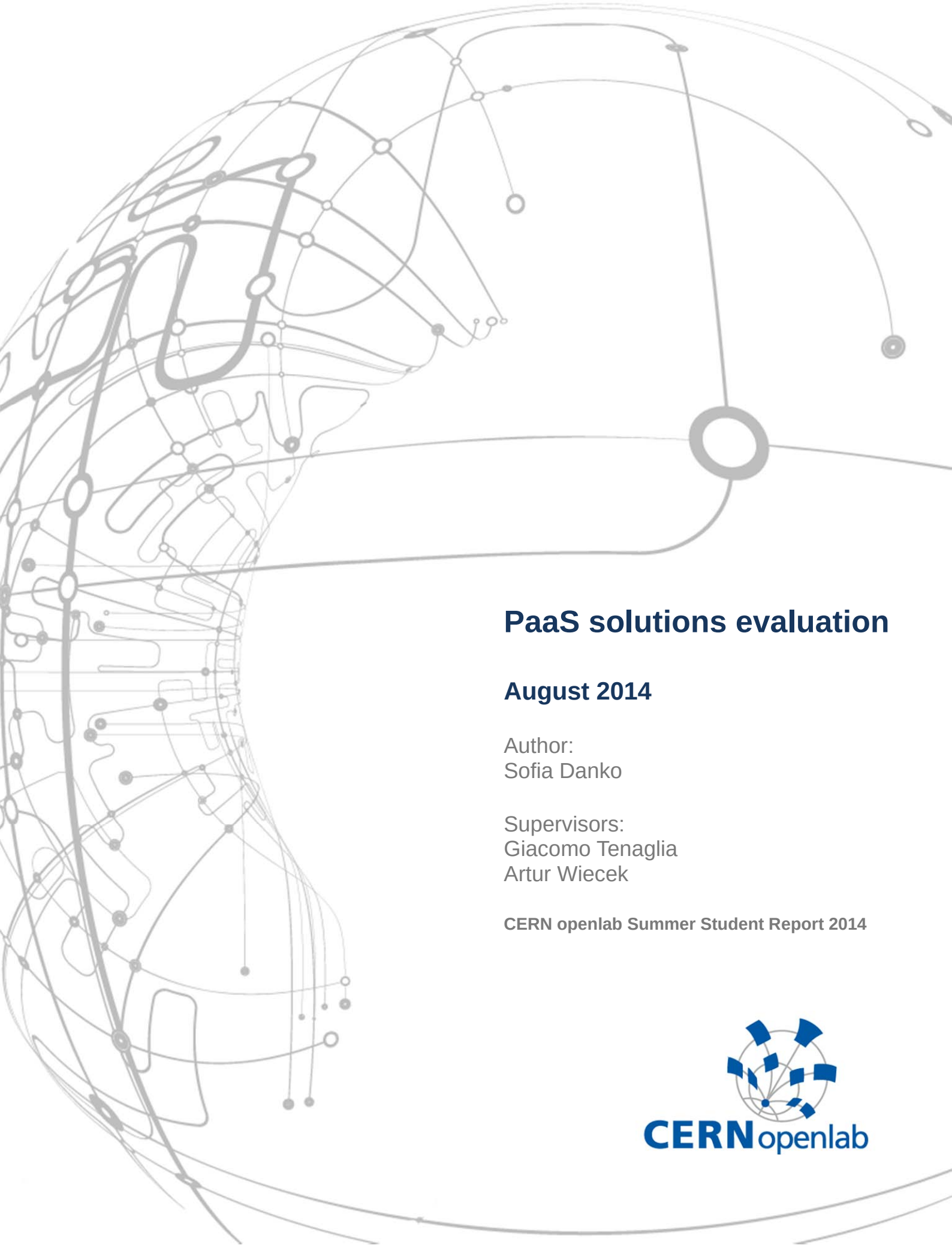

# PaaS solutions evaluation

## August 2014

Author:
Sofia Danko

Supervisors:
Giacomo Tenaglia
Artur Wiecek

**CERN openlab Summer Student Report 2014**

CERN openlab

# Project Specification

OpenShift Origin is an open source software developed mainly by Red Hat to provide a multi-language PaaS. It is meant to allow developers to build and deploy their applications in a uniform way, reducing the configuration and management effort required on the administration side.

The aim of the project is to investigate how to deploy OpenShift Origin at CERN, and to which extent it could be integrated with CERN "Middleware on Demand" service.

The student will be exposed to modern cloud computing concepts such as PaaS, and will work closely with the IT middleware experts in order to evaluate how to address service needs with a focus on deployment in production.

Some of the tools that are going to be heavily used are Puppet and Openstack to integrate with the IT infrastructure.

# Abstract

The report is a brief summary of Platform as a Service (PaaS) solutions evaluation including investigation the current situation at CERN and Services on Demand provision, homemade solutions, external market analysis and some information about PaaS deployment process.

This first part of the report is devoted to the current status of the process of deployment OpenShift Origin at existing infrastructure at CERN, as well as specification of the common issues and restrictions that were found during this process using different machines for test.

Furthermore, the following open source software solutions have been proposed for the investigation of possible PaaS provision at CERN:

- OpenShift Online;
- Cloud Foundry;
- Deis;
- Paasmaster;
- Cloudify;
- Stackato;
- WSO2 Stratos.

CERN 'homemade' solution called Middleware Manager Service was also examined as well as series of proprietary software to compare the features and possibilities of integration. The second part of report contains information about Cloud Foundry PaaS solution and about the process of its deployment at CERN.

# Table of Contents

# 1 Introduction

Cloud computing is a promising approach to the modern computing, comparatively new despite the fact that some of the experts [1] reckon that prerequisites for the cloud technology originated in 1950s. Nowadays the term “cloud computing’ refers to delivery of computing resources to the user on demand as a service rather than a product.

The basic components of cloud computing could be presented as so-called horizontal approach diagram:

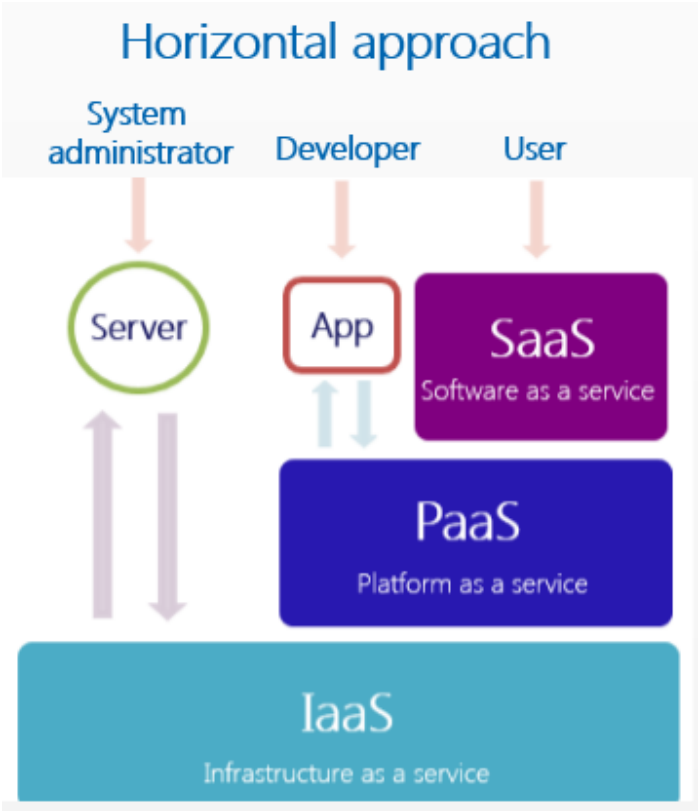


Fig. 1. Cloud computing horizontal approach

First layer is Infrastructure as a Service (IaaS) which basically means that external provider is responsible for computer resources, servers, OS, etc. CERN IT has implemented Openstack as IaaS, and it offers users to launch instances with preferred boot source and flavour in an easy and fast way. Using IaaS means that once this powerful tool works, developers don’t need to think about buying machines, servers, distribution, etc., which allows to concentrate more on development instead of maintenance.

Next level of abstraction is Platform as a Service, which could offer the same simplification to the app development. Basic development tools such as runtimes (Java, Ruby, Python, Perl, PHP, etc.), middleware (Jboss, Tomcat, etc.), frameworks (Django, Drupal, Flask, Rails, etc.) and services (MySQL, Postgresql, MongoDB, Jenkins, etc.).

Software is a service is a concept that helps developers to provide application software and databases on demand to the end user. Usually most of hardware and software maintenance is carried out as outsourcing.

An essential part of this report is dedicated to PaaS concepts as the next stage of cloud computing development at CERN.

# 2 CERN background

## 2.1 J2EE Public Service

J2EE Public Service [2] is a central server infrastructure for deployment of java server-side (servlet/jsp) applications at CERN started in 2007. As for the applications themselves, the target was medium-sized and relatively important, but non-mission-critical applications.

J2EE Public Service provided server-side infrastructure for deployment of java (servlet/jsp) web applications as well as:

- scalable server-side infrastructure,
- deployment mechanism,
- monitoring tools (currently available only to service managers),
- mechanism that will prevent an excessive use of resources that could result in the system going low on resources and crashing,
- procedures for easy reinstallation of the service in case of hardware failure,
- backup and recovery procedures,
- user documentation.

J2EE Public Service as a homemade solution was rethought and improved, and a transition to the Middleware Manager was started.

## 2.2 Middleware Manager

The Middleware Manager Service (MWMGR) [3] aims to provide facilities for managing Middleware Services offered by the Infrastructure & Middleware Services (IMS) section of the CERN IT/DB group and delivered to other CERN departments and/or users. Management of Middleware Services means:

- Definition of new Middleware entities (=instances) in a central repository
- Deployment (SW installation & configuration) of entities on target hosts
- Operations (start, stop, status, (re)configure…)
- Deletion (clean-up, de-registration)

**Types of servers**

There are 3 types of servers involved:

- Control machines: host the components involved in the business logic part of the platform:
    - Application Servers running the Manager Application
    - Web Servers serving the manager application
    - Daemon component
- Webtier machines: host the Web tier components serving users application
- Worker machines: host the Application Server components running users applications
- **Operating systems**

Operating systems supported for this project are SLC6/RHEL6 or newest releases.

- **Configuration Management**

All systems involved are managed by Puppet.

- **Special Users and Groups**

Users involved in this project are:

- sysctl:de: The Syscontrol user.
    - Used from MWMGR Control machines to run the Daemon Component and send Syscontrol commands to managed entities
- jps:jps: The Manager Application user.
    - Used from MWMGR Control machines to run the Manager Application (owner of the Container where application mwctl is deployed)

- jps001:2771 to jps999:2771: The User Applications users
    - Used from MWMGR workers machines to run the Users Applications (owners of the Containers where users applications are deployed)

# 3 OpenShift solutions providing PaaS

## 3.1 OpenShift Origin

### 3.1.1 Overview

OpenShift Origin is the open source upstream of OpenShift, the next generation application hosting platform developed by Red Hat. OpenShift takes care of infrastructure, middleware, and management so that developers can focus on their apps.

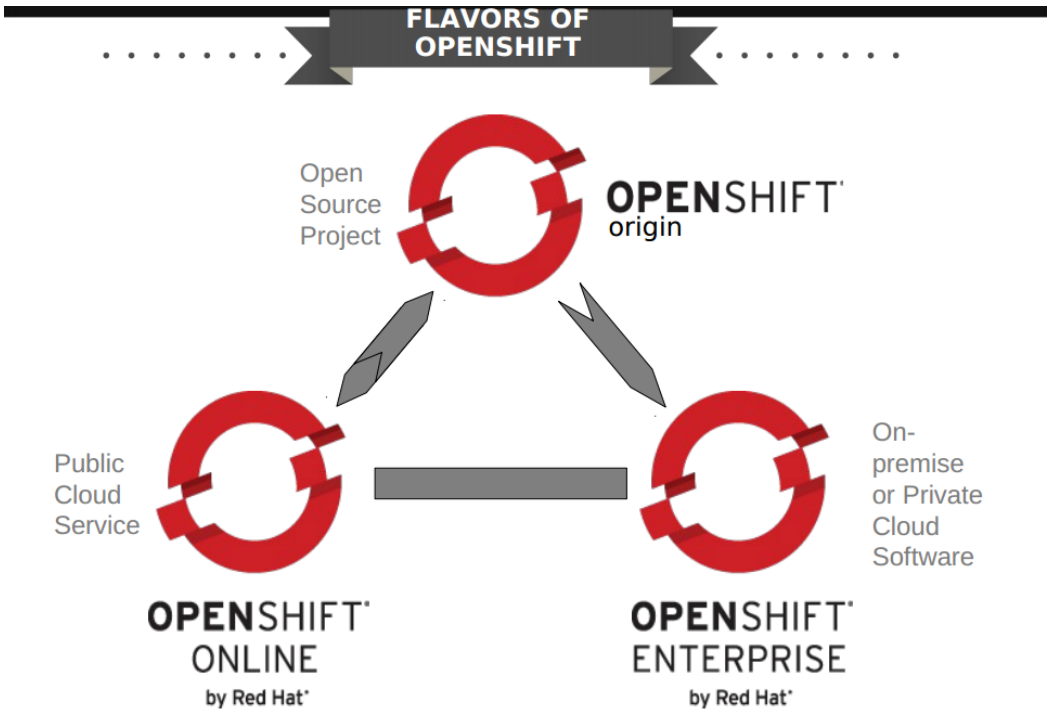


Fig. 2. Three types of OpenShift (credits: Red Hat)

OpenShift Origin includes support for a wide variety of language runtimes and data layers including Java EE6, Ruby, PHP, Python, Perl, MongoDB, MySQL, and PostgreSQL.

### 3.1.2 Installation

## 3.1.2.1 Prerequisites

Before OpenShift Origin can be installed, the following services must be available in your network:

- DNS
- MongoDB
- ActiveMQ

And the hosts (or nodes) in your system must have the following clients installed:

- NTP
- MCollective

After:

1. Setup Yum repositories
   1.1. Configure the openshift-dependencies RPM repository
   1.2. Configure the openshift-origin RPM repository
   1.3. Include the EPEL repository to install Puppet and update
2. Update the Operating System using SSH & yum
3. Configure the Clock to Avoid Time Skew
4. Firewall setup
5. Setting up the Ruby Environment

## 3.1.2.2 The Broker

The Broker is the central dispatcher in the OpenShift Origin service.

The installation includes:

1. DNS configuration
2. Add the Broker Host to DNS
3. DHCP Client and Hostname
4. Install the MongoDB server

**Server used:**

- broker host

**Tools used:**

- text editor
- yum

- sed
- chkconfig
- lokkit
- openssl
- ssh-keygen
- fixfiles
- restorecon

### 3.1.2.3 The Web Console

**Server used:**

- broker host

**Tools used:**

- text editor
- yum
- service
- chkconfig

The OpenShift Origin Web Console is written in Ruby and will provide a graphical user interface for users of the system to create and manage application gears that are deployed on the gear hosts.

### 3.1.2.4 The Node

**Servers used:**

- Node host
- Broker host

**Tools used:**

- text editor
- yum

- ntpdate
- dig
- oo-register-dns
- cat
- scp
- ssh

### 3.1.3 Configuration

The aim of deployment OpenShift Origin was to ease maintenance effort for the administrators, in other words to simplify the process of deployment our own stack. But at a certain stage we decided to shift our attention to other open source Platform as a Service solutions because the deployment of OpenShift Origin required heavy infrastructural modifications which are not possible at that very moment.

### 3.1.4 OpenShift Origin Virtual Machine Deployment

The command for starting download is **wget**. The archive can be downloaded from OpenShift mirror site:

```
$ wget https://mirror.openshift.com/pub/origin-
server/release/3/images/openshift-origin.zip
```

The command for extracting files from the archive is **unzip**.

```
$ unzip openshift-origin
```

It takes significant time to unpack the archive. The **ls -1t** command shows three extracted files:

```
$ ls -1t
openshift-origin.vbox
openshift-origin.vmdk
openshift-origin.vmx
```

Multicast DNS services are built-in on Windows (zeroconf) system, so configuration is not needed.

VirtualBox runs on Windows 7 (32-bit) as a host operating system. Multicast DNS services are built-in on Windows (zeroconf) system, so configuration is not needed.

For a good start OpenShift Deployment Guide recommends to set 1 GB for a machine with 4 GB of RAM. In this case the appropriate size is 768 MB.

Select VM "Hard Drive" Image

In most cases at this stage Virtualbox will create a new virtual hard drive. Here the virtual disk image contains the OpenShift Origin virtual machine.

New Virtual Machine at Oracle VM VirtualBox Manager Translation (NAT) is used by default, but it does not let user connect back into the virtual machine, that is why a second network adaptor configuration is needed. Adapter 2 is selected as a second adapter.

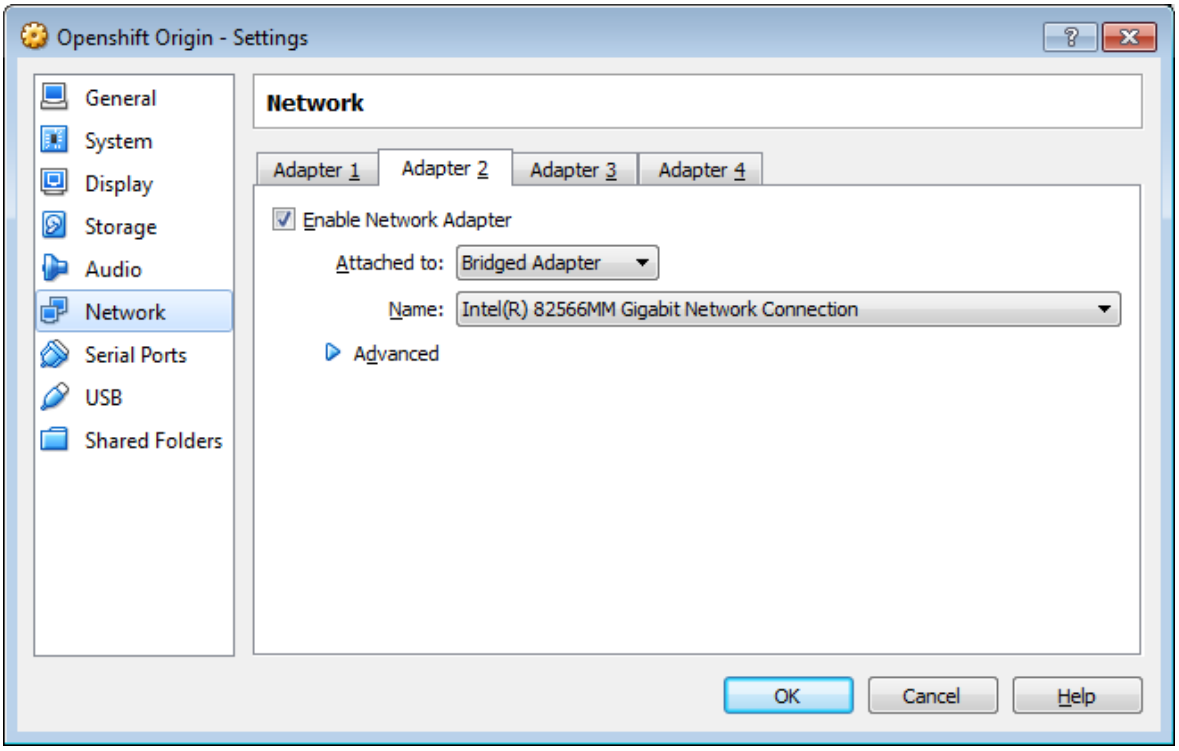


Fig. 3. Bridged Adapter configuration

After pushing 'OK' the necessary steps of configuration are finished. By pushing the 'Start' button, VirtualBox will display the VM console as a black window and the boot process can be observed.

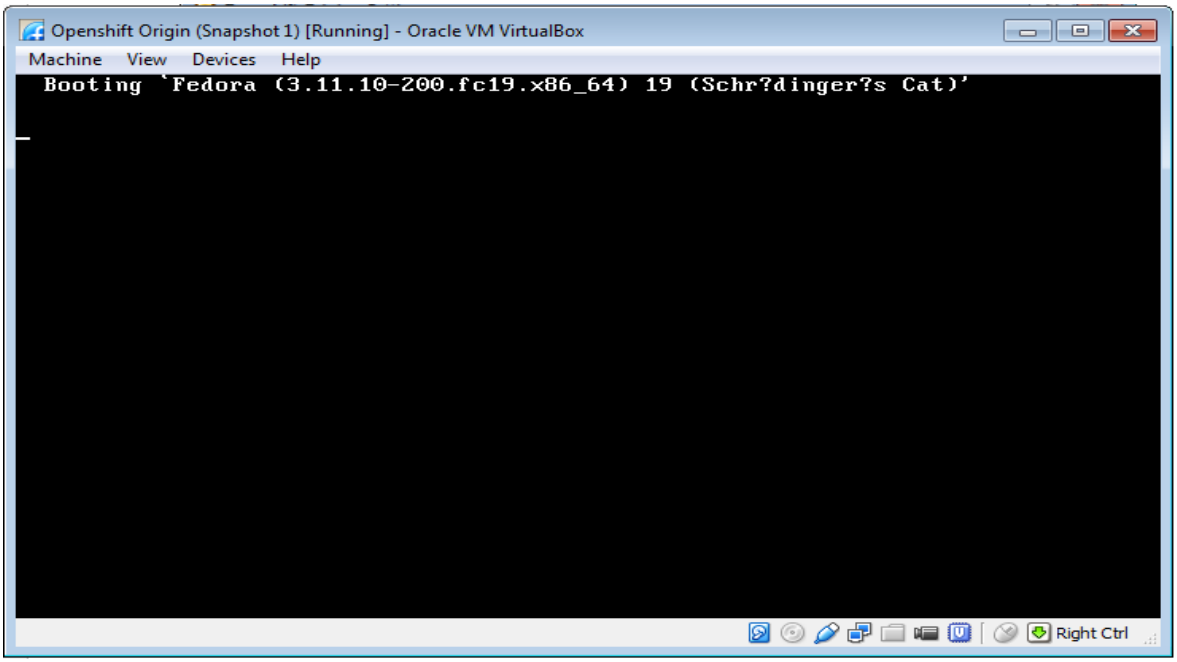


Fig. 4. Booting process

## 3.2 OpenShift Online

### 3.2.1 Overview

OpenShift Online by Red Hat is a Platform as a Service (PaaS) that enables developers to build and deploy web applications. OpenShift Online provides a wide selection of programming languages and frameworks including Java, Ruby, and PHP. It also provides integrated developer tools to support the application lifecycle, including Eclipse integration, JBoss Developer Studio, and Jenkins. OpenShift Online uses an open source ecosystem to provide a platform for mobile applications, database services, and more.

OpenShift Online is built on Red Hat Enterprise Linux, which provides a secure and scalable multi-tenant operating system to address the needs of enterprise-class applications as well as providing integrated application runtimes and libraries.

### 3.2.2 Features

OpenShift Online enables users to create, deploy and manage applications online. It provides disk space, CPU resources, memory, network connectivity, and an Apache or JBoss server. For most types of applications, OpenShift Online creates a file system layout that you can use as a template for building an application. OpenShift Online also generates a limited DNS so your application is accessible online.

The two main functional units of OpenShift Online are the broker and cartridges.
The **Broker** is the single point of contact for all application management activities. It is responsible for managing user logins, DNS, application state, and general orchestration of the applications.

Cartridges represent pluggable components that can be combined within a single application. These include programming languages, database engines, and various management tools.

### 3.2.3 Client tools

#### 3.2.3.1 Installation

Using the following command at Git Bash (**rhc** stands for The OpenShift Client tools)

```
$ gem install rhc
```

to install client tools, the result is displayed at Figure 5.

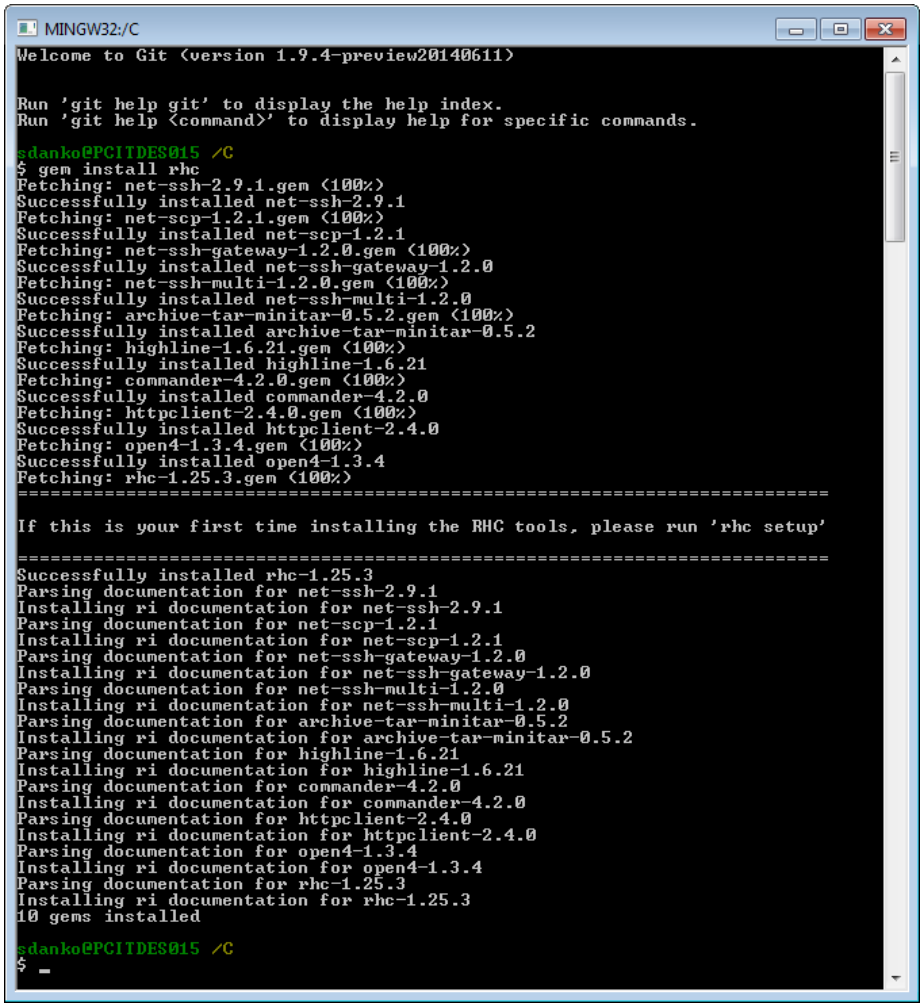


Fig. 5. Installing client tools for OpenShift

### 3.2.3.2 Configuration

Starting the Setup Wizard

The setup wizard can be launched by running the **rhc** setup command:

```
$ rhc setup
OpenShift Client Tools (RHC) Setup Wizard

This wizard will help you upload your SSH keys, set your
application namespace, and check that other programs like Git are
properly installed.
```

The following actions are required:

1. Sign in
2. Authorization Tokens generation
3. Configuration File
4. SSH Keys generation in case of initial configuration
5. Public SSH key uploading to OpenShift server
6. Setting private SSH key as 'readable only for me'
7. Creating first domain (entering its namespace)
8. Checking applications

The types of applications that can be created with the associated commands are shown at Figure 6.

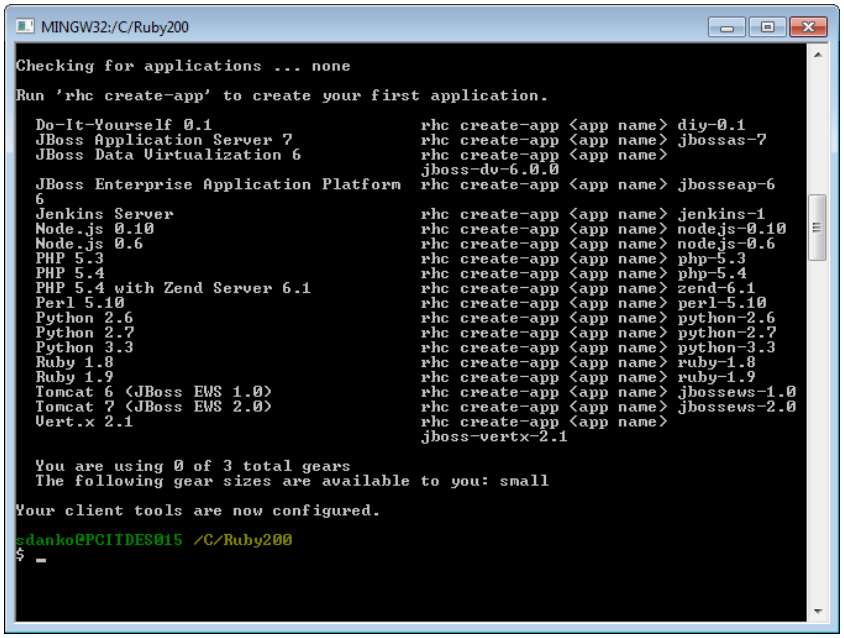


Fig. 6. Completing of configuration

Updating the client tools can be provided by running the following command:

```
$ gem update rhc
```

### 3.2.4 Applications

The process of PaaS deployment using OpenShift Online can be divided into 3 steps:

1. ‘Choose a web programming cartridge or kick the tires with a quickstart. After you create the application you can add cartridges to enable additional capabilities like databases, metrics, and continuous build support with Jenkins’ [4].

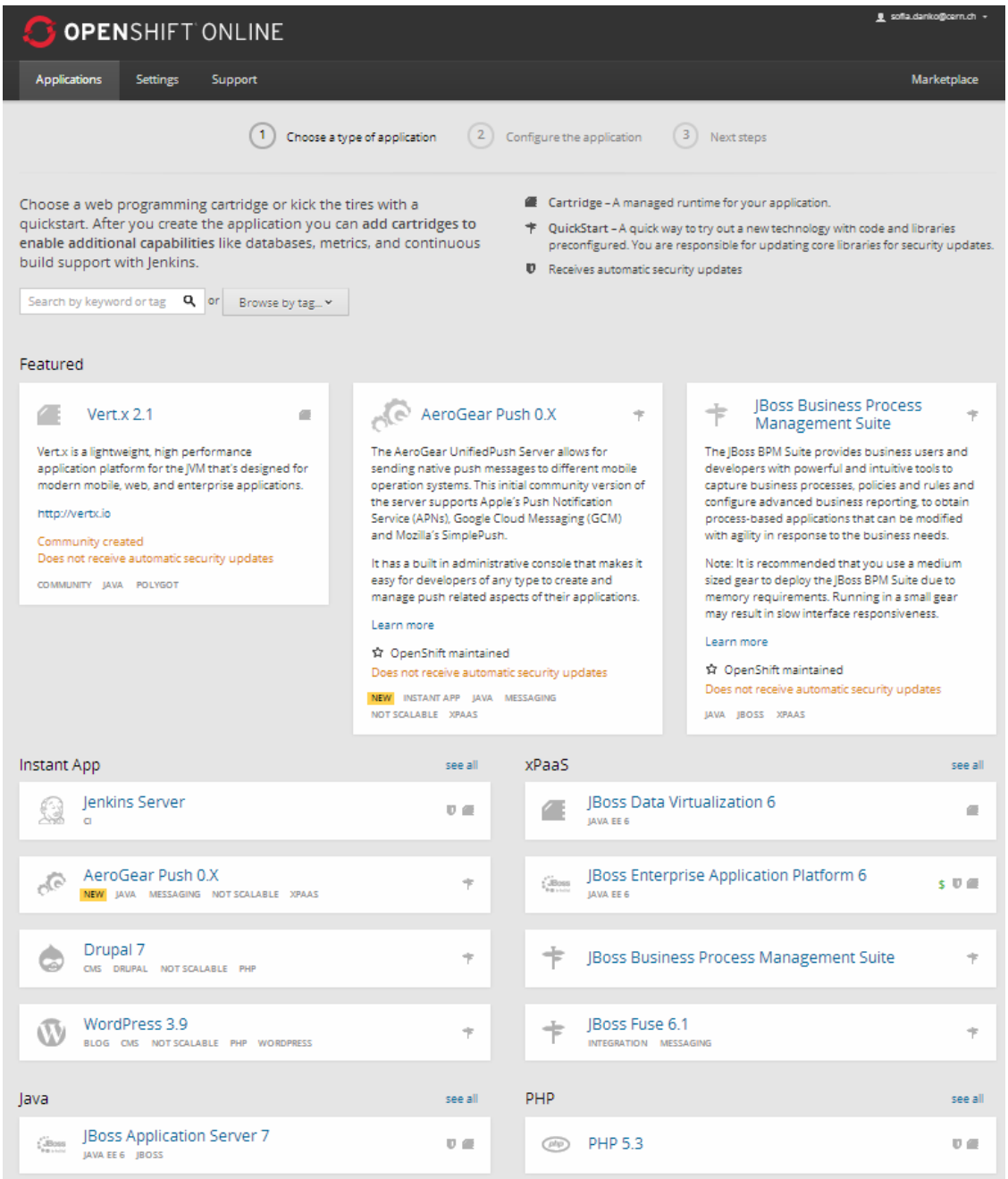


Fig. 7. Type of application selection (https://openshift.redhat.com)

2. Configuration is simple: user sets the name (Public URL), other properties are optional.

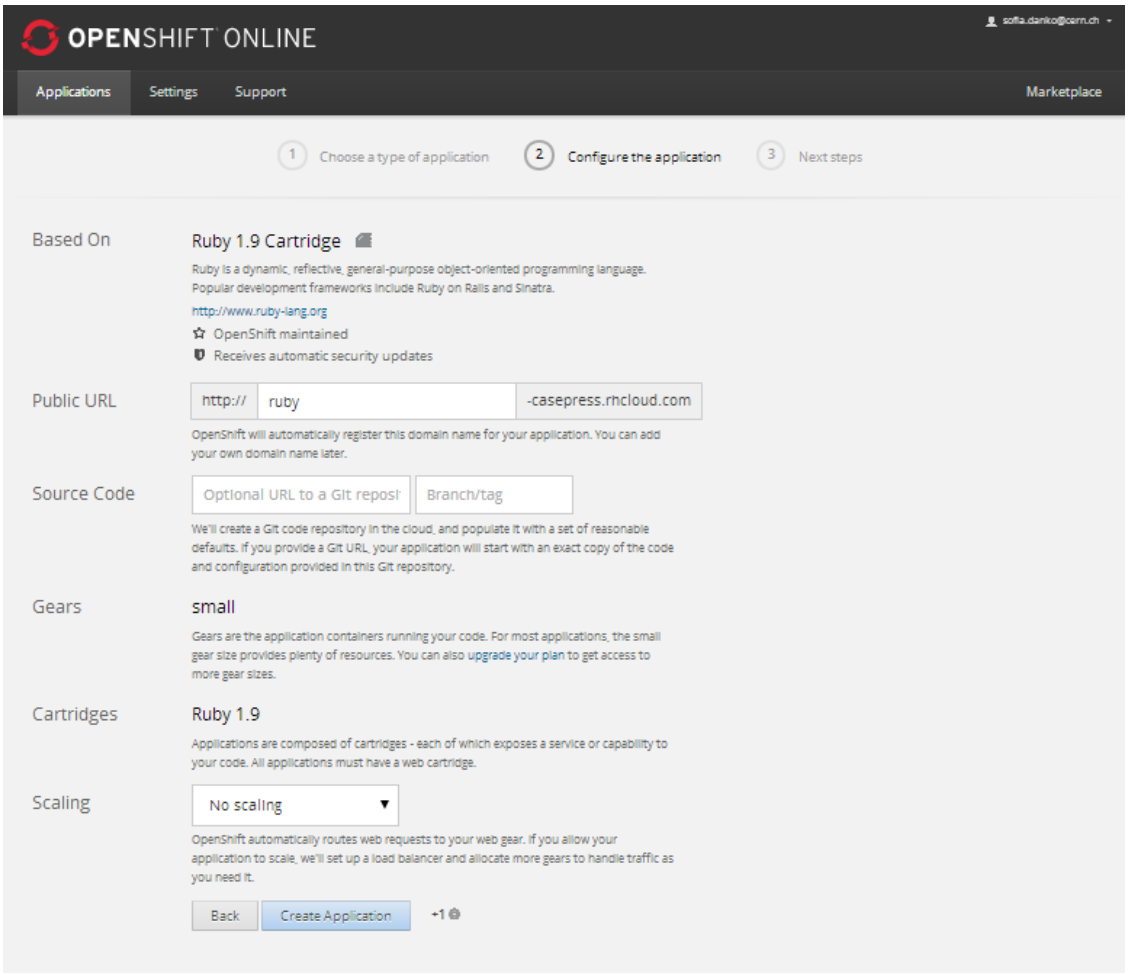


Fig. 8. Application configuration (https://openshift.redhat.com)

3. Basically the new application is created and configured.

The process of app configuration using the tools provided by Online version of OpenShift seems to be easy and fast. This could help to get basic idea of PaaS configuration and set the requirements to the solution that is going to be deployed for further usage: it should be simple, usable, fast and easy scalable.

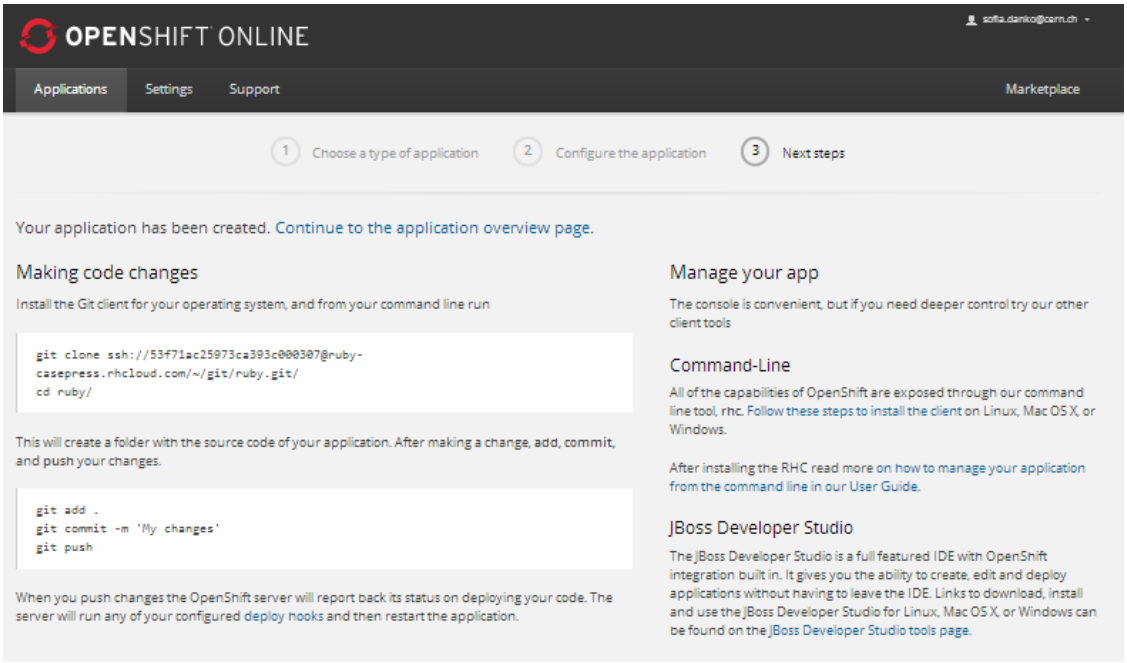


Fig. 9. Final step of application creation using OpenShift Online (https://openshift.redhat.com)

# 4 Cloud Foundry

Cloud Foundry is another open source Platform as a service private hosting solution provided by Pivotal that offers developers to use different runtimes and services in order to simplify the process of PaaS maintenance.

**Table 1. Cloud Foundry Runtimes**

| **Language** | **Versions** |
|---|---|
| Go | *1.1, 1.2* |
| Groovy | *1.5.*, 1.6.*, 1.7.*, 1.8.*, 2.0.*, 2.1.** |
| Java | *1.6.*, 1.7.*, 1.8.** |
| Node | *0.4.*, 0.6.*, 0.8.*, 0.10.** |
| Ruby | *1.8.7, 1.9.2, 1.9.3, 2.0.0* |
| Scala | |

**Table 2. Cloud Foundry Frameworks**

| **Name** | **Runtime** |
|---|---|
| Grails | Groovy |
| Play | Java |

| Name | Runtime |
|---|---|
| Rails | Ruby |
| Sinatra | Ruby |
| Spring | Java |

The Cloud Foundry architecture is shown on Figure 10.

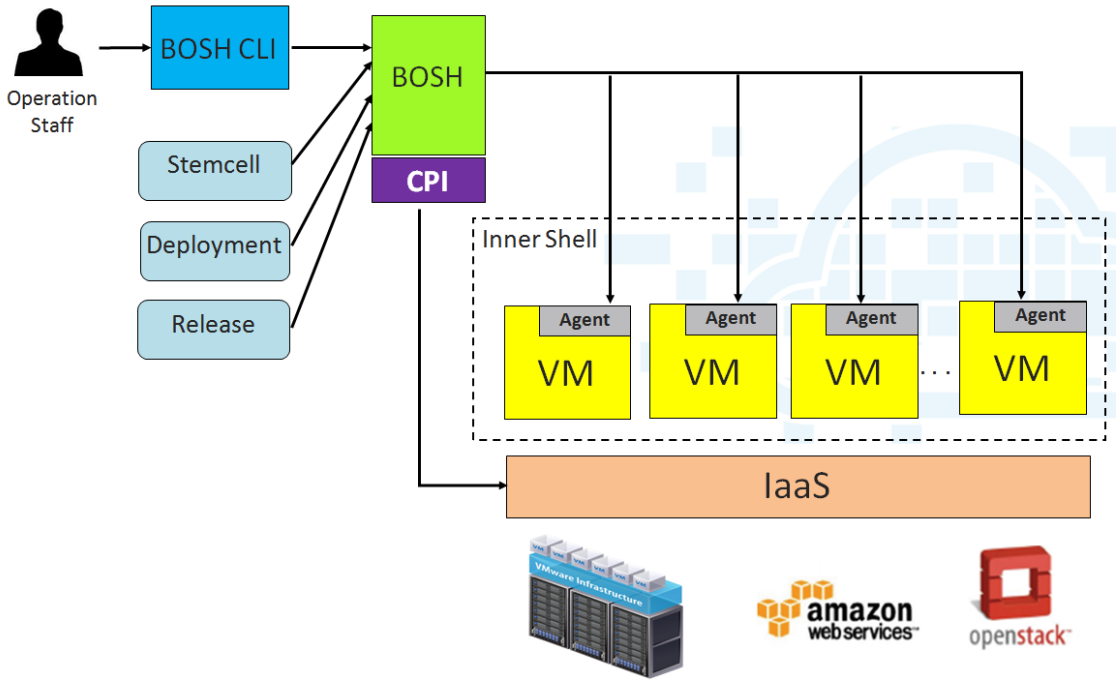


Fig. 10. Cloud Foundry architecture (from http://www.think-foundry.com)

Essential component of Cloud Foundry is BOSH. BOSH creates an Agent on every VM that BOSH deploys. A BOSH Agent listens for instructions from the BOSH Director to the VM and carries out those instructions.

The BOSH Command Line Interface (CLI) is the primary operator interface to BOSH. An operator uses the CLI to interact with the BOSH Director and perform actions on the cloud. The Director controls VM creation and deployment, as well as other software and service lifecycle events.

Stemcell is a generic VM image that BOSH clones and configures to during deployment.

Release is a collection of configuration files, job definitions, source code, package definitions and accompanying information needed to make a software component deployable by BOSH.

Deployment is an encapsulation of software and configuration that BOSH can deploy to the cloud.
CPIA Cloud Provider Interface is a software layer between BOSH and an IaaS (cloud).

Figure 11 represents BOSH components:

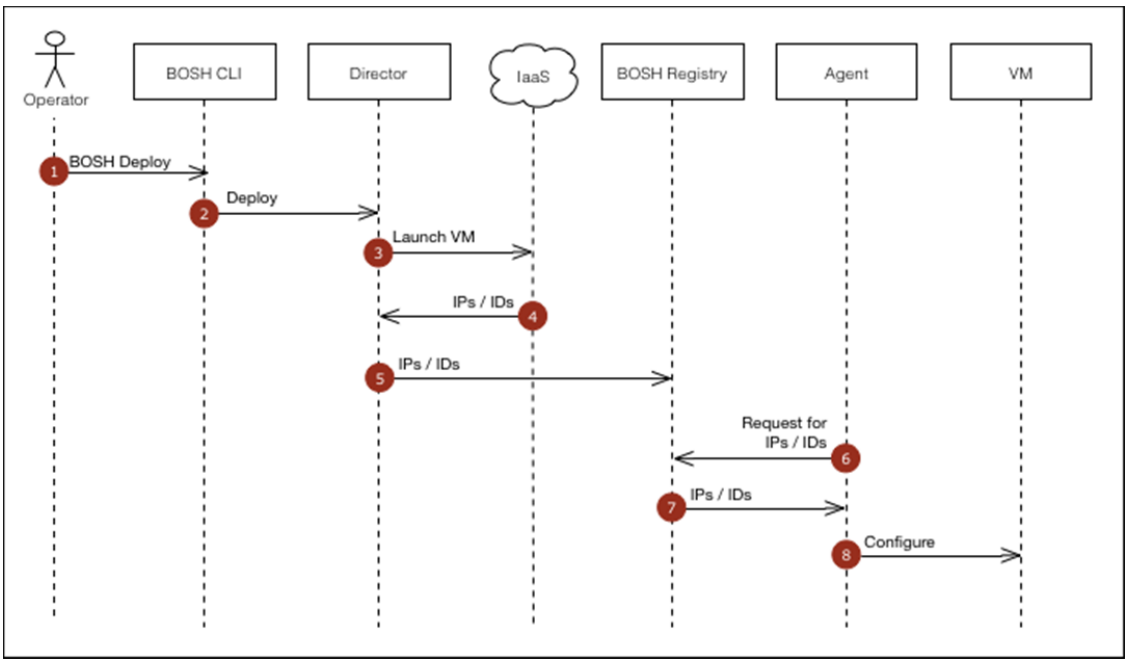


Fig. 11. BOSH components (from http://docs.cloudfoundry.org)

The interaction between the components can be described this way:

1. Through the BOSH CLI, the operator takes an action (e.g. deploy for the first time, scaling up deployment) which requires creating a new VM.
2. The CLI passes the instruction to the Director.
3. The Director uses the CPI to tell the IaaS to launch a VM.
4. The IaaS provides the Director with information (IP addresses and IDs) the Agent on the VM needs to configure the VM.
5. The Director updates the BOSH Registry with the configuration information for the VM.
6. The Agent running on the VM requests the configuration information for the VM from the BOSH Registry.

7. The BOSH Registry responds with the IP addresses and IDs.
8. The Agent uses the IP addresses and IDs to configure the VM.

At the current stage of the project the deployment of Cloud Foundry is in-between 3 and 4 points.

# 5 Future proposals

For the future investigation we would like to propose to have a look at the following external solutions:

- **Deis;**
- **Paasmaster;**
- **Cloudify;**
- **Stackato;**
- **WSO2 Stratos.**

In addition we would like to evaluate infrastructural changes needed to deploy OpenShift Origin and use industry-standard tools in our PaaS services, to ease maintenance and extend capabilities. At the moment CERN just offers Java PaaS to the developers, and we could investigate how the same platform could be used to provide also other languages/applications after testing Cloud Foundry and some other solutions. At the current stage the developers are setting up their own application stacks and this could be centralized and that could boil down to increasing efficiency.

# 6 Discussion and conclusion

The concept of cloud computing is a very important direction of modern computing development. Providing Platform as a Service tools to the developers can ease the maintenance and increase the productivity by extending capabilities.

During 2 months of the project a variety of different PaaS solutions was examined, as well as new ideas for deployment were proposed.

Some restrictions to deployment planned solutions were found, as well as infrastructural changes needed to be done were evaluated.

# 7 References


1. Strachey, Christopher (June 1959). "Time Sharing in Large Fast Computers". Proceedings of the International Conference on Information processing, UNESCO. Paper B.2.19: 336–341.
2. http://j2ee-public-service.web.cern.ch/j2ee-public-service/
3. https://twiki.cern.ch/twiki/bin/view/DB/Private/MiddlewareManagerDocumentation
4. https://openshift.redhat.com/app/console/application_types
5. http://www.paasify.it/vendor/cloud%20foundry
6. http://proquest.tech.safaribooksonline.de/book/operating-systems-and-server-administration/virtualization/9781449323400
7. http://openshift.github.io/documentation/oo_deployment_guide_comprehensive.html
8. http://openshift.github.io/documentation/oo_administration_guide.html
9. http://openshift.github.io/documentation/oo_user_guide.html
10. http://openshift.github.io/documentation/oo_install_users_guide.html
11. http://openshift.github.io/documentation/oo_notes_running_a_local_ddns.html
12. https://access.redhat.com/documentation/en-US/OpenShift/2.0/html/User_Guide/sect-OpenShift-User_Guide-Creating_an_Application.html#sect-OpenShift-User_Guide-Cloning_Application_Files
13. https://github.com/openshift/origin-server/blob/master/documentation/oo_install_users_guide.adoc
14. https://access.redhat.com/documentation/en-US/OpenShift/2.0/pdf/User_Guide/OpenShift-2.0-User_Guide-en-US.pdf
15. http://openshift.github.io/documentation/oo_system_architecture_guide.html
16. http://www.slideshare.net/scitronpousty/build-your-own-paas-using-openshift-origin
17. https://www.openshift.com/developers/deploying-and-building-applications
18. http://docs.cloudfoundry.org/
19. http://www.openqrm-enterprise.com/fileadmin/DATA/Whitepapers/openQRM-documentation-24022010.pdf
20. http://www.think-foundry.com/img/fig0.png