Complete Dynamic Multi-cloud Application Management

Project no. 644925

Innovation Action

Co-funded by the Horizon 2020 Framework Programme of the European Union

Call identifier: H2020-ICT-2014-1

Topic: ICT-07-2014 – Advanced Cloud Infrastructures and Services

Start date of project: January 1st, 2015 (36 months duration)

Deliverable D7.2

Overlay with focus on Component Manager

Due date:	31/10/2015
Submission date:	30/11/2015
Deliverable leader:	QSC AG
Editors list:	D. Hacker (QSC AG), José Aznar (i2CAT)

Dissemination Level

☒	PU:	Public
☐	PP:	Restricted to other programme participants (including the Commission Services)
☐	RE:	Restricted to a group specified by the consortium (including the Commission Services)
☐	CO:	Confidential, only for members of the consortium (including the Commission Services)

List of Contributors

Participant	Short Name	Contributor
Interoute S.P.A.	IRT	Domenico Gallico, Matteo Biancani
Sixsq SARL	SIXSQ	Charles Loomis, Rob Branchat
QSC AG	QSC	Doris Hacker, Maria Kurkouli
Technische Universitaet Berlin	TUB	Mathias Slawik, Ilke Zilci
Fundacio Privada I2CAT, Internet I Innovacio Digital A Catalunya	I2CAT	José Aznar, Eduard Escalona
Universiteit Van Amsterdam	UVA	Miroslav Zivkovic, Yuri Demchenko
Centre National De La Recherche Scientifique	CNRS	Christophe Blanchet, Mohamed Bedri

Change history

Version	Date	Partners	Description/Comments
V0.1	30/09/2015	QSC	First ToC
V5.0	05/11/2015	I2CAT	Final ToC
V6.0	11/12/2015	I2CAT	First integrated version of the document.
V7.0	19/11/2015	I2CAT	Second integrated version of the document.
V8.0	23/11/2015	QSC	Third integrated version of the document
V9.0	26/11/2015	I2CAT	Final version – ready for submission.
VFinal	30/11/2015	IRT	Final version submitted

Table of Contents

List of Figures

Executive Summary

This document is the accompanying release note for the first CYCLONE demonstrator set up, which was set as an internal milestone of the project and carried out during the General Assembly meeting in Berlin on 7-Oct-2015 (M10 of the project).

The focus of CYCLONE demonstration has been twofold. At first, we aimed at demonstrating the integration of specific security components with SlipStream Component Manager; these components are the Federated Identity Provider and the Distributed Logging System. Then, we demonstrated the integration of the Federated Identity Provider to extend two different applications: a common WordPress deployment and one of the CYCLONE Bioinformatics use cases (Securing human biomedical data) described in D3.1 [1]. SlipStream owns a predominant role in this first demonstrator acting as Component Manager.

1. Introduction

This document represents the release notes of the first CYCLONE demonstrator set up. The demonstrator was developed as an internal milestone of the project and achieved during the General Assembly meeting in Berlin on 7-Oct-2015 (M10 of the project). The demonstration has focused at first on the integration of specific security components with SlipStream Component Manager, and in particular we have worked on the Federated Identity Provider and the Distributed Logging System. Then, we demonstrated the integration of the Federated Identity Provider to extend two different applications: a common WordPress deployment and one of the CYCLONE Bioinformatics use cases (Securing human biomedical data) described in D3.1 [1].

This demonstrator extends the current functionalities of the involved CYCLONE software components. It also represents a step forward towards the demonstration of the project use cases in general and some specific CYCLONE functionalities in particular, thus advancing state of the art in current Cloud federated environments.

The document starts the discussion in section 2, focused on the SlipStream component deployment and required connectors to CYCLONE Cloud Service Provider (CSP) platform and infrastructures. SlipStream represents the visible face of the preliminary demonstrations and therefore there is a more exhaustive explanation on how it has been deployed to set up the demonstrator. Section 3 identifies the requirements of the testbed set up for the demonstration and discusses the integration of the network and security architecture components with SlipStream for the demonstrator. In section 4 we briefly discuss the functionalities that were demonstrated together with preliminary results. Finally, on section 5 we identify next steps in terms of "*how to extend current the current demonstrator performance, looking forward to progressively integrate additional functionalities to demonstrate the use cases*".

2. CYCLONE Component Manager: SlipStream

One of the main targets of the CYCLONE project is to provide to Application Service Providers (ASPs) with the means to facilitate deployment, management, and use of their complex, multi-cloud applications. SlipStream, a multi-cloud application deployment engine, was selected to be the core part of the CYCLONE toolset. Within CYCLONE framework, SlipStream takes the role of Component Manager. SlipStream, enables application developers to simplify and automate the deployment and management of applications on federated cloud infrastructures, and can allow to reduce costs and application provisioning times. The principal components of the SlipStream service are depicted in Figure 1, and briefly described in the following paragraphs.

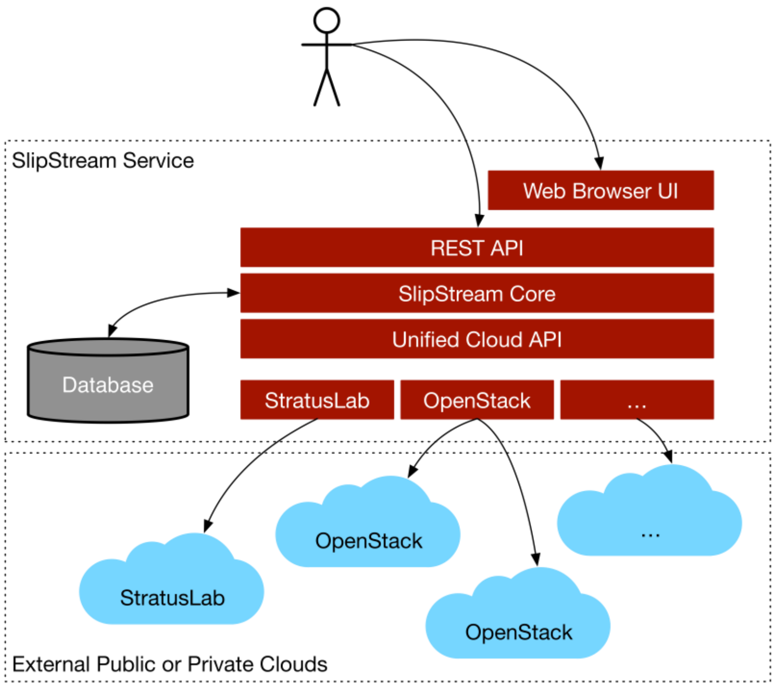


Figure 1: SlipStream Components

2.1. SlipStream general deployment over the testbed infrastructure

2.1.1. SlipStream connectors to the CSPs

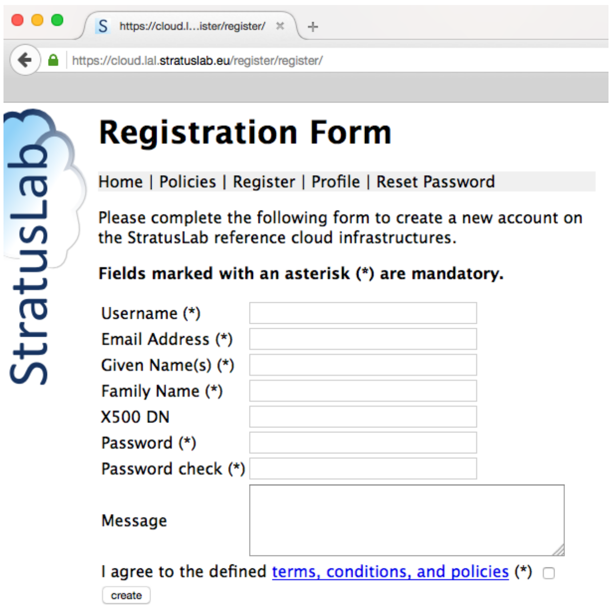
Although one of the goals of CYCLONE is to enable the deployment of multi-cloud applications, for this first demonstrator we mainly focused on the StratusLab cloud infrastructure that was available at CNRS-LAL, and in particular we used the StratusLab connector of SlipStream. As the project progresses, we will increase the number of SlipStream connectors in use in order to evolve towards a more extensive multi-cloud environment for future demonstrators.

2.1.2. CSP components be connected to SlipStream

The cloud infrastructure maintained at CNRS-LAL is subject to be used by registered users only by presenting valid credentials. The user registration at CNRS-LAL cloud is an automated process carried out by using the web page:

https://cloud.lal.stratuslab.eu/register/.

Filling the required fields on that page automatically sends an email to CNRS-LAL cloud administrator who allows or denies the registration. As soon as the user registration is accepted, the user can immediately start using CNRS-LAL cloud by configuring its own local client, as explained in the web page [2].


https://cloud.l...ister/register/
https://cloud.lal.stratuslab.eu/register/register/
StratusLab

Registration Form

Home | Policies | Register | Profile | Reset Password

Please complete the following form to create a new account on the StratusLab reference cloud infrastructures.

Fields marked with an asterisk (*) are mandatory.

Username (*)	
Email Address (*)	
Given Name(s) (*)	
Family Name (*)	
X500 DN	
Password (*)	
Password check (*)	
Message	

I agree to the defined terms, conditions, and policies (*) ☐

create

Figure 2: StratusLab account registration

Beyond CNRS-LAL cloud, users may have access to different cloud deployments. This is where SlipStream plays a fundamental role: by using SlipStream service, users automate application deployment on a multi-cloud environment. SlipStream usage has a different levels configuration model: cloud configuration, SlipStream configuration and user configuration.

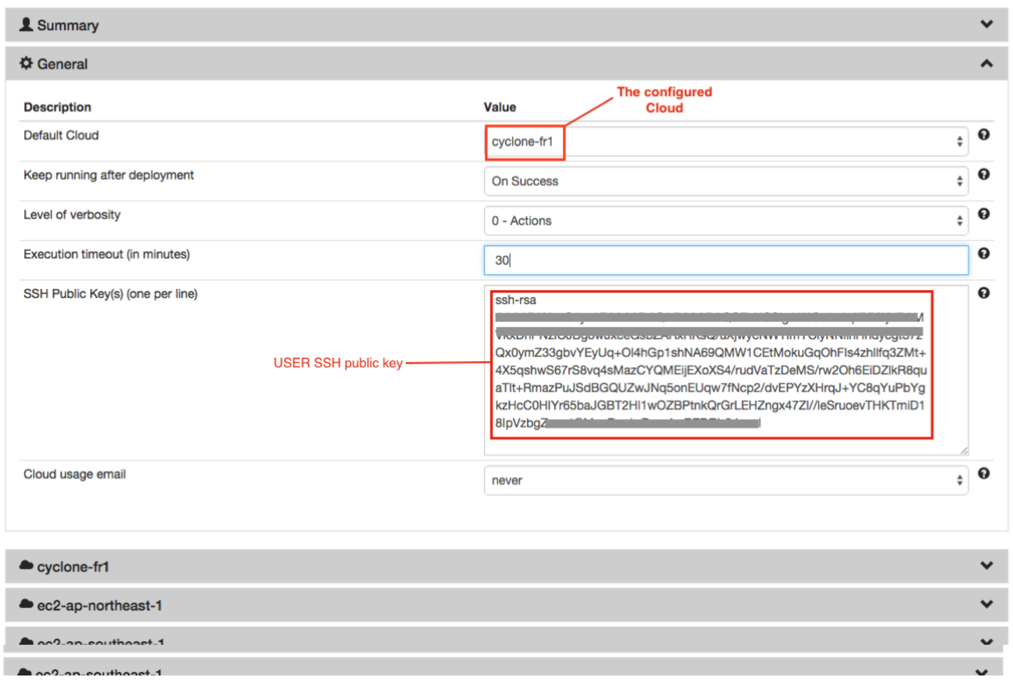
2.1.2.1. Cloud configuration

Once a user is registered, two different mechanisms can be used for authentication:

- Username/password pair generated by the StratusLab registration service
- Digital certificate issued by an IGTF accredited Certification Authority. In this case authentication can be done directly using this certificate or by generating a VOMS proxy certificate signed from the above.

Then, the user must install the StratusLab client tools that allow him to access remotely the cloud, control the lifecycle of Virtual Machine instances. The user has to edit the specific tool configuration file «/home/sluser/stratuslab/stratuslab.cfg » to add the mentions CNRS-LAL endpoints :

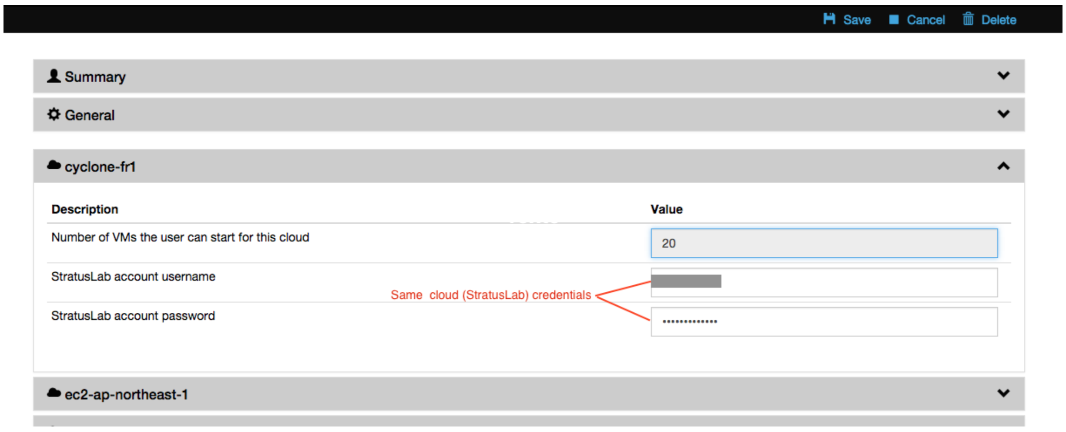
- endpoint = https://cloud.lal.stratuslab.eu/one-proxy/xmlrpc
- pdisk_endpoint = https://pdisk.lal.stratuslab.eu/pdisk
- marketplace_endpoint = https://marketplace.stratuslab.eu/marketplace



Description	Value
Default Cloud	cyclone-fr1
Keep running after deployment	On Success
Level of verbosity	0 - Actions
Execution timeout (in minutes)	30
SSH Public Key(s) (one per line)	ssh-rsa [redacted] Qx0ymZ33gbvYEyUq+Ol4hGp1shNA69QMW1CEtMokuGqOhFls4zhllfq3ZMt+ 4X5qshwS67rS8vq4sMazCYQMEijEXoXS4/rudVaTzDeMS/rw2Oh6EiDZlkR8qu aTlt+RmazPuJSdBGQUZwJNq5onEUqw7fNcp2/dvEPYzXHrqJ+YC8qYuPbYg kzHcC0HlYr65baJGBT2Hl1wOZBPtnkQrGrLEHZngx47Zl//leSruoevTHKTmiD1 8IpVzbgZ[redacted]
Cloud usage email	never

Figure 3: SlipStream User Authentication parameters

For more details, see the StratusLab user tutorial (http://stratuslab.readthedocs.org/en/latest/user-tutorial/index.html) or user guide (http://stratuslab.readthedocs.org/en/latest/user-guide/index.html).



Description	Value
Number of VMs the user can start for this cloud	20
StratusLab account username	[redacted]
StratusLab account password	•••••••••••••

Figure 4: SlipStream Cloud Configuration

2.1.2.2. *SlipStream user configuration*

The user must configure a SlipStream account by providing an email and password, also by adding a valid ssh public key.

Then, SlipStream service must be configured using at least one cloud credentials via a SlipStream connector of this cloud, these credentials depends of the CSP, but, generally, a cloud username (or an API key) and a password (or a secret key)

3. From demonstrator requirements to Slipstream integration

3.1. Testbed requirements

The objective of the CYCLONE testbed is to provide a platform that can be used for testing, improving the CYCLONE software components and report continuous feedback to the developers. Initially, the testbed consist of three distributed geographical sites. The distributed testbed, interconnected through the public internet includes computing, storage and network resources. The complexity of the underlying cloud infrastructure is hidden, and the developers and application service providers in CYCLONE have control over operating systems and deployed applications on demand that are accessed remotely via Internet. Furthermore, the CYCLONE testbed supports the development of an IaaS cloud architecture that works as a distributed federated cloud to address efficient multi-tenancy. Another purpose of the testbed is the use of network virtualization approaches to address network resources utilization.

Crucial parts of the testbed are the different components that compose CYCLONE software and that will be deployed over the testbed infrastructure. The integrated software components of the testbed provide the available services by CYCLONE such as application management services or authentication and federation services. The software needs to communicate and exchange information in order to accomplish the requirements of software developers to improve, test them and get continuous feedback.

The CYCLONE testbed is built up based on the requirements specification derived by the two use cases in combination with the requirements of software developers.

The creation of the CYCLONE testbed has been executed according to the following steps:

- Definition of the distributed local testbeds composing the overall physical infrastructure, including the complete specification of their capabilities (storage, network, etc.). As actual sites CNRS-LAL, Interoute and QSC are providing test cloud infrastructure.
- Identification of the interfaces (web and command-line client) that enable the interaction with cloud platforms.
- Deployment and configuration of the selected software to be used in the platform.
- Integration with different cloud providers.

The overall CYCLONE testbed infrastructure and the overall architecture is described in a previous deliverable D7.1 [3].

Figure 5 depicts the view of the CYCLONE testbed architecture. The three geographically distributed testbed located at CNRS/LAL, IRT and QSC AG sites compose the CYCLONE testbed infrastructure. The connection to GÈANT and other NRENs to extend the multi-domain scope of the testbed has been considered to be done in later phases of the project.

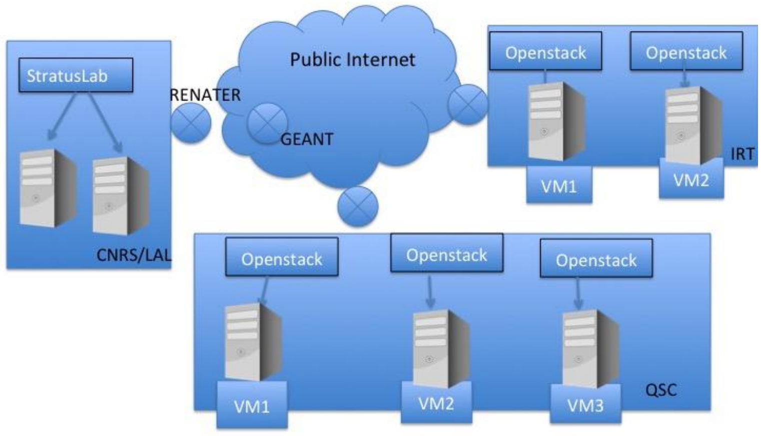


Figure 5: Testbed high level view

3.2. SlipStream integration with security architecture components

The demonstrator includes the deployment of two security architecture components: the CYCLONE federation provider and the CYCLONE distributed logging system. Both are deployed on local VMWare resources at TUB facilities, but can be easily deployed to other clouds, e.g., OpenStack or StratusLab. The public deployment descriptions can be found within Open Source Repositories on GitHub [4] and [5].

The CYCLONE Federation Provider is based on Keycloak [6] and offers federated login via EduGain service [7] into the connected applications. It employs the OpenID-Connect protocol, which combines OAuth, JSON Web Tokens (JWT) and a RESTful Authentication API.

The CYCLONE Distributed Logging is based on the ELK-Stack [11] (Elasticsearch [8], Logstash [9], Kibana3 [10]), a popular open source choice for logging persistence, middleware, and a dashboard. It is already integrated with the CYCLONE Federation Provider so that only people from the same *schacHomeOrganization* can access their tenant's logs. Currently, all developing partners are working on integrating their solutions with the Logging system, e.g. StratusLab and SlipStream. TUB has already integrated logging Kibana logins to the Distributed Logging.

There are two integration use cases between SlipStream and these security architecture components. Both use cases are described in the next subsections:

3.2.1. Deployment of security architecture components via SlipStream

There is ongoing work to make the security architecture components deployable via SlipStream. This is done through the creation of *Dockerfiles* and *Docker Compose compositions* for security components as well as using these Docker-based assets within SlipStream deployment recipes.

The results are published and updated continuously on the public CYCLONE GitHub repository [12] and the results of the demonstrations are presented in section 4.

3.2.2. Extension of SlipStream with security architecture functionality

In a second approach, SlipStream will be extended by functionality provided by the CYCLONE security architecture. This includes the extension of the SlipStream login service with OpenID-Connect in order to integrate the EduGain identities provided by the Federated Identity Provider as well as extending the SlipStream logging with remote logging to the distributed logging system.

4. First CYCLONE demonstrator

4.1. CYCLONE SlipStream Instance deployment

For this demonstrator we leveraged the SlipStream instance that was already available at CNRS-LAL premises in the same infrastructure as the targeted StratusLab resources. For this reason, there was no specific work needed to release and configure it.

In future demonstrators we will be using the "nuv.la" service, which is the publicly available SlipStream instance directly managed by SixSq. The required configuration changes and connectors for the CYCLONE testbed are already in place and applications are being migrated from the SlipStream service at CNRS-LAL to "nuv.la". Additionally, in the future the endpoint will be customized to allow a DNS name in the CYCLONE domain and to allow the project to use its custom branding.

4.2. Deployment of the Federated Identity Provider

The Federated Identity Provider is our solution to facilitate the integration of the educational identity providers with the cloud service providers on the basis of CYCLONE. The demonstration showcases CNRS, DFN-AAI (Deutsche Forschungsnetz Authentication and Authorization Infrastructure), TU Berlin and UvA identity providers. Cloud application users can authenticate to cloud services using their educational identities. Cloud service providers implement the integration only with the single endpoint (CYCLONE Keycloak) following the Open ID Connect Authorization Flow.

The Federated Identity Provider [4] can be deployed using the source provided above. It includes a *Dockerfile* for building the container, a README, as well as the configuration file (keycloak-export.json). The CYCLONE Federation Provider is basically a Keycloak installation, preconfigured in a docker container. There is more in-depth documentation available in the [13] and [14].

When built and run, the FP listeners on http://localhost:8080 by default. There are some preconfigured users and clients for testing purposes that are shown below.

4.2.1. Preconfigured Users

Keycloak can also be utilized as a standalone Identity Provider. The table below lists its preconfigured users which can be used for testing purposes by cloud service providers which are implementing the integration. Moreover, this capability enables CYCLONE to give access to cloud services by adding users, which do not have eduGAIN identities, directly to Keycloak.

Username	Password
admin	admin
owner	owner
user	user
guest	guest

4.2.2. Preconfigured OpenID-Connect Clients

The table below lists preconfigured OpenId-Connect Clients which can be used by cloud service providers before they complete the registration with the Federated Identity Provider to receive their dedicated client identities and redirect URIs.

Client ID	Allowed Redirect URIs
slipstream	*
portal	*
test	*

4.2.3. Authentication with keycloak

The authentication flow uses *OpenId-Connect* [15] as provided by Keycloak, specifically authentication using the authorization code flow, that can be found in [16]. Keycloak also supports authentication with SAML [17], if configured. The following numbered steps explain the protocol that Open ID Clients, namely the cloud service providers which want their users to authenticate via the Federated Identity Provider, need to implement:

1. Go or redirect user to: (optionally set kc_idp_hint to choose Idp for user, if brokered) /auth/realms/master/protocol/*OpenID-Connect*/auth?client_id=(client_id)&redirect_uri=(redirect_uri)&response_type=code
2. User logs into keycloak or a brokered idp and is sent to the redirect_uri with a code:(redirect_uri)/?code=(code)
3. Use this code to retrieve the actual json web tokens (jwts) by requesting:

```
POST /auth/realms/master/protocol/OpenID-Connect/token
Content-Type: application/x-www-form-urlencoded

Encode the following as application/x-www-form-urlencoded:
     grant_type : authorization_code
     code : (code)
     redirect_uri : (redirect_uri)
     client_id : (client_id)

Response:
{
    "access_token": "(base64-encoded jwt)",
    "expires_in": "(access_token expiration time)",
    "refresh_expires_in": "(refresh token expiration time)",
    "refresh_token": "(base64-encoded jwt)",
    "token_type": "bearer",
    "id_token": "(base64-encoded jwt)",
    ″not-before-policy″: ″(not-before-policy)″,
    "session-state": "(session-state)"
}
```

4. Some user information is included in the access_token and id_token already (if configured), some can be retrieved by requesting:

```
GET /auth/realms/master/protocol/OpenID-Connect/userinfo
Authorization: Bearer (access_token)

Response:
```

```
{
    ″email″: ″(email)″,
    "name": "(name)",
    "preferred_username": "(preferred_username)",
    ...
}
```

5. Use refresh_token to retrieve new tokens before expiration:

```
POST /auth/realms/master/protocol/OpenID-Connect/token
Content-Type: application/x-www-form-urlencoded

Encode the following as application/x-www-form-urlencoded:
     grant_type : refresh_token
     refresh_token : (refresh_token)
     redirect_uri : (redirect_uri)
     client_id : (client_id)
```

```
Response:
{
    "access_token": "(base64-encoded jwt)",
    "expires_in": "(access_token expiration time)",
    "refresh_expires_in": "(refresh token expiration time)",
    "refresh_token": "(base64-encoded jwt)",
    "token_type": "bearer",
    "id_token": "(base64-encoded jwt)",
    ″not-before-policy″: ″(not-before-policy)″,
    "session-state": "(session-state)"
}
```

6. To log out of the SSO session, point user's browser to http /auth/realms/master/tokens/logout?redirect_uri=(redirect_uri)

4.2.4. Screenshots of current deployment

On the screen shown in Figure 6, currently tested identity providers are listed. Users can log in to Keycloak directly or select one of the identity providers.

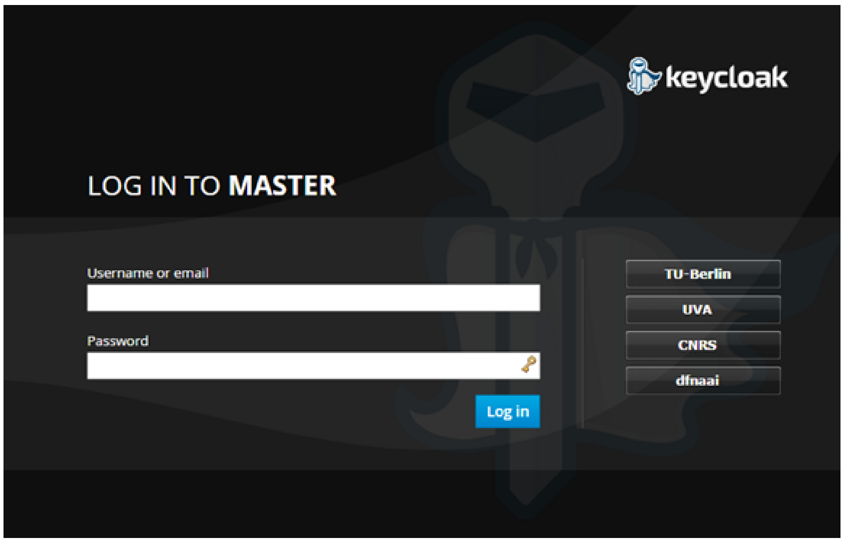


Figure 6: Login form showing currently configured Federated Identity providers.

4.3. Deployment of the Distributed Logging System

The repository [5] contains the cyclone logging components responsible for consolidating the various logs in the CYCLONE ecosystem, consisting of the ELK stack (Elasticsearch, Logstash, Kibana3) and a filter-proxy utilizing keycloak-nodejs [18] . Authentication is provided by the cyclone federation provider.

4.3.1. Configuration

- **Elasticsearch** can be configured by editing the files in config/elasticsearch/.
- The **Logstash** configuration file is in config/logstash. Further information regarding this file is provided on the Logstash homepage. Logstash supports various input, filter, codec and output plugins.
- The **cyclone logging filter proxy** can be configured by editing the source files in components/cyclone-logging-filter-proxy/ directly, specifically app.js and logging.js.
- **Kibana** is configured by editing kibana_config_template.js and kibana_dashboard_template.js in components/cyclone-logging-filter-proxy/.

4.3.2. How to run

Prerequisites: Create an *OpenID-Connect* client in the cyclone-federation-provider and export the client adapter configuration in the Keycloak JSON format to keycloak.json. Put this file into the cyclone-logging-filter-proxy directory.

Then deploy with docker-compose [19], i.e. docker-compose up.

4.4. Extension of WordPress by Federated Identity Provider Login

This is a demonstration for the integration of a widely used application, in this case WordPress, with the Federated Identity Provider of the CYCLONE project. It allows cloud application users to deploy a WordPress instance as a Service on Docker compatible clouds and log in using their eduGAIN identities via CYCLONE's Keycloak instance. It consists of a preconfigured WordPress instance with the Generic Open Id Connect Plugin and an installation script for customized images to configure WordPress by updating its tables at runtime. The Generic OpenID Connect Plugin [20] implements the Open ID Connect Authorization Code Flow.

The steps to achieve this are as follows: To start a container with defaults, on the project root run:

```
docker-compose build

docker-compose up -d
```

When cloud application users send a request to *wordpress-site-url/wp.login.php*, they are redirected to the Keycloak Federated Identity Provider. They can select the identity provider of their choice and log in. Keycloak returns the token to the WordPress instance and the Generic Open Id Connect Plugin creates a WordPress user for the eduGAIN identity.

To build the Docker image, the latest version of Docker and Docker Compose is needed. This image was tested with Docker 1.8.3 and Docker-Compose 1.4.2 on Ubuntu Trusty.

The WordPress configuration can be made in two steps by editing:

- the docker compose configuration file
  - docker-compose.yml if no external data volume is needed
  - docker-compose.yml to be able to edit the container's files outside the container
- the components/wordpress/wp-cli.yml file.

The configuration for the WordPress build in the docker-compose.yml must be consistent with the wp-cli.yml file:

```
wordpress:
  build: components/wordpress
  environment:
    - WORDPRESS_DB_PASSWORD=password
  links:
    - wordpressdb:mysql
  ports:
    - "80:80"
```

The Docker Compose syntax for the port binding is HOST:CONTAINER. With the configuration above, WordPress will run on port 80 on the container and can be accessed on the host machine on port 80 as well. The environment variable WORDPRESS_DB_PASSWORD must be the same as MYSQL_ROOT_PASSWORD, since root is the default user for the MySQL database.

Same values for the database and the port should be configured in wp-cli.yml. This file is used by the [wp-cli](http://wp-cli.org/) command line Wordpress management tool to be able to connect to WordPress' database to install the OpenId Connect Plugin and add its configuration. For example:

```
wordpress:
  ports:
    - "80:80"
```

requires

```
url: "http://localhost"
```

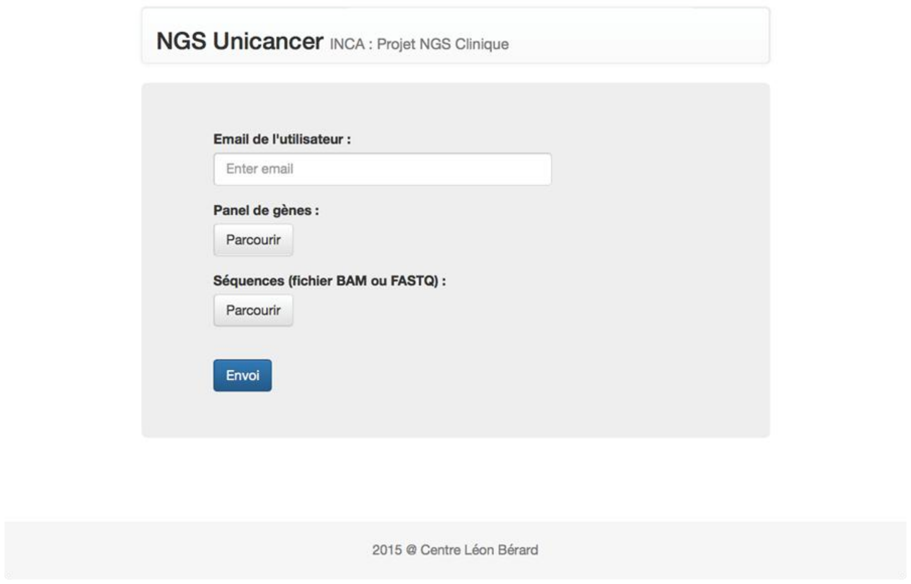
If cloud application users want to try out a local installation of WordPress with The Generic OpenID Connect Plugin installed, they can use CYCLONE Federated Identity WordPress Demo for Local Installation. This repository is a trimmed version of the WordPress in Docker Demo. The cloud application user copies the WP files to the server and adds the necessary tables to the MySQL database by restoring the MySQL dump (cyclone_wp_openid.sql).

4.5. Extension of UC1 - Securing human biomedical data analysis by Federated Identity Provider

Thanks to the steady drop of genome sequencing technology costs (NGS), an increasing number of clinicians are including biological results obtained with these technologies in their day-to-day diagnosis practice. Today, much genomics analyses are realized on the exome, which is the expressed part (5%) of the genome. However, the full genome sequencing is being envisaged and will be soon included in daily medical practices.

The UC1 - Securing human biomedical data is a single-VM deployment but with enhanced security features. The VM was instrumented with the Federation Provider and deployed on the testbed infrastructure (in CNRS LAL's site).

Some of the genomic data processed on the IFB cloud platform will concern human biomedical data and will thus be subject to strict privacy restrictions. The demo is done only on non-personally identifiable data (either anonymized benchmark data or simulated data): human data are extracted from existing public reference datasets such as the European Nucleotide Archive or the Ensemble Genomes resource. The cloud appliance NGS-Unicancer [21] is developed by the bioinformatics platform of the Centre Léon Bérard (Lyon, France) in the context of the project NGS-Clinique (INCA - Institut National du Cancer). It provides a simple web interface to launch the analysis pipeline (see Figure 7).


NGS Unicancer INCA : Projet NGS Clinique

Email de l'utilisateur :

Enter email

Panel de gènes :

Parcourir

Séquences (fichier BAM ou FASTQ) :

Parcourir

Envoi

2015 @ Centre Léon Bérard

Figure 7:Web interface of the NGS-Unicancer cloud appliance.

This bioinformatics pipeline get input of BAM [22] or FastQ [23] files (BAM and FAstQ are two file formats for short genomics sequences), some reference data (Human HG19 data) and produce as output a CSV file including mutations, aligned BAM file, FastQC. The average running time is 1h30.

The workflow for the demo scenario illustrated in Figure 8 is the following:

1. Deploy the appliance NGS-Unicancer through the IFB's web interface in one click (steps 1-3).
2. Mount automatically in the VM the community reference dataset (HG19).
3. Use the CYCLONE federation provider to manage the access to the VM web interface based on the user identity in the federation (step 4).
4. Upload user data with end-to-end security (step 5).
5. Provide biomedical staff with a simple web interface to analyse human genomics data (step 7).

The figure shows also the application components while describing the different steps of the workflow.

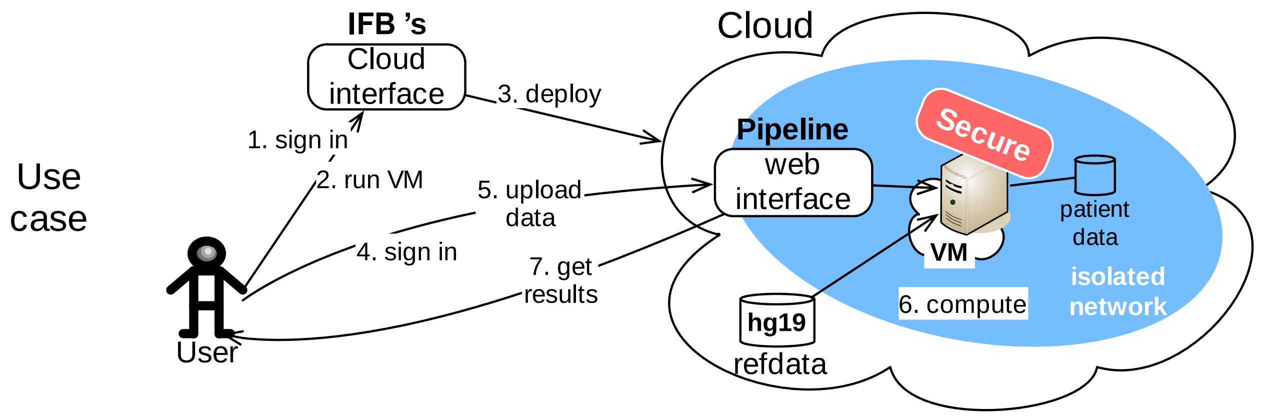


Figure 8: Functional schema of the use case "Securing human biomedical data".

4.6. Extension of UC2 - Cloud virtual pipeline for microbial genomes analysis with SlipStream deployment recipes

The team of the IFB-MIGALE platform (one of the bioinformatics platforms of the IFB) developed an environment for the annotation and visualization of microbial genomes. The platform automatically launches a set of bioinformatics tools to analyze the data and stores the results of the tools in a PostgreSQL relational database (See Figure 9). These tools use several public reference data collections. A web interface allows the user to consult the results, and perform the manual annotation (manual annotation means adding manually metadata and biological knowledge to the genome sequence). But installing the platform requires solid skills in system administration since many bioinformatics tools with different dependences, a relational database management system, a web server and servlet container, etc. must be installed.

The first tests were done with a generic bioinformatics appliance built previously by IFB (the appliance BIO compute node [24]), and requiring a cluster mode. BIO compute node is an image, based on centos, which provide a variety of bioinformatics tools highly used by life science researchers. The image was exported from the IFB's cloud and registered in the StratusLab Marketplace. Then a deployment recipe [25] based on SlipStream instantiates the complete application with all the required VMs on the testbed infrastructure in CNRS LAL's site. This was demonstrated in the F2F meeting in Berlin.

In SlipStream the project is called "*Bio_Compute*" and consists of:

- A typical Biocompute node
- A BioCompute_master, which has 4 parameters associated (flag_ready, id.dsa.node, instanceid and hostname) and a deployment recipe
- A BioCompute_slave, which has 3 parameters associated (id.dsa.node, instanceid and hostname) and a configuration recipes.
- A deployment named cluster_BioCompute that brings together a master and several slaves.

The configuration recipe of the master executes the following tasks:

1- Clean all the yum repositories,
2- Generate a password-less key and configure the ssh access between the master and the slaves,
3- Export the NFS share from the master to the slaves and open the iptables,
4- Configure the cluster messaging system.

Simultaneously the recipe of the slaves runs these instructions:

1- Clean all the yum repositories,
2- Generate a password-less key and configure the ssh access between the master and the slaves,
3- Modify the scripts ifb-qsub and ifb-worker with the IP address of the master,
4- Mount the master NFS share,
5- Run the ifb-worker command

The following figure shows the application components and the scenario of the usage of the federation provider to extend the UC 1 NGS biomedical analysis.

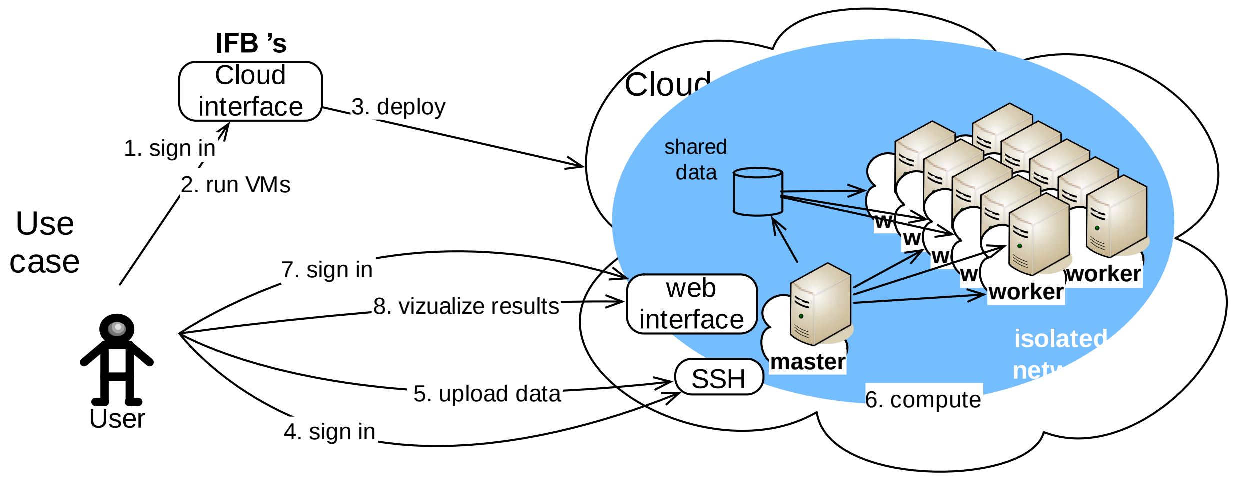


Figure 9: Functional schema of the use case "Cloud virtual pipeline for microbial genomes analysis".

5. Conclusions and Next Steps

The initial definition and setup of the CYCLONE testbed, has served as basis to deploy the first demonstrator. The testbed will progressively be developed and enhanced to be open for future requirements. The collected software tools are gradually being to improve the deployment of applications in hybrid cloud federated environments. SlipStream as Component Manager has resulted fundamental to deploy previous described components and demonstrate them.

5.1. Next Steps

For the next steps, the different options for collaboration with other European Cloud Projects will be evaluated to extend the experience and integrate into an overall testbed view. Additionally, it has been already identified a list of items to extend the current demonstrator performance, looking forward to progressively integrate additional functionalities and requirements to demonstrate the use cases.

5.1.1. SlipStream

Planned next steps in this area are:

- Integrating the Service Catalogue with the Open Service Compendium of TUB
- Implementation of a Brokering API within SlipStream supporting matchmaking of CIMI resources

5.1.2. CYCLONE Security Architecture

There are a number of tasks planned for Y2 and Y3 in implementing and demonstrating additional security architecture features, namely:

- Integrating SimpleSAMLphp into the Federation Provider for increased usability and to support all EduGain identity providers via a single SAML endpoint
- Implementation of federated authorization, e.g., through integration of XACML or OAuth 2.0
- Integration of secure end-to-end encryption technologies, such as the TUB-developed TCTP
- Offering the Federation Provider as a CYCLONE Service for the academic community

5.1.3. CYCLONE OpenNaaS network management platform components

At the current stage of the project, the inter-CSP network connectivity management has not been demonstrated, so that the integration between the OpenNaaS and the SlipStream has not yet been set up, but actions have been driven to develop APIs and connectors in order to integrate with CSP platforms. It is expected to incrementally extend the networking management options to the inter-CSP domain and make them available to the SlipStream component during the next phases of the project. More specifically:

- Enable with firewalling mechanism.
- VPN capabilities
- Load Balancing

- Guarantee QOS SLAs in multi-cloud application deployments
- Enable automatic DHCP configuration options
- Enable automatic DNS configuration options
- Dynamic network discovery (local and global cloud federation levels)

References

[1] CYCLONE deliverable D3.1: "Evaluation of Use Cases". Available at: http://www.cyclone-project.eu/deliverables.html

[2] http://stratuslab.readthedocs.org/en/latest/user-guide/03-client-install.html#installation.

[3] [1]CYCLONE deliverable D7.1: "Description of testbed". Available at: http://www.cyclone-project.eu/deliverables.html

[4] CYCLONE Federation Provider repository. Available at: https://github.com/cyclone-project/cyclone-federation-provider

[5] GitHub CYCLONE logging system repository. Available at: https://github.com/cyclone-project/cyclone-logging

[6] Keycloak. http://keycloak.jboss.org/

[7] EduGain. http://services.geant.net/edugain/Pages/Home.aspx

[8] Elasticsearch: https://github.com/elastic/elasticsearch

[9] Logstash: https://github.com/elastic/logstash

[10] Kibana3: https://github.com/elastic/kibana/tree/kibana3

[11] ELK stack. https://www.elastic.co/webinars/introduction-elk-stack

[12] CYCLONE GitHub public repository https://github.com/cyclone-project

[13] http://keycloak.jboss.org/docs

[14] http://docs.docker.com/

[15] http://openid.net/connect/

[16] http://openid.net/specs/*OpenID-Connect*-core-1_0.html#CodeFlowAuth

[17] http://saml.xml.org/

[18] https://github.com/keycloak/keycloak-nodejs

[19] https://docs.docker.com/compose/

[20] https://wordpress.org/plugins/generic-*OpenID-Connect*/

[21] http://marketplace.france-bioinformatique.fr:8081/metadata/IqDPi0GvoJOhFo00BqoHlqIVH40/Christophe.BLANCHET@france-bioinformatique.fr/2015-09-25T10:22:30Z

[22] https://genome.ucsc.edu/FAQ/FAQformat.html#format5.1

[23] https://en.wikipedia.org/wiki/FASTQ_format

[24] https://marketplace.stratuslab.eu/marketplace/metadata/KpuPQj5xVmBbMVf3rfflJXRLlh2/Christophe.BLANCHET@france-bioinformatique.fr/2015-09-17T13:22:15Z

[25] https://nuv.la/module/cyclone/BIO_Compute/1984