The Bigstep Metalcloud is an IaaS service focused on offering low latency and high performance compute and storage services. The service offers the same level of flexibility that is typical from a cloud provider but using only bare metal.
CHAPTER 1

Introduction

The Bigstep Metalcloud is an IaaS service focused on offering low latency and high performance compute and storage services. The service offers the same level of flexibility that is typical from a cloud provider but using only bare metal:

- Instant Provisioning
- Stop-and-resume
- Upgrades and downgrades
- Quick hardware change
- Variable storage size

1.1 Important concepts

A typical user infrastructure architecture would look similar to this:
A user would create one or more **Instance Arrays** in an **infrastructure** and connect them with one or more **Networks**.

- An **infrastructure** groups servers, storages, IPs etc that are needed for a particular “project”.
- An **InstanceArray** is a cluster of similar servers grouped together.
- A **Network** is a dedicated **Layer 2** broadcast domain

### 1.2 Instance Arrays and Drive Arrays

Multiple **Instances** are grouped together in **Instance Arrays**. An Instance Array can be “resized” which means the same “InstanceArray-wide” settings will apply to the new instances. This is helpful for managing application clusters.

An InstanceArray can be connected to a DriveArray which in turn groups Drives. Each Instance will get an associated Drive.

The separation between storage and compute enables **Instance Arrays** to be suspended and then resumed later, hardware can be changed with a reboot etc.
1.3 Infrastructures

An infrastructure is a collection of Instances and Drives in a datacenter (availability region). From a security perspective infrastructures are equivalent to an AWS VPC.

1.4 Servers and Instances

Instances are logical-level concepts. Behind each Instance there is a Server but which particular one is of no importance to the user. The server can be swapped with another while the Instance’s properties such as access credentials, DNS records etc are preserved.

The mapping between an instance and a server is done at “provisioning” time and is created when the user “requests” a compute resource. An instance can be mapped to another server if the user needs more resources or needs a replacement.

Instances have associated “Drives” and “Networks”. Drives can be moved across storages and Networks typically span multiple switches.

Instances can be “stopped” which will release the server to the pool but will keep all other configurations such as server type, network, OS, firewall configurations etc.

The Metalcloud supports installing the OS on a local drive. When releasing the server associated with an instance the content of all local drives will be wiped.
This is how Servers This is often the very first step in using the Metal Cloud. Make sure you have created an account with us by signing up here

2.1 Creating an instance array using the Infrastructure Editor

In the MetalCloud servers (called Instances) are grouped in InstanceArrays. By default an infrastructure is created for you called “my-infrastructure” in a datacenter geographically close to you.

1. Click on the Create your first InstanceArray
2. Select your configuration, number of servers, operating system, drive size and boot type. The server will not be deployed now but rather will wait for the Deploy button to be clicked. (step #5)

Certain servers types support deploying the operating system on a local drive (or a collection of local drives in an RAID 0 array). Local drives do not allow switching the server but are less expensive and carry higher capacities and, if using local NVMe higher performance.

You will notice that a structure has been created on your UI:
it Includes both an instance array and a drive array both with a count of 1. That is normal. If you want to perform further modifications such as firewall rules or selecting another configuration by click on it.

3. Select firewall configuration

By default all traffic is blocked except if it originates from what our systems detects as being your IP. You need to explicitly enable additional IPs or ports before you deploy.
4. Deploy the infrastructure

Operations in the Metal Cloud are not immediately deployed. In fact they can be reverted until the infrastructure is “Deployed”. Click on the big “Deploy” button from the bottom of the screen.
The deploy operation should take between 3 and 10 minutes. At the end of it the instance array will be in an active state.

5. After the deploy the instance array is in “active” state. Click on the instance array:

This will pop-up the access credentials window:
Here you can find, for each instance (server):

1. the quick ssh access link

2. root password

Note: It is recommended that you register your public SSH key in the Account settings section so that it gets automatically added on the hosts at deploy time.

Where to go from here:

1. Managing infrastructures

2. Managing instance arrays
CHAPTER 3

Metalcloud instances

The Metalcloud offers various bare metal server configurations to choose from:
### Table 1: Instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.8.8</td>
<td>1 x Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz 8 GB RAM 4 x 1Gbit NIC</td>
</tr>
<tr>
<td>M.8.16</td>
<td>1 x Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz 16 GB RAM 4 x 1Gbit NIC</td>
</tr>
<tr>
<td>M.8.32</td>
<td>1 x Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz 32 GB RAM 4 x 1Gbit NIC</td>
</tr>
<tr>
<td>M.8.32.2N</td>
<td>1 x Intel(R) Xeon(R) CPU E3-1585L v5 @ 3.00GHz 32 GB RAM 2 x 954.0 GB NVME 2 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.8.32.v2</td>
<td>1 x Intel(R) Xeon(R) CPU E3-1585L v5 @ 3.00GHz 32 GB RAM 2 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.24.96</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz 48 GB RAM 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.24.48</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz 48 GB RAM 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.20.64</td>
<td>1 x Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz 64 GB RAM 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.16.64</td>
<td>1 x Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz 64 GB RAM 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.40.128</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz 128 GB RAM 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.20.128.1G</td>
<td>1 x Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz 128 GB RAM 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.32.128.1N</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz 128 GB RAM 1 x 745.0 GB NVME 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.40.128.1N.v3</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz 128 GB RAM 1 x 745.0 GB NVME 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.40.128.1N</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz 128 GB RAM 1 x 373.0 GB NVME 4 x 10Gbit NIC</td>
</tr>
<tr>
<td>M.24.128</td>
<td>2 x Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz 128 GB RAM 4 x 10Gbit NIC</td>
</tr>
</tbody>
</table>
Metalcloud pricing

The Metalcloud’s pricing depends on a number of factors and the services used. Depending on the contract service it can be similar to a public cloud or similar to a dedicated servers provider.

Make sure you check the pricing page on our website for offers that might help you get over the complexity if you’re just getting started.

4.1 Instance pricing

Metalcloud bare metal instances can be consumed both as on-demand and as reserved resources. More details on the various instance types can be found here: Metalcloud instances

4.1.1 On-demand bare metal servers (instances) pricing

Metalcloud bare metal instances allow for on-demand consumption. They can be provisioned and de-provisioned at any time by the customer and paid only for the duration of the utilization.
If no reservations or packages are applied, instances are priced depending on their utilization. Utilization is metered on a per-second basis but the prices are displayed per hour. Metering starts from the moment a server is allocated to an instance and is stopped when it is stopped or deleted. The time the instance was allocated to a user (“ACTIVE” state) is charged at the end of the billing cycle typically 1 month.

### 4.1.2 Reservation pricing

Reservations will override on-demand prices for matching instances and are up to 40% lower than on-demand pricing, making them suitable for more predictable scenarios.
4.1.3 How reservations work

The actual utilization of instances can still be used on-demand, the same reservation can cover two separate instances during a month.

Reservations are invoiced at the beginning of the utilization period.

4.2 Storage pricing

For a typical 720 hour month the pricing of the various storage options is as follows:

Reservations will override on-demand prices for matching instances and are up to 40% lower than on-demand pricing making them suitable for more predictable scenarios.

<table>
<thead>
<tr>
<th>Storage type</th>
<th>Euro</th>
<th>GBP</th>
<th>USD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD-backed iSCSI Drives</td>
<td>€0.10/GB</td>
<td>£0.08/GB</td>
<td>$0.10/GB</td>
</tr>
<tr>
<td>HDD-backed iSCSI Drives</td>
<td>€0.04/GB</td>
<td>£0.04/GB</td>
<td>$0.04/GB</td>
</tr>
<tr>
<td>Data Lake</td>
<td>€0.03/GB</td>
<td>£0.02/GB</td>
<td>$0.03/GB</td>
</tr>
</tbody>
</table>

If the duration is smaller than 720 hours the charge will be proportional. SharedDrive prices are the same as regular Drives.

4.3 IP space pricing

Pricing per 1 ip, used for a typical 720 hours month:

<table>
<thead>
<tr>
<th>IP Address type</th>
<th>Euro</th>
<th>GBP</th>
<th>USD</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 address</td>
<td>€2.43/ip</td>
<td>£2.13/ip</td>
<td>$2.55/ip</td>
</tr>
<tr>
<td>IPv6 address</td>
<td>€0.00/ip</td>
<td>£0.00/ip</td>
<td>$0.00/ip</td>
</tr>
</tbody>
</table>

IPv4 addresses are charged individually but are allocated in contiguous blocks to infrastructures:

- /30 (2 usable ip address)
- /29 (6 usable ip address)
- /28 (14 usable ip address)
- /27 (30 usable ip address)

If additional instances are added to the infrastructure these subnets will grow if the next contiguous ip space is unallocated. Otherwise a separate block will be allocated.

4.4 Data Transfer prices

Prices for data transfer depend on the destination of the transfer.
<table>
<thead>
<tr>
<th>Traffic type</th>
<th>Euro</th>
<th>GBP</th>
<th>USD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>egress</strong> (from the Metalcloud to the internet)</td>
<td>€0.05/ip</td>
<td>£0.04/ip</td>
<td>$0.05/ip</td>
</tr>
<tr>
<td><strong>ingress</strong> (from the internet to the Metalcloud)</td>
<td>€0.00/ip</td>
<td>£0.00/ip</td>
<td>$0.00/ip</td>
</tr>
<tr>
<td><strong>inter-datacenter</strong> (between two Metalcloud datacenters)</td>
<td>€0.01/ip</td>
<td>£0.01/ip</td>
<td>$0.01/ip</td>
</tr>
</tbody>
</table>
CHAPTER 5

Additional resources

You can find additional resources at the following links:

- Terraform Provider
- CLI
- User API Documentation
- GO SDK
- Additional SDKs
Instance Arrays are collections of identical servers used as a single entity.

### 6.1 Creating an infrastructure using the UI

1. In the infrastructure selector on the very top of the editor select **Create an infrastructure**

![Image of the editor interface showing infrastructure creation]

2. Add the name and select your datacenter.
3. Click Create

6.2 Creating an infrastructure using the CLI

This tutorial uses the CLI. Visit using the CLI for more details.

```
metalcloud-cli infrastructure create -label test1
```

6.3 Listing infrastructures using the CLI

Check if your infrastructure has been created:

```
metalcloud-cli infra list
```

Infrastructures I have access to (as alex.bordei@bigstep.com) in datacenter uk-reading

```
+-------+-------------------------------+----------------------------------+
| ID    | LABEL | OWNER | REL. |
|--------+-------+--------+------|
| 26356  | test4444 | alex.d@d.com | ordered |
| 26358  | test4443 | alex.d@d.com | ordered |
| 26359  | test1  | alex.d@d.com | ordered |
```

(continues on next page)
6.4 Changing an infrastructure’s properties using the UI

1. Access the infrastructure’s properties using the **Infrastructure Properties** tab:

Once deployed, an infrastructure’s datacenter cannot be changed any more.

The name of the infrastructure can be changed. This will have an effect on all long form DNS records such as instance array names in the form: `instance-58417.instance-array-37139.vanilla.my-infrastructure.8186.bigstep.io`

6.5 Deleting an infrastructure using the CLI

```
$ metalcloud-cli infrastructure delete -id demo
```

Where to go from here:

1. *Creating an InstanceArray*
CHAPTER 7

Managing instance arrays

This is how Servers This is often the very first step in using the Metal Cloud. Make sure you have created an account with us by signing up here

7.1 Deploying an instance array using the Infrastructure Editor

In the MetalCloud servers (called Instances) are grouped in InstanceArrays. By default an infrastructure is created for you called “my-infrastructure” in a datacenter geographically close to you.

1. Click on the Create your first InstanceArray
2. Select your configuration, number of servers, operating system, drive size and boot type.

Certain servers types support deploying the operating system on a local drive (or a collection of local drives in an RAID 0 array). Local drives do not allow switching the server but are less expensive and carry higher capacities and, if using local NVMe's higher performance.

3. Alter firewall rules

By default all traffic is blocked except if it originates from what our systems detects as being your IP. You need to explicitly enable additional IPs or ports before you deploy.
4. Deploy the infrastructure

Operations in the MetalCloud are not immediately deployed. In fact they can be reverted until the infrastructure is “Deployed”. Click on the big “Deploy” button from the bottom of the screen.
7.2 Retrieving server access credentials using the UI

Once an infrastructure is active you can access the servers’ access credentials by clicking on the instance array.

This will pop-up the access credentials window:
Here you can find, for each instance (server):

1. the quick ssh access link
2. root password

Clicking on an instance opens up further information:

The host can be any one of the:

1. Hosts’ public ip address (89..)
2. Hosts’s permanent DNS entry: instance-58417.bigstep.io

Note: It is recommended that you register your public SSH key in the Account settings section so that it gets automatically added on the hosts at deploy time.

### 7.3 Deploying an instance array using the CLI

This tutorial uses the CLI. Visit using the CLI for more details.

1. List available templates

```
$ metalcloud-cli volume-template list
Volume templates I have access to as user alex.d@d.com:
+-------+-----------------------------------------+-------------------------------+
| ID | LABEL | NAME | SIZE | STATUS | FLAGS | URL |
+-------+-----------------------------------------+-------------------------------+
| 6 | ubuntu-12-04 | Ubuntu-12.04 | 40960 | deprecateddeny provision | | 
+-------+-----------------------------------------+-------------------------------+
```

(continues on next page)
2. Provision an Instance

```bash
$ metalcloud-cli instance-array create -boot pxe_iscsi -firewall-management-disabled -infra demo -instance-count 1 -label gold
```

3. Add a drive array to the instance

Use the ID of the template, for instance 18 for CentOS 7.1

```bash
$ metalcloud-cli drive-array create -ia gold -infra demo -size 100000 -label gold-da -template 18
```

4. Deploy the infrastructure

```bash
$ metalcloud-cli infrastructure deploy -id demo
```

### 7.4 Retrieving server access credentials using the CLI

To retrieve your ssh credentials use the following command:

```bash
$ metalcloud-cli instance-array get -id gold -show-credentials
```

### 7.5 Checking the power status of an instance using the CLI

To retrieve your instances status use:

```bash
metalcloud-cli instance-array get -id workers -show-power-status
```

### 7.6 Rebooting an instance using the CLI

To perform power operations on a particular instance use:

```bash
metalcloud-cli instance power-control -id 1023 -operation reset
```

Possible operations are:

- on - power on
- off - power off (hard)
- reset - Reboot (hard)
• soft - Soft shutdown (ACPI via the operating system)
Managing drive arrays

Drives are iSCSI based block devices attached to servers at runtime. They can be operating system drives or unformatted drives.

**Drive arrays** are collections of identical drives that users can control as a single entity.

### 8.1 Creating a drive array using the UI

There are multiple ways to create a drive array from the **Infrastructure Editor**:

1. Click on an instance array and go to the **DriveArray** tabs
2. Click on Create **DriveArray** button

![Create DriveArray UI](image)

### 8.2 Listing drive arrays of an instance array using the UI

1. Click on an instance array and go to the **DriveArray** tabs
8.3 Creating a drive array using the CLI

```bash
$ metalcloud-cli drive-array create -ia gold -infra complex-demo -size 100000 -label da
```

Typically drive arrays will expand with their instance array. To stop that from happening use `-no-expand-with-ia`

8.4 Listing drive arrays of an infrastructure using the CLI

```bash
$ metalcloud-cli drive-array list -infra complex-demo
Drive Arrays I have access to as user alex.@d.com:
+-------+-------------------------------+-----------+-----------+-----------+---------
<table>
<thead>
<tr>
<th>ID</th>
<th>LABEL</th>
<th>STATUS</th>
<th>SIZE (MB)</th>
<th>TYPE</th>
<th>DRV_CNT</th>
<th>TEMPLATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>47859</td>
<td>da</td>
<td>ordered</td>
<td>100000</td>
<td>iscsi_ssd</td>
<td>1</td>
<td>gold</td>
</tr>
<tr>
<td>45928</td>
<td>drive-array-45928</td>
<td>active</td>
<td>40960</td>
<td>iscsi_ssd</td>
<td>2</td>
<td>workers</td>
</tr>
<tr>
<td>45929</td>
<td>drive-array-45929</td>
<td>active</td>
<td>40960</td>
<td>iscsi_ssd</td>
<td>1</td>
<td>master</td>
</tr>
<tr>
<td>47799</td>
<td>gold-da</td>
<td>ordered</td>
<td>100000</td>
<td>iscsi_ssd</td>
<td>1</td>
<td>gold</td>
</tr>
<tr>
<td>47858</td>
<td>test</td>
<td>ordered</td>
<td>40960</td>
<td>iscsi_ssd</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
+-------+-------------------------------+-----------+-----------+-----------+---------
```

Total: 5 Drive Arrays

8.5 Deleting a drive array via the CLI

To delete a drive array use
8.6 Manually logging into the iscsi target

Most of the time the drives will simply appear in the operating system at a reboot. However sometimes manual intervention is required. Refer to *Manually managing iSCSI connections* for more information.
The MetalSoft CLI is a powerful tool that allows you to interact quicker with the MetalSoft via the console and automate various aspects of it using bash, scripts etc.

**9.1 Installation**

To install on Mac OS X:

```
brew tap bigstepinc/homebrew-repo
brew install metalcloud-cli
```

To install on any CentOS/Redhat Linux distribution:

```
$ sudo rpm -i https://github.com/bigstepinc/metalcloud-cli/releases/download/v1.0.3/→metalcloud-cli_1.0.3_linux_amd64.rpm
```

To install on any Debian/Ubuntu distributions:

```
curl -sLO https://github.com/bigstepinc/metalcloud-cli/releases/download/v1.0.3/→metalcloud-cli_1.0.3_linux_amd64.deb && sudo dpkg -i metalcloud-cli_1.0.3_linux_→amd64.deb
```

To install on Windows: Binaries are available [here](https://github.com/bigstepinc/metalcloud-cli/releases/latest):

```
https://github.com/bigstepinc/metalcloud-cli/releases/latest
```

To install using `go get` (this should also work on Windows):

```
go get github.com/bigstepinc/metalcloud-cli
```
9.2 Getting the API key

In the Metalcloud’s Infrastructure Editor go to the upper left corner and click on your email. Then go to Settings > API & SDKs > API credentials

Copy the api key. It should be of the form:

Configure credentials as environment variables:

```bash
export METALCLOUD_API_KEY="<your key>"
export METALCLOUD_ENDPOINT="https://api.bigstep.com"
export METALCLOUD_USER_EMAIL="<your email>"
export METALCLOUD_DATACENTER="uk-reading"
```

9.3 Getting a list of supported commands

Use `metalcloud-cli help` for a list of supported commands.

9.4 Getting started

To create an infrastructure, in the default datacenter, configured via the `METALCLOUD_DATACENTER` environment variable):

```bash
metalcloud-cli infrastructure create -label test \return-id
```

<table>
<thead>
<tr>
<th>ID</th>
<th>LABEL</th>
<th>OWNER</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>complex-demo</td>
<td><a href="mailto:d.d@sdd.com">d.d@sdd.com</a></td>
</tr>
<tr>
<td>REL.</td>
<td>STATUS</td>
<td>CREATED UPDATED</td>
</tr>
</tbody>
</table>
To create an instance array in that infrastructure, get the ID of the infrastructure from above (12345):

```bash
metalcloud-cli instance-array create -infra 12345 -label master -proc 1 -proc-core-count 8 -ram 16
```

To view the id of the previously created drive array:

```bash
metalcloud-cli instance-array list -infra 12345
```

+-------+---------------------+---------------------+-----------+
| ID | LABEL | STATUS | INST_CNT |
+-------+---------------------+---------------------+-----------+
| 54321 | master | ordered | 1 |
+-------+---------------------+---------------------+-----------+

Total: 1 Instance Arrays

To create a drive array and attach it to the previous instance array:

```bash
metalcloud-cli drive-array create -infra 12345 -label master-da -ia 54321
```

To view the current status of the infrastructure

```bash
metalcloud-cli infrastructure get -id 12345
```

Infrastructures I have access to (as test@test.com)

+-------+----------------+-------------------------------+-------------------------------+-----------+
| ID | OBJECT_TYPE | LABEL | DETAILS | STATUS |
+-------+----------------+-------------------------------+-------------------------------+-----------+
| 36791 | InstanceArray | master | 1 instances (16 RAM, 8 cores, 1 disks) | ordered |
| 47398 | DriveArray | master-da | 1 drives - 40.0 GB iscsi_ssd (volume_template:0) attached to: 36791 | ordered |
+-------+----------------+-------------------------------+-------------------------------+-----------+

Total: 2 elements

### 9.5 Use flags at the end

Flags, noted with (Flag) in the command help are options that take no arguments. They need to be used at the end of the command line.

### 9.6 Condensed format

The CLI also provides a “condensed format” for most of it’s commands:

- instance-array = ia
- drive-array = da
- infrastructure = infra
- list = ls
- delete = rm

9.5. Use flags at the end 39
This allows commands such as:

```
metalcloud-cli infra ls
```

### 9.7 Using label instead of IDs

Most commands also take a label instead of an id as a parameter. For example:

```
metalcloud-cli infra show -id complex-demo
```

### 9.8 Permissions

Some commands depend on various permissions. For instance you cannot access another user’s infrastructure unless you are a delegate of it.
This is a terraform plugin for controlling Bigstep Metalcloud resources.

### 10.1 Terraform provider requirements

- Terraform 0.12.x

### 10.2 Using the Provider

A terraform `main.tf` template file, for an infrastructure with a single server would look something like this:

```terraform
provider "metalcloud" {
  user_email = var.user_email
  api_key = var.api_key
  endpoint = var.endpoint
}

data "metalcloud_volume_template" "centos76" {
  volume_template_label = "centos7-6"
}

resource "metalcloud_infrastructure" "my-infra216" {

  infrastructure_label = "my-terraform-infra216"
  datacenter_name = var.datacenter

  prevent_deploy = true

  network{
    network_type = "san"
    network_label = "san"
  }
```
network{
    network_type = "wan"
    network_label = "internet"
}

network{
    network_type = "lan"
    network_label = "private"
}

instance_array {
    instance_array_label = "example-master"
    instance_array_instance_count = 2
    instance_array_ram_gb = 8
    instance_array_processor_count = 1
    instance_array_processor_core_count = 8

    interface{
        interface_index = 0
        network_label = "san"
    }

    interface{
        interface_index = 1
        network_label = "internet"
    }

    interface{
        interface_index = 2
        network_label = "private"
    }
}

drive_array{
    drive_array_label = "example-master-os-drive"
    drive_array_storage_type = "iscsi_hdd"
    drive_size_mbytes_default = 49000
    volume_template_id = tonumber(data.metalcloud_volume_template.centos76.id)
}

firewall_rule {
    firewall_rule_description = "test fw rule"
    firewall_rule_port_range_start = 22
    firewall_rule_port_range_end = 22
    firewall_rule_source_ip_address_range_start = "0.0.0.0"
    firewall_rule_source_ip_address_range_end = "0.0.0.0"
    firewall_rule_protocol = "tcp"
    firewall_rule_ip_address_type = "ipv4"
}

instance_array {
    instance_array_label = "example-slave"
    instance_array_instance_count = 1
    instance_array_ram_gb = 8
}
instance_array_processor_count = 1
instance_array_processor_core_count = 8

drive_array{
  drive_array_label = "example-slave-os-drive"
  drive_array_storage_type = "iscsi_hdd"
  drive_size_mbytes_default = 49000
  volume_template_id = tonumber(data.metalcloud_volume_template.centos76.id)
}

firewall_rule {
  firewall_rule_description = "test fw rule"
  firewall_rule_port_range_start = 22
  firewall_rule_port_range_end = 22
  firewall_rule_source_ip_address_range_start = "0.0.0.0"
  firewall_rule_source_ip_address_range_end = "0.0.0.0"
  firewall_rule_protocol = "tcp"
  firewall_rule_ip_address_type = "ipv4"
}

Initialize the provider with your API key and the api endpoint either by using environment variables or -var.

```
export TF_VAR_api_key="<yourkey>"
export TF_VAR_user="test@test.com"
export TF_VAR_endpoint="https://api.bigstep.com/metal-cloud"
export TF_VAR_datacenter="uk-reading"
```

Initialize the provider:

```
terraform init
```

To deploy this infrastructure export the following variables (or use -var):

The plan phase:

```
terraform plan
```

The apply phase:

```
terraform apply
```

To delete the infrastructure:

```
terraform destroy
```
Creating an iSCSI OS template

Custom iSCSI templates allow users to build their own images. Typically a user would start from a standard Metalcloud template and modify it:

This tutorial uses the CLI. Visit using the CLI for more details.

### 11.1 Creating a “golden” drive

1. List available templates

   ```
   $ metalcloud-cli volume_template list
   Volume templates I have access to as user alex.d@d.com:
   +-------+-----------------------------------------+-------------------------------+
   | ID | LABEL | NAME             | SIZE | STATUS               | FLAGS |
   +-------+-----------------------------------------+-------------------------------+
   | 6 | ubuntu-12-04 | Ubuntu-12.04         | 40960 | deprecated_deny_provision |           |
   | 13 | centos6-5   | CentOS6.5            |       |                      |       |
   | 14 | centos6-6   | CentOS6.6            |       |                      |       |
   | 18 | centos71vl  | CentOS 7.1           |       |                      |       |
   +-------+-----------------------------------------+-------------------------------+
   Total: 4 Volume templates
   ```

2. Provision an Instance
3. Add a drive array to the instance

   Use the ID of the template, for instance `18` for CentOS 7.1

   ```bash
   $ metalcloud-cli drive-array create -ia gold -infra demo -size 100000 -label gold-da -template 18
   ```

4. Deploy the infrastructure

   ```bash
   $ metalcloud-cli infrastructure deploy -id demo
   ```

5. Login or connect to the server and perform required modifications, test etc.

   ```bash
   $ metalcloud-cli instance-array get -id gold -show-credentials
   ```

6. Shutdown the server to avoid in-flight data from not being serialized

   ```bash
   $ metalcloud-cli instance power-control -id 58413 -operation soft
   ```

7. List the drives of the `gold-da` drive array

   ```bash
   $ metalcloud-cli drive-array get -id gold-da
   ```

11.2 Creating a a volume template from the golden drive

   1. Create a template from the first drive

   ```bash
   $ metalcloud-cli volume-template create -id 74270 -boot-methods-supported pxe-iscsi -boot-type hybrid -label "centos7.1-custom" -description "Custom 7.1" -template -name "Custom Centos 7.1"
   ```

   The “create volume_template” operation takes the following arguments:
Notes on server boot methods:

- `-boot-methods-supported` refers to the type of mechanism employed to boot this template. For iSCSI drives use `pxe_iscsi`.
- `-boot-type` refers to the boot process of the server which can be either legacy BIOS or EFI. Hybrid refers to templates that support both mechanisms.

2. Check that the volume template has been created

```
$ metalcloud-cli volume-template list
```

```
Volume templates I have access to as user alex.d@d.com:
+-------+-----------------------------------------+-------------------------------+---------------------------+-----------+
| ID    | LABEL         | NAME                      | SIZE  | STATUS             | FLAGS         |
+-------+-----------------------------------------+-------------------------------+---------------------------+-----------+
| 6     | ubuntu-12-04 | Ubuntu-12.04              | 40960 | deprecated_deny_provision |               |
| 13    | centos6-5   | CentOS6.5                  | 40960 | deprecated_allow_expand |               |
| 14    | centos6-6   | CentOS6.6                  | 41000 | deprecated_allow_expand |               |
| 18    | centos71v1  | CentOS 7.1                 | 40960 | deprecated_allow_expand |               |
| 19    | centos71-custom | CustomCentOS 7.1       | 40960 | deprecated_allow_expand |               |
+-------+-----------------------------------------+-------------------------------+---------------------------+-----------+
Total: 5 Volume templates
```

Please note that the volume template is a private template meaning only it’s owner has access to it.

Now the template can be used as any other template.
$ metalcloud-cli drive-array create -ia gold -infra demo -size 100000 -label gold-da -template 19
Manually managing iSCSI connections

Most iscsi related operations are automatic but there are situations where manual intervention is necessary. This guide describes how to retrieve iscsi credentials, login into targets, format drives, mount them etc.

12.1 Retrieving the drive array’s connection details from the UI

1. Click on on an instance array and go to the DriveArray tabs
2. Select one of the drive arrays
3. Click on Show Drives
4. Scroll down to iSCSI credentials
12.2 Retrieving iSCSI access credentials using the CLI

Typically drive arrays that are part of the same target as other drives that are mounted in the operating system automatically. However if you need to manually mount a drive use the following:

This is the LUN that you will need to mount:

```
$ metalcloud-cli drive-array get -id drive-array-45929 -show-credentials
```

<table>
<thead>
<tr>
<th>ID</th>
<th>LABEL</th>
<th>STATUS</th>
<th>SIZE (MB)</th>
<th>TYPE</th>
<th>ATTACHED TO</th>
<th>CREDENTIALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>71587</td>
<td>drive-71587</td>
<td>active</td>
<td>40960</td>
<td>iscsi_ssd</td>
<td>instance-56008</td>
<td>CentOS 7.4(#78)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target: 100.96.0.12 Port:3260 IQN: iqn.2013-01.com.bigstep:storage.dd.6fjo87t.dd</td>
<td>CentOS none</td>
</tr>
</tbody>
</table>

This are the initiator credentials. These should already be configured on the instance but if they are not use the following to retrieve them:
12.3 Logging into the iSCSI target from Linux (CentOS/Redhat)

This process is operating system version but in general lines it requires a user to configure it’s iscsi initiator (the server) and login into the target (the storage).

1. Install iscsi support

   ```bash
   yum install iscsi-initiator-utils
   ```

2. Set node.startup to automatic.

   ```bash
   vi /etc/iscsi/iscsid.conf
   [...] 
   node.startup = automatic
   [...] 
   ```

3. Start the iSCSI discovery

   ```bash
   iscsiadm -m discovery -t st -p 100.96.0.12
   iscsiadm -m node
   ```

4. Login the iSCSI target. Notice the .33 at the end of the IQN. That’s the LUN ID from the drive array’s credentials.

   ```bash
   iscsiadm -m node --targetname "iqn.2013-01.com.bigstep:storage.dd.6fjo87t.dd.33" --portal "100.96.0.12:3260" --login
   ```

5. Identifying the drive by looking at dmesg

   ```bash
   $ dmesg
   ```

6. Formatting & mounting the drive The drive is now visible in the OS just like any other drive:
$ mkfs.ext3 /dev/sdb1
$ mount /dev/sdb1 /mnt
$ ls -l

12.4 Logging into a drive on Windows using PowerShell

1. Set iSCSI Initiator service to started and automatic

   Set-Service msiscsi -startuptype "automatic"
   Start-Service msiscsi

2. Check iSCSI Initiator Configuration Initiator Name

   (Get-WmiObject -Namespace root\wmi -Class MSiSCSIInitiator_MethodClass) | ? { $_.iSCSINodeName -ne '' } | Select-Object ISCSINodeName

3. Set iSCSI Initiator Configuration Initiator Name. Note -NewNodeAddress is retrieved from the CLI by running metalcloud-cli instance-array get -id workers -show-iscsi-credentials

   $AddInP = (Get-InitiatorPort) | Select-Object NodeAddress
   Set-InitiatorPort -NodeAddress $AddInP.NodeAddress -NewNodeAddress "iqn.2020-03.com.bigstep.storage:instance-0000"

4. Add a new portal

   New-IscsiTargetPortal -TargetPortalAddress "100.96.0.192" -AuthenticationType OneWayCHAP -ChapUsername "ss" -ChapSecret "ss"

5. Add a new target

   $GIT = Get-IscsiTarget | Where-Object { $_.IsConnected -like "False" }

6. Connect to the new target (disconnect if already connected)

   Disconnect-IscsiTarget -NodeAddress $GIT.NodeAddress -Confirm:$false
   Connect-IscsiTarget -nodeaddress $GIT.NodeAddress

7. Make persistent You can skip this step or set the -IsPersistent to false if reconnect not required at reboot

   Connect-IscsiTarget -nodeaddress $GIT.NodeAddress -IsPersistent $True

8. Connect to the target

   Connect-IscsiTarget -nodeaddress $GIT.NodeAddress -IsPersistent $False -AsJob

12.5 Prepare and format disk in Powershell

To make all attached disks online

Get-Disk | Where-Object IsOffline -Eq $True | Set-Disk -IsOffline $False
To Initialize all raw disks. This will initialize the disks and create new partitions and format the drive without confirmation.

```powershell
```

### 12.6 Disconnect from all iscsi targets in Windows using Powershell Or Just disconnect Offline target

To disconnect all connections. You may receive an error if there are files open on the associated drive. This will disconnect without confirmation.

```powershell
$GIT = Get-IscsiTarget | Where-Object {$_.IsConnected -like "True"}
Disconnect-IscsiTarget -NodeAddress $GIT.NodeAddress -Confirm:$false
```

Or you can use the below. This will disconnect without confirmation.

```powershell
Get-IscsiTarget | Where-Object IsConnected -Eq $True | Disconnect-IscsiTarget -Confirm:$false
```

Warning, the above two will disconnect ALL iSCSI drives. If your operating system is on iSCSI, it is safer to use the below two commands.

**Offline disks which are not Boot**

```powershell
Get-Disk | Where-Object IsBoot -Eq $False | Set-Disk -IsOffline $True
```

Disconnect iSCSI connection of offline disk

```powershell
Get-Disk | Where-Object -FilterScript {($_.BusType -Eq "iSCSI") -and ($_.IsOffline -Eq $True)} | Get-IscsiSession | Get-IscsiTarget | Disconnect-IscsiTarget -Confirm:$false
```

### 12.7 Disable indexing on a drive in Powershell

To disable indexing on a drive, you must first create a function.

```powershell
function Disable-Indexing{
    Param($Drive)
    $obj = Get-WmiObject -Class Win32_Volume -Filter "DriveLetter='$Drive'"
    $indexing = $obj.IndexingEnabled
    if($"$indexing" -eq $True)
    {
        write-host "Disabling indexing of drive $Drive"
        $obj | Set-WmiInstance -Arguments @{$IndexingEnabled=$False} | Out-Null
    }
}
```

To save as a function Go to C:\Program Files\WindowsPowerShell\Modules and create a folder called Indexing Save it in C:\Program Files\WindowsPowerShell\Modules\Indexing and save it as Indexing.psm1 as a script

Usage

```
```

**12.6. Disconnect from all iscsi targets in Windows using Powershell Or Just disconnect Offline** target
Disable-Indexing "d:"
where "d:" is the drive
CHAPTER 13

Additional resources

- Terraform Provider
- CLI
- GO SDK
- User API Documentation
- Extended API Documentation
- Additional SDKs