

# REST API



Nathan Rini  
SchedMD

SLUG 2019

Copyright 2019 SchedMD  
[www.schedmd.com](http://www.schedmd.com)

# Contents



- What is REST API
- What is OpenAPI
- What is the Slurm REST API
- Security
- Authentication with REST API
- IPv4 & IPv6
- Examples

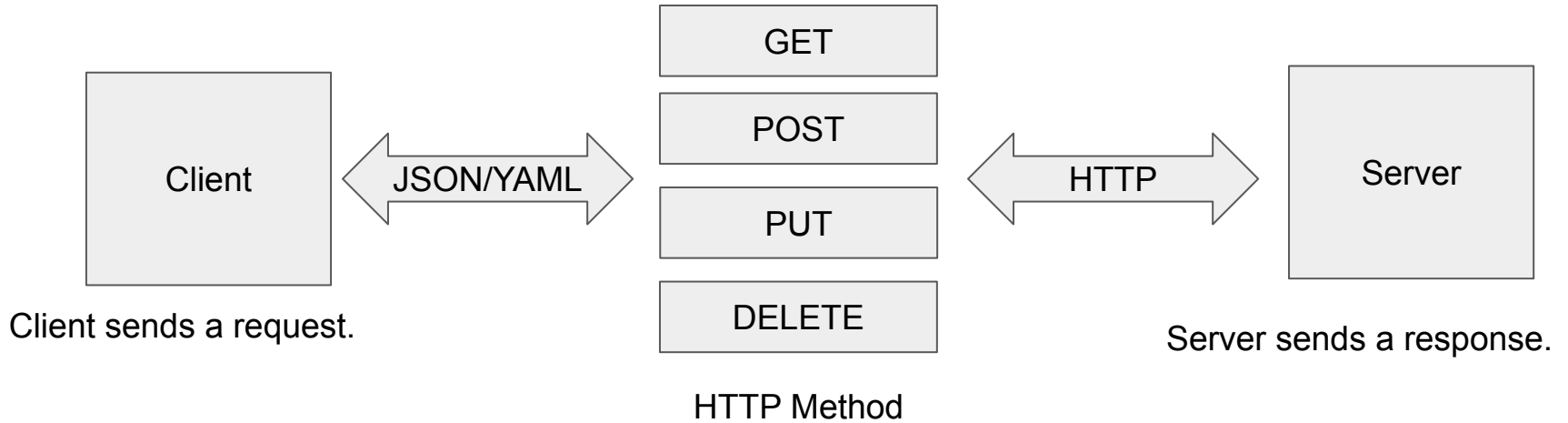
# What is REST API



”REST is acronym for REpresentational State Transfer. It is architectural style for distributed hypermedia systems and was first presented by Roy Fielding in 2000 in his famous dissertation.” --<https://restfulapi.net/>

- Guiding Principles of REST
- Client–server
- Stateless
- Cacheable
- Uniform interface
- Layered system
- Code on demand (optional)

# What is REST API



# What is OpenAPI (aka Swagger)



OpenAPI Specification allows you to describe your entire REST API, including:

- Available endpoints (/users) and operations on each endpoint (GET /users, POST /users)
- Operation parameters Input and output for each operation
- Authentication methods
- Contact information, license, terms of use and other information.
- API specifications can be written in YAML or JSON. The format is easy to learn and readable to both humans and machines.

Copyright 2019 SchedMD

[www.schedmd.com](http://www.schedmd.com)

<https://swagger.io/docs/specification/about/>

# What is OpenAPI

- **Relatively** simple specification for what your program will receive and return.
- Free (and some less free) tools that will work with the specification.
- Pretty documentation automatically generated:

## Slurm Rest API 0.9 OAS3

API to access and control Slurm.

[Terms of service](#)

[GNU General Public License](#)

### default



**GET** /slurm/v1/diag/ get diagnostics

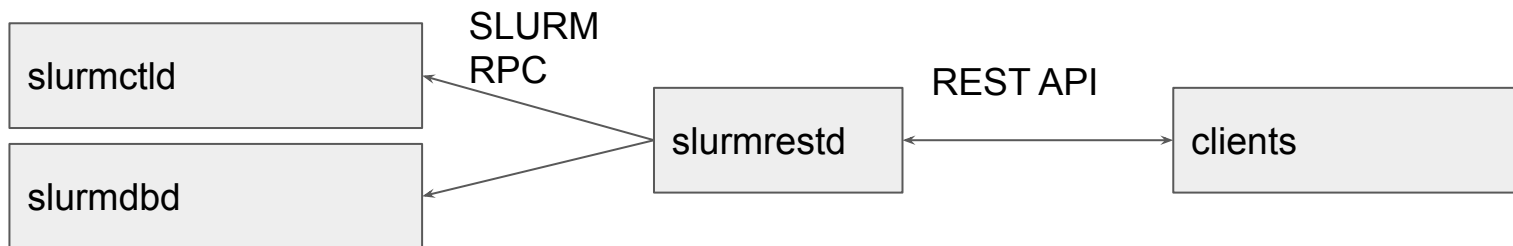
**GET** /slurm/v1/ping/ ping test

**GET** /slurm/v1/jobs/ get list of jobs

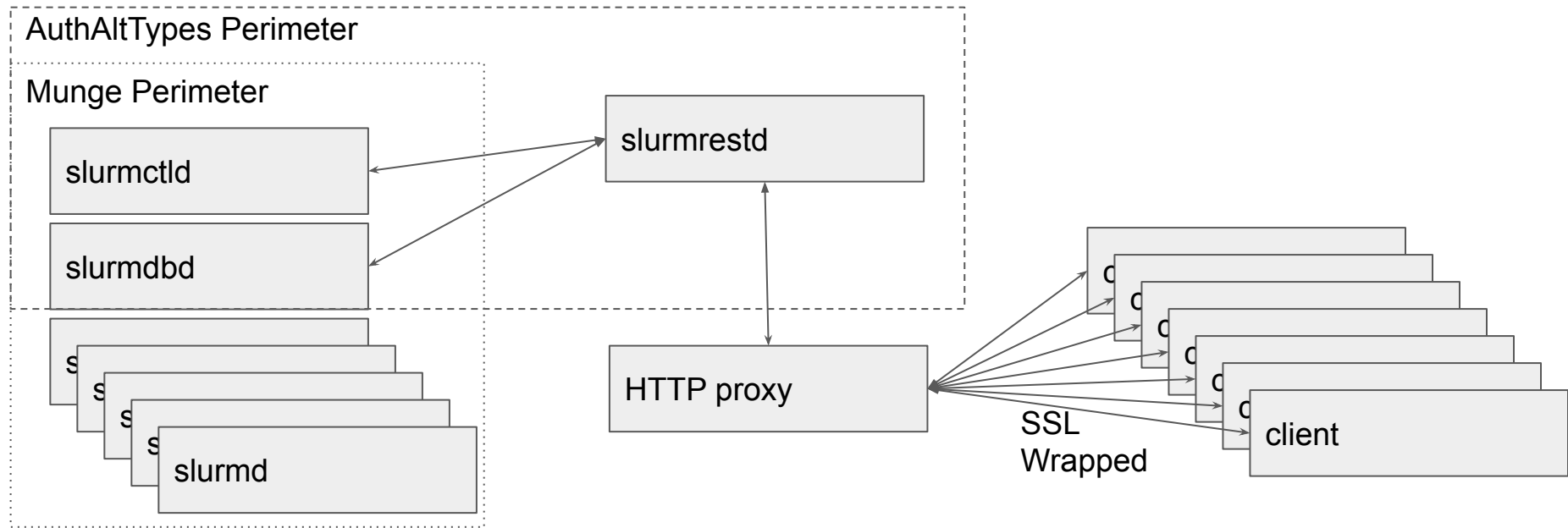
**GET** /slurm/v1/job/{job\_id} get job info

# What is the Slurm REST API

A tool that runs inside of the Slurm perimeter that will translate JSON/YAML requests into Slurm RPC requests.

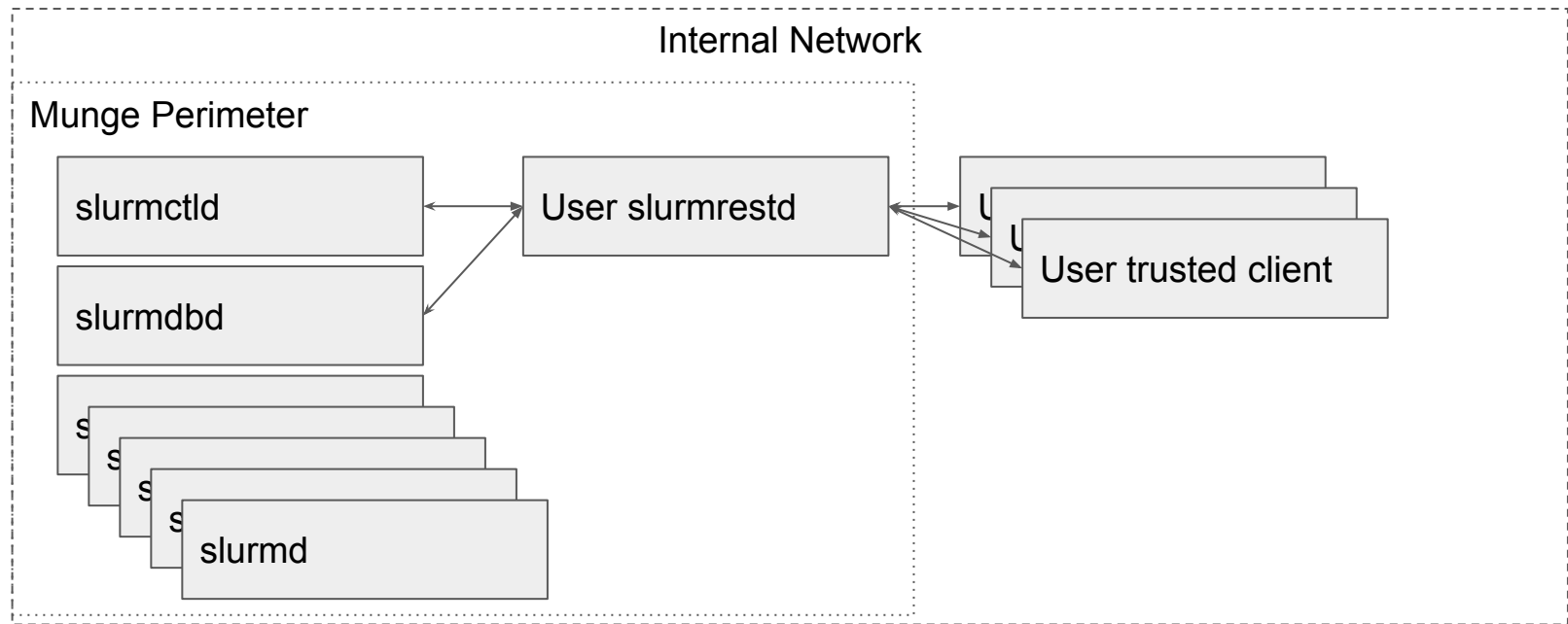


# Slurm REST API Architecture (External REST)



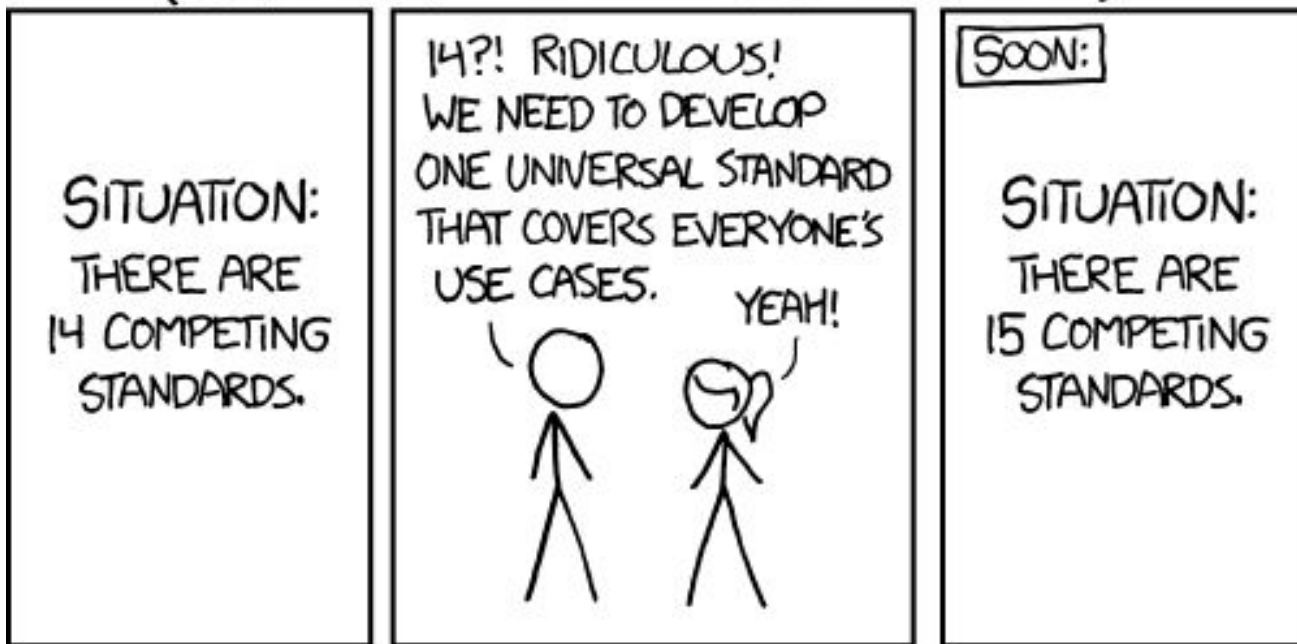


# Slurm REST API Architecture (User Listen Mode)



# REST Standards

HOW STANDARDS PROLIFERATE:  
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)



Slurm's REST API is designed to be as compliant as possible with the relevant REST and IETF standards.

<https://xkcd.com/927/>

# slurmrestd - REST API daemon

- **Daemon Mode**
  - Listen on given set of IP/PORTs for user connections.
- **Inet Mode**
  - Single user connected via STDIN and STDOUT.
    - Started per connection by xinetd, inetd, or systemd socket activation.
  - Script send commands via PIPE directly.
- **Any user can run the daemon.**
- **Execution User:**
  - unprivileged - All commands run as execution user
  - privileged user - All commands can run as requested users

# Design



1. slurmrestd acts as a framework for requests.
  - a. Modular addition of new REST commands via registering URL paths.
2. Built in data type to translate JSON and YAML.
3. Built in support for generating OpenAPI specification.
4. Conformant HTTP server to handle interfacing with clients and proxies.

# Inet Mode Example (unprivileged user)

- `$ echo -e "GET /slurm/v1/diag HTTP/1.1\r\nAccept: */*\r\n" | slurmrestd`
- `slurmrestd: operations_router: /slurm/v1/diag for pipe:[722494]`
- `HTTP/1.1 200 OK`
- `Content-Length: 973`
- 
- `{`
- `"parts_packedg": 1,`
- `"req_timeg": 1567712456,`
- `... JSON continues ...`

# Listen Mode Example (listen on 3 ports)

- `$ slurmrestd -k test -vvv localhost:9997 [::1]:7272 10.10.0.1:38484`
- `slurmrestd: debug: Reading slurm.conf file: /etc/slurm.conf`
- `slurmrestd: debug: Interactive mode activated (TTY detected on STDIN)`
- `slurmrestd: debug: listen: localhost:9997 fd: 3`
- `slurmrestd: debug: _socket_thread_listen: thread for fd: 3`
- `slurmrestd: debug: listen: ip6-localhost:7272 fd: 4`
- `slurmrestd: debug: _socket_thread_listen: thread for fd: 4`
- `slurmrestd: debug: listen: spheron:38484 fd: 5`
- `slurmrestd: debug: server listen mode activated`
- `slurmrestd: debug: _socket_thread_listen: thread for fd: 5`

# Inet Mode Example (unprivileged user with AltAuth)

- `$ echo -e "GET http://localhost/slurm/v1/diag HTTP/1.1\r\nAccept: */*\r\n" | slurmrestd -f etc/slurm.token.conf`
- `slurmrestd: operations_router: /slurm/v1/diag for pipe:[1052487]`
- `HTTP/1.1 200 OK`
- `Content-Length: 973`
- `{`
- `"parts_packedg": 1,`
- `"req_timeg": 1568051342,`
- `"req_time_startg": 1568050812,`
- `"server_thread_count": 3,`
- `... JSON continues ...`

# YAML Mode Example

- `$ echo -e "GET http://localhost/slurm/v1/diag HTTP/1.1\r\nAccept: text/yaml\r\n" | slurmrestd`
  - `slurmrestd: error: _on_url: URL Schema currently not supported for pipe:[782247]`
  - `slurmrestd: error: _on_url: URL host currently not supported for pipe:[782247]`
  - `slurmrestd: operations_router: /slurm/v1/diag for pipe:[782247]`
  - `HTTP/1.1 200 OK`
  - `Content-Length: 1210`
  - `%YAML 1.1`
  - `---`
  - `!!str parts_packedg: !!int 1`
- .... YAML Continues ....



# Security



- Authentication
  - All remote clients must be authenticated via HTTP headers:
    - X-SLURM-API-KEY or X-SLURM-USER-TOKEN
    - X-SLURM-USER-NAME
- Authorization / Mutation
  - Can be offloaded to authenticating proxy.
  - Requests are parsed by `cli_filter`.
- Denial
  - JSON/YAML requests will not even be parsed without authentication and clients will be rejected with 403 errors.

# Security - User & Request Isolation

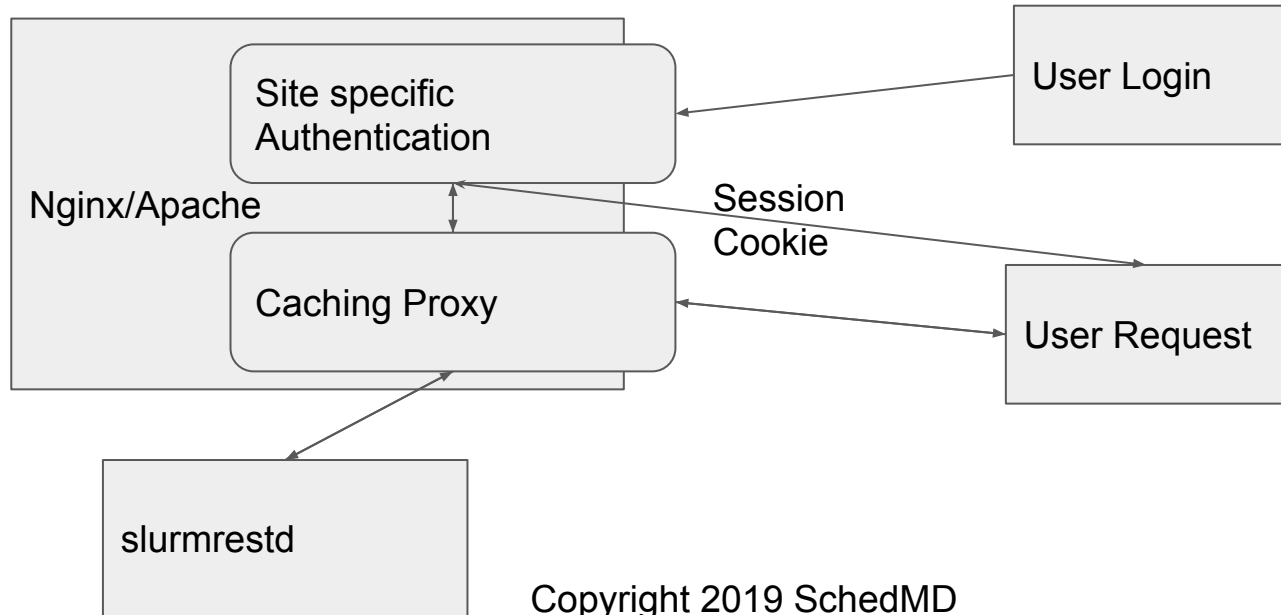
- User per POSIX thread
  - Utilizes new AuthAltTypes framework.
  - Allows slurmrestd to exist outside of Munge.
- Authentication Types
  - Local
    - Only allowed for piping directly into slurmrestd
  - Authenticating Proxy
    - Proxy is given a key that will allow proxy to authenticate as any user
  - User Token
    - Each user will have a token to authenticate.

# Authentication with REST API



1. Authenticating proxy to verify users and user requests prior to being sent to REST API.
2. Authentication can be:
  - a. Site specific
    - i. Too many authentication schemes to support them all.
    - ii. Filter/modify user requests
  - b. Scalable
    - i. External/Slow authentication systems can be:
      1. Cached or Offloaded
      2. Parallelized

# Authenticating HTTP Proxy Example



# Simple Authenticating HTTP Proxy Demo

- `$ cd src/contribs/auth_proxy_example/simple`
- `$ make #starts docker auth-proxy:latest`
- `$ slurmrestd localhost:9999 & #start background daemon`
- `$ curl 'http://localhost:8080/slurm/v1/ping'`
  - `<html><head><title>401 Authorization Required</title></head>`
  - `<body><center><h1>401 Authorization Required</h1></center>`
  - `<hr><center>nginx</center></body></html>`
- `$ curl --user test:test -c /tmp/cookiejar 'http://localhost:8080/auth/'`  
`'http://localhost:8080/slurm/v1/ping'`
  - `<p>Hello test.</p><p>You entered test as your password.</p>`
  - `{ "errors": [], "ping": { "spheron": { }, "ping": "UP", "status": 0, "mode": "primary" } }`

# IPv4 & IPv6 Support



- slurmrestd supports IPv6 and IPv4 clients (or proxies).
- slurmrestd can act as REST gateway into IPv4 cluster to external IPv6 clients.
  
- slurmrestd must be able to talk to slurmctld/slurmdbd over IPv4.

# IPv6 Mode Example

- `$ slurmrestd -vv [::1]:9999 &`
- `slurmrestd: slurmrestd: debug: listen: ip6-localhost:9999 fd: 3`
- `$ curl -s 'http://[::1]:9999/slurm/v1/ping'`
- `slurmrestd: debug: parse_http: Accepted HTTP connection from ::%2698417920:49016`
- `slurmrestd: debug: connection ::%2698417920:49016 url path: /slurm/v1/ping query: (null)`
- `slurmrestd: operations_router: /slurm/v1/ping for ::%2698417920:49016`
- `{ "errors": [ ], "ping": { "spheron": { }, "ping": "UP", "status": 0, "mode": "primary" } }`
- `slurmrestd: debug: parse_http: Closed HTTP connection from ::%2698417920:49016`

# Example Queries



- Job submission (sbatch)
- Heterogeneous Job ("HetJob") submission
- Array Job Submission
- Cancelling job (scancel)
- Viewing jobs (scontrol show job)
- slurmctld diagnostics (sdiag)

HTTP/JSON/YAML have been compacted or even trimmed to fit on the slides. Examples are to show what is possible. Please see generated openapi reference for exact implementation documentation.



# Job submission request example (sbatch)

- `$ cat demo`
- `POST /slurm/v1/job/submit HTTP/1.1`
- `Accept: text/yaml`
- `Content-Type: application/json`
- `Content-Length: 348`
- 
- `{"job": {"account": "test", "ntasks": 20, "name": "test18.1", "nodes": [2, 4], "current_working_directory": "/tmp/", "user_id": 1000, "group_id": 1000, "environment": {"PATH": "/bin:/usr/bin:/usr/local/bin/", "LD_LIBRARY_PATH": "/lib/./lib64:/usr/local/lib"}}, "script": "#!/bin/bash\nnecho it works"}`

# Job submission example (sbatch)

- `$ slurmrestd 2>/dev/null < demo`
- `HTTP/1.1 200 OK`
- `Content-Length: 131`
- `%YAML 1.1`
- `---`
- `!!str errors: !!seq []`
- `!!str job_id: !!int 115`
- `!!str step_id: !!str BATCH`
- `!!str job_submit_user_msg: !!null null`
- `...`
- `$ squeue -j 115`
- | JOBID            | PARTITION | USER    | ST   | TIME         | NODES |
|------------------|-----------|---------|------|--------------|-------|
| NODELIST(REASON) |           |         |      |              |       |
| 115              | debug     | nate PD | 0:00 | 4 (Priority) |       |

# Job submission request example (sbatch HetJob)

- `$ cat demo`
- `POST /slurm/v1/job/submit HTTP/1.1`
- `Accept: text/yaml`
- `Content-Type: application/json`
- `Content-Length: 654`
- `{"job": [{"account": "test", "ntasks": 20, "name": "test18.1", "nodes": [2, 4], "current_working_directory": "/tmp/", "user_id": 1000, "group_id": 1000, "environment": {"PATH": "/bin:/usr/bin:/usr/local/bin/", "LD_LIBRARY_PATH": "/lib:/lib64:/usr/local/lib"}}, {"account": "test", "ntasks": 2, "name": "test18.3", "nodes": [2, 4], "current_working_directory": "/tmp/", "user_id": 1000, "group_id": 1000, "environment": {"PATH": "/bin:/usr/bin:/usr/local/bin/", "LD_LIBRARY_PATH": "/lib:/lib64:/usr/local/lib"}} ], "script": "#!/bin/bash\nsrunch echo it works" }`

# Job submission example (sbatch HetJob)

- `$ slurmrestd < demo`
- `HTTP/1.1 200 OK`
- `Content-Length: 131`
- 
- `%YAML 1.1`
- `---`
- `!!str errors: !!seq []`
- `!!str job_id: !!int 118`
- `!!str step_id: !!str BATCH`
- `!!str job_submit_user_msg: !!null null`
- `...`

# Job submission example (sbatch HetJob)

- `$ squeue -j 118`

- |   | JOBID            | PARTITION | USER | ST | TIME | NODES    |
|---|------------------|-----------|------|----|------|----------|
|   | NODELIST(REASON) |           |      |    |      |          |
| • | 118+0            | debug     | nate | PD | 0:00 | 4 (None) |
| • | 118+1            | debug     | nate | PD | 0:00 | 4 (None) |

# Job submission request example (sbatch HetJob)

- `$ cat demo`
- `POST /slurm/v1/job/submit HTTP/1.1`
- `Accept: text/yaml`
- `Content-Type: application/json`
- `Content-Length: 374`
- `{"job":{"account":"test","array":"1-1000","ntasks":20,"name":"test18.1","nodes": [2,4],"current_working_directory":"/tmp/","user_id":1000,"group_id":1000,"environment":{"PATH":"/bin:/usr/bin:/usr/local/bin/","LD_LIBRARY_PATH":"/lib:/lib64:/usr/local/lib"}}, "script":"#!/bin/bash\nsrun echo it works"}`

# Job submission example (sbatch array)

- `$ slurmrestd < demo`
- `HTTP/1.1 200 OK`
- `Content-Length: 132`
- 
- `%YAML 1.1`
- `---`
- `!!str errors: !!seq []`
- `!!str job_id: !!int 1202`
- `!!str step_id: !!str BATCH`
- `!!str job_submit_user_msg: !!null null`
- `...`

# Job submission example (sbatch array)

- `$ squeue -j 1202`
- | JOBID            | PARTITION | USER | ST | TIME | NODES         |
|------------------|-----------|------|----|------|---------------|
| NODELIST(REASON) |           |      |    |      |               |
| 1202_[528-1000]  | debug     | nate | PD | 0:00 | 4 (Resources) |
| 1202_527         | debug     | nate | R  | 0:00 | 4 host[5-8]   |
| 1202_526         | debug     | nate | R  | 0:00 | 4 host[1-4]   |
- 
-



# Cancelling job example (sbatch array)

- `$ cat demo`
- `DELETE /slurm/v1/job/2202 HTTP/1.1`
- `Accept: text/yaml`
- `$ slurmrestd < demo`
- `HTTP/1.1 200 OK`
- `Content-Length: 41`
- `%YAML 1.1`
- `---`
- `!!str errors: !!seq []`
- `...`
- `$ scontrol show job 2202|grep JobState`
- `JobState=CANCELLED Reason=None Dependency=(null)`

# View job example (sbatch array)

- `$ echo -e 'GET /slurm/v1/job/2203 HTTP/1.1\r\nACCEPT: text/json\r\n\r\n' | slurmrestd`
- `{"account":"test","accrue_time":1568158776,"admin_comment":"","array_job_id":0,"array_task_id":{"type":"array"},"array_max_tasks":0,"array_task_str":"","assoc_id":4,"batch_features":"","batch_flag":true,"batch_host":"host1","bitflags":["JOB_WAS_RUNNING"],"boards_per_node":0,"burst_buffer":"","burst_buffer_state":"","cluster":"linux","cluster_features":"","command":"","comment":"","contiguous":false,"core_spec":{"type":"core"},"thread_spec":{"type":"thread"},"cores_per_socket":{"type":"core"},"billable_tres":12,"cpus_per_task":{"type":"cpu"},"cpu_freq_min":{"type":"cpu"},"cpu_freq_max":{"type":"cpu"},"cpu_freq_gov":{"type":"cpu"},"cpus_per_tres":"","deadline":0,"delay_boot":0,"dependency":"","derived_ec":0,"eligible_time":1568158776,"end_time":1599694776,"exc_nodes":"","execution_node_by_index":[],"exit_code":0,"features":"","fed_origin_str":"","fed_siblings_active":0,"fed_siblings_active_str":"","fed_siblings_viable":0,"fed_siblings_viable_str":"","gres_detail":[],"group_id":1000,"job_id":2203,"job_resources":{"type":"job"},"job_st`

# Diagnostics query example (sbatch array)

- `echo -e 'GET /slurm/v1/diag HTTP/1.1\r\nACCEPT: text/json\r\n\r\n' | slurmrestd`
- `{"parts_packedg":1,"req_timeg":1568158683,"req_time_startg":1568153903,"server_thread_count":3,"agent_queue_size":0,"agent_count":0,"dbd_agent_queue_size":0,"gettimeofday_latency":18,"schedule_cycle_max":12041,"schedule_cycle_last":18,"schedule_cycle_sum":1008716,"schedule_cycle_counter":1138,"schedule_cycle_depth":5109,"schedule_queue_len":0,"jobs_submitted":2130,"jobs_started":2083,"jobs_completed":2082,"jobs_canceled":54,"jobs_failed":0,"jobs_pending":0,"jobs_running":0,"job_states_ts":1568158674,"bf_backfilled_jobs":1,"bf_last_backfilled_jobs":1,"bf_backfilled_pack_jobs":0,"bf_cycle_counter":130,"bf_cycle_sum":37282,"bf_cycle_last":65,"bf_cycle_max":690,"bf_last_depth":1,"bf_last_depth_try":1,"bf_depth_sum":3356,"bf_depth_try_sum":260,"bf_queue_len":0,"bf_queue_len_sum":0,"bf_when_last_cycle":1568158222,"bf_active":0}`



# QUESTIONS?

Copyright 2019 SchedMD  
[www.schedmd.com](http://www.schedmd.com)

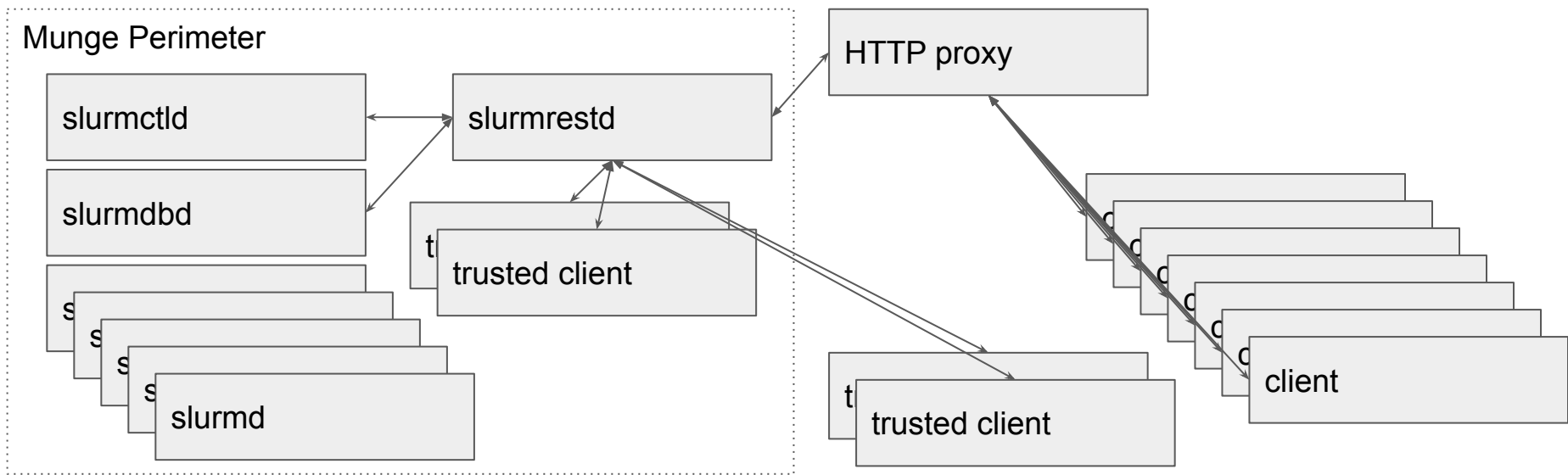


# Backup Slides

Copyright 2019 SchedMD  
[www.schedmd.com](http://www.schedmd.com)



# Slurm REST API Architecture (Internal REST)



# Simple Authenticating HTTP Proxy (authorization)

- `$ cat auth.php`

- `<?php`
- `if (!isset($_SERVER['PHP_AUTH_USER']) || !isset($_SERVER['PHP_AUTH_PW']) ||`
- `!($_SERVER['PHP_AUTH_USER'] == "test" && $_SERVER['PHP_AUTH_PW'] ==`
- `"test")) {`
- `header("WWW-Authenticate: Basic realm="test");`
- `header('HTTP/1.0 401 Unauthorized');`
- `echo 'Please try<br>User: test Password: test';`
- `exit;`
- `} else {`
- `session_start();`
- `$_SESSION['user_id'] = 1000;`
- `echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}</p>";`
- `echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";`
- `}`
- `?>`



# Simple Authenticating HTTP Proxy (validation)

- `$ cat validate.php`

- `<?php`
- `error_reporting(E_ALL);`
- `session_start();`
- 
- `if (session_status() == PHP_SESSION_NONE || !isset($_SESSION['user_id'])) {`
- `header("HTTP/1.1 401 Unauthorized");`
- `} else {`
- `header("X-SLURM-USER-ID: ".$_SESSION['user_id']);`
- `}`
- 
- `?>`

# Simple Authenticating HTTP Proxy (nginx config)

- `$ cat nginx.conf # TRIMMED`
  - `location ~ ^/slurm/(.*)$ {`
  - `auth_request /validate/;`
  - `auth_request_set $userid $upstream_http_x_slurm_user_id;`
  - `proxy_pass http://localhost:9999;`
  - `proxy_set_header X-SLURM-USER-ID $userid;`
  - `proxy_set_header X-SLURM-API-KEY test;`
  - `proxy_set_header Host $host;`
  - `proxy_set_header X-Real-IP $remote_addr;`
  - `proxy_redirect ~^http://[^\:]+\:d+(/.)$ http://localhost:9999/$1;`
  - `}`
  - `location = /validate/ { internal; auth_request off; index index.php; }`
  - `location = /auth/ { auth_request off; index index.php; }`