Scheduling & Orchestration

5th High Performance Container Workshop - ISC19

Scope and Introduction

This segment focuses on **SCHEDULING & ORCHESTRATION** aspects.

We are going to touch on what was said before.

Docker SWARM: Light-weight orchestrator

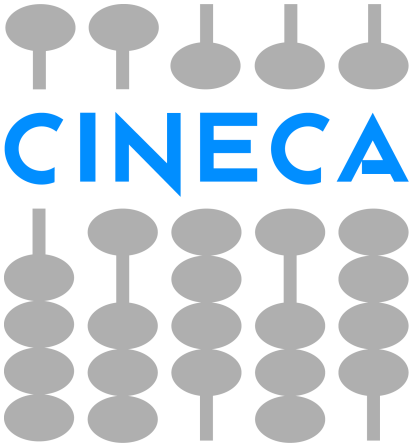
HPCW @ ISC2019

Abdulrahman Azab

Dept. of Research Computing

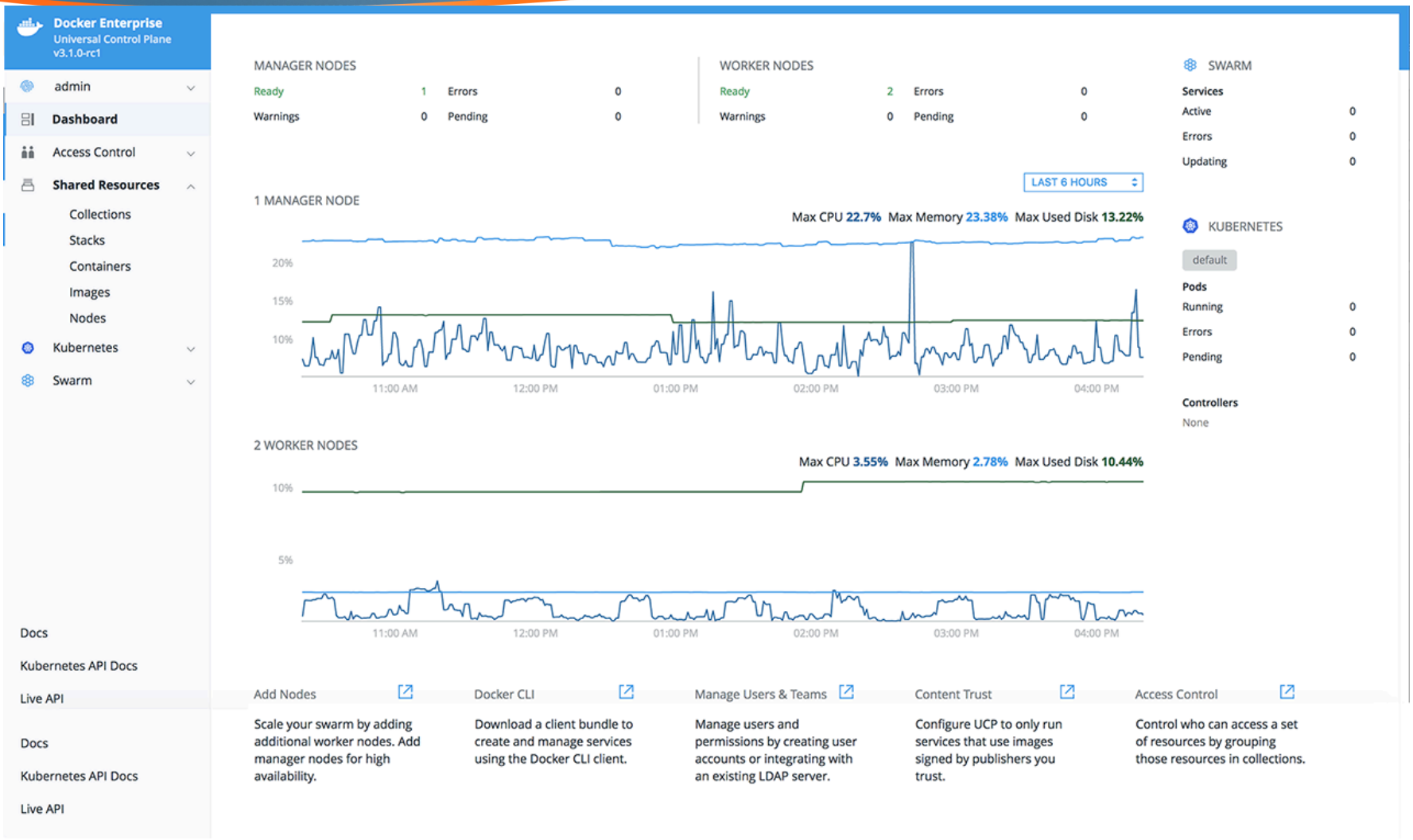
University of Oslo, Norway

PRACE 6IP WP6.2.3: The deployment of containers and full virtualised tools into HPC infrastructures

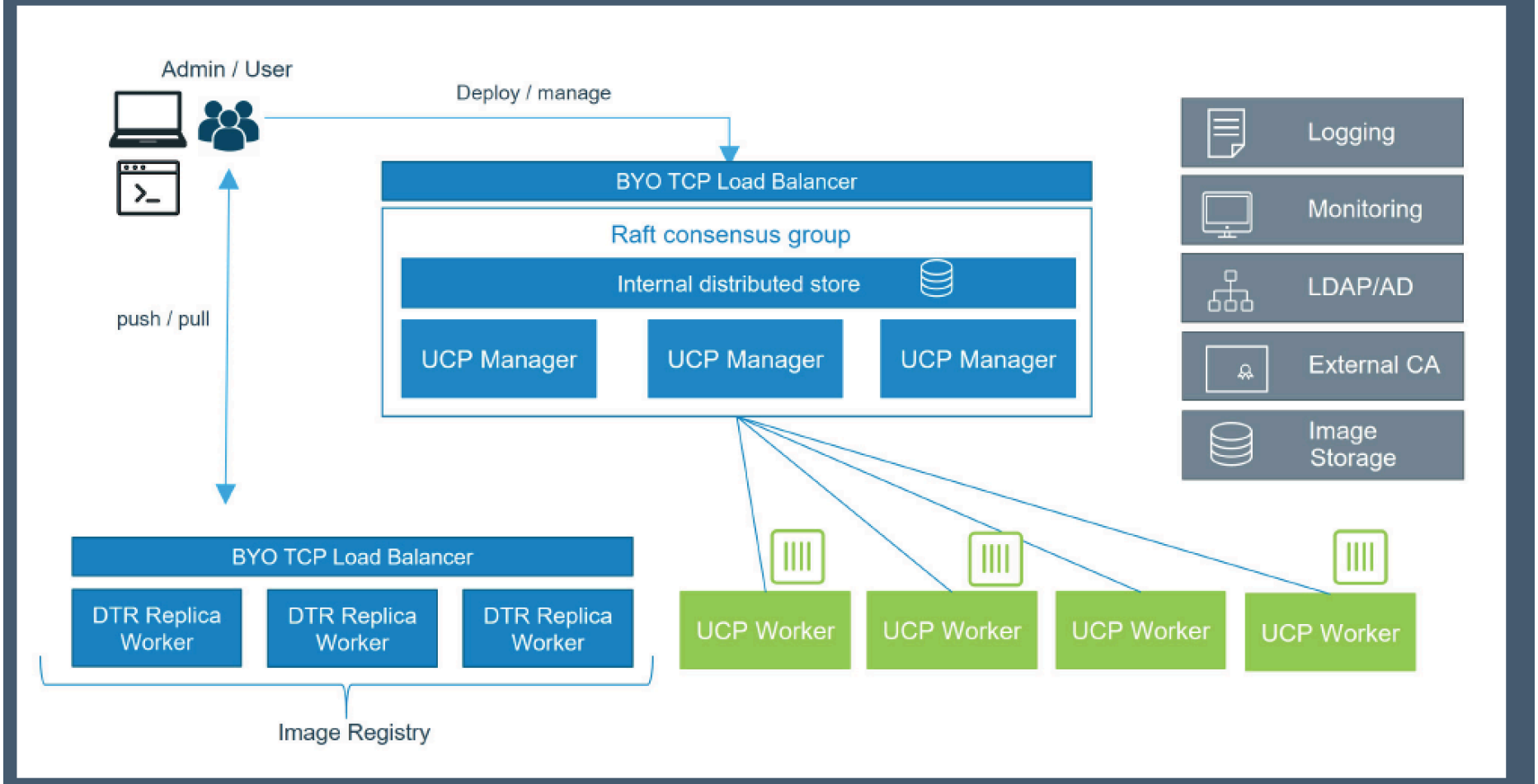


ISC High Performance
The HPC Event.

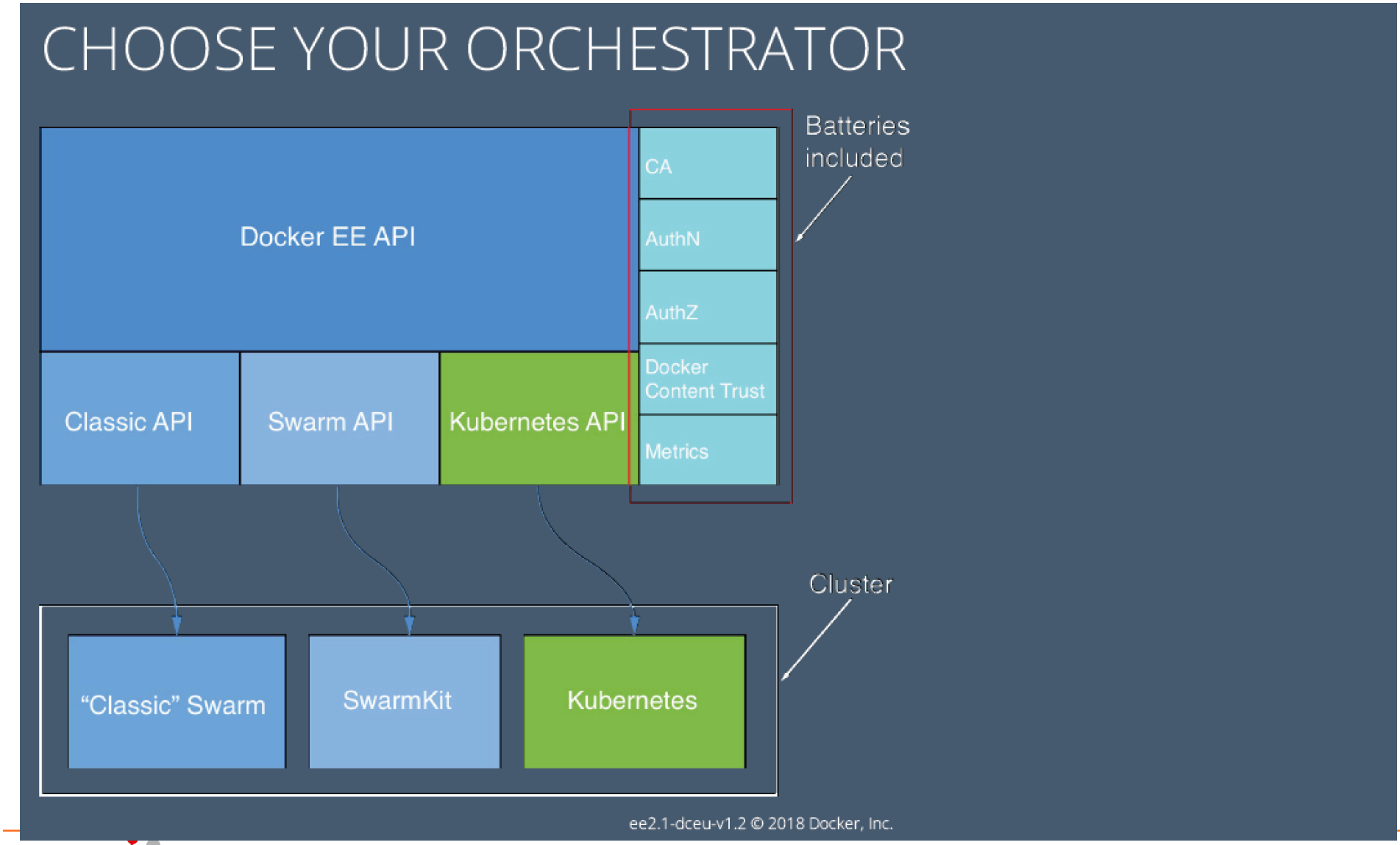
Docker UCP


Docker Enterprise
Universal Control Plane
v3.1.0-rc1
admin
Dashboard
Access Control
Shared Resources
Collections
Stacks
Containers
Images
Nodes
Kubernetes
Swarm
Docs
Kubernetes API Docs
Live API
Docs
Kubernetes API Docs
Live API
MANAGER NODES
Ready 1 Errors 0
Warnings 0 Pending 0
WORKER NODES
Ready 2 Errors 0
Warnings 0 Pending 0
SWARM
Services
Active 0
Errors 0
Updating 0
LAST 6 HOURS
1 MANAGER NODE
Max CPU 22.7% Max Memory 23.38% Max Used Disk 13.22%
20%
15%
10%
11:00 AM
12:00 PM
01:00 PM
02:00 PM
03:00 PM
04:00 PM
KUBERNETES
default
Pods
Running 0
Errors 0
Pending 0
Controllers
None
2 WORKER NODES
Max CPU 3.55% Max Memory 2.78% Max Used Disk 10.44%
10%
5%
11:00 AM
12:00 PM
01:00 PM
02:00 PM
03:00 PM
04:00 PM
Add Nodes
Scale your swarm by adding additional worker nodes. Add manager nodes for high availability.
Docker CLI
Download a client bundle to create and manage services using the Docker CLI client.
Manage Users & Teams
Manage users and permissions by creating user accounts or integrating with an existing LDAP server.
Content Trust
Configure UCP to only run services that use images signed by publishers you trust.
Access Control
Control who can access a set of resources by grouping those resources in collections.

DOCKER EE ARCHITECTURE

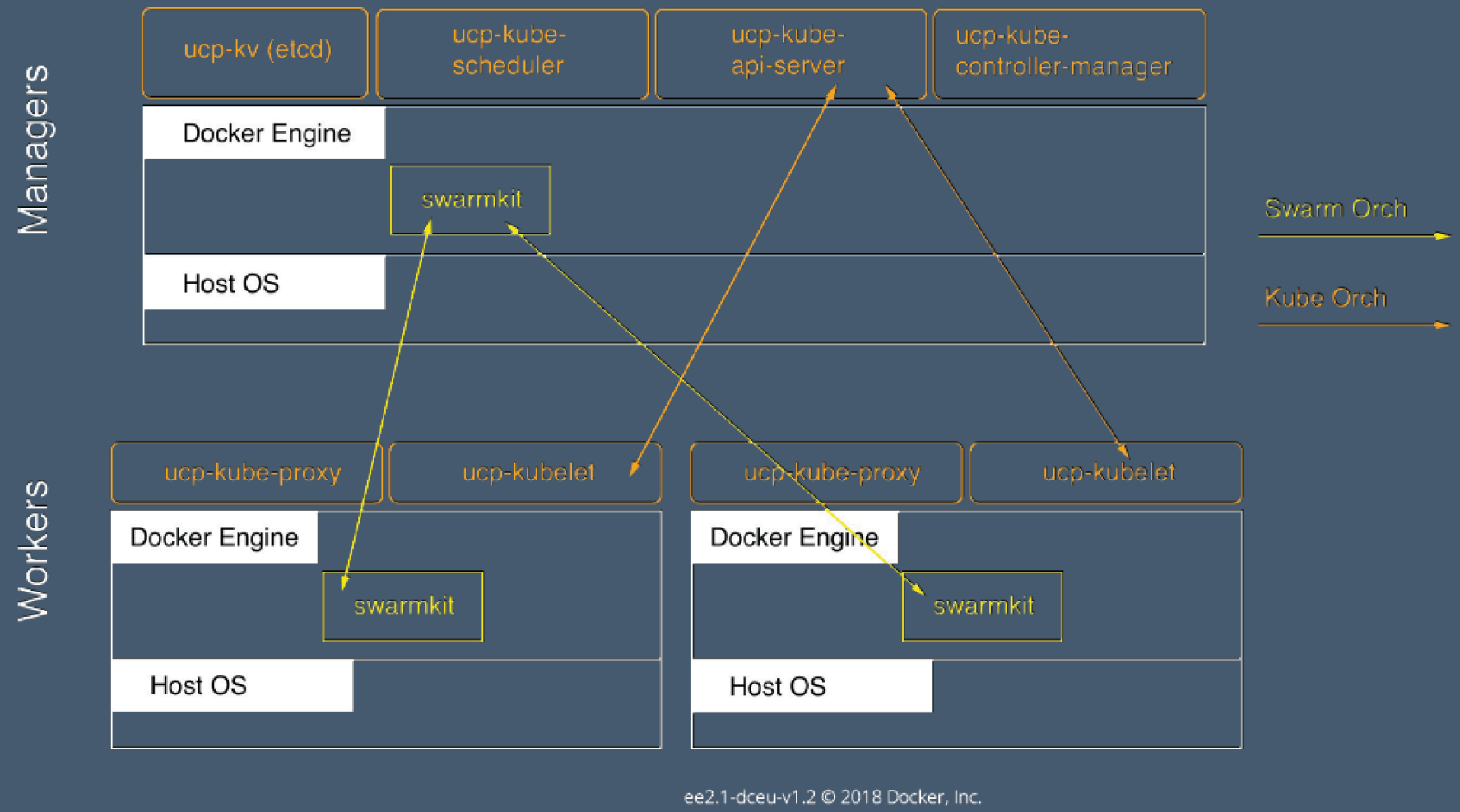


```
[centos@ucp-manager-0 ~]$ UCP_IP=<ucp-manager-0 IP>
[centos@ucp-manager-0 ~]$ UCP_FQDN=<ucp-manager-0 FQDN>
[centos@ucp-manager-0 ~]$ docker container run --rm -it --name
ucp \
-v /var/run/docker.sock:/var/run/docker.sock \
docker/ucp:3.1.0 install \
--admin-username admin \
--admin-password adminadmin \
--san ${UCP_IP} \
--san ${UCP_FQDN}
```

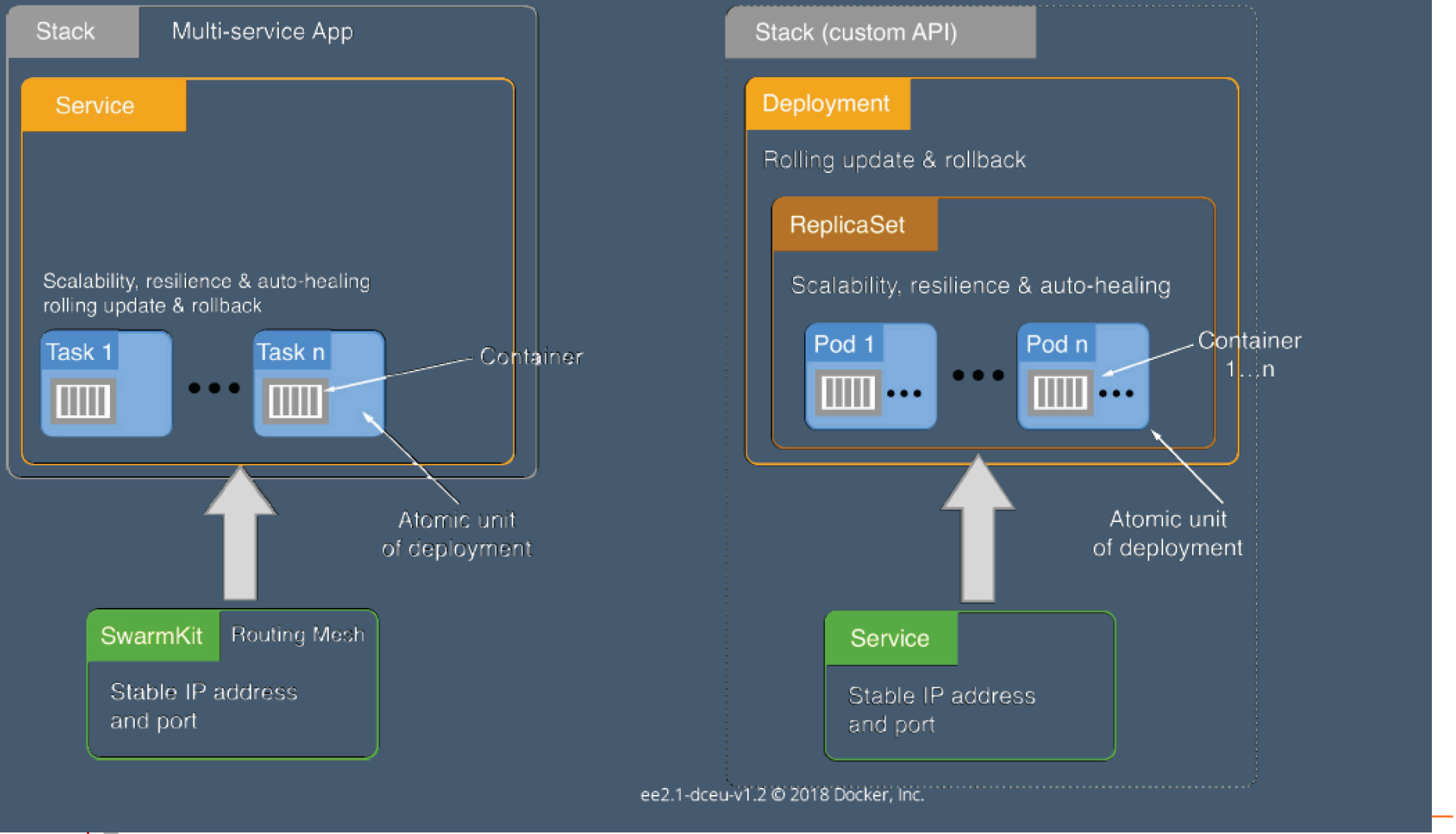

CHOOSE YOUR ORCHESTRATOR
Batteries included
CA
AuthN
AuthZ
Docker Content Trust
Metrics
Docker EE API
Classic API
Swarm API
Kubernetes API
Cluster
"Classic" Swarm
SwarmKit
Kubernetes
ee2.1-dceu-v1.2 © 2018 Docker, Inc.

Orchestration

ORCHESTRATORS IN UCP



ORCHESTRATION COMPONENTS



Container Network Operations

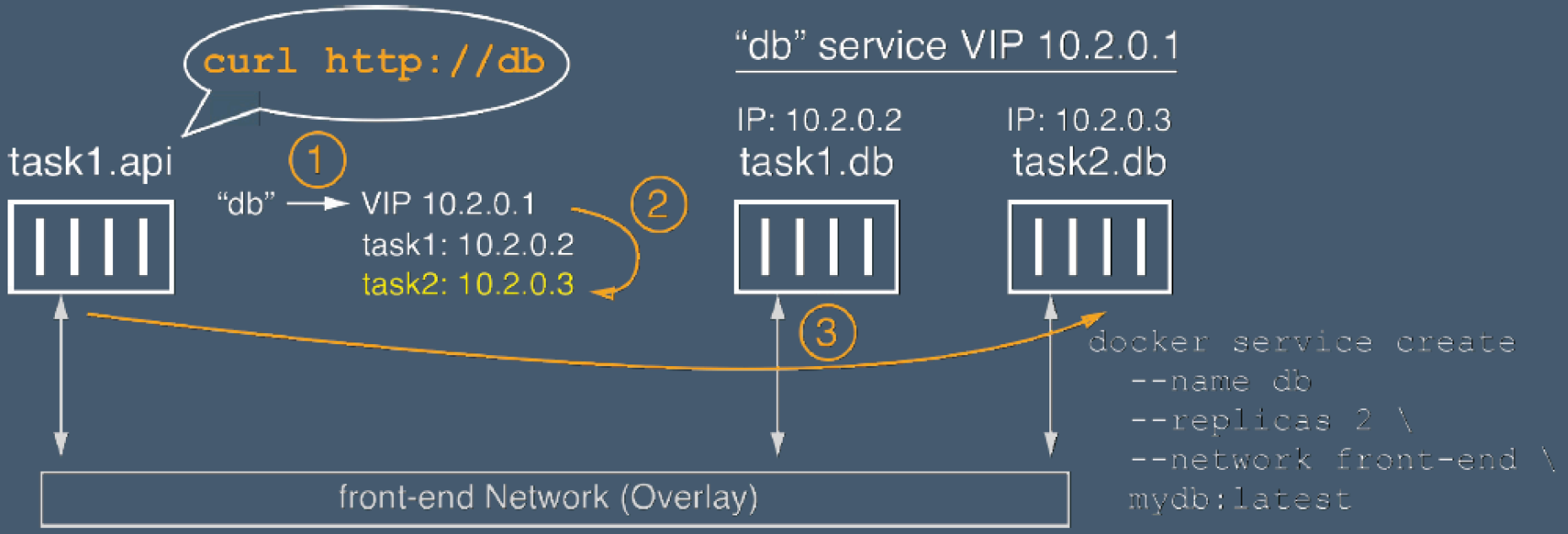
CHOOSING THE RIGHT NETWORKING TOOL

Three questions:

- Is the request originator internal or external to your cluster?
- Are the destination containers stateless or stateful?
- Are you using Swarm or Kubernetes?

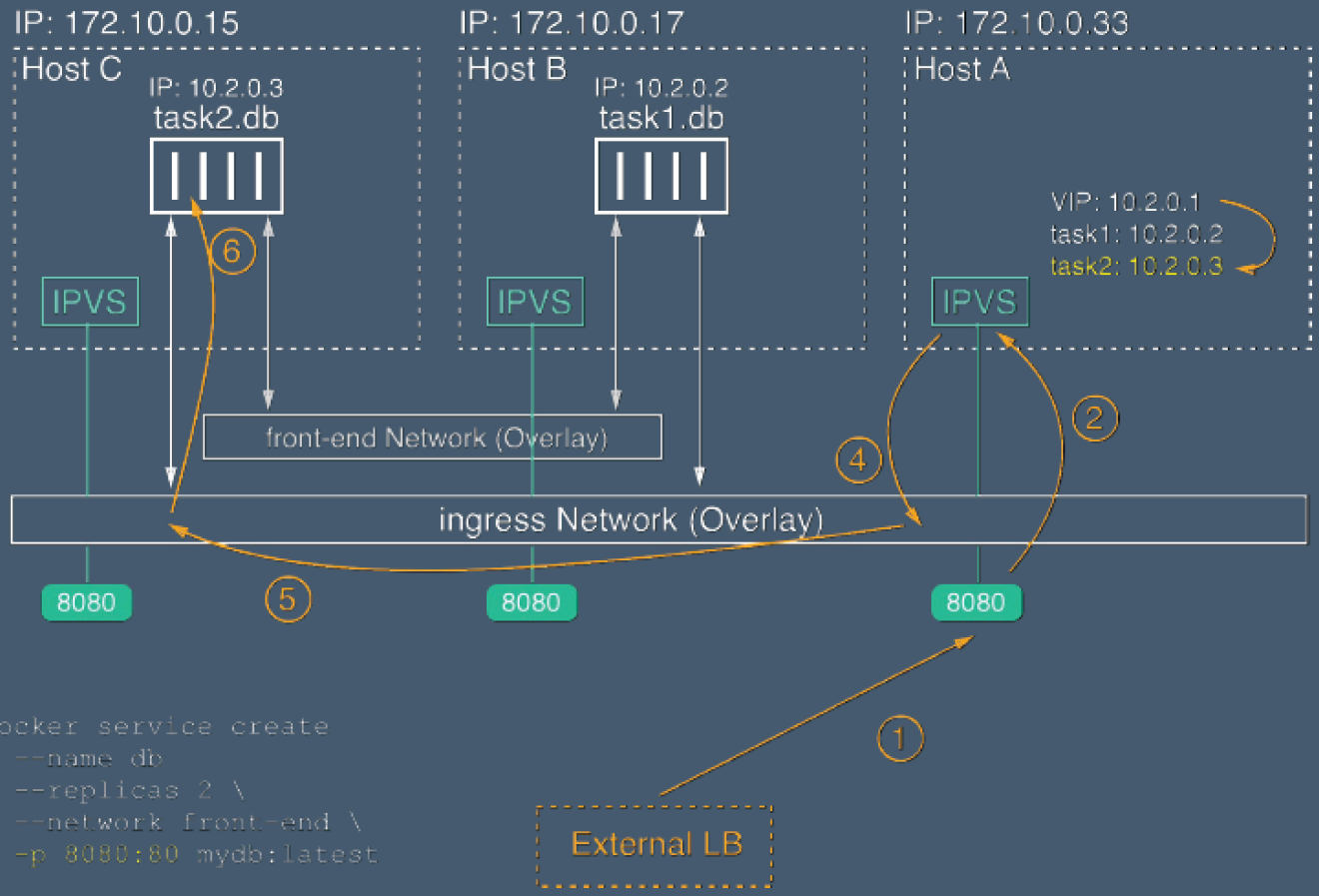
INTERNAL / STATELESS / SWARM

Solution: Swarm VIPs



EXTERNAL / STATELESS / SWARM

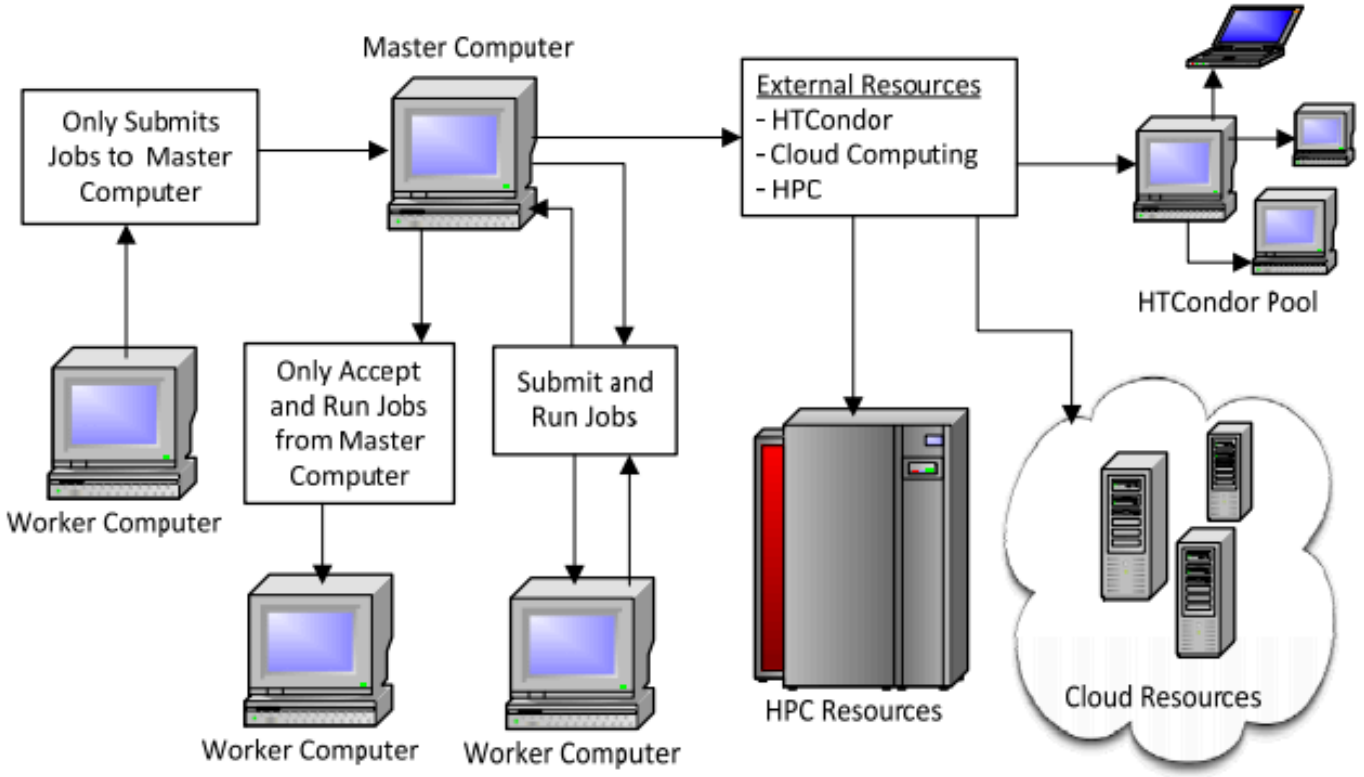
Solution: Swarm L4 Routing Mesh




```
docker service create
  --name db
  --replicas 2 \
  --network front-end \
  -p 8080:80 mydb:latest
```

Use case:
Containerised HPC
With HTConcor and swarm

HTCondor


Master Computer
Only Submits Jobs to Master Computer
External Resources
- HTCondor
- Cloud Computing
- HPC
HTCondor Pool
Worker Computer
Only Accept and Run Jobs from Master Computer
Submit and Run Jobs
HPC Resources
Cloud Resources
Worker Computer
Worker Computer

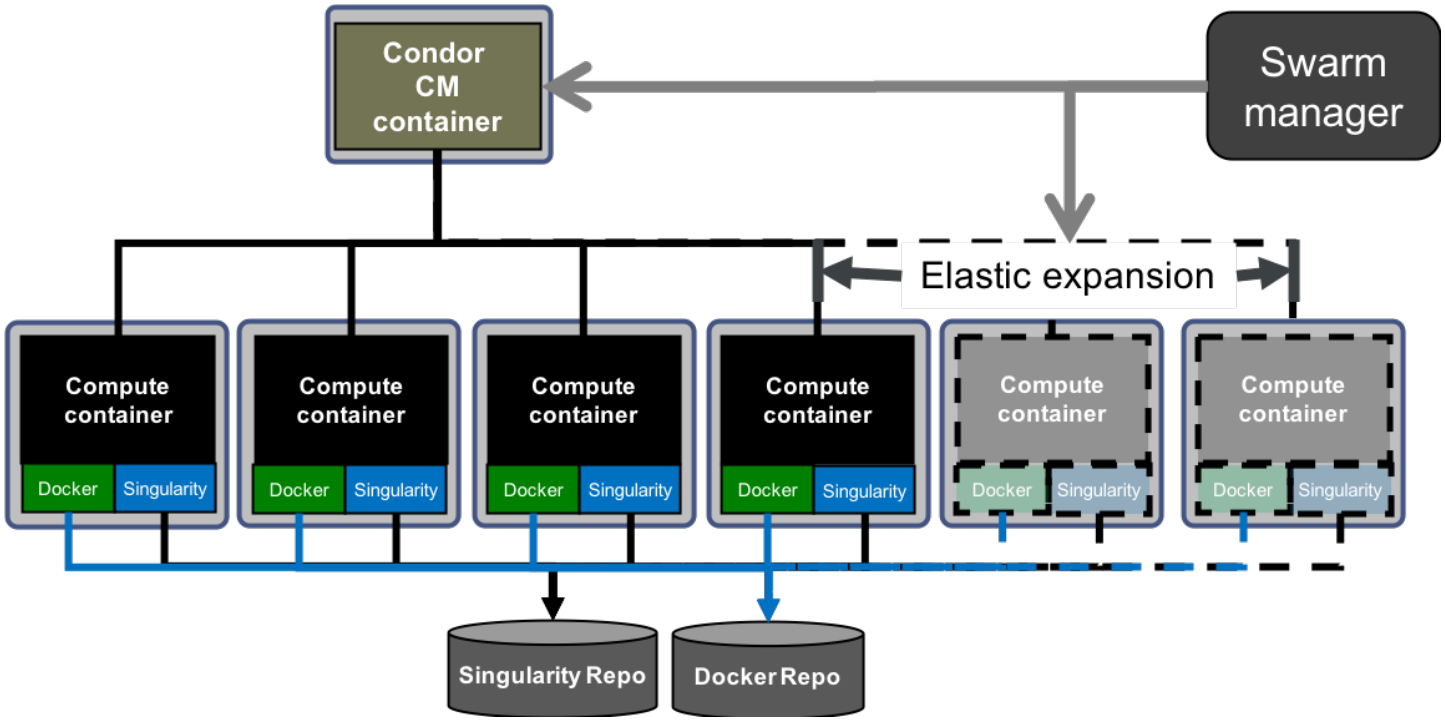
HTCondor VM universe

```
universe = vm
executable = vmware_sample_job
log = simple.vm.log.txt
vm_type = vmware
vm_memory = 64
vmware_dir = C:\condor-test
vm_checkpoint = true
queue
```

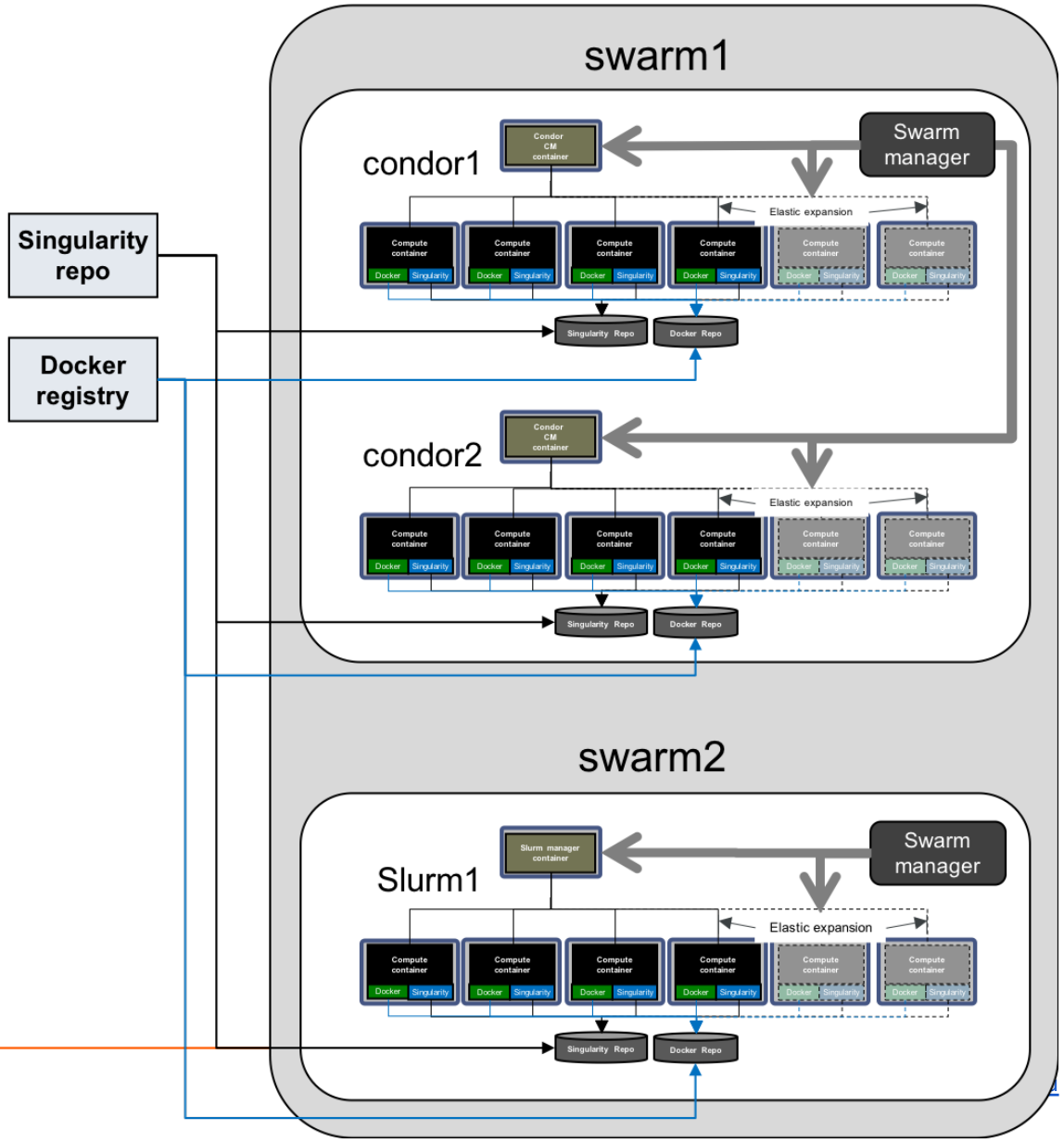
HTCondor Docker universe

```
universe = docker
docker_image = debian
executable = /bin/cat
arguments = /etc/hosts
output = out.$(Process)
error = err.$(Process)
request_memory = 100M
queue 10
```

Containerised HTCondor on Swarm



Docker only cluster



```
[cloud-user@swarm-1 ~]$ docker node ls
ID                            HOSTNAME              STATUS              AVAILABILITY        MANAGER STATUS
RSION
smzp49rwzhc6g5m3rbnwa4o41 *   swarm-1.novalocal     Ready               Active              Leader
e
2x9m0zr61qnm7gvnhwwjgn263      swarm-2.novalocal     Ready               Active
e
n99dyxddexkr0w0wq8ysz3tw3      swarm-3.novalocal     Ready               Active
```

```
[cloud-user@swarm-1 ~]$ docker service ls
ID                  NAME                                MODE                REPLICAS
                                  PORTS
5rme9l2t03vi        galaxy_galaxy-htcondor              replicated          1/1
axy-htcondor:18.01b
z6seh1nyhjep        galaxy_galaxy-htcondor-executor     replicated          2/2
axy-htcondor-executor:18.01b
vjq5y0dd5ni5        galaxy_galaxy-htcondor-executor-big replicated          1/1
axy-htcondor-executor:18.01b
```

```
[cloud-user@swarm-1 ~]$ docker service ps galaxy_galaxy-htcondor-executor | grep Running
59zcdu1us630        galaxy_galaxy-htcondor-executor.1       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-3
.novalocal   Running             Running 2 months ago
4dygqif85wi2        galaxy_galaxy-htcondor-executor.2       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-3
.novalocal   Running             Running 2 months ago
```

```
[cloud-user@swarm-1 ~]$ docker service scale galaxy_galaxy-htcondor-executor=5
galaxy_galaxy-htcondor-executor scaled to 5
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service converged
[cloud-user@swarm-1 ~]$ docker service ps galaxy_galaxy-htcondor-executor | grep Running
59zcdu1us630        galaxy_galaxy-htcondor-executor.1       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-3
.novalocal   Running             Running 2 months ago
4dygqif85wi2        galaxy_galaxy-htcondor-executor.2       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-3
.novalocal   Running             Running 2 months ago
t1cadorzttuh        galaxy_galaxy-htcondor-executor.3       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-2
.novalocal   Running             Running 15 seconds ago
av7mt5t7i6jk        galaxy_galaxy-htcondor-executor.4       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-2
.novalocal   Running             Running 15 seconds ago
920zv43mmaox        galaxy_galaxy-htcondor-executor.5       quay.io/bgruening/galaxy-htcondor-executor:18.01b   swarm-2
.novalocal   Running             Running 15 seconds ago
```

```
[cloud-user@swarm-3 ~]$ docker exec 53463399ae3a condor_status
Name               OpSys      Arch   State     Activity LoadAv Mem   ActvtyTime

2b928ae59394       LINUX      X86_64 Unclaimed Idle      0.160 1024  0+00:00:03
5ed8939b3952       LINUX      X86_64 Unclaimed Idle      0.000 2048 77+05:43:00
2073aae2920d       LINUX      X86_64 Unclaimed Idle      0.160 1024  0+00:00:03
a4740170e2cf       LINUX      X86_64 Unclaimed Idle      0.000 1024 77+05:46:51
c3d66659047a       LINUX      X86_64 Unclaimed Idle      0.000 1024 77+05:46:55
c55e3685352c       LINUX      X86_64 Unclaimed Idle      0.160 1024  0+00:00:03

                     Total Owner Claimed Unclaimed Matched Preempting Backfill  Drain

       X86_64/LINUX     6     0       0         6       0          0        0      0

              Total     6     0       0         6       0          0        0      0
```


Q&A

Kubernetes and HPC

Daniel Gruber
The UberCloud

Kubernetes Intro: Native Batch Job Submission

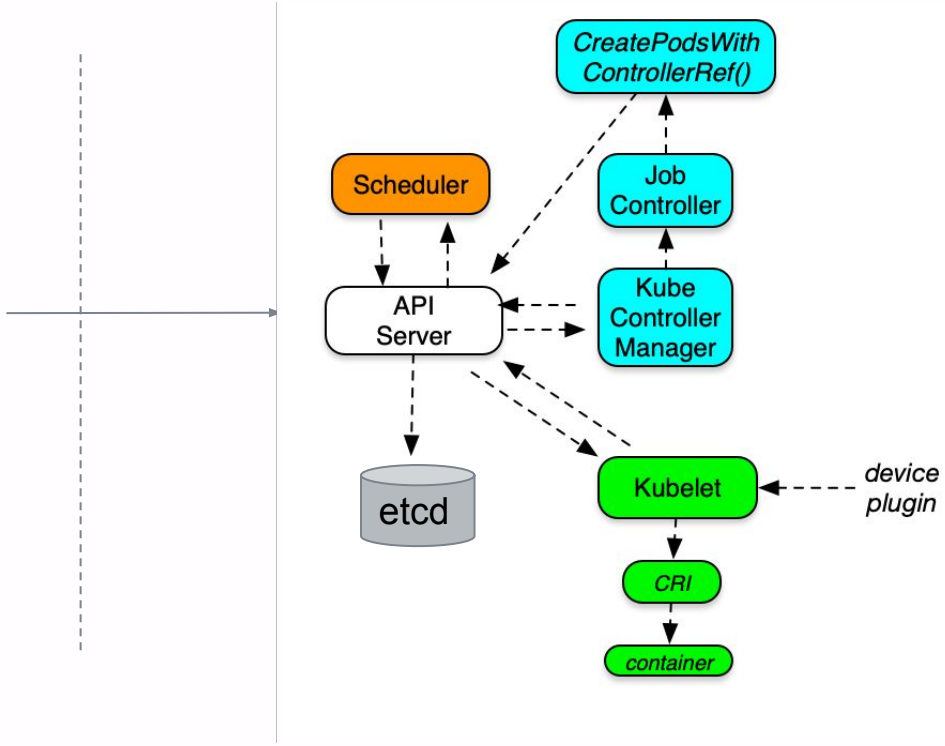
```
kubectl apply -f job.yaml
```

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: sleep
spec:
  template:
    spec:
      containers:
      - name: sleep
        image: busybox
        command: ["sleep",  "100"]
      restartPolicy: Never
```

Kubernetes Intro: Native Batch Job Submission

```
kubectl apply -f job.yaml
```

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: sleep
spec:
  template:
    spec:
      containers:
      - name: sleep
        image: busybox
        command: ["sleep",  "100"]
      restartPolicy: Never
```



Deployment of more complex applications:

helm, Operators

Kubernetes and HPC

HPC Requirements	Kubernetes features
➢ Fast job submission times	❖ *kubectl*, API
➢ Multi-node jobs (distributed memory)	❖ No direct MPI Support
➢ Support for GPUs, FPGAs	❖ GPUs supported via *device plugin*
➢ Easily accessible by HPC applications	❖ AI
➢ Job queueing capabilities	❖ Pods, since 1.9
➢ Job prioritization	❖ 1.14
➢ Job pre-emption	❖ 1.14

Kubernetes and HPC (continued)

HPC Requirements	Kubernetes features
➢ Advance reservation	❖ No
➢ License management capabilities	❖ No, Scheduler Extenders
➢ Support for Infiniband, Omnipath	❖ device plugin
➢ Full hardware utilization	❖ More daemons, typically virtualized, CPU manager
➢ Resource control	❖ Memory and CPU resources, Resource quotas
➢ Accounting and reporting	❖ Yes
➢ Storage	❖ PVs and PVCs, storageclass, hostpath

References

- Kubeflow: https://github.com/kubeflow/kubeflow
- MPI Operator: https://github.com/kubeflow/mpi-operator
- Poseidon: https://github.com/kubernetes-sigs/poseidon
- CPU Manager: https://kubernetes.io/docs/tasks/administer-cluster/cpu-management-policies/
- Kube batch: https://github.com/kubernetes-sigs/kube-batch
- Volcano: https://github.com/volcano-sh/volcano
- Slurm Operator: https://github.com/sylabs/slurm-operator
- Lustre: https://github.com/kvaps/kube-lustre
- DRMAA2: https://github.com/dgruber/drmaa2os
- qsub for k8s: https://github.com/dgruber/qsub
- RDMA: https://github.com/Mellanox/k8s-rdma-sriov-dev-plugin
- Univa's Command: http://www.univa.com/products/navops.php
- UberCloud: https://www.theubercloud.com/

Thank you

Daniel.Gruber@TheUberCloud.com

Some of my strange thoughts: www.drmaa2.org (--> https://www.gridengine.eu)

https://github.com/dgruber

AWS Storage Blog

New on the Open Source blog: Using the FSx for Lustre CSI Driver with Amazon EKS

by Sean White | on 16 APR 2019 | in Amazon Elastic Container Service, Amazon Elastic Container Service For Kubernetes, Amazon FSx For Lustre, Storage | Permalink | Comments | Share

From time to time the AWS Storage Blog will point you to storage related blog posts from other AWS blog channels that we think you will find interesting or helpful for your storage use cases. We wanted to share a post from the Open Source Blog that highlights a walk-through of using Amazon FSx for Lustre Container Storage Interface (CSI) Driver with Amazon EKS.

The Amazon Elastic Container Service for Kubernetes (Amazon EKS) team has been busy building CSI drivers for all our storage solutions, including Amazon FSx for Lustre: a fully-managed file system which has integrations with S3 and is optimized for compute-intensive workloads such as high-performance computing (HPC) and machine learning. Because the AWS FSx CSI Driver is potentially useful to any Kubernetes user, we've donated it to Kubernetes SIG-AWS. This blog post focuses on deploying the AWS FSx CSI driver to an Amazon EKS cluster.

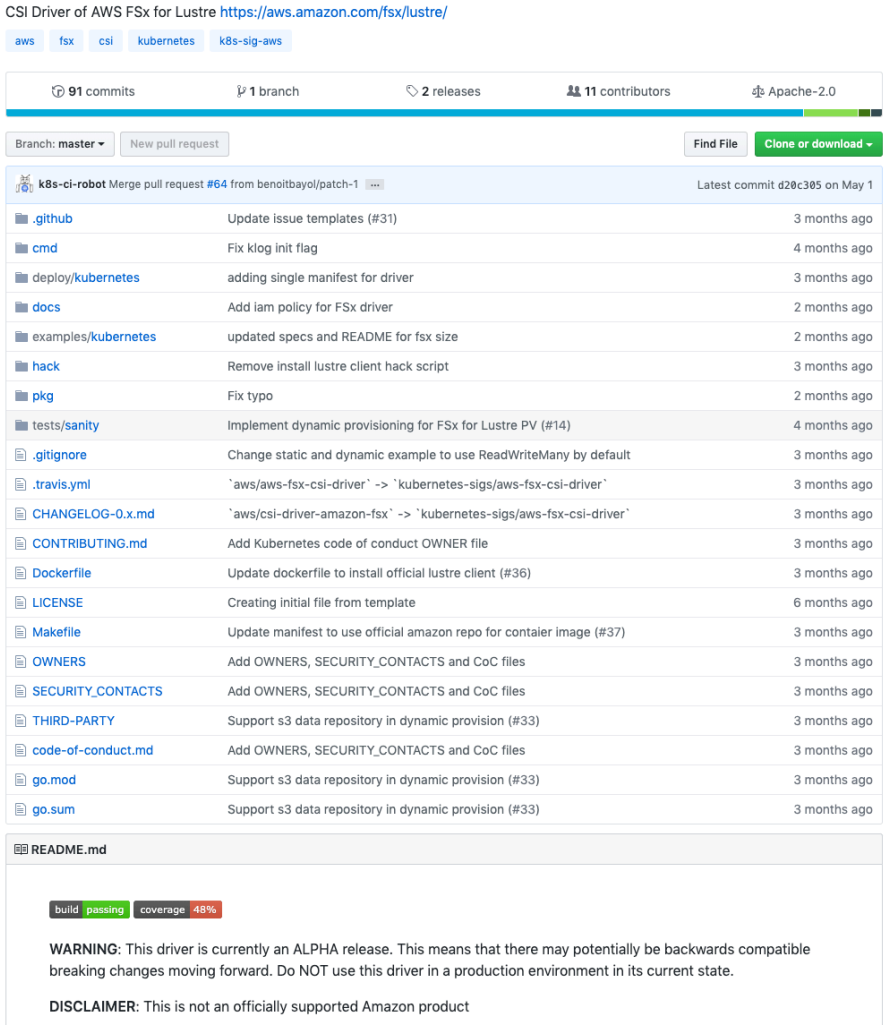
You can now dynamically provision Amazon FSx for Lustre filesystems for your high performance computing workloads, and have a container automatically connected into this filesystem. FSx for Lustre provides a high-performance file system optimized for fast processing of workloads such as machine learning, high performance computing (HPC), video processing, financial modeling, and electronic design automation (EDA). These workloads commonly require data to be presented via a fast and scalable file system interface, and typically have data sets stored on long-term data stores like Amazon S3.

You can also use dynamic provisioning to consume the same persistent volume claim from multiple pods from different nodes. If you need to start your containers with a dataset automatically available, for example loading a machine learning training data set, you can even connect your StorageClass to an existing Amazon S3 bucket by using Dynamic Provisioning with Data Repository. Other examples can be found in the FSx for Lustre CSI Driver documentation.

To learn more, see the full AWS Open Source Blog post.

https://aws.amazon.com/blogs/opensource/using-fsx-lustre-csi-driver-amazon-eks/

https://github.com/kubernetes-sigs/aws-fsx-csi-driver



CSI Driver of AWS FSx for Lustre https://aws.amazon.com/fsx/lustre/

aws fsx csi kubernetes k8s-sig-aws

91 commits | 1 branch | 2 releases | 11 contributors | Apache-2.0

Branch: master | New pull request | Find File | Clone or download

k8s-ci-robot Merge pull request #64 from benoitbayol/patch-1 — Latest commit d20c305 on May 1

.github	Update issue templates (#31)	3 months ago
cmd	Fix klog init flag	4 months ago
deploy/kubernetes	adding single manifest for driver	3 months ago
docs	Add iam policy for FSx driver	2 months ago
examples/kubernetes	updated specs and README for fsx size	2 months ago
hack	Remove install lustre client hack script	3 months ago
pkg	Fix typo	2 months ago
tests/sanity	Implement dynamic provisioning for FSx for Lustre PV (#14)	4 months ago
.gitignore	Change static and dynamic example to use ReadWriteMany by default	3 months ago
.travis.yml	`aws/aws-fsx-csi-driver` -> `kubernetes-sigs/aws-fsx-csi-driver`	3 months ago
CHANGELOG-0.x.md	`aws/csi-driver-amazon-fsx` -> `kubernetes-sigs/aws-fsx-csi-driver`	3 months ago
CONTRIBUTING.md	Add Kubernetes code of conduct OWNER file	3 months ago
Dockerfile	Update dockerfile to install official lustre client (#36)	3 months ago
LICENSE	Creating initial file from template	6 months ago
Makefile	Update manifest to use official amazon repo for contaier image (#37)	3 months ago
OWNERS	Add OWNERS, SECURITY_CONTACTS and CoC files	3 months ago
SECURITY_CONTACTS	Add OWNERS, SECURITY_CONTACTS and CoC files	3 months ago
THIRD-PARTY	Support s3 data repository in dynamic provision (#33)	3 months ago
code-of-conduct.md	Add OWNERS, SECURITY_CONTACTS and CoC files	3 months ago
go.mod	Support s3 data repository in dynamic provision (#33)	3 months ago
go.sum	Support s3 data repository in dynamic provision (#33)	3 months ago

README.md

build passing | coverage 48%

WARNING: This driver is currently an ALPHA release. This means that there may potentially be backwards compatible breaking changes moving forward. Do NOT use this driver in a production environment in its current state.

DISCLAIMER: This is not an officially supported Amazon product

Using K8s Operators for RDMA Workloads

Dror Goldenberg, Liel Shoshan - Mellanox Technologies

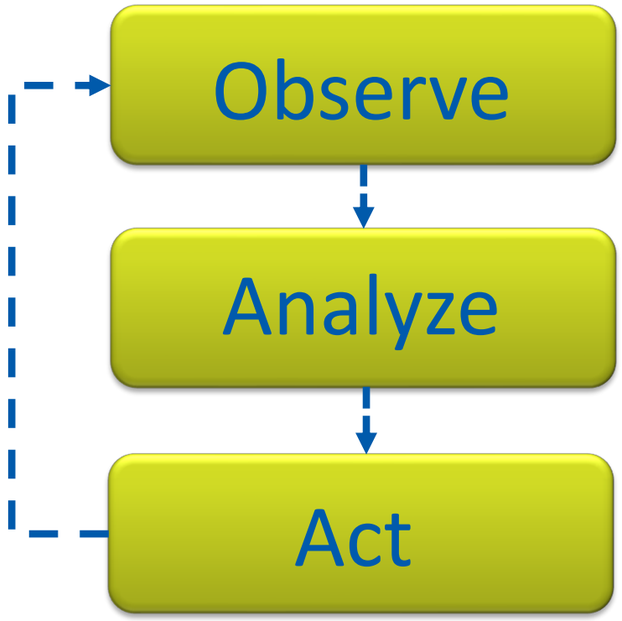
ISC Container Workshop Frankfurt, June 2019

Operators

What are Operators?

- **Operator** - method of packaging, deploying and managing a Kubernetes application

- Enables developers to
  - Extend and add new functionalities
  - Automate administration tasks as if they were a native Kubernetes component
- Operator = set of application-specific custom controllers



Node is configured with 48 SR-IOV VFs

Desired state
- Each node is configured with 64 SR-IOV VFs

Reconcile
- Configure the node with 64 SR-IOV VFs

Operators

- **Why are Operators better than other tools out there?**
  - Using standard Kubernetes tools, CLI and API
  - Can monitor the cluster, change pods/services, scale up/down and call endpoints of the running applications
  - Smarter and more tailored than generic tools

- **Who builds an Operator?**
  - Best built by those that are experts in the “business logic” of installing, running and upgrading an application
  - Creation of an Operator typically starts by automating an application installation and self-service provisioning capabilities
  - Evolves to take on more complex automation

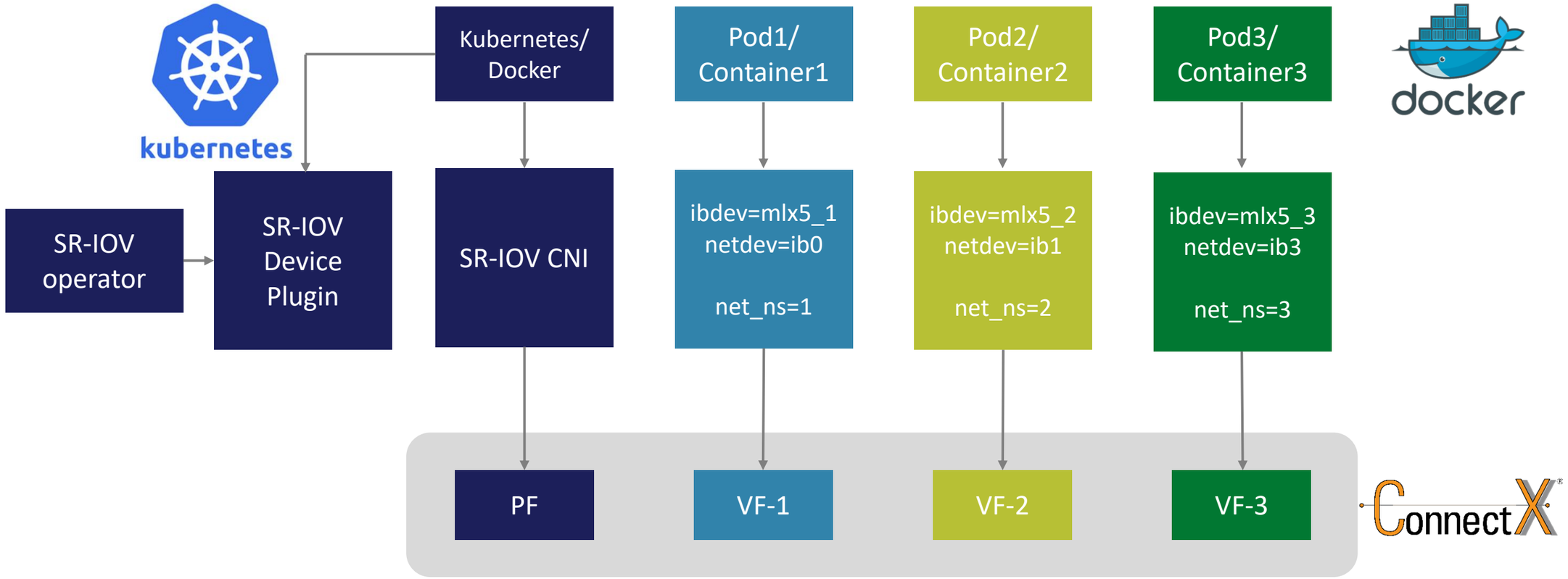
K8s RDMA Solution

RDMA/RoCE in K8s

- SR-IOV based solution
  - Each Container is assigned with an SR-IOV VF
  - Consist of the following K8s plugins
    - SR-IOV Device Plugin for Kubernetes on GitHub
      - SR-IOV VF provisioning
    - SR-IOV CNI Plugin for Kubernetes on GitHub
- Virtual networking in shared device mode
  - An IB device is shared between different pods

RDMA at the Container Level using SR-IOV

Technical Overview



Mellanox ConnectX Adapter Card with SR-IOV Enabled

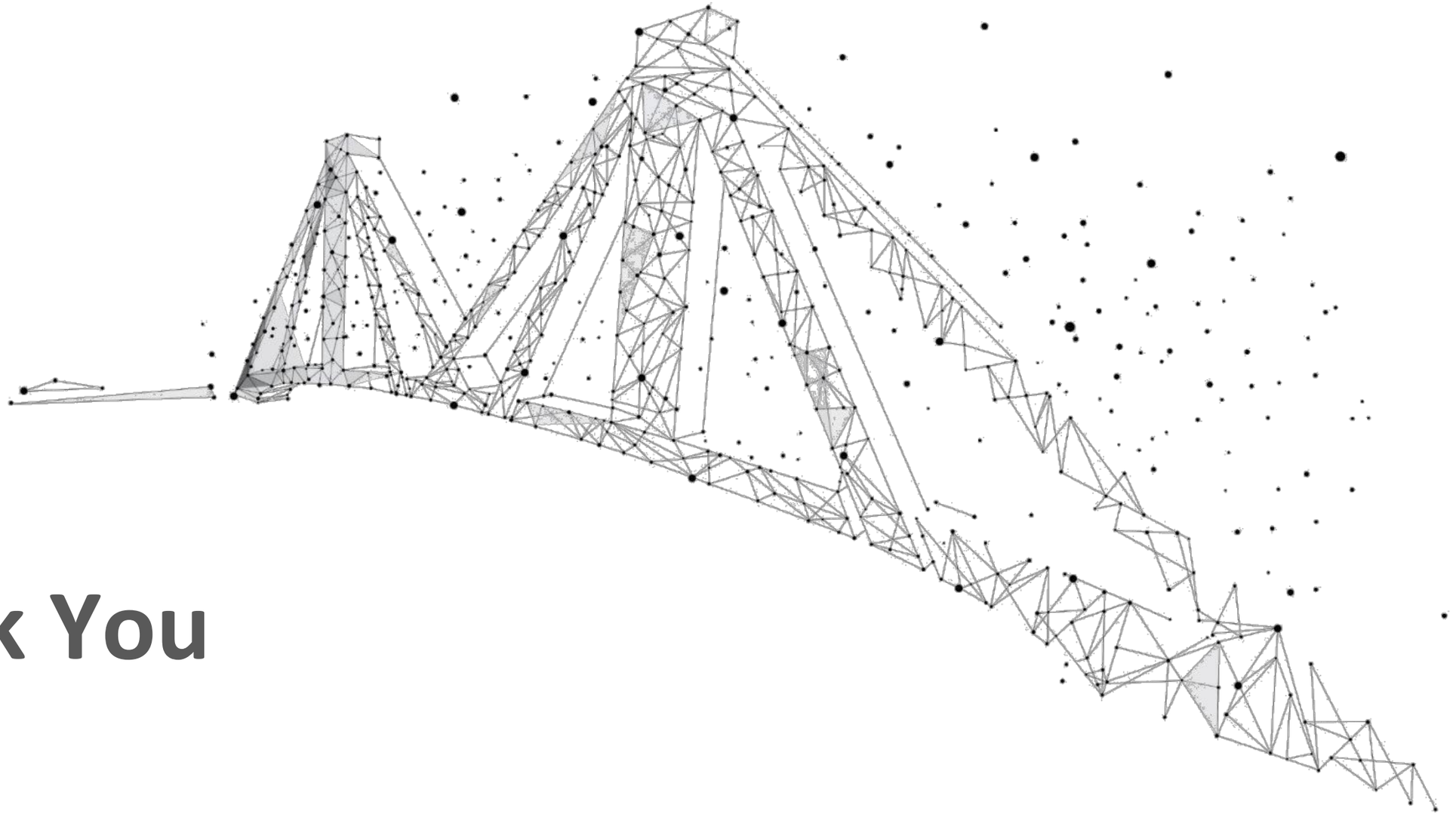
- Every container/POD has an IB device (mlx5_1,2,3) and netdevice

Operators for RDMA Workloads

- Why do we need operators for running RDMA workloads?
  - Running RDMA/RoCE workloads requires several configurations
  - Automating these configuration allows ease of use

- Required configurations
  - Driver installation
  - Link state configuration – Infiniband/Ethernet

  - For SR-IOV based solution:
    - FW configuration
      - SR-IOV enable
      - Setting Number of VF's
    - VFs activation
    - GUID and MAC address configuration



Thank You

Singularity, SLURM, and Kubernetes - 5 min

Michael Bauer - HPC Container Workshop ISC19

20 June 2019

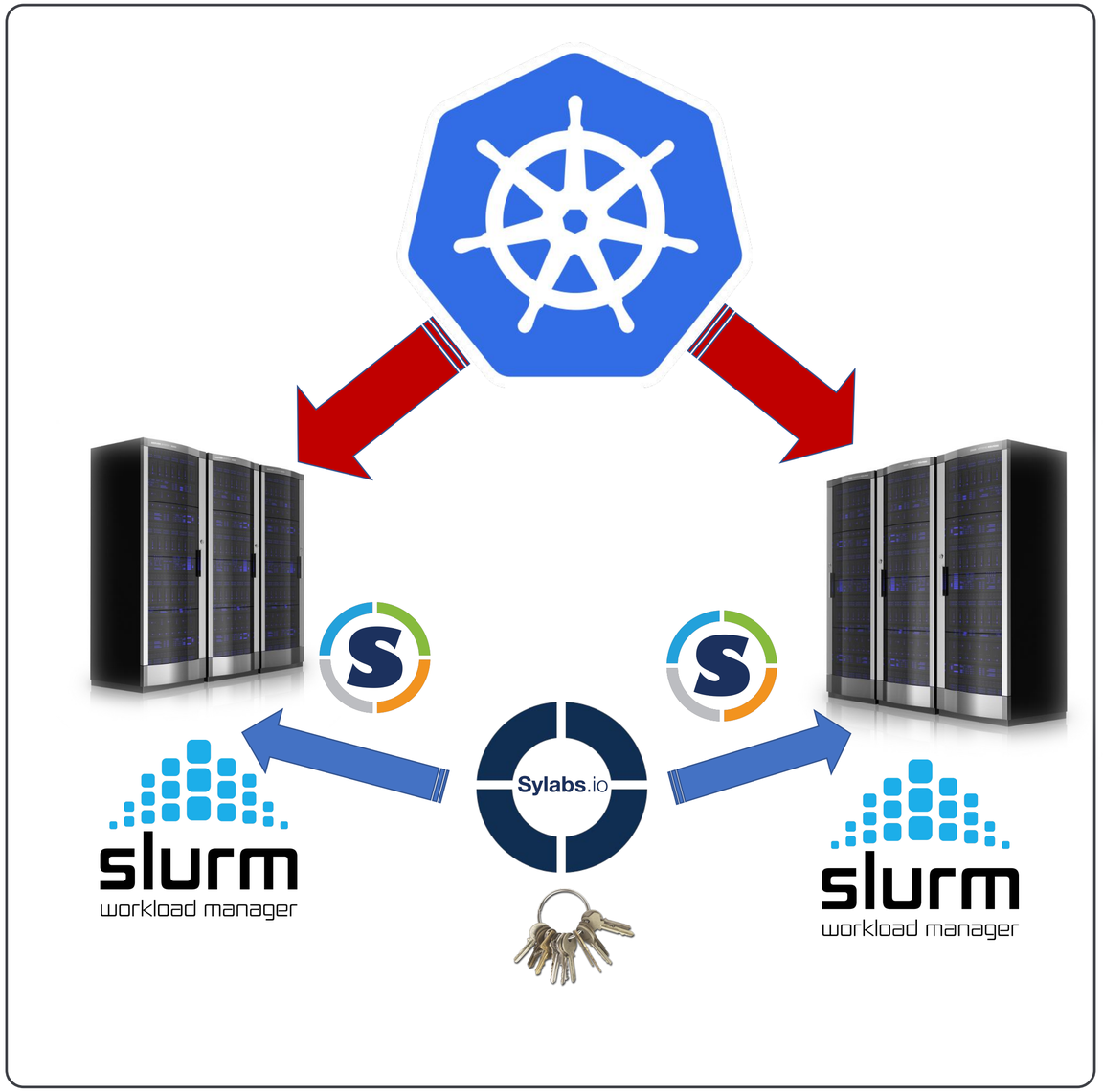
Running Singularity with K8s

- Singularity-CRI now in **v1.0.0-alpha.3** release
  - *Please try and provide feedback!*
- Run all workloads (K8s microservices, HPC, EPC, etc…) with one unifying runtime
- Native support for **GPUs**, **Infiniband/RDMA** *(coming soon)*, other HPC-oriented hardware via **Container Device Plugin**

Multi-Cluster Scheduling

Kubernetes - scheduling to multiple HPC clusters - using Singularity containers to distribute the workloads.

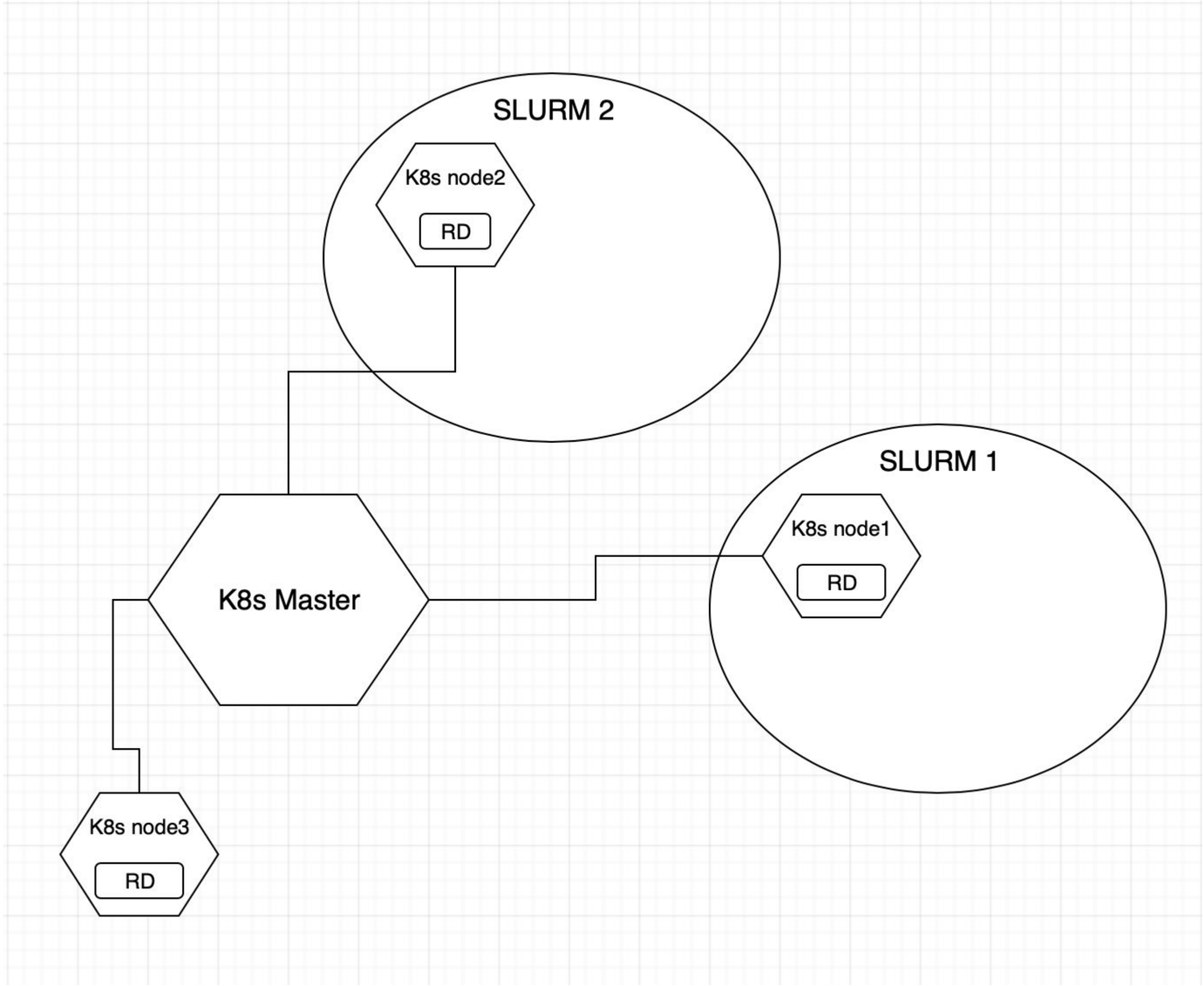
All as unprivileged user.



Multi-Cluster Scheduling to SLURM via K8s

- Submit SLURM job request to Kubernetes server using new job kind “SlurmJob”
- Custom K8s Resource Daemon lives on each SLURM master node, lets K8s understand what resources exist
- Use default or custom K8s scheduling algorithm to dispatch jobs to different clusters

Singularity, SLURM, K8s



K8s SlurmJob API

```
apiVersion: slurm.sylabs.io/v1alpha1
kind: SlurmJob
metadata:
  name: cow
spec:
  batch: |
    #!/bin/sh
    ##SBATCH --nodes=1 --cpus-per-task=1
    srun singularity pull library://sylabsed/examples/lolcow
    srun singularity run lolcow_latest.sif
    srun rm lolcow_latest.sif
  nodeSelector:
    containers: singularity
```

PORTABLE CONTAINERS ORCHESTRATION AT SCALE WITH NEXTFLOW

Paolo Di Tommaso, Seqera Labs

ISC-HPC 2019 - Frankfurt

nextflow

seqeralabs

ENABLING TECHNOLOGY

nextflow

code

orchestration

dependencies

deployment

sharing & reproducibility

WHAT DO YOU MEAN?

```
# satellite sequences reported by RepeatMasker.
zcat rmsk.txt.gz \
  | grep Satellite \
| cut -f6,7,8 \
| sed s,^chr,, \
| perl -pe 's/^[^\s_]+_([^\s_]+)_random/$1.1/' \
| tr "gl" "GL" \
| sort -k1,1N -k2,2n \
| bgzip > hs37d5.satellite.bed.gz
```

Credits of *Heng Li*, https://goo.gl/2nF5NC

THE NEXTFLOW WAY

```
process filtering {
    input: file 'rmsk.txt.gz' from sequences_ch
    output: file 'hs37d5.satellite.bed.gz' into results_ch

    '''
    # satellite sequences reported by RepeatMasker.
    zcat rmsk.txt.gz \
        | grep Satellite \
        | cut -f6,7,8 \
        | sed s,^chr,, \
        | perl -pe 's/^[^\s_]+_([^\s_]+)_random/$1.1/' \
        | tr "gl" "GL" \
        | sort -k1,1N -k2,2n \
        | bgzip > hs37d5.satellite.bed.gz
    '''
}


Channel.fromPath('data/rmsk.txt.gz') | filtering | publishTo { '/path' }
```

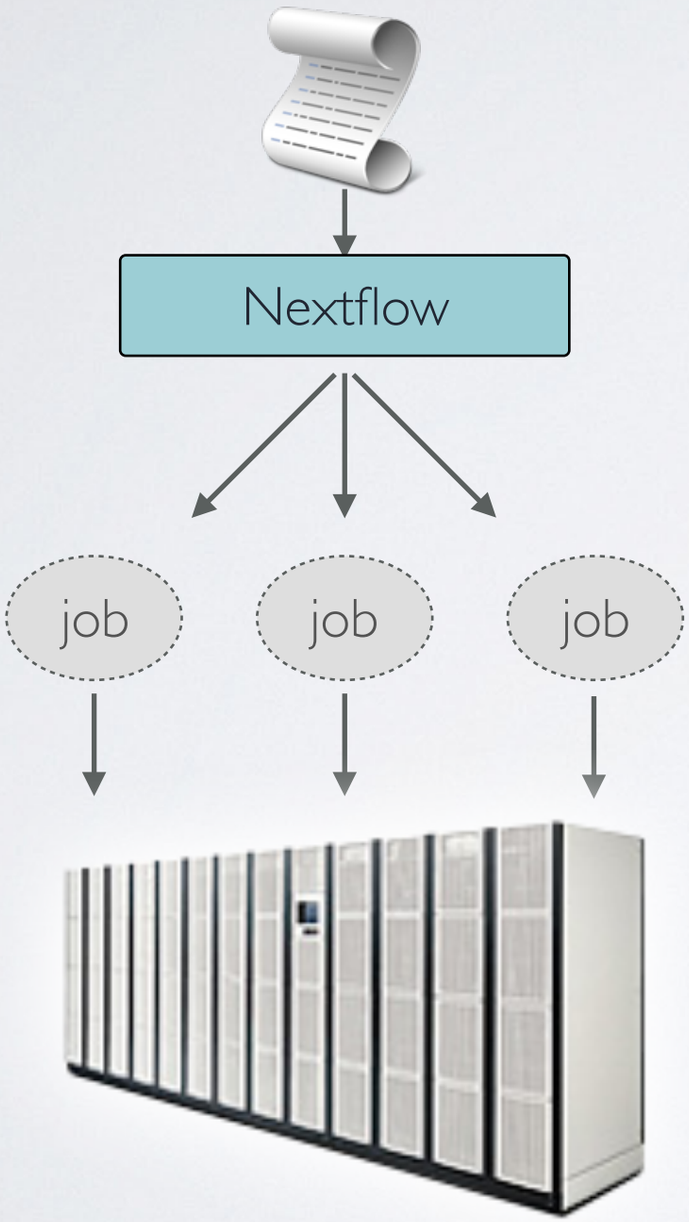
THE NEXTFLOW WAY

```
process filtering {
    input: file 'rmsk.txt.gz' from sequences_ch
    output: file 'hs37d5.satellite.bed.gz' into results_ch

    '''
    # satellite sequences reported by RepeatMasker.
    zcat rmsk.txt.gz \
        | grep Satellite \
        | cut -f6,7,8 \
        | sed s,^chr,, \
        | perl -pe 's/^[^\s_]+_([^\s_]+)_random/$1.1/' \
        | tr "gl" "GL" \
        | sort -k1,1N -k2,2n \
        | bgzip > hs37d5.satellite.bed.gz
    '''
}


Channel.fromPath('data/*.txt.fq') | filtering | publishTo { '/path' }
```

CONTAINERISATION

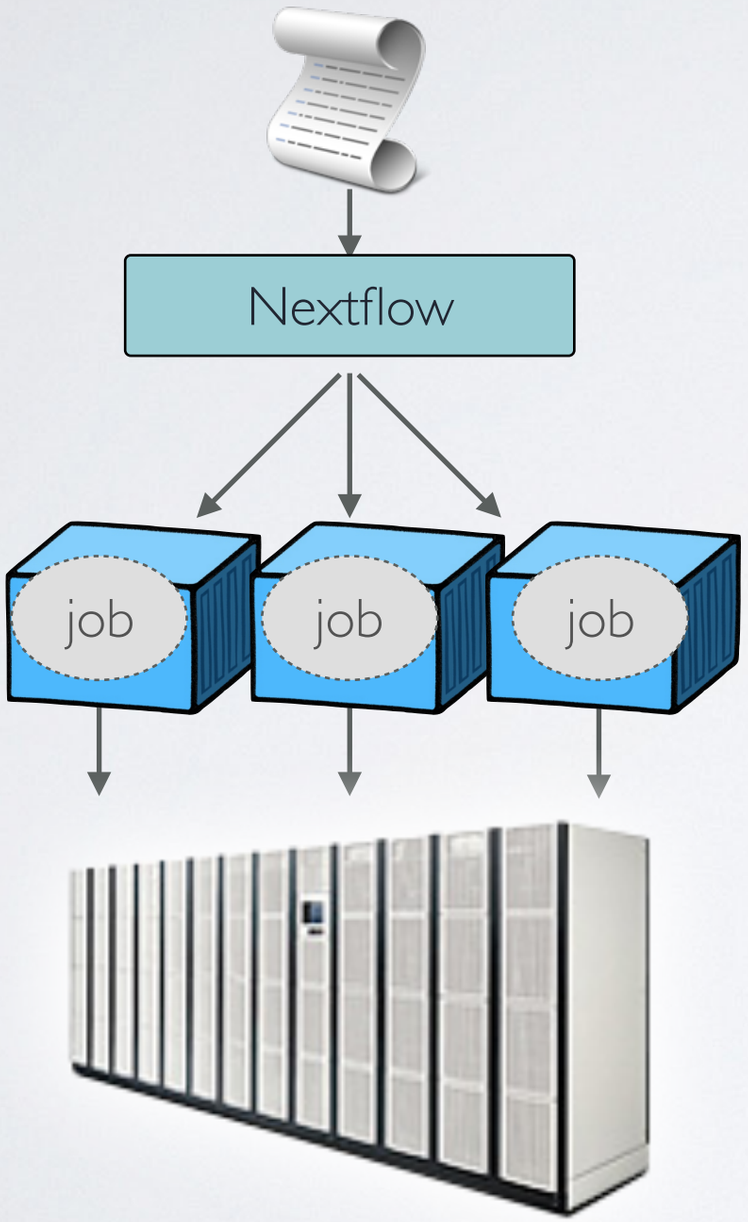


- Nextflow envisioned the use of software containers to fix computational reproducibility

- Mar 2014 (ver 0.7), support for Docker

- Dec 2016 (ver 0.23), support for Singularity

CONTAINERISATION



- Nextflow envisioned the use of software containers to fix computational reproducibility
- Mar 2014 (ver 0.7), support for Docker
- Dec 2016 (ver 0.23), support for Singularity

PORTABILITY


```
nextflow run your-script.nf -with-docker your/image
```

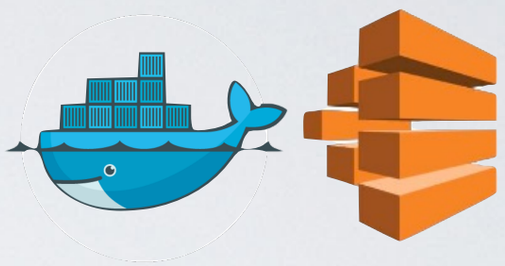
PORTABILITY

nextflow

```
process {
 executor = 'slurm'
 queue = 'my-queue'
 memory = '8 GB'
 cpus = 4
 container = 'user/image'
}
```

slurm
workload manager

PORTABILITY



nextflow

```
process {
 executor = 'awsbatch'
 queue = 'my-queue'
 memory = '8 GB'
 cpus = 4
 container = 'user/image'
}
```

WHO IS USING NEXTFLOW?

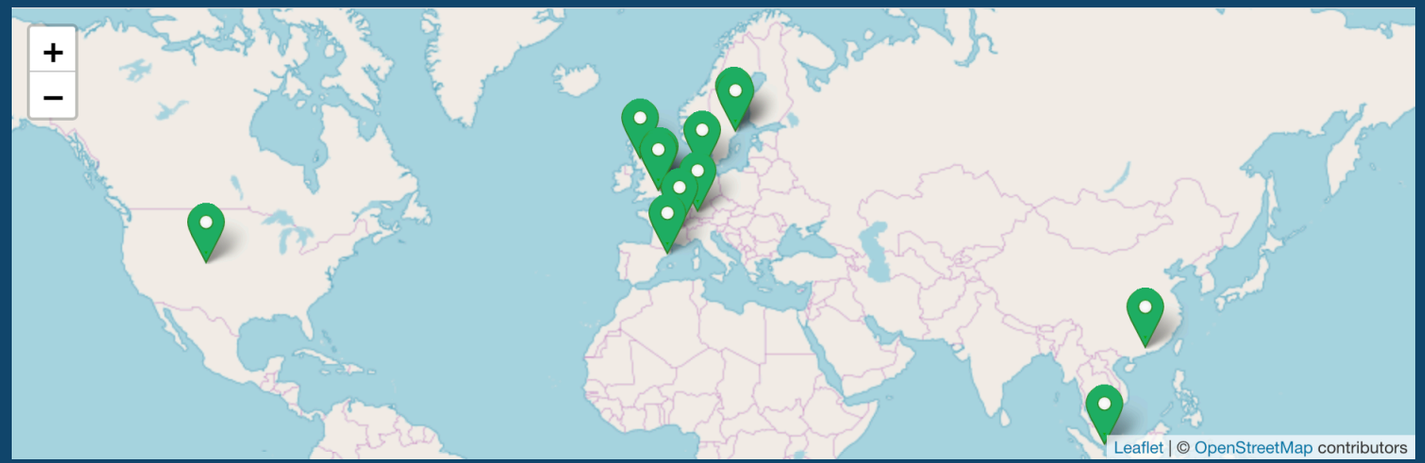
EMBL-EBI
Roche
THE FRANCIS CRICK INSTITUTE
FRED HUTCH CURES START HERE™
SciLifeLab
READCOOR
University of Colorado Boulder
wellcome sanger institute
International Agency Research on Cancer
World Health Organization
Institut Pasteur
Curtin University
Boehringer Ingelheim
illumina®
JGI JOINT GENOME INSTITUTE DEPARTMENT OF ENERGY
UT Southwestern Medical Center
NATIONAL MARROW DONOR PROGRAM®
Universität Bielefeld
UCSF
HARVARD UNIVERSITY
mi genomics medicine ireland
Colorado State University
gritstone ONCOLOGY
bina
BENAROYA RESEARCH INSTITUTE Virginia Mason
QBiC Quantitative Biology Center
ACHILLES THERAPEUTICS
Karolinska Institutet
Weill Cornell Medical College
enterome bioscience
PAWSEY supercomputing centre
BERKELEY LAB
FABRIC GENOMICS™
VIR
Agency for Science, Technology and Research SINGAPORE
SYNTHETIC GENOMICS

nf-core

20
pipelines

38
members

12+
institutions



CANCER RESEARCH UK | BEATSON INSTITUTE

NATIONAL GENOMICS INFRASTRUCTURE

SciLifeLab

THE FRANCIS CRICK INSTITUTE

QBiC

A*STAR Genome Institute of Singapore

SLU

SYSUCC SINCE 1964

中山大學 肿瘤防治中心 SUN YAT-SEN UNIVERSITY CANCER CENTER

International Agency for Research on Cancer
World Health Organization

University of Colorado Boulder

wellcome sanger institute

CRG Centre for Genomic Regulation

THANK YOU

nextflow

http://nextflow.io

seqeralabs

http://seqera.io

AWS Batch

aws

Introducing AWS Batch

Fully Managed

No software to install or servers to manage. AWS Batch provisions, manages, and scales your infrastructure

Integrated with AWS

Natively integrated with the AWS Platform, AWS Batch jobs can easily and securely interact with services such as Amazon S3 and DynamoDB

Cost-optimized Resource Provisioning

AWS Batch automatically provisions compute resources tailored to the needs of your jobs using Amazon EC2 and EC2 Spot

AWS Batch Concepts

- Job Queue
- Compute Environments
- Job Definitions
- Jobs
  - Single jobs vs Array jobs vs Multi-node Parallel jobs
- Scheduler

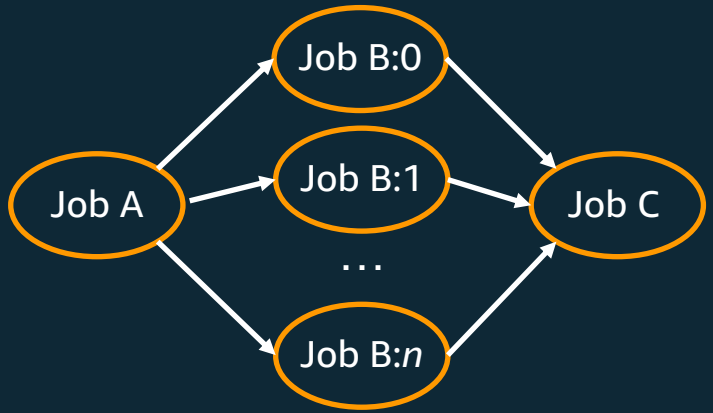
aws

Easily run massively parallel jobs

Instead of submitting a large number of independent "**simple jobs**", we also support "**array jobs**" that run many copies of an application against an array of elements.

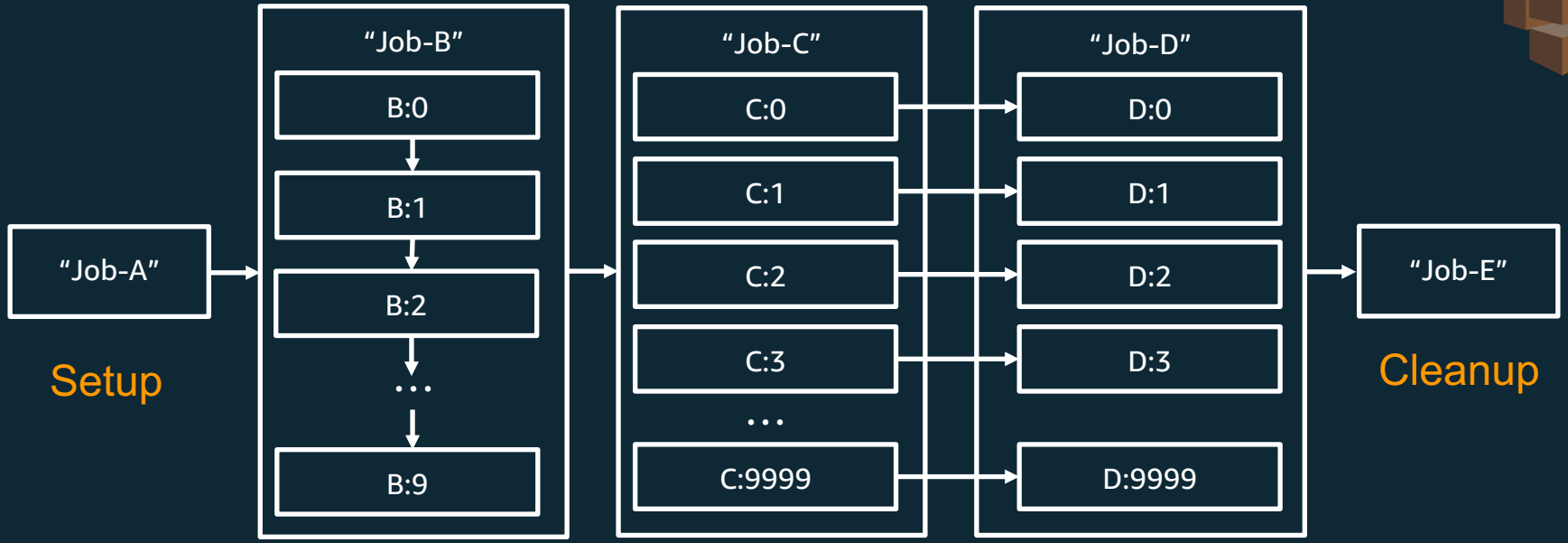
Array jobs are an efficient way to run:

- Parametric sweeps
- Monte Carlo simulations
- Processing a large collection of objects



aws

Array Job Dependency Models

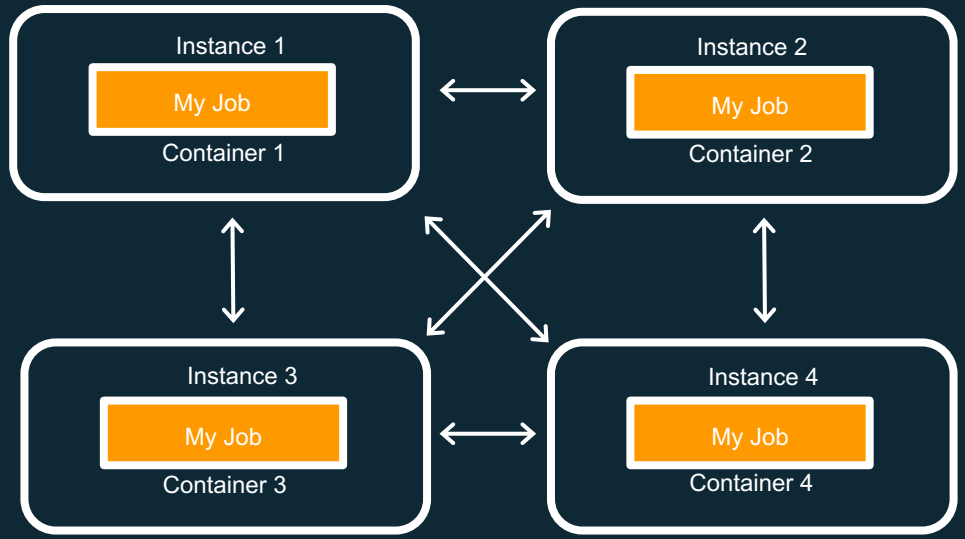



```
$ aws batch submit-job –depends-on 606b3ad1-aa31-48d8-92ec-f154bfc8215f ...
```

Multi-node Parallel Jobs on AWS Batch

- Scale jobs across multiple instances with AWS Batch support for Multi-node Parallel (MNP) jobs

Use AWS Batch to efficiently run larger-scale tightly coupled High Performance Computing (HPC) applications and distributed GPU model training without the need to launch, configure, and manage EC2 resources directly



aws

Typical AWS Batch Job Architecture

Input Files

S3 Events Trigger Lambda Function or CloudWatch Event to Submit Batch Job

Job Definition

IAM Role for Batch Job

Application Image

Job Resource Requirements and other parameters

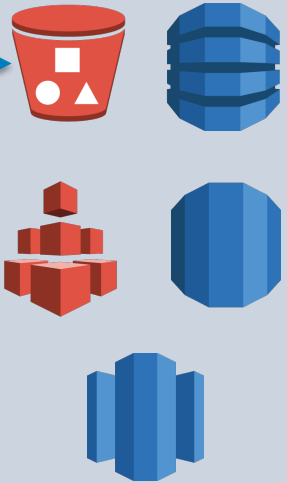
AWS Batch Execution

Queue of Runnable Jobs

AWS Batch Compute Environments

AWS Batch Scheduler

AWS Batch Job Output



aws