Cloud Test Suite - CERN IT Documentation

CERN IT-DI-EFP

Dec 30, 2021

Cloud Test Suite Documentation

This tool is intended to be used to test and validate commercial cloud services across the stack for research and education environments and it is being used as a validation tool for commercial cloud services procurement in European Commission sponsored projects such as , and . For more information about the European Open Science Cloud (EOSC) visits its website .

Please find the repository hosting the source code of this tool.

CHAPTER 1

1. Getting started

This section guides the user on the whole process of preparation for running the suite, from installation of dependencies to configuration of the tool.

To ease this process, we do recommend using a Docker image that we provide which already contains the dependencies below described (Terraform, Ansible, kubectl, etc). Refer to section *Using Docker* to use the Docker image we provide to avoid dealing with required packages and dependencies.

1.1 1.1 Dependencies

This test-suite requires some packages, please see below.

1.1.1 Terraform

Terraform is the tool that creates the VMs that will later conform a Kubernetes cluster. Download and install on your machine.

1.1.2 Ansible

Ansible is used to configure the VMs and install packages on them in order to bootstrap the cluster. Follow the steps to install it.

1.1.3 Kubernetes client

In order to manage the Kubernetes cluster remotely, install on your machine.

1.1.4 Python

Python version 3 is required. The following python packages are required, install them with pip3:

- pyyaml
- jsonschema
- kubernetes
- requests

1.1.5 jq

Linux' jq is a flexible and lightweight command-line processor for JSON. In the test suite it is used to work with JSON data produced by commands such as kubectl.

1.2 1.2 SSH key

A SSH key pair is needed to establish connections to the VMs to be created later. The paths to both the private and public keys are required in the configuration file. Note errors may occur if your key does not have the right permissions. Set these correctly using the following command:

```
$ chmod 600 path/to/key
```

1.3 1.3 Security groups

By default the test suite takes care of security groups and other firewall measurements. Ultimately, the traffic that should be allowed is:

Port	Protocol	Functionality
	ICMP	Connectivity test
22	TCP	SSH
6443	TCP	Kubernetes API
10250	TCP	API which allows node access
8472	UDP	Flannel overlay network, k8s pods communication

1.4 1.4 Permissions

During its run, the suite creates files and folders inside the directory *EOSC-Testsuite*: Kubernetes resource definition files, log and results files and folders, etc. Therefore, the user running the suite must have enough permissions to accomplish such tasks.

1.5 1.5 Clone and prepare

1.5.1 Cloning repository

Clone the repository as follows and cd into it. Note this step is not needed when using the provided Docker image, the repository is already cloned there.

```
$ git clone https://github.com/cern-it-efp/EOSC-Testsuite.git
$ cd EOSC-Testsuite
```

1.5.2 Configuration

Two YAML files have to be filled to configure the run. Examples of these two files can be found inside the *examples* folder of the repository.

`testsCatalog.yaml`

This file gathers details related to the tests that should be deployed. Refer to the section *Test Catalog* to learn how to fill this file.

`configs.yaml`

This file gathers general details required to provision the infrastructure. The file also contains a section named *costCalculation*. Refer to the section *Cost of run calculation* to learn more about the estimated cost calculation feature and to understand how to fill that part.

General variables:

Name	Explanation / Values
providerName	Terraform name of the provider. (required)
pathToKey	Path to the location of your private key, to be used for ssh connections. (required)
flavor	Flavor to be used for the main cluster. (required)
openUser	User to be used for ssh connections.
storageCapacity	Storage size to have on the VM.

Provider/cloud specific variables:

1.5.2.1 AWS

AWS specific variables for configs.yaml:

Name	Explanation / Values
region	The region in which to create the compute instances. (required)
availabilityZone	The availability zone within the region where the compute instances should be created. (required)
sharedCredentialsFile	The authentication method supported is AWS shared credential file. Specify here the absolute path to such file. (required)
ami	AMI for the instances. (required)
keyName	Name of the key for the instances. (required)

It is also possible to use EKS to provision the cluster, for this refer to section *Using existing clusters*.

1.5.2.2 Azure

Install az CLI and configure credentials with 'az login'. Note resource group, security group, and subnet have to be created in advance.

Azure specific variables for configs.yaml:

Name	Explanation / Values
location	The region in which to create the compute instances. (required)
subscriptionId	ID of the subscription. (required)
resourceGroupName	Specifies the name of the Resource Group in which the Virtual Machine should exist. (required)
pubSSH	Public SSH key of the key specified at configs.yaml's pathToKey. (required)
securityGroupID	The ID of the Network Security Group to associate with the VMs's network interfaces (required)
subnetId	Reference to a subnet in which the NIC for the VM has been created. (required)
image.publisher	Specifies the publisher of the image used to create the virtual machines.
image.offer	Specifies the offer of the image used to create the virtual machines.
image.sku	Specifies the SKU of the image used to create the virtual machines.
image.version	Specifies the version of the image used to create the virtual machines.

The image section is optional but in case it is provided, all its 4 variables must be set. Omitting the image section defaults to:

- publisher = OpenLogic
- offer = CentOS
- sku = 7.5
- version = latest

It is also possible to use AKS to provision the cluster, for this refer to section *Using existing clusters*.

1.5.2.3 CloudSigma

CloudSigma specific variables for configs.yaml:

Name	Explanation / Values
storageCapacity	VM's disk size. The VMs will boot from disk, this sets the size of it. Measured in bytes. (required)
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
authFile	Path to a YAML file containing the credentials. See the details below this table. (required)
openUser	Must be set to *cloudsigma*. (required)
location	Location where the deployments should be performed. (required)
clone_drive_id	UUID of a library drive to clone. Can be taken from the URL when browsing drives at https://gva.cloudsigma.com/ui/4.0/library. (required)
flavor.memory	VM's RAM measured in bytes (max is 137438953472). (required)
flavor.cpu	VM's CPU Clock speed measured in MHz (max is 124000). (required)
staticIPs	Array of Static IPs to use, instead of automatically obtained ones.

Credentials file's example:

```
username: someone@email.com # UI email
password: abcd123! # UI password
```

1.5.2.4 Exoscale

Exoscale specific variables for configs.yaml:

Name	Explanation / Values
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
zone	The zone in which to create the compute instances. (required)
template	OS Image to be used for the VMs. (required)
storageCapacity	VM's disk size. (required)
authFile	Path to the file containing the Exoscale credentials. See below the structure of such file. (required)
securityGroups	Security groups array.

Exoscale credentials file's structure:

```
[cloudstack]
key = EXOe3ca3e7621b7cd7a20f7e0de
secret = 2_JvzFcZQL_Rg1nZSRNVheYQh9oYlL5aX3zX-eILiL4
```

1.5.2.5 GCP

GCP specific variables for configs.yaml:

Name	Explanation / Values
zone	The zone in which to create the compute instances. (required)
pathToCredentials	Path to the GCP JSON credentials file (note this file has to be downloaded in advance from the GCP console). (required)
image	Image for the instances. (required)
project	Google project under which the infrastructure has to be provisioned. (required)
gpuType	Type of GPU to be used. Needed if the Deep Learning test was selected at testsCatalog.yaml.

It is also possible to use GKE to provision the cluster, for this refer to section *Using existing clusters*. You will have to too.

1.5.2.6 IBM Cloud

IBM specific variables for configs.yaml:

Name	Explanation / Values
authFile	Path to the yaml file containing the credentials. See below the structure of such file. (required)
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
os_reference_code	OS image for the VMs. (required)
datacenter	Location where the deployments should be performed. (required)
flavor	VM flavor. (required)
network_speed	The connection speed (in Mbps) for the instance's network components. (required)

Credentials file's example:

```
iaas_classic_username: someone@email.com #
iaas_classic_api_key: abcd123! # Classic Infrastructure API Keys
```

1.5.2.7 Ionos Cloud

Ionos specific variables for configs.yaml:

Name	Explanation / Values
storageCapacity	VM's disk size. The VMs will boot from disk, this sets the size of it. (required)
authFile	Path to the yaml file containing the credentials. See below the structure of such file. (required)
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
image_name	OS image for the VMs. (required)
location	Location where the deployments should be performed. (required)
flavor.cores	VM number of cores. (required)
flavor.ram	The amount of memory for the server in MB. (required)

Credentials file's example:

```
username: someone@email.com # UI email
password: abcd123! # UI password
```

1.5.2.8 Layershift (Jelastic PaaS)

Layershift specific variables for configs.yaml:

Name	Explanation / Values
nodes	Size of the cluster to create, including the master. (required)
cores	Number of cores the cluster nodes should have. (required)
tokenPath	Path to the file containing the token. See below the structure of such file. (required)
podsOnMaster	If true, the taint on the master node will be removed so pods can run there. (required)

Credentials file's structure:

```
token: e3ca3e7620f7e0de1b7cd7a2
```

Create and configure cluster:

```
$ python3 main.py -c ../../../../examples/layershift/configs.yaml
```

On success, the cluster is ready and the test suite can be run, using the same configs file and the option *–onlyTest*.

Destroy cluster:

```
$ python3 main.py -d -c ../../../../examples/layershift/configs.yaml
```

1.5.2.9 Oracle Cloud Infrastructure

OCI specific variables for configs.yaml:

Name	Explanation / Values
ssh_public_key_path	Path to the public key belonging to your private key at pathToKey. This will be injected to the VMs (required)
authFile	Path to the yaml file containing the OTC credentials. See below the structure of such file. (required)
image_ocid	The OCID of the image to be used on the VMs. The OCID list can be consulted here: https://docs.oracle.com/en-us/iaas/images/centos-7x/ (required)
compartment_ocid	Compartment's OCID. (required)
availability_domain	Availability domain to be used. (required)
subnet_ocid	The OCID of the subnet to be used. (required)
storageCapacity	VM's disk size.
useFlexShape	Indicates if the selected flavor is a flexible shape (such as VM.Optimized3.Flex). If this is set to false and the used flavor is a flexible one, the instances will be created with the default configuration values for the specified shape.

When using a flexible shape, specify also the following arguments (note these would be only considered if useFlexShape is set to true):

Name	Explanation / Values
ocpus	Specify the cores for the VMs.
memoryInGbs	Specify the memory in GB for the VMs.

Oracle Cloud Infrastructure credentials file's must be a YAML file containing only the following variables:

Name	Explanation / Values
privateKeyPath	Path to the private key to be used to authenticate to OCI. This is not the key to be used to ssh into the machines.
userOcid	User's OCID.
tenancyOcid	Tenancy's OCID.
fingerprint	Authentication key's fingerprint.
region	Region to be used.

It is also possible to use OKE to provision the cluster, for this refer to section *Using existing clusters*.

1.5.2.10 OpenStack

Regarding authentication, download the OpenStack RC File containing the credentials from the Horizon dashboard and source it.

OpenStack specific variables for configs.yaml:

Name	Explanation / Values
storageCapacity	VM's disk size. The VMs will boot from disk, this sets the size of it. (required)
imageID	OS Image ID to be used for the VMs. (required)
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
securityGroups	Security groups array.
region	The region in which to create the compute instances. If omitted, the region specified in the credentials file is used.
availabilityZone	The availability zone in which to create the compute instances.
networkName	Name of the newtork to be used.

1.5.2.11 T-Systems' Open Telekom Cloud

To allow the VMs access the internet, Shared SNAT has to be enabled on the default VPC, which will be used for the suite run.

OTC specific variables for configs.yaml:

Name	Explanation / Values
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
storageCapacity	VM's disk size. The VMs will boot from disk, this sets the size of it. (required)
authFile	Path to the yaml file containing the OTC credentials. See below the structure of such file. (required)
imageID	ID of the image to be used on the VMs. (required)
domainName	OTC Domain Name. (required)
tenantName	OTC Tenant Name. (required)
securityGroups	Security groups array.

Open Telekom Cloud credentials file's structure:

```
accK: 123456789abcd # access key
secK: 123456789abcd # security key
```

1.5.2.12 Orange's Flexible Engine

Orange's Flexible Engine specific variables for configs.yaml:

Name	Explanation / Values
storageCapacity	VM's disk size. The VMs will boot from disk, this sets the size of it. (required)
authFile	Path to the yaml file containing the credentials. See below the structure of such file. (required)
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
imageID	ID of the image to be used on the VMs. (required)
openUser	User for initial SSH connections. (required)
region	Region where the deployments should be performed. (required)
availabilityZone	Availability Zone. (required)
flavor	VM flavor. (required)
bandwidth	VM's public bandwidth in Mbps, maximum 1000. (required)
staticIPs	Array of Elastic IPs to use, instead of automatically allocated ones. If used, the number of nodes to be provisioned is determined by the size of this array.

Credentials file's structure:

```
accessKey: 123456789abcd # access key
secretKey: 123456789abcd # security key
```

1.5.2.13 Yandex Cloud

Yandex Cloud specific variables for configs.yaml:

Name	Explanation / Values
storageCapacity	VM's disk size. The VMs will boot from disk, this sets the size of it. Measured in GB. (required)
pathToPubKey	Path to your public key, which is injected into the VMs. (required)
serviceAccountKeyFile	Path to a JSON file containing the credentials. (required)
openUser	Default user for SSH connections. (required)
cloudID.	Cloud ID. (required)
folderID	Folder ID (required)
platformID	Platform ID. (required)
zone	Zone where the deployments should be performed. (required)
imageID	OS image ID. (required)
flavor.cores	VM number of cores. (required)
flavor.memory	The amount of memory for the server in GB. (required)
staticIPs	Array of Static IPs to use, instead of automatically obtained ones.

To run the suite on a provider/cloud that is not listed above (supporting Terraform or not), refer to the section *No Terraform runs*.

1.6 1.6 Using Docker

A Docker image has been built and pushed to Docker hub. This image allows you to skip section "Dependencies" and jump to "SSH key", you can see the Dockerfile here.

Run the container (pulls the image first):

```
$ docker run -it cernefp/tslauncher
```

You will get a terminal on the container, directly inside the cloned repository.

CHAPTER 2

2. Tests Catalog

This section describes the tests that the test suite contains. In the root of the cloned repository, you will find a file named *testsCatalog.yaml* in which you have to specify the tests you want to run. The test suite relies on this YAML file to deploy the tests on the clusters. To run a certain test set its *run* variable to *True*. Please find below, a description of each test as well as additional parameters for the configuration of each.

2.1 Distributed Training of a GAN using GPUs

The 3DGAN application is a prototype developed to investigate the possibility to use a Deep Learning approach to speed-up the simulation of particle physics detectors. The benchmark measures the total time needed to train a 3D convolutional Generative Adversarial Network (GAN) using a data-parallel approach on distributed systems. It is based on MPI for communication. As such, it tests the performance of single nodes (GPUs cards) but also latency and bandwidth of nodes interconnects and data access. The training uses a Kubernetes cluster (GPU flavored) with Kubeflow and MPI.

If selected, the suite will provision a Kubernetes cluster -GPU flavored- specifically for this test. The test suite **assumes NVIDIA drivers are installed**. Therefore, this test can only run using an OS image that includes it. For this test, apart from the *run* variable, the following can be set in the *testsCatalog.yaml* file:

Name	Explanation / Values
nodes	Number of nodes to be used for the deployment. Default: max number of nodes available.
flavor	Name of the flavor to be used for this test's cluster. (required)

- Contributors/Owners: Sofia Vallecorsa (CERN) - sofia.vallecorsa AT cern.ch; Jean-Roch Vlimant (Caltech)
- /

2.2 progressive Growing of GANs using GPUs

Single-node training, based on . The is a generic dataset, used only as an example for this use case.

If selected, the suite will provision a single-node Kubernetes cluster -GPU flavored- specifically for this test. The test suite **assumes NVIDIA drivers are installed**. Therefore, this test can only run using an OS image that includes it. For this test, apart from the *run* variable, the following can be set in the *testsCatalog.yaml* file:

Name	Explanation / Values
flavor	Name of the flavor to be used for this test's cluster. (required)
images_amount (1, 980)	Number of images to use from the data set. Minimum 1, maximum 980. (required)
kimg	Number of images provided to the network for its training. Note 1 kimg = 1000 images. Minimum 1, maximum 12000. (required)
gpus	Number of GPUs to use. Accepted values are 1, 2, 4 and 8. If this parameter is not used, all the available GPUs on the VM will be used.

- Contributors/Owners: Sofia Vallecorsa (CERN) - sofia.vallecorsa AT cern.ch
-

2.3 S3 endpoint tests

A simple S3 test script to test functionality of S3-like endpoints, checking the following: S3 authentication (access key + secret key), PUT, GET, GET with prefix matching, GET chunk and GET multiple chunks.

For this test, apart from the *run* variable, the following ones must be set on the *testsCatalog.yaml* file:

Name	Explanation / Values
endpoint	Endpoint under which your S3 bucket is reachable. This URL must not include the bucket name but only the host.
accessKey	Access key for S3 resource management.
secretKey	Secret key for S3 resource management.

Note that the provider has to allow using S3 clients such as s3cmd or aws-cli. For example, specifically for GCP, interoperability has to be enabled.

- Contributors/Owners: Oliver Keeble (CERN) - oliver.keeble AT cern.ch
-

2.4 Data Export: From the commercial cloud provider to Zenodo

When using cloud credits, when the credit is exhausted, data can be repatriated or moved to a long-term data storage service. The example used in this test uses service maintained by CERN, verifying that the output data can be taken from the cloud provider to Zenodo.

- Contributors/Owners: Ignacio Peluaga (CERN) - ignacio.peluaga.lozada AT cern.ch
-

2.5 CPU Benchmarking

Benchmarking relying on a suite containing several High Energy Physics (HEP) based benchmarks. For this test, the VM should have at least 4 cores. Please refer to the repository below for more details and information.

- Contributors/Owners: Domenico Giordano (CERN) - domenico.giordano AT cern.ch
-

2.6 Networking performance measurements

perfSONAR is a network measurement toolkit designed to provide federated coverage of paths, and help to establish end-to-end usage expectations.

In this test, a perfSONAR testpoint is created using a containerised approach on the cloud provider infrastructure. The following tests are run between the provisioned testpoint and another perfSONAR server that the user specifies in the test's configuration (see below):

- throughput: A test to measure the observed speed of a data transfer and associated statistics between two endpoints.
- rtt: Measure the round trip time and related statistics between hosts.
- trace: Trace the path between IP hosts.
- latency: Measure one-way latency and associated statistics between hosts. Note this test does not work if the node is behind NAT.

The endpoint for these tests must be specified at testsCatalog.yaml's *perfsonarTest.endpoint* variable. Note if the server on the provided endpoint does not allow or support any of these tests, those will fail but the others would still be carried out. Use endpoints from:

-
-
-
- Contributors/Owners: Shawn Mckee (University of Michigan) - smckee AT umich.edu; Marian Babik CERN) - marian.babik AT cern.ch
-

2.7 DODAS: Dynamic On Demand Analysis Services test

DODAS is a system designed to provide a high level of automation in terms of provisioning, creating, managing and accessing a pool of heterogeneous computing and storage resources, by generating clusters on demand for the execution of HTCondor workload management system. DODAS allows to seamlessly join the HTCondor Global Pool of CMS to enable the dynamic extension of existing computing resources. A benefit of such an architecture is that it provides high scaling capabilities and self-healing support that results in a drastic reduction of time and cost, through setup and operational efficiency increases.

If one wants to deploy this test, the machines in the general cluster (to which such test is deployed), should have rather large disks as the image for this test is 16GB. To set the disk size use the *storageCapacity* variable from configs.yaml.

- Contributors/Owners: Daniele Spiga (INFN) - daniele.spiga@pg.infn.it ; Diego Ciangottini (INFN) - diego.ciangottini@cern.ch
-

CHAPTER 3

3. Run

Once the configuration steps are completed, the Test-Suite is ready to be run:

```
EOSC-Testsuite$ chmod +x test_suite
EOSC-Testsuite$ ./test_suite <options>
```

The suite uses a combination of Linux' watch and cat commands to display the logs of the run. Hence, the suite will fail if no terminal/TTY is available. To change this behaviour use the option *noWatch* as defined in the options below.

Once the provisionment steps are completed (Kubernetes cluster up and running) and pods are deployed, the run will finish once all the deployed tests complete. If for any reason you want to stop the run before completion, delete all the pods and this will finish the run.

3.1 Options

-c, --configs	Specifies a custom location of the general configurations YAML file. If omitted, EOSC-Testsuite/configs.yaml will be used.
-t, --testsCatalog	Specifies a custom location of the tests catalog YAML file. If omitted, EOSC-Testsuite/testsCatalog.yaml will be used.
--noTerraform	Option to skip terraform provisionment, only ansible-playbook bootstrapping. To be used on providers that do not support terraform.
-o, --onlyTest	Run without creating the infrastructure (VMs and cluster), only deploy tests. Not valid for the first run.
--destroy <cluster>	No test suite run, only destroy provisioned infrastructure. Argument can be: (note a quote wrapped and space separated subset of these can also be specified, for example "dlTest shared") 'shared': Destroy the shared cluster. 'dlTest': Destroy the GPU cluster. 'hpcTest': Destroy the HPC cluster.

'all': Destroy all clusters.

--destroyOnCompletion <clusters> Destroy infrastructure once the test suite completes its run. Same arguments as for '–destroy' apply.

--customNodes <value> Set the number of instances that should be deployed for the shared cluster. If omitted, the suite will provision as many nodes as tests of the general ones (s3Test, dataRepatriationTest, cpuBenchmarking, perfsonarTest and dodasTest) were selected.

--usePrivateIPs By default the test suite signs the cluster's certificate for the master's NAT IP address. This is required in order to be able to communicate with the cluster through the internet because in most of the cases, providers do not allocate public IPs to the VMs (no network interfaces with public IP) but use NAT. If this option is used, the test suite will not sign the certificate for the master's NAT IP address.This means, the cluster can be reached only from the network it is connected to. For this scenario, a bastion VM should be used.

When using the docker approach, *–net=host* should be used in the Docker run command. With that option, the container will use the network used by its host. Without it, the container wouldn't be able to communicate with the nodes, as it would not be in the same network as them and the nodes will not have public IPs.

--noWatch Makes the test suite not use the watch function, hence disabling logs.

--freeMaster Disables running pods (tests and/or benchmarks) on the master node. For the shared and proGAN's clusters, an extra node would be added. For example, if 3 tests were selected, 4 nodes would be created. For the other clusters, the number of nodes created would still be the one the user specified on the tests catalog YAML file.

--publish Upload results to CERN's S3. More information can be found here.

3.2 Other commands

This sections provides a set of commands that can be used to obtain logs and in general, check the status of the run. Note all these assume the location is EOSC-Testsuite, this is, inside the cloned repository.

Once the test suite is running, you can view the Terraform provisionment logs by doing:

```
$ tail -f EOSC-Testsuite/logs
```

You can see the Ansible bootstrapping logs by doing:

```
$ tail -f EOSC-Testsuite/src/logging/ansibleLogs*
```

Once the bootstrapping has completed and tests are deployed, you can see the pods statuses by doing:

```
$ watch kubectl get pods --kubeconfig EOSC-Testsuite/src/tests/shared/config
```

For tests other than those that are deployed in the general cluster, see their pods by doing:

```
$ watch kubectl --kubeconfig EOSC-Testsuite/src/tests/dlTest/config get pods # For␣
↪dlTest cluster
$ watch kubectl --kubeconfig EOSC-Testsuite/src/tests/dlTest/config get pods # For␣
↪proGANTest cluster
$ watch kubectl --kubeconfig EOSC-Testsuite/src/tests/hpcTest/config get pods # For␣
↪hpcTest cluster
```

(continues on next page)

(continued from previous page)

Once the pods are deployed, the suite run can be stopped by destroying pods. This is useful for example when pods go "Evicted" or "ImagePullBackOff". Examples:

```
$ kubectl --kubeconfig EOSC-Testsuite/src/tests/shared/config delete pods --all #␣
↪destroy all pods on the shared cluster
$ kubectl --kubeconfig EOSC-Testsuite/src/tests/shared/config delete pod dodas-pod #␣
↪destroy DODAS pod
```

The following aliases are available when using the provided Docker image:

Alias	Equivalence
tfLogs	'tail -f /EOSC-Testsuite/logs'
ansibleLogs	'tail -f /EOSC-Testsuite/src/logging/ansibleLogs*'
watchPods	'watch kubectl get pods –kubeconfig /EOSC-Testsuite/src/tests/shared/config -owide'

CHAPTER 4

4. Using existing clusters

It's possible to use this tool for testing providers that support Kubernetes as a Service. This means the provider offers the user a way for simply creating a cluster. In case one wants to validate a provider that offers this and want to take advantage of it, skip the Terraform steps and when running the test-suite, use option *–onlyTest*.

Note that the kubeconfig files have to be placed in the following locations, according to whether the test was selected or not:

Test / Cluster	Path to kubeconfig
Shared cluster	EOSC-Testsuite/src/tests/shared/config
dlTest cluster	EOSC-Testsuite/src/tests/dlTest/config
hpcTest cluster	EOSC-Testsuite/src/tests/hpcTest/config
proGANTest cluster	EOSC-Testsuite/src/tests/proGANTest/config

CHAPTER 5

5. No Terraform runs

It's possible to use this tool for testing providers that do not support Terraform. In these cases, the suite will skip the Terraform provision phase and would just bootstrap the cluster and deploy the tests. Therefore the VMs have to be provisioned beforehand and their IPs written down at configs.yaml.

The configs.yaml file for these type of run takes the following variables:

Name	Explanation / Values
providerName	Name of the provider being tested. (required)
pathToKey	Path to the location of your private key, to be used for ssh connections. (required)
openUser	User to be used for ssh connections. (required)
clusters.shared	IPs of the shared cluster's VMs. (Required if at least one of the general tests was selected)
clusters.hpcTest	IPs of the hpcTest cluster's VMs. (Required if hpcTest was selected)
clusters.dlTest	IPs of the dlTest cluster's VMs. (Required if dlTest was selected)
clusters.proGANTest	IPs of the proGANTest cluster's VMs. (Required if proGANTest was selected)

For making use of this feature use the option '–noTerraform' when running the suite.

CHAPTER 6

6. Results

Once all the selected tests finish the run, the test-suite has completed its execution. The results and logs of the validation exercise can be seen at /results (JSON format). Prior to completing the runs, a message will be printed to the console showing the exact path to the results. There you will find a file *general.json* containing general details, estimated cost and brief test results information and also a directory *detailed* containing more detailed information for each test.

6.1 Upload results to CERN's S3

Prior to the run, define the environment variables S3_ACC_KEY and S3_SEC_KEY and later use the option –publish. The results of the run should be on the bucket after the run completes.

CHAPTER 7

7. Cost of run calculation

An approximative cost of running the test-suite will be calculated in case the prices are specified at configs.yaml under the *costCalculation* section. In this configuration file, one must specify the price per hour for the different resources:

Name	Explanation / Values
generalInstancePrice	Price per hour of VM with the flavor chosen for the general cluster.
GPUInstancePrice	Price per hour of VM with the flavor chosen for the GPU cluster.
HPCInstancePrice	Price per hour of VM with the flavor chosen for the HPC cluster.
s3bucketPrice	S3 bucket price.

If a price value is required for the cost calculation but the *costCalculation* section is not properly filled (For example, S3 Endpoint test was set to True but *s3bucketPrice* was not set), no approximation will be given at all.

At the end of the run, the resulting approximated cost will be added to the file containing general test suit run results. In case this information isn't needed, simply leave the values on the section *costCalculation* empty. Note that this is a cost estimate and not an exact price.

The formula used is as follows:

(Number of VMs created) x (Price of a VM per hour) x (Time in hours the VMs were used for the test-suite run) + (Cost of other resources)

Where "Cost of other resources" are the cost of resources which are not simple compute, like storage. For example in the case of the S3 Endpoint test:

(Price of a S3 bucket per hour) x (Time in hours the bucket was used for the test)

Note that the price per request or data amount (GB) are not considered here as these are not significant since less than 10 requests are done for this test and for very small data sets. Note also that only the cost of the running time of the VM is considered, so if your provider charges for VM creation and not only for the time it is running, the cost obtained will vary to the real one.

CHAPTER 8

Projects

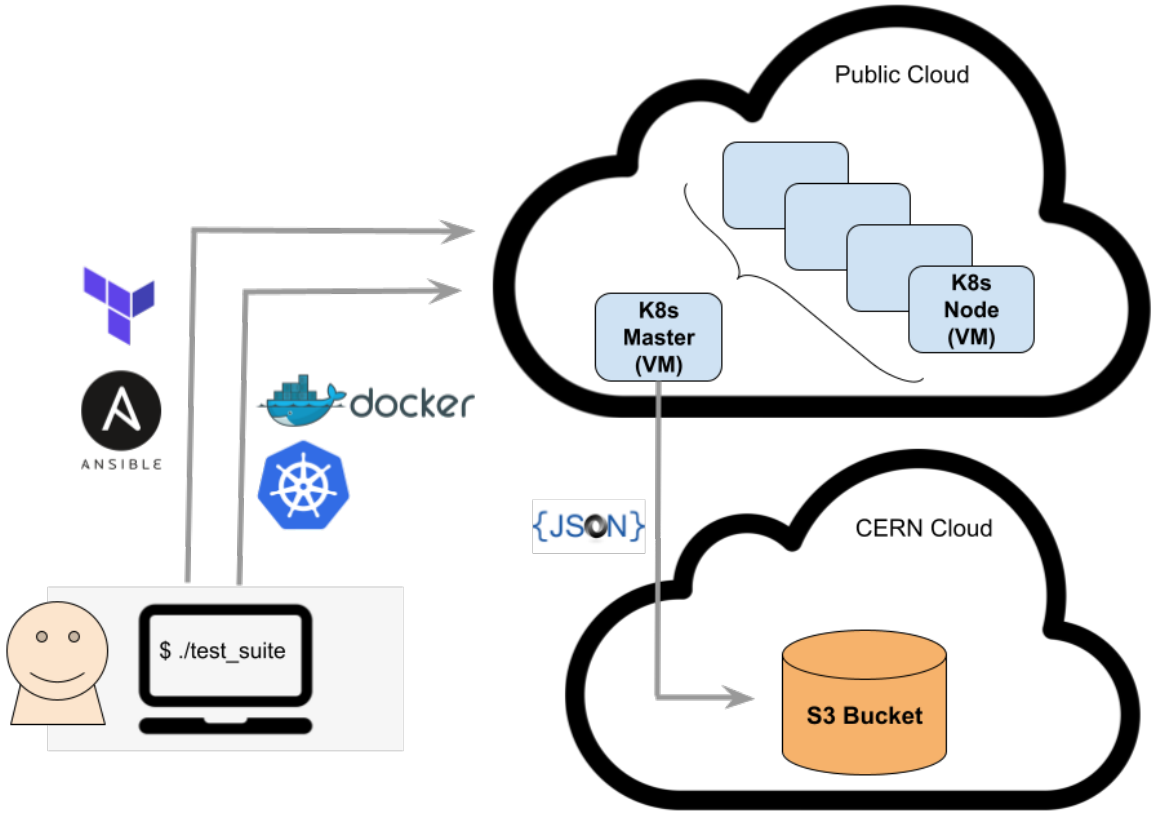
This page contains project specific information concerning the test suite.

8.1 Open Clouds for Research Environments (OCRE)

8.1.1 Motivation

The test-suite is intended to be used to test and validate cloud services across the stack for research and education environments. The development of the test suite originates from the HNSciCloud project, where automation was lacking as tests deployments were executed manually, in a scattered manner with no result tracking capabilities. This Test-Suite is being used as a validation tool for cloud services procurement in European Commission sponsored projects such as OCRE and ARCHIVER. The testing and validation in the scope of OCRE will be used as part of the selection criteria for the adoption funds available in the project, resulting in a OCRE certification process for the 27 platforms involved in the project.

Below, an overview of the tool and overall test validation in the context of the OCRE project.



8.1.2 Scope of the Testing and Validation

The test suite is capable of evaluating different services across the cloud stacks:

- **Networking** - Network performance tests using .
- **Standard CPU benchmarks** - Runs of a set of to benchmark CPU performance.
- **Basic S3 storage features validation** - Check the availability of the main S3 object store functionalities .
- **Basic Data Repatriation** - Tests data backup exports from the cloud provider to the repository service.
- **Machine Learning** - Set of benchmarks for single and multiple node algorithm training of advanced GAN models (, and), on accelerator architectures (i.e. GPUs).
- **DODAS check** - is a scientific software framework running a test workload from one of the LHC experiments, . Success running this test indicates the DODAS framework can generate clusters on demand using public cloud, for batch workload execution based on the workload management system.

The provides additional details about these tests.

The testing and validation in the scope of OCRE will be used as part of the selection criteria for the adoption funds available in the project.

8.1.3 Results

Results of the runs will be stored as JSON files on an S3 bucket on the CERN cloud's CEPH service. OCRE consortium members will be able to access all results, whilst vendors only their own results.

To provide segregated access, pre-signed S3 URLs will be used. Each vendor will be provided a list of pre-signed URLs that should be used to obtain the result files. Automation of the download of those result files is possible. Please use to do it.

The CERN development team is developing a dashboard that will parse the data from the JSON files, for a more user-friendly visualisation.

No tests results will be made public without agreement of the respective vendor.

The OCRE consortium aims to create a certification process for the platforms that are successfully validated. This process will be agreed and communicated to the respective vendors before being put in place.

In addition to the GÉANT contract management team, two CERN members shall be involved in the interactions:

- A CERN representative will handle all communication between the vendors technical representatives and the CERN developers.
- A CERN technical representative will be responsible for deploying and performing validation tests. In addition, he must have permissions to create additional local user accounts in the award subscription, in case other members need access to run or complete those tests.

8.1.4 Requirements

In order to perform multiple runs of the test set including the Machine Learning benchmarks, an modest subscription credit (**5000€**, valid for 6 months) is required for the full stack of services available in the platforms.

In case a platform does not offer accelerator architectures (i.e. GPUs, FPGAs or similar vendor specific) the required amount of credits can be lower.

The number/type of tests is expected to evolve during 2021 in areas such as HPCaaS and . Any new additional test will be documented and this page updated. It shall not imply requests of additional credit.

8.1.5 Timeline

Access to the platforms should be provided to the testing team by the latest in mid April 2021. Results will start to be available between May and June 2021.

8.1.6 Main Technical Contacts

To handle communication effectively, please use the mailing list: cloud-test-suite AT cern.ch

The contact details of the CERN technical representative that will be responsible for deploying and performing validation tests will be provided later.

8.1.7 Licensing

The framework is licensed under . Tests included might have their specific licenses. For more details, please refer to the .

8.1.8 Resources

-
-

The test-suite executes four main steps:

1) Infrastructure provisioning: VMs are created using Terraform and then with these several Kubernetes clusters are bootstrapped by Ansible according to the selected tests.
2) Deploy the tests: Kubernetes resource definition files (YAML) are used to deploy the tests.
3) Harvest results: at the end of each test run a result file -written in JSON- is created. This file is collected from the pod running on the cluster.
4) Optionally, destroy resources.

The test set described in the Tests Catalog section of the documentation is originally based on the tests used in the PCP project funded by the European Commission.

The developers would like to thank all test owners and contributors to this project.

The latest version of the suite has been tested on:

OS running on provider's VMs	CentOS7, Ubuntu 18.04, Ubuntu 20.04
Providers / clouds	AWS Google Cloud Microsoft Azure Oracle Cloud Infrastructure Exoscale (CloudStack) T-Systems' Open Telekom Cloud (OpenStack) CERN Private Cloud (OpenStack) CloudFerro (OpenStack) CloudStack OpenStack Ionos CloudSigma OVH IBM CityNetwork X-Ion Layershift Orange's Flexible Engine Yandex Cloud

The suite is continuously tested in new cloud providers. As tests are concluded, the cloud providers names will be added to the table above.

CHAPTER 9

Release notes

(Note the versions are numbered with the date of the release: YEAR.MONTH)

`21.12 - latest`

- Included deployment support for Ionos, CloudSigma, OVH, IBM, CityNetwork, X-Ion, Layershift, Orange and Yandex.
- Updated CPU Benchmark: using new Hepscore based implementation.
- Enabled prompt asking for confirmation prior to starting the run in case of existing Terraform files.
- Network test (perfSONAR) update: added reverse throughput measurement & retries.
- Complete networking/VPC provisioning: AWS, Openstack, IBM, Orange, OCI, Ionos and Yandex.
- Added option –freeMaster to disable running tests/benchmarks on the master node.
- Updated results upload to CERN's private Object Storage.
- Implemented a dashboard to visually display results. While the actual deployment is not public, its source code can be found .

`21.4`

- Cluster certificate additionally signed for NAT IP (no need to use bastion method, with this solution the cluster can be reached from outside of the provider network. However, previous allocation of floating IPs is now required).
- Added –usePrivateIPs option for bastion's method.
- Removed CloudStack Terraform support (the provider's repository by HashiCorp).
- Allowed both project-wide and VM-specific ssh key on GCP.
- Improved configuration: select network.
- Updated Distributed GAN test: included NNLO implementation ; more configuration (dataset size).
- Added ProGAN test.
- Allowed subset of costs (general configuration YAML file) and tests (tests catalog YAML file).

- Allowed relative paths for -c and -t.
- Updated CPU benchmark, based on the HEP Benchmarking Suite.
- Added option –noWatch to run without displaying logs, without watch command.

20.6

- Improved support for running on Oracle Cloud Infrastructure and T-Systems' OTC.
- Added option –customNodes to set the number of instances that should be deployed for the shared cluster.
- Using Terraform's yamldecode with configs.yaml for variables instead of Python's replace function with placeholders.
- Disabled general Terraform support: only the providers and clouds that support Terraform and are present on the table above are fully supported by this suite. To run on another provider (supporting Terraform or not), the option '–noTerraform' has to be used.

20.2

- Using Ansible for VM configuration instead of Terraform's provisioners.
- Added support for non-Terraform providers (only bootstrap phase).
- Added options to destroy provisioned infrastructure.
- Added options to specify custom paths to configs.yaml and testsCatalog.yaml.
- Added support to use Ubuntu on VMs.

19.12

- Project restructured.
- Improved support for running on Google, AWS, Azure, Exoscale, OpenStack and CloudStack.

19.8

- Parallel creation of clusters, with different flavors according to tests needs.
- New logging system to keep parallel running tests logs sorted.
- Restructured configuration: moved configuration files to */configurations* and created new files taking HCL code (terraform configuration code) to keep *configs.yaml* clean.
- Automated allowance of root ssh by copying open user's authorized_keys to root's ~/.ssh as well as *sshd_config* modification.
- Usage of Kubernetes API instead of Kubernetes CLI.
- For network test (perfSONAR), usage of API instead of pscheduler CLI.
- New test: Dynamic On Demand Analysis Service, provided by INFN.
- Added configurations validation with jsonschema.
- Created Docker image to run a Test-Suite launcher container: rapidly creates a ready to use Test-Suite launcher.

19.4

- New tests: network performance (perfSONAR) and CPU benchmarking.

19.2

- First release.

CHAPTER 10

Contact

For more information contact ignacio.peluaga.lozada AT cern.ch

CHAPTER 11

License