General information

An **Upgrade** button appears on the **Dashboard** screen whenever a new Harvester version that you can upgrade to becomes available. For more information, see Start an upgrade.

For air-gapped environments, see Prepare an air-gapped upgrade.

Preventing Corruption of VM Images During Upgrade

:::caution

Before proceeding with the upgrade to Harvester **v1.4.0**, please make sure the **BackingImage** CRD is updated to the Longhorn **v1.7.2** version beforehand.

If this step is skipped, it may lead to backing image corruption, as described in this known Longhorn issue.

:::

To prevent the issue from occurring, you can manually update the `BackingImage` CRD before upgrading Harvester.

1. Patch the **Harvester managedchart** to avoid related errors and warnings.

```
kubectl patch managedchart harvester \
-n fleet-local \
--type='json' \
-p='[
  {
    "op":"add",
    "path":"/spec/diff/comparePatches/-",
    "value": {
      "apiVersion":"apiextensions.k8s.io/v1",
      "jsonPointers":["/spec","/metadata/annotations", "/metadata/labels",
"/status"],
      "kind":"CustomResourceDefinition",
      "name":"backingimages.longhorn.io"
    }
  }
]'
```

1. Apply the **Longhorn v1.7.2** `BackingImage` CRD.

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.15.0
  labels:
    app.kubernetes.io/name: longhorn
    app.kubernetes.io/instance: longhorn
    app.kubernetes.io/version: v1.7.2
    longhorn-manager: ""
  name: backingimages.longhorn.io
```

```
spec:
  conversion:
    strategy: Webhook
    webhook:
      clientConfig:
        service:
          name: longhorn-conversion-webhook
          namespace: longhorn-system
          path: /v1/webhook/conversion
          port: 9501
      conversionReviewVersions:
      - v1beta2
      - v1beta1
  group: longhorn.io
  names:
    kind: BackingImage
    listKind: BackingImageList
    plural: backingimages
    shortNames:
    - lhbi
    singular: backingimage
  scope: Namespaced
  versions:
  - additionalPrinterColumns:
    - description: The backing image name
      jsonPath: .spec.image
      name: Image
      type: string
    - jsonPath: .metadata.creationTimestamp
      name: Age
      type: date
    name: v1beta1
    schema:
      openAPIV3Schema:
        description: BackingImage is where Longhorn stores backing image object.
        properties:
          apiVersion:
            description: |-
              APIVersion defines the versioned schema of this representation of an
object.
              Servers should convert recognized schemas to the latest internal value,
and
              may reject unrecognized values.
              More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#resources
            type: string
          kind:
            description: |-
              Kind is a string value representing the REST resource this object
represents.
              Servers may infer this from the endpoint the client submits requests to.
              Cannot be updated.
              In CamelCase.
              More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#types-kinds
            type: string
          metadata:
            type: object
          spec:
            x-kubernetes-preserve-unknown-fields: true
          status:
            x-kubernetes-preserve-unknown-fields: true
        type: object
    served: true
    storage: false
    subresources:
      status: {}
  - additionalPrinterColumns:
    - description: The system generated UUID
      jsonPath: .status.uuid
      name: UUID
      type: string
    - description: The source of the backing image file data
      jsonPath: .spec.sourceType
      name: SourceType
      type: string
    - description: The backing image file size in each disk
      jsonPath: .status.size
      name: Size
      type: string
    - description: The virtual size of the image (may be larger than file size)
      jsonPath: .status.virtualSize
      name: VirtualSize
      type: string
    - jsonPath: .metadata.creationTimestamp
      name: Age
      type: date
    name: v1beta2
    schema:
      openAPIV3Schema:
        description: BackingImage is where Longhorn stores backing image object.
        properties:
          apiVersion:
            description: |-
              APIVersion defines the versioned schema of this representation of an
object.
              Servers should convert recognized schemas to the latest internal value,
and
              may reject unrecognized values.
              More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#resources
            type: string
          kind:
            description: |-
```

```
              Kind is a string value representing the REST resource this object
represents.
              Servers may infer this from the endpoint the client submits requests to.
              Cannot be updated.
              In CamelCase.
              More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#types-kinds
            type: string
          metadata:
            type: object
          spec:
            description: BackingImageSpec defines the desired state of the Longhorn
              backing image
            properties:
              checksum:
                type: string
              diskFileSpecMap:
                additionalProperties:
                  properties:
                    evictionRequested:
                      type: boolean
                  type: object
                type: object
              diskSelector:
                items:
                  type: string
                type: array
              disks:
                additionalProperties:
                  type: string
                description: Deprecated. We are now using DiskFileSpecMap to assign
                  different spec to the file on different disks.
                type: object
              minNumberOfCopies:
                type: integer
              nodeSelector:
                items:
                  type: string
                type: array
              secret:
                type: string
              secretNamespace:
                type: string
              sourceParameters:
                additionalProperties:
                  type: string
                type: object
              sourceType:
                enum:
                - download
                - upload
                - export-from-volume
                - restore
                - clone
                type: string
            type: object
          status:
            description: BackingImageStatus defines the observed state of the Longhorn
              backing image status
            properties:
              checksum:
                type: string
              diskFileStatusMap:
                additionalProperties:
                  properties:
                    lastStateTransitionTime:
                      type: string
                    message:
                      type: string
                    progress:
                      type: integer
                    state:
                      type: string
                  type: object
                nullable: true
                type: object
              diskLastRefAtMap:
                additionalProperties:
                  type: string
                nullable: true
                type: object
              ownerID:
                type: string
              size:
                format: int64
                type: integer
              uuid:
                type: string
              virtualSize:
                description: Virtual size of image, which may be larger than physical
                  size. Will be zero until known (e.g. while a backing image is
uploading)
                format: int64
                type: integer
            type: object
        type: object
    served: true
    storage: true
    subresources:
      status: {}
```

3. Start the upgrade process.

Known issues

---

1. A VM with a container disk can't be migrated which makes the upgrade stuck in pre-drain status

:::tip

Manually stop the VMs to continue the upgrade process.

:::

When upgrading from v1.3.2 to v1.4.0, the upgrade process may become stuck if a VM with a container disk cannot be migrated. There is some limitation of live migration.

For more information, see [Issue #7005](#).

---

2. Upgrade stuck on waiting for Harvester bundle

When upgrading from v1.3.2 to v1.4.0, the upgrade process may become stuck on waiting for the Harvester bundle to become ready. This issue is caused by a race condition when the Fleet agent (`fleet-agent`) is redeployed.

The following error messages indicate that the issue exists.

```
> kubectl get bundles -n fleet-local
NAME                                              BUNDLEDEPLOYMENTS-READY   STATUS
mcc-harvester                                     0/1
ErrApplied(1) [Cluster fleet-local/local: encountered 2 deletion errors. First is:
admission webhook "validator.harvesterhci.io" denied the request: Internal error
occurred: no route match found for DELETE /v1, Kind=Secret harvester-
system/sh.helm.release.v1.harvester.v2]
mcc-harvester-crd                                 0/1
ErrApplied(1) [Cluster fleet-local/local: admission webhook
"validator.harvesterhci.io" denied the request: Internal error occurred: no route
match found for DELETE /v1, Kind=Secret harvester-
system/sh.helm.release.v1.harvester-crd.v1]
```

You can run the following script to fix the issue.

```
#!/bin/bash

patch_fleet_bundle() {
  local bundleName=$1
  local generation=$(kubectl get -n fleet-local bundle ${bundleName} -o
jsonpath='{.spec.forceSyncGeneration}')
  local new_generation=$((generation+1))
  patch_manifest="$(mktemp)"
  cat > "$patch_manifest" <<EOF
{
  "spec": {
    "forceSyncGeneration": $new_generation
  }
}
EOF
  echo "patch bundle to new generation: $new_generation"
  kubectl patch -n fleet-local bundle ${bundleName}  --type=merge --patch-file
$patch_manifest
  rm -f $patch_manifest
}

echo "removing harvester validating webhook"
kubectl delete validatingwebhookconfiguration harvester-validator

for bundle in mcc-harvester-crd mcc-harvester
do
  patch_fleet_bundle ${bundle}
done

echo "removing longhorn services"
kubectl delete svc longhorn-engine-manager -n longhorn-system --ignore-not-
found=true
kubectl delete svc longhorn-replica-manager -n longhorn-system --ignore-not-
found=true
```

---

3. Upgrade stuck on waiting for Fleet

When upgrading from v1.3.2 to v1.4.0, the upgrade process may become stuck on waiting for Fleet to become ready. This issue is caused by a race condition when Rancher is redeployed.

Check the Harvester logs and Fleet history for the following indicators:

- The manifest pod is stuck in the `deployed` status.
- The upgrade is pending with a chart version that has been deployed.

Example:

```
> kubectl logs -n harvester-system -l harvesterhci.io/upgradeComponent=manifest
wait helm release cattle-fleet-system fleet fleet-104.0.2+up0.10.2 0.10.2 deployed

> helm history -n cattle-fleet-system fleet
REVISION        UPDATED                         STATUS          CHART
APP VERSION     DESCRIPTION
26              Tue Dec 10 03:09:13 2024        superseded      fleet-
103.1.5+up0.9.5 0.9.5           Upgrade complete
27              Sun Dec 15 09:26:54 2024        superseded      fleet-
103.1.5+up0.9.5 0.9.5           Upgrade complete
28              Sun Dec 15 09:27:03 2024        superseded      fleet-
103.1.5+up0.9.5 0.9.5           Upgrade complete
29              Mon Dec 16 05:57:03 2024        deployed        fleet-
103.1.5+up0.9.5 0.9.5           Upgrade complete
30              Mon Dec 16 05:57:13 2024        pending-upgrade fleet-
103.1.5+up0.9.5 0.9.5           Preparing upgrade
```

You can run the following command to fix the issue.

```
helm rollback fleet -n cattle-fleet-system <last-deployed-revision>
```

---

4. Upgrade will start over again unexpectedly after clicking the "Dismiss it" button

When you use Rancher to upgrade Harvester, the Rancher UI displays a dialog with a button labeled "Dismiss it". Clicking this button may result in the following issues:

- The `status` section of the `harvesterhci.io/v1beta1/upgrade` CR is cleared, causing the loss of all important information about the upgrade.
- The upgrade process starts over again unexpectedly.

This issue affects Rancher v2.10.x, which uses v1.0.2, v1.0.3, and v1.0.4 of the Harvester UI Extension. All Harvester UI versions are not affected. The issue will be fixed in Harvester UI Extension v1.0.5 and v1.5.0.

To avoid this issue, perform either of the following actions:

- Use the Harvester UI to upgrade Harvester. Clicking the "Dismiss it" button on the Harvester UI does not result in unexpected behavior.
- Instead of clicking the button on the Rancher UI, run the following command against the cluster:

```
kubectl -n harvester-system label upgrades -l harvesterhci.io/latestUpgrade=true
harvesterhci.io/read-message=true
```

Related issue:

- [BUG] upgrade controller does not handle read-message well due to UI menu `Dismiss it` wipes upgrade CR's status