Slurm Bridge: Slurm Scheduling Superpowers in Kubernetes

Tim Wickberg – tim@schedmd.com
Chief Technology Officer, SchedMD

Alan Mutschelknaus - alan@schedmd.com
Senior Cloud Developer, SchedMD

KubeCon | CloudNativeCon
North America 2025

What is Slurm?

- Leading HPC Workload Manager
  - Workload Manager = Scheduler + Resource Manager
    - Roughly equivalent to "Orchestrator"
  - Scheduler:
    - Prioritize and decide which jobs to run on which parts of the system
  - Resource Manager:
    - Track node state and resources
    - Launch jobs
- Manages the majority of the TOP500 supercomputers
  - Also manages most AI/ML training workloads
  - Scales beyond 15,000 nodes in the cluster
- Open-Source
  - GPL-v2+

Who are SchedMD?

- Developers of Slurm – and Slinky
- Spun off from LLNL in 2012 to support Slurm's rapid adoption
  - Founders are Moe and Danny, the "MD" in SchedMD
- SchedMD provides commercial support for Slurm - and Slinky - alongside
  - Training
  - Consultation
  - Custom Development

What is Slinky?

slurm
+
=
slinky

What is Slinky?

- Toolkit of projects to integrate Slurm with Kubernetes
- Open Source
  - Apache-2.0
- Major components:
  - Slurm-operator
  - Slurm-bridge
  - Associated tooling

slinky

Why both?

- Systems faced with increasing demand for batch-style workloads
- AI/ML folks are running Kubernetes for Inference
  - But Slurm for Training workloads
- More traditional HPC systems are being asked to support more flexible workloads
  - But still need resource constraints, efficient queueing, and enough policy control to manage finite system resources
- Running and maintaining both traditional HPC and Cloud Native clusters simultaneously wastes resources

Why both?

- How can we converge the two environments?
- Slinky exists at intersection of the HPC and Cloud Native environments
  - Slurm Operator provides for a traditional Slurm HPC environment within an overarching Kubernetes system
  - Slurm Bridge provides for HPC scheduling semantics for both traditional Slurm batch jobs and emerging cloud-native workloads
    - And gives systems engineers a central place to prioritize both workloads

Additional Capabilities

- Slurm can provide scheduling advantages for pure-Kubernetes environments
  - Efficient multi-node scheduling and resource allocation
  - Planning around future system state - "backfill" - allowing deferred execution of multi-node workloads while not blocking current jobs from scheduling
  - Network topology management – e.g., for NVIDIA DGX systems – ensuring optimal placement for multi-node workloads
    - And ensuring de-fragmentation
    - Managed by the `topology/block` plugin in Slurm

Slurm Bridge

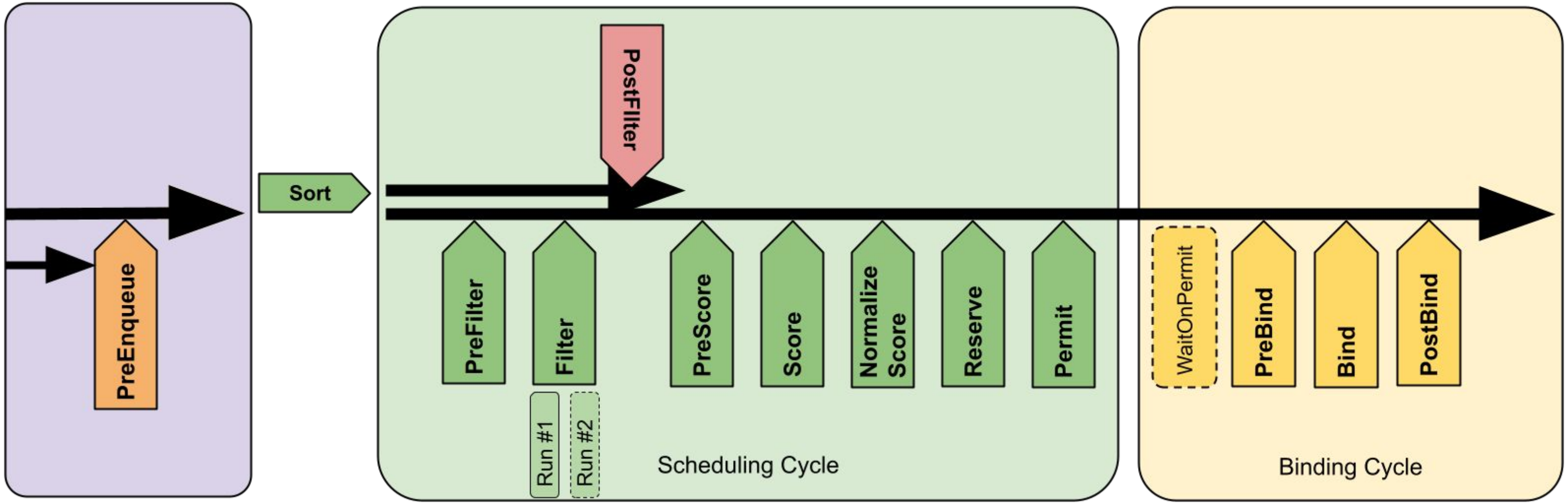
Slurm Bridge

- Slurm as a Kubernetes scheduler using the Scheduling Framework
  - Uses PreEnqueue, PreFilter, Filter, and PostFilter for placement decision
  - Uses PreBind to generate DRA ResourceClaims

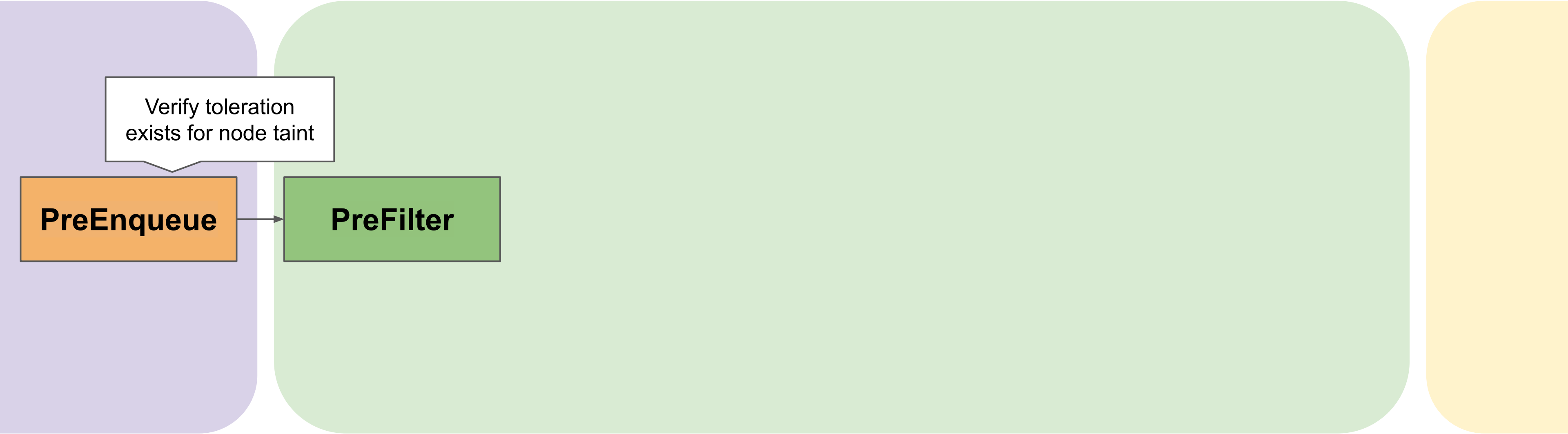
- Translate pod resources into a Slurm placeholder job
  - Placeholder job in Slurm will determine when and where pod(s) run
  - Placeholder job is an “external job” in Slurm
  - Will leverage new Workload resource coming in 1.36

- Scheduled by Slurm, launched by kubelet
  - Slurm schedules to nodes running slurmd or “external nodes” without slurmd

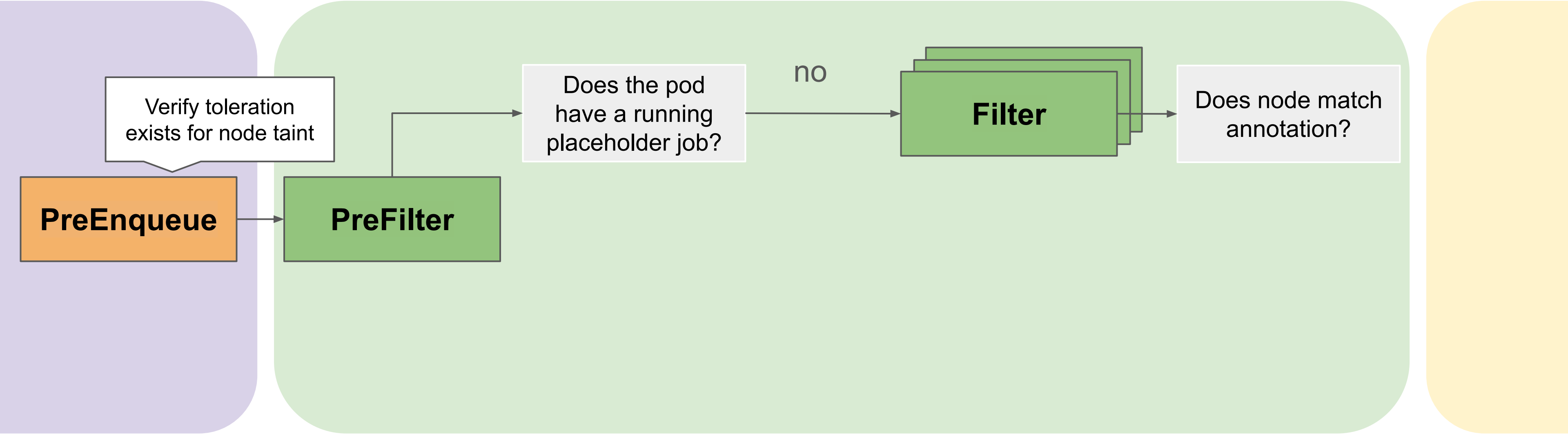
Kubernetes Scheduler Framework - Slurm Bridge Plugins



Kubernetes Scheduler Framework - Slurm Bridge Plugins



Kubernetes Scheduler Framework - Slurm Bridge Plugins



Verify toleration exists for node taint

PreEnqueue

PreFilter

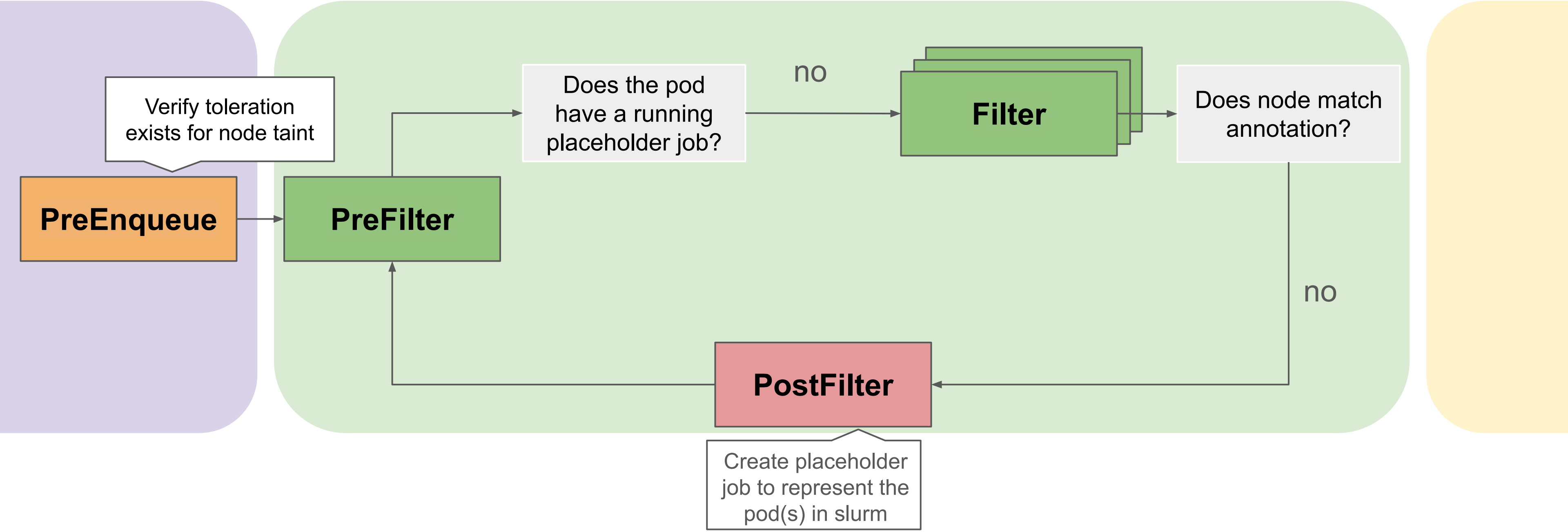
Does the pod have a running placeholder job?

no

Filter

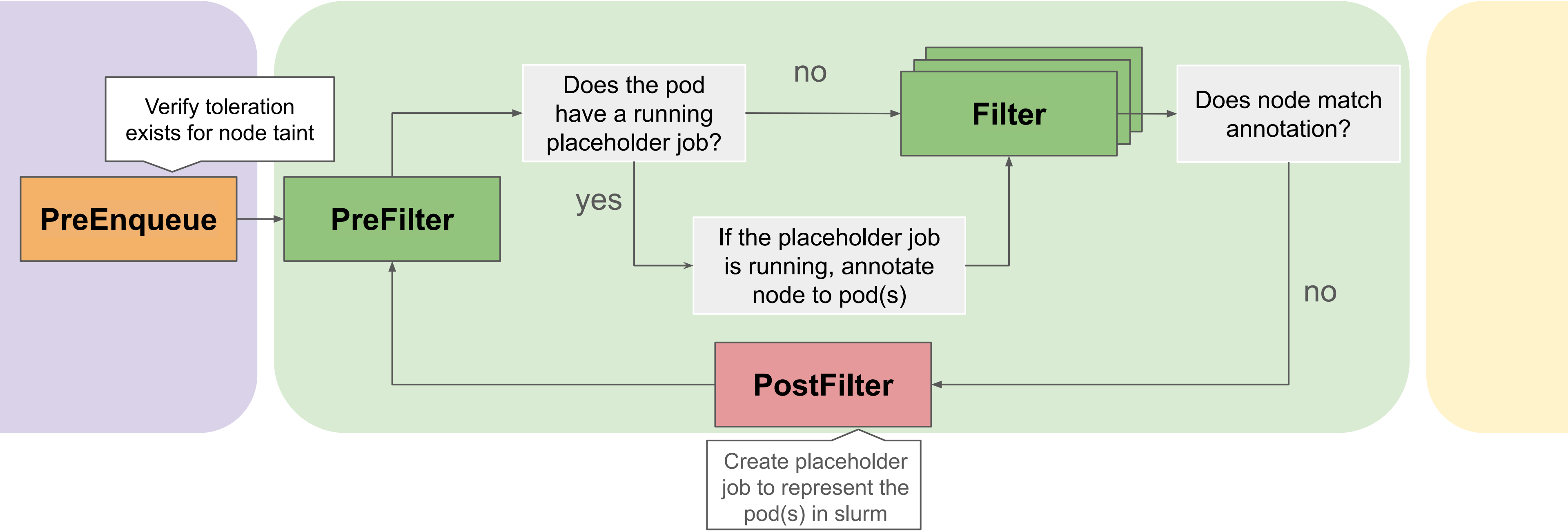
Does node match annotation?

Kubernetes Scheduler Framework - Slurm Bridge Plugins

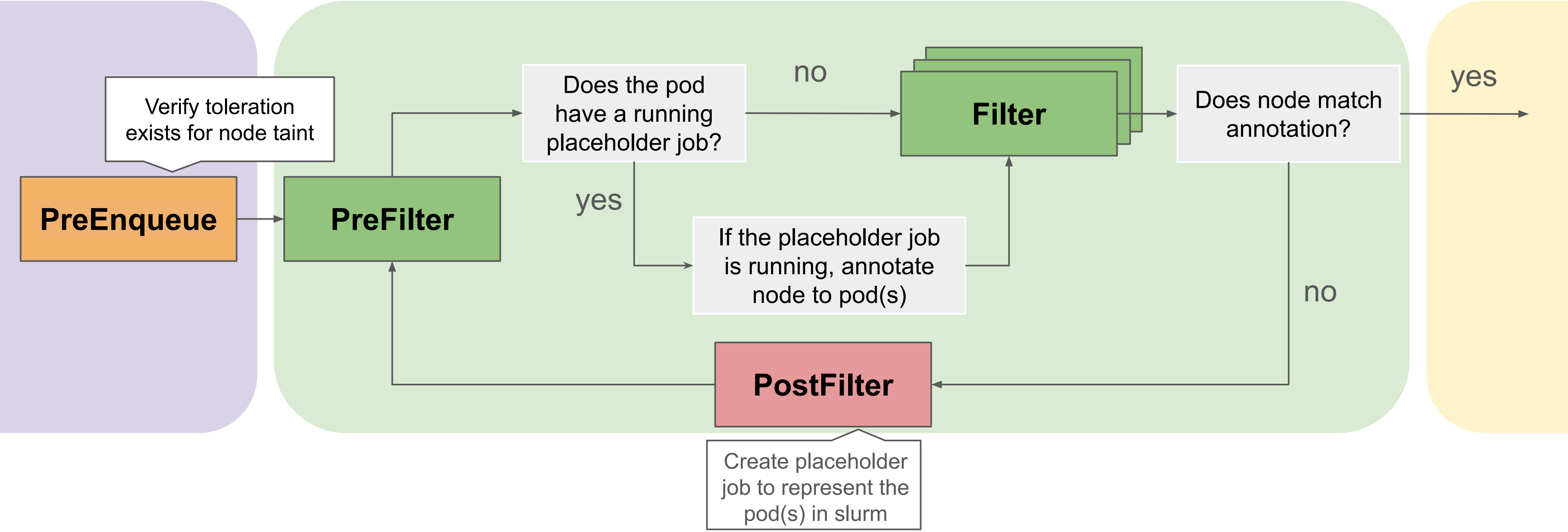


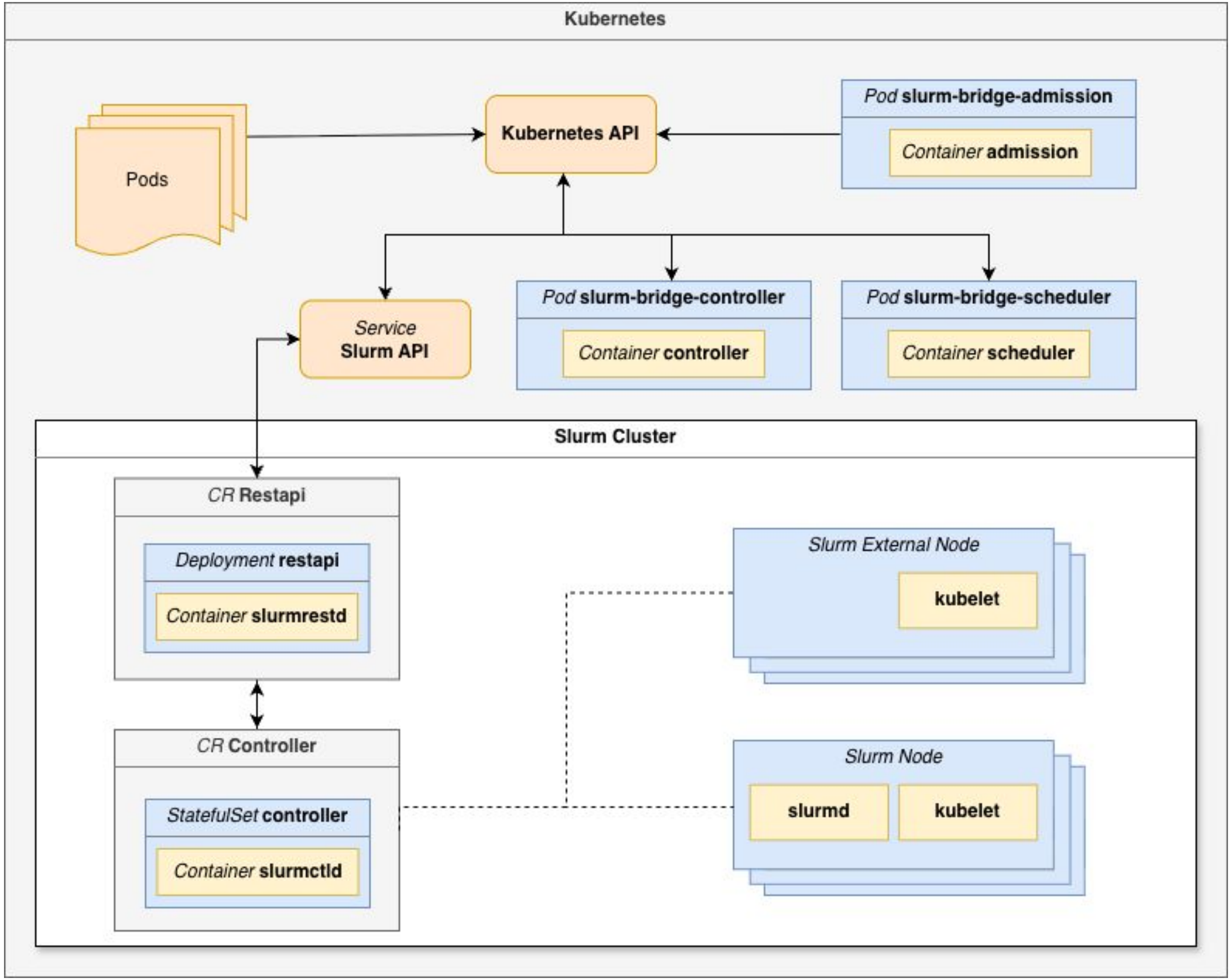
Verify toleration exists for node taint
PreEnqueue
PreFilter
Does the pod have a running placeholder job?
no
Filter
Does node match annotation?
no
PostFilter
Create placeholder job to represent the pod(s) in slurm

Kubernetes Scheduler Framework - Slurm Bridge Plugins


Verify toleration exists for node taint
PreEnqueue
PreFilter
Does the pod have a running placeholder job?
no
yes
Filter
Does node match annotation?
If the placeholder job is running, annotate node to pod(s)
no
PostFilter
Create placeholder job to represent the pod(s) in slurm

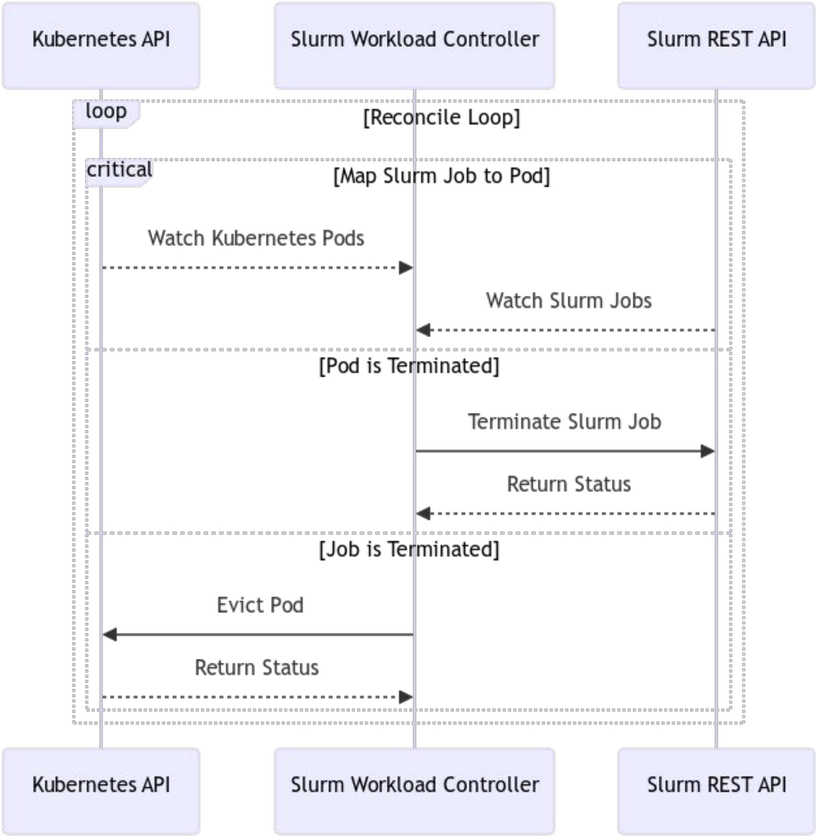
Kubernetes Scheduler Framework - Slurm Bridge Plugins


Verify toleration exists for node taint
PreEnqueue
PreFilter
Does the pod have a running placeholder job?
no
yes
Filter
If the placeholder job is running, annotate node to pod(s)
Does node match annotation?
yes
no
PostFilter
Create placeholder job to represent the pod(s) in slurm


Kubernetes
Pods
Kubernetes API
Pod **slurm-bridge-admission**
Container **admission**
Service **Slurm API**
Pod **slurm-bridge-controller**
Container **controller**
Pod **slurm-bridge-scheduler**
Container **scheduler**
Slurm Cluster
CR **Restapi**
Deployment **restapi**
Container **slurmrestd**
CR **Controller**
StatefulSet **controller**
Container **slurmctld**
Slurm External Node
kubelet
Slurm Node
slurmd
kubelet

Slurm Workload Controller - Sequence

- Workload controller reconciles state between Kubernetes and Slurm control planes
- Slurm is the source-of-truth for Bridged Nodes
- Responsible for cleaning up:
    - Slurm jobs after pods complete/terminate
    - Pods after Slurm job complete/terminate
    - Generated ResourceClaims

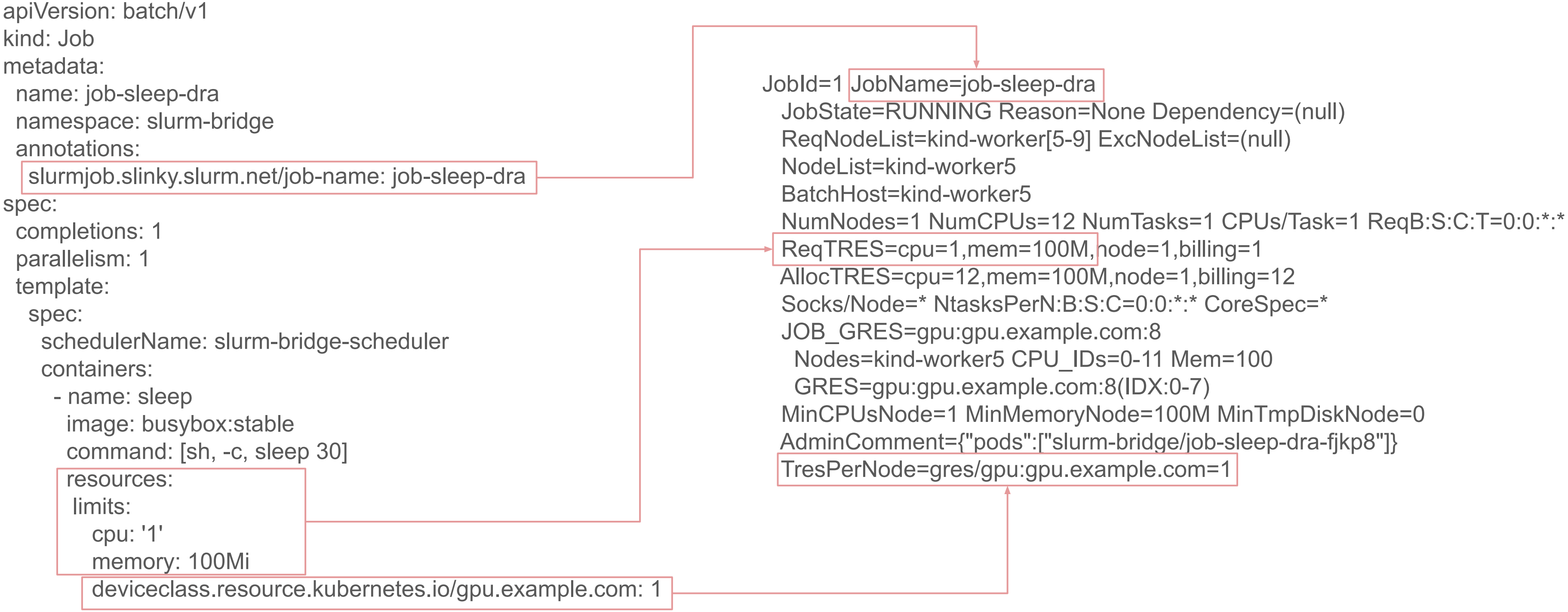


slurm + = slinky | SCHEDMD

Slurm Bridge Features

- Workload Aware, multi-node scheduling
  - Job, JobSet, PodGroup, LeaderWorkerSet
  - Multiple pods may map to a single placeholder job
- Topology aware scheduling
  - Uses Slurm's definition of node topology
- DRA Support
  - Slurm's Generic Resource (GRES) model DRA ResourceClaims
  - Uses DRA feature DRAExtendedResource

Pod Translation



```
apiVersion: batch/v1
kind: Job
metadata:
  name: job-sleep-dra
  namespace: slurm-bridge
  annotations:
    slurmjob.slinky.slurm.net/job-name: job-sleep-dra
spec:
  completions: 1
  parallelism: 1
  template:
    spec:
      schedulerName: slurm-bridge-scheduler
      containers:
        - name: sleep
          image: busybox:stable
          command: [sh, -c, sleep 30]
          resources:
           limits:
              cpu: '1'
              memory: 100Mi
              deviceclass.resource.kubernetes.io/gpu.example.com: 1
```

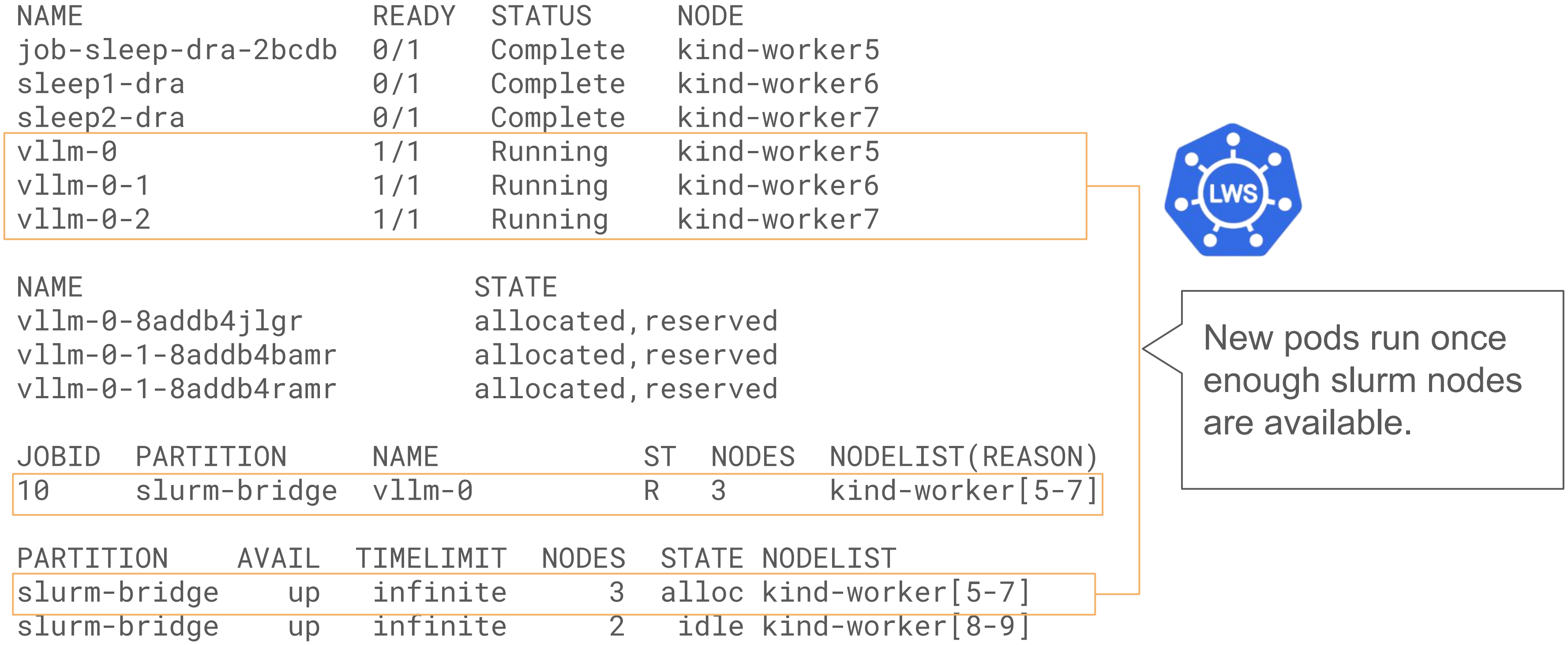
```
JobId=1 JobName=job-sleep-dra
   JobState=RUNNING Reason=None Dependency=(null)
   ReqNodeList=kind-worker[5-9] ExcNodeList=(null)
   NodeList=kind-worker5
   BatchHost=kind-worker5
   NumNodes=1 NumCPUs=12 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   ReqTRES=cpu=1,mem=100M,node=1,billing=1
   AllocTRES=cpu=12,mem=100M,node=1,billing=12
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
   JOB_GRES=gpu:gpu.example.com:8
     Nodes=kind-worker5 CPU_IDs=0-11 Mem=100
     GRES=gpu:gpu.example.com:8(IDX:0-7)
   MinCPUsNode=1 MinMemoryNode=100M MinTmpDiskNode=0
   AdminComment={"pods":["slurm-bridge/job-sleep-dra-fjkp8"]}
   TresPerNode=gres/gpu:gpu.example.com=1
```

```
NAME                 READY  STATUS     NODE
job-sleep-dra-2bcdb  1/1    Running    kind-worker5
sleep1-dra           1/1    Running    kind-worker6
sleep2-dra           1/1    Running    kind-worker7

NAME                       STATE
job-sleep-dra-2bcdb5lmtm   allocated,reserved
sleep1-drapjdll            allocated,reserved
sleep2-dradjtx4            allocated,reserved

JOBID  PARTITION     NAME             ST  NODES  NODELIST(REASON)
9      slurm-bridge  podgroup-sleep   R   2      kind-worker[6-7]
8      slurm-bridge  job-sleep-dra    R   1      kind-worker5

PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
slurm-bridge    up   infinite      3  alloc kind-worker[5-7]
slurm-bridge    up   infinite      2   idle kind-worker[8-9]
```

```
NAME                 READY  STATUS    NODE
job-sleep-dra-2bcdb  1/1    Running   kind-worker5
sleep1-dra           1/1    Running   kind-worker6
sleep2-dra           1/1    Running   kind-worker7
vllm-0               0/1    Pending   <none>
vllm-0-1             0/1    Pending   <none>
vllm-0-2             0/1    Pending   <none>

NAME                          STATE
job-sleep-dra-2bcdb5lmtm   allocated,reserved
sleep1-drapjdll            allocated,reserved
sleep2-dradjtx4            allocated,reserved

JOBID  PARTITION     NAME             ST  NODES   NODELIST(REASON)
9      slurm-bridge  podgroup-sleep   R   2       kind-worker[6-7]
8      slurm-bridge  job-sleep-dra    R   1       kind-worker5
10     slurm-bridge  vllm-0           PD  3       (Resources)

PARTITION     AVAIL  TIMELIMIT  NODES  STATE NODELIST
slurm-bridge     up   infinite      3  alloc kind-worker[5-7]
slurm-bridge     up   infinite      2   idle kind-worker[8-9]
```

LWS

New pods pending on “job 10” and 3 idle nodes…



```
NAME                  READY   STATUS      NODE
job-sleep-dra-2bcdb   0/1     Complete    kind-worker5
sleep1-dra            0/1     Complete    kind-worker6
sleep2-dra            0/1     Complete    kind-worker7
vllm-0                1/1     Running     kind-worker5
vllm-0-1              1/1     Running     kind-worker6
vllm-0-2              1/1     Running     kind-worker7

NAME                         STATE
vllm-0-8addb4jlgr            allocated,reserved
vllm-0-1-8addb4bamr          allocated,reserved
vllm-0-1-8addb4ramr          allocated,reserved

JOBID  PARTITION     NAME               ST  NODES  NODELIST(REASON)
10     slurm-bridge  vllm-0             R   3      kind-worker[5-7]

PARTITION     AVAIL  TIMELIMIT  NODES  STATE NODELIST
slurm-bridge     up   infinite      3  alloc kind-worker[5-7]
slurm-bridge     up   infinite      2   idle kind-worker[8-9]
```

New pods run once enough slurm nodes are available.

Demo

Questions

https://github.com/SlinkyProject

KubeCon | CloudNativeCon

North America 2025