Virtual Core for Data Science Documentation

Release 0.1

Tiago Antao

Jul 01, 2018

Contents

Warning: virtual-core has been superseded by Skates!

Virtual Core is a turn-key solution to deploy a complete set of data-science core services that can be used as a base infrastructure for big data analysis. It was developed to support bioinformatics data analysis at the University of Montana.

The solution is based on Docker and it currently includes:

- A LDAP container, along with a web interface (phpldapadmin)
- Zabbix-based monitoring.
- PostgreSQL database server.
- An extensible software container, currently including Python and R tools for data-science (based on Anaconda)
- A user container, where users can log-in and run all data-science applications
- A file server, capable of routing data from other file servers (e.g. communicate with a Samba server and exposing an NFS interface)
- Cluster software (currently only SLURM)
- A SLURM-compute container that can be deployed across a cluster (via Docker swarm)
- A exploratory analysis server, which includes a Jupyter hub
- Plain web server

Most communications are SSL secured and the system can work as a adhoc SSL certification authority.

The containers can be split across a cluster with Docker Swarm.

A wizard is included to get a single-machine configuration up and running.

The system can be extended with flavors (Natural Language Processing, Finance, costumer analysis).

We currently have a flavor for bioinformatics with a Galaxy container, extra analysis software (based on bioconda) and the file router can be configured to receive data from Illumina Sequencers.

Warning: Virtual Core is currently in production an the University of Montana, but its installation by others is still quite hard. We are working hard to produce documentation, but that is still under heavy development (as you can see here). Some functionality is still being finalized. That being said, you are most welcome to try this (pre-alpha quality) software. If you need any help, do not hesitate to contact me.

CHAPTER 1

Contents

1.1 Installation

Here you can find a set of containers to help creating a data science core architecture, for example:

- LDAP server
- PostgreSQL server
- File router (NFS and Samba)
- Galaxy server
- Zabbix server
- Software server (i.e. a lot of pre-installed software)
- Interative Compute server (a place for users to login)
- Exploratory Analysis server (JupyterHub with JupyterLab)
- SLURM grid configuration

There is a focus on bioinformatics, but the infrastructure can be used for other applications.

1.1.1 Base images

We use Alpine Linux for simple servers (very small footprint) and Ubuntu for larger images. It might happen that some containers are derived from Debian ones.

1.1.2 Dependencies

- Python 2.7 (for ansible) **and** 3.5+
- PyYAML
- Docker

- Ansible
- docker-py (python-docker on ubuntu)

Todo: Check Python version for ansible (conda…)

If you use the setup wizard (strongly recommended for a first install)

- Flask
- openssl and pyOpenSSL (if you need to generate keys)

(explain with conda)

1.1.3 Installation

Virtual Core **uses Docker swarm mode.** You can check the Swarm mode documentation, though we provide basic instructions:

```
docker swarm init
```

Virtual core requires a **local Docker registry**. Again you can find documentation on the Docker site, but you probably only need to do this:

```
docker run -d -p 5000:5000 --name registry registry:2
```

This will install all your servers on the local machine. If you have a very large big-iron machine, this might be what you want. If you have a cluster, this is still a reasonable starting point, though you will have some work to do, especially on the security front.

1. Make sure you have a docker with swarm mode installation running.
2. Use the wizard to configure the most complicated stuff: `./run_wizard.sh`. Typically this will be on http://127.0.0.1:7000/
3. Create an overlay virtual network called *virtual_core* `docker network create --driver overlay --attachable --subnet 172.18.0.0/24 virtual_core`. **Make sure this is configured everytime you start the system**. **Also make sure the subnet is OK for you**.
4. Create a directory that will store all your docker volumes. This might need to be very big. Lets call this your base directory.
5. `cp etc/hosts.sample etc/hosts` (you will want to edit this in the future)
6. Make sure all variables on `ansible/` are correctly defined, especially on the common role
7. `python src/create_directory_structure.py <base_directory>`
8. `cd _instance/ansible; ansible-playbook --ask-pass -i ../../etc/hosts main.yml`

1.1.4 Installation on a cluster

Swarm node joining

The registry shoud be on the head node.

CONTINUE

1.2 Notes on containers

1.2.1 Support containers

These are available in dockerhub. There are two different flavors: Alpine and Ubuntu-based.

All Virtual Core containers are based on either alpine-supervisor-sshd or tiagoantao/ubuntu-supervisor-sshd. These have the following features:

1. Processes are controlled by supervisord
2. Includes a zabbix monitor
3. A SSH deamon is installed * A authorized_key has to be supplied. Used for root access.
4. The entry point is */sbin/server_run.sh*. A file */sbin/prepare_magic.sh* can be installed and will be called before supervisor (e.g. to use passed environment variables to setup stuff)

Todo: Discuss shared volumes and the need of consistent mount points

Todo: Talk about docker swarm

1.2.2 LDAP service

Both containers are based on Alpine Linux. This will probably change in the future as Alpine Linux is not a good solution for most other services.

ldap

Attention: The container exposes only the SSL port, but a non-SSL port is available inside the docker network.

1.2.3 PostgreSQL service and container

The PostgreSQL database is configured by default to authenticate via LDAP, but some databases are trusted to other hosts, these are normally related to other containers. For example the zabbix database is controlled by the user zabbix which is trusted to login from the host zabbix. Note a few things:

1. This is a default behavior that you can change during the configuration stage
2. All databases necessary by other containers are created along with access rights **even for containers that are not going to be installed**. Note that the database schemas should be created by the respective containers.
3. Make sure you are happy with access rules, especially if you spread containers across multiple machines. In that case you have some tweaking to do.

1.2.4 Web service

uwsgi - same directory as static, extension cgi

traverse…

discuss architecture

1.2.5 File router

Please see the dedicated page

1.2.6 Software container

1.2.7 Compute nodes

We provide support for clusters via SLURM. You can be in one of three situations:

1. You already have a cluster infrastructure not based on SLURM. If that is your case we do not support this directly. But if your cluster is based on free software we would be most happy to try to support it, please do contact us!
2. You already have a SLURM based configuration. In this case you will have to make a few changes to the configuration. The wizard will help you here.
3. You have nothing installed. We will take care of setting up a complete SLURM solution.

1.2.8 User server

based on software container. It is also a SLURM head node (if desired).

1.2.9 Exploratory analysis (Jupyter Lab) server

based on user server

share the home dirs…

1.3 The file router container

The file router container serves two purposes:

- Export NFS volumes
- Interacting with Windows machines

1.3.1 Export NFS volumes

To export NFS volumes, configure the NFS exports file appropriately (see an example on copy/etc/exports.sample

1.3.2 Interacting with Windows machines

There are a few use cases here:

- Interacting with services that are on Windows (For example, Illumina sequencers)
- Exporting volumes to users

> **Warning:** Integration of LDAP and Windows authentication is nothing short of a mess. There are plenty of alternatives on how to do it, but unless you *really* have to do it, you might want to consider avoiding it. There is plenty of documentation on the web, and we will not

If you want users to be able to mount your volumes as samba shares and have integrated LDAP authentication, we offer a ad-hoc script */usr/bin/change_password.py* that does sync between samba and LDAP. It works this way:

1. The user already has an account on LDAP
2. You create an account for the user on Samba using `pdbedit -a ldap_uid`. Use `smbpasswd ldap_uid`, password will be `boot`
3. The user logs in, ASAP, on the file_router and uses `change_password.py boot` to sync the passwords
4. From now on the user can login on the file_router to change the password using `change_password.py`

   (indeed password change can only happen on the file_router or the passwords will be out of sync)

Yes, this is ugly, but LDAP/Samba/Windows AD integration is ugly.

Note that for ad-hoc users (for example, in the bioinformatics case, interacting with a Illumina sequencer) you can just mantain a separate account just on samba without LDAP sync.

1.4 Customizing your system

_instance directory

1.5 LDAP authentication based on the virtual core

1.5.1 Ubuntu

apt-get install libpam-ldap ldap-utls

ldap-utils is recommended, not required.

On the configuration you will need to supply your LDAP URL (with https) and your base DN. You probably will want to allow users to change their passwords (with impact on the LDAP server). The server is not authenticated.

This will change the PAM configuration on /etc/pam.d. It will also change /etc/ldap.conf

If you are using your own certificate authority, you will need to add the certificate of the authority, by changing the *TLS_CACERT* parameter on /etc/ldap/ldap.conf . Be careful with auto-reconfiguration

Finally do

pam-auth-update

And you should be done

https://wiki.debian.org/LDAP/PAM

(uid limitation - pam_filter)

1.5.2 CentOS 5

We do this manually

- Make sure openldap-clients and nss_ldap is installed
- Copy your CA certificate to /etc/openldap/cacerts
- Make sure /etc/ldap.conf has (among other things):

```
URI ldaps://PATH_TO_YOUR_LDAP_SERVER
BASE your_base
pam_password exop
ssl on
port 636
tls_cacertfile /etc/openldap/cacerts/cacert.pem
```

- Make sure /etc/openldap/ldap.conf has (among other things):

```
URI ldaps://PATH_TO_YOUR_LDAP_SERVER
BASE your_base
TLS_CACERT /etc/openldap/cacerts/cacert.pem
```

- Edit /etc/nsswitch.conf to include ldap (on password, group and shadow)
- Edit at least /etc/pam.d/system-auth and add in appropriate places:

```
auth        sufficient    pam_ldap.so use_first_pass
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
password    sufficient    pam_ldap.so use_authtok
session     optional      pam_ldap.so
```

- Restart nscd

1.5.3 CentOS 6

- Install openldap-clients, sssd, pam_ldap and nss-pam-ldapd
- Make sure sssd is running
- on /etc/nsswitch.con use sss instead of ldap
- Here is an example of a sssd.conf file:

```
[sssd]
domains = LDAP
services = nss
config_file_version = 2

[nss]
filter_groups = root
filter_users = root

[domain/LDAP]
enumerate=true
cache_credentials = TRUE

id_provider = ldap
auth_provider = ldap
```

(continues on next page)

(continued from previous page)

```
ldap_schema = rfc2307
chpass_provider = ldap

ldap_uri = YOU_SERVER
ldap_search_base = YOUR_BASE
```

- Copy your CA certificate to /etc/openldap/cacerts
- Make sure /etc/pam_ldap.conf has (among other things):

```
URI ldaps://PATH_TO_YOUR_LDAP_SERVER
BASE your_base
pam_password exop
ssl on
port 636
tls_cacertfile /etc/openldap/cacerts/cacert.pem
```

- Make sure /etc/openldap/ldap.conf has (among other things):

```
URI ldaps://PATH_TO_YOUR_LDAP_SERVER
BASE your_base
TLS_CACERT /etc/openldap/cacerts/cacert.pem
```

- Edit at least /etc/pam.d/system-auth and add in appropriate places:

```
auth        sufficient    pam_ldap.so use_first_pass
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
password    sufficient    pam_ldap.so use_authtok
session     optional      pam_ldap.so
```

1.5.4 CentOS 7

needs review

follow instructions for centos 5, caveats:

On CentOS install nss_ldap and nss-pam-ldapd

/etc/nslcd.conf - ldap server (instead of /etc/ldap.conf)

make sure nslcd is started

make sure /etc/pam.d/system-auth is the only file of interest (e.g. password-auth)

authconfig –enableldap –enableldapauth –ldapserver=ldap://ldap.YOUR-DOMAIN:389/
–ldapbasedn="BASE-DN" –enablecache –disablefingerprint –kickstart

https://wiki.centos.org/AdrianHall/CentralizedLDAPAuth

1.6 Use case: Bioinformatics

1.6.1 Connecting with a sequencer

Tested on Illumina MiSeq

Samba

Sequencer side

Put machine on a workgroup (default on the container is GENOMICS)

Create a user for login and add a password

1.6.2 Galaxy

1.6.3 Software flavor