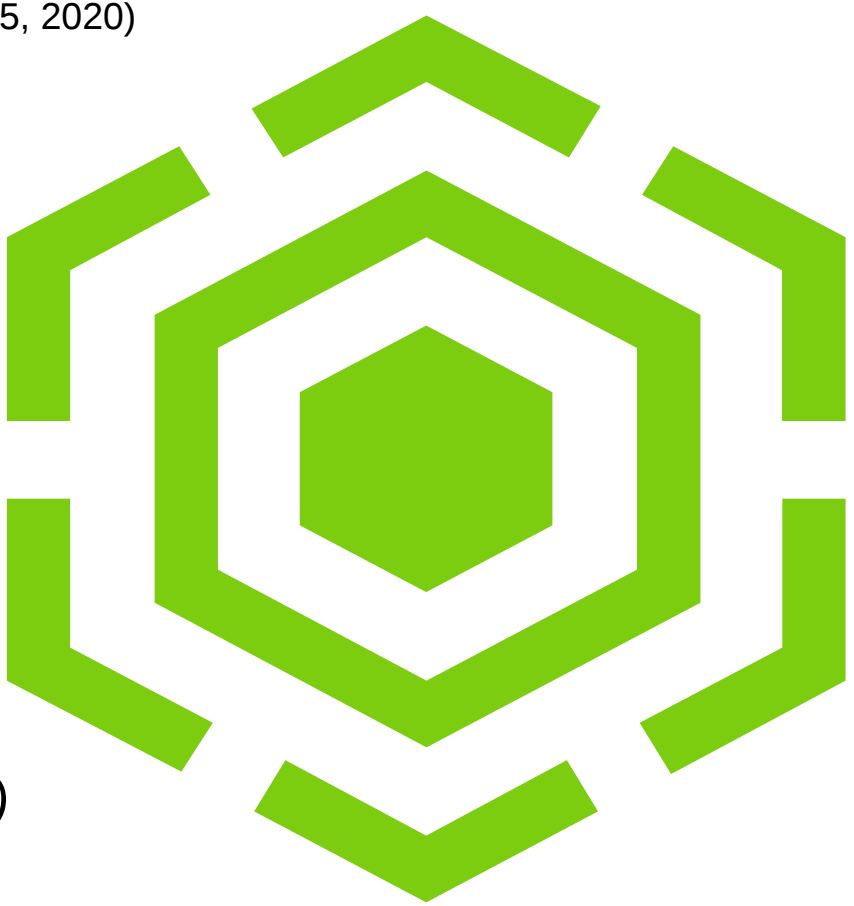
IBM Spectrum Scale Strategy Days, Ehningen, Germany (March 5, 2020)

IBM Spectrum Scale:

How to enable AI Workloads with OpenShift and IBM Spectrum Scale



Presentation:
Gero Schmidt (gerosch@de.ibm.com)
Live Demo:
Przemyslaw Podfigurny (ppod@de.ibm.com)

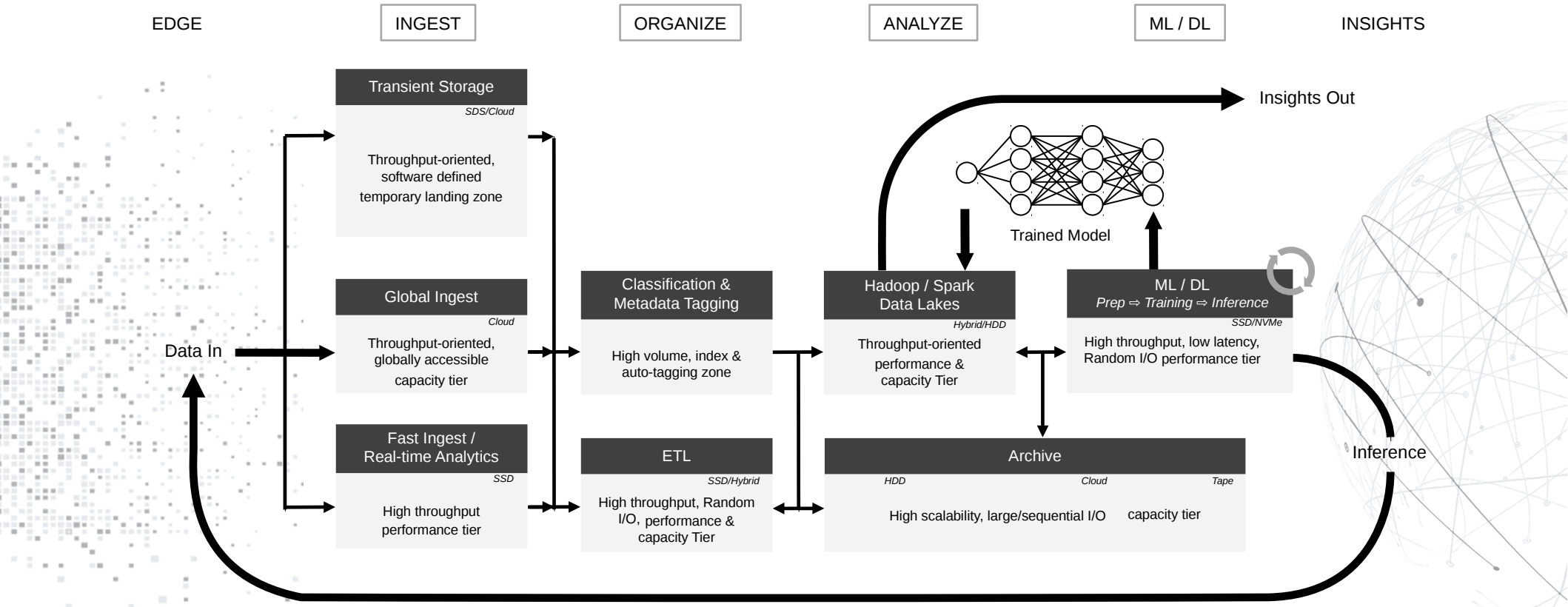
Big Data & Analytics, Germany, IBM Spectrum Scale Development

IBM

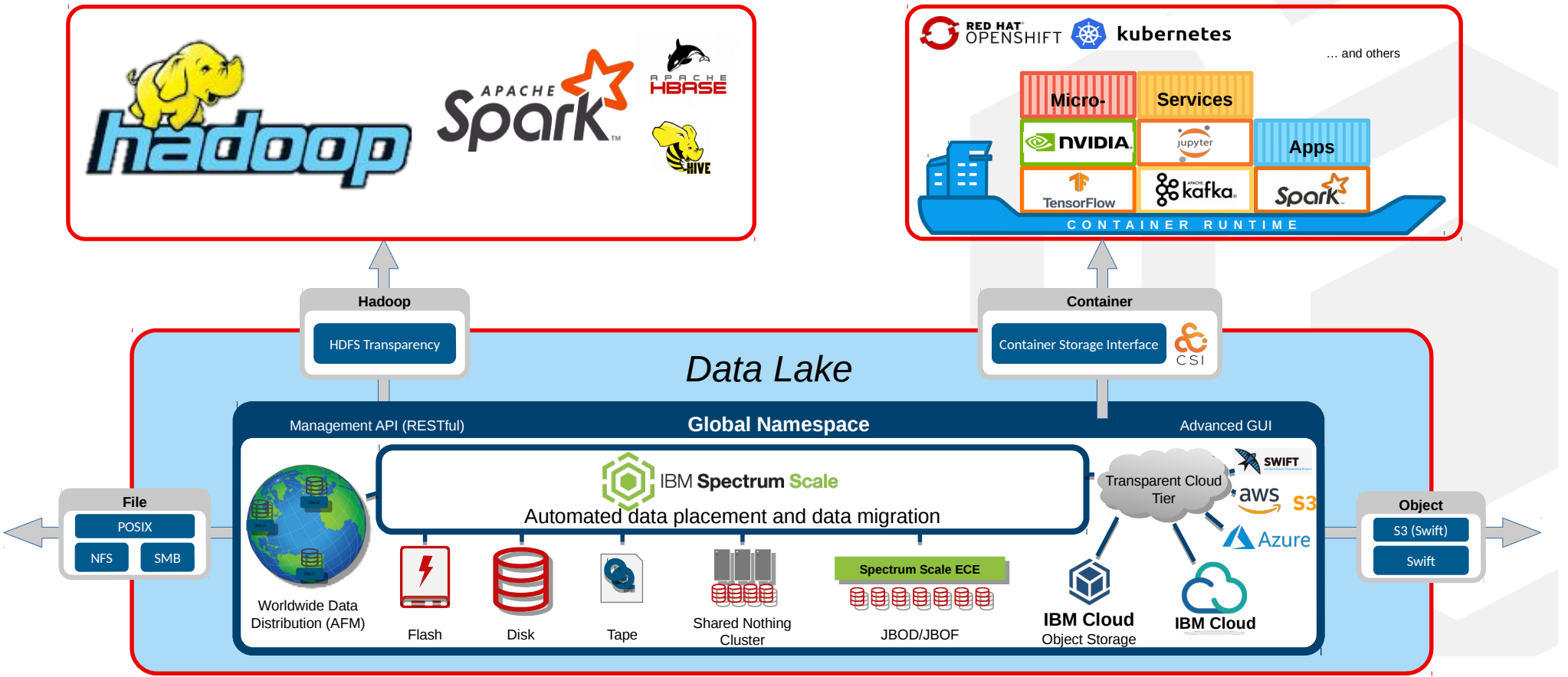
Agenda

- ***(1) Presentation***
  - ***AI Data Pipeline and Workloads at Scale***
  - ***OpenShift 4.x Architecture***
  - ***IBM Spectrum Scale CSI Driver***
  - ***Persistent Storage Concepts in Kubernetes/OpenShift (PV, PVC)***
  - ***Use Cases for Storage Provisioning with IBM Spectrum Scale CSI for AI Workloads***
- ***(2) Live Demo:***
  - ***AI Use Case Example with IBM Spectrum Scale CSI and TensorFlow in OpenShift 4.x***
- **Questions**

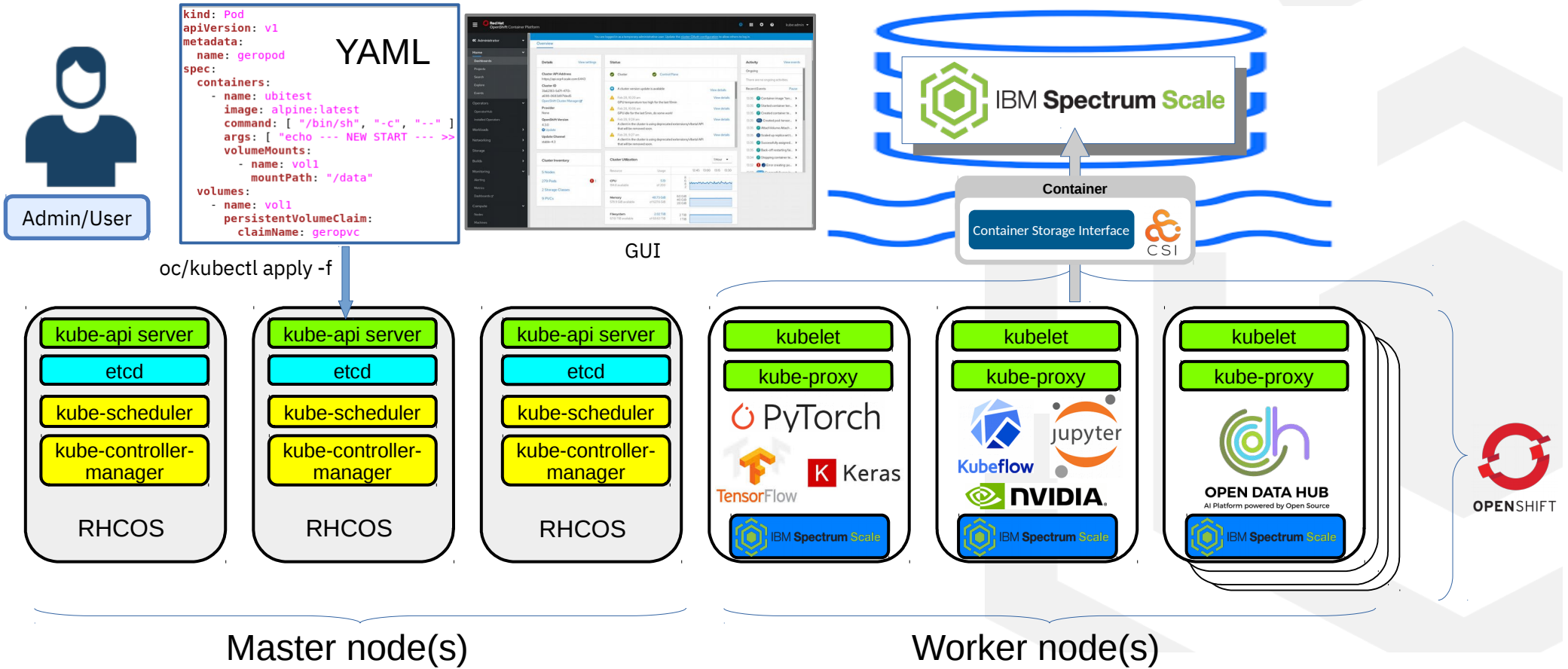
Enterprise AI Data Pipeline



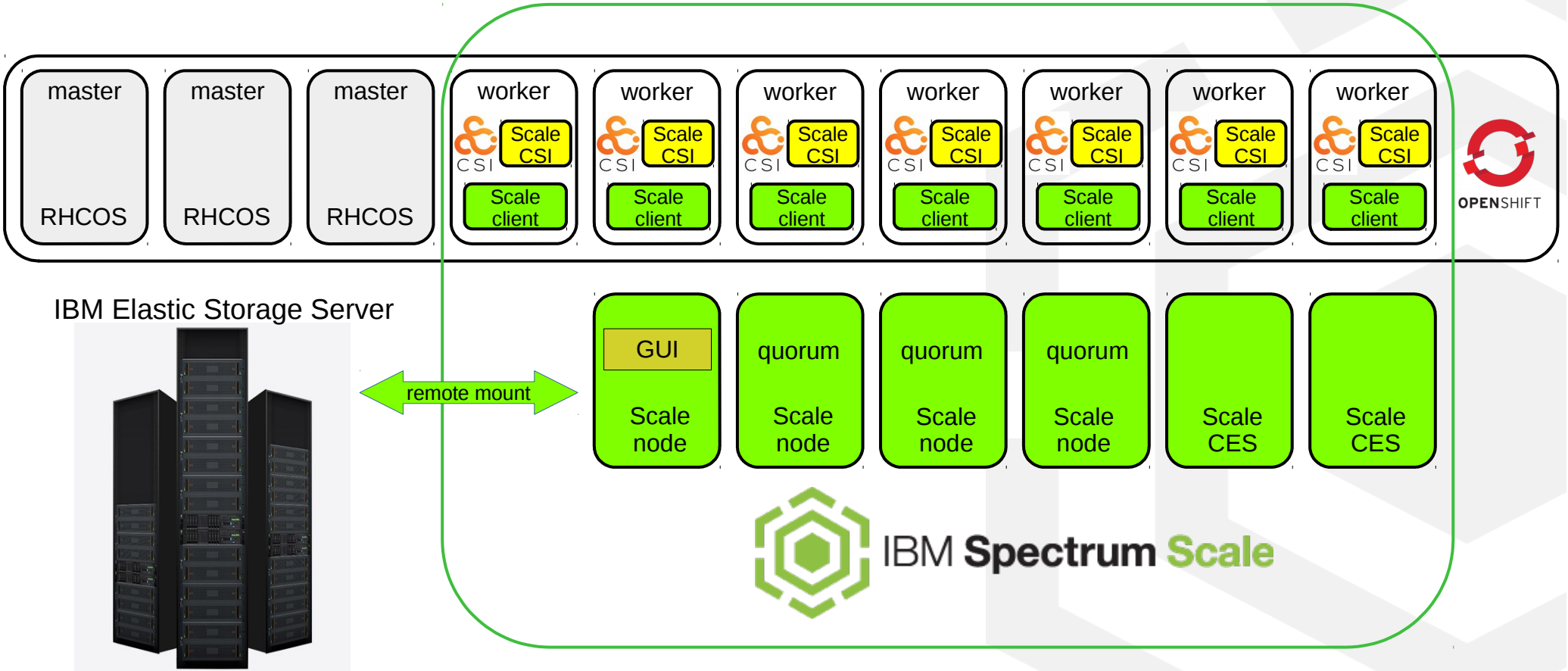
AI Workloads at Scale



OpenShift 4.x Architecture



OpenShift 4.x and IBM Spectrum Scale 5.0.4.1 Integration



IBM Spectrum Scale CSI Driver – Preparation

Create a IBM Spectrum Scale user group "CsiAdmin" if one does not already exist

```
# /usr/lpp/mmfs/gui/cli/mkusergrp CsiAdmin --role csiadmin
```

Create an IBM Spectrum Scale user in the "CsiAdmin" group

```
# /usr/lpp/mmfs/gui/cli/mkuser <username> -p <password> -g CsiAdmin
```

Initialize GUI

```
# /usr/lpp/mmfs/gui/cli/initgui
```

Test GUI access from Kubernetes nodes

```
# curl --insecure -u '<username>:<password>' \
       -X GET https://[GUI-SERVER]:443/scalemgmt/v2/cluster
```

Verify that quota is enabled on Spectrum Scale file systems used for fileset-based dynamic provisioning

`# mmlsfs fs0 -Q` **Otherwise enable quota with** `# mmchfs fs0 -Q yes`

Enforce quota for root

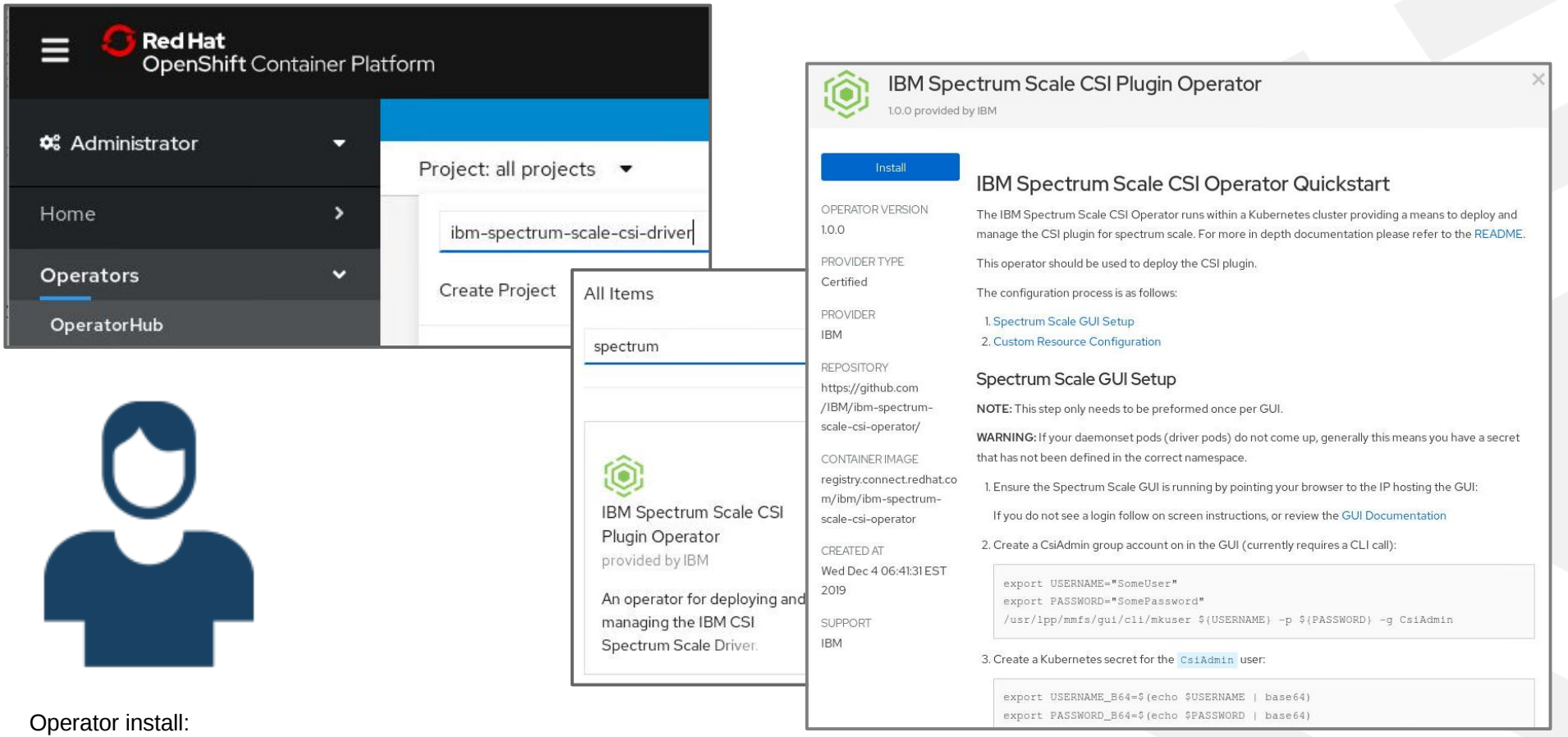
```
#  mmchconfig enforceFilesetQuotaOnRoot=yes -i
```

Install IBM Spectrum Scale client on Kubernetes *worker nodes* & add to IBM Spectrum Scale cluster

Red Hat OpenShift: `# mmchconfig controlSetxattrImmutableSELinux=yes -i`

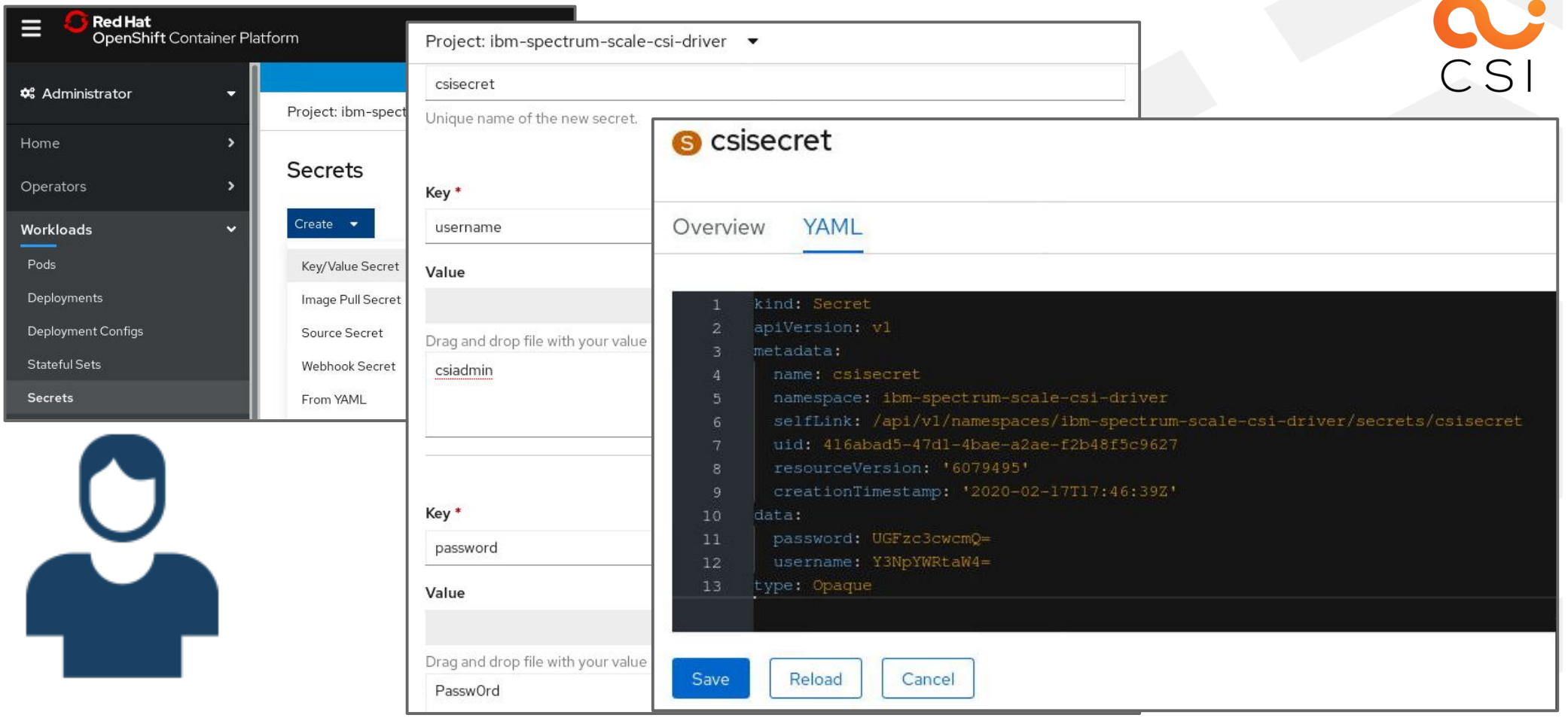
Spectrum Scale pre-install tasks: https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_instal_prereq.html

IBM Spectrum Scale CSI Driver Installation in OpenShift (1/3)

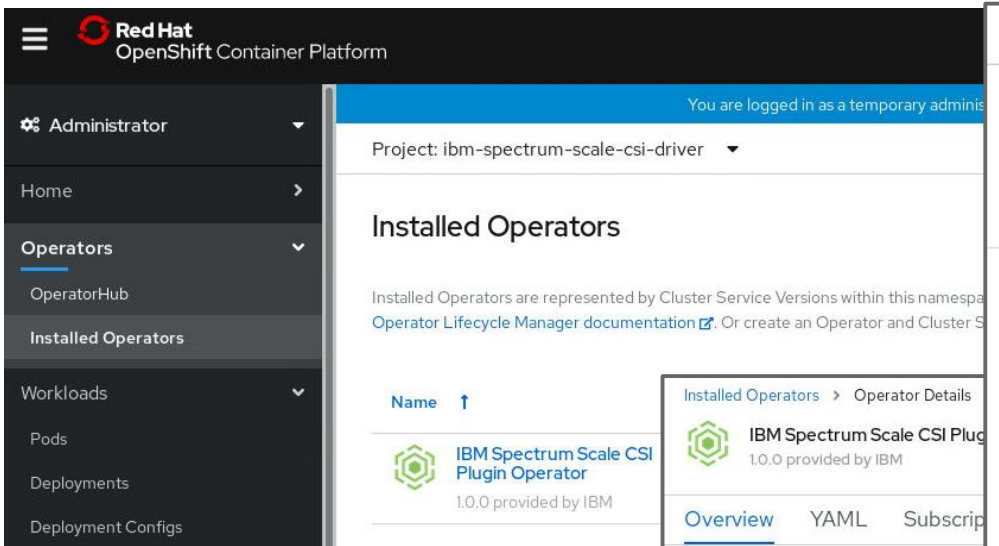


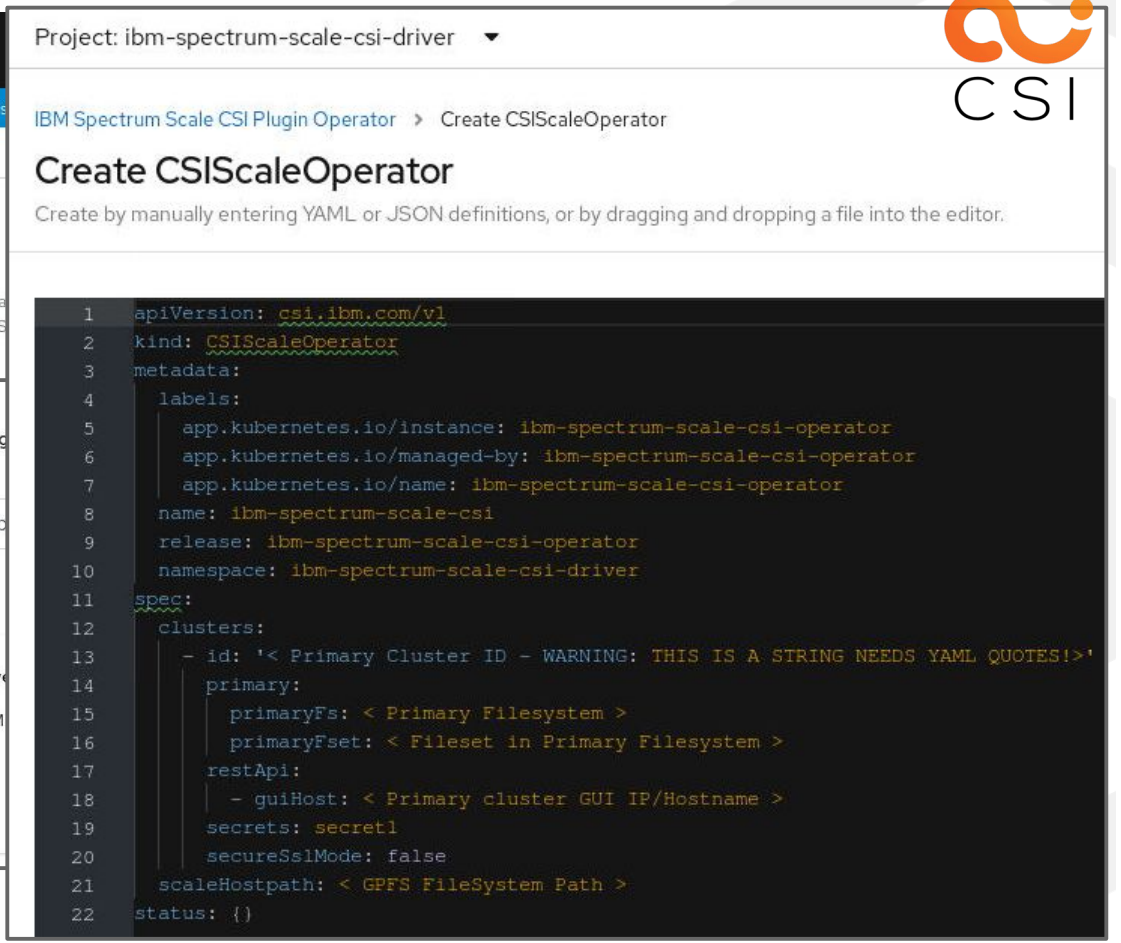
Operator install:
- https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_install_usingOLM.html

IBM Spectrum Scale CSI Driver Installation in OpenShift (2/3)



IBM Spectrum Scale CSI Driver Installation in OpenShift (3/3)





IBM Spectrum Scale CSI Driver Installation (manual steps/K8s)

(1) Create the ibm-spectrum-scale-csi-driver namespace and secret

```
# kubectl create namespace ibm-spectrum-scale-csi-driver
# kubectl create secret generic csisecret --from-literal=username=csiadmin  \
                                          --from-literal=password='Passw0rd' -n ibm-spectrum-scale-csi-driver
```

(2) Start the operator from an auto-generated YAML file

```
# curl -O https://raw.githubusercontent.com/IBM/ibm-spectrum-scale-csi-operator/v1.0.0/generated/installer/ibm-spectrum-scale-csi-operator.yaml
# kubectl apply -f ibm-spectrum-scale-csi-operator.yaml
```

(3) Configure and apply the custom resource file (CR) to match the Spectrum Scale install

```
# curl -O https://raw.githubusercontent.com/IBM/ibm-spectrum-scale-csi-operator/v1.0.0/stable/ibm-spectrum-scale-csi-operator-bundle/operators/ibm-spectrum-scale-csi-operator/deploy/crds/ibm-spectrum-scale-csi-operator-cr.yaml
# vi ibm-spectrum-scale-csi-operator-cr.yaml
>   scaleHostpath: "/gpfs/fs0"
>     - id: "6174907451206751632"
>       secrets: "csisecret"
>         primaryFs: "fs0"
>         primaryFset: "csi-fset"
>       - guiHost: "vm08.bda.scale.com"

# kubectl apply -f ibm-spectrum-scale-csi-operator-cr.yaml
```

Manual install:
- https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.4/com.ibm.spectrum.scale.csi.v5r04.doc/bl1csi_install_usingops.html
- https://ibm-spectrum-scale-csi-operator.readthedocs.io/en/latest/deployment/manual.html

IBM Spectrum Scale CSI Driver – Components

```
[root@vm01 csi]# kubectl get all -n ibm-spectrum-scale-csi-driver -o wide
NAME                                                  READY   STATUS    RESTARTS   AGE    NODE
pod/ibm-spectrum-scale-csi-7xxzx                      2/2     Running   0          12m    vm07.bda.scale.com
pod/ibm-spectrum-scale-csi-attacher-0                 1/1     Running   0          12m    vm05.bda.scale.com
pod/ibm-spectrum-scale-csi-bv9nn                      2/2     Running   0          12m    vm05.bda.scale.com
pod/ibm-spectrum-scale-csi-operator-5f8846dfbf-zhqct  2/2     Running   0          26m    vm06.bda.scale.com
pod/ibm-spectrum-scale-csi-provisioner-0              1/1     Running   0          12m    vm07.bda.scale.com
pod/ibm-spectrum-scale-csi-xsvhg                      2/2     Running   0          12m    vm06.bda.scale.com

NAME                                              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/ibm-spectrum-scale-csi-operator-metrics   ClusterIP   10.233.44.169   <none>        8383/TCP   25m

NAME                                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/ibm-spectrum-scale-csi   3         3         3       3            3           <none>          12m

NAME                                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ibm-spectrum-scale-csi-operator   1/1     1            1           26m

NAME                                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/ibm-spectrum-scale-csi-operator-5f8846dfbf  1         1         1       26m

NAME                                                READY   AGE
statefulset.apps/ibm-spectrum-scale-csi-attacher    1/1     12m
statefulset.apps/ibm-spectrum-scale-csi-provisioner 1/1     12m
```

IBM Spectrum Scale CSI driver video blogs: https://developer.ibm.com/storage/2019/12/26/ibm-spectrum-scale-csi-driver-video-blogs/

Kubernetes Storage Concepts – Persistent Volumes (PVs)

CSI

- **Static provisioning**

Admin provisions static **PVs** in OpenShift/Kubernetes

User claims volume from pool of pre-provisioned **PVs** through **Persistent Volume Claim (PVC)**

- **Dynamic provisioning**

Admin creates **StorageClass** in OpenShift/Kubernetes

User claims volume from **StorageClass** through **Persistent Volume Claim (PVC)** (*self-service provisioning*)

Kubernetes: Persistent Volumes – *Static Provisioning*

```
$ kubectl apply -f scale-dev_pv.yaml
$ cat scale-dev_pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: scale-data-pv01
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  csi:
    driver: ibm-spectrum-scal
    volumeHandle: "volumeHand
    "clusterID;FSUID;path=/gp
```

```
$ cat myapp-pvc01.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myapp-pvc01
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
```

```
$ cat myapp.yaml
kind: Pod
apiVersion: v1
metadata:
  name: myapp
spec:
  containers:
    - name: demo
      image: alpine:3.8
      command: [ "/bin/sh", "-c", "--" ]
      args: [ while true; …; done;" ]
      volumeMounts:
        - name: vol01
          mountPath: "/data"
  volumes:
    - name: vol01
      persistentVolumeClaim:
        claimName: myapp-pvc01
```

This only works in this simple fashion if **no *default* StorageClass** is defined!

created

vol

bound

vol

PV (Persistent Volume)

Kubernetes: Persistent Volumes – *Dynamic Provisioning*

```
$ kubectl apply -f sc_scale.yaml
$ cat sc_scale.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: scale-dev
provisioner: spectrumscale.
parameters:
    volBackendFs: "fs0"
    clusterId: "61749074512
reclaimPolicy: Delete
```

```
$ cat myapp-pvc01.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myapp-pvc01
spec:
  storageClassName: "scale-dev"
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
```

created ↕ bound

PV (Persistent Volume)

```
$ cat myapp.yaml
kind: Pod
apiVersion: v1
metadata:
  name: myapp
spec:
  containers:
    - name: demo
      image: alpi
      command: [
      args: [ while true; …; done;" ]
      volumeMounts:
        - name: vol01
          mountPath: "/data"
  volumes:
    - name: vol01
      persistentVolumeClaim:
        claimName: myapp-pvc01
```

- PV is *auto-created* upon user request.
- With a ***default*** **StorageClass** defined ***storageClassName*** can be omitted.
- **Labels** & **Selector** cannot be used!

Understand the *default* StorageClass

Define a ***default*** StorageClass:

```
# kubectl patch storageclass scale-ifset -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"true"}}}'
storageclass.storage.k8s.io/scale-ifset patched

# kubectl get sc
NAME                    PROVISIONER                    AGE
local-path              rancher.io/local-path          35d
local-storage           kubernetes.io/no-provisioner   35d
scale-dfset             spectrumscale.csi.ibm.com      3d22h
scale-ifset (default)   spectrumscale.csi.ibm.com      3d22h
```

With **no *default*** StorageClass defined:

- **PVC** with ***no* or *empty* storageClassName** is mapped to pool of ***statically provisioned*** volumes
- **PVC** with ***non-empty* storageClassName** is ***dynamically provisioned***

With a ***default*** StorageClass defined:

- **PVC** with ***no* storageClassName** is **dynamically provisioned** from ***default*** StorageClass
- **PVC** with ***empty* storageClassName (“”)** is mapped to pool of ***statically provisioned*** volumes
- PVC with ***non-empty* storageClassName** is ***dynamically provisioned***

Kubernetes: Persistent Volumes – *Static Provisioning (refined)*

```
$ kubectl apply -f scale-data-pv01.yaml
$ cat scale-data-pv01.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: scale-data-pv01
  labels:
    type: data
    dept: datascience
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  csi:
    driver: ibm-spectrum-scal
    volumeHandle: "volumeHand
    "clusterID;FSUID;path=/gp
```

```
$ cat myapp-pvc01.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myapp-pvc01
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  selector:
    matchLabels:
      type: data
      dept: datascience
```

```
$ cat myapp.yaml
kind: Pod
apiVersion: v1
metadata:
  name: myapp
spec:
  containers:
    - name: demo
      image: alpine:3.8
      command: [ "/bin/sh", "-c", "--" ]
      args: [ while true; …; done;" ]
      volumeMounts:
        - name: vol01
          mountPath: "/data"
  volumes:
    - name: vol01
      persistentVolumeClaim:
        claimName: myapp-pvc01
```

- This works even if a ***default* StorageClass** is defined!
- **Labels** & **Selector** allow to request *specific* PVs from pool.

created

bound

PV (Persistent Volume)

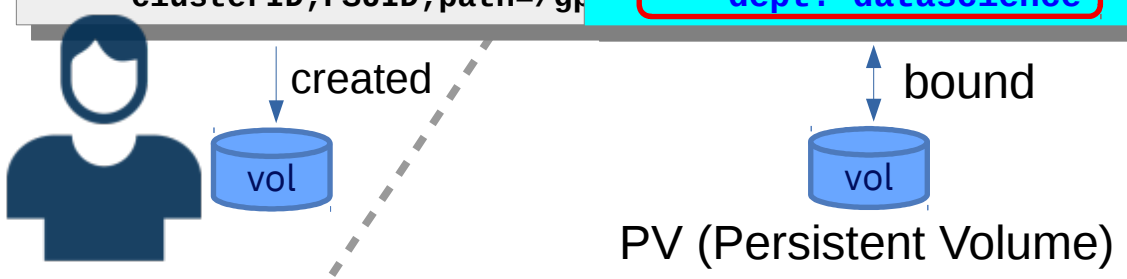
Kubernetes: Persistent Volumes – *Static Provisioning (refined 2)*

```
$ kubectl apply -f scale-data-pv01.yaml
$ cat scale-data-pv01.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: scale-data-pv01
  labels:
    type: data
    dept: datascience
spec:
  storageClassName: static
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  csi:
    driver: ibm-spectrum-scal
    volumeHandle: "volumeHand
    "clusterID;FSUID;path=/gp
```

```
$ cat myapp-pvc01.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myapp-pvc01
spec:
  storageClassName: static
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  selector:
    matchLabels:
      type: data
      dept: datascience
```

```
$ cat myapp.yaml
kind: Pod
apiVersion: v1
metadata:
  name: myapp
spec:
  containers:
    - name: demo
      image: alpine:3.8
      command: [ "/bin/sh", "-c", "--" ]
      args: [ while true; …; done;" ]
      volumeMounts:
        - name: vol01
          mountPath: "/data"
  volumes:
    - name: vol01
      persistentVolumeClaim:
        claimName: myapp-pvc01
```

- This works even if a ***default* StorageClass** is defined!
- **Labels** & **Selector** allow to request *specific* PVs from pool.



IBM Spectrum Scale CSI Driver: *StorageClass* Types

Fileset-based volumes (dependent/independent)

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: scale-silver
provisioner: spectrumscale.csi.ibm.com
parameters:
    volBackendFs: "fs1"
    clusterId: "17797813605352210071"
    uid: "1000"
    gid: "1000"
    filesetType: "dependent"|"independent"
    parentFileset: "k8s-development"
reclaimPolicy: Delete
```

Directory-based (lightweight) volumes:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: scale-bronze
provisioner: spectrumscale.csi.ibm.com
parameters:
    volBackendFs: "fs1"
    volDirBasePath: "k8s-storage"
reclaimPolicy: Delete
```

volBackendFs	IBM Spectrum Scale file system on which directory or fileset based volume is to be created In case of remotely mounted file system, this is the local file system name.
volDirBasePath	Base path under which all volumes with this storage class will be created. This path must exist.
clusterId	Cluster ID on which fileset should be created. In case of remotely mounted filesystem, this should be the remote cluster ID.
filesetType	*dependent* or *independent*; default is *independent* if not specified
parentFileset	Parent fileset name in case of *dependent* fileset (optional)
inodeLimit	inode limit for fileset (optional)
uid	uid that will be assigned to the fileset – user must exist (optional)
gid	guid that will be assigned to fileset – group must exist (optional)

```
reclaimPolicy:
 - Retain
 - Recycle
 - Deleted.
```

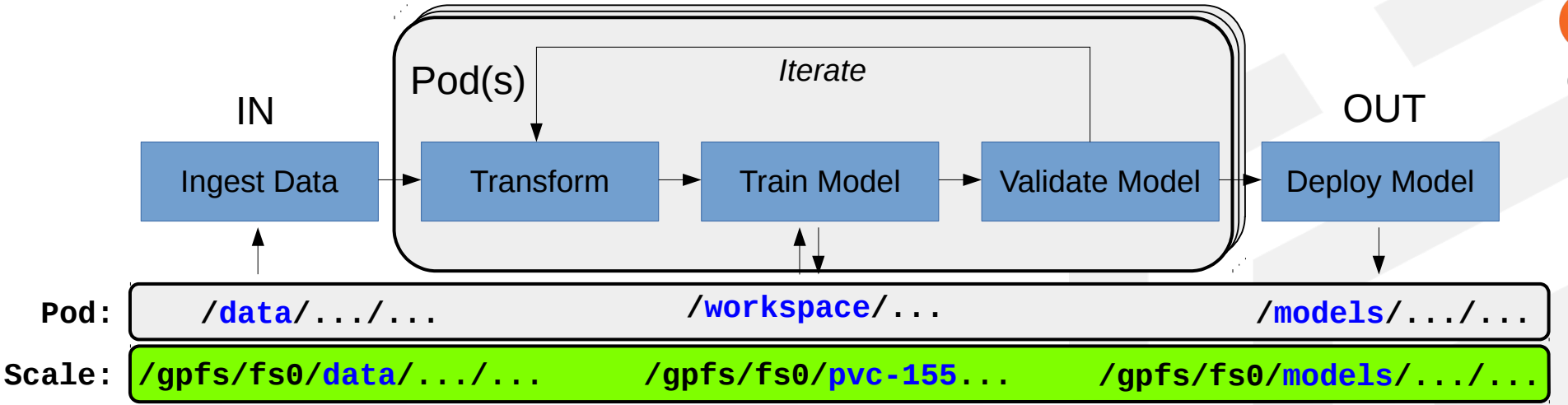
OpenShift/Kubernetes – *Storage Resource Quota*

You can limit the ***total sum of storage resources*** that can be requested in a given ***namespace***.
In addition, you can limit consumption of storage resources based on an associated ***StorageClass***.

```
# cat storage-quota.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-quota
spec:
  hard:
    requests.storage: 50Gi
    persistentvolumeclaims: 5
    scale-ifset.storageclass.storage.k8s.io/requests.storage: 10Gi
    scale-ifset.storageclass.storage.k8s.io/persistentvolumeclaims: 4
    scale-dfset.storageclass.storage.k8s.io/requests.storage: 5Gi
    scale-dfset.storageclass.storage.k8s.io/persistentvolumeclaims: 4
```

See: https://kubernetes.io/docs/concepts/policy/resource-quotas/#storage-resource-quota

AI Use Cases for Storage Provisioning with IBM Spectrum Scale



(1) **Static Provisioning** for

Shared (pre-existing) data for training and models (provisioned by *admin*):

/gpfs/fs0/data (ro) → static pv (e.g. “scale-data-pv01”)

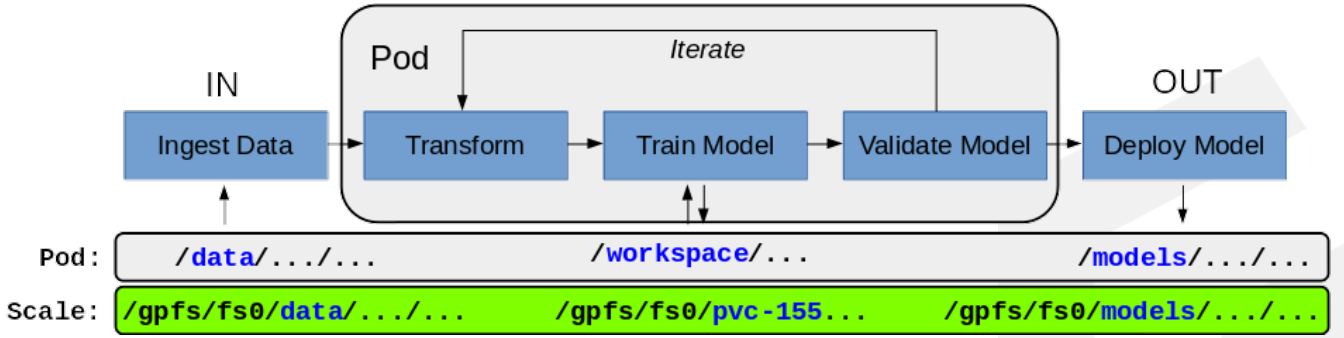
/gpfs/fs0/models (rw) → static pv (e.g. “scale-models-pv01”)

(2) **Dynamic Provisioning** for

Individual workspace in Pod for model building and validation during iterations (self-service provisioned/*user*)

/gpfs/fs0/pvc-155c0216-cc4f-4091-823f-9b626dd17a1e → StorageClass (e.g. ”scale”)

(I) Admin provides Storage for User



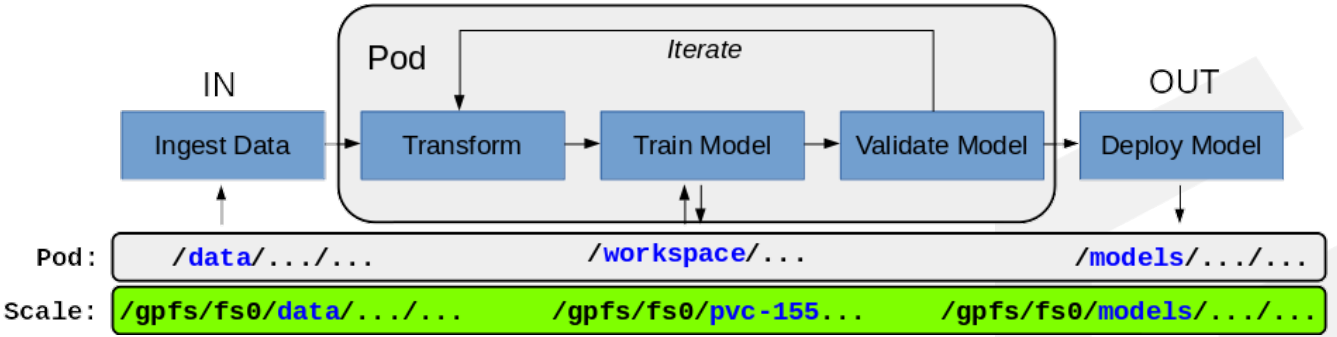

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: scale-data-pv01
  labels:
    type: data
    dept: datascience
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadOnlyMany(*)
  csi:
    driver: ibm-spectrum-
    volumeHandle: "Cluste
    readOnly: true(*)
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: scale-models-pv01
  labels:
    type: models
    dept: datascience
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  csi:
    driver: ibm-spectrum-scale-csi
    volumeHandle: "ClusterId;FSID;path=/gpfs/fs0/models"
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: scale-ifset
provisioner: spectrumscale.csi.ibm.com
parameters:
    volBackendFs: "fs0"
    clusterId: "6174907451206751632"
reclaimPolicy: Delete
```

(*) Not yet supported by IBM Spectrum Scale CSI v1.0.0

(II) User requests Storage for Namespace/Project




```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: "scale-data-pvc"
spec:
  storageClassName: ""
  accessModes:
  - ReadOnlyMany(*)
  resources:
    requests:
      storage: 100Gi
  selector:
    matchLabels:
      type: data
      dept: datascience
```

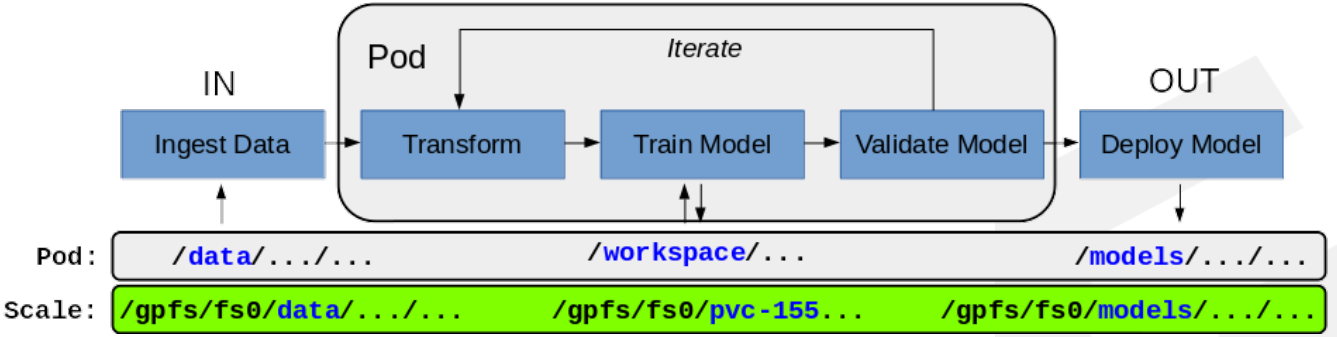
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: "scale-models-pvc"
spec:
  storageClassName: ""
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  selector:
    matchLabels:
      type: models
      dept: datascience
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: "workspace-pvc"
spec:
  storageClassName: scale-ifset
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

(*) Not yet supported by IBM Spectrum Scale CSI v1.0.0

(III) Storage for User is bound to Persistent Volume Claims (PVCs)

CSI



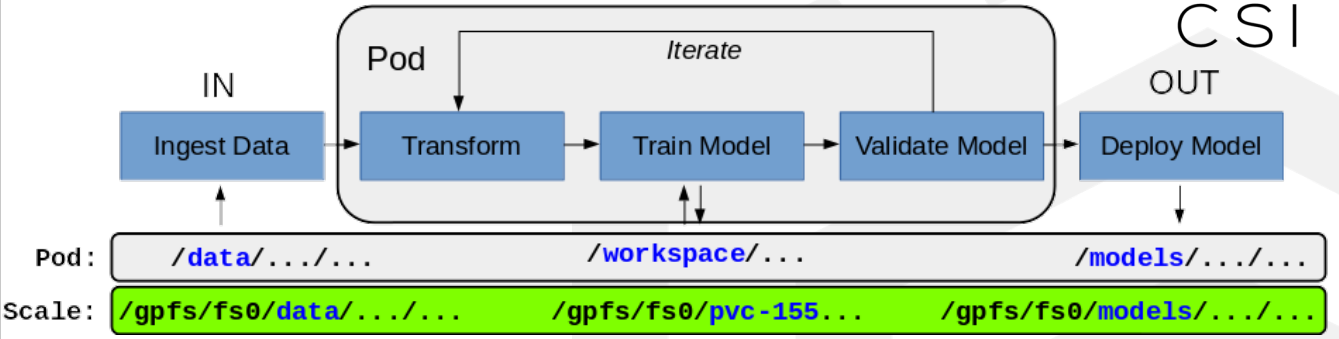

```
# kubectl get pvc
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
scale-data-pvc      Bound    scale-data-pv02                            100Gi      ROX                           33m
scale-models-pvc    Bound    scale-models-pv01                          100Gi      RWX                           28h
workspace-pvc       Bound    pvc-aaf83bc1-2c33-401d-8c3a-713be508b96f   1Gi        RWX            scale-ifset    28h

# kubectl get pv
NAME                    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                       STORAGECLASS
pvc-aaf83bc1-[]-[]-[]-[] 1Gi        RWX            Delete           Bound       default/workspace-pvc       scale-ifse
scale-data-pv01         100Gi      ROX            Retain           Bound       default/scale-data-pvc
scale-models-pv01       100Gi      RWX            Retain           Bound       default/scale-models-pvc
```

Persistent Volumes *access modes*:

RWO = ReadWriteOnce ROX = ReadOnlyMany RWX = ReadWriteMany

(IV) User starts Pod with AI Workload and provisioned Storage

```
kind: Pod
apiVersion: v1
metadata:
  name: aipod
spec:
  containers:
    - name: tensorflow
      image: tensorflow:latest-gpu
      command: [ "/bin/sh", ...]
      args: [ "...cmd..." ]
      volumeMounts:
      - name: data
        mountPath: "/data"
        readOnly: true
      - name: models
        mountPath: "/models"
      - name: workspace
        mountPath: "/workspace"
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: scale-data-pvc
  - name: models
    persistentVolumeClaim:
      claimName: scale-models-pvc
  - name: workspace
    persistentVolumeClaim:
      claimName: workspace-pvc
```




```
/data:

/gpfs/fs0/data/
|-training
  |-images
    |-traffic
      |-pedestrians
      |-traffic-lights
|-inference
  |-ingest
    |-20190826
    |-20190915

/models:

/gpfs/fs0/models/
|-tensorflow
  |-object-detection
    |-pedestrians
    |-traffic-lights
```

```
/workspace:

/gpfs/fs0/pvc-aaf83bc1-2c33-401d-8c3a-713be508b96f
|- pvc-aaf83bc1-2c33-401d-8c3a-713be508b96f-data/
   |-my_saved_models
     |-model01
     |-model02
   |-my_scatch_space
     |-feature_engineering
     |-data_transformation
```

Pod

© IBM Corporation

OpenShift – NVIDIA GPU Driver Installation

Install Node Feature Discovery Operator

```
# git clone https://github.com/openshift/cluster-nfd-operator.git
# cd cluster-nfd-operator
# make deploy
```

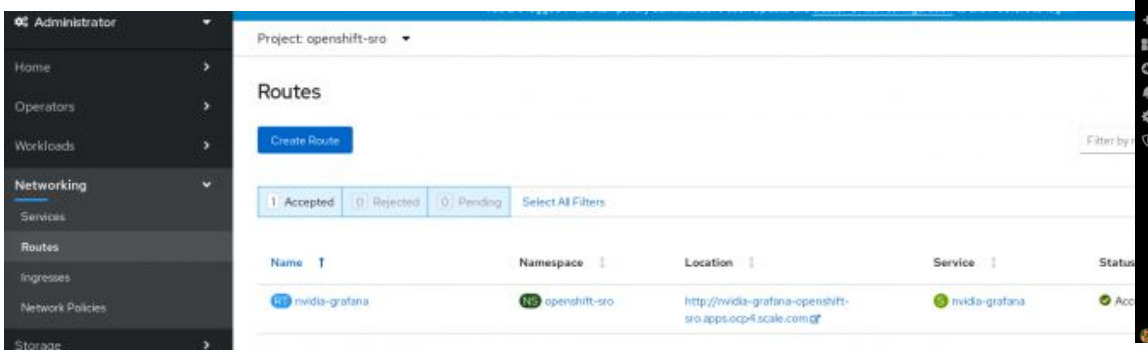
Disable nouveau on RHEL7x worker nodes

```
# cat /etc/modprobe.d/blacklist-nouveau.conf
blacklist nouveau
options nouveau modeset=0
```

Install NVIDIA Special Resource Operator

```
# git clone https://github.com/openshift-psap/special-resource-operator.git
# cd special-resource-operator
# PULLPOLICY=Always make deploy
```

Monitor NVIDIA GPU Load on Grafana Dashboard



Test GPU support with gpu-burn example workload

```
# git clone https://github.com/openshift-psap/gpu-burn.git
# cd gpu-burn
# oc create -f gpu-burn.yaml
```

See blog post:
Enabling and monitoring Nvidia GPUs on Openshift 4 for AI workloads
https://developer.ibm.com/storage/2020/02/10/...

References: IBM Storage for Red Hat OpenShift with CSI

IBM Knowledge Center: Spectrum Scale - CSI
https://www.ibm.com/support/knowledgecenter/...

IBM Spectrum Scale CSI driver video blogs
https://developer.ibm.com/storage/2019/12/26/ibm-spectrum-scale-csi-driver-video-blogs/

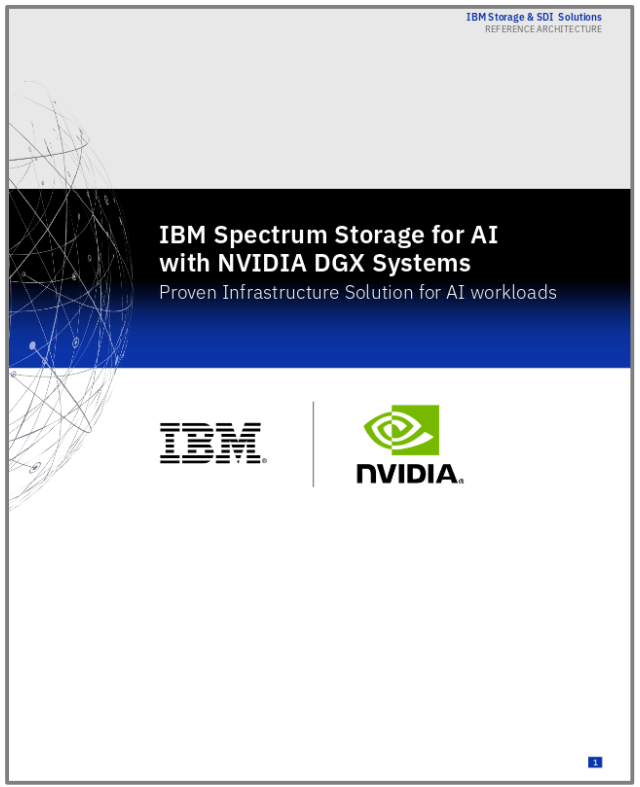
Github IBM/ibm-spectrum-scale-csi
https://github.com/IBM/ibm-spectrum-scale-csi

Playlist for Spectrum Scale CSI driver demos
https://youtu.be/tPnQNyjxFjk

IBM Spectrum Scale CSI Operator
https://ibm-spectrum-scale-csi-operator.readthedocs.io/en/latest/index.html

IBM Storage for Red Hat OpenShift Blueprint
http://www.redbooks.ibm.com/abstracts/redp5565.html?Open

References: IBM Storage for AI



IBM Spectrum Storage for AI with NVIDIA DGX Systems
https://www.ibm.com/downloads/cas/MNEQGQVP

Looking at IBM Spectrum Storage for AI with NVIDIA DGX performance
https://www.ibm.com/downloads/cas/MJLMALGL

How to use IBM Spectrum Scale with CSI Operator 1.0 on Openshift 4.2
https://developer.ibm.com/storage/2020/01/20/...

How to set up a remote cluster with IBM Spectrum Scale
https://developer.ibm.com/storage/2020/01/27/...

Enabling and monitoring Nvidia GPUs on Openshift 4 for AI workloads
https://developer.ibm.com/storage/2020/02/10/...

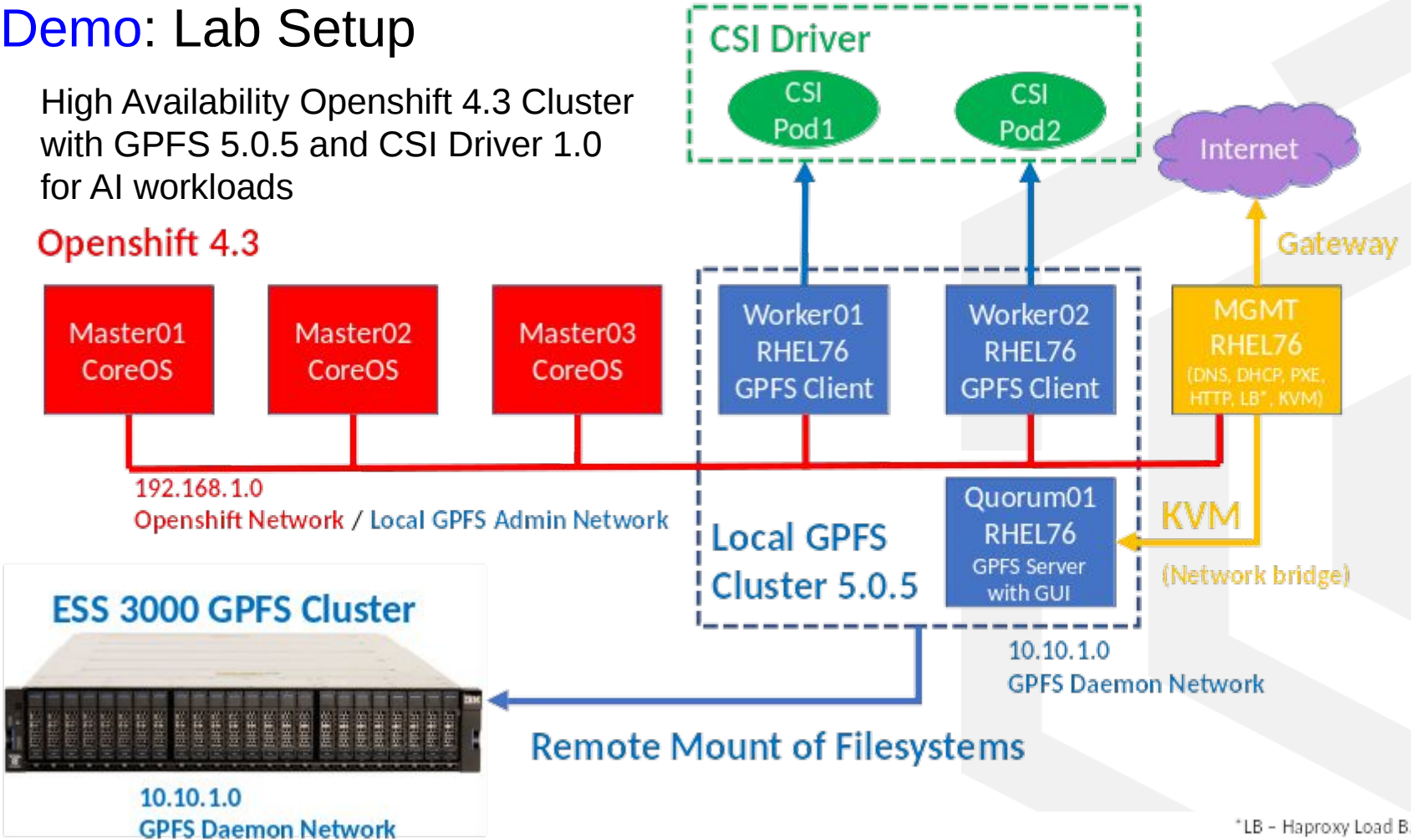
Live Demo

AI use case example with IBM Spectrum Scale CSI and TensorFlow in OpenShift 4.3

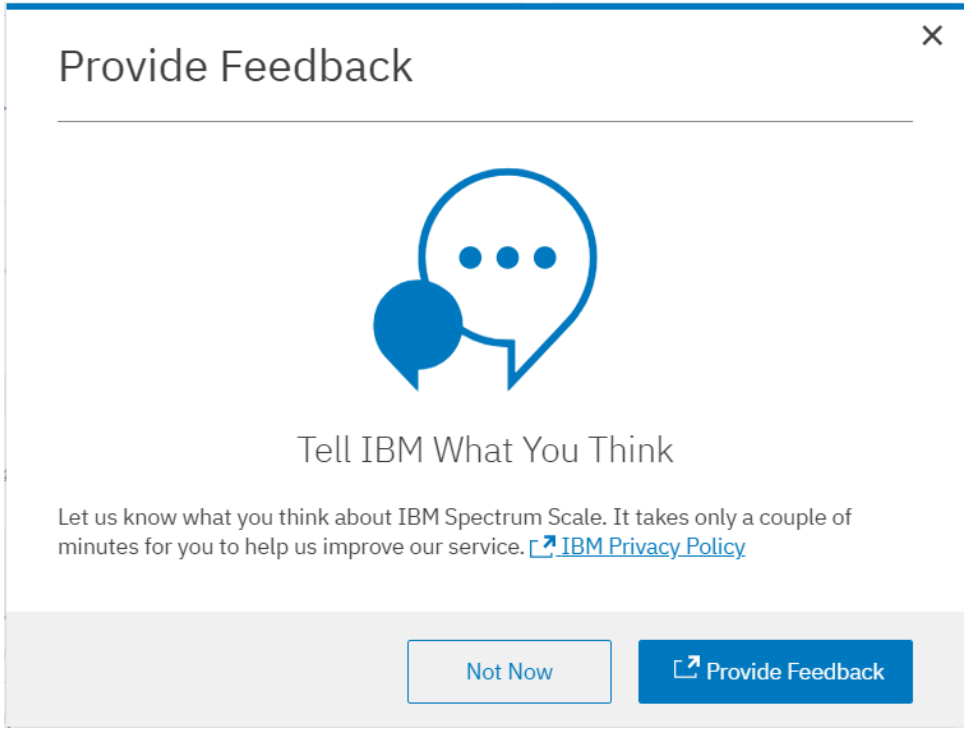
Przemyslaw Podfigurny (ppod@de.ibm.com)

Live Demo: Lab Setup

High Availability Openshift 4.3 Cluster with GPFS 5.0.5 and CSI Driver 1.0 for AI workloads



Thank you!



Please help us to improve Spectrum Scale with your feedback

- If you get a survey in email or a popup from the GUI, please respond
- We read every single reply

Questions?

DISCLAIMER: All product plans, directions and intent are subject to change or withdrawal without notice. References to IBM products, programs or services do not imply that they will be available in all countries in which IBM operates. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or services names may be trademarks or services marks of others.