Read the Docs Template Documentation

Release 1.0

Read the Docs

Jul 16, 2022

BASIC USAGE GUIDE

CHAPTER

ONE

INTRODUCTION TO CLOUD COMPUTING, CONTAINERS, MICROSERVICES AND DEVOPS

Cloud Computing has reached virtually all areas of society and its impact on service development, production, provision and consumption is manifold and far-reaching. It lowers innovation barriers and thereby impacts industry, small and large businesses, governments and society, and offers significant benefits for everyone.

According to Gartner Inc *, while cloud computing has been an application deployment and infrastructure management paradigm for many years now, the cloud market is still expanding, reaching an impressive $200bn milestone projection for 2016 with an increasing growth rate of 16%. In this digital economy, Small and Medium Enterprises (SMEs) and today's Startups are migrating core services and products of their business to the cloud. Recent studies shows that in 2015 more than 37% of SMEs have embraced the cloud to run parts of their business, while projections show that by 2020 this number will grow and reach 80%. However, properly preparing for tomorrow's cloud challenges is crucial if one wants to unleash the full potential of the technology.*

Below is a set of resources in the form of dissemination and scientific articles, implementation examples, blog entries, videos, tutorials and courses and at different levels of difficulty. The purpose of collecting this information is to give potential users of the Rainbow platform quick access to useful and high quality resources on different related topics, namely, general information on Cloud Computing, information on the challenges one must face when deciding to adopt this technology and, finally, aspects related to the agile processes of software development for Cloud Computing.

Moreover, you can read some useful information about analytic services, decision making, auto-scaling and monitoring using the RAINBOW platform.

CHAPTER

TWO

WHAT IS RAINBOW!

A framework that allows the design and deployment of secure and elastic by design cloud applications and services.

CHAPTER

THREE

RAINBOW TECHNOLOGY STACK

Developed on top of popular and open-source frameworks including Kubernetes, Docker, CoreOS to support multi-cloud application runtime management

CHAPTER

FOUR

WHY RAINBOW?

- All Rainbow apps are packaged and enabled by Docker Runtime Engine to create and isolate the containerized execution environment. * Docker Runtime Engine and Docker Compose are tools sufficient for small deployments, but limited to a single host.
- Kubernetes can support the orchestration of large-scale distributed containerized deployments spanning across multiple hosts. * However Kubernetes has limitations in regard on the provisioning and deprovisioning of infrastructure resources and the auto-scaling. * Also Kubernetes cannot support cross-cloud deployments.
- Underlying containerized environment based on CoreOS which enables fast boot times and secure-out-of-the Docker runtime. * Enhanced by security service to filter network traffic and apply privacy preserving ruling.
- Rainbow Smart Orchestrator is suitable for Highly Available (HA) host management. * Taps into auto-scaling offered by cloud offerings to estimate and assess app elasticity behavior and scaling effects. * Low-cost and self-adaptive monitoring to reduce network traffic propagation.
- Rainbow Smart Orchestrator enables deployments across multiple cloud sites. * Cross-cloud network overlay is provided.
- Compatibility with Docker Compose is preserved as an extension of Docker Compose is used to describe, configure and deploy multi-container applications using YAML syntax.

CHAPTER

FIVE

ONLINE DOCUMENTATION CONTENTS

5.1 Supported OS

Servers:

We support every distrubution system, but currently the scripts work only for debian and ubuntu. if there is a hard limitation please contact in order to extend the scripts for other distributions

RPI:

We support only 64bit ubuntu based distributions due to limitation of k8s working with calico as network plugin

Nvidia jetson/xavier/etc:

Kernerl recompilation is essential to enable flags.

CJDNS enablement for Xavier:

Instructions can be found under https://gitlab.com/rainbow-project1/rainbow-installation/-/tree/main#cjdns-enablement-for-xavier

5.2 Setup

Here you can find instructions for the installation of Rainbow Platform. This page references the repository under https://gitlab.com/rainbow-project1/rainbow-installation

5.2.1 Requirements

For the advanced platform package, that includes everything, the minimum requirements are:

Component	#vCPUs	RAM(GB)
k8s master	4	10
k8s worker server	2	8
k8s rpi worker	2	8
k8s jetson worker	2	8
Rainbow Dashboard	2	8

For the core package platform package, that does not include complex analytics, RAM can be dropped to 4GB.

Component	#vCPUs	RAM(GB)
k8s master	4	10
k8s worker server	2	4
k8s rpi worker	2	4
k8s jetson worker	2	4
Rainbow Dashboard	2	4

5.2.2 Master Setup

For master setup you will need to follow the steps below:

1. Clone the repository.
2. Go to ***installation-scripts*** directory and give execution rights to all scripts.
3. Update the following lines of the ***rainbow-v2-master.sh*** script with the appropriate values.

```
docker_server="<DOCKER_SERVER>"
docker_username="<DOCKER_USERNAME>"
docker_password="<DOCKER_PASSWORD>"
docker_email="<DOCKER_EMAIL>"
```

4. Execute ***rainbow-v2-master.sh*** script.
5. After script execution it prints out the **kubernetes** and **cjdns** credentials. Those credentials are necessary for the next steps, workers setup and dashboard setup.

5.2.3 Workers Setup

For master setup you will need to follow the steps below:

1. Clone the repository.
2. Go to ***installation-scripts*** directory and give execution rights to all scripts.
3. Update the following lines of the ***rainbow-v2-worker.sh*** script with the appropriate values. *serverIP* is the master's ip, the other values are the **cjdns** credentials from the **Master** setup step.

```
serverIP="<SERVER_IP>"
port="<SERVER_PORT>"
password="<PASSWORD>"
publicKey="<PUBLICKEY>"
serverHostname="<HOSTNAME>"
serverIPv6="<IPv6>"
```

4. Execute ***rainbow-v2-worker.sh*** script.

5.2.4 Special Case

If there are NVIDIA Xavier devices on the cluster go to ***xavier-device*** directory and follow the instructions

5.3 Analytic Stack Setup

This page references the repository under https://gitlab.com/rainbow-project1/rainbow-installation

5.3.1 Data Storage and Analytic Services on Master

For the data storage and processing stack, users initially must execute a docker-compose file on master that includes all necessary services.

Navigate to the repository and go to the ***analytic-stack/master*** directory where the *docker-compose.yaml* file is provided.

Next we provide details about the parameters of the docker-compose file. For providing the details, edit the ***.env*** file inside the directory.

```
NODE_IPV6="..." # Node's IPV6
NODE_IPV4="..." # Node's IPV4
PROVIDER_HOSTS="..." # The IPs of the nodes that the system will retrieve its data (all␣
↪nodes' ips)
NODE_HOSTNAME="..." # The hostname or ip of the node
STORM_NIMBUS_CONFIG_FILE="..." # The path of Storm Nimbus configuration file
STORAGE_PLACEMENT="..." # Enables and disables the placement algorithm of the storage.␣
↪Default is False.
STORAGE_DATA_FOLDER="...." # The folder that the data of storage will be stored
```

Storm Nimbus and Storm UI Configurations

Generally, the configuration of Nimbus needs no alteration. However, users can update the following file accordingly. Furthermore, users can add also other configurations of Storm Framework. Finally, users can introduce other scheduling strategies (including RAINBOW's strategies) via this confiugration file. For instance, if users set storm.scheduler equals to ResourceAwareScheduler and its strategy to be EnergyAwareStrategy, the execution will try to minimize the energy consumption.

Configuration file exists in ***analytic-stack/master/storm*** directory

A typical configuration file is the following:

```
storm.zookeeper.servers:
     - "cluster-head-IP" # update with master's IPv4
nimbus.seeds: [ "cluster-head-IP" ]  # update with master's IPv4
storm.log.dir: "/logs"
storm.local.dir: "/data"
storm.local.hostname: "cluster-head-IP" # update with master's IPv4
supervisor.slots.ports:
     - 6700
     - 6701
     - 6702
     - 6703
```

(continues on next page)

(continued from previous page)

```
nimbus.thrift.max_buffer_size: 20480000
supervisor.thrift.max_buffer_size: 20480000
worker.childopts: "-Xmx%HEAP-MEM%m -Xloggc:artifacts/gc.log -
↪XX:+HeapDumpOnOutOfMemoryError    XX:HeapDumpPath=artifacts/heapdump"
topology.component.cpu.pcore.percent: 1000.0
topology.component.resources.onheap.memory.mb: 512.0
#storm.scheduler: "org.apache.storm.scheduler.resource.ResourceAwareScheduler"
#topology.scheduler.strategy: "eu.rainbowh2020.Schedulers.EnergyAwareStrategy"
```

Stack Execution

In order to execute the stack, users have only to run the following command in the ***analytic-stack/master*** directory.

```
docker-compose up
```

And the system will start all services. We should note that users have to run firstly the services of master and after that all the other storage and processing services on the rest of the nodes.

5.3.2 Monitoring, Data Storage and Analytic Services on Edge Nodes

For the data storage and processing stack on Edge Nodes, users must execute a docker-compose file on all nodes.

Navigate to the repository and go to the ***analytic-stack/nodes*** directory where the *docker-compose.yaml* file is provided.

Next we provide details about the parameters of the docker-compose file.

```
MONITORING_CONFIGURATION_FILE="..." # The path of monitoring agent configuration file
STORAGE_RAINBOW_HEAD="..." # Cluster head's IPV6
STORAGE_NODE_NAME="..." # The hostname or ip of the node
STORAGE_PLACEMENT="..." # Enables and disables the placement algorithm of the storage.␣
↪Default is False.
STORAGE_DATA_FOLDER="...." # The folder that the data of storage will be stored
```

Monitoring Agent Configurations

```
node_id: "node_id"  # user need to provide a node id (e.g. hostname)

sensing-units:
  general-periodicity: 1s # general sensing rate
  DefaultMonitoring: # Node-level metrics will be enable, users can disable this by␣
↪removing it
    periodicity: 1s
#    disabled-groups: # metric-groups that the system will not start at all metric␣
↪groups include CPU, memory, disk, network
##      - "disk"
#    metric-groups: # override the sensing preferences on specific groups
#      - name: "memory"
#        periodicity: 15s # change a static periodicity
#      - name: "cpu"
```

(continues on next page)

(continued from previous page)

```
  UserDefinedMetrics: # specific implementation of sensing interface for user-defined␣
↪metrics
    periodicity: 1s
    sources:
      - "/"
  ContainerMetrics: # Container-level monitoring metrics
    periodicity: 1s

dissemination-units: # users can enable multiple dissemination units however in rainbow␣
↪we use Ignite as storage
 IgniteExporter:
    hostname: ignite-server
    port: 50000

#adaptivity:   # optional adaptivity properties
#  sensing:  # adaptivity in sensing units
#    DockerProbe:  # e.g., enable adaptivity for the container metrics
#      target_name: demo_test|cpu_ptc  # and set as target metric the cpu percentage of␣
↪demo_test container
#      minimum_periodicity: 1
#      maximum_periodicity: 15
#      confidence: 0.95
#  dissemination:  # adaptivity in dissemination
#    all:  # the system sends adaptively all metrics to the storage
#      minimum_periodicity: 1
#      maximum_periodicity: 15
#      confidence: 0.95
#    metric_id:  # or it can send adaptively only specific metrics
#    - minimum_periodicity: 5s
#      maximum_periodicity: 35s
#      confidence: 95
#
```

For more information about the monitoring and its configuration please check its repository https://gitlab.com/rainbow-project1/rainbow-monitoring

Storm Worker Configurations

Generally, we need only to update the IPs of the Storm Worker configurations. However, users can update the following file accordingly to alter the processing characteristics of a node. Specifically, users can add other configurations from Storm Framework (since we utilize storm as execution engine).

Configuration file exists in ***analytic-stack/nodes/storm*** directory.

A typical configuration file is the following:

```
nimbus.seeds: ["cluster-head-IP"] # The cluster head IPV4
ui.port: 8080
storm.zookeeper.servers:
   - "cluster-head-IP" # The cluster head IPV4
storm.local.hostname: "node-IP" # As hostname we need to provide the node's IPV4
supervisor.slots.ports:
```

(continues on next page)

(continued from previous page)

```
    - 6700
    - 6701
    - 6702
    - 6703
```

Stack Execution

In order to execute the stack, users have only to run the following command:

```
docker-compose up
```

For the execution on ARM processors, user need to execute specific docker-compose files. For instance, for 32 bit arm processors users need to run the following command:

```
docker-compose -f docker-compose-arm32.yaml up
```

or for 64 bit arm processors, users should execute the following command:

```
docker-compose -f docker-compose-arm64.yaml up
```

Ports and Networking

Apache Storm needs specific ports to be open namely or users can configure other ports in the storm configurations:

Default Port	Storm Config	Client Hosts/Processes	Server						
2181	storm.zookeeper.port	Nimbus, Supervisors, and Worker processes	Zookeeper						
6627	nimbus.thrift.port	Storm clients, Supervisors, and UI	Nimbus						
6628	supervisor.thrift.port	Nimbus	Supervisors						
8080	ui.port	Client Web Browsers	UI						
8000	logviewer.port	Client Web Browsers	Logviewer						
3772	drpc.port	External DRPC Clients	DRPC						
3773	drpc.invocations.port	Worker Processes	DRPC						
3774	drpc.http.port	External HTTP DRPC Clients	DRPC						
670{0,1,2,3}	supervi-sor.slots.ports	Worker Processes	Worker Processes						

Furthermore, Storage Agents need ports *50000*, *47500*, *47100* to be open as well.

5.4 Dashboard Setup

Here you can find instructions for the Dashboard Setup, which is the last step of the whole setup procedure. This page references the repository under https://gitlab.com/rainbow-project1/rainbow-installation.

For installation a token is required, which can be requested at ktheodosiou@ubitech.eu and avasileiou@ubitech.eu

1. Clone the repository.
2. Go to ***dashboard*** directory and give execution rights to all scripts.
3. Update the following lines of the ***rainbow-dashboard.sh*** script with the appropriate values. *serverIP* is the master's ip, the other values are the **cjdns** credentials from the **Rainbow Setup - Master** setup step.

```
serverIP="<SERVER_IP>"
port="<SERVER_PORT>"
password="<PASSWORD>"
publicKey="<PUBLICKEY>"
serverHostname="<HOSTNAME>"
serverIPv6="<IPv6>"
```

4. Execute ***rainbow-dashboard.sh***
5. Update the following lines of the ***.env*** file with the appropriate values. Server here is the local machine where the setup takes place.

```
SERVER_IP=<SERVER_IP>
SERVER_BASE_PATH=<VOLUME_DATA_PATH>
```

6. Execute the following command

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

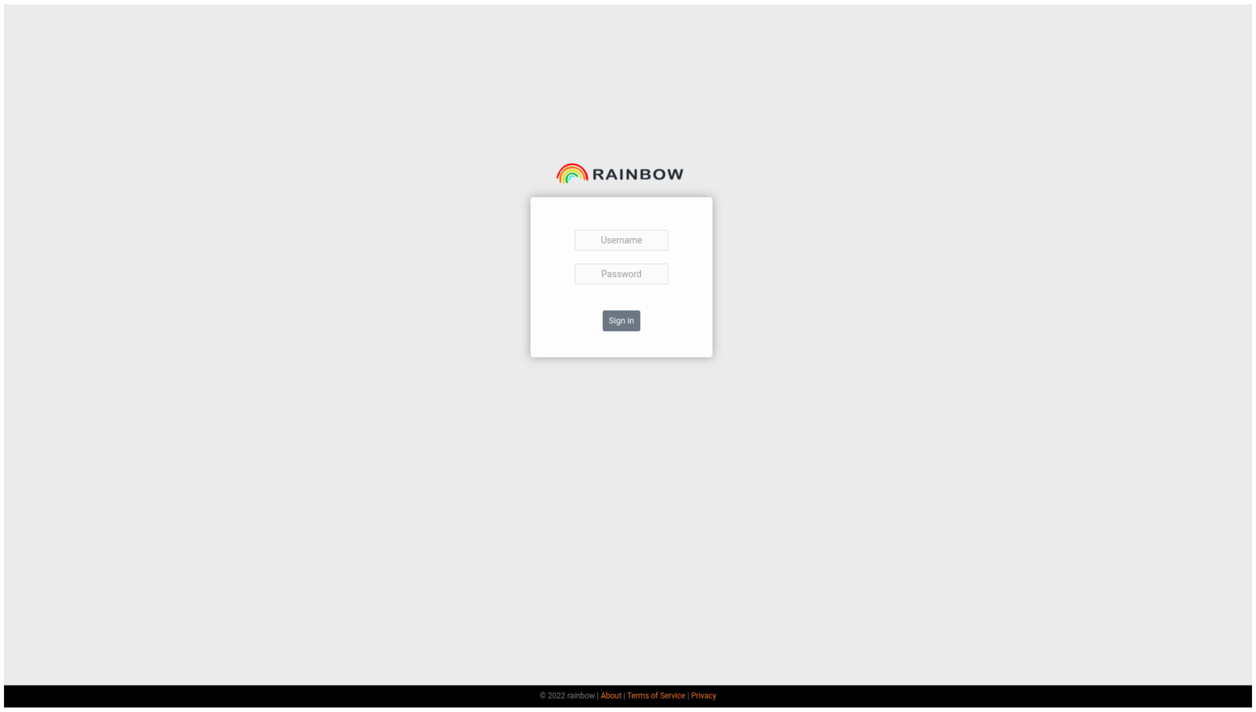
5.5 Getting started with Rainbow Dashboard

Here you can find instructions for the basic functionalities of Rainbow Platform.

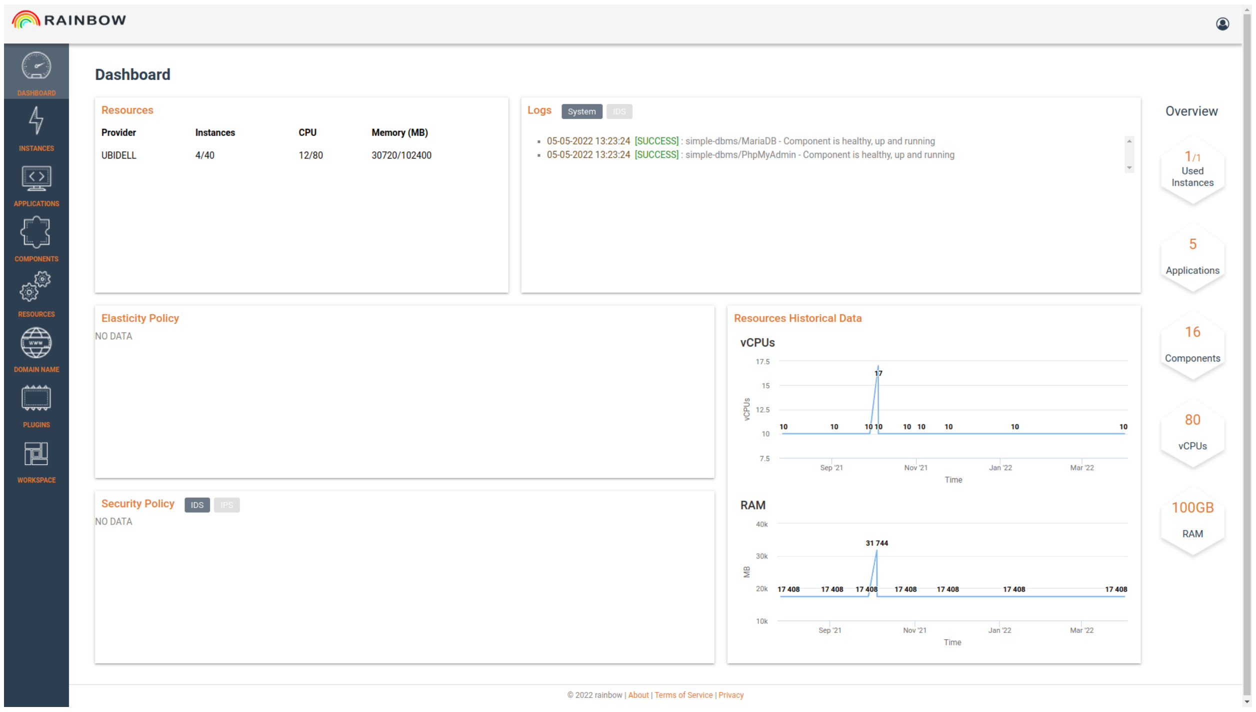
5.5.1 Login/Logout

Login

- Provide your login credentials and click the <SIGN IN> button.



- Upon successful authentication the following screen will be presented.



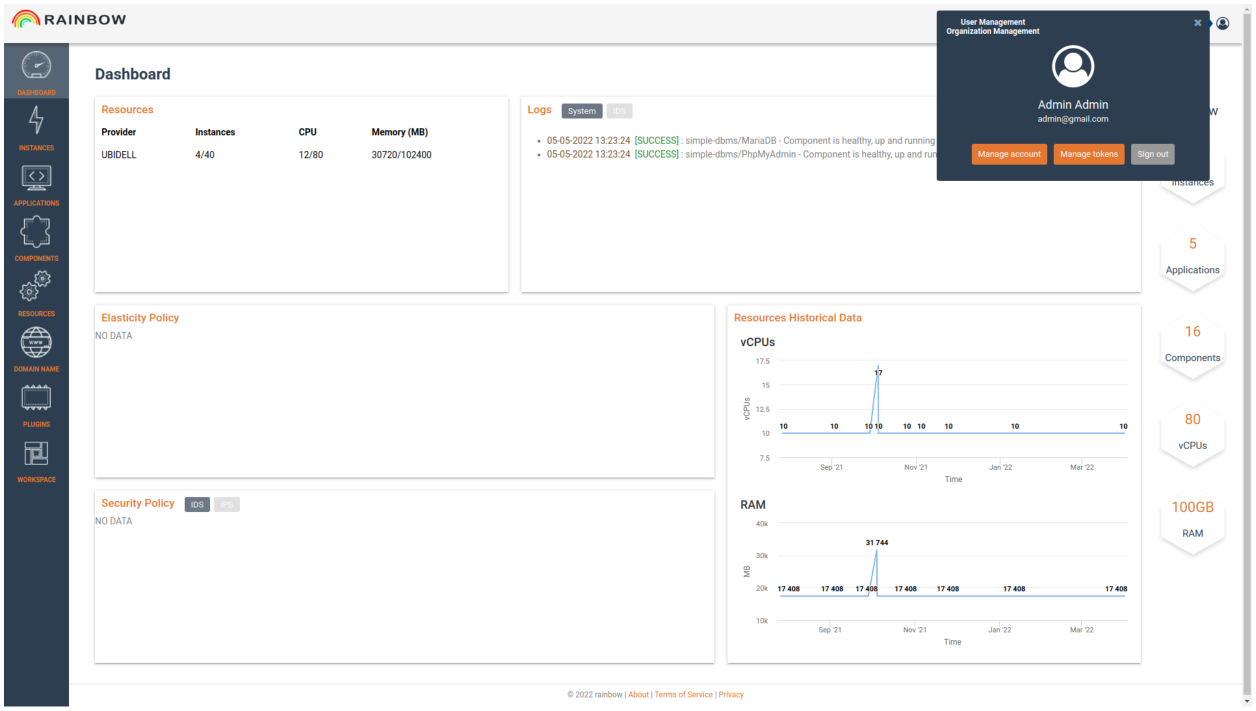
Resources

Provider	Instances	CPU	Memory (MB)
UBIDELL	4/40	12/80	30720/102400

```
• 05-05-2022 13:23:24 [SUCCESS] : simple-dbms/MariaDB - Component is healthy, up and running
• 05-05-2022 13:23:24 [SUCCESS] : simple-dbms/PhpMyAdmin - Component is healthy, up and running
```

Logout

- In order to perform logout click the <Log-out > Button.

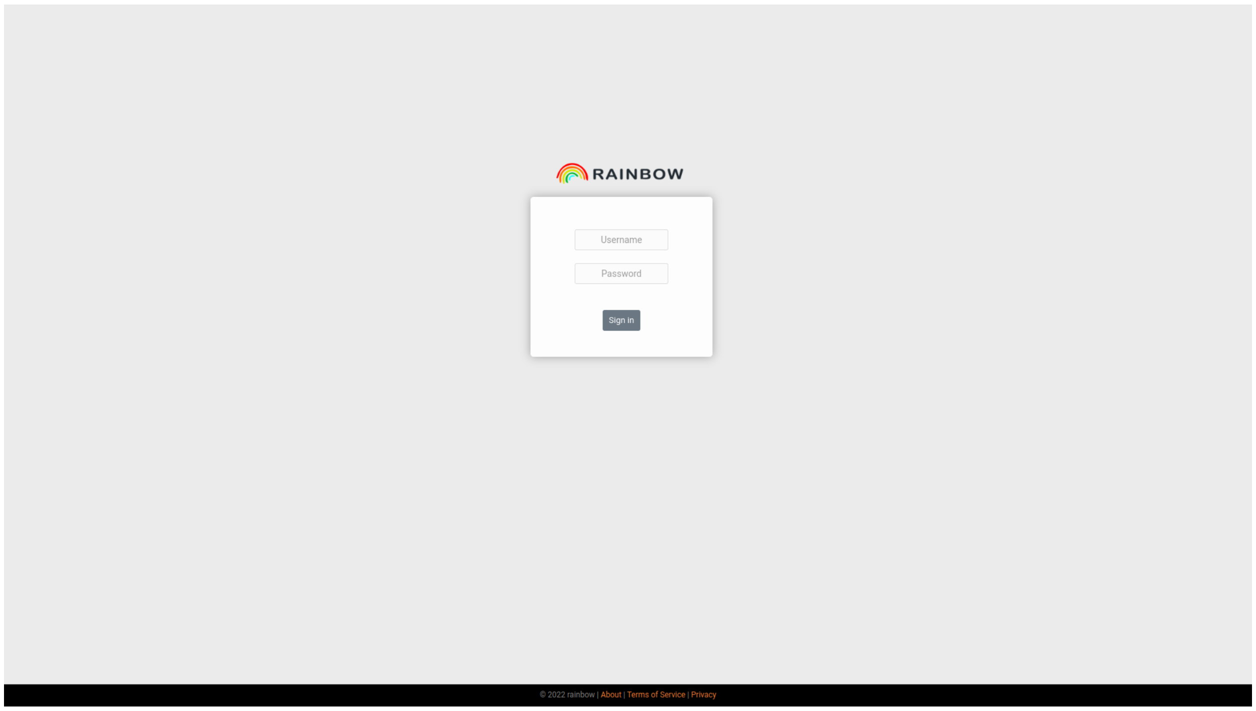


Resources

Provider	Instances	CPU	Memory (MB)
UBIDELL	4/40	12/80	30720/102400

```
• 05-05-2022 13:23:24 [SUCCESS] : simple-dbms/MariaDB - Component is healthy, up and running
• 05-05-2022 13:23:24 [SUCCESS] : simple-dbms/PhpMyAdmin - Component is healthy, up and run
```

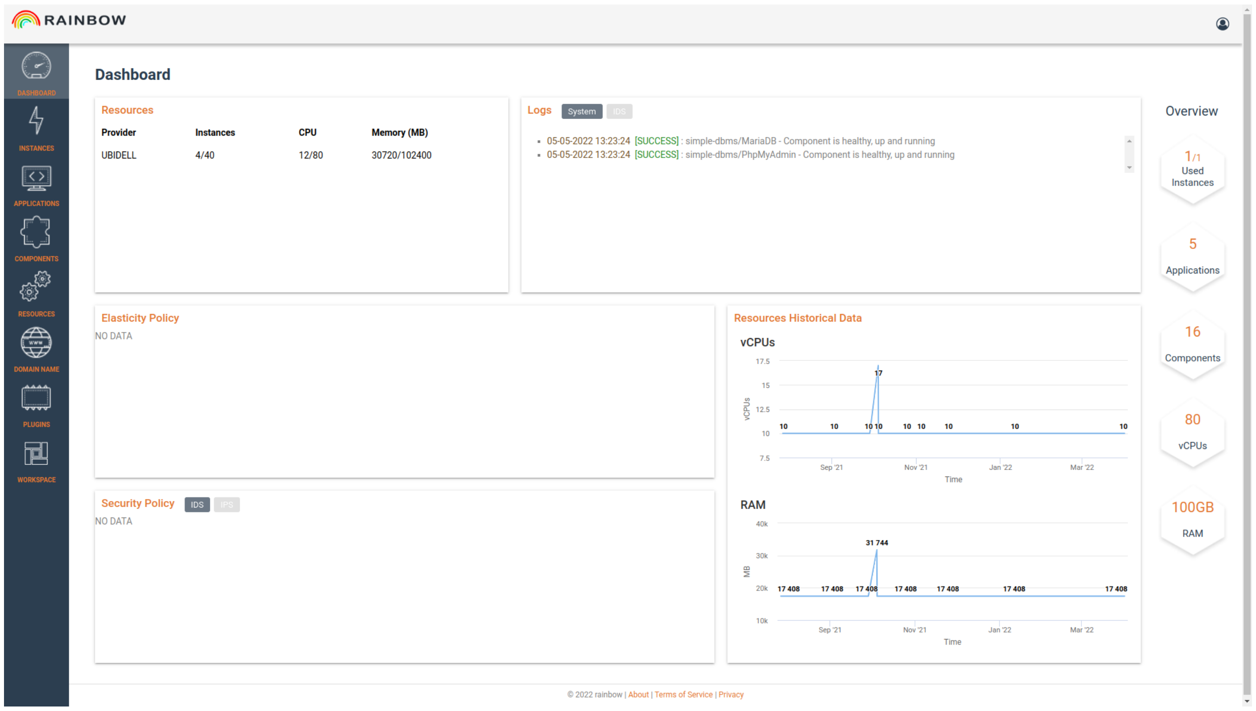
- Upon successful logout the following screen will be presented.



5.5.2 Dashboard Main View

The main dashboard:

- Contains useful information on the cloud resources usage (CPU, Memory, Instances)
- Provides health tracking logs on the running instances
- Lists elasticity and security policies
- Presents a total overview of the existing components, applications, vCPUS and Ram available.



Dashboard

Resources

Provider	Instances	CPU	Memory (MB)
UBIDELL	4/40	12/80	30720/102400

Logs

```
• 05-05-2022 13:23:24 [SUCCESS] : simple-dbms/MariaDB - Component is healthy, up and running
• 05-05-2022 13:23:24 [SUCCESS] : simple-dbms/PhpMyAdmin - Component is healthy, up and running
```

Elasticity Policy

NO DATA

Security Policy

NO DATA

Resources Historical Data

© 2022 rainbow | About | Terms of Service | Privacy

5.5.3 Platform Usage Video

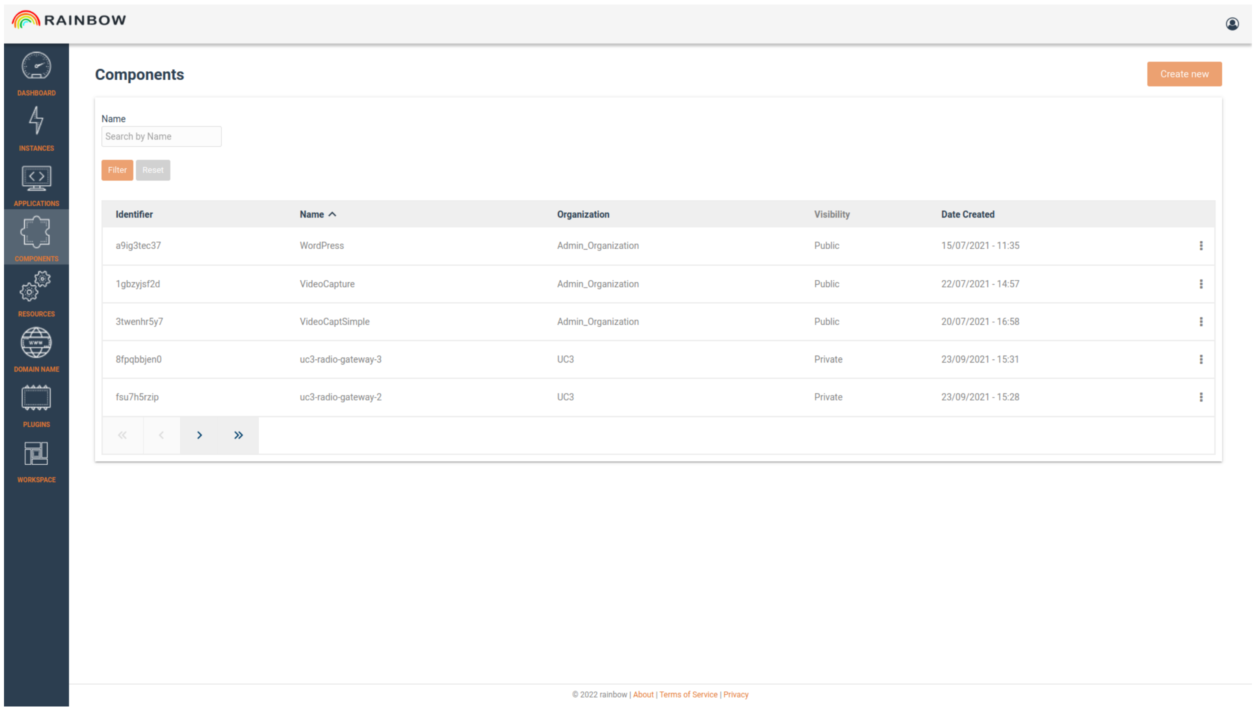
This screencast can also help you get started:

5.6 Components

In Rainbow Platform each service is represented by a Component which is instantiated as a docker process in a unique Virtual Machine. Then applications are formed by connecting individuals components (VMs) together.

5.6.1 View Available Components

By selecting Components from the left sidebar menu, the user is redirected to the list of the available components.



Components

Name
Search by Name

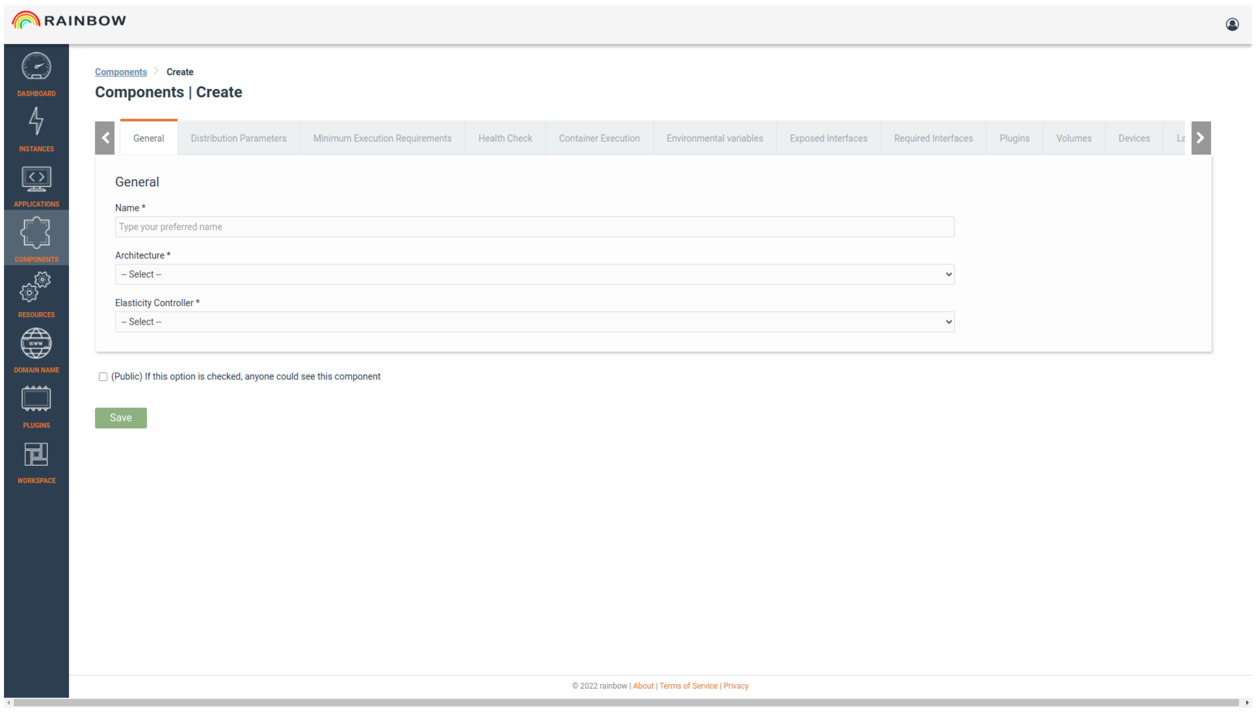
Identifier	Name	Organization	Visibility	Date Created
a9ig3tec37	WordPress	Admin_Organization	Public	15/07/2021 - 11:35
1gbzyjsf2d	VideoCapture	Admin_Organization	Public	22/07/2021 - 14:57
3twenhr5y7	VideoCaptSimple	Admin_Organization	Public	20/07/2021 - 16:58
8fpqbbjen0	uc3-radio-gateway-3	UC3	Private	23/09/2021 - 15:31
fsu7h5rzip	uc3-radio-gateway-2	UC3	Private	23/09/2021 - 15:28

© 2022 rainbow | About | Terms of Service | Privacy

5.6.2 Create a Component

The user can create a new component by clicking on the button <Create new> at the top right corner of the page. Then there is a tabbed form where the configuration of the component takes place.

- General Settings can be configured as the component name, machine architecture, and elasticity controller scale function.



Components | Create

General

Name *
Type your preferred name

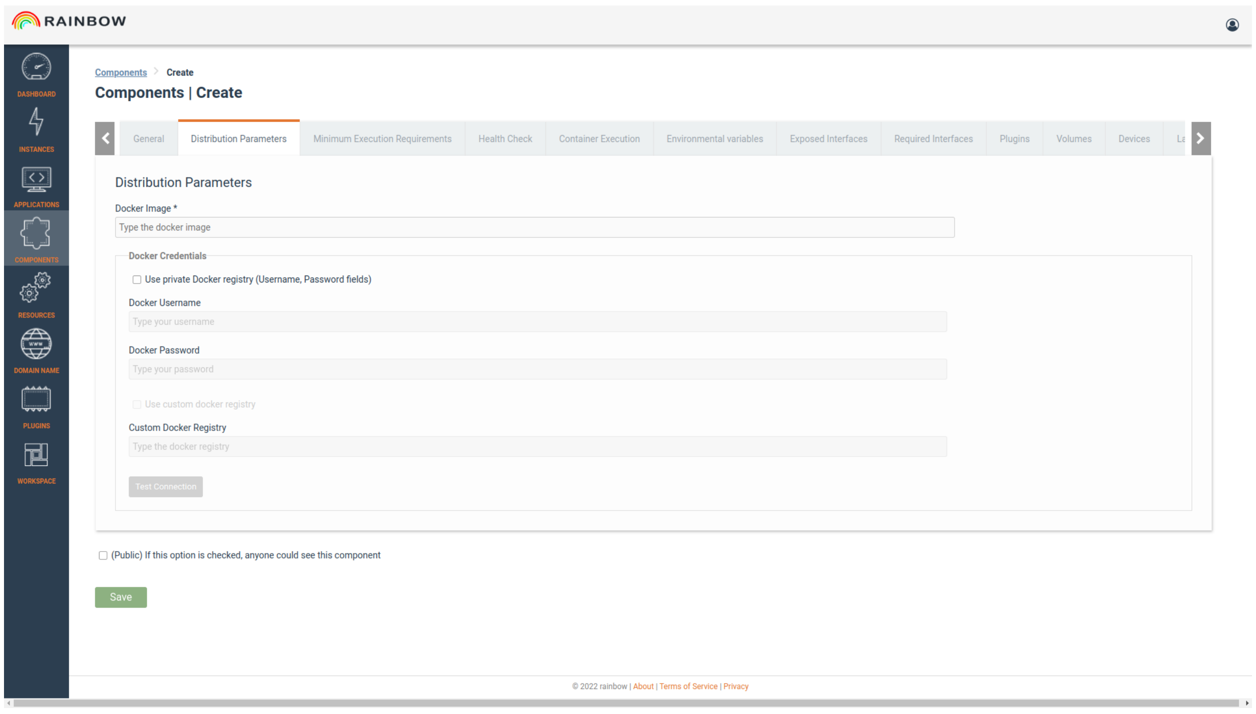
Architecture *
-- Select --

Elasticity Controller *
-- Select --

☐ (Public) If this option is checked, anyone could see this component

© 2022 rainbow | About | Terms of Service | Privacy

- **The "blueprint" of the component, which is the docker image and the registry of it, are set in the Distribution Parameters tab.**
    - Custom Docker registries are supported.



Components | Create

Distribution Parameters

Docker Image *
Type the docker image

Docker Credentials

☐ Use private Docker registry (Username, Password fields)

Docker Username
Type your username

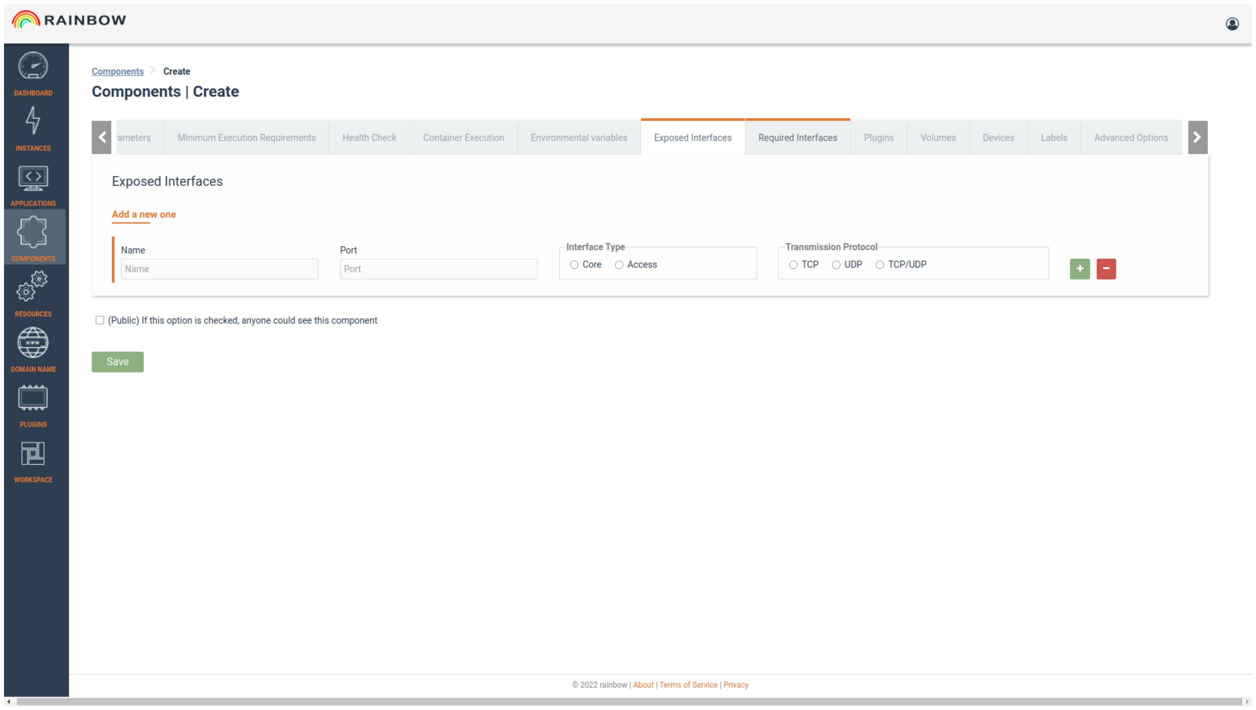
Docker Password
Type your password

☐ Use custom docker registry

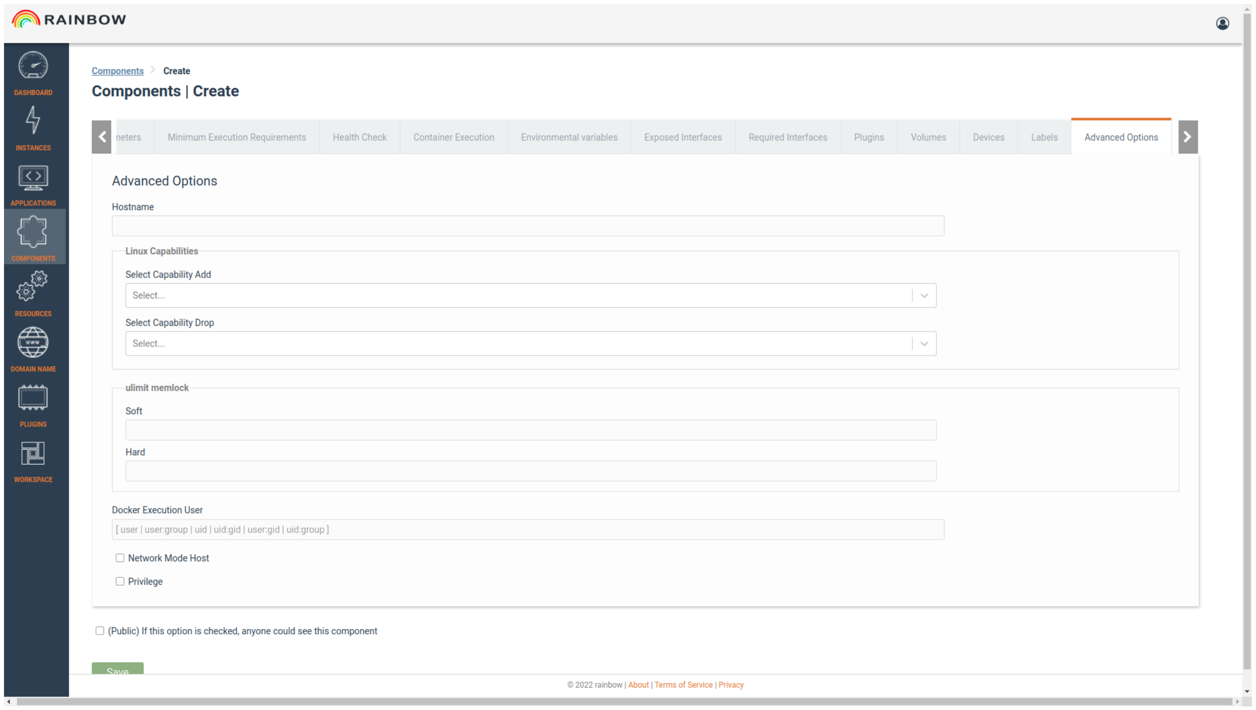
Custom Docker Registry
Type the docker registry

☐ (Public) If this option is checked, anyone could see this component

- **The communication protocol is setting by exposing or requiring interfaces, which is the network port that the Component exposes or requires to communicate.**
    - A user can select one of the existing interfaces, like a TCP access through port 80, or define a new interface.
    - For the definition of the required interface, an existing exposed interface of another component has to be selected.



- Advanced Options can be set such as run the component image in host mode, priviliged mode, add System Capablities, assign hostname



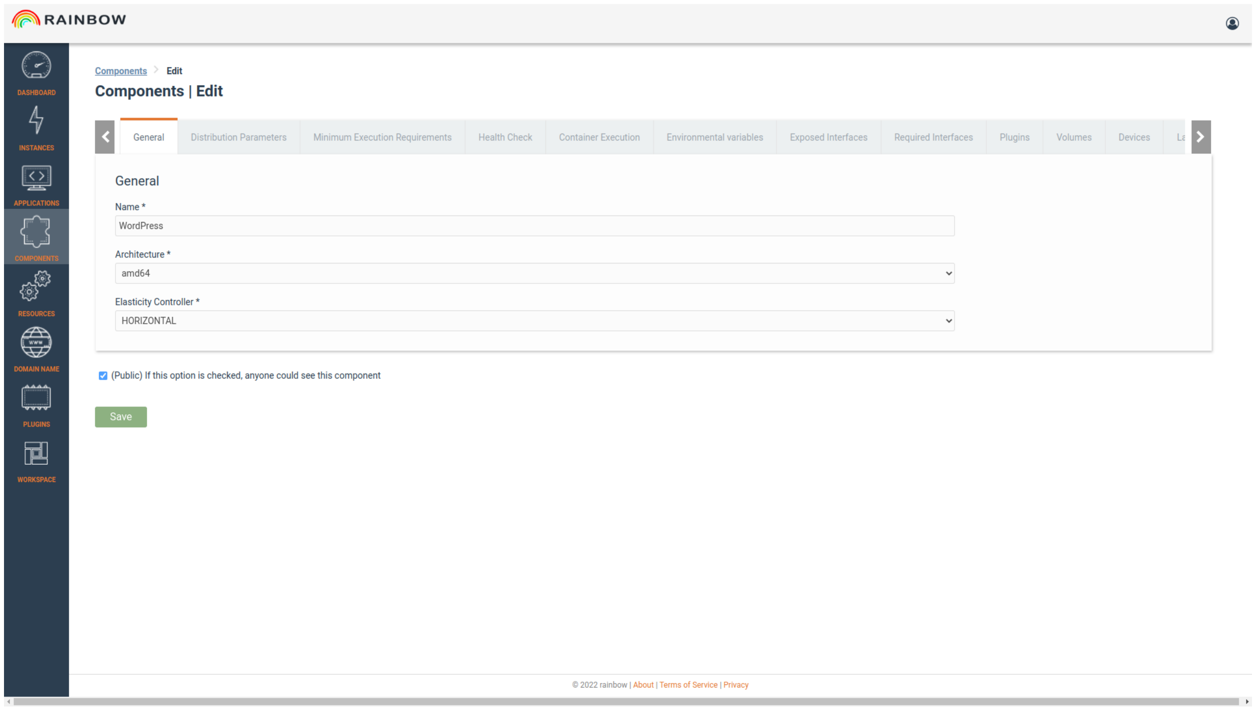
The user can also:

- Specify resources limit for the Component (VCPUs, RAM, Storage, Hypervisor Type)
- Set Enviromental Variables
- Define Health Check Commands and Entrypoint Commands

- Map Volumes
- Add Plugins that are available in the Plugins Section
- Add labels to the component

5.6.3 View/Edit a Component

The user can view and edit the parameters of an existing component by clicking Edit button in the right corner of the item list.



Components | Edit

General

Name *
WordPress

Architecture *
amd64

Elasticity Controller *
HORIZONTAL

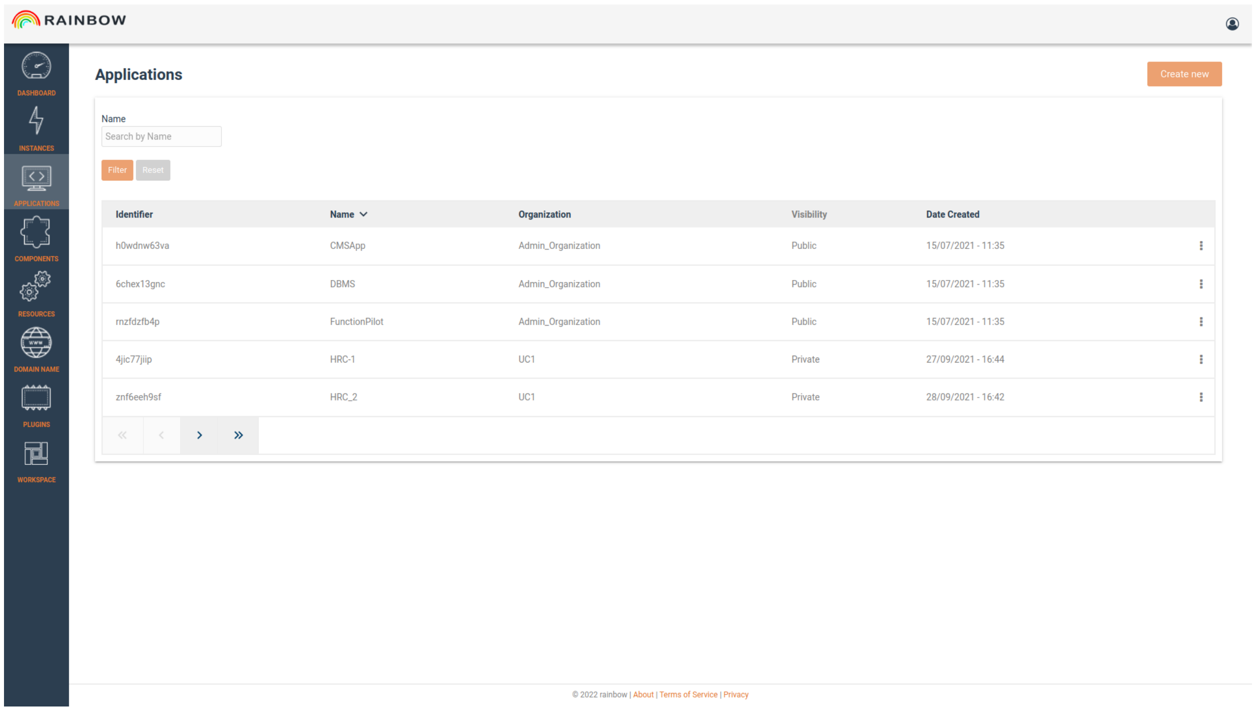
☑ (Public) If this option is checked, anyone could see this component

5.7 Applications

- After the needed components have been defined, the user can proceed with the definition of the application.
- The application will be created through the help of a visual graph editor and then will be available for deployment.

5.7.1 View Available Applications

By selecting Applications from the left sidebar menu, the user is redirected to the list of the available applications.



Applications

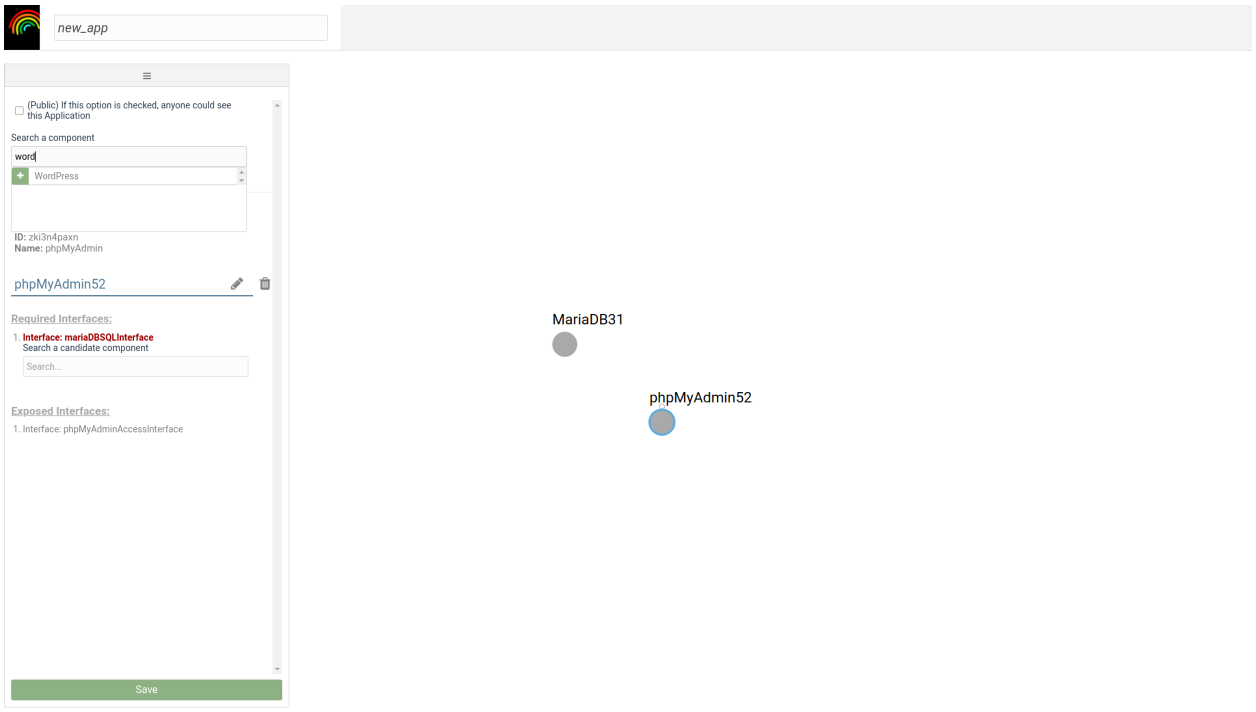
Name
Search by Name

Identifier	Name	Organization	Visibility	Date Created
h0wdnw63va	CMSApp	Admin_Organization	Public	15/07/2021 - 11:35
6chex13gnc	DBMS	Admin_Organization	Public	15/07/2021 - 11:35
rnzfdzfb4p	FunctionPilot	Admin_Organization	Public	15/07/2021 - 11:35
4jic77jiip	HRC-1	UC1	Private	27/09/2021 - 16:44
znf6eeh9sf	HRC_2	UC1	Private	28/09/2021 - 16:42

© 2022 rainbow | About | Terms of Service | Privacy

5.7.2 Create An Application

- To create a new Application, the user has to select <Create new> at the top right corner of the page.
- The user is redirected to the visual edito, where the application components are presented as the nodes of a graph.
- The connection between the nodes is describing the interfaces between the components - services.



(Public) If this option is checked, anyone could see this Application

Search a component
word

ID: zki3n4paxn
Name: phpMyAdmin

phpMyAdmin52

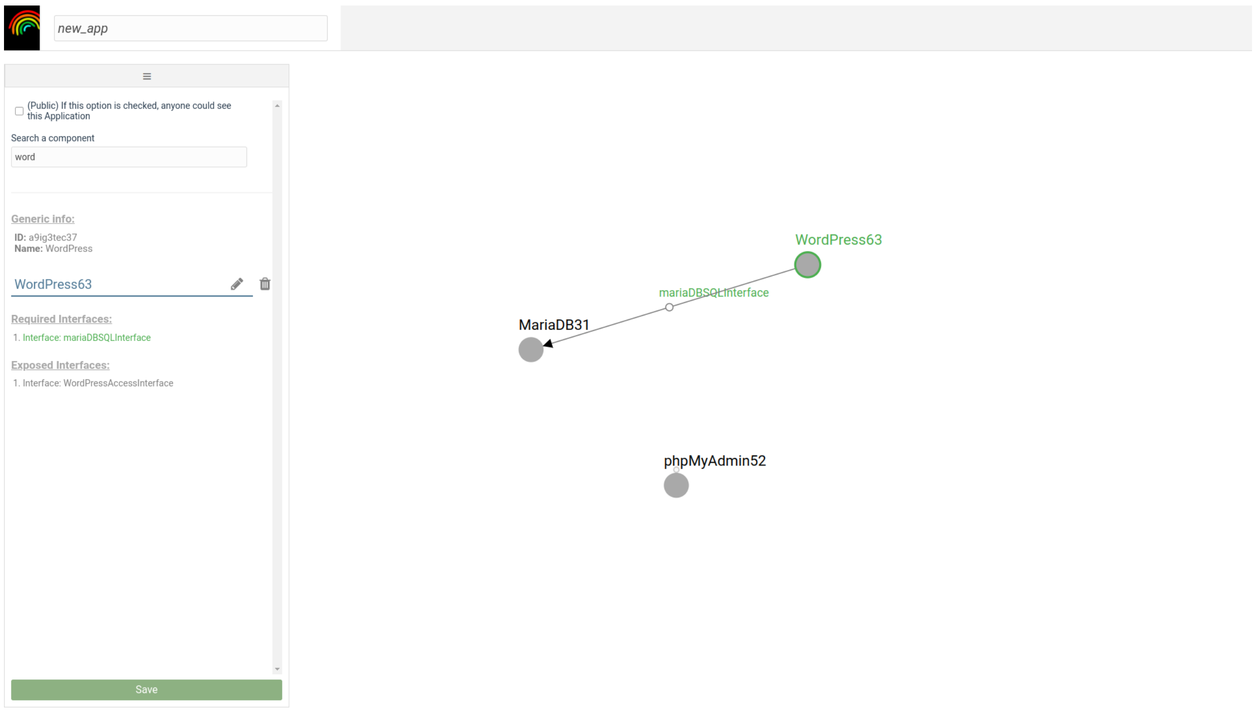
Required Interfaces:

1. Interface: mariaDBSQLInterface
   Search a candidate component
   Search...

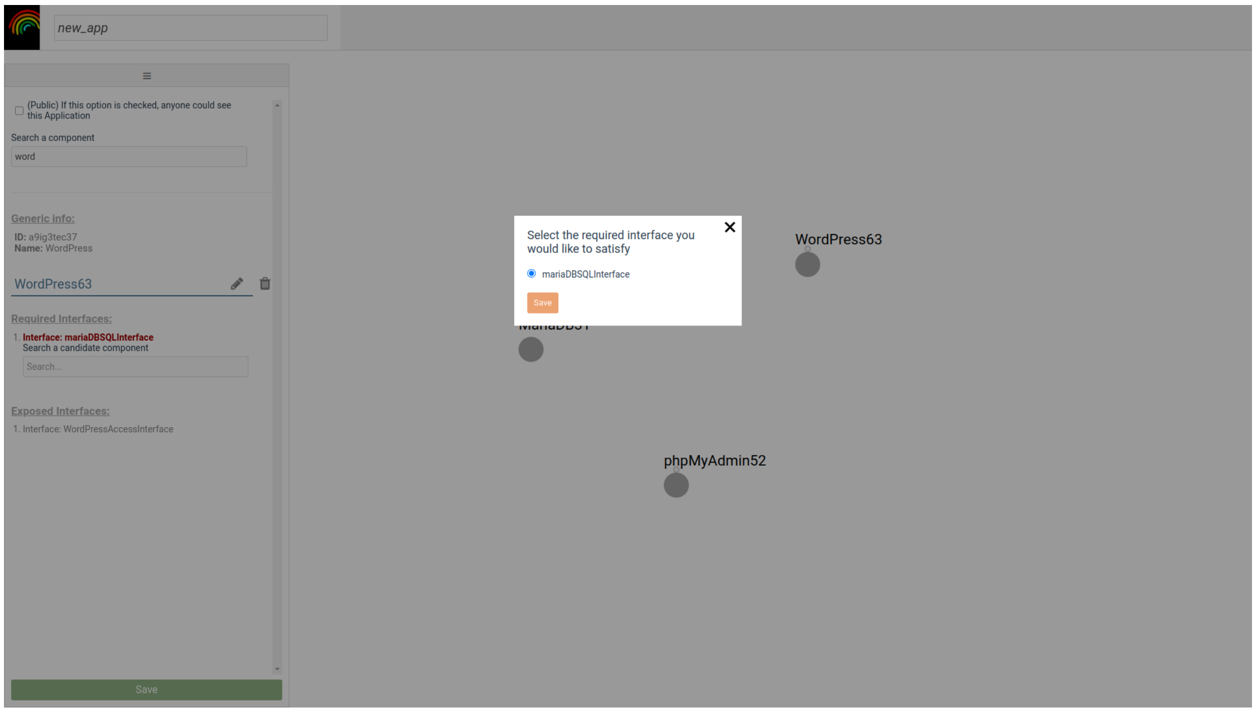
Exposed Interfaces:

1. Interface: phpMyAdminAccessInterface

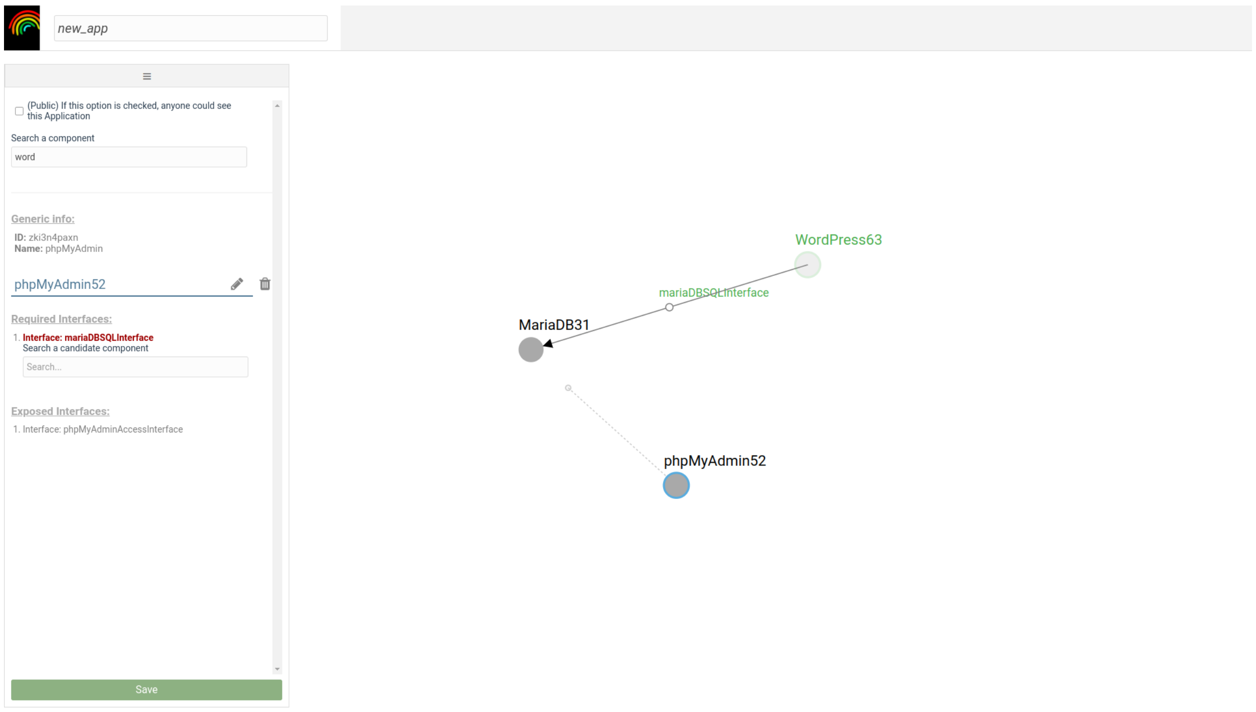
- Through the left side panel, the components can be retrieved and added to the editor.



- By selecting the required interface and dragging it to another node, the connection between the interfaces of the components can be done.


new_app
≡
(Public) If this option is checked, anyone could see this Application
Search a component
word
Generic info:
ID: a9ig3tec37
Name: WordPress
WordPress63
Required Interfaces:
1. Interface: mariaDBSQLInterface
Search a candidate component
Search...
Exposed Interfaces:
1. Interface: WordPressAccessInterface
Save
Select the required interface you would like to satisfy
mariaDBSQLInterface
Save
WordPress63
phpMyAdmin52

- This procedure is followed until all required interfaces have been connected in order to save a valid application graph.



new_app

(Public) If this option is checked, anyone could see this Application

Search a component: word

Generic info:

ID: zki3n4paxn
Name: phpMyAdmin

phpMyAdmin52

Required Interfaces:

1. **Interface: mariaDBSQLInterface**
   Search a candidate component
   Search...

Exposed Interfaces:

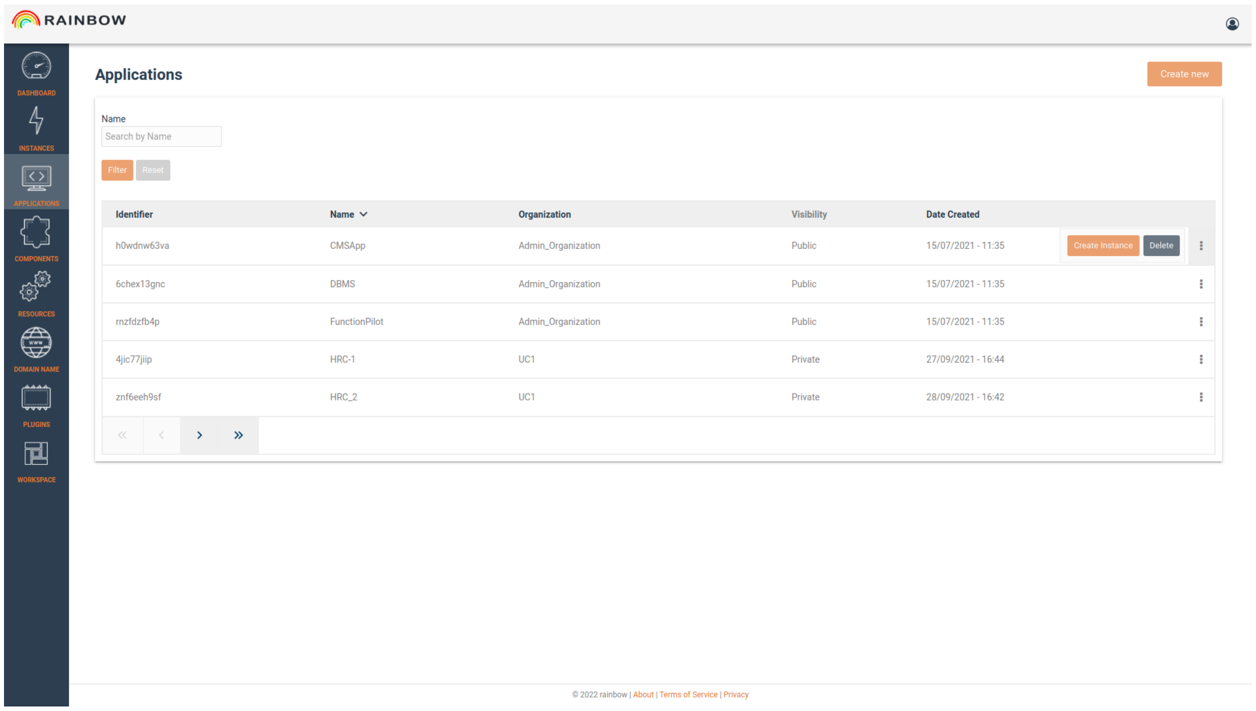
1. Interface: phpMyAdminAccessInterface

5.8 Application Instances

After an application has been configured, an instance of it can be deployed to the cloud.

5.8.1 Create an Application Instance

- To create a new Application Instance, the user has to select <Create Instance> at the top right side of application list item.



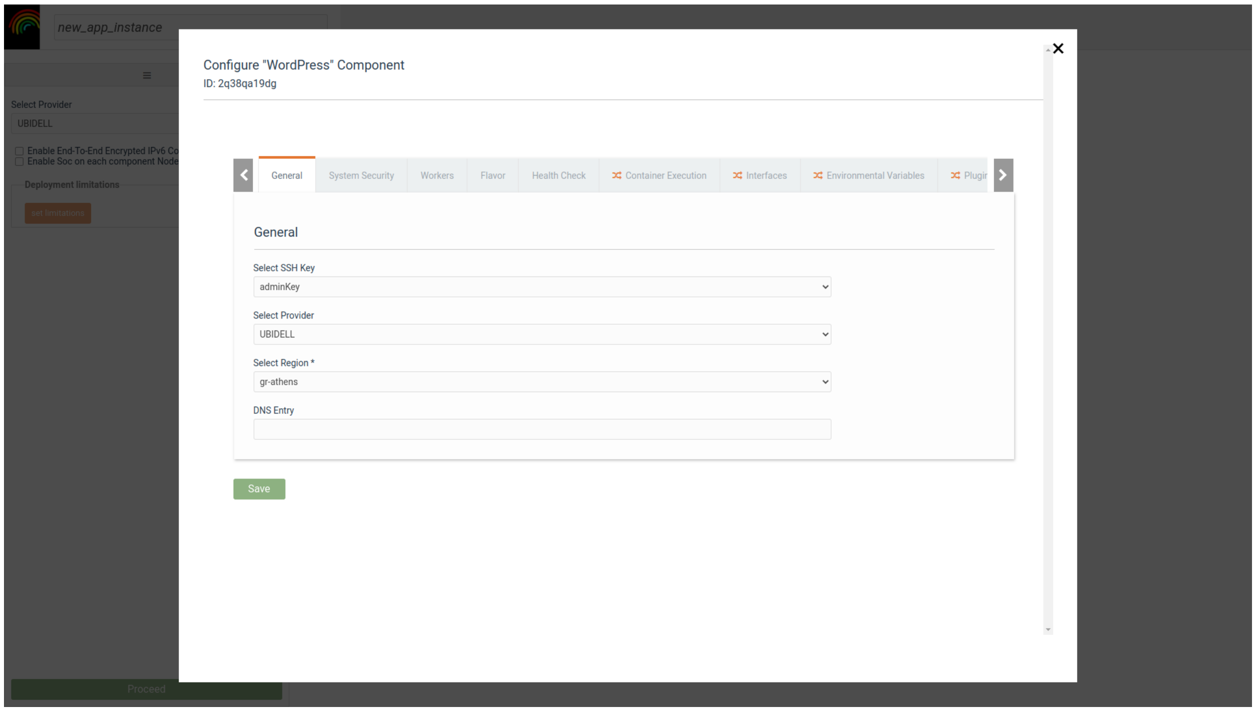
Applications

Name: Search by Name

Identifier	Name	Organization	Visibility	Date Created
h0wdnw63va	CMSApp	Admin_Organization	Public	15/07/2021 - 11:35
6chex13gnc	DBMS	Admin_Organization	Public	15/07/2021 - 11:35
rnzfdzfb4p	FunctionPilot	Admin_Organization	Public	15/07/2021 - 11:35
4jic77jiip	HRC-1	UC1	Private	27/09/2021 - 16:44
znf6eeh9sf	HRC_2	UC1	Private	28/09/2021 - 16:42

Prior to the deployment some settings can be configured:

- Genereal settings concerning the deployment infrastructure.



Configure "WordPress" Component
ID: 2q38qa19dg

General

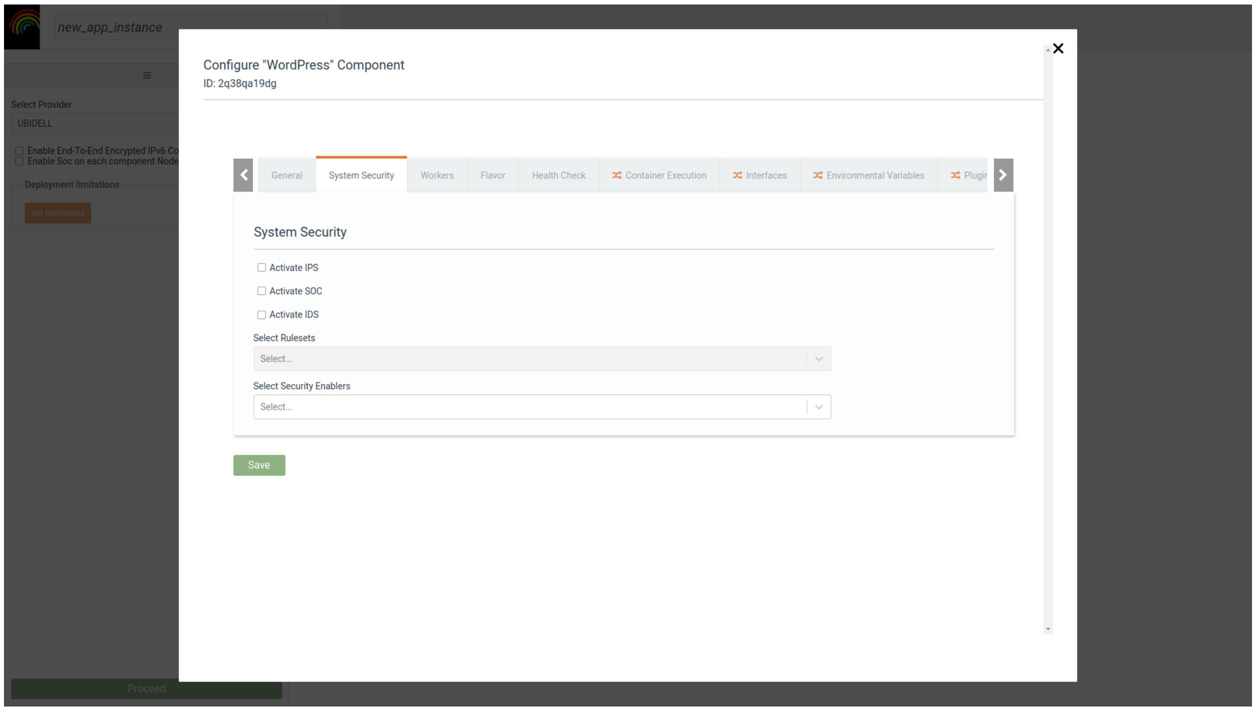
Select SSH Key: adminKey

Select Provider: UBIDELL

Select Region *: gr-athens

DNS Entry:

- Security mechanisms such as Intrusion Prevention Mechanism (IPS), Security Operations Center (SOC), Intrusion Detection Mechanism (IDS)



Configure "WordPress" Component
ID: 2q38qa19dg

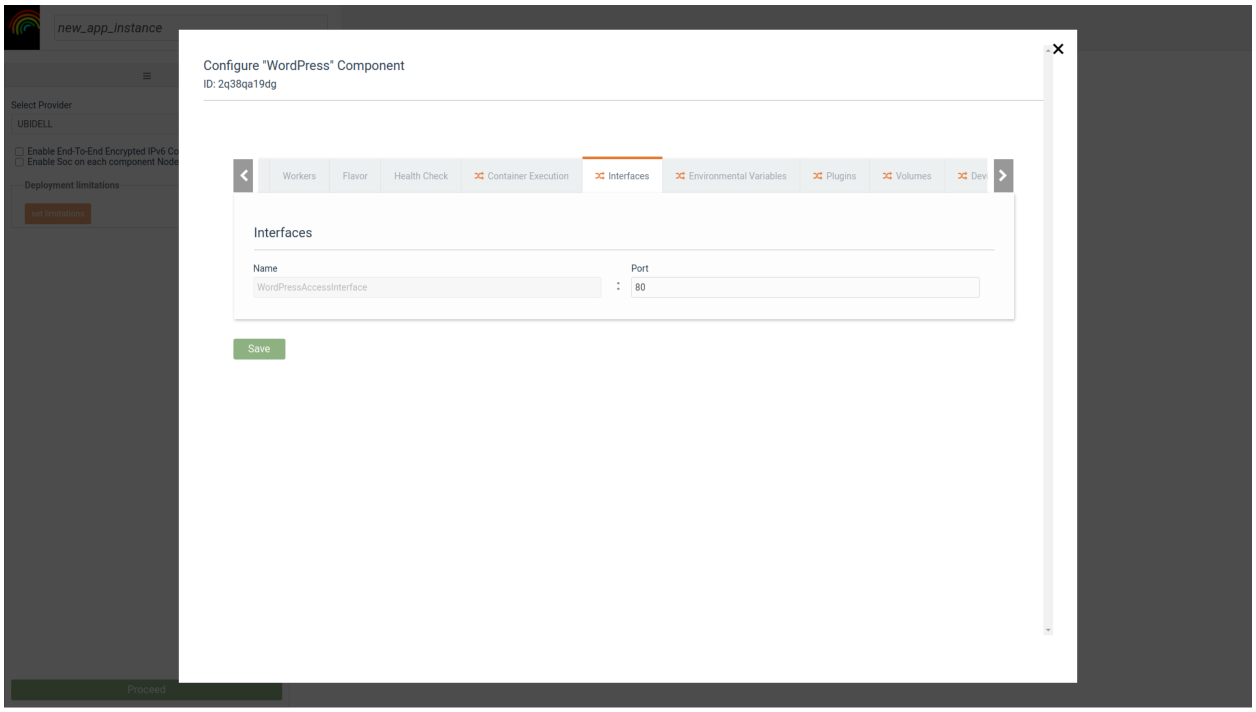
System Security

- [ ] Activate IPS
- [ ] Activate SOC
- [ ] Activate IDS

Select Rulesets: Select...

Select Security Enablers: Select...

- Interface parameters

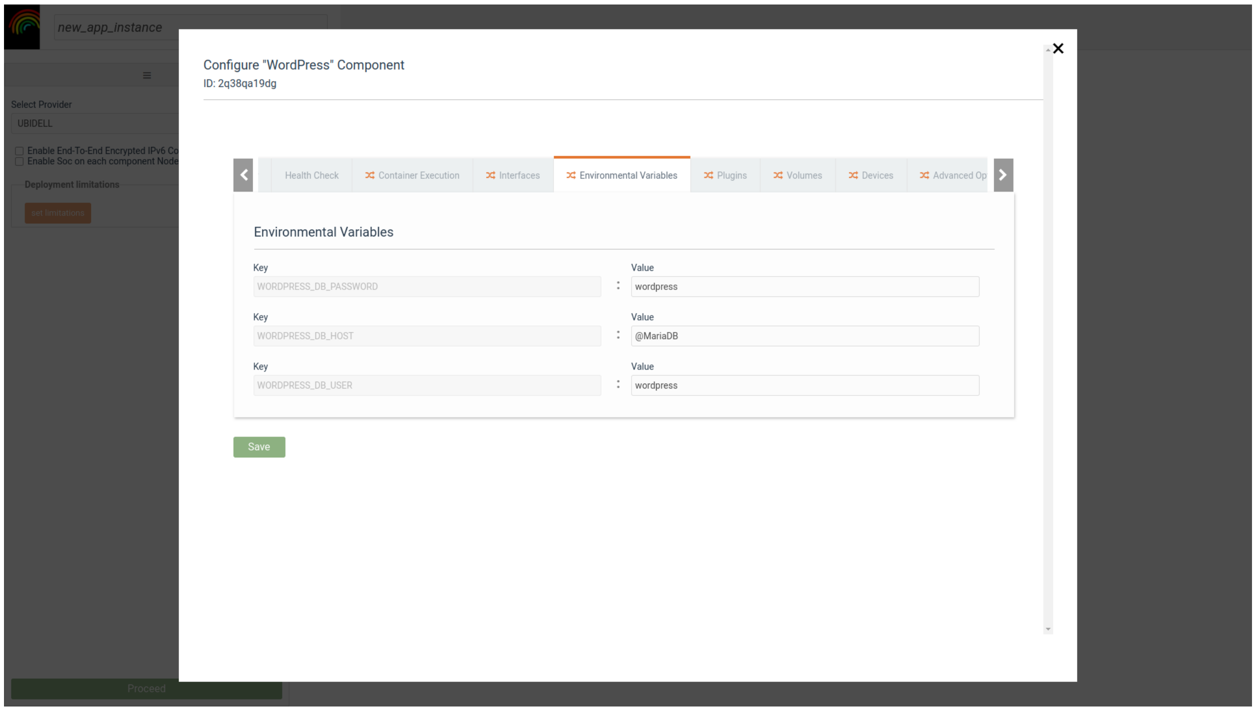


Configure "WordPress" Component
ID: 2q38qa19dg

Interfaces

Name	Port
WordPressAccessInterface	80

- Environmental variables of the components.

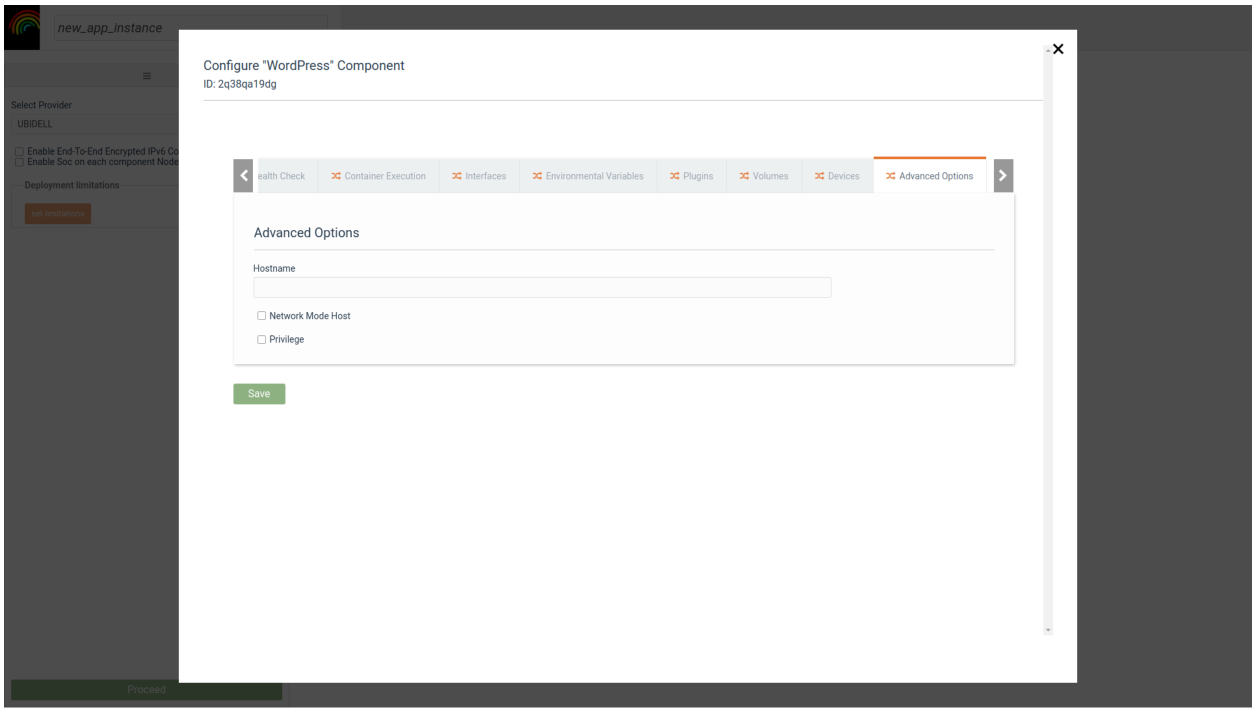


Configure "WordPress" Component
ID: 2q38qa19dg

Environmental Variables

Key		Value
WORDPRESS_DB_PASSWORD	:	wordpress
WORDPRESS_DB_HOST	:	@MariaDB
WORDPRESS_DB_USER	:	wordpress

- Advanced options such as network mode host or priviliged mode



Configure "WordPress" Component
ID: 2q38qa19dg

Advanced Options

Hostname

☐ Network Mode Host

☐ Privilege

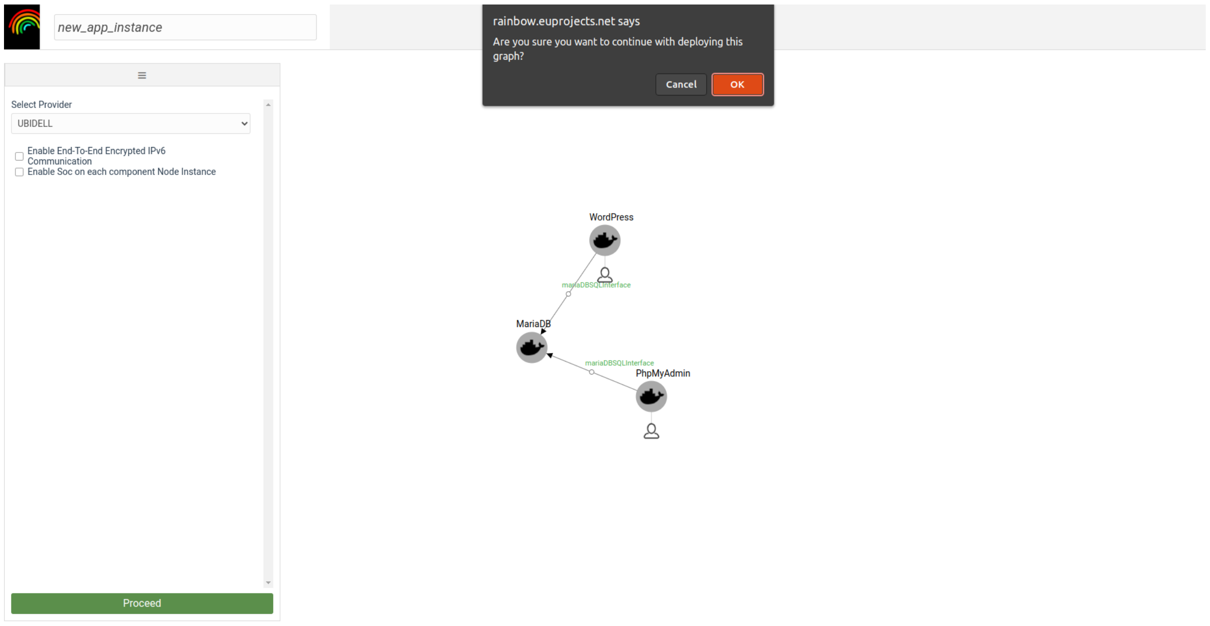
The user can also:

- Specify the minimum and maximum amount of workers per node that control the scalability profile of the application.
- Set Health Check Commands and Entrypoint Execution Commands
- Map Volumes

- Mange Component Plugins
- Manage components' labels

5.8.2 Deploy an Application Instance

- By pressing "Proceed" the deployment starts.



rainbow.euprojects.net says

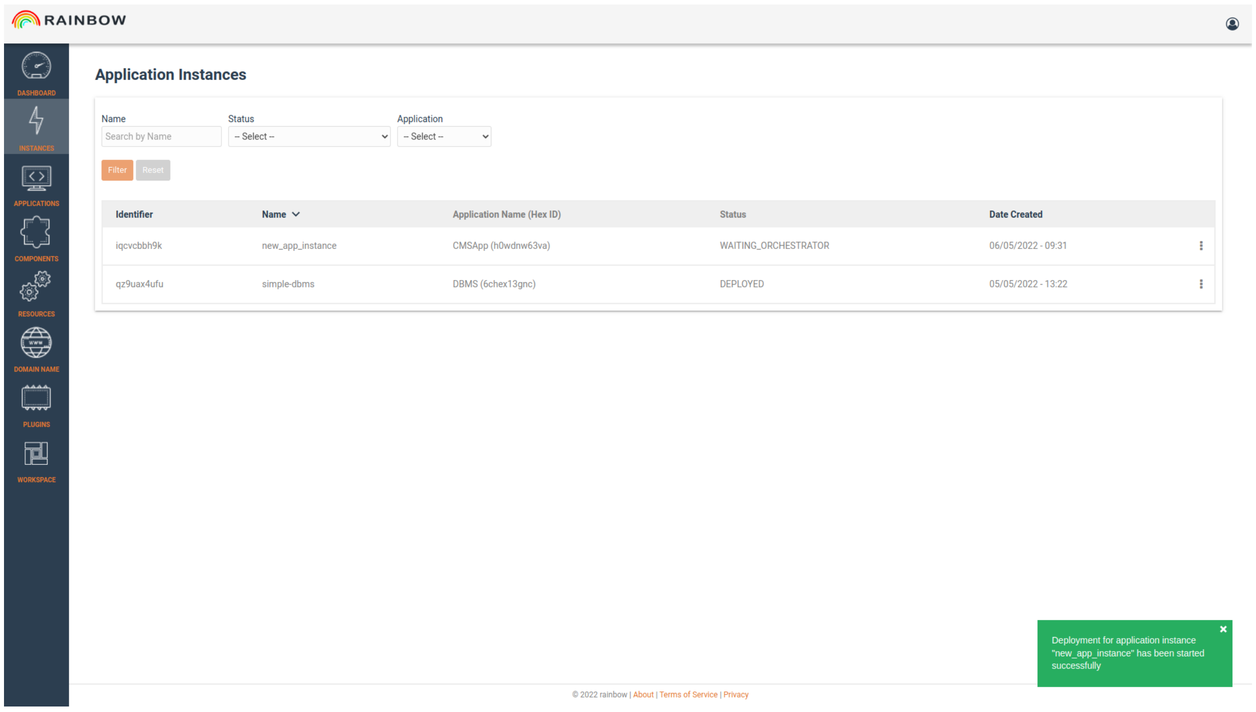
Are you sure you want to continue with deploying this graph?

Select Provider: UBIDELL

☐ Enable End-To-End Encrypted IPv6 Communication

☐ Enable Soc on each component Node Instance

- At the Instances View, the user can see the list of deployed instances, identifiers and their status.



Application Instances

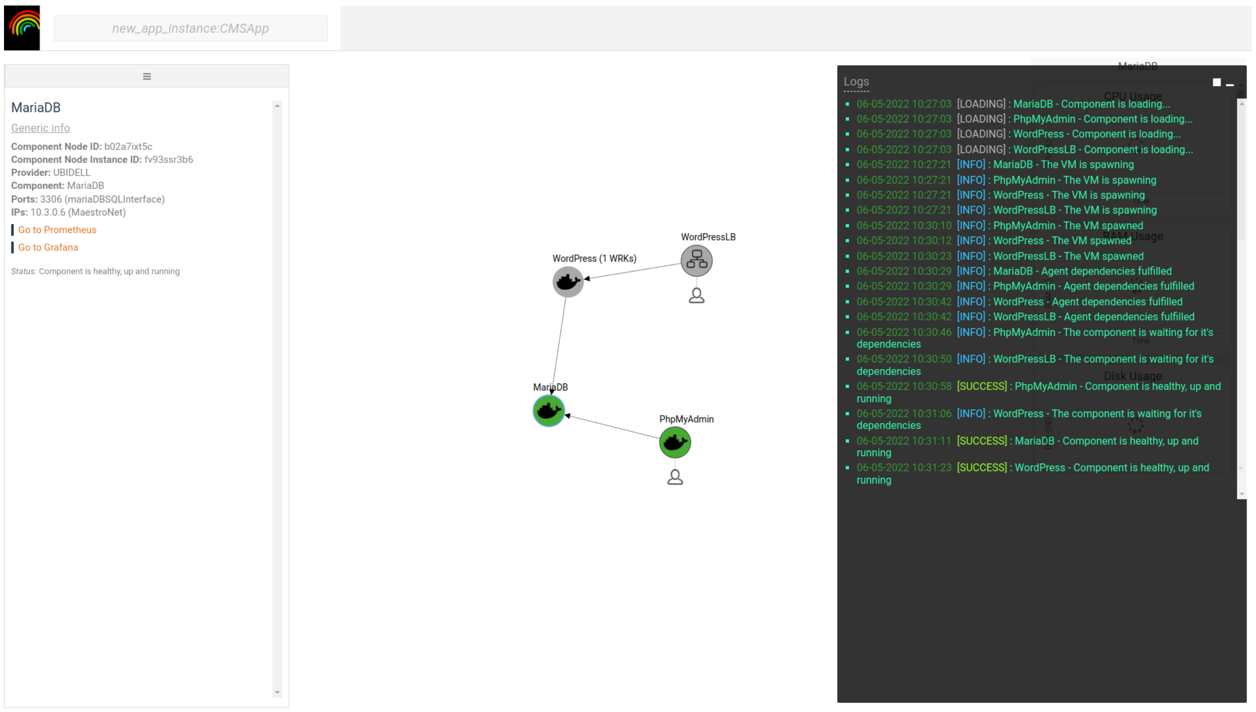
Name: Search by Name — Status: -- Select -- — Application: -- Select --

Identifier	Name	Application Name (Hex ID)	Status	Date Created
iqcvcbbh9k	new_app_instance	CMSApp (h0wdnw63va)	WAITING_ORCHESTRATOR	06/05/2022 - 09:31
qz9uax4ufu	simple-dbms	DBMS (6chex13gnc)	DEPLOYED	05/05/2022 - 13:22

Deployment for application instance "new_app_instance" has been started successfully

© 2022 rainbow | About | Terms of Service | Privacy

- **Deployment procedure needs few minutes to finish. The user is constantly informed by viewing the logs aggregated from all the nodes of the application.**
  - The total deployment time depends on the cloud infrastructure selected, as the spawning of new VMs might take more time in some IaaS.
  - Total time is also affected by the network delays between the cloud infrastructure and the docker registry that is used to fetch the components container image.



MariaDB

Generic info

Component Node ID: b02a7ixt5c
Component Node Instance ID: fv93ssr3b6
Provider: UBIDELL
Component: MariaDB
Ports: 3306 (mariaDBSQLInterface)
IPs: 10.3.0.6 (MaestroNet)

Go to Prometheus

Go to Grafana

Status: Component is healthy, up and running

Logs

```
06-05-2022 10:27:03 [LOADING] : MariaDB - Component is loading...
06-05-2022 10:27:03 [LOADING] : PhpMyAdmin - Component is loading...
06-05-2022 10:27:03 [LOADING] : WordPress - Component is loading...
06-05-2022 10:27:03 [LOADING] : WordPressLB - Component is loading...
06-05-2022 10:27:21 [INFO] : MariaDB - The VM is spawning
06-05-2022 10:27:21 [INFO] : PhpMyAdmin - The VM is spawning
06-05-2022 10:27:21 [INFO] : WordPress - The VM is spawning
06-05-2022 10:27:21 [INFO] : WordPressLB - The VM is spawning
06-05-2022 10:30:10 [INFO] : PhpMyAdmin - The VM spawned
06-05-2022 10:30:12 [INFO] : WordPress - The VM spawned
06-05-2022 10:30:23 [INFO] : WordPressLB - The VM spawned
06-05-2022 10:30:29 [INFO] : MariaDB - Agent dependencies fulfilled
06-05-2022 10:30:29 [INFO] : PhpMyAdmin - Agent dependencies fulfilled
06-05-2022 10:30:42 [INFO] : WordPress - Agent dependencies fulfilled
06-05-2022 10:30:42 [INFO] : WordPressLB - Agent dependencies fulfilled
06-05-2022 10:30:46 [INFO] : PhpMyAdmin - The component is waiting for it's
dependencies
06-05-2022 10:30:50 [INFO] : WordPressLB - The component is waiting for it's
dependencies
06-05-2022 10:30:58 [SUCCESS] : PhpMyAdmin - Component is healthy, up and
running
06-05-2022 10:31:06 [INFO] : WordPress - The component is waiting for it's
dependencies
06-05-2022 10:31:11 [SUCCESS] : MariaDB - Component is healthy, up and
running
06-05-2022 10:31:23 [SUCCESS] : WordPress - Component is healthy, up and
running
```

- **When deployment finishes all nodes turn green**
  - On the instance list the application is shown as "DEPLOYED".

5.9 Application Monitoring and Scaling

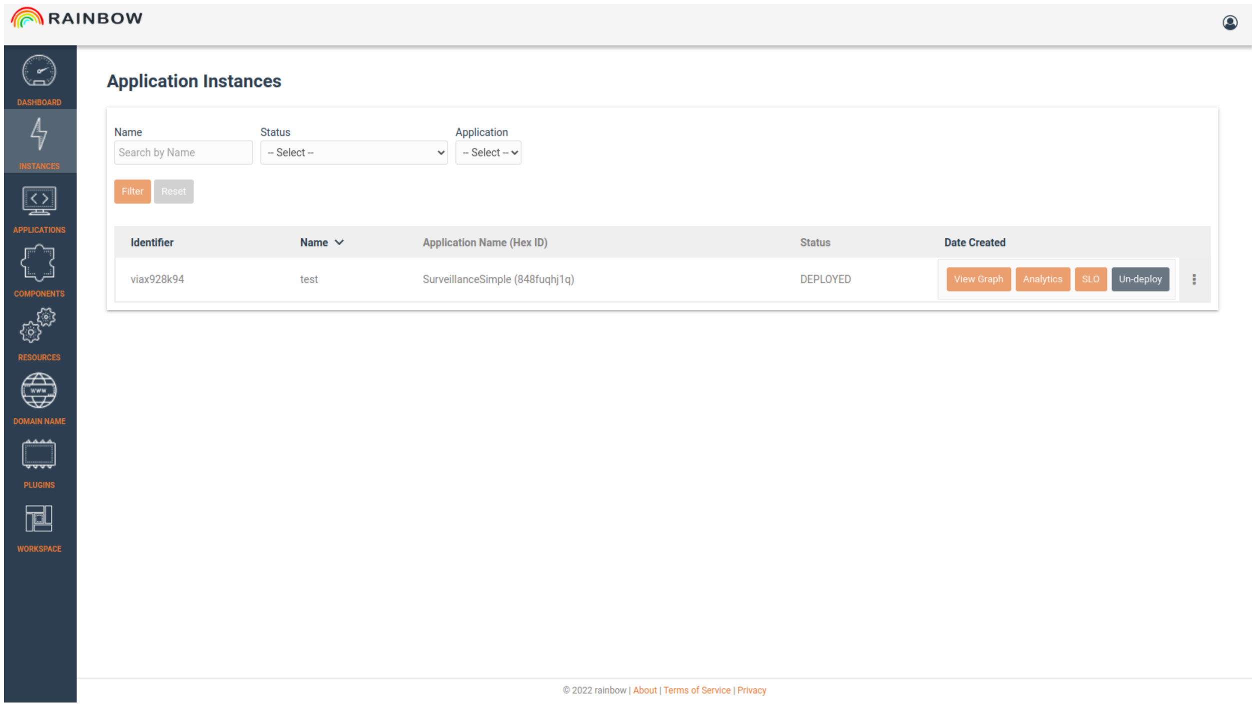
5.9.1 Monitoring Applications instances

- Monitoring metrics are presented for each one of the application nodes

5.9.2 Application Elasticity

- From the application instance list, the user must select the "Elasticity Policies" option for the deployed application, in order to configure how the application scales.
- By selecting the appropriate function, user can to aggregate the monitoring results in various ways.
- For the monitored parameter we select the metric and it's dimension from appropriate lists.
- An operand shall be added to the policy and the threshold that the policy shall confirm to.
- The period field is used to set the size of the time window that the metric values are collected and aggregated for the policy enforcement.

- **On the scaling action we can select the component to scale in or scale out, and the number of workers to scale.**
  - After a scaling action is performed, some time is needed for having the component workers deployed. For this reason we should ensure that for this period we don't fire additional scaling actions.
  - This is done through the "Inertia" field that is used to define the time in minutes that after a scaling action is done,no further action is performed.
- Multiple scaling actions can be added.
- The policy can be saved and will be enforced to the application within few seconds.
- In this example we had initially only one worker of the WordPress component.
- **But due to the scaling rule, an additional node has been created.**
  - A load balancer had been already deployed from the initial deployment since we had defined that this component might need multiple workers.
  - The scaling action is visible to the user through the logs and the number on workers in the "WordPress" node in the graphs.

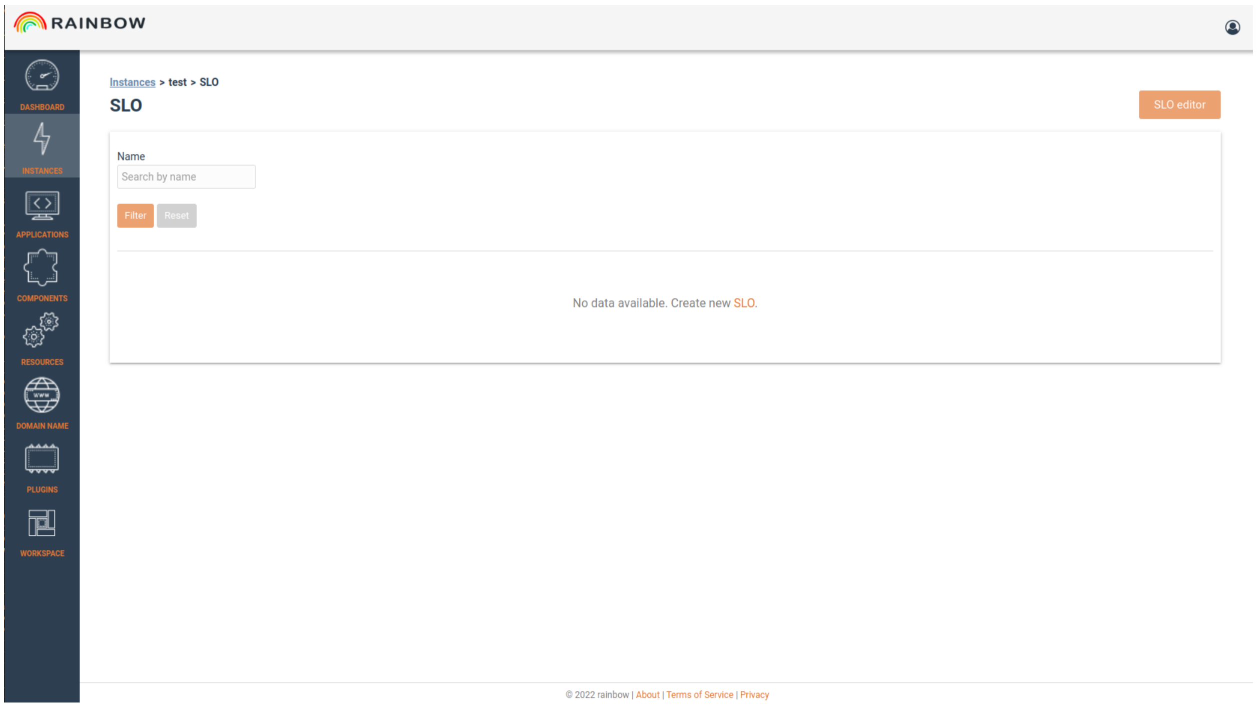


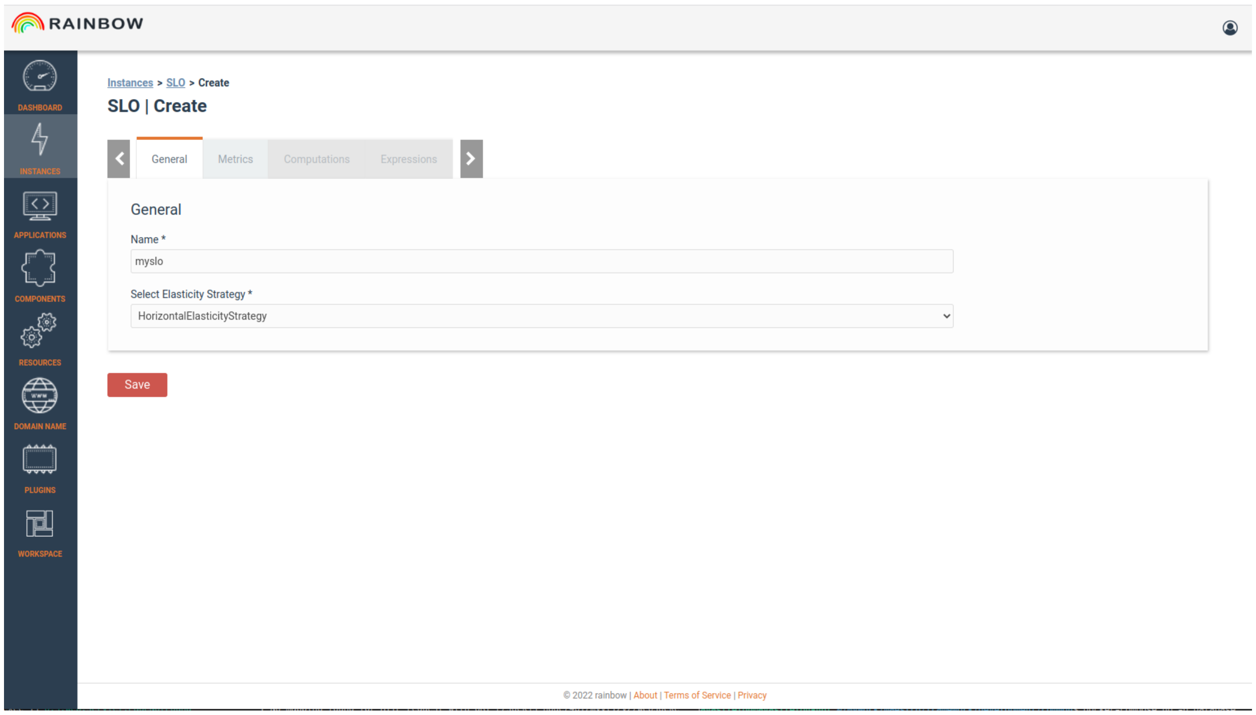
Application Instances

Name: Search by Name
Status: -- Select --
Application: -- Select --

Identifier	Name	Application Name (Hex ID)	Status	Date Created
viax928k94	test	SurveillanceSimple (848fuqhj1q)	DEPLOYED	

© 2022 rainbow | About | Terms of Service | Privacy



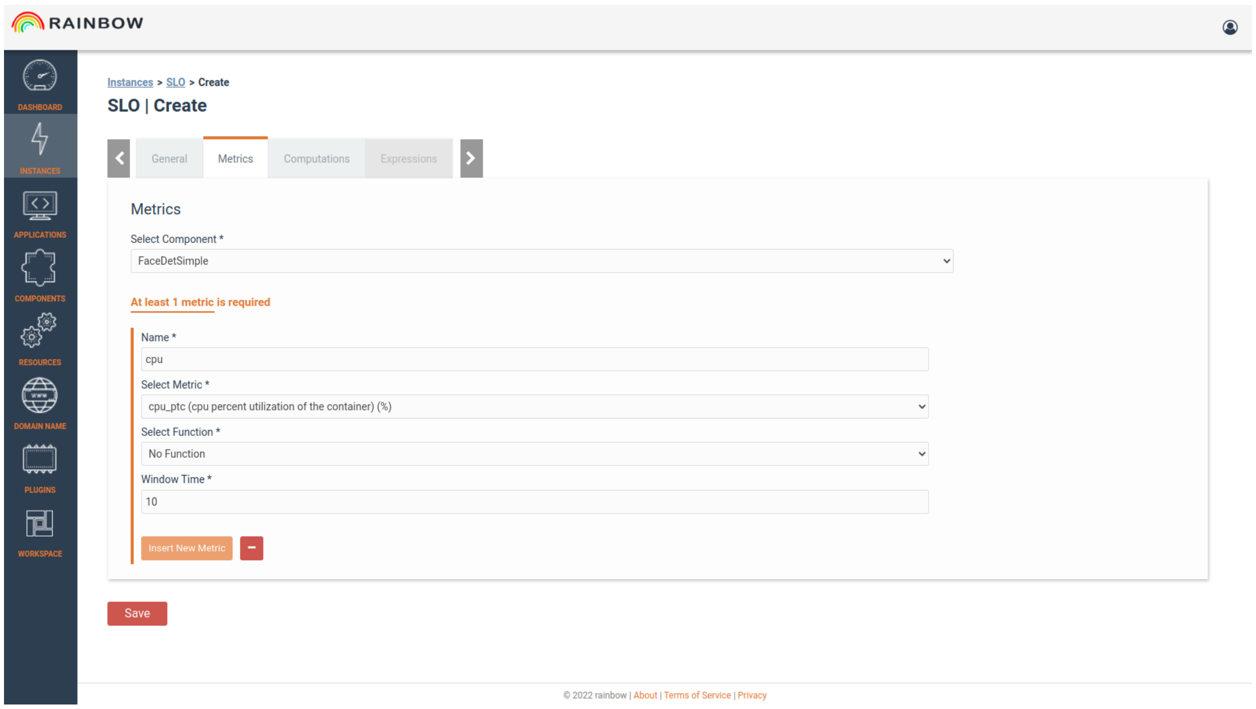


Name *

myslo

Select Elasticity Strategy *

HorizontalElasticityStrategy



SLO | Create

Metrics

Select Component * FaceDetSimple

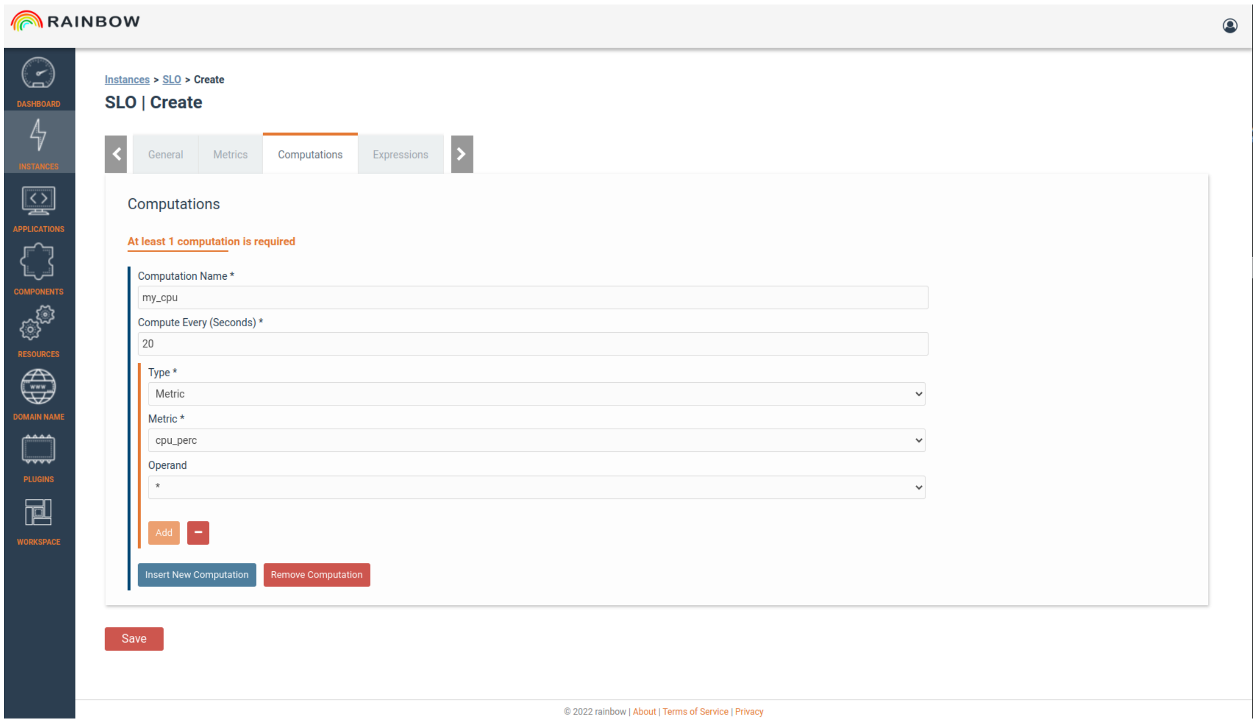
At least 1 metric is required

Name * cpu

Select Metric * cpu_ptc (cpu percent utilization of the container) (%)

Select Function * No Function

Window Time * 10



SLO | Create

Computations

At least 1 computation is required

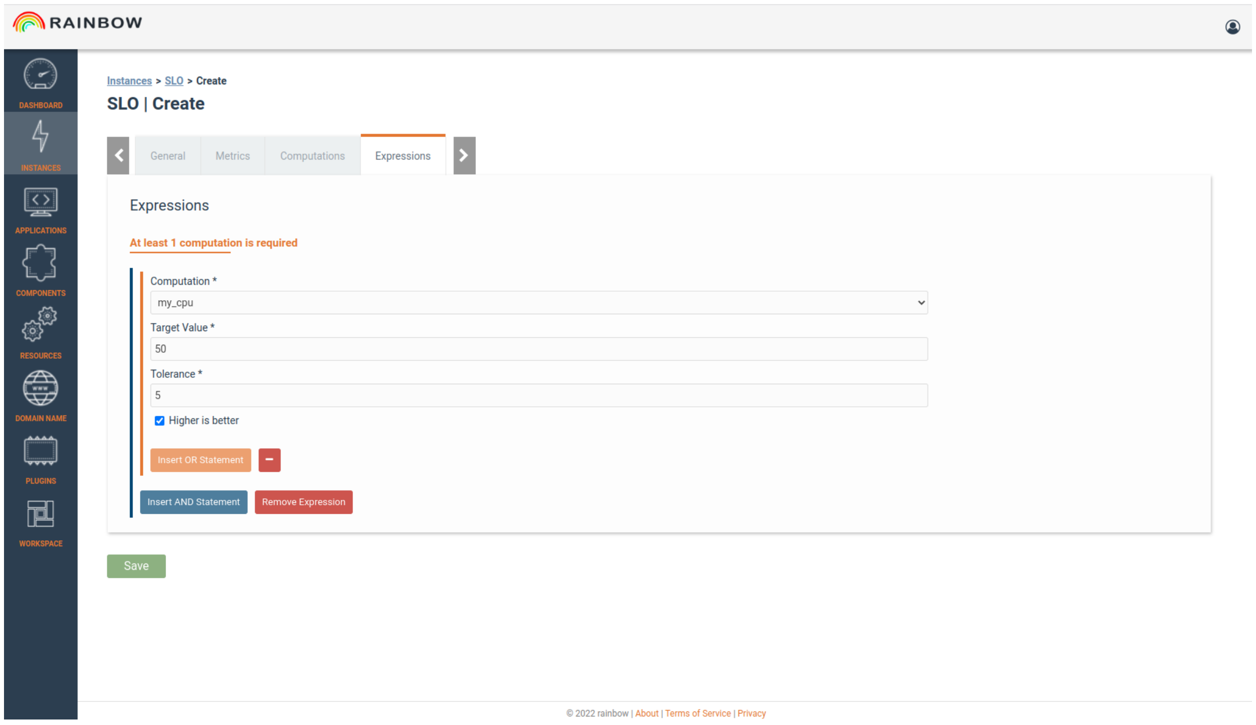
Computation Name * my_cpu

Compute Every (Seconds) * 20

Type * Metric

Metric * cpu_perc

Operand *



Instances > SLO > Create

SLO | Create

Expressions

At least 1 computation is required

Computation *: my_cpu

Target Value *: 50

Tolerance *: 5

[✓] Higher is better

© 2022 rainbow | About | Terms of Service | Privacy

5.9.3 Analytics

5.10 Cloud Resources

5.10.1 Manage Cloud Resources

- Rainbow user can add cloud resources in compatible cloud providers in order to allow the deployment of an application
- OpenStack, Amazon AWS and Google Cloud are supported
- Appropriate forms for of the each cloud providers are available;
- OpenStack
- Amazon AWS
- Google Cloud

5.10.2 Manage Your Keys

You can add keys for your account with the appropriate form.