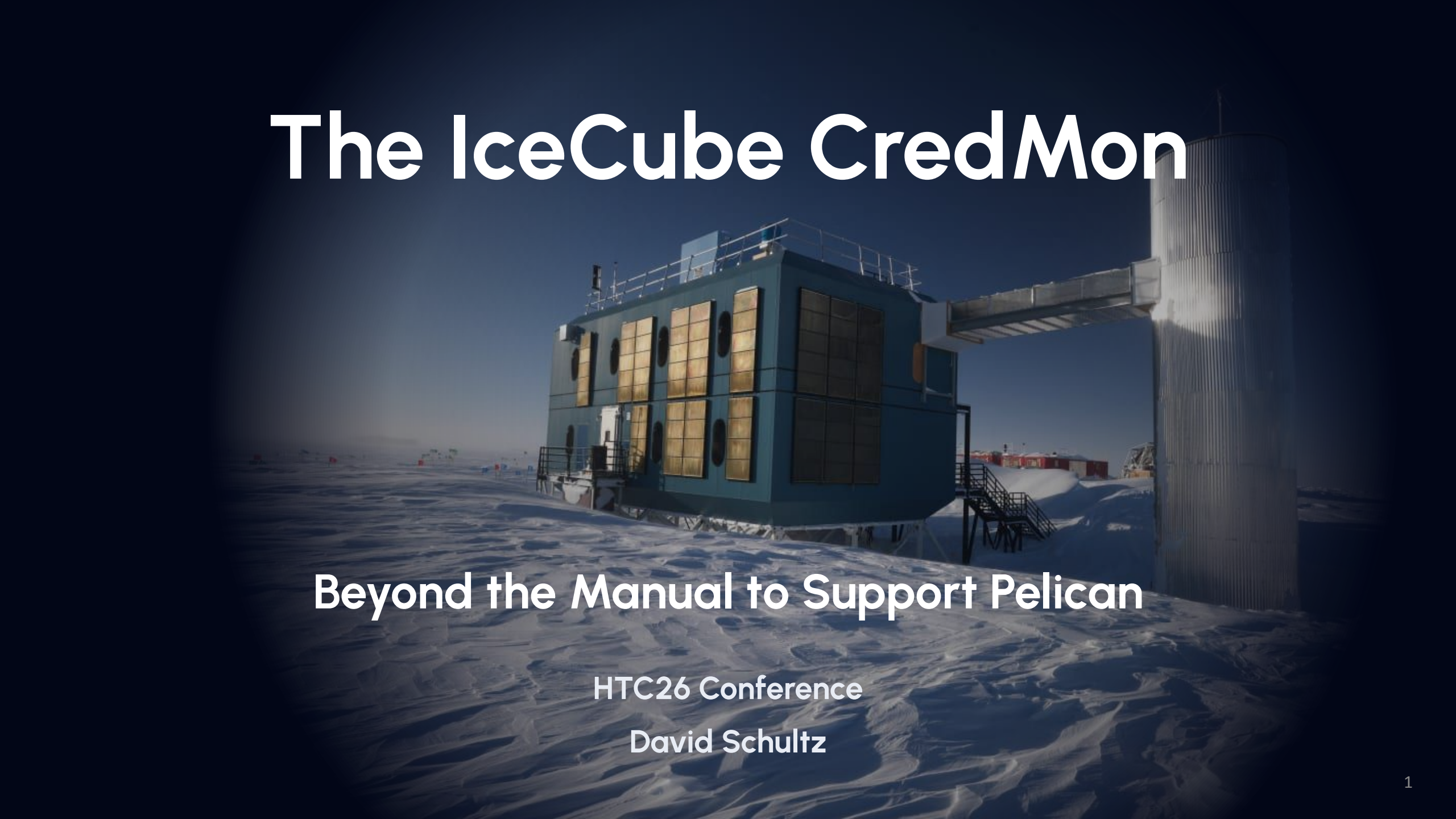
# The IceCube CredMon



## Beyond the Manual to Support Pelican

HTC26 Conference

David Schultz

# DATA FROM THE SOUTH POLE

Located at the Amundsen-Scott South Pole Station, the IceCube Neutrino Observatory instruments a cubic kilometer of clear Antarctic ice to detect high-energy cosmic particles

IceCube can process terabytes of data per day, distributed across a global compute infrastructure

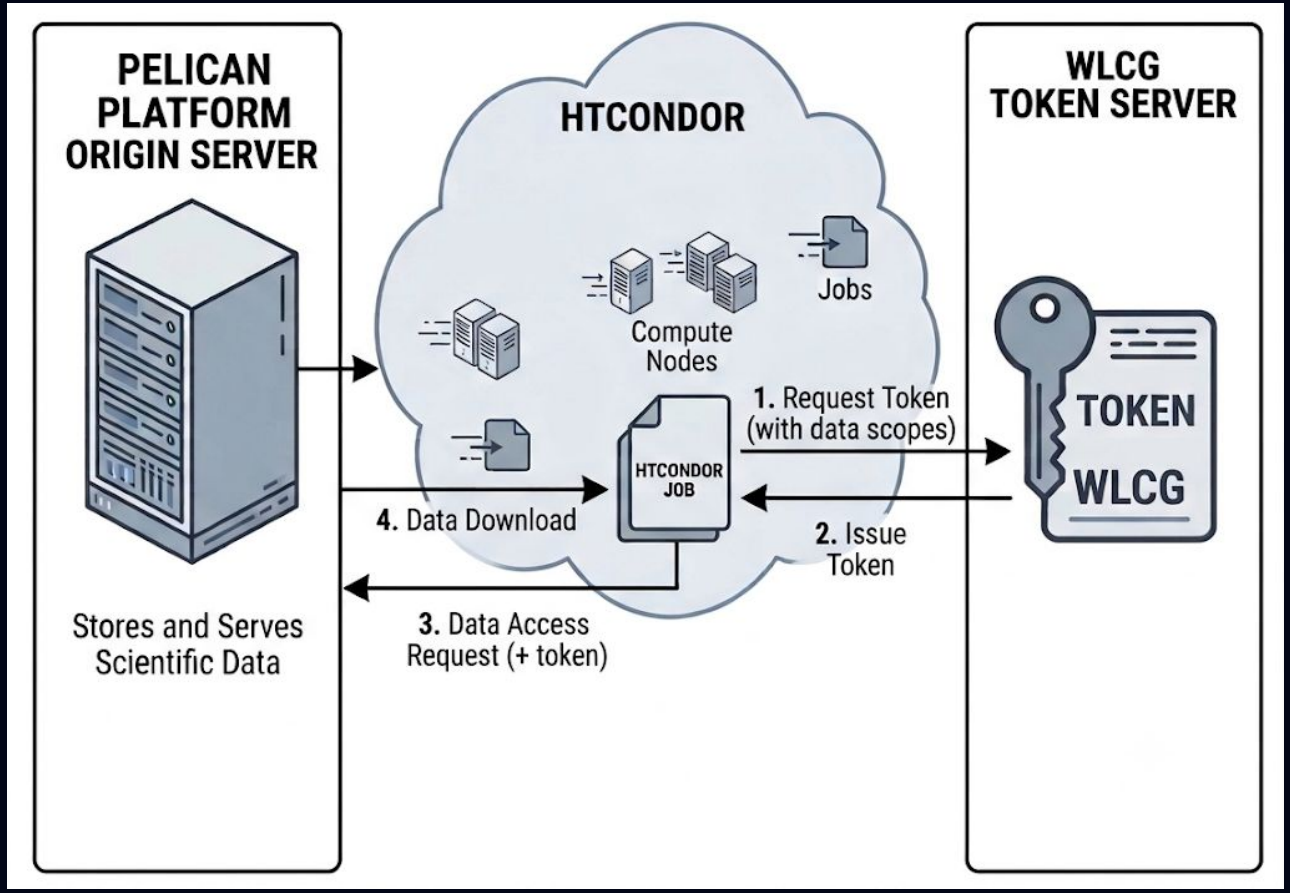
- Over 5000 detectors buried in the ice, producing data 24/7/365
- Real-time multi-messenger alerts
- High-throughput processing across the OSPool

# CONTEXT: OSDF AND PELICAN

Leveraging the Pelican Platform and the Open Science Data Federation (OSDF).

IceCube has a central token server that can issue WLCG-compliant data tokens.

- Namespace: osdf:///icecube/wipac/
- Data tokens required for secure data access
- OSDF Pelican Director manages redirection to our Origin server



# THE FRICTION OF WEB-AUTH FLOWS

### Browser Access

Default OAuth2 CredMon expects a web browser redirect. On submit nodes, this creates a "GUI wall" that breaks out of the ssh terminal.

### Manual Intervention

Clicking a link breaks automatic workflows like DAGman or Python submissions.

# THE ICECUBE CREDMON

0

User Interaction Required

We developed a custom CredMon in Rust to address security and workflow requirements:

- **No User Interaction:** Should be able to work without a human
- **Authentication:** SSH login is sufficient - no extra login needed (authorization check to read/write happens on token issuer)
- **Centralized Secrets:** No client IDs or secrets in metadata files

# RUST: PERFORMANCE AND SAFETY

- **Memory Safety:** Eliminates runtime segfaults and buffer overflows in sensitive auth code
- **High-Throughput:** Minimal overhead during token generation and refresh
- **Easily Deployable:** One static binary across all submit nodes, regardless of OS

# SECURITY MODEL

Since users authenticate via SSH/LDAP/password, the local session creation establishes trust.

The job submit part of the CredMon runs SUID, fetching the user ID from the calling context while accessing protected client secrets and token storage.

The refresh part of the CredMon runs as any other HTCondor daemon, managed by the HTCondor master

# END-TO-END WORKFLOW

1. **Submit:** Researcher runs condor_submit
2. **Call:** Submit triggers CredMon token creation process
3. **Fetch:** Token requested from issuer and saved to HTCondor
4. **Job Start:** HTCondor sends access token to worker
5. **Transfer:** Pelican uses access token for OSDF access

# HTCONDOR CONFIG SNIPPETS

```
# The two binaries, to create and refresh tokens
SEC_CREDENTIAL_STORER = /usr/bin/condor_credmon_rust_client
CREDMON_OAUTH = /usr/sbin/condor_credmon_rust

# In order for condor to not print out a url, we claim we are a Vault credmon
VAULT_CREDMON_PROVIDER_NAMES = myprovider
# The base path to the issuer, for dynamic discovery.
myprovider_ISSUER = https://my.issuer.here
myprovider_CLIENT_ID = XXXXXX
myprovider_CLIENT_SECRET_FILE = /etc/condor/.secrets/XXXXXX-client-secret
# Hack: actually tell the STORER which provider this is
myprovider_DEFAULT_OPTIONS = myprovider
```

# HTCONDOR CONFIG SNIPPETS

```
# This is the minimum time in seconds that access tokens must have
#   before they expire when they are fetched by credmon.  It must
#   be set to be less than the expiration time assigned by the token
#   issuer.
# NOTE: this must be larger than SEC_CREDENTIAL_REFRESH on the EP,
#   which is by default 300
CREDMON_OAUTH_TOKEN_MINIMUM=360

# This is the time in seconds between fetching new access tokens.
#   If not set, the default is half of CREDMON_OATH_TOKEN_MINIMUM.
CREDMON_OAUTH_TOKEN_REFRESH=60
```

# OPERATIONAL COMPARISON

| Metric / Capability | Default Web-Interactive Flow | IceCube Rust CredMon |
|---|---|---|
| **User Submission Step** | Requires external web browser auth | Fully automated (zero click) |
| **Headless Environment** | Incompatible / Fails DAGMans | Fully supported & automated |
| **Secrets Isolation** | Secrets scattered in token storage | Secret only stored in /etc/condor/.secrets |
| **Operational Failure Rate** | High (human intervention issues) | ~0% (limited only by hardware outages) |

# OPERATIONAL SUCCESS METRICS

~0

Token Auth Failures

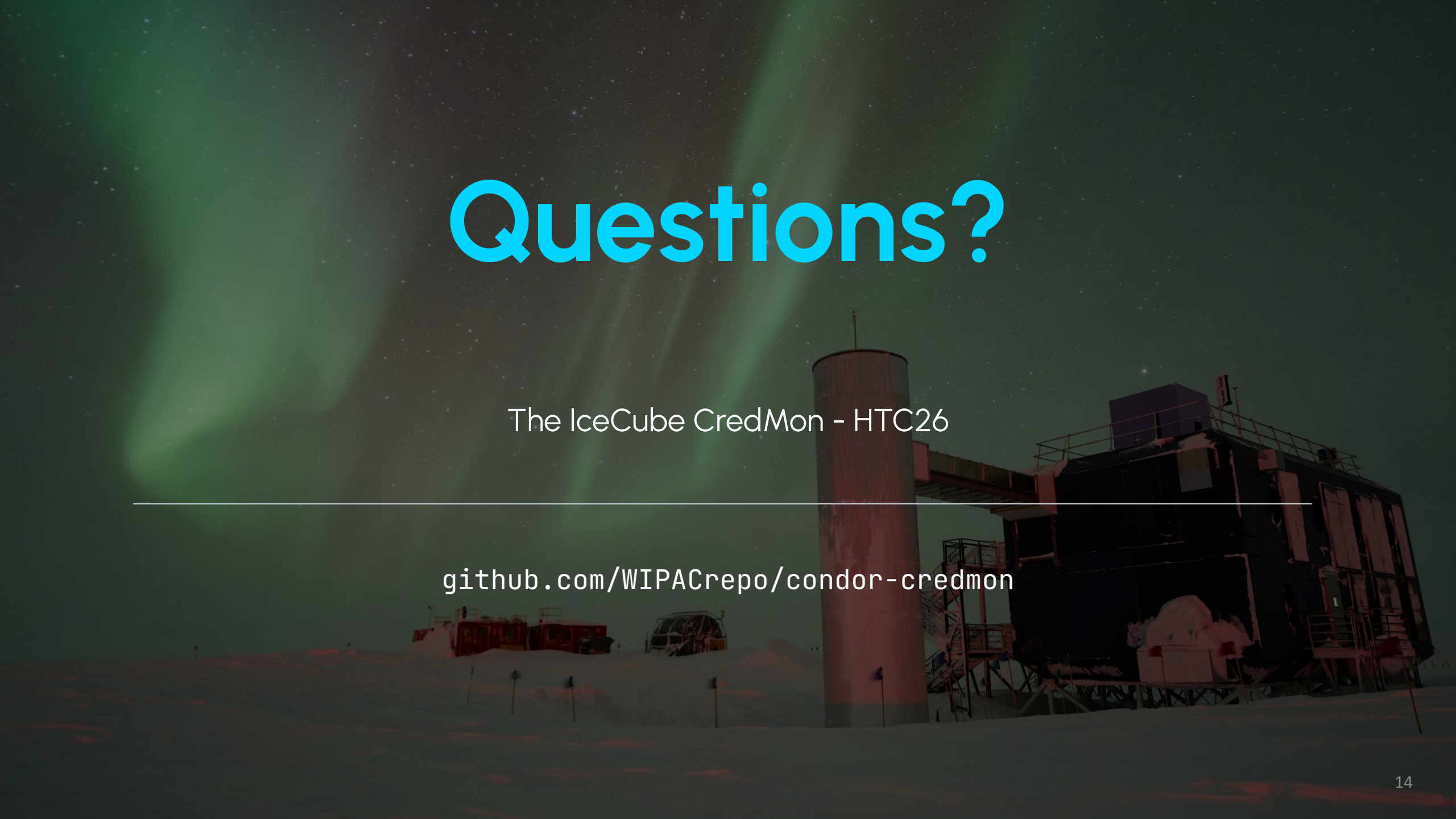
Excluding direct hardware or network outages, the IceCube CredMon has maintained near-perfect reliability across millions of jobs.

- Resilience against transient token server blips
- Successful management of high-volume token creation / refreshes
- User feedback: "It just works"

# FUTURE

While this was designed for IceCube, it can definitely be used by others.

Talking with the HTCondor team about a PREPARE job state, to remove the SUID requirement and a number of hacks in the submit process..


Questions?
The IceCube CredMon - HTC26
github.com/WIPACrepo/condor-credmon