Available as of v1.1.0

With the vm-import-controller addon users can import their virtual machines from VMware and OpenStack into Harvester.

To use the VM import feature, users need to enable the vm-import-controller addon.

By default, vm-import-controller leverages ephemeral storage, which is mounted from /var/lib/kubelet.

During the migration, a large VM's node could run out of space on this mount, resulting in subsequent scheduling failures.

To avoid this, users are advised to enable PVC-backed storage and customize the amount of storage needed. According to the best practice, the PVC size should be twice the size of the largest VM being migrated. This is essential as the PVC is used as scratch space to download the VM, and convert the disks into raw image files.

vm-import-controller

Currently, the following source providers are supported:

- VMware
- OpenStack

API

The vm-import-controller introduces two CRDs.

Sources

Sources allow users to define valid source clusters.

For example:

```
apiVersion: migration.harvesterhci.io/v1beta1
kind: VmwareSource
metadata:
  name: vcsim
  namespace: default
spec:
  endpoint: "https://vscim/sdk"
  dc: "DCO"
  credentials:
    name: vsphere-credentials
    namespace: default
```

The secret contains the credentials for the vCenter endpoint:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: vsphere-credentials
  namespace: default
stringData:
  "username": "user"
  "password": "password"
```

As part of the reconciliation process, the controller will log into vCenter and verify whether the `dc` specified in the source spec is valid.

Once this check is passed, the source is marked as ready and can be used for VM migrations.

```
$ kubectl get vmwaresource.migration
NAME      STATUS
vcsim   clusterReady
```

For OpenStack-based source clusters, an example definition is as follows:

```yaml
apiVersion: migration.harvesterhci.io/v1beta1
kind: OpenstackSource
metadata:
  name: devstack
  namespace: default
spec:
  endpoint: "https://devstack/identity"
  region: "RegionOne"
  credentials:
    name: devstack-credentials
    namespace: default
```

The secret contains the credentials for the OpenStack endpoint:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: devstack-credentials
  namespace: default
stringData:
  "username": "user"
  "password": "password"
  "project_name": "admin"
  "domain_name": "default"
  "ca_cert": "pem-encoded-ca-cert"
```

The OpenStack source reconciliation process attempts to list VMs in the project and marks the source as ready.

```
$ kubectl get opestacksource.migration
NAME       STATUS
devstack   clusterReady
```

VirtualMachineImport

The VirtualMachineImport CRD provides a way for users to define a source VM and map to the actual source cluster to perform VM export/import.

A sample VirtualMachineImport looks like this:

```yaml
apiVersion: migration.harvesterhci.io/v1beta1
kind: VirtualMachineImport
metadata:
  name: alpine-export-test
  namespace: default
spec:
  virtualMachineName: "alpine-export-test"
  folder: "Discovered VM"
  networkMapping:
  - sourceNetwork: "dvSwitch 1"
    destinationNetwork: "default/vlan1"
  - sourceNetwork: "dvSwitch 2"
    destinationNetwork: "default/vlan2"
  storageClass: "my-storage-class"
  sourceCluster:
    name: vcsim
    namespace: default
    kind: VmwareSource
    apiVersion: migration.harvesterhci.io/v1beta1
```

This will trigger the controller to export the VM named "alpine-export-test" on the VMware source cluster to be exported, processed and recreated into the Harvester cluster.

This can take a while based on the size of the virtual machine, but users should see `VirtualMachineImages` created for each disk in the defined virtual machine.

If the source virtual machine is placed in a folder, you can specify the folder name in the optional `folder` field.

The list of items in `networkMapping` will define how the source network interfaces are mapped to the Harvester Networks.

If a match is not found, each unmatched network interface is attached to the default `managementNetwork` .

The `storageClass` field specifies the StorageClass to be used for images and provisioning persistent volumes during the import process. If not specified, the default StorageClass will be used.

Once the virtual machine has been imported successfully, the object will reflect the status:

```
$ kubectl get virtualmachineimport.migration
NAME                    STATUS
```

```
alpine-export-test      virtualMachineRunning
openstack-cirros-test   virtualMachineRunning
```

Similarly, users can define a VirtualMachineImport for an OpenStack source as well:

```
apiVersion: migration.harvesterhci.io/v1beta1
kind: VirtualMachineImport
metadata:
  name: openstack-demo
  namespace: default
spec:
  virtualMachineName: "openstack-demo" #Name or UUID for instance
  networkMapping:
  - sourceNetwork: "shared"
    destinationNetwork: "default/vlan1"
  - sourceNetwork: "public"
    destinationNetwork: "default/vlan2"
  sourceCluster:
    name: devstack
    namespace: default
    kind: OpenstackSource
    apiVersion: migration.harvesterhci.io/v1beta1
```

:::note OpenStack allows users to have multiple instances with the same name. In such a scenario, users are advised to use the Instance ID. The reconciliation logic tries to perform a name-to-ID lookup when a name is used. :::

Known issues

- **Source virtual machine name is not RFC1123 compliant**: When creating a virtual machine object, the vm-import-controller add-on uses the name of the source virtual machine, which may not meet the Kubernetes object naming criteria. You may need to rename the source virtual machine to allow successful completion of the import.