BIG DATA EUROPE

Coordination and Support Action

Big Data Europe – Empowering Communities with Data Technologies

Project Number: 644564 Start Date of Project: 01/01/2015 Duration: 36 months

Deliverable 5.1: Generic Big Data Integrator Instance I

Dissemination Level	Public
Due Date of Deliverable	Month 16, 01/05/2016
Actual Submission Date	Month 19, 15/07/2016
Work Package	WP5, Big Data Integrator Instances
Task	T5.1
Type	Other
Approval Status	Approved
Version	1.0
Number of Pages	33
Filename	D5.2_Generic_Big_Data_Integrator_Instance_I.pdf

Abstract: Documentation of the Generic Big Data Integrator Instance. This instance is used for generic functionality and usability tests, while it also targets the Big Data community in general.

Project funded by the European Union's Horizon 2020 Research and Innovation Programme (2014 – 2020)

History

Version	Date	Reason	Revised by
0.00	21/01/16	Document structure	S. Konstantopoulos
0.01	18/02/16	Description of the base system (Section 2)	I. Mouchakis, K. Mitrogeorgos, S. Konstantopoulos, and G. Stavrinos
0.02	26/04/16	Updated description of the base system, following decision (WP3) to switch to Docker Swarm instead of Mesos	I. Mouchakis, K. Mitrogeorgos, and G. Stavrinos
0.03	17/05/16	Usage guides for Spark, Virtuoso, Semagrow, 4store, GeoTriples, Sextant, Silk (Section 3)	E. Pauwels, A. Versteden, G. Papadakis, I. Mouchakis, A. Charalambidis
0.04	15/06/16	Usage guide for Hive (Sect 3). In Sect. 2, minor updates and corrections regarding accessing the cluster through the gateway machine. Synchronized the Wiki pages on Github with this document. Corrections on adding new node to the cluster.	I. Mouchakis
0.05	27/06/16	Peer review	A. Versteden, E. Pauwels
0.06	13/07/16	Address peer review comments	I. Mouchakis, G. Stavrinos, K. Mitrogeorgos, S. Konstantopoulos
1.0	17/07/16	Final version	S. Konstantopoulos

Author List

Organisation	Name	Contact Information
NCSR-D	Ioannis Mouchakis	gmouchakis@iit.demokritos.gr
NCSR-D	Konstantinos Mitrogeorgos	konmitr@iit.demokritos.gr
NCSR-D	Georgios Stavrinos	gstavrinos@iit.demokritos.gr
NCSR-D	Stasinos Konstantopoulos	konstant@iit.demokritos.gr
NCSR-D	Angelos Charalambidis	acharal@iit.demokritos.gr
TenForce	Erika Pauwels	erika.pauwels@tenforce.com
TenForce	Aad Versteden	aad.versteden@tenforce.com
UoA	Georgios Papadakis	gpapadis@di.uoa.gr

Executive Summary

This report documents the deployment of the Big Data Integrator Platform and provides instructions on how to reproduce identical deployments. Various configurations and component mixtures of this generic platform will be used in the BDE pilots to serve exemplary use cases of the Horizon 2020 Societal Challenges.

The instructions in this document include (a) installation of the base system; (b) network topology and configuration; and (c) the components available as docker images and how they can be spawned and accessed to create pilot applications.

Abbreviations and Acronyms

BDI	Big Data Integrator
LOD	Linked Open Data
RAID	Redundant Array of Independent Disks
QEMU	Machine Emulator and Virtualizer

Table of Contents

List of Figures

1. Introduction

1.1 Purpose

This report documents the deployment of the generic *Big Data Integrator Platform (BDI)* at the NCSR-D cluster. Although the examples are based on this particular cluster, the instructions are generic and explanations are provided on how to generalize to different clusters.

1.1 Methodology

Section 2 of this document was first prepared by documenting all steps while the cluster was setup and the BDI platform was deployed. The draft produced in this manner was then given to a third person who was not involved in the first setup; this person re-did the whole installation from scratch while we were noting and improving places in the text that proved to be unclear for the third person. This process had to be repeated for parts of Section 2 as the project evolved from Apache Mesos to Docker Swarm for the base provisioning.

Section 3 archives the current state of the README and Wiki instructions from the BDE Github repositories.

2. Base System

Our example setup has one gateway node, one master computation node, and four worker nodes. The gateway node is the entry point to the infrastructure by the users and is also used for monitoring. The master node executes the head or master process for all tools that require such a head. The gateway and master nodes need considerably less CPU and memory resources than the slave nodes; in our setup we have mounted older and weaker servers on the rack for these nodes, allocating all four new servers as worker nodes.

All nodes are Virtual Machines (VM) deployed in separate physical machines. Each worker node uses two virtual drive images, one for the OS installation and one for data. This allows us to easily backup and restore the OS image or the data image in order to:

- Replace the OS installation image with a trusted OS image. This ensures that even if the computation nodes are compromised, any undetected back-doors are removed. This is applied periodically and after a breach is detected.
- Replace the OS installation image with an updated OS distribution to take advantage of new versions of the software bundled in the updated distribution.
- Backup and restore the data image, both for safety and to alternate between experiments that do not simultaneously fit in our storage.

2.1 Base Installation and Configuration

We used Debian 8 for all physical machines, as it is a stable and secure distribution. We assume a Debian 8 machine exposing only ssh to be trusted. The physical drive is partitioned as follows:

/	25 GB (RAID 1) this partition will host the OS
/srv	5934 GB (RAID 0) this partition will host the VM images
swap	8183 MB (RAID 1)

Virtual machines are executed on *Kernel-based Virtual Machine (KVM)* virtualization infrastructure[1] for the Linux kernel. The following command installs the necessary Debian packages:

```
sh> sudo apt-get install postfix htop tmux sudo qemu-kvm libvirt-bin virtinst bridge-utils kpartx
```

2.2 Network Topology

Each physical machine that hosts the master and slaves VMs has four physical network interfaces, used as follows:

eth0: management network

eth1: computation network (br10)

eth2: currently unused

eth3: SAN network (br172)

The physical machine of the gateway node has 4 interfaces used as follows:

eth0: management network

eth1: computation network (br10)

eth2: public network (brDMZ)

eth3: SAN network (br172)

The gateway node VM has an external IP through the brDMZ bridge in order to be accessible from external networks at:

bdegw.iit.demokritos.gr or **143.233.226.33**

The master and slave (worker) nodes have cluster-internal IPs (br10) and are only accessible from the gateway node.

2.3 Virtual Node Installation

We have used Ubuntu 14.04.3 for the master and worker node VMs, as Ubuntu regularly receives security fixes which can be installed as automatic updates. Version 14.04 is a stable

[1] cf. http://www.linux-kvm.org

release and will remain supported until April 2019. We will use the native Ubuntu software repositories to install the following:

- General purpose tools: ntp, htop, nmon, wget, git, etc.
- iptables-persistent in order to give internet access to the slave nodes.
- The docker-engine
- nginx for accessing services from outside the cluster's NAT.

In the virt-install commands given below we use the /srv/vmimages directory to create and store the VM images. Any directory can be used, as long as it exists before the virt-install commands are executed.

2.3.1 Ansible

We use Ansible 2.1.0 to configure the VM nodes and install and configure the base components. To install Ansible follow the instructions provided in the Ansible documentation.[2]

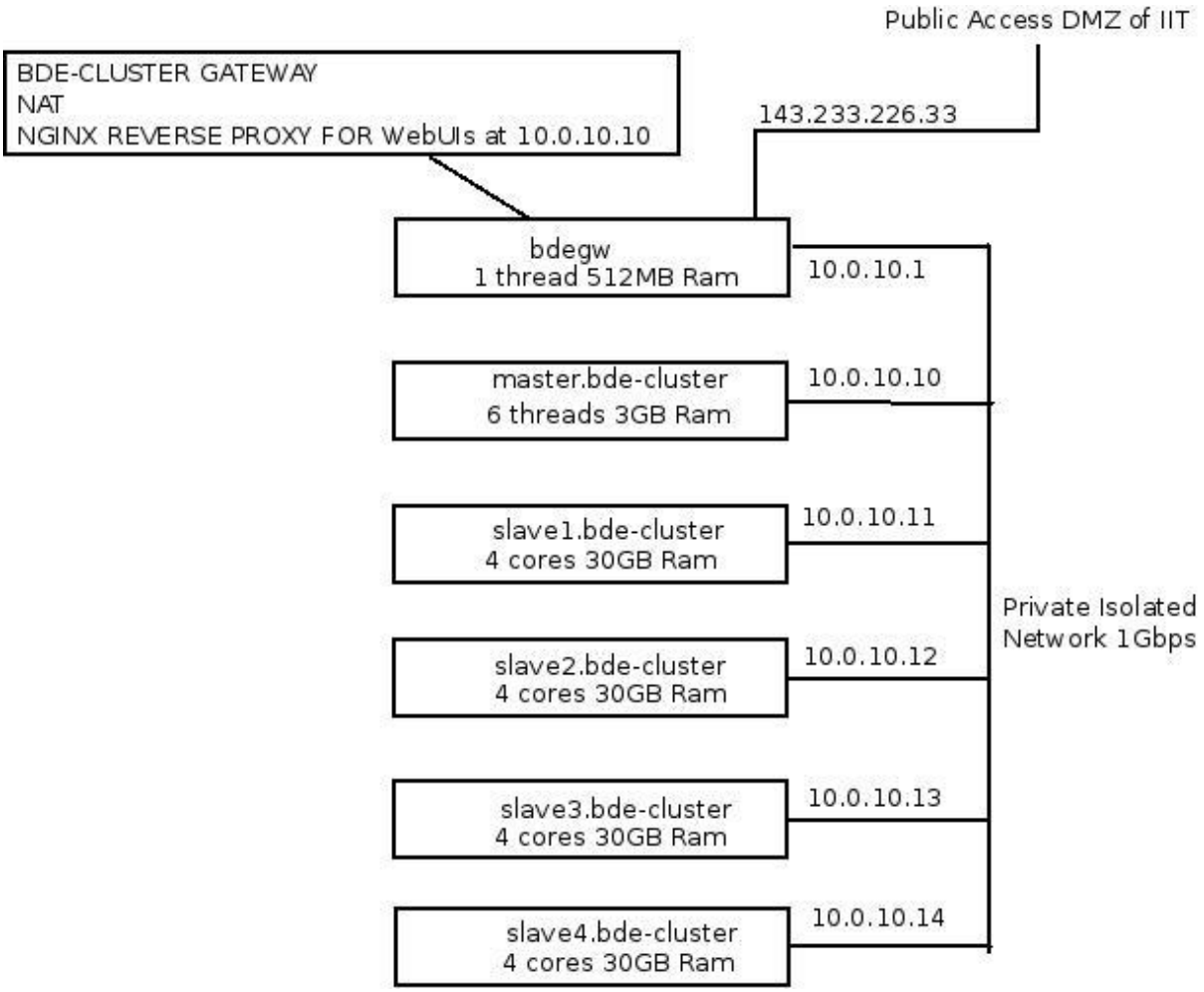

Figure 1: BDE cluster topology

[2] cf. http://docs.ansible.com/ansible/intro_installation.html

The BDE Ansible script is maintained in the big-data-europe/ansible Github repository.[3] To checkout a local copy, issue:

```
sh> git clone https://github.com/big-data-europe/ansible.git
```

In the remainder of this document, we will write <ANSIBLE_HOME> to refer to the directory where the BDE Ansible script has been downloaded.

This script configures the network interfaces, installs auxiliary packages, deploys and configures Docker Swarm. Cassandra and Hadoop are also deployed in the Swarm by default, in order to ease the transition from the previous version of the BDE infrastructure where Cassandra and Hadoop were installed natively. Cassandra and Hadoop might be removed from the default Swarm deployment in future versions. There are several configuration files to modify the Ansible script.

- ***<ANSIBLE_HOME>/ansible.cfg*** defines the username used for deployment.
- ***<ANSIBLE_HOME>/hosts*** configures the way your machine connects to the VMs over ssh.
- ***<ANSIBLE_HOME>/files/hosts*** defines the hostnames of the VMs in the cluster, it is used as the VMs hosts file.
- ***<ANSIBLE_HOME>/files/bdegw_interfaces*** configures the network interface that will be used by the gateway VM.

2.3.2 Gateway

On physical machine misc, execute the following command to create the gateway VM image /srv/vmimages/bdegw.img:

```
sh> sudo virt-install -n bdegw --ram 512 --vcpus=1 --os-type linux --disk
path=/srv/vmimages/bdegw.img,device=disk,bus=virtio,sparse=true,format=raw,size=25 --
location 'http://de.archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/' --graphics none
--accelerate --network bridge:br10,model=virtio --network bridge:brDMZ,model=virtio --extra-
args 'console=ttyS0,115200n8 serial' --autostart
```

Network configuration is as follows:

Hostname: bdegw

Domain: iit.demokritos.gr

Primary network interface: eth1

IP: 143.233.226.33

Netmask: 255.255.255.128

Gateway: 143.233.226.1

DNS: 143.233.226.2 143.233.226.3

Disk partitioning:

/ vda1 20 GB

swap vda2 all remaining space

Username: iitadmin

When the installer asks about automatic updates, choose “no”

[3] cf. https://github.com/big-data-europe/ansible

When the installer asks to choose software to install select “openssh server”

When the installer asks about the default system clock choose “UTC”

This machine acts as the gateway to all other nodes (master and slaves).It is publicly accessible from

bdegw.iit.demokritos.gr or **143.233.226.33**

Before continuing with the installation of the other nodes, the bdegw node must be configured so that the other nodes have Internet access.

For security reasons bdegw is accessible from ssh on port 222. The first thing you have to do is to change ssh port from default 22 to 222. To do so login to misc and then connect to bdegw with

```
sh> sudo virsh console bdegw
```

Login and then run the following commands to change the ssh port

```
sh> ssh iitadmin@bdegw.iit.demokritos.gr
sh> sudo sed -i 's/Port 22/Port 222/g' /etc/ssh/sshd_config
sh> sudo /etc/init.d/ssh restart
```

Then setup password-less ssh access from your machine by issuing:

```
sh> ssh -p222 iitadmin@bdegw.iit.demokritos.gr 'cat >> ~/.ssh/authorized_keys' < ~/.ssh/id_*.pub
```

After that configure bdegw with the ansible script by running:

```
sh> cd <ANSIBLE_HOME>
sh> ansible-playbook playbook.yaml -i hosts --limit gateway -K -vvvv
```

After the scripts finishes you are ready to configure the rest of the VMs.

2.3.3 Master Node

On physical machine melos0, execute the following command to create the master node VM image /srv/vmimages/master.img:

```
sh> sudo virt-install -n master --ram 3900 --vcpus=8 --os-type linux --disk
path=/srv/vmimages/master.img,device=disk,bus=virtio,sparse=true,format=raw,size=100 --location
'http://de.archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/' --graphics none --
accelerate --network bridge:br10,model=virtio --extra-args 'console=ttyS0,115200n8 serial' --
autostart
```

Network configuration is as follows:

Hostname: master

Domain name : bde-cluster

IP Address: 10.0.10.10

Netmask : 255.255.255.0

Gateway: 10.0.10.1

DNS 143.233.226.2 143.233.226.3

Disk partitioning:

/ vda1 96 GB

swap vda2 all remaining space

Username: iitadmin

When the installer asks about automatic updates, choose “no”

When the installer asks to choose software to install select “openssh server”

When the installer asks about the default system clock choose “UTC”

2.3.4 Worker Nodes

On the worker physical machines, execute the following command to create the worker node VM images:

```
sh> sudo virt-install -n <VM_NAME> --ram 8192 --vcpus=4 --os-type linux --disk
path=/srv/vmimages/<VM_NAME>.img,device=disk,bus=virtio,sparse=true,format=raw,size=25 --
disk
path=/srv/vmimages/<VM_NAME>_data.img,device=disk,bus=virtio,sparse=true,format=raw,size=2
500 --location 'http://de.archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/' --graphics
none --accelerate --network bridge:br10,model=virtio --extra-args 'console=ttyS0,115200n8 serial' -
-autostart
```

Replacing <VM_NAME> with slave1, slave2, etc.

Network configuration is as follows:

Hostname: slave1
Domain name : bde-cluster
IP Address: 10.0.10.11
Netmask : 255.255.255.0
Gateway: 10.0.10.1
DNS: 143.233.226.2 143.233.226.3

Disk partitioning: The / partition contains all system libraries and executables. The /srv partition contains data. This way the base system can be updated or replaced by just replacing the vda1 image without having to copy the data stored in the vdb1 image.

/ vda1 20 GB

swap vda2 all remaining space

/srv vdb1 whole disk

Username: iitadmin

When the installer asks about automatic updates, choose “no”

When the installer asks to choose software to install select “openssh server”

When the installer asks about the default system clock choose “UTC”

2.3.5 Master and Slave Node Configuration

All the VMs are configured using the Ansible script as described in the previous section.

To run the script you must first setup password-less ssh access to the nodes. To do so follow the instructions provided in the “BDE Cluster SSH Access” section of this document.

Then execute the ansible script:

```
sh> cd <ANSIBLE_HOME>
sh> ansible-playbook playbook.yaml -i hosts -K –vvvv
```

2.4 Recovery from Failures

In this section we describe how to recover from failures such as a VM crash or a hardware failure. There are two scenarios, a failure on the master node or a failure or on slave node.

2.4.1 Master Node Failure

If the master node stops then login in melos0 machine and start the master VM using:

```
sh> sudo virsh start master
```

In case of hardware failure then you should restore the setup from RAID1 and follow the steps above to restart the VM and services.

2.4.2 Slave Node Failure

In case a slave VM crashed then first you need to restart it from its host machine:

```
sh> sudo virsh start <VM_NAME>
```

Since HDFS, Cassandra, and in general all big data storage solutions used in BDE support replication, there is no data redundancy at the RAID level. To recover from the failure you should first start by setting up the new physical machine using steps described above (Sections 2.1 and 2.3). Then login to the physical machine and create the VM. In this scenario we will re-create slave1 with IP 10.0.10.11; replace the name and IP as appropriate:

```
sh> virt-install -n slave1 --ram 8192 --vcpus=4 --os-type linux --disk
path=/srv/vmimages/slave1.img,device=disk,bus=virtio,sparse=true,format=raw,size=25 --disk
path=/srv/vmimages/slave1_data.img,device=disk,bus=virtio,sparse=true,format=raw,size=2500 --
location 'http://de.archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/' --graphics none --
accelerate --network bridge:br10,model=virtio --extra-args 'console=ttyS0,115200n8 serial' --
autostart
```

Disk partitioning and network configuration is as described above. After finishing the installation, proceed to configure with Ansible.

To add the new slave to the docker swarm login to the node and run:

```
sh> docker -H=tcp://slave1:2375 run -d --restart always --name swarm-agent swarm join --
advertise=10.0.10.11:2375 consul://10.0.10.10:8500
```

2.5 Adding a New Node

In this section we describe how to add a new node in the cluster.

First you have to create a new VM to host the services. In the physical machine create a new VM with KVM and assign a new name e.g. slave5 with IP 10.0.10.15. In case you are replacing a node that has crashed due to hardware failure you should use the appropriate name that was used for that node, e.g. slave4

To create the VM run on the physical machine:

```
sh> virt-install -n slave5 --ram 8192 --vcpus=4 --os-type linux --disk
path=/srv/vmimages/slave5.img,device=disk,bus=virtio,sparse=true,format=raw,size=25 --disk
path=/srv/vmimages/slave5_data.img,device=disk,bus=virtio,sparse=true,format=raw,size=2500 --
location 'http://de.archive.ubuntu.com/ubuntu/dists/trusty/main/installer-amd64/' --graphics none --
accelerate --network bridge:br10,model=virtio --extra-args 'console=ttyS0,115200n8 serial' --
autostart
```

Installation-time choices, network configuration and disk partition are as in Section 2.5.2 above.

Add all nodes (bdegw, master and slaves) to /etc/hosts of all nodes:

```
10.0.10.10      master.bde-cluster      master
10.0.10.11      slave1.bde-cluster      slave1
10.0.10.12      slave2.bde-cluster      slave2
10.0.10.13      slave3.bde-cluster      slave3
10.0.10.14      slave4.bde-cluster      slave4
10.0.10.15      slave5.bde-cluster      slave5
```

In the <ANSIBLE_HOME>/files/hosts add line:

```
10.0.10.15  slave5.bde-cluster    slave5
```

as above. In the <ANSIBLE_HOME>/hosts file add in the “slaves” section the following line:

```
slave5.bde-cluster ansible_ssh_port=22 ansible_ssh_user=iitadmin
```

Then run the ansible script to configure the new node

```
sh> cd <ANSIBLE_HOME>
sh> ansible-playbook playbook.yaml -i hosts --limit slave5 -K -vvvv
```

To add the new slave to the docker swarm log into the node and run:

```
sh> docker -H=tcp://slave5:2375 run -d --restart always --name swarm-agent swarm join --
advertise=10.0.10.15:2375 consul://10.0.10.10:8500
```

2.6 Cloning a Node

Use the following commands to clone a VM. In the following example we clone slave1 to slave2. Replace the names accordingly for a different scenario (e.g. slave2 to slave3)

First you must shutdown the VM you want to clone using

```
sh> sudo virsh shutdown slave1
```

Then copy original images to cloned. Use sparse=always to copy only the real size and not the virtual disk space reserved by QEMU:

```
sh> sudo cp -p --sparse=always /srv/vmimages/slave1.img /srv/vmimages/slave2.img
```

```
sh> sudo cp -p --sparse=always /srv/vmimages/slave1_data.img /srv/vmimages/slave2_data.img
```

Clone the KVM XML definitions

```
sh> sudo virt-clone -o slave1 -n slave2 --file=/srv/vmimages/slave2.img --
file=/srv/vmimages/slave2_data.img --preserve-data
```

mount the images to edit the network configuration

```
sh> losetup /dev/loop0 /srv/vmimages/slave2.img
sh> kpartx -a /dev/loop0
sh> mount /dev/mapper/loop0p1 /mnt/
```

change the hostname

```
sh> echo slave2 > /mnt/etc/hostname
```

ensure that the following records exist to the /mnt/etc/hosts file otherwise add the missing records.

```
10.0.10.1   bdegw.bde-cluster    bdegw
10.0.10.10  master.bde-cluster   master
10.0.10.11  slave1.bde-cluster   slave1
10.0.10.12  slave2.bde-cluster   slave2
10.0.10.13  slave3.bde-cluster   slave3
10.0.10.14  slave4.bde-cluster   slave4
```

To change the ip address of the cloned VM edit file /mnt/etc/network/interfaces and replace line “address” with the IP of the new VM. e.g. for slave2 replace address 10.0.10.11 with address 10.0.10.12

Unmount the images, free up the resources, and start the VM:

```
sh> umount
sh> kpartx -d /dev/loop0
sh> losetup -d /dev/loop0
sh> sudo virsh start slave2
```

2.7 Pilot Swap

In this scenario the user wants to swap between pilots and keep the data of the previous pilot. To do so follow the steps bellow.

First shutdown the master and slave VMs using

```
sh> virsh shutdown <VMname>
```

where <VMname> the name of the VM in all host machines starting from the master. Then replace all the images of the stopped VMs with the images the new pilot is using. Then startup all VMs by using

```
sh> virsh start <VMname>
```

Always keep backup of the replaced images.

For example to replace the master VM log in melos0 machine and run

```
sh> virsh shutdown master
```

Then if you have an image for the master of the new pilot in /srv/vmimages/master_pilot2.img first take a backup of the running pilot 1 and then replace the image with pilot 2 image by running

```
sh> mv /srv/vmimages/master.img /srv/vmimages/master_pilot1.img
sh> mv /srv/vmimages/master_pilot2.img /srv/vmimages/master.img
```

After the mv command finishes start-up the VM using

```
sh> virsh start master
```

Note: You should always first replace all the images of master and slaves and then startup the VMs starting from the master VM.

2.8 BDE Cluster SSH Access

To access a machine within the cluster you must first ssh to the cluster gateway bdegw.iit.demokritos.gr (143.233.226.33) and from there ssh to the internal machine of your choice.

For example if you want to connect to master.bde-cluster you can run the following command

```
sh> ssh -A -t -p222 iitadmin@bdegw.iit.demokritos.gr ssh -A -t 10.0.10.10
```

which combines two ssh commands, one to connect to bdegw and from there to the master node.

To run the Ansible scripts you must first configure ssh and then enable password-less authentication on each VM you create.

First configure your ssh client through the ~/.ssh/config file. Add the following configuration to the file and you will be able to connect to the private hosts with one command.

```
Host bdegw
 HostName bdegw.iit.demokritos.gr
 User iitadmin
 Port 222
Host master.bde-cluster
 User iitadmin
 ProxyCommand ssh -q bdegw nc -q0 master.bde-cluster 22
Host slave1.bde-cluster
 User iitadmin
 ProxyCommand ssh -q bdegw nc -q0 slave1.bde-cluster 22
Host slave2.bde-cluster
 User iitadmin
 ProxyCommand ssh -q bdegw nc -q0 slave2.bde-cluster 22
Host slave3.bde-cluster
 User iitadmin
```

```
 ProxyCommand ssh -q bdegw nc -q0 slave3.bde-cluster 22
Host slave4.bde-cluster
 User iitadmin
 ProxyCommand ssh -q bdegw nc -q0 slave4.bde-cluster 22
```

After that you should be able to connect for example to slave1.bde-cluster machine by issuing

```
sh> ssh slave1.bde-cluster
```

Then enable password-less authentication in all VMs. This can be done by running

```
sh> ssh-copy-id <host>
```

Where <host> the host you want to configure. For example to enable password-less authentication for slave1 run

```
sh> ssh-copy-id slave1.bde-cluster
```

2.9 Port Forwarding

The base installation also deploys a nginx server on the bdegw node. By using nginx you can make ports and UIs available through the public bdegw.iit.demokritos.gr URL.

To do so you must add a server section in file /etc/nginx/conf.d/tcp_forwarding.conf on bdegw node like below

```
server {
 listen          143.233.226.33:<EXPOSED_PORT>;
 location / {
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://<HOST>:<PORT>/;
 }
}
```

Replace **<PORT>** and **<HOST>** with those you want to forward and **<EXPOSED_PORT>** with the port that will be publicly available. Then you must restart nginx with

```
sh> sudo service nginx restart
```

After that you can access the service from bdegw.iit.demokritos.gr:**<PORT>**.

For example if you want to expose the Web UI running on 10.0.10.10:11111 the server section should be the following

```
server {
 listen          143.233.226.33:11111;
 location / {
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass http://10.0.10.10:11111;
 }
}
```

Then issue

```
sh> sudo service nginx restart
```

After that you can access the GUI from bdegw.iit.demokritos.gr:11111.

Another way to temporarily forward ports is to use ssh tunneling. To do so from your machine run

```
sh> ssh -L localhost:<local_port>:<VM_IP>:<VM_port> -p222 iitadmin@bdegw.iit.demokritos.gr
```

and replace **<local_port>** with the local port you are going to use, **<VM_IP>** the IP of the node you want to access and **<VM_port>** the port in the VM you want to access. For example if you want to access a service listening on port 9042 in slave4 from localhost:9042 run

```
sh> ssh -L localhost:9042:10.0.10.14:9042 -p222 iitadmin@bdegw.iit.demokritos.gr
```

2.10 Docker Networking

The base installation configures a Docker network named “bde_net”. This network uses the overlay driver and creates the 10.10.0.12/16 subnet. Our testing setup has Cassandra and Hadoop containers always running, they are part of the bde_net network. If you need to access these, your container or pipeline need to run in the same network. We recommend the bde_net network for all BDE components. To create the network in the ansible script we used the command

```
sh> docker -H=tcp://master:4000 network create --driver overlay --subnet=10.10.0.0/16 bde_net
```

3. Other BDE Components

Although the components of a pipeline are all contained in the docker-compose.yml specifying the pipeline, components can be started manually for testing purposes. In this section we describe how to deploy BDE components. All dockers can be deployed in any node. You can define the the node by setting the node's hostname on the constraint:node parameter. All dockers should be deployed in the Docker Swarm and all Docker commands can be executed from any slave node or from the master node. More info about each container can be found at BDE wiki[4].

3.1 Apache Spark

To deploy a standalone Spark cluster with one Spark master and multiple Spark workers you must first deploy the master and then the workers. To deploy the master issue

```
sh> docker -H tcp://master:4000 run --name spark-master -h spark-master --net=bde_net -e
constraint:node==master -d bde2020/spark-master:1.5.1-hadoop2.6
```

[4] https://github.com/big-data-europe/README/wiki

After the command finishes a container will be deployed on the master node running a Spark master that accepts connections on spark://spark-master:7077 on the bde_net network. Then you can deploy the workers. For example to deploy a Spark worker at slave1 node issue

```
sh> docker -H tcp://master:4000 run --name spark-worker-1 --net=bde_net -e
constraint:node==slave1 -d bde2020/spark-worker:1.5.1-hadoop2.6
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'

services:

 spark-master:
      image: bde2020/spark-master:1.5.1-hadoop2.6
      hostname: spark-master
      container_name: spark-master
      environment:
      - "constraint:node==master"
      networks:
      - default

 spark-worker-1:
      image: bde2020/spark-worker:1.5.1-hadoop2.6
      container_name: spark-worker-1
      environment:
      - "constraint:node==slave1"
      networks:
      - default
      depends_on:
      - spark-master

 spark-worker-2:
      image: bde2020/spark-worker:1.5.1-hadoop2.6
      container_name: spark-worker-2
      environment:
      - "constraint:node==slave2"
      networks:
      - default
      depends_on:
      - spark-master
```

```
  spark-worker-3:
      image: bde2020/spark-worker:1.5.1-hadoop2.6
      container_name: spark-worker-3
      environment:
      - "constraint:node==slave3"
      networks:
      - default
      depends_on:
      - spark-master

  spark-worker-4:
      image: bde2020/spark-worker:1.5.1-hadoop2.6
      container_name: spark-worker-4
      environment:
      - "constraint:node==slave4"
      networks:
      - default
      depends_on:
      - spark-master

networks:
 default:
      external:
      name: bde_net
```

3.2 OpenLink Virtuoso

To deploy a Virtuoso Docker container for example on the slave1 node issue

```
sh> docker -H tcp://master:4000 run --name my-virtuoso -p 8890:8890 -p 1111:1111 -e
DBA_PASSWORD=myDbaPassword -e SPARQL_UPDATE=true -e
DEFAULT_GRAPH=http://www.example.com/my-graph -v /my/path/to/the/virtuoso/db:/data --
net=bde_net -e constraint:node==slave1 -d tenforce/virtuoso
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'

services:

 virtuoso:
      image: tenforce/virtuoso
```

```
    container_name: my-virtuoso
    ports:
    - "8890:8890"
    - "1111:1111"
    environment:
    - "DBA_PASSWORD=myDbaPassword"
    - "SPARQL_UPDATE=true"
    - "DEFAULT_GRAPH=http://www.example.com/my-graph"
    - "constraint:node==slave1"
    volumes:
    - /my/path/to/the/virtuoso/db:/data
    networks:
    - default

networks:
 default:
    external:
    name: bde_net
```

3.3 Semagrow and Sevod-Scraper

In order to use Semagrow you must first provide a sevod[5] description for the federated datasets. If you want to create one you can use the docker container for the sevod-scraper tool. The tool creates a sevod description both for a triple store and Cassandra.

To create sevod description from a RDF dump run

```
sh> docker run --rm -it -v </path/to/output>:/output -v </path/to/dump>:/dump semagrow/sevod-scraper rdfdump /dump/<dump_name> <endpoint_url> <flags> /output/<output_name>
```

Where:

- **/path/to/output** the directory to write the output
- **/path/to/dump** the directory that contains the dump
- **dump_name** the filename of the dump
- **endpoint_url** the endpoint URL where the dump is stored
- **flags** the flags to run sevod-scraper
- **output_name** the the filename of the output which will be located at **/path/to/output**/output_name

To create sevod description from Cassandra run

```
sh> docker run --rm -it -v </path/to/output>:/output -v semagrow/sevod-scraper cassandra <cassandra_ip> <cassandra_port> <keyspace> <base_url> /output/<output_name>
```

Where:

[5] https://www.w3.org/2015/03/sevod

- **/path/to/output** the directory to write the output
- **cassandra_ip** the IP of the Cassandra store
- **cassandra_port** the port of the Cassandra store
- **keyspace** the Cassandra keyspace to scrap
- **base_ur**l the base url to use in the output
- **output_name** the the filename of the output which will be located at **/path/to/output**/output_name

To deploy a Semagrow Docker container, for example on the master node, issue

```
sh> docker -H tcp://master:4000 run -d --name semagrow -p 8080:8080 -v
/path/to/sevod:/etc/default/semagrow -e constraint:node==master semagrow/semagrow
```

To deploy Semagrow with the Cassandra extension run

```
sh> docker -H tcp://master:4000 run -d --name semagrow -p 8080:8080 -v
/path/to/sevod:/etc/default/semagrow -e constraint:node==master semagrow/semagrow-cassandra
```

3.4 4store

To deploy a 4store cluster you must first deploy all datanode containers in the slave nodes and then the 4store master in the master node. To deploy the datanodes issue

```
sh> docker -H tcp://master:4000 run -d --name 4store1 --net=bde_net -v /srv/4store:/var/lib/4store -
e constraint:node==slave1 bde2020/4store

sh> docker -H tcp://master:4000 run -d --name 4store2 --net=bde_net -v /srv/4store:/var/lib/4store -
e constraint:node==slave2 bde2020/4store

sh> docker -H tcp://master:4000 run -d --name 4store3 --net=bde_net -v /srv/4store:/var/lib/4store -
e constraint:node==slave3 bde2020/4store

sh> docker -H tcp://master:4000 run -d --name 4store4 --net=bde_net -v /srv/4store:/var/lib/4store -
e constraint:node==slave4 bde2020/4store
```

Then deploy the 4store master container on the master node by running

```
sh> docker -H tcp://master:4000 run -d --name 4store-master --net=bde_net -v
/srv/4store:/var/lib/4store -e constraint:node==master -e
STORE_NODES="4store1;4store2;4store3;4store4" bde2020/4store-master
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'


services:


 4store1:
      image: bde2020/4store
      hostname: 4store1
      container_name: 4store1
      volumes:
      - /srv/4store:/var/lib/4store
      environment:
```

```
    - "constraint:node==slave1"
    networks:
    - default

4store2:
    image: bde2020/4store
    hostname: 4store2
    container_name: 4store2
    volumes:
    - /srv/4store:/var/lib/4store
    environment:
    - "constraint:node==slave2"
    networks:
    - default

4store3:
    image: bde2020/4store
    hostname: 4store3
    container_name: 4store3
    volumes:
    - /srv/4store:/var/lib/4store
    environment:
    - "constraint:node==slave3"
    networks:
    - default

4store4:
    image: bde2020/4store
    hostname: 4store4
    container_name: 4store4
    volumes:
    - /srv/4store:/var/lib/4store
    environment:
    - "constraint:node==slave4"
    networks:
    - default

4store-master:
```

```
    image: bde2020/4store-master
    hostname: 4store-master
    container_name: 4store-master
    volumes:
    - /mnt/synology/4store-dumps:/dumps
    networks:
    - default
    environment:
    - STORE_NODES=4store1;4store2;4store3;4store4
    - "constraint:node==master"
    depends_on:
    - 4store1
    - 4store2
    - 4store3
    - 4store4

networks:
 default:
    external:
    name: bde_net
```

3.5 GeoTriples

To deploy GeoTriples in slave1 run

```
sh> docker -H tcp://master:4000 run -it --name geotriples --net=bde_net -e
constraint:node==slave1 bde2020/geotriples bash
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'


services:


 geotriples:
    image: bde2020/geotriples
    container_name: geotriples
    environment:
    - "constraint:node==slave1"
    networks:
    - default
```

```
networks:
 default:
      external:
      name: bde_net
```

3.6 Sextant

To deploy Sextant for example in the master node run

```
sh> docker -H tcp://master:4000 run --name sextant -p 9999:8080 -e constraint:node==master --
net=bde_net -d bde2020/sextant
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'

services:

 sextant:
      image: bde2020/sextant
      container_name: sextant
      environment:
      - "constraint:node==master"
      ports:
      - "8890:8890"
      networks:
      - default

networks:
 default:
      external:
      name: bde_net
```

If you want to make Sextant's UI publicly available you must forward the UI through the bdegw node. To do so log into bdegw and add in file /etc/nginx/conf.d/tcp_forwarding.conf the following section

```
server {
 listen              143.233.226.33:9999;
 location / {
          proxy_set_header X-Forwarded-Host $host;
          proxy_set_header X-Forwarded-Server $host;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
          proxy_pass http://10.0.10.10:9999/;
```

```
 }
}
```

Then restart nginx with

```
sh> sudo service nginx restart
```

The Sextant interface will now be available at bdegw.iit.demokritos.gr:9999

3.7 Silk

To deploy Sextant for example in the master node run

```
sh> docker -H tcp://master:4000 run --name silk -p 9999:8080 -e constraint:node==master --
net=bde_net -d bde2020/silk-workbench
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'


services:


 silk:
      image: bde2020/silk-workbench
      container_name: silk
      environment:
      - "constraint:node==master"
      ports:
      - "8890:8890"
      networks:
      - default


networks:
 default:
      external:
      name: bde_net
```

If you want to make Silk's UI publicly available you must forward the UI through the bdegw node. To do so log into bdegw and add in file /etc/nginx/conf.d/tcp_forwarding.conf the following section:

```
server {
 listen              143.233.226.33:9000;
 location / {
          proxy_set_header X-Forwarded-Host $host;
          proxy_set_header X-Forwarded-Server $host;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
            proxy_pass http://10.0.10.10:9000/;
        }
    }
```

Then restart nginx with

```
sh> sudo service nginx restart
```

The Silk interface will now be available at bdegw.iit.demokritos.gr:9000

3.8 PostGIS

To deploy PostGIS for example at slave1 run

```
sh> docker -H tcp://master:4000 run -d --name postgis --net=bde_net -p 5432:5432 -v
/srv/postgis:/var/lib/postgresql/9.4/main -e constraint:node==slave1 bde2020/postgis
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'


services:


 postgis:
      image:bde2020/postgis
      container_name: postgis
      environment:
      - "constraint:node==slave1"
      ports:
      - "5432:5432"
      volumes:
      - /srv/postgis:/var/lib/postgresql/9.4/main
      networks:
      - default


networks:
 default:
      external:
      name: bde_net
```

3.9 Strabon

To deploy Strabon first you have to setup a postgis with hostname postgis (or simply run command from 3.8) at and then run

```
sh> docker -H tcp://master:4000 run -d --name strabon --net=bde_net -p 9999:8080 -e
constraint:node==slave1 bde2020/strabon
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'

services:

 strabon:
      image: bde2020/strabon
      container_name: strabon
      environment:
      - "constraint:node==slave1"
      ports:
      - "9999:8080"
      networks:
      - default

networks:
 default:
      external:
      name: bde_net
```

3.10 Hive

To deploy Hive you must first deploy the Hadoop cluster and then configure Hive to connect to it. The Hive container starts hiveserver2 on port 10000 so you can then connect remotely to Hive and issue queries. To deploy HIve for example on slave2 accessible at “hive:10000” run:

```
sh> docker -H tcp://master:4000 run -d --name hive -v /srv/hive:/hive-metastore --net=bde_net -e
constraint:node==slave2 bde2020/hive
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'

services:

 hive:
      image: bde2020/hive
      container_name: hive
      environment:
```

```
      - "constraint:node==slave2"
      volumes:
      - /srv/hive:/hive-metastore
      networks:
      - default

networks:
 default:
      external:
      name: bde_net
```

3.11 Cassandra

To deploy Cassandra for example on slave1 run

```
sh> docker -H tcp://master:4000 run -d --name cassandra-cluster-1 -v
/srv/cassandra:/var/lib/cassandra --net=bde_net -e constraint:node==slave1 bde2020/cassandra
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'
services:

 cassandra-cluster-1:
      image: bde2020/cassandra
      container_name: cassandra-cluster-1
      volumes:
      - /srv/cassandra:/var/lib/cassandra
      environment:
      - "constraint:node==slave1"
      networks:
      - default

networks:
 default:
      external:
      name: bde_net
```

3.12 Hadoop

To deploy a Hadoop cluster with the namenode running on master and the datanodes on the slaves you must first clone the docker-hadoop repository and go into the clone directory by running on any node

```
sh> git clone https://github.com/big-data-europe/docker-hadoop.git
sh> docker-hadoop
```

Then start the cluster by running

```
sh> docker -H tcp://master:4000 run -d --name namenode -v
/srv/hadoop/namenode:/hadoop/dfs/name --net=bde_net -h namenode.hadoop -e
constraint:node==master -e CLUSTER_NAME=bde-cluster --env-file=./hadoop.env
bde2020/hadoop-namenode:1.0.0

sh> docker -H tcp://master:4000 run -d --name datanode1 -v
/srv/hadoop/datanode:/hadoop/dfs/data --net=bde_net -h datanode1.hadoop -e
constraint:node==slave1 --env-file=./hadoop.env bde2020/hadoop-datanode:1.0.0

sh> docker -H tcp://master:4000 run -d --name datanode2 -v
/srv/hadoop/datanode:/hadoop/dfs/data --net=bde_net -h datanode2.hadoop -e
constraint:node==slave2 --env-file=./hadoop.env bde2020/hadoop-datanode:1.0.0

sh> docker -H tcp://master:4000 run -d --name datanode3 -v
/srv/hadoop/datanode:/hadoop/dfs/data --net=bde_net -h datanode3.hadoop -e
constraint:node==slave3 --env-file=./hadoop.env bde2020/hadoop-datanode:1.0.0

sh> docker -H tcp://master:4000 run -d --name datanode4 -v
/srv/hadoop/datanode:/hadoop/dfs/data --net=bde_net -h datanode4.hadoop -e
constraint:node==slave4 --env-file=./hadoop.env bde2020/hadoop-datanode:1.0.0
```

For the BDE platform you should use the following docker compose snippet

```
version: '2'


services:


 namenode:
      image: bde2020/hadoop-namenode:1.0.0
      hostname: namenode
      container_name: namenode
      domainname: hadoop
      volumes:
      - /mnt/synology/hadoop/namenode:/hadoop/dfs/name
      environment:
      - CLUSTER_NAME=bde_cluster
      - "constraint:node==master"
      env_file:
      - ./hadoop.env
      networks:
```

```
      - default

datanode1:
      image: bde2020/hadoop-datanode:1.0.0
      hostname: datanode1
      container_name: datanode1
      domainname: hadoop
      volumes:
      - /srv/hadoop/datanode:/hadoop/dfs/data
      env_file:
      - ./hadoop.env
      environment:
 - "constraint:node==slave1"
      networks:
 - default
      depends_on:
      - namenode

datanode2:
      image: bde2020/hadoop-datanode:1.0.0
      hostname: datanode2
      container_name: datanode2
      domainname: hadoop
      volumes:
      - /srv/hadoop/datanode:/hadoop/dfs/data
      env_file:
      - ./hadoop.env
      environment:
 - "constraint:node==slave2"
      networks:
 - default
      depends_on:
      - namenode

datanode3:
      image: bde2020/hadoop-datanode:1.0.0
      hostname: datanode3
      container_name: datanode3
```

```
        domainname: hadoop
        volumes:
        - /srv/hadoop/datanode:/hadoop/dfs/data
        env_file:
        - ./hadoop.env
        environment:
    - "constraint:node==slave3"
        networks:
    - default
        depends_on:
        - namenode

  datanode4:
        image: bde2020/hadoop-datanode:1.0.0
        hostname: datanode4
        container_name: datanode4
        domainname: hadoop
        volumes:
        - /srv/hadoop/datanode:/hadoop/dfs/data
        env_file:
        - ./hadoop.env
        environment:
    - "constraint:node==slave4"
        networks:
    - default
        depends_on:
        - namenode

networks:
  default:
        external:
        name: bde_net
```

4. Conclusion and Future

The base platform has been deployed on the NCSR infrastructure. The installation procedure was independently tested. The cluster as installed here will be used as a platform for testing the SC implementations. This infrastructure will continue to be updated as necessary.

Currently, components should know the IP of the component they are composed with. In order to facilitate completely moving from raw docker commands to docker-compose.yml files/snippets we need to address this in the next iteration of the Generic Big Data Integrator Instance. This will allow components to be deployed with the BDE Pipeline UI.