Slinky - Slurm 🤝 Kubernetes

Skyler Malinowski
Alan Mutschelknaus
Marlow Warnicke

SchedMD

- Developers and maintainers of the open source Slurm Workload Manager
- Offer commercial support for Slurm, alongside training and development
- Plan to release Slinky as open-source, targeting November (KubeCon)
- Will offer commercial support for Slinky, alongside training and development

What is Slinky?

A collection of projects and initiatives to enable Slurm on Kubernetes:

- Slurm-operator
- Kubernetes Deployment
  - Helm Chart
  - Container Images
- Any underlying resource drivers
- Resource Scheduler Plugin

HPC vs. Cloud Native Historical assumptions

HPC

- Underlying software is mutable
    - Users assume fine-grained control
- Users are often systems experts that understand infrastructure
    - Have a tolerance for complexity
- Access to compute handled by a resource manager or scheduling system
- Users own the node entirely during computation
- Assumption of node homogeneity

Cloud Native

- Underlying software is immutable
- Users are not systems experts, do not think in terms of parallel
    - Limited tolerance for complexity
- Users share nodes
    - Can introduce jitter
    - Can blow through bandwidth
- Assumption of heterogeneous nodes
- Not a ton of attention given to network topology

Overview - Easily Transferable Slurm Scheduling Features

- **GPU reservations and scheduling/allocation of jobs:** Advanced and timed reservations of resources for a particular group and/or job
- **Priority scheduling:** Determine job execution order based on priorities and weights such as age
- **Fair share:** Resources are distributed equitably among users based on historical usage.
- **Quality of Service (QoS):** set of policies, such as limits of resources, priorities, and preemption and backfilling.

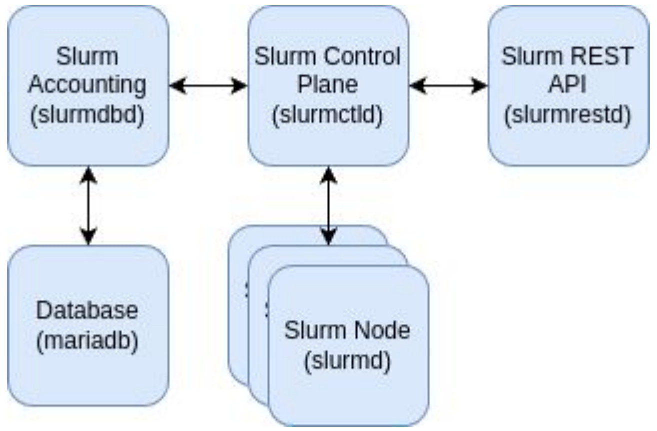
Overview - Easily Transferable Slurm Scheduling Features

- **Job accounting**: Information for every job and job step executed
- **Job dependencies**: Allow users to specify relationships between jobs, from start, succeed, fail, or a particular state.
- **Granular job allocation** (e.g. CPU, Memory, GPU)
- **Network topology**: Properties taken into account such as node proximity, bandwidth, and latency.
- **Workflows with partitioning**: Divide cluster resource into sections for job management

Architecture - Slurm Operator

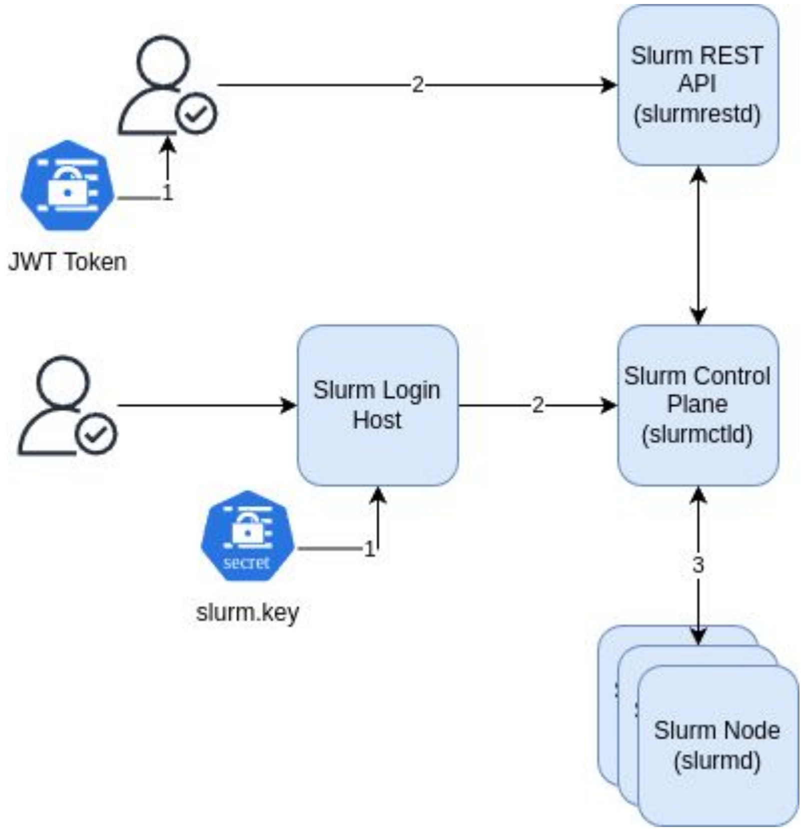
Slurm Architecture - Daemons

- Slurmctld
  - Slurm Control-Plane
  - Slurm API
    - Slurm Daemon
    - Client Commands
- Slurmd
  - Slurm Compute Node Agent
- Slurmstepd
  - Slurm Job Agent
- Slurmrestd
  - Slurm REST API
- Slurmdbd
  - Slurm Database Agent
- Sackd
  - Slurm Auth/Cred Agent

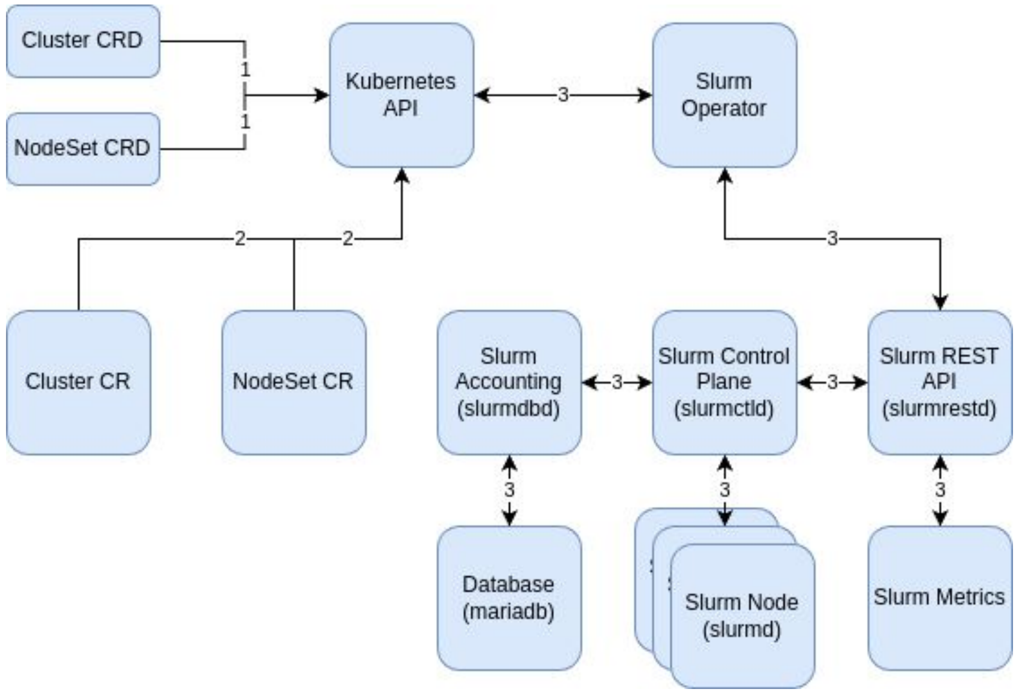

Slurm Accounting (slurmdbd)
Slurm Control Plane (slurmctld)
Slurm REST API (slurmrestd)
Database (mariadb)
Slurm Node (slurmd)

Jobs

1. User can be authenticated with Slurm
2. User submits a Slurm job.
3. Job runs until completion.



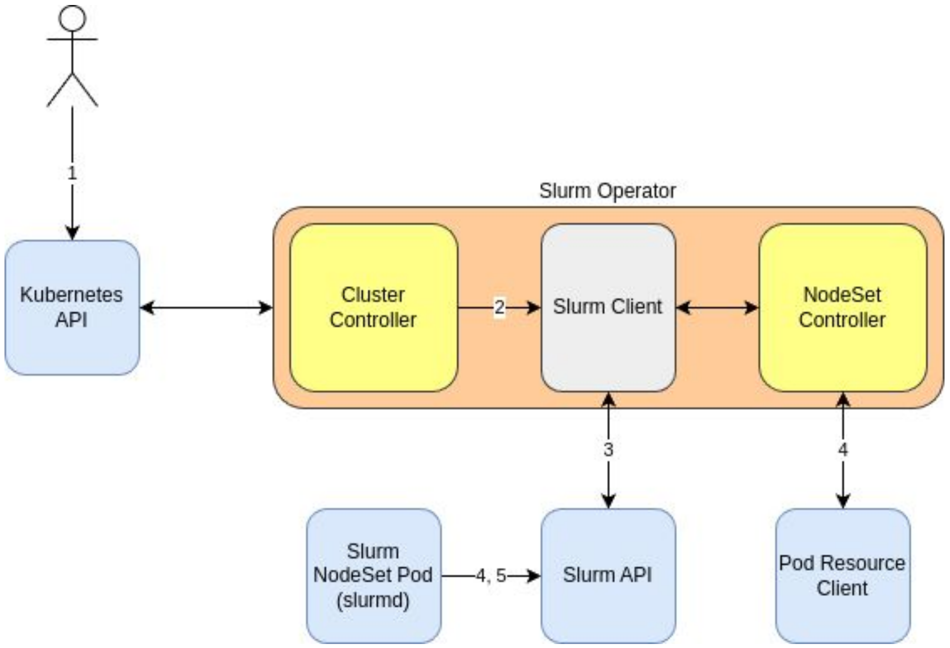
Big Picture



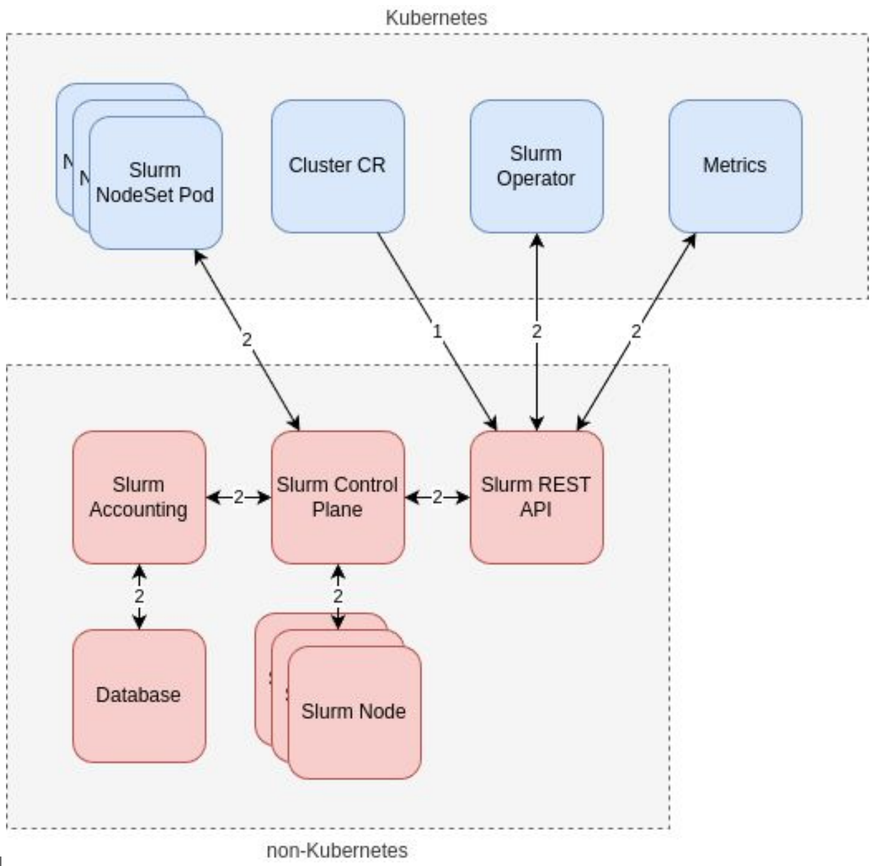
1. Install Slinky Custom Resource Definitions (CRDs).
2. Add/Delete/Update Slinky Custom Resource (CR).
3. Network Communication.

Slurm-Operator

1. User installs Slinky CRs.
2. Cluster Controller creates Slurm Client from Cluster CR.
3. Slurm Client starts informer to poll Slurm resources.
4. NodeSet Controller creates NodeSet Pods from NodeSet CR
   a. The `slurmd` registers to `slurmctld` on startup.
5. NodeSet Controller terminates NodeSet Pod after fully draining Slurm node
   a. NodeSet Pod deletes itself from Slurm on preStop.



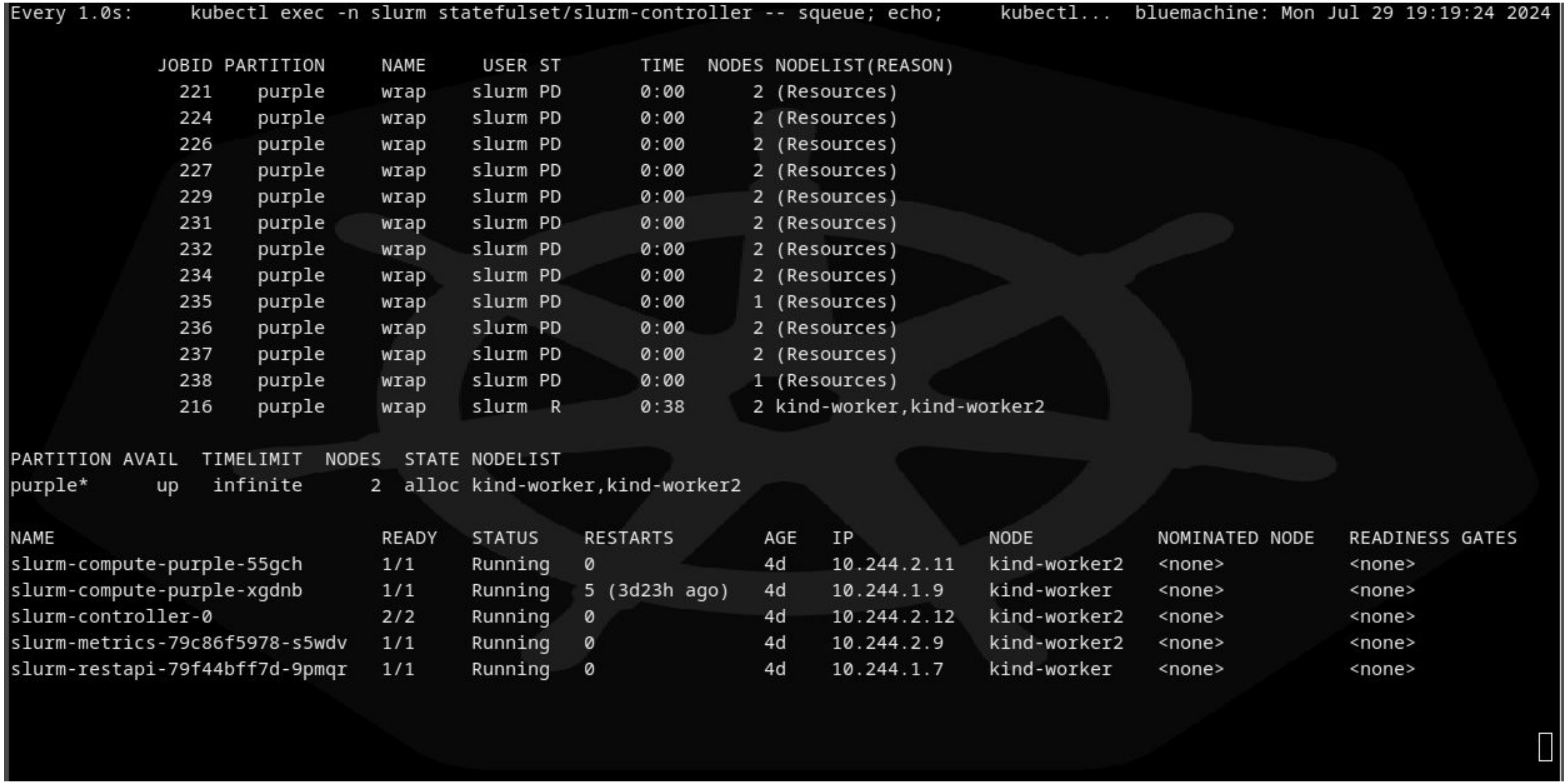
Slurm: Kubernetes + non-Kubernetes



1. References a resource.
2. Network Communication.

- Slurm components (e.g. slurmctld, slurmd, slurmrestd, slurmdbd) can reside anywhere.
  - Kubernetes
  - Bare-metal
  - Virtual Machine
- Communication is key!

Demo - Slurm Operator

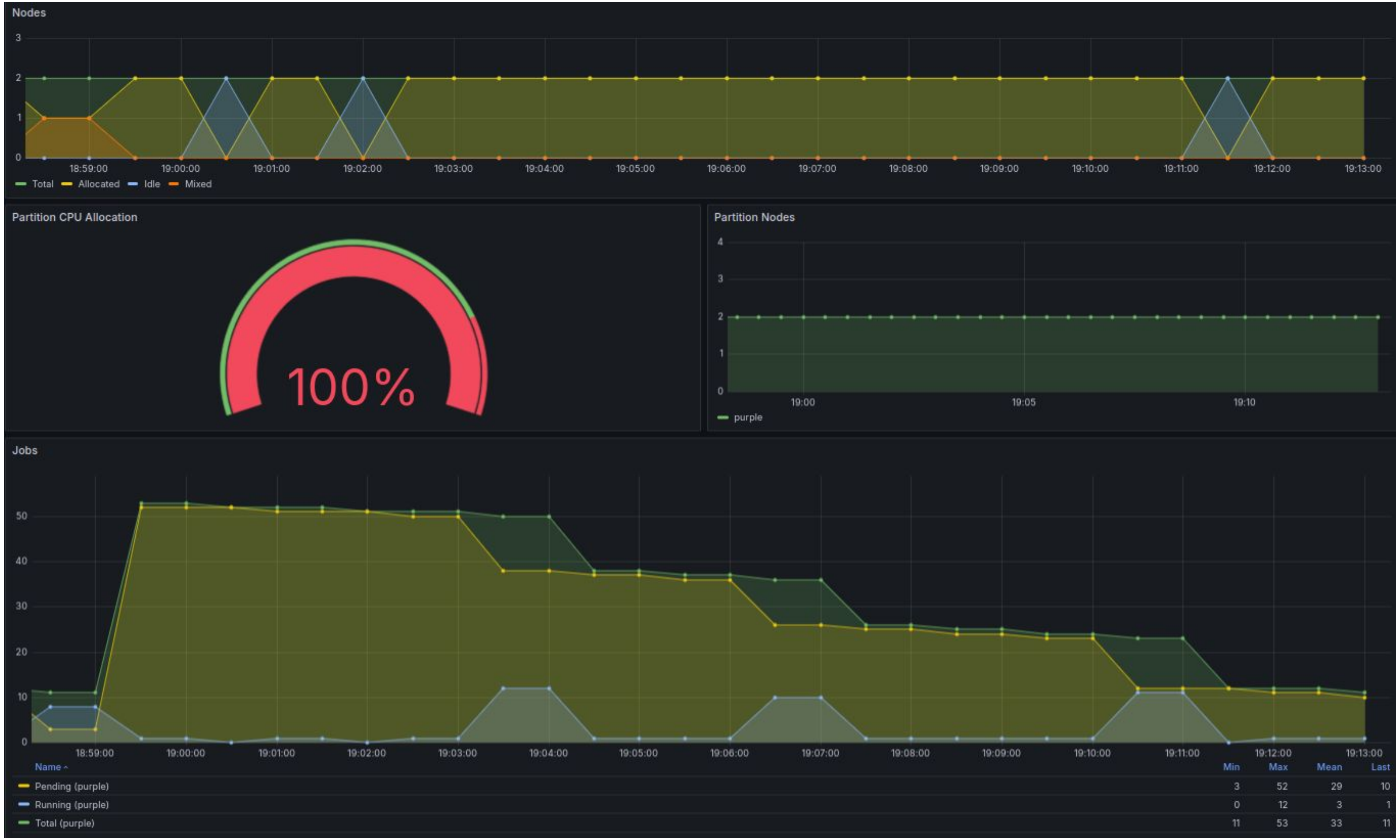


```
Every 1.0s:     kubectl exec -n slurm statefulset/slurm-controller -- squeue; echo;     kubectl...  bluemachine: Mon Jul 29 19:19:24 2024

             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               221    purple     wrap    slurm PD       0:00      2 (Resources)
               224    purple     wrap    slurm PD       0:00      2 (Resources)
               226    purple     wrap    slurm PD       0:00      2 (Resources)
               227    purple     wrap    slurm PD       0:00      2 (Resources)
               229    purple     wrap    slurm PD       0:00      2 (Resources)
               231    purple     wrap    slurm PD       0:00      2 (Resources)
               232    purple     wrap    slurm PD       0:00      2 (Resources)
               234    purple     wrap    slurm PD       0:00      2 (Resources)
               235    purple     wrap    slurm PD       0:00      1 (Resources)
               236    purple     wrap    slurm PD       0:00      2 (Resources)
               237    purple     wrap    slurm PD       0:00      2 (Resources)
               238    purple     wrap    slurm PD       0:00      1 (Resources)
               216    purple     wrap    slurm  R       0:38      2 kind-worker,kind-worker2

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
purple*      up   infinite      2  alloc kind-worker,kind-worker2

NAME                             READY   STATUS    RESTARTS        AGE   IP            NODE           NOMINATED NODE   READINESS GATES
slurm-compute-purple-55gch       1/1     Running   0               4d    10.244.2.11   kind-worker2   <none>           <none>
slurm-compute-purple-xgdnb       1/1     Running   5 (3d23h ago)   4d    10.244.1.9    kind-worker    <none>           <none>
slurm-controller-0               2/2     Running   0               4d    10.244.2.12   kind-worker2   <none>           <none>
slurm-metrics-79c86f5978-s5wdv   1/1     Running   0               4d    10.244.2.9    kind-worker2   <none>           <none>
slurm-restapi-79f44bff7d-9pmqr   1/1     Running   0               4d    10.244.1.7    kind-worker    <none>           <none>
```



Name ^	Min	Max	Mean	Last
Pending (purple)	3	52	29	10
Running (purple)	0	12	3	1
Total (purple)	11	53	33	11

slurm SCHEDMD

Future Work

Future Work

- Slurm finer-grained management of kubelet resource allocations (e.g. CPUs, GPUs, Core pinning)
  - Current Kubernetes cannot mix pinned and unpinned cores, let alone more complex versions of core assignment
  - Increase pluggable infrastructure of Kubernetes - current CPU and memory manager leaves much to be desired. Replace either with DRA or with some other model.
- Network Topology Aware Scheduling in Slurm
  - Using NFD combined with Slurm internals
- Add Slurm scheduling extension to handle resource scheduling for the cluster
  - Map current scheduling concepts not in Slurm, e.g. affinity/anti-affinity
- Slurm scheduler component

SCHEDMD

The Slurm Company