devcloud-kvm

Like devcloud, devcloud-kvm is designed for development purposes. It allows you to run a full cloudstack environment in a VM. The reason to use devcloud-kvm over devcloud would be if you intend to develop and test the KVM agent and host related code. If you want to test the Xen hypervisor, use the normal devcloud.

Note: kindly consider a more modern KVM appliance based CloudStack development with MonkeyBox.

Prerequisites

1. Linux system with kernel 3.1 or newer, kvm_intel or kvm_amd module. Parameter 'nested' should be set to true for module, so your vm can run vms. Consult your platform's modprobe configs for how to force this if necessary. You can check the current setting via "cat /sys/module/kvm_intel /parameters/nested".
2. libvirt and virsh utilities installed.
3. 30GB free disk space. The devcloud-kvm disk image is ~1.4G, but it needs room to grow as you use it. The final size will depend on how much you do.

> **Mac Users**
>
> Note that VMware Fusion 5 supports nesting as well. Follow the instructions (INSTALL-HOWTO.txt) found bundled with the vm package in devcloud-kvm-fusion.tar.gz (CentOS) or devcloud-kvm-fusion-ubuntu.tar.gz (Ubuntu 12.04) for the 'Setup' portion, and then move on to 'Building'.

Setup

Use a pre-created image

1. Download devcloud-kvm.tar from here (CentOS) or here (Ubuntu).
2. Extract devcloud-kvm.tar to your preferred location
3. cd into devcloud-kvm, define networks:

```
virsh net-define network-devcloud-kvm-0.xml
virsh net-define network-devcloud-kvm-1.xml
virsh net-start devcloud-kvm-0
virsh net-start devcloud-kvm-1
virsh net-autostart devcloud-kvm-0
virsh net-autostart devcloud-kvm-1
```

4. Open devcloud-kvm.xml, change file path of the qcow2 to point to the path on your machine
5. Define and start VM:

```
virsh define devcloud-kvm.xml
virsh start devcloud-kvm
```

6. Connect to localhost:50 via VNC to get to the VM's console. Optionally add an /etc/hosts entry for it so you can get to it by name. Login for root is 'password'. You can also ssh to 192.168.100.10 or 172.17.10.10 as root.

Create the VM from scratch

1. Build the image manually by following the instructions here.

Building (CentOS devcloud-kvm)

> For this tutorial we'll assume that you start in the /root directory and that you'll clone the code there

> Both the CentOS and Ubuntu versions of devcloud-kvm use the same networking/ip setup. If you intend to use both at the same time, you'll need to make your own adjustments

1. install your IDE of choice, git checkout cloudstack.

```
git clone https://git-wip-us.apache.org/repos/asf/cloudstack.git
```

2. CD into the locally cloned repo, compile, deploy database, and start cloudstack:
   The first time, it's a good idea to build the RPMs and install them. This puts all of the dependencies in the right places. In subsequent development, you can build just the portion you're working on and copy it to the right place (or just build RPMs again and install those):

```
# make sure you have rpm-build package installed
sudo yum install rpm-build
# build
cd cloudstack
git checkout <insert branch you want to work on here>
cd packaging
./package.sh -d centos63
#install
cd ../../dist/rpmbuild/RPMS/x86_64
rm -f cloudstack-baremetal-agent*
rpm -Uvh cloudstack*
# sed debug mode in cloudstack agent
sed -i 's/INFO/DEBUG/g' /etc/cloudstack/agent/log4j-cloud.xml
#deploy database
cloudstack-setup-databases cloud:password@localhost --deploy-as root
#deploy management server
cloudstack-setup-management
#wait 30 seconds so any db upgrades can complete
mysql -e "update cloud.configuration set value=8096 where name='integration.api.port'"
mysql -e "update cloud.configuration set value='true' where name='system.vm.use.local.storage'"
mysql -e "update cloud.configuration set value='false' where name='consoleproxy.restart'"
mysql < /root/cloudstack/tools/devcloud-kvm/devcloud-kvm.sql
service cloudstack-management restart
```

   Management server should be starting, may take 30 seconds. You can try going to http://172.17.10.10:8080/client to check.
3. Deploy fresh advanced networking zone via marvin autoconfig (requires integration port to be open on 8096, either manually via global config, or via the mysql statement above):

```
cd /root/cloudstack
mvn -P developer,systemvm clean install
cd tools
easy_install marvin/dist/Marvin-0.1.0.tar.gz
#for KVM-based hypervisor
python2.7 marvin/marvin/deployDataCenter.py -i devcloud-kvm/devcloud-kvm-advanced.cfg
#slightly different network config for VMware Fusion based hypervisor
python2.7 marvin/marvin/deployDataCenter.py -i devcloud-kvm/devcloud-kvm-advanced-fusion.cfg
```

   ⚠ Deploying the marvin configuration will start the cloud-agent on the devcloud-kvm host when it adds the host to the zone. Subsequent code changes, recompiles, when the zone is already configured will require a manual restart via 'service cloudstack-agent restart'.

4. Stop/restart management server to pick up global config changes.
5. Connect to http://172.17.10.10:8080/client for the web UI, or do whatever you set up devcloud-kvm to do!

Building (Ubuntu devcloud-kvm)

⚠ For this tutorial we'll assume that you start in the /root directory and that you'll clone the code there

⚠ Both the CentOS and Ubuntu versions of devcloud-kvm use the same networking/ip setup. If you intend to use both at the same time, you'll need to make your own adjustments

1. install your IDE of choice, git checkout cloudstack.

```
git clone https://git-wip-us.apache.org/repos/asf/cloudstack.git
```

2. CD into the locally cloned repo, compile, deploy database, and start cloudstack:
   The first time, it's a good idea to build the DEBs and install them. This puts all of the dependencies in the right places. In subsequent development, you can build just the portion you're working on and copy it to the right place (or just build DEBs again and install those):

```
# build
cd cloudstack
git checkout <insert branch you want to work on here>
mvn clean install -P developer,systemvm
dpkg-buildpackage -uc -us
#install
dpkg -i ../*.deb
# sed debug mode in cloudstack agent
sed -i 's/INFO/DEBUG/g' /etc/cloudstack/agent/log4j-cloud.xml
#deploy database
cloudstack-setup-databases cloud:password@localhost --deploy-as root
#deploy management server
cloudstack-setup-management
#wait 30 seconds so any db upgrades can complete
mysql -e "update cloud.configuration set value=8096 where name='integration.api.port'"
mysql -e "update cloud.configuration set value='true' where name='system.vm.use.local.storage'"
service cloudstack-management restart
```

Management server should be starting, may take 30 seconds. You can try going to http://172.17.10.10:8080/client to check.

3. Deploy fresh advanced networking zone via marvin autoconfig (requires integration port to be open on 8096, either manually via global config, or via the mysql statement above):

```
mvn -P developer,systemvm clean installmvn -P developer,systemvm clean install

cd tools
mvnpython marvin/marvin/deployDataCenter.py -i devcloud-kvm/devcloud-kvm-advanced.cfg
```


Deploying the marvin configuration will start the cloud-agent on the devcloud-kvm host when it adds the host to the zone. Subsequent code changes, recompiles, when the zone is already configured will require a manual restart via 'service cloudstack-agent restart'.

4. Stop/restart management server to pick up global config changes.
5. Connect to http://172.17.10.10:8080/client for the web UI, or do whatever you set up devcloud-kvm to do!

Working with an existing devcloud-kvm

this is out of date. Running directly from mvn seems to be broken at the moment, or I don't know how to do it. For now probably best to generate RPMs and reinstall them

If you're developing, pulled a new branch, etc on a devcloud-kvm that you've already set up, follow these steps to redeploy new code.

1. Stop your management server by killing your jetty process (CTRL+C)
2. Stop cloudstack-agent:

```
service cloudstack-agent stop
```

3. Compile your new code:

```
mvn -P developer,systemvm clean install
```

- Optional, redeploy database. If you do this all of your existing VMs will be orphaned, need to be manually shut down and deleted, unless you cleaned them up before you stopped the management server. You generally only need to do this if you want to start from scratch or the database schema has changed.

```
cloudstack-setup-databases cloud:password@localhost --deploy-as root
cd /root/cloudstack
mysql < tools/devcloud-kvm/devcloud-kvm.sql
```

4. Start your management server:

```
mvn -pl :cloud-client-ui jetty:run
```

- If you didn't wipe out your database, start the cloudstack-agent:

  ```
  service cloudstack-agent start
  ```

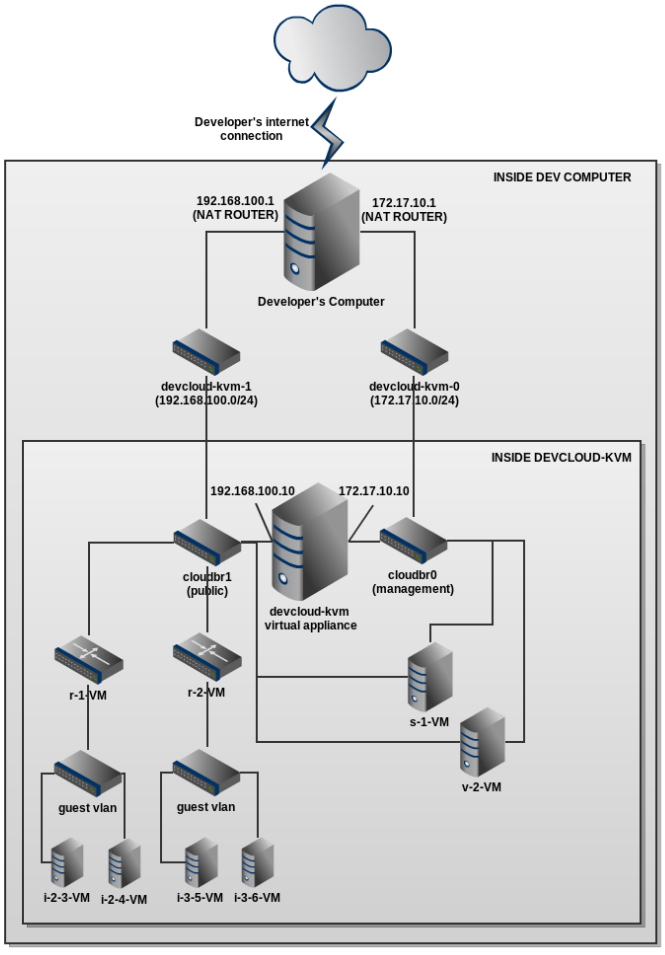
- If you DID wipe out your database, redeploy your config:

  ```
  python tools/marvin/marvin/deployDataCenter.py -i tools/devcloud-kvm/devcloud-kvm-advanced.cfg
  ```

  Then restart your management server as above.

> The above assumes you want to run the management server out of the checked-out repo code. It compiles the code in place and runs it from the tree. Changes to the agent, changes to the systemvm.iso, or any other installed code will need to be copied into place on the system. If you're not strictly working on management server code it's probably best to just rebuild the RPMs and install those, then restart cloudstack-management and cloudstack-agent.

Network Diagram



Using devcloud-kvm to test

Testing with marvin is conducted using Python, and requires Python 2.7. The devcloud-kvm image has Python 2.7 as an altinstall, therefore any testing should be done by executing "python2.7", and any easy-installs should be done via 'easy_install-2.7'. Example:

```
easy_install-2.7 tools/marvin/dist/Marvin-0.1.0.tar.gz
```

Other than that, you should largely be able to follow the instructions at https://cwiki.apache.org/confluence/display/CLOUDSTACK/Testing+with+Python

Keep in mind that some of the tests will require a specific zone configuration.