

---

# **mist.client Documentation**

*Release 1.8.0*

**Chris Loukas**

July 23, 2016



<b>1</b>	<b>Quickstart</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>The mist command</b>	<b>7</b>
<b>4</b>	<b>Python API</b>	<b>17</b>
<b>5</b>	<b>Ansible modules</b>	<b>25</b>
<b>6</b>	<b>Package Info</b>	<b>35</b>



**Version** 1.2.1

**Author** Mist.io Inc

**Source** <https://github.com/mistio/mist.client>

**License** GPL v3

Mist is a Python and a command line interface for managing and monitoring servers across clouds from any device that can access the web. To use it you need an account with the freemium <https://mist.io> service.



---

## Quickstart

---

Install mist using pip:

```
pip install mist
```

Let's add our backends, for example an ec2 and an Openstack:

```
mist add-backend --provider ec2 --region ec2_ap_northeast --name EC2 --ec2-api-key ALKID98IGGYUG --ec2-secret-key ...  
mist add-backend --provider openstack --name Openstack --openstack-username admin --openstack-tenant-id ...
```

We can now provision new machines just like that:

```
mist create-machine --backend EC2 --name mongo.myserver --location_id 0 --size_id m1.small --image_id ...  
mist create_machine --backend Openstack --name mongo2.myopenstackserver --location_id 0 --size_id 2 --image_id ...
```

We can tag machines into groups:

```
mist tag mongo.myserver --new-tag dev  
mist tag mongo2.myopenstackserver --new-tag dev
```

We can run batch commands to all machines in the dev group:

```
mist run --command "apt-get update -y" --tag dev
```

And even enable monitoring with a single command:

```
mist enable-monitoring mongo.myserver
```





---

## Installation

---

### 2.1 Install using pip

This is the easiest way to obtain the mist package:

```
pip install mist
```

### 2.2 Bash completion

To enable bash completion, you have to do the following:

```
sudo activate-global-python-argcomplete
```

If you are on Mac OSX, you have to do the following:

```
activate-global-python-argcomplete --dest=/usr/local/opt/bash-completion/etc/bash_completion.d
```

And then add the following line in your `~/.bashrc`:

```
eval "$(register-python-argcomplete /usr/bin/mist)"
```

If you are on Mac OSX, you have to add the following line to your `~/.bash_profile`:

```
eval "$(register-python-argcomplete mist)"
```



---

## The mist command

---

### 3.1 Introduction

`mist` will prompt for your `mist.io` email and password. You have the option to create a config file at `~/.mist`. By having this config file you'll be able to use the `mist` command without providing your credentials every time. The config file will look like this:

```
[mist.credentials]
email=user@mist.io
password=mist_password
```

To see your accounts' specific information:

```
mist user-info
```

Output:

```
User Details:
+-----+-----+-----+-----+
| country | company_name | number_of_servers | name | number_of_people |
+-----+-----+-----+-----+
| Greece | Mist | 1-5 | John Doe | 1-5 |
+-----+-----+-----+-----+

Current Plan:
+-----+-----+-----+-----+-----+-----+-----+-----+
| machine_limit | promo_code | title | started | isTrial | has_expired |
+-----+-----+-----+-----+-----+-----+-----+
| 20 | | Startup | Mon Oct 28 18:49:50 2013 | True | False |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

### 3.2 Backends

With `mist` you can handle multiple machines on multiple providers from one interface, the `mist.io` service. In order to do so, the very first thing to do when using `mist.io` is to ensure that you have added your backends. After doing that you'll be able to provision, monitor and in general handle all your machines on all those providers.

### 3.2.1 Supported Providers

Before you add a new backend, you'll find it useful to see a list of all the providers that mist.io supports:

```
mist list-providers
```

Output:

```
Other Server          bare_metal
Azure                azure
EC2                  ec2                Tokyo                ec2_ap_northeast
EC2                  ec2                Singapore            ec2_ap_southeast
EC2                  ec2                Sydney               ec2_ap_southeast_2
EC2                  ec2                Ireland              ec2_eu_west
EC2                  ec2                Sao Paulo            ec2_sa_east
EC2                  ec2                N. Virginia          ec2_us_east
EC2                  ec2                N. California        ec2_us_west
EC2                  ec2                Oregon               ec2_us_west_oregon
Google Compute Engine gce
NephoScale           nephoscale
DigitalOcean          digitalocean
Linode                linode
OpenStack             openstack
Rackspace             rackspace          Dallas                dfw
Rackspace             rackspace          Chicago               ord
Rackspace             rackspace          N. Virginia          iad
Rackspace             rackspace          London                lon
Rackspace             rackspace          Sydney                syd
Rackspace             rackspace          Hong Kong             hkg
Rackspace             rackspace          US-First Gen          rackspace_first_gen:us
Rackspace             rackspace          UK-First Gen          rackspace_first_gen:uk
SoftLayer             softlayer
HP Helion Cloud       hpcloud            US West               region-a.geo-1
HP Helion Cloud       hpcloud            US East               region-b.geo-1
Docker                docker
VMware vCloud         vcloud
Indonesian Cloud      indonesian_vcloud
KVM (via libvirt)     libvirt
```

**Note:** With every *list* action, you can have the output in a more *pretty* format by providing the `--pretty` flag.

For example, `mist list-providers --pretty` will return this output:

Provider	Provider ID	Region	Region ID
Other Server	bare_metal	-	-
Azure	azure	-	-
EC2	ec2	Tokyo	ec2_ap_northeast
EC2	ec2	Singapore	ec2_ap_southeast
EC2	ec2	Sydney	ec2_ap_southeast_2
EC2	ec2	Ireland	ec2_eu_west
EC2	ec2	Sao Paulo	ec2_sa_east
EC2	ec2	N. Virginia	ec2_us_east
EC2	ec2	N. California	ec2_us_west
EC2	ec2	Oregon	ec2_us_west_oregon
Google Compute Engine	gce	-	-
NephoScale	nephoscale	-	-

DigitalOcean	digitalocean	-	-
Linode	linode	-	-
OpenStack	openstack	-	-
Rackspace	rackspace	Dallas	dfw
Rackspace	rackspace	Chicago	ord
Rackspace	rackspace	N. Virginia	iad
Rackspace	rackspace	London	lon
Rackspace	rackspace	Sydney	syd
Rackspace	rackspace	Hong Kong	hkg
Rackspace	rackspace	US-First Gen	rackspace_first_gen:us
Rackspace	rackspace	UK-First Gen	rackspace_first_gen:uk
SoftLayer	softlayer	-	-
HP Helion Cloud	hpcloud	US West	region-a.geo-1
HP Helion Cloud	hpcloud	US East	region-b.geo-1
Docker	docker	-	-
VMware vCloud	vcloud	-	-
Indonesian Cloud	indonesian_vcloud	-	-
KVM (via libvirt)	libvirt	-	-

From here on you'll need your desired provider's id in order to use it when adding a new backend.

### 3.2.2 Backend Actions

Add an EC2 backend:

```
mist add-backend --provider ec2 --region ec2_ap_northeast --ec2-api-key AKIAHKIB7OIJCX7YLI03JA --ec2-
```

Add a Rackspace backend:

```
mist add-backend --provider rackspace --region iad --rackspace-username my_username --rackspace-api-
```

Add a Nephoscale backend:

```
mist add-backend --provider nephoscale --nepho-username nepho_username --nepho-password nepho_passwd
```

Add a DigitalOcean backend:

```
mist add-backend --provider digitalocean --digi-token kjhdkfh897dfodlkfjlkhd90sdfusldkfjkljsdf098lk
```

Add a Linode backend:

```
mist add-backend --provider linode --linode-api-key dkljflkjlkjgddgijgd00987ghudGgcf9G1kj
```

Add an OpenStack backend:

```
mist add-backend --provider openstack --openstack-username demo --openstack-password mypass --opensta
```

Add a Softlayer backend:

```
mist add-backend --provider softlayer --softlayer-username soft_username --softlayer-api-key kjhdkj
```

Add a HP Cloud backend:

```
mist add-backend --provider hpcloud --region region-a.geo-1 --hp-username hp_username --hp-password r
```

Add a Azure backend:

To add a Azure backend you have to download to a file the Azure certificate.

```
mist add-backend --provider azure --azure-sub-id lkjoiy8-kjddjkhd-987-hd9d --azure-cert-path /home/user
```

Add a Docker backend:

```
mist add-backend --provider docker --docker-host 10.0.0.1 --docker-port 4243
```

Add a Bare Metal Server (or any server):

```
mist add-backend --provider bare_metal --bare-hostname 198.230.89.3 --bare-user root --bare-port 22
```

Add a Google Compute Engine backend:

To add a GCE backend you have to download the private key file

```
mist add-backend --provider gce --gce-email 46234234246-3oiuoiu0980989873yui@developer.gserviceaccount
```

Add VMware(vCloud) backend:

```
mist add-backend --provider vcloud --vcloud-username admin --vcloud-password ioiuYoIU --vcloud-org
```

Add Indonesian vCloud backend:

```
mist add-backend --provider indonesian_vcloud --indonesian-username admin --indonesian-password kjOI
```

Add KVM(via libvirt) backend:

```
mist add-backend --provider libvirt --libvirt-hostname 10.0.0.1 --libvirt-user root --libvirt-key My
```

You can now see a list of all your added backends:

```
mist list-backends
```

Output:

openstackaf0.mist.io	2Mn2ZnCoXhK3ywqzGn1fzWVmSse6	bare_metal
Icehouse	4ukW6Juooqa8bTu2YgM4mE8RAsk7	openstack
EC2 AP Sydney	25ykPERh5D17DyoeKsCgw35DLmwv	ec2_ap_southeast_2
Openstack Juno	2u5yKqXmDiZ7BHck1u17FFcmFS2m	openstack
HP Helion Cloud	3WwgPBXETjdeMEbM5fUCACSvedGT	hpcloud
Google Compute Engine	g6T3HYae2ZMcHfHyFGKvtMG6PZU	gce
Docker	B3rbEA6bteaqMWJ4obVbgbqrXWf	docker
openstackdfe.mist.io	XMdRN2u3NVASmM14BuHo4HJnS15	bare_metal

**Note:** You can use the `--pretty` flag. `mist list-backends --pretty` will return:

Name	ID	Provider	State
openstackaf0.mist.io	2Mn2ZnCoXhK3ywqzGn1fzWVmSse6	bare_metal	online
Icehouse	4ukW6Juooqa8bTu2YgM4mE8RAsk7	openstack	online
EC2 AP Sydney	25ykPERh5D17DyoeKsCgw35DLmwv	ec2_ap_southeast_2	online
Openstack Juno	2u5yKqXmDiZ7BHck1u17FFcmFS2m	openstack	online
HP Helion Cloud	3WwgPBXETjdeMEbM5fUCACSvedGT	hpcloud	online
Google Compute Engine	g6T3HYae2ZMcHfHyFGKvtMG6PZU	gce	online
Docker	B3rbEA6bteaqMWJ4obVbgbqrXWf	docker	online
openstackdfe.mist.io	XMdRN2u3NVASmM14BuHo4HJnS15	bare_metal	online

You can also display information about a specific backend, either by providing the backend's name or ID. The following commands are equivalent:

```
mist describe-backend Icehouse
mist describe-backend 4ukW6Juooqa8bTu2YgM4mE8RAsk7
mist describe-backend --id 4ukW6Juooqa8bTu2YgM4mE8RAsk7
mist describe-backend --name Icehouse
```

Output:

```
+-----+-----+-----+-----+
| Title | ID | Provider | State |
+-----+-----+-----+-----+
| Icehouse | 4ukW6Juooqa8bTu2YgM4mE8RAsk7 | openstack | online |
+-----+-----+-----+-----+

Machines:
+-----+-----+-----+-----+
| Name | ID | State | Public Ips |
+-----+-----+-----+-----+
| atlanta | c9411bbe-2bb2-4a88-996c-d831272b426e | running | 109.59.77.32 |
+-----+-----+-----+-----+
```

You have the option to rename a backend:

```
mist rename-backend Icehouse --new-name Openstack_Icehouse
```

Finally you can delete a backend. The following two commands are equivalent:

```
mist delete-backend Docker
```

## 3.3 Keys

By uploading your SSH keys to mist.io you can access all your machines through mist.io, have a shell prompt from your browser and even let mist.io take care of enabling monitoring to your machines. You also can have mist.io run commands to your machines during provisioning or after an alert is triggered.

### 3.3.1 Add a new key

You can use one of your existing keys and upload it to mist.io for further usage:

```
mist add-key --name MyKey --key-path /home/user/.ssh/mist_key
```

Or you can ask mist.io to auto-generate a key for you:

```
mist add-key --name AutogeneratedKey --auto-generate
```

### 3.3.2 Keys Actions

To list your keys:

```
mist list-keys
```

Output:

```
Dummy
testkey
ParisDemo
TestKey
DemoKey
```

Or use the `--pretty` flag. `mist list-keys --pretty`:

```
+-----+-----+
| Name | Is Default |
+-----+-----+
| Dummy | False |
| testkey | False |
| DemoKey | True |
| TestKey | False |
| ParisDemo | False |
+-----+-----+
```

You can also inspect a specific key:

```
mist describe-key Dummy
```

Output:

```
Name: Dummy
```

```
Private key:
```

```
-----BEGIN RSA PRIVATE KEY-----
```

```
MIIEpAIBAAKCAQEAz1aWE6y8uB3PQJh1Vdc1RpZyRlBFQpN8c2edGIP/SfdAeGT3
QdOoTJfKvZTxk99YJG/cRPzan19PAjZXJjYX1CiyF5YpJivRfN7j/QzzMJv6ouK/
62WxyjwWxDa9pixAQj2na9N0Gn8sqIIFxFqEXW0wFkac3A4I8vke8AZrRitGw3MO
FoIfrZjcicW6U2b4XLgK3vLSIe5myN9bgAqTPYPOLm/m8Rz3cv+1B0qCbPZEHBG3
2zoLTG40F6JgmekUrNSQhKaEWJjwLJRRj4aEtw7WeSbP3lnVNm0ch34j4+vVIp0L
+hFYAt9gjI2p/aa/YRg++H5Wfvpvz21POWw4pTQIDAQABAoIBAA4ai7bm5yd3D6QL
Oc1vDDazAS77QtrWgX6wK6WBRRpY8U+/PnqB1U7wDOOtZolyheJkoY0nzg872HoE
DEWTJGfQJNz/byKlLejamJOcD+bclV4DIp72mC6vi7TpLF1jZTOcUgkppxouUHFd
9tp2dc6NINpDD2SAP+cvPwWAYkdJhuKI/cruyZ2y6b+FNC0JPF0f1yB6gwD3KAj0
YjcvDRjDaZKwFej+97YaKt37FuQuUjOKIruMytlcxm9qzQfSPeubfHEya61dL+Za
epJjm4NN5+x9PqSGhNpSbj1KwEbI67zNLLovEep7IC/7Et4rXm3/OtbNg1Kb/s67
YAifgVkcGyEA1H6PgSHp2Y12m+fIBFLyQqWOW3DjBV267h+R26pLOLFQCBaONZjS
35Ru+prQeCIGRbGD5BC/DP19qkk0VIuVY1KIRfgrYEmS2Uq+h24htpaqw+Ehqs6
yN7q5ppikONP8wJ+y25u8TN8kssZm8U8Q3qOCgnZ2prP/ctB1eefkvcCgYEA+cnG
7ygDoHv7sdmGDkAAkuU0skhpaZD4CV2XvWtS61vAu4V3xPkLAAi43rUuPqO/R7LG
br3CaDDe3PJ0jXsZjTgPm6eIz5hsglm3aoaQ6cDBJS1B9B488eDLkT816CH2IAuf
XsmqNKWFVcn+oWlLkYdZWP49+S8er7ulKfOEENsCgYB8RRO5qlKvdyqxXKi91qB1
V4rcctVjMwCaN/4+H+zj4iOYR1CdiaVxOcZ5asZaTEUMxxbh7uU8PJccWjlvZD5V
xPyLJuq79EMcLrkkTMUMmip96ZCdZcL4LF3lxPNjlnxwGrp6UDgzeS/WTU7JqVxn
/i1JN5+fV8BhpVf4N8A72wKBgQDrP2eF8W+JA3uGglDItupTb1500dHFRGz11RnF
oYBUfPNFKGw11z7Qh2ZlCMnm4JzTT8Gmpjyjl/Msr1/fxVq8YpUyOsSUjv8SvKAL
SXTNUWYWN0t4N8o6GvZdctWmi+WbRjbx1IfiUUKEBNs070k6B/jT4Y5IUmJaKyVg
HyHwJQKBgQDOBYoJjancXX4H7sW8rah5j7Lj3LYfTc2kwLUv9NeROd+gdVPZt9PT
SWbT/d+7foCyMwWIK3eCT71FHsiR8nNivet8AFjnm3aa8xTgvJwZlCLhvyWA3Fht
8NpVCBubPk4+fs2x0j/D3Uwgho51XXztngE/R3nr1XeB7xDSJm1iEA==
```

```
-----END RSA PRIVATE KEY-----
```

```
Public key:
```

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDPVpYTrLy4Hc9AmGVV1zVGlnJGUEVcK3xzZ50Yg/9J90B4ZPdE06hMkWRV1PGT...
```

You have the option to rename a key:



```
mist rename-key Dummy --name MyKey --new-name RenamedKey
```

And delete one:

```
mist delete-key Dummy
```

## 3.4 Machines

Now that you have added your backends and keys you can provision and monitor any machine on any of your providers.

Before you provision a machine, you'll need to provide some information, regarding the OS Image to use, the size of the machine and on which Backend's location. All of these information differ with each provider. However you can list all of them and choose your desired values.

### 3.4.1 Images

To see all the available images for a backend. *The --backend option can be either the backend's id or name. Both will do.*

```
mist list-images --backend Juno
```

Output:

```
Fedora-x86_64-20-20140618-sda          755c8a98-882f-4dd2-9598-5c01c039e63a
cirros-0.3.2-x86_64-uec              cbcc00f7-6ec0-41a5-ad42-3008143a77b2
cirros-0.3.2-x86_64-uec-ramdisk      586360b9-06f4-4353-9f62-7191a9f95d64
cirros-0.3.2-x86_64-uec-kernel       475ae832-7d2a-4b0b-a4d9-63e7d170a223
```

And with the `--pretty` flag, `mist list-images --backend Juno --pretty`:

```
+-----+-----+
|           Name           |           ID           |
+-----+-----+
| Fedora-x86_64-20-20140618-sda | 755c8a98-882f-4dd2-9598-5c01c039e63a |
|   cirros-0.3.2-x86_64-uec   | cbcc00f7-6ec0-41a5-ad42-3008143a77b2 |
| cirros-0.3.2-x86_64-uec-ramdisk | 586360b9-06f4-4353-9f62-7191a9f95d64 |
|   cirros-0.3.2-x86_64-uec-kernel | 475ae832-7d2a-4b0b-a4d9-63e7d170a223 |
+-----+-----+
```

The list of images can be huge, especially on providers such as EC2. My default mist.io will return a list of the most used images. You can however use the `--search` option. If you provide `--search all` mist.io will provide all available images. If you want to narrow your search you can search for a specific image:

```
mist list-images --backend DigitalOcean --search all
mist list-images --backend DigitalOcean --search gentoo
```

From the returned list you 'll need your desired image's ID to be used with machine creation.

### 3.4.2 Sizes - Locations/Regions

Each provider offers different options for machine sizes and locations/regions to choose from. For each of them you'll need the corresponding ID:

```
mist list-sizes --backend DigitalOcean
mist list-sizes --backend DigitalOcean --pretty
```

Output:

Name	ID
CS05-SSD - 0.5GB, 1Core, 25GB, 10 Gbps	219
CS1-SSD - 1GB, 1Core, 25GB, 10 Gbps	221
CS2.1-SSD - 2GB, 1Core, 37GB, 10 Gbps	223
CS2.2-SSD - 2GB, 2Core, 50GB, 10 Gbps	225
CS4.2-SSD - 4GB, 2Core, 75GB, 10 Gbps	227
CS4.4-SSD - 4GB, 4Core, 100GB, 10 Gbps	229
CS8.4-SSD - 8GB, 4Core, 150GB, 10 Gbps	231
CS8.8-SSD - 8GB, 8Core, 200GB, 10 Gbps	233
CS16.8-SSD - 16GB, 8Core, 300GB, 10 Gbps	235
CS16.16-SSD - 16GB, 16Core, 400GB, 10 Gbps	237
CS32.8-SSD - 32GB, 8Core, 600GB, 10 Gbps	239
CS32.16-SSD - 32GB, 16Core, 800GB, 10 Gbps	241
CS64.20-SSD - 64GB, 20Core, 1600GB, 10 Gbps	243
CS05 - 0.5GB, 1Core, 25GB, 1 Gbps	5
CS1 - 1GB, 1Core, 50GB, 1 Gbps	3
CS2.1 - 2GB, 1Core, 75GB, 1 Gbps	46
CS2.2 - 2GB, 2Core, 100GB, 1 Gbps	7
CS4.2 - 4GB, 2Core, 150GB, 1 Gbps	48
CS4.4 - 4GB, 4Core, 200GB, 1 Gbps	9
CS8.4 - 8GB, 4Core, 300GB, 1 Gbps	50
CS8.8 - 8GB, 8Core, 400GB, 1 Gbps	11
CS16.8 - 16GB, 8Core, 600GB, 1 Gbps	52
CS16.16 - 16GB, 16Core, 800GB, 1 Gbps	1
CS32.8 - 32GB, 8Core, 1000GB, 1 Gbps	56
CS32.16 - 32GB, 16Core, 1200GB, 1 Gbps	54

```
mist list-locations --backend DigitalOcean
mist list-locations --backend DigitalOcean --pretty
```

Output:

Name	ID
SJC-1	86945
RIC-1	87729

### 3.4.3 Create a new machine

Now that you have gathered the information needed for machine creation you can tell mist to provision a machine on a specific backend. Alongside the image, location and size ID's you'll also need to provide a keys' name to be assigned to the newly created machine:

```
mist create-machine --backend EC2 --name dev.machine --image ami-bddaa2bc --size t1.micro --location
```

### 3.4.4 Machine Actions

You can list all your machines on all your Backends, or list machines on a specific backend:

```
mist list-machines
mist list-machines --backend Docker
```

You can start, stop, reboot or destroy a machine. To specify a machine you can either directly use the machine's name or ID, or pass the `--id`, `--name` flags:

```
mist reboot db-server-1
mist destroy db-server-1
```

You can also probe a machine. By probing a machine you verify that sshd is up and running and that you have access to the machine with the previously assigned key:

```
mist probe db-server-1
```

After creating a new machine it might take a little time for the probe to be successful.

You can also tag machine:

```
mist tag db-server-1 --new-tag dbservers
```

Tagging will be useful later when you want to group your machines across different clouds and run multiple commands and configuration scripts.

## 3.5 Monitoring

Mist.io offers plans for monitoring your machines. By default it will install a `collectd` instance pre-configured with some basic metrics and send the results to mist.io's servers. By visiting mist.io you can see live graphs of your monitored machines.

Furthermore, you have a huge list of `collectd` plugins that you can add to your machine and even upload custom python scripts to be used as `collectd` plugins, allowing you to monitor...well, almost everything.

### 3.5.1 Enable monitoring

In order to enable monitoring on a machine with name `dbServer`:

```
mist enable-monitoring dbServer
```

Now, your `dbServer` machine has `collectd` installed and you can visit mist.io to see live graphs (note that the first time you enable `collectd` it may take some time for the package to install).

To disable monitoring on a machine:

```
mist disable-monitoring dbServer
```

### 3.5.2 Add Metrics

`Collectd` supports a huge list of custom metrics/plugins. To see all available plugins/metrics for a monitored machine:

```
mist list-metrics --machine dbServer
```

If you wish to add one of those metrics you have to use the metric's id. For example, to add the metric `users`:

```
mist add-metric --machine dbServer --metric-id users
```

Mist.io supports custom, python plugins. For example, if you have a `~/plugin.py`:

```
import random

def read():
    # return random value
    return random.random()
```

You can add it by providing the `--custom_plugin` parameter and providing a plugin name with the `--plugin` parameter:

```
mist add-custom-metric --machine dbServer --metric-name my_custom_metric --file-path ~/plugin.py --u
```

## 3.6 Run commands

With `mist` command line tool you can run a bash command in multiple tagged servers at once. For example to run a command on all your dev servers:

```
mist run --command "touch something" --tag dev
```

### Output

```
Found tagged machines
Found key association for machine: atlanta

Finished in machine: atlanta
```

## 4.1 Introduction

Now that you have the `mist` package you can import `MistClient`:

```
from mistclient import MistClient
client = MistClient(email="yourmail@mist.io", password="yourpassword")
```

## 4.2 Backends

A backend can be an IaaS cloud, a Docker host, or any single server.

### 4.2.1 Supported Providers

Mist.io supports a big list of providers including EC2, Rackspace, SoftLayer, Digital Ocean, Nephoscale, Openstack, Docker, HP Cloud and any single server.

In order to see the list of all supported providers:

```
client.supported_providers
```

The result will look like this:

```
[{'provider': 'bare_metal', 'regions': [], 'title': 'Other Server'},
 {'provider': 'azure', 'regions': [], 'title': 'Azure'},
 {'provider': 'ec2',
  'regions': [{'id': 'ec2_ap_northeast', 'location': 'Tokyo'},
               {'id': 'ec2_ap_southeast', 'location': 'Singapore'},
               {'id': 'ec2_ap_southeast_2', 'location': 'Sydney'},
               {'id': 'ec2_eu_west', 'location': 'Ireland'},
               {'id': 'ec2_sa_east', 'location': 'Sao Paulo'},
               {'id': 'ec2_us_east', 'location': 'N. Virginia'},
               {'id': 'ec2_us_west', 'location': 'N. California'},
               {'id': 'ec2_us_west_oregon', 'location': 'Oregon'}],
  'title': 'EC2'},
 {'provider': 'gce', 'regions': [], 'title': 'Google Compute Engine'},
 {'provider': 'nephoscale', 'regions': [], 'title': 'NephoScale'},
 {'provider': 'digitalocean', 'regions': [], 'title': 'DigitalOcean'},
 {'provider': 'linode', 'regions': [], 'title': 'Linode'},
 {'provider': 'openstack', 'regions': [], 'title': 'OpenStack'},
```

```
{'provider': u'rackspace',
  u'regions': [{u'id': u'dfw', u'location': u'Dallas'},
    {u'id': u'ord', u'location': u'Chicago'},
    {u'id': u'iad', u'location': u'N. Virginia'},
    {u'id': u'lon', u'location': u'London'},
    {u'id': u'syd', u'location': u'Sydney'},
    {u'id': u'hkg', u'location': u'Hong Kong'},
    {u'id': u'rackspace_first_gen:us', u'location': u'US-First Gen'},
    {u'id': u'rackspace_first_gen:uk', u'location': u'UK-First Gen'}],
  u'title': u'Rackspace'},
{'provider': u'softlayer', u'regions': [], u'title': u'SoftLayer'},
{'provider': u'hpcloud',
  u'regions': [{u'id': u'region-a.geo-1', u'location': u'US West'},
    {u'id': u'region-b.geo-1', u'location': u'US East'}],
  u'title': u'HP Helion Cloud'},
{'provider': u'docker', u'regions': [], u'title': u'Docker'},
{'provider': u'vcloud', u'regions': [], u'title': u'VMware vCloud'},
{'provider': u'indonesian_vcloud',
  u'regions': [],
  u'title': u'Indonesian Cloud'},
{'provider': u'libvirt', u'regions': [], u'title': u'KVM (via libvirt)'}
```

## 4.2.2 Add Backend

Before anything you must add your Backends to the mist.io service. By doing that you'll be able to handle all your machines from the mist.io service or the service's API.

In order to add a backend, you'll need the provider information from the supported providers you listed before. For example to add a "Rackspace LON" backend:

```
client.add_backend(provider="rackspace", title="My Rack London", region="lon", username="rack_username")
```

See also `mist.client.add_backend` method for detailed information about the different params for each backend.

After adding a new backend, `mist.backends` are automatically updated.

## 4.2.3 Backend actions

You can see all of your added backends:

```
client.backends()
```

This will return a list of all your added backends:

```
[Backend => EC2 AP NORTHEAST, ec2_ap_northeast, D1g9abwqGUmQuZKGGBMfCgw8AUQ,
Backend => openstackaf0.mist.io, bare_metal, 2Mn2ZnCoXhK3ywqzGn1fzWVmSse6,
Backend => Icehouse, openstack, 4ukW6Juooqa8bTu2YgM4mE8RAsk7,
Backend => EC2 AP Sydney, ec2_ap_southeast_2, 25ykPERh5D17DyoeKsCgw35DLmvw,
Backend => Openstack Juno, openstack, 2u5yKqXmDiZ7BHCK1u17FFcmFS2m,
Backend => HP Helion Cloud, hpcloud, 3WwgPBXETjdeMEbM5fUCACSvedGT,
Backend => Google Compute Engine, gce, g6T3HYae2ZMchfHyFGKVtMG6PZU,
Backend => Docker, docker, B3rbEA6bteaQMwJ4obVbgbqrXWf,
Backend => openstackdfe.mist.io, bare_metal, XMdRN2u3NVASm14BuHo4HJnS15]
```

You can also choose a backend by providing either the backend's name or id:

```
backend = client.backends(id="XMdRN2u3NVASm14BuHo4HJnS15") [0]
backend = client.backends(name="Docker") [0]
```

You can also search in all the backends' ids and names:

```
backend = client.backends(search="OpenStack") [0]
```

Your new backend object has a lot of attributes and methods:

```
backend.id
backend.info
backend.images
...
```

See `mistclient.model.Backend` class for detailed information.

You have the option to rename a backend:

```
backend.rename("newName")
```

Finally, you can delete a backend:

```
backend.delete()
```

## 4.3 Keys

By uploading your SSH keys to mist.io you can access all your machines through mist.io, have a shell prompt from your browser and even let mist.io take care of enabling monitoring to your machines. You also can have mist.io run commands to your machines during provisioning or after an alert is triggered.

### 4.3.1 Add a new key

When adding a new key, you have 2 choices. Either upload a local ssh-key to mist.io, or ask mist.io to generate one for you.

When uploading a local ssh-key, you have to provide the private ssh-key as a string. So first you can:

```
with open("/home/user/.ssh/my_key") as f:
    private = f.read()
```

You now have the private key and can add a new key to mist.io:

```
client.add_key(key_name="MyKey", private=private)
```

Or have mist.io generate a random one for you:

```
private = client.generate_key()
client.add_key(key_name="MyKey", private=private)
```

After adding a new key, `client.keys` will be automatically updated.

### 4.3.2 Keys actions

To see all added keys:

```
client.keys()
```

The result will be a list like this:

```
[Key => Dummy,  
Key => ParisDemo2,  
Key => testkey,  
Key => DemoKey,  
Key => TestKey,  
Key => ParisDemo]
```

You can now search for key names:

```
key = client.keys(search="Paris")[0]
```

You have the option to set a key as the default one. This becomes handy if you want mist.io to auto-assign this key to a machine if you leave the association blank:

```
key.set_default()
```

You can rename the key:

```
key.rename("newName")
```

Finally, to delete the key:

```
key.delete()
```

See `mistclient.model.Key` class for detailed information.

## 4.4 Machines

Before you can provision a machine, you have to know some data that are necessary for the creation of a machine. Every backend has different OS Images, locations, machine sizes. You can list all the available options after you have chosen a backend:

```
backend = client.backends(search="NephoScale")
```

### 4.4.1 Images

You can list all available OS Images in a backend:

```
backend.images
```

This will return a list of all available images. From the desired image you will need the image's id in order to create a machine with that image:

```
[{'extra': {'architecture': u'x86',  
           u'billable_type': None,  
           u'cores': None,  
           u'disks': None,  
           u'pcpus': None,  
           u'storage': None,  
           u'uri': u'https://api.nephoscale.com/image/server/3/'},  
  u'id': u'3',  
  u'name': u'Linux CentOS 5.5 32-bit',
```



```
u'star': True},
{u'extra': {u'architecture': u'x86_64',
  u'billable_type': None,
  u'cores': None,
  u'disks': None,
  u'pcpus': None,
  u'storage': None,
  u'uri': u'https://api.nephoscale.com/image/server/5/'},
  u'id': u'5',
  u'name': u'Linux CentOS 5.5 64-bit',
  u'star': True},
{u'extra': {u'architecture': u'x86',
  u'billable_type': None,
  u'cores': None,
  u'disks': None,
  u'pcpus': None,
  u'storage': None,
  u'uri': u'https://api.nephoscale.com/image/server/23/'},
  u'id': u'23',
  u'name': u'Linux Debian Server 5.05 32-bit',
  u'star': True},
{u'extra': {u'architecture': u'x86',
  u'billable_type': None,
  u'cores': None,
  u'disks': None,
  u'pcpus': None,
  u'storage': None,
  u'uri': u'https://api.nephoscale.com/image/server/43/'},
  u'id': u'43',
  u'name': u'Linux Ubuntu Server 10.04 LTS 32-bit',
  u'star': True},
{u'extra': {u'architecture': u'x86',
  u'billable_type': None,
  u'cores': None,
  u'disks': None,
  u'pcpus': None,
  u'storage': None,
  u'uri': u'https://api.nephoscale.com/image/server/45/'},
  u'id': u'45',
  u'name': u'Linux CentOS 5.7 32-bit',
  u'star': True},
{u'extra': {u'architecture': u'x86_64',
  u'billable_type': None,
  u'cores': None,
  u'disks': None,
  u'pcpus': None,
  u'storage': None,
  u'uri': u'https://api.nephoscale.com/image/server/49/'},
  u'id': u'49',
  u'name': u'Linux Ubuntu Server 10.04 LTS 64-bit',
  u'star': True},
{u'extra': {u'architecture': u'x86_64',
  u'billable_type': None,
  u'cores': None,
  u'disks': None,
  u'pcpus': None,
  u'storage': None,
  u'uri': u'https://api.nephoscale.com/image/server/51/'},
```

```
u'id': u'51',
u'name': u'Linux Debian Server 6.0.3 64-bit',
u'star': True},
{'extra': {'architecture': u'x86_64',
u'billable_type': None,
u'cores': None,
u'disks': None,
u'pcpus': None,
u'storage': None,
u'uri': u'https://api.nephoscale.com/image/server/55/'},
u'id': u'55',
u'name': u'Linux Debian 5.0.9 64-bit',
u'star': True}]

image_id = backend.images[0]['id']
```

You also have the option to search for an image. Especially in EC2 backends, the result of the search will include community and public images:

```
backend.search_image("Debian")
```

## 4.4.2 Sizes

To list available machine sizes for the chosen backend:

```
backend.sizes
```

From the list of all available sizes, you'll also need the id of the desired size:

```
[{'bandwidth': None,
u'disk': 25,
u'driver': u'NephoScale',
u'id': u'219',
u'name': u'CS05-SSD - 0.5GB, 1Core, 25GB, 10 Gbps',
u'price': None,
u'ram': 512},
{'bandwidth': None,
u'disk': 25,
u'driver': u'NephoScale',
u'id': u'221',
u'name': u'CS1-SSD - 1GB, 1Core, 25GB, 10 Gbps',
u'price': None,
u'ram': 1024},
...

size_id = backend.sizes[0]['id']
```

## 4.4.3 Locations

Some backends have different locations for you to provision a machine to. You can list them:

```
backend.locations
```

From the list of available locations, you'll need the id of the desired location:

```
[{'country': u'US', 'id': u'86945', 'name': u'SJC-1'},
 {'country': u'US', 'id': u'87729', 'name': u'RIC-1'}]

location_id = backend.locations[0]
```

#### 4.4.4 Create machines

In order to create a machine you basically need to have chosen a backend, a key, image\_id, location\_id, size\_id and a name for the machine:

```
backend.create_machine(name="production.server", key=key, image_id=image_id, location_id=location_id,
```

In some backends some extra information is needed. You can see `mistclient.model.Backend.create_machine` method for more details.

#### 4.4.5 Machine actions

You can see a list of all your created machines for a given backend:

```
client.machines()
```

Or for a specific backend:

```
backend.machines()
```

You can choose one:

```
machine = client.machines(search="dev")[0]
machine = client.machines(name="dbserver1")[0]
```

Machines support actions like:

```
machine.reboot()
machine.start()
machine.stop()
machine.destroy()
```

After creating a machine, the machine may take some time to be up and running. You can see that by using `machine.probe()`. Machine probe, if successful will show that the machine is up and running and that the key association was successful. It will also return some useful information about the machine like the machine's uptime etc.

In case you want, you can associate another ssh-key to the machine, provided you have uploaded that key to mist.io service:

```
machine.associate_key(key_id, host="187.23.43.98")
```

The host of the machine can be found in the `machine.info['public_ips']` list. You can also provide two more parameters. `ssh_user` and `ssh_port`.

## 4.5 Monitoring

### 4.5.1 Enable monitoring

In case you have an account with the mist.io service (<https://mist.io>), you can enable monitoring to a machine:

```
machine.enable_monitoring()
```

This will take some time, cause mist.io will auto-install collectd and configure it to send monitoring data to mist.io servers. One way to see that the process has finished and you have data coming is:

```
machine.get_stats()
```

In case enabling monitoring has finished you'll get your monitoring data in a dict.

### 4.5.2 Advanced monitoring options

By default, mist.io's collectd will be configured with some metrics, like Disk usage, CPU usage etc. However, mist.io supports a huge list of collectd plugins that you can choose from:

```
machine.available_metrics
```

Using your desired metric id, you can add that to a monitored machine. For example to have data about the number of users that are currently logged in, we can use the `users` metric:

```
machine.add_metric("users")
```

### 4.5.3 Custom metrics

Since the last updates of mist.io, you can now upload custom python metrics that can literally monitor anything. These plugins are simple python files that you can upload to the machine. They can be as simple as:

```
import random

def read():
    # return random value
    return random.random()
```

Or more complex, taking care of pings to other servers etc.

To upload a custom plugin to a monitored machine, all you need is the python file's path in your computer, and a name for the plugin:

```
machine.add_python_plugin(name="Random", python_file="/home/user/random.py")
```

Some more advanced options can be used, determining the `value_type`, the `unit` etc. You can see `mistclient.model.Machine.add_python_plugin` method for more info.

---

## Ansible modules

---

Once you have installed the `mist` package you'll be able to use the `mist` ansible modules in your playbooks. The easiest way to do so is to run the `mistplay` command, which is a wrapper of `ansible-playbook`:

```
mistplay main.yml
```

### 5.1 `mist_providers` - Lists all available providers supported by the `mist.io` service

- *Synopsis*
- *Options*
- *Examples*

#### 5.1.1 Synopsis

New in version 1.7.1.

Returns a list of all available providers and the corresponding regions that you can add and control through `mist.io` service. `mist_email` and `mist_password` can be skipped if `~/.mist` config file is present. See documentation for config file <http://mist.readthedocs.org/en/latest/cmd/cmd.html>

#### 5.1.2 Options

#### 5.1.3 Examples

```
- name: List supported providers, simple case
  mist_providers:
    mist_email: your@email.com
    mist_password: yourpassword
    provider: all
    register: providers

- name: List supported provider having ~/.mist config file present
  mist_providers:
    provider: all
```

```
register: providers

- name: List only ec2 provider options
  mist_providers:
    mist_email: your@email.com
    mist_password: yourpassword
    provider: ec2
  register: providers
```

## 5.2 mist\_backends - Manage backends in the mist.io service

- *Synopsis*
- *Options*
- *Examples*

### 5.2.1 Synopsis

New in version 1.7.1.

Manage multi-cloud backends through mist.io service. You can add/remove multiple backends from multiple providers through mist.io service. Before you can provision, monitor etc machines through mist.io, you have to first add a backend to the mist.io service. Mist.io supports EC2, Rackspace, Openstack, Linode, Google Compute Engine, SoftLayer, Digital Ocean, Nephoscale, Bare metal servers, Docker containers, HP Cloud, Azure, VmWare - Vcloud, KV“libvirt“, *mist\_email* and *mist\_password* can be skipped if *~/.mist* config file is present. See documentation for config file <http://mistclient.readthedocs.org/en/latest/cmd/cmd.html>

### 5.2.2 Options

### 5.2.3 Examples

```
- name: Add EC2 backend
  mist_backends:
    title: MyEC2
    provider: ec2
    api_key: kjhf98y9lkj0909kj90edffwwf432fd
    api_secret: LKHLKjlkdlkho8976dhjkjhd987987
    region: ec2_ap_northeast
    state: present

- name: Add Rackspace backend
  mist_backends:
    title: MyRackspace
    provider: rackspace
    region: dfw
    username: rack_username
    api_key: sadlkjnjkhbi0HBCG
    state: present

- name: Add Nephoscale backend
  mist_backends:
```

```
title: MyNepho
provider: nephoscale
username: nepho_user
password: nepho_pass
state: present

- name: Add SoftLayer backend
  mist_backends:
    title: MySoftLayer
    provider: softlayer
    username: SL09890
    api_key: kjhdskjhad987987098sdlkhjlajslkj
    state: present

- name: Add Digital Ocena backend
  mist_backends:
    title: MyDigi
    provider: digitalocean
    token: oiulksdjkhjd0987098lkahkjdhkj....
    state: present

- name: Add Google Compute Engine backend
  mist_backends:
    title: GCE
    provider: gce
    email: my.gce.email@gce
    project_id: electron-25
    private_key: /path/to/locally/stored/private_key
    state: present

- name: Add Azure backend
  mist_backends:
    title: AZURE
    provider: azure
    subscription_id: lkjafh-08jhkl-09kljlj...
    certificate: /path/to/locally/saved/certificate
    state: present

- name: Add Linode backend
  mist_backends:
    title: MyLinode
    provider: linode
    api_key: dlkjdljkd0989yKGFgjgc86798ohkl
    state: present

- name: Add Bare Metal (or any server with ssh access)
  mist_backends:
    title: MyOtherServer
    provider: bare_metal
    machine_ip: 190.20.10.45
    machine_user: myuser
    machine_key: name_of_key_added_to_mist.io
    machine_port: 22
    state: present

- name: Add vCloud backend
  mist_backends:
    title: MyVCLLOUD
```

```
provider: vcloud
username: vuser
password: vpass
organization: Mist.io
host: compute.idcloudonline.com
state: present

- name: Add Indonesian vCloud backend
mist_backends:
  title: IndoVCLLOUD
  provider: indonesian_vcloud
  username: vuser
  password: vpass
  organization: Mist.io
  state: present

- name: Add KVM(libvirt) backend
mist_backends:
  title: MyKVM
  provider: libvirt
  machine_hostname: 190.198.23.0
  machine_user: root
  machine_key: name_of_key_added_to_mist.io
  state: present

- name: Add HP Cloud backend
mist_backends:
  title: MyHP
  provider: hpcloud
  region: region-a.geo-1
  username: hpuser
  password: hppass
  tenant_name: my_tenant
  state: present

- name: Add Openstack backend
mist_backends:
  title: MyOPENSTACK
  provider: openstack
  username: user
  password: pass
  tenant_name: admin
  auth_url: http://190.132.20.22:5000
  region: my_region_if_exists
  state: present

- name: Add Docker backend
mist_backends:
  title: MyDOCKER
  provider: docker
  docker_host: 190.189.1.2
  docker_port: 4243
  auth_user: user if I have Basic HTTP AUTH setup
  auth_password: pass if I have Basic HTTP AUTH setup
  key_file: path to key file if I have TLS setup
  cert_file: path to cert file if I have TLS setup

- name: List information about DigitalOcean backend
```



```
mist_backends:
  mist_email: your@email.com
  mist_password: yourpassword
  backend: DigitalOcean
  register: backend
```

## 5.3 mist\_images - Lists all available OS images for a backend

- *Synopsis*
- *Options*
- *Examples*

### 5.3.1 Synopsis

New in version 1.7.1.

Returns a list of all available OS images that the given backend supports. *mist\_email* and *mist\_password* can be skipped if *~/.mist* config file is present. See documentation for config file <http://mistclient.readthedocs.org/en/latest/cmd/cmd.html>.

### 5.3.2 Options

### 5.3.3 Examples

```
- name: List default images for NephoScale backend
mist_images:
  mist_email: your@email.com
  mist_password: yourpassword
  backend: NephoScale
  register: images

- name: Search for gentoo images in backend with id i984JHdkjhKj
mist_images:
  mist_email: your@email.com
  mist_password: yourpassword
  backend: i984JHdkjhKj
  search: gentoo
  register: images
```

## 5.4 mist\_sizes - Lists all available machine sizes for a backend

- *Synopsis*
- *Options*
- *Examples*

### 5.4.1 Synopsis

New in version 1.7.1.

Returns a list of all available machine sizes for a given backend *mist\_email* and *mist\_password* can be skipped if *~/.mist* config file is present. See documentation for config file <http://mistclient.readthedocs.org/en/latest/cmd/cmd.html>

### 5.4.2 Options

### 5.4.3 Examples

```
- name: List sizes for a backend
  mist_sizes:
    mist_email: your@email.com
    mist_password: yourpassword
    backend: DigitalOcean
  register: sizes
```

## 5.5 mist\_locations - Lists all available locations/regions for a backend

- *Synopsis*
- *Options*
- *Examples*

### 5.5.1 Synopsis

New in version 1.7.1.

Returns a list of all available locations/regions for a given backend *mist\_email* and *mist\_password* can be skipped if *~/.mist* config file is present. See documentation for config file <http://mistclient.readthedocs.org/en/latest/cmd/cmd.html>

### 5.5.2 Options

### 5.5.3 Examples

```
- name: List locations for a backend
  mist_locations:
    mist_email: your@email.com
    mist_password: yourpassword
    backend: DigitalOcean
  register: locations
```

## 5.6 mist\_keys - Manage ssh-keys from mist.io service

- *Synopsis*
- *Options*
- *Examples*

### 5.6.1 Synopsis

New in version 1.7.1.

By uploading your SSH keys to mist.io you can access all your machines through mist.io, have a shell prompt from your browser and even let mist.io take care of enabling monitoring to your machines. You also can have mist.io run commands to your machines during provisioning or after an alert is triggered. *mist\_email* and *mist\_password* can be skipped if *~/.mist* config file is present. See documentation for config file <http://mistclient.readthedocs.org/en/latest/cmd/cmd.html>

### 5.6.2 Options

### 5.6.3 Examples

```
- name: Add local key named my_key to mist.io
mist_keys:
  mist_email: your@email.com
  mist_password: yourpassword
  name: myKey
  state: present
  key: /home/user/.ssh/my_key

- name: Auto-generate key and save locally
mist_keys:
  mist_email: your@email.com
  mist_password: yourpassword
  name: autoKey
  state: present
  auto_generate: true
  save_locally: true
  local_save_path: /path/to/save

- name: Delete key named myKey
mist_keys:
  mist_email: your@email.com
  mist_password: yourpassword
  name: myKey
  state: absent

- name: List info for key named myKey
mist_keys:
  mist_email: your@email.com
  mist_password: yourpassword
  name: myKey
  register: key
```

## 5.7 mist - Provision, monitor and manage machines with the mist.io service

- *Synopsis*
- *Options*
- *Examples*

### 5.7.1 Synopsis

New in version 1.7.1.

Manage machines in all of your added backends You can add/remove multiple backends from multiple providers through mist.io service. *mist\_email* and *mist\_password* can be skipped if *~/.mist* config file is present. See documentation for config file <http://mistclient.readthedocs.org/en/latest/cmd/cmd.html>

### 5.7.2 Options

### 5.7.3 Examples

```
- name: Provision Ubuntu machine to EC2
  mist:
    mist_email: your@email.com
    mist_password: yourpassword
    backend: EC2
    state: present
    name: MyMachine
    key: myKey
    image_id: ami-bddaa2bc
    size_id: m1.small
    location_id: 0

- name: Provision SUSE machine on EC2 and enable monitoring
  mist:
    mist_email: your@email.com
    mist_password: yourpassword
    backend: EC2
    state: present
    name: MyMachine
    key: myKey
    image_id: ami-9178e890
    size_id: m1.small
    location_id: 0
    monitoring: true
    wait_for_stats: true

- name: List info for machine with name dbServer
  mist:
    mist_email: your@email.com
    mist_password: yourpassword
    backend: EC2
    name: dbServer
    register: machine
```

```
- name: Enable monitoring and add custom plugin.py
  mist:
    mist_email: your@email.com
    mist_password: yourpassword
    backend: EC2
    name: dbServer
    state: present
    key: newKey
    wait: true
    monitoring: true
    wait_for_stats: true
    metric: MyPlugin
    python_file: /home/user/plugin.py
```



---

## Package Info

---

### 6.1 Changelog

#### 6.1.1 Release 0.3.0 (released Nov 18,2014)

Featured added:

- Repackage `mist.client` to `mist`
- Refactor `mistclient.machines` and `mistclient.backends`
- `client.machines`, `client.backends`, `client.keys` are now lists instead of dicts
- Refactor the `mist` command line tool
- Add `mist run` capability

#### 6.1.2 Release 0.1.0 (released Sep 3, 2014)

Features added:

- `mist` command line interface
- Add `client.backend_from_name`, `client.backend_from_id` and `client.search_backend` methods
- Add `backend.machine_from_name`, `backend.machine_from_id`, `backend.machine_from_ip` and `backend.search_machine` methods
- `client.backends` is now a dict with backend ids as `dict.keys`
- `backend.machines` is now a dict with machine ids as `dict.keys`

Bugs fixed:

- #5: Fix pip hanging up when installing requirements for the first time
- #6: Fix `mist sync` when syncing Bare Metal Backends