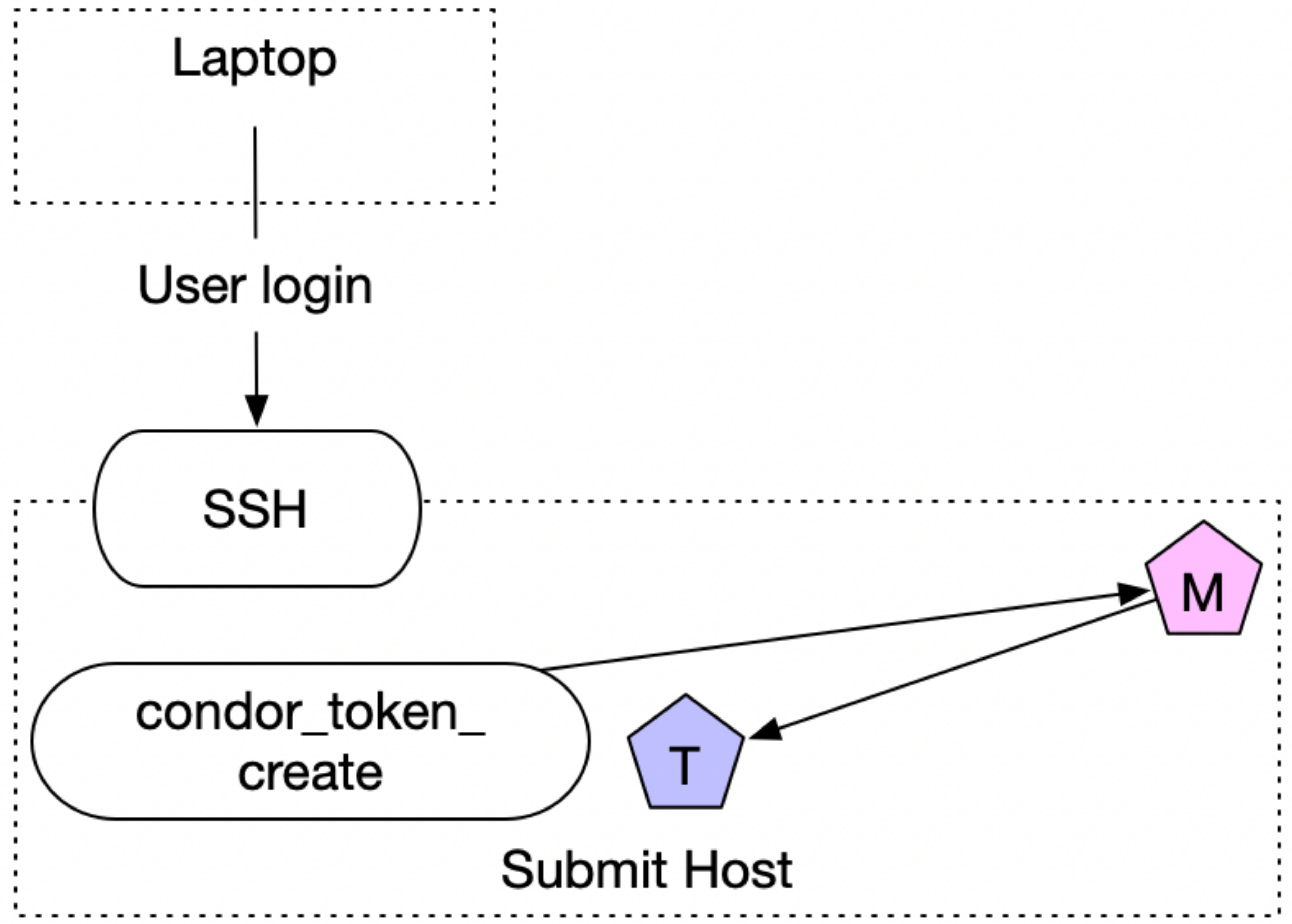
The trust domain is configured by

- TRUST_DOMAIN
- Defaults to $(COLLECTOR_HOST)

Bootstrapping Trust – Creating an IDTOKEN

Anyone who can read the master password can issue any token they want using **condor_token_create**.

```
$ sudo condor_token_create \
    -identity brian.bockelman@collector.example.com \
    -lifetime 3600 \
    -authz READ -authz WRITE
```

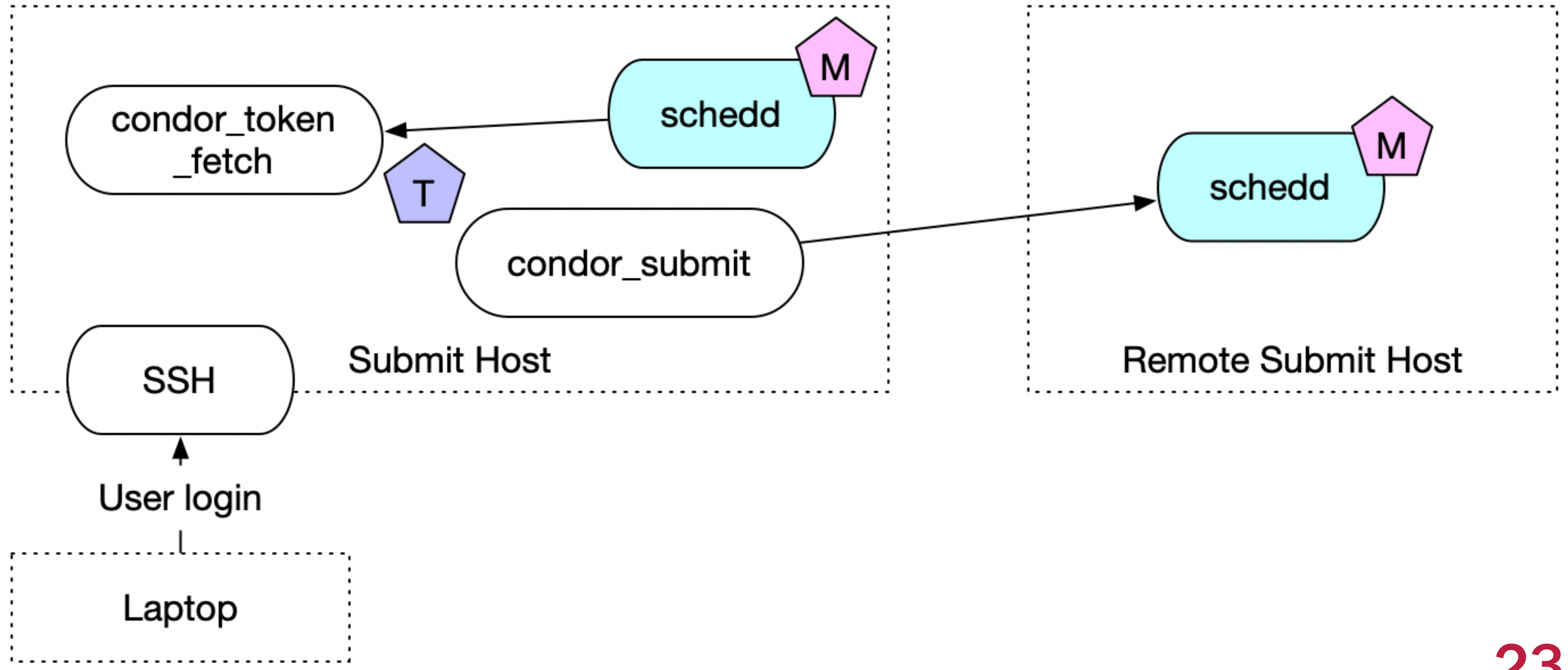


eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.eyJleHAiOjE1ODk4Mjk4MzUsImlhdCI6MTU4OTgyNjIzNSwiaXNzIjoiY29sbGVjdG9yLmV4YW1wbGUuY29tIiwic2NvcGUiOiJjb25kb3I6L1JFQUQgY29uZG9yOi9XUklURSIsInN1YiI6ImJyaWFuLmJvY2tlbG1hbkBjb2xsZWN0b3IuZXhhbXBsZS5jb20if
Q.NxOw5f9GsmGgwV0TezisZwmtqRbRuGHvj8G1r5esdLI

Fetching an IDTOKEN

Does authentication work now – but you need to squirrel away an IDTOKEN for future use? **condor_token_fetch** to the rescue!

- This tool authenticates with a daemon and asks the daemon to sign a token on behalf of the user's identity. Resulting identity is identical to authenticated ID.
- **Use case:** I have an SSH login to a local schedd but want to remotely submit to a schedd in the same trust domain.

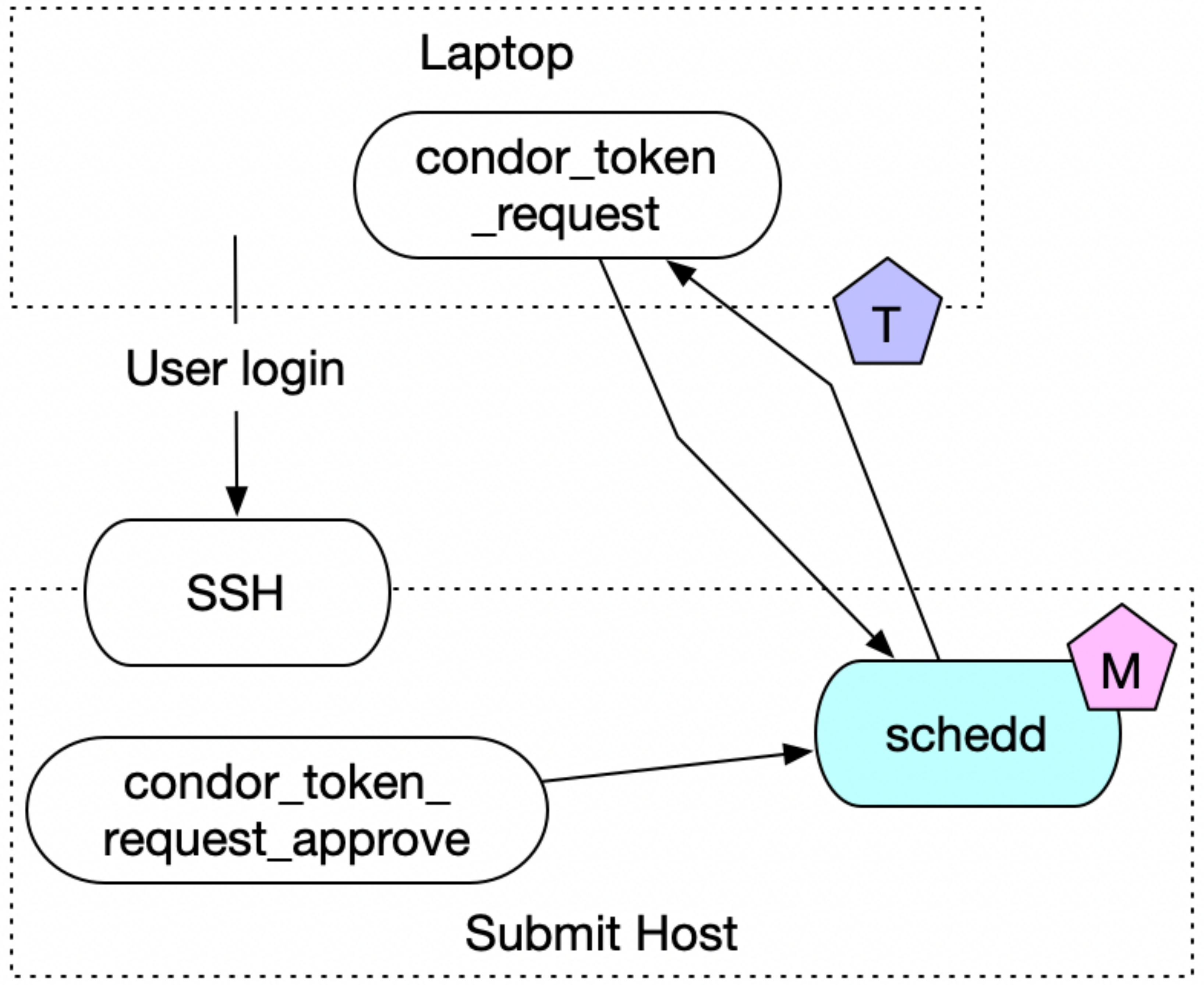


Requesting an IDTOKEN

Want to get an IDTOKEN on a machine without authenticating?

- **condor_token_request** allows an anonymous user to request a token for an arbitrary identity X.
  - The token request can be approved either by an admin or a user authenticated as X.
  - Anyone can ask. Few can approve!
- **Use case**: I have an SSH login on a schedd and want to start submitting jobs from my laptop.
  - **Solution**: Request a token from my laptop; login to the submit host and approve the request.
- ***DO NOT COPY/PASTE TOKENS.*** Instead, use **condor_token_request**!

The startd, master, and schedd will automatically request tokens from the collector if authentication fails.



Bootstrapping Trust – Autoapproval

Token requests are a handy tool for securely bootstrapping authentication – provides a tool for secure movement of credentials.

- Great tool for adding a new worker node by hand.
- Crappy tool for adding 1,000 worker nodes!

Auto-approval mode for the rescue! Automatically approves certain requests for a specific network and time duration

How do I install a new cluster?

- On first start, a collector automatically generates a new master password.
- Enable auto-approval mode for hosts coming from the new subnet.
- If a schedd, startd, or master cannot authenticate with the collector, they will automatically request a token. If the requests come from the correct subnet, the token will be immediately issued.
- Wait for all the hosts to show up in condor_status then disable auto-approval mode.

Token Revocation

How do you “undo” a token? What happens if a user says “sorry, but my laptop got stolen and my IDTOKEN was on it?”

- An IDTOKEN must be signed by a password.
  - Remove the password and the token is invalid!
  - Erm … but all tokens are invalid. ☹
- You can add a user to the DENY_* lists to remove all authorizations for that identity.
  - But then the user can never use HTCondor again ☹
- In 8.9.7, HTCondor has SEC_TOKEN_BLACKLIST_EXPR.
  - Any token matching this ClassAd expression will be rejected.
  - If you know the token unique ID, you can blacklist only that token. ☺
  - You can also reject all tokens owned by a specific user that were issued before last Friday. ☺

Closing Thoughts

IDTOKENS strives to balance simplicity and fine-grained auth.

- Importantly, we provide tools to setup your HTCondor pool with IDTOKENS.

Not all is unicorns and rainbows!

- Some workflows still require central manager -> worker node auth: think `condor_fetchlog` or `condor_defrag`.
  - Giving an IDTOKEN to the worker node doesn't help because IDTOKENS identify clients; here, the worker node is a server and needs the master password.
  - Should be fixed before 9.0.
- Documentation exists in the manual but it could admittedly use some love.

Would love to hear how you use these new tools!

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

morgridge.org

FEARLESS SCIENCE