TRV Test: Hopsworks install, step by step

Purpose: install Hopsworks 5.2.2 with Helm chart 5.2.3 on an OpenShift 4.20 cluster, step by step: the commands, and what each must print. Each step has a section with the same number in the annex, which holds the requirements, the reasons, and what to do when a step does not go as written. Steps marked **(platform team)** change the cluster outside the Hopsworks namespace.

Draft, 2026-11-03.

0. Prepare the workstation

Run every command on this page in one bash shell. On macOS the default shell is zsh, which reads some of these commands differently:

```
echo "${BASH_VERSION:-not bash: run bash and start again}"

```

It must print a bash version. Otherwise run `bash` and keep that shell open until the install is done.

0a. Downloads.

```
export INSTALL_DIR="${INSTALL_DIR:-$HOME/hopsworks-install}"
mkdir -p "$INSTALL_DIR" && chmod 700 "$INSTALL_DIR" && cd "$INSTALL_DIR"
for f in SHA256SUMS hopsworks-prechecks.tar.gz hopsworks-5.2.3.tgz consul-1.3.1.tgz values.images-5.2.3.yaml values.placement-5.2.3.yaml hopsworks-prechecks-jobs.yaml values.sizing-5.2.3.yaml; do
  curl -fsSLO "https://repo.hops.works/dev/alex/trv/$f" || echo "download failed: $f"
done
sha256sum -c SHA256SUMS 2>/dev/null || shasum -a 256 -c SHA256SUMS

```

0b. Settings.

```
cd "$INSTALL_DIR"
tar -xzf hopsworks-prechecks.tar.gz && export CHECKS="$PWD/hopsworks-prechecks"
cp "$CHECKS/hopsworks-install.env" . && chmod 600 hopsworks-install.env
${EDITOR:-vi} hopsworks-install.env                    # fill in every <...> except the three from step 4b
set -a; source hopsworks-install.env; set +a
grep -n '"<' hopsworks-install.env | grep -v 'from step 4b' || echo "every setting filled in"
echo "base images $BASE_IMAGE_VERSION"
read -rs -p "cache robot token: " MIRROR_TOKEN; echo; export MIRROR_TOKEN
read -rs -p "build robot token: " BUILD_TOKEN; echo; export BUILD_TOKEN
read -rs -p "S3 access key: " AWS_ACCESS_KEY_ID; echo; export AWS_ACCESS_KEY_ID
read -rs -p "S3 secret key: " AWS_SECRET_ACCESS_KEY; echo; export AWS_SECRET_ACCESS_KEY

```

0c. Workstation check, logged in with `oc` as the account that installs:

```
bash "$CHECKS/workstation/check.sh"

```

Pass: `0 fail, 0 unknown`. It checks the tools, that the account is cluster-admin, `vm.max_map_count` on every node (one short-lived `oc debug` pod per node), and the

Helm version. Everything else about the cluster is checked from inside it in step 4b.

1. Registry trust and pull secret (platform team)

1a. Trust Quay's CA on the nodes. Leave out if Quay's certificate is publicly trusted (annex 0, No custom CA).

```
oc create configmap registry-ca -n openshift-config --from-file="$QUAY_HOST"="$CA_BUNDLE"
oc patch image.config.openshift.io/cluster --type=merge \
  -p '{"spec":{"additionalTrustedCA":{"name":"registry-ca"}}}'

```

1b. Add the two robots to the cluster pull secret.

```
oc get secret/pull-secret -n openshift-config -o jsonpath='{.data.\.dockerconfigjson}' | base64 -d > pull-secret.json
jq --arg m "$QUAY_HOST/$MIRROR_ORG" --arg ma "$(printf '%s:%s' "$MIRROR_ROBOT" "$MIRROR_TOKEN" | base64 | tr -d '\n')" \
   --arg b "$QUAY_HOST/$BUILD_ORG"  --arg ba "$(printf '%s:%s' "$BUILD_ROBOT" "$BUILD_TOKEN" | base64 | tr -d '\n')" \
   '.auths[$m] = {auth: $ma} | .auths[$b] = {auth: $ba}' pull-secret.json > pull-secret.new.json
oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=pull-secret.new.json
rm -f pull-secret.json pull-secret.new.json

```

1c. Wait for the nodes to take the change. A few minutes; no drain, no reboot.

```
sleep 60    # the new configuration takes up to a minute to reach the pools
oc wait mcp --all --for=condition=Updated --timeout=20m
oc get mcp  # every pool UPDATED True, UPDATING False, DEGRADED False

```

2. Hopsworks namespace (platform team for the node selector)

```
oc new-project "$NAMESPACE"
oc annotate namespace "$NAMESPACE" --overwrite openshift.io/node-selector="${NODE_LABEL:-node-role.kubernetes.io/worker=}"
RANGE=$(oc get namespace "$NAMESPACE" -o jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.uid-range}')
SUPP_GROUPS=$(oc get namespace "$NAMESPACE" -o jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.supplemental-groups}')
START=${RANGE%%/*}
echo "uid-range $RANGE  supplemental-groups $SUPP_GROUPS  magicuser $((START + 10)) magicgroup $START"

```

Write the OpenShift overlay with the namespace's ids:

```
tar -xzOf "$CHART" hopsworks/values.openshift.yaml > values-openshift.yaml
sed -i.orig -E -e "s/^(magicuser: &magicuser )[0-9]+$/\1$((START + 10))/" \
               -e "s/^(magicgroup: &magicgroup )[0-9]+$/\1$START/" values-openshift.yaml
grep -cE "^magicuser: &magicuser $((START + 10))$" values-openshift.yaml   # must print 1
grep -cE "^magicgroup: &magicgroup $START$" values-openshift.yaml          # must print 1

```

Probe a pull through the Quay cache:

```
IMG="$QUAY_HOST/$MIRROR_ORG/hopsworks/hwutils:1.10"
oc -n "$NAMESPACE" run pull-probe --image="$IMG" --restart=Never --overrides="$(jq -nc --arg i "$IMG" '{spec: {
  securityContext: {runAsNonRoot: true, seccompProfile: {type: "RuntimeDefault"}},
  containers: [{name: "pull-probe", image: $i, command: ["true"],
```

```
    securityContext: {allowPrivilegeEscalation: false, capabilities: {drop:
["ALL"]}}}]}}')"
sleep 60; oc -n "$NAMESPACE" get pod pull-probe -o wide   # Completed (or Error from the
command): pulled; NODE: a worker node
oc -n "$NAMESPACE" delete pod pull-probe

```

The pod must reach `Completed` (or `Error` from the command) on a worker node. `ErrImagePull` means step 1 is incomplete: annex 2.

3. SCC grants (platform team)

```
oc adm policy add-scc-to-user privileged "system:serviceaccount:$NAMESPACE:filebeat"
# I2
oc adm policy add-scc-to-user anyuid "system:serviceaccount:$NAMESPACE:image-builder"
# I3

```

The HopsFS CSI node plugin's SCC (I1), created by the platform team (option B). To let the chart create it during step 8 instead (option A), skip this block, leave `CSI_SCC` empty, and skip the option B block in step 7 (annex 3).

```
CSI_SCC="$RELEASE-$NAMESPACE-hopsfs-csi-node-plugin"
oc apply -f - <<EOF
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: $CSI_SCC
allowPrivilegedContainer: true
allowPrivilegeEscalation: true
allowHostDirVolumePlugin: true
allowHostNetwork: false
allowHostPID: false
allowHostIPC: false
allowHostPorts: false
readOnlyRootFilesystem: false
allowedCapabilities: []
defaultAddCapabilities: []
requiredDropCapabilities: []
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
seccompProfiles:
  - runtime/default
volumes:
  - hostPath
  - projected
users:
  - system:serviceaccount:$NAMESPACE:hopsfs-csi-node-sa
EOF
oc get scc "$CSI_SCC" -o jsonpath='{.users}{"\n"}'    # must name system:serviceaccount:
<namespace>:hopsfs-csi-node-sa

```

4. Secrets and the CA in the namespace

Builds the build robot's credential file, then creates the object store Secret, the registry Secret and the CA ConfigMap. Leave out the `hopsworks-extra-ca` line without a custom CA.

```
B64=$(printf '%s:%s' "$BUILD_ROBOT" "$BUILD_TOKEN" | base64 | tr -d '\n')
( umask 077; jq -n --arg h "$QUAY_HOST" --arg a "$B64" \
    '{auths: {($h): {auth: $a}, ($h + ":443"): {auth: $a}}}' > "$REGISTRY_CREDS_FILE" );
unset B64
oc -n "$NAMESPACE" create secret generic aws-credentials \
  --from-literal=access-key-id="$AWS_ACCESS_KEY_ID" --from-literal=secret-access-
key="$AWS_SECRET_ACCESS_KEY"
oc -n "$NAMESPACE" create secret generic registry-creds --
type=kubernetes.io/dockerconfigjson \
  --from-file=.dockerconfigjson="$REGISTRY_CREDS_FILE"
rm -f "$REGISTRY_CREDS_FILE"
oc -n "$NAMESPACE" create configmap hopsworks-extra-ca --from-file=ca.crt="$CA_BUNDLE"

```

4a. The ProGet feed key for custom commands

```
curl -fsSLo apt-feed.gpg ${CA_BUNDLE:+--cacert "$CA_BUNDLE"} "$APT_KEY_URL"
test -s apt-feed.gpg && echo "feed key saved"

```

ProGet signs its Debian feed with its own key, not Ubuntu's, and apt refuses a feed whose key it does not hold (`NO_PUBKEY`). Hopsworks 5.2 has no setting that installs a key into environment builds, so users upload `apt-feed.gpg` with every custom command that installs Ubuntu packages (step 10). The apt sources in step 7 point at it there (`signed-by=/srv/hops/build/apt-feed.gpg`). Keep the file to hand to users.

4b. Checks inside the namespace

Runs the cluster, Quay, object store and ProGet checks as Jobs in the Hopsworks namespace, with the Secrets and CA from step 4, then proves the grants of step 3 with probe pods. Every object it creates is removed at the end. What each check covers: annex 4b.

```
CSI_SCC="${CSI_SCC:-}" bash "$CHECKS/jobs/run.sh" 2>&1 | tee prechecks-namespace.out;
echo "exit ${PIPESTATUS[0]}"

```

Pass: every line `pass` in the summary at the end, and `exit 0` . A line may add what a later step handles: `openshift-cluster` "needs install steps 6 and 6b (cert-manager)", `s3-object-store` "needs the legacy checksum settings" (step 7 sets them, step 8 checks them). A failure prints its fix; fix it and run 4b again.

A Job that does not finish, with its pod in `ImagePullBackOff` or a timeout: run the laptop prechecks P2 to P4 from the workstation. If they pass, the fault is on the path from the pod network (egress rules, the proxy); if they fail, the service or its credentials.

If the bucket does not exist yet, run 4b with `SKIP_S3=1` and run it again without before step 8 (annex 4b).

Then copy the three settings the summary ends with into `hopsworks-install.env` , and note P1.5's cert-manager state (`none` , `running` or `leftover CRDs`) for steps 6 and 6b:

```
${EDITOR:-vi} hopsworks-install.env                    # KSERVE_CERT_MANAGER,
S3_LEGACY_CHECKSUMS, S3_DISABLE_CERT_CHECKING
set -a; source hopsworks-install.env; set +a

```

```
grep -n '"<' hopsworks-install.env || echo "every setting filled in"

```

5. Consul and the `.consul` forward (platform team for the DNS patch)

```
cat > values-consul.yaml <<EOF
global:
  enabled: true
  datacenter: dc1
  image: "$QUAY_HOST/$MIRROR_ORG/hashicorp/consul:1.16.3"
  imageK8S: "$QUAY_HOST/$MIRROR_ORG/hashicorp/consul-k8s-control-plane:1.3.1"
  gossipEncryption: {autoGenerate: true}
  tls: {enabled: true}
  acls: {manageSystemACLs: true}
  metrics: {enabled: true, enableAgentMetrics: true, enableGatewayMetrics: false}
  openshift: {enabled: true}
domain: consul
fullnameOverride: hopsworks-consul
server:
  enabled: true
  replicas: 3
  connect: false
  resources:
    requests: {memory: 200Mi, cpu: 100m}
    limits: {memory: 350Mi, cpu: 100m}
dns: {enabled: true}
client: {hostNetwork: true, dnsPolicy: ClusterFirstWithHostNet}
ui: {enabled: true}
syncCatalog:
  enabled: true
  toConsul: true
  toK8S: false
  default: true
  resources:
    requests: {memory: 100Mi, cpu: 50m}
    limits: {memory: 100Mi, cpu: 50m}
connectInject: {enabled: false}
EOF
helm upgrade --install hopsworks-consul "$CONSUL_CHART" -n "$NAMESPACE" \
  --values values-consul.yaml --wait --timeout 15m
CONSUL_IP=$(oc -n "$NAMESPACE" get svc hopsworks-consul-dns -o
jsonpath='{.spec.clusterIP}')
echo "consul dns: $CONSUL_IP"                 # must not be empty
oc get dns.operator/default -o jsonpath='{.spec.servers}{"\n"}'   # empty: run the patch
below
oc patch dns.operator/default --type=merge -p "{\"spec\":{\"servers\":
[{\"name\":\"consul-server\",\"zones\":[\"consul\"],\"forwardPlugin\":
{\"policy\":\"Random\",\"upstreams\":[\"$CONSUL_IP\"]}}]}}"

```

`helm` must end with `STATUS: deployed`. If the `get` listed other forwards, add this one to the list with `oc edit dns.operator/default` instead of patching. Fewer than 3 schedulable nodes: annex 5.

6. cert-manager CRDs at v1.21.1

Skip if P1.5 printed `state: none`.

```
crd_versions() { oc get crd -o json | jq -r '.items[] | select(.spec.group |
endswith("cert-manager.io"))
  | "\(.metadata.name) \(.metadata.labels["app.kubernetes.io/version"] // "-")"'; }
crd_versions                                          # the version they have now
tar -xzOf "$CHART" hopsworks/charts/kserve/crds/cert-manager-crds.yaml > cert-manager-
crds-v1.21.1.yaml
oc apply --server-side --force-conflicts --field-manager=helm -f cert-manager-crds-
v1.21.1.yaml
crd_versions                                          # every one v1.21.1
```

```

```

Record the Certificates, for Verify 10:

```
certs() { oc get certificates -A -o json | jq -r '.items[] | [.metadata.namespace,
.metadata.name,
  (.status.conditions[]? | select(.type == "Ready") | .status), .status.revision,
.status.notAfter] | @tsv'; }
certs | tee certificates-before.tsv

```

6b. Secrets of a removed cert-manager's ClusterIssuers

Only if P1.5 printed `leftover CRDs`. List the ClusterIssuers and the Secret each reads:

```
oc get clusterissuers -o json | jq -r '.items[] | [.metadata.name,
  (.spec | (.ca.secretName // .acme.privateKeySecretRef.name //
.vault.auth.tokenSecretRef.name
    // .vault.auth.appRole.secretRef.name // .venafi.tpp.credentialsRef.name //
.venafi.cloud.apiTokenSecretRef.name // "-"))] | @tsv' | tee clusterissuer-secrets.tsv

```

Copy the listed Secrets into the Hopsworks namespace, once the platform team agrees: a CA ClusterIssuer's Secret is a CA private key. The alternative and what else happens to the Certificates: annex 6b.

```
OLD_NS=cert-manager
for s in $(awk -F'\t' '$2 != "-" {print $2}' clusterissuer-secrets.tsv); do
  oc -n "$OLD_NS" get secret "$s" -o json \
    | jq --arg ns "$NAMESPACE" 'del(.metadata.uid, .metadata.resourceVersion,
.metadata.creationTimestamp,
        .metadata.managedFields, .metadata.ownerReferences) | .metadata.namespace = $ns'
\
    | oc apply -f -
done

```

7. Hopsworks values

```
cat > values-site.yaml <<EOF
global:
  _hopsworks:
    imageRegistry: "$QUAY_HOST/$MIRROR_ORG"
    openshift:
      enabled: true
    kyverno:
      enabled: false
    minio:
      enabled: false          # the switch for the bundled MinIO; a top-level
minio.enabled does not remove it
    managedDockerRegistery:
      enabled: true
      domain: "$QUAY_HOST"
      namespace: "$BUILD_ORG"
      port: "443"
    managedObjectStorage:
      enabled: true
      s3:
        bucket:
          name: $S3_BUCKET
        region: $S3_REGION
        endpoint: "$S3_ENDPOINT"
        secret:
          name: aws-credentials
          access_key_id: access-key-id
          secret_key_id: secret-access-key
```

```
    backups:
      enabled: true

hopsworks:
  terminal:
    enabled: true
  velero:
    backup:
      enabled: false          # no Velero on this cluster; also skips the chart's Velero check
  dockerRegistry:
    preset:
      secrets:
        - registry-creds
  variables:
    kube_kserve_installed: true
    docker_operations_managed_docker_secrets: 'registry-creds'
    docker_operations_push_insecure: 'true'   # see Known limitations
  create_certificate:
    trustManager:
      enabled: true
      configMapName: hopsworks-extra-ca
      key: ca.crt
      aliasPrefix: hopsworks
  dockerImage:
    packageAuth:
      caCertsConfigMap: hopsworks-extra-ca
    pypi:
      global_parameters:
        index-url: "$PYPI_INDEX_URL"
    apt:
      sources:
        - "deb [signed-by=/srv/hops/build/apt-feed.gpg] $APT_FEED_URL resolute main universe"
        - "deb [signed-by=/srv/hops/build/apt-feed.gpg] $APT_FEED_URL resolute-updates main universe"
        - "deb [signed-by=/srv/hops/build/apt-feed.gpg] $APT_FEED_URL resolute-security main universe"

hopsfs:
  objectStorage:
    s3:
      disableCertChecking: $S3_DISABLE_CERT_CHECKING
      disableChecksumValidation: $S3_LEGACY_CHECKSUMS

olk:
  opensearch:
    backup:
      repositories:
        default:
          s3:
            endpoint: "$S3_ENDPOINT"
            region: $S3_REGION
            path_style_access: true
            protocol: "https"
            legacy_md5_checksum_calculation: $S3_LEGACY_CHECKSUMS
          settings:
            bucket: $S3_BUCKET
            base_path: opensearch_backup
          credentials:
            secret_from: aws-credentials
            access_key_id: access-key-id
            secret_key_id: secret-access-key

rondb:
  rondb:
    backups:
      s3:
        provider: Other
        endpoint: "$S3_ENDPOINT"
        bucketName: $S3_BUCKET
```

```
        region: $S3_REGION
        keyCredentialsSecret:
          name: aws-credentials
          key: access-key-id
        secretCredentialsSecret:
          name: aws-credentials
          key: secret-access-key

consul:
  fullnameOverride: hopsworks-consul

kserve:
  cert_manager:
    enabled: $KSERVE_CERT_MANAGER
EOF
grep -n '\$' values-site.yaml || echo "every variable expanded"

```

Name the SCC from step 3 (option B; in a new shell, set `CSI_SCC` again first):

```
cat >> values-site.yaml <<EOF

hopsfs-csi:
  openshift:
    existingSecurityContextConstraints:
      name: "$CSI_SCC"
EOF
grep -n 'name: "' values-site.yaml | tail -1            # must show the SCC name, not name: ""

```

The image values and the placement values:

```
sed "s|@MIRROR@|$QUAY_HOST/$MIRROR_ORG|g" values.images-5.2.3.yaml > values-images.yaml
grep -c '@MIRROR@' values-images.yaml        # must print 0

```

```
if [ -n "$NODE_LABEL" ]; then
  sed -e "s|@NODE_LABEL_KEY@|${NODE_LABEL%%=*}|g" -e "s|@NODE_LABEL_VALUE@|${NODE_LABEL#*=}|g" \
    values.placement-5.2.3.yaml > values-placement.yaml
else
  echo '{}' > values-placement.yaml               # no label: no placement values
fi
grep -c '@NODE_LABEL' values-placement.yaml   # must print 0

```

8. Install Hopsworks

First check the rendered chart against the cluster; it changes nothing:

```
bash "$CHECKS/hopsworks-render/check.sh" \
  "$SIZING_VALUES" values-openshift.yaml values-site.yaml values-images.yaml values-placement.yaml

```

Pass: `0 fail, 0 unknown`. What its seven checks cover: annex 8.

```
helm upgrade --install "$RELEASE" "$CHART" -n "$NAMESPACE" --timeout 90m \
  --values "$SIZING_VALUES" --values values-openshift.yaml --values values-site.yaml \
  --values values-images.yaml --values values-placement.yaml

```

Helm returns once the release is recorded; the install settles in the background in about an hour:

```
watch -n 30 "oc -n $NAMESPACE get pods | grep -v -E 'Running|Completed'"

```

Settled: the only lines left are the header, `create-repos` jobs (they fail with a private CA, annex Known limitations) and failed attempts of Jobs that later completed; `hopsworks-instance` and `hopsworks-admin` show all containers ready. If it does not settle: annex 8a, 8b.

8c. With a private CA on Quay: let model deployments resolve their images (HWORKS-3664). Put the original value back before every `helm upgrade`, and patch again after it (annex 8c):

```
SKIP=$(oc -n "$NAMESPACE" get configmap config-deployment -o jsonpath='{.data.registries-skipping-tag-resolving}')
echo "$SKIP" | tee skip-registries.orig       # <quay>/<cache org>,<quay>:443
oc -n "$NAMESPACE" patch configmap config-deployment --type=merge \
  -p "$(jq -nc --arg v "$SKIP,$QUAY_HOST" '{data: {"registries-skipping-tag-resolving": $v}}')"

```

9. Every Hopsworks project (recurring, platform team)

Before a user creates a project, create its namespace and give its name to the user:

```
PROJECT_NS=<namespace for the project>
oc new-project "$PROJECT_NS"
oc annotate namespace "$PROJECT_NS" --overwrite \
  openshift.io/node-selector="${USER_NODE_LABEL:-node-role.kubernetes.io/worker=}" \
  openshift.io/sa.scc.uid-range="$(oc get namespace "$NAMESPACE" -o jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.uid-range}')" \
  openshift.io/sa.scc.supplemental-groups="$(oc get namespace "$NAMESPACE" -o jsonpath='{.metadata.annotations.openshift\.io/sa\.scc\.supplemental-groups}')"

```

Projects are created with that name: `hopsworks.create_project("<project>", namespace="<PROJECT_NS>")`, or the namespace field in the UI.

10. Custom commands that install Ubuntu packages (users)

Upload `apt-feed.gpg` from step 4a with the script, and point apt at the feed alone:

```
#!/bin/bash
set -euo pipefail
mkdir -p /srv/hops/build/no-parts                    # an empty parts directory: hides the default Ubuntu sources
FEED_ONLY=(-o Dir::Etc::SourceList=/etc/apt/sources.list.d/proxy_sources.list
           -o Dir::Etc::SourceParts=/srv/hops/build/no-parts)
sudo apt-get "${FEED_ONLY[@]}" update
sudo apt-get "${FEED_ONLY[@]}" install -y --no-install-recommends <package>

```

Verify

1. Every pod admitted as designed, `restricted-v2` except `filebeat-*` (`privileged`), `hopsfs-csi-node-*` (the I1 SCC) and `image-builder` build pods (`anyuid`):

```
oc get pods -n "$NAMESPACE" -o custom-columns=POD:.metadata.name,SCC:.metadata.annotations.openshift\\.io/scc,STATUS:.status.ph
```

```
ase

```

2. No admission refusals: `oc get events -n "$NAMESPACE" | grep -i 'unable to validate against any security context constraint'` prints nothing.

3. `helm status "$RELEASE" -n "$NAMESPACE"` shows `deployed`.

4. Open the host from `oc -n "$NAMESPACE" get route,ingress`, log in as `admin@hopsworks.ai` with the initial password `admin`, and change it at once.

5. `oc -n "$NAMESPACE" get jobs | grep preset` shows Complete.

6. In one project namespace from step 9: an environment build, a Jupyter server, a Python job, a Spark job, an agent job, a model deployment and a web terminal session; then check 1 for that namespace.

7. A custom command from step 10 installing `jq` completes.

8. No image from outside Quay: `oc get pods -A -o jsonpath='{range .items[*]}{.metadata.namespace}{" "}{range .spec.containers[*]}{.image}{" "}{end}{"\n"}{end}' | grep -E "^($NAMESPACE|<project namespaces>) " | grep -v "$QUAY_HOST/"` prints nothing.

9. No Hopsworks pod off its nodes: annex Verify 9.

10. After step 6: `certs | diff certificates-before.tsv -` shows only Certificates that were due for renewal; `oc get clusterissuers,issuers -A` all Ready.