



Addressing tokens dynamic generation, propagation, storage and renewal to secure the GlideinWMS pilot based jobs and system

Marco Mambelli, Bruno Moreira Coimbra, Kyle Knoepfel (speaker)

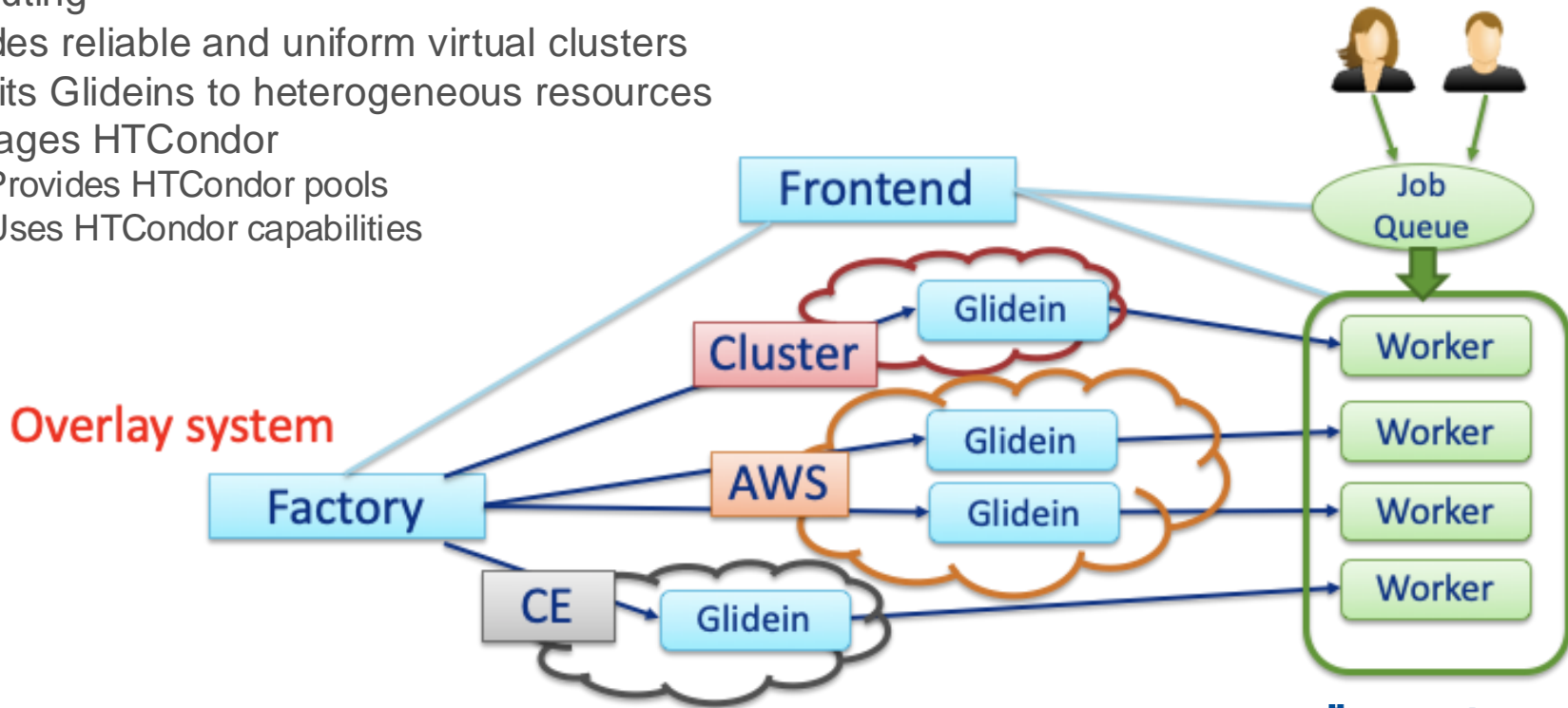
CHEP2024

Krakow, Poland, October 19-25, 2024

This work was supported by the Fermi National Accelerator Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy - FERMILAB-SLIDES-24-0284-CSAID

GlideinWMS

- GlideinWMS is a pilot-based resource provisioning tool for distributed High Throughput Computing
- Provides reliable and uniform virtual clusters
- Submits Glideins to heterogeneous resources
- Leverages HTCondor
 - Provides HTCondor pools
 - Uses HTCondor capabilities



Glidein: pilot job for node testing and customization

- Scouts for resources and validates the Worker node
 - Cores, memory, disk, GPU, OS, software installed, CVMFS, ...
- Customizes the Worker node
 - Environment, GPU libraries, Starting containers (Apptainer, ...)
- Provides monitoring and audit
- Runs one or more jobs in parallel or sequentially via HTCondor
- **Stores and uses Pilot credentials (e.g. VO Credentials, CE Credentials)**
- **Safely receives and stores Job credentials**

(Glidein) Factory

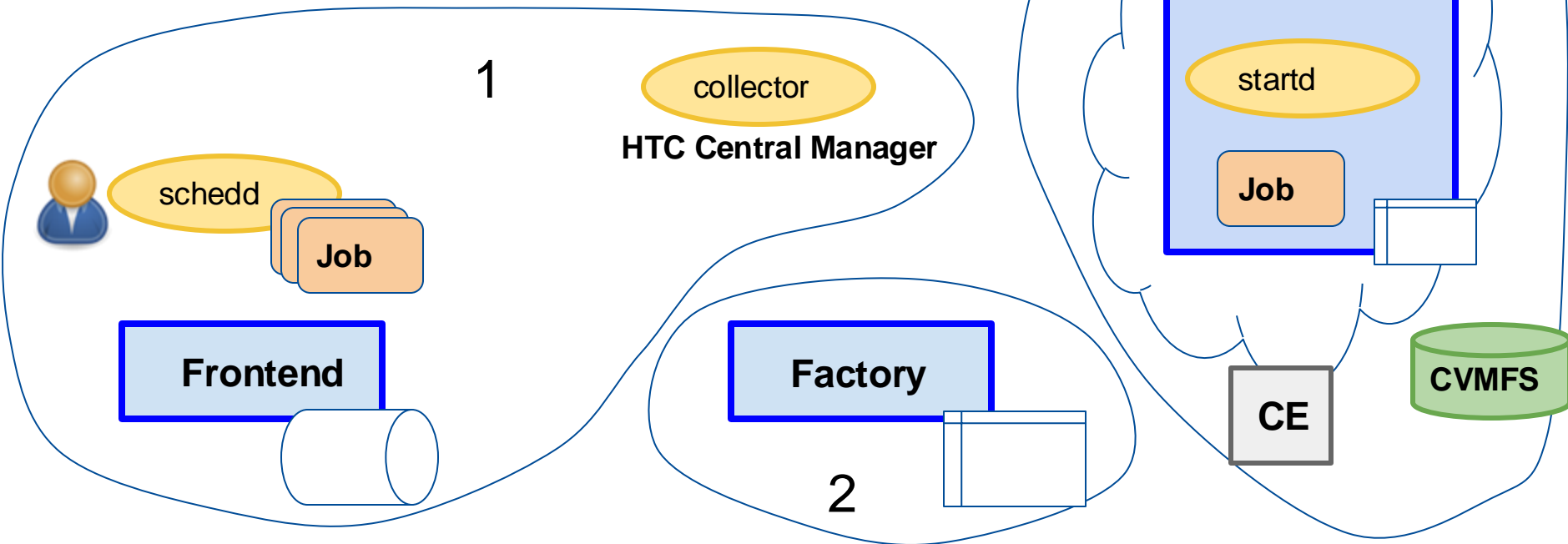
- Knows how to submit Glideins to sites
 - Sites are described in a configuration (curated, in VCS, or auto-generated)
 - **Authentication method, supported VOs**, expected resources, ..
 - Only trusted and tested sites are included in production
- Condor does the heavy lifting of submissions.
- **Keeps a cache of credentials used or forwarded to Glideins**

(VO) Frontend or client

- Pressure-based system controlling the Factory Glideins requests
 - Monitors job requests and available entries (sites)
 - Works keeping a certain number of Glideins running or idle at the sites
 - Limits Glideins requests to enforce policies, avoid spikes and overloads
 - Jobs, requests, and credentials are partitioned in “groups”
- **Manages credentials and delegates them to the Factory and Glidein**
 - **Stores and owns auto-generated or VO-managed credentials**
 - **Has some long-term credentials, forwarding short term ones**
 - **Manual input, interacts with IAMs**
- Other software, like the HEPCloud Decision Engine, can perform similar functionalities, so we call these GlideinWMS clients because provide instructions served by the Factory

Security domains

1. Submit Infrastructure - VO Services, experiments
2. Framework/Infrastructure Services - Shared
3. Resources



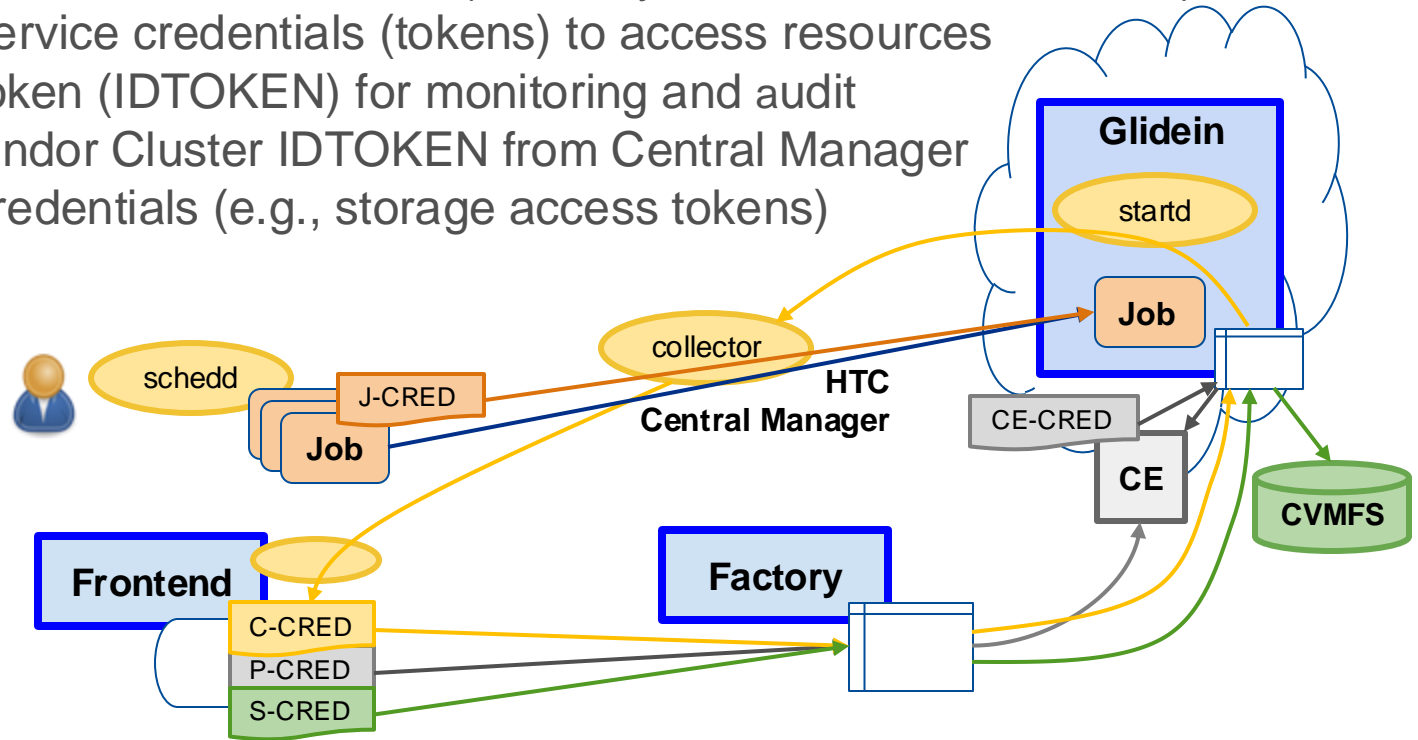
Security and Credentials in GlideinWMS

Access resources, provide secure infrastructure, and user job needs

- IDTOKENS and HTCondor security (was GSI) Internally
- Credentials have different purpose
 - Pilot submission
 - Pilot infrastructure operation (virtual cluster security, monitoring, ...)
 - Pilot VO services
 - User job operation
- Credentials are of different types, both identity or capability based
 - Tokens, GSI, SSH keys, ...
- Agnostic about the credential type
 - Provider and service must be compatible
- Credentials may have parameters
 - Project ID, VMID and VMATYPE
- Hierarchic groups of credentials from users

Token authentications

- P-CRED Pilot submission credential (SSH key, SciToken/ WLCG token)
- S-CRED VO Service credentials (tokens) to access resources
- CE-CRED CE Token (IDTOKEN) for monitoring and audit
- C-CRED HTCondor Cluster IDTOKEN from Central Manager
- J-CRED Job credentials (e.g., storage access tokens)

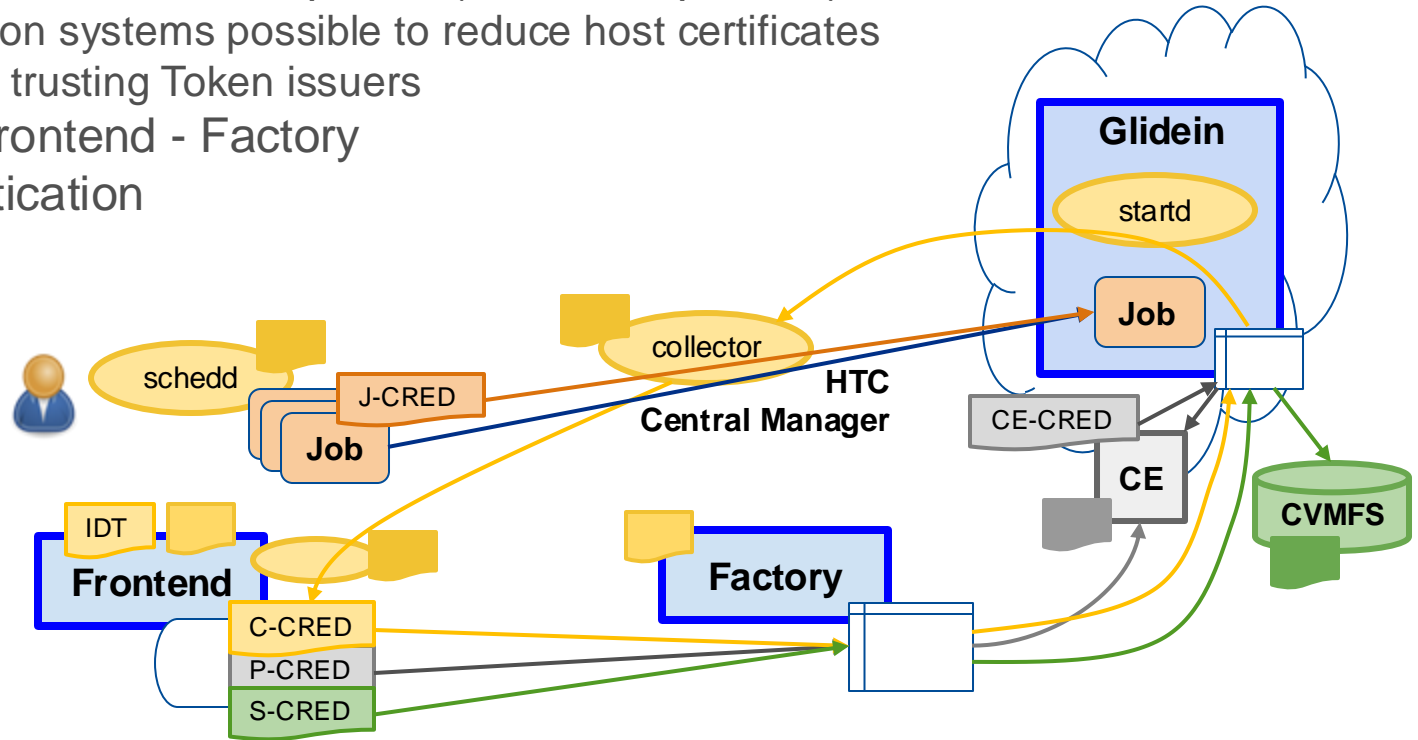


Token authentications - Hosts

Host certificates used to bootstrap TLS (DN not important)

Different authentication systems possible to reduce host certificates

- Service providers trusting Token issuers
- IDTOKEN for Frontend - Factory
- HTCSS authentication



Token Support Milestones

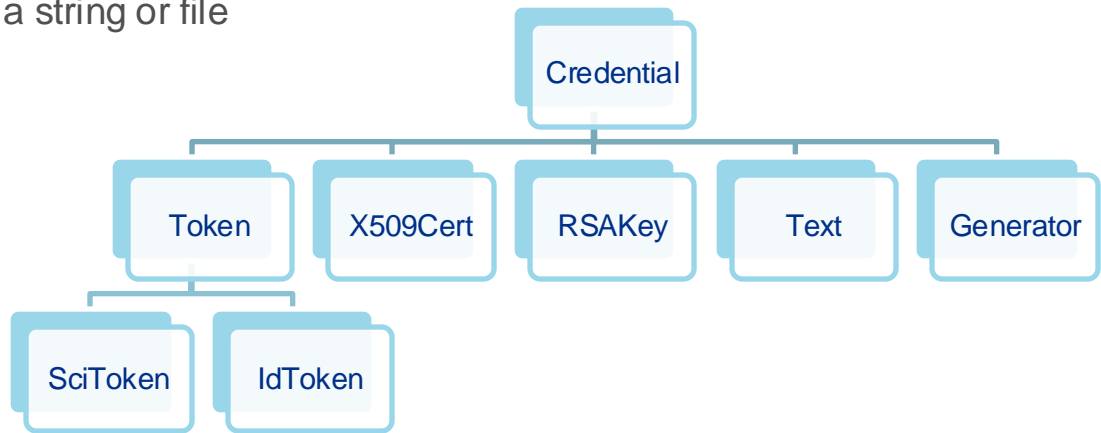
- 2019/05 –Use of tokens (security without x509 certificates) in roadmap
- 2019/10/22 – **Proof of concept** (Manually modified GlideinWMS using tokens and no proxy)
- New versions with incrementally new token features in Python 2 and Python 3
- 2021/09/02 – v3.7.5 Fix IDTOKEN generation, SciToken credential per-entry, **CMS and OSG production**
- 2022 – v3.9.x to 3.10.1 - More automation, configurability, better handling of transition, credential generator plugins, token support for Grid/Batch universe, GCE, and AWS, support HTCondor-CE collector
- 2024/07 – Test versions (3.11.0 RC1/2) with refactored credentials in GlideinWMS
- 2024/11 – Expected HEPCloud Decision Engine test version with same refactored credentials library
- 2024/12 – Expected 3.11.0 release with refactored credentials

Challenges from tokens transition

- Increased granularity in time and space to increase security
 - Short duration – need some refresh mechanism
 - Per-site credentials – need dynamic generation due to scale
- Different types and purposes
 - Complex requirement matching (auth_method)
 - Handling of fallbacks and transitions
- Other GlideinWMS clients have similar needs
 - GlideinWMS shares developers with HEPCloud Decision Engine
 - Decided to have a common library usable by both to avoid duplicate effort

New Credentials have Classes

- Class hierarchy for credential types
 - All credentials have a set of common methods and attributes
 - Factory and Frontend operations are agnostic to the credential type
 - Easy to add new types in the future
 - Credential types can be inferred from a string or file
- Better code
 - More structured
 - Patterns like Factory and Composite



Formalized Credential Purpose, Parameters, and match

- Credential Purposes
 - Determine how a credential is used (e.g. CE authentication, or joining virtual cluster)
 - Allow to distinguish credentials sent along with the Glidein
 - Standard (hardcoded) and custom (user-defined) purposes
- Credential Parameters
 - Credential qualifiers are now independent parameters (e.g. ProjectID, VMID, VMTYPE)
- New mechanism for matching credentials/parameters available at client and required by resources
 - At least one match per group
 - Fallback possible if multiple matches within a group

`auth_method="scitoken,grid_proxy;vmid;vmtype"`

Use SciToken or proxy for site authentication (or fallback if both are available), and provide also VM ID and VM type


Credential and Parameters Generators

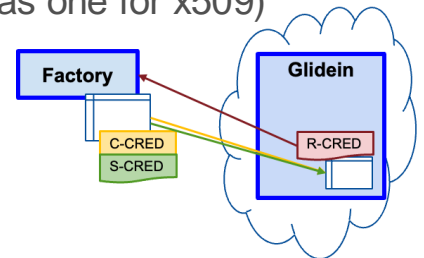
- Generic generator framework available for credentials and (security) parameters
- Extend the current functionality of credential generators
 - Per-site tokens generation at run-time
- Built-in generators, e.g. RoundRobinGenerator
- Customized in configuration file
- Simplified existing configurations
 - Before entries were replicated for each AWS VM type or zone to spread requests (combinatorial explosion)
 - Now one entry is doing round-robin

```
<credential
  absfname="RoundRobinGenerator"
  context="{ 'items': ['cred1', 'cred2', 'cred3'], 'type': 'text' }"
  purpose="payload"
  security_class="frontend"
  trust_domain="grid"
  type="generator"
/>

<parameter
  name="VMId"
  value="RoundRobinGenerator"
  context="{ 'items': ['vm1', 'vm2', 'vm3'], 'type': 'string' }"
  type="generator"
/>
```

Credentials expiration and renewal

- Credentials are renewed or generated at the Frontend/client
- Factory receives always updated credentials
- From here we rely on components that do not provide renewal mechanisms
 - HPC sites have long queue wait times, C-CRED and S-CRED may expire
 - If the virtual cluster crashes or is restarted will lose session information and Glideins will be unable to rejoin with expired C-CRED
- Way forward
 - Configurable credential lifetime as workaround
 - Discussion with HTCondor to provide a refresh mechanism for tokens (there was one for x509)
 - Considered implementing credential refresh in GlideinWMS
 - Glidein fetching updates from Factory via service/refresh token
 - Use HashiCorp Vault 



Summary

- GlideinWMS token and hybrid support in production since end of 2021, still improving
- When generalizing credentials (e.g. using tokens) remember to consider
 - Time and space granularity
 - Type and purpose
 - Multiple credentials
 - Dynamic/Late credentials generation/request
- Added refactored credential handling in library for GlideinWMS services
 - Better code with classes and patterns
 - Type and purpose allow clear handling and matching of credentials and parameters
 - Custom and built-in generators allow dynamic and late credentials and simplify configuration
 - Credentials renewal is still a work in progress

<https://github.com/glideinWMS/glideinwms>

