The following is a list of advanced settings that you can use in Harvester. You can modify the `settings.harvesterhci.io` custom resource using both the Harvester UI and the `kubectl` command.

---

## General Settings

### `additional-ca`

**Definition**: Additional trusted CA certificates that enable Harvester to access external services.

:::caution

Changing this setting might cause single-node clusters to temporarily become unavailable or inaccessible.

:::

**Default value**: None

**Example**:

```
-----BEGIN CERTIFICATE-----
SOME-CA-CERTIFICATES
-----END CERTIFICATE-----
```

### `auto-disk-provision-paths` [Experimental]

**Definition**: Setting that allows Harvester to automatically add disks that match the specified glob pattern as VM storage.

This setting only adds formatted disks that are mounted to the system. When specifying multiple patterns, separate values using commas.

:::caution

This setting is applied to **all nodes** in the cluster. All data in the storage devices **will be destroyed**.

:::

**Default value**: None

**Example**:

The following example adds disks that match the glob pattern `/dev/sd*` or `/dev/hd*`:

```
/dev/sd*,/dev/hd*
```

### `auto-rotate-rke2-certs`

**Versions**: v1.3.0 and later

**Definition**: Setting that allows you to automatically rotate certificates for RKE2 services. This setting is disabled by default.

Use the field `expiringInHours` to specify the validity period of each certificate ( `1` to `8759` hours). Harvester automatically replaces the certificate before the specified period ends.

For more information, see the **Certificate Rotation** section of the Rancher and RKE2 documentation.

**Default value**: `{"enable":false,"expiringInHours":240}`

**Example**:

```
{"enable":true,"expiringInHours":48}
```

### `backup-target`

**Definition**: Custom backup target used to store VM backups.

For more information, see the Longhorn documentation.

**Default value**: None

**Example**:

```
{
  "type": "s3",
  "endpoint": "https://s3.endpoint.svc",
  "accessKeyId": "test-access-key-id",
  "secretAccessKey": "test-access-key",
  "bucketName": "test-bup",
  "bucketRegion": "us-east-2",
  "cert": "",
  "virtualHostedStyle": false
}
```

### `cluster-registration-url`

**Definition**: URL used to import the Harvester cluster into Rancher for multi-cluster management.

When you configure this setting, a new pod called `cattle-cluster-agent-*` is created in the namespace `cattle-system` for registration purposes. This pod uses the container image `rancher/rancher-agent:related-version`, which is not packed into the Harvester ISO and is instead determined by Rancher. The `related-version` is usually the same as the Rancher version. For example, when you register Harvester to Rancher v2.7.9, the image is `rancher/rancher-agent:v2.7.9`. For more information, see Find the required assets for your Rancher version in the Rancher documentation.

Depending on your Harvester settings, the image is downloaded from either of the following locations:

- Harvester containerd-registry: You can configure a private registry for the Harvester cluster.
- Docker Hub (docker.io): This is the default option when you do not configure a private registry in Rancher.

Alternatively, you can obtain a copy of the image and manually upload it to all Harvester nodes.

**Default value**: None

**Example**:

```
https://172.16.0.1/v3/import/w6tp7dgwjj549l88pr7xmxb4x6m54v5kcplvhbp9vv2wzqrrjhrc7c_c-m-zxbbbck9.yaml
```

### `containerd-registry`

**Definition**: Configuration of a private registry created for the Harvester cluster.

The value is stored in the `registries.yaml` file of each node (path: `/etc/rancher/rke2/registries.yaml`). For more information, see Containerd Registry Configuration in the RKE2 documentation.

For security purposes, Harvester automatically removes the username and password configured for the private registry after those credentials are stored in the `registries.yaml` file.

**Example**:

 containerd-registry

```
{
  "Mirrors": {
    "docker.io": {
      "Endpoints": ["https://myregistry.local:5000"],
      "Rewrites": null
    }
  },
  "Configs": {
    "myregistry.local:5000": {
      "Auth": {
        "Username": "testuser",
        "Password": "testpassword"
      },
      "TLS": {
        "InsecureSkipVerify": false
      }
    }
  }
}
```

### `csi-driver-config`

**Versions**: v1.2.0 and later

**Definition**: Configuration necessary for using third-party CSI drivers installed in the Harvester cluster.

You must configure the following information before using features related to backups and snapshots:

- Provisioner for the installed third-party CSI driver
- `volumeSnapshotClassName` : Name of the `VolumeSnapshotClass` used to create volume snapshots or VM snapshots.
- `backupVolumeSnapshotClassName` : Name of the `VolumeSnapshotClass` used to create VM backups.

**Default value**:

```
{
  "driver.longhorn.io": {
```

```
    "volumeSnapshotClassName": "longhorn-snapshot",
    "backupVolumeSnapshotClassName": "longhorn"
  }
}
```

### `default-vm-termination-grace-period-seconds`

**Versions**: v1.2.0 and later

**Definition**: Number of seconds Harvester waits before forcibly shutting down a VM that was stopped using the Harvester UI.

Harvester sends a graceful shutdown signal to any VM that is stopped using the Harvester UI. If the graceful shutdown process is not completed within the specified number of seconds, Harvester forcibly shuts down the VM.

**Default value**: `120`

### `http-proxy`

**Definition**: HTTP proxy used to access external services, including downloading of images and backup to S3 services.

:::caution

Changing this setting might cause single-node clusters to temporarily become unavailable or inaccessible.

:::

**Default value**: `{}`

**Supported options and values**:

- Proxy URL for HTTP requests: `"httpProxy": "http://<username>:<pswd>@<ip>:<port>"`
- Proxy URL for HTTPS requests: `"httpsProxy": "https://<username>:<pswd>@<ip>:<port>"`
- Comma-separated list of hostnames and/or CIDRs: `"noProxy": "<hostname | CIDR>"`

You must specify key information in the `noProxy` field if you configured the following options or settings:

| Configured option/setting | Required value in `noProxy` | Reason |
|---|---|---|
| `httpProxy` and `httpsProxy` | Harvester node's CIDR | Not specifying the node's CIDR can break the Harvester cluster. |
| `cluster-registration-url` | Host of `cluster-registration-url` | The host information allows you to access the Harvester cluster from Rancher. |

Harvester appends necessary addresses to user-specified `noProxy` values (for example, `localhost,127.0.0.1,0.0.0.0,10.0.0.0/8,longhorn-system,cattle-system,cattle-system.svc,harvester-system,.svc,.cluster.local`). This ensures that internal traffic flows as expected.

**Example**:

```
{
  "httpProxy": "http://my.proxy",
  "httpsProxy": "https://my.proxy",
  "noProxy": "some.internal.svc,172.16.0.0/16"
}
```

### `log-level`

**Definition**: Log level for the Harvester host.

**Default value**: `info`

**Supported options and values**:

- `panic` : Least verbose log level
- `fatal`
- `error`
- `warn` , `warning`
- `info`
- `debug`
- `trace` : Most verbose log level

**Example**:

```
debug
```

### `longhorn-v2-data-engine-enabled` [Experimental]

**Versions**: v1.4.0 and later

**Definition**: Setting that enables and disables the Longhorn V2 Data Engine.

When set to `true` , Harvester automatically loads the kernel modules required by the Longhorn V2 Data Engine, and attempts to allocate 1024 × 2 MiB-sized huge pages (for example, 2 GiB of RAM) on all nodes.

Changing this setting automatically restarts RKE2 on all nodes but does not affect running virtual machine workloads.

:::tip

If you encounter error messages that include the phrase "not enough hugepages-2Mi capacity", allow some time for the error to be resolved. If the error persists, reboot the affected nodes.

To disable the Longhorn V2 Data Engine on specific nodes (for example, nodes with less processing and memory resources), go to the **Hosts** screen and add the following label to the target nodes:

- label: `node.longhorn.io/disable-v2-data-engine`
- value: `true`

:::

**Default value**: `false`

**Example**:

```
true
```

### `ntp-servers`

**Versions**: v1.2.0 and later

**Definition**: NTP servers for time synchronization on Harvester nodes.

You can define NTP servers during installation and update the addresses after installation.

:::caution

Changes to the server address list are applied to all nodes.

:::

**Default value**: ""

**Example**:

```
{
  "ntpServers": [
    "0.suse.pool.ntp.org",
    "1.suse.pool.ntp.org"
  ]
}
```

### `overcommit-config`

**Definition**: Percentage of physical compute, memory, and storage resources that can be allocated for VM use.

Overcommitting is used to optimize physical resource allocation, particularly when VMs are not expected to fully consume the allocated resources most of the time. Setting values greater than 100% allows scheduling of multiple VMs even when physical resources are notionally fully allocated.

**Default values**: `{ "cpu":1600, "memory":150, "storage":200 }`

With the default values, it would be possible to schedule the following:

- 16x the number of physical CPUs on a host
- 1.5x the amount of physical RAM on a host
- 2x the amount of physical storage in Longhorn

A VM that is configured to use 2 CPUs (equivalent to 2,000 milliCPU) can consume the full allocation as long as the resources are available. However, if the host is running heavy workloads and an overcommit value is set (for example, 1600%), Harvester only requests 125 milliCPU from the Kubernetes scheduler (2000/16 = 125 milliCPU).

**Example**:

```
{
  "cpu": 1000,
  "memory": 200,
```

```
  "storage": 300
}
```

### `additional-guest-memory-overhead-ratio`

**Definition**: The ratio to futher tune the VM `memory overhead`.

Each VM is configured with a memory value, this memory is targeted for the VM guest OS to see and use. In Harvester, the VM run in a virt-launcher pod. CPU/VM resource limits are translated and applied to the launcher pod Resource requests and limits of Pod and container. Kubevirt ensures certain amount of memory is reserved in the pod for managing the virtualization process. Harvester and KubeVirt summarize such additional memory as the VM `Memory Overhead`. The `Memory Overhead` is computed by a complex formula formula. However, sometimes the OOM(Out Of Memory) can still happen and the related VM is killed by the Harvester OS, the direct cause is that the whole POD/Container exceeds its memory limits. From practice, the `Memory Overhead` varies on different kinds of VM, different kinds of VM operating system, and also depends on the running workloads on the VM.

This setting is for more flexibly tuning the VM `Memory Overhead`.

**Default values**: `"1.5"`

Valid values: `""`, `"0"` and from `"1.0"` to `"10.0"`.

A VM that is configured to have `1 CPU, 2 Gi Memory, 1 Volume and 1 NIC` will get around `240 Mi Memory Overhead` when the ratio is `"1.0"`. When the ratio is `"1.5"`, the `Memory Overhead` is `360 Mi`. When the ratio is `"3"`, the `Memory Overhead` is `720 Mi`.

A VM that is configured to have `1 CPU, 64 Gi Memory, 1 Volume and 1 NIC` will get around `250 Mi Memory Overhead` when the ratio is `"1.0"`. The VM memory size does not have a big influence on the computing of `Memory Overhead`. The overhead of `guest OS pagetables` needs one bit for every 512b of RAM size.

The yaml output of this setting on a new cluster:

```
apiVersion: harvesterhci.io/v1beta1
default: "1.5"
kind: Setting
metadata:
  name: additional-guest-memory-overhead-ratio
value: ""
```

When the `value` field is `""`, the `default` field is used.

When the `value` field is `"0"`, the `additional-guest-memory-overhead-ratio` setting is not used, Harvester will fallback to the legacy Reserved Memory which is used in Harvester v1.3.x, v1.2.x and earlier versions. When a new VM is created and the `Reserved Memory` field on WebUI is not filled, this VM will get the `100Mi default Reserved Memory`.

If you have already set a valid value on the `spec.configuration.additionalGuestMemoryOverheadRatio` field of `kubevirt` object before Harvester v1.4.0 and then upgrade to v1.4.0, Harvester will fetch and convert it to the `value` field of this setting on the upgrade path. After that, Harvester will always use this setting to sync to the `kubevirt` object.

This setting and the VM configuration field Reserved Memroy are both taken into account to get a final `Total Memory Overhead` for each VM.

The `Total Memory Overhead` = automatically computed `Memory Overhead` + Harvester `Reserved Memory` .

The following table shows how they work together.

| VM Configured Memory | Reserved Memory | additional-guest-memory-overhead-ratio | Guest OS Memory | POD Container Memory Limit | Total Memory Overhead |
|---|---|---|---|---|---|
| 2 Gi | ""(not configured) | "0.0" | 2 Gi - 100 Mi | 2 Gi + 240 Mi | ~340 Mi |
| 2 Gi | 256 Mi | "0.0" | 2 Gi - 256 Mi | 2 Gi + 240 Mi | ~500 Mi |
| --- | --- | --- | --- | --- | --- |
| 2 Gi | ""(not configured) | "1.0" | 2 Gi | 2 Gi + 240*1.0 Mi | ~240 Mi |
| 2 Gi | ""(not configured) | "3.0" | 2 Gi | 2 Gi + 240*3.0 Mi | ~720 Mi |
| --- | --- | --- | --- | --- | --- |
| 2 Gi | ""(not configured) | "1.5" | 2 Gi | 2 Gi + 240*1.5 Mi | ~360 Mi |
| 2 Gi | 256 Mi | "1.5" | 2 Gi - 256 Mi | 2 Gi + 240*1.5 Mi | ~620 Mi |

The related information can be fetched from those objects:

```
When `additional-guest-memory-overhead-ratio` is set as "1.5".

The VM object:
...
        memory:
          guest: 2Gi    // Guest OS Memory
        resources:
          limits:
            cpu: "1"
            memory: 2Gi // VM Configured Memory

The POD object:
...
    resources:
      limits:
        cpu: "1"
        devices.kubevirt.io/kvm: "1"
```

```
        devices.kubevirt.io/tun: "1"
        devices.kubevirt.io/vhost-net: "1"
        memory: "2532309561"                // POD Container Memory Limit
```

**Example**:

```
2.0
```

:::note

To reduce the chance of hitting OOM, Harvester suggests to:

- Configure this setting with value `"2"` to give all the VMs `~480 Mi` `Memory Overhead`, if the cluster has no memory resource pressure.
- Configure the `Reserved Memory` field to have a bigger `Total Memory Overhead`, if some VMs are important. The rules based on experiences are:
  - When `VM Configured Memory` is between `5 Gi` and `10 Gi`, the `Total Memory Overhead` is `>= VM Configured Memory * 10%`.
  - When `VM Configured Memory` is greater than `10 Gi`, the `Total Memory Overhead` is `>= 1 Gi`.
  - Keep observing, tuning and testing to get the best values for each VM.
- Avoid configuring the `spec.configuration.additionalGuestMemoryOverheadRatio` field of `kubevirt` object directly.

The bigger `Total Memory Overhead` does not mean that the amount of memory is used up all the time, it is set to tolerant the peak and hence avoid hitting OOM.

There is no `one-fit-all` solution.

:::

:::important

If you have set the `Reserved Memory` field for each VM and plan to keep the legacy Reserved Memory, after the cluster is upgraded to Harvester v1.4.0, you can set the `additional-guest-memory-overhead-ratio` setting to `"0"`.

Changing the `additional-guest-memory-overhead-ratio` setting affects the VMs per following rules:

- It will affect the VMs newly created after this change immediately.
- It will not affect the running VMs immediately, those VMs will get the new `Memory Overhead` calculated from the setting after those VMs are rebooted.
- When a VM has a user configured `Reserved Memory`, this is always kept.
- When the value changes between `"0"` and the range `["", "1.0" .. "10.0"]`, the existing VMs which have the `100Mi default Reserved Memory` will keep it, the existing VMs which do not have `100Mi default Reserved Memory` will not get it automatically.

:::

### `release-download-url`

**Definition**: URL for downloading the software required for upgrades.

Harvester retrieves the ISO URL and checksum value from the `${URL}/${VERSION}/version.yaml` file that is accessible through the configured URL.

**Default value**: `https://releases.rancher.com/harvester`

**Example (version.yaml)**:

```
apiVersion: harvesterhci.io/v1beta1
kind: Version
metadata:
  name: ${VERSION}
  namespace: harvester-system
spec:
  isoChecksum: ${ISO_CHECKSUM}
  isoURL: ${ISO_URL}
```

### `server-version`

**Definition**: Version of Harvester that is installed on Harvester nodes.

**Example**:

```
v1.0.0-abcdef-head
```

### `ssl-certificates`

**Definition**: SSL certificates for the Harvester UI and API.

:::caution

Changing this setting might cause single-node clusters to temporarily become unavailable or inaccessible.

:::

**Default value**: `{}`

**Example**:

```
{
  "ca": "-----BEGIN CERTIFICATE-----\nSOME-CERTIFICATE-ENCODED-IN-PEM-FORMAT\n-----
END CERTIFICATE-----",
  "publicCertificate": "-----BEGIN CERTIFICATE-----\nSOME-CERTIFICATE-ENCODED-IN-
PEM-FORMAT\n-----END CERTIFICATE-----",
  "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nSOME-PRIVATE-KEY-ENCODED-IN-PEM-
FORMAT\n-----END RSA PRIVATE KEY-----"
}
```

### `ssl-parameters`

**Definition**: Enabled SSL/TLS protocols and ciphers of the Harvester UI and API.

:::info important

If you misconfigure this setting and are unable to access the Harvester UI and API, see Troubleshooting.

:::

**Default value**: None

**Supported options and values**:

- `protocols` : Enabled protocols.
- `ciphers` : Enabled ciphers.

For more information about the supported options, see `ssl-protocols` and `ssl-ciphers` in the Ingress-Nginx Controller documentation.

If you do not specify any values, Harvester uses `TLSv1.2` and `ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305` .

**Example**:

```
{
  "protocols": "TLSv1.2 TLSv1.3",
  "ciphers": "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305"
}
```

### `storage-network`

**Definition**: Segregated storage network for Longhorn traffic.

By default, Longhorn uses the management network, which is limited to a single interface and shared with cluster-wide workloads. If your implementation requires network segregation, you can use a storage network to isolate Longhorn in-cluster data traffic.

:::info important

Shut down all VMs before configuring this setting.

Specify an IP range in the IPv4 CIDR format. The number of IPs must be four times the number of your cluster nodes.

:::

**Default value**: ""

**Example**:

```
{
  "vlan": 100,
  "clusterNetwork": "storage",
  "range": "192.168.0.0/24"
}
```

### `support-bundle-image`

**Versions**: v1.2.0 and later

**Definition**: Support bundle image, with various versions available in [rancher/support-bundle-kit](rancher/support-bundle-kit).

**Default value**: `support-bundle-kit` image that is packed into the Harvester ISO and is specific to each Harvester release.

**Example**:

In this example, the default image tag of the cluster is `v0.0.25`.

```
{
  "repository": "rancher/support-bundle-kit",
  "tag": "v0.0.25",
  "imagePullPolicy": "IfNotPresent"
}
```

**Supported options and values**:

The value is a JSON object literal that contains the following key-value pairs:

- `repository`: Name of the repository that stores the support bundle image.
- `tag`: Tag assigned to the support bundle image.
- `imagePullPolicy`: Pull policy of the support bundle image. The supported values are `IfNotPresent`, `Always`, and `Never`. For more information, see [Image pull policy](Image pull policy) in the Kubernetes documentation.

**Notes**:

The CLI shows the following `support-bundle-image` setting object:

```
apiVersion: harvesterhci.io/v1beta1
default: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.25","imagePullPolicy":"IfNotPresent"}'  // default value,
automatically set
kind: Setting
metadata:
  name: support-bundle-image
...
status: {}
```

After some time, a newer image tag ( `v0.0.36` ) is specified in the `value` field using the Harvester UI.

```
apiVersion: harvesterhci.io/v1beta1
default: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.25","imagePullPolicy":"IfNotPresent"}'
kind: Setting
metadata:
  name: support-bundle-image
...
status: {}
value: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.36","imagePullPolicy":"IfNotPresent"}'  // your setting value
```

Eventually, this cluster is upgraded and the object changes again.

```
apiVersion: harvesterhci.io/v1beta1
default: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.38","imagePullPolicy":"IfNotPresent"}'  // default value,
automatically updated after upgrade
kind: Setting
metadata:
  name: support-bundle-image
...
status: {}
value: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.36","imagePullPolicy":"IfNotPresent"}'  // your setting value is kept
unchanged
```

The value of `tag` in the `value` field is `v0.0.36`, while the value of `tag` in the `default` field is `v0.0.38`.

To clear the outdated setting and use the default image tag, run the following command, remove the `value` field, and save the changes.

```
$ kubectl edit settings.harvesterhci.io support-bundle-image
```

The object appears as follows after the `value` field is removed.

```
apiVersion: harvesterhci.io/v1beta1
default: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.38","imagePullPolicy":"IfNotPresent"}'
kind: Setting
metadata:
  name: support-bundle-image
...
status: {}
```

The **Use the default value** button on the Harvester UI can be used to copy the contents of the `default` field to the `value` field.

The object appears as follows after the changes are saved.

```
apiVersion: harvesterhci.io/v1beta1
default: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.38","imagePullPolicy":"IfNotPresent"}'  // default
kind: Setting
metadata:
  name: support-bundle-image
...
status: {}
value: '{"repository":"rancher/support-bundle-
kit","tag":"v0.0.38","imagePullPolicy":"IfNotPresent"}'  // copied from default
```

When the cluster is upgraded in the future, the contents of the `value` field may become outdated again because the default image tag is likely to change.

:::note

- The value of `tag` in the `default` field is always based on the image that is packed into the Harvester ISO. This field is automatically updated whenever the cluster is upgraded.
- The `default` field is used when the `value` field is not set or is left empty. Harvester checks if the default image is stored in the cluster and is up-to-date.
- Configuring this setting is not required. If you decide to specify a different image tag in the `value` field, remember that this tag may become outdated when the cluster is upgraded.
- Use the command `$ kubectl edit settings.harvesterhci.io support-bundle-image` to clear the `value` field.
- The **Use the default value** button on the Harvester UI only copies the contents of the `default` field to the `value` field. You may use this to replace an outdated image tag, but the copied tag will eventually become outdated as well (when the cluster is upgraded and the `default` field is updated).
- If your cluster is in an air-gapped environment and you specified a non-default image tag in the `value` field, ensure that the image is available in the local containerd registry. Harvester is unable to generate a support bundle if the image is not available.

:::

### `support-bundle-namespaces`

**Versions**: v1.2.0 and later

**Definition**: Additional namespaces that you can use when generating a support bundle.

By default, the support bundle only collects resources from the following predefined namespaces:

- cattle-dashboards
- cattle-fleet-local-system
- cattle-fleet-system
- cattle-fleet-clusters-system
- cattle-monitoring-system
- fleet-local
- harvester-system
- local
- longhorn-system
- cattle-logging-system

Namespaces that you select are appended to the predefined namespaces list.

**Default value**: None

### `support-bundle-timeout`

**Versions**: v1.2.0 and later

**Definition**: Number of minutes Harvester allows for the completion of the support bundle generation process.

The process is considered to have failed when the data collection and file packing tasks are not completed within the configured number of minutes. Harvester does not continue or retry support bundle generation processes that have timed out. When the value is `0`, the timeout feature is disabled.

**Default value**: `10`

### `support-bundle-expiration`

**Versions**: v1.3.0 and later

**Definition**: Number of minutes Harvester waits before deleting a support bundle that has been packaged but not downloaded (either deliberately or unsuccessfully) or retained.

You can specify a value greater than or equal to 0. When the value is 0, Harvester uses the default value.

**Default value**: `30`

### `support-bundle-node-collection-timeout`

**Versions**: v1.3.1 and later

**Definition**: Number of minutes Harvester allows for collection of logs and configurations (Harvester) on the nodes for the support bundle.

If the collection process is not completed within the allotted time, Harvester still allows you to download the support bundle (without the uncollected data). You can specify a value greater than or equal to 0. When the value is 0, Harvester uses the default value.

**Default value**: `30`

### `upgrade-checker-enabled`

**Definition**: Setting that automatically checks for available Harvester upgrades.

**Default value**: `true`

**Example**:

```
false
```

### `upgrade-checker-url`

**Definition**: URL used to check for available Harvester upgrades.

This setting can only be used if the `upgrade-checker-enabled` setting is set to `true`.

**Default value**: `https://harvester-upgrade-responder.rancher.io/v1/checkupgrade`

**Example**:

```
https://your.upgrade.checker-url/v99/checkupgrade
```

### `upgrade-config`

**Definition**: Upgrade-related configuration.

**Default value**: `{"imagePreloadOption":{"strategy":{"type":"sequential"}}, "restoreVM": false}`

**Supported options and fields**:

- `imagePreloadOption` : Options for the image preloading phase.

  The full ISO contains the core operating system components and all required container images. Harvester can preload these container images to each node during installation and upgrades. When workloads are scheduled to management and worker nodes, the container images are ready to use.

- `strategy` : Image preload strategy.

- `type` : Type of image preload strategy.

  - `sequential` : Harvester preloads the container images from the target ISO to each node. This is the default option.

  - `skip` : Harvester does not preload the container images from the target ISO to each node. **Do not use this option in production environments.**

    :::info important

    If you decide to use `skip` , ensure that the following requirements are met:

    - You have a private container registry that contains all required images.
    - Your cluster has high-speed internet access and is able to pull all images from Docker Hub when necessary.

    Note any potential internet service interruptions and how close you are to reaching your Docker Hub rate limit. Failure to download any of the required images may cause the upgrade to fail and may leave the cluster in a middle state.

    :::

  - `parallel` (**experimental**): Nodes preload images in batches. You can adjust this using the `concurrency` option.

- `concurrency` : Number of nodes that can simultaneously preload images. This option takes effect only when `type` is set to `parallel` .

  The default value is `0` , which is equivalent to following the cluster's node counts. Using `0` allows the system to dynamically follow the scale of the cluster. Values higher than the cluster's node counts are treated as `0` , while lower values are considered invalid and are rejected by Harvester.

  :::note

  Harvester deploys an upgrade-repo service on the cluster that serves as an HTTP server for nodes that need to preload the container images. When a `concurrency` value is set, each batch of nodes downloads the container images from this upgrade-repo in parallel. Because of this, you must consider the speed of the Harvester management network and the read speed of the default disk for Longhorn.

  :::

- `restoreVM`: Option that enables Harvester to automatically restore running VMs after a single-node cluster is upgraded. The default value is `false`, which causes all VMs to be stopped after the upgrade is completed. When set to `true`, Harvester restarts VMs that were running before the upgrade was started. VMs that were paused before the upgrade are not restarted.

**Example**:

```
{
  "imagePreloadOption": {
    "strategy": {
      "type": "parallel",
      "concurrency": 2
    }
  },
  "restoreVM": true
}
```

### `vip-pools`

**Versions**: Deprecated as of v1.2.0 (Use IP pools instead.)

**Definition**: Global or namespace-specific IP address pools of the VIP by CIDR or IP range.

**Default value**: `{}`

**Example**:

```
{
  "default": "172.16.0.0/24,172.16.1.0/24",
  "demo": "172.16.2.50-172.16.2.100,172.16.2.150-172.16.3.200"
}
```

### `vm-force-reset-policy`

**Definition**: Setting that allows you to force rescheduling of a VM when the node that it is running on becomes unavailable.

When the state of the node changes to `Not Ready`, the VM is force deleted and rescheduled to an available node after the configured number of seconds.

When the node becomes unavailable or is powered off, the VM only restarts and does not migrate.

**Default value**: `{"enable":true, "period":300}`

**Example**:

```
{
  "enable": "true",
  "period": 300
}
```

### `volume-snapshot-class`

**Definition**: VolumeSnapshotClassName for the VolumeSnapshot and VolumeSnapshotContent when restoring a VM to a namespace that does not contain the source VM.

**Default value**: `longhorn`

**Example**:

`longhorn`

---

## UI Settings

### `branding`

**Versions**: v1.2.0 and later

**Definition**: Setting allows you to globally rebrand the Harvester UI by customizing the product name, logos, and color scheme.

**Default value**: **Harvester**

:::caution

Because this setting is part of the `settings.management.cattle.io` custom resource, you cannot configure any of the supported options (for example, `Logo` and `Primary Color`) using a Harvester configuration file.

:::

 containerd-registry

**Supported options and values**:

- **Private Label**: Product name or other text that replaces "Harvester" in most locations on the Harvester UI.
- **Logo**: Logo image in the top-level navigation header. You must upload logos for both light and dark modes.
- **Favicon**: Small image displayed next to the page title in the browser tab.
- **Primary Color**: Main color used throughout the Harvester UI.
- **Link Color**: Color used for link text throughout the Harvester UI.

### `ui-index`

**Definition**: HTML index location for the Harvester UI.

**Default value**: `https://releases.rancher.com/harvester-ui/dashboard/latest/index.html`

**Example**:

```
https://your.static.dashboard-ui/index.html
```

### `ui-path`

**Definition**: Path that describes the location of `index.html`, which is used to access the Harvester UI.

`ui-path` serves as the entry point to the Harvester UI and is active only in the following situations:

- The value of `ui-source` is `bundled`.
- The value of `ui-source` is `auto`, but `ui-index` is unable to retrieve the HTML file.

**Default value**: `/usr/share/harvester/harvester`

**Examples**:

`index.html` is stored in a container in `/home/samplefolder`. The value of `ui-source` is `bundled`.

Scenario 1: The value of `ui-path` is `/home/samplefolder`. Whenever you access the Harvester UI, the content of `/home/samplefolder/index.html` is displayed.

Scenario 2: The value of `ui-index` points to a page that is unavailable or non-existent (for example, `notexist-example.com/index.html`). When you access the Harvester UI for the first time, the content of `/home/samplefolder/index.html` is displayed. However, if you modify the `ui-index` setting to use the default value and access the Harvester UI again, the content of `/home/samplefolder/index.html` is still displayed (even if the new `ui-index` value points to an available page). For more information, see Issue #6066.

### `ui-plugin-index`

**Definition**: JavaScript address for the Harvester plugin (when accessing Harvester from Rancher).

**Default value**: `https://releases.rancher.com/harvester-ui/plugin/harvester-latest/harvester-latest.umd.min.js`

**Example**:

```
https://your.static.dashboard-ui/*.umd.min.js
```

### `ui-source`

**Definition**: Setting that allows you to configure how to load the UI source.

**Default value**: `auto`

**Supported values**:

- `auto`: Automatically detects whether to use the bundled UI or not.
- `external`: Uses the external UI source.
- `bundled`: Uses the bundled UI source.

**Example**:

```
external
```