

AZURE, CLOUD, DEVOPS, KUBERNETES

Test OpenShift 4 at low cost in Azure

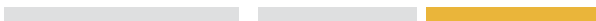
28.11.2023 by DevOps


If you want to learn and test OpenShift 4 (the last release is 4.14 at this time of writing), you may not want to go straight with the installation of a full cluster. Using the OpenShift Service of a cloud provider is costly. Even if you want to install it all by yourself in the cloud, you'll still need several virtual machines with enough CPU and memory that will also not be cheap in the end.

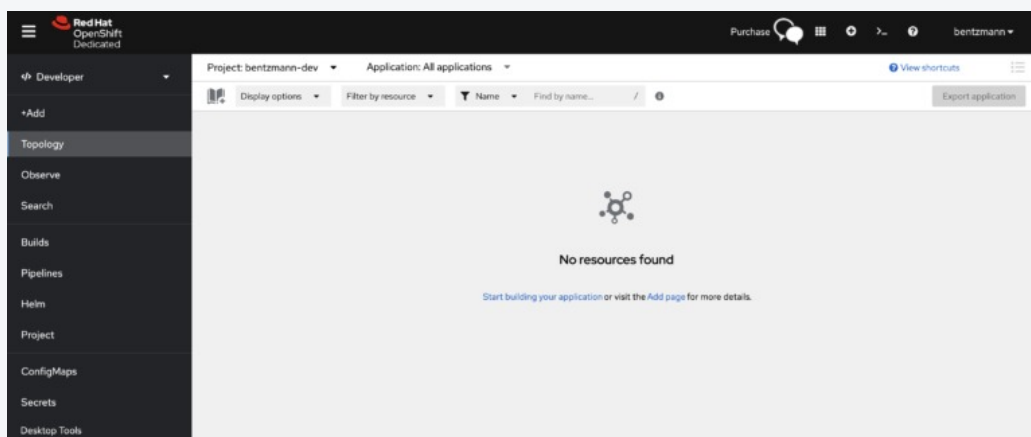
If like me, you want to quickly set it up and play with it at low or even no cost then read on, I've cleared the path for you.

For those of you who are not sure what OpenShift is, it is the Kubernetes solution provided by Red Hat. It is very much used because Red Hat is a popular and stable operating system used in production by many companies. It is then a natural path to use OpenShift for deploying microservice applications in


We use cookies on our website to provide you with the most relevant experience by remembering your preferences. No personal data is stored. By clicking on "Accept All", you consent to the use of ALL cookies. However, you can visit "Cookie Settings" to provide controlled consent. [Read More](#)



If you are a developer and just want to do some tests to deploy your workload in an OpenShift cluster, then there is an easy solution provided by Red Hat at no cost. Just register for free and get access to a Developer Sandbox. You can find all the information [here](#) . In a few minutes you have a cluster ready for you to play with. Your data are kept for 30 days, after that time you can create another Sandbox to continue your exploration.



Above is the OpenShift Web console you can access from your browser. On the top left you can have the Developer or Administrator view and on the top right you can also get a prompt where you can use the oc tool to interact with the cluster. So there are already a lot of things you can do, but you are not kubeadmin of this cluster.

If you need to completely administer an OpenShift cluster, you can then move to the next step: Install a mini cluster that will use just one node. This solution is now called OpenShift Local and you'll find all the information [here](#) . It was formerly called CodeReady Containers. In the version of OpenShift 3, you could use minishift that provided the same kind of experience. This solution is at no cost if you install it on your laptop/workstation but you need to have enough CPU and memory. In my case I didn't want to clutter my laptop so I've

decided to use a virtual machine in the cloud instead.

Moderately difficult solution at low cost

This solution is not as easy as the previous one because there are a few flaming hoops to go through. However as you've found this blog, congratulations! You can just sit back, relax and follow all the instructions along!

The first flaming hoop is that the virtual machine from the cloud provider needs to support nested virtualization. OpenShift Local itself creates a virtual machine which will be the node used to deploy our cluster. Hence the nested virtualization. AWS EC2 doesn't support it. At least not at a good price as you need to choose an EC2 type that is an OnPrem machine. Azure and Google Cloud Platform have some virtual machine types that supports nested virtualization at a decent price. I've chosen Azure with which I'm familiar with.

After a few attempts, I've found the right configuration required to make it all work and I'll share it with you so you will be successful on your first try!

I've selected the operating system Red Hat 8.7 with the size Standard D4s_v3 that supports nested virtualization with 4xCPU, 16GiB memory (and 64GiB Disk by default). This VM will cost US\$175 per month if it stays up 24/7. This is not cheap but still low cost in comparison of setting up a real OpenShift cluster in the cloud that can cost several thousands a month. Now here comes the second flaming hoop! In the way OpenShift Local run its installation, a disk of 64GiB is not enough, you need to choose 128GiB for the OS disk size (and standard SSD is fine). For the other parameters you can keep

the default values or adjust according to your preferences. At this point you can create and launch your virtual machine.

Before connecting to it through ssh, you can already plan to open some ports to be able to later connect to the OpenShift Web Console from your laptop. In the NSG that is bound to your public IP Address, modify the Inbound Security Rule with the following destination ports 80,6443,443 in addition to 22. I also like to put my IP Address as source as a good security practice. You can now ssh to the public IP Address of your virtual machine:

```
1 | $ ssh -i ".ssh/mykey.pem" enb@<public_ip_of_your_vm>
```

The third flaming hoop

The first thing to notice is that even if I've chosen 128GiB of disk, only the default 64GiB are setup in my virtual machine. Second, OpenShift Local will download and install its packages in my home folder. It needs a bit more than 32G of free space to do so. Here comes the flaming:

```
1 | [enb@enb-OpenShift ~]$ df -Th
2 | Filesystem                                Type      Size  Used Avail
3 | devtmpfs                                  devtmpfs  7.8G    0  7.8G
4 | tmpfs                                     tmpfs     7.8G    0  7.8G
5 | tmpfs                                     tmpfs     7.8G  8.6M  7.8G
6 | tmpfs                                     tmpfs     7.8G    0  7.8G
7 | /dev/mapper/rootvg-rootlv                 xfs       2.0G   74M  2.0G
8 | /dev/mapper/rootvg-usrlv                  xfs       10G   1.8G  8.3G
9 | /dev/sda1                                 xfs       496M  107M  390M
10 | /dev/mapper/rootvg-tmplv                  xfs       2.0G   47M  2.0G
11 | /dev/mapper/rootvg-homelv                 xfs       1014M   40M  975M
12 | /dev/mapper/rootvg-varlv                  xfs       8.0G  325M  7.7G
13 | /dev/sda15                                vfat      495M   5.8M  489M
14 | tmpfs                                     tmpfs     1.6G    0  1.6G
```

My /home mount is using a filesystem that has a size of 1014M

with 975M available! We then need to do a bit of system administration and use our superpower as root. If you are not comfortable with this part, I promise it will not be too long, just a few commands and you will be done with it:

```
1 [enb@enb-OpenShift ~]$ sudo -i
2 [root@enb-OpenShift ~]#
3
4 [root@enb-OpenShift ~]# lsblk -f
5 NAME                                FSTYPE          LABEL  UUID
6 sda
7 |-sda1                             xfs              3be24f71-c584-49
8 |-sda2                             LVM2_member      R2fg21-J3fd-S6As
9 ||-rootvg-tmplv                    xfs              a6a26328-7157-41
10 ||-rootvg-usrlv                    xfs              d0a5ec53-dea1-4b
11 ||-rootvg-homelv                   xfs              2658b270-62cc-45
12 ||-rootvg-varlv                    xfs              3aba65b1-d5fb-49
13 ||`-rootvg-rootlv                 xfs              3549dfce-5fcb-4d
14 |-sda14
15 `--sda15                          vfat             3328-E458
16 sdb
17 `--sdb1                            ext4             5ddb46ed-6a87-49
18 sr0
```

Our /home mount is on the logical volume rootvg-homelv and use the disk sda2. So let's check that physical volume as well as the sda2 disk:

```
1 [root@enb-OpenShift ~]# pvscan
2 PV /dev/sda2   VG rootvg   lvm2 [<63.02 GiB
3   Total: 1 [<63.02 GiB] / in use: 1 [<63.02 GiB] / i
4
5 root@enb-OpenShift ~]# lsblk /dev/sda2
6 NAME                                MAJ:MIN  RM  SIZE  RO  TYPE  MOUNTPOINT
7 sda2                                8:2      0   63G   0  part
8 |-rootvg-tmplv                    253:0    0    2G   0  lvm   /tmp
9 |-rootvg-usrlv                    253:1    0   10G   0  lvm   /usr
10 |-rootvg-homelv                   253:2    0    1G   0  lvm   /home
11 |-rootvg-varlv                    253:3    0    8G   0  lvm   /var
12 `--rootvg-rootlv                 253:4    0    2G   0  lvm   /
```

There is 40 GiB free in LVM2 and sda2 uses only 63G. We will then install the growpart package to extend its storage capacity:

```

1 [root@enb-OpenShift ~]# yum install cloud-utils-grow
2
3 [root@enb-OpenShift ~]# growpart /dev/sda 2
4 CHANGED: partition=2 start=2050048 old: size=1321656
5
6 [root@enb-OpenShift ~]# lsblk /dev/sda2
7 NAME                                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
8 sda2                                8:2      0  127G  0 part
9 |-rootvg-tmplv                      253:0    0    2G  0 lvm   /tmp
10 |-rootvg-usrlv                      253:1    0   10G  0 lvm   /usr
11 |-rootvg-homelv                     253:2    0    1G  0 lvm   /home
12 |-rootvg-varlv                     253:3    0    8G  0 lvm   /var
13 `--rootvg-rootlv                   253:4    0    2G  0 lvm   /

```

Now sda2 has a size of 127G and we can finally resize our physical and logical volume:

```

1 [root@enb-OpenShift ~]# pvresize /dev/sda2
2 Physical volume "/dev/sda2" changed
3 1 physical volume(s) resized or updated / 0 physic
4
5 [root@enb-OpenShift ~]# pvscan
6 PV /dev/sda2   VG rootvg                lvm2 [<127.02 Gi
7 Total: 1 [<127.02 GiB] / in use: 1 [<127.02 GiB] /
8
9 [root@enb-OpenShift ~]# lvresize -r -L +64G /dev/map
10 Size of logical volume rootvg/homelv changed from
11 Logical volume rootvg/homelv successfully resized.
12 meta-data=/dev/mapper/rootvg-homelv isize=512    agc
13          =                               sectsz=4096 attr=2
14          =                               crc=1      finobt
15          =                               reflink=1   bigtim
16 data      =                               bsize=4096 blocks
17          =                               sunit=0     swidth
18 naming    =version 2                      bsize=4096 ascii-
19 log       =internal log                    bsize=4096 blocks
20          =                               sectsz=4096 sunit=
21 realtime =none                            extsz=4096 blocks
22 data blocks changed from 262144 to 17039360

```

And that's it! Let's check out the result and get out of here!

```

1 [root@enb-OpenShift ~]# df -Th
2 Filesystem                Type      Size  Used Avail
3 devtmpfs                  devtmpfs  7.8G   0    7.8G
4 tmpfs                     tmpfs     7.8G   0    7.8G
5 tmpfs                     tmpfs     7.8G  8.6M   7.8G
6 tmpfs                     tmpfs     7.8G   0    7.8G
7 /dev/mapper/rootvg-rootlv xfs       2.0G   74M   2.0G
8 /dev/mapper/rootvg-usrlv xfs       10G   1.8G   8.3G
9 /dev/sda1                 xfs       496M  107M  390M
10 /dev/mapper/rootvg-tmplv xfs       2.0G   47M   2.0G
11 /dev/mapper/rootvg-homelv xfs       65G  516M   65G
12 /dev/mapper/rootvg-varlv xfs       8.0G  574M   7.5G
13 /dev/sda15               vfat      495M   5.8M  489M
14 tmpfs                    tmpfs     1.6G   0    1.6G
15
16 [root@enb-OpenShift ~]# exit
17 logout
18 [enb@enb-OpenShift ~]$

```

We can see now that our /home partition has 65G available which is enough for the installation of OpenShift Local. Let's do it!

OpenShift local installation

With our virtual machine now correctly configured, this is a walk in the park:

```

1 $ mkdir crc
2 $ wget https://mirror.openshift.com/pub/openshift-v4/
3 $ tar -xvf crc-linux-amd64.tar.xz
4 $ mv crc-linux-2.29.0-amd64/* ./crc
5 $ rm crc-linux-amd64.tar.xz && sudo rm -r crc-linux-2
6 $ cd crc && sudo chmod +x crc
7 $ export PATH=$PATH:/home/enb/crc
8 $ crc setup
9 $ crc start

```

After the command “crc setup” you can choose to contribute or not to statistics and after the “crc start” you will have to enter a pull secret. You are given the link to connect to and you need to

connect to your free Red Hat Developer account to get it.

It takes several minutes to start and when finished, the administrator and developer account information are given to connect to the cluster. You can for example connect to it as a developer by using the oc tool to interact with the cluster:

```
1 | $ eval $(crc oc-env)
2 | $ oc login -u developer -p developer https://api.crc.
```

Last flaming hoop

At this stage there are a few more tips I'd like to share to keep your VM persistent. At some point you are going to shut your VM down and start it again later. The first thing to do is to check and keep note of the DNS configuration of your VM:

```
1 | $ cat /etc/resolv.conf
```

This configuration will change at some point (more on that in my next blog) when you restart the NetworkManager service and so also your VM. So copy the content of this file somewhere safe as you may need to restore it later before starting up your cluster again.

Now, before shutting your VM down, it is important to exit and stop your cluster properly with the following command:


```
1 | $ oc logout
2 | $ crc stop
```

This will keep your cluster stable and consistent then you can shut down the VM. When you will start it again, check the content of /etc/resolv.conf is identical to its original

configuration (at this stage it may be) otherwise modify it. Then only start your cluster:

```
1 | $ crc start
```

Wrap up

Congratulations! You have successfully installed an OpenShift cluster and have access to it through the command line interface. However at this stage you are still connected to the Azure VM through ssh. Wouldn't it be great to interact with the cluster directly from your laptop? Interact in command line but also by using the OpenShift Web console. That will be the topic of my next [blog](#) . Stay tuned!

 Post Views: 2,097



Thumbnail:
[60x60]

by

DevOps

No comments yet.

Leave a Reply:

Email

Username

Comment



Submit

Cancel

BE SHARING

Related blog articles

ANSIBLE, DEVOPS

Parallel execution of Ansible roles

10.06.2025 by **Martin Bracher**



AZURE, CLOUD

Azure Bootcamp Switzerland 2025

10.06.2025 by **Nicolas Jardot**



AZURE

Azure Bootcamp Switzerland 2025, an Azure day in Bern

06.06.2025 by **Adrien Devaux**



CLOUD, ORACLE



ExaCC/CS: modify database parameters in batch mode using dbaascli

03.06.2025 by **Jérôme Witt**

ABOUT

dbi services is a company specialized in IT consulting and services. We are experts in innovative and efficient data infrastructures and platforms. Tailor-made solutions is what we offer to our customers thanks to our consultants, whose skills and knowledge are constantly evolving thanks to continuous training.

Copyright © 2010-2025 dbi services SA.

All rights reserved.

WE ARE CLOSE TO YOU

Delémont >

Nyon >

Bâle >

Berne >

Zurich >

