oVirt

Creating Oracle Databases on KVM using the deploycluster tool with oVirt API

Mitch DSouza
PreSales Consultant

August/2021

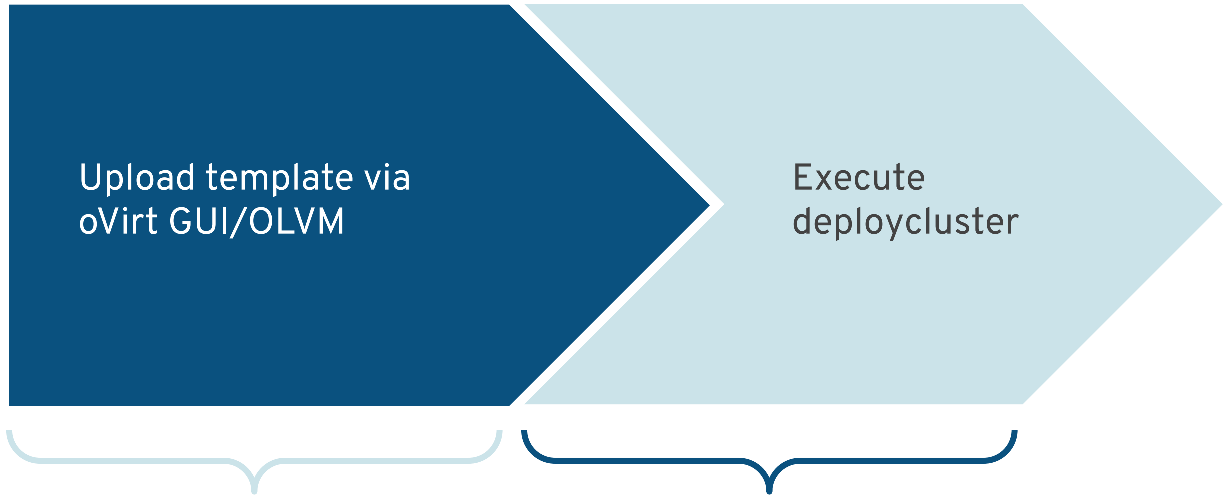
What is deploycluster ?

- A tool to create Oracle database clusters from a VM template
- Able to create n-node RAC, single instance, and single instance HA database VM’s
- End deployment fully certified to work on customer on-prem environments
- Rapid, repeatable and well defined methodology
- Tool created over 10 years ago to work on Oracle’s OVM (Xen) hypervisor
- Original tool written in python
- Continuous enhancements and fixes with customer feedback
- Version 3 is the terminal release for OVM

What is new with deploycluster4 ?

- Oracle’s virtualization strategy to the KVM hypervisor required a tool rewrite
- Xen and oVirt API is incompatible (as expected) so no easy transition
- oVirt rich API set much more powerful leading to an enhanced tool
- Google “go” was chosen as the language for deploycluster4
- A more modern compiled language with concurrency, threading, type checking, etc
- Single binary, with no dependencies
- Database templates had to be tweaked to support cloud-init
- Tool leverages cloud-init for provisioning networking, ssh keys, and database configuration
- Tool can be used in one-shot or modern IaC deployment pipelines
- Synchronous or asynchronous operations
- Non-database use cases with JSON output

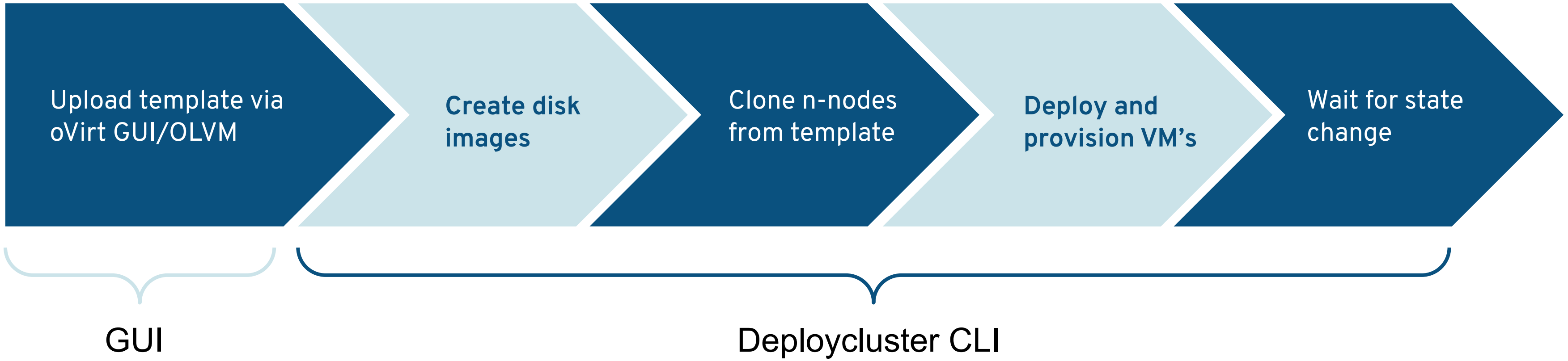
One-shot deployment



GUI

Deploycluster CLI

Pipeline deployment flow


Upload template via oVirt GUI/OLVM
Create disk images
Clone n-nodes from template
Deploy and provision VM’s
Wait for state change
GUI
Deploycluster CLI

A quick note about the first step

Upload template via oVirt GUI/OLVM

- The only step currently requiring the oVirt GUI
- There is no API to upload OVA templates
- We can achieve this step however by decomposing the OVA tar file, interpreting the XML client side, uploading the disk images, and recreating the OVA from a template with the oVirt API
- Experimental code (that is functional) is in our internal repo for deploycluster already to do this step but need more testing before release

Exercising oVirt API

The tool heavily exercises the API.

All sorting, parsing and decisions are made client side, so we typically gather all information early on with the tool.

What API calls are used:

- Listing of domains, cluster, disks, networks (gathering stage only)
- Creation of disks in multiple storage domains
- Cloning VM’s
- Adjusting networks, cpu, memory, disk attachments on clones
- Starting VM’s
- Removing VM’s and disk attachments
- Provisioning YAML for cloud-init

API Bugs ?

In quite extensive testing, we have not discovered many/any.

On heavily loaded oVirt manager or busy KVM nodes we will get spurious HTML returned from wildfly which will confuse the JSON parser and return unrepeatable errors. Retrying the same API call will succeed. This has proven hard to get a repeatable test case (seems this was fixed Aug 9th in commit 29b63f3)

The return from API calls do not necessarily populate all expected fields in the related structure (VM, Disk, Network, etc). Several “follow” API calls are required to get information we need.

All things considered, we were pleased with the stability of the oVirt API.

Sample one-shot 2-node RAC

```
% deploycluster -C OLVM* --createdisk asm0,asm1,asm2 --clonemem 4G --clonenet nic0,nic1=private -M vm1,vm2 -N netconfig.ini -P params.ini --clonedeploy
```

This is a one shot command where we do the following in the background

- Create 3 named virtual disks that will be marked shared of size 5G each (default size, and storagedomain of the template can be changed if required)
- Clone 2 nodes (specified in the netconfig.ini) named vm1,vm2
- Modify the clones to specify them with 4GB and having 2 nics, one on the default ovirtmgmt and the other on the private network profile called “private”
- Attach the 3 created disks to each VM
- Generate the cloud-init YAML file that provisions ssh-keys for the cloud-user and the inter-server communication keys
- Start up the VMs
- Note, the execution is asynchronous, so the above command will exit in a few seconds having fired off the command to the oVirt API

Pipeline version of the same deployment

Create the ASM disks for the RAC nodes (note the disks can be on different storage domains)

- % deploycluster --createdisk asm0,asm1,asm2 --storagedomain SD

Create the 2 clones named vm1, vm2 (VM's will remain in the DOWN state after cloning). Note we use the --cloneattach to attach the previously created disks, but you can attach iSCSI/FC luns if required.

- % deploycluster -C OLVM* -clonenet nic0,nic1=private --clonemen 4G -M vm1,vm2 \
  --cloneattach asm0,asm1,asm2

Now deploy the cloned VM's, passing the --wait 10m to make the command synchronous and wait for the VM's to be up. This let's the pipeline catch errors or timeouts

- % deploycluster -M vm1,vm2 -N netconfig.ini -P params.ini --wait 10m

Command usage and checks

Since most of the heavy lifting for cluster creation is done on the VM template itself, we do some basic sanity checks client side before the VM's are kicked off

- Checks on netconfig.ini (bad ip, ip duplicates, netmasks, node name format, dns, etc)
- Checks on params.ini (basic checks only on database configuration)
- Check disks are shared (for RAC, not SI or SIHA)
- Checks for interfaces on same network
- Check for ssh keys

Notice that all the basic connection info for the deploycluster command is read from a config file so we don't expose passwords, and we are less prone to typo mistakes. The deploycluster.ini is very important as it has user, host, password, cluster, datacenter, and other options and should have appropriate permissions.

Important to note, with great power comes great responsibility. You should never use the admin@internal account for deploycluster. A suitable RBAC user should be used.

Other use cases

Non Oracle Database RAC use cases are possible

1. Use --json to dump and machine parse vm, templates, disks, storage domains, clusters, networks, etc. configuration

```
$ deploycluster -L --json
```

2. Create multiple disk images for attachment to VM's

```
$ deploycluster --createdisk \
      namefmt=disk-%d,count=5,size=10g,storagedomain=nfs
```

3. Start an existing VM (named vm1) and wait for an IP

```
$ deploycluster -M vm1 --noexec --waitforip
```

Other use cases (contd…)

Non Oracle Database RAC use cases are possible

4. Create multiple VM’s from a template and adjust vm specs

```
$ deploycluster -M namefmt=vm%d,count=5 -C templatename --clonemem  4G \
            --clonecpu 4,2,1 --clonenet nic0,nic1=mynet
```

5. Remove VM’s

```
$ deploycluster -M vm1,vm2 --remove
```

Demo

oVirt

Thank you!

https://ovirt.org/

users@ovirt.org

@ovirt