Virtual machine import to KubeVirt

Operator SDK use case

@jakubdzon Jakub Dżon

About me

- Senior Software Engineer at Red Hat
- One of **vm-import-operator** developers
- Polish Java User Group co-leader
- GeeCON conference co-founder

Agenda

- Keywords
- The Operator SDK – introduction
- DEMO: Working with the Operator SDK
- The Virtual Machine Import Operator
- DEMO: Virtual Machine import from oVirt to KubeVirt
- Virtual Machine Import Operator design
- Summary
- Q&A

Keywords


oVirt

KubeVirt

CDI

CR

CRD

Controller

Operator

HCO

Operator SDK

https://github.com/operator-framework/operator-sdk

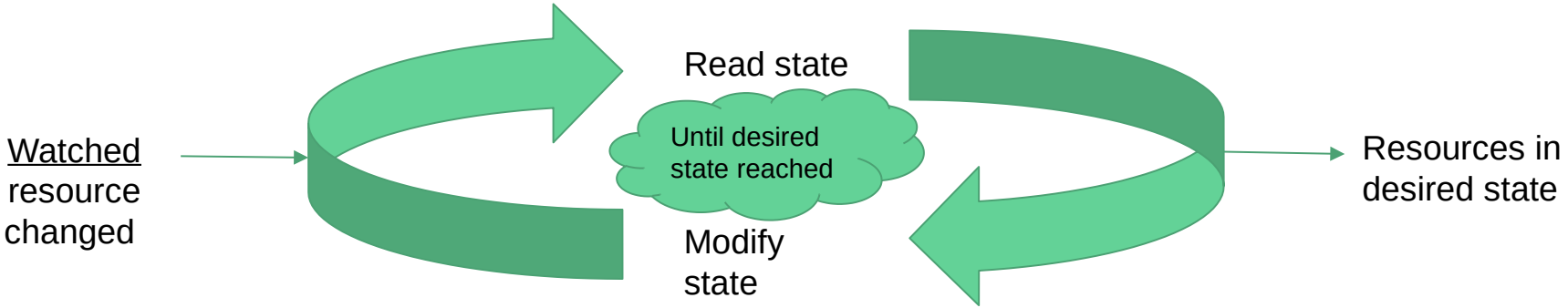
Operator Framework

1. Operator SDK
2. Operator Lifecycle Manager
3. Operator Hub – operatorhub.io

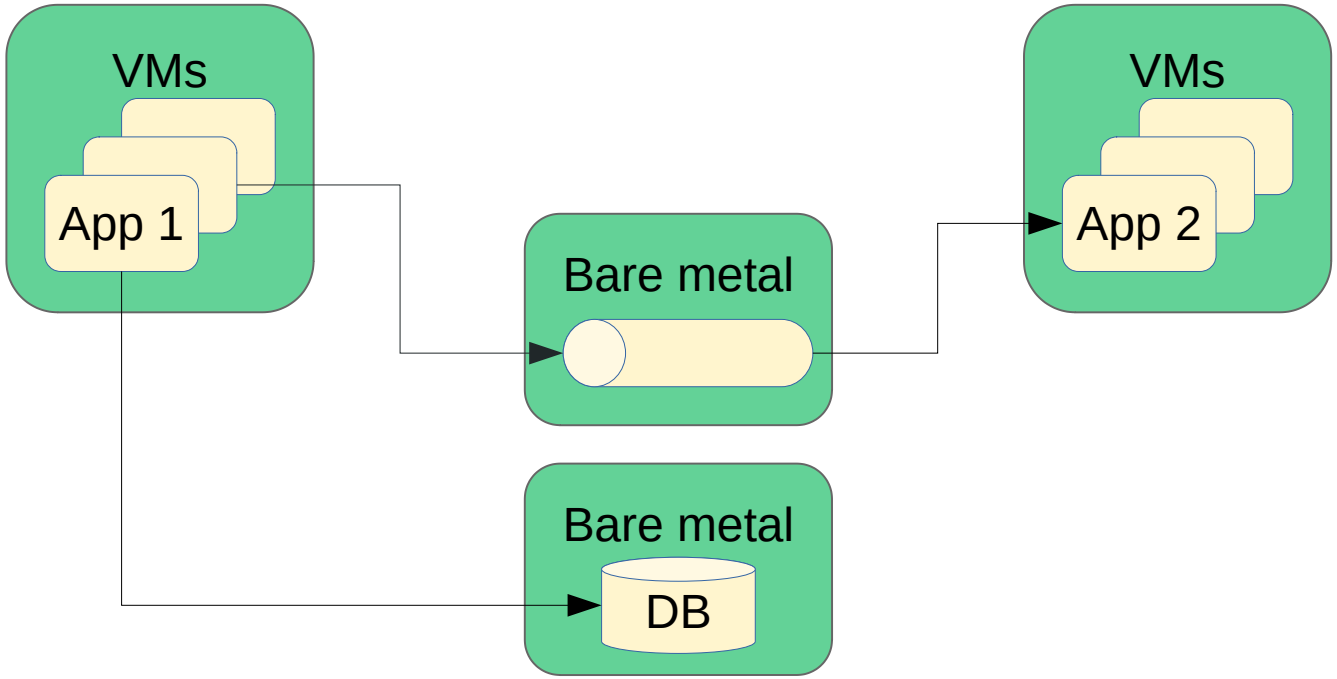
Operator SDK

- Hides Kubernetes-related complexity
- Provides robust scaffolding
- Developers can focus on writing the “business logic” by filling the gaps in the generated code

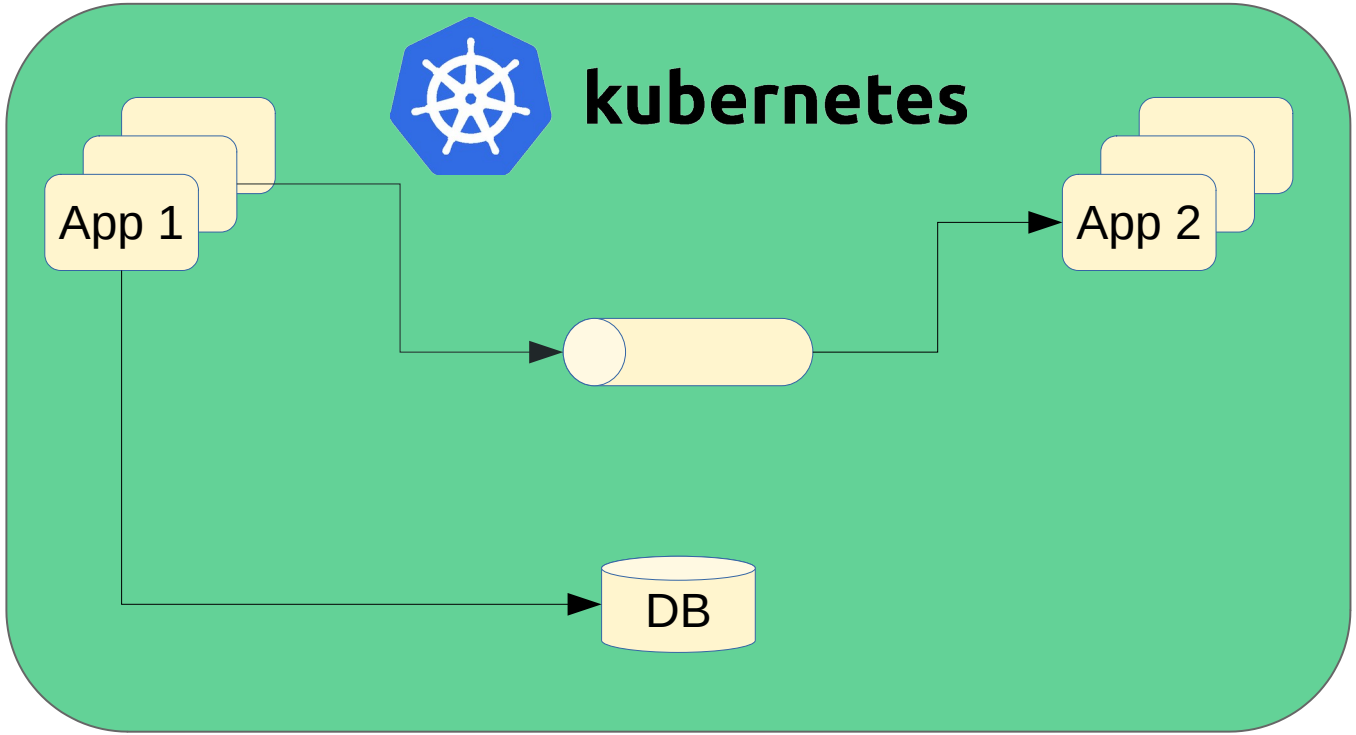
Operator SDK - reconciliation



Operator SDK - reconciliation



Operator SDK - reconciliation



Working with Operator SDK

1. Bootstrap the operator project

Working with Operator SDK

1. Bootstrap the operator project
2. Generate API

Working with Operator SDK

1. Bootstrap the operator project
2. Generate API
3. Fill the generated model with domain-specific details

Working with Operator SDK

1. Bootstrap the operator project
2. Generate API
3. Fill the generated model with domain-specific details
4. Generate manifests

Working with Operator SDK

1. Bootstrap the operator project
2. Generate API
3. Fill the generated model with domain-specific details
4. Generate manifests
5. Create the reconciliation code

Working with Operator SDK

1. Bootstrap the operator project
2. Generate API
3. Fill the generated model with domain-specific details
4. Generate manifests
5. Create the reconciliation code
6. Run it!

DEMO

The Virtual Machine Import Operator

https://github.com/kubevirt/vm-import-operator

The Virtual Machine Operator

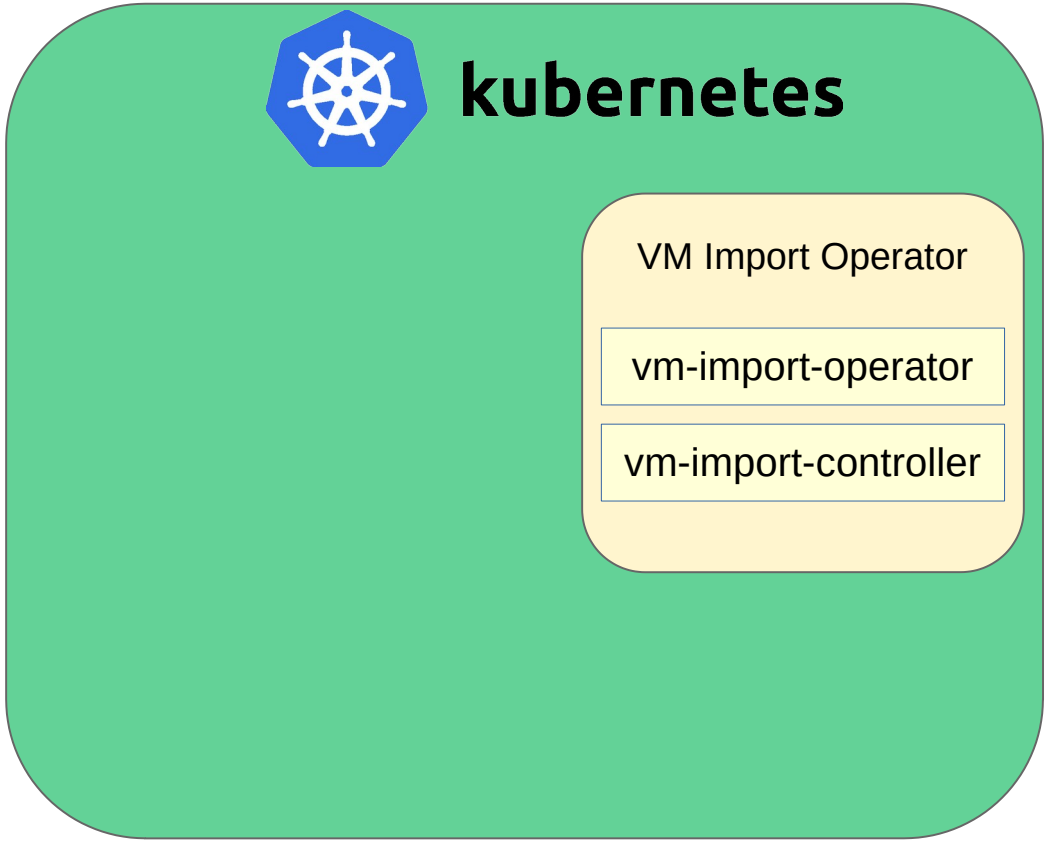
Imports compatible virtual machines to Kubevirt

- from KVM hypervisor managed by oVirt by copying disks as they are - using CDI;
- from VMware hypervisor by copying and converting the virtual machine disks using CDI and virt-v2v

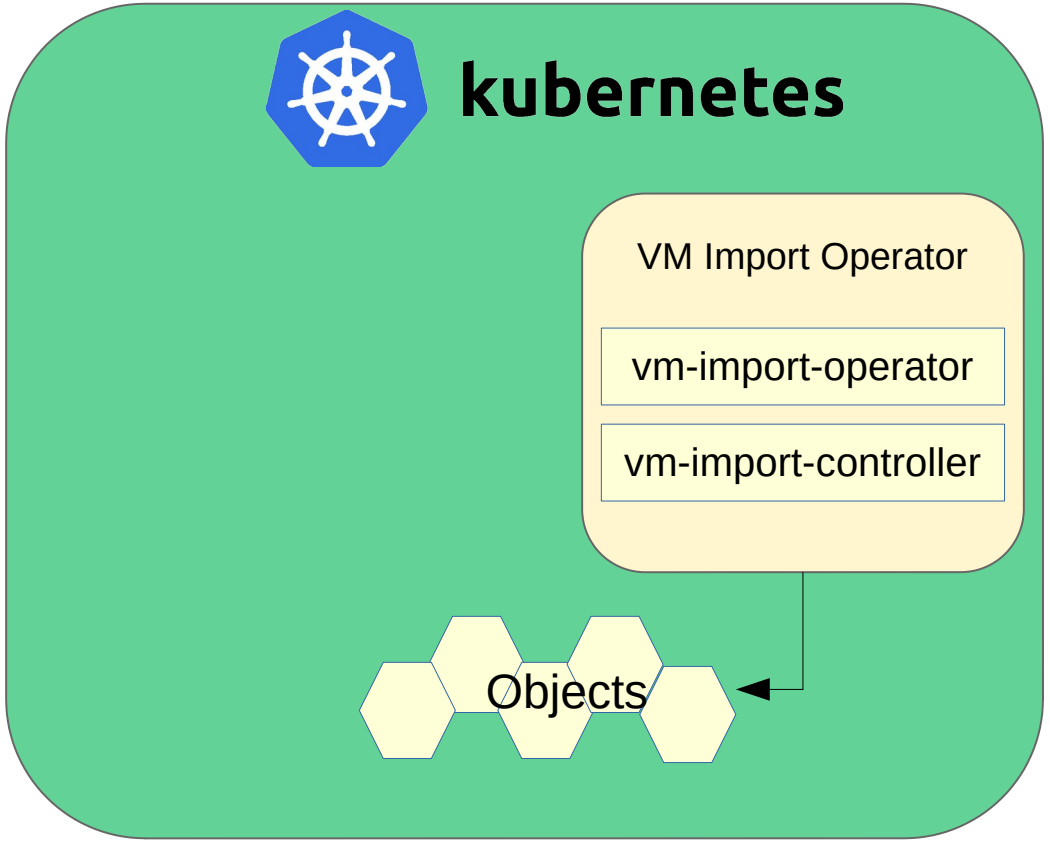
The Virtual Machine Operator

Written in Go using Operator SDK

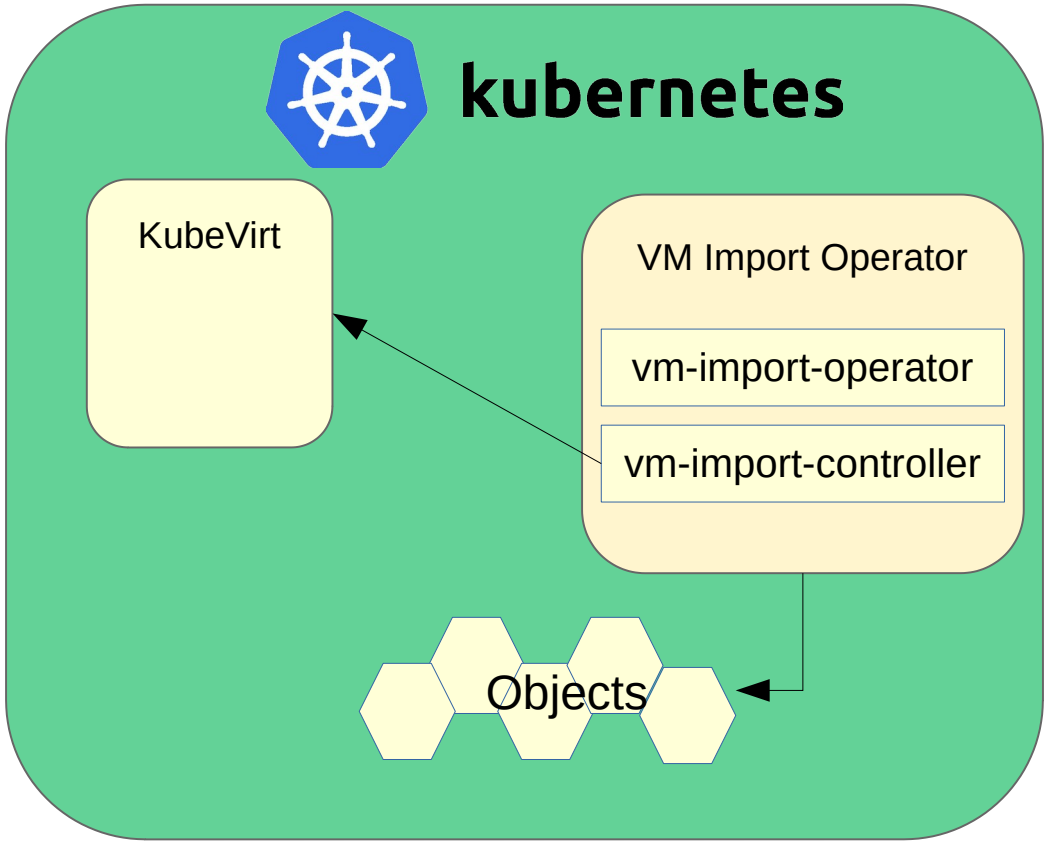
The Virtual Machine Operator



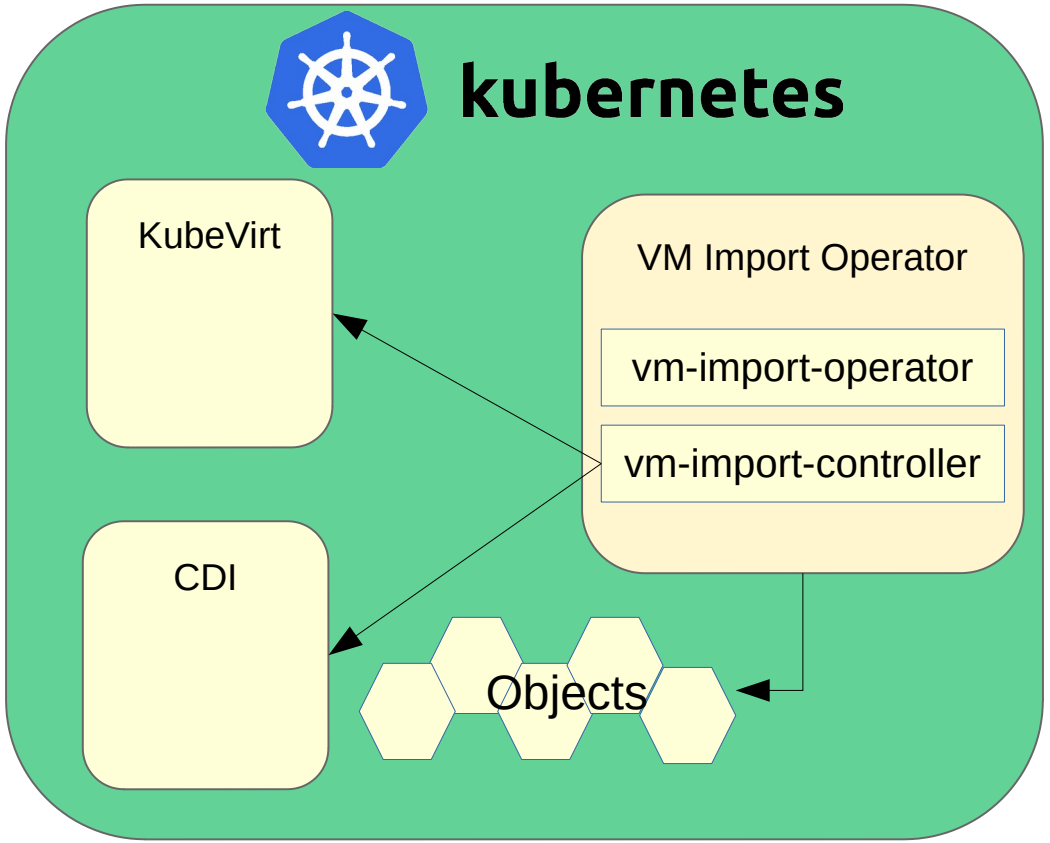
The Virtual Machine Operator



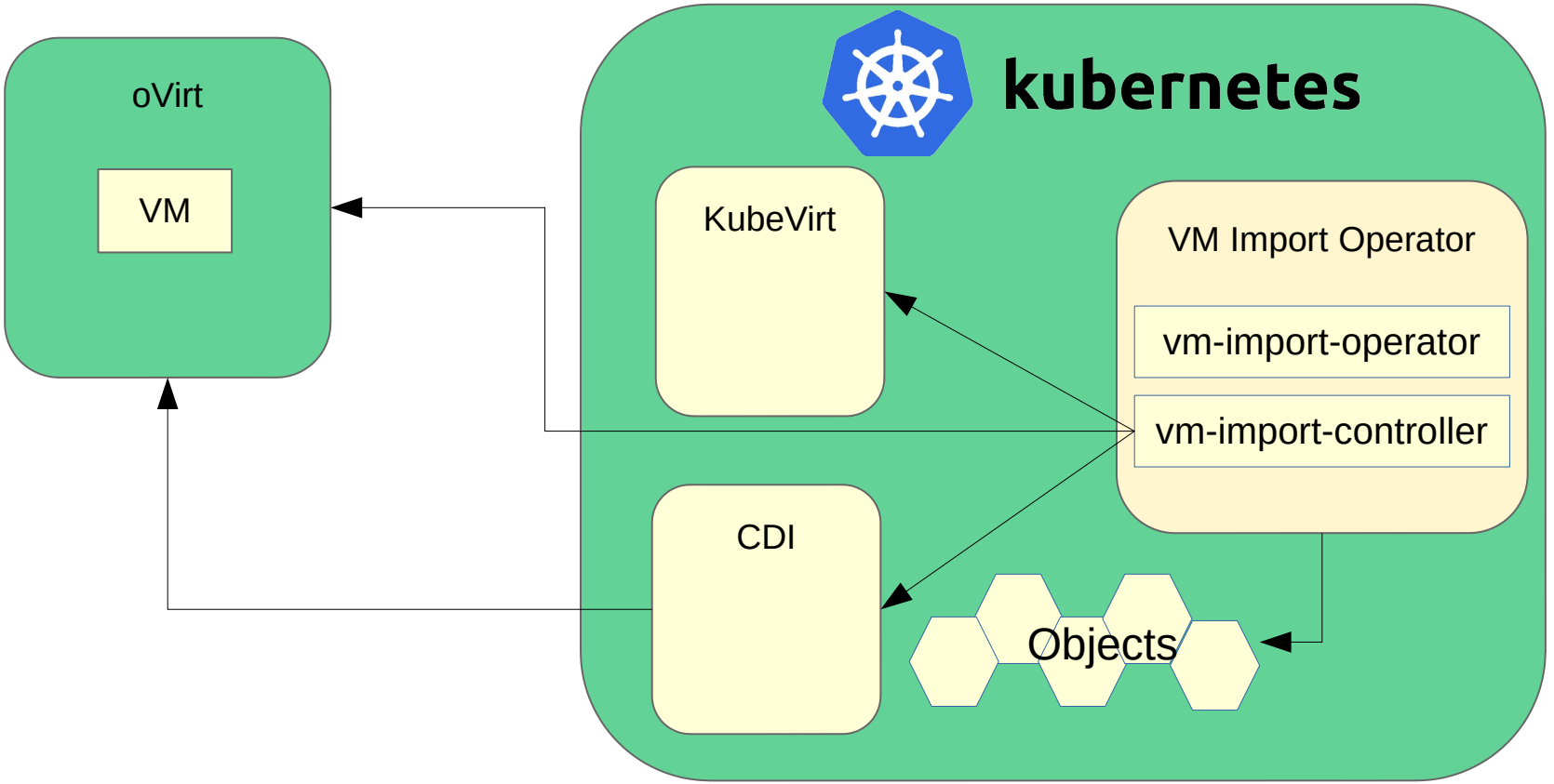
The Virtual Machine Operator



The Virtual Machine Operator



The Virtual Machine Operator



DEMO

Virtual Machine Import Operator design

Virtual Machine Import Pods

1. vm-import-controller
2. vm-import-operator

Virtual Machine Import: vm-import-controller

Executes the virtual machine import

Virtual Machine Import: vm-import-operator

Manages the lifecycle of the controller

Virtual Machine Import: vm-import-operator

Built using

controller-lifecycle-operator-sdk

https://github.com/kubevirt/controller-lifecycle-operator-sdk

SDK for building Controller Lifecycle Operators

Library helping in building operators that manage lifecycle of Kubernetes applications.

SDK for building Controller Lifecycle Operators

- API
- SDK
- Callbacks
- Resources
- OpenAPI
- Reconciler

SDK for building Controller Lifecycle Operators

```go
// Status represents status of a operator configuration resource; must be inlined in the
// operator configuration resource status
type Status struct {
    Phase Phase `json:"phase,omitempty"`
    // A list of current conditions of the resource
    Conditions []conditions.Condition `json:"conditions,omitempty" optional:"true"`
    // The version of the resource as defined by the operator
    OperatorVersion string `json:"operatorVersion,omitempty" optional:"true"`
    // The desired version of the resource
    TargetVersion string `json:"targetVersion,omitempty" optional:"true"`
    // The observed version of the resource
    ObservedVersion string `json:"observedVersion,omitempty" optional:"true"`
}
```

SDK for building Controller Lifecycle Operators

```go
// CrManager defines interface that needs to be provided for the reconciler to operate
type CrManager interface {
    // IsCreating checks whether creation of the managed resources will be executed
    IsCreating(cr controllerutil.Object) (bool, error)
    // Creates empty CR
    Create() controllerutil.Object
    // Status extracts status from the cr
    Status(cr runtime.Object) *sdkapi.Status
    // GetAllResources provides all resources managed by the cr
    GetAllResources(cr runtime.Object) ([]runtime.Object, error)
    // GetDependantResourcesListObjects returns resource list objects of dependant resources
    GetDependantResourcesListObjects() []runtime.Object
}
```

SDK for building Controller Lifecycle Operators

https://github.com/kubevirt/controller-lifecycle-operator-sdk/tree/master/examples/sample-operator

```
// SampleConfigStatus defines the observed state of SampleConfig
type SampleConfigStatus struct {
    sdkapi.Status `json:",inline"`
}
```

SDK for building Controller Lifecycle Operators

https://github.com/kubevirt/controller-lifecycle-operator-sdk/tree/master/examples/sample-operator

```go
// IsCreating checks whether creation of the managed resources will be executed
func (m *CrManager) IsCreating(cr controllerutil.Object) (bool, error) {
    config := cr.(*v1alpha1.SampleConfig)
    return config.Status.Conditions == nil || len(config.Status.Conditions) == 0, nil
}

// Create creates empty CR
func (m *CrManager) Create() controllerutil.Object {
    return new(v1alpha1.SampleConfig)
}

// Status extracts status from the cr
func (m *CrManager) Status(cr runtime.Object) *sdkapi.Status {
    return &cr.(*v1alpha1.SampleConfig).Status.Status
}
```

Virtual Machine Import API

1. **VirtualMachineImport** CRD
2. **ResourceMapping** CRD
3. Kubernetes **Secret**

Virtual Machine Import - reconciliation

1. **Fetch VM metadata from the provider (oVirt or VMware)**

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. **Validate source VM**

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. **Stop source VM**

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. Stop source VM
4. **Create disk-less KubeVirt VM**

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. Stop source VM
4. Create disk-less KubeVirt VM
5. **Create DataVolumes**

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. Stop source VM
4. Create disk-less KubeVirt VM
5. Create DataVolumes
6. **Monitor DataVolumes import until done**

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. Stop source VM
4. Create disk-less KubeVirt VM
5. Create DataVolumes
6. Monitor DataVolumes import until done
7. ***Convert guest [VMware only]***

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. Stop source VM
4. Create disk-less KubeVirt VM
5. Create DataVolumes
6. Monitor DataVolumes import until done
7. *Convert guest [VMware only]*
8. ***Monitor guest conversion until done [VMware only]***

Virtual Machine Import - reconciliation

1. Fetch VM metadata from the provider (oVirt or VMware)
2. Validate source VM
3. Stop source VM
4. Create disk-less KubeVirt VM
5. Create DataVolumes
6. Monitor DataVolumes import until done
7. *Convert guest [VMware only]*
8. *Monitor guest conversion until done [VMware only]*
9. **Start target VM**

Summary

Thank you!

@jakubdzon

Jakub Dżon

Q&A

@jakubdzon

Jakub Dżon