Fermilab

June 11, 2026

GlideinWMS Improvements and Challenges

Marco Mambelli for the GlideinWMS team

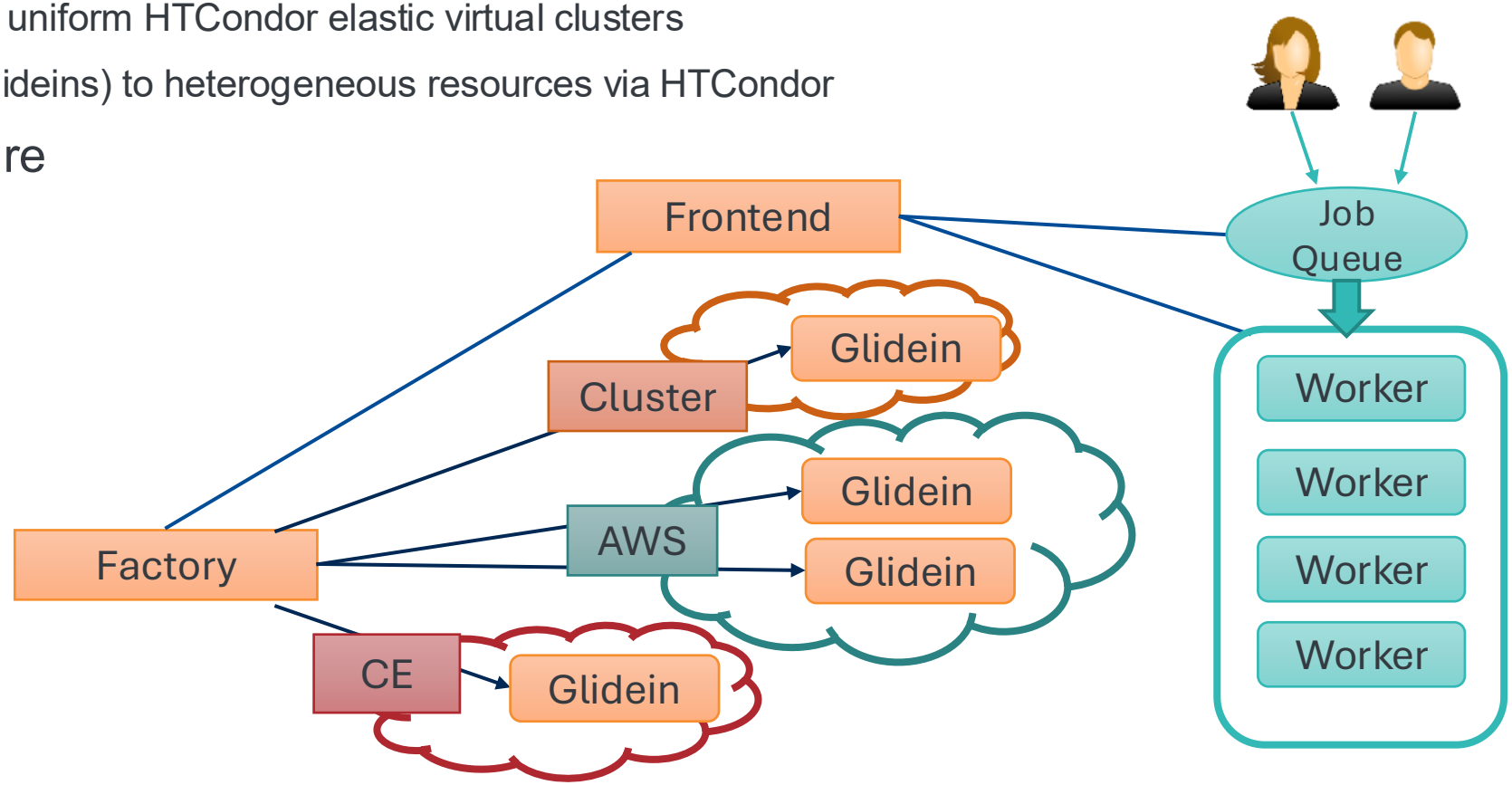
Throughput Computing Week 2026

U.S. DEPARTMENT of ENERGY

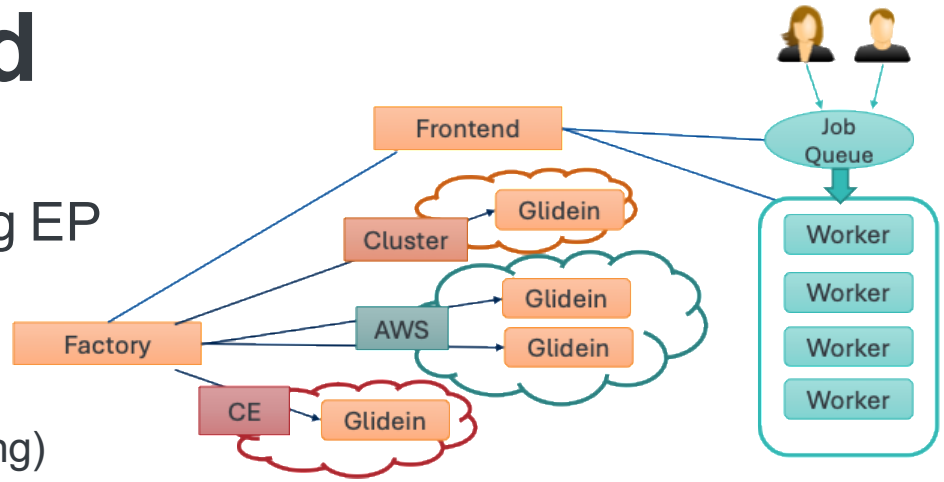
Fermi National Accelerator Laboratory is managed by FermiForward for the U.S. Department of Energy Office of Science

Glidein Workload Management System

- GlideinWMS is a pilot-based resource provisioning tool for distributed High Throughput Computing and leveraging heavily HTCondor:
  - Provides reliable and uniform HTCondor elastic virtual clusters
  - Submits pilot jobs (Glideins) to heterogeneous resources via HTCondor
- Distributed architecture
  - Glidein
  - Factory
  - Frontend

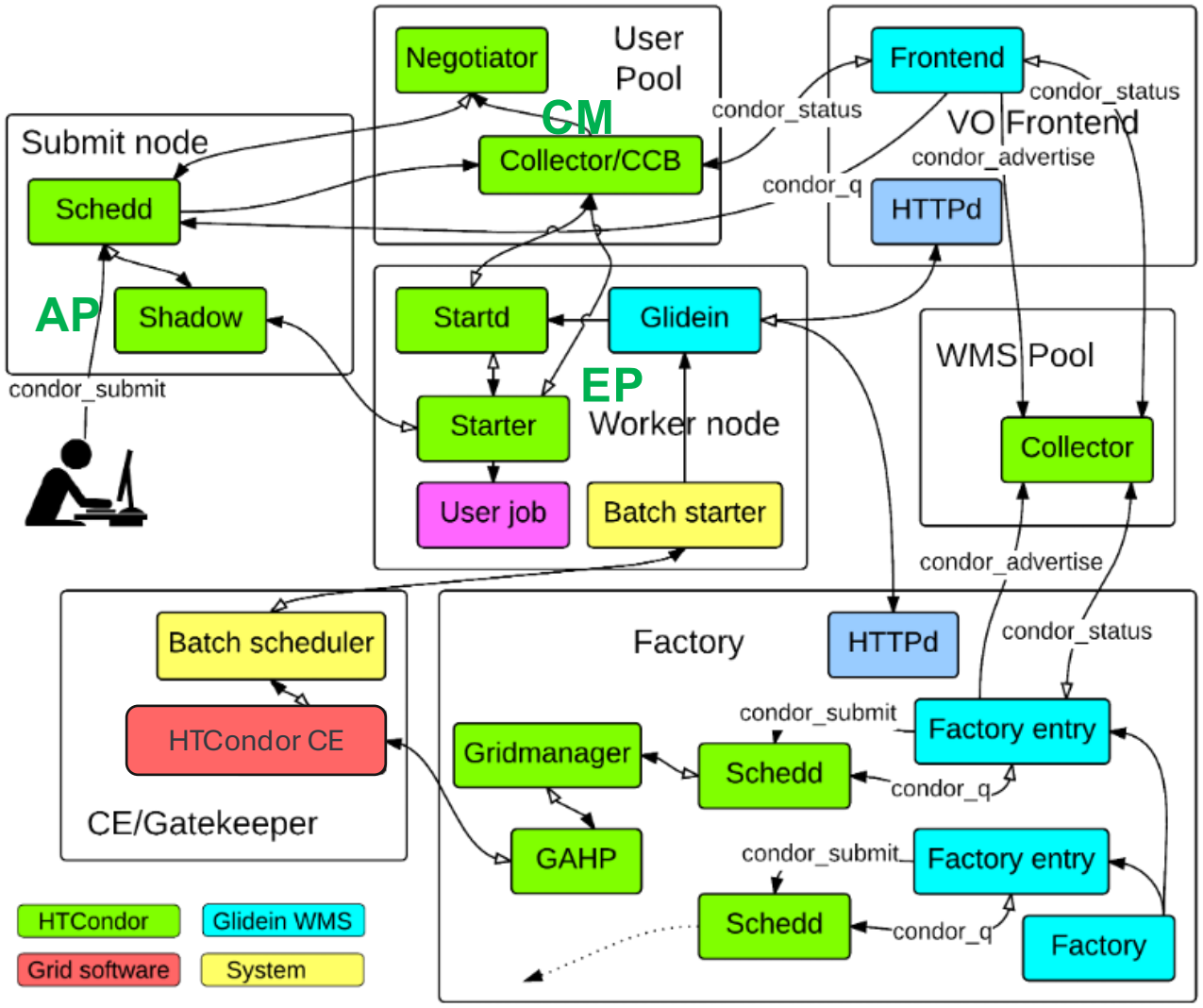


Glidein, Factory and Frontend



- Glidein (Pilot): node testing and customization, starting EP
  - Scouts for resources and validates the Worker node
  - Customizes the Worker node
  - Can manage/partition the Worker Node (space and time sharing)
  - Provides a reliable and customized Execution Point to an HTCondor Pool
- Factory: Glidein submission
  - A Glidein Factory knows how to submit to sites
  - Site entries described in the configuration (static or auto-generated)
  - HTCondor does the heavy lifting of submissions: to local and remote clusters, CEs, hosted CEs, clouds, and HPC resources
- Frontend: resource provisioning decisions
  - Monitors jobs to see how many Glideins are needed for all managed pools
  - Requests Glideins submission/removal from the Factory depending on jobs needs and available resources
  - Pressure-based system
  - Manages credentials and delegates them to the Factory and Glidein

Zoom-in on HTCondor components



Contained operation effort

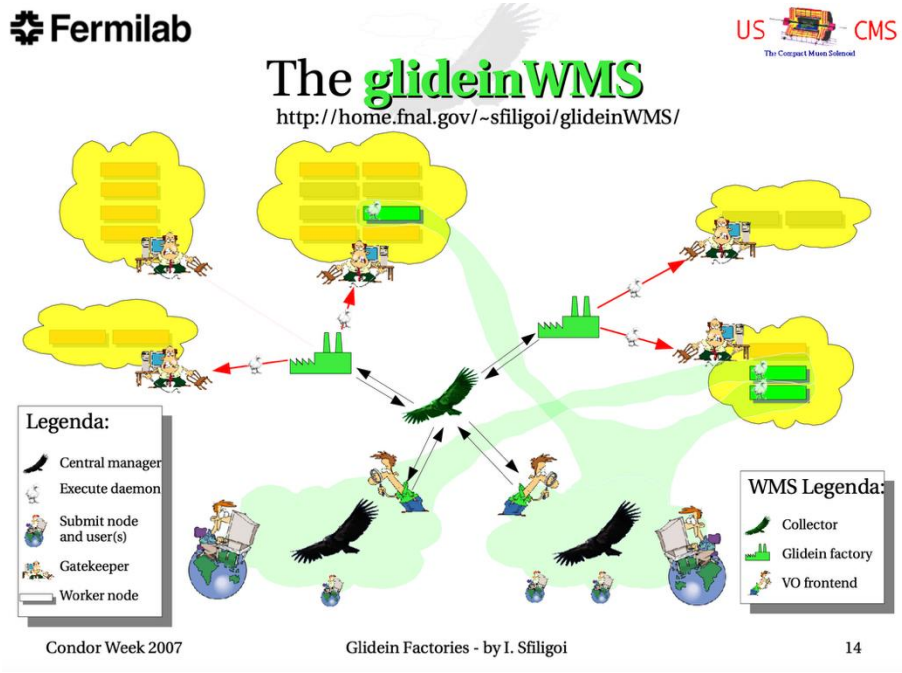
- Big VO operation (Shared Factories and CMS resources)
  - About 2.5 FTE
  - Most of it to develop monitoring and improve the system
- VO operation (Frontend and one Pool)
  - About 0.1 FTE
  - Most of that is monitoring and checking on sites

Evolved with HTCondor and Python

Thanks for the support!

- Almost 20 years old (October 2006).
  - CDF Analysis Facility using Glideins (2005)
  - Used for CMS analysis and production, later FIFE and DUNE
  - RPM package, later also container

Thanks to VDT and OSG!



- v2 and v3 (2009-12) monitoring, Python logging and new Factory API
- v3.5 (2019) Factory using a single user (no more condor switchboard)
- v3.7/v3.7.1 (2020) Token demonstrator
  - HTCondor token-auth->IDTOKENS+SciTokens
- v3.9/v3.9.3 (2020) Python 3

https://research.cs.wisc.edu/htcondor/CondorWeek2007/presentations/igor_GlideinFactories_final.pdf

Current/next steps

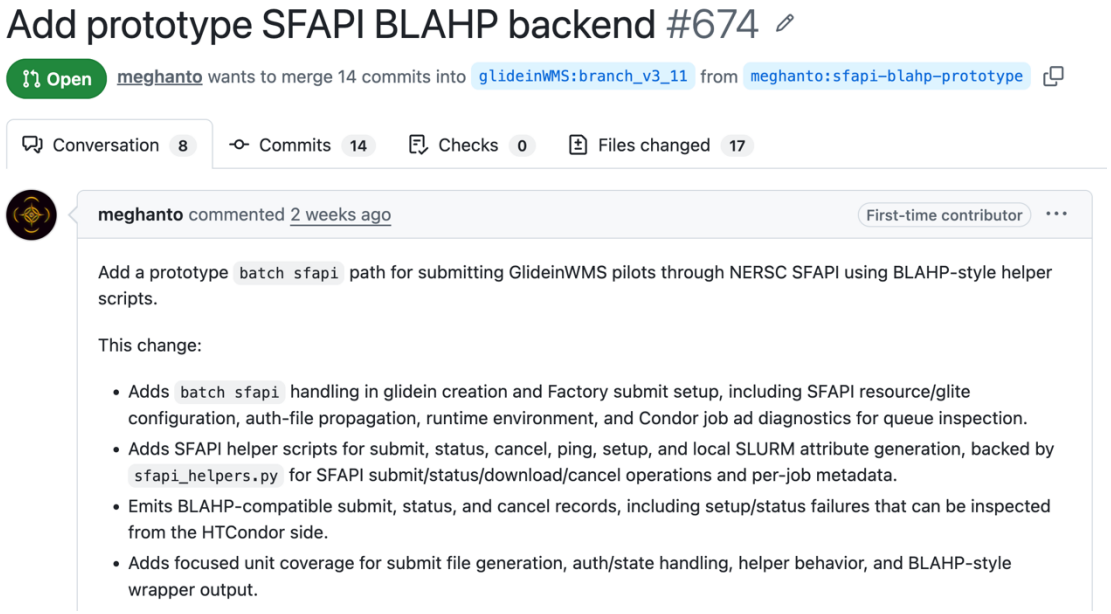
- v3.11 (2025) New credentials model and generators
- v3.11.5 (2026) On-demand CVMFS provisioning
- v3.11.6 (2026) condor_ssh_to_job in containers
- v3.11.6 (2026) Support for more HPC resources

Glidein Submission via HTCondor

- 2006 HTCondor Pool (vanilla universe)
- 2006 Grid (HTCondor-G/GRAM CE gt2, then gt5 - grid universe, gt2/gt5) up to HTC 9.x
- 2006-16 Batch systems (grid universe, batch (pbs, lsf, sge, or slurm)
- 2009-21 NorduGrid ARC CE, then v1 and finally REST (grid universe, arc)
- 2014-16 AWS, OpenStack/OpenNebula (grid universe, ec2)
- 2014 Remote cluster, aka BOSCO (ssh bridge - grid universe, batch and remote_cluster)
- 2015 HTCondor-CE/HTCondor-C (grid universe, condor)
- 2016 GCE (grid universe, gce)
- 2019 Azure (grid universe, azure)
- 2019 Split-starter and Lumberjack for no network HPC (only prototypes, no production)
- 2026? More HPC sites
  - NERSC Superfacility API
  - Integrated Research Infrastructure (IRI) API
  - Globus Compute

SuperFacility API support

- Used already for monitoring allocations, need to start using it for job management
- Code under development by Meghanto Majumder
- Thanks to the Pegasus team for sharing documentation and reference implementation
  - https://pegasus.isi.edu/docs/5.1.3-dev.0/user-guide/deployment-scenarios.html#nersc-perlmutter-via-sfapi
  - https://pegasus.isi.edu/docs/5.1.3-dev.0/reference-guide/nersc-sfapi-submission.html
- Test from development machines at Fermilab to Perlmutter at NERSC



Add prototype SFAPI BLAHP backend #674

Open meghanto wants to merge 14 commits into glideinWMS:branch_v3_11 from meghanto:sfapi-blahp-prototype

Conversation 8 Commits 14 Checks 0 Files changed 17

meghanto commented 2 weeks ago First-time contributor

Add a prototype `batch sfapi` path for submitting GlideinWMS pilots through NERSC SFAPI using BLAHP-style helper scripts.

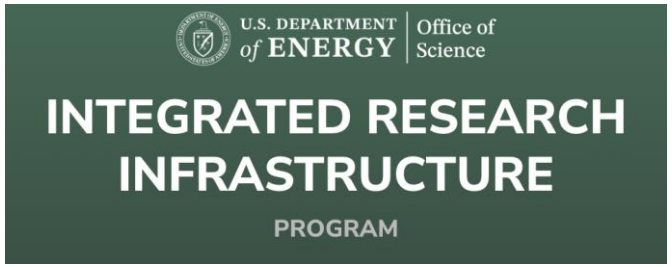
This change:

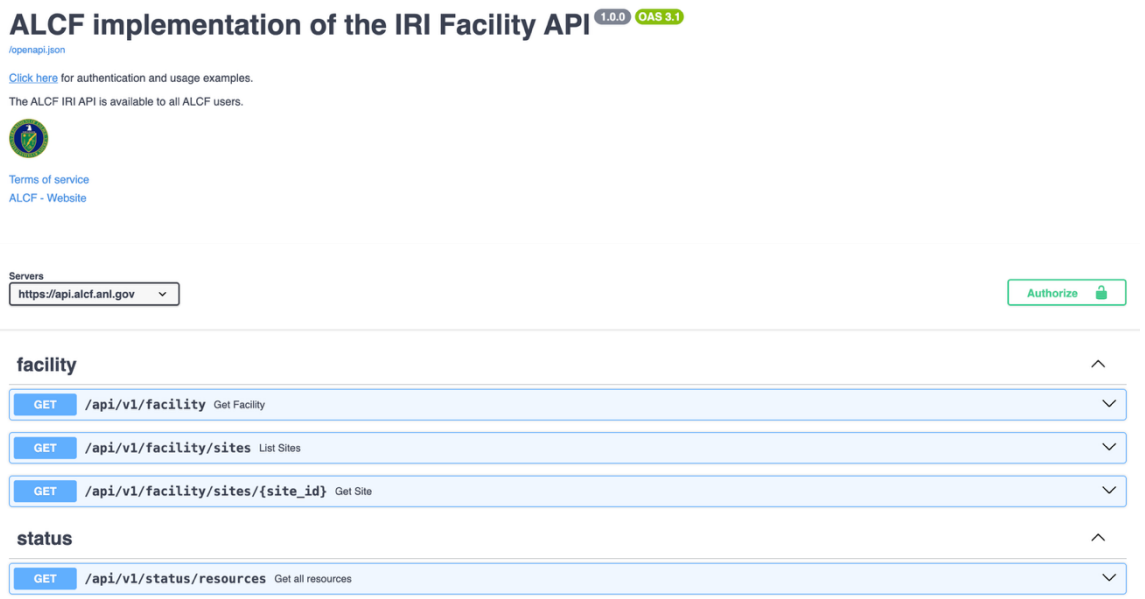
- Adds `batch sfapi` handling in glidein creation and Factory submit setup, including SFAPI resource/glite configuration, auth-file propagation, runtime environment, and Condor job ad diagnostics for queue inspection.
- Adds SFAPI helper scripts for submit, status, cancel, ping, setup, and local SLURM attribute generation, backed by `sfapi_helpers.py` for SFAPI submit/status/download/cancel operations and per-job metadata.
- Emits BLAHP-compatible submit, status, and cancel records, including setup/status failures that can be inspected from the HTCondor side.
- Adds focused unit coverage for submit file generation, auth/state handling, helper behavior, and BLAHP-style wrapper output.

```
Name                                    OpSys     Arch    State     Activity LoadAv Mem   ActivityTime

slot1@glidein_1709526_362407216@x1306c7s6b0n1h0.chn.perlmutter.nersc.gov LINUX     X86_64  Unclaimed Idle      0.000 515140  0+00:00:00

                     Total Owner Claimed Unclaimed Matched Preempting  Drain Backfill BkIdle

X86_64/LINUX     1     0     0         1       0          0     0        0      0

       Total     1     0     0         1       0          0     0        0      0
```

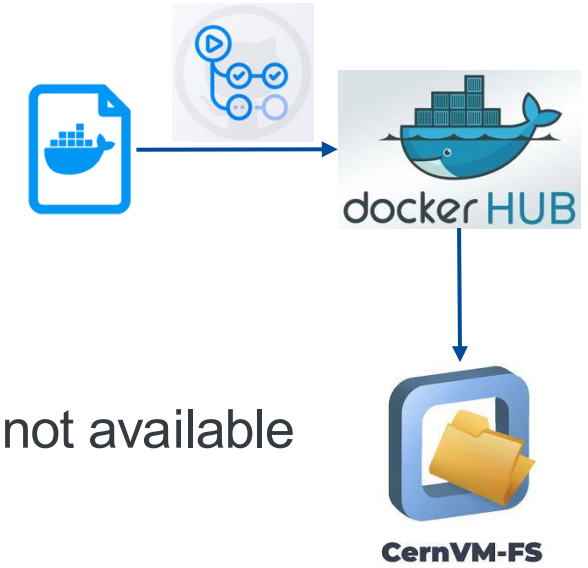
Integrated Research Infrastructure (IRI) API

- Common API to submit and monitor ASCR facilities (ALCF, OLCF, NERSC)
- https://github.com/doe-iri/iri-facility-api-python
- Planning to run some tests at least at NERSC
  - Using ALCF will add connectivity problems
- Long term ask is for HTCondor to support the IRI API





On-demand CVMFS provisioning



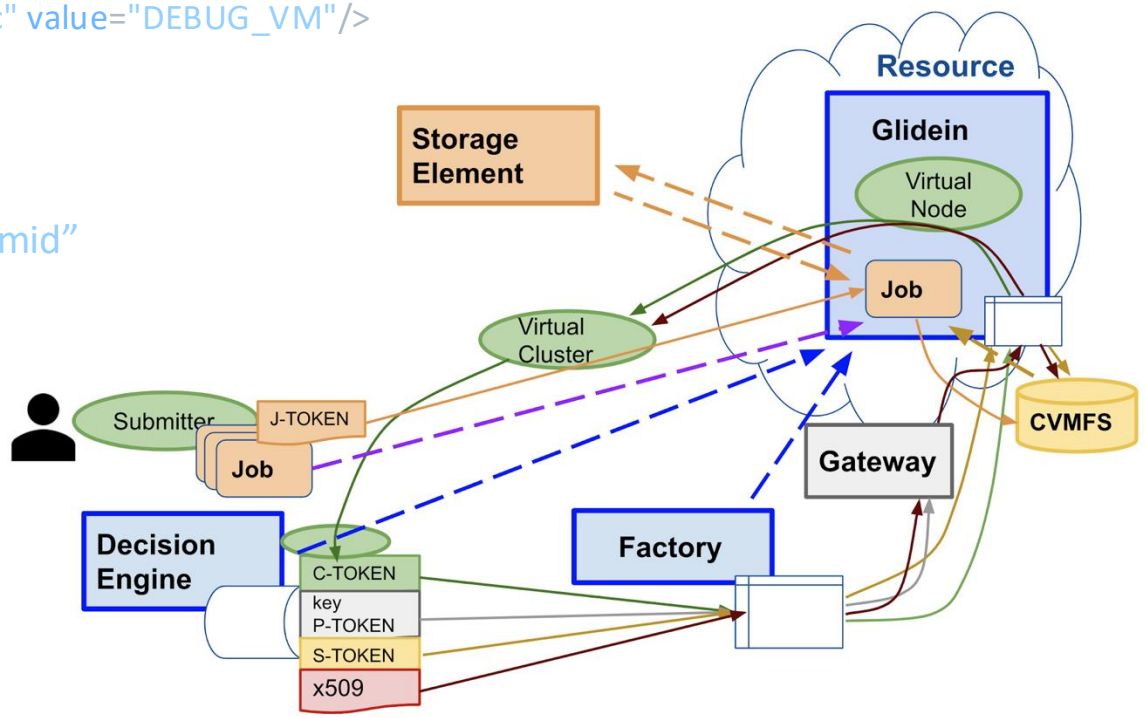
- VOs specify CVMFS requirements
- Resources may have CVMFS available
- cvmfsexec used to provision CVMFS on-demand when required and not available
  - Different cvmfsexec options used depending on OS and capabilities
    - cvmfsexec (mode 2 and 3)
    - mountrepo/umountrepo (mode 1) – nonstandard mountpoint
  - System and VO-defined list of repositories
- Apptainer allows to bind the repositories on the well-known /cvmfs mount point

New credentials and generators (v3.11)

- Multiple credentials and parameters
  - Frontend:

```
<credentials>
        <credential absfname="/opt/gwms/credentials/pilot.scitoken" purpose="request"
                    security_class="frontend" trust_domain="grid" type="scitoken"/>
        <credential absfname="dynamic_credential.py" purpose="payload"
                    security_class="frontend" trust_domain="grid" type="generator"/>
</credentials>
<parameters>
        <parameter name="ProjectId" type="generator" value="dynamic_parameter.py"/>
        <parameter name="VMId" type="static" value="DEBUG_VM"/>
</parameters>
```

  - Factory: auth_method="scitoken,grid_proxy;projectid;vmid"



Easier configuration with generators - old

Project ID is “allocation1” for SiteA, “allocation2” for SiteB, not needed otherwise

```
<security …
 <credentials />
</security>
<group name="main" enabled="True">
 <config ignore_down_entries="True" ramp_up_attenuation="3">
 …
 </config>
 <match match_expr="True" start_expr="True" policy_file="/path/to/python-policy-file">
  <factory query_expr="(GLIDEIN_Site!="SiteA) && (GLIDEIN_Site!="SiteB)” >
   <match_attrs />
   <collectors />
  </factory>
  <job query_expr="True">
   <match_attrs />
   <schedds />
  </job>
 </match>
 <security>
  <credentials>
   <credential absfname="/etc/osg/tokens/my_token.scitoken" security_class="frontend"
     trust_domain="OSG" type="scitoken"    />
  </credentials>
 </security>
 <attrs />
 <files />
</group>
```

```
<group name="group1" enabled="True"> …
<match match_expr="True" start_expr="True" policy_file="/path/to/python-policy-file">
  <factory query_expr="(GLIDEIN_Site=="SiteA)”>
   <match_attrs />
   <collectors />
  </factory>
  <job … />
</match>
 <security>
  <credentials>
   <credential absfname="/etc/osg/tokens/my_token.scitoken" security_class="frontend"
     trust_domain="OSG" type="scitoken" project_id="allocation1" />
  </credentials>
 </security> …
</group>
<group name="group2" enabled="True"> …
<match match_expr="True" start_expr="True" policy_file="/path/to/python-policy-file">
  <factory query_expr="(GLIDEIN_Site=="SiteB")">
   <match_attrs />
   <collectors />
  </factory>
  <job … />
</match>
 <security>
  <credentials>
   <credential absfname="/etc/osg/tokens/my_token.scitoken" security_class="frontend"
     trust_domain="OSG" type="scitoken" project_id="allocation2" />
  </credentials>
 </security> …
</group>
```

Easier configuration with generators - new

Project ID is “allocation1” for SiteA, “allocation2” for SiteB, not needed otherwise

```
<security …
 <credentials>
    <credential absfname="/etc/osg/tokens/my_token.scitoken" security_class="frontend" trust_domain="OSG" type="scitoken" />
 </credentials>
 <parameters>
  <parameter name="ProjectId" type="generator" value="EntryConditionGenerator”
                        context=“{‘discriminator’=‘name’, dict={‘SiteA’:’allocation1’, ‘SiteB’:’allocation2’}, default=‘’} />
 </parameters>
</security> …
<group name="main" enabled="True">
 <config ignore_down_entries="True" ramp_up_attenuation="3">
 …
 </config>
 <match match_expr="True" start_expr="True" policy_file="/path/to/python-policy-file">
  <factory query_expr=”True">
   <match_attrs />
   <collectors />
  </factory>
  <job query_expr="True">
   <match_attrs />
   <schedds />
  </job>
 </match>
 <security>
  <credentials/>
 </security>
 <attrs />
 <files />
</group>
```

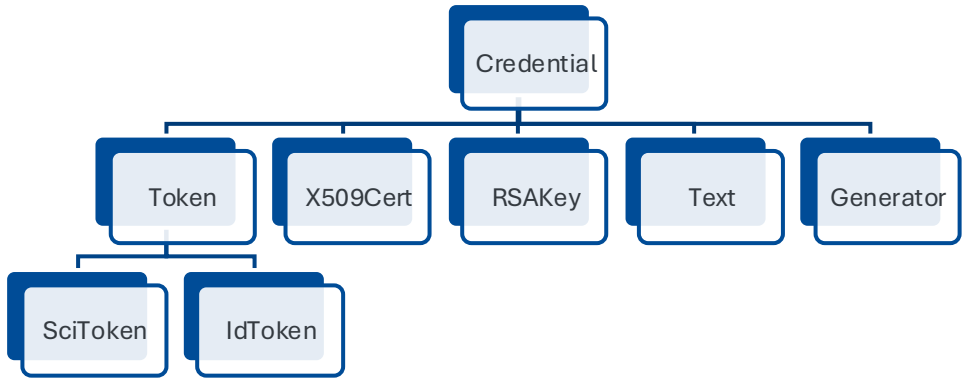
Shorter!

No duplication and consistency problems

Also with new credentials and generators

- Refactoring of the code
- Easy to define custom generators in Python
- IdTokenGenerator to customize the tokens to join the cluster
- Self-signed SciTokens for testing
  - Generates SciTokens according to the VO desires and uses its key to sign them
  - Advertises the public key with standard OAuth 2.0 authorization server metadata RFC
  - Just add a couple of lines in the CE mapfile

- Robust renewal of tokens missing
  - Problem when CM restarts – Glideins cannot rejoin the pool



GlideinWMS simplifies resource provisioning

- Glidein based Workload Management System used in production for 20 years
- Aiming to improve the provisioning and simplify operations
- Currently
  - Providing access to more HPC resources
  - Making CVMFS available on all nodes
  - Simplifying operations around credentials and submit parameters
- Thanks to the Condor team for all the support, we will keep requesting more performance and features to scale and satisfy GlideinWMS stakeholders' requests
  - Support of more HPC resources using IRI API and similar
  - Credential renewal mechanism working for all resources