Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements

Cloud testbed setup

Deliverable 5.6

Deliverable:	D5.6
Title:	Cloud testbed setup
Editor(s):	Darren Whigham (FLEXI)
Contributor(s):	Craig Sheridan (FLEXI)
Reviewers:	José Ignacio Requeno (ZAR), Youssef RIDENE (NETF)
Type (R/P/DEC):	Other
Version:	1.0
Date:	26/07/2016
Status:	Final version
Dissemination level:	Public
Download page:	http://www.dice-h2020.eu/deliverables/
Copyright:	Copyright © 2016, DICE consortium – All rights reserved

DICE partners

ATC:	Athens Technology Centre
FLEXI:	Flexiant Limited
IEAT:	Institutul E Austria Timisoara
IMP:	Imperial College of Science, Technology & Medicine
NETF:	Netfective Technology SA
PMI:	Politecnico di Milano
PRO:	Prodevelop SL
XLAB:	XLAB razvoj programske opreme in svetovanje d.o.o.
ZAR:	Unversidad De Zaragoza

The DICE project (February 2015-January 2018) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869

Executive summary

This deliverable is a report on the testbed setup by Flex for use within the DICE project. As well as a detail a work in progress description of the fault-injection tools and techniques developed on top of this testbed. These have been developed into the DICE Fault Injection Tool (FTI) and will be released as part of the DICE IDE and documented in detail within month 24.

Glossary

FCO	Flexiant Cloud Orchestrator
VM	Virtual Machine
FTI	Fault Injection Tool
AWS	Amazon Web Services

Table of contents

List of figures

List of Tables

1. Introduction

This deliverable will detail the first iteration of the Cloud testbed that has been developed for use within the DICE project. An overview of both the physical hardware and Cloud Orchestration Software that is used to provide this testbed.

This testbed has been provided as a close to real life platform to allow development and testing of the DICE tools. This helps to ensure that the goals of the DICE tools are achieved. These goals are three-fold. First, they aim to simplify the development and deployment of the data-intensive applications so that it becomes more accessible to the new and less experienced programmers. Second, the emphasis is on the tools' ability to provide recommendations on how the vital application components should be configured in order to achieve efficient and reliable application runtime. Existing tools require users' expertise and an extensive set of tweaking the configuration to find the optimal solution. Finally, the DICE tools aim to provide feedback on the quality of the application being developed.

In addition the DICE Fault Injection Tool (FIT) will also be discussed within this deliverable as well as its role within the DICE project. Operating data intensive applications unavoidably involves dealing with various failures. The industry has long recognised that instead of designing software to never fail, it is a much more viable strategy to make it reliable and resilient to failures. To properly test applications during development in this respect, the Fault Injection Tool presented in this report will be invaluable in a quality-driven application development and in the realisation of the DevOps. The DICE Fault Injection Tool (FIT) provides the ability for the tool user to generate faults either at on the VM itself or as an admin on Cloud. The purpose of the FIT is to allow cloud platform owners a means to test the resiliency of a cloud installation as an application target. With this approach, the designers can use robust testing, showing where to harden the application before it reaches a commercial environment and allows a user/application owner to test and understand their application design/deployment in the event of a cloud failure or outage. Thus allowing for the mitigation of risk in advance of a cloud based deployment.

2. Testbed overview

Flexiant have provided a Cloud testbed for use within the DICE project. This testbed has 11 compute nodes, which are split between two clusters and run the most recent at time of writing version 5.0.15 of Flexiant Cloud Orchestrator (FCO). Over the course of the project lifetime this testbed is periodically updated to new releases of FCO. Feedback on bugs are passed from the project to the FCO development team as to ensure the FCO product improved not only for the project but for all Flexiant customers. This ensures that developers within DICE have access to the latest Cloud Orchestration software as well as ensuring features and bug fixes are provided.

Flexiant Cloud Orchestrator is a turnkey product enabling service providers and enterprises to design, create and manage broad range of cloud solutions. Flexiant Cloud Orchestrator can manage the entire cloud solution, from hardware, network and storage management to metering, billing & customer/end-user self-service from infrastructure to complex applications.

Hardware Resources and Infrastructure

Flexiant has built and made available a testbed that is cleanly and efficiently managed by dedicated engineers and allows access of different levels using multiple APIs, such as an admin API and user API. The testbed can have master and standard billing entities set, product offers can be created and manipulated, to provide users different access to different networking and physical hardware as required.

The testbed consists of two clusters using CEPH as the storage solution. The specifications of the individual nodes is shown within Table 1 and the overview of cluster 1 is shown in Figure 1. Each cluster uses separate CEPH storage to and to accommodate this cluster has CEPH multiple CEPH Monitors and CEPH Nodes.

Table 1 FCO Cluster 1 specifications of individual nodes

Cluster 1	CPU Cores	CPU Type	RAM
Compute Node 1	16	AMD Opteron Processor 6366 HE	RAM 128GB
Compute Node 2	8	AMD Opteron Processor 6212	RAM 64GB
Compute Node 3	16	AMD Opteron Processor 6366 HE	RAM 128GB

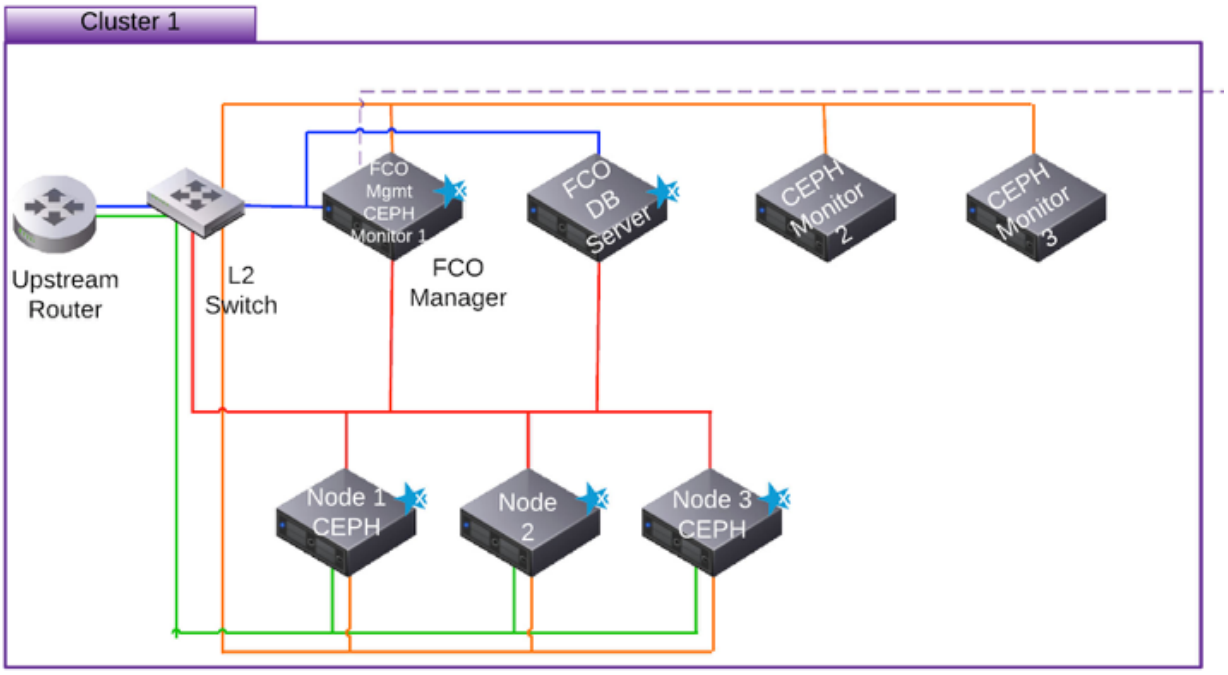

Figure 1 - FCO Cluster 1

The cluster 2 overview and specifications of all the respective nodes are given in Table 2

Table 2 - FCO Cluster 2 specifications of individual nodes

Cluster 2	CPU	CPU Type	RAM
Compute Node 1	8	Dual-Core AMD Opetron™ Processor 8820	RAM 64GB
Compute Node 2	4	AMD Opetron™ Processor 6320	RAM 32GB
Compute Node 3	4	AMD Opetron™ Processor 6320	RAM 32GB
Compute Node 4	8	AMD Opteron™ Processor 6212	RAM 64GB
Compute Node 5	8	AMD Opteron™ Processor 6212	RAM 64GB
Compute Node 6	16	Quad-Core AMD Opteron™ Processor 8356	RAM 64GB
Compute Node 7	16	AMD Opetron™ Processor 6366 HE	RAM 128GB
Compute Node 8	16	AMD Opteron Processor 6366 HE	RAM 128GB

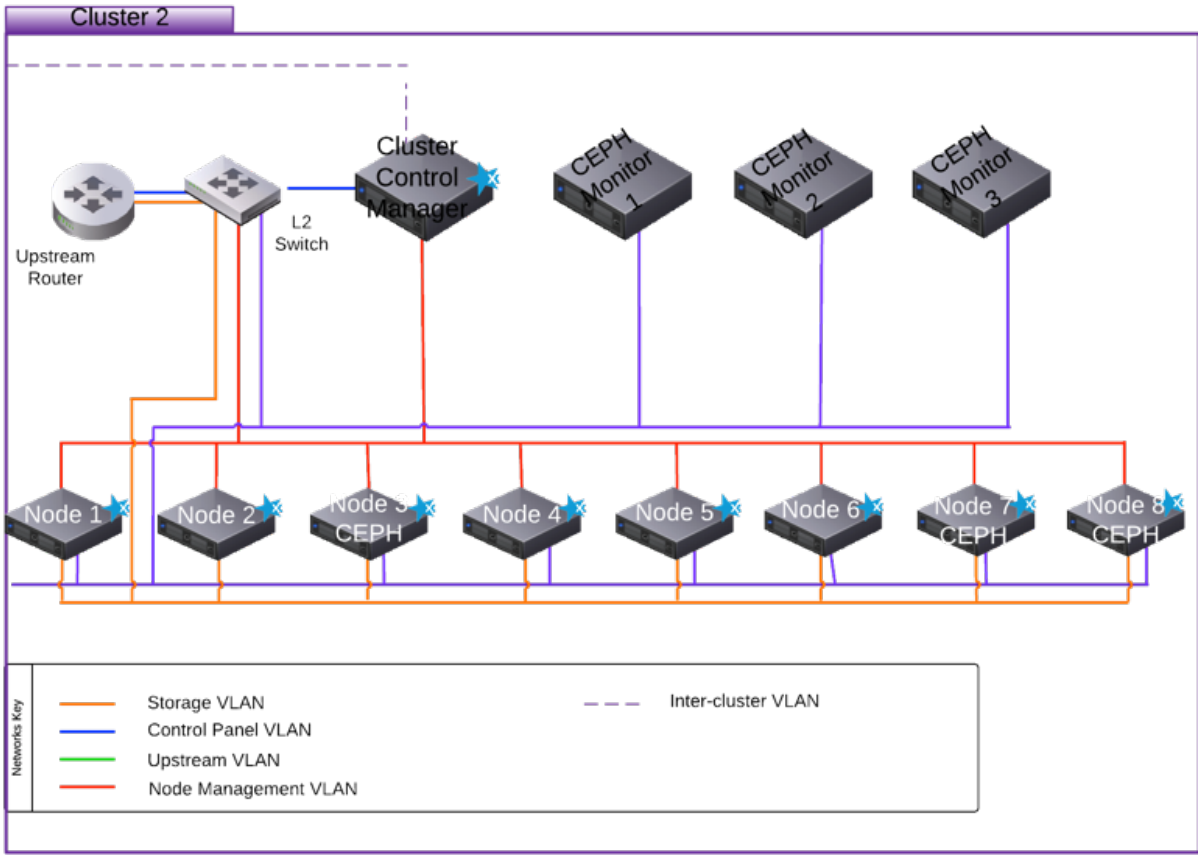

Figure 2 - FCO Cluster 2

The testbed can be accessed using the Flexiant Cloud Orchestrator (FCO) on https://cp.sd1.flexiant.net. The UI is shown in Figure 3.

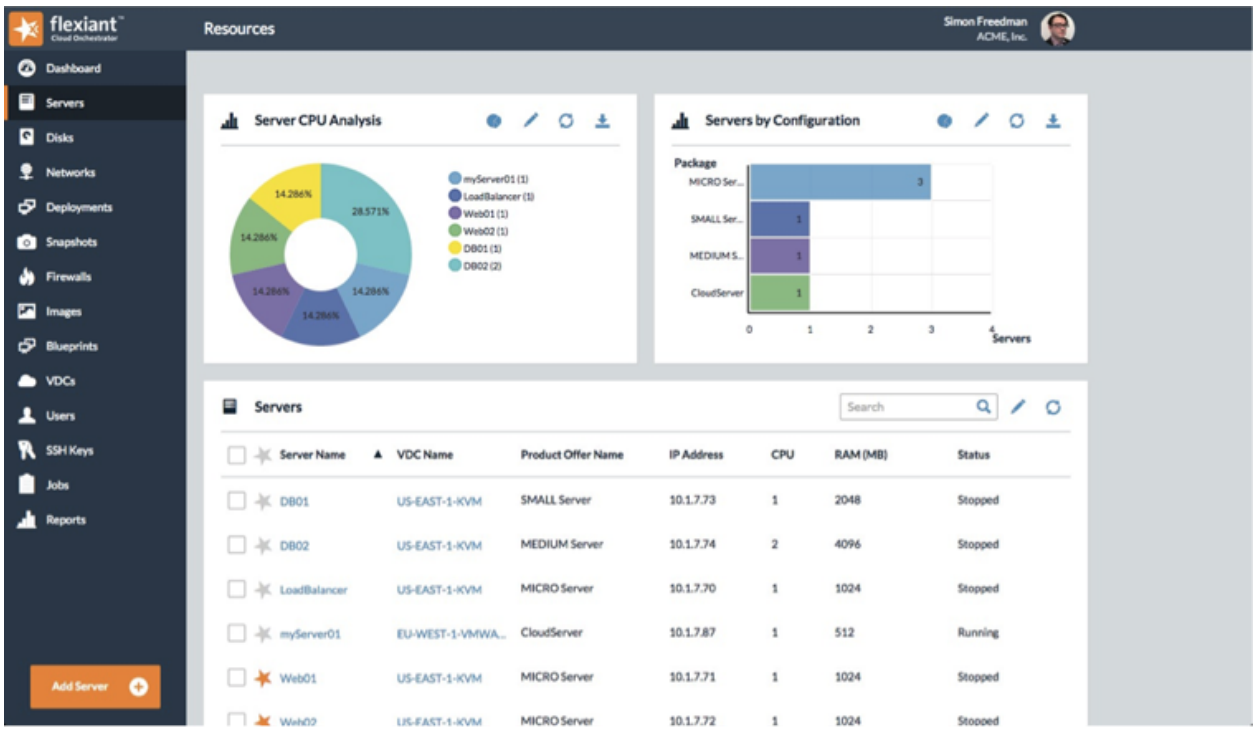

Figure 3 - FCO User Interface

Configuration, setup and usage

The platform can be accessed via a user interface built on top of a SOAP API and also provides RESTful services. The platform has built in support for various reseller and billing models along with integration to 3rd party tools. Multi-cluster support is available from a single interface with all popular hypervisor types (KVM, Xen, VMWare, HyperV) available and controllable via the Flexiant management stack.

Account Structure

Flexiant Cloud Orchestrator has a number of levels of account management [1] to enable delegation of control to the appropriate level (see Figure 4).

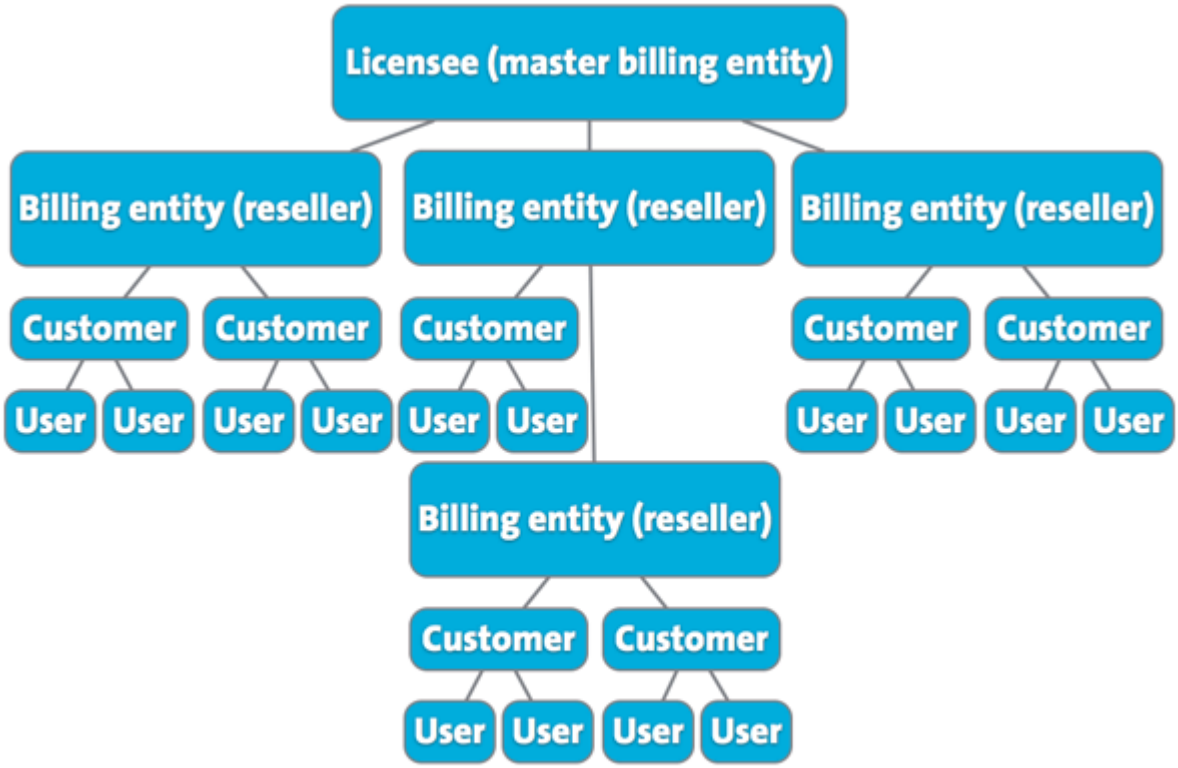

Figure 4 Flexiant Account Structure Overview

At the top level is the Licensee; they have permissions over the whole platform. In a billing context, we refer to the licensee as the master billing entity or MBE. Licensee access enables them to manage the hardware platform, specific products and billing entities.

Billing entities are organisations that the licensee sells platform access to. These in turn have customer accounts, which contain user accounts. A billing entity can also have its own child billing entities, which sell platform access to their own customers. Billing entities that have one or more child billing entities (i.e. another billing entity that resells access to the first billing entity's portion of the platform) are referred to as parent billing entities.

Within the DICE project we will use the licensee functionality to provide account access to the testbed for multiple users and accounts. This allows multiple users to deploy onto the platform at the same time.

Application Stack Overview

Flexiant Cloud Orchestrator has been developed to be ran as a modular application stack [2]. Though the default installation will install the entire application stack onto one machine, the modular nature of the application stack allows FCO to span multiple locations. This greatly increases scalability and flexibility. This will ensure the platform for use within the DICE project is scalable to meet the requirements as the project develops.

Billing Capabilities

Flexiant Cloud Orchestrator can generate invoice line items or complete invoices [3], including branded output via html, pdf or e-mail. FCO can also integrate with existing merchant providers to automatically collect payments from customers via credit card and can also be integrated into existing billing solutions which undertake some or all of the functions above, it can then be synchronised via the Admin API.

IP Migration

IP migration [4] is available to transfer an IP address between virtual machines. This means that the network traffic that was formerly routed to an old virtual machine can be routed to a new one.

Keys

Keys [5] are labels that can be given to resources. They can be used to filter widgets or to determine the optimal node placement for resources using Dynamic Workload Placement [6]. Keys can be set by either the master billing entity, billing entities, or customers, and are set when managing the appropriate resource. The master billing entity can set keys on all resources, billing entities can set keys on resources belonging to the billing entity (such as customers, for example), and customers can set keys on resources belonging to them, for example servers.

Metering

Metering [6] allows the use of applying charges based on a time interval or usage as defined within a product offering. These can be defined from a wide variety of sources such as seen within Table 3.

Table 3 Flexiant Metering overview

By time interval:	**By Usage:**
Virtual Machine Running time (consisting of a CPU and RAM combination) IP Address usage VLAN Usage Firewall Usage Image Usage Snapshot Usage Disk Capacity Software Licenses (in conjunction with images)	Disk IO Usage. Network Data Transfer

Products and Product Offers

Products represent the resources that Flexiant Cloud Orchestrator manages and orchestrates. Each product is linked to a type of resource. Product offers determine how products can be used, and how they are billed.

Tags

Tags [8] allow you to sort customers and billing entities into categories, and are used to determine the billing entities and customers that a product offer is available to.

Setup

Flexiant Cloud Orchestrator is used to manage an entire cloud solution. Figure 5 breaks down what the platform covers. The User interface is what a customer will use to interact with the platform. Customer Management & Billing is a subsection allows management of all customer accounts and billing entities. Cluster Management allows management of individual clusters, nodes and overall hardware of the platform. Hypervisor, network and storage control are used to control the underlying Hypervisors, network and storage available to the platform.

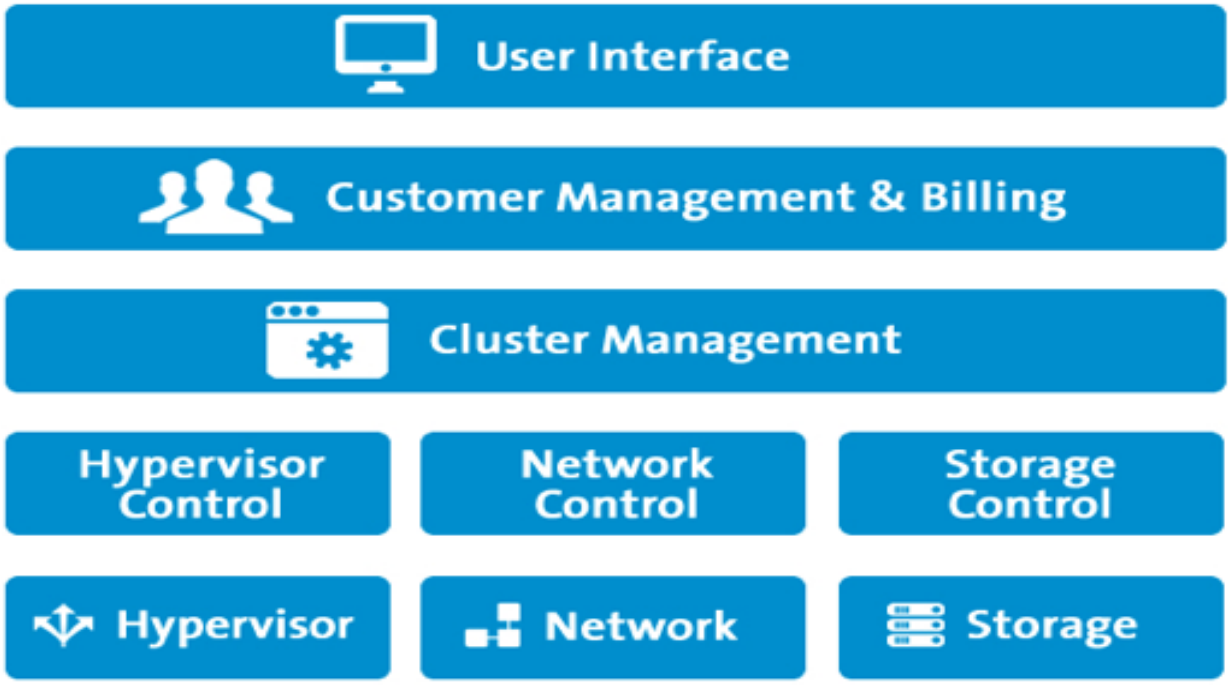

Figure 5 - FCO Breakdown

FCO in DICE

Within the DICE project, the testbed as described above has been made available for use by all partners within the project. This testbed as detailed has been split into two clusters, with Computer Node 3 fully dedicated for experimentation and benchmarking within the DICE project. In addition each WP has made use of the testbed in a number of ways as detailed in the following list.

WP3

Within WP3, PMI used the test bed to determine Hive queries (which are mapped into map reduce and Tez jobs) performance profiles and to validate queuing network and stochastic well formed networks performance predictions. Results have been reported into deliverable D3.8. Moreover, PMI plan to deploy the next release of its D-SPACE4Cloud tool on the test bed to provide access to project partners.

WP4

Within WP4 the Cloudera Distribution for Hadoop has been deployed on the testbed. In addition the DICE Monitoring Platform has also been deployed across a number of VMs within the testbed.

IMP with WP4 have used Cloudera Distribution for Hadoop to run a series of experiments on Spark technology within Oryx 2 deployment. The objective of experiments was to observe how Oryx 2 performance changes as a result of manipulating Spark and Oryx 2 configuration settings.

WP5

In terms of the WP5's own perspective, the testbed has been used to host the latest version of the DICE deployment service. Additionally, a Jenkins CI service has been deployed to test the T5.2's CI results and to use it for internal unit testing and integration testing of the deployment tool itself. The test bed is also used for testing the DICE technology library work and in addition exploit our own FCO plug-in for Cloudify to execute the blueprints on FCO deployed Cloud platforms.

Configuration Optimization hosts a service for demonstrating and testing the tool. They rely on the DICE deployment service instance mentioned earlier to deploy test clusters and topologies that are subject to the configuration optimization.

For testing the integration between configuration optimization and delivery tools (deployment service and CI) we have used FCO. Also we have run several experiments with WordCount system. We are also doing ATC case study with configuration optimziation tool. In addition the FIT is also tested on VMs and cloud platform offered by Flexiant.

WP6

Within WP6 the NETF use case (eGov Fraud Detection) heavily relies on Flexiant Testbed. INETF builds on FCO a Cassandra and Spark clusters which are respectively used to store the data and gather/process it when needed. The Flexiant Testbed hosts also a webservice which is in charge of the communication between the graphical user interface and the clusters. Indirectly, NETF use case uses also all the services used by DICE tools such as the monitoring platform, the configuration optimization etc.

3. Fault Injection Tool

3.1.Introduction

The DICE Fault Injection Tool [9] is currently being developed to generate faults within Virtual Machines and at the Cloud Provider Level. The FIT provides the ability for the tool user to generate faults either at the VM level or at the cloud level. The purpose of the FIT is to allow cloud platform owners/Application VM owners a means to test the resiliency of a cloud installation as an application target. With this approach, the designers can use robust testing, showing where to harden the application before it reaches a commercial environment and allows a user/application owner to test and understand their application design/deployment in the event of a cloud failure or outage. Thus allowing for the mitigation of risk in advance of a cloud based deployment. By using this tool developers and cloud operators can offering their best services to all customers due to the reasons above and simplify testing within the DevOps paradigm.

3.2.Design

The DICE FIT has been designed and developed in a modular fashion. This allows the replacement of any function that carries out Faults as well as potential to extend the tool as required. The FIT was designed to be as lightweight on the VMs as possible, so for this reason only existing well-tested tools have been implemented for causing faults. The FIT downloads, installs and configures only what is required at the time, because of this no unnecessary tools or dependences are installed. To access the VM level and issue commands the DICE tools uses JSCH (http://www.jcraft.com/jsch/) to SSH to the Virtual Machines to execute commands to download, install, configure and run tools used to cause faults within VMs. To execute a fault, this must be done via the command line. The example shown below will stress the VM's memory and use 512 MB on the VM with the IP 111.222.333.444 using a ssh key to authenticate.

--stressmem 2 512m ubuntu@111.222.333.444 -no c://SSHKEYS/VMkey.key

Any argument that requires VM password or SSHKeypath use "-no" for argument not used. A flow diagram can been seen in figure 4. It showcases the actions that the FIT takes once a command has been selected.

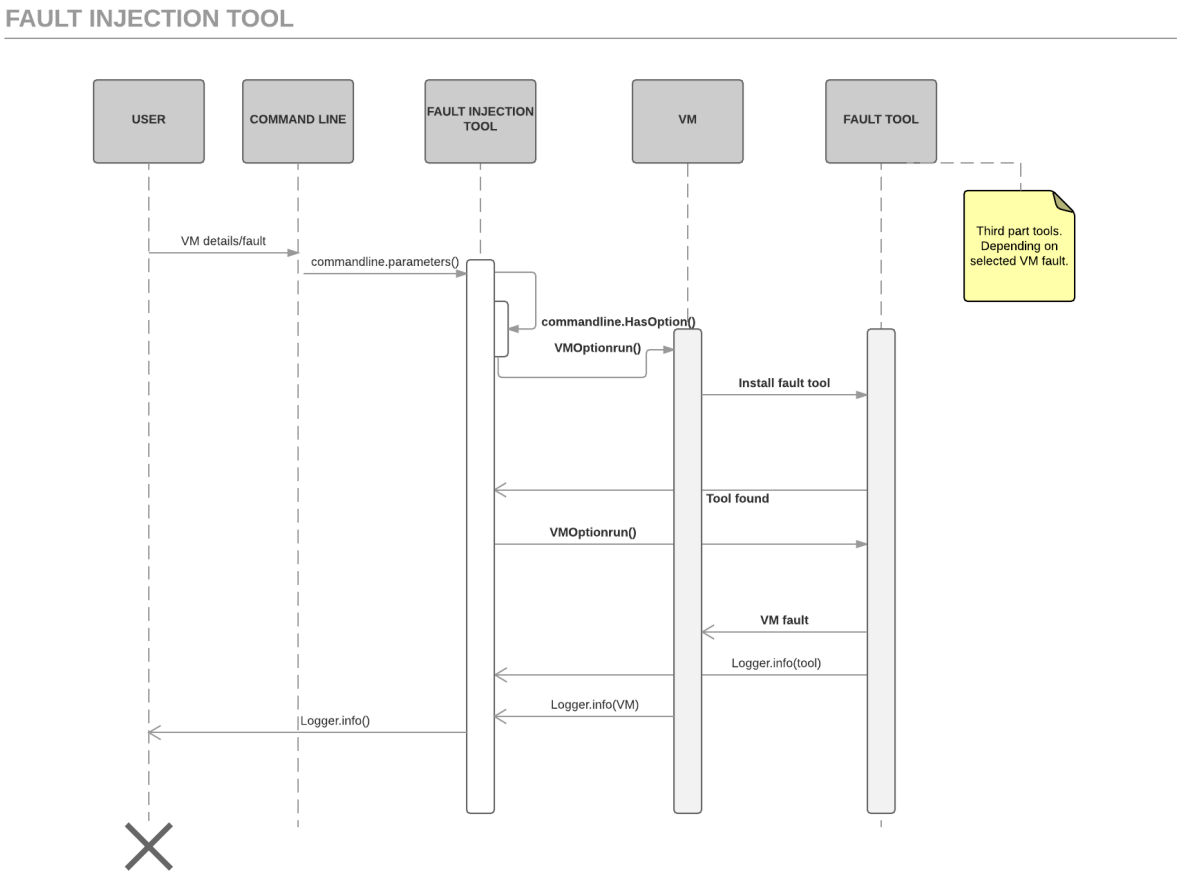

Figure 6 - DICE FIT Flow Diagram

Due to the extensive number of tools available, these existing tools were used to provide the FIT the ability to generate faults within the VM.

The current VM/User Faults are:

- Shutdown random VM (Ignore tagged VM with "noshutdown" in random selection)
- High CPU for VM (Using Stress tool)
- High Memory usage for VM (Using Memtest tool)
- Block VM external access (Using ufw)
- High Bandwidth usage. (Using iperf, requires external iperf server ip to be passed)
- High Disk I/O usage (Using Bonnie ++)
- Stop service running on VM
- Shutdown random VM from whitelist provided by user (Note the whitelist does not check if VM exists or is a in a running state)
- Call YCSB on VM running MongoDB to begin workload test.

- Run JMeter plan

3.3.FTI in practice Testing

During the continued development of the DICE FIT, testing has been carried out continuously. At the time of writing the FIT has been developed and tested with the following Operating Systems:

- Ubuntu (14.04, 15.10)
- Centos 7 (With set Repo configured & wget installed)

Testing of each Fault tool has been carried out by using the DICE FIT. An example showing the memory usage on a target node as shown within the WP4 D-MON [10] display before the FTI was used Figure 7 (left) and after Figure 7 (right)

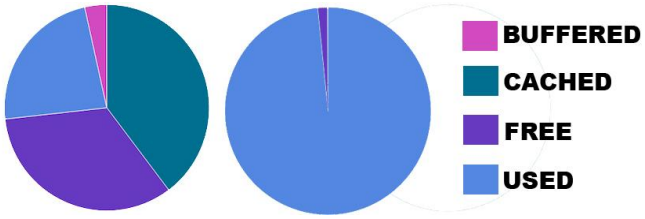

Figure 7 - Memory utilization of test VM

Further testing will be carried out by each use case partner once the initial version of the full DICE tool kit have been released as required to be integrated into each use case.

3.4.Future work

The future work within the lifetime of the project will focus on developing the range of faults at the VM level as well as develop the cloud level faults and implement them within the tool. This work will continue until M24 and will be detailed within D5.4 DICE testing tools - Initial version.

Other future work will could potentially focus on expanding on the types of Cloud level faults that are supported and potentially catering for different topologies, operating systems and vendor agnostic cloud provider infrastructure where possible. In addition containerised environments are also a consideration future FIT targets if possible during the course of the project lifetime. Finally other target cloud APIs such as AWS could be added to the FIT to expand the VM/User level fault selections.

4.Conclusion

In this deliverable we have documented the testbed that has been designed and deployed for use within the DICE project. This testbed has been developed to match a typically cloud architecture and uses the latest version of FCO for deployments. This will be improved over time with new iterations of FCO to improve features and fix bugs during the course of the project lifetime. The DICE Fault Injection Tool aimed at developers using a quality driven DevOps approach to be able to test their applications under the conditions of failing parts of the infrastructure or misbehaving services. The tool is easy to use and integrate in the Continuous Deployment workflow. In the current implementation of the tool, we show support of VM level faults.

References

[1] http://docs.flexiant.com/display/DOCS/Account+Structure

[2] http://docs.flexiant.com/display/DOCS/Application+Stack+Overview

[3] http://docs.flexiant.com/display/DOCS/Billing+Capabilities

[4] http://docs.flexiant.com/display/DOCS/IP+Migration

[5] http://docs.flexiant.com/display/DOCS/Keys

[6] http://docs.flexiant.com/display/DOCS/Metering

[7] http://docs.flexiant.com/display/DOCS/How+the+Dynamic+Workload+Placement+System+Works

[8] http://docs.flexiant.com/display/DOCS/Tags

[9] https://github.com/dice-project/DICE-Fault-Injection-Tool

[10] DICE H2020 - Monitoring and data warehousing tools – Initial version Deliverable 4.1

