vmadm – virtual machine administration

Lars Wirzenius

2022-06-18 08:31

Contents

1 Data files for scenarios

This section has some data files used by scenarios.

File: **smoke.yaml**

```
smoke: {}
```

File: **other.yaml**

```
other: {}
```

File: **ssh_key**

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEAoWeplbhiwaVu0mqxVP07uk3pqrjcWrWzFGbk5PEalbW5L0ynlhWe
YebZ7Vjx4Ek/MpGBWiK6/HmLikJCnQcR1ux/JHo0zcEbv6w20WF+cMU5+I8OEVaKRk6cTJ
Rq1nruQpFj6CwIULSM81AJ6wxqfbKhuj9RuKISOtcGTBULWvUdrjcV553rPntq+GO/BsZp
UB/6NKLiPHwZ7MUStCKEnxbNi7rTusI8s0efRXQvU0+8Ln3eZFzEc8bJjxnl8zXGqrxxsQ
bObnBhXJQpKkkcJNxjRKV2UIZQiKEyKEbtCDwKf+N9LZcfcyTIYcA5WbDjl0axWKXVlTEk
wAChAi0mMQAAA8jfx6KY38eimAAAAAdzc2gtcnNhAAABAQChZ6mVuGLBpW7SarFU/Tu6Te
mquNxatbMUZuTk8RqVtbkvTKeWFZ5h5tntWPHgST8ykYFaIrr8eYuKQkKdBxHW7H8kejTN
wRu/rDbRYX5wxTn4jw4RVopGTpxMlGrWeu5CkWPoLAhQtIzzUAnrDGp9sqG6P1G4ohI61w
ZMFQta9R2uNxXnnes+e2r4Y78GxmlQH/o0ouI8fBnsxRK0IoSfFs2LutO6wjyzR59FdC9T
T7wufd5kXMRzxsmPGeXzNcaqvHGxBvRucGFclCkqSRwk3GNEpXZQhlCIoTIoRu0IPAp/43
0tlx9zJMhhwDlZsOOXRrFYpdWVMSTAAKECLSYxAAAAAwEAAQAAAQAJ417pVD2AnZD3hR/O
FCGHnWRWWDLvv7fz5QXa3MaDK3nn4utVb4efedQaDVvsILCleXKSQhRwiUW6N6r7EcbPAv
gbFP2NKWp4yKUnGLD1Wa/egW0cNNAN1J0Qt/r/ntJf86ZKQABWaWlFMr8Yzk7r2ni7/0sT
Y6J4RloVaSij7s1uZ76sTw/REGF/BNX0BC1FTDlQE3jTQptEYxGbLGQFYYhDl37Zv3Emnf
j7ZA8pkwrUn6mPy5JEZTjp2MgFD8oF8XxzfXWtFJP9UuDUJrLcQD8hOunUrPQbnazpxOHw
0vvF7K7B9l0mLJX7UmQlBW8Op+tv1jJugJ29rDjT3FcRAAAAgFtDhZCLc2ihQiOK2zEqb5
bUk4x9a42othQQNOvMEkwCLxmKTVjoYrClSp+9j6blkKESiGCxAu8MC7Hc8JV2NOkIrorl
K35KgwCiLyEXPedQ/+ZPoO1a4ZIGHPfxosbmh9byJYgvDQ4E8gqRUlEhtYSOzyeObfXh7T
7QcNtKOvo+AAAAgQDVK194Jl97URrb23+jgFPfWeNb1daeLUc/DYCDaJgHVom8zAxKPsHP
sYdy5dPGNTiMttdk2JpzqAmo7G1/QDPB5sHNcWEAQEOPGagYGFJhBMhB9Ug6iJek14h9nS
/m6BfVy3fQW062sztE7dw5lzGTTQajJWX7z4VDm4VSaraHXQAAAIEAwdW4Xk8xS2Jdp3Gr
/+OYG4+90rLpidS1z0SgT+a8NeLA3KXwDhdunHDxGn9QKgSCZ8ogHmcGN4x07jI3+3ajdK
7Xe/QIlNLAnXEX7Kkrx3+FmakCXs/aN5xTA0J6s2Hyj9MwJIX0C+EmAzxEcIbHvbKxKVBp
V4cecTlFJGBtUOUAAAAMbGl3QGV4b2xvYmUxAQIDBAUGBw==
-----END OPENSSH PRIVATE KEY-----
```

Note that we use *tilde expansion* in filenames to indicate they go into the home directory. The Subplot test runner sets the `HOME` environment variable to a suitable directory, so that the files don't go into the actual home directory of the person running the generated test program

File: **config.yaml**

```yaml
image_directory: ~/images
default_base_image: ~/base.qcow2
default_image_gib: 5
default_memory_mib: 2048
default_cpus: 1
default_generate_host_certificate: true
default_autostart: true
ca_key: ~/ca_key
user_ca_pubkey: ~/user_ca_pubkey
authorized_keys:
  - ~/.ssh/id_rsa.pub
```

File: **fullconfig.json**

```
{
    "image_directory": "~/images",
    "default_base_image": "~/base.qcow2",
    "default_image_gib": 5,
    "default_memory_mib": 2048,
    "default_cpus": 1,
    "default_generate_host_certificate": true,
    "default_autostart": true,
    "default_networks": [
        "network=default"
    ],
    "default_allow_authorized_keys": null,
    "ca_key": "~/ca_key",
    "user_ca_pubkey": "~/user_ca_pubkey",
    "authorized_keys": [
        "~/.ssh/id_rsa.pub"
    ]
}
```

File: **spec.yaml**

```
foo:
  ca_key: ~/other_ca
```

File: **fullspec.json**

```
[
    {
        "name": "foo",
        "ssh_keys": [
            "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQChZ6mVuGLBpW7SarFU/Tu6TemquNxatbMUZuTk8Rc
        ],
        "networks": ["network=default"],
        "rsa_host_key": null,
        "rsa_host_cert": null,
        "dsa_host_key": null,
        "dsa_host_cert": null,
        "ecdsa_host_key": null,
        "ecdsa_host_cert": null,
        "ed25519_host_key": null,
        "ed25519_host_cert": null,
        "base": "~/base.qcow2",
        "image": "~/images/foo.qcow2",
        "image_size_gib": 5,
        "memory_mib": 2048,
        "cpus": 1,
        "generate_host_certificate": true,
        "autostart": true,
```

```
        "ca_key": "~/other_ca",
        "user_ca_pubkey": "~/user_ca_pubkey",
        "allow_authorized_keys": true
    }
]
```

File: **ssh__key__pub**

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQChZ6mVuGLBpW7SarFU/Tu6TemquNxatbMUZuTk8RqVtbkvTKeWFZ5
```

File: **ca__key**

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACABAgbX2ZOvZUO42nZDbKYOaovzfaSH1uiXKjBFydy2igAAAJBWl8ZtVpfG
bQAAAAtzc2gtZWQyNTUxOQAAACABAgbX2ZOvZUO42nZDbKYOaovzfaSH1uiXKjBFydy2ig
AAAECD6VUD9Cl/oDBtGumplYGWkbYCWXTFDAb6CaeXyf1ErQECBtfZk69lQ7jadkNspg5q
i/N9pIfW6JcqMEXJ3LaKAAAADGxpd0BleG9sb2JlMQE=
-----END OPENSSH PRIVATE KEY-----
```

File: **ssh__config**

```
host *
  userknownhostsfile=ssh/known_hosts
  stricthostkeychecking=accept-new
  identityfile=.ssh/id_rsa
  identitiesonly=yes
  passwordauthentication=no
```

File: **known__hosts**

```
@cert-authority * ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAECBtfZk69lQ7jadkNspg5qi/N9pIfW6JcqME
```

2 Cloud-init configuration

This scenario verifies that vmadm creates the cloud-init configuration correctly.

given an installed vmadm
and file **init.yaml**
and file **config.yaml**
and file **.ssh/id__rsa.pub** from **init__ssh__key__pub**
and file **user__ca__pubkey** from **ssh__key__pub**
and file **expected/init-test/meta-data** from **init-metadata**
and file **expected/init-test/user-data** from **init-userdata**
when I run **vmadm cloud-init --config config.yaml init.yaml actual**
then directories **actual/init-test** and **expected/init-test** are identical

File: **init.yaml**

```
init-test:
  ssh_key_files:
  - .ssh/id_rsa.pub
  rsa_host_key: rsa-private
  rsa_host_cert: rsa-certificate
  dsa_host_key: dsa-private
  dsa_host_cert: dsa-certificate
  ecdsa_host_key: ecdsa-private
  ecdsa_host_cert: ecdsa-certificate
  ed25519_host_key: ed25519-private
  ed25519_host_cert: ed25519-certificate
  base: /home/liw/tmp/debian-10-openstack-amd64.qcow2
  image: images/init.qcow2
  image_size_gib: 5
  memory_mib: 2048
  cpus: 1
```

File: **init__ssh__key__pub**

```
init-authz-key
```

File: **init-metadata**

```
# Amazon EC2 style metadata
local-hostname: init-test
```

File: **init-userdata**

```
#cloud-config
ssh_authorized_keys:
  - init-authz-key
ssh_keys:
  rsa_private: rsa-private
  rsa_certificate: rsa-certificate
  dsa_private: dsa-private
  dsa_certificate: dsa-certificate
  ecdsa_private: ecdsa-private
  ecdsa_certificate: ecdsa-certificate
  ed25519_private: ed25519-private
  ed25519_certificate: ed25519-certificate
user_ca_pubkey: >
  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQChZ6mVuGLBpW7SarFU/Tu6TemquNxatbMUZuTk8RqVtbkvTKeWFZ
allow_authorized_keys: true
```

3 Create a virtual machine

This scenario verifies that vmadm can create a virtual machine and that the user can log into it as root via SSH after it has booted. This requires that the

environment it set up so that virtual machines can be addressed by name.

First some setup.

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **smoke.yaml**
and file **config.yaml**
and file **ca_key**
and file **user_ca_pubkey** from **ssh_key_pub**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
when I run **chmod -R u=rwX,go= .ssh**

Then we create the VM, and verify that we can log in.

when I run **vmadm new --config config.yaml smoke.yaml**
and I run **ssh -F .ssh/config debian@smoke hostname**
then stdout contains "**smoke**"
when I run **ssh -F .ssh/config debian@smoke df -h /**
then stdout matches regex **4\.\dG**
when I run **ssh -F .ssh/config debian@smoke free -m**
then stdout matches regex **Mem:\s+19\d\d\s**

Then we shut it down, twice. The second time is to verify shutting down works even if the VM is already shut down.

when I run **vmadm shutdown --config config.yaml smoke.yaml**
and I run **vmadm shutdown --config config.yaml smoke.yaml**

Then we start it back up again and verify we can log in. Then we start it again, while it's already running, to verify that that works.

when I run **vmadm start --config config.yaml smoke.yaml**
and I run **ssh -F .ssh/config debian@smoke hostname**
and I run **vmadm start --config config.yaml smoke.yaml**

Finally, we delete it twice.

when I run **vmadm delete --config config.yaml smoke.yaml**
and I run **vmadm delete --config config.yaml smoke.yaml**

4 Manage several specs at once

This scenario verifies that vmadm can manage virtual machines from several specifications at once.

First some setup.

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **smoke.yaml**
and file **other.yaml**
and file **config.yaml**
and file **ca_key**
and file **user_ca_pubkey** from **ssh_key_pub**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
when I run **chmod -R u=rwX,go= .ssh**

Then we create the VMs. We don't verify that each works, beyond that we can log in.

when I run **vmadm new --config config.yaml smoke.yaml other.yaml**
and I run **ssh -F .ssh/config debian@smoke hostname**
then stdout contains "**smoke**"
when I run **ssh -F .ssh/config debian@other hostname**
then stdout contains "**other**"

Then we shut them all down.

when I run **vmadm shutdown --config config.yaml smoke.yaml other.yaml**

Then we start them back up again and verify we can log in.

when I run **vmadm start --config config.yaml smoke.yaml other.yaml**
and I run **ssh -F .ssh/config debian@smoke hostname**
and I run **ssh -F .ssh/config debian@other hostname**

Then we recreate them.

when I run **vmadm recreate --config config.yaml smoke.yaml other.yaml**
and I run **ssh -F .ssh/config debian@smoke hostname**
and I run **ssh -F .ssh/config debian@other hostname**

Finally, we delete them.

when I run **vmadm delete --config config.yaml smoke.yaml other.yaml**

5 Give useful error if image for new VM already exists

This scenario verifies that if the VM image file already exists, vmadm gives a useful error message.

given an installed vmadm
and a Debian 10 OpenStack cloud image

and file **smoke.yaml**
and file **config.yaml**
and file **ca_key**
and file **user_ca_pubkey** from **ssh_key_pub**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
and file **images/smoke.qcow2** from **dummy.qcow2**
when I try to run **vmadm new --config config.yaml smoke.yaml**
then exit code is **1**
and stderr contains "**images/smoke.qcow2**"
and stderr contains "**exists**"

File: **dummy.qcow2**

```
This dummy file pretends to be a QCOW2 image file.
```

6 Dump config

This scenario verifies that vmadm can show its actual configuration.

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **.config/vmadm/config.yaml** from **config.yaml**
and file **fullconfig.json**
when I run **vmadm config**
then stdout, as JSON, matches file **fullconfig.json** with tilde expansion

7 Dump specification

This scenario verifies that vmadm can show the actual specification it will use.

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **.config/vmadm/config.yaml** from **config.yaml**
and file **ca_key**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
and file **spec.yaml**
and file **fullspec.json**
when I run **vmadm spec spec.yaml**
then stdout, as JSON, matches file **fullspec.json** with tilde expansion

8 Configure networks

vmadm must allow the user to specify any kind of network that `virt-install` supports, including bridge ones. This scenario verifies that a bridge can be specified.

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **.config/vmadm/config.yaml** from **bridgeconfig.yaml**
and file **ca_key**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
and file **bridgespec.yaml**
when I run **vmadm config**
then stdout contains "**bridge=br0**"
when I run **vmadm spec bridgespec.yaml**
then stdout contains "**bridge=br1**"

File: **bridgeconfig.yaml**

```
image_directory: ~/images
default_base_image: ~/base.qcow2
default_image_gib: 5
default_memory_mib: 2048
default_cpus: 1
default_generate_host_certificate: true
default_autostart: true
ca_key: ~/ca_key
authorized_keys:
  - ~/.ssh/id_rsa.pub
default_networks:
- bridge=br0
```

File: **bridgespec.yaml**

```
foo:
  networks:
  - bridge=br1
```

9 Check that network name is OK

vmadm must check that the virtual network name is OK. The scenarios in this chapter verify that for the configuration and spec files.

9.1 Check network name in config

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **.config/vmadm/config.yaml** from **bad-network-config.yaml**
and file **ca_key**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
when I try to run **vmadm config**
then command fails
and stderr contains "**: br0**"

File: **bad-network-config.yaml**

```
image_directory: ~/images
default_base_image: ~/base.qcow2
default_image_gib: 5
default_memory_mib: 2048
default_cpus: 1
default_generate_host_certificate: true
default_autostart: true
ca_key: ~/ca_key
authorized_keys:
  - ~/.ssh/id_rsa.pub
default_networks:
- br0
```

9.2 Check network name in spec

given an installed vmadm
and a Debian 10 OpenStack cloud image
and file **.config/vmadm/config.yaml** from **config.yaml**
and file **ca_key**
and file **.ssh/id_rsa** from **ssh_key**
and file **.ssh/id_rsa.pub** from **ssh_key_pub**
and file **.ssh/config** from **ssh_config**
and file **.ssh/known_hosts** from **known_hosts**
and file **bad-network-spec.yaml**
when I try to run **vmadm spec bad-network-spec.yaml**
then command fails
and stderr contains "**: br0**"

File: **bad-network-spec.yaml**

```
foo:
  networks:
```

```
3     - br0
```

10 Colophon

This is a document meant to be processed with [Subplot][] into an HTML document, a PDF document, and an executable program.