Setting-up the Infrastructure for the PaNOSC Data Transfer pilot

Document Log

Issue	Data	Comment	Author
v1.0	20/04/2020	First version of the documentation	Giuseppe La Rocca/EGI Foundation
v2.0	14/05/2020	Final version	Giuseppe La Rocca/EGI Foundation Marco De Simone/CERIC-ERIC
v3.0	27/11/2020	Added new section to upgrade the OneZone and Provider services	Andrea Manzi/EGI Foundation

Terminology

The EGI glossary of terms is available at: https://wiki.egi.eu/wiki/Glossary.

Table of Contents

About this document

The present document is organized in four different sections:

- PART I: Provides guidelines on how to install and configure the OneZone service.
- PART II: Provides guidelines on how to install and configure the Oneprovider service.
- PART III: Outlines how to deploy a Kubernetes cluster with the EC3/IM.
- PART IV: Describes how to tailor the EGI Notebooks service to address the needs of the PaNOSC community.
- PART V: Described how to set-up a Binder instance for the project.

Background

There are three pilots use cases identified in the PaNOSC (WP6):

- **1.) Transfer of data to analysis services** provided by another organisation (e.g.: EGI). Typical scenario: an user wants to analyse datasets produced during one of the experiments in the EGI infrastructure using Jupyter notebooks service.

- **2.) Archiving use case**: Transfer data from a production RI to external cold storage. The volume of data is in the order of 300TB up to 10PB.
    - Initial data transfer tests between STFC and ILL are working.
        - FTS3 client and globus-url-copy

- **3.) User's data transfer**: an user wants to transfer datasets to his/her home PC. The PaNOSC community is interested in a solution which allows users to download GB to TB of datasets to their local computers. The solution has to be reliable and support different protocols (the http:// protocol is not reliable above a few GB). Some RIs of the PaNOSC community (e.g.: ESFR, DESY, EuXFEL, Diamond) are already familiar with the Globus online solution (free version), other RI facilities are planning to use this service in the near future. The volume of data involved in the data transfer is in the order of 100TB up to 10PB.

In this document we will focus on the technical requirements to set-up the pilot tested for supporting the first use case. For this pilot the following services/solutions will be used:

- EGI Notebooks service for supporting big data analytics,
- Onedata for federating data sets from the PaNOSC RIs, and
- UmbrellaID as Community proxy for the PaNOSC community.

The Onedata solution

Onedata is a global data management system, providing easy access to distributed storage resources, supporting a wide range of use cases from personal data management to data-intensive scientific computations.

Getting started

This section provides a quick overview of the basic concepts behind Onedata as well as basic download and usage instructions.

Onedata for users

This section explains in more depth how to access, manage and share data using the Onedata platform.

Onedata for system administrators

This section describes how to deploy Onedata services, including Onezone for creating storage provider federations, as well as Oneprovider services to join existing Onezone federations and provide storage resources to users.

Onedata system requirements

Onedata services have certain minimum system requirements, which should be considered before deployment.

OneZone

Requirements	Minimal	Optimal
No. of VMs	1	2 + 1 for every 5000 users
CPU	8 vCPU cores	16 vCPU core
RAM	32 GB	64 GB
Local disk	SATA	SSD
Local storage space	20GB	40GB
OS (Docker deployment)	Any Docker compatible	Any Docker compatible
OS (Package deployment)	Ubuntu (16.04), CentOS 7	Ubuntu (16.04), CentOS 7

Oneprovider

Requirements	Minimal	Optimal
No. of VMs	1	2 + 1 for every 5000 users
CPU	8 vCPU cores	16 vCPU core
RAM	32 GB	64 GB
Local disk	SATA	SSD
Local storage space	20GB + 8MB for each 1000 files	40GB + 8MB for each 1000 files
OS (Docker deployment)	Any Docker compatible	Any Docker compatible
OS (Package deployment)	Ubuntu (16.04), CentOS 7	Ubuntu (16.04), CentOS 7

Please note that these numbers are rough estimates and depend on actual data access patterns of users, network capacity and underlying storage performance.

PART I - Install and configure the OneZone service

To start to get familiar with the OneZone service, please refer to this documentation. In this documentation we are going to install an instance of the OneZone service on a VM running with the following technical requirements:

OS	Ubuntu (bionic-x86_64)
Flavor	standard.2core-16ram (2 vCPU cores, 16GB of RAM and 80GB of local disk)
Hostname	onezone-panosc.egi.eu An entry for this hostname was registered in the DNS

Pre-requisites

In order to ensure optimum performance of the **Onezone** service, several low-level settings need to be tuned on the host machine. This applies to both Docker based as well as package based installations, in particular to nodes where Couchbase database instances are deployed. **After these settings are modified, the machine needs to be rebooted**.

Setting up certificates

Since release 18.02.0-rc10, Onezone supports automatic certificate management backed by Let's Encrypt. To use this option, it is only necessary to enable this feature in Onezone Docker Compose configuration file (see above) or via GUI. If you prefer to obtain and install certificates for Onezone service manually, modify the Docker Compose file to mount PEM files inside the container using paths listed in TLS certificate management.

Firewall configuration in Onezone

Due to the fact that Onedata consists of several services which need to communicate between different sites, several ports need to be opened to the outside of the local network. Below is a detailed list of ports, which need to be opened and their designation in the Onezone service.

Port	Description
53/TCP	DNS (Required in Onezone)
53/UDP	DNS (Required in Onezone)
80/TCP	HTTP (Optional - automatically redirected to 443)
443/TCP	HTTPS, REST, Oneclient

Increase maximum number of opened files

In order to install **Onezone** service on one of the supported operating systems, first make sure that the maximum limit of opened files is sufficient (preferably 63536, but below `/proc/sys/fs/file-max`).

The limit can be checked using:

```
]$ ulimit -n
1024
```

If necessary, increase the limit using:

```
]$ sudo sh -c 'echo "* soft nofile 63536" >> /etc/security/limits.conf'
]$ sudo sh -c 'echo "* hard nofile 63536" >> /etc/security/limits.conf'
```

It might be also necessary to setup the limit in /etc/systemd/system.conf:

```
]$ sudo sh -c 'echo DefaultLimitNOFILE=65536 >> /etc/systemd/system.conf'
]$ sudo systemctl daemon-reexec
```

Disable systemd-resolved service

```
]$ sudo systemctl stop systemd-resolved
]$ sudo systemctl disable systemd-resolved
```

Swap preference settings

Make sure that the swap preference (i.e. *swappiness*) is set to `0` (or at most `1` - see here for details):

```
]$ cat /proc/sys/vm/swappiness
60
```

and if necessary decrease it using:

```
]$ sudo sh -c 'echo "vm.swappiness=0" >> /etc/sysctl.d/50-swappiness.conf'
```

Disable Transparent Huge Pages feature

By default, many Linux machines have the Transparent Huge Pages feature enabled, which improves apparent performance of machines running multiple applications at once, however it deteriorates the performance of most database-heavy applications, such as **Onezone**.

These settings can be checked using the following commands (here the output shown is the expected setting):

```
]$ cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
always madvise [never]

]$ cat /sys/kernel/mm/transparent_hugepage/defrag
always madvise [never]
```

If any of the settings is different than the above, they should be changed permanently, which can be achieved for instance by creating a simple **systemd** unit file `/etc/systemd/system/disable-thp.service`:

```
[Unit]
Description=Disable Transparent Huge Pages

[Service]
Type=oneshot
ExecStart=/bin/sh -c "/bin/echo 'never' |
/usr/bin/tee/sys/kernel/mm/transparent_hugepage/enabled"
ExecStart=/bin/sh -c "/bin/echo 'never' |
/usr/bin/tee/sys/kernel/mm/transparent_hugepage/defrag"

[Install]
WantedBy=multi-user.target
```

and enabling it on start using:

```
]$ sudo systemctl enable disable-thp.service
]$ sudo systemctl start disable-thp.service
```

Increase network performance

By default the Linux network stack is not configured for high speed large file transfer across WAN links. This is done to save memory resources. You can easily tune Linux network stack by increasing network buffers size for high-speed networks that connect server systems to handle more network packets.

Set the max OS send buffer size (wmem) and receive buffer size (rmem) to 16 MB for queues on all protocols. Add these settings in the `/etc/sysctl.conf` file:

```
]$ sysctl -w net.core.wmem_max=16777216
]$ sysctl -w net.core.rmem_max=16777216
```

Reboot the machine to make changes effective.

Setting the hostname

Make sure that the machine has a resolvable, domain-style hostname (it can be Fully Qualified Domain Name or just a proper entry in `/etc/hostname` and `/etc/hosts`) - for this tutorial it is set to `onezone-panosc.egi.eu`.

```
]$ cat /etc/hosts
127.0.0.1 onezone-panosc

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

```
]$ cat /etc/hostname
onezone-panosc
```

Following command examples assumes an environment variable `ONEZONE_HOST` is available, for instance:

```
]$ export ONEZONE_HOST="onezone-panosc.egi.eu"
```

Python

Make sure that python 2.x is installed on the machine.

```
]$ python -V
Python 2.7.15+
```

Docker based setup

Onezone installation using Docker is very straightforward. This type of deployment requires that docker and docker-compose have been installed on your server.

Install Docker

```
]$ sudo apt install -y apt-transport-https \
                       ca-certificates curl software-properties-common

]$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
]$ sudo add-apt-repository "deb [arch=amd64]
```

```
https://download.docker.com/linux/ubuntu bionic stable"
]$ sudo apt update
]$ sudo apt install -y docker-ce
]$ sudo usermod -aG docker ${USER}
```

Install Docker Compose

```
]$ sudo curl -L
https://github.com/docker/compose/releases/download/1.21.2/docker-compose-
`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
]$ sudo chmod +x /usr/local/bin/docker-compose
```

DNS records setup for subdomain delegation

Onezone has its own DNS server, automatically deployed on every node of the cluster. Its purpose is to simplify cluster scaling and allow for subdomain delegation for Oneproviders - i.e. allocating subdomains of the Onezone domain for the providers and resolving DNS queries in their name. In order for the subdomain delegation to work properly, it is necessary to set-up a DNS zone delegation in the DNS server responsible for your domain. It should delegate management of the Onezone domain and its subdomains to the Onezone DNS server.

Assuming your Onezone domain is `onezone-panosc.egi.eu.com` you need to set the following records at your DNS provider: NS records pointing to `ns1.onezone.org` and `ns2.onezone.org`, etc. Number of those subdomains depends on the number of nodes in your Onezone cluster. If there are more than 10 nodes, only the first ten should be inserted.

In the DNS responsible for the `onezone-panosc.egi.eu` domain (usually the server is administered by the domain provider, or there is a dedicated DNS server for your organization), set the following records:

Domain	Record	Value
onezone-panosc.egi.eu.	NS	ns1.onezone-example.com.
onezone-panosc.egi.eu.	NS	ns2.onezone-example.com.
onezone-panosc.egi.eu.	NS	ns3.onezone-example.com.
ns1.onezone-example.com.	A	150.1.0.2
ns2.onezone-example.com.	A	150.1.0.3
ns3.onezone-example.com.	A	150.1.0.4

This way, all queries concerning the `onezone-panosc.egi.eu` domain will be routed to the DNS servers running on Onezone cluster nodes.

Customizing Onezone Docker Compose script

In case of Docker based deployment all configuration information needed to install **Onezone** can be included directly in the Docker Compose script. This tutorial assumes that all **Onezone** configuration and log files will be stored in the folder `/opt/onedata/onezone` on the host machine, but you can use any directory to which Docker has access to. Make sure the partition where the `/opt` directory is mounted has at least 20GB of free space for logs and database files.

The following assumes you have prepared the following directory structure:

```
]$ sudo mkdir -p /opt/onedata/onezone
]$ sudo mkdir /opt/onedata/onezone/certs
]$ sudo mkdir /opt/onedata/onezone/persistence
```

Create the following Docker Compose file in `/opt/onedata/onezone/`

```
]$ sudo cat /opt/onedata/onezone/docker-compose.yml
version: '2'

networks:
  default:
    driver: bridge
    driver_opts:
      com.docker.network.driver.mtu: 1442

services:
  node1.onezone:
    restart: always
    image: onedata/onezone:19.02.1
    hostname: node1.onezone.local
    # dns: 8.8.8.8 # uncomment if container can't ping any domain
    container_name: onezone
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
      # Onezone persistence
      - "./data/persistence:/volumes/persistence"
      # Onezone config
      - "./data/configs/overlay.config:/etc/oz_worker/overlay.config"
      # OpenID & SAML config
```

```
      - "./data/secret/auth.config:/etc/oz_worker/auth.config"
      # Custom images used on the login page
      - "./data/images/umbrella.png:/var/www/html/oz_worker/custom/umbrella.png"
      - "./data/images/xdc.png:/var/www/html/oz_worker/custom/xdc.png"
      - "./data/images/deep.png:/var/www/html/oz_worker/custom/deep.png"
    ports:
      - "53:53"
      - "53:53/udp"
      - "80:80"
      - "443:443"
      - "9443:9443"
    environment:
     #ONEPANEL_DEBUG_MODE: "true" # prevents container exit on configuration error
     ONEPANEL_BATCH_MODE: "true"
      ONEPANEL_LOG_LEVEL: "none" # prints logs to stdout (possible values: none,
debug, info, error), by default set to info
      ONEPANEL_EMERGENCY_PASSPHRASE: ${EMERGENCY_PASSPHRASE}
       # Used for test purposes, disable when you have your own web cert mounted
via volumes.
      # Cert will be generated only if none is found under the cert path.
      # If enabled, a new web cert will be generated with CN matching the
      #   ONEPANEL_GENERATED_CERT_DOMAIN and signed by OnedataTestWebServerCa
      # NOTE: The generation will be performed upon every startup, any
      #   existing certs will be backed up and placed in the same directory.
      # WARNING: This feature is devised for test purposes and must not
      #   be used in production.
      ONEPANEL_GENERATE_TEST_WEB_CERT: "false"  # default: false
      # The generated test web cert will be issued for below domain.
      ONEPANEL_GENERATED_CERT_DOMAIN: ""  # default: ""
      # If enabled, onepanel will trust any server that has a cert signed by
      #   the OnedataTestWebServerCa.
      # WARNING: This feature is devised for test purposes and must not
      #   be used in production.
      ONEPANEL_TRUST_TEST_CA: "false"  # default: false

      ONEZONE_CONFIG: |
        cluster:
          domainName: "onezone-panosc.egi.eu"
          nodes:
           n1:
```

```
            hostname: "node1"
        managers:
          mainNode: "n1"
          nodes:
            - "n1"
        workers:
          nodes:
            - "n1"
        databases:
          nodes:
            - "n1"
      onezone:
        name: "PaNOSC"
        domainName: "onezone-panosc.egi.eu"
        letsEncryptEnabled: true
```

To enable federated authentication mechanism and local access, update the `/opt/onedata/onezone/data/secret/auth.config` file with the settings:

```
#{
    version => 3,
        % Allows to log in to Onezone using username & password.
        basicAuthConfig => #{
        enabled => true
    },

    openidConfig => #{
        % Enable OpenID login protocol - if disabled, all OpenID IdPs will be
        % hidden from the login page.
        enabled => true,
        [..]
    },
    supportedIdps => [
        {basicAuth, #{
            % Configuration of the login page button
            displayName => "username & password",
            % Some predefined icons are built-in into the GUI app.
            % Their paths start with /assets/images/auth-providers.
            % For a custom icon, put it in:
            %   /var/www/html/oz_worker/custom/<path>
            % And reference it here like this: /custom/<path>
```

```
            iconPath => "/assets/images/auth-providers/basicauth.svg",
            % Background color is useful for icons with transparency. Moreover,
            % after selecting an IdP, its icon disappears and a spinner is
            % shown - hence background color should be kept the same as icon's
            % background color (if any).
            iconBackgroundColor => "#4BD187",
           % Which protocol is used for this IdP - basicAuth is the only
            % valid one for basicAuth IdP.
            protocol => basicAuth
        }},
```

In this configuration the OpenID setting has been used to enable UmbrellaID as community proxy service for the PaNOSC community. Additional AAI providers can also be included in the configuration if needed.

```
{UmbrellaID, #{
   displayName => "UmbrellaID",
   iconPath => "/custom/umbrella.png",
   iconBackgroundColor => "#0455A1",
   protocol => openid,
   protocolConfig => #{
   plugin => default_oidc_plugin,
   pluginConfig => #{
       clientId => "XXX-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
       clientSecret => "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
       endpoints => #{
       xrds => "https://proxy.umbrellaid.org/.well-known/openid-configuration",
       authorize => {xrds, "authorization_endpoint"},
       accessToken => {xrds, "token_endpoint"},
       userInfo => {xrds, "userinfo_endpoint"}
       },
       scope => "openid email profile",
       accessTokenAcquireMethod => post,
       clientSecretPassMethod => urlencoded,
       accessTokenPassMethod => urlencoded
    }
    }
}}
```

```
]
}.
```

Custom icon guidelines

To use your custom icon on the login page, place it on the Onezone host under the path `/var/www/html/oz_worker/custom/<path>` and reference it in the config like this: `iconPath => "/custom/<path>"`. If you are using docker-compose, simply mount your icon by adding a volume, for example:

```
]$ cat docker-compose.yaml
[..]
  volumes:
   - "/path/to/your/icon.svg:/var/www/html/oz_worker/custom/my-icon.svg"
```

To obtain the best visual effects, please check the online documentation available here.

Install Docker images

To install the necessary Docker images on the machine run:

```
]$ cd /opt/onedata/onezone
]$ docker-compose -f /opt/onedata/onezone/docker-compose.yml pull
```

Start OneZone

```
]$ cd /opt/onedata/onezone
]$ ./onezone.sh start
Pulling node1.onezone (docker.onedata.org/onezone:19.02.1)...
[..]
Creating network "onezone_default" with the default driver
Creating onezone ... done
```

Checking Logs

```
]$ ./onezone.sh logs
Copying missing persistent files...
Done.
Starting oz_panel...
[  OK  ] oz_panel started
```

```
Waiting for the existing cluster to start...
Existing Onezone deployment resumed work.

Container details:
* IP Address: 172.18.0.2
* Ports: -

]$ ./onezone panel-logs
[..]
"Node 'cluster_manager@node1.onezone.local' not responding to pings."
[E 2019-11-27 15:13:41.071 <0.521.0>] Command "service couchbase-server status" exited with code 3 and output
" * couchbase-server is not running"
[..]
[I 2019-11-27 15:14:28.047 <0.864.0>] Setting up Onezone panel service in Onezone
[I 2019-11-27 15:14:28.210 <0.864.0>] Deployed static GUI files (7c7c3aa91c703bf38397f42fd51f2b0a73d992160be2c92e65a2d24546900a9a)
[I 2019-11-27 15:14:28.220 <0.864.0>] Onezone panel service successfully set up in Onezone
```

Configure the volume space in OneZone

Open `https://$ONEZONE_HOST:9443` using any web browser and continue through the steps described here. After this step succeeds, Onezone should be ready and accessible at `https://$ONEZONE_HOST`.

Upgrade the OneZone

In order to upgrade the OneZone service to a new version (either a new bugfix release or a new minor or major) the following steps are needed:

1. Stop the service by running the command:

   ```
   ]$ cd /opt/onedata/onezone
   ]$ ./onezone.sh stop
   Stopping onezone ... done
   Removing onezone ... done
   Removing network onezone_default
   ```

2. Backup the persistent folder (`opt/onedata/onezone/data/persistence`) in case a rollback is needed
3. Update the image to upgrade to in the file `/opt/onedata/onezone/docker-compose.yml` (image parameter)

4. Restart the service by running the command

```
]$ cd /opt/onedata/onezone
]$ ./onezone.sh start
   Creating network "onezone_default" with driver "bridge"
   Pulling node1.onezone (onedata/onezone:20.02.3)...
   20.02.3: Pulling from onedata/onezone
   …
   Status: Downloaded newer image for onedata/onezone:20.02.3
   Creating onezone ... done
```

PART II - Install and configure the Oneprovider services

To start to get familiar with the Oneprovider service, please refer to this documentation. In this documentation we are going to install Oneprovider service at CESNET-MCC and two Oneprovider services at CERIC-ERIC on VMs having the following technical requirements:

OS	Ubuntu (bionic-x86_64)
Flavor	hpc.16cores-32ram (16 vCPU cores, 32GB of RAM and 80GB of local disk)
Storage	An additional block storage of 800GB is mounted in /mnt
Hostname	oneprovider-pn-cesnet
Location	CESNET

OS	Ubuntu (bionic-x86_64)
Flavor	hpc.16cores-32ram (16 vCPU cores, 32GB of RAM and 120GB of local virtio disk)
Storage	An additional NFS volume storage of 1024GB is mounted in /mnt
Hostname	oneprovider-pn-ceric
Location	CERIC-ERIC

OS	Ubuntu (bionic-x86_64)
Flavor	hpc.16cores-32ram (16 vCPU cores, 32GB of RAM and 120GB of local virtio disk)
Storage	Only local, 120gb virtio disk
Hostname	ceric-op-cache
Location	CERIC-ERIC

Pre-requisites

Oneprovider machines (if the deployment is dockerized - dockers too) require the following parameters set for network buffers:

```
sysctl -w net.core.wmem_max=16777216
sysctl -w net.core.rmem_max=16777216
```

Setting up certificates

Since release 18.02.0-beta5, Oneprovider supports automatic certificate management backed by Let's Encrypt. To use this option, it is only necessary to enable this feature in Oneprovider Docker Compose configuration file (see above) or via GUI.

If you prefer to obtain and install certificates for Oneprovider service manually, modify the Docker Compose file to mount PEM files inside the container using paths listed in TLS certificate management.

Firewall configuration in Oneprovider

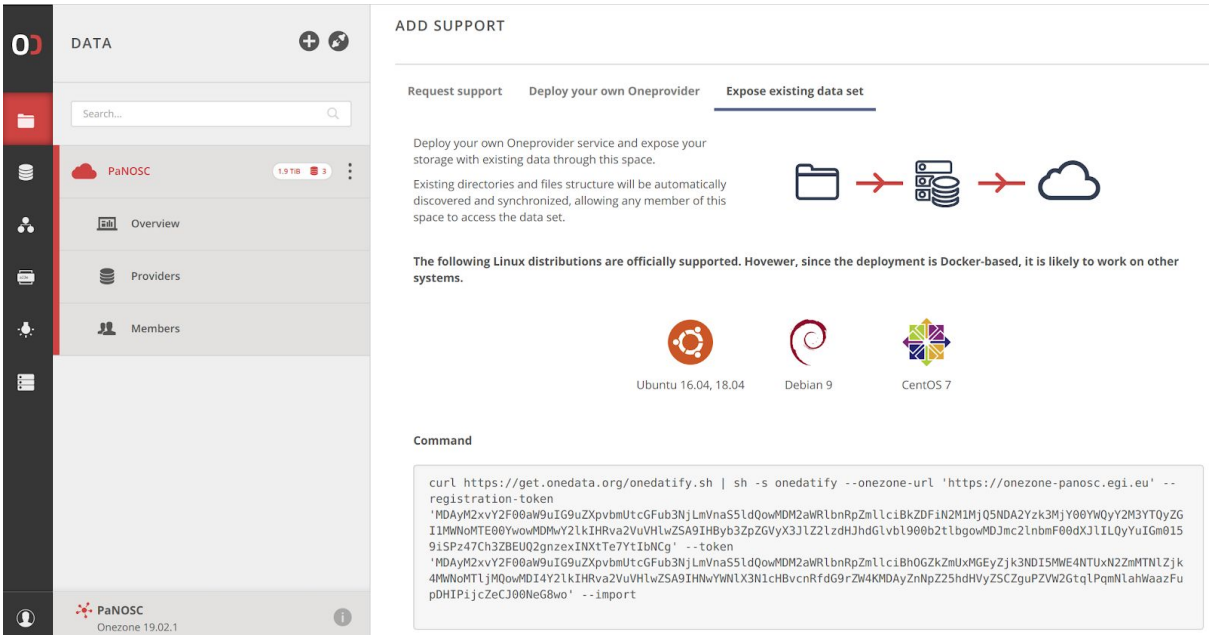
Oneprovider requires the following ports to be opened:

Port	Description
80/TCP	HTTP (Optional - automatically redirected to 443)
443/TCP	HTTPS, REST, Oneclient
6665/TCP	Onedata data transfer channel (Oneprovider RTransfer)
9443/TCP	Onepanel web interface (can be limited to local network)

Deploy the Oneprovider server

The easiest way to deploy Oneprovider is using Onedatify. To generate the online Onedatify command in Onezone:

- Go to Onezone interface,
- Select `Add support...` option under the space name, and
- Select the tab `Expose existing data set`:



CESNET Oneprovider deployment via onedatify

Run the command on the target node:

```
]$ curl https://get.onedata.org/onedatify.sh | sh -s onedatify \
--onezone-url 'https://onezone-panosc.egi.eu'\
--registration-token 'MDAxNWxvY2F00[..]8YKDdNCpy8Hd9bOrUF' \
--token 'MDAxNWxvY2F00[..]8YKDdNCpy8Hd9bOrUF' --import
[..]

Downloaded Onedatify installation script /tmp/onedatify_19.02.1.rc2.sh
The Onedatify script allows you to easily install Oneprovider service and connect
to an existing data space.
This script will perform following changes on this machine:
- install the latest versions of Docker and Docker Compose
- change machine 'ulimit' to the value found in /proc/sys/fs/file-max
- disable swappiness and Transparent Huge Pages feature
- open all ports required by Oneprovider
Oneprovider requires valid SSL certificates
to register with Onezone, you can provide your own certificates
or Onedatify script will help you generate Let's Encrypt certificates.
In order to install Onedatify service prepare the following items:
- a fully qualified domain name (FQDN) of this machine
- an operating system that supports systemd
- the following ports need to be open:
    80/TCP      HTTP
    443/TCP     HTTPS
    6665/TCP    Onedata data transfer channel (RTransfer)
    9443/TCP    Onepanel web interface - needed only for system administrators

If you want to expose existing data via Onedata,
```

```
the storage with the data must be accessible for this machine through a POSIX path
(e.g. NFS mount point).
If you expect this provider to transfer data to other providers, make sure that
this VM has a direct connection (public IP or dedicated routing), to other
providers you want to transfer data to.

Are you ready to proceed with the installation (y|n)?: y
[..]
########### Installing Onedatify Package ###########
Onedatify installation process started…
Docker installation found.
Docker-compose binary was detected on the machine.
Installing onedatify package:
onedatify       19.02.1.rc2.systemd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  Grub-pc-bin
Use 'sudo apt autoremove' to remove it.
The following packages will be REMOVED:
  Onedatify
The following held packages will be changed:
  Onedatify
0 upgraded, 0 newly installed, 1 to remove and 99 not upgraded.
[..]
(Reading database ... 90871 files and directories currently installed.)
Preparing to unpack .../onedatify_19.02.1.rc2.systemd_amd64.tGHcNNO.deb …
Unpacking onedatify (19.02.1.rc2.systemd) over (19.02.1.rc2.systemd) …
Setting up onedatify (19.02.1.rc2.systemd) …
onedatify set on hold.
########################################
Package installation successful.
onedata/oneprovider:19.02.1-rc2

########################################
version=onedata/oneprovider:19.02.1-rc2
zone_fqdn=onezone-panosc.egi.eu
admin_password=******************
admin_email=giuseppe.larocca@egi.eu
name=cesnet
fqdn=cesnet-op.onezone-panosc.egi.eu
persistence_volume=/opt/onedata/onedatify/oneprovider_conf
overlay_panel_volume=/opt/onedata/onedatify/op-panel-overlay.config
overlay_worker_volume=/opt/onedata/onedatify/op-worker-overlay.config
subdomain=cesnet-op
domain_name=onezone-panosc.egi.eu
onezone_registration_token=MDAyM2xvY[..]rhBfIQWt00vt1qeq8YKDdNCpy8Hd9bOrUFLvFQCg
subdomain_delegation=true
lets_encrypt_enabled=true
```

```
key=
cert=
cacert=
extra_cacerts=
geo_latitude=50.10496139526367
geo_longitude=14.382089614868164
########################################

You will now be asked to provide configuration parameters:
########## Basic parameters ############
Please enter the preety name of your oneprovider (default: cesnet):
Enter a new administration password for the Oneprovider or the password will be
auto generated:
Your new password to login into Oneprovider is: ******************
Please enter the email address that will be used as the emergency contact for this
provider (default: giuseppe.larocca@egi.eu): <ENTER>
Absolute path where to keep oneprovider configuration and metadata (default:
/opt/onedata/onedatify/oneprovider_conf): <ENTER>
Please enter the latitude of your provider (auto-detected default:
50.10496139526367): <ENTER>
Please enter the longitude of your provider (auto-detected default:
14.382089614868164): <ENTER>

############ Certificates ##############
You can choose to provide your own certificates or use subdomain delegation and
request Onezone to help in generating the Let's Encrypt certificate for your
subdomain.
Do you want to use a subdomain delegation to acquire a FQDN for your Oneprovider?
(y/n, default: y)?: y
Please enter the subdomain for your oneprovider (auto-detected default:
oneprovider-pn): cesnet-op
The FQDN of your Oneprovider is: cesnet-op.onezone-panosc.egi.eu
Do you want Onezone to create a Let's Encrypt certificate for Oneprovider (y/n,
default: y)? y
INFO: Let's Encrypt allows you to create only 5 duplicate certificates per week!
Use this option with care!

############ Storage Configuration ##############
What storage type would you like to use? (default: posix, possible options: posix
| s3 | ceph | glusterfs | swift ): posix
An absolute path to a directory you want to expose trough Onedata from POSIX
storage (default: /tmp): /mnt
Please provide the url to the LUMA server, or leave it blank:
Expose storage as read only? (y/n, default: n): n
Created symlink /etc/systemd/system/multi-user.target.wants/onedatify.service →
/usr/lib/systemd/system/onedatify.service.

######## Starting Oneprovider ########
Installation is about to start, you can observe detailed logs by executing
following command in a second window:
onedatify logs
```

```
Waiting for oneprovider to start.........................
Congratulations! Oneprovider has started successfully.
The size of space support (you can use B, KiB, MiB, up to EiB, 1KiB = 1024B, eg.
1GiB) (auto-detected: 786.4477GiB): <ENTER>

Supporting a space with ****** [B] of storage onedatify on oneprovider
oneprovider-pn.onezone-panosc.egi.eu
Checking Oneprovider reddiness..

Your storage was successfully supported and exposed.
Depending on the amount of exposed data it might take some time to scan it. Please
be patient.

To configure your Oneprovider please visit:
https://oneprovider-pn.onezone-panosc.egi.eu:9443

In order to access your data please visit:
https://onezone-panosc.egi.eu

Generated password to access the Onepanel emergency account is: **************
```

CERIC-ERIC Oneprovider deployment

Oneprovider-pn-ceric

```
]$ curl https://get.onedata.org/onedatify.sh | sh -s onedatify --onezone-url
'https://onezone-panosc.egi.eu'                          --registration-token
'MDAyM2xvY2F[..]02LvhLtaEykE2njnED1015Cg' --token 'MDAyM2xvY2F00aW[..]' --import

[..]
In order to install Onedatify service prepare the following items:
- a fully qualified domain name (FQDN) of this machine
- an operating system that supports systemd
- the following ports need to be open:
    80/TCP      HTTP
    443/TCP     HTTPS
    6665/TCP    Onedata data transfer channel (RTransfer)
    9443/TCP    Onepanel web interface - needed only for system administrators

If you want to expose existing data via Onedata, the storage with the data must be
accessible for this machine through a POSIX path (e.g. NFS mount point).
```

If you expect this provider to transfer data to other providers,
make sure that this VM has a direct connection (public IP or dedicated routing),
to other providers you want to transfer data to.

Are you ready to proceed with the installation (y|n)?: y
[..]

########### Installing Onedatify Package ###########
Onedatify installation process started…
Docker installation found.
Docker-compose binary was detected on the machine.
Installing onedatify package:
onedatify   19.02.1.systemd
[...]
Preparing to unpack .../onedatify_19.02.1.systemd_amd64.bRjByY0.deb …
Unpacking onedatify (19.02.1.systemd) over (19.02.1.systemd) …
Setting up onedatify (19.02.1.systemd) …
onedatify set on hold.
##

Package installation successful.
onedata/oneprovider:19.02.1
[...]

##

You will now be asked to provide configuration parameters:
########## Basic parameters ###########

Please enter the preety name of your oneprovider (default: ceric): **ceric**
Enter a new administration password for the Oneprovider or the password will be auto generated:
Your new password to login into Oneprovider is: ********************

Please enter the email address that will be used as the emergency contact for this provider : **marco.desimone@elettra.eu**

Absolute path where to keep oneprovider configuration and metadata (default: /opt/onedata/onedatify/oneprovider_conf):

Please enter the latitude of your provider (auto-detected default: 45.662498474121094):
Please enter the longitude of your provider (auto-detected default: 13.797300338745117):

########### Certificates #############
You can choose to provide your own certificates or use subdomain delegation and request Onezone to help in generating the Let's Encrypt certificate for your subdomain.

Do you want to use a subdomain delegation to acquire a FQDN for your Oneprovider? (y/n, default: y)?: **y**

Please enter the subdomain for your oneprovider (auto-detected default: ceric-op):
The FQDN of your Oneprovider is: **ceric-op.onezone-panosc.egi.eu**

########### Storage Configuration #############

What storage type would you like to use? (default: posix, possible options: posix | s3 | ceph | glusterfs | swift): **posix**

An absolute path to a directory you want use to support your Onedata POSIX storage (default: /tmp): **/mnt/ceric-datasets**

Created symlink /etc/systemd/system/multi-user.target.wants/onedatify.service → /usr/lib/systemd/system/onedatify.service.

####### Starting Oneprovider ########

Installation is about to start, you can observe detailed logs by executing following command in a second window: onedatify logs

Waiting for oneprovider to start.....................
Congratulations! Oneprovider has started successfully.

The size of space support (you can use B, KiB, MiB, up to EiB, 1KiB = 1024B, eg. 1GiB) (auto-detected: 1023.1768GiB):

Supporting a space with 1098627678208 [B] of storage onedatify on oneprovider ceric-op.onezone-panosc.egi.eu

Checking Oneprovider reddiness..

Your storage was successfully supported and exposed.
Depending on the amount of exposed data it might take some time to scan it. Please be patient.

To configure your Oneprovider please visit:
https://ceric-op.onezone-panosc.egi.eu:9443

In order to access your data please visit:
https://onezone-panosc.egi.eu

Generated password to access the Onepanel emergency account is: ************

CERIC op-cache

]$ curl https://get.onedata.org/onedatify.sh | sh -s onedatify --onezone-url 'https://onezone-panosc.egi.eu' --registration-token 'MDAyM2xvY2F00aW[..] 0SKcvuXXlQMGNKJWIGTaZ9bZCg' --token 'MDAyM2xvY2F00aW9uIG9uZ[..]b31xX2jUT8DsCyYoQo' [...]
########## Basic parameters ############
Please enter the preety name of your oneprovider (default: ceric-op-cache): ceric-cache
Enter a new administration password for the Oneprovider or the password will be auto generated:
Your new password to login into Oneprovider is: ****************
Please enter the email address that will be used as the emergency contact for this provider (default: marco.desimone@elettra.eu): **<ENTER>**
Absolute path where to keep oneprovider configuration and metadata (default: /opt/onedata/onedatify/oneprovider_conf): **<ENTER>**
Please enter the latitude of your provider (auto-detected default: 45.662498474121094): **<ENTER>**
Please enter the longitude of your provider (auto-detected default: 13.797300338745117): **<ENTER>**

########### Certificates #############

You can choose to provide your own certificates or use subdomain delegation and request Onezone to help in generating the Let's Encrypt certificate for your subdomain.
Do you want to use a subdomain delegation to acquire a FQDN for your Oneprovider? (y/n, default: y)?: **y**
Please enter the subdomain for your oneprovider (auto-detected default: oneprovider-pn-ceric-cache): **ceric-cache**
The FQDN of your Oneprovider is: ceric-cache.onezone-panosc.egi.eu

########## Storage Configuration #############
What storage type would you like to use? (default: posix, possible options: posix | s3 | ceph | glusterfs | swift): **posix**
An absolute path to a directory you want to expose trough Onedata from POSIX storage (default: /tmp): **/cache**
Please provide the url to the LUMA server, or leave it blank: **<ENTER>**
Expose storage as read only? (y/n, default: n): **n**
Created symlink /etc/systemd/system/multi-user.target.wants/onedatify.service → /usr/lib/systemd/system/onedatify.service.

####### Starting Oneprovider ########
Installation is about to start, you can observe detailed logs by executing following command in a second window:
onedatify logs

Waiting for oneprovider to start.....................
Congratulations! Oneprovider has started successfully.

The size of space support (you can use B, KiB, MiB, up to EiB, 1KiB = 1024B, eg. 1GiB) (auto-detected: 117.6148GiB): **<ENTER>**
Do you want to regularly scan storage for any changes in files? (y/n, default: y): **y**

Supporting a space with 126287937536 [B] of storage onedatify on oneprovider ceric-cache.onezone-panosc.egi.eu
Checking Oneprovider reddiness..

Your storage was successfully supported and exposed.
Depending on the amount of exposed data it might take some time to scan it. Please be patient.

```
To configure your Oneprovider please visit:
https://ceric-cache.onezone-panosc.egi.eu:9443

In order to access your data please visit:
https://onezone-panosc.egi.eu

Generated password to access the Onepanel emergency account is:
**********
```

CERIC-ERIC NOTES

There's a small bug in the onedatify script, it does not accept domain name with a dash symbol (ceric-eric.eu) so I had to use elettra.eu domain and then fix the web gui once the oneprovider was online, the issue has been submitted to onedata developers.

The two CERIC-ERIC oneproviders are located in the Elettra's DMZ lan, they have internal IP addresses and there is a reverse 1to1 public NAT for both, it is managed by a sonic-wall firewall, their FQDN is associated with their public IP address:

- `ceric-op.onezone-panosc.egi.eu     -> 140.105.207.31 <-> 172.19.31.6`
- `ceric-cache.onezone-panosc.egi.eu -> 140.105.207.32 <-> 172.19.31.7`

Since they cannot use the public ip address to connect to each other and also to improve the network performance, the /etc/hosts files on both machine have been modified as follow, in order to force to use the internal lan (25Gbit):

```
127.0.0.1      localhost
172.19.31.6    ceric-op.onezone-panosc.egi.eu
172.19.31.6    oneprovider-pn.ceric-eric.eu    oneprovider-pn
172.19.31.7    ceric-cache.onezone-panosc.egi.eu
172.19.31.7    op-cache-01.ceric-eric.eu       op-cache-01 ceric-cache
```

Upgrade the Oneprovider server

In order to upgrade the OneProvider service to a new version (either a new bugfix release or a new minor or major) the following steps are needed:

1. Stop the service by running the command:

   ```
   ]$ sudo systemctl stop onedatify.service
   ```

2. Backup the persistent folder (/opt/onedata/onedatify/oneprovider_conf/) in case a rollback is needed
3. Update the image to upgrade to in the file `/opt/onedata/onedatify/config` (version parameter)
4. Restart the service by running the command

```
]$ sudo systemctl start onedatify.service
```

Configure LUMA

TODO

PART III - Create a K8s cluster with EC3

In this section it will be described how a Kubernetes (K8s) virtual cluster composed by 1 master node and 2 workers is configured to spawn Jupyter notebooks on demand for users. For the configuration of the K8s cluster it will be used the Elastic Cloud Compute Cluster (EC3) framework developed by the Polytechnic University of Valencia (UPV).

Configure the environment

EC3 has an official Docker container image available in Docker Hub that can be used instead of installing the CLI. You can download it by typing:

```
]$ sudo docker pull grycap/ec3
```

Listing available EC3 templates

To list the available templates, use the command:

```
]$ cd $HOME
]$ sudo docker run -v /tmp/.ec3/clusters:/root/.ec3/clusters grycap/ec3 templates

name                summary
--------------------------------------------------------------------------------
blcr            Tool for checkpoint applications
bowtie2         Fast and sensitive read alignment
centos-ec2      CentOS 6.5 amd64 on EC2
chronos         An open-source tool to orchestrate job execution in a Mesos cluster
ckptman         Tool to automatically checkpoint applications running on Spot
                Instances
consul          Service Discovery and Configuration Made Easy
docker          An open-source tool to deploy applications inside software
                Containers
docker-compose A tool for defining and running multi-container Docker applications
kubefaas        Install FaaS frameworks in a kubernetes cluster
kubernetes      Install and configure a cluster using the grycap.kubernetes ansible
                role
[..]
```

Listing running clusters

To list the running clusters, use the command:

```
]$ cd $HOME
]$ sudo docker run -v /tmp/.ec3/clusters:/root/.ec3/clusters grycap/ec3 list

   name        state           IP           nodes
-------------------------------------------------------
  cluster      configured    212.189.145.XXX    0
```

Authorization file

The authorization file stores in plain text the credentials to access the cloud providers, the IM service and the VMRC service. Each line of the file is composed of pairs of key and value separated by semicolon, and refers to a single credential. The key and value should be separated by "=", that is **an equal sign preceded and followed by one white space at least**.

E.g.: Creation of an auth file to use OIDC token.

```
]$ cat auth_OIDC_CESNET-MCC_K8s.dat
id = cesnetostegi; type = OpenStack; host = https://identity.cloud.muni.cz;
username = egi.eu; tenant = openid; domain = panosc.eu; auth_version =
3.x_oidc_access_token; password = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
```

The Keystone Identity URL for the cloud provider can be retrieved from the EGI GOCDB.

Templates to configure the K8s cluster

The K8s virtual cluster will be configured with the following templates:

- **kubernetes (default template)**
- **ubuntu-1604-OIDC-CESNET_K8s (custom template)**
- **configure_nfs (custom template)**
- **refreshtoken (custom template)**

Custom templates are stored in the `$HOME/ec3/templates.`

Templates used to configure the K8s cluster

This section contains the templates used to configure the cluster.

Cluster_configure.radl

```
]$ cat ec3/templates/configure_nfs.radl
# http://www.server-world.info/en/note?os=CentOS_6&p=nfs&f=1
# http://www.server-world.info/en/note?os=CentOS_7&p=nfs
```

```
description nfs (
    kind = 'component' and
    short = 'Tool to configure shared directories inside a network.' and
   content = 'Network File System (NFS) client allows you to access shared
directories from Linux client. This recipe installs nfs from the repository and
shares the /home/ubuntu directory with all the nodes that compose the cluster.
Webpage: http://www.grycap.upv.es/clues/'
)
network public (
    outports contains '111/tcp' and
    outports contains '111/udp' and
    outports contains '2046/tcp' and
    outports contains '2046/udp' and
    outports contains '2047/tcp' and
    outports contains '2047/udp' and
    outports contains '2048/tcp' and
    outports contains '2048/udp' and
    outports contains '2049/tcp' and
    outports contains '2049/udp' and
    outports contains '892/tcp' and
    outports contains '892/udp' and
    outports contains '32803/tcp' and
    outports contains '32769/udp'
)
system front (
    ec3_templates contains 'nfs' and
    disk.0.applications contains (name = 'ansible.modules.grycap.nfs')
)
configure front (
@begin
  - roles:
      - { role: 'grycap.nfs', nfs_mode: 'front', nfs_exports: [{path: "/home",
export: "wn*.localdomain(rw,async,no_root_squash,no_subtree_check,insecure)"}] }
@end
)
system wn ( ec3_templates contains 'nfs' )
configure wn (
@begin
  - roles:
    - { role: 'grycap.nfs', nfs_mode: 'wn', nfs_client_imports: [{ local: "/home",
remote: "/home", server_host: '{{
hostvars[groups["front"][0]]["IM_NODE_PRIVATE_IP"] }}' }] }
@end
)
include nfs_misc (
  template = 'openports'
)
```

ubuntu-1604-OIDC-CESNET_K8s.radl

```
description ubuntu-1604-occi-CESNET (
    kind = 'images' and
```

```
    short = 'Ubuntu 16.04' and
    content = 'Image for EGI Ubuntu 16.04 LTS [Ubuntu/16.04/KVM]'
)
network public (
    provider_id = 'public-cesnet-78-128-251-GROUP' and
    outports contains '443/tcp' and
    outports contains '80/tcp'
)
network private (provider_id = 'auto_allocate_network')

system front (
    cpu.arch = 'x86_64' and
    cpu.count >= 8 and
    memory.size >= 16384m and
    disk.0.os.name = 'linux' and
    # PaNOSC tenant
    disk.0.image.url = 'ost://identity.cloud.muni.cz/e8d75fc1-ac32-4851-90b5-b4c925e9e6f8'
and
    disk.0.os.credentials.username = 'ubuntu'
)
system wn (
    cpu.arch = 'x86_64' and
    cpu.count >= 8 and
    memory.size >= 16384m and
    ec3_max_instances = 5 and # maximum number of working nodes in the cluster
    disk.0.os.name = 'linux' and
    # PaNOSC tenant
     disk.0.image.url = 'ost://identity.cloud.muni.cz/e8d75fc1-ac32-4851-90b5-b4c925e9e6f8'
and
    disk.0.os.credentials.username = 'ubuntu'
)
```

refreshtoken.radl

```
description refreshtoken (
    kind = 'component' and
    short = 'Tool to refresh LToS access token.' and
    content = 'Tool to refresh LToS access token.'
)
configure front (
@begin
  - vars:
      CLIENT_ID: ef4d5286-0db3-4c06-87ff-6a27ec97cb85
CLIENT_SECRET:
O-UODpEZZiceW3X47Kx_RDYnd5KJwrm-UzHJK_4Z5tfyKF5RbtbluJcnzeGzPz8xmlfbCWEjKjDj_aBT3H
lhlA
      REFRESH_TOKEN_FILE:
        ec3_file: refresh_token.py
    tasks:
    - name: Create dir /usr/local/ec3/
      file: path=/usr/local/ec3/ state=directory
```

```
    - copy:
        dest: /usr/local/ec3/refresh_token.py
        content: "{{REFRESH_TOKEN_FILE}}"
        mode: 0700
    - cron:
        name: "refresh token"
        minute: "*/5"
        job: "[ -f /usr/local/ec3/auth.dat ] && /usr/local/ec3/refresh_token.py {{
CLIENT_ID }} {{ CLIENT_SECRET }}"
        user: root
        cron_file: refresh_token
        state: present
@end
)
```

The refresh_token.py has to be copied in the `$HOME/ec3/templates`.

Create a K8s cluster

To create a K8s cluster, use the command:

```
]$ sudo docker run -v /home/centos/:/tmp/ \
 -v /home/centos/ec3/templates:/root/.ec3/templates \
 -v /tmp/.ec3/clusters:/root/.ec3/clusters grycap/ec3 launch k8s_cluster \
 kubernetes ubuntu-1604-OIDC-CESNET_K8s configure_nfs refreshtoken \
 -a /tmp/auth_OIDC_CESNET-MCC_K8s.dat
Creating infrastructure
Infrastructure successfully created with ID:
b5a86286-3e93-11ea-9f7c-fae41df0e987
Front-end state: launching
Front-end state: pending,
IP: NoneFront-end state: running,
IP: NoneFront-end state: running, IP: XXX.XXX.XXX.XXX
Front-end configured with IP XXX.XXX.XXX.XXX
Transferring infrastructure
Front-end ready!
```

Access the K8s cluster

To access the configured K8s cluster, use the command:

```
]$ cd $HOME
]$ sudo docker run -ti -v /tmp/.ec3/clusters:/root/.ec3/clusters
grycap/ec3 ssh k8s_cluster
[..]
```

Use the clues daemon to start workers in the K8s cluster:

```
]$ clues poweron wn1.localdomain wn2.localdomain
[..]
```

Use the `is_cluster_ready` CLI to check the status of the cluster. When the cluster is configured change the status of the workers to prevent clues shutdowns the sites when idle:

```
]$ clues disable wn1.localdomain wn2.localdomain
]$ clues status

node             state  enabled   time stable   (cpu,mem) used   (cpu,mem) total
-------------------------------------------------------------------------------
wn1.localdomain   used  disabled  21h06'28"     0.55,1394606080   8,16713998336
wn2.localdomain   used  disabled  21h06'28"     0.15,320864256    8,16714006528
wn3.localdomain   off   enabled   21h37'20"     0,0               1,1073741824
[..]
```

Listing running EC3 clusters

To list running clusters, use the command:

```
]$ cd $HOME
]$ sudo docker run -v /tmp/.ec3/clusters:/root/.ec3/clusters grycap/ec3
list
      name         state        IP        nodes
-------------------------------------------------
  k8s_cluster  configured  XXX.XXX.XXX.XXX    0
```

Destroy the EC3 cluster

To destroy the configured K8s cluster, use the command:

```
]$ cd $HOME
]$ sudo docker run -ti -v /tmp/.ec3/clusters:/root/.ec3/clusters
grycap/ec3 destroy k8s_cluster
WARNING: you are going to delete the infrastructure (including frontend
and nodes).
Continue [y/N]? Y
Success deleting the cluster!
```

PART IV - Configure the K8s cluster to spawn Jupyter notebooks

A Kubernetes cluster deployed into a cloud resource provider is in charge of managing the containers that will provide the service. On this cluster there are:

- 1 master node that manages the whole cluster;
- Support for load balancer or alternatively 1 or more edge nodes with a public IP and corresponding public DNS name (e.g. notebooks.egi.eu) where a k8s ingress HTTP reverse proxy redirects requests from users to other components of the service. The HTTP server has a valid certificate from one CA recognised at most browsers (e.g. Let's Encrypt).
- 1 or more nodes that host the JupyterHub server, the notebooks servers where the users will run their notebooks. The JupyterHub is deployed using the JupyterHub helm charts. These nodes should have enough capacity to run as many concurrent user notebooks as needed. Main constraint is usually memory.
- Support for Kubernetes PersistentVolumeClaims for storing the persistent folders. Default EGI-Notebooks installation uses NFS, but any other volume type with ReadWriteOnce capabilities can be used.

Configure the volume space in the K8s master node

Access the K8s master node and change the settings in the `/etc/exports`

```
]$ mkdir /volumes
]$ /home
wn*.localdomain(rw,async,no_root_squash,no_subtree_check,insecure)
/volumes wn*.localdomain(rw,no_root_squash,no_subtree_check)
kubeserver.localdomain(rw,no_root_squash,no_subtree_check)

]$ exportfs -ra
]$ sudo exportfs
/volumes        kubeserver.localdomain
/home           wn*.localdomain
/volumes        wn*.localdomain
```

Check the status of the K8s cluster:

```
]$ kubectl get nodes
NAME                     STATUS   ROLES    AGE     VERSION
kubeserver.localdomain   Ready    master   14m     v1.15.7
wn1.localdomain          Ready    <none>   4m21s   v1.15.7
wn2.localdomain          Ready    <none>   4m16s   v1.15.7
```

Helm

Install Helm in the K8s master node:

```
]$ wget https://get.helm.sh/helm-v3.0.2-linux-amd64.tar.gz
]$ tar zxf helm-v3.0.2-linux-amd64.tar.gz
]$ cd linux-amd64
]$ sudo mv helm /usr/local/bin/helm
]$ helm version
version.BuildInfo{Version:"v3.0.2",
GitCommit:"19e47ee3283ae98139d98460de796c1be1e3975f",
GitTreeState:"clean", GoVersion:"go1.13.5"}
```

nfs-provider

Install the nfs-provider in the K8s master node:

```
]$ sudo helm repo add stable
https://kubernetes-charts.storage.googleapis.com
"stable" has been added to your repositories

]$ sudo helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈ Happy Helming!⎈

]$ sudo helm install nfs-provisioner stable/nfs-client-provisioner \
                     --namespace kube-system \
                     --set nfs.server=192.168.8.57 \
                     --set storageClass.defaultClass=true \
                     --set nfs.path=/volumes \
                     --set
tolerations[0].effect=NoSchedule,tolerations[0].key=node-role.kubernetes.i
o/master

NAME: nfs-provisioner
LAST DEPLOYED: Fri Jan 24 12:42:24 2020
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

nginx-ingress

Create the yaml file:

```
]$ cat ingress.yaml
controller:
  tolerations:
  - effect: NoSchedule
```

```
    key: node-role.kubernetes.io/master
  service:
    type: NodePort
    externalIPs:
    - (static IP of the master node)
  config:
    proxy-body-size: '0'
    ssl-protocols: 'TLSv1 TLSv1.1 TLSv1.2'
    ssl-ciphers:
'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES
128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:E
[..]'
defaultBackend:
  tolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
```

Install nginx-ingress in the K8s master node:

```
]$ sudo helm install -name cluster-ingress --namespace kube-system -f ingress.yaml
stable/nginx-ingress
NAME: cluster-ingress
LAST DEPLOYED: Fri Jan 24 12:47:28 2020
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The nginx-ingress controller has been installed.

Get the application URL by running these commands:
  export HTTP_NODE_PORT=$(kubectl --namespace kube-system get services -o
jsonpath="{.spec.ports[0].nodePort}" cluster-ingress-nginx-ingress-controller)
  export HTTPS_NODE_PORT=$(kubectl --namespace kube-system get services -o
jsonpath="{.spec.ports[1].nodePort}" cluster-ingress-nginx-ingress-controller)
  export NODE_IP=$(kubectl --namespace kube-system get nodes -o
jsonpath="{.items[0].status.addresses[1].address}")

  echo "Visit http://$NODE_IP:$HTTP_NODE_PORT to access your application via
HTTP."
  echo "Visit https://$NODE_IP:$HTTPS_NODE_PORT to access your application via
HTTPS."

An example Ingress that makes use of the controller:

  apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    annotations:
```

```
      kubernetes.io/ingress.class: nginx
    name: example
    namespace: foo
  spec:
    rules:
      - host: www.example.com
        http:
          paths:
            - backend:
                serviceName: exampleService
                servicePort: 80
              path: /
    # This section is only required if TLS is to be enabled for the Ingress
    tls:
        - hosts:
            - www.example.com
          secretName: example-tls


If TLS is enabled for the Ingress, a Secret containing the certificate and key
must also be provided:

  apiVersion: v1
  kind: Secret
  metadata:
    name: example-tls
    namespace: foo
  data:
    tls.crt: <base64 encoded cert>
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
```

Install the cert-manager certificate

Update the K8s front-node hostname. You can register a new entry for the front-node in a DNS, or use the Dynamic DNS service for the EGI FedCloud infrastructure.
For this particular scenario, the Dynamic DNS service of EGI was used. To update the hostname of the master, after you have obtained the host secret from the service, just run the command:

```
]$ curl
https://notebooks-panosc.fedcloud-tf.fedcloud.eu:X97JBX27S7@nsupdate.fedcl
oud.eu/nic/update
```

Create the cert-manager namespace:

```
]$ sudo kubectl create namespace cert-manager
namespace/cert-manager created

]$ sudo kubectl apply --validate=false -f
https://raw.githubusercontent.com/jetstack/cert-manager/release-0.12/deplo
```

```
y/manifests/00-crds.yaml
[..]

]$ sudo helm repo add jetstack https://charts.jetstack.io
"jetstack" has been added to your repositories

]$ sudo helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "jetstack" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈ Happy Helming!⎈

]$ cat certmanager.yaml
webhook:
  enabled: false
ingressShim:
  defaultIssuerName: letsencrypt-prod
  defaultIssuerKind: ClusterIssuer
tolerations:
- effect: NoSchedule
  key: node-role.kubernetes.io/master
cainjector.tolerations:
- effect: NoSchedule
  key: node-role.kubernetes.io/master
```

Install the cert-manager in the K8s master node:

```
]$ sudo helm install --name certs-man \
                     --namespace cert-manager --version=0.12.0 \
                     -f certmanager.yaml jetstack/cert-manager
NAME: certs-man
LAST DEPLOYED: Fri Jan 24 13:09:32 2020
NAMESPACE: cert-manager
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
cert-manager has been deployed successfully!
[..]
```

ClusterIssuer

Create the ClusterIssuer in the K8s master node:

```
]$ cat clusterissuer.yaml
apiVersion: cert-manager.io/v1alpha2
kind: ClusterIssuer
```

```
metadata:
  name: letsencrypt-prod
spec:
  acme:
    email: giuseppe.larocca@egi.eu
    server: https://acme-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
      name: cluster-issuer-account-key
    # Add a single challenge solver, HTTP01 using nginx
    solvers:
    - http01:
        ingress:
          class: nginx
```

Install the ClusterIssuer in the K8s master node

```
]$ sudo kubectl apply -f clusterissuer.yaml
clusterissuer.cert-manager.io/letsencrypt-prod created
```

JupyterHub

Create the jupyter configuration file for PaNOSC in the K8s master node:

```
proxy:
  # FIXME NEEDS INPUT
  secretToken: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
  service:
    #type: 8888
    type: NodePort

ingress:
  enabled: true
  annotations:
    kubernetes.io/tls-acme: "true"
  hosts:
  # FIXME NEEDS INPUT
  - notebooks-panosc.fedcloud-tf.fedcloud.eu
  tls:
  - hosts:
    # FIXME NEEDS INPUT
    - notebooks-panosc.fedcloud-tf.fedcloud.eu
    secretName: acme-tls-catchall

singleuser:
  storage:
    capacity: 10Gi
    dynamic:
```

```
      storageAccessModes: ["ReadWriteMany"]
  memory:
    limit: 1G
    guarantee: 128M
  cpu:
    limit: 1
    guarantee: .02
  image:
    name: eginotebooks/single-user
    tag: "2044ab8"

hub:
  image:
    name: enolfc/hub
    tag: "20200316.02"
  extraEnv:
    JUPYTER_ENABLE_LAB: 1
    OAUTH2_AUTHORIZE_URL:
https://proxy.umbrellaid.org/saml2sp/OIDC/authorization
    OAUTH2_TOKEN_URL: https://proxy.umbrellaid.org/OIDC/token
    OAUTH_CALLBACK_URL:
https://notebooks-panosc.fedcloud-tf.fedcloud.eu/hub/oauth_callback
  extraConfig:
    volume-handling: |-
      from egi_hub_addons.spawner import datahub_pod_modifier
      c.KubeSpawner.modify_pod_hook =
datahub_pod_modifier(onezone_url='https://onezone-panosc.egi.eu',
oneprovider_host='oneprovider-pn.onezone-panosc.egi.eu',
force_proxy_io=True,
force_direct_io=False)
      c.JupyterHub.authenticate_prometheus = False
      c.JupyterHub.log_level = 'DEBUG'
      c.JupyterHub.template_paths =
['/usr/local/share/egi-hub/templates/']

auth:
  state:
    enabled: true
    # FIXME NEEDS INPUT
    cryptoKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
  type: custom
  admin:
    access: true
    users:
      - XXXXXXXXXXXXXXX
  custom:
```

```
    className: egiauthenticator.oidc.OIDCAuthenticator
    config:
      client_id: "XXX-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
      client_secret: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
      oauth_callback_url:
"https://notebooks-panosc.fedcloud-tf.fedcloud.eu/hub/oauth_callback"
      token_url: "https://proxy.umbrellaid.org/OIDC/token"
      userdata_url: "https://proxy.umbrellaid.org/OIDC/userinfo"
      scope: ["openid", "profile", "email"]
      client_auth_method: "client_secret_post"
      username_key: "name"
      onezone_url: "https://onezone-panosc.egi.eu"
      onezone_idp: "UmbrellaID"
      oneprovider_host: "cesnet-pn.onezone-panosc.egi.eu"

]$ sudo helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "jetstack" chart repository
...Successfully got an update from the "jupyterhub" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈ Happy Helming!⎈
```

Create the namespace:

```
]$ sudo kubectl create namespace panosc
namespace/panosc created
```

Install the Jupyterhub in the namespace:

```
]$ sudo helm install panosc --namespace panosc --version 0.9.0-beta.2 -f
panosc.yaml jupyterhub/jupyterhub
```

PART V - Configure a Binder instance to spawn Jupyter notebooks

Registry

Create the registry.yaml configuration file for helm:

```
]$ cat registry.yaml
persistence:
  enabled: false

secrets:
  htpasswd: "binder:$2y$05$1Gc[..]AsSlLbNVak11s4VtK"

ingress:
  enabled: true
  annotations:
    kubernetes.io/tls-acme: "true"
    ingress.kubernetes.io/proxy-body-size: "0"
  hosts: [registry-panosc.ops.fedcloud.eu]
  tls:
  - hosts: [registry-panosc.ops.fedcloud.eu]
    secretName: acme-tls-registry
```

Create the namespace:

```
]$ sudo kubectl create namespace registry
namespace/registry created
```

Install the registry in the namespace:

```
]$ sudo helm install registry stable/docker-registry \
        -f registry.yaml --namespace registry
[..]
```

Binder

Create the binder.yaml configuration file for helm:

```
]$ cat binder.yaml
---
config:
  BinderHub:
    auth_enabled: true
    hub_url: https://binder-panosc.fedcloud-tf.fedcloud.eu/hub
    use_registry: true
    image_prefix: registry-panosc.ops.fedcloud.eu/binder-
    build_image: jupyter/repo2docker:0.10.0
```

```
    use_named_servers: true

registry:
  url: "https://registry-panosc.ops.fedcloud.eu"
  username: binder
  password: **********************

dind:
  enabled: true
  daemonset:
    extraArgs:
    - --mtu=1392

ingress:
  enabled: true
  annotations:
    kubernetes.io/tls-acme: "true"
  hosts: [binder-panosc.fedcloud-tf.fedcloud.eu]
  tls:
  - hosts:
    - binder-panosc.fedcloud-tf.fedcloud.eu
    secretName: acme-tls-binder

service:
  type: NodePort

jupyterhub:
  proxy:
    secretToken: "****************************"
    service:
      type: NodePort

  ingress:
    enabled: true
    annotations:
      kubernetes.io/tls-acme: "true"
    hosts: [binder-panosc.fedcloud-tf.fedcloud.eu]
    tls:
    - hosts: [binder-panosc.fedcloud-tf.fedcloud.eu]
      secretName: acme-tls-binder

  singleuser:
    storage:
      type: none
      capacity: 1Gi
      dynamic:
```

```
        storageAccessModes: ["ReadWriteMany"]
    memory:
      limit: 3.5G
      guarantee: 1G
    cpu:
      limit: 2
      guarantee: .02
    cmd: jupyterhub-singleuser
    imagePullSecret:
      enabled: true
      registry: registry-panosc.ops.fedcloud.eu
      username: binder
      password: ******************************************

  hub:
    baseUrl: "/hub"
    allowNamedServers: true
    namedServerLimitPerUser: 0
    image:
      name: enolfc/hub
      tag: "20200316.02"
    extraConfig:
      volume-handling: |
        c.JupyterHub.authenticate_prometheus = False
        c.JupyterHub.log_level = 'DEBUG'
        c.JupyterHub.template_paths =
['/usr/local/share/egi-hub/templates/']
    extraEnv:
      OAUTH2_AUTHORIZE_URL:
https://proxy.umbrellaid.org/saml2sp/OIDC/authorization
      OAUTH2_TOKEN_URL: https://proxy.umbrellaid.org/OIDC/token
    services:
      binder:
        admin: true
        oauth_redirect_uri:
"http://binder-panosc.fedcloud-tf.fedcloud.eu/oauth_callback"
        oauth_client_id: "binder-egi"
        apiToken: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

  auth:
    type: custom
    state:
      enabled: true
      cryptoKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
    admin:
      access: true
```

```
      users:
        - efernandez
        - efernandez2
        - glarocca5
    custom:
      className: egiauthenticator.oidc.OIDCAuthenticator
      config:
        client_id: "XXX-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
        client_secret: "**********************************"
        oauth_callback_url:
"https://binder-panosc.fedcloud-tf.fedcloud.eu/hub/oauth_callback"
        token_url: "https://proxy.umbrellaid.org/OIDC/token"
        userdata_url: "https://proxy.umbrellaid.org/OIDC/userinfo"
        scope: ["openid", "profile", "email"]
        client_auth_method: "client_secret_post"
        username_key: "name"
        onezone_url: "https://onezone-panosc.egi.eu"
        onezone_idp: "eduTEAMS"
        oneprovider_host: "cesnet-op.onezone-panosc.egi.eu"

  cull:
    timeout: 900
    every: 300


]$ sudo helm install binder jupyterhub/binderhub \
      --version=0.2.0-n153.h7db895c \
      --namespace=binder -f binder.yaml
```