Clients for Slurm's REST API

(aka slurmrestd clients)

Nathan Rini

SC'25

slurm | SCHEDMD

Questions?

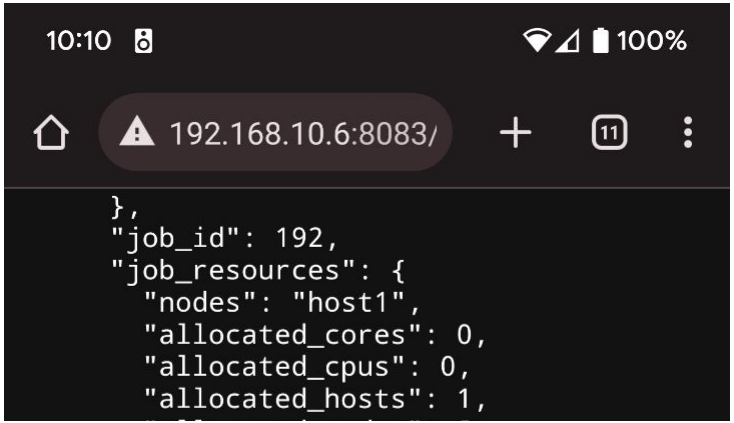
- Please feel free to interrupt at any time with questions.
  - I will ask you wait if the question is answered later in the presentation.
- Comments and constructive complaints are always welcome
  - I may ask to defer discussion to finish the presentation on time or if question is not applicable to 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.

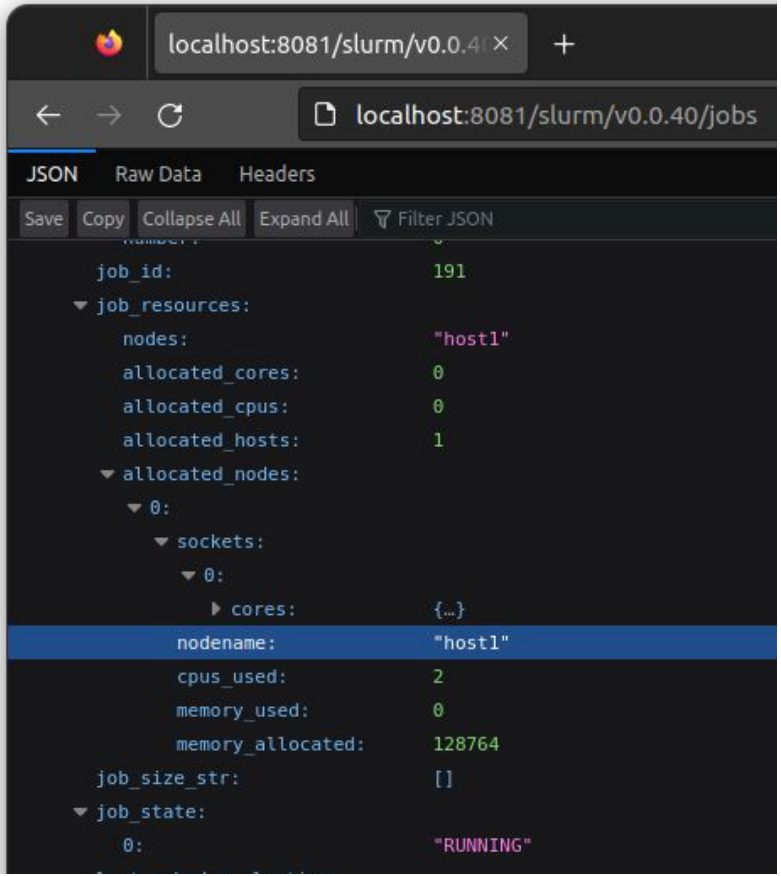
Intro to slurmrestd

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.

Setup

Slurm's JWT Authentication

- Setup procedure: https://slurm.schedmd.com/jwt.html
- Supported algorithms:
  - HS256 (shared secret)
  - RS256 (public key)
- 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
  - Popular tools: Keycloak and Amazon Cognito and Microsoft Entra ID
  - Python example: https://slurm.schedmd.com/jwt.html#compatibility
  - Users and groups on cluster must match token's username exactly
- Implementation is limited and can not be used to communicate with slurmd daemons.
- SLURM_JWT environment variable should not be set in user CLI environments.

Compiling slurmrestd with JWT support

- 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
checking for jwt.h... yes
```

- 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
```

Configure JWT authentication

- Slurmrestd and JWT are installed automatically if all prerequisites are detected
  - Optionally require slurmrestd and JWT

```
./configure --enable-slurmrestd --with-jwt
make -j
make install
```

  - Verify in config.log

```
configure:29013: checking whether to compile slurmrestd
configure:29044: result: yes
```

```
configure:22801: checking for jwt.h
configure:22801: /usr/local/gcc/15.1.0//bin/gcc -c -DNUMA_VERSION1_COMPATIBILITY -g -O2
-fno-omit-frame-pointer  conftest.c >&5
configure:22801: $? = 0
configure:22801: result: yes
configure:22810: checking for jwt_add_header in -ljwt
configure:22839: /usr/local/gcc/15.1.0//bin/gcc -o conftest -DNUMA_VERSION1_COMPATIBILITY -g -O2
-fno-omit-frame-pointer   conftest.c -ljwt  -lm -lresolv  >&5
configure:22839: $? = 0
configure:22851: result: yes
```

Start slurmrestd to service clients

- Start Slurm cluster
- Start slurmrestd daemon manually:

```
env SLURM_JWT=daemon slurmrestd -a jwt :6820
```

- Start slurmrestd via systemd:

```
systemctl enable slurmrestd
systemctl start slurmrestd
systemctl start slurmrestd
```

Example use cases via Python Client

Preparation: Compile and install generated OpenAPI client

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

```
$ slurmrestd -f /dev/null --generate-openapi-spec -d v0.0.44 >
openapi.json
$ env OPENAPI_GENERATOR_VERSION=7.17.0 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 .
```

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:6820/",
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:

```
>>> pprint([a for a in dir(slurmdb) if re.match(r'slurmdb', a)])
['slurmdb_v0044_delete_account',
 'slurmdb_v0044_delete_account_with_http_info',
 'slurmdb_v0044_delete_account_without_preload_content',
 'slurmdb_v0044_delete_association',
 'slurmdb_v0044_delete_association_with_http_info',
 'slurmdb_v0044_delete_association_without_preload_content',
 'slurmdb_v0044_delete_associations',
...(truncated)...

>>> pprint([a for a in dir(slurm) if re.match(r'slurm', a)])
['slurm_v0044_delete_job',
 'slurm_v0044_delete_job_with_http_info',
 'slurm_v0044_delete_job_without_preload_content',
 'slurm_v0044_delete_jobs',
 'slurm_v0044_delete_jobs_with_http_info',
 'slurm_v0044_delete_jobs_without_preload_content',
...(truncated)...
```

Query jobs information

- Get job_id of all jobs known to slurmctld:

```
resp = slurm.slurm_v0044_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_v0044_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_v0044_get_jobs()
c=0
for job in resp.jobs:
    if 'RUNNING' in job.job_state:
        c += job.tasks.number
pprint(c)
```

Example Job submission

- Submit example job:

```
from openapi_client.models.v0044_job_desc_msg import
V0044JobDescMsg as JobDesc
from openapi_client.models.v0044_job_submit_req import
V0044JobSubmitReq as JobReq

job = JobReq(job=JobDesc(script='#!/bin/bash\nsrun uptime',
environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/'],current_work
ing_directory='/tmp/'))
print(slurm.slurm_v0044_post_job_submit(job).job_id)
```

Example Array Job submission

- Submit example job:

```
from openapi_client.models.v0044_job_desc_msg import
V0044JobDescMsg as JobDesc
from openapi_client.models.v0044_job_submit_req import
V0044JobSubmitReq as JobReq

job = JobReq(job=JobDesc(script='#!/bin/bash\nsrun
uptime',array='100',environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/
sbin/'],current_working_directory='/tmp/'))
print(slurm.slurm_v0044_post_job_submit(job).job_id)
```

Example Heterogeneous (HetJob) submission

- Submit example job:

```
from openapi_client.models.v0044_job_desc_msg import
V0044JobDescMsg as JobDesc
from openapi_client.models.v0044_job_submit_req import
V0044JobSubmitReq as JobReq

job = JobReq(jobs=[JobDesc(script='#!/bin/bash\nsrun
uptime',environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/'],curr
ent_working_directory='/tmp/'),
    JobDesc(environment=['PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/'],
    current_working_directory='/tmp/'),])
print(slurm.slurm_v0044_post_job_submit(job).job_id)
```

Control Jobs

- Cancel a job:

```
slurm.slurm_v0044_delete_job(job_id='3694')
```

- Signal a job with SIGINT:

```
slurm.slurm_v0044_delete_job(job_id='3694', signal='SIGINT')
```

Control Jobs

- Change job name:

```
from openapi_client.models.v0044_job_desc_msg import V0044JobDescMsg as
JobDesc

job = JobDesc(name='updated test job')
print(slurm.slurm_v0044_post_job(job_id='3697', v0044_job_desc_msg=job))
```

- Change job task count:

```
from openapi_client.models.v0044_job_desc_msg import V0044JobDescMsg as
JobDesc

job = JobDesc(tasks=15)
print(slurm.slurm_v0044_post_job(job_id='3697', v0044_job_desc_msg=job))
```

Example use cases via Slinky slurm-client (Golang)

Preparation: Compile and install generated OpenAPI client

- Prerequisite:
  - Install golang
- Compile and install Slinky's slurm-client

```
$ mkdir rest-client
$ go mod init example.com/rest-client
go: creating new go.mod: module example.com/rest-client
$ go get github.com/SlinkyProject/slurm-client
```

Golang client common driver code [1/3]

- Common code required:

```
package main

import (
        "context"
        "fmt"
        api "github.com/SlinkyProject/slurm-client/api/v0044"
        client "github.com/SlinkyProject/slurm-client/pkg/client"
        types "github.com/SlinkyProject/slurm-client/pkg/types"
        "k8s.io/utils/ptr"
        "os/exec"
        "strings"
)
```

Golang client common driver code [1/3]

- Common code required: Gets a JWT from scontrol:

```
func getJWT() string {
        out, _ := exec.Command("scontrol", "token",
"lifespan=90").Output()
        token := strings.TrimPrefix(string(out), "SLURM_JWT=")
        token = strings.TrimSuffix(token, "\n")
        // fmt.Printf("JWT: %v\n", token)
        return token
}
```

Golang client common driver code [1/3]

- Common code required: Gets a JWT from scontrol:

```
func main() {
        config := &client.Config{
                Server:    "http://localhost:6820",
                AuthToken: getJWT(),
        }

        slurmClient, _ := client.NewClient(config)

        // Optional for Caching
        ctx := context.Background()
        go slurmClient.Start(ctx)

        // INSERT CODE HERE
}
```

Inspection of generated GoLang client

- Inspect HTTP methods implemented as functions and structs using Delve in Golang:

```
$ go install github.com/go-delve/delve/cmd/dlv@latest
$ ~/go/bin/dlv debug .
...(navigate to break to inspect object)...
(dlv) p slurmClient
github.com/SlinkyProject/slurm-client/pkg/client.Client(*github.com/SlinkyProj
ect/slurm-client/pkg/client.client) *{
        mu: sync.RWMutex {
                w: (*sync.Mutex)(0xc00028a000),
                writerSem: 0,
                readerSem: 0,
                readerCount: (*"sync/atomic.Int32")(0xc00028a010),
                readerWait: (*"sync/atomic.Int32")(0xc00028a014),},
        informers:
map[github.com/SlinkyProject/slurm-client/pkg/object.ObjectType]github.com/Sli
nkyProject/slurm-client/pkg/client.InformerCache [],
        cancel: context.WithCancel.func1 {
                c *context.cancelCtx = *(*context.cancelCtx)(0xc000288050)
        },
...(truncated)...
```

Query jobs information

- Get job_id of all jobs known to slurmctld:

```
jobs := &types.V0044JobInfoList{}

if err := slurmClient.List(ctx, jobs); err != nil {
        panic(err)
}

for _, job := range jobs.Items {
        fmt.Printf("JobID: %v\n", *job.StepId.JobId.Number)
}
```

- Example output:

```
$ go run .
JobID: 153
JobID: 154
```

Query jobs information

- Get total number of tasks of all running jobs:

```
var tasks int32 = 0
jobs := &types.V0044JobInfoList{}
if err := slurmClient.List(ctx, jobs); err != nil {
        panic(err)
}
for _, job := range jobs.Items {
        tasks += *job.Tasks.Number
}
fmt.Printf("Tasks: %v\n", tasks)
```

- Example output:

```
$ go run .
Tasks: 22
```

Example Job submission

- Submit example job:

```
        jobInfo := &types.V0044JobInfo{}
        req := api.V0044JobSubmitReq{
                Job: &api.V0044JobDescMsg{
                        CurrentWorkingDirectory: ptr.To("/tmp"),
                        Environment: &api.V0044StringArray{
"PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/",
                        },
                        Script: ptr.To("#!/bin/bash\nsrun uptime"),
                },
        }

        if err := slurmClient.Create(ctx, jobInfo, req); err != nil {
                panic(err)
        }

        fmt.Printf("Submitted JobID: %v\n", *jobInfo.JobId)
```

Example Array Job submission

- Submit example job:

```
        jobInfo := &types.V0044JobInfo{}
        req := api.V0044JobSubmitReq{
                Job: &api.V0044JobDescMsg{
                        CurrentWorkingDirectory: ptr.To("/tmp"),
                        Environment: &api.V0044StringArray{
"PATH=/bin/:/sbin/:/usr/bin/:/usr/sbin/",
                        },
                        Script: ptr.To("#!/bin/bash\nsrun uptime"),
                        Array:  ptr.To("1-100"),
                },
        }

        if err := slurmClient.Create(ctx, jobInfo, req); err != nil {
                panic(err)
        }

        fmt.Printf("Submitted JobID: %v\n", *jobInfo.JobId)
```

Example Heterogeneous (HetJob) submission

- Submit example job:

```
jobInfo := &types.V0044JobInfo{}
 req := api.V0044JobSubmitReq{
        Jobs: &[]api.V0044JobDescMsg{api.V0044JobDescMsg{
                CurrentWorkingDirectory: ptr.To("/tmp"),
                Environment: &api.V0044StringArray{
                        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin",
                },
                Script: ptr.To("#!/usr/bin/env bash\nsleep 30"),
        }, api.V0044JobDescMsg{
                CurrentWorkingDirectory: ptr.To("/tmp"),
                Environment: &api.V0044StringArray{
                        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin",
                },
                Script: ptr.To("#!/usr/bin/env bash\nsleep 30"),
        }},
 }

 if err := slurmClient.Create(ctx, jobInfo, req); err != nil {
        panic(err)
 }

 fmt.Printf("Submitted JobID: %v\n", *jobInfo.JobId)
```

Control Jobs

- Cancel a job:

```
jobInfo := &types.V0044JobInfo{
        V0044JobInfo: api.V0044JobInfo{
                JobId: &[]int32{288}[0],
        },
  }
  if err := slurmClient.Delete(ctx, jobInfo); err != nil {
        panic(err)
  }

  fmt.Printf("Cancelled JobID: %v\n", *jobInfo.JobId)
```

Control Jobs

- Change job comment:

```
jobInfo := &types.V0044JobInfo{
        V0044JobInfo: api.V0044JobInfo{
                JobId: &[]int32{289}[0],
        },
}
req := api.V0044JobDescMsg{
        Comment: ptr.To("updated comment"),
}
if err := slurmClient.Update(ctx, jobInfo, req); err != nil {
        panic(err)
}

fmt.Printf("Updated JobID: %v\n", *jobInfo.JobId)
```

Control Jobs

- Change job name:

```
jobInfo := &types.V0044JobInfo{
        V0044JobInfo: api.V0044JobInfo{
                JobId: &[]int32{289}[0],
        },
}
req := api.V0044JobDescMsg{
        Name: ptr.To("new job name here"),
}
if err := slurmClient.Update(ctx, jobInfo, req); err != nil {
        panic(err)
}

fmt.Printf("Updated JobID: %v\n", *jobInfo.JobId)
```

Control Jobs

- Change job task count:

```
jobInfo := &types.V0044JobInfo{
        V0044JobInfo: api.V0044JobInfo{
                JobId: &[]int32{306}[0],
        },
}
req := api.V0044JobDescMsg{
        Tasks: &[]int32{16}[0],
}
if err := slurmClient.Update(ctx, jobInfo, req); err != nil {
        panic(err)
}

fmt.Printf("Updated JobID: %v\n", *jobInfo.JobId)
```

Questions?


SCHEDMD
The Slurm Company