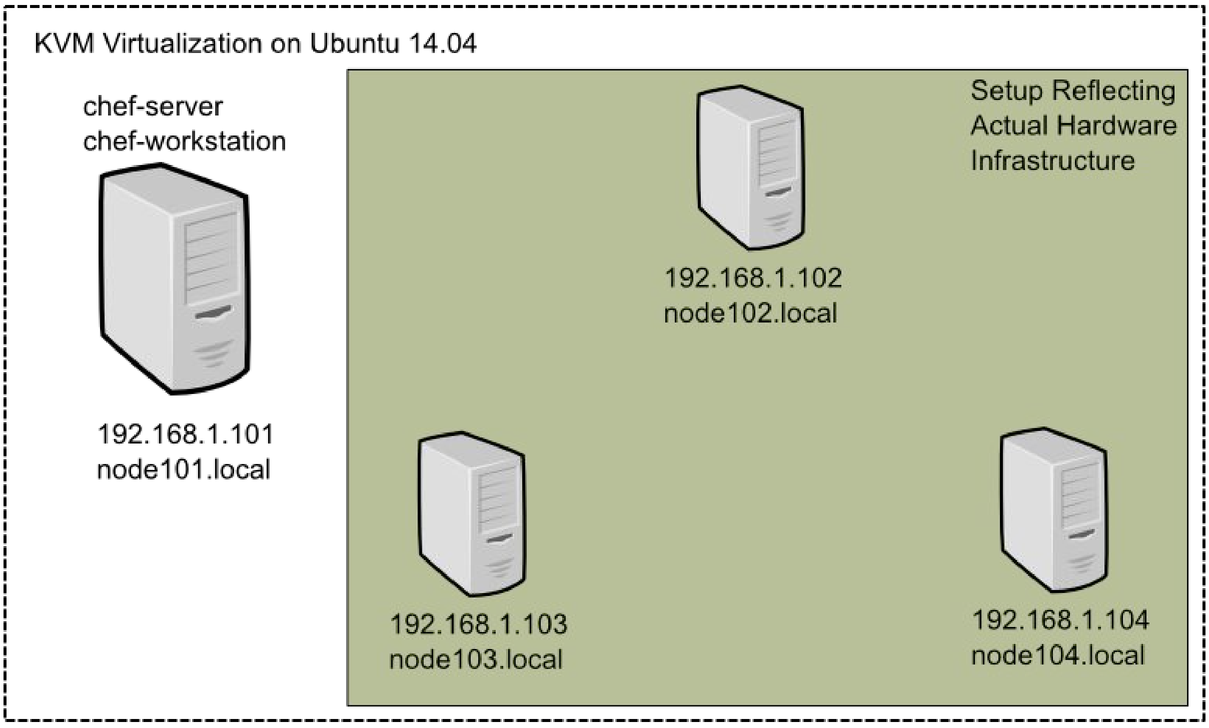
Written by: Ivan Ermilov, Hajira Jabeen
Editors: Ricardo Usbeck

Abstract

In this whitepaper, we show how to deploy the basic BDE platform using CHEF on local infrastructure. Base BDE platform consists of HDFS, Zookeeper, Mesos and several frameworks installed on top of Mesos.
The whitepaper is structured as follows: in “Infrastructure Description” we provide an overview over how to replicate our configuration using Ubuntu with KVM virtualization, in “Chef Initial Setup” we describe how to install and use chef-server in a local infrastructure, in “Creating Cookbook for Installation of BDE Platform” we guide through the BDE Platform cookbook creation process (HDFS, Zookeeper, Mesos, Marathon, Chronos).
If you just want to use the BDE Platform cookbook and see how to deploy it on your own infrastructure, skip to section “Using the BDE Platform Cookbooks”.

Infrastructure Description

We will deploy HDFS, Zookeeper and Mesos on 3 nodes. For the setup we use Ubuntu 14.04 machine with 16 GB RAM and 250 GB SSD drive with kvm and bridged networking configuration. KVM installation instructions can be found here: ttps://help.ubuntu.com/community/KVM/Installation Bridged networking configuration can be found here: https://help.ubuntu.com/community/KVM/Networking Essentially, we will run 4 VMs with 2 GB RAM and 20 GB disk space per VM. One of the VMs will be used for chef-server and chefDK installation.



Host configuration

Bridge configuration for the VM host is done as follows:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.5
netmask 255.255.255.0
network 192.168.1.0
gateway 192.168.1.1
broadcast 192.168.1.255
dns-nameservers 8.8.8.8

auto br0
iface br0 inet static
    address 192.168.1.5
    netmask 255.255.255.0
    network 192.168.1.0
    gateway 192.168.1.1
    broadcast 192.168.1.255
    dns-nameservers 8.8.8.8
    bridge_ports eth0
    bridge_stp off
    bridge_fd 9
    bridge_hello 2
    bridge_maxage 12
```

Node Configuration

On example of node104.local
/etc/hostname

```
node104.local
```

/etc/hosts

```
127.0.0.1  localhost
192.168.1.101   node101.local
192.168.1.102   node102.local
192.168.1.103   node103.local
192.168.1.104   node104.local

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

/etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.1.104
    netmask 255.255.255.0
    network 192.168.1.0
    gateway 192.168.1.1
    broadcast 192.168.1.255
    dns-nameservers 8.8.8.8
```

Chef Initial Setup

Chef-server can be installed from officially provided debian package. For chef installation refer to the official documentation:
https://docs.chef.io/release/server_12-2/install_server.html#standalone

On the same machine install ChefDK:
https://downloads.chef.io/chef-dk/ubuntu/

After installation of chef-server you should have a user and organization on the server as well as generated keys for them. The user and organization keys are generated in the folders you specified in *chef-server-ctl* command. For instance, you may have bde-user.pem key for user and aksw-validator.pem key for organization. They are required to configure access to Chef server.

To configure ChefDK to work with your server run:

```
knife configure
```

After following instructions you will have a configuration generated in /home/bde-user/.chef folder (where bde-user will be your username). We recommend to store your keys directly in .chef folder and to use relative paths in your configuration as follows:

```
bde-user@node101:~/.chef$ cat knife.rb
chef_dir = "/home/bde-user/.chef"
log_level                 :info
log_location              STDOUT
node_name                 'bde-user'
client_key                "#{chef_dir}/bde-user.pem"
validation_client_name    "aksw-validator"
validation_key            "#{chef_dir}/aksw-validator.pem"
chef_server_url
"https://node101.local:443/organizations/aksw"
syntax_check_cache_path  "#{chef_dir}/syntax_check_cache"
cookbook_path [ "#{chef_dir}/../chef-repo/cookbooks" ]
#ssl_verify_mode :verify_none
```

Inside ~/.chef folder, create trusted_certs and copy your server certificate there:

```
mkdir ~/.chef/trusted_certs
cp /var/opt/opscode/nginx/ca/node101.local.crt
~/.chef/trusted_certs
```

To check the configuration of the server run:

```
bde-user@node101:~/.chef$ knife ssl check
Connecting to host node101.local:443
Successfully verified certificates from `node101.local'
```

After successful configuration you should be able to check that your user was added on the server:

```
bde-user@node101:~/.chef$ knife user list
bde-user
bde-user@node101:~/.chef$ knife user show bde-user
display_name: Ivan Ermilov
email:        bde-user@informatik.uni-leipzig.de
first_name:   Ivan
last_name:    Ermilov
middle_name:
username:     bde-user
```

Creating a cookbook for installation of BDE platform

Cookbook environment setup

TODO: look for a non-official repo with last ruby version
Install ruby on the workstation (node101):

```
sudo apt-get install git-core curl zlib1g-dev build-essential
libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3
libxml2-dev libxslt1-dev libcurl4-openssl-dev
python-software-properties libffi-dev
cd
git clone git://github.com/sstephenson/rbenv.git .rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
exec $SHELL

git clone git://github.com/sstephenson/ruby-build.git
~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >>
~/.bashrc
exec $SHELL

rbenv install 2.2.3
rbenv global 2.2.3
ruby -v
```

First you have to initialize git repository in ~/chef-repo/cookbooks folder:

```
cd ~/chef-repo/cookbooks
git init
git add README.md
git commit -m “initial commit”
```

Remove the “example” cookbook as it will get in a way when uploading the cookbooks to the server:

```
rm -rf ~/chef-repo/cookbooks/example
```

Berkshelf installs cookbooks into ~/.berkshelf directory by default. Berkshelf can not automatically locate your knife.rb configuration file for chef connection parameters. To reuse knife.rb configuration in the berkshelf, you have to export *KNIFE_HOME* env variable:

```
export KNIFE_HOME=/home/bde-user/.chef/
```

If you have ~/.berkshelf/config.json file created, it will have precedence over *KNIFE_HOME* env variable. As we do not want duplicate configurations for knife and berkshelf, add config.json only if necessary (and only parameters which are necessary). After this you can upload cookbooks to the chef-server without ssl verification:

```
berks upload --no-ssl-verify
```

Berkshelf configuration can be used to specify --no-ssl-verify option for all berks commands. Edit ~/.berkshelf/config.json to look like:

```
{
  "ssl": {
    "verify": false
  }
}
```

For a complete overview on the options for config.json refer to the official documentation: http://berkshelf.com/

BDE-Cluster Cookbook

We will create all the recipes inside one cookbook, which will be then used to bootstrap the whole BDE base platform. To create an empty cookbook run the following commands:

```
cd ~/chef-repo/cookbooks
berks cookbook bde-cluster
```

The first component we will install is an HDFS.

HDFS

For HDFS installation, we need one namenode and three datanodes. The namenode will reside on node102. Datanodes will be deployed on node102, node103 and node104.

Namenode

First add the dependencies for *hadoop* and *java* cookbooks into Berksfile and metadata.rb.

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ cat
metadata.rb
name             'bde-cluster'
maintainer       'YOUR_NAME'
maintainer_email 'YOUR_EMAIL'
license          'All rights reserved'
description      'Installs/Configures bde-cluster'
long_description 'Installs/Configures bde-cluster'
version          '0.1.5'

depends 'java'
depends 'hadoop', '~> 2.0.9'
```

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ cat Berksfile
source "https://supermarket.chef.io"

metadata

cookbook 'java'
cookbook 'hadoop', '~> 2.0.9'
```

Hadoop requires Java 7 to function properly. Therefore, we specify the Java version to be installed in default.rb attributes file. Here, we specify dfs.name.dir parameter explicitly to be able to format HDFS conditionally (see hdfs-namenode.rb recipe):

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster/attributes$ cat
default.rb
default['java']['jdk_version'] = '7'
default['hadoop']['hdfs_site']['dfs.name.dir'] =
'file:///tmp/hadoop-hdfs/dfs/name'
```

Now, we can use *hadoop* cookbook as a library cookbook and make a wrapper for the installation of a HDFS namenode. First, put the Java installation to the default.rb recipe for the bde-cluster cookbook as we will use the same Java version on all the nodes:

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster/recipes$ cat
default.rb

include_recipe 'java::default'
```

Now we can create hdfs-namenode.rb recipe to deploy HDFS namenode. The following recipe will install namenode, format the HDFS, enable and start a service running the namenode.

```
include_recipe 'bde-cluster::default'
include_recipe 'hadoop::default'
include_recipe 'hadoop::hadoop_hdfs_namenode'

ruby_block 'initaction-hdfs-namenode-format' do
  block do
    resources('execute[hdfs-namenode-format]').run_action(:run)
  end
  not_if {
::File.directory?("#{node['hadoop']['hdfs_site']['dfs.name.dir'][7
..-1]}") }
end

ruby_block 'service-hadoop-hdfs-namenode-start-and-enable' do
  block do
    %w(enable start).each do |action|
resources('service[hadoop-hdfs-namenode]').run_action(action.to_sy
m)
    end
  end
end
```

Now to bootstrap the node102 to run hdfs-namenode.rb recipe, run the following commands:

```
berks install
berks upload
knife bootstrap 192.168.1.102 --ssh-user bde-user --ssh-password
'password' --sudo --use-sudo-password --node-name node102
--run-list 'recipe[bde-cluster::hdfs-namenode]'
```

Now you can see namenode running by logging into node102.local via ssh and using *htop* or *top* or by pointing your browser to http://node102.local:50070/ (host node102.local should be visible from your machine).

Troubleshooting

In case you have a problem that the node can not access the server to push the update at the end of the recipe run, run the following command:

```
gem install knife-acl
knife acl bulk add group clients nodes '.*' update,read
```

This situation may occur, if you have existing entry for the node in the chef-server but the node was rolled back to some initial state (cloned or restored via backup)

In case you have the following server response during node bootstrapping:

```
Since Server API v1, all keys must be updated via the keys
endpoint.
```

You have to remove client from chef workstation and run bootstrapping procedure again. To remove the client run (substitute node104 with you node):

```
knife client delete node104
```

Datanodes

We will create a recipe for datanode and deploy it on node102, node103 and node104.

First we have to add additional line into our default.rb attributes, that will point to HDFS namenode:

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster/attributes$ cat
default.rb
default['java']['jdk_version'] = '7'
default['hadoop']['hdfs_site']['dfs.name.dir'] =
'file:///tmp/hadoop-hdfs/dfs/name'
default['hadoop']['core_site']['fs.defaultFS'] =
'hdfs://node102.local'
```

The recipe for datanode is almost the same as for the namenode, except that we do not need to format it:

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster/recipes$ cat
hdfs-datanode.rb
include_recipe 'bde-cluster::default'
include_recipe 'hadoop::default'
include_recipe 'hadoop::hadoop_hdfs_datanode'

ruby_block 'service-hadoop-hdfs-datanode-start-and-enable' do
  block do
    %w(enable start).each do |action|

resources('service[hadoop-hdfs-datanode]').run_action(action.to_sy
m)
    end
  end
end
```

Increase the cookbook version to upload it to chef-server:

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ cat
metadata.rb
name             'bde-cluster'
maintainer       'YOUR_NAME'
maintainer_email 'YOUR_EMAIL'
license          'All rights reserved'
description      'Installs/Configures bde-cluster'
long_description 'Installs/Configures bde-cluster'
version          '0.1.7'

depends 'java'
depends 'hadoop', '~> 2.0.9'
```

Then we can run:

```
berks install
berks upload
knife bootstrap 192.168.1.103 --ssh-user bde-user --ssh-password
‘password' --sudo --use-sudo-password --node-name node103
--run-list 'recipe[bde-cluster::hdfs-datanode]'
knife bootstrap 192.168.1.104 --ssh-user bde-user --ssh-password
‘password' --sudo --use-sudo-password --node-name node104
--run-list 'recipe[bde-cluster::hdfs-datanode]'
```

As we already have hdfs-namenode.rb in run_list on node102, we can not just bootstrap the node with hdfs-datanode.rb recipe, but we have to add new recipe to the run_list:

```
knife node run_list add node102
"recipe[bde-cluster::hdfs-datanode]"
```

And check if it is added to the run_list:

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ knife node
show node102
Node Name:   node102
Environment: _default
FQDN:        node102.local
IP:          192.168.1.102
Run List:    recipe[bde-cluster::hdfs-namenode],
recipe[bde-cluster::hdfs-datanode]
Roles:
Recipes:     bde-cluster::hdfs-namenode, bde-cluster::default,
java::default, java::set_attributes_from_version, java::openjdk,
apt::default, java::default_java_symlink, java::set_java_home,
hadoop::default, hadoop::repo, hadoop::_hadoop_checkconfig,
hadoop::_compression_libs, hadoop::hadoop_hdfs_namenode,
hadoop::_hadoop_hdfs_checkconfig, hadoop::_system_tuning,
selinux::disabled, selinux::_common, sysctl::default,
sysctl::service
Platform:    ubuntu 14.04
Tags:
```

And update the node by running chef-client using ssh:

```
ssh -t bde-user@node102.local "sudo chef-client"
```

Now we have running setup with one namenode and three datanodes.

Zookeeper

For Zookeeper, we will use a cookbook provided by Bloomberg from github. It is not available on Chef Supermarket, thus the dependency list will include git repository directly:

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ cat Berksfile
source "https://supermarket.chef.io"

metadata

cookbook 'java'
cookbook 'hadoop', '~> 2.0.9'
cookbook 'zookeeper-cluster',
git:"https://github.com/bloomberg/zookeeper-cookbook.git"
```

Zookeeper ensemble is configured via data bag. To create one, we first export *EDITOR* env variable (we use vim, you can also export nano or vi), and then run corresponding knife command:

```
export EDITOR=”vim”

knife data bag create config zookeeper
```

Here we create item “*zookeeper*” for data bag “*config*”. You will be prompted to populate the data bag item (knife will open *$EDITOR* for you). We create “*production*” key and populate it with the FQDN (Fully Qualified Domain Name, i.e. node102.local) of our nodes.

```
{
  "id": "zookeeper",
  "production": [
    "node102.local",
    "node103.local",
    "node104.local"
  ]
}
```

The next step is to make a wrapper for *zookeeper-cluster::default* recipe. Essentially, we have to configure “*instance_name*” to resolve to node FQDN and point “ensemble” key to “*production*” key from “*zookeeper*” item of “*config*” data bag.

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ cat
recipes/zookeeper-node.rb
data_bag_zookeeper = data_bag_item('config', 'zookeeper')
node.default['zookeeper-cluster']['config']['instance_name'] =
node["fqdn"]
node.default['zookeeper-cluster']['config']['ensemble'] =
data_bag_zookeeper['production']
include_recipe 'zookeeper-cluster::default'
```

Do not forget to increase the version of the cookbook in metadata.rb and run berks commands to install new versions of packages and upload them to chef-server:

```
cd ~/chef-repo/cookbooks/bde-cluster
berks install
berks upload
```

Now we can add our recipe to run lists of our nodes (which are the same nodes specified in ensemble).

```
knife node run_list add node102 "bde-cluster::zookeeper-node"
knife node run_list add node103 "bde-cluster::zookeeper-node"
knife node run_list add node104 "bde-cluster::zookeeper-node"
```

And run chef-client remotely on all the nodes to apply the changes:

```
ssh -t bde-user@node102.local "sudo chef-client"
ssh -t bde-user@node103.local "sudo chef-client"
ssh -t bde-user@node104.local "sudo chef-client"
```

Now we can ssh into the nodes and check the status of the Zookeeper installation. As we can see in the following listings, node104 has zookeeper in mode "*leader*" while node103 is "*follower*".

```
bde-user@node104:/var/lib/zookeeper$ echo "stat" | nc localhost
2181
Zookeeper version: 3.5.0-alpha-1615249, built on 08/01/2014 22:13
GMT
Clients:
 /0:0:0:0:0:0:0:1:48714[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/0/0
Received: 1
Sent: 0
Connections: 1
Outstanding: 0
Zxid: 0x100000000
Mode: leader
Node count: 5

bde-user@node103:/var/log/zookeeper$ echo "stat" | nc localhost
2181
Zookeeper version: 3.5.0-alpha-1615249, built on 08/01/2014 22:13
GMT
Clients:
 /0:0:0:0:0:0:0:1:44625[0](queued=0,recved=1,sent=0)

Latency min/avg/max: 0/0/0
Received: 1
Sent: 1
Connections: 1
Outstanding: 0
Zxid: 0x0
Mode: follower
Node count: 5
```

For further reading on Zookeeper please refer to **ZooKeeper: Distributed Process Coordination by Flavio Junqueira, Benjamin Reed.**

Mesos

For Mesos installation, we will use cookbook provided by mdsol. Add the corresponding dependency in the Berksfile.

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster$ cat Berksfile
source "https://supermarket.chef.io"

metadata
```

```
cookbook 'java'
cookbook 'hadoop', '~> 2.0.9'
cookbook 'zookeeper-cluster',
git:"https://github.com/bloomberg/zookeeper-cookbook.git"
cookbook 'mesos',
git:"https://github.com/mdsol/mesos_cookbook.git"
```

Now we can wrap *mesos::master* recipe into our specific mesos-master.rb.

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster/recipes$ cat
mesos-master.rb
data_bag_zookeeper = data_bag_item('config', 'zookeeper')
zk_string =
"zk://#{data_bag_zookeeper['production'].join(':2181,')}:2181/meso
s"
node.default['mesos']['master']['flags']['cluster'] = "BDECluster"
node.default['mesos']['master']['flags']['zk'] = zk_string
node.default['mesos']['master']['flags']['quorum'] = "1"
node.default['mesos']['master']['flags']['hostname'] =
node['fqdn']
node.default['mesos']['master']['flags']['ip'] = node['ipaddress']

include_recipe 'mesos::master'
```

Increase the cookbook version in metadata.rb and update our cookbook on the chef-server.

```
berks install
berks upload
```

Then add mesos-master to corresponding node. In our setup, we run only one master with quorum equals to one (see the mesos-master.rb recipe above).

```
knife node run_list add node102 "bde-cluster::mesos-master"
ssh -t bde-user@node102.local "sudo chef-client"
```

To wrap mesos::slave recipe, we just have to specify zookeeper connection string.

```
bde-user@node101:~/chef-repo/cookbooks/bde-cluster/recipes$ cat
mesos-slave.rb
data_bag_zookeeper = data_bag_item('config', 'zookeeper')
zk_string =
"zk://#{data_bag_zookeeper['production'].join(':2181,')}:2181/meso
s"
node.default['mesos']['slave']['flags']['master'] = zk_string

include_recipe 'mesos::slave'
```

Again increase the cookbook version in metadata.rb and update our cookbook on the chef-server.

```
berks install
berks upload
```

Then we can add mesos-slave.rb recipe to all three nodes in our setup:

```
knife node run_list add node102 "bde-cluster::mesos-slave"
knife node run_list add node103 "bde-cluster::mesos-slave"
knife node run_list add node104 "bde-cluster::mesos-slave"
ssh -t bde-user@node102.local "sudo chef-client"
ssh -t bde-user@node103.local "sudo chef-client"
ssh -t bde-user@node104.local "sudo chef-client"
```

Now you have working setup of mesos with 1 master node and 3 slaves. You can navigate to http://node102.local:5050/ and confirm that everything is running correctly.

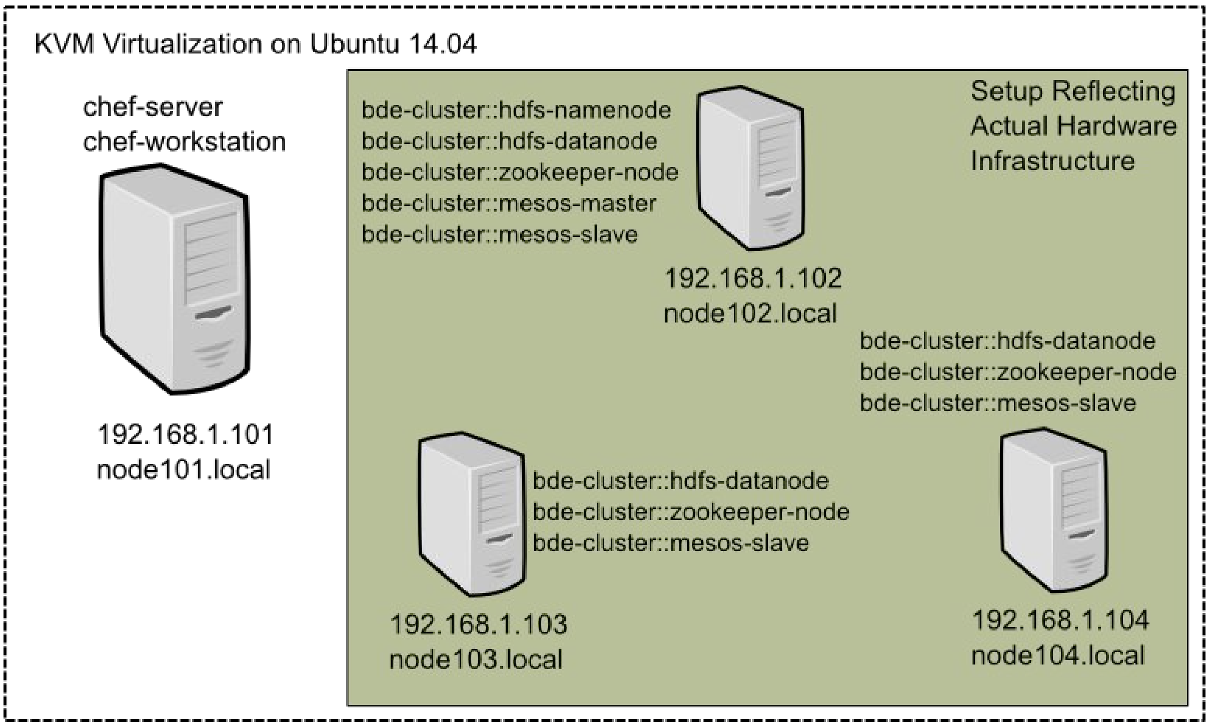
Marathon

The cookbook provided by community is still under development. Thus Marathon can be installed by hand after base platform deployment via chef.

Using the BDE Platform Cookbooks

To be able to use chef cookbooks in your local infrastructure, you have to setup chef-server with chef tools on your workstation. We will setup it on node101.local. Please, refer to instructions provided in section “Chef Initial Setup”.

We will deploy the architecture shown in the following figure:


KVM Virtualization on Ubuntu 14.04
chef-server
chef-workstation
192.168.1.101
node101.local
Setup Reflecting Actual Hardware Infrastructure
bde-cluster::hdfs-namenode
bde-cluster::hdfs-datanode
bde-cluster::zookeeper-node
bde-cluster::mesos-master
bde-cluster::mesos-slave
192.168.1.102
node102.local
bde-cluster::hdfs-datanode
bde-cluster::zookeeper-node
bde-cluster::mesos-slave
192.168.1.103
node103.local
bde-cluster::hdfs-datanode
bde-cluster::zookeeper-node
bde-cluster::mesos-slave
192.168.1.104
node104.local

Node102 will run hdfs-namenode and mesos-master. All nodes will run hdfs-datanode, zookeeper-node and mesos-slave.

Clone BDE base platform cookbook into ~/chef-repo/cookbooks folder.

```
bde-user@node101:~/chef-repo/cookbooks$ git clone
https://github.com/big-data-europe/chef-base-platform-cookbook
cd chef-base-platform-cookbook
```

Edit recipes/hdfs-datanode.rb to point to your namenode (node102.local):

```
node.default['hadoop']['core_site']['fs.defaultFS'] =
'hdfs://node102.local'
```

Create data bag for zookeeper and specify hostnames, where you will run zookeeper:

```
knife data bag create config zookeeper

{
  "id": "zookeeper",
  "production": [
    "node102.local",
    "node103.local",
    "node104.local"
  ]
}
```

Change the name of your mesos cluster in recipes/mesos-master.rb:

```
node.default['mesos']['master']['flags']['cluster'] = "BDECluster"
```

Install dependencies and upload the cookbook to chef-server.

```
berks install
berks upload
```

If you have problems uploading to chef-server using berks, edit ~/.berkshelf/config.json to disable ssl verification:

```
{
  "ssl": {
    "verify": false
  }
}
```

And bootstrap the nodes:

```
knife bootstrap 192.168.1.102 --ssh-user bde-user --ssh-password
'password' --sudo --use-sudo-password --node-name node102
--run-list
'recipe[bde-cluster::hdfs-namenode],recipe[bde-cluster::hdfs-datan
ode],recipe[bde-cluster::zookeeper-node],recipe[bde-cluster::mesos
-master],recipe[bde-cluster::mesos-slave]'
```

```
knife bootstrap 192.168.1.103 --ssh-user bde-user --ssh-password
'password' --sudo --use-sudo-password --node-name node103
--run-list
'recipe[bde-cluster::hdfs-datanode],recipe[bde-cluster::zookeeper-
node],recipe[bde-cluster::mesos-slave]'

knife bootstrap 192.168.1.104 --ssh-user bde-user --ssh-password
'password' --sudo --use-sudo-password --node-name node104
--run-list
'recipe[bde-cluster::hdfs-datanode],recipe[bde-cluster::zookeeper-
node],recipe[bde-cluster::mesos-slave]'
```