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

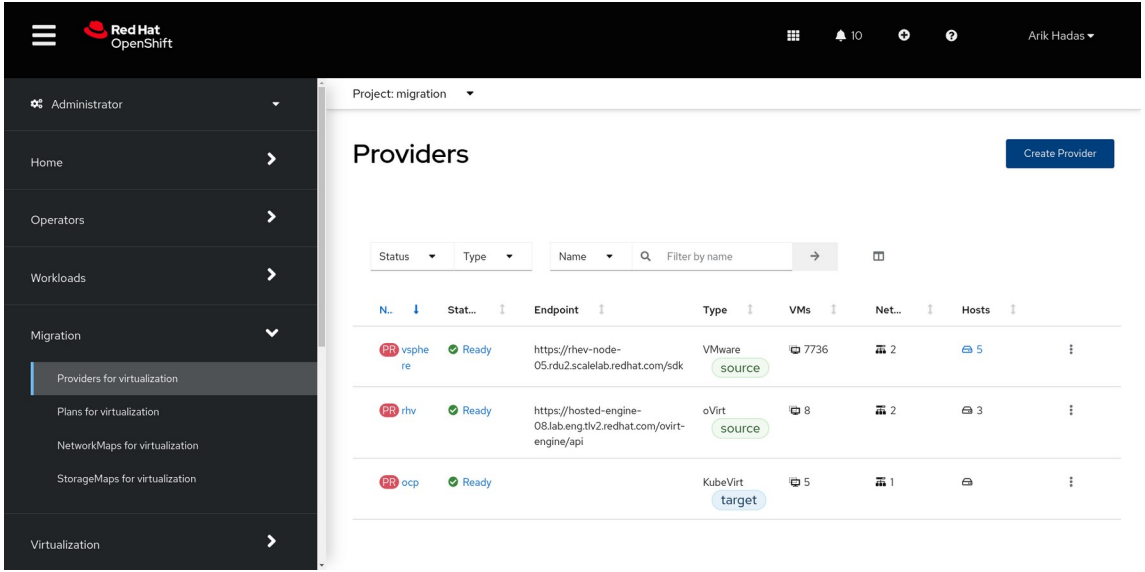
"Migrate virtual machines at scale to OpenShift in a few simple steps. Provide source and destination credentials, map infrastructure, and create migration plans"

MTV’s mission statement

How to start a migration in Forklift

Few simple steps using an easy to use UI

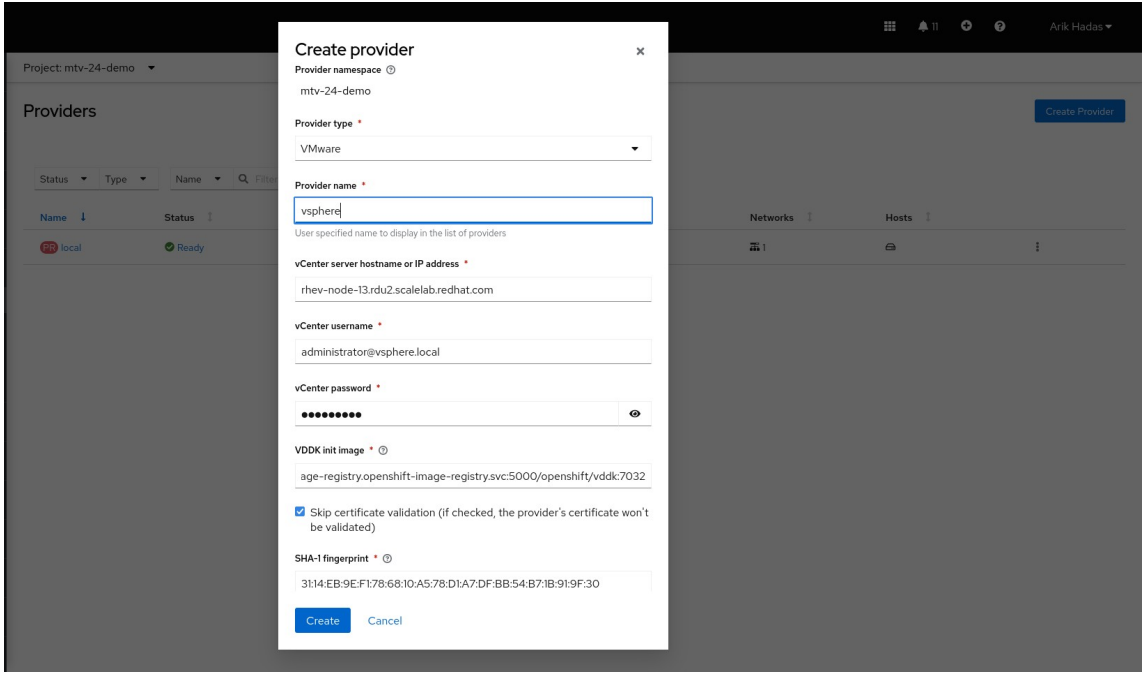
- Define a source provider
- Define a target provider (optional)
- Create a migration plan
  - Select virtual machines
  - Set Network mappings
  - Set Storage mappings
- Start the migration plan



How to start a migration in Forklift

Few simple steps using an easy to use UI

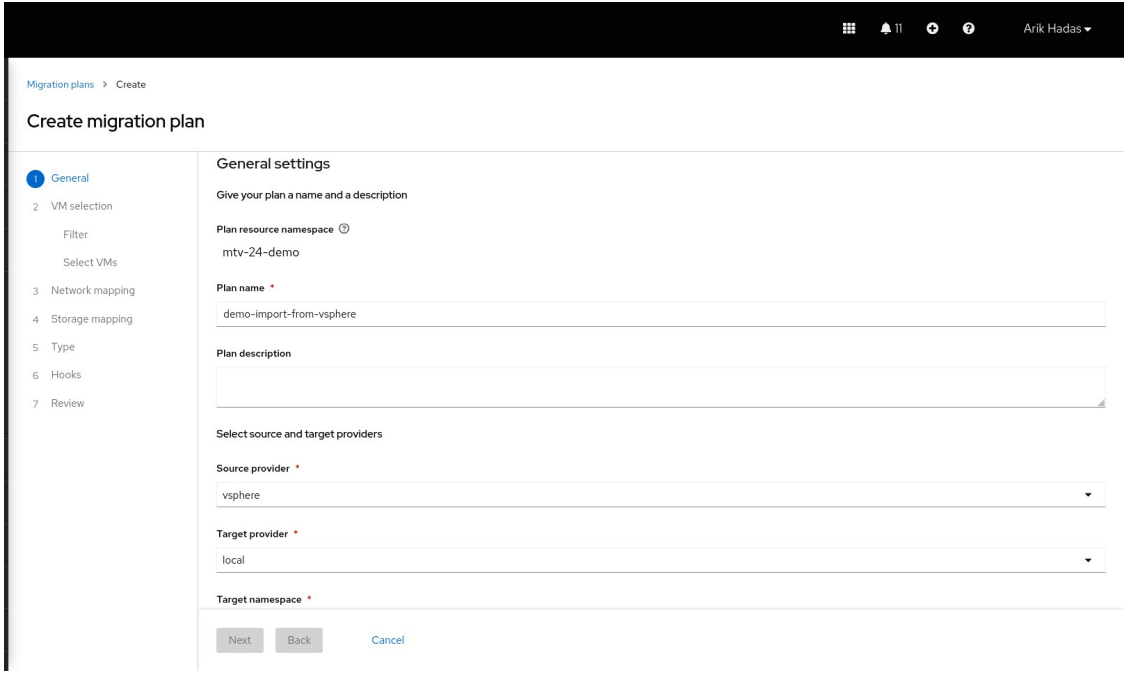
- **Define a source provider**
- Define a target provider (optional)
- Create a migration plan
  - Select virtual machines
  - Set Network mappings
  - Set Storage mappings
- Start the migration plan



How to start a migration in Forklift

Few simple steps using an easy to use UI

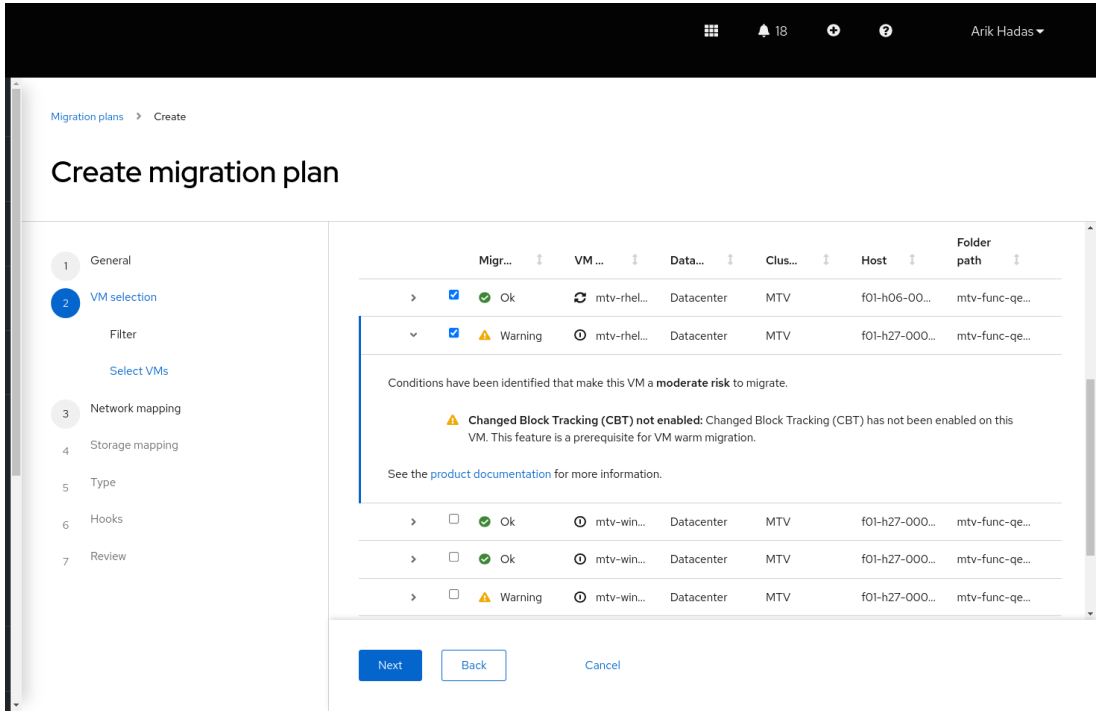
- Define a source provider
- Define a target provider (optional)
- **Create a migration plan**
  - Select virtual machines
  - Set Network mappings
  - Set Storage mappings
- Start the migration plan



How to start a migration in Forklift

Few simple steps using an easy to use UI

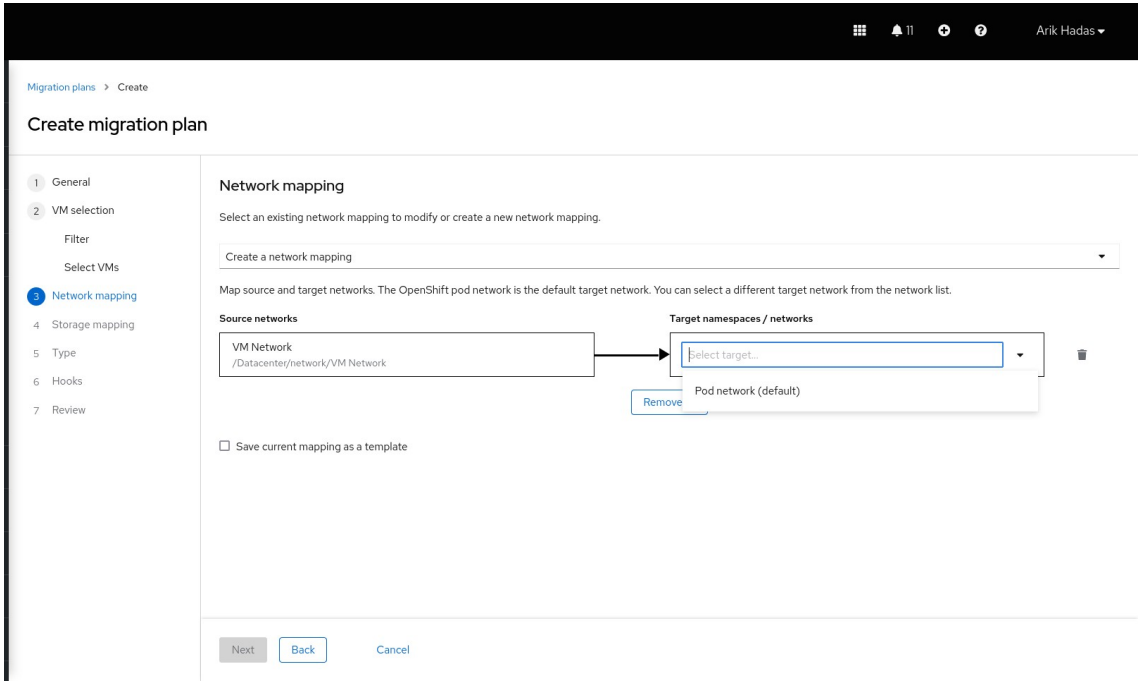
- Define a source provider
- Define a target provider (optional)
- Create a migration plan
  - **Select virtual machines**
  - Set Network mappings
  - Set Storage mappings
- Start the migration plan



How to start a migration in Forklift

Few simple steps using an easy to use UI

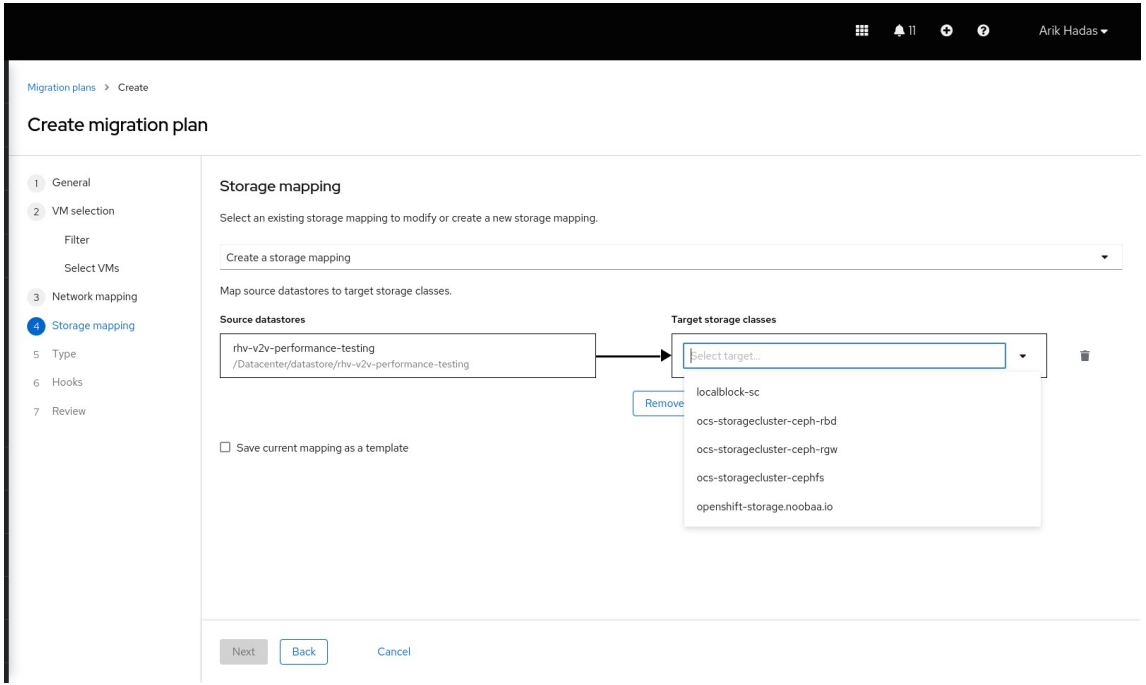
- Define a source provider
- Define a target provider (optional)
- Create a migration plan
  - Select virtual machines
  - **Set Network mappings**
  - Set Storage mappings
- Start the migration plan



How to start a migration in Forklift

Few simple steps using an easy to use UI

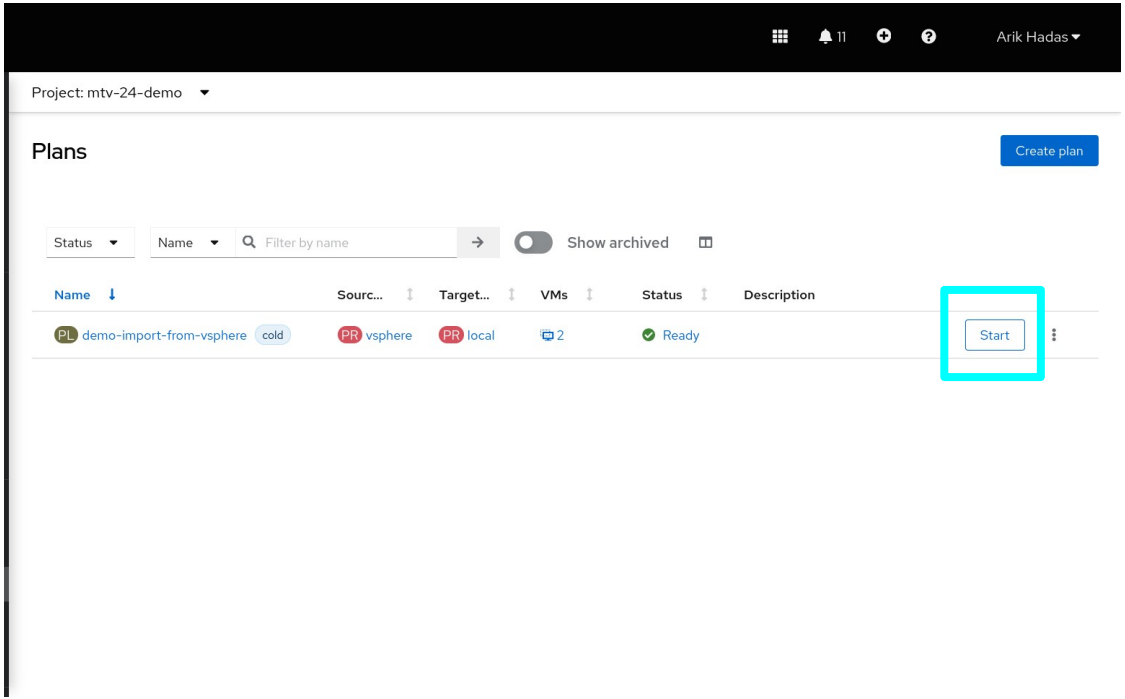
- Define a source provider
- Define a target provider (optional)
- Create a migration plan
  - Select virtual machines
  - Set Network mappings
  - **Set Storage mappings**
- Start the migration plan



How to start a migration in Forklift

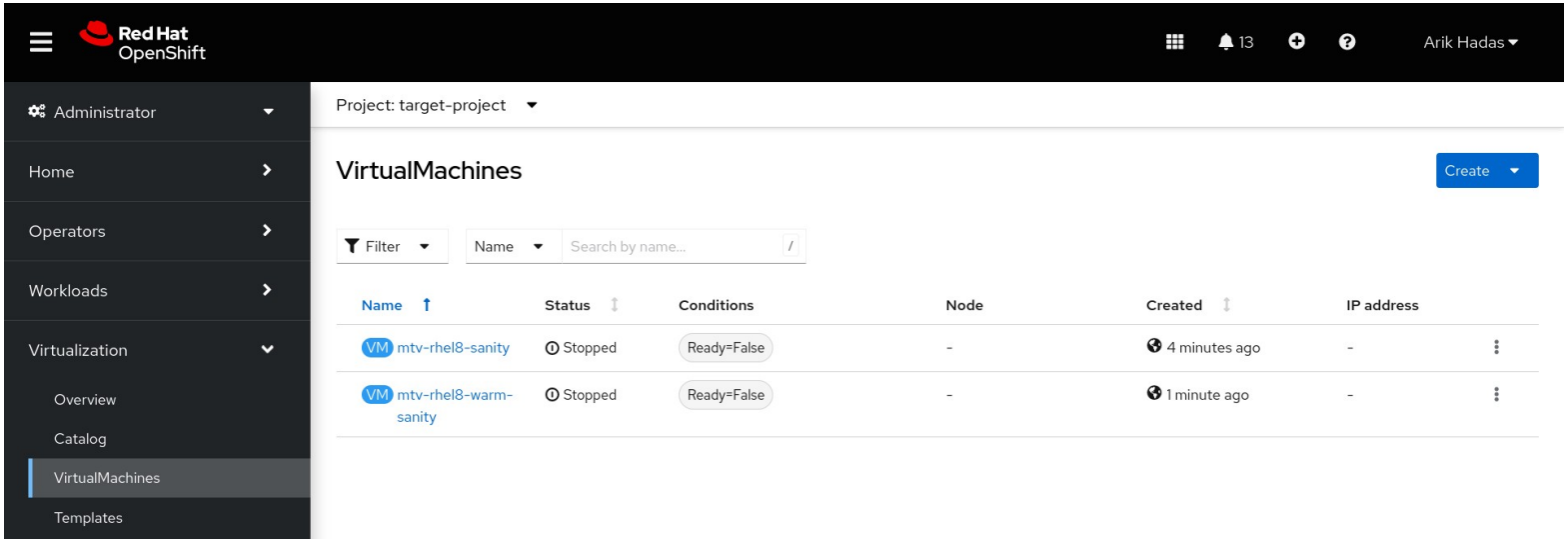
Few simple steps using an easy to use UI

- 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**



Finally: virtual machines in KubeVirt

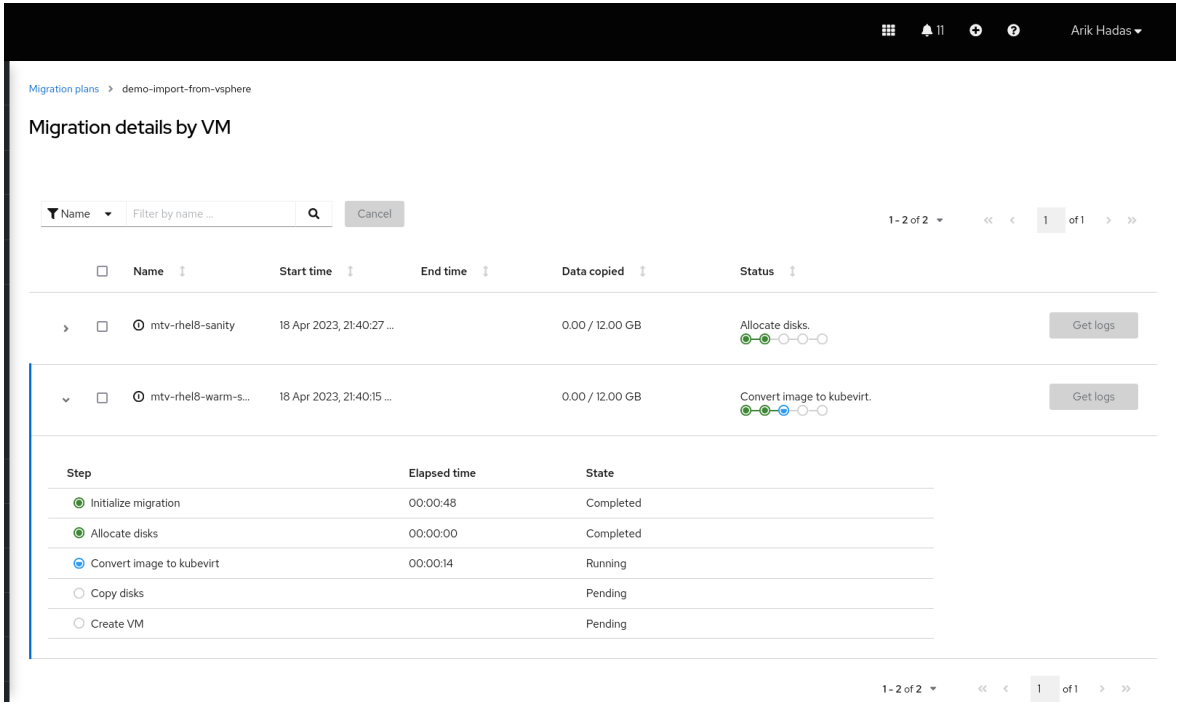
The migrated VMs can start in containers within the **target** k8s / OpenShift cluster



But what happens in between?

What is going on during the migration

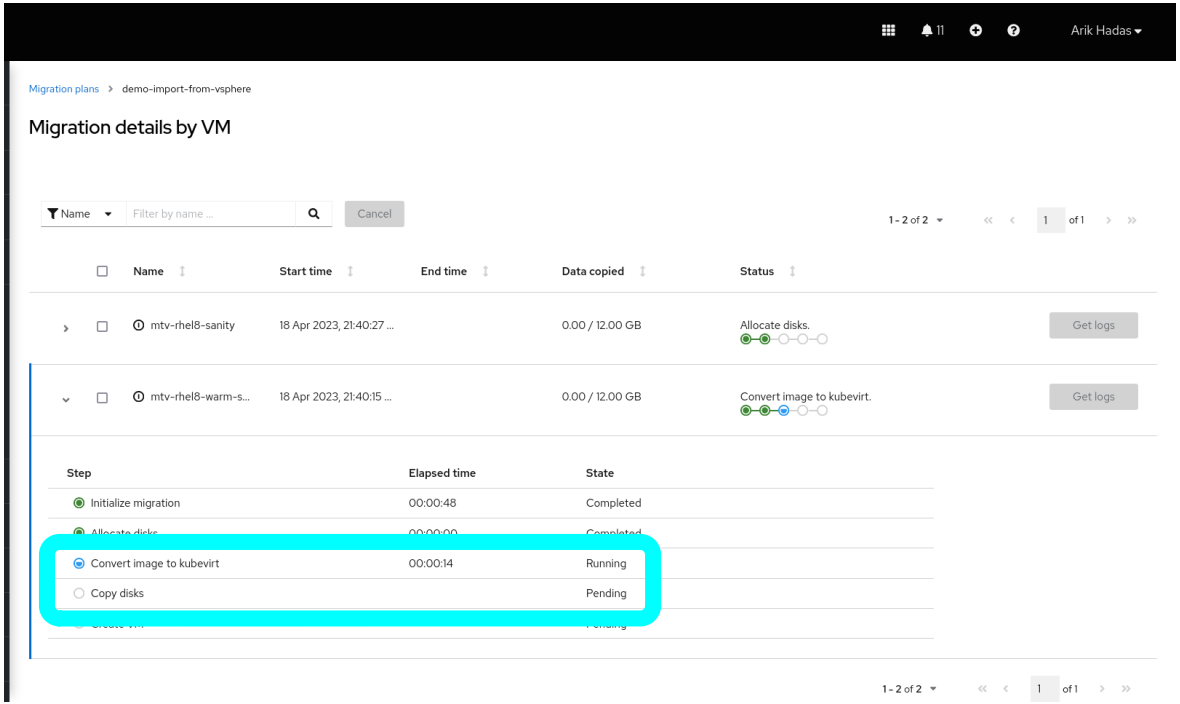
- Disks are converted (optional)
- Disks are copied
- VM configuration is converted



But what happens in between?

Our focus today

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



Volume populators

The volume populators feature

Setting up PVC and data source CR


```yaml
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
```

```yaml
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

```yaml
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
```

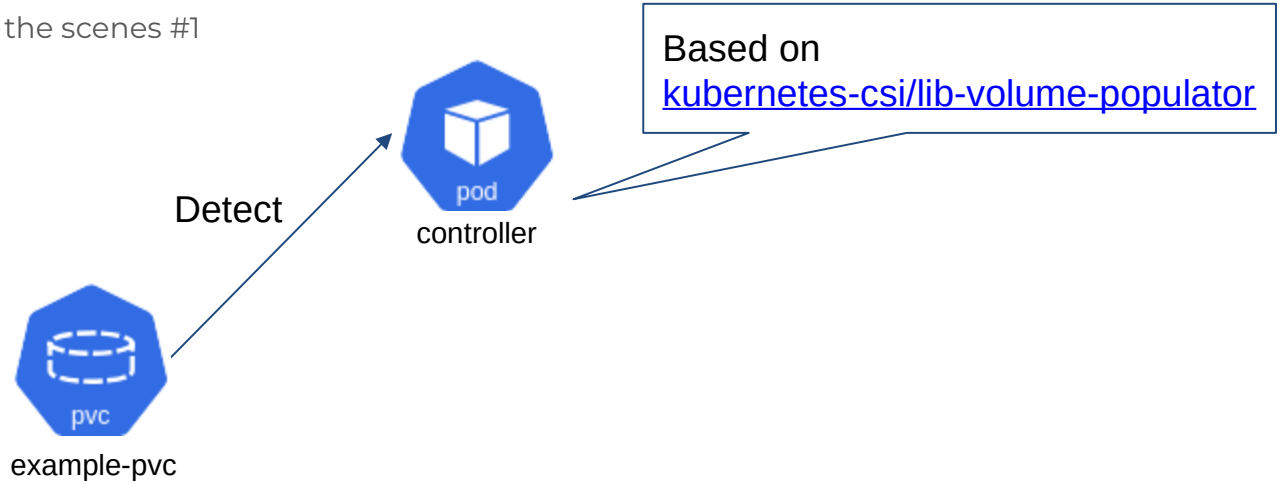
```
kubectl wait --for=condition=Complete job/example-job
```

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

```
Hello, world!
```

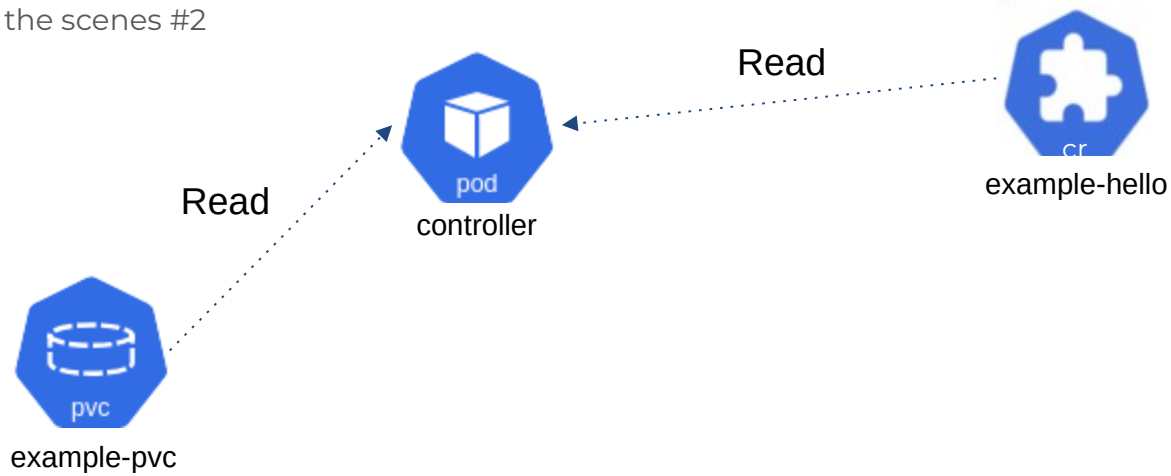
The volume populators feature

What happened behind the scenes #1



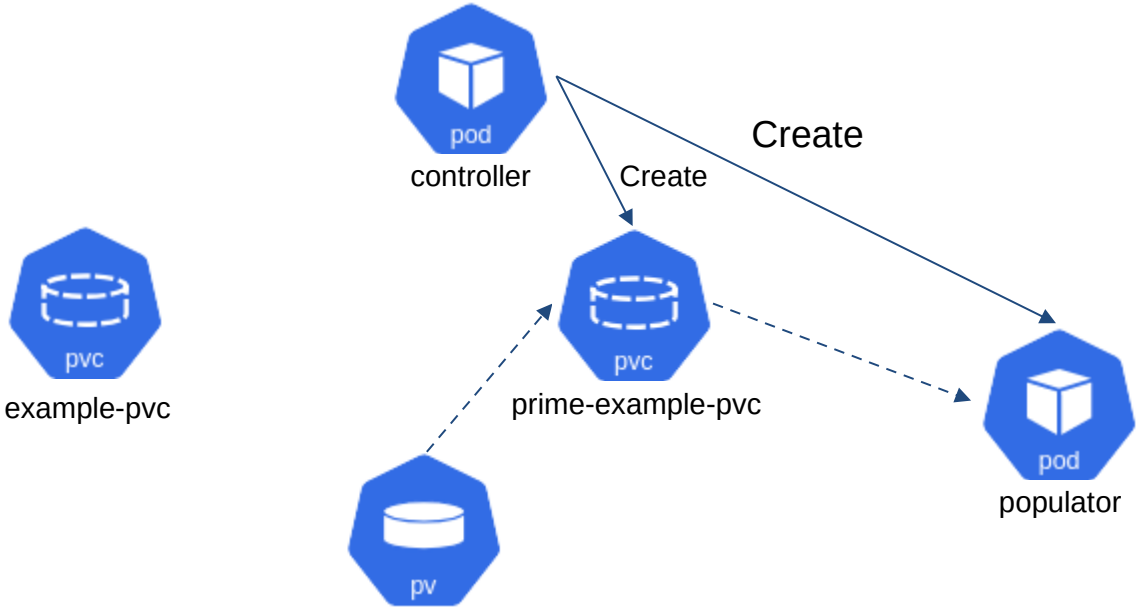
The volume populators feature

What happened behind the scenes #2



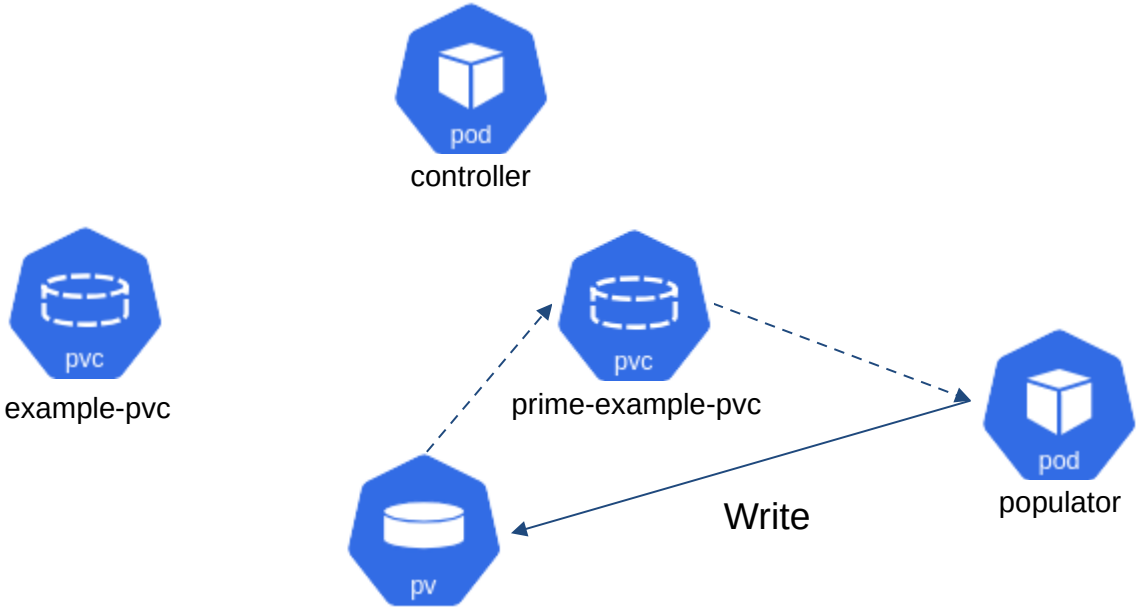
The volume populators feature

What happened behind the scenes #3



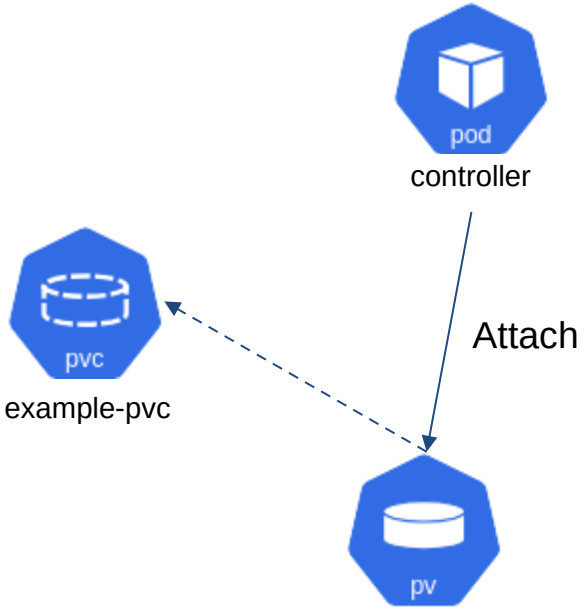
The volume populators feature

What happened behind the scenes #4

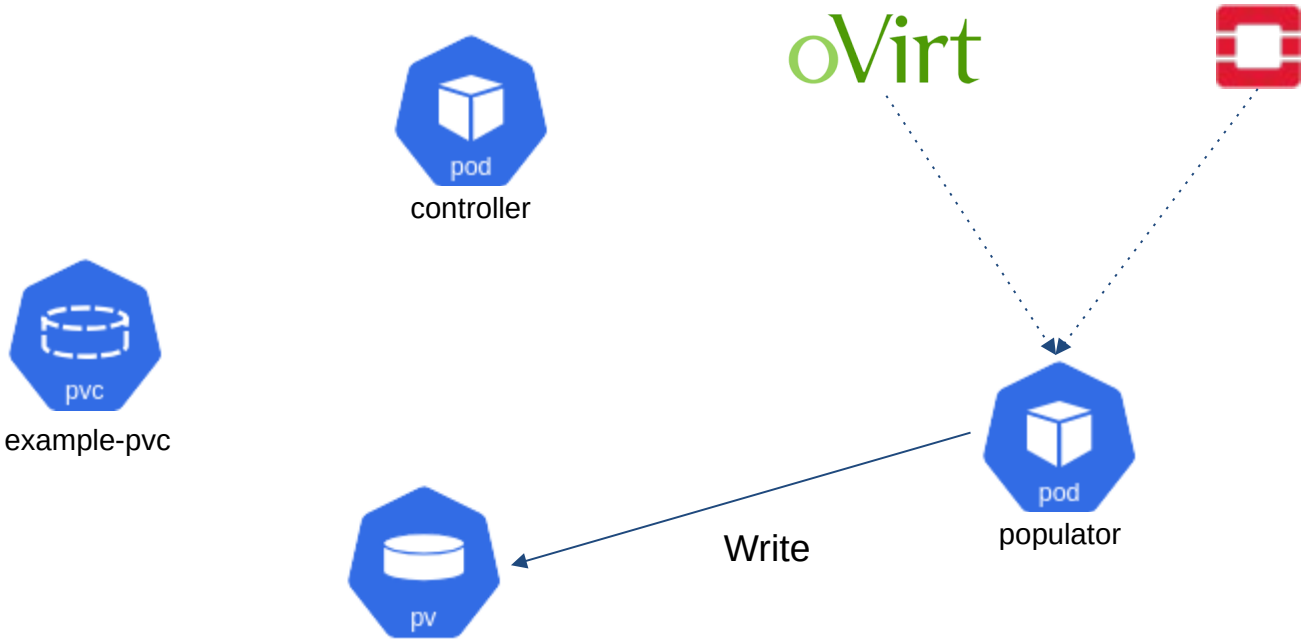


The volume populators feature

What happened behind the scenes #5



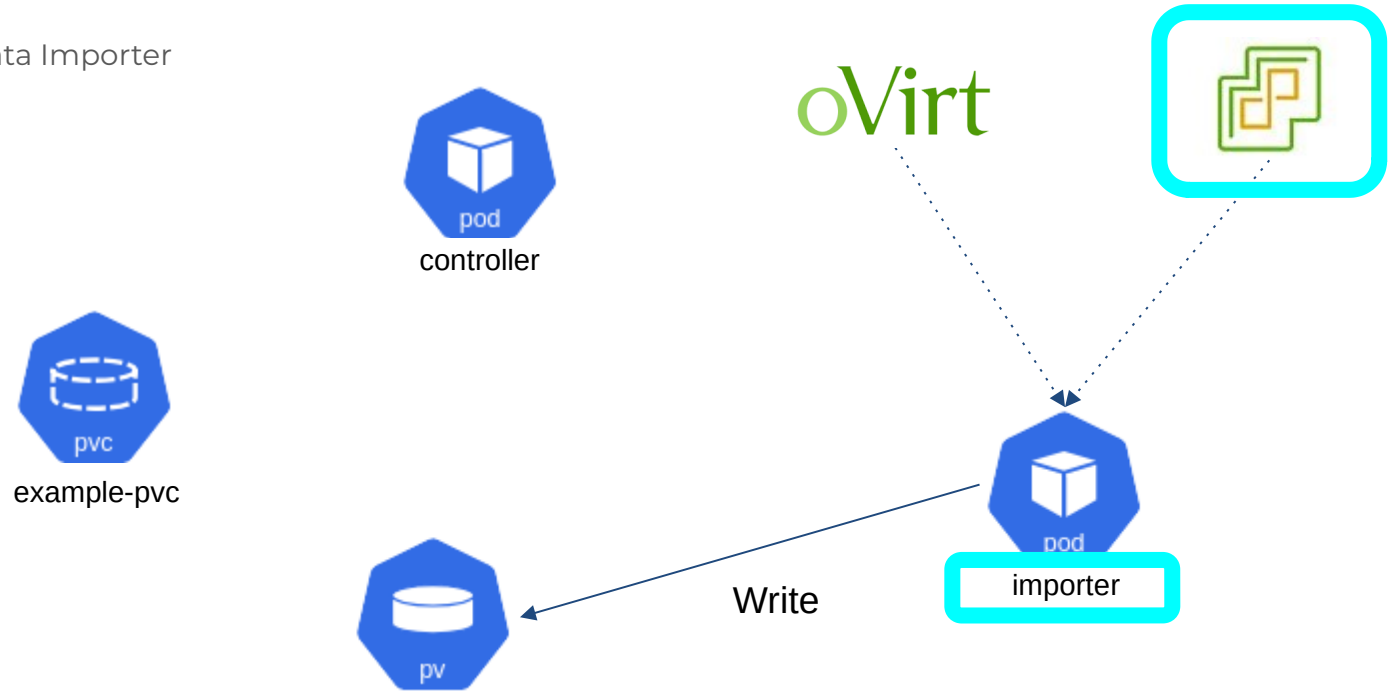
Volume populators in Forklift



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

```yaml
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

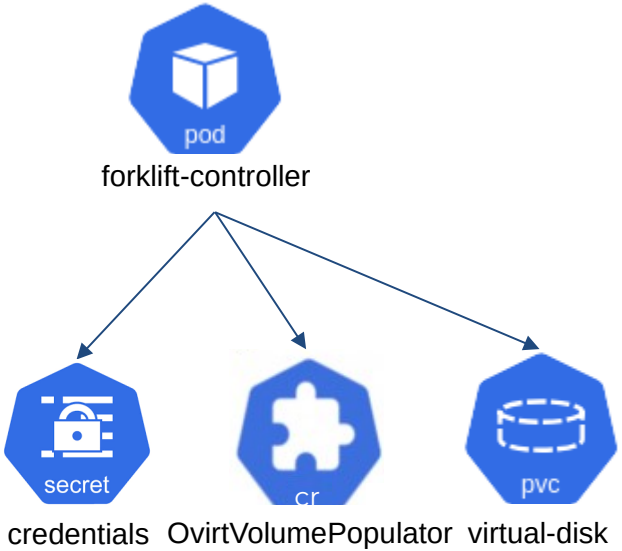
Volume Populators

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

Our volume populators

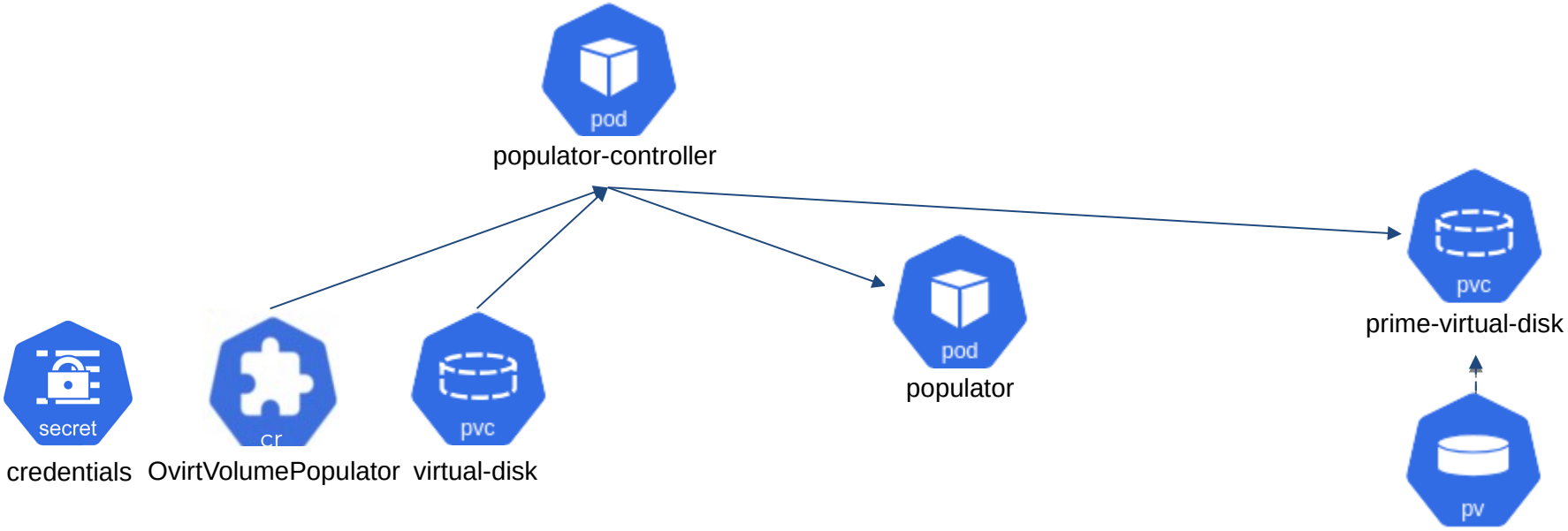
Volume populator for oVirt / RHV

The forklift-controller creates relevant resources during migration



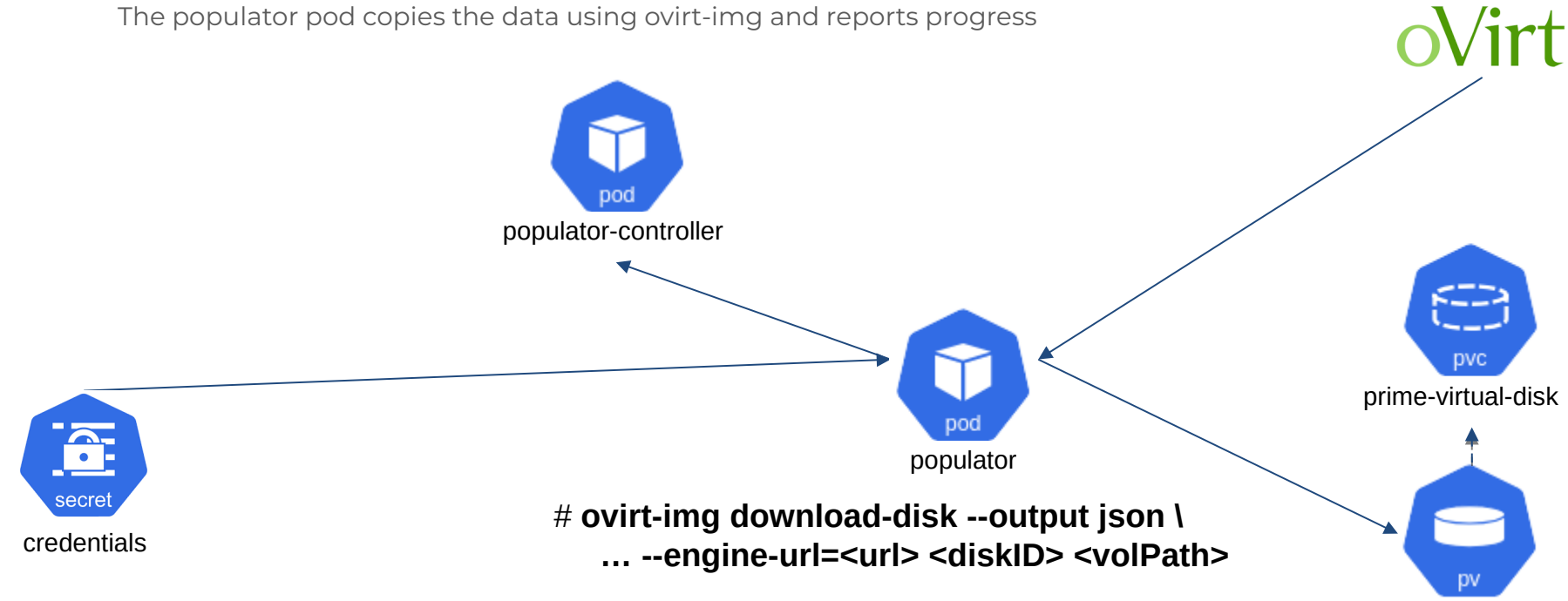
Volume populator for oVirt / RHV

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



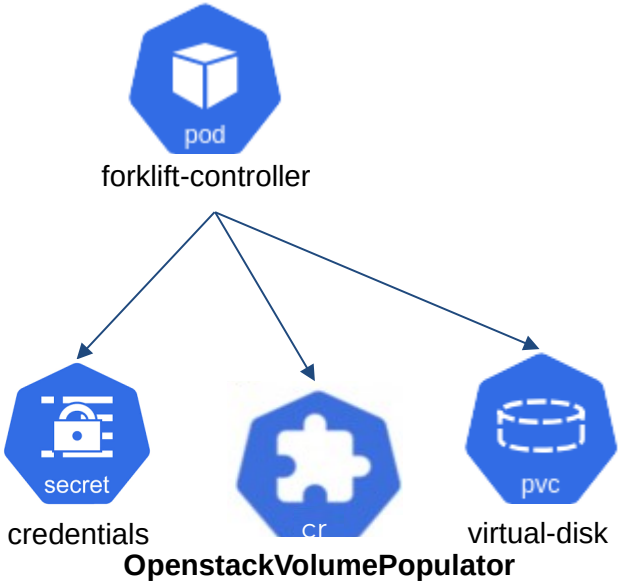
Volume populator for oVirt / RHV

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



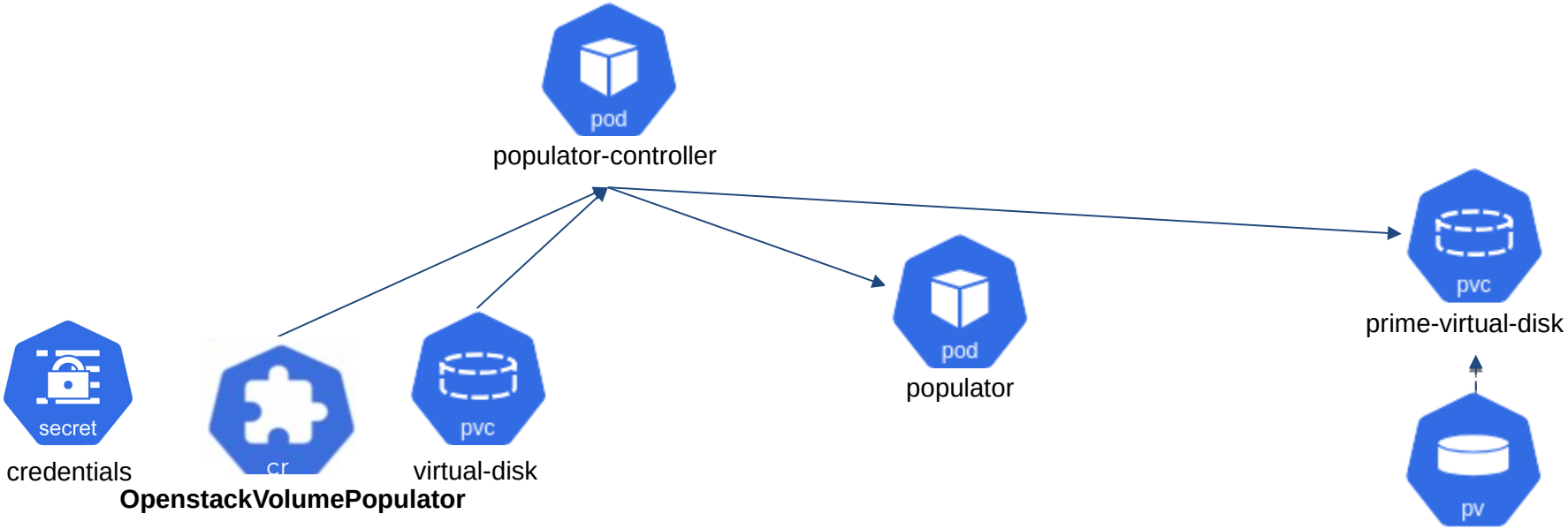
Volume populator for OpenStack

The forklift-controller creates relevant resources during migration



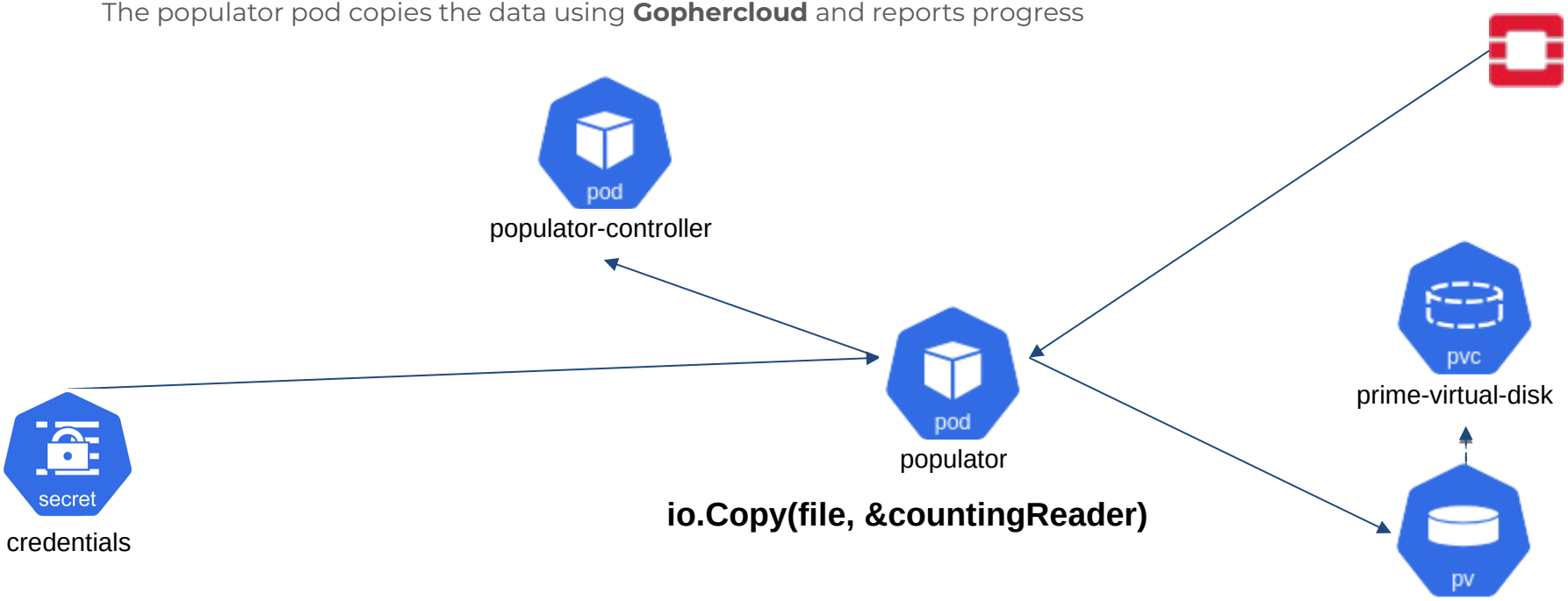
Volume populator for OpenStack

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

Challenges and Insights

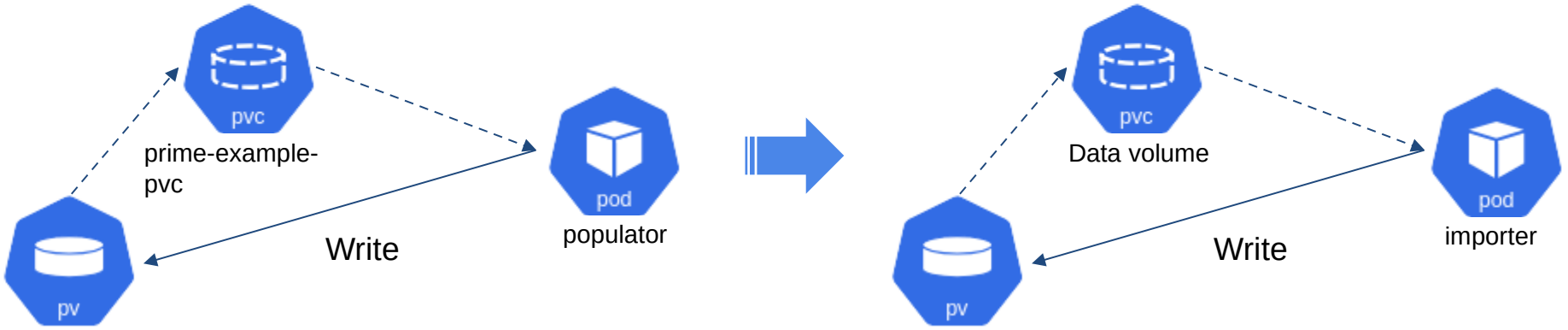
Delegate volume allocation to CDI

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)

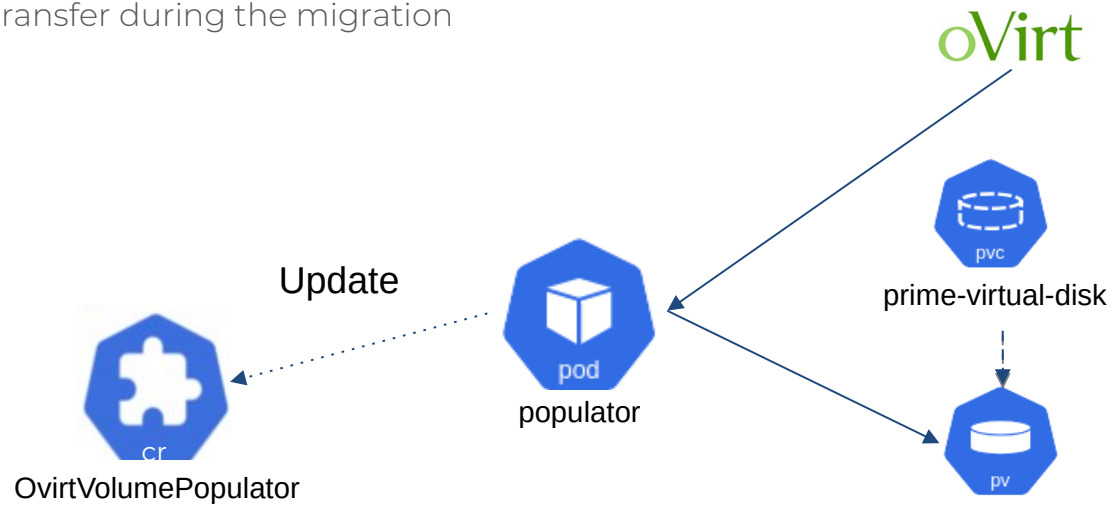
Thus we report the progress of the data transfer during the migration

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

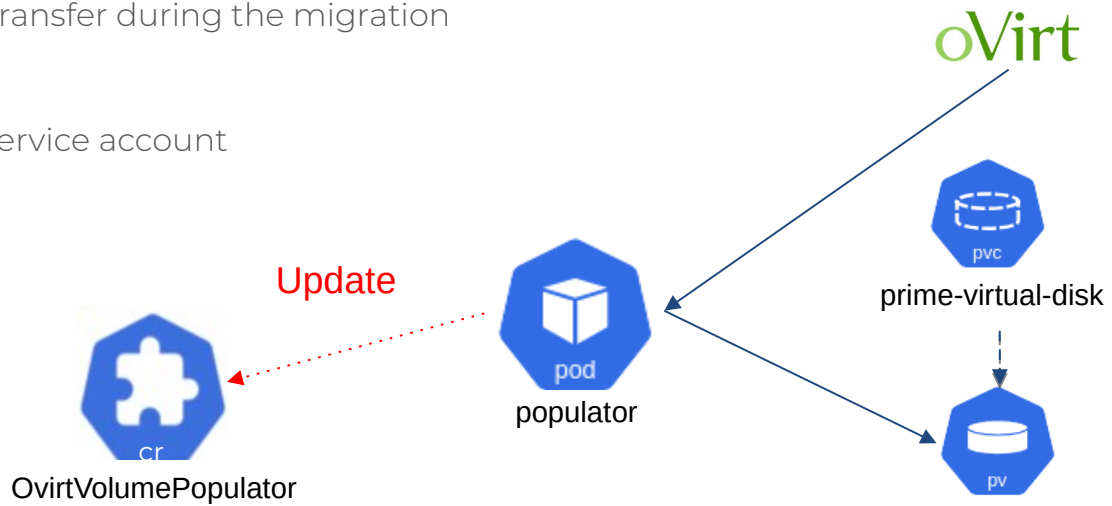


Progress reporting

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



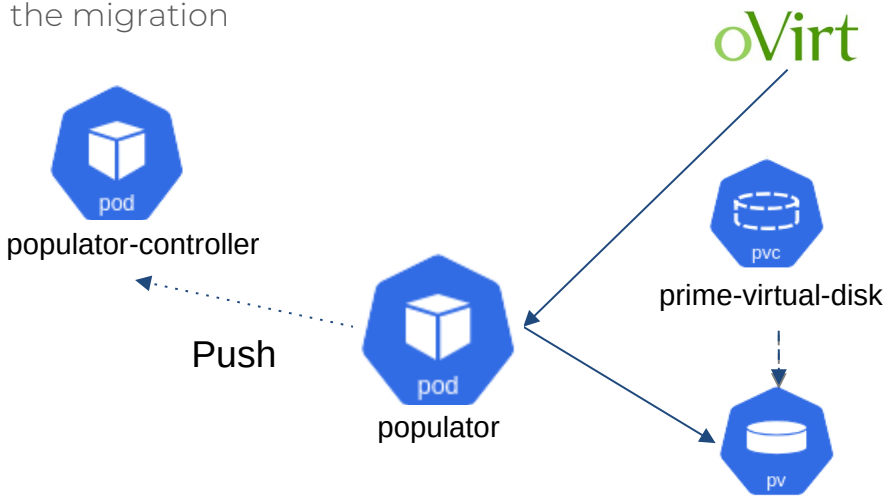
Progress reporting

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

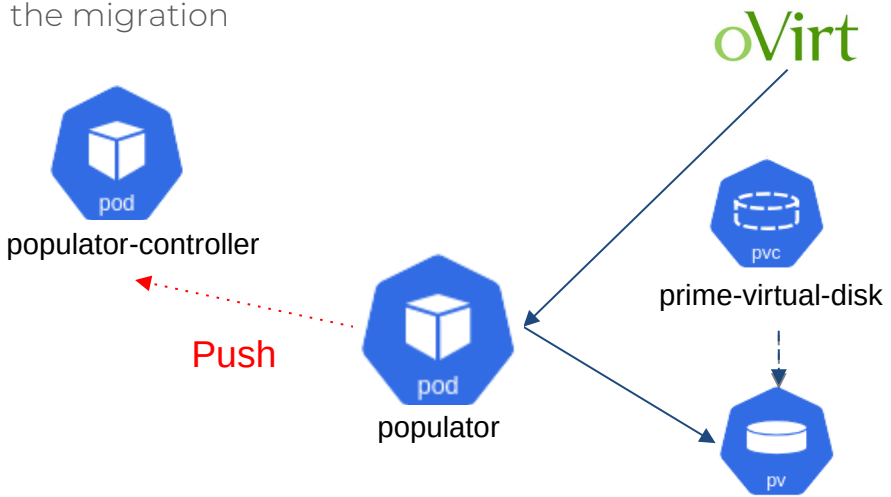
Copying data from remote environments can take significant time (few hours)

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

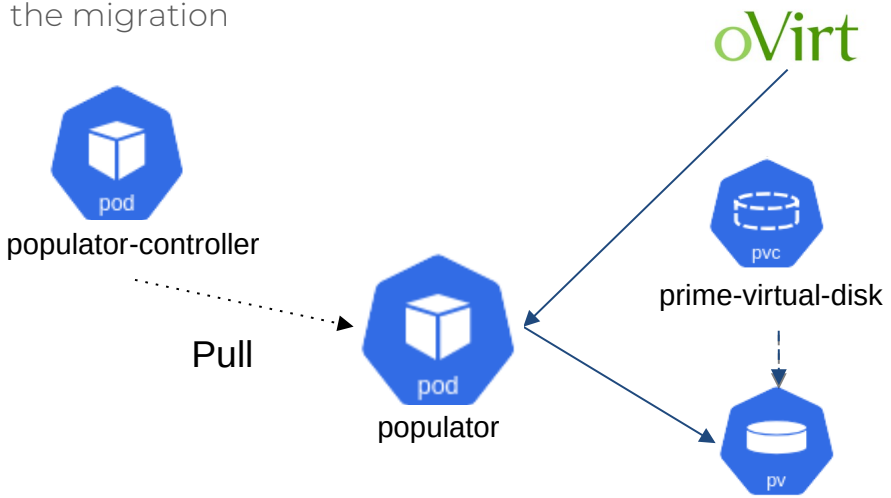
Copying data from remote environments can take significant time (few hours)

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

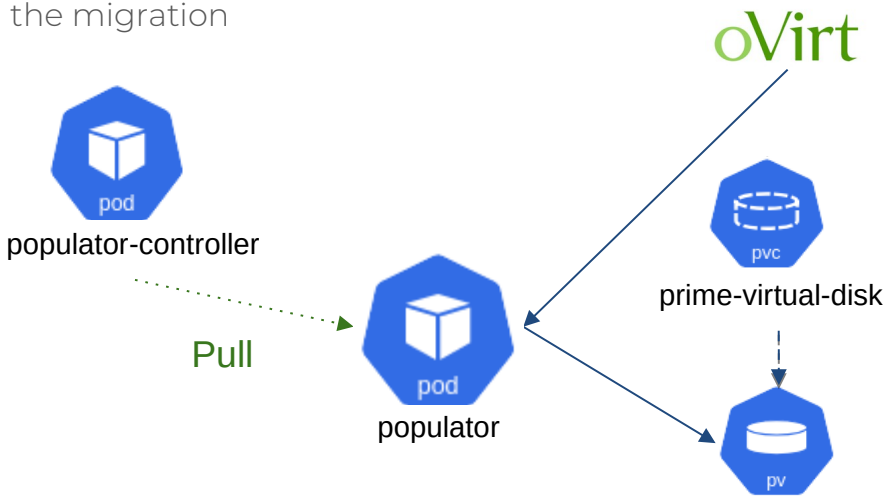
Copying data from remote environments can take significant time (few hours)

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

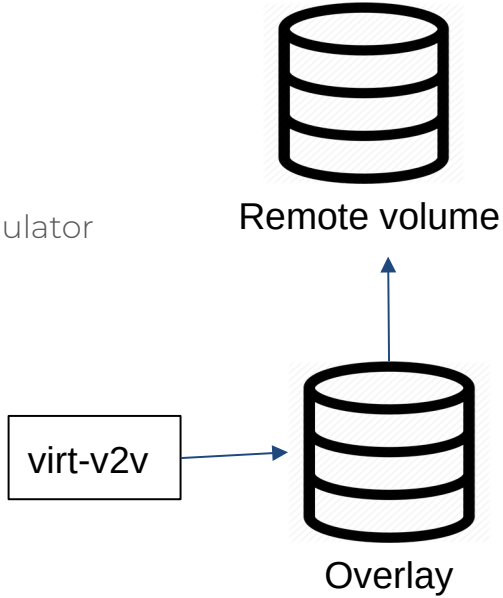
Blocked use of volume populators for statically provisioned storage classes in Forklift

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



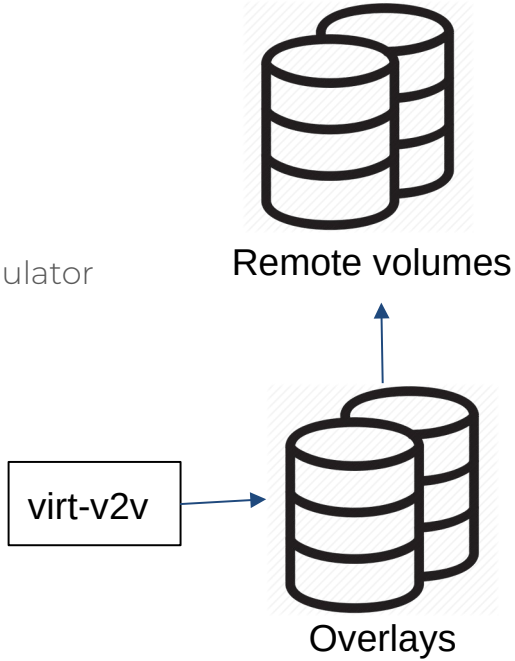
Conversion of multi-volume disks

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

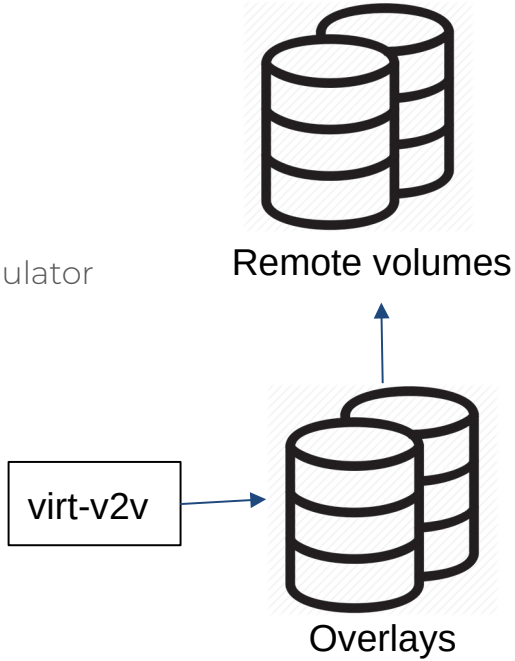
- To inspect the content of a disk
- To modify the content of a disk

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

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

Questions?

Thank you!