

DEVCONF.cz

**Case Study:
Volume Populators for Virtual Disks?**

Arik Hadas

Associate Manager and Principal Software Engineer

My journey at Red Hat

I have been working at Red Hat for over a decade:

- 2012 - 2018 oVirt / Red Hat Virtualization
- 2018 - 2019 KubeVirt / OpenShift Virtualization
- 2019 - 2020 Other (non-virtualization) OpenShift areas
- 2020 - 2022 oVirt / Red Hat Virtualization
- 2022 - Forklift / Migration Toolkit for Virtualization



Forklift / MTV

What is it about?

Forklift is an extension to Kubernetes that migrates virtual machines to KubeVirt / OpenShift Virtualization from traditional virtualization platforms

Forklift 2.4:

- vSphere
- oVirt / Red Hat Virtualization
- OpenStack / Red Hat OpenStack Platform

Planned in Forklift 2.5:

- OVA
- KubeVirt to KubeVirt

Designed for Mass Migration of VMs



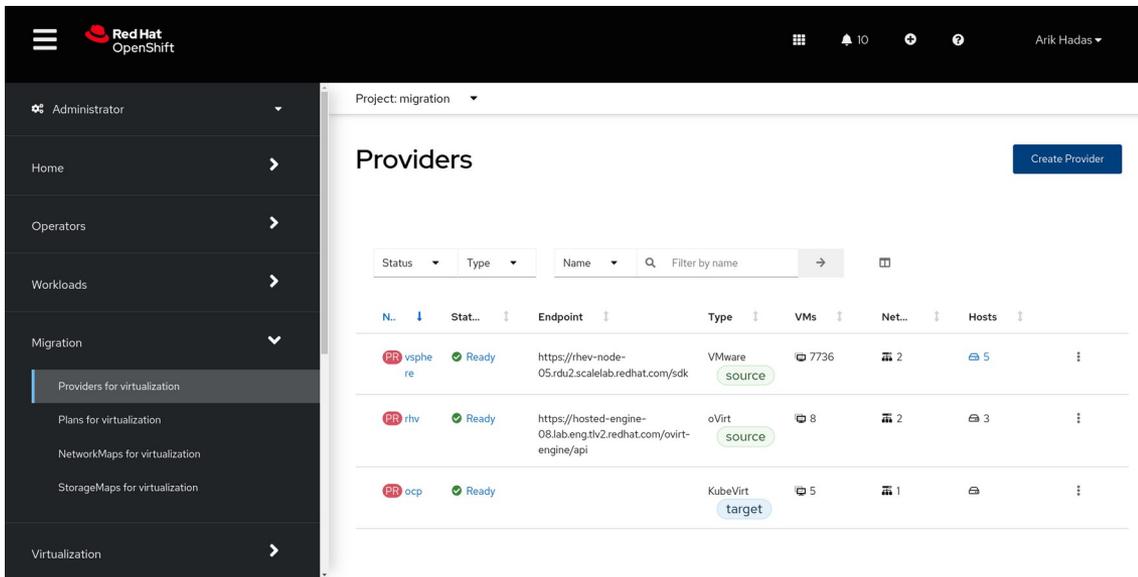
"Migrate virtual machines at scale to OpenShift in a few simple steps. Provide source and destination credentials, map infrastructure, and create migration plans"

MTV's mission statement

How to start a migration in Forklift

Few simple steps using an easy to use UI

- Define a source provider
 - Select virtual machines
 - Set Network mappings
 - Set Storage mappings
- Start the migration plan



Project: migration

Providers

Create Provider

Status Type Name Filter by name

| N. | Stat... | Endpoint | Type | VMs | Net... | Hosts |
|----------|---------|-------------------------------------------------------------------|-----------------|------|--------|-------|
| PR vsphe | Ready | https://hev-node-05.rdu2.scalelab.redhat.com/sdk | VMware source | 7736 | 2 | 5 |
| PR rhv | Ready | https://hosted-engine-08.lab.eng.tlv2.redhat.com/ovirt-engine/api | oVirt source | 8 | 2 | 3 |
| PR ocp | Ready | | KubeVirt target | 5 | 1 | |

How to start a migration in Forklift

Few simple steps using an easy to use UI

- **Define a source provider**
- Define a target provider (optional)
- Create a migration plan
 - Select virtual machines
 - Set Network mappings
 - Set Storage mappings
- Start the migration plan

The screenshot shows the 'Create provider' dialog in the Forklift UI. The dialog is titled 'Create provider' and has a close button (X). It contains the following fields and options:

- Provider namespace**: mtv-24-demo
- Provider type**: VMware (selected from a dropdown)
- Provider name**: vsphere (text input field)
- vCenter server hostname or IP address**: rhev-node-13.rdu2.scalelab.redhat.com
- vCenter username**: administrator@vsphere.local
- vCenter password**: masked with dots and an eye icon to toggle visibility
- VDDK init image**: age-registry.openshift-image-registry.svc:5000/openshift/vddk:7032
- Skip certificate validation**: checked checkbox (if checked, the provider's certificate won't be validated)
- SHA-1 fingerprint**: 31:14:EB:9E:F1:78:68:10:A5:78:D1:A7:DF:BB:54:B7:1B:91:9F:30

At the bottom of the dialog are 'Create' and 'Cancel' buttons. In the background, the 'Providers' table is visible with a 'local' provider in a 'Ready' state.

How to start a migration in Forklift

Few simple steps using an easy to use UI

- Define a source provider
- Define a target provider (optional)
- **Create a migration plan**
 - Select virtual machines
 - Set Network mappings
 - Set Storage mappings
- Start the migration plan

Migration plans > Create

Create migration plan

- 1 General
- 2 VM selection
 - Filter
 - Select VMs
- 3 Network mapping
- 4 Storage mapping
- 5 Type
- 6 Hooks
- 7 Review

General settings

Give your plan a name and a description

Plan resource namespace ⓘ
mtv-24-demo

Plan name *
demo-import-from-vmware

Plan description

Select source and target providers

Source provider *
vsphere

Target provider *
local

Target namespace *

Next Back Cancel

How to start a migration in Forklift

Few simple steps using an easy to use UI

- Define a source provider
- Define a target provider (optional)
- Create a migration plan
 - Select virtual machines
 - Set Network mappings
 - Set Storage mappings
- Start the migration plan

Migration plans > Create

Create migration plan

- 1 General
- 2 VM selection
 - Filter
 - Select VMs
- 3 Network mapping
- 4 Storage mapping
- 5 Type
- 6 Hooks
- 7 Review

| | Migr... | VM... | Data... | Clus... | Host | Folder path |
|---|----------------------------------------------------------|-------------------------------------------------|------------|---------|----------------|----------------|
| > | <input checked="" type="checkbox"/> Ok | <input checked="" type="checkbox"/> mtv-rhel... | Datacenter | MTV | f01-h06-00... | mtv-func-qe... |
| ▼ | <input checked="" type="checkbox"/> Warning | <input checked="" type="checkbox"/> mtv-rhel... | Datacenter | MTV | f01-h27-000... | mtv-func-qe... |

Conditions have been identified that make this VM a **moderate risk** to migrate.

Warning Changed Block Tracking (CBT) not enabled: Changed Block Tracking (CBT) has not been enabled on this VM. This feature is a prerequisite for VM warm migration.

See the [product documentation](#) for more information.

| | | | | | | |
|---|-----------------------------------------------|------------------------------------------------|------------|-----|----------------|----------------|
| > | <input type="checkbox"/> Ok | <input checked="" type="checkbox"/> mtv-win... | Datacenter | MTV | f01-h27-000... | mtv-func-qe... |
| > | <input type="checkbox"/> Ok | <input checked="" type="checkbox"/> mtv-win... | Datacenter | MTV | f01-h27-000... | mtv-func-qe... |
| > | <input type="checkbox"/> Warning | <input checked="" type="checkbox"/> mtv-win... | Datacenter | MTV | f01-h27-000... | mtv-func-qe... |

[Next](#) [Back](#) [Cancel](#)

How to start a migration in Forklift

Few simple steps using an easy to use UI

- Define a source provider
- Define a target provider (optional)
- Create a migration plan
 - Select virtual machines
 - **Set Network mappings**
 - Set Storage mappings
- Start the migration plan

The screenshot shows the 'Create migration plan' interface in Forklift. The left sidebar contains a progress indicator with seven steps: 1. General, 2. VM selection, 3. Filter, 4. Select VMs, 5. Network mapping (highlighted), 6. Storage mapping, 7. Type, 8. Hooks, and 9. Review. The main content area is titled 'Create migration plan' and 'Network mapping'. It includes a dropdown menu to 'Create a network mapping', a text area for instructions, and a mapping diagram. The diagram shows a 'Source networks' box containing 'VM Network' and '/Datacenter/network/VM Network' connected by an arrow to a 'Target namespaces / networks' dropdown menu. The dropdown menu is open, showing 'Select target...' and 'Pod network (default)'. A 'Remove' button is located below the dropdown. At the bottom, there are 'Next', 'Back', and 'Cancel' buttons.

How to start a migration in Forklift

Few simple steps using an easy to use UI

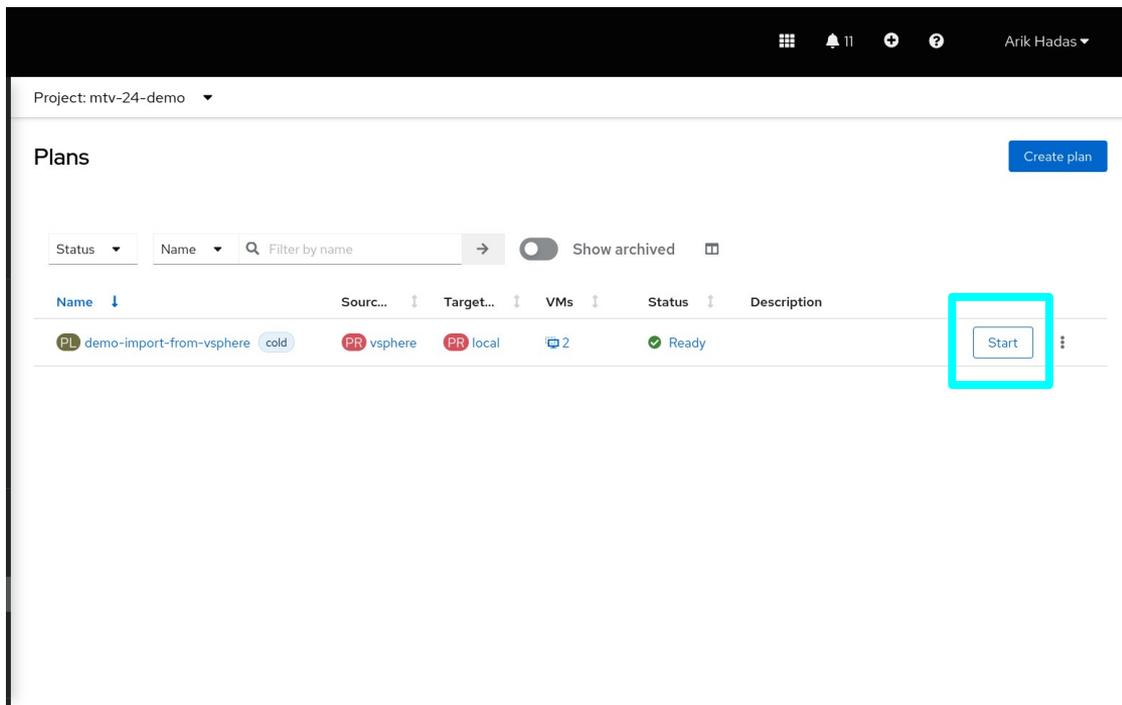
- Define a source provider
- Define a target provider (optional)
- Create a migration plan
 - Select virtual machines
 - Set Network mappings
 - **Set Storage mappings**
- Start the migration plan

The screenshot shows the 'Create migration plan' interface in the Forklift console. The left sidebar contains a navigation menu with steps: 1 General, 2 VM selection, 3 Network mapping, 4 Storage mapping (selected), 5 Type, 6 Hooks, and 7 Review. The main content area is titled 'Create migration plan' and has a sub-section 'Storage mapping'. It instructs the user to 'Select an existing storage mapping to modify or create a new storage mapping.' Below this is a dropdown menu with the option 'Create a storage mapping'. The next instruction is 'Map source datastores to target storage classes.' Under 'Source datastores', there is a box containing 'rhv-v2v-performance-testing' and its path '/Datacenter/datastore/rhv-v2v-performance-testing'. An arrow points from this box to a 'Target storage classes' dropdown menu. The dropdown menu is open, showing a search field 'Select target...' and a list of storage classes: 'localblock-sc', 'ocs-storagecluster-ceph-rbd', 'ocs-storagecluster-ceph-rgw', 'ocs-storagecluster-cephfs', and 'openshift-storage.noobaa.io'. A 'Remove' button is visible next to the dropdown. At the bottom of the page, there are 'Next', 'Back', and 'Cancel' buttons.

How to start a migration in Forklift

Few simple steps using an easy to use UI

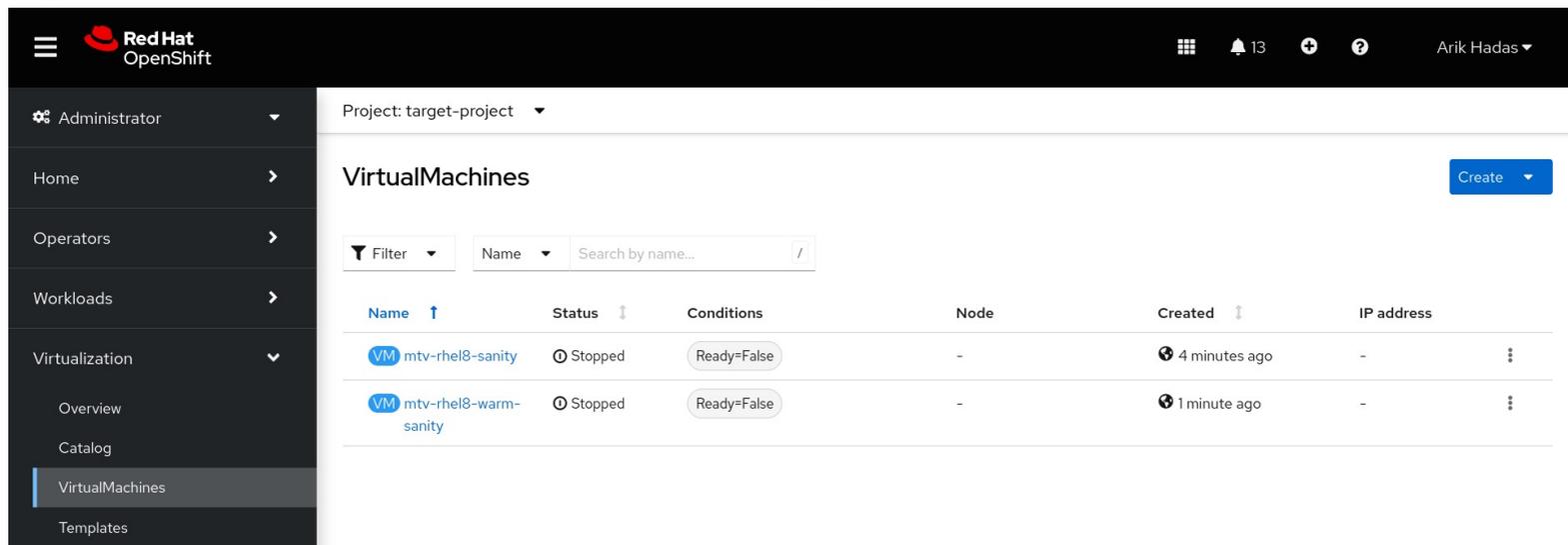
- Define a source provider
- Define a target provider (optional)
- Create a migration plan
 - Select virtual machines
 - Set Network mappings
 - Set Storage mappings
- **Start the migration plan**



The screenshot displays the Forklift web interface for a project named 'mtv-24-demo'. The main section is titled 'Plans' and features a 'Create plan' button in the top right. Below this, there are search and filter controls, including a 'Filter by name' search bar and a 'Show archived' toggle. A table lists migration plans with columns for Name, Source, Target, VMs, Status, and Description. One plan is visible: 'demo-import-from-vsphere' with a 'cold' tag, source 'vsphere', target 'local', 2 VMs, and a 'Ready' status. A red 'Start' button is highlighted with a red box in the rightmost column of this row. The top right of the interface shows user information 'Arik Hadas' and system icons.

Finally: virtual machines in KubeVirt

The migrated VMs can start in containers within the **target** k8s / OpenShift cluster



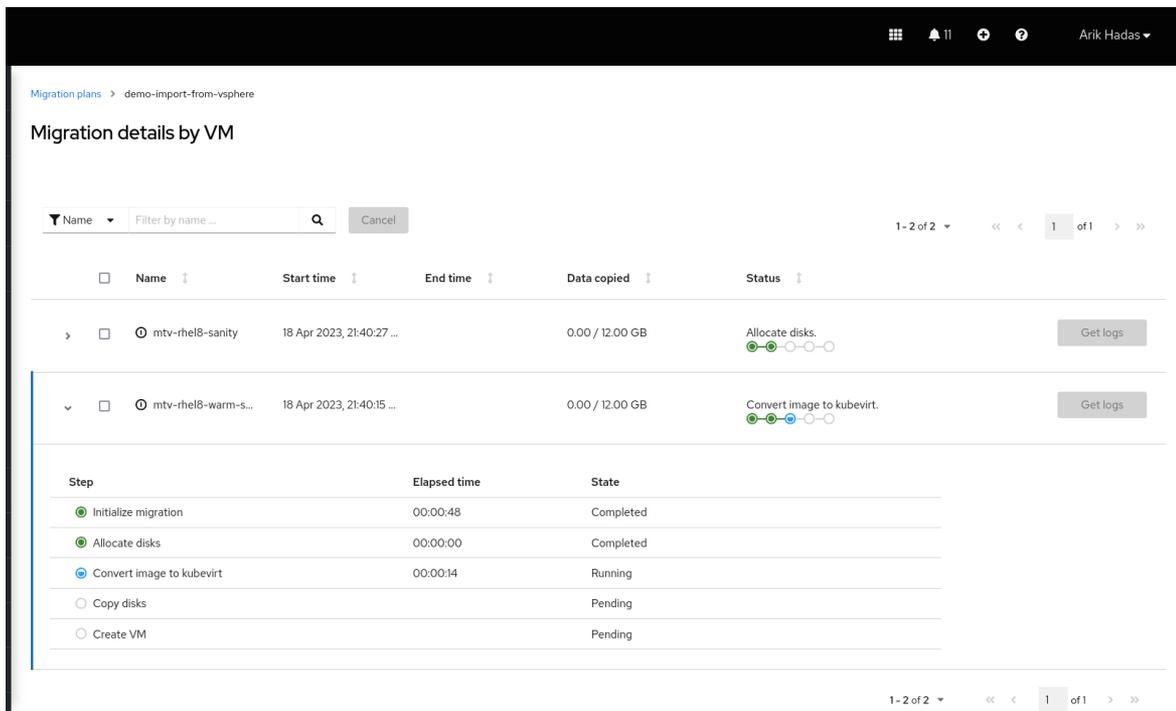
The screenshot displays the Red Hat OpenShift console interface. The top navigation bar includes the Red Hat OpenShift logo, a hamburger menu, and user information for 'Arik Hadas'. The left sidebar shows navigation options: Administrator, Home, Operators, Workloads, Virtualization (selected), Overview, Catalog, VirtualMachines (highlighted), and Templates. The main content area is titled 'VirtualMachines' and shows a table of VMs in the 'target-project' namespace. Two VMs are listed, both in a 'Stopped' state with 'Ready=False' conditions.

| Name ↑ | Status ↓ | Conditions | Node | Created ↓ | IP address |
|--------------------------|-----------|-------------|------|-----------------|------------|
| VM mtv-rhel8-sanity | ⊘ Stopped | Ready=False | - | 🕒 4 minutes ago | - |
| VM mtv-rhel8-warm-sanity | ⊘ Stopped | Ready=False | - | 🕒 1 minute ago | - |

But what happens in between?

What is going on during the migration

- Disks are converted (optional)
- Disks are copied
- VM configuration is converted



Migration plans > demo-import-from-vsphere

Migration details by VM

▼ Name Filter by name ... 1-2 of 2 << < 1 of 1 > >>

| <input type="checkbox"/> | Name | Start time | End time | Data copied | Status | <input type="button" value="Get logs"/> |
|--------------------------|----------------------------------------------|---------------------------|----------|-----------------|--------------------------------------|-----------------------------------------|
| > | <input type="checkbox"/> mtv-rhel8-sanity | 18 Apr 2023, 21:40:27 ... | | 0.00 / 12.00 GB | Allocate disks. ●●○○○○ | <input type="button" value="Get logs"/> |
| ▼ | <input type="checkbox"/> mtv-rhel8-warm-s... | 18 Apr 2023, 21:40:15 ... | | 0.00 / 12.00 GB | Convert image to kubevirt. ●●●●○○ | <input type="button" value="Get logs"/> |

| Step | Elapsed time | State |
|-----------------------------|--------------|-----------|
| ● Initialize migration | 00:00:48 | Completed |
| ● Allocate disks | 00:00:00 | Completed |
| ● Convert image to kubevirt | 00:00:14 | Running |
| ○ Copy disks | | Pending |
| ○ Create VM | | Pending |

1-2 of 2 << < 1 of 1 > >>

But what happens in between?

Our focus today

- **Disks are converted (optional)**
- **Disks are copied**
- VM configuration is converted

The screenshot displays a web interface for managing migration plans. The main heading is "Migration details by VM". Below this, there is a search bar and a table listing migration tasks. The table has columns for Name, Start time, End time, Data copied, and Status. Two tasks are visible: "mtv-rhel8-sanity" and "mtv-rhel8-warm-s...". The second task is expanded to show a detailed step-by-step progress table.

| Name | Start time | End time | Data copied | Status |
|---------------------|---------------------------|----------|-----------------|----------------------------|
| mtv-rhel8-sanity | 18 Apr 2023, 21:40:27 ... | | 0.00 / 12.00 GB | Allocate disks. |
| mtv-rhel8-warm-s... | 18 Apr 2023, 21:40:15 ... | | 0.00 / 12.00 GB | Convert image to kubevirt. |

| Step | Elapsed time | State |
|---------------------------|--------------|-----------|
| Initialize migration | 00:00:48 | Completed |
| Allocate disks | 00:00:00 | Completed |
| Convert image to kubevirt | 00:00:14 | Running |
| Copy disks | | Pending |

The image features a vibrant purple background with several overlapping, semi-transparent circular shapes in various shades of purple, from light lavender to deep indigo. The text "Volume populators" is centered in a clean, white, sans-serif font.

Volume populators

The volume populators feature

Setting up PVC and data source CR



```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
  dataSourceRef:
    apiGroup: hello.example.com
    kind: Hello
    name: example-hello
  volumeMode: Filesystem
```

```
apiVersion: hello.example.com/v1alpha1
kind: Hello
metadata:
  name: example-hello
spec:
  fileName: example.txt
  fileContents: Hello, world!
```



The volume populators feature

Let's use this PVC

```
apiVersion: batch/v1
kind: Job
metadata:
  name: example-job
spec:
  template:
    spec:
      containers:
        - name: example-container
          image: busybox:latest
          command:
            - cat
            - /mnt/example.txt
          volumeMounts:
            - name: vol
              mountPath: /mnt
      restartPolicy: Never
      volumes:
        - name: vol
          persistentVolumeClaim:
            claimName: example-pvc
```

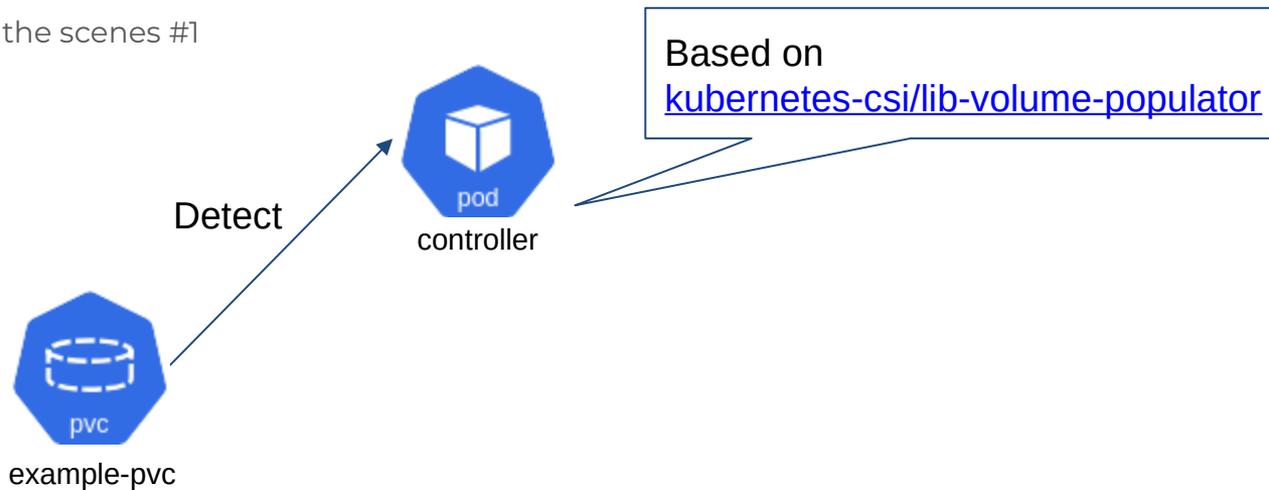
```
kubectl wait --for=condition=Complete job/example-job
```

```
kubectl logs job/example-job
```

```
Hello, world!
```

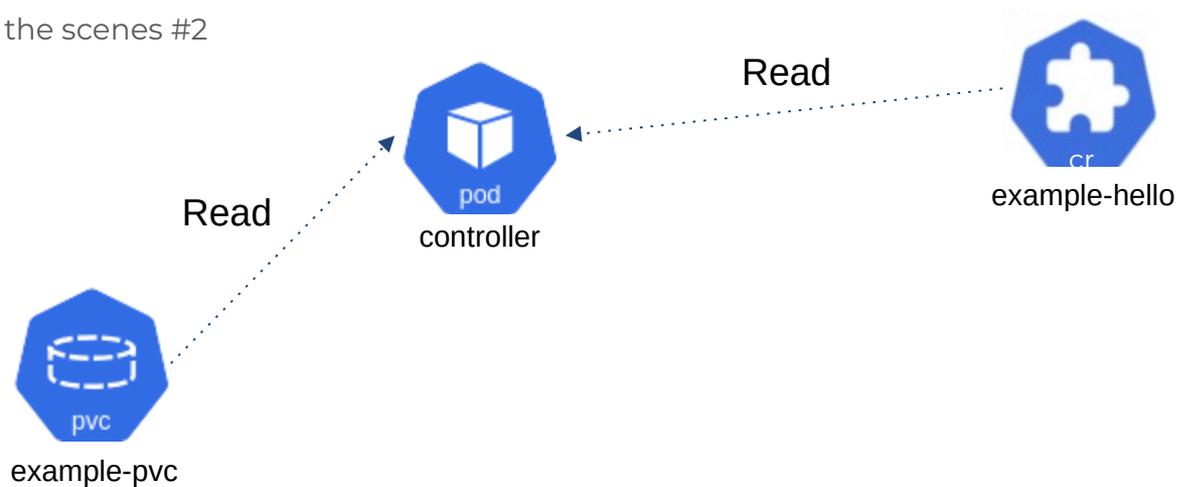
The volume populators feature

What happened behind the scenes #1



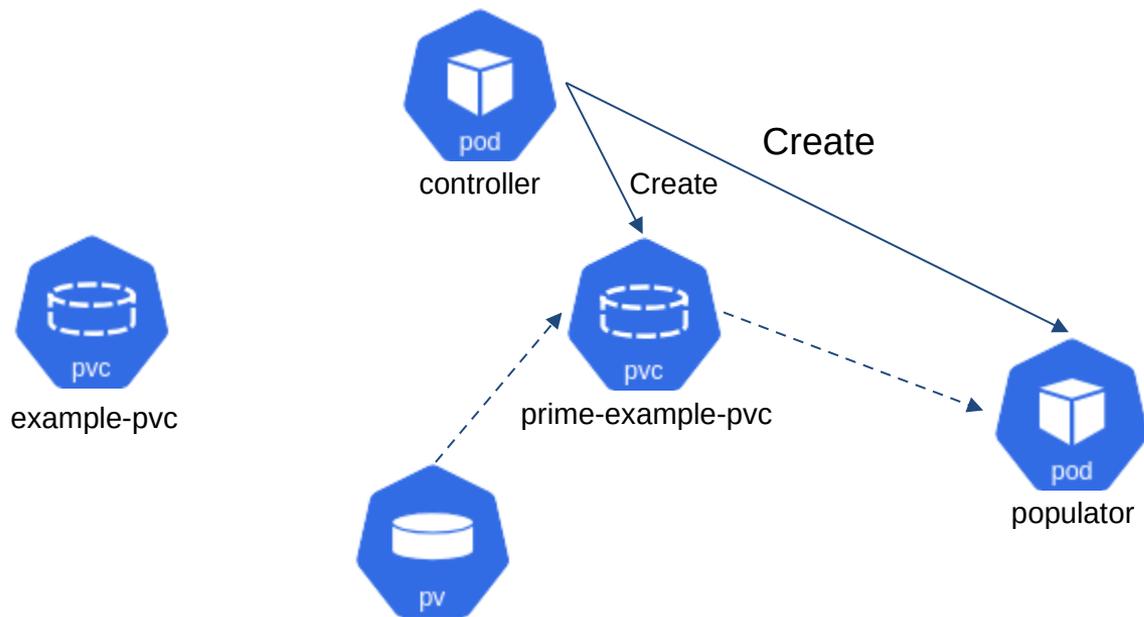
The volume populators feature

What happened behind the scenes #2



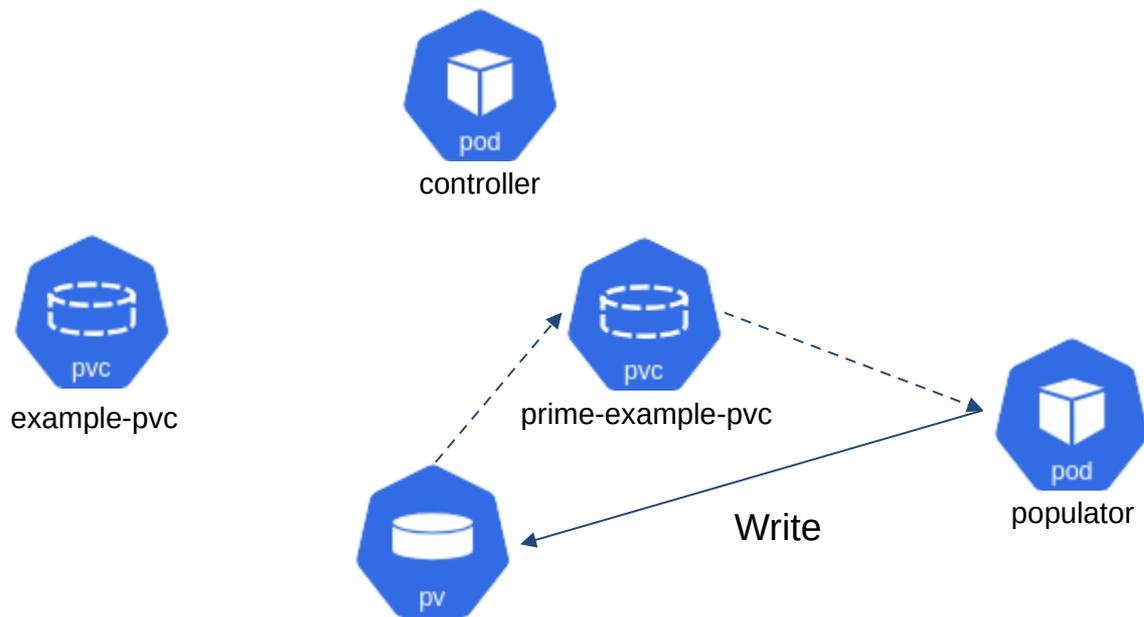
The volume populators feature

What happened behind the scenes #3



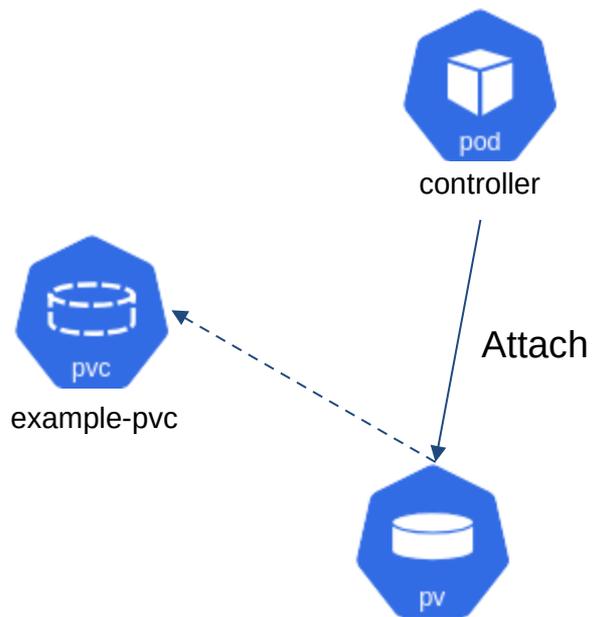
The volume populators feature

What happened behind the scenes #4

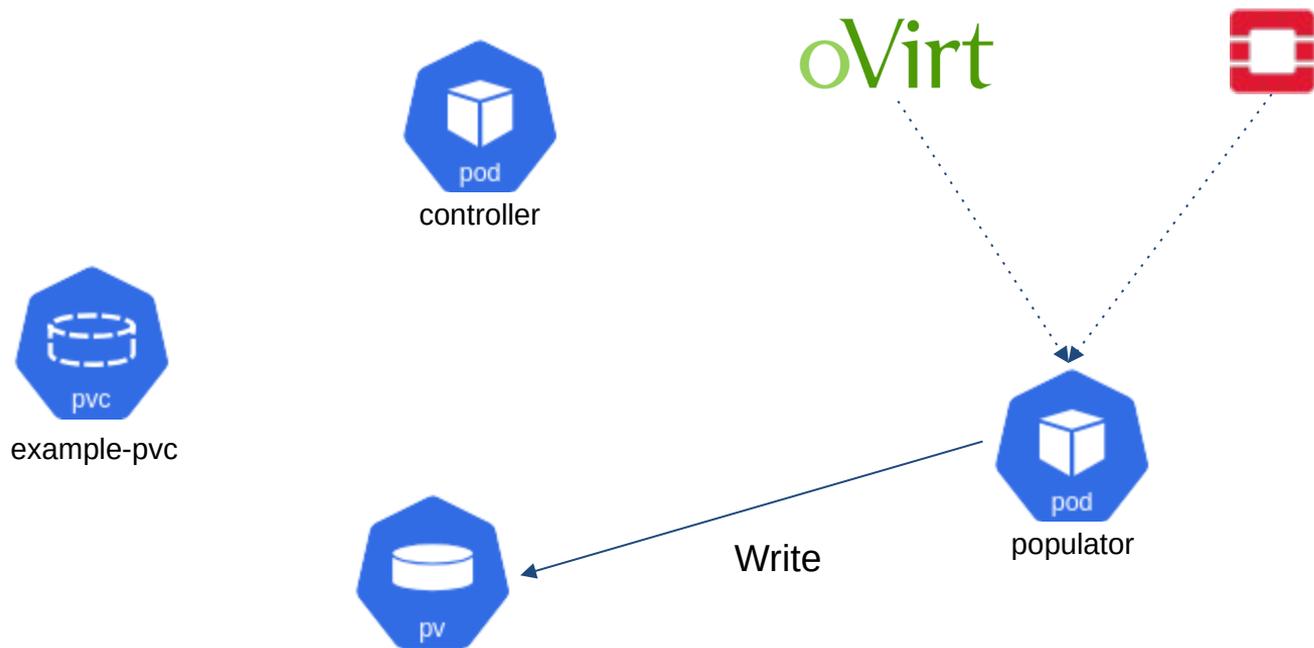


The volume populators feature

What happened behind the scenes #5



Volume populators in Forklift

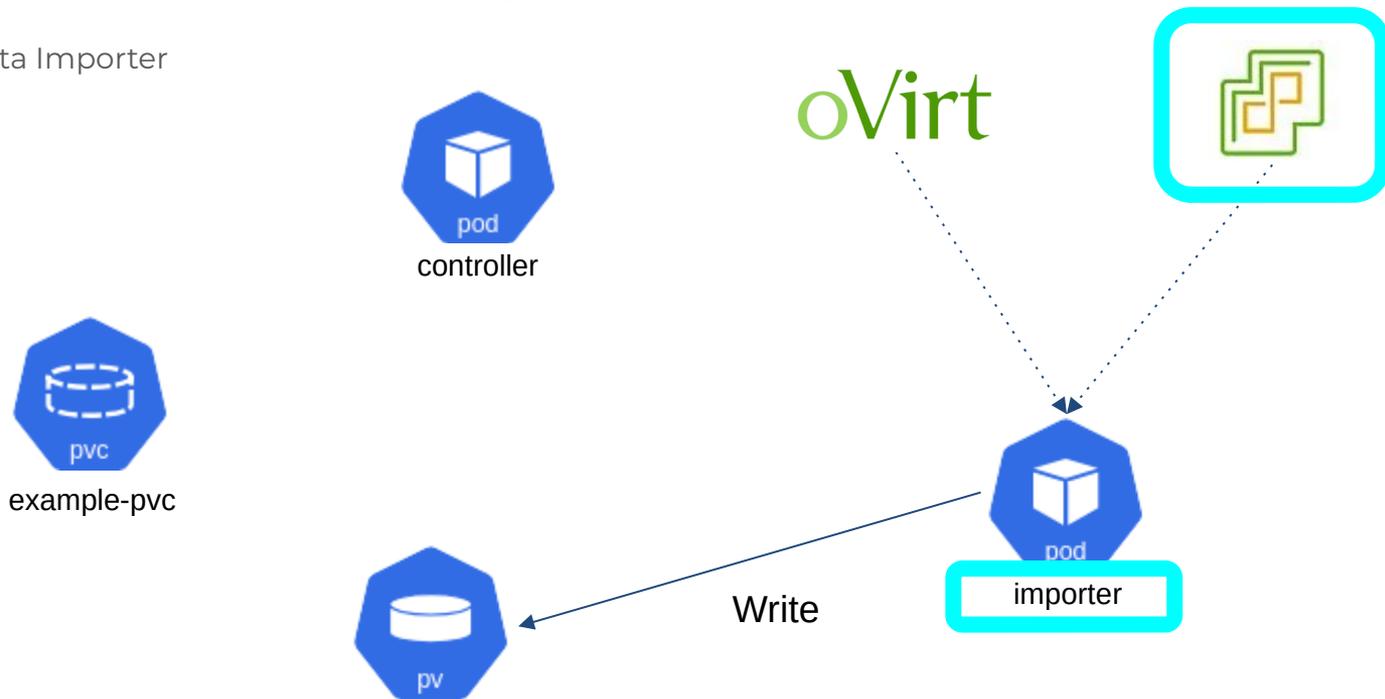


Looks familiar?



Implementation with CDI

CDI == Containerized Data Importer



Containerized Data Importer (CDI)

What is it about?

- Importing data to k8s/Openshift
 - Predated volume populators
- Based on Data Volume
- Supports additional sources
- Used by Forklift
 - Partially replaced in Forklift 2.4

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: "test-dv"
spec:
  source:
    imageio:
      url: "http://<ovirt engine url>/ovirt-engine/api"
      secretRef: "endpoint-secret"
      certConfigMap: "tls-certs"
      diskId: "1"
  pvc:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: "500Mi"
```

Comparison of both solutions

CDI

Extension to Kubernetes

Importer per-disk

Supports multi-stage (warm) migration

Implementation of client code in Go

Extensible

Integrated in KubeVirt

PV allocation tailored to virtual disks

Volume Populators

Integrated in Kubernetes

Populator pod per-disk

No support for multi-stage migration

Can use “native” clients

Pluggable

Integration in KubeVirt: WIP

No VM-awareness

Why did we choose volume populators

CDI

Extension to Kubernetes

Importer per-disk

Supports multi-stage (warm) migration

Implementation of client code in Go

Extensible

Integrated in KubeVirt

PV allocation tailored to virtual disks

Volume Populators

Integrated in Kubernetes

Populator pod per-disk

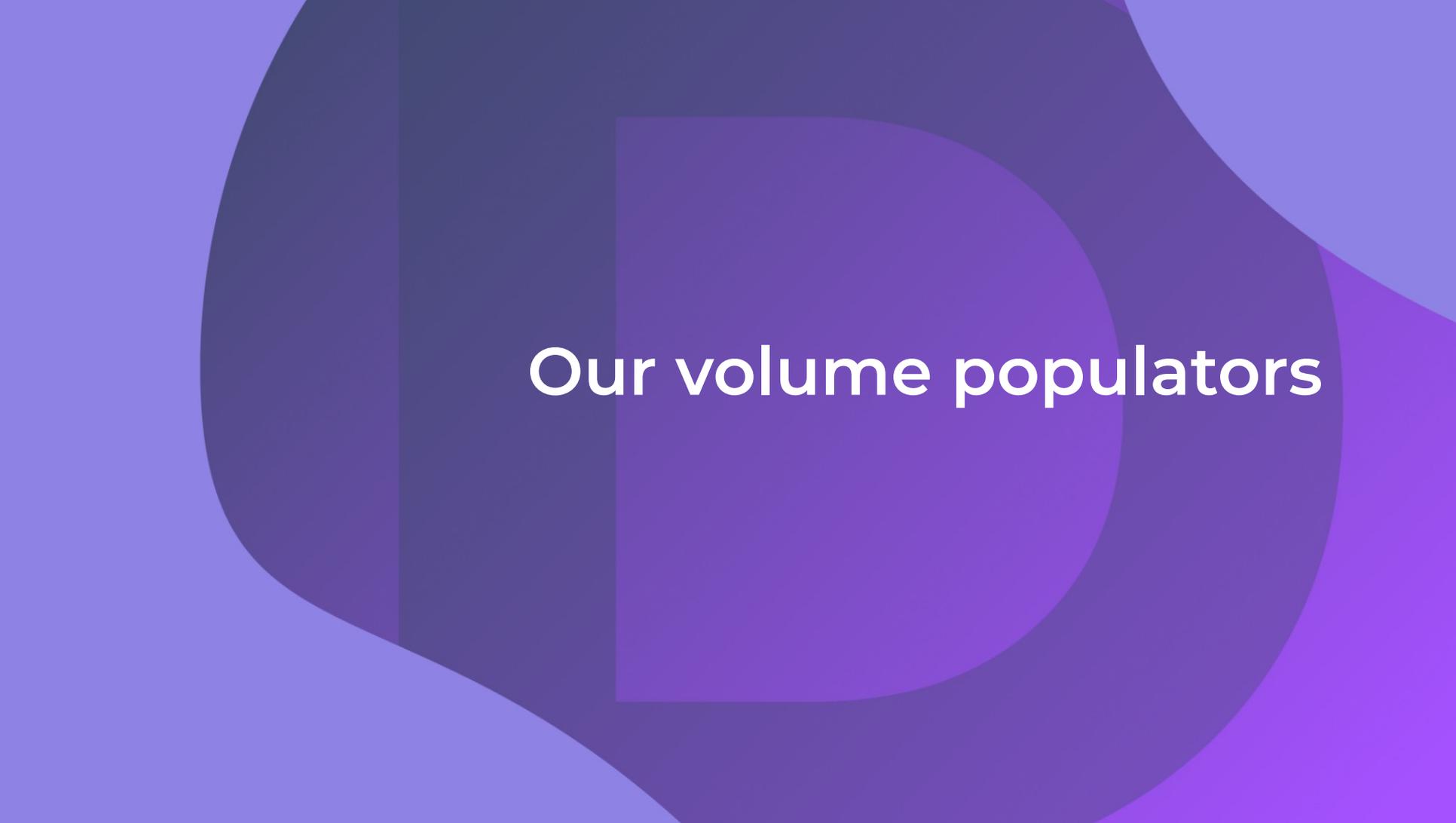
No support for multi-stage migration

Can use “native” clients

Pluggable

Integration in KubeVirt: WIP

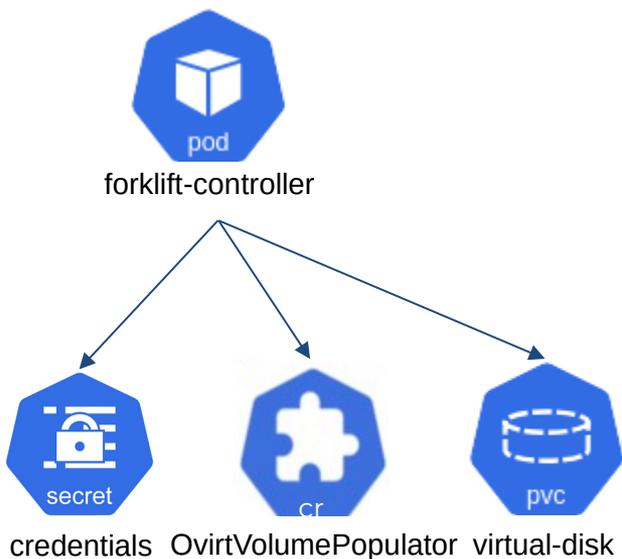
No VM-awareness

The background consists of several overlapping, semi-transparent purple shapes. A large, dark purple circle is on the left, partially overlapping a medium purple circle on the right. A light purple shape is at the top right, and a bright purple shape is at the bottom right. The text is centered in the middle of these shapes.

Our volume populators

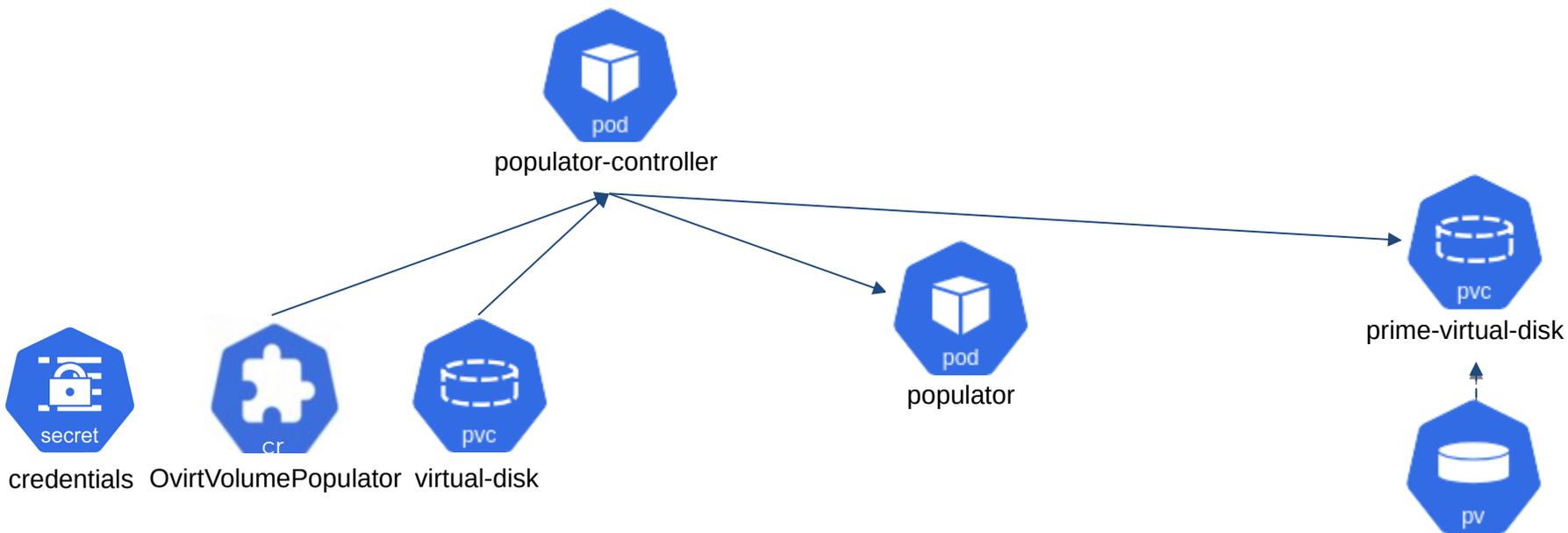
Volume populator for oVirt / RHV

The forklift-controller creates relevant resources during migration



Volume populator for oVirt / RHV

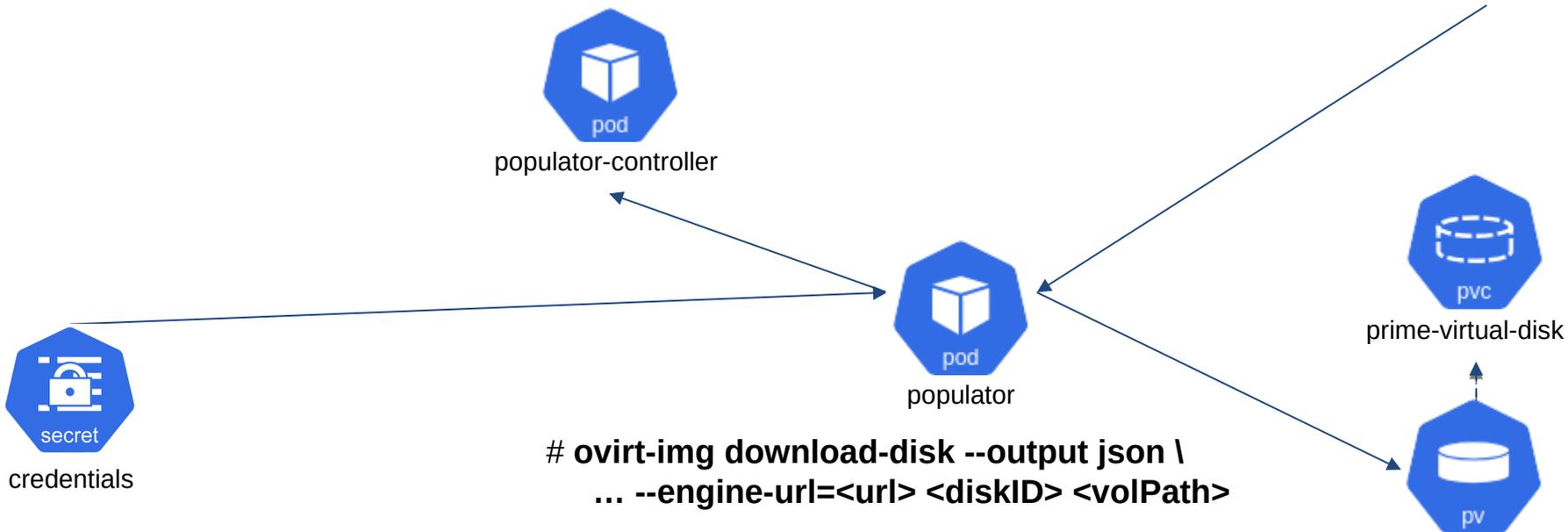
The populator-controller detects PVC + ovfp and create populator pod and prime-PVC



Volume populator for oVirt / RHV

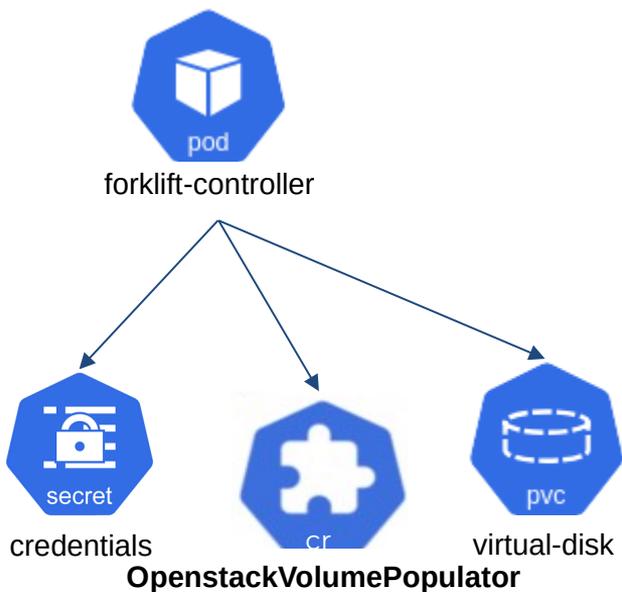
The populator pod copies the data using ovirt-img and reports progress

oVirt



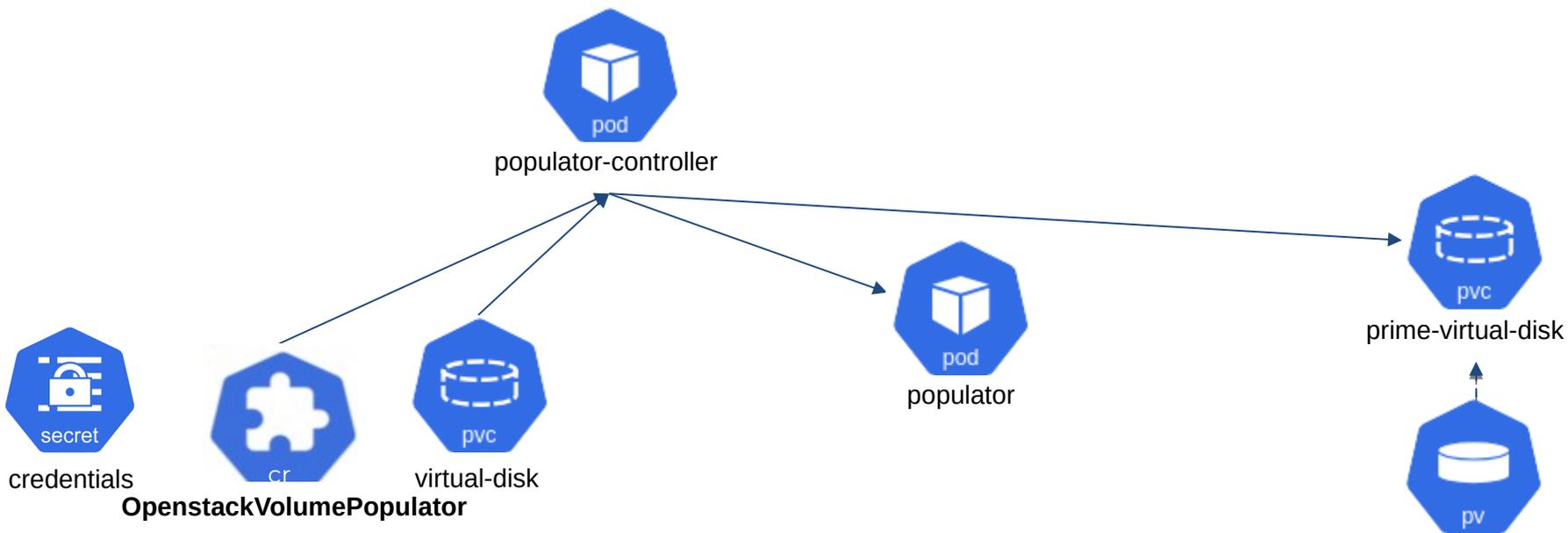
Volume populator for OpenStack

The forklift-controller creates relevant resources during migration



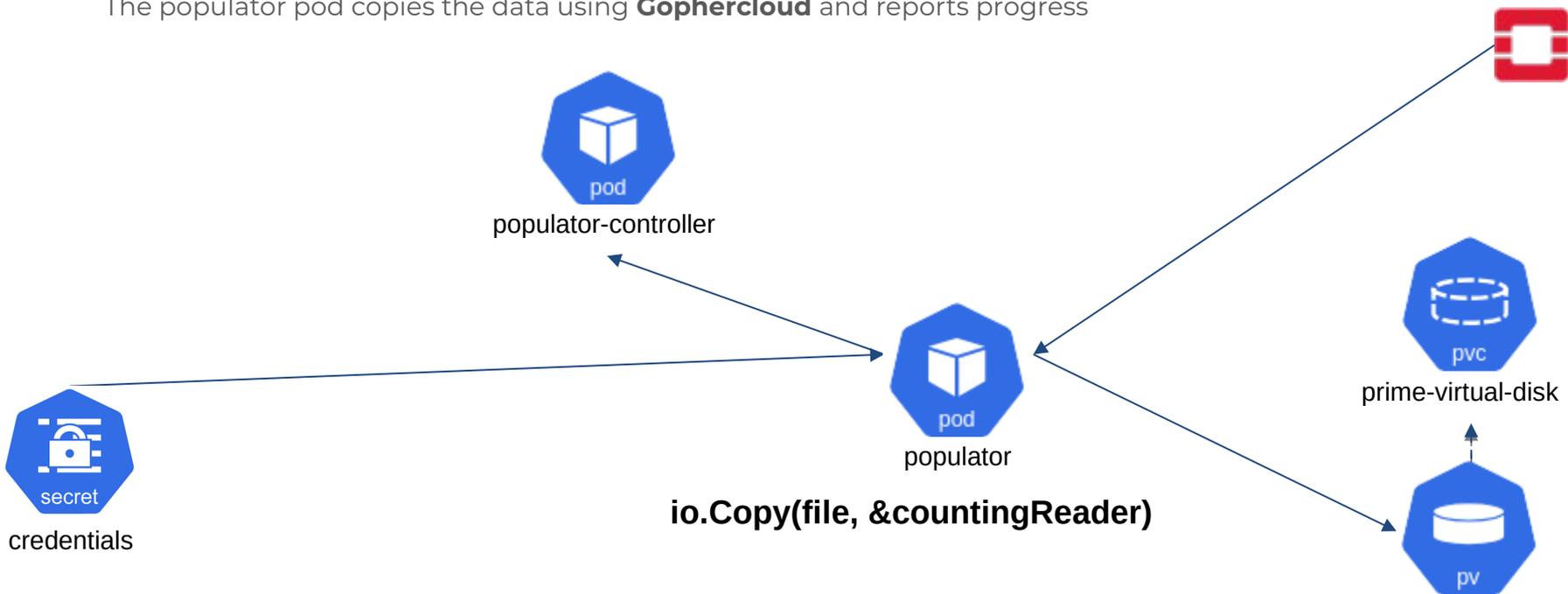
Volume populator for OpenStack

The populator-controller detects PVC + **osvp** and create populator pod and prime-PVC



Volume populator for OpenStack

The populator pod copies the data using **Gophercloud** and reports progress



Achievements

- Improved migrations from oVirt / RHV
 - Transfers were accelerated
 - Added “insecure transfers”
 - Can potentially deprecate CDI code
- Introduced migrations from OpenStack
 - Similar transfer mechanism to that of oVirt / RHV
 - Without adding code to CDI

The background features a complex arrangement of overlapping shapes in various shades of purple, from light lavender to deep, dark violet. The shapes include large circles and irregular, organic forms that create a layered, abstract composition. The text is centered within this design.

Challenges and Insights

Delegate volume allocation to CDI

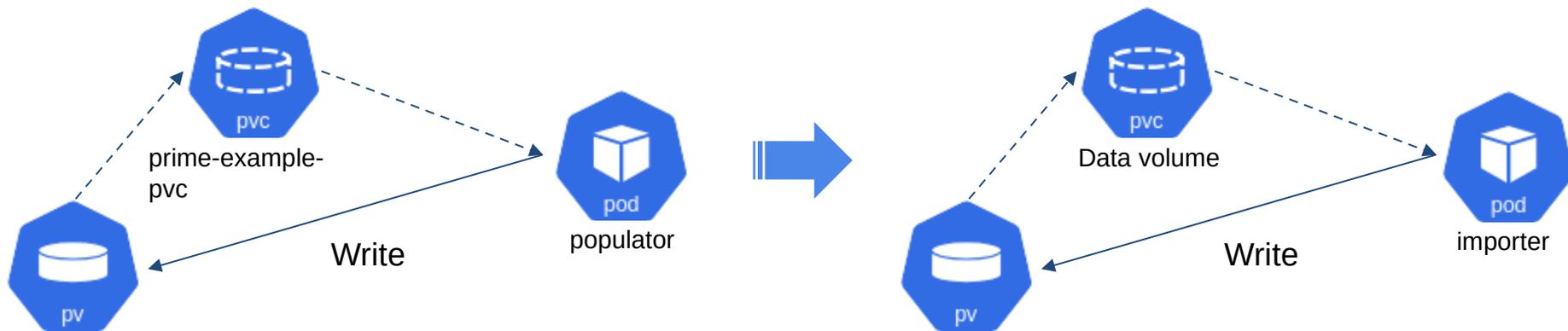
We planned to create Data Volumes with source = Blank



Delegate volume allocation to CDI

We planned to create Data Volumes with source = Blank

However, this ended up in the populated data being overridden by CDI with empty data



Delegate volume allocation to CDI

We planned to create Data Volumes with source = Blank

However, this ended up in the populated data being overridden by CDI with empty data

As a result, we create PVCs instead

... and had to implement similar logic for:

- Access modes
- Disk overhead

Progress reporting

Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

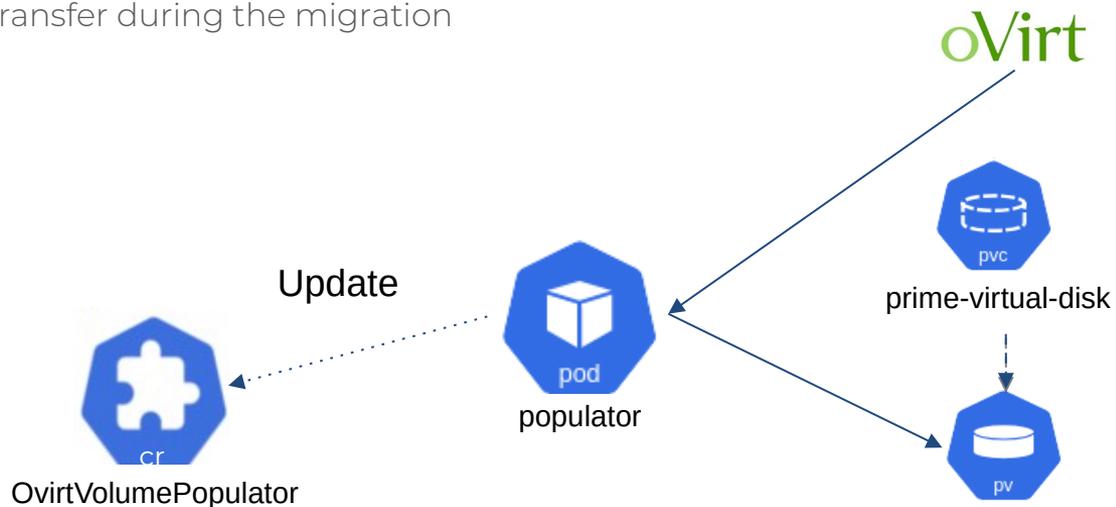


Progress reporting

Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

Attempt #1: Update the CR

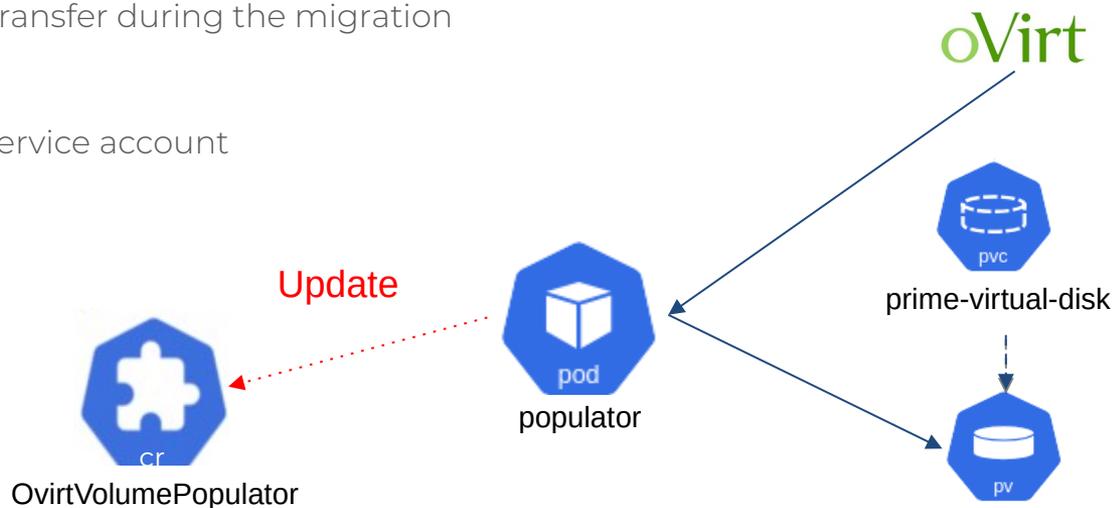


Progress reporting

Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

Attempt #1: Update the CR - Requires a service account



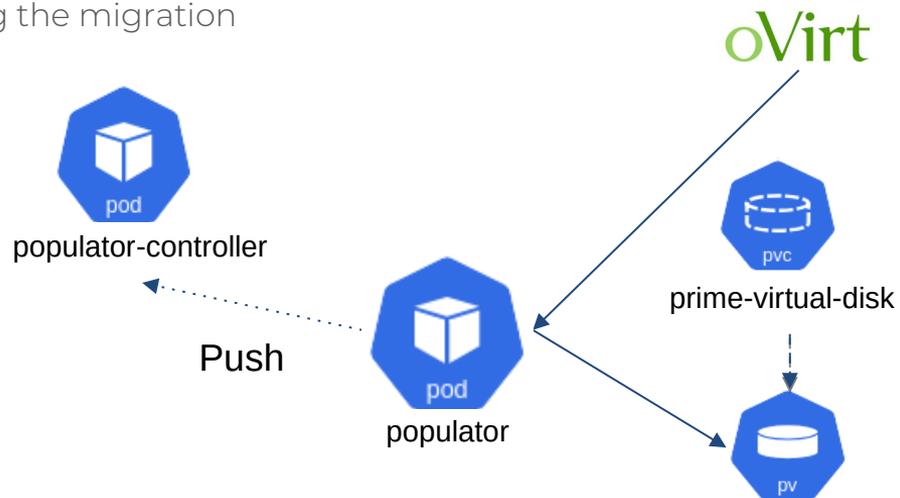
Progress reporting

Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

Attempt #1: Update the CR

Attempt #2: Push reports to the controller



Progress reporting

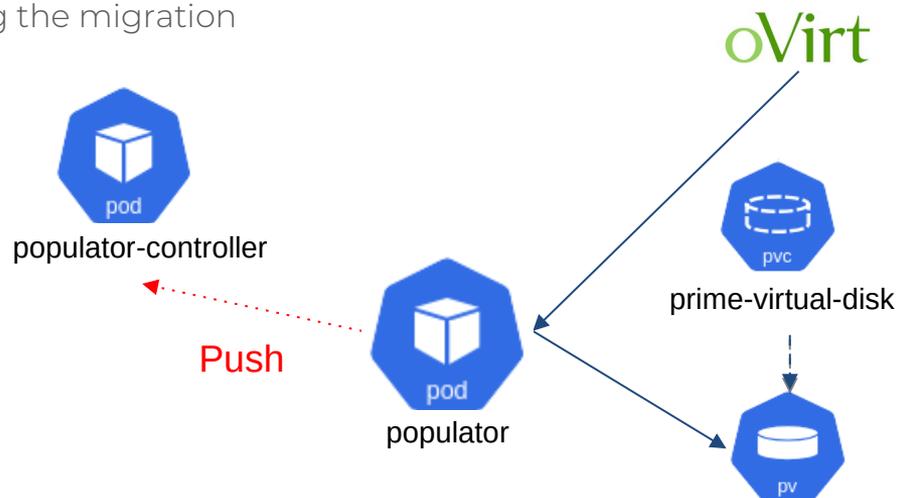
Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

Attempt #1: Update the CR

Attempt #2: Push reports to the controller -

Requires the populator pod to 'know'
about the controller



Progress reporting

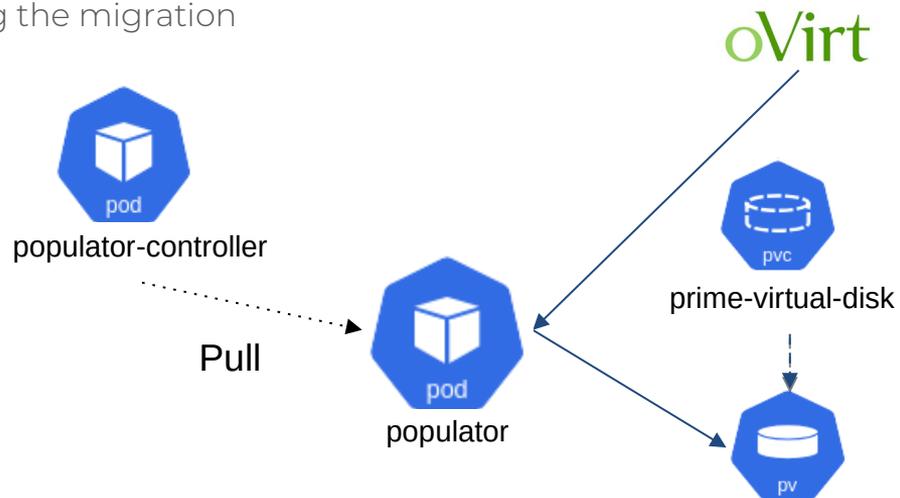
Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

Attempt #1: Update the CR

Attempt #2: Push reports to the controller

Attempt #3: Pull metrics from populator pods



Progress reporting

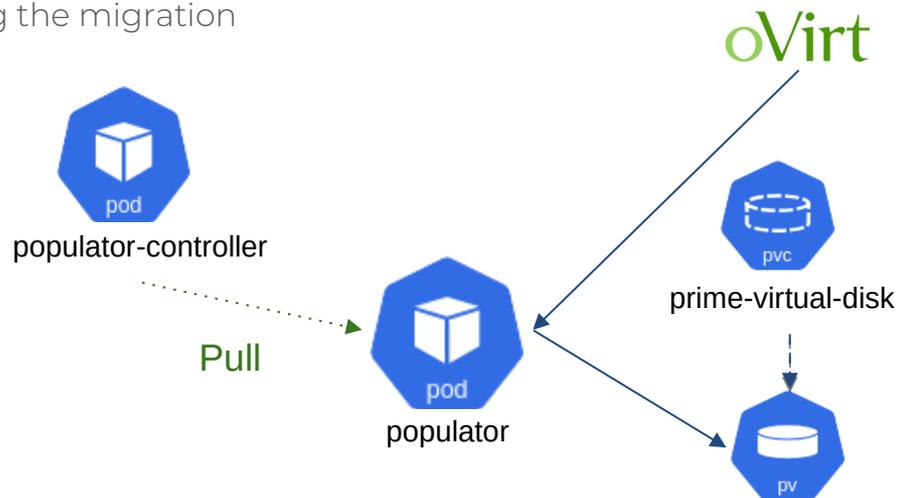
Copying data from remote environments can take significant time (few hours)

Thus we report the progress of the data transfer during the migration

Attempt #1: Update the CR

Attempt #2: Push reports to the controller

✓ Attempt #3: Pull metrics from populator pods



Dynamic volume provisioning

Our populators worked well in our development environments



Dynamic volume provisioning

Our populators worked well in our development environments

However, migrations sometimes failed on QE environments

Further analysis revealed it failed on statically provisioned storage classes

- An issue in the lib-volume-populator library

Dynamic volume provisioning

Our populators worked well in our development environments

However, migrations sometimes failed on QE environments

Further analysis revealed it failed on statically provisioned storage classes

- An issue in the lib-volume-populator library

Discussed with maintainers of kubernetes-csi/lib-volume-populator

- Have not reached a consensus on a way to resolve this

Blocked use of volume populators for statically provisioned storage classes in Forklift

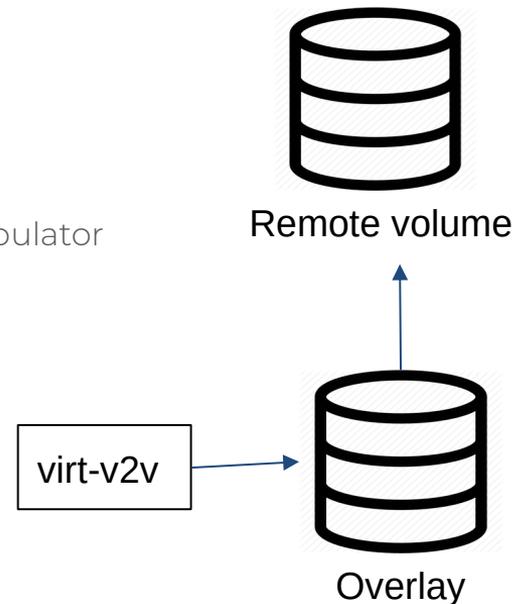


Conversion of multi-volume disks

When converting a VM from vSphere, virt-v2v operates on an overlay

- To inspect the content of a disk
- To modify the content of a disk

Conversion of a single-volume disk can be implemented with a volume populator



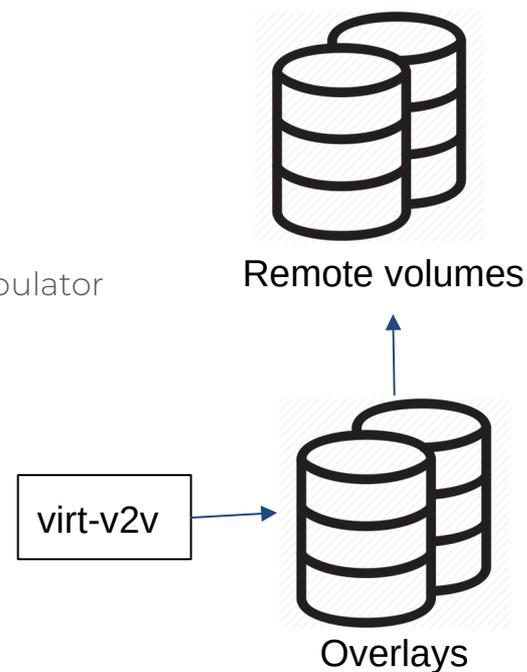
Conversion of multi-volume disks

When converting a VM from vSphere, virt-v2v operates on an overlay

- To inspect the content of a disk
- To modify the content of a disk

Conversion of a single-volume disk can be implemented with a volume populator

However, this breaks with multi-volume disks



Conversion of multi-volume disks

When converting a VM from vSphere, virt-v2v operates on an overlay

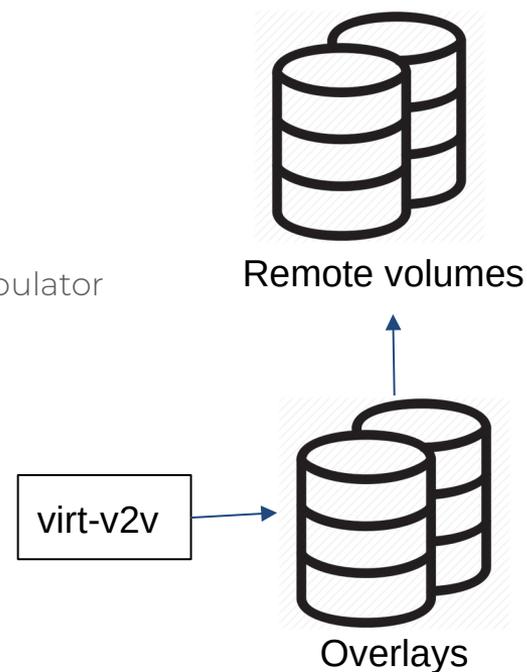
- To inspect the content of a disk
- To modify the content of a disk

Conversion of a single-volume disk can be implemented with a volume populator

However, this breaks with multi-volume disks

Filed [kubernetes-csi/lib-volume-populator#40](#)

We think about modifying the code in (our fork of) lib-volume-populator



Migrations to remote clusters

When migrating to another cluster

1. The CR needs to be posted to this cluster
2. The populator pod needs to run on this cluster



Migrations to remote clusters

When migrating to another cluster

1. The CR needs to be posted to this cluster
2. The populator pod needs to run on this cluster

First question: how to define the CRD(s) on a remote cluster

Second question: where should the controller run



Migrations to remote clusters

When migrating to another cluster

1. The CR needs to be posted to this cluster
2. The populator pod needs to run on this cluster

First question: how to define the CRD(s) on a remote cluster

Second question: where should the controller run

Approach #1: CRDs would be defined and processed by controllers on the source cluster



Migrations to remote clusters

When migrating to another cluster

1. The CR needs to be posted to this cluster
2. The populator pod needs to run on this cluster

First question: how to define the CRD(s) on a remote cluster

Second question: where should the controller run

Approach #1: CRDs would be defined and processed by controllers on the source cluster

- Managing CRDs on a remote cluster is complex
- Reporting the progress to a remote cluster is challenging

Migrations to remote clusters

When migrating to another cluster

1. The CR needs to be posted to this cluster
2. The populator pod needs to run on this cluster

First question: how to define the CRD(s) on a remote cluster

Second question: where should the controller run

Approach #1: CRDs would be defined and processed by controllers on the source cluster

Approach #2:: let CDI deploy CRDs and control the flow



Migrations to remote clusters

When migrating to another cluster

1. The CR needs to be posted to this cluster
2. The populator pod needs to run on this cluster

First question: how to define the CRD(s) on a remote cluster

Second question: where should the controller run

Approach #1: CRDs would be defined and processed by controllers on the source cluster

✓ Approach #2:: let CDI deploy CRDs and control the flow (WIP)

More adaptations to migration flow

- Create populator pods in target namespaces
 - To ease debugging and cleanup (owner-references)
 - No need to hide prime-PVCs and populator pods
- Limit restarts of populator pods to 3 (by default)
 - To conclude migration failed and gather logs
- Correlate created resources with a migration using a label
 - For cleanup of migration resources
- Respect selected transfer network

The background consists of several overlapping, semi-transparent purple shapes in various shades, creating a layered, abstract effect. The colors range from a light lavender to a deep, dark purple. The word "Conclusion" is centered in the right half of the image, overlaid on a medium-purple shape.

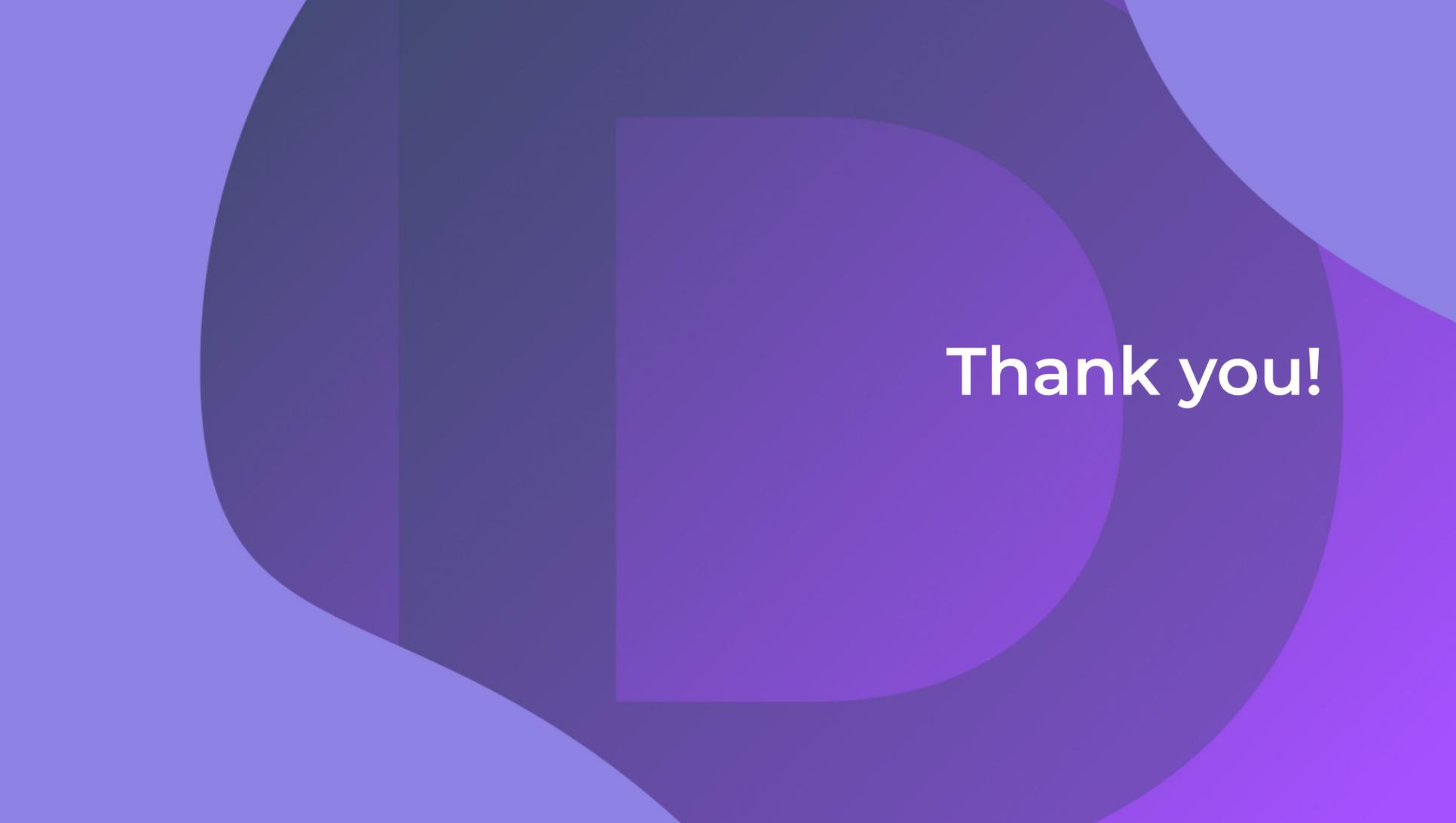
Conclusion

So, volume populators for virtual disks?

- Generally yes
 - We achieved the basic functionality for VM migration
 - Can be integrated into KubeVirt with CDI (WIP)
- But doesn't completely fit for VM migration
 - We had to introduce a variety of changes to the controller library
 - ... and eventually forked it
 - Additional work is required for advanced functionality
 - Remote migrations, warm migrations, conversion of multi-volume disks

The background consists of several overlapping, semi-transparent purple shapes in various shades, creating a layered, abstract effect. The colors range from a light lavender to a deep, dark purple. The shapes are irregular and organic, with some appearing as large, rounded rectangles or ovals. The overall composition is modern and minimalist.

Questions?

The background consists of several overlapping, semi-transparent purple shapes in various shades, creating a layered, abstract effect. The colors range from a light lavender to a deep, dark purple. The shapes are rounded and organic in form, with some appearing as large, soft-edged rectangles or ovals. The overall composition is clean and modern.

Thank you!