Support Action

Big Data Europe – Empowering Communities with Data Technologies

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

Deliverable 5.5
Generic Big Data Integrator Instance II

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due Date of Deliverable</td>
<td>Month 24, 31/12/2016</td>
</tr>
<tr>
<td>Actual Submission Date</td>
<td>31/1/2017</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP5, Big Data Integrator Instances</td>
</tr>
<tr>
<td>Task</td>
<td>T5.1</td>
</tr>
<tr>
<td>Type</td>
<td>Other</td>
</tr>
<tr>
<td>Approval Status</td>
<td>Submitted</td>
</tr>
<tr>
<td>Version</td>
<td>Final</td>
</tr>
<tr>
<td>Number of Pages</td>
<td></td>
</tr>
<tr>
<td>Filename</td>
<td></td>
</tr>
</tbody>
</table>

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.

The information in this document reflects only the author’s views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided “as is” without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.
History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1/08/2016</td>
<td>Document structure and initial draft, assuming D5.1 as a starting point.</td>
<td>S. Konstantopoulos</td>
</tr>
<tr>
<td>0.01</td>
<td>22/09/2016</td>
<td>Solved the issue noted in D5.1, Section 4 regarding using Docker Compose</td>
<td>I. Mouchakis</td>
</tr>
<tr>
<td>0.02</td>
<td>10/10/2016</td>
<td>Improvement of the platform installation scripts and update of the relevant instructions in Section 2.3</td>
<td>I. Mouchakis</td>
</tr>
<tr>
<td>0.03</td>
<td>02/11/2016</td>
<td>Updated the description of the Docker network topology (Section 2) to reflect changes in BDE platform.</td>
<td>I. Mouchakis, A. Charalambidis</td>
</tr>
<tr>
<td>0.04</td>
<td>10/12/2016</td>
<td>Updated usage guides for Spark, Virtuoso, Semagrow, GeoTriples, Sextant, Silk (Section 3)</td>
<td>A. Versteden, G. Papadakis, I. Mouchakis, A. Charalambidis</td>
</tr>
<tr>
<td>0.05</td>
<td>15/12/2016</td>
<td>Testing of the deployment instructions with three experiment subjects and editorial changes and improvements.</td>
<td>I. Mouchakis, G. Stavrinos, A. Charalambidis, S. Konstantopoulos and two NCSR-D staff not working for BDE</td>
</tr>
<tr>
<td>0.06</td>
<td>19/01/2016</td>
<td>Final synchronization with README files on Github</td>
<td>I. Mouchakis</td>
</tr>
<tr>
<td>0.07</td>
<td>24/01/2017</td>
<td>Peer review</td>
<td>H. Jabeen, A. Versteden</td>
</tr>
<tr>
<td>0.08</td>
<td>25/01/2017</td>
<td>Address peer review comments</td>
<td>I. Mouchakis, S. Konstantopoulos</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>Final version</td>
<td></td>
</tr>
</tbody>
</table>

Author List

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Name</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCSR-D</td>
<td>Ioannis Mouchakis</td>
<td><a href="mailto:gmouchakis@iit.demokritos.gr">gmouchakis@iit.demokritos.gr</a></td>
</tr>
<tr>
<td>Institution</td>
<td>Name</td>
<td>Email</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>NCSR-D</td>
<td>Georgios Stavrinos</td>
<td><a href="mailto:gstavrinos@iit.demokritos.gr">gstavrinos@iit.demokritos.gr</a></td>
</tr>
<tr>
<td>NCSR-D</td>
<td>Stasinos Konstantopoulos</td>
<td><a href="mailto:konstant@iit.demokritos.gr">konstant@iit.demokritos.gr</a></td>
</tr>
<tr>
<td>NCSR-D</td>
<td>Angelos Charalambidis</td>
<td><a href="mailto:acharal@iit.demokritos.gr">acharal@iit.demokritos.gr</a></td>
</tr>
<tr>
<td>UoA</td>
<td>G. Papadakis</td>
<td></td>
</tr>
<tr>
<td>TenForce</td>
<td>Aad Versteden</td>
<td><a href="mailto:aad.versteden@tenforce.com">aad.versteden@tenforce.com</a></td>
</tr>
<tr>
<td>UoB</td>
<td>Hajira Jabeen</td>
<td><a href="mailto:hajirajabeen@gmail.com">hajirajabeen@gmail.com</a></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDI</td>
<td>Big Data Integrator</td>
</tr>
<tr>
<td>LOD</td>
<td>Linked Open Data</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks</td>
</tr>
<tr>
<td>QEMU</td>
<td>Machine Emulator and Virtualizer</td>
</tr>
</tbody>
</table>
# Table of Contents

Contents

1. Introduction .................................................................................................................. 9  
   1.1 Purpose .................................................................................................................. 9  
   1.2 Methodology ......................................................................................................... 9  
2. Base System .................................................................................................................. 10  
   2.1 Base Installation and Configuration .................................................................. 10  
   2.2 Network Topology ............................................................................................... 11  
   2.3 Virtual Node Installation ................................................................................... 11  
      2.3.1 Ansible .......................................................................................................... 11  
      2.3.2 Gateway ....................................................................................................... 13  
      2.3.3 Master node ................................................................................................. 14  
      2.3.4 Slave nodes ................................................................................................. 15  
      2.3.5 Master and slave node configuration ........................................................... 16  
   2.4 Recovery from Failures ....................................................................................... 16  
      2.4.1 Master node failure ...................................................................................... 16  
      2.4.2 Slave node failure ....................................................................................... 17  
   2.5 Adding a New Node ............................................................................................ 17  
   2.6 Cloning a Node .................................................................................................... 18  
   2.7 Pilot Swap ........................................................................................................... 20  
   2.8 BDE Cluster SSH Access ................................................................................... 20  
   2.9 Port Forwarding .................................................................................................... 21  
3. Other BDE components ............................................................................................... 23  
   3.1 Apache Spark ....................................................................................................... 23  
   3.2 OpenLink Virtuoso .............................................................................................. 25  
   3.3 Semagrow and sevod-scraper ............................................................................ 26  
   3.4 4store ................................................................................................................... 27  
   3.5 GeoTriples ........................................................................................................... 29
3.6 Sextant .......................................................... 30
3.7 Silk.................................................................. 31
3.8 PostGIS............................................................ 32
3.9 Strabon............................................................ 32
3.10 Hive ............................................................... 33
4. Conclusion.......................................................... 39
List of Tables

Figure 1: BDE Cluster Topology

12
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.

1.2 Methodology
Section 2 of this document was first prepared by documenting all steps while the cluster was set up 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 executed the whole installation procedure from scratch while we were noting and improving places in the text that proved to be unclear for the third person. The trial was repeated with two further trial subjects who are NCSR-D staff, not working for the BDE action and who had never deployed or used Big Data tools.

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 slave 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 slave nodes.

All nodes are Virtual Machines (VM) deployed in separate physical machines. Each slave 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
```

\(^1\) cf. http://www.linux-kvm.org
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 nodes have cluster-internal IPs (br10) and are only accessible from the gateway node.

2.3 Virtual Node Installation
We have used Ubuntu 14.04.5 for the master and slave node VMs, as Ubuntu regularly receives security fixes which can be installed as automatic updates. Version 14.04 is a stable 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.
- 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.\(^2\)

The BDE Ansible script is maintained in the big-data-europe/ansible Github repository. 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 the Docker Engine, Docker Compose and Docker Swarm. The script on the “master” branch of the ansible repository is always configured for the NCSR cluster. If your cluster is different edit the following files accordingly.

- `<ANSIBLE_HOME>/ansible.cfg` defines the username used for deployment.
- `<ANSIBLE_HOME>/hosts` configures the way your machine connects to the VMs over ssh and Ansible variables for the hosts.
- `<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 the physical machine that will host the gateway VM execute the following command as root to create the VM:

```
sh> 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”
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 the physical machine that hosts the gateway VM bdegw and then connect to it 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
```

When prompted for SUDO password enter the password for “bdegw”.

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

### 2.3.3 Master node
On physical machine that will host the master node VM execute the following command as root to create VM:

```
sh> 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=300 --location cd
'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 290 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 Slave nodes

On the slave physical machines, execute the following command as root to create the slave node VM images:

```sh
```

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 “2.8 BDE Cluster SSH Access” section of this document. Then execute the ansible script:

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

When prompted for SUDO password enter the password for master and slaves.

After ansible finishes a Docker Swarm cluster should be deployed in all nodes. To verify the installation wait for two minutes for Swarm to configure itself and then login in any master or slave node and execute:

```
sh> docker -H tcp://master:4000 info
```

If Swarm was configured correctly the above command will list all nodes in the Swarm cluster marked as “Healthy”.

### 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 the physical machine that hosts the master VM and start the 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

If a slave node stops then login in its host machine and start it using:

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

To recover from a hardware 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 configure the node and add it to Docker Swarm run:

```
sh> cd <ANSIBLE_HOME>
sh> ansible-playbook playbook.yaml -i hosts --limit slave1.bde-cluster -K -vvvv
```

When prompted for SUDO password enter the password for “slave1”.

To verify that the node was added correctly in the Swarm cluster wait for two minutes for Swarm to configure itself and then login in any master or slave node and execute:

```
sh> docker -H tcp://master:4000 info
```

If Swarm was configured correctly the above command will list all nodes in the Swarm cluster marked as “Healthy”.

### 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.

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
```
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 there is a section named “slaves”. In that section add the following line:

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

The new node should be also added in your ~/.ssh/config file as described in section 2.8 BDE cluster SSH access.

Then run the ansible script to configure the new node and add it to Docker Swarm

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

When prompted for SUDO password enter the password for “slave5”.

To verify that the node was added correctly in the Swarm cluster wait for two minutes for Swarm to configure itself and then login in any master or slave node and execute:

```
sh> docker -H tcp://master:4000 info
```

If Swarm was configured correctly the above command will list all nodes in the Swarm cluster marked as “Healthy”.

### 2.6 Cloning a Node

Use the following commands as root 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> 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> cp -p --sparse=always /srv/vmimages/slave1.img:q /srv/vmimages/slave2.img
sh> cp -p --sparse=always /srv/vmimages/slave1_data.img
/srv/vmimages/slave2_data.img

Clone the KVM XML definitions

sh> 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

e nsure 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 /mnt
sh> kpartx -d /dev/loop0
sh> losetup -d /dev/loop0
sh> 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 the physical machine that hosts the master VM 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 run the Ansible scripts you must first configure ssh and then enable password-less authentication on each VM you create. The same procedure is proposed for accessing the cluster with SSH.

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. Replace `iitadmin` with the name of your user.

```
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

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

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

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

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

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

Alternatively you can connect to the cluster without configuring SSH. 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 as user iitadmin you can run the following command

```bash
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.

### 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

```nginx
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

```nginx
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

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 wiki4.

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 -e constraint:node==master -d bde2020/spark-master:1.5.1-hadoop2.6

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. 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 -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:

4 https://github.com/big-data-europe/README/wiki
image: bde2020/spark-master:1.5.1-hadoop2.6
hostname: spark-master
container_name: spark-master
environment:
  - "constraint:node==master"

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

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"

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"

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"
  depends_on:
  - spark-master

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

```
```

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

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

```bash
sh> docker run --rm -it -v </path/to/output>:/output -v </path/to/dump>:/dump semagrow/sevod-scraper rddfump /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

```bash
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:
- `/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_url` 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

\(^5\) https://www.w3.org/2015/03/sevod
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
```

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 -v
/srv/4store:/var/lib/4store -e constraint:node==slave1 bde2020/4store

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

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

sh> docker -H tcp://master:4000 run -d --name 4store4 -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 -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"

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

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

4store4:
  image: bde2020/4store
  hostname: 4store4
  container_name: 4store4
  volumes:
  - /srv/4store:/var/lib/4store
  environment:
  - "constraint:node==slave4"
4store-master:
    image: bde2020/4store-master
hostname: 4store-master
container_name: 4store-master
volumes:
    - /mnt/synology/4store-dumps:/dumps
environment:
    - STORE_NODES=4store1;4store2;4store3;4store4
    - "constraint:node==master"
depends_on:
    - 4store1
    - 4store2
    - 4store3
    - 4store4

3.5 GeoTriples
To deploy GeoTriples in slave1 run

```
sh> docker -H tcp://master:4000 run -it --name geotriples -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"
```
3.6 Sextant

To deploy Sextant for example in the master node run

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

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

```yaml
version: '2'

services:

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

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

```yaml
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
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 -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"
```

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 at slave1 run

```
```

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

```yaml
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
```

### 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 -p 9999:8080 -e constraint:node==slave1 bde2020/strabon
```

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

```yaml
version: '2'
```
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 -e constraint:node==slave2 bde2020/hive
```

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

```
services:

hive:
  image: bde2020/hive
  container_name: hive
  environment:
    - "constraint:node==slave2"
  volumes:
    - /srv/hive:/hive-metastore
```
3.11 Cassandra

To deploy Cassandra for example on slave1 run

```
sh> docker -H tcp://master:4000 run -d --name cassandra-cluster-1 -e constraint:node==slave1 bde2020/cassandra
```

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

```yaml
version: '2'
services:

  cassandra-cluster-1:
    image: bde2020/cassandra
    container_name: cassandra-cluster-1
    environment:
      - "constraint:node==slave1"

To deploy a Cassandra cluster on all nodes use the following docker compose snippet

```yaml
version: '2'
services:

  cassandra-cluster-1:
    image: bde2020/cassandra
    container_name: cassandra-cluster-1
    restart: always
    environment:
      - "CASSANDRA_BROADCAST_ADDRESS=cassandra-cluster-1"
      - "constraint:node==slave1"

  cassandra-cluster-2:
    image: bde2020/cassandra
    container_name: cassandra-cluster-2
    restart: always
```
environment:
- "CASSANDRA_BROADCAST_ADDRESS=cassandra-cluster-2"
- "CASSANDRA_SEEDS=cassandra-cluster-1"
- "constraint:node==slave2"

depends_on:
- cassandra-cluster-1

cassandra-cluster-3:
  image: bde2020/cassandra
  container_name: cassandra-cluster-3
  restart: always
  environment:
    - "CASSANDRA_BROADCAST_ADDRESS=cassandra-cluster-3"
    - "CASSANDRA_SEEDS=cassandra-cluster-1"
    - "constraint:node==slave3"
  depends_on:
    - cassandra-cluster-2

cassandra-cluster-4:
  image: bde2020/cassandra
  container_name: cassandra-cluster-4
  restart: always
  environment:
    - "CASSANDRA_BROADCAST_ADDRESS=cassandra-cluster-4"
    - "CASSANDRA_SEEDS=cassandra-cluster-1"
    - "constraint:node==slave4"
  depends_on:
    - cassandra-cluster-3

3.12 Hadoop
To deploy a Hadoop cluster with the namenode running on master and the data nodes 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 -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 -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 -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 -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 -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:
    - /srv/hadoop/namenode:/hadoop/dfs/name
  environment:
```

```
- CLUSTER_NAME=bde_cluster
- "constraint:node==master"

env_file:
- ./.hadoop.env

datanode1:
  image: bde2020/hadoop-datanode:1.0.0
  hostname: datanode1
domainname: hadoop
container_name: datanode1

volumes:
- /srv/hadoop/datanode:/hadoop/dfs/data

env_file:
- ./.hadoop.env

environment:
- "constraint:node==slave1"
depends_on:
- namenode

datanode2:
  image: bde2020/hadoop-datanode:1.0.0
  hostname: datanode2
domainname: hadoop
container_name: datanode2

volumes:
- /srv/hadoop/datanode:/hadoop/dfs/data

env_file:
- ./.hadoop.env

environment:
- "constraint:node==slave2"
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"
  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"
  depends_on:
  - namenode
4. Conclusion

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.

In the table below we present records taken during the test deployments (command count to deploy the platform and time to deploy) with three subjects, two of them not working for BDE and not familiar with the platform:

1. G. Stavrinos: Expert Linux user; part of the NCSR-D BDE team; also subject for earlier BDE installation experiments.
2. G. Siantikos: Expert Linux user; not part of the NCSR-D BDE team; research and development background in signal analysis; has never used or installed any Apache Big Data tools previously.
3. D. Sgouropoulos: Expert Linux user; not part of the NCSR-D BDE team; research and development background in machine learning; has never used or installed any Apache Big Data tools previously.

The first subject (G. Stavrinos) issued more commands and took more time to deploy the platform because he was the first to test it and provided comments and corrections on the procedure. The instructions were then updated and the two last subjects completed the guide with less commands and in less time. Deployment time may vary depending on network speed.

Deployment Records

<table>
<thead>
<tr>
<th>Subject</th>
<th>Command count</th>
<th>Time to deploy</th>
</tr>
</thead>
<tbody>
<tr>
<td>G. Stavrinos</td>
<td>60</td>
<td>1h55m</td>
</tr>
<tr>
<td>G. Siantikos</td>
<td>56</td>
<td>1h22m</td>
</tr>
<tr>
<td>D. Sgouropoulos</td>
<td>56</td>
<td>1h26m</td>
</tr>
</tbody>
</table>