Simplifying KubeVirt: New tools for easier VM management

FOSDEM 2025

Virtualization and Cloud Infrastructure

Felix Matouschek

About me

Felix Matouschek
fmatouschek@redhat.com
Software Engineer @ Red Hat

https://github.com/0xFelix
https://matouschek.org

Agenda

- Introduction to KubeVirt
- Enhancements to the KubeVirt CLI
- Automating the VM lifecycle with Ansible
- Demo
- Next steps

Introduction to KubeVirt

- Run and manage VMs on Kubernetes
- Bring VMs into the container world
- Combine virtualized and containerized workloads

Go to https://kubevirt.io or https://kubevirt.io/user-guide to learn more

KubeVirt VirtualMachine

```yaml
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: vm-d9qgd
spec:
  runStrategy: Always
  template:
    spec:
      domain:
        devices: {}
        memory:
          guest: 512Mi
      terminationGracePeriodSeconds: 180
      volumes:
      - containerDisk:
          image: quay.io/containerdisks/fedora:latest
        name: vm-d9qgd-containerdisk-0
```

Go to https://kubevirt.io/api-reference to learn more

Reminder: Instancetypes and preferences

- CRDs combining resource sizing or runtime preference settings
- One of each can be referenced by `VirtualMachines`
- KubeVirt ships pre-defined instancetypes and preferences

Go to https://github.com/kubevirt/common-instancetypes to learn more

KubeVirt's CLI: virtctl

- Control the lifecycle of VMs
- Create VMs, Instancetypes and Preferences
  - Fixed set of CLI flags to adjust parameters
  - Outputs manifests which can be piped into `kubectl` or `oc`
- Other quality of life features
  - Remote access to SSH, SCP and VNC

Enhancements in the KubeVirt 1.4 cycle

- `virtctl create vm` gained several new features
  - Generating `cloud-init` configurations
  - Injecting credentials
  - Support for `sysprep` volumes
- Many more clean ups and fixes
  - Unified volume import

Generating cloud-init configurations

- Login and use your VM right after creating it

```
$ virtctl create vm --user=myuser --ssh-key="$(<~/.ssh/id_ed25519.pub)"
```

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
[...]
      volumes:
      - cloudInitNoCloud:
          userData: |-
            #cloud-config
            user: myuser
            ssh_authorized_keys:
              - ssh-ed25519 AAAA...
        name: cloudinitdisk
[...]
```

Go to https://kubevirt.io/user-guide/user_workloads/creating_vms to learn more

Injecting credentials

- Dynamically inject public keys or passwords from a `Secret`

```
$ virtctl create vm --user=myuser --access-cred=src:my-keys
```

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
[...]
      accessCredentials:
      - sshPublicKey:
          propagationMethod:
            qemuGuestAgent:
              users:
              - myuser
          source:
            secret:
              secretName: my-keys
[...]
```

Automating the VM lifecycle

- Ansible is a ubiquitous tool to automate IT tasks
- KubeVirt now provides the `kubevirt.core` collection
  - Lean bindings that support common tasks
  - Reuse existing tooling and approaches
  - Manage day 2 activites at scale

Go to https://github.com/kubevirt/kubevirt.core to learn more

Manage your VMs with Ansible

- `kubevirt.core.kubevirt_vm`
  - Create, update or delete VMs
  - Supports changing most parameters found on VMs
  - Wait for changes to be propagated
- `kubevirt.core.kubevirt_{vm,vmi}_info`
  - Look up VMs
  - Gather information about a specific VM
  - Read-only

Populate your inventory dynamically

- `kubevirt.core.kubevirt` inventory plugin
  - Fetch and update inventory data from a live cluster
  - Allows Ansible to interact with KubeVirt VMs
    - Tries to discover SSH connectivity automatically
  - Designed to handle large-scale environments
    - Group and filter by cluster name, namespaces and labels

Demo



1. How to prepare the environment
2. Creating an instancetype and preference with `virtctl`
3. Creating a VM with Ansible
4. Populating the Ansible inventory
5. Running a playbook deploying an application

Find demo files at https://github.com/0xFelix/fosdem2025-demo

How to prepare the environment

- Prerequisites: Ansible, Docker or Podman, kubectl, virtualization enabled

1. Install collection from Ansible Galaxy

```
$ ansible-galaxy collection install kubevirt.core
```

2. Install virtctl

```
$ curl -L https://github.com/kubevirt/kubevirt/releases/download/v1.4.0/virtctl-v1.4.0-linux-amd64 -o ~/.local/bin/virtctl
$ chmod +x ~/.local/bin/virtctl
```

3. Checkout the repository and bring up a cluster

```
$ git clone https://github.com/kubevirt/kubevirt.core.git
$ cd kubevirt.core && make cluster-up
```

Creating an instancetype with virtctl

```
$ virtctl create instancetype --name my-it --cpu 2 --memory 2Gi | kubectl create -f -
```

```
apiVersion: instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterInstancetype
metadata:
  name: my-it
spec:
  cpu:
    guest: 2
  memory:
    guest: 2Gi
```

Creating a preference with virtctl

```
$ virtctl create preference --name my-pref --cpu-topology cores | kubectl create -f -
```

```
apiVersion: instancetype.kubevirt.io/v1beta1
kind: VirtualMachineClusterPreference
metadata:
  name: my-pref
spec:
  cpu:
    preferredCPUTopology: cores
```

Creating a VM with Ansible

```
$ ansible-playbook create-testvm.yml
```

- Using the `kubevirt.core.kubevirt_vm` module
- Applying our instancetype and preference
- Adding a secondary network for connectivity
- Statically injecting an SSH public key

Playbook: https://github.com/0xFelix/fosdem2025-demo/blob/main/create-testvm.yml

Populating the Ansible inventory

Inventory configuration

```
$ cat inventory.kubevirt.yml
plugin: kubevirt.core.kubevirt
namespaces:
  - default
network_name: bridge-network
```

Listing discovered hosts

```
$ ansible-inventory -i inventory.kubevirt.yml --list
[...]
"hostvars": {
    "default-testvm": {
        "ansible_host": "172.19.1.1",
[...]
```

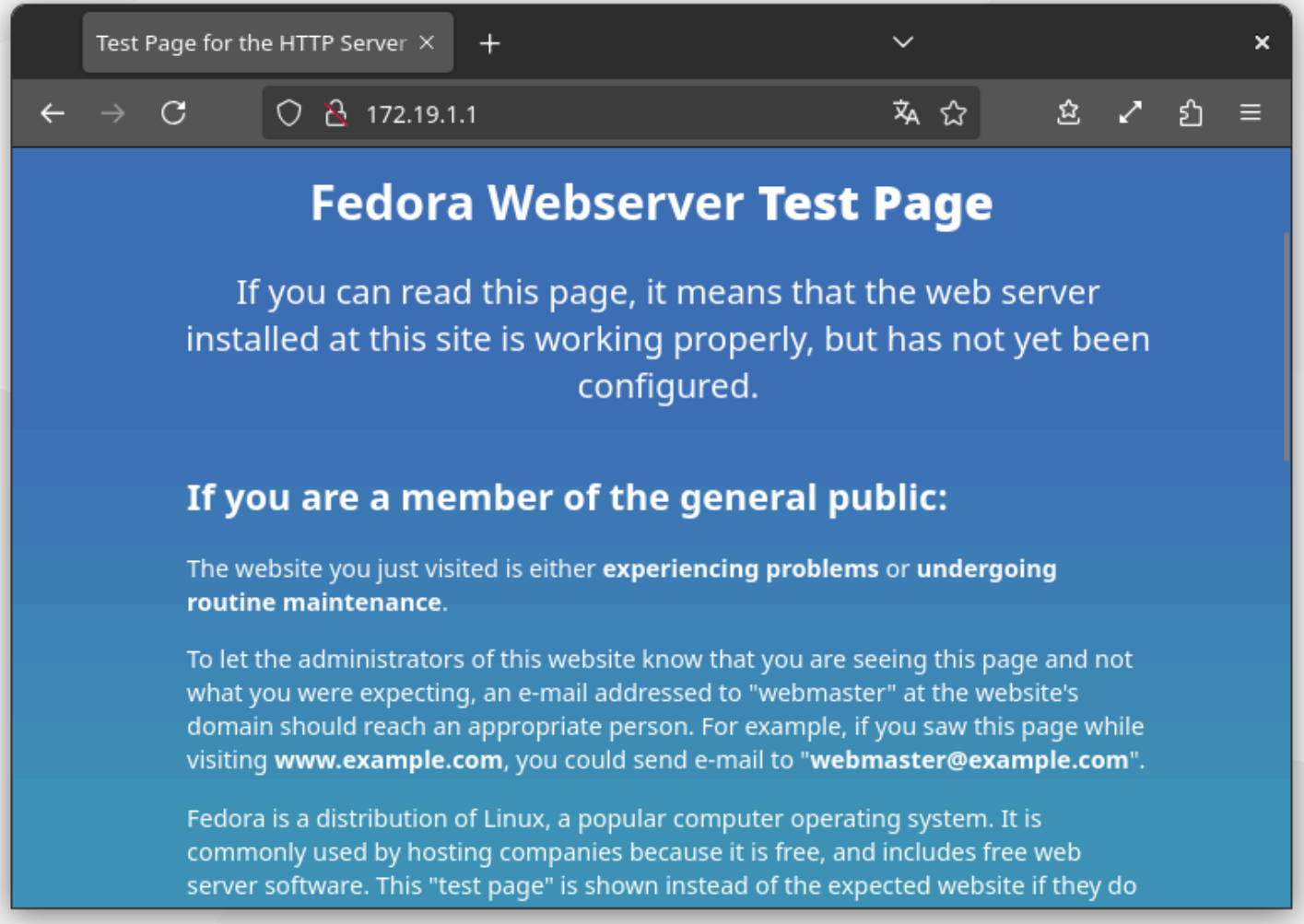
Deploying an application on the VM

```
$ cat httpd.yml
- hosts: default-testvm
  become: true
  tasks:
    - name: Install httpd
      ansible.builtin.package:
        name: httpd
    - name: Start httpd
      ansible.builtin.service:
        name: httpd
        state: started
```

Running the playbook

```
$ ansible-playbook -i inventory.kubevirt.yml -u fedora --private-key mykey httpd.yml
```

It works!


Test Page for the HTTP Server
172.19.1.1
Fedora Webserver Test Page
If you can read this page, it means that the web server installed at this site is working properly, but has not yet been configured.
If you are a member of the general public:
The website you just visited is either experiencing problems or undergoing routine maintenance.
To let the administrators of this website know that you are seeing this page and not what you were expecting, an e-mail addressed to "webmaster" at the website's domain should reach an appropriate person. For example, if you saw this page while visiting www.example.com, you could send e-mail to "webmaster@example.com".
Fedora is a distribution of Linux, a popular computer operating system. It is commonly used by hosting companies because it is free, and includes free web server software. This "test page" is shown instead of the expected website if they do

Next steps

- Make `virtctl` easier to extend
- Address usability issues in `kubevirt.core`
- Carefully add new features to the collection
  - Support fact caching
  - We need your feedback!

Questions?