Berth Documentation

Release 0.1.0

Berth Authors

Jul 11, 2019

Contents

Berth is a deliberately minimalist VM runner for Kubernetes.

CHAPTER 1

Berth

Berth is a deliberately minimalist VM runner for Kubernetes.

I'm not 100% sold on the name; before merging we could change it. . .

1.1 TL;DR Installation Guide

```
# Make sure you have Helm 2.5.x and Kubernetes 1.6.x
#
# edit values.yaml; set class_name and ssh key
#
helm package berth
helm install --name=berth ./berth-0.1.0.tgz # ...
kubectl get pods -o wide
```

You should be able to SSH to your VM at the Kubernetes IP for the container which you can retrieve with *kubectl get all -o wide*. VNC access is available on port 5900.

```
ssh -i ./you-ssh-private-key root@ip.of.vm.pod
```

1.1.1 Example

Quick installation / sample

1.1.2 Why this?

The requirements are very narrow right now and the existing alternatives don't align well at present. This will likely change in time at which point we can realign the internal implementation.

1.1.3 Minimalist requirements

- Run VMs from inside of Kubernetes
- Work with Calico
- Have VM life-cycle match that of pods
- Have VMs benefit from Kubernetes resiliency
- Allow for persistent storage
- Allow for state injection/access from a ConfigMaps

1.2 Requirements

- Helm 2.5.x
- Kubernetes 1.6.x

This does not need to be installed as part of the OpenStack chart collection.

1.3 How it works

At a high level, it works like this:

- Create a SNAT/DNAT enabled linux bridge.
- Assign the bridge a private IP address from a small /30 subnet (controlled with *VM_IP* and *VM_GW*)
- Plug the VM network interface into the bridge.
- Run a dnsmasq process to allocate the VM the right name-servers, and DNS search strings extracted from the parent container. Assign the private IP address to the VM and have it use the bridges IP as its default gateway.
- Setup SNAT/DNAT on the parent container to do 1:1 mapping of all ports, all protocols to the VM, except for TCP:5900 to allow for VNC access (can be controlled with NO_VNC environment variable).
- At this point, VM essentially assumes Pod Assigned IP.
- Feed any meta-data or user-data down into the VM by leveraging these ConfigMap mounts with the same name and turning them into an ISO presented to the guest.

The startvm.sh entry-point supports several environment variables:

- *IMG_SOURCE* which is an http or https URL that contains a qcow2 image. It can also be a full path to a local file baked into the container image, e.g. "/image.qcow"
- *IMG_TARGET* the name to save the image above as in the shared volume.

It also supports two files, which should be mounted as ConfigMaps if using Kubernetes at */userdata* and */metadata* as YAML files containing, obviously meta-data and user-data as YAML that will be fed to the VM as a config-drive iso.

The "pet" version of the image, which is created using qemu-img -b to base it on the source, is stored in a separate volume dedicated to the VM itself, and named after the container hostname.

There are a few other parameters you can control as an operator:

- *VM_IP* is the IP address the VM should be allocated by DHCP. The container will 1:1 NAT except for port 5900 for VNC access (defaults to 192.168.254.2)
- *VM_GW* is the gateway IP address the VM should use for its default route (defaults to 192.168.254.1)