Running Slinky (Slurm) on Azure AKS

GPU and InfiniBand Enabled HPC on Kubernetes

Kris Buytaert
April 2026

Who am I ?

- I used to be a developer
- Then I became an Ops person
- Open Source Consultant @inuits.eu / o11y.eu
- Organiser of #devopsdays, #cfgmgmtcamp, #loadays, ...
- Cofounder of all of the above
- Everything is a Freaking DNS Problem
- DNS : devops needs sushi
- Used to be the OpenMosix release manager
- @krisbuytaert on mastodon.social/bsky/github/

Introduction

- HPC environments are traditionally deployed on bare metal clusters
- Slurm is one of the leading Job Schedulers in the HPC community
- Slinky is a Slurm "distribution" for K8s

This project was to validate :

- Slurm on AKS leveraging EESSI
- GPU / Infiniband supported on Slurm

With a focus on:

- Functional validation
- Not benchmarking or performance tuning.

Git repository :

https://github.com/krisbuytaert/slinky-aks-demo

- Docs
- IaC Code
- Example sbatch jobs
- Example Docker Containers

All code + full examples is in repo.
Only showing snippets in the presentation.

What is Slinky

Slinky integrates Slurm with Kubernetes.

- Slurm Operator
- Slurm Bridge
- Slinky Containers

We focussed on the Slurm Operator, not the Bridge.

Slinky provides:

- Slurm operator
- Slurm containerized components (slurm-worker, slurm-restapi slurm-ctld, slurmdb , slurm-logind etc)
- Slurm nodeset for the slurm-worker instances

Tools

Tools used :

- Azure 'az'
- Opentofu
- Helm
- EESSI
- ArgoCD

Azure 'az'

The Azure CLI tool.

We need to authenticate before running OpenTofu:

```
az login --tenant $TENANTID
```

And we also want to link up our Azure Container Registry to our AKS Cluster

```
az aks update --name slinky-aks  --resource-group slinky \
   --attach-acr $ACRREGISTRY
```

OpenTofu for IAC

The AKS cluster was deployed using **OpenTofu**.

We created a reproducible resource group and AKS cluster with different Nodepools

- System nodepool
- HPC nodepool
- GPU nodepool

Example VM types:

- `Standard_HC44rs` (InfiniBand HPC)
- `Standard_NC4as_T4_v3` (GPU)

EESSI on K8s

EESSI is the European Environment for Scientific Software Installations (EESSI)
The EESSI project proves repositories and pre build tools for HPC purposes that will allow people to deploy more stable and more consistent HPC clusters.

```
helm install cvmfs-csi -f cvmfs-csi/values.yaml \
     oci://registry.cern.ch/kubernetes/charts/cvmfs-csi
```

Now a filesystem on /cvmfs/software.eessi.io/ is available with the EESSI software available for usage.

Installing Slinky

Slinky, the Slinky operator (and a number of prerequisites) are installed using **Helm**.

```
helm install slurm-operator-crds \
     oci://ghcr.io/slinkyproject/charts/slurm-operator-crds
helm install slurm-operator \
     oci://ghcr.io/slinkyproject/charts/slurm-operator  \
     --namespace=slinky --create-namespace
helm install -f slurm/values.yaml  \
     slurm oci://ghcr.io/slinkyproject/charts/slurm \
     -n slurm
```

This deploys the Slurm control components.

Verifying Slurm Resources

```
$kubectl  get pods -n slurm
NAME                                  READY   STATUS    RESTARTS   AGE
slurm-controller-0                    3/3     Running   0          16m
slurm-login-slinky-564cbcd67f-qmbt7   1/1     Running   0          16m
slurm-restapi-75dd5868fb-dlhjz        1/1     Running   0          16m
slurm-worker-slinky-0                 2/2     Running   0          16m
slurm-worker-slinky-1                 2/2     Running   0          16m
slurm-worker-slonky-0                 2/2     Running   0          16m
software-eessi-io-pod                 1/1     Running   0          19m
$ kubectl  get nodeset -n slurm
NAME                 REPLICAS   UPDATED   READY   AGE
slurm-worker-slinky   2          4         4       17m
slurm-worker-slonky   1          2         2       17m
```

Slurm Cluster View

Once deployed:

ssh into the slogin pod

```
# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE
slinky    up   infinite     10     idle
slonky    up   infinite      2     idle
```

From here you can run sbatch jobs just as on a regular Slurm cluster.
List the squeue etc .

InfiniBand

HPC workloads require:

- Low latency communication
- High throughput

We wanted to validate the Infiniband offering available in AKS for these purposes, so we needed to test:

- Support on AKS
- Support in Slinky

Detecting the hardware

- NodeFeatureRule to discover the hardware and add labels on the hardware
- Network Operator to manage the drivers and different network tools (ipam etc)

Deploy the nodefeature rule with matching rules for your hardware

```
helm install --wait --create-namespace   \
     -n network-operator node-feature-discovery  \
     node-feature-discovery --create-namespace --repo \
     https://kubernetes-sigs.github.io/node-feature-discovery/charts
kubectl apply -f ./network-operator-nfd.yaml
```

Configuring the NodeFeatureRule

The ./network-operator-nfd.yaml file should contain the pci id of the hardware you are using

```
 device: {op: In, value: ["101c", "101e", "1018", "1016"]}
```

This should result in a

```
$ kubectl  describe node aks-hpcnode-29405625-vmss000000
| grep present
    feature.node.kubernetes.io/pci-15b3.present=true
```

Install the Network Operator

Deploy the operator:

```
helm install nvidia-network-operator \
  nvidia/network-operator -f ./values.yaml
```

This installs:

- MOFED (Melanox OpenFabric) drivers
- multus cni (support for multiple network interfaces)
- nv-ipam-controller (ipam)
- kube-ipoib-cni (ip over ib)

Building Custom Containers

The git repo contains different example containers which you can build an publish:

```
docker build -t slinkyregistry.azurecr.io/dev/slurmd .
docker push slinkyregistry.azurecr.io/dev/slurmd
```

We need this for:

- Slurm worker nodes
- Test workloads
- InfiniBand Test images

Building and Testing IB {.fragile}

```
metadata:
  name: ipoib-test-pod-1
  annotations:
    k8s.v1.cni.cncf.io/networks: slinky-ipoibnetwork
  spec:
    containers:
    - image: slinkyregistry.azurecr.io/ibtest
    ...
       requests:
         cpu: 20
         rdma/rdma\_shared\_device\_a: 1
       limits:
         cpu: 20
         rdma/rdma\_shared\_device\_a: 1
```

Testing IB (2/2)

A pod with IB enabled should have an output like :

```
kubectl -n slinky  describe pod ipoib-test-pod-1  | grep multus
  Normal  AddedInterface  85s   multus             Add eth0 [10.244.4.126
  Normal  AddedInterface  85s   multus             Add net1 [192.168.42.1
```

We can now test multiple tools to verify if IB really works.

- ping
- ib_read_lat
- ucx_perftest

Ucx_perftest output

```
Client on slinky-3 using interface: mlx5_0:1 Server on slinky-0 using int
|              |                |       overhead (usec)         |   bandwidt
|    Stage     | # iterations | 50.0%ile | average | overall |  average |
Final:                   100      0.000   323.792   323.792     3088.41
```

Higher troughput and lower latency for ib (mlx) vs ethernet (eth0)

```
Server on slinky-2 using interface: eth0 Client on slinky-3 using interfa
|              |                |       overhead (usec)         |   bandwidt
|    Stage     | # iterations | 50.0%ile | average | overall |  average |
Final:                   100      5.000  2200.689  2200.689      454.40
```

Adding this to Slinky pods

Slurm defines a default slinky nodeset,
We need to modify the nodeset so it has

- A different image
- A label so the IB network gets picked up
- Resources limits / requirements
- Multiple mounts (share / eessi)

In order to do so we can edit the values.yaml

(look at the git repo for full context)

Resource requirements

```
podSpec:
    # -- The pod resource limits and requests.
    # Ref: https://kubernetes.io/docs/concepts/configuration/manage-res
    resources:
       limits:
         cpu: 100
         rdma/rdma_shared_device_a: 1
       requests:
         cpu: 10
         rdma/rdma_shared_device_a: 1
    securityContext:
      capabilities:
        add:
        - IPC_LOCK
      privileged: true
```

Mountpoints

```
  volumeMounts:
        - name: software-eessi-io-vol
          mountPath: /cvmfs/software.eessi.io
        - name: eessii-share
          mountPath: /mnt/shared
```

```
volumes:
        - name: software-eessi-io-vol
          persistentVolumeClaim:
            claimName: software-eessi-io-pvc
        - name: eessii-share
          persistentVolumeClaim:
            claimName: eessii-share-claim
```

Network annotation & different image

```
metadata:
     annotations:
       k8s.v1.cni.cncf.io/networks: slinky-ipoibnetwork
```

```
 image:
       repository: slinkyregistry.azurecr.io/slinkypoc/slurmd
       tag: latest
```

Running Gromacs

We used Gromacs from the EESSI repo to test all Slinky variations
Gromacs needs a larger Dataset when running on multiple CPU's (TestCaseB)

```
Running on 7 nodes with total 308 cores, 308 processing units
  Cores per node:           44
  Logical processing units per node:   44
  OS CPU Limit / recommended threads to start per node:   44
  ...
[1771945140.098412] [slinky-4:678  :0]          module.c:282  UCX  DEBUG
[1771945141.638798] [slinky-4:678  :0]            time.c:22   UCX  DEBUG
[1771945141.638874] [slinky-4:678  :0]     ucp_context.c:2257 UCX  INFO
[1771945141.638883] [slinky-4:678  :0]     ucp_context.c:2015 UCX  DEBUG
[1771945141.638885] [slinky-4:678  :0]     ucp_context.c:2022 UCX  DEBUG
```

Adding GPU support to K8s

Install the NVidia GPU operator

```
$ helm install --wait --generate-name \
    -n gpu-operator --create-namespace \
    nvidia/gpu-operator
NAME: gpu-operator-1772014947
LAST DEPLOYED: Wed Feb 25 11:22:29 2026
NAMESPACE: gpu-operator
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

This deploys a number of pods that will allow support for the GPU's in the

Pods deployed

```
$kubectl  get pods -n gpu-operator -o wide
NAME                                                              READY
gpu-feature-discovery-pws2n                                       1/1
gpu-operator-1772014947-node-feature-discovery-gc-9f9b967978r9f   1/1
gpu-operator-1772014947-node-feature-discovery-worker-wh9bp       1/1
gpu-operator-69f6d789d-mb48l                                      1/1
nvidia-container-toolkit-daemonset-zvkv2                          1/1
nvidia-cuda-validator-x2px7                                       0/1
nvidia-dcgm-exporter-k7jxv                                        1/1
nvidia-device-plugin-daemonset-kjsbr                              1/1
nvidia-operator-validator-xjn82                                   1/1
```

GPU Detection inside the container

There are testpods/jobs out there, try them first.

We created a new nodeset in Slinky (slonky) that had the appropriate hardware requirements

```
resources:
   limits:
     nvidia.com/gpu: 1
```

Example output

Then we ran a slurm job test-gpu which leverages the EESSI GPU drivers libraries

```
sbatch  -p slonky test-gpu.sbatch
Submitted batch job 112
root@slurm-login-slinky-cc57dbb77-j82kn:/mnt/shared# squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
112    slonky  gmx_gpu     root  R       0:02      2 slonky-[0-1]
```

GPU Support in Slurm

Sbatch doesn't need to be told it requires a GPU if you give it one.
But if you tell it the job needs a GPU you need to reconfigure Slurm so it knows.

Adapted the values.yaml again

```
AutoDetect=nvidia
Gres: gpu:1
```

This way Slurm knows better which instances have GPU if you require the job to have GPU.

Running Gromacs in Slinky with GPU.

Using the_script.sbatch

```
 `Command line:
  gmx_mpi mdrun -s topol.tpr -maxh 0.50 -notunepme -noconfout -nsteps 100
```

```
Running on 2 nodes with total 8 cores, 8 processing units, 2 compatible G
  Cores per node:            4
  Logical processing units per node:    4
  OS CPU Limit / recommended threads to start per node:    4
  Compatible GPUs per node:  1
  All nodes have identical type(s) of GPUs
Hardware detected on host slonky-1 (the node of MPI rank 0):
  CPU info:
    Vendor: AMD
```

Conclusion

We successfully demonstrated:

- Slurm running on Kubernetes
- GPU workloads in containers
- InfiniBand networking
- All code is in the repo

This shows Kubernetes can support **HPC environments**.

Questions