Slurm’s REST API

(aka slurmrestd)

Nathan Rini

SC'24

Questions?

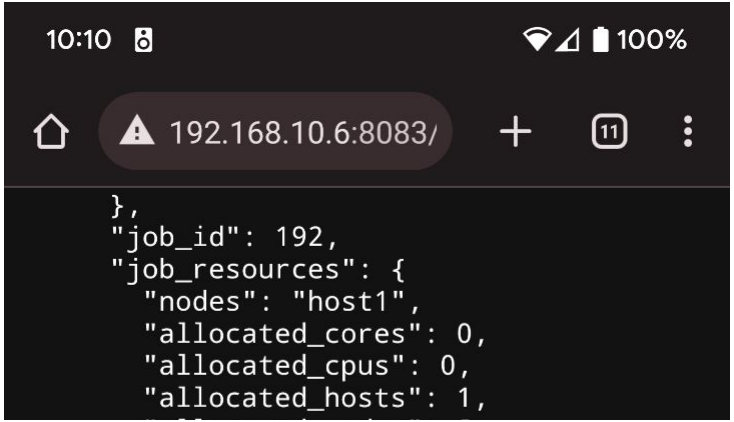
- Please feel free to interrupt at any time with questions.
  - I will ask you wait if the question is answered later in the presentation.
- Comments and constructive complaints are always welcome
  - I may ask to defer discussion to finish the presentation on time or if question is not applicable to other sites.
- If you don't get your questions answered or you have more follow up questions, please open a ticket or find me after.
- Your feedback on slurmrestd matters to us and helps us with the future roadmap.

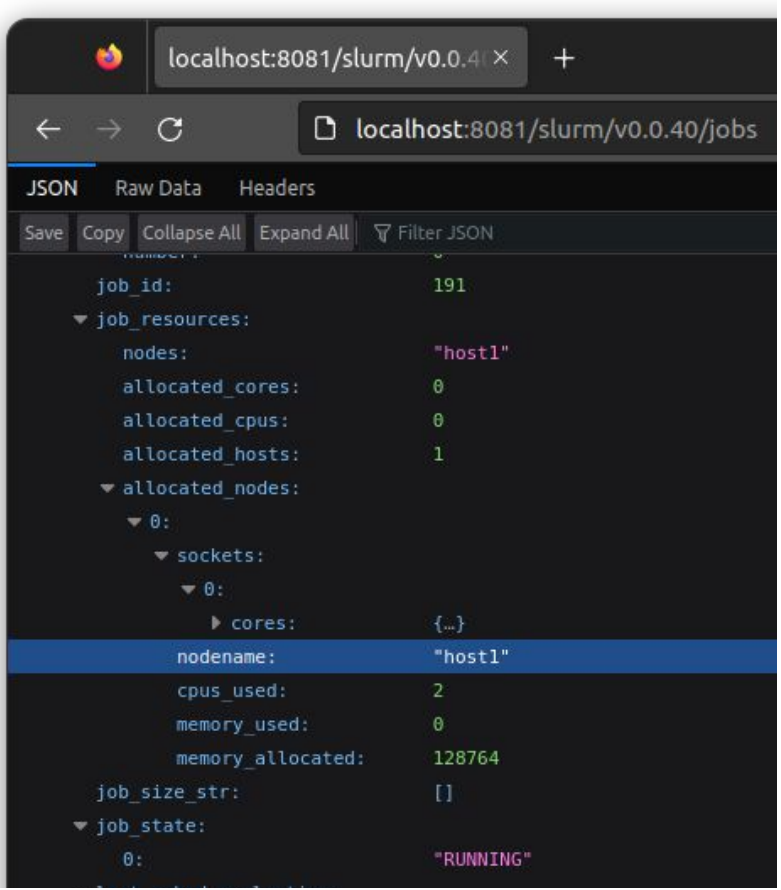
Intro to slurmrestd

What is the Slurm REST API?

Short answer:

Slurm without command line





What is the Slurm REST API?

- Slightly longer answer:
  - Allows users to query Slurm via HTTP requests (AKA a RESTful API)
    - Supports data formatted as JSON or YAML
  - OpenAPI v3 (aka Swagger v3) compliant to allow sites to easily generate clients
  - JSON Web Token (JWT) authentication for clients outside of MUNGE security perimeter
    - Allowing for partially trusted clients or using site authentication service

- Exhaustive answers with live demos can be covered during Slurm onsite trainings:
  - Please email sales@schedmd.com to set up a training session.

slurmrestd - documentation

- See the following documentation for detailed explanations of the contents discussed in the slides:
  - REST API Quick Start Guide
  - REST API Reference
  - REST API Client Writing Guide
  - REST API Generated Documentation from OpenAPI Schema
  - REST API Release Notes
  - SLURM Release NEWS
  - SLURM Release Notes
  - REST API Support matrix

We are trying to improve the documentation in every release. If something is missing, please open a ticket via bugzilla and we can look into documenting it.

Example use cases via CLI

Preparation: Setup shell

- Example bash functions query and post to slurmrestd via TCP socket:

```
#!/bin/bash
function rest_query()
{
	[ -z "$1" ] && echo "$0 {hostname:port} {Query Path}" >&2 && return 1
	export $(unset SLURM_JWT; scontrol token lifespan=10)
	curl -s -H "X-SLURM-USER-TOKEN:${SLURM_JWT}" -X GET "http://${1}/${2}"
}
function rest_post()
{
	[ -z "$1" ] && echo "$0 {hostname:port} {Query Path} {file to post}" >&2 && return 1
	export $(unset SLURM_JWT; scontrol token lifespan=10)
	mime_type="$(file -i ${3} | cut -d ' ' -f2 - | tr -d ';')"
	curl -s -H "X-SLURM-USER-TOKEN:${SLURM_JWT}" -H "Content-Type: ${mime_type}" -X POST \
		--data-binary "@${3}" "http://${1}/${2}"
}
export -f rest_query rest_post
```

- User name is only required to act as a proxy, otherwise user encoded in token in used:

```
-H "X-SLURM-USER-NAME:$(whoami)"
```

- Make sure to always call `--data-binary` and not `--data` when using curl to avoid the payload being corrupted by curl's auto type conversion.

Query jobs information

- Get job_id of all jobs known to slurmctld:

```
$ rest_query localhost:8080 slurm/v0.0.42/jobs | jq -r '.jobs[].job_id'
193
194
```

- Get state of first Array Job task with state of Job 194_1:

```
$ rest_query localhost:8080 slurm/v0.0.42/job/194_1 | jq -r
'.jobs[].job_state[]'
PENDING
```

- Get total number of tasks of all running jobs:

```
$  rest_query localhost:8080 slurm/v0.0.42/jobs | jq -r '.jobs[] |
select(.job_state[] == "RUNNING") | .tasks.number' | awk '{ sum += $1; } END
{ print sum; }'
10
```

Example Job description

- job.json:

```
{
  "script": "#!/bin/bash\nsrun uptime",
  "job": {
      "environment": [
      "PATH=/bin/:/usr/bin/:/sbin/"
      ],
      "account": "test",
      "name": "test slurmrestd job",
      "current_working_directory": "/tmp/",
      "memory_per_node": {
      "set": true,
      "number": 100
      },
      "tasks": 5,
      "nodes": "2-10"
  }
}
```

Example Array Job description

- array_job.json:

```
{
  "script": "#!/bin/bash\nsrun uptime",
  "job": {
      "environment": [
      "PATH=/bin/:/usr/bin/:/sbin/"
      ],
      "current_working_directory": "/tmp/",
      "account": "test",
      "array": "100",
      "name": "test slurmrestd array job",
      "memory_per_node": {
      "set": true,
      "number": 100
      },
      "tasks": 5,
      "nodes": "2-10"
  }
}
```

Example HetJob description

- het_job.json:

```
{
  "script": "#!/bin/bash\nsrun uptime",
  "jobs": [
      {
      "environment": [
      "PATH=/bin/:/usr/bin/:/sbin/"
      ],
      "current_working_directory": "/tmp/",
      "account": "test",
      "name": "test slurmrestd job",
      "memory_per_node": {
      "set": true,
      "number": 100
      },
      "tasks": 5,
      "nodes": "2-10"
      },
      {
      "memory_per_node": {
      "set": true,
      "number": 15
      },
      "tasks": 1,
      "nodes": "1",
      "current_working_directory": "/tmp/",
      "environment": [
      "PATH=/bin/:/usr/bin/:/sbin/"
      ]
      },
      {
      "current_working_directory": "/tmp/",
      "nodes": "1",
      "environment": [
      "PATH=/bin/:/usr/bin/:/sbin/"
      ]
      }
  ]
}
```

Submit example jobs

```
$ rest_post localhost:8080 /slurm/v0.0.42/job/submit job.json | jq -r
'.job_id'
231

$ rest_post localhost:8080 /slurm/v0.0.42/job/submit array_job.json | jq -r
'.job_id'
232

$ rest_post localhost:8080 /slurm/v0.0.42/job/submit het_job.json | jq -r
'.job_id'
233
```

Control Jobs

- Cancel a job:

```
$ rest_query localhost:8080 slurm/v0.0.42/job/236 -X DELETE
```

- Signal a job with SIGINT:

```
$ rest_query localhost:8080 slurm/v0.0.42/job/236?signal=SIGINT -X DELETE
```

- Change number tasks in a job:
  - change_job.json

```
{"tasks": 10}
```

```
$ rest_post localhost:8080 slurm/v0.0.42/job/239 change_job.json
```

Example use cases via Python Client

Preparation: Compile and install generated OpenAPI client

- Prerequisite:
  - Install openapi-generator-cli
- Compile and install library for client

```
$ slurmrestd --generate-openapi-spec -d v0.0.42 -s
slurmdbd,slurmctld > openapi.json
$ openapi-generator-cli generate -i openapi.json -g python -o
py_api_client
$ virtualenv test
$ source test/bin/activate
$ cd py_api_client/
$ pip install -r requirements.txt .
```

Note: openapi-generator.tech python generator's recent change to pydantic caused significant syntax changes in field naming and structures from prior v0.0.40 examples.

Preparation: start and configure python interactive

- Run python3 in interactive mode and setup environment for all examples:

```
$ python3
from pprint import pprint
import openapi_client
import subprocess
import os
import re
from openapi_client import ApiClient as Client
from openapi_client import Configuration as Config
c = openapi_client.Configuration(host = "http://localhost:8080/",
access_token = subprocess.run(['scontrol', 'token',
'lifespan=9999'], check=True, capture_output=True,
text=True).stdout.replace('SLURM_JWT=','').replace('\n',''))
slurm = openapi_client.SlurmApi(openapi_client.ApiClient(c))
slurmdb = openapi_client.SlurmdbApi(openapi_client.ApiClient(c))
```

Inspection of generated OpenAPI client

- HTTP methods are implemented as functions in slurm and slurmdb objects:

```
print([a for a in dir(slurmdb) if re.match(r'slurmdb', a)])
['slurmdb_v0042_delete_account',
'slurmdb_v0042_delete_account_with_http_info',
'slurmdb_v0042_delete_account_without_preload_content',
'slurmdb_v0042_delete_association',
'slurmdb_v0042_delete_association_with_http_info',
'slurmdb_v0042_delete_association_without_preload_content',
'slurmdb_v0042_delete_associations',
...(truncated)...

print([a for a in dir(slurm) if re.match(r'slurm', a)])
['slurm_v0042_delete_job', 'slurm_v0042_delete_job_with_http_info',
'slurm_v0042_delete_job_without_preload_content', 'slurm_v0042_delete_jobs',
'slurm_v0042_delete_jobs_with_http_info',
'slurm_v0042_delete_jobs_without_preload_content', 'slurm_v0042_delete_node',
'slurm_v0042_delete_node_with_http_info',
...(truncated)...
```

Query jobs information

- Get job_id of all jobs known to slurmctld:

```
resp = slurm.slurm_v0042_get_jobs()
for job in resp.jobs:
     print(job.job_id)
```

- Get state of first Array Job task with state of all jobs known to slurmctld:

```
resp = slurm.slurm_v0042_get_jobs()
for job in resp.jobs:
    for state in job.job_state:
        print(state)
```

Query jobs information

- Get total number of tasks of all running jobs:

```
resp = slurm.slurm_v0042_get_jobs()
c=0
for job in resp.jobs:
        if 'RUNNING' in job.job_state:
                c += job.tasks.number
print(c)
```

Example Job description

- Submit example job:

```
from openapi_client.models.v0042_job_desc_msg import
V0042JobDescMsg as JobDesc
from openapi_client.models.v0042_job_submit_req import
V0042JobSubmitReq as JobReq

job = JobReq(script='#!/bin/bash\nsrun uptime',
job=JobDesc(environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/'],
current_working_directory='/tmp/'))
print(slurm.slurm_v0042_post_job_submit(job).job_id)
```

Example Array Job description

- Submit example job:

```
from openapi_client.models.v0042_job_desc_msg import
V0042JobDescMsg as JobDesc
from openapi_client.models.v0042_job_submit_req import
V0042JobSubmitReq as JobReq

job = JobReq(script='#!/bin/bash\nsrun uptime',
job=JobDesc(array='100',environment=['PATH=/bin/:/sbin/:/usr/bin/:/
usr/sbin/'],current_working_directory='/tmp/'))
print(slurm.slurm_v0042_post_job_submit(job).job_id)
```

Example HetJob description

- Submit example job:

```
from openapi_client.models.v0042_job_desc_msg import
V0042JobDescMsg as JobDesc
from openapi_client.models.v0042_job_submit_req import
V0042JobSubmitReq as JobReq

job = JobReq(script='#!/bin/bash\nsrun uptime', jobs=[
    JobDesc(environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/'],
current_working_directory='/tmp/'),
    JobDesc(environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/'],
    current_working_directory='/tmp/'),])
print(slurm.slurm_v0042_post_job_submit(job).job_id)
```

Control Jobs

- Cancel a job:

```
slurm.slurm_v0042_delete_job(job_id='3694')
```

- Signal a job with SIGINT:

```
slurm.slurm_v0042_delete_job(job_id='3694', signal='SIGINT')
```

Control Jobs

- Change job name:

```
from openapi_client.models.v0042_job_desc_msg import V0042JobDescMsg as
JobDesc

job = JobDesc(name='updated test job')
print(slurm.slurm_v0042_post_job(job_id='3697', v0042_job_desc_msg=job))
```

- Change job task count:

```
from openapi_client.models.v0042_job_desc_msg import V0042JobDescMsg as
JobDesc

job = JobDesc(tasks=15)
print(slurm.slurm_v0042_post_job(job_id='3697', v0042_job_desc_msg=job))
```

Slurm CLI: YAML & JSON

JSON and YAML for the command line

- Functionality from slurmrestd has been added to existing CLI commands to provide JSON and YAML output:

`sshare --json`	`sshare --yaml`
`sacct --json`	`sacct --yaml`
`sacctmgr --json`	`sacctmgr --yaml`
`scontrol --json`	`scontrol --yaml`
`sdiag --json`	`sdiag --yaml`
`sinfo --json`	`sinfo --yaml`
`sprio --json`	`sprio --yaml`
`squeue --json`	`squeue --yaml`

Query jobs information

- Get job_id of all jobs known to slurmctld:

```
$ squeue -json | jq -r '.jobs[].job_id'
193
194
```

- Get state of first Array Job task with state of all jobs known to slurmctld:

```
$ scontrol show job -json 194_1 | jq -r '.jobs[].job_state[]'
PENDING
```

- Get total number of tasks of all running jobs:

```
$ scontrol show jobs -json | jq -r '.jobs[] | select(.job_state[] ==
"RUNNING") | .tasks.number' | awk '{ sum += $1; } END { print sum; }'
10
```

Selecting data_parser plugin version (23.11+)

- CLI commands
  - –yaml/–json without an argument defaults to latest version (v0.0.42 on 24.11)

```
sinfo –json=v0.0.41
sinfo –json=v0.0.42
sinfo –json
```

```
sinfo –yaml=v0.0.41
sinfo –yaml=v0.0.42
sinfo –yaml
```

  - Set `DataParserParameters=v0.0.41` in slurm.conf to change default (24.11+)
- Slurmrestd (supports loading multiple data_parser plugins at once)
  - 24.05:

```
slurmrestd -d v0.0.39,v0.0.40,v0.0.41 -s slurmctld,slurmdbd
```

  - 24.11:

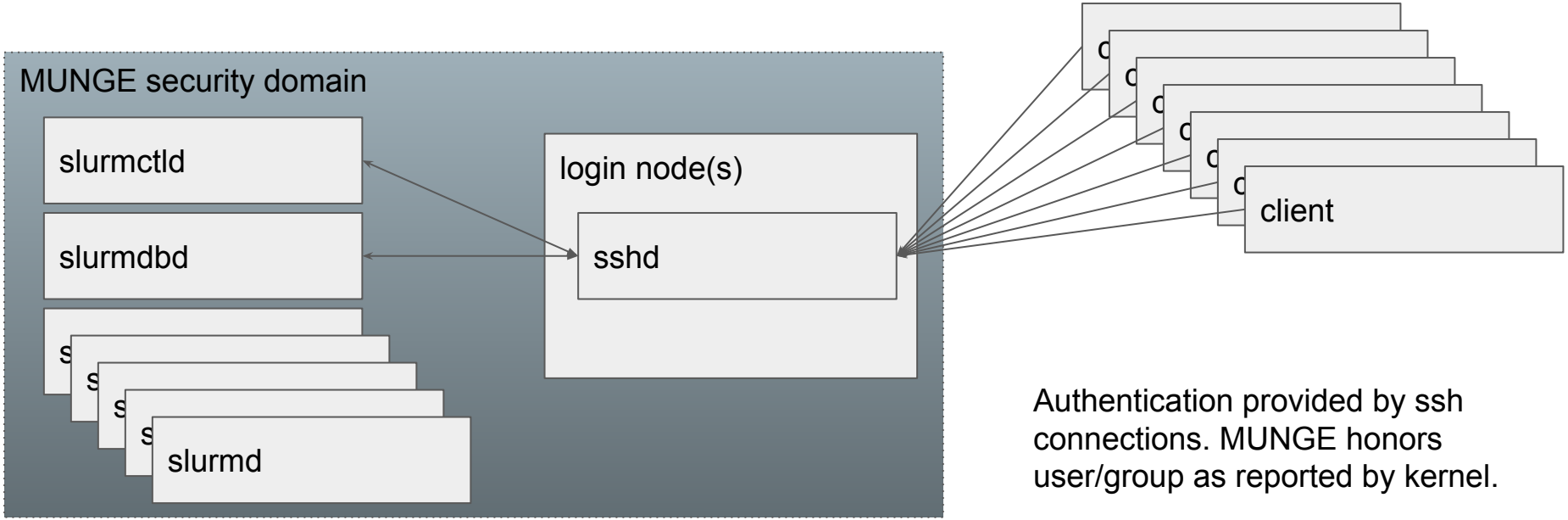
```
slurmrestd -d v0.0.41,v0.0.42 -s slurmctld,slurmdbd
```

Command Line - OpenAPI specification

- Requesting OpenAPI schema for output (23.11+, v0.0.40+)
  - `sinfo -json=v0.0.42+spec_only`
- Produces output similar to an OpenAPI schema to allow caller to know the format of the expected result.
  - *sinfo* has no equivalent request in slurmrestd.
- OpenAPI standard only applies to URL paths:
  - Only the schema for output is returned instead of a full OpenAPI specification

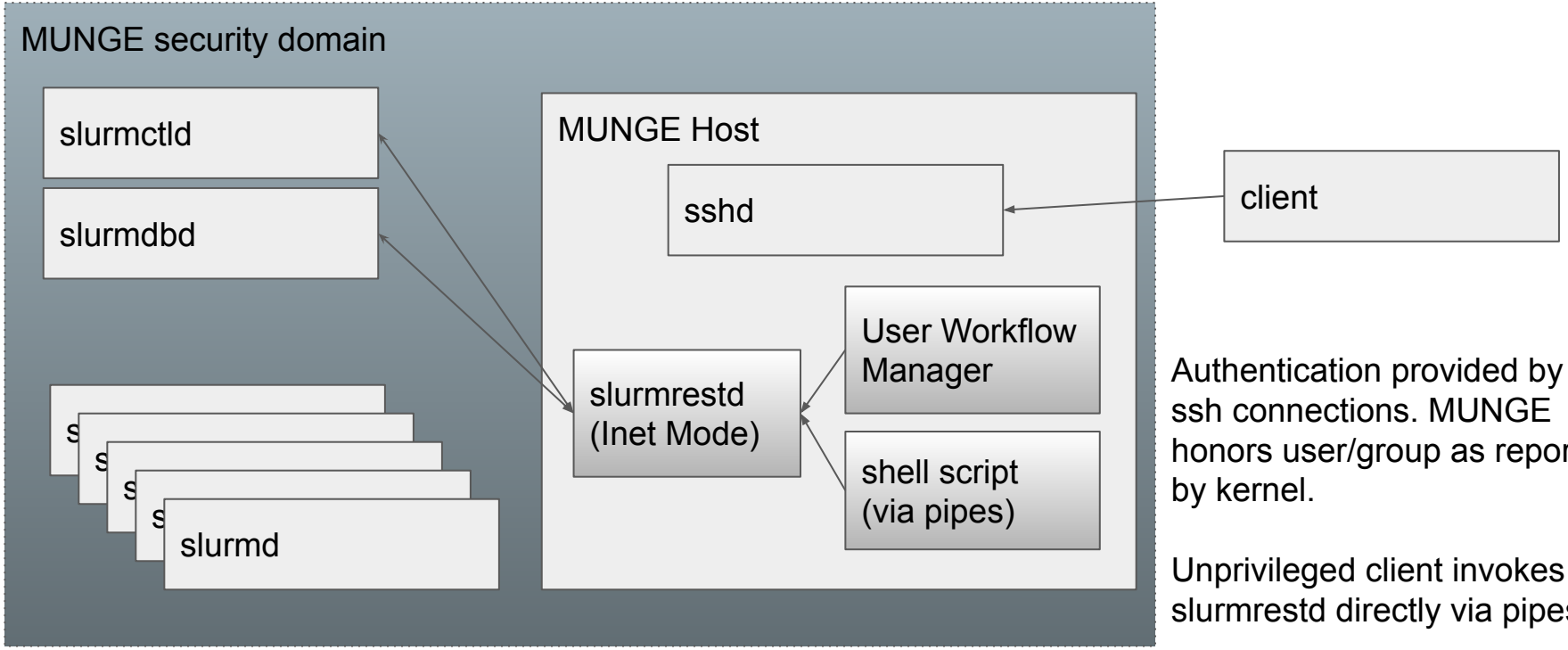
Ways to deploy Slurm’s REST API

MUNGE and SSH based Slurm (aka Slurm without REST API)



Authentication provided by ssh connections. MUNGE honors user/group as reported by kernel.

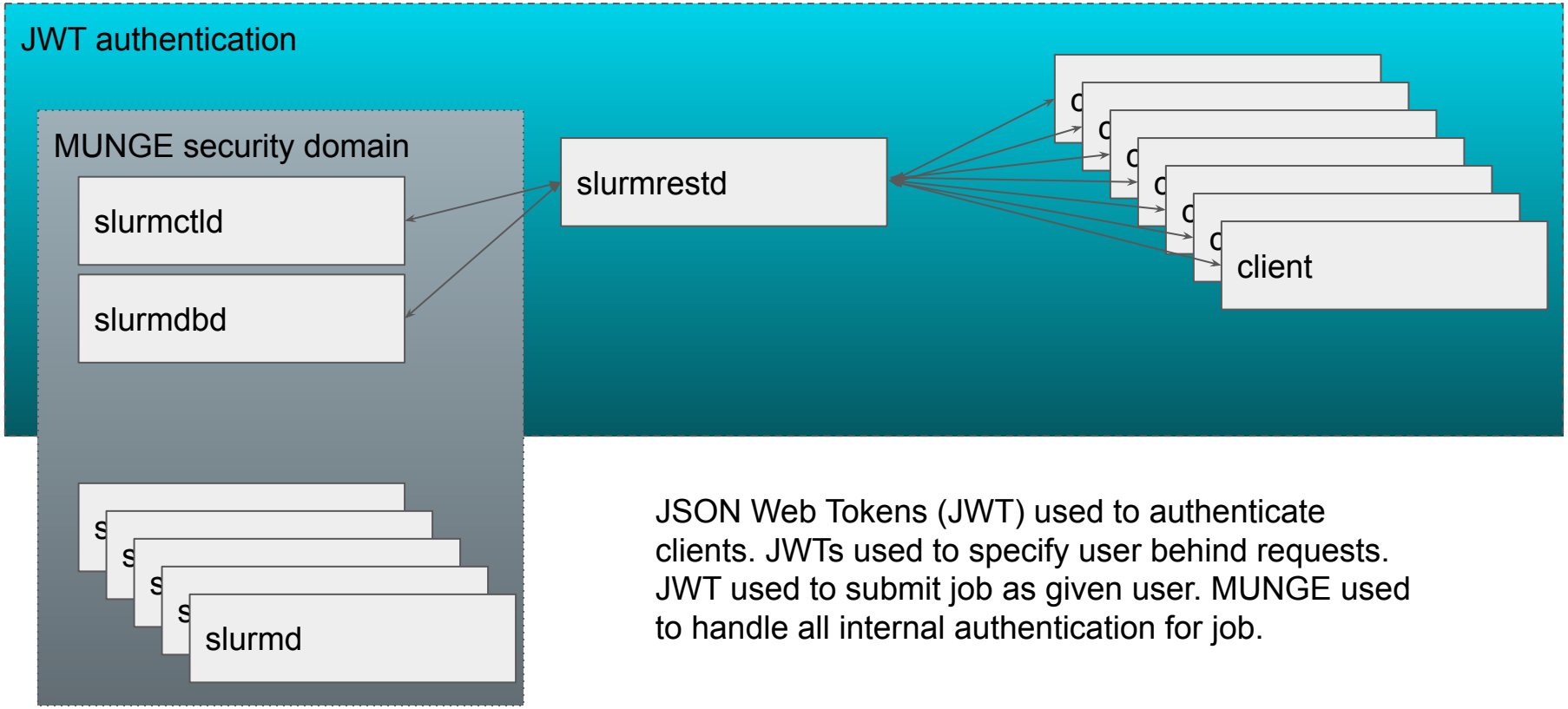
Slurm REST API using only MUNGE and command line



Authentication provided by ssh connections. MUNGE honors user/group as reported by kernel.

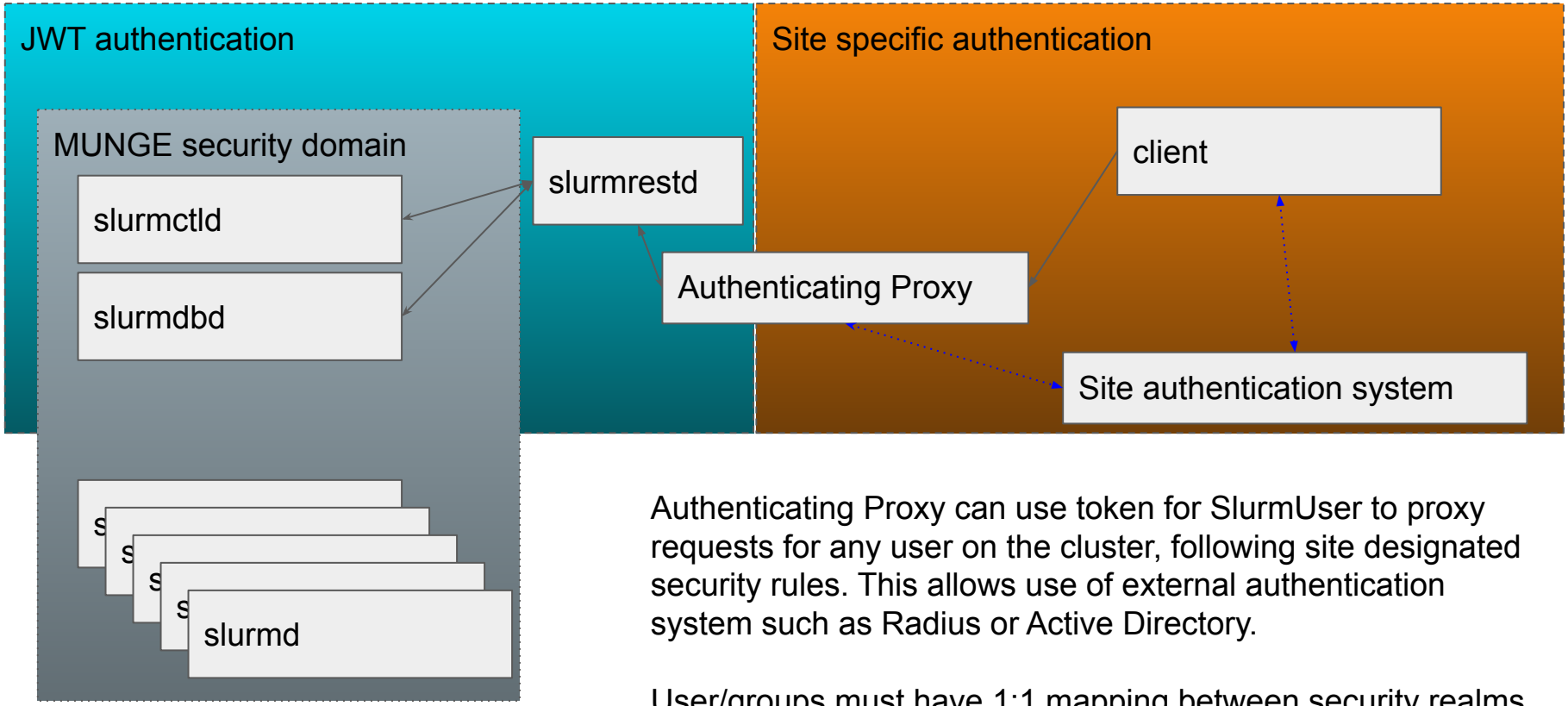
Unprivileged client invokes slurmrestd directly via pipes.

Slurm REST API using JSON web tokens in an existing cluster



JSON Web Tokens (JWT) used to authenticate clients. JWTs used to specify user behind requests. JWT used to submit job as given user. MUNGE used to handle all internal authentication for job.

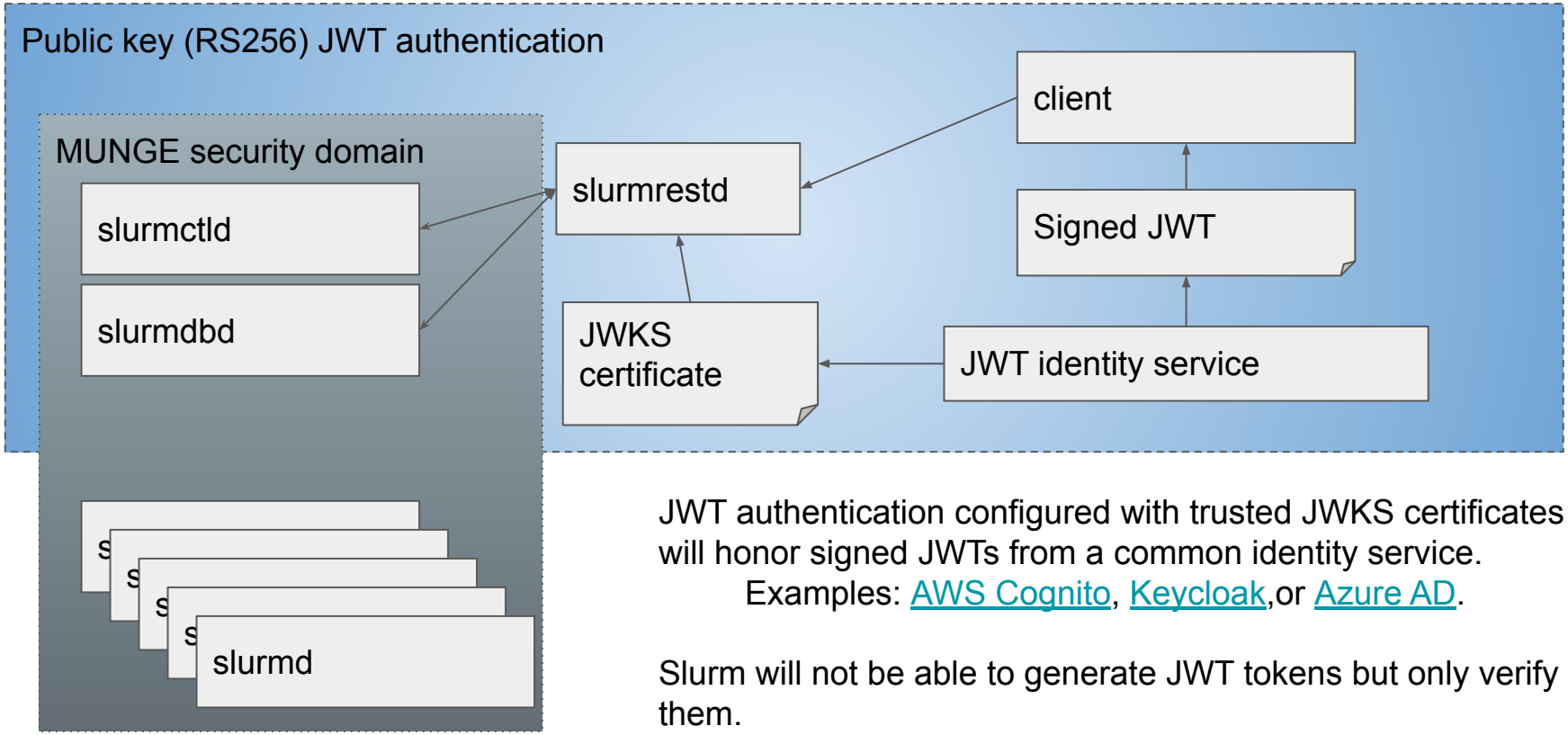
Slurm REST API for the whole site or even the Internet



Authenticating Proxy can use token for SlurmUser to proxy requests for any user on the cluster, following site designated security rules. This allows use of external authentication system such as Radius or Active Directory.

User/groups must have 1:1 mapping between security realms. All client connections must be TLS wrapped by proxy.

Slurm REST API from the Cloud

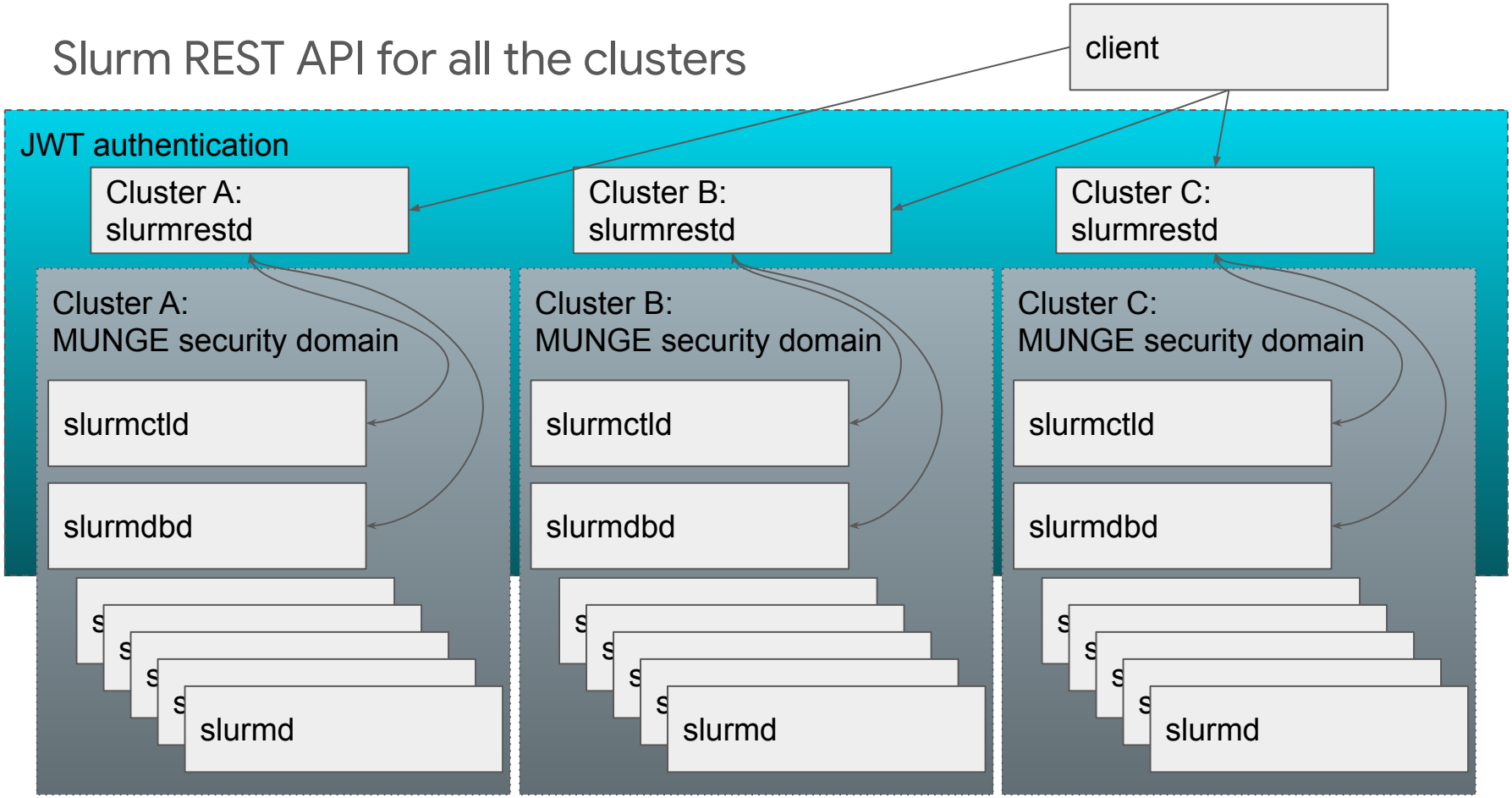


JWT authentication configured with trusted JWKS certificates will honor signed JWTs from a common identity service.

Examples: AWS Cognito, Keycloak,or Azure AD.

Slurm will not be able to generate JWT tokens but only verify them.

Slurm REST API for all the clusters



Setup

Slurm's JWT Authentication

- Setup procedure: https://slurm.schedmd.com/jwt.html
- RFC7519 compliant implementation
- Supported algorithms:
  - HS256 (shared secret)
  - RS256 (public key)
- Users and groups on cluster must match token's user exactly
- Implementation is limited and can not be used to communicate with slurmd daemons.
- How to generate token in Slurm (HS256 algorithm only):

```
$ scontrol token
SLURM_JWT=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

- Sites should consider generating JWTs outside of Slurm for automatic configuration of clients
  - Example: https://slurm.schedmd.com/jwt.html#compatibility
  - The SLURM_JWT environment variable should not be set in user environments.

slurmrestd - Slurm's REST API implementation

- How to compile
  - Follow normal configuration procedure first
  - slurmrestd will be automatically compiled if all prerequisites are present
    - ```
      checking whether to compile slurmrestd... yes
      checking for slurmrestd default port... 6820
      ```
  - Possible to explicitly request slurmrestd
    - ```
      ../configure --enable-slurmrestd
      ```
- slurmrestd is just another unprivileged binary callable by any user
  - Installed at *EPREFIX/sbin/slurmrestd*
    - Possible to change install path when calling configure:
      - ```
        ../configure --prefix=$NEW_INSTALL_PATH
        ```
      - ```
        ../configure --sbindir=$NEW_INSTALL_PATH
        ```
- slurmrestd must be able to communicate with slurmctld and slurmdbd via TCP connections.

slurmrestd - Invoked directly

- Call slurmrestd directly from a shell script or from a in-cluster workflow manager
  - Avoids requiring any new authentication for the cluster
  - Requires that client handle HTTP communications
- Example (truncated):

```
$ echo -e 'GET /slurm/v0.0.42/jobs HTTP/1.1\r\n' | slurmrestd
HTTP/1.1 200 OK
Content-Length: 8758
Content-Type: application/json

{
  "jobs": [
    {
      "job_id": 192,
      "job_state": [
        "RUNNING"
      ],
```

slurmrestd - Proxying

- slurmrestd is HTTP standards compliant
  - Any tool that can work with a web server should work with slurmrestd
- Sites are suggested to setup a proxy between clients and slurmrestd
  - Use Nginx, Apache, or proxy du-jour
- Add caching
  - Configure caching in proxy as desired
  - Reduce strain on slurmctld and slurmdbd for repeated requests
- Always wrap communications with TLS
  - **Never** directly expose slurmrestd (or any of Slurm) to the Internet.
- Use authentication proxy functionality in the proxy to use existing site authentication instead of MUNGE or Slurm's JWT implementation.
  - Example: https://github.com/naterini/docker-scale-out/tree/master/proxy
  - Avoid users having to grab a new JWT by calling `scontrol token`

slurmrestd - Running as a listening daemon

- Start daemon listening on IPv4 localhost TCP port 8080, IPv6 localhost TCP port 8080, IPv6 and IPv4 on all interfaces TCP port 8181, streaming Unix socket at /path/to/unix.socket with Slurm-24.11 content plugins only using JWT authentication for a Slurm-24.11 install.

```
$ env SLURM_JWT=daemon slurmrestd unix:/path/to/unix.socket 127.0.0.1:8080
ip6-localhost:8080 :8181 -a jwt -s slurmctld,slurmdbd -d v0.0.42
```

- Start daemon listening on IPv4 localhost TCP port 8080, IPv6 localhost TCP port 8080, IPv6 and IPv4 on all interfaces TCP port 8181, streaming Unix socket at /path/to/unix.socket with **Slurm-24.05** content plugins only using JWT authentication for a Slurm-24.11 install.

```
$ env SLURM_JWT=daemon slurmrestd unix:/path/to/unix.socket 127.0.0.1:8080
ip6-localhost:8080 :8181 -a jwt -s slurmdbd,slurmctld -d v0.0.41
```

slurmrestd - Running as a listening daemon via systemd

- Sites are encouraged to use slurmrestd.service for systemd
  - Compiled and placed in build directory as etc/slurmrestd.service
  - Drop-in unit should be used to change values instead of modifying template.
    - Make sure to always update the systemd unit during an upgrade
    - Using `systemctl edit slurmrestd.service` is suggested for editing drop-ins.
- Example setup procedure to have slurmrestd listen port 8080 on all interfaces:

```
cp $BUILD_PATH/etc/slurmrestd.service
/usr/lib/systemd/system/slurmrestd.service
echo ‘SLURMRESTD_LISTEN=:8080’ > /etc/default/slurmrestd
systemctl daemon-reload
systemctl start slurmrestd
```

Optimization

slurmrestd: Fast mode Parser (v0.0.40+,23.11+)

- Attempt to process queries as fast as possible by sacrificing warning checks and readability for humans.
- This flag should **only** be used for production systems with production clients that have already **been fully tested without** the fast flag active.
- Other enhancements may be added in future releases to improve processing speed.
- Example:

```
slurmrestd -d v0.0.40+fast -s slurmdbd,slurmctld $LISTEN_PORTS
```

slurmrestd: Compact JSON/YAML (v0.0.40+,23.11+)

- Generated JSON/YAML outputs are by default done with extra characters to improve readability.
  - For some sites with large number of requests to slurmrestd, skipping unnecessary whitespace characters can have considerable performance benefit in processing time and reduced network usage.
- Environment variables to activate compact mode:
  - `SLURMRESTD_YAML=compact`
  - `SLURMRESTD_JSON=compact`
- Example:

```
env SLURMRESTD_JSON=compact SLURMRESTD_YAML=compact slurmrestd
-d v0.0.42 -s slurmdbd,slurmctld $SLURMRESTD_LISTEN
```

Client compatibility

slurmrestd - Plugins lifetime matrix

- See REST API Support matrix for updates
- All paths in slurmrestd requests include the relevant plugin version:
  - Example: http://$HOST/slurmdb/**v0.0.42**/jobs

Added in Slurm release	Content Plugins (-s)	Data_parser Plugin (-d) [23.11+]	Removal in Slurm release
23.02	v0.0.39,dbv0.0.39	v0.0.39	24.11
23.11	slurmctld,slurmdbd	v0.0.40	25.05 (v0.0.40 only)
24.08		v0.0.41	25.11
24.11		v0.0.42	26.05
25.05		v0.0.43	26.11

- Unversioned slurmctld and slurmdbd content plugins added in Slurm-23.11 have no planned removal date.

Compatibility Testing

- slurmrestd is currently tested using:
  - Golang codegen
    - Used as client generator for Slinky (Slurm's Kubernetes project)
  - *openapi-generator-cli* generated python client
    - Tests use static driver code against generated python clients
    - New test units are required for each data_parser version and the major version of *openapi-generator-cli*.
    - Arguably the most popular client generator for OpenAPI due to heritage from Swagger.
  - *curl*
    - Direct queries of slurmrestd using hand crafted requests
    - Use of curl for site scripting is **not** advised
- Breaking changes of existing clients of the same version are considered a bug.
- General goal of reducing changes required for porting to newer versions.
  - Depending on the relevant change(s), requests in prior accepted formats may still be accepted but with warnings sent to client.

openapi-generator.tech: OpenAPI Standard Compliance

- *openapi-generator.tech* created clients can not handle or refuse unexpected data types
  - In most cases, the client will assert but others just result in a segfault.
- OpenAPI standard includes *oneOf()* and *anyOf()* operators to allow for polymorphism
  - Allows return of *null* when a field isn't set.
  - Slurm makes heavy use of polymorphism internally.
    - slurmrestd designed to handle polymorphic formats
- *openapi-generator.tech*'s generator is not monolithic
  - Uses a plugin based approach to create generators for many languages and some languages have more than one generator.
  - Clients for each language have varying level of OpenAPI standard support
- *openapi-generator.tech* generated clients will crash when handed (some) schemas using *oneOf()*. All usage of *oneOf()* has been removed (v0.0.37+) to avoid breaking clients.

To Infinity and... Assert!

- Slurm makes heavy use of Infinity or Unlimited, usually as a way to disable a limit.
- ECMA-404 JSON does not support a value of *infinity* (or *±infinity* or *±NaN*)
  - Most JSON parsers actually support *infinity*
    - Some silently convert to max of the internal type:

```
$ echo infinity | jq
1.7976931348623157e+308
```

  - OpenAPI standard does not support (or explicitly ban) use of *infinity*
    - *openapi-generator-cli* python client will assert upon receiving *infinity*
- slurmrestd supports *infinity* (and *NaN* which is not used)
  - slurmrestd can automatically convert "*inf*", "*+inf*", "*-inf*", "*infinity*", "*+infinity*", "*-infinity*" string values to OpenAPI number format for inputs.
    - Warnings will still be issued about non-compliance with OpenAPI specification for such format conversions for any given field.
    - slurmrestd should **not** output *infinity* or *NaN* to avoid breaking clients.

slurmrestd and the non-compliant clients?

- Several sites opened tickets against slurmrestd for broken clients
  - slurmrestd was written against the OpenAPI standard
    - In theory, the non-compliance to the OpenAPI standard of any one client should be fixed by clients.
    - This effectively set the bar for entry too high for most sites who were not writing their own clients.
- slurmrestd's workarounds - (tagged with "NO_VAL" in parser/schema naming)
  - All use of *oneOf()* and *anyOf()* removed (20.11+)
  - *Infinity*, *null*, and *NaN* will not be dumped as result of a request (20.11+)
  - *Infinity* and *null* must be presented as booleans fields in representative object (23.02+)
    - Example: `{ "set": true, "infinite": true, "number": 0}`
  - All fields present and populated in dumped responses (23.02+)

Ambiguities of JSON

- Slurm uses libjson-c to parse and dump JSON
  - libjson-c is RFC7159 compliant but not ECMA-262 compliant
    - Compliance is fully explained in Ticket#18299 Comment#7
- Known possible JSON incompatibilities with clients:
  - libjson-c may dump “infinity” and “NaN” as values which not all JSON parsers will consider valid JSON. (ticket#20817)
    - Please open a ticket if this is ever encountered.
  - Floating point numbers may only parse as 32 bit floats and lose precision
  - UTF-16, UTF-32, UTF Byte Order Marks are unsupported
  - UTF-8 parsing may treat code points addressed by 15 bits or more as multiple characters
    - Note: Slurm is binary safe for all strings internally but will not “correct” the parsed output from libjson-c as received.

OpenAPI Specification

OpenAPI Specification

- slurmrestd generates the OpenAPI Schema based on runtime arguments.
  - Sites should always specify the plugins via `-s`, `-a` and `-d` (23.11+) they plan to use explicitly via arguments instead of the default of loading all plugins found for production servers.
- Previously, we tried to have a single static specification as a static file (openapi.json).
  - Maintaining the OpenAPI Specification by hand in git ended up causing more problems than it solved as git kept mangling the content and formatting.
  - Schemas are now generated by slurmrestd and the openapi.json is a basic template in the source code (23.02+)
    - Same code that generates the output also generates the schema to keep everything as coherent as possible.
    - The generated OpenAPI schemas at “http://$HOST/openapi/v3” should always be used instead of the openapi.json in the source code.

Improving the OpenAPI Specification

- String Schemas with Enum (23.02+)
  - OpenAPI standard provides the Enum array to allow strings with well defined values to enumerated out.
  - slurmrestd internally tracks most of these well defined strings as flags.
    - Many fields have been converted to flags to make it easier for users to know possible values (23.11)
- Path parameters are now generated (23.11+)
  - All possible parameters should now be in generated OpenAPI specification including enum strings.
- Boolean query parameters in the URI without a value will be considered to be *true*.
  - Example: http://$HOST/slurmdb/v0.0.42/associations?with_usage
  - *openapi-generator-cli* clients will need to pass “true” or “false” in the query objects as the internal schema checker will reject a value of *None*.

OpenAPI Specification Versioning

- Format and layout of schemas are designed to be consistent between all Slurm releases where the versioned plugin is originally tagged in the release.
  - A query to v0.0.40 endpoint in Slurm-23.11 should work the same as a query to v0.0.40 endpoint in Slurm-24.05 and Slurm-24.11.
  - Schemas changes during patchset releases are only done to correct breaking issues, such as ones causing most *openapi-generator-cli* clients to crash.
  - Schemas between different data_parser versions are not guaranteed to be compatible and in some cases may be entirely different. Make sure to test clients when porting between versions.
- OpenAPI Specifications are tagged with the data_parser plugin version and have same version stability.

Major release changes

Changes in Slurm-24.05

- Removal of v0.0.38 endpoints.
- Added v0.0.41 endpoints.
- Partial support for gracefully handling soft memory limits (ticket#19899,18406)
- Add easily overridable environment variable SLURMRESTD_LISTEN in systemd unit slurmrestd.service (ticket#18693)
- Populating “deprecated” fields in OpenAPI schema (ticket#17916)
- Change OpenAPI schema to reduce “$ref” entries with `+prefer_refs` flags to reverse change (ticket#20378)
- Add `slurmrestd --generate-openapi-spec` arg to allow generating OpenAPI schema without running daemon or slurm.conf being present (ticket#19303)
- Support running slurmrestd without slurmdbd configured/online (ticket#19899)
- Add comment descriptions to all fields in >=v0.0.41 in OpenAPI schema (ticket#16961)
- Size buffering per kernel hints to reduce memory usage (ticket#19641)

Changes in Slurm-24.11

- Removal of v0.0.39 endpoints. (ticket#18484)
- Added v0.0.42 endpoints. (ticket#18484)
- Removal of all deprecated fields in v0.0.42 endpoints. (ticket#19938)
- Add `GET slurm/v0.0.42/nodes` endpoint (ticket#19745)
- Error with Authentication Failure [401] instead of Internal Server Error [500] (ticket#18516)
- Added support for 'slurmrestd -d latest' arg (ticket#20615)
- Added 'DataParserParameters' to slurm.conf (ticket#21121)
- Switch to `+prefer_refs` flag as default with `+minimize_refs` flag to allow reverse of change (ticket#20378)
- New formatting and machine friendly for `scontrol ping –json` and 'GET /slurm/v0.0.42/ping` (ticket#20324)
- New 'sacctmgr ping –json` and 'GET /slurmdb/v0.0.42/ping` endpoint (Issue#17)
- Latency improvements (ticket#20114)

Planned changes in Slurm-25.05

- Removal of v0.0.40 endpoints. (Issue#50141)
- Add v0.0.43 endpoints. (Issue#50141)
- Improved test units with better coverage of API calls (ticket#20464,21341)

Any other changes will be announced in Tim Wickberg's *24.05, 24.11 and Beyond* presentation at the Slurm BOF at 12:15pm - 1:15pm EST in B203.

Questions?

SCHEDMD

The Slurm Company