



Grant Agreement No.: 723174

Call: H2020-ICT-2016-2017

Topic: ICT-38-2016 - MEXICO: Collaboration on ICT

Type of action: RIA



## D3.3: SmartSDK Platform Manager

Revision: v.1.0

Work package	WP 3
Task	Task 3.3
Due date	31/05/2017
Submission date	31/05/2017
Deliverable lead	FBK
Version	1.0
Authors	Daniele Pizzolli (FBK), Daniel Zozin (FBK)
Reviewers	Tomas Aliaga (MARTEL), Miguel G. Mendoza (ITESM)

Abstract	This deliverable documents the software usage, installation and maintenance of the SmartSDK platform. Particular emphasis is reserved to the integration with the FIWARE Lab and to the deployment of SmartSDK recipes.
Keywords	Container Orchestration, FIWARE Lab, docker, rancher, docker-compose, docker stack

### Disclaimer

The information, documentation and figures available in this deliverable, is written by the SmartSDK (A FIWARE-based Software Development Kit for Smart Applications for the needs of Europe and Mexico) – project consortium under EC grant agreement 723174 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

### Copyright notice

© 2016 - 2018 SmartSDK Consortium

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		R
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CI	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to SmartSDK project and Commission Services	

\* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

OTHER: Software, technical diagram, etc.

## EXECUTIVE SUMMARY

---

The SmartSDK Platform Manager allows to register, configure, manage and monitor the deployment of SmartSDK recipes.

It can be installed on the FIWARE Lab nodes. It allows the creation of fully separated environments. Every environment is composed by grouping a set of hosts. Adding an host to an existing environment is a straightforward procedure and a number of cloud providers are already supported, including the FIWARE Lab itself.

This deliverable documents why rancher was selected as the base software for the SmartSDK Platform, the usage of the SmartSDK Platform Manager, highlights the main use cases, and offers some advice in order to deploy the SmartSDK recipes.

In the end we will discuss future plans, with the objective to have even a more integrated platform at developer disposal.

## TABLE OF CONTENTS

<b>EXECUTIVE SUMMARY .....</b>	<b>3</b>
<b>TABLE OF CONTENTS .....</b>	<b>4</b>
<b>LIST OF FIGURES .....</b>	<b>6</b>
<b>ABBREVIATIONS .....</b>	<b>7</b>
<b>GLOSSARY .....</b>	<b>8</b>
<b>1 INTRODUCTION .....</b>	<b>9</b>
1.1 The SmartSDK Platform Manager .....	9
1.2 SmartSDK Overall Architecture.....	10
1.3 Structure of the deliverable.....	11
1.4 Audience .....	11
<b>2 SMARTSDK PLATFORM USAGE.....</b>	<b>12</b>
2.1 Introduction.....	12
2.2 Environment Templates.....	12
2.2.1 The Number of Swarm Managers.....	12
2.2.2 Rancher IPsec plugin MTU (FIWARE LAB) .....	13
2.2.3 Configure the Rancher IPsec plugin MTU .....	14
2.2.4 (Optional) Rancher NFS plugin.....	14
2.3 Environment .....	14
2.4 Host requirements.....	14
2.5 Hosts on the FIWARE Lab.....	14
2.6 Setup the OpenStack project for hosting a SmartSDK platform environment. ....	14
2.7 OpenStack client Setup.....	15
2.7.1 Install python-openstackclient.....	15
2.7.2 Load the OpenStack client settings.....	15
2.8 Start with a clean OpenStack and docker-machine environment .....	15
2.9 Add proper security group roles .....	16
2.10 Docker Machine.....	16
2.10.1 Resolve dependencies for docker-machine .....	17
2.10.2 Install the docker-machine.....	17
2.10.3 Start with a clean docker-machine environment .....	17
2.10.4 Setup docker-machine from the command line interface .....	17
2.11 Setup security groups.....	18
2.12 Download and install rancher-compose.....	19
2.13 Download and install rancher CLI.....	19
2.14 Create API keys .....	19
2.15 Write CLI configuration .....	19
2.16 Provision of rancher hosts using machine drivers .....	20
2.16.1 Set host parameters .....	20
2.16.2 Create and access hosts.....	21
<b>3 SMARTSDK PLATFORM ADVANCED USAGE .....</b>	<b>22</b>
3.1 User management integrated with FIWARE Lab Oauth.....	22
3.2 Enable the FIWARE Lab Oauth.....	23
3.3 Machine driver and User Interface Plugin for FIWARE Lab Nodes .....	24
3.4 Using a VPN for overcoming NAT issues .....	25

3.5	Provision of rancher hosts using custom hosts .....	26
<b>4</b>	<b>DEPLOY SMARTSDK RECIPES ON SMARTSDK PLATFORM .....</b>	<b>27</b>
4.1	Deployment using rancher-compose .....	27
4.2	Deployment using docker stack deploy .....	27
4.3	Test deployments .....	28
4.3.1	Swarm deployment with replication (docker stack) .....	28
4.3.2	Rancher compose example from catalog (community-recipes) .....	28
4.3.3	Swarm deployment example (smartsdk-recipes) .....	28
4.3.4	Rancher compose example (fiware tour demo app) .....	29
<b>5</b>	<b>SMARTSDK PLATFORM INSTALLATION .....</b>	<b>30</b>
5.1.1	Install docker-compose .....	30
5.1.2	Set some useful variables .....	30
5.2	Configure Access Control .....	31
5.3	Set registration URL .....	32
5.4	(Optional) Configuration of the NFS server .....	32
5.5	Enable the catalog in the web interface .....	32
5.6	Import environment templates .....	33
5.7	Create environments based on templates .....	33
5.8	Activate FIWARE Lab machine driver .....	33
5.9	Activate openstack machine driver .....	34
5.10	Upgrade Rancher Server .....	34
5.10.1	Notes based on the guide .....	34
5.10.2	Setup some variables for the new version .....	34
5.10.3	Stop the running server .....	34
5.10.4	Data persistence, keep a reference .....	34
5.10.5	Do the upgrade .....	34
5.10.6	Check the values .....	35
<b>6</b>	<b>CONCLUSION AND FUTURE WORK .....</b>	<b>36</b>
6.1	User Interface for docker-compose v3 .....	36
6.2	User management integrated with FIWARE Lab for SmartSDK platform .....	36
6.3	Overlay network .....	36
6.4	Persistent storage configuration .....	36
6.5	SmartSDK Platform Manager in HA .....	36
<b>APPENDIX A</b>	<b>ADVANCED FUNCTIONALITIES .....</b>	<b>37</b>
A.1	How Docker Swarm networking works .....	37
A.2	How exposing services through load balancers works .....	37
A.3	Issues with rancher hosts for natted machines .....	37
A.4	Advanced analysis of the issues related to non-standard MTU usage .....	38
A.5	Install modern openstackclient with pip .....	38
A.6	List Available Images in an OpenStack Project .....	39
A.7	List Available Flavors in an OpenStack Project .....	39
A.8	Useful script to manage Docker .....	39
A.9	Change the MTU of all interfaces to 1400 .....	39
A.10	Workaround for sudo complaint .....	39
A.11	Add the docker group to the current user .....	39
A.12	Workaround to view display error on FIWARE Lab portal .....	39
A.13	Rancher General cleanup .....	40

LIST OF FIGURES

---

**Figure 1: The SmartSDK overall picture .....9**  
**Figure 2: Simple overview of the SmartSDK architecture .....10**  
**Figure 3: Simple overview of the SmartSDK usage.....11**  
**Figure 4: Setting the manager number in environment template settings.....13**  
**Figure 5: Screenshot of the Access control Setup for FIWARE Lab Oauth.....22**

## ABBREVIATIONS

---

<b>API</b>	Application Programming Interface
<b>CLI</b>	Command Line Interface
<b>DNS</b>	Domain Name System
<b>FQDN</b>	Fully Qualified Domain Name
<b>GE</b>	Generic Enabler
<b>GEri</b>	Generic Enabler reference implementation
<b>HA</b>	High Availability
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IoT</b>	Internet of Things
<b>IP</b>	Internet Protocol
<b>TCP</b>	Transmission Control Protocol
<b>UI</b>	User Interface
<b>VPN</b>	Virtual Private Network
<b>VM</b>	Virtual Machine

## GLOSSARY

---

**OpenStack project.** Projects represent the base unit of “ownership” in OpenStack, in that all resources in OpenStack should be owned by a specific project. In OpenStack Identity, a project must be owned by a specific domain. (Definition source: OpenStack glossary).



# 1 INTRODUCTION

The SmartSDK Platform Manager allows to:

- ➔ Register
- ➔ Configure
- ➔ Manage
- ➔ Monitor

the deployment of SmartSDK recipes. The documentation of SmartSDK recipes is detailed in the SmartSDK recipes deliverable. The relationship with the other components of SmartSDK is detailed in the Figure 1.

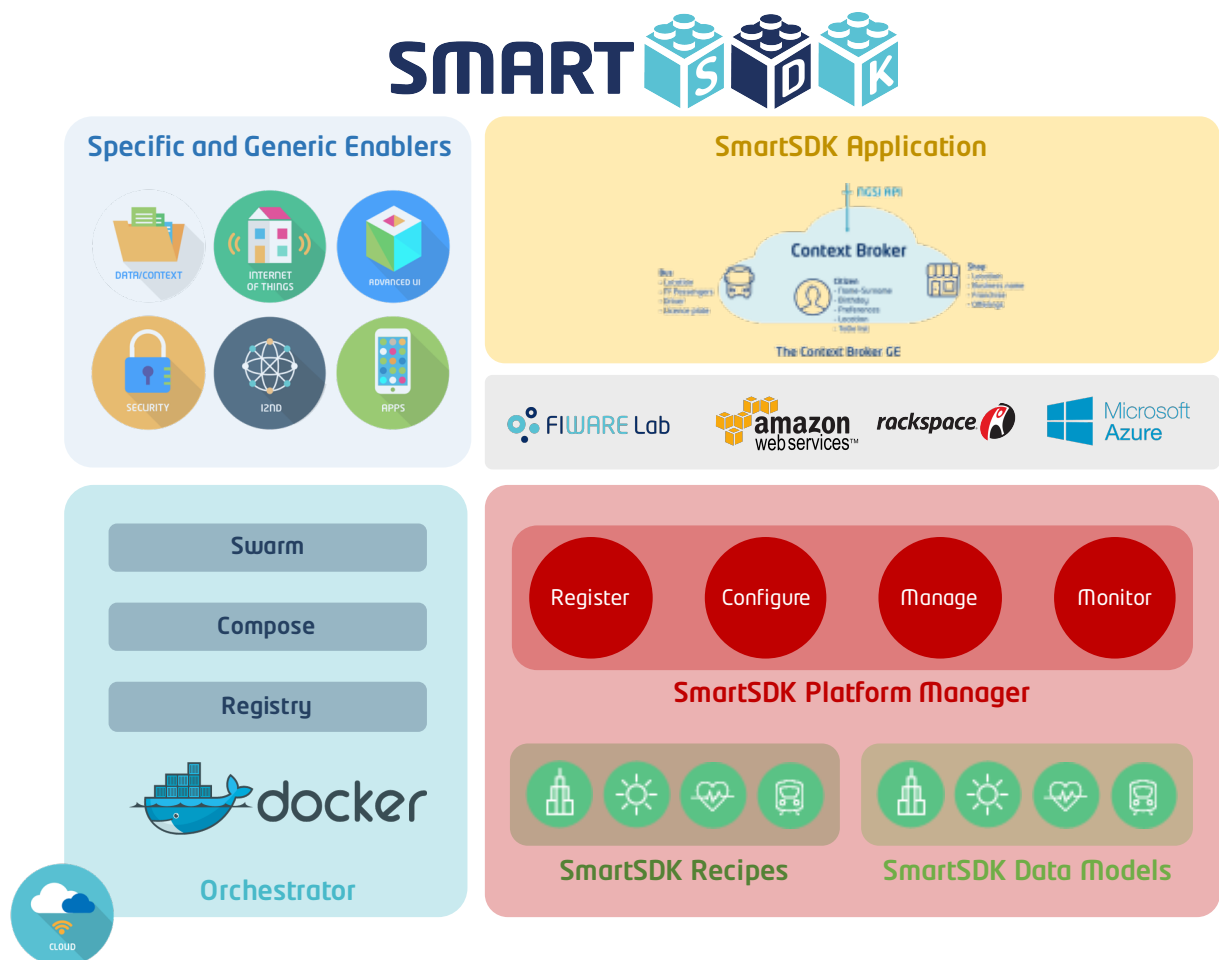


Figure 1: The SmartSDK overall picture

## 1.1 The SmartSDK Platform Manager

The SmartSDK Platform Manager duties are related to the deployment of SmartSDK recipes. It must also be compliant with the “design principles” of the SmartSDK project. Developing all the requested features from scratch did not make sense. We think that rancher is mature enough to support our initial needs, and in the reasonable future it will fully support our needs. In the areas where rancher is lacking we develop custom extensions or document suitable workarounds.

So far, at the current state of the project, is possible to use the SmartSDK Platform Manager to deploy SmartSDK recipes on the FIWARE Lab.

Rancher was chosen as the SmartSDK Platform Manager because it fits well the need and the requirement of the project. Rancher respect all the “design principles” of the SmartSDK project. In details:

- ➔ **Restful APIs.** Rancher offers most of its functionalities via API and are well documented on the Rancher API documentation site.
- ➔ **Reusability and Openness.** Rancher is released under an Apache License Version 2.0, The development is done on github. Rancher is still a young project, currently there are over 1500 open issues, but over 7000 were closed since November 2014. Rancher relies on other third party technologies, all of them are released under an Open Source License.
- ➔ **Cloudification and Microservices.** The Rancher application itself is made by different components, respecting the good design patterns for microservices-based application. The deployment of applications by Rancher can be done using a catalog or docker-compose stack recipes, currently one of the most advanced ways available to deploy applications in the cloud.
- ➔ **Market and community relevance.** Rancher has a very active community, that mostly discuss on the official forum. By supporting the deployment of application using docker swarm mode and Kubernetes it can be adopted by two broad and growing communities.

## 1.2 SmartSDK Overall Architecture

The SmartSDK Platform Manager uses rancher as a base to offer his services. A SmartSDK Platform Manager user will be able to instantiate one or multiple “environments” in order to deploy his application. See Figure 2 for a component representation. See Figure 3 for a reference of the steps involved.

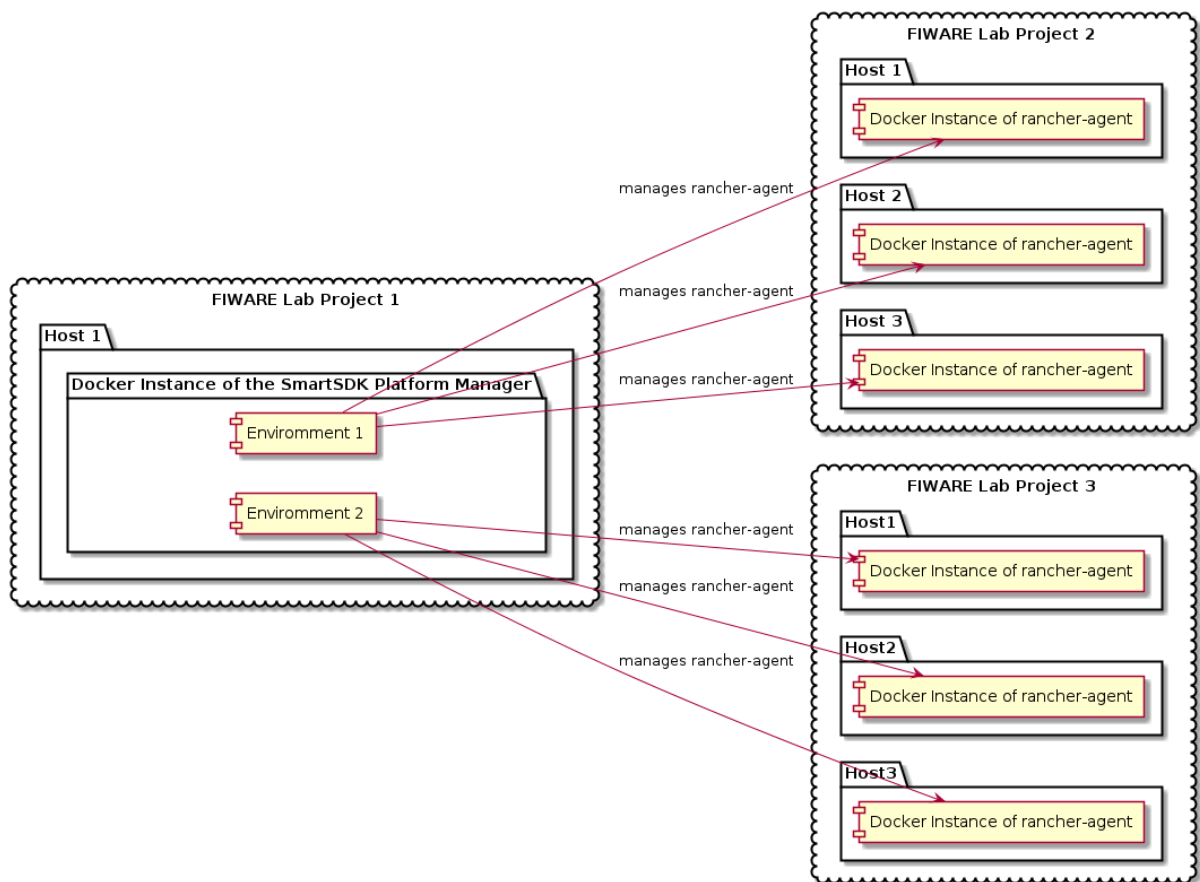


Figure 2: Simple overview of the SmartSDK architecture



Figure 3: Simple overview of the SmartSDK usage

### 1.3 Structure of the deliverable

The deliverable is structured as follow:

- ➔ INTRODUCTION will introduce the main concepts of the SmartSDK Platform.
- ➔ SMARTSDK PLATFORM USAGE will introduce the usage of SmartSDK.
- ➔ SMARTSDK PLATFORM ADVANCED USAGE will detail some advanced configurations.
- ➔ DEPLOY SMARTSDK RECIPES ON SMARTSDK PLATFORM show some SmartSDK recipes deployment examples.
- ➔ SMARTSDK PLATFORM INSTALLATION will introduce the installation and administration of SmartSDK.
- ➔ CONCLUSION AND FUTURE WORK summarizes the plan for future work in relation to the whole SmartSDK project.

### 1.4 Audience

This deliverable is mainly intended for:

- ➔ Developers interested into deploying SmartSDK application recipes.
- ➔ Operators interested to deploy the SmartSDK Platform Manager in a production context.

## 2 SMARTSDK PLATFORM USAGE

---

### 2.1 Introduction

A number of steps need to be followed in order to have a working docker swarm cluster. First an environment template must be configured, then an environment must be added, then some hosts running docker must added and finally docker must be configured for swarm mode on each of those hosts. Each step can be completed by choosing multiple options. For each option we will detail the pros and cons. We will spend some time especially to detail the solutions or the workarounds that works well on a FIWARE Lab installation.

### 2.2 Environment Templates

The SmartSDK Platform uses environment templates in order to offer some configured templates with default values. User can choose one of the default template, or start the creation of a new template.

Each template contains a predefined set of services and configuration for the Environment. For example you may add to the template, or simply reconfigure, the “Rancher IPsec” overlay network, or the “Rancher NFS” or the “Portainer.io” web user interface.

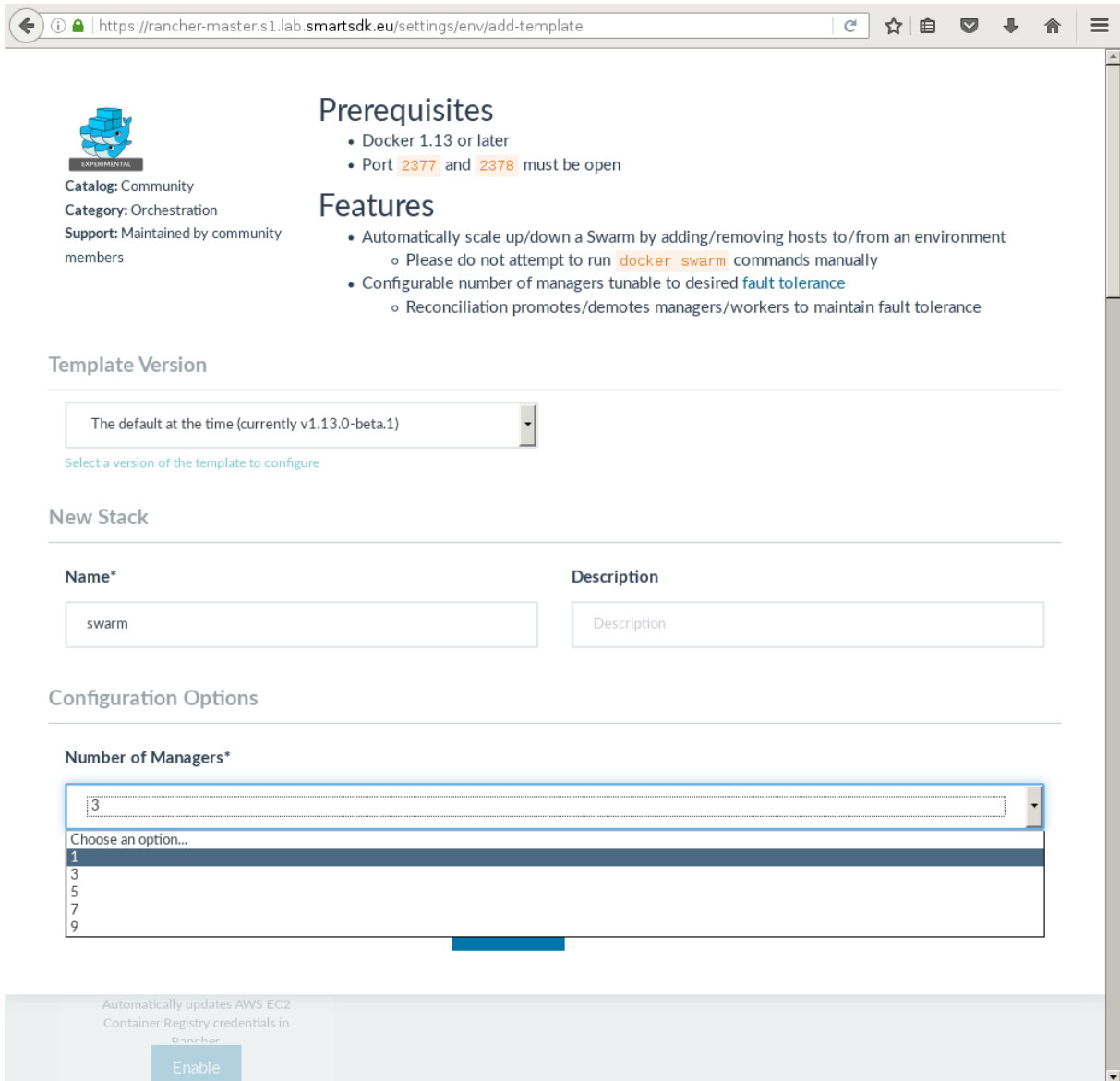
The SmartSDK Platform allow the creation of templates for “Docker Swarm”.

Proper attention must be dedicated for the configuration of the:

- ➔ The Number of Swarm Managers
- ➔ Rancher IPsec plugin MTU (FIWARE Lab)
- ➔ (Optional) Rancher NFS plugin

#### 2.2.1 The Number of Swarm Managers

The Number of Swarm Managers in the template will affect the high availability of the swarm mode. The default number is 3, but can be lowered to 1 for simple installation for evaluation purposes where the high availability of the managers is not needed. See the screenshot at Figure 4.



**Prerequisites**

- Docker 1.13 or later
- Port **2377** and **2378** must be open

**Features**

- Automatically scale up/down a Swarm by adding/removing hosts to/from an environment
  - Please do not attempt to run **docker swarm** commands manually
- Configurable number of managers tunable to desired **fault tolerance**
  - Reconciliation promotes/demotes managers/workers to maintain fault tolerance

**Template Version**

The default at the time (currently v1.13.0-beta.1)

Select a version of the template to configure

**New Stack**

Name*	Description
swarm	Description

**Configuration Options**

**Number of Managers\***

3

Choose an option...

- 1
- 3
- 5
- 7
- 9

Automatically updates AWS EC2 Container Registry credentials in Rancher

Enable

**Figure 4: Setting the manager number in environment template settings.**

## 2.2.2 Rancher IPsec plugin MTU (FIWARE LAB)

### 2.2.2.1 How to find out the MTU of your host

If your provider does not offer any documentation regarding the default MTU you can search the MTU value by yourself.

In order to find out the MTU of the device connected to the default gateway (which usually is the one that allow also local area network connectivity) of your host, connect to it and issue the following commands:

```
# Find out the device that is the default gateway
DEFAULT_GW_DEV=$(ip route | awk '/^default/ {print $NF; exit}')
# Find out the MTU of the default gateway
DEFAULT_GW_MTU=$(ip addr show "${DEFAULT_GW_DEV}" | grep -oP '(?<=mtu ) [0-9]*')
printf "%s\n" "${DEFAULT_GW_MTU}"
```

If the value is lower than the common value of 1500 bytes, you should take additional care because

the overlay network created to allow the communication between the swarm cluster nodes assumes the default value of 1500 bytes.

A wrong configuration of the MTU can prevent proper communication between nodes and make the system totally unusable.

### 2.2.3 Configure the Rancher IPsec plugin MTU

The MTU for the Rancher IPsec plugin must be lower or equal of the MTU of the host (found out following the previous section).

For example on the Spain2 FIWARE Lab node the MTU is currently 1400 bytes.

### 2.2.4 (Optional) Rancher NFS plugin

It is possible to add to the environment template an NFS server in order to have a shared storage service available to the SmartSDK Platform Manager. The NFS server must be reachable from the swarm nodes. The parameters to configure are:

- ➔ the IP address of the server
- ➔ the exported path

See (Optional) Configuration of the NFS server for further information.

## 2.3 Environment

Starting from a template the user can instantiate an Environment. In order to have a fully functional environment, a number of hosts equal or greater than the number of swarm managers selected in the template must be added to the environment.

## 2.4 Host requirements

The host requirements varies from version to version. In general the exact requirements are listed in the template. The minimal setup will be satisfied with the following:

- ➔ A modern Linux distribution
- ➔ Installation of docker 1.13 or later
- ➔ Port 2377 and 2378 must be open between the hosts

## 2.5 Hosts on the FIWARE Lab

It is possible, and encouraged, to add nodes running on a project in the FIWARE Lab. We assume that the entire project is dedicated to the environment.

## 2.6 Setup the OpenStack project for hosting a SmartSDK platform environment.

Before the host creation, the OpenStack project need to be configured with proper access rules and images.

The following settings are working settings but not the best secure setup. For example you may want to restrict incoming connection to management ports only from a well known subset of IPs.

## 2.7 OpenStack client Setup

- ➔ Install `python-openstackclient`, in order to have the tool called `openstack`
- ➔ Load your local `openrc` file in order to have the setting for your account loaded

### 2.7.1 Install python-openstackclient

To install the OpenStack command line client, on a modern debian-based Linux distribution you need to issue the command:

```
sudo apt install --yes python3-openstackclient
```

This will install the distribution supplied client, that from time to time maybe is an outdated version. You can install the latest OpenStack client using various methods. See for example: Install modern `openstackclient` with `pip`.

### 2.7.2 Load the OpenStack client settings

To load your settings and credential you need to set some well known OpenStack related variables. One of the most common way to do it is to source the `openrc` file.

```
. openrc
```

## 2.8 Start with a clean OpenStack and docker-machine environment

This may cause data loss! Double check the source of the correct `openrc` file.

- ➔ Clean up servers (also known as hosts):

```
openstack server list -f value -c ID | xargs -trn1 openstack server delete
openstack server list
```

- ➔ Clean up volumes:

```
openstack volume list -f value -c ID | xargs -trn1 openstack volume delete
openstack volume list
```

- ➔ Clean up security groups:

This will not work with `python-openstackclient` 2.3.0 shipped with ubuntu 16.04. See Install modern `openstackclient` with `pip` for a workaround.

```
openstack security group list -f value -c ID | xargs -trn1 openstack
security group delete
openstack security group rule list -f value -c ID default | \
  xargs -trn1 openstack security group rule delete
openstack security group set default --description 'empty default'
openstack security group list
```

- ➔ Clean up keypair:

```
openstack keypair list -f value -c Name | xargs -trn1 openstack keypair
delete
openstack keypair list
```

You may also want to cleanup:

- ➔ Clean up floating IP reservation
- ➔ Clean up snapshots
- ➔ Clean up storage containers

- ➔ Clean up images
- ➔ Clean up flavors

## 2.9 Add proper security group roles

The configuration of the networking for the hosts that belong to the rancher cluster offers a lot of options. In this section we will setup security roles for a group of host that is started on a generic OpenStack installation that allows the binding of a public IP for each host in order to allow the direct connectivity with the rancher server.

- ➔ Allow cluster communication among rancher nodes:

```
openstack security group create rancher-cluster \
  --description "Security group among rancher nodes"

openstack security group rule create rancher-cluster \
  --protocol tcp --dst-port 22:22 --remote-group rancher-cluster
```

- ➔ Allow IPsec according to the documentation:

```
openstack security group rule create rancher-cluster \
  --protocol udp --dst-port 500:500 --remote-group rancher-cluster
openstack security group rule create rancher-cluster \
  --protocol udp --dst-port 4500:4500 --remote-group rancher-cluster
```

- ➔ Allow access to health-check:

```
openstack security group rule create rancher-cluster \
  --protocol tcp --dst-port 80:80 --remote-group rancher-cluster
```

- ➔ Allow access to docker-engine and docker-swarm daemons:

```
# Port 2376 2377 2388
openstack security group rule create rancher-cluster \
  --protocol tcp --dst-port 2376:2378 --remote-group rancher-cluster
```

- ➔ If you use docker machine for the public network, the ports must be public available:

```
openstack security group rule create rancher-cluster \
  --protocol tcp --dst-port 2376:2378
```

- ➔ Allow access for swarm ingress network:

```
openstack security group rule create rancher-cluster \
  --protocol tcp --dst-port 7946:7946 --remote-group rancher-cluster
openstack security group rule create rancher-cluster \
  --protocol udp --dst-port 7946:7946 --remote-group rancher-cluster
openstack security group rule create rancher-cluster \
  --protocol udp --dst-port 4789:4789 --remote-group rancher-cluster
```

- ➔ (Optional) Allow NFSv4 port access to cluster security group:

```
openstack security group rule create rancher-cluster \
  --protocol tcp --dst-port 2049:2049 --remote-group rancher-cluster
```

## 2.10 Docker Machine

Docker Machine enables the fast deployment of new hosts with docker installed and ready to use. Docker machine relies on specific components called “machine drivers”, in order to interface with the underlying cloud. The “machine drivers” are pluggable components. OpenStack is already supported. The FIWARE Lab is supported by using the OpenStack native driver, or by using the already mentioned Machine driver and User Interface Plugin for FIWARE Lab Nodes.



### 2.10.1 Resolve dependencies for docker-machine

docker-machine is a young and fast moving project. Chances are that your distribution is shipping and outdated version, if any. In order to satisfy the requirements, even from a stripped bare image.

➔ Install curl

```
sudo apt install --yes curl
```

### 2.10.2 Install the docker-machine

➔ You may want to have a look to the official documentation.

➔ Install docker-machine:

```
MACHINE_VERSION="v0.10.0"
MACHINE_BASE_URL="https://github.com/docker/machine/releases/download"
OS_NAME="$(uname -s)"
OS_ARCH="$(uname -m)"
MACHINE_URL="${MACHINE_BASE_URL}/${MACHINE_VERSION}/docker-
machine/${OS_NAME}-${OS_ARCH}"
curl -s -L "${MACHINE_URL}" \
  > /tmp/docker-machine && \
  chmod +x /tmp/docker-machine && \
  sudo cp /tmp/docker-machine /usr/local/bin/docker-machine
```

➔ Show the docker machine version:

```
docker-machine version
```

### 2.10.3 Start with a clean docker-machine environment

This may cause data loss! Do the following step only if you want to destroy all the hosts created with docker-machine.

➔ Clean up local docker-machine:

```
docker-machine ls -q -f Name | xargs -trnl docker-machine rm --force
docker-machine ls
```

### 2.10.4 Setup docker-machine from the command line interface

Note that rancher (the base software of SmartSDK Platform) has some problems with docker machine. There is an open bug that always deletes OpenStack keypairs when a VM is removed with "docker-machine rm" (see #3261 and #7570).

➔ Set docker machine parameters:

```
export MACHINE_DOCKER_INSTALL_URL='https://releases.rancher.com/install-
docker/17.03.sh'
export MACHINE_DRIVER='openstack'
```

➔ Define the parameters related to the security groups previously defined in the section Add proper security group roles:

```
export OS_SECURITY_GROUPS='external,rancher-cluster'
```

➔ Define variables related to the underlying OpenStack installation. The following defaults are also used on most nodes of the FIWARE Lab nodes:

```
# Usually the name is 'default'
export OS_DOMAIN_NAME='default'
```

```
# This is the usual network names on the nodes of FIWARE Lab, check also
with
# openstack network list --column Name
export OS_NETWORK_NAME='node-int-net-01'
export OS_FLOATINGIP_POOL='public-ext-net-01'
```

- ➔ Define the variables relative to the images and flavor. Usually those are specific of a node. See List Available Images in an OpenStack Project and List Available Flavors in an OpenStack Project in order to find suitable values.

```
export OS_IMAGE_NAME='base_ubuntu_14.04'
export OS_FLAVOR_NAME='m1.small'
```

You may want to try different images, for example newer ubuntu or debian like base\_ubuntu\_16.04 or base\_debian\_8.

- ➔ The user used for ssh connection is image specific, usually it takes the name of the Linux distribution or your cloud provider should have some specific documentation.
- ➔ Usual value for ubuntu images:

```
export OS_SSH_USER='ubuntu'
```

- ➔ Usual value for debian images:

```
export OS_SSH_USER='debian'
```

- ➔ To create a specific keypair to OpenStack and tell docker machine to use it issue the following commands:

```
openstack keypair create --public-key "~/ssh/id_rsa_deployer.pub"
deployer
export OS_KEYPAIR_NAME="deployer"
export OS_PRIVATE_KEY_FILE=~/.ssh/id_rsa_deployer
```

- ➔ Or, on the contrary, to use automatically generated keys (a different one for each host), you may want to unset the specific environment variables:

```
unset OS_PRIVATE_KEY_FILE
unset OS_KEYPAIR_NAME
```

## 2.11 Setup security groups

- ➔ Allow access from external:

```
openstack security group create external \
  --description "Allow external access, for ssh, http, https, proxy"
openstack security group rule create external \
  --protocol tcp --dst-port 22:22
openstack security group rule create external \
  --protocol tcp --dst-port 80:80
openstack security group rule create external \
  --protocol tcp --dst-port 443:443
openstack security group rule create external \
  --protocol tcp --dst-port 8080:8080
```

- ➔ If the master is provisioned with docker-machine, you must also add a security rule for the docker-machine connection (it would be better to restrict a bit the source IP addresses):

```
openstack security group rule create external \
  --protocol tcp --dst-port 2376:2378
```

## 2.12 Download and install rancher-compose

To deploy docker-compose recipes (version 2) we need `rancher-compose`.

```
cd
wget -c https://releases.rancher.com/compose/v0.12.3/rancher-compose-linux-
amd64-v0.12.3.tar.gz
tar xvzf rancher-compose-linux-amd64-v0.12.3.tar.gz
cd rancher-compose-v0.12.3/
sudo mv --force rancher-compose /usr/local/bin/
rancher-compose --version
```

## 2.13 Download and install rancher CLI

To deploy docker-compose stack recipes (version 3) we need `rancher`. Version 3 recipes are not fully supported, but there is a known workaround. See Deployment using docker stack deploy.

```
cd
wget -c https://releases.rancher.com/cli/v0.5.1/rancher-linux-amd64-
v0.5.1.tar.gz
tar xvzf rancher-linux-amd64-v0.5.1.tar.gz
cd rancher-v0.5.1/
sudo cp rancher /usr/local/bin/
rancher --version
```

## 2.14 Create API keys

In order to access controlled environment you need to provide the API keys to `rancher` and to `rancher-compose`.

Note: the api key need to be related to the environment, event there is an account related api that may have more power (admin account).

If you delete the default enabled environment the first one will get the ID of 17a.

Complete with the values available in the rancher web interface the following variables:

## 2.15 Write CLI configuration

```
export RANCHER_URL=
export RANCHER_ENVIRONMENT=
export RANCHER_ACCESS_KEY=
export RANCHER_SECRET_KEY=
```

Use the variables to create a configuration file:

```
mkdir -p "${HOME}/.rancher"
cat <<EOF > "${HOME}/.rancher/cli.json"
{
  "accessKey":"${RANCHER_ACCESS_KEY}",
  "secretKey":"${RANCHER_SECRET_KEY}",
  "url":"${RANCHER_URL}/schemas",
  "environment":"${RANCHER_ENVIRONMENT}"
}
EOF
cat "${HOME}/.rancher/cli.json"
```

Now we are ready to use the command line client to send command to our rancher environment.

## 2.16 Provision of rancher hosts using machine drivers

The simplest way create and connect rancher hosts to the rancher server is to use the so called machine drivers.

The machine driver require some very specific parameters, most of the time you can supply the parameter by using environmental variables or command line parameters.

You should be able to find out the parameters from your OpenStack cloud provider by yourself by using the `openstackclient`.

The parameters are quite a few and the first time it is easy to get lost. Read this section carefully. We use a script to rename variables that do not need to be understood in order to use the machine drivers.

### 2.16.1 Set host parameters

Set Docker version:

```
export MACHINE_DOCKER_INSTALL_URL='https://releases.rancher.com/install-docker/17.03.sh'
```

To list the available files use the Google Cloud Storage API: `https://releases.rancher.com/?delimiter=&prefix=install-docker`

#### 2.16.1.1 Reasonable parameters for FIWARE Lab

A working environment for the Spain2 node of the Fiware Lab:

```
export MACHINE_DRIVER='fiwarelab'
export ENGINE_OPT='mtu=1400'
export FIWARELAB_DOMAIN_NAME='default'
export FIWARELAB_SEC_GROUPS='rancher-cluster,external'
export FIWARELAB_NET_NAME='node-int-net-01'
export FIWARELAB_FLAVOR_NAME='m1.small'
export FIWARELAB_IMAGE_NAME='base_ubuntu_16.04'
export FIWARELAB_IMAGE_NAME='base_debian_8'
export FIWARELAB_SSH_USER='debian'
export FIWARELAB_FLOATINGIP_POOL='public-ext-net-01'
```

➔ Export OpenStack variables to be seen by FIWARE Lab driver,

note that we are forced to do fancy stuff because of partial support for values with spaces by `env` and `set`:

```
NAMES_VALUES=$(
  env | \
  grep '^OS_' | \
  sed -e "s:=:=':" -e "s:$:':" `: # add quotes in dumb way` | \
  sed -e 's/^OS_/FIWARELAB_/` `: # rename the variables` | \
  tr '\n' ' '
)

eval export "${NAMES_VALUES}"
unset NAMES_VALUES

# If you have multiple regions active, you need to specify one, in
# order to not get the multiple possible endpoint match error
export FIWARELAB_REGION="Spain2"

set | egrep '^(OS|FIWARELAB)_'
```

### 2.16.1.2 Reasonable parameters for a generic OpenStack cloud

```
export MACHINE_DRIVER='openstack'
export OPENSTACK_NETWORK_NAME='vlab'
export OPENSTACK_FLAVOR_NAME='m1.small'
export OPENSTACK_IMAGE_NAME='GNU/Linux Ubuntu Server 16.04 x86_64 (Default
user: ubuntu)'
export OPENSTACK_SSH_USER='ubuntu'
export OPENSTACK_SEC_GROUPS='rancher-cluster,external'
export OPENSTACK_FLOATINGIP_POOL='vlab_external'
```

- ➔ Export OpenStack variables to be seen by rancher (see rancher bug #7647), note that we are forced to do fancy stuff because of partial support for values with spaces by env and set:

```
NAMES_VALUES=$(
  env | \
  grep '^OS_' | \
  sed -e "s:=:=':" -e "s:$:':" `: # add quotes in dumb way` | \
  sed -e 's/^OS_/OPENSTACK_/' `: # rename the variables` | \
  tr '\n' ' '
)

eval export "${NAMES_VALUES}"
unset NAMES_VALUES

# If you have multiple regions active, you need to specify one, in
# order to not get the multiple possible endpoint match error
export OPENSTACK_REGION="Spain2"

set | egrep '^(OS|OPENSTACK)_'
```

### 2.16.2 Create and access hosts

Before creating a host double check the configured rancher environment with `rancher config`

- ➔ Create host:

```
rancher hosts create rancher-node-01
```

- ➔ List the hosts status

```
rancher hosts ls
```

- ➔ Host, once up, is accessible via:

```
rancher ssh rancher-node-01
```

- ➔ NOTE: the rancher CLI has some minor annoyances. The `rm` subcommand is not scoped like the `create` one, so you need to issue the command as `rancher rm $HOSTID` instead of using the more human friendly name.

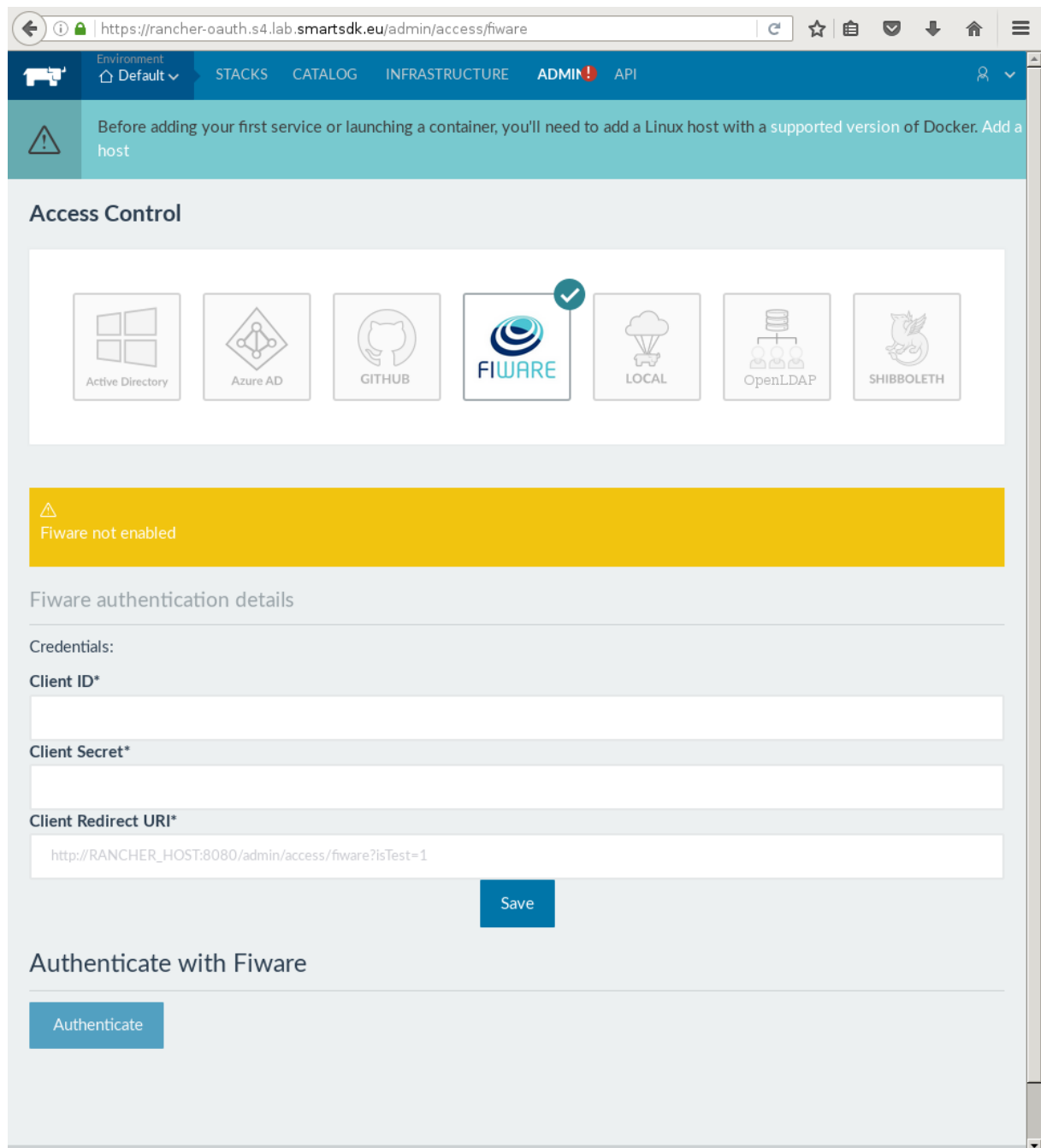
After adding at least 3 nodes (by default) or (change the number by following The Number of Swarm Managers) you should wait few minutes in order to give the cluster the time to startup.

Now you are ready to follow the detailed guide to **DEPLOY SMARTSDK RECIPES ON SMARTSDK PLATFORM**.

### 3 SMARTSDK PLATFORM ADVANCED USAGE

#### 3.1 User management integrated with FIWARE Lab OAuth

By using a custom build of rancher it is possible to use the OAuth authentication supplied by the FIWARE Lab. The use of the FIWARE Lab OAuth endpoint simplifies the user management on the Platform and offers an integrated user experience<sup>1</sup>.



**Figure 5: Screenshot of the Access control Setup for FIWARE Lab OAuth**

<sup>1</sup>This component is developed outside SmartSDK and are documented here for completeness.

### 3.2 Enable the FIWARE Lab Oauth

Unfortunately, to our knowledge, the FIWARE Lab Oauth does not support multiple source and callback URLs, which are required to register the application by using the rancher web interface. However, it is possible to complete the registration by using the rancher APIs by following the procedure described next.

- ➔ Remember the SmartSDK Platform Manager URL:

```
BASE_URL="https://rancher-oauth.s4.lab.smartsdk.eu"
```

- ➔ In the FIWARE Lab create a new application.

The mandatory data are, assuming the URL of the SmartSDK Platform Manager is `${BASE_URL}`

- ➔ URL: `${BASE_URL}/login`
- ➔ Callback URL: `${BASE_URL}/`

If you use the wrong URLs, the FIWARE Lab interface will show a warning and then an error and will not let you to complete the login procedure.

Take note of the generated credentials “OAuth2 Credentials”:

- ➔ Client ID
- ➔ Client Secret

Put the values in the following variables in order to use the subsequent code snippets:

```
CLIENT_ID=
CLIENT_SECRET=
```

- ➔ Install prerequisites:

```
sudo apt install --yes curl jq
```

- ➔ Prepare the json to create a new admin account keys:

```
cat <<EOD > post-apikey-data
{
  "type": "apikey",
  "accountId": "1a1",
  "name": "admin-api-name",
  "description": "admin-api-description",
  "created": null,
  "kind": null,
  "removeTime": null,
  "removed": null,
  "uuid": null
}
EOD
```

- ➔ Get the response and extract the Access Key (username) and Secret Key (password):

```
API_PATH="v2-beta/accounts/${ACCOUNT_ID}"
API_URL="${BASE_URL}/${API_PATH}"
RESPONSE="$(curl -s \
  'https://rancher-oauth.s4.lab.smartsdk.eu/v2-beta/apikey' \
  -H 'Accept: application/json' \
  -H 'Content-Type: application/json' \
  -d @post-apikey-data)"

CATTLE_ACCESS_KEY="$(printf '%s' "${RESPONSE}" | jq --raw-
output .publicValue)"
```

```
CATTLE_SECRET_KEY="$(printf '%s' "${RESPONSE}" | jq --raw-output .secretValue)"
```

- ➔ We inferred that the first person that login with the FIWARE OAuth will get the Account Id value of 1a7:

```
ACCOUNT_ID=1a7
```

- ➔ Prepare the json for the access control with FIWARE Lab:

```
API_PATH="v1-auth/config"
API_URL="${BASE_URL}/${API_PATH}"
REDIRECT_URI="${BASE_URL}/"

cat <<EOD > post-data-oauth
{
  "type": "config",
  "provider": "fiwareconfig",
  "enabled": true,
  "accessMode": "unrestricted",
  "allowedIdentities": [],
  "fiwareconfig": {
    "clientId": "${CLIENT_ID}",
    "clientSecret": "${CLIENT_SECRET}",
    "redirecturi": "${REDIRECT_URI}"
  }
}
EOD
```

- ➔ Enable the access control with FIWARE Lab:

```
curl -s \
-H "Content-Type: application/json" \
-d @post-data-oauth \
"${API_URL}"
```

- ➔ Now the access control is enable and it is possible to interact with rancher only after authenticating with the admin API keys or by using the FIWARE Lab.
- ➔ Login using the browser to confirm that the FIWARE Lab authentication works. Please note that is normal that the FIWARE Lab takes 10-15 seconds to reply in each step.
- ➔ Now, using the admin account keys we promote the first user as an admin:

```
API_PATH="v2-beta/accounts/${ACCOUNT_ID}"
API_URL="${BASE_URL}/${API_PATH}"
curl \
-u "${CATTLE_ACCESS_KEY}:${CATTLE_SECRET_KEY}" \
-X PUT \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
-d '{"kind": "admin"}' \
"${API_URL}"
```

- ➔ Using the web interface logout and login to see the new “ADMIN” tab.

### 3.3 Machine driver and User Interface Plugin for FIWARE Lab Nodes

The default graphical user interface of SmartSDK Platform Manager requires filling about 20 fields in order to start a docker swarm environment on a generic OpenStack project. The ICCLab (Cloud



Computing Lab) at ZHAW Zurich University of Applied Sciences developed a plugin to simplify the configuration, available on the [ui-driver-fiwarelab](#) project on gitlab<sup>2</sup>.

### 3.4 Using a VPN for overcoming NAT issues

It is possible to create an environment with rancher agents that are not associated to unique public IPs (i.e. connecting to remote rancher server from a natted network).

In order to satisfy the rancher requirement (every agent need to have a different IP) we will set up a VPN.

This unfortunately can not be easily automated with rancher machine drivers.

The overall procedure is the following:

- ➔ Install a VPN server in the same subnet of the rancher-master host (o even on the same host of the rancher-master). This host must be reachable from all the other hosts (rancher-master and rancher-agents).
- ➔ Start the VPN service.
- ➔ Join the VPN with rancher-master.
- ➔ Join the VPN with any other host that will become a rancher-agent.
- ➔ Start the rancher-agents as custom hosts (most of the time you will specify the private VPN ip as the "public IP" in the terminology of the rancher web interface, also known as CATTLE\_AGENT\_IP). Unfortunately the rancher web interface makes some confusion about the requirement: what is labeled as "public" needs only to be reachable and unique, not really "public".

One reasonable easy VPN service is n2n, for detailed information look at the [n2n howto](#).

The following snippet show an example installation:

```
# Define some useful variables
SUPERNODE_IP=203.0.113.1
RANCHER_MASTER_IP_ON_VPN=192.0.2.1
RANCHER_MASTER_PORT=443
MTU=1300
VPN_PORT=1194
# Infer RANCHER_HOST_ENV_TOKEN from the long command line interface
# from the rancher add host interface (it is the same for all the hosts)
RANCHER_HOST_ENV_TOKEN=
# install n2n
sudo apt install n2n
# start the server on port 1194
sudo supernode -l "${VPN_PORT}"
# The NODE_IP_ON_VPN must be different for each host and for the
# rancher-master the same of RANCHER_MASTER_IP
NODE_IP_ON_VPN=192.0.2.2
# Set up a long enough shared secret
SHARED_SECRET="REPLACE ME WITH A LONG ASCII TEXT"
# on the rancher master and on each rancher-agents node join the server
nohup sudo edge -c vpn4rancher -d vpn4rancher -k "${SHARED_SECRET}" \
  -l "${SUPERNODE_IP}:${VPN_PORT}" -M "${MTU}" -a "${NODE_IP_ON_VPN}" &
# on each node join the rancher-server (with a modified cut and paste
# from the rancher add host interface)
sudo docker run -e CATTLE_AGENT_IP="${NODE_IP_ON_VPN}" -d --privileged \
```

<sup>2</sup>This component is developed outside SmartSDK and are documented here for completeness.

```
-v /var/run/docker.sock:/var/run/docker.sock \  
-v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.1 \  
"http://${RANCHER_MASTER_IP}:${RANCHER_MASTER_PORT}/v1/scripts/${RANCHER_HOST_ENV_TOKEN}"
```

### 3.5 Provision of rancher hosts using custom hosts

This is the procedure to use when you are not able to use machine drivers, for example for lack of public IPs.

“Add hosts to rancher using custom hosts” allows to specify which IP addresses hosts has to use to communicate with each other.

This is done by specifying the `CATTLE_AGENT_IP` environment variable when launching the rancher/agent container.

It seems not possible by now to pass this env variable to the rancher agent container launched via machine drivers: <https://github.com/rancher/go-machine-service/blob/master/handlers/bootstrap.go>

In particular launching with a `CATTLE_AGENT_IP` reachable from all hosts in the environment is needed when they are behind a NAT (not exposed with a public IP). Otherwise they will pick the IP detected by the server, which will likely be an IP shared with other hosts of the NAT. This in turn doesn't allow the ipsec inter-host networking to work properly.

## 4 DEPLOY SMARTSDK RECIPES ON SMARTSDK PLATFORM

Note that there are MTU issues with swarmkit in the FIWARE lab. See Advanced analysis of the issues related to non-standard MTU usage.

### 4.1 Deployment using rancher-compose

Rancher compose allows to run applications services specified in docker-compose version 2 files. currently docker-compose v3 files are not supported directly, but for a workaround see next section.

To deploy services specified in a `docker-compose.yml` file, move in the same directory and run

```
rancher-compose up -d
```

This will create a stack in rancher containing the started services.

A `rancher-compose.yml` file format can be used to specify deployment rules (such as scalability parameters).

The documentation is still in early phases (see the bug report #4657). The file schema can be inferred by looking at the code.

The upstream documentation is available.

Look at the examples in the next sections for a guidance in order to have an application deployed.

### 4.2 Deployment using docker stack deploy

When an environment is managed in docker swarm mode (docker  $\geq$  1.12), the application deployment can be managed by passing docker-compose v3 files to a swarm manager node.

The latest rancher server (v1.6.0) doesn't provide an easy way to do this. A multi-step procedure follows.

1. Be sure to have docker 1.13.x or 17.03.0-ce installed on your host.
2. Find a node labeled as manager:

rancher hosts				
ID	HOSTNAME	STATE	CONTAINERS	IP
LABELS				
1h11	rancher-node-01.novalocal	active	14	192.168.242.90
swarm=manager				
1h12	rancher-node-02.novalocal	active	12	192.168.242.91
swarm=manager				
1h13	rancher-node-03.novalocal	active	12	192.168.242.93
swarm=manager				
1h14	rancher-node-04.novalocal	active	12	192.168.242.92

3. Ask a manager to deploy the application:

```
rancher --host 1h11 docker stack deploy --compose-file docker-compose.yml
testApp
```

Note that the deployed application stacks will not be seen nor managed by rancher as stacks but only as independent containers.

Look at the examples in the next sections for a guidance in order to have an application deployed.

## 4.3 Test deployments

### 4.3.1 Swarm deployment with replication (docker stack)

The following docker-compose creates 3 replica containers, reachable from any of the swarm nodes.

The `docker-compose.yml`:

```
cat <<EOF > docker-compose.yml
version: '3'
services:
  whoami:
    image: jwilder/whoami

    ports:
      - "80:8000"

    deploy:
      replicas: 3
networks:
  default:
    driver: bridge
    driver_opts:
      com.docker.network.driver.mtu: \${DOCKER_MTU:-1500}
EOF
```

To launch the application follow the example:

```
# We need to set the variable for this recipe (default is 1500)
export DOCKER_MTU=1400
# Just a trick to find the first available manager
SWARM_MGR=$(rancher hosts ls | awk '/swarm=manager/ { print $1; exit}')
export SWARM_MGR
rancher --host "${SWARM_MGR}" docker stack deploy --compose-file docker-
compose.yml whoami
```

### 4.3.2 Rancher compose example from catalog (community-recipes)

The following example will deploy a wordpress instance.

The template from the catalog must be completed with some answers at runtime.

```
export DOCKER_MTU=1400
mkdir -p /tmp/recipes/
cd /tmp/recipes/
git clone https://github.com/rancher/community-catalog.git
cd community-catalog/swarm-templates/wordpress/0/
ls
cat docker-compose.yml
# NOTE: interactive begin
rancher-compose up -d
rancher-compose rm
# NOTE: interactive end
```

### 4.3.3 Swarm deployment example (smartsdk-recipes)

Replicated deployment using swarm orchestration inside rancher

```
# We need to set the variable for this recipe (default is 1500)
export DOCKER_MTU=1400
mkdir -p /tmp/recipes
```

```
cd /tmp/recipes
git clone https://github.com/martel-innovate/smartsdk-recipes.git
cd smartsdk-recipes/recipes/utils/mongodb/replica
cat docker-compose.yml
# Find a swarm manager node registered with rancher
rancher host
# Just a trick to find the first available manager
SWARM_MGR=$(rancher hosts ls | awk '/swarm=manager/ { print $1; exit}')
export SWARM_MGR
rancher --host "${SWARM_MGR}" docker stack deploy --compose-file docker-
compose.yml mongoReplica
```

#### 4.3.4 Rancher compose example (fiware tour demo app)

The following example tries to deploy the FIWARE TourGuide-App.

Containers requiring volumes from others (`volumes_from`) is not yet supported and will be deprecated in docker-compose v3. See also the bug.

```
export DOCKER_MTU=1400
mkdir -p /tmp/recipes
cd /tmp/recipes
git clone https://github.com/Fiware/tutorials.TourGuide-App.git
cd tutorials.TourGuide-App
ls
cat docker-compose.yml
rancher-compose up -d
rancher-compose stop
rancher-compose rm
```

You now should be familiar enough with `docker-compose.yml` to change the file to work with rancher.

## 5 SMARTSDK PLATFORM INSTALLATION

We will start with a host with a docker installation  $\geq 1.13$ . Then we will install `docker-compose` and deploy the `rancher-master`.

### 5.1.1 Install docker-compose

- ➔ See the docker compose official documentation.
- ➔ For your convenience see also how to "add the docker group to the current user".
- ➔ Install docker compose:

```
COMPOSE_VERSION=1.11.1
COMPOSE_BASE_URL="https://github.com/docker/compose/releases/download"
OS_NAME="$(uname -s)"
OS_ARCH="$(uname -m)"
COMPOSE_URL="${COMPOSE_BASE_URL}/${COMPOSE_VERSION}/docker-compose-
${OS_NAME}-${OS_ARCH}"
sudo curl -s -L "${COMPOSE_URL}" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

### 5.1.2 Set some useful variables

You want to set the `RANCHER_HOSTNAME` to a fully qualified host name. If it is reachable from the Internet it will get a proper SSL certificate signed by letsencrypt. For testing purposes you can use `http://xip.io/` or `http://nip.io/` with a public floating IP.

Note that by using those test services you may not be able to get certificates because of a rate/total certificate limit on letsencrypt:

- ➔ ACME server returned an error: `urn:acme:error:rateLimited :: There were too many requests of a given type :: Error creating new cert :: Too many certificates already issued for: nip.io`

Set rancher version, server host name, email where to send certificate renewal alerts and MTU:

```
export RANCHER_VERSION="v1.6.0"
export RANCHER_MASTER_INTERNAL_IP="192.0.2.1"
export RANCHER_MASTER_FLOATING_IP="203.0.113.1"
export DOMAIN="example.com"
export RENEWAL_EMAIL="user@example.com"
export RANCHER_HOSTNAME="rancher-master.s1.lab.smartsdk.eu.example.com"
export DOCKER_MTU="1400"

# To use with the browser
export RANCHER_URL="https://${RANCHER_HOSTNAME}"

# To use for the register host setting in rancher admin panel
export RANCHER_HOST="http://${RANCHER_MASTER_INTERNAL_IP}:8080"

echo ${RANCHER_URL}
echo ${RANCHER_HOST}
```

See more about `RANCHER_HOST` in section "Set registration URL".

Create the docker compose to deploy a rancher server accessible through a TLS termination proxy:

```
cat <<EOF > docker-compose.yml
version: '2'
```

```

services:
  nginx-proxy:
    container_name: tls-proxy
    image: jwilder/nginx-proxy
    environment:
      - DEFAULT_HOST=${RANCHER_HOSTNAME}
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock:ro
      - /srv/conf/nginx-proxy/certs:/etc/nginx/certs:ro
      - /srv/conf/nginx-proxy/vhost.d:/etc/nginx/vhost.d
      - /srv/conf/nginx-proxy/httpdocs:/usr/share/nginx/html

  lets:
    container_name: letsencrypt
    image: jrcs/letsencrypt-nginx-proxy-companion
    volumes_from:
      - nginx-proxy
    volumes:
      - /srv/conf/nginx-proxy/certs:/etc/nginx/certs:rw
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /srv/conf/nginx-proxy/vhost.d:/etc/nginx/vhost.d
      - /srv/conf/nginx-proxy/httpdocs:/usr/share/nginx/html
    depends_on:
      - nginx-proxy

  rancher:
    container_name: rancher-server
    image: rancher/server:${RANCHER_VERSION}
    ports:
      - "8080:8080"
    environment:
      - VIRTUAL_HOST=${RANCHER_HOSTNAME}
      - VIRTUAL_PORT=8080
      - LETSENCRYPT_HOST=${RANCHER_HOSTNAME}
      - LETSENCRYPT_EMAIL=${RENEWAL_EMAIL}
      - LETSENCRYPT_TEST=true
    volumes:
      # TODO: something seems wrong here, why mount a local lib into
      # the container?
      - /var/lib/mysql:/var/lib/mysql:rw

networks:
  default:
    driver_opts:
      com.docker.network.driver.mtu: ${DOCKER_MTU}
EOF

```

Launch server and proxy:

```
docker-compose up -d
```

Look at the log:

```
docker-compose logs rancher
```

## 5.2 Configure Access Control

Once the rancher-master is started, the first thing to do is to setup the access control. This usually

means to set the administrative user and password.

It is also possible to set the User management integrated with FIWARE Lab Oauth.

For additional advanced configuration see the access control documentation.

### 5.3 Set registration URL

The registration URL is needed by hosts to register with the server.

In order for the overlay network to work, the hosts has to be mutually reachable using the IPs as collected by the rancher server (shown by issuing `rancher hosts`).

This is needed because the default network plugin establishes direct IPsec connections between hosts, using the IP addresses collected by the server during host registration. If those addresses are not reachable by a host, the hosted containers cannot communicate with containers hosted in other hosts.

This is the case when many hosts register themselves using the same IP address (i.e. when they are behind a NAT). The source IP collected by the server will not enable hosts to see each other using that address.

TODO: we use the remote host name `RANCHER_HOST` to overcome the 80 to 443 redirect done by the reverse proxy.

Rancher allow private IP, but seems to discourage them, as the tool tip say:

Are you sure all the hosts you will create will be able to reach `http://192.168.243.107:8080`? It looks like a private IP or local network.

See also the detailed documentation on host registration.

### 5.4 (Optional) Configuration of the NFS server

➔ A reasonable starting point for a NFS setup (NFSv4 should be enabled by default):

```
sudo apt install --yes nfs-kernel-server
sudo mkdir -p /srv/export
echo '/srv/export *(rw,no_root_squash,no_subtree_check)' | sudo tee -a
/etc/exports
sudo exportfs -rav
sudo systemctl restart nfs-kernel-server || sudo /etc/init.d/nfs-kernel-
server restart
systemctl status nfs-kernel-server || sudo /etc/init.d/nfs-kernel-server
status
# check that the correct port is bounded
ss -ltn | grep -q :2049 && echo OK || echo KO
# TODO NOTE Take care of the internal ip for the nfs configuration for
the rancher env
ip addr show dev eth0 | grep 'inet ' | tr '/' ' ' | awk '{print $2}'
```

➔ Please remember that the NFS server address and export path must be kept in sync with the environment configuration.

### 5.5 Enable the catalog in the web interface

➔ Enable the catalog

Catalogs contain templates that allow users to easily deploy applications by answering a few questions



## 5.6 Import environment templates

You may want to provide already customized environment templates to the user. This example show how to add an Environment with the NFS storage configured.

➔ Environment templates can be defined as:

```
export DOCKER_MTU=1400
export NFS_SERVER_ADDR=10.0.0.2

cat <<EOF > swarmNFS.yaml
description: Swarm with NFS template
isPublic: "true"
name: SwarmNFS
stacks:
- name: healthcheck
  template_id: library:infra*healthcheck
- name: network-services
  template_id: library:infra*network-services
- name: portainer
  template_id: community:infra*portainer
- answers:
  DOCKER_BRIDGE: docker0
  MTU: "$DOCKER_MTU"
  RANCHER_DEBUG: false
  SUBNET: 10.42.0.0/16
  name: ipsec
  template_id: library:infra*ipsec
- answers:
  MOUNT_DIR: /srv/export
  MOUNT_OPTS: ""
  NFS_SERVER: $NFS_SERVER_ADDR
  NFS_VERS: nfsvers=4
  RANCHER_DEBUG: "false"
- name: scheduler
  template_id: library:infra*scheduler
- name: swarm
  template_id: community:infra*swarm
EOF
```

➔ then imported with:

```
rancher env template import swarmNFS.yaml
```

It seems that the `isPublic` setting is not saved in the environment. It has to be set later manually from the web interface.

## 5.7 Create environments based on templates

➔ Example creation of a new environment called `testEnv` using the `swarmNFS` template:

```
rancher env create -t swarmNFS testEnv
```

## 5.8 Activate FIWARE Lab machine driver

To install the `fiwarelab` driver on the `rancher` server follow the instructions at: Machine driver and User Interface Plugin for FIWARE Lab Nodes.

## 5.9 Activate openstack machine driver

You can use the Web UI or the following CLI snippet.

```
# NOTE: this only works when access control is disabled
# OpenStack driver ID is "lmd9"
export DRIVER_ID='lmd9'
curl -X POST ${RANCHER_URL}/v2-beta/machinedrivers/${DRIVER_ID}/?action=activate
```

## 5.10 Upgrade Rancher Server

### 5.10.1 Notes based on the guide

<https://docs.rancher.com/rancher/v1.6/en/upgrading/#single-container>

### 5.10.2 Setup some variables for the new version

```
docker-machine ssh rancher-server
NEW_SERVER_VERSION=v1.6.0
```

### 5.10.3 Stop the running server

```
# Find the server container name
ORIGINAL_SEVER_CONTAINER="$(docker ps | grep rancher/server: | awk '{print $1}')"
ORIGINAL_SEVER_VERSION="$(docker ps | grep -o 'rancher/server:[a-z0-9.-]*' | cut -d : -f2)"

echo $ORIGINAL_SEVER_CONTAINER
echo $ORIGINAL_SEVER_VERSION

docker stop "${ORIGINAL_SEVER_CONTAINER}"

docker ps
```

### 5.10.4 Data persistence, keep a reference

If there is no rancher\_data container, create it to keep a reference on the volumes

```
RANCHER_DATA_ID="$(docker ps -qa --no-trunc --filter name='^/rancher-data$')"
```

```
echo $RANCHER_DATA_ID
```

```
if [ -z "${RANCHER_DATA_ID}" ];
then
  echo creating a data container
  docker stop "${ORIGINAL_SEVER_CONTAINER}"
  docker create --volumes-from "${ORIGINAL_SEVER_CONTAINER}" \
    --name rancher-data rancher/server:"${ORIGINAL_SEVER_VERSION}"
fi
```

### 5.10.5 Do the upgrade

```
docker pull rancher/server:"${NEW_SERVER_VERSION}"
```

```
# run once the rancher server
docker run --volumes-from rancher-data --restart=unless-stopped \
  -p 8080:8080 rancher/server:${NEW_SERVER_VERSION}"
```

Upgrade the version in the `docker-compose.yml`.

```
sed -i.bak -e
"s~rancher/server:${ORIGINAL_SEVER_VERSION}~rancher/server:${NEW_SERVER_VER
SION}~" \
  docker-compose.yml
```

Restart the service as usual:

```
docker-compose up -d
```

### 5.10.6 Check the values

```
CURRENT_SEVER_CONTAINER="$(docker ps | grep rancher/server: | awk '{print
$1}')"
CURRENT_SEVER_VERSION="$(docker ps | grep -o 'rancher/server:[a-z0-9.-]*' |
cut -d : -f2)"

echo $CURRENT_SEVER_CONTAINER
echo $CURRENT_SEVER_VERSION

docker logs "${CURRENT_SEVER_CONTAINER}" 2>&1
```

## 6 CONCLUSION AND FUTURE WORK

---

The SmartSDK Platform Manager is already working, installed and properly configured on a testing project on the FIWARE Lab.

At the current state of the project is possible to use the SmartSDK Platform Manager to deploy SmartSDK recipes.

In order to provide an even better user experience and to add more features a number of enhancements need to be developed or integrated.

For the near future we will concentrate on the following points.

### 6.1 User Interface for docker-compose v3

The current user interface does not support the advanced features offered by docker-compose format v3. It only support a basic view of the running docker instances.

The portainer project should support this in the near future. Se the current open issue for further information.

### 6.2 User management integrated with FIWARE Lab for SmartSDK platform

We plan to automate the build and the documentation of the custom docker image presented in the section "User management integrated with FIWARE Lab Oauth".

### 6.3 Overlay network

To overcome the limitation of rancher networking we plan to explore and document other VPN alternatives: for example `freelan`, `vde2` and `gvpe`.

### 6.4 Persistent storage configuration

- ➔ There are also few discrepancies regarding the support of shared storage in docker compose v2 and v3 to be further investigated.
- ➔ The optional persistent storage configuration can be enhanced, for example by using a separate storage network.

### 6.5 SmartSDK Platform Manager in HA

Good production practices for cloud distributed application encourage the setup of highly available services. Currently our testing environment do not have the resources to offer HA, but we plan to overcome this issue in the near future.

## APPENDIX A    ADVANCED FUNCTIONALITIES

### A.1 How Docker Swarm networking works

Docker swarm allows to create overlay networks that connect containers on different swarm nodes.

The default swarm network driver uses VXLAN secured with point-to-point IPSEC tunnels to provide L2 networking among containers. IPsec requires that nodes can directly reach each other with UDP traffic (no natting).

### A.2 How exposing services through load balancers works

Docker swarm uses internal load balancers that expose ports on the swarm nodes.

Incoming requests on swarm nodes are forwarded by the node load balancer to one of the replicated containers through the swarm ingress network.

The incoming request will always be forwarded to a running container, even those who arrives on nodes on which the container is not running.

### A.3 Issues with rancher hosts for natted machines

It is possible to use rancher machine drivers to start virtual machines that, after the setup are not reachable from within the rancher overlay network.

For example, if you do not have any more free floating IP you can still start virtual machines with the openstack driver by unsetting `OPENSTACK_FLOATINGIP_POOL`:

```
unset OPENSTACK_FLOATINGIP_POOL
```

then creating new hosts:

```
rancher hosts create rancher-node-02
rancher hosts create rancher-node-03
```

and list them:

```
rancher host
```

ID	HOSTNAME	STATE	CONTAINERS	IP
1h27	rancher-node-01.novalocal	active	14	
130.206.126.142	swarm=wait_leader			
1h30	rancher-node-02.novalocal	active	11	
130.206.122.186	swarm=manager			
1h31	rancher-node-03.novalocal	active	11	
130.206.122.186	swarm=manager			

Note that the two new host have the same IP, that is the IP used by OpenStack to do the NAT for hosts that do not have a floating IP. An SSH connection as well as the establishment of the overlay network with them is impossible.

```
rancher --debug ssh 1h30
```

```
ssh: connect to host 130.206.122.186 port 22: Connection refused
```

To overcome this issue it is possible to follow the guide at [Using a VPN for overcoming NAT issues](#).

## A.4 Advanced analysis of the issues related to non-standard MTU usage

The MTU for the network interfaces of the Spain2 FIWARE Lab VMs differs from the standard one of 1500 bytes.

This requires to explicitly specify it for every newly created network that uses the Linux bridge driver, otherwise packets could be corrupted by the network stack. Interfaces connected on a bridge need to have all the same MTU (see here).

The predefined `docker0` and `docker_gwbridge` are both affected, as they use the Linux bridge driver.

The MTU of the `docker0` bridge network can be set by passing the `--mtu=${DOCKER_MTU}` value to the docker daemon.

The `docker_gwbridge` bridge network used in swarm is also affected. It is used as the default gateway for containers created in swarm mode.

It is created automatically when the swarm is initialized and picks the default (non-configurable!) MTU.

The MTU can be set by (re)creating it with desired parameters before initializing the swarm node (see here):

```
docker network create -o com.docker.network.bridge.enable_icc=false \
-o com.docker.network.bridge.enable_ip_masquerade=true \
-o com.docker.network.driver.mtu=${DOCKER_MTU} \
docker_gwbridge
```

Also **every container** using bridge networks, has to be started by specifying the MTU to assign to the containers interfaces. For docker-compose v2 and v3 declare it in the networks section (see here):

```
networks:
  default:
    driver: bridge
    driver_opts:
      com.docker.network.driver.mtu: ${DOCKER_MTU}
```

To pass the MTU with docker-machine use the following snippet:

```
# Set the MTU equal to the one of the default gateway interface
export DOCKER_MTU=1400
docker-machine create rancher-server --engine-opt mtu="${DOCKER_MTU}"
```

Explicitly setting an MTU value for the docker bridge avoids network issues in case the default route has an MTU different from 1500 (see #22028 and Customize the docker0 bridge).

## A.5 Install modern openstackclient with pip

To install the openstackclient you need to satisfy some build dependencies. Please follow the following snippet.

```
# Enable the deb-src sources in /etc/apt/source.list
sudo sed -i.bak -e 's:# deb-src :deb-src :' /etc/apt/sources.list
sudo apt update
sudo apt install --yes virtualenvwrapper
sudo apt build-dep --yes python-openstackclient
sudo apt build-dep --yes python-netifaces

# There are issues if python-openstackclient==3.9.0 and
# python-novaclient==8.0.0 on #openstack the working suggestion by
# dtroyer was to
pip install python-novaclient==7.1.0
```

```
# and wait the fixing version python-openstackclient==3.10.0
. /etc/bash_completion.d/virtualenvwrapper
mkvirtualenv osclient
workon osclient
pip install python-openstackclient
```

## A.6 List Available Images in an OpenStack Project

```
openstack image list --column Name
```

## A.7 List Available Flavors in an OpenStack Project

```
openstack flavor list --column Name
```

## A.8 Useful script to manage Docker

➔ stop all docker containers

```
sudo docker ps -aq | xargs -tr sudo docker stop
```

➔ remove all docker containers

```
sudo docker ps -aq | xargs -tr sudo docker rm
```

➔ remove all docker volumes

```
sudo docker volume ls -q | xargs -tr sudo docker volume rm
```

## A.9 Change the MTU of all interfaces to 1400

```
ip addr show | grep ^[0-9] | awk '{print $2}' | tr -d : | xargs -I X sudo
ifconfig X mtu 1400
```

## A.10 Workaround for sudo complaint

Usually the hosts created in cloud environment do not have proper fully qualified host name. Sudo complains a bit with the following output:

```
# sudo: unable to resolve host $HOSTNAME
```

To fix the verbose sudo warning about the missing FQDN use the following:

```
echo 127.0.1.1 $(hostname) | sudo tee -a /etc/hosts
```

## A.11 Add the docker group to the current user

To avoid typing sudo every time before the docker command you may want to issue:

```
sudo usermod -aG docker "${USER}"
echo exit an login to make the group change effective
echo or launch another login shell
```

## A.12 Workaround to view display error on FIWARE Lab portal

In order to get the “Connect to VM display (view display)” working in the FIWARE Lab portal, you need to enable 3rd party cookies. If not, you will see the error: Failed to connect to

server (code: 1006).

## A.13 Rancher General cleanup

```
# Select your rancher host from docker swarm masters manually
RANCHER_HOST=1h3
rancher --host "${RANCHER_HOST}" docker ps -a
rancher --host "${RANCHER_HOST}" docker ps -a -q --filter status=created | \
    xargs -r rancher --host "${RANCHER_HOST}" docker rm
rancher --host "${RANCHER_HOST}" docker ps -a -q --filter status=exited | \
    xargs -r rancher --host "${RANCHER_HOST}" docker rm
rancher --host "${RANCHER_HOST}" docker ps -a
```