Cloudmesh REST Interface for Virtual Clusters

Gregor von Laszewski 1,* , Fugang Wang 1 , and Badi Abdhul-Wahid 1

Draft v0.0.1, April 28, 2017

This document summarizes a number of objects that are instrumental for the interaction with Clouds, Containers, and HPC systems to manage virtual clusters. TBD

© 2017 https://creativecommons.org/licenses/. The authors verify that the text is not plagiarized.

Keywords: CLoudmesh, REST, NIST

https://github.com/cloudmesh/rest/tree/master/docs

CONTENTS					33		4.2.3 Azure Vm	
2	1	1.1	oduction Lessons Learned	2 2	34 35	5	HPC 5.1 Batch Job	7 7
4		1.2	Hybrid Cloud	2	36	6	Virtual Cluster	7
5		1.3	Design by Example	2	37		6.1 Cluster	7
6		1.4 1.5	Tools to Create the Specifications	2	38		6.2 Compute Resource	8
7		1.6	Document Creation	3	39		6.3 Computer	8
8		1.7	Conversion to Word	3	40		6.4 Compute Node	8
10		1.8	Interface Compliancy	3	41		6.5 Virtual Cluster	9
		1.0	interface compilately	U	42		6.6 Virtual Compute node	9
11	2	Use	r and Profile	3	43		6.7 Virtual Machine	9
12		2.1	Profile	3	44		6.8 Mesos	9
13		2.2	User	3				
14		2.3	Organization	3	45	7	Containers	9
15		2.4	Group/Role	4	46		7.1 Container	
	3	Data		4	47		7.2 Kubernetes	10
16 17	3	3.1	Var	4		0	Doulovement	10
18		3.2	Default	4	48	8	Deployment 8.1 Deployment	10
19		3.3	File	4	49		8.1 Deployment	10
20		3.4	File Alias	5	50	9	Mapreduce	10
21		3.5	Replica	5	51		9.1 Hadoop	
22		3.6	Virtual Directory	5	52		9.2 Mapreduce	
23		3.7	Database	5			7-2	
24		3.8	Stream	5	53	10	Security	11
	_				54		10.1 Key	11
25	4	IaaS		6				
26		4.1	Openstack	6	55	11	Microservice	11
27			4.1.1 Openstack Flavor	6	56		11.1 Microservice	
28			4.1.2 Openstack Image	6	57		11.2 Reservation	11
29		4.2	4.1.3 Openstack Vm	6 6		10	Network	11
30		1.4	4.2.1 Azure Size	6	58	14	INCLIMOTE	11
32			4.2.2 Azure Image	7	59	Α	Schema Command	12
<i></i>				•	-			

¹ School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

^{*}Corresponding authors: laszewski@gmal.com

В	Schema	12
C	Contributing	24
D	Using the Cloudmesh REST Service	24
D	D.1 Element Definition	25
	D.2 Yaml	25
	D.3 Cerberus	25
_		
E	Mongoengine	25
F	Cloudmesh Notation	25
	F.1 Defining Elements for the REST Service	25
	F.2 DOIT	25
	F.3 Generating service	25
G	ABC	25
Н	Cloudmesh Rest	25
	H.1 Prerequistis	25
	H.1.1 Install Mongo on OSX	25
	H.1.2 Install Mongo on OSX	25
	H.2 Introduction	26
	H.3 Yaml Specification	26
	H.4 Json Specification	26
	H.5 Conversion to Eve Settings	26
	H.5.1 Managing Mongo	26
	H.5.2 Manageing Eve	26
I	REST with Eve	26
	I.1 Overview of REST	26
	I.2 REST and eve	27
	I.2.1 Installation	27
	I.2.2 Starting the service	27
	I.3 Creating your own objects	27
	I.4 Towards cmd5 extensions to manage eve and	
	mongo	27
Ţ	CMD5	28
•	J.1 Resources	28
	J.2 Execution	28
	J.3 Create your own Extension	28
	J.4 Excersise	29
1.	INTRODUCTION	
In-	this document we summarize elementary objects that	are
	portant to for the NBDRA.	
1.1	. Lessons Learned	
TH	BD	
1.2	. Hybrid Cloud	
	•	$\overline{}$
TH	BD .	

To accelerate discussion among the team we use an approach to define objects and its interfaces by example. These examples are than taken in a later version of the document and a schema is generated from it. The schema will be added in its complete form to the appendix B. While focusing first on examples it allows us to speed up our design and simplifies

104

105

106

107

discussions of the objects and interfaces eliminating getting lost in complex syntactical specifications. The process and specifications used in this document will also allow us to automatically create a implementation of the objects that can be integrated into a reference architecture as provided by for example the cloudmesh client and rest project [?].

An example object will demonstrate our approach. The following object defines a JSON object representing a user.

```
Listing 1.1: User profile

{

    "profile": {
        "description": "The Profile of a user",
        "uuid": "jshdjkdh...",
        "context:": "resource",
        "email": "laszewski@gmail.com",
        "firstname": "Gregor",
        "lastname": "von Laszewski",
        "username": "gregor"
}

}
```

Such an object can be transformed to a schema specification while introspecting the types of the original example. The resulting schema object follows the Cerberus [?] specification and looks for our object as follows:

```
profile = {
   'description': { 'type': 'string'},
   'email': { 'type': 'email' },
   'firstname': { 'type': 'string'},
   'lastname': { 'type': 'string' },
   'username': { 'type': 'string' }
}
```

As mentioned before, the AppendixB will list the schema that is automatically created from the definitions.

1.4. Tools to Create the Specifications

The tools to create the schema and object are all available opensource and are hosted on github. It includes the following repositories:

cloudmesh.common

https://github.com/cloudmesh/cloudmesh.common

cloudmesh.cmd5

https://github.com/cloudmesh/cloudmesh.cmd5

cloudmesh.rest

https://github.com/cloudmesh/cloudmesh.rest

cloudmesh/evegenie

https://github.com/cloudmesh/evegenie

1.5. Installation of the Tools

The current best way to install the tools is from source. A convenient shell script conducting the install is located at:

TBD

Once we have stabilized the code the package will be available from pypi and can be installed as follows:

```
pip install cloudmesh.rest
pip install cloudmesh.evengine
```

142 1.6. Document Creation

It is assumed that you have installed all the tools. TO create the document you can simply do

git clone https://github.com/cloudmesh/cloudmesh.resttdcccdcloudmesh.rest/doccsmake

This will produce in that directory a file called object.pdf containing this document.

1.7. Conversion to Word

We found that it is inconvenient for the developers to maintain this document in Microsoft Word as typically is done for other documents. This is because the majority of the information contains specifications that are directly integrated in a reference implementation, as well as that the current majority of contributors are developers. We would hope that editorial staff provides direct help to improve this document, which even can be done through the github Web interface and does not require any access either to the tools mentioned above or the availability of LATEX.

The files are located at:

 https://github.com/cloudmesh/cloudmesh.rest/tree/master/ docs

1.8. Interface Compliancy

Due to the extensibility of our interfaces it is important to introduce a terminology that allows us to define interface compliancy. We define it as follows

Full Compliance: These are reference implementations that provide full compliance to the objects defined in this document. A version number will be added to assure the snapshot in time of the objects is associated with the version. This reference implementation will implement all objects.

Partially Compliance: These are reference implementations that provide partial compliance to the objects defined in this document. A version number will be added to assure the snapshot in time of the objects is associated with the version. This reference implementation will implement a partial list of the objects. A document is accompanied that lists all objects defined, but also lists the objects that are not defined by the reference architecture.

Full and extended Compliance: These are interfaces that in addition to the full compliance also introduce additional interfaces and extend them.

2. USER AND PROFILE

In a multiuser environment we need a simple mechanism of associating objects and data to a particular person or group. While we do not want to replace with our efforts more elaborate solutions such as proposed by eduPerson (http://software.internet2.edu/eduperson/internet2-mace-dir-eduperson-201602.html) or others [?], we need a very simple way of distinguishing users. Therefore we have introduced a number of simple objects including a profile and a user.

2.1. Profile

A profile is simple the most elementary information to distinguish a user profile. It contains name and e-mail information. It may have an optional uuid and/or use a unique e-mail to distinguish a user.

```
Listing 2.1: User profile

{
    "profile": {
        "description": "The Profile of a user",
        "uuid": "jshdjkdh...",
        "context:": "resource",
        "email": "laszewski@gmail.com",
        "firstname": "Gregor",
        "lastname": "von Laszewski",
        "username": "gregor"
    }
}
```

2.2. User

In contrast to the profile a user contains additional attributs that define the role of the user within the system.

```
Listing 2.2: user

{

"user": {

"uuid": "jshdjkdh...",

"context:": "resource",

"email": "laszewski@gmail.com",

"firstname": "Gregor",

"lastname": "von Laszewski",

"username": "gregor",

"roles": ["admin", "user"]

}

}

}
```

2.3. Organization

An important concept in many applications is the management of a roup of users in a virtual organization. This can be achieved through two concepts. First, it can be achieved while useing the profile and user resources itself as they contain the ability to manage multiple users as part of the REST interface. The second concept is to create a virtual organization that lists all users of this virtual organization. The third concept is to introduce groups and roles either as part of the user definition or as part of a simple list similar to the organization

```
Listing 2.3: user

{

"organization": {

"users": [

"objectid:user"

5

]

}
}
```

247

248

249

250 251

252

253

254

255

256

257

258

259

261

264

265

267

274

2.4. Group/Role

215

216

217

219

221

222

A group contains a number of users. It is used to manage authorized services.

```
Listing 2.4: group
    "group": {
       "name": "users",
       "description": "This group contains all
    users",
       "users": [
         "objectid:user"
      ٦
}
```

A role is a further refinement of a group. Group members can have specific roles. A good example is that ability to formulate a group of users that have access to a repository. However the role defines more specifically read and write privileges to the data within the repository.

```
Listing 2.5: role
    "role": {
      "name": "editor",
      "description": "This role contains all
    editors",
       "users": [
         "objectid:user"
}
```

3. DATA

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

Data for Big Data applications are delivered through data providers. They can be either local providers contributed by a user or distributed data providers that refer to data on the internet. At this time we focus on an elementary set of abstractions related to data providers that offer us to utilize variables, files, virtual data directories, data streams, and data filters.

Variables are used to hold specific contents that is associated in programming language as a variable. A variable has a name, value and type.

Default is a special type of variable that allows adding of a 270 context. defaults can than created for different contexts. 271

Files are used to represent information collected within the 273 context of classical files in an operating system.

Streams are services that offer the consumer a stream of data. 276 Streams may allow the initiation of filters to reduce the amount of data requested by the consumer Stream Filters operate in streams or on files converting them to streams

Batch Filters operate on streams and on files while working in the background and delivering as output Files

Virtual directories and non-virtual directories are collection of files that organize them. For our initial purpose the distinction between virtual and non-virtual directories is non-essential and we will focus on abstracting all directories to be virtual. This could mean that the files are physically hosted on different disks. However, it is important to note that virtual data directories can hold more than files, they can also contain data streams and data filters.

variables are used to store a simple values. Each variable can have a type. The variable value format is defined as string to allow maximal probability. The type of the value is also provided.

```
Listing 3.1: var
{
  "var": {
    "name": "name of the variable",
    "value": "the value of the variable as
    string",
    "type": "the datatype of the variable such
    as int, str, float, ..."
  }
}
```

3.2. Default

A default is a special variable that has a context associated with it. This allow su to define values that can be easily retrieved based on its context. A good example for a default would be the image name for a cloud where the context is defined by the cloud name.

```
Listing 3.2: default
{
  "default": {
    "value": "string",
    "name": "string",
    "context": "string
                         - defines the context
    of the default (user, cloud, ...)"
  }
}
```

A file is a computer resource allowing to store data that is being processed. The interface to a file provides the mechanism to appropriately locate a file in a distributed system. Identification include the name, and andpoint, the checksum and the size. Additional parameters such as the lasst access time could be stored also. As such the Interface only describes the location of the file

The **file** object has *name*, *endpoint* (location), *size* in GB, MB, Byte, *checksum* for integrity check, and last *accessed* timestamp.

```
Listing 3.3: file
{
    "file": {
         "name": "report.dat",
```

```
"endpoint":

"file://gregor@machine.edu:/data/report.dat",

"checksum":

{"md5":"8c324f12047dc2254b74031b8f029ad0"},

"accessed": "1.1.2017:05:00:00:EST",

"created": "1.1.2017:05:00:00:EST",

"modified": "1.1.2017:05:00:00:EST",

"size": ["GB", "Byte"]
}
}
}
```

3.4. File Alias

279

281

282

284

285

286

287

289

290

291

292

293

A file could have one alias or even multiple ones.

```
Listing 3.4: file alias

{
    "file_alias": {
        "alias": "report-alias.dat",
        "name": "report.dat"
}
}
```

3.5. Replica

In many distributed systems, it is of importance that a file can be replicated among different systems in order to provide faster access. It is important to provide a mechanism that allows to trace the pedigree of the file while pointing to its original source

```
Listing 3.5: replica
      "replica": {
        "name": "replica_report.dat",
        "replica": "report.dat",
        "file://gregor@machine.edu:/data/replica_report.da
        "checksum": {
             "md5":
        "8c324f12047dc2254b74031b8f029ad0"
        },
        "accessed": "1.1.2017:05:00:00:EST",
        "size": [
10
                                                             305
11
           "GB",
           "Byte"
                                                             306
12
                                                             307
        ٦
13
      }
                                                             308
14
   }
                                                             309
```

3.6. Virtual Directory

A collection of files or replicas. A virtual directory can contain an number of entities cincluding files, streams, and other virtual directories as part of a collection. The element in the collection can either be defined by uuid or by name.

```
"endpoint": "http://.../data/",
"protocol": "http",
"collection": [
    "report.dat",
    "file2"
]

10
}
}
```

3.7. Database

297

298

A **database** could have a name, an *endpoint* (e.g., host:port), and protocol used (e.g., SQL, mongo, etc.).

```
Listing 3.7: database

{

"database": {

"name": "data",

"endpoint": "http://.../data/",

"protocol": "mongo"

}

}
```

3.8. Stream

A stream proveds a stream of data while providing information about rate and number of items exchanged while issuing requests to the stream. A stream my return data items in a specific fromat that is defined by the stream.

```
Listing 3.8: stream

{

"stream": {

"name": "name of the variable",

"format": "the format of the data

da 
exchanged in the stream",

"attributes": {

"rate": 10,

"limit": 1000

}

}

}

}
```

Examples for streams could be a stream of random numbers but could also include more complex formats such as the retrieval of data records.

Services can subscribe, unsubscribe from a stream, while also applying filters to the subscribed stream.

```
Listing 3.9: filter

{

"filter": {

"name": "name of the filter",

"function": "the function of the data

exchanged in the stream"

}

}
```

Filter needs to be refined

313 4. IAAS

317

318

In this section we are defining resources related to Infrastructure as a Service frameworks. This includes specific objects useful for OpenStack, Azure, and AWS, as well as others.

4.1. Openstack

4.1.1. Openstack Flavor

```
Listing 4.1: openstack flavor

{

"openstack_flavor": {

"os_flv_disabled": "string",

"uuid": "string",

"os_flv_ext_data": "string",

"ram": "string",

"os_flavor_acces": "string",

"vcpus": "string",

"swap": "string",

"rxtx_factor": "string",

"disk": "string"

}

}
```

320 4.1.2. Openstack Image

```
Listing 4.2: openstack image
    {
      "openstack_image": {
        "status": "string",
        "username": "string",
        "updated": "string",
        "uuid": "string",
        "created": "string",
        "minDisk": "string",
        "progress": "string",
        "minRam": "string",
10
        "os_image_size": "string",
11
        "metadata": {
12
          "image_location": "string",
13
          "image_state": "string",
14
          "description": "string",
15
          "kernel_id": "string",
16
          "instance_type_id": "string",
17
          "ramdisk_id": "string",
18
19
          "instance_type_name": "string",
20
          "instance_type_rxtx_factor": "string",
          "instance_type_vcpus": "string",
21
          "user_id": "string",
22
          "base_image_ref": "string",
23
          "instance_uuid": "string",
24
          "instance_type_memory_mb": "string",
25
          "instance_type_swap": "string",
26
          "image_type": "string",
27
28
          "instance_type_ephemeral_gb": "string",
29
          "instance_type_root_gb": "string",
          "network_allocated": "string",
30
31
          "instance_type_flavorid": "string",
          "owner_id": "string"
32
        }
33
      }
34
```

```
5 }
```

4.1.3. Openstack Vm

```
Listing 4.3: openstack vm
    {
      "openstack_vm": {
        "username": "string",
        "vm_state": "string",
        "updated": "string",
        "hostId": "string",
        "availability_zone": "string",
        "terminated_at": "string",
        "image": "string",
        "floating_ip": "string",
10
        "diskConfig": "string",
11
12
        "key": "string",
13
        "flavor__id": "string",
        "user_id": "string",
14
        "flavor": "string",
15
        "static_ip": "string",
16
        "security_groups": "string",
17
        "volumes_attached": "string",
18
        "task_state": "string",
19
20
        "group": "string",
        "uuid": "string",
21
        "created": "string",
22
        "tenant_id": "string",
23
        "accessIPv4": "string",
24
        "accessIPv6": "string",
25
        "status": "string",
26
27
        "power_state": "string",
        "progress": "string",
28
        "image__id": "string",
29
        "launched_at": "string",
30
        "config_drive": "string"
31
32
   }
```

4.2. Azure

324

325

4.2.1. Azure Size

The size description of an azure vm

```
Listing 4.4: azure-size
      "azure-size": {
        "_uuid": "None",
        "name": "D14 Faster Compute Instance",
        "extra": {
          "cores": 16,
          "max_data_disks": 32
        "price": 1.6261,
10
        "ram": 114688,
        "driver": "libcloud",
11
12
        "bandwidth": "None",
        "disk": 127,
13
        "id": "Standard_D14"
14
15
```

```
16 }
```

4.2.2. Azure Image

```
Listing 4.5: azure-image
     "azure_image": {
       "_uuid": "None",
       "driver": "libcloud",
        "extra": {
          "affinity_group": "",
          "category": "Public",
          "description": "Linux VM image with
       coreclr-x64-beta5-11624 installed to
       /opt/dnx. This image is based on Ubuntu
       14.04 LTS, with prerequisites of CoreCLR
       installed. It also contains PartsUnlimited
       demo app which runs on the installed
       coreclr. The demo app is installed to
       /opt/demo. To run the demo, please type
       the command /opt/demo/Kestrel in a
       terminal window. The website is listening
       on port 5004. Please enable or map a
       endpoint of HTTP port 5004 for your azure
       VM.".
          "location": "East Asia; Southeast
       Asia; Australia East; Australia
       Southeast; Brazil South; North Europe; West
       Europe; Japan East; Japan West; Central
       US; East US; East US 2; North Central
       US; South Central US; West US",
          "media_link": "",
10
         "os": "Linux",
11
          "vm_image": "False"
12
13
       },
       "id": "03f55de797f546a1b29d1....",
14
       "name": "CoreCLR x64 Beta5 (11624) with
       PartsUnlimited Demo App on Ubuntu Server
       14.04 LTS"
     }
16
   }
```

4.2.3. Azure Vm

331

332

An Azure virtual machine

```
Listing 4.6: azure-vm
      "azure-vm": {
        "username": "string",
        "status": "string",
        "deployment_slot": "string",
        "cloud_service": "string",
        "image": "string",
        "floating_ip": "string",
        "image_name": "string",
        "key": "string",
10
11
        "flavor": "string",
        "resource_location": "string",
12
        "disk_name": "string",
13
        "private_ips": "string",
```

```
"group": "string",
  15
           "uuid": "string",
  16
           "dns_name": "string",
  17
           "instance_size": "string",
  18
           "instance_name": "string",
  19
           "public_ips": "string",
  20
           "media_link": "string"
  21
  22
      }
  23
335
```

6 5. HPC

5.1. Batch Job

```
Listing 5.1: batchjob
        "batchjob": {
          "output_file": "string",
          "group": "string",
          "job_id": "string",
          "script": "string, the batch job script",
          "cmd": "string, executes the cmd, if None
          path is used",
          "queue": "string",
          "cluster": "string",
          "time": "string",
          "path": "string, path of the batchjob, if
  11
          non cmd is used",
          "nodes": "string",
  12
          "dir": "string"
  13
  14
     }
  15
338
```

6. VIRTUAL CLUSTER

6.1. Cluster

341

345

The cluster object has name, label, endpoint and provider. The *endpoint* defines.... The *provider* defines the nature of the cluster, e.g., its a virtual cluster on an openstack cloud, or from AWS, or a bare-metal cluster.

```
Listing 6.1: cluster
                                                      </>
    {
      "cluster": {
        "label": "c0",
        "endpoint": {
           "passwd": "secret",
           "url": "https"
        },
         "name": "myCLuster",
        "provider": [
           "openstack",
10
           "aws",
11
12
           "azure",
           "eucalyptus"
13
14
      }
15
   }
```

6.2. Compute Resource

347

348

349

350

351

352

355

356

357

360

361

363

365

366

367

368

369

372

373

374

375

377

378

379

380

381

382

An important concept for big data analysis it the representation of a compute resource on which we execute the analysis. We define a compute resource by name and by endpoint. A compute resource is an abstract concept and can be instantiated through virtual machines, containers, or bare metal resources. This is defined by the "kind" of the compute resource

compute_resource object has attribute *endpoint* which specifies ... The *kind* could be *baremetal* or *VC*.

```
Listing 6.2: compute resource

{

    "compute_resource": {

        "name": "Compute1",

        "endpoint": "http://.../cluster/",

        "kind": "baremetal"

}

}
```

6.3. Computer

This defines a **computer** object. A computer has name, label, IP address. It also listed the relevant specs such as memory, disk size, etc.

```
Listing 6.3: computer

{
    "computer": {
        "ip": "127.0.0.1",
        "name": "myComputer",
        "memoryGB": 16,
        "label": "server-001"
    }
}
```

6.4. Compute Node

A node is composed of multiple components:

- 1. Metadata such as the name or owner.
- 2. Physical properties such as cores or memory.
- Configuration guidance such as create_external_ip, security_groups, or users.

The metadata is associated with the node on the provider end (if supported) as well as in the database. Certain parts of the metadata (such as owner) can be used to implement access control. Physical properties are relevant for the initial allocation of the node. Other configuration parameters control and further provisioning.

In the above, after allocation, the node is configured with a user called hello who is part of the wheel group whose account can be accessed with several SSH identities whose public keys are provided (in authorized_keys).

Additionally, three ssh keys are generated on the node for the hello user. The first uses the ed25519 cryptographic method with a password read in from a GPG-encrypted file on the Command and Control node. The second is a 4098-bit RSA key also password-protected from the GPG-encrypted

file. The third key is copied to the remote node from an encrypted file on the Command and Control node.

This definition also provides a security group to control access to the node from the wide-area-network. In this case all ingress and egress TCP and UDP traffic is allowed provided they are to ports 22 (SSH), 443 (SSL), and 80 and 8080 (web).

```
Listing 6.4: node
      "node_new": {
         "authorized_keys": [
           "ssh-rsa AAAA...",
           "ssh-ed25519 AAAA...",
           "...etc"
        ],
         "name": "example-001",
         "external_ip": "",
         "loginuser": "root",
10
11
         "create_external_ip": true,
         "internal_ip": "",
12
         "memory": 2048,
13
         "owner": "",
14
         "cores": 2,
15
16
         "users": {
           "name": "hello",
17
           "groups": [
18
             "wheel"
19
20
        },
21
         "disk": 80.
22
         "security_groups": [
23
24
             "ingress": "0.0.0.0/32",
25
26
             "egress": "0.0.0.0/32",
27
             "ports": [
               22,
28
               443.
29
               80,
30
               8080
31
32
             "protocols": [
33
                "tcp",
34
                "udp"
35
             ]
36
           }
37
38
        ],
         "ssh_keys": [
39
40
           {
             "to": ".ssh/id_rsa",
41
             "password": {
42
                "decrypt": "gpg",
43
                "from": "yaml",
44
                "file": "secrets.yml.gpg",
45
                "key": "users.hello.ssh[0]"
46
47
             },
48
             "method": "ed25519",
49
             "ssh_keygen": true
           }
50
           {
51
             "to": ".ssh/testing",
52
53
             "password": {
```

404

405

```
"decrypt": "gpg",
               "from": "yaml",
55
               "file": "secrets.yml.gpg",
56
               "key": "users.hello.ssh[1]"
57
             },
58
             "bits": 4098,
59
             "method": "rsa",
60
             "ssh_keygen": true
61
          },
62
63
             "decrypt": "gpg",
64
             "from":
65
        "secrets/ssh/hello/copied.gpg",
             "ssh_keygen": false,
66
             "to": ".ssh/copied"
67
68
        ]
69
      }
70
    }
```

6.5. Virtual Cluster

392

393

394

395

397

399

A virtual cluster is an agglomeration of virtual compute nodes that constitute the cluster. Nodes can be assembled to be baremetal, virtual machines, and containers. A virtual cluster contains a number of virtual compute nodes.

```
Listing 6.5: virtual cluster

{

    "virtual_cluster": {
        "name": "myvc",
        "frontend": "objectid:virtual_machine",
        "nodes": [
            "objectid:virtual_machine"
        ]
    }
}
```

6.6. Virtual Compute node

```
Listing 6.6: virtual compute node
      "virtual_compute_node": {
        "name": "data",
        "endpoint": "http://.../cluster/",
        "metadata": {
          "experiment": "exp-001"
        },
        "image": "Ubuntu-16.04",
        "ip": [
          "TBD"
10
        ],
11
        "flavor": "TBD",
12
        "status": "TBD"
13
14
   }
```

6.7. Virtual Machine

Virtual Machine Virtual machines are an emulation of a computer system. We are maintaining a very basic set of infor-

mation. It is expected that through the endpoint the virtual machine can be introspected and more detailed information can be retrieved.

```
Listing 6.7: virtual machine
      "virtual_machine" :{
        "name": "vm1",
        "ncpu": 2,
        "RAM": "4G",
        "disk": "40G",
        "nics": ["objectid:nic"
        "OS": "Ubuntu-16.04",
        "loginuser": "ubuntu",
11
        "status": "active",
12
        "metadata":{
13
        "authorized_keys": [
14
           "objectid:sshkey"
15
        ]
16
17
   }
```

6.8. Mesos

```
Refine
    Listing 6.8: mesos
      "mesos-docker": {
        "instances": 1,
         "container": {
          "docker": {
             "credential": {
               "secret": "my-secret",
               "principal": "my-principal"
             "image": "mesosphere/inky"
11
          },
          "type": "MESOS"
12
        },
13
        "mem": 16.0,
14
        "args": [
15
          "argument"
16
17
18
        "cpus": 0.2,
         "id": "mesos-docker"
19
20
   }
21
```

7. CONTAINERS

7.1. Container

409

411

412

This defines **container** object.

```
Listing 7.1: container

{
    "container": {
    "name": "container1",
```

```
"endpoint": "http://.../container/",
        "ip": "127.0.0.1",
        "label": "server-001",
        "memoryGB": 16
    }
}
```

7.2. Kubernetes

Listing 7.2: kubernetes

REFINE 416

```
{
      "kubernetes": {
         "kind": "List",
         "items": [
           {
             "kind": "None",
             "metadata": {
               "name": "127.0.0.1"
             },
             "status": {
10
                "capacity": {
11
                  "cpu": "4"
12
               },
13
                "addresses": [
14
                  {
15
                     "type": "LegacyHostIP",
16
                     "address": "127.0.0.1"
17
18
               ]
19
             }
20
21
           },
22
             "kind": "None",
23
             "metadata": {
24
                "name": "127.0.0.2"
25
             },
26
27
             "status": {
                "capacity": {
28
                  "cpu": "8"
29
               },
30
                "addresses": [
31
32
                    "type": "LegacyHostIP",
33
                     "address": "127.0.0.2"
34
                  },
35
36
                     "type": "another",
37
                     "address": "127.0.0.3"
38
                  }
39
               ]
40
             }
41
42
           }
43
        ],
         "users": [
44
45
           {
             "name": "myself",
46
             "user": "gregor"
47
           },
```

```
{
  49
                 "name": "e2e",
  50
                 "user": {
  51
                    "username": "admin",
  52
                    "password": "secret"
  53
  54
  55
            ]
  56
         }
  57
       }
  5
418
```

8. DEPLOYMENT

8.1. Deployment

420

421

A **deployment** consists of the resource *cluster*, the location provider, e.g., AWS, OpenStack, etc., and software stack to be 422 deployed (e.g., hadoop, spark). 423

```
Listing 8.1: deployment
        "deployment": {
             "cluster": [{ "name": "myCluster"},
                          { "id" : "cm-0001"}
                         ],
             "stack": {
                 "layers": [
                      "zookeeper",
                      "hadoop",
                      "spark",
10
11
                      "postgresql"
12
                 ],
                  "parameters": {
13
                      "hadoop": {
14
         "zookeeper.quorum": [ "IP", "IP", "IP"]
15
16
                 }
17
            }
18
        }
   }
```

9. MAPREDUCE

9.1. Hadoop

424

425

426

427

428

429

430

A **hadoop** definition defines which *deployer* to be used, the parameters of the deployment, and the system packages as requires. For each requirement, it could have attributes such as the library origin, version, etc.

```
Listing 9.1: hadoop
  "hadoop": {
    "deployers": {
      "ansible":
    "git://github.com/cloudmesh_roles/hadoop"
    },
    "requires": {
      "java": {
        "implementation": "OpenJDK",
        "version": "1.8",
```

441

443

445

446

447

448

449

450

451

452

455

456

```
"zookeeper": "TBD",
  10
                "supervisord": "TBD"
  11
             }
  12
                                                                442
          },
  13
           "parameters": {
  14
             "num_resourcemanagers": 1,
  15
             "num_namenodes": 1,
  16
  17
             "use_yarn": false,
             "use_hdfs": true,
  18
             "num_datanodes": 1,
  19
             "num_historyservers": 1,
  20
  21
             "num_journalnodes": 1
  22
  23
        }
      }
  24
432
```

9.2. Mapreduce

433

This defines a mapreduce deployment with its layered com-434 ponents. 435

```
Listing 9.2: mapreduce
        "mapreduce": {
            "layers": [
                 "hadoop"
            ],
            "hdfs_datanode": "IP ADDRESS",
            "java": {
                 "platform": "OpenJDK",
                 "version": "1.8"
            },
            "supervisord": "",
  11
            "hdfs_namenode": "IP ADDRESS",
  12
            "zookeeper": "IP ADDRESS",
  13
            "yarn_historyserver": "IP ADDRESS",
  14
            "hdfs_journalnode": "IP ADDRESS",
  15
            "yarn_resourcemanager": "IP ADDRESS"
  16
  17
        }
      }
436
```

10. SECURITY

10.1. Key

439

```
Listing 10.1: key
                                                </>>
  "sshkey": {
    "comment": "string",
    "source": "string",
    "uri": "string",
    "value": "ssh-rsa",
     "fingerprint": "string, unique"
  }
}
```

11. MICROSERVICE

11.1. Microservice

introduce registry we can register many things to it latency provide example on how to use each of them, not just the object definition example

necessity of local direct attached storage. Mimd model to storage Kubernetis, mesos can not spin up? Takes time to spin them up and coordinate them. While setting up environment takes more than using the microservice, so we must make sure that the micorservices are used sufficiently to offset spinup cost.

limitation of resource capacity such as networking.

Benchmarking to find out thing about service level agreement to access the

A system could be composed of from various microservices, and this defines each of them.

```
Listing 11.1: microservice
                                                </>>
"microservice" :{
    "name": "ms1",
    "endpoint": "http://.../ms/",
    "function": "microservice spec"
    }
}
```

11.2. Reservation

```
Listing 11.2: reservation
    {
      "reservation": {
        "hosts": "string",
        "description": "string",
        "start_time": [
           "date",
           "time"
        "end_time": [
           "date",
10
           "time"
11
        ]
12
      }
13
    }
```

12. NETWORK

We are looking for volunteers to contribute here.

461 A. SCHEMA COMMAND

462 B. SCHEMA

463 TBD

```
Listing B.1: schema
    profile = {
        'schema': {
             'username': {
                 'type': 'string'
             'context:': {
                'type': 'string'
            },
             'description': {
                'type': 'string'
11
12
             'firstname': {
                 'type': 'string'
13
            },
14
             'lastname': {
15
                 'type': 'string'
16
17
             'email': {
18
                 'type': 'string'
19
20
            },
             'uuid': {
21
22
                 'type': 'string'
23
        }
24
   }
25
26
27
    virtual_machine = {
28
        'schema': {
             'status': {
29
                 'type': 'string'
30
            },
31
             'authorized_keys': {
32
                 'type': 'list',
33
                 'schema': {
34
                      'type': 'objectid',
35
                      'data_relation': {
36
37
                          'resource': 'sshkey',
                          'field': '_id',
38
                          'embeddable': True
39
                     }
40
                 }
41
            },
42
             'name': {
43
                 'type': 'string'
44
            },
45
             'nics': {
46
                 'type': 'list',
47
48
                 'schema': {
                      'type': 'objectid',
49
50
                      'data_relation': {
51
                          'resource': 'nic',
                          'field': '_id',
52
                          'embeddable': True
53
                     }
54
                 }
55
```

```
},
56
              'RAM': {
57
                 'type': 'string'
58
59
              'ncpu': {
60
                 'type': 'integer'
61
62
             },
              'loginuser': {
63
                 'type': 'string'
64
             },
65
              'disk': {
66
67
                 'type': 'string'
             },
              '0S': {
                  'type': 'string'
70
             },
71
              'metadata': {
72
                  'type': 'dict',
73
                  'schema': {}
74
75
76
         }
77
78
    kubernetes = {
79
         'schema': {
80
             'items': {
81
                  'type': 'list',
82
83
                  'schema': {
84
                      'type': 'dict',
                      'schema': {
85
                           'status': {
86
                                'type': 'dict',
87
                                'schema': {
88
89
                                    'capacity': {
90
                                         'type':
     → 'dict',
                                         'schema': {
91
                                             'cpu': {
92
93
     → 'type': 'string'
                                             }
94
                                        }
95
                                    },
                                    'addresses': {
98
                                        'type':
     → 'list',
                                         'schema': {
99
                                             'type':
100
     → 'dict',
                                             'schema':
101
     ← {
102
     → 'type': {
103
     → 'type': 'string'
                                                 },
104
105
     → 'address': {
106
     → 'type': 'string'
                                                 }
107
```

```
}
                                                                                  'bandwidth': {
 108
                                                                    166
                                            }
                                                                                      'type': 'string'
                                                                    167
 109
                                       }
                                                                                  },
 110
                                                                    168
                                   }
                                                                                  'mtu': {
 111
                                                                    169
                              },
                                                                                      'type': 'integer'
 112
                                                                    170
                              'kind': {
 113
                                                                    171
                                   'type': 'string'
                                                                                  'broadcast': {
 114
                                                                    172
                              },
                                                                                      'type': 'string'
                                                                    173
 115
                               'metadata': {
                                                                                  },
                                                                    174
 116
                                   'type': 'dict',
                                                                                  'mac': {
 117
                                                                    175
                                   'schema': {
                                                                                      'type': 'string'
 118
                                                                    176
 119
                                        'name': {
                                                                    177
                                                                                  'type': {
 120
                                            'type':
                                                                    178
       → 'string'
                                                                                      'type': 'string'
                                                                    179
 121
                                                                    180
                                   }
                                                                                  'gateway': {
 122
                                                                    181
                              }
                                                                                     'type': 'string'
 123
                                                                    182
                         }
 124
                                                                    183
                    }
                                                                             }
 125
                                                                    184
               },
                                                                    185
 126
 127
                'kind': {
                                                                    186
                    'type': 'string'
                                                                    187
                                                                         virtual_compute_node = {
 128
                },
 129
                                                                    188
                                                                             'schema': {
                'users': {
                                                                    189
                                                                                  'status': {
 130
                    'type': 'list',
                                                                                      'type': 'string'
 131
                                                                    190
                     'schema': {
 132
                                                                    191
                         'type': 'dict',
 133
                                                                    192
                                                                                  'endpoint': {
                         'schema': {
                                                                    193
                                                                                      'type': 'string'
 135
                              'name': {
                                                                    194
                                  'type': 'string'
                                                                                  'name': {
 136
                                                                    195
                              },
                                                                                      'type': 'string'
 137
                                                                    196
                                                                                  },
                              'user': {
                                                                    197
 138
                                   'type': 'dict',
                                                                                  'ip': {
 139
                                                                    198
                                   'schema': {
                                                                                       'type': 'list',
 140
                                                                    199
                                        'username': {
                                                                                       'schema': {
 141
                                                                    200
                                             'type':
                                                                                           'type': 'string'
 142
                                                                    201
       → 'string'
                                                                    202
                                       },
                                                                                  },
 143
                                                                   203
                                                                                  'image': {
                                        'password': {
 144
                                                                   204
                                            'type':
                                                                                      'type': 'string'
 145
                                                                   205
                                                                                  },
       → 'string'
                                                                   206
                                        }
                                                                                  'flavor': {
                                                                    207
                                   }
 147
                                                                                      'type': 'string'
                              }
                                                                                  },
 148
                                                                   209
                         }
                                                                                  'metadata': {
 149
                                                                   210
                    }
                                                                                       'type': 'dict',
 150
                                                                   211
               }
                                                                                       'schema': {
 151
                                                                   212
           }
                                                                                           'experiment': {
 152
                                                                    213
      }
                                                                                                 'type': 'string'
 153
                                                                    214
 154
                                                                    215
                                                                                       }
      nic = {
 155
                                                                    216
                                                                                 }
           'schema': {
 156
                                                                   217
                'name': {
 157
                                                                   218
                    'type': 'string'
 158
                                                                   219
               },
 159
                                                                   220
                'ip': {
                                                                        openstack_flavor = {
 160
                                                                   221
                    'type': 'string'
                                                                             'schema': {
                                                                   222
                                                                                  'os_flv_disabled': {
                                                                    223
                'mask': {
                                                                                     'type': 'string'
 163
                                                                   224
                    'type': 'string'
                                                                    225
 164
               },
                                                                                  'uuid': {
                                                                   226
 165
466
```

```
'type': 'string'
                                                                                   'public_ips': {
 227
                                                                    288
                },
                                                                                       'type': 'string'
 228
                                                                    289
                'os_flv_ext_data': {
                                                                                   },
 229
                                                                    290
                                                                                   'media_link': {
                    'type': 'string'
 230
                                                                    291
                },
                                                                                       'type': 'string'
 231
                                                                    292
                'ram': {
 232
                                                                    293
                    'type': 'string'
                                                                                   'key': {
 233
                                                                    294
                },
                                                                                       'type': 'string'
 234
                                                                    295
                'os_flavor_acces': {
                                                                                  },
 235
                                                                    296
                    'type': 'string'
                                                                                   'flavor': {
 236
                                                                    297
                },
                                                                                       'type': 'string'
 237
                                                                    298
 238
                'vcpus': {
                                                                    299
 239
                    'type': 'string'
                                                                    300
                                                                                   'resource_location': {
                },
                                                                                       'type': 'string'
 240
                                                                    301
                'swap': {
                                                                                   },
 241
                                                                    302
                    'type': 'string'
                                                                                   'instance_size': {
 242
                                                                    303
                },
                                                                                       'type': 'string'
 243
                                                                    304
                'rxtx_factor': {
                                                                                   },
 244
                                                                    305
                    'type': 'string'
                                                                                   'disk_name': {
 245
                                                                    306
                },
                                                                                       'type': 'string'
 246
                                                                    307
                'disk': {
 247
                                                                    308
                     'type': 'string'
                                                                                   'uuid': {
 248
                                                                    309
                                                                                       'type': 'string'
 249
                                                                    310
           }
 250
                                                                    311
      }
                                                                              }
 251
                                                                    312
 252
                                                                    313
      azure_vm = {
 253
                                                                    314
 254
           'schema': {
                                                                         azure_size = {
                                                                    315
                'username': {
                                                                              'schema': {
 255
                                                                    316
                    'type': 'string'
                                                                                   'ram': {
 256
                                                                    317
                },
                                                                                       'type': 'integer'
 257
                                                                    318
                                                                                  },
                'status': {
 258
                                                                    319
                     'type': 'string'
                                                                                   'name': {
 259
                                                                    320
 260
                },
                                                                    321
                                                                                       'type': 'string'
 261
                'deployment_slot': {
                                                                    322
                                                                                  },
 262
                    'type': 'string'
                                                                    323
                                                                                   'extra': {
                },
                                                                                        'type': 'dict',
 263
                                                                    324
                'group': {
                                                                                        'schema': {
 264
                                                                    325
                     'type': 'string'
                                                                                            'cores': {
 265
                                                                    326
                },
                                                                                                 'type': 'integer'
 266
                                                                    327
                'private_ips': {
 267
                                                                    328
                    'type': 'string'
                                                                                            'max_data_disks': {
                                                                    329
                                                                                                 'type': 'integer'
 269
                'cloud_service': {
 270
                                                                    331
                    'type': 'string'
                                                                                       }
 271
                                                                    332
                },
                                                                                  },
 272
                                                                    333
                'dns_name': {
                                                                                   'price': {
 273
                                                                    334
                    'type': 'string'
                                                                                       'type': 'float'
 274
                                                                    335
                },
                                                                                  },
 275
                                                                    336
                                                                                   '_uuid': {
                'image': {
 276
                                                                    337
                     'type': 'string'
                                                                                       'type': 'string'
 277
                                                                    338
                                                                                  },
 278
                                                                    339
                                                                                   'driver': {
                'floating_ip': {
 279
                                                                    340
                     'type': 'string'
                                                                                       'type': 'string'
 280
                                                                    341
                },
                                                                                  },
 281
                                                                    342
                'image_name': {
                                                                                   'bandwidth': {
 282
                                                                    343
 283
                     'type': 'string'
                                                                    344
                                                                                       'type': 'string'
 284
                                                                    345
                'instance_name': {
                                                                                   'disk': {
 285
                                                                    346
                     'type': 'string'
                                                                                       'type': 'integer'
                                                                    347
 286
                                                                                  },
                },
 287
                                                                    348
468
```

```
349
                                                                    410
                    'type': 'string'
                                                                                   'updated': {
 350
                                                                    411
                                                                                        'type': 'string'
 351
                                                                    412
           }
 352
                                                                    413
      }
                                                                                   'hostId': {
 353
                                                                    414
                                                                                        'type': 'string'
 354
                                                                    415
      openstack_vm = {
                                                                                   },
 355
                                                                    416
                                                                                   'floating_ip': {
           'schema': {
 356
                                                                    417
                'vm_state': {
                                                                                        'type': 'string'
 357
                                                                    418
                     'type': 'string'
 358
                                                                    419
                },
                                                                                   'static_ip': {
 359
                                                                    420
                                                                                        'type': 'string'
 360
                'availability_zone': {
                                                                    421
                                                                                   },
 361
                    'type': 'string'
                                                                    422
                                                                                   'key': {
 362
                                                                    423
                'terminated_at': {
                                                                                       'type': 'string'
 363
                                                                    424
                    'type': 'string'
 364
                                                                    425
                },
                                                                                   'flavor__id': {
 365
                                                                    426
                'image': {
                                                                                        'type': 'string'
 366
                                                                     427
                    'type': 'string'
 367
                                                                     428
                                                                                   'group': {
 368
                                                                     429
                'diskConfig': {
                                                                                        'type': 'string'
 369
                                                                    430
                     'type': 'string'
 370
                                                                    431
 371
                },
                                                                                   'task_state': {
                                                                    432
                'flavor': {
                                                                                        'type': 'string'
 372
                                                                    433
                     'type': 'string'
 373
                                                                    434
                                                                                   'created': {
 374
                                                                     435
                'security_groups': {
                                                                                        'type': 'string'
 375
                                                                     436
                    'type': 'string'
 377
                                                                                   'tenant_id': {
                                                                    438
                                                                                        'type': 'string'
                'volumes_attached': {
 378
                                                                    439
                    'type': 'string'
 379
                                                                    440
                },
                                                                                   'status': {
 380
                                                                    441
                'user_id': {
                                                                                        'type': 'string'
 381
                                                                    442
 382
                    'type': 'string'
                                                                    443
                                                                              }
 383
                                                                     444
                'uuid': {
 384
                                                                    445
                    'type': 'string'
 385
                                                                    446
                },
                                                                         cluster = {
                                                                    447
 386
                'accessIPv4': {
                                                                              'schema': {
 387
                                                                    448
                                                                                   'provider': {
                     'type': 'string'
 388
                                                                    449
                },
                                                                                        'type': 'list',
 389
                                                                    450
                'accessIPv6': {
 390
                                                                     451
                                                                                        'schema': {
                    'type': 'string'
                                                                                             'type': 'string'
                                                                     452
                },
                                                                                        }
 392
                                                                    453
                'power_state': {
                                                                                   },
 393
                                                                    454
                    'type': 'string'
                                                                                   'endpoint': {
 394
                                                                    455
                                                                                        'type': 'dict',
 395
                                                                    456
                                                                                        'schema': {
                'progress': {
 396
                                                                     457
                                                                                             'passwd': {
                     'type': 'string'
 397
                                                                     458
                                                                                                 'type': 'string'
 398
                'image__id': {
 399
                                                                     460
                     'type': 'string'
                                                                                             'url': {
 400
                                                                     461
                },
                                                                                                  'type': 'string'
 401
                                                                     462
                'launched_at': {
 402
                                                                    463
                     'type': 'string'
                                                                                        }
 403
                                                                     464
                                                                                   },
 404
                                                                     465
                                                                                   'name': {
 405
                'config_drive': {
                     'type': 'string'
                                                                     467
                                                                                        'type': 'string'
                },
                                                                                   },
 407
                                                                     468
                'username': {
                                                                                   'label': {
                                                                     469
 408
                     'type': 'string'
                                                                                        'type': 'string'
                                                                    470
 409
470
```

```
}
  471
                                                                     532
           }
                                                                                    'image_type': {
  472
                                                                     533
      }
                                                                                        'type': 'string'
  473
                                                                     534
  474
                                                                     535
                                                                                    'is_public': {
      computer = {
  475
                                                                     536
  476
           'schema': {
                                                                                        'type': 'string'
                                                                     537
                'ip': {
                                                                                   },
  477
                                                                     538
                                                                                    'owner_id': {
                     'type': 'string'
  478
                                                                     539
                },
                                                                                        'type': 'string'
  479
                                                                     540
                'name': {
  480
                                                                     541
                     'type': 'string'
                                                                                    'architecture': {
  481
  482
                                                                                        'type': 'string'
                'memoryGB': {
  483
                                                                                   },
                     'type': 'integer'
                                                                                    'virtualization_type': {
  484
                                                                     545
                },
                                                                                        'type': 'string'
  485
                                                                     546
                'label': {
                                                                                   },
  486
                                                                     547
                     'type': 'string'
                                                                                    'uuid': {
  487
                                                                     548
                                                                                        'type': 'string'
  488
                                                                     549
           }
  489
                                                                     550
                                                                              }
  490
                                                                     551
  491
                                                                     552
      libcloud_image = {
  492
                                                                     553
  493
           'schema': {
                                                                          user = {
                                                                     554
  494
                'username': {
                                                                               'schema': {
                                                                     555
                     'type': 'string'
                                                                                    'username': {
  495
                                                                     556
                                                                                        'type': 'string'
  496
                                                                     557
  497
                'status': {
                                                                     558
                     'type': 'string'
                                                                                    'context:': {
                                                                                        'type': 'string'
  499
                                                                     560
                                                                                   },
                'updated': {
  500
                                                                     561
                                                                                    'uuid': {
                     'type': 'string'
  501
                                                                     562
                },
                                                                                        'type': 'string'
  502
                                                                     563
                'description': {
  503
                                                                     564
  504
                     'type': 'string'
                                                                     565
                                                                                    'firstname': {
  505
                                                                     566
                                                                                        'type': 'string'
  506
                'owner_alias': {
                                                                     567
                                                                                   },
                     'type': 'string'
                                                                                    'lastname': {
  507
                                                                     568
                },
                                                                                        'type': 'string'
  508
                                                                     569
                                                                                   },
                'kernel_id': {
  509
                                                                     570
                     'type': 'string'
                                                                                    'roles': {
  510
                                                                     571
                                                                                        'type': 'list',
  511
                                                                     572
                'hypervisor': {
                                                                                        'schema': {
                     'type': 'string'
                                                                                             'type': 'string'
  514
                                                                     575
                'ramdisk_id': {
                                                                                   },
  515
                                                                     576
                     'type': 'string'
                                                                                    'email': {
  516
                                                                     577
                },
                                                                                        'type': 'string'
  517
                                                                     578
                'state': {
  518
                                                                     579
                     'type': 'string'
                                                                              }
  519
                                                                     580
  520
                                                                     581
                'created': {
  521
                                                                     582
                     'type': 'string'
                                                                          file = {
  522
                                                                     583
                },
                                                                               'schema': {
  523
                                                                     584
                'image_id': {
                                                                                    'endpoint': {
  524
                                                                     585
                     'type': 'string'
                                                                                         'type': 'string'
  525
                                                                     586
  526
                                                                     587
                                                                                    'name': {
  527
                'image_location': {
                     'type': 'string'
                                                                                        'type': 'string'
  529
                                                                     590
                'platform': {
                                                                                    'created': {
  530
                                                                     591
                     'type': 'string'
                                                                                        'type': 'string'
  531
                                                                     592
472
```

```
},
  593
                                                                    650
                'checksum': {
                                                                                                          }
                                                                    651
  594
                                                                                                     }
                     'type': 'dict',
  595
                                                                    652
                     'schema': {
                                                                                                 }
  596
                                                                    653
                          'md5': {
                                                                                            }
  597
                                                                    654
                                                                                       }
                               'type': 'string'
                                                                    655
  598
                                                                                  }
  599
                                                                    656
                                                                              }
                                                                    657
  600
                },
  601
                                                                    658
                'modified': {
  602
                                                                    659
                    'type': 'string'
                                                                         mapreduce = {
  603
                                                                    660
                },
  604
                                                                    661
                                                                              'schema': {
                'accessed': {
                                                                                   'layers': {
                     'type': 'string'
                                                                                        'type': 'list',
  606
                                                                    663
                },
                                                                                       'schema': {
  607
                                                                    664
                'size': {
                                                                                            'type': 'string'
                                                                    665
  608
                     'type': 'list',
  609
                                                                    666
                     'schema': {
                                                                                  },
  610
                                                                    667
                         'type': 'string'
                                                                                   'hdfs_datanode': {
  611
                                                                    668
                                                                                       'type': 'string'
  612
                                                                    669
                }
  613
           }
                                                                    671
                                                                                   'java': {
  614
  615
      }
                                                                    672
                                                                                        'type': 'dict',
                                                                                        'schema': {
  616
                                                                    673
      deployment = {
                                                                                            'platform': {
  617
                                                                    674
           'schema': {
                                                                                                 'type': 'string'
  618
                                                                    675
                'cluster': {
  619
                                                                    676
                     'type': 'list',
                                                                    677
                                                                                            'version': {
  621
                     'schema': {
                                                                    678
                                                                                                 'type': 'string'
                          'type': 'dict',
  622
                                                                    679
                          'schema': {
                                                                                       }
  623
                                                                    680
                              'id': {
                                                                                  },
  624
                                                                    681
                                                                                   'supervisord': {
                                   'type': 'string'
  625
                                                                    682
  626
                                                                    683
                                                                                       'type': 'string'
                          }
  627
                                                                    684
                     }
                                                                                   'yarn_historyserver': {
  628
                                                                    685
                },
                                                                                       'type': 'string'
  629
                                                                    686
                'stack': {
                                                                                   },
  630
                                                                    687
                     'type': 'dict',
                                                                                   'zookeeper': {
  631
                                                                    688
                     'schema': {
                                                                                       'type': 'string'
  632
                                                                    689
                          'layers': {
                                                                                   },
  633
                                                                    690
                               'type': 'list',
                                                                                   'hdfs_namenode': {
                                                                                       'type': 'string'
                               'schema': {
                                   'type': 'string'
  636
                                                                    693
  637
                                                                                   'hdfs_journalnode': {
                                                                    694
                          },
                                                                                       'type': 'string'
  638
                                                                    695
                          'parameters': {
  639
                                                                    696
                               'type': 'dict',
                                                                                   'yarn_resourcemanager': {
  640
                                                                    697
                               'schema': {
                                                                                       'type': 'string'
  641
                                                                    698
                                   'hadoop': {
  642
                                                                    699
                                        'type': 'dict',
                                                                              }
  643
                                                                    700
                                        'schema': {
  644
                                                                    701
  645
                                                                    702
       → 'zookeeper.quorum': {
                                                                         group = {
                                                                    703
                                                  'type':
                                                                              'schema': {
  646
                                                                    704
                                                                                   'users': {
       → 'list',
                                                                    705
                                                                                        'type': 'list',
  647
                                                  'schema':
                                                                    706
                                                                                        'schema': {
                                                                    707
                                                                                            'type': 'objectid',
  648
                                                                    708
       → 'type': 'string'
                                                                                            'data_relation': {
                                                                    709
                                                  }
                                                                                                 'resource': 'user',
                                                                    710
 649
474
```

```
'field': '_id',
                                                                                      'name': {
 711
                                                                      772
                                'embeddable': True
                                                                       773
 712
                          }
 713
                                                                       774
                     }
                                                                                }
  714
                                                                       775
                 },
 715
                                                                       776
                 'name': {
 716
                                                                       777
                     'type': 'string'
                                                                            virtual_cluster = {
 717
                                                                      778
                 },
                                                                                 'schema': {
                                                                      779
 718
                 'description': {
                                                                                      'nodes': {
 719
                                                                       780
                      'type': 'string'
  720
                                                                       781
                                                                                          'schema': {
  721
                                                                       782
           }
  722
                                                                       783
  723
       }
                                                                       784
 724
                                                                       785
      role = {
                                                                            725
            'schema': {
  726
                                                                       786
                 'users': {
  727
                                                                       787
                      'type': 'list',
                                                                                               }
  728
                                                                       788
                      'schema': {
                                                                                          }
  729
                                                                       789
                           'type': 'objectid',
                                                                                     },
  730
                                                                       790
                           'data_relation': {
                                                                                      'frontend': {
  731
                                                                       791
                                'resource': 'user',
  732
                                                                       792
  733
                                'field': '_id',
                                                                       793
                                'embeddable': True
  734
                                                                       794
                          }
  735
                                                                       795
                     }
  736
                                                                       796
                 },
  737
                                                                       797
                 'name': {
                                                                                     },
                     'type': 'string'
                                                                                      'name': {
  739
                                                                       799
                },
 740
                                                                       800
                 'description': {
 741
                                                                       801
                      'type': 'string'
                                                                                }
 742
                                                                       802
                }
                                                                           }
  743
                                                                       803
           }
  744
                                                                       804
  745
       }
                                                                       805
                                                                           libcloud_flavor = {
  746
                                                                       806
                                                                                 'schema': {
       virtual_directory = {
                                                                                      'uuid': {
 747
                                                                       807
            'schema': {
 748
                                                                       808
                 'endpoint': {
                                                                                     },
 749
                                                                       809
                     'type': 'string'
                                                                                      'price': {
  750
                                                                       810
                },
  751
                                                                       811
                                                                                     },
  752
                 'protocol': {
                                                                                      'ram': {
                     'type': 'string'
                },
  754
                                                                       814
                 'name': {
  755
                                                                       815
                     'type': 'string'
                                                                                      'bandwidth': {
  756
                                                                       816
                 },
  757
                                                                       817
                                                                                     },
                 'collection': {
  758
                                                                       818
                     'type': 'list',
                                                                                      'flavor_id': {
  759
                                                                       819
                      'schema': {
  760
                                                                       820
                           'type': 'string'
  761
                                                                       821
                                                                                      'disk': {
                                                                       822
  762
                }
                                                                       823
  763
            }
  764
                                                                       824
                                                                                      'cpu': {
  765
                                                                       825
  766
                                                                       826
       file_alias = {
  767
                                                                       827
            'schema': {
                                                                                }
                                                                       828
                 'alias': {
 769
                                                                       829
                     'type': 'string'
 770
                                                                       830
                },
                                                                           batchjob = {
                                                                      831
 771
476
```

```
'type': 'string'
'type': 'list',
    'type': 'objectid',
    'data_relation': {
        'resource':
        'field': '_id',
         'embeddable': True
'type': 'objectid',
'data_relation': {
    'resource': 'virtual_machine',
    'field': '_id',
    'embeddable': True
'type': 'string'
```

```
'schema': {
                                                                                    'name': {
  832
                                                                     893
                'output_file': {
                                                                                        'type': 'string'
  833
                                                                     894
                     'type': 'string'
                                                                                   },
  834
                                                                     895
                                                                                    'memoryGB': {
  835
                                                                     896
                'group': {
                                                                                        'type': 'integer'
  836
                                                                     897
                     'type': 'string'
  837
                                                                     898
                                                                                    'label': {
                },
  838
                                                                     899
                'job_id': {
                                                                                        'type': 'string'
  839
                                                                     900
                    'type': 'string'
  840
                                                                     901
                },
                                                                              }
  841
                                                                     902
                'script': {
  842
                                                                     903
  843
                     'type': 'string'
                                                                     904
                },
  844
                                                                     905
                                                                          sshkey = {
                'cmd': {
                                                                               'schema': {
  845
                                                                     906
                     'type': 'string'
                                                                                   'comment': {
  846
                                                                     907
                                                                                        'type': 'string'
  847
                                                                     908
                'queue': {
                                                                                   },
  848
                                                                     909
                                                                                    'source': {
                     'type': 'string'
  849
                                                                     910
                                                                                        'type': 'string'
  850
                                                                     911
                'cluster': {
                                                                                   },
  851
                                                                     912
                     'type': 'string'
                                                                                    'uri': {
  852
                                                                     913
                },
                                                                                        'type': 'string'
  853
                                                                     914
  854
                'time': {
                                                                     915
                                                                                    'value': {
  855
                     'type': 'string'
                                                                     916
                },
                                                                                        'type': 'string'
  856
                                                                     917
                'path': {
  857
                                                                     918
                    'type': 'string'
  858
                                                                     919
                                                                                    'fingerprint': {
                },
                                                                     920
                                                                                        'type': 'string'
                'nodes': {
  860
                                                                     921
                     'type': 'string'
                                                                              }
                                                                     922
  861
                },
                                                                         }
                                                                     923
  862
                'dir': {
  863
                                                                     924
                     'type': 'string'
                                                                          stream = {
  864
                                                                     925
                }
                                                                               'schema': {
  865
                                                                     926
  866
           }
                                                                     927
                                                                                    'attributes': {
  867
                                                                     928
                                                                                        'type': 'dict',
                                                                                         'schema': {
  868
                                                                     929
       organization = {
                                                                                             'rate': {
  869
                                                                     930
            'schema': {
                                                                                                  'type': 'integer'
  870
                                                                     931
                'users': {
                                                                                             },
  871
                                                                     932
                     'type': 'list',
                                                                                             'limit': {
  872
                                                                     933
                     'schema': {
  873
                                                                     934
                                                                                                  'type': 'integer'
                          'type': 'objectid',
                          'data_relation': {
                                                                                        }
  875
                                                                     936
  876
                               'resource': 'user',
                                                                                   },
                                                                     937
                               'field': '_id',
                                                                                    'name': {
  877
                                                                     938
                               'embeddable': True
                                                                                        'type': 'string'
  878
                                                                     939
                                                                                   },
                          }
  879
                                                                     940
                     }
                                                                                   'format': {
  880
                                                                     941
                }
                                                                                        'type': 'string'
  881
                                                                     942
           }
  882
                                                                     943
                                                                              }
  883
                                                                     944
  884
                                                                     945
       container = {
  885
                                                                     946
           'schema': {
                                                                          database = {
  886
                                                                     947
                'ip': {
                                                                               'schema': {
  887
                                                                     948
                                                                                   'endpoint': {
  888
                     'type': 'string'
                                                                     949
  889
                                                                     950
                                                                                        'type': 'string'
                'endpoint': {
                                                                                   },
  890
                                                                     951
                     'type': 'string'
                                                                                   'protocol': {
  891
                                                                     952
                },
                                                                                        'type': 'string'
                                                                     953
  892
478
```

```
},
 954
                                                                    1015
                'name': {
                                                                                             'instance_type_memory_mb': {
 955
                                                                    1016
                     'type': 'string'
                                                                                                  'type': 'string'
 956
                                                                    1017
                                                                                             },
 957
                                                                    1018
           }
                                                                                             'user_id': {
 958
                                                                    1019
      }
                                                                                                  'type': 'string'
 959
                                                                    1020
                                                                                             },
 960
                                                                    1021
      default = {
                                                                                             'description': {
 961
                                                                    1022
           'schema': {
                                                                                                  'type': 'string'
 962
                                                                    1023
                'context': {
 963
                                                                    1024
                     'type': 'string'
                                                                                             'kernel_id': {
 964
                                                                    1025
 965
                },
                                                                    1026
                                                                                                 'type': 'string'
 966
                'name': {
                                                                    1027
                                                                                             },
                     'type': 'string'
                                                                                             'instance_type_name': {
 967
                                                                    1028
                },
                                                                                                 'type': 'string'
 968
                                                                    1029
                'value': {
 969
                                                                    1030
                    'type': 'string'
                                                                                             'ramdisk_id': {
 970
                                                                    1031
                                                                                                  'type': 'string'
 971
                                                                    1032
           }
 972
                                                                    1033
      }
                                                                                             'instance_type_id': {
 973
                                                                    1034
                                                                                                 'type': 'string'
 974
                                                                    1035
 975
      openstack_image = {
                                                                    1036
 976
           'schema': {
                                                                    1037
                                                                                             'instance_type_ephemeral_gb':
 977
                'status': {

→ {

                    'type': 'string'
                                                                                                 'type': 'string'
 978
                                                                    1038
                },
 979
                                                                    1039
                'username': {
 980
                                                                    1040
                                                                                             'instance_type_rxtx_factor': {
                     'type': 'string'
                                                                                                  'type': 'string'
                                                                    1041
 982
                                                                    1042
                'updated': {
 983
                                                                                             'image_type': {
                                                                    1043
                     'type': 'string'
                                                                                                 'type': 'string'
 984
                                                                    1044
                },
 985
                                                                    1045
                'uuid': {
                                                                                             'network_allocated': {
 986
                                                                    1046
 987
                     'type': 'string'
                                                                    1047
                                                                                                  'type': 'string'
 988
                                                                    1048
 989
                'created': {
                                                                    1049
                                                                                             'instance_type_flavorid': {
                     'type': 'string'
                                                                                                  'type': 'string'
 990
                                                                    1050
                },
                                                                                             },
 991
                                                                    1051
                'minDisk': {
                                                                                             'instance_type_vcpus': {
 992
                                                                    1052
                     'type': 'string'
                                                                                                  'type': 'string'
 993
                                                                    1053
                },
 994
                                                                    1054
 995
                'progress': {
                                                                    1055
                                                                                             'instance_type_root_gb': {
                     'type': 'string'
                                                                                                 'type': 'string'
                },
                                                                                             },
 997
                                                                    1057
                'minRam': {
                                                                                             'base_image_ref': {
 998
                                                                    1058
                                                                                                  'type': 'string'
                    'type': 'string'
 999
                                                                    1059
                },
 1000
                                                                    1060
                'os_image_size': {
                                                                                             'instance_type_swap': {
 1001
                                                                    1061
                                                                                                  'type': 'string'
                     'type': 'string'
 1002
                                                                    1062
                },
                                                                                             },
 1003
                                                                    1063
                'metadata': {
                                                                                             'owner_id': {
 1004
                                                                    1064
                     'type': 'dict',
                                                                                                  'type': 'string'
 1005
                                                                    1065
                     'schema': {
 1006
                                                                    1066
                          'instance_uuid': {
                                                                                        }
 1007
                                                                    1067
                                                                                   }
                               'type': 'string'
 1008
                                                                    1068
                                                                              }
 1009
                                                                    1069
                          'image_location': {
                                                                    1070
 1010
                              'type': 'string'
                                                                    1071
                          },
                                                                          azure_image = {
 1012
                                                                    1072
                          'image_state': {
                                                                               'schema': {
 1013
                                                                    1073
                               'type': 'string'
                                                                                   '_uuid': {
                                                                    1074
 1014
480
```

```
'type': 'string'
                                                                                                         },
 1075
                                                                     1136
                },
                                                                                                         'zookeeper': {
 1076
                                                                     1137
                 'driver': {
                                                                                                              'type': 'string'
 1077
                                                                     1138
                     'type': 'string'
 1078
                                                                     1139
                },
                                                                                                         'supervisord': {
 1079
                                                                     1140
                 'id': {
                                                                                                              'type': 'string'
 1080
                                                                     1141
                     'type': 'string'
                                                                     1142
 1081
                },
                                                                                                   }
 1082
                                                                     1143
                 'name': {
                                                                                              }
 1083
                                                                     1144
                     'type': 'string'
                                                                                          }
 1084
                                                                     1145
                },
                                                                                     },
 1085
                                                                     1146
                 'extra': {
 1086
                                                                     1147
                                                                                     'parameters': {
 1087
                     'type': 'dict',
                                                                     1148
                                                                                          'type': 'dict',
                     'schema': {
                                                                                          'schema': {
 1088
                                                                     1149
                          'category': {
                                                                                               'num_resourcemanagers': {
 1089
                                                                     1150
                               'type': 'string'
                                                                                                    'type': 'integer'
 1090
                                                                     1151
                          },
                                                                                              },
 1091
                                                                     1152
                          'description': {
                                                                                               'num_namenodes': {
 1092
                                                                     1153
                               'type': 'string'
                                                                                                   'type': 'integer'
 1093
                                                                     1154
 1094
                                                                     1155
                          'vm_image': {
                                                                                               'use_yarn': {
 1095
                                                                     1156
                               'type': 'string'
                                                                                                    'type': 'boolean'
                                                                     1157
 1096
                          },
                                                                     1158
 1097
                           'location': {
                                                                                               'num_datanodes': {
 1098
                                                                     1159
                                                                                                   'type': 'integer'
                               'type': 'string'
 1099
                                                                     1160
 1100
                                                                     1161
 1101
                          'affinity_group': {
                                                                     1162
                                                                                               'use_hdfs': {
                               'type': 'string'
                                                                                                   'type': 'boolean'
 1102
                                                                     1163
 1103
                                                                     1164
                          'os': {
                                                                                               'num_historyservers': {
                                                                     1165
 1104
                               'type': 'string'
                                                                                                   'type': 'integer'
 1105
                                                                     1166
                          },
                                                                                               },
 1106
                                                                     1167
                          'media_link': {
                                                                                               'num_journalnodes': {
 1107
                                                                     1168
 1108
                               'type': 'string'
                                                                     1169
                                                                                                    'type': 'integer'
 1109
                                                                     1170
                     }
                                                                                          }
 1110
                                                                     1171
                }
                                                                                    }
 1111
                                                                     1172
           }
                                                                                }
 1112
                                                                     1173
 1113
                                                                     1174
 1114
                                                                     1175
                                                                           compute_resource = {
      hadoop = {
 1115
                                                                     1176
           'schema': {
                                                                                'schema': {
                                                                     1177
                 'deployers': {
                                                                                     'kind': {
                      'type': 'dict',
                                                                                          'type': 'string'
                                                                     1179
 1118
                      'schema': {
 1119
                                                                     1180
                          'ansible': {
                                                                                     'endpoint': {
 1120
                                                                     1181
                               'type': 'string'
                                                                                          'type': 'string'
 1121
                                                                     1182
                                                                                    },
 1122
                                                                     1183
                     }
                                                                                     'name': {
 1123
                                                                     1184
                },
                                                                                         'type': 'string'
 1124
                                                                     1185
                 'requires': {
 1125
                                                                     1186
                                                                                }
                      'type': 'dict',
 1126
                                                                     1187
                      'schema': {
                                                                     1188
 1127
                          'java': {
 1128
                                                                     1189
                               'type': 'dict',
                                                                           node_new = {
 1129
                                                                     1190
                                'schema': {
                                                                                'schema': {
 1130
                                                                     1191
 1131
                                    'implementation': {
                                                                     1192
                                                                                     'authorized_keys': {
 1132
                                         'type': 'string'
                                                                     1193
                                                                                          'type': 'list',
                                    },
                                                                                         'schema': {
 1133
                                                                     1194
                                    'version': {
                                                                                               'type': 'string'
 1134
                                                                     1195
                                         'type': 'string'
                                                                                          }
 1135
                                                                     1196
482
```

```
'schema': {
 1197
                                                                     1258
                 'name': {
                                                                                                              'type': 'integer'
 1198
                                                                     1259
                     'type': 'string'
                                                                                                         }
 1199
                                                                     1260
                },
                                                                                                    },
 1200
                                                                     1261
                 'external_ip': {
                                                                                                    'protocols': {
 1201
                                                                     1262
                                                                                                         'type': 'list',
                     'type': 'string'
 1202
                                                                     1263
                },
                                                                                                         'schema': {
 1203
                                                                     1264
                 'memory': {
                                                                                                              'type': 'string'
 1204
                                                                     1265
                     'type': 'integer'
 1205
                                                                     1266
                },
                                                                                                    }
 1206
                                                                     1267
                                                                                               }
                 'create_external_ip': {
 1207
                                                                     1268
                                                                                          }
 1208
                     'type': 'boolean'
                                                                     1269
                },
                                                                                     },
 1209
                                                                     1270
                 'internal_ip': {
                                                                                     'users': {
 1210
                                                                     1271
                     'type': 'string'
                                                                                          'type': 'dict',
 1211
                                                                     1272
                                                                                          'schema': {
 1212
                                                                     1273
                 'loginuser': {
                                                                                               'name': {
 1213
                                                                     1274
                     'type': 'string'
                                                                                                    'type': 'string'
 1214
                                                                     1275
                                                                                               },
 1215
                                                                     1276
                 'owner': {
                                                                                               'groups': {
 1216
                     'type': 'string'
                                                                                                    'type': 'list',
 1217
                },
                                                                                                    'schema': {
 1218
                                                                     1279
                 'cores': {
                                                                                                         'type': 'string'
 1219
                                                                     1280
                     'type': 'integer'
 1220
                                                                     1281
                },
                                                                                               }
 1221
                                                                     1282
                 'disk': {
                                                                                          }
 1222
                                                                     1283
                     'type': 'integer'
                                                                                     }
 1223
                                                                     1284
                                                                                }
                },
                                                                     1285
                 'ssh_keys': {
 1225
                                                                     1286
                      'type': 'list',
                                                                     1287
 1226
                      'schema': {
                                                                           filter = {
 1227
                                                                     1288
                           'type': 'dict',
                                                                                'schema': {
 1228
                                                                     1289
                           'schema': {
                                                                                     'function': {
 1229
                                                                     1290
 1230
                               'from': {
                                                                     1291
                                                                                          'type': 'string'
 1231
                                    'type': 'string'
                                                                     1292
 1232
                                                                     1293
                                                                                     'name': {
                               'decrypt': {
                                                                                          'type': 'string'
 1233
                                                                     1294
                                    'type': 'string'
 1234
                                                                     1295
                                                                                }
                               },
 1235
                                                                     1296
                                'ssh_keygen': {
 1236
                                                                     1297
                                    'type': 'boolean'
 1237
                                                                     1298
                               },
                                                                           reservation = {
                               'to': {
                                                                                'schema': {
                                    'type': 'string'
                                                                                     'start_time': {
 1240
                                                                     1301
                                                                                          'type': 'list',
 1241
                                                                     1302
                          }
                                                                                          'schema': {
 1242
                                                                     1303
                     }
                                                                                               'type': 'string'
 1243
                                                                     1304
                },
 1244
                                                                     1305
                                                                                     },
                 'security_groups': {
 1245
                                                                     1306
                     'type': 'list',
                                                                                     'hosts': {
                      'schema': {
                                                                                          'type': 'string'
 1247
                                                                     1308
                           'type': 'dict',
 1248
                                                                     1309
                                                                                     'description': {
                           'schema': {
 1249
                                                                     1310
                               'ingress': {
                                                                                          'type': 'string'
 1250
                                                                     1311
                                                                                     },
                                    'type': 'string'
 1251
                                                                     1312
                                                                                     'end_time': {
 1252
                                                                     1313
 1253
                               'egress': {
                                                                     1314
                                                                                          'type': 'list',
                                    'type': 'string'
                                                                     1315
                                                                                          'schema': {
                               },
                                                                                               'type': 'string'
 1255
                                                                     1316
                               'ports': {
                                                                     1317
 1256
                                    'type': 'list',
                                                                                     }
 1257
                                                                     1318
484
```

```
1319
                                                                     1380
      }
                                                                          mesos_docker = {
 1320
                                                                     1381
                                                                                'schema': {
 1321
                                                                     1382
                                                                                    'container': {
      replica = {
 1322
                                                                     1383
           'schema': {
                                                                                         'type': 'dict',
 1323
                                                                     1384
                'endpoint': {
                                                                                         'schema': {
 1324
                                                                     1385
                     'type': 'string'
                                                                                              'docker': {
 1325
                                                                     1386
                                                                                                   'type': 'dict',
 1326
                                                                     1387
                 'name': {
                                                                                                   'schema': {
 1327
                                                                     1388
                     'type': 'string'
                                                                                                        'credential': {
 1328
                                                                     1389
                },
                                                                                                             'type': 'dict',
 1329
                                                                     1390
 1330
                 'checksum': {
                                                                     1391
                                                                                                             'schema': {
 1331
                     'type': 'dict',
                                                                     1392
                                                                                                                  'secret': {
                     'schema': {
                                                                                                                       'type':
 1332
                                                                     1393
                         'md5': {
                                                                           → 'string'
 1333
                               'type': 'string'
 1334
                                                                     1394
                                                                                                                  'principal': {
 1335
                                                                     1395
                     }
                                                                                                                       'type':
 1336
                                                                     1396
                },
 1337
                                                                              'string'
                 'replica': {
                                                                     1397
 1338
                    'type': 'string'
                                                                                                             }
 1339
                                                                     1398
                                                                                                        },
 1340
                                                                     1399
                 'accessed': {
                                                                                                        'image': {
 1341
                                                                     1400
                     'type': 'string'
                                                                                                             'type': 'string'
 1342
                                                                     1401
 1343
                                                                     1402
                'size': {
                                                                                                   }
 1344
                                                                     1403
                     'type': 'list',
 1345
                                                                     1404
                                                                                              },
                                                                                              'type': {
                     'schema': {
                         'type': 'string'
                                                                                                  'type': 'string'
 1347
                                                                     1406
 1348
                                                                     1407
                }
                                                                                         }
 1349
                                                                     1408
                                                                                    },
           }
 1350
                                                                     1409
      }
                                                                                     'mem': {
 1351
                                                                     1410
                                                                                       'type': 'float'
 1352
                                                                     1411
 1353
      microservice = {
                                                                     1412
           'schema': {
 1354
                                                                     1413
                                                                                     'args': {
                'function': {
                                                                                         'type': 'list',
 1355
                                                                     1414
                     'type': 'string'
                                                                                         'schema': {
                                                                     1415
 1356
                                                                                              'type': 'string'
 1357
                                                                    1416
                 'endpoint': {
 1358
                                                                     1417
                     'type': 'string'
 1359
                                                                     1418
                                                                                     'cpus': {
                },
                 'name': {
                                                                                        'type': 'float'
                    'type': 'string'
                                                                     1421
 1362
                                                                                     'instances': {
                                                                     1422
 1363
                                                                                        'type': 'integer'
           }
 1364
                                                                     1423
                                                                                    },
 1365
                                                                     1424
                                                                                     'id': {
 1366
                                                                     1425
      var = {
                                                                                        'type': 'string'
 1367
                                                                     1426
           'schema': {
 1368
                                                                     1427
                                                                               }
                 'type': {
                                                                     1428
 1369
                     'type': 'string'
 1370
                                                                     1429
                },
 1371
                                                                     1430
                 'name': {
                                                                          libcloud_vm = {
                                                                     1431
 1372
                                                                               'schema': {
                    'type': 'string'
 1373
                                                                     1432
                                                                                    'username': {
 1374
                                                                    1433
                                                                                        'type': 'string'
 1375
                 'value': {
                                                                     1434
 1376
                     'type': 'string'
                                                                     1435
                                                                                     'status': {
 1377
                                                                     1436
           }
                                                                                        'type': 'string'
 1378
                                                                     1437
                                                                                    },
      }
 1379
                                                                     1438
486
```

```
'root_device_type': {
 1439
                      'type': 'string'
 1440
                 },
 1441
                 'private_ips': {
 1442
                      'type': 'string'
 1443
                 },
 1444
                 'instance_type': {
 1445
                      'type': 'string'
 1446
                 },
 1447
                 'image': {
 1448
                      'type': 'string'
 1449
                 },
 1450
 1451
                 'private_dns': {
                      'type': 'string'
 1452
                 },
 1453
                 'image_name': {
 1454
                      'type': 'string'
 1455
                 },
 1456
                 'instance_id': {
 1457
                      'type': 'string'
                 },
 145
                 'image_id': {
 1460
                      'type': 'string'
 1461
                 },
 1462
                 'public_ips': {
 1463
 1464
                      'type': 'string'
                 },
 1465
                 'state': {
 1466
                      'type': 'string'
 1467
                 },
 1468
                 'root_device_name': {
 1469
                      'type': 'string'
 1470
                 },
 1471
 1472
                 'key': {
 1473
                      'type': 'string'
 1474
                 },
                 'group': {
 1475
                      'type': 'string'
 1476
                 },
 1477
                 'flavor': {
 1478
 1479
                      'type': 'string'
                 },
 1480
                 'availability': {
                      'type': 'string'
 1482
                 },
 1483
                 'uuid': {
 1484
                      'type': 'string'
 1485
 1486
            }
 1487
       }
 1489
 1490
 1491
       eve_settings = {
 1492
            'MONGO_HOST': 'localhost',
 1493
            'MONGO_DBNAME': 'testing',
 1494
            'RESOURCE_METHODS': ['GET', 'POST',
 1495
           'DELETE'],
            'BANDWIDTH_SAVER': False,
 1496
            'DOMAIN': {
 1497
                 'profile': profile,
 1498
488
```

```
'virtual_machine': virtual_machine,
 1499
               'kubernetes': kubernetes,
 1500
               'nic': nic,
 1501
               'virtual_compute_node':
 1502
           virtual_compute_node,
               'openstack_flavor': openstack_flavor,
 1503
                'azure-vm': azure_vm,
 1504
               'azure-size': azure_size,
 1505
               'openstack_vm': openstack_vm,
 1506
               'cluster': cluster,
 1507
               'computer': computer,
 1508
               'libcloud_image': libcloud_image,
 1509
 1510
               'user': user,
               'file': file,
 1511
               'deployment': deployment,
 1512
               'mapreduce': mapreduce,
 1513
               'group': group,
 1514
               'role': role,
 1515
               'virtual_directory':
 1516
           virtual_directory,
               'file_alias': file_alias,
 1517
               'virtual_cluster': virtual_cluster,
 1518
               'libcloud_flavor': libcloud_flavor,
 1519
               'batchjob': batchjob,
 1520
               'organization': organization,
 1521
 1522
               'container': container,
 1523
               'sshkey': sshkey,
               'stream': stream,
 1524
               'database': database,
 1525
               'default': default,
 1526
               'openstack_image': openstack_image,
 1527
               'azure_image': azure_image,
 1528
               'hadoop': hadoop,
 1529
 1530
               'compute_resource': compute_resource,
 1531
               'node_new': node_new,
 1532
               'filter': filter,
               'reservation': reservation,
 1533
                'replica': replica,
 1534
                'microservice': microservice,
 1535
               'var': var,
 1536
               'mesos-docker': mesos_docker,
 1537
               'libcloud_vm': libcloud_vm,
 1538
           },
      }
 1540
489
```

C. CONTRIBUTING

492

493

494

495

496

497

498

499

500

We invite you to contribute to this paper and its discussion to improve it. Improvements can be done with pull requests. We suggest you do *small* individual changes to a single section and object rather than large changes as this allows us to integrate the changes individually and comment on your contribution via github.

Once contributed we will appropriately acknoledge you either as contributor or author. Please discuss with us how we best acknowledge you.

D. USING THE CLOUDMESH REST SERVICE

Components are written as YAML markup in files in the resources/samples directory.

For example: 503

```
Listing D.1: profile
        "profile": {
           "description": "The Profile of a user",
           "uuid": "jshdjkdh...",
           "context:": "resource",
           "email": "laszewski@gmail.com",
           "firstname": "Gregor",
           "lastname": "von Laszewski",
                                                              513
           "username": "gregor"
                                                              514
  10
      }
  1
                                                              516
504
                                                              517
```

D.1. Element Definition

Each resource should have a description entry to act as documentation. The documentation should be formated as reStructuredText. For example:

D.2. Yaml

505

506

507

```
entry = yaml.read(''')
                                                       522
profile:
  description: |
                                                       523
    A user profile that specifies general information
    about the user
  email: laszewski@gmail.com, required
  firtsname: Gregor, required
                                                       526
  lastname: von Laszewski, required
  height: 180
                                                       527
,,,}
                                                       528
```

D.3. Cerberus

```
schema = {
'profile': {
  'description': {'type': 'string'}
                 {'type': 'string', 'required': True}
  'email':
  'firtsname':
                 {'type': 'string', 'required': True}
                 {'type': 'string', 'required': True5}3
  'lastname':
                 {'type': 'float'}
  'height':
```

E. MONGOENGINE

```
class profile(Document):
    description = StringField()
    email = EmailField(required=True)
    firstname = StringField(required=True)
    lastname = StringField(required=True)
    height = FloatField(max_length=50)
```

F. CLOUDMESH NOTATION

```
profile:
    description: string
    email: email, required
    firstname: string, required
    lastname: string, required
    height: flat, max=10
```

```
proposed command
```

```
cms schema FILENAME --format=mongo -o OUTPUT
cms schema FILENAME --format=cerberus -o OUTPUT
cms schema FILENAME --format=yaml -o OUTPUT
```

reads FILENAME in cloudmesh notation and returns format

```
cms schema FILENAME --input=evegenie -o OUTPUT
  reads eavegene example and create settings for eve
```

F.1. Defining Elements for the REST Service

To manage a large number of elements defined in our REST service easily, we manage them trhough definitions in yaml files. To generate the appropriate settings file for the rest service, we can use teh following command:

```
cms admin elements <directory> <out.json>
```

where

520 521

529

536

537

539

540

541

- <directory>: directory where the YAML definitions re-
- <out.json>: path to the combined definition

For example, to generate a file called all.json that integrates all yml objects defined in the directory resources/samples you can use the following command:

```
cms elements resources/samples all.json
```

F.2. DOIT

cms schema spec2tex resources/specification resources/tex

F.3. Generating service

With evegenie installed, the generated JSON file from the above step is processed to create the stub REST service definitions.

G. ABC

README.rst

H. CLOUDMESH REST

H.1. Prerequistis

- · mongo instalation
- eve instalation
- cloudmesh cmd5
- · cloudmesh rest

H.1.1. Install Mongo on OSX

```
brew update
542
    brew install mongodb
543
544
    # brew install mongodb --with-openssl
545
```

H.1.2. Install Mongo on OSX

ASSIGNMET TO STUDENTS, PROVIDE PULL REQUEST 547 WITH INSTRUCTIONS

612

615

617

618

619

620

630

633

634

635

636

637

638

642

643

649

H.2. Introduction

551

552

553

554

559

560

561

562

563

564

573

578

579

580

583

584

585

586

587

588

589

590

593

594

598

600

With the cloudmesh REST framework it is easy to create REST services while defining the resources in the service easily with examples. The service is based on eve and the examples are defined in yml to be converted to json and from json with 605 evegenie into a valid eve settings file.

Thus oyou can eother wite your examples in yaml or in json. The resources are individually specified in a directory. The directory can contain multiple resource files. We recomment that for each resource you define your own file. Conversion of the specifications can be achieved with the schema 610 command.

H.3. Yaml Specification

Let us first introduce you to a yaml specification. Let us assume that your yaml file is called profile.yaml and located in a directory called 'example':

```
profile:
565
      description: The Profile of a user
566
      email: laszewski@gmail.com
567
      firstname: Gregor
568
      lastname: von Laszewski
569
      username: gregor
570
```

As eve takes json objects as default we need to convert it first to json. This is achieved wih:

```
cd example
cms schema convert profile.yml profile.json
```

This will provide the json file profile.json as Listed in the next section

H.4. Json Specification

A valid json resource specification looks like this:

```
{
  "profile": {
    "description": "The Profile of a user",
    "email": "laszewski@gmail.com",
    "firstname": "Gregor",
    "lastname": "von Laszewski",
    "username": "gregor"
  }
}
```

H.5. Conversion to Eve Settings

The json files in the ~/sample directory need now to be converted to a valid eve schema. This is achieved with tow commands. First, we must concatenate all json specified resource examples into a single json file. We do this with:

```
cms schema cat . all.json
```

As we assume you are in the samples directory, we use a 644 . for the current location of the directory that containes the samples. Next, we need to convert it to the settings file. THis can be achieved with the convert program when you specify a json file:

```
cms schema convert all.json
```

THe result will be a eve configuration file that you can use to start an eve service. The file is called all.settings.py

H.5.1. Managing Mongo

Next you need to start the mongo service with

```
cms admin mongo start
```

You can look at the status and information about the service with:

```
cms admin mongo info
cms admin mongo status
```

If you need to stop the service you can use:

cms admin mongo stop

H.5.2. Manageing Eve

Now it is time to start the REST service. THis is done in a separate window with the following commands:

```
cms admin settings all.settings.json
cms admin rest start
```

The first command coppies the settings file to

```
~/cloudmesh/eve/settings.py
```

This file is than used by the start action to start the eve service. Please make sure that you execute this command in a separate window, as for debugging purposses you will be able to monitor this way interactions with this service

```
Testing - OLD ~
```

```
# install mongo and eve
make setup
make install
             # installs the code and integrates it into cmd5
make deploy
make test
```

classes lessons rest.rst

I. REST WITH EVE

I.1. Overview of REST

REST stands for REpresentational State Transfer. REST is an architecture style for designing networked applications. It is based on stateless, client-server, cacheable communications protocol. Although not based on http, in most cases, the HTTP protocol is used. In contrast to what some others write or say, REST is not a standard.

RESTful applications use HTTP requests to:

- post data: while creating and/or updating it,
- read data: while making queries, and
- · delete data.

Hence REST uses HTTP for the four CRUD operations:

- Create
- Read
- Update
- Delete

As part of the HTTP protocol we have methods such as GET, PUT, POST, and DELETE. These methods can than be used to implement a REST service. As REST introduces collections and items we need to implement the CRUD functions for them. The semantics is explained in the Table illustrationg how to implement them with HTTP methods.

Source: https://en.wikipedia.org/wiki/Representational state transfei

713

714

715

716

717

720

721

733

734

735

739

740

741

755

756

I.2. REST and eve

653

655

656

657

658

662

665

666

668

669

671

675

676

677

678

680

681

683

684

685

689

691

692

693

694

696

699

700

701

702

703

Now that we have outlined the basic functionality that we need, we lke to introduce you to Eve that makes this process rather trivial. We will provide you with an implementation example that showcases that we can create REST services without writing a single line of code. The code for this is located at https://github.com/cloudmesh/rest

This code will have a master branch but will also have a dev branch in which we will add gradually more objects. Objects in the dev branch will include:

- · virtual directories
- virtual clusters
- job sequences
- inventories

;You may want to check our active development work in the dev branch. However for the purpose of this class the 719 master branch will be sufficient.

I.2.1. Installation

First we havt to install mongodb. The instalation will depend on your operating system. For the use of the rest service it 723 is not important to integrate mongodb into the system upon 724 reboot, which is focus of many online documents. However, for us it is better if we can start and stop the services explicitly for now.

On ubuntu, you need to do the following steps:

On windows 10, you need to do the following steps:

elect Windows 10. YOu could be using the online documentation

TO BE CONTRIBUTED BY THE STUDENTS OF THE CLASS as homeworkevegenion

provided by starting it on Windows, or rinning it in 732a

On OSX you can use homebrew and install it with:

brew update

brew install mongodb

In future we may want to add ssl authentication in which case

need to install it as follows:

brew install mongodb -with-openssl

I.2.2. Starting the service

We have provided a convenient Makefile that currently only works for OSX. It will be easy for you to adapt it to Linux. Certainly you can look at the targes in the makefile and replicate them one by one. Improtaht targest are deploy and test.

When using the makefile you can start the services with:

make deploy

IT will start two terminals. IN one you will see the mongo service, in the other you will see the eve service. The eve service will take a file called sample.settings.py that is base on sample json for the start of the eve service. The mongo servide is configured in suc a wahy that it only accepts incimming connections from the local host which will be suffiicent fpr our case. The mongo data is written into the \$USER/.cloudmesh directory, so make sure it exists.

To test the services you can say:

YOu will se a number of json text been written to the screen.

I.3. Creating your own objects

The example demonstrated how easy it is to create a mongodb and an eve rest service. Now lets use this example to creat your own. FOr this we have modified a tool called evegenie to install it onto your system.

The original documentation for evegenie is located at:

http://evegenie.readthedocs.io/en/latest/

However, we have improved evegenie while providing a commandline tool based on it. The improved code is located at:

· https://github.com/cloudmesh/evegenie

You clone it and install on your system as follows:

```
cd ~/github
git clone https://github.com/cloudmesh/evegenie
cd evegenie
python setup.py install
pip install .
```

This shoudl install in your system evegenie. YOu can verify this by typing:

which evegenie

TO BE CONTRIBUTED BY THE STUDENTS OF THE CLASS as homework of you see the path evegenie is installed. With evegenie installed its usaage is simple:

> dbsager: container. evegenie --help evegenie FILENAME

It takes a json file as input and writes out a settings file for the use in eve. Lets assume the file is called sample.json, than the settings file will be called sample.settings.py. Having the evegenie programm will allow us to generate the settings files easily. You can include them into your project and leverage the Makefile targets to start the services in your project. In case you generate new objects, make sure you rerun evegenie, kill all previous windows in whcih you run eve and mongo and restart. In case of changes to objects that you have designed and run previously, you need to also delete the mongod database.

I.4. Towards cmd5 extensions to manage eve and mongo

Naturally it is of advantage to have in cms administration commands to manage mongo and eve from cmd instead of targets in the Makefile. Hence, we propose that the class develops such an extension. We will create in the repository the extension called admin and hobe that students through collaborative work and pull requests complete such an admin command.

The proposed command is located at:

 https://github.com/cloudmesh/rest/blob/master/ cloudmesh/ext/command/admin.py

816

823

825

828

829

830

837

841

842

843

844

845

846

850

851

852

853

855

It will be up to the class to implement such a command. 810 Please coordinate with each other.

The implementation based on what we provided in the Make file seems straight forward. A great extensinion is to load the objects definitions or eve e.g. settings.py not from 813 the class, but forma place in .cloudmesh. I propose to place the file at:

```
.cloudmesh/db/settings.py
```

the location of this file is used whne the Service class is 817 initialized with None. Prior to starting the service the file 818 needs to be copied there. This could be achived with a set 819 commad. classes lesson python cmd5.rst

J. CMD5

757

758

759

760

761

765

767

768

769

770

771

772

773

774

779

780

783

785

786

787

788

791

794

795

797

803

ลกล

807

ลกล

CMD is a very useful package in python to create command line shells. However it does not allow the dynamic integration of newly defined commands. Furthermore, addition to cmd need to be done within the same source tree. To simplify 826 developping commands by a number of people and to have a dynamic plugin mechnism, we developed cmd5. It is a rewrite on our ealier effords in cloudmesh and cmd3.

J.1. Resources

The source code for cmd5 is located in github:

https://github.com/cloudmesh/cmd5

Installation from source -

We recommend that you use a virtualenv either with virtualenv or pyenv. This can be either achieved vor virtualenv with:

virtualenv ~/ENV2

or for pyenv, with:

pyenev virtualenv 2.7.13 ENV2

Now you need to get two source directories. We assume yo place them in ~/github:

```
mkdir ~/github
789
    cd ~/github
790
```

```
git clone https://github.com/cloudmesh/common.git
git clone https://github.com/cloudmesh/cmd5.git
git clone https://github.com/cloudmesh/extbar.git
```

cd ~/github/common 796

python setup.py install

pip install . 798

cd ~/github/cmd5

python setup.py install 801 pip install . 802

cd ~/github/extbar 804 805

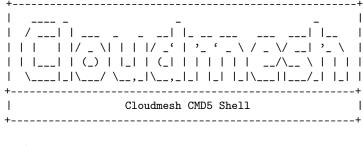
python setup.py install pip install .

The cmd5 repository contains the shell, while the extbar directory contains the sample to add the dynamic commands 858 foo and bar.

J.2. Execution

To run the shell you can activate it with the cms command. cms stands for cloudmesh shell:

It will print the banner and enter the shell:



To see the list of commands you can say

cms> help

To see the manula page for a specific command, please use:

help COMMANDNAME

J.3. Create your own Extension

One of the most important features of CMD5 is its ability to extend it with new commands. This is done via packaged name spaces. This is defined in the setup.py file of your enhancement. The best way to create an enhancement is to take a look at the code in

https://github.com/cloudmesh/extbar.git

Simply copy the code and modify the bar and foo commands to fit yor needs.

make sure you are not copying the .git directory. Thus we recommend that you copy it explicitly file by file or directory by directory

It is important that all objects are defined in the command itself and that no global variables be use in order to allow each shell command to stand alone. Naturally you should develop API libraries outside of the cloudmesh shell command and reuse them in order to keep the command code as small as possible. We place the command in:

cloudmsesh/ext/command/COMMANDNAME.py

An example for the bar command is presented at:

 https://github.com/cloudmesh/extbar/blob/master/ cloudmesh/ext/command/bar.py

It shows how simple the command definition is (bar.py):

```
from __future__ import print_function
from cloudmesh.shell.command import command
from cloudmesh.shell.command import PluginCommand
```

class BarCommand(PluginCommand):

```
860
         def do_bar(self, args, arguments):
861
              11 11 11
862
863
              ::
                Usage:
864
                       command -f FILE
                       command FILE
866
                       command list
867
                This command does some useful things.
868
                Arguments:
869
                    FILE
                             a file name
870
                Options:
871
                     -f
                              specify the file
872
              print(arguments)
874
```

An important difference to other CMD solutions is that our commands can leverage (besides the standrad definition), docopts as a way to define the manual page. This allows us to use arguments as dict and use simple if conditions to interpret the command. Using docopts has the advantage that contributors are forced to think about the command and its options and document them from the start. Previously we used not to use docopts and argparse was used. However we noticed that for some contributions the lead to commands that were either not properly documented or the developers delivered ambiguous commands that resulted in confusion and wrong ussage by the users. Hence, we do recommend that you use docopts.

The transformation is enabled by the @command decorator that takes also the manual page and creates a proper help message for the shell automatically. Thus there is no need to introduce a separate help method as would normally be needed in CMD.

J.4. Excersise

875

876

877

878

879

883

884

885

886

887

890

891

892

893

894

895

899

CMD5.1: Install cmd5 on your computer.

CMD5.2: Write a new command with your firstname as the command name.

CMD5.3: Write a new command and experiment with docopt syntax and argument interpretation of the dict with if conditions.

CMD5.4: If you have useful extensions that you like us to
 add by default, please work with us.