

# Flex Cloud VM Control

*FlexCLI is a simple unix shell script that executes Curl requests via the Flex Cloud API. This console app, demonstrates a few possibilities for interaction with the api.*

## API Essentials

## API Key Generation

## Autoscaling

- Add Autoscaling Rules
- Get List of Autoscaling Rules for VS
- Remove Autoscaling Rules

## Backups

- Add/Edit Note
- Convert Backup to Template
- Create Backup
- Delete Backup
- Get List of All VS Backups
- Get List of Incremental Backups
- Get List of Normal Backups
- Restore Disk from Backup

## Credit

- Get Credit

## DNS

- Add DNS Record
- Add DNS Zone
- Delete DNS Record
- Delete DNS Zone
- Edit DNS Record
- Get DNS Zones
- Get List of DNS Zone Records
- Get List of Name Servers

## Firewall Rules

- Add Firewall Rule
- Get Firewall Rules
- Update Firewall Rules

## Flex Cloud VM Control

## Getting Started with Superb Flex Cloud

## IP Addresses

- Get IP Address Joins

## Logs

- Get List of Log Items
- Get List of Transactions
- Get list of VS Transaction

## Network Interfaces

## Contents

## Setup

Click here to download the zipped file: <https://github.com/superbDev/flexCLI/archive/master.zip>, save/relocate to an appropriate directory such as `/usr/local/flexTools`.

Or find the latest stable release <https://github.com/superbDev/flexCLI/releases>

### Using terminal:

1. Navigate to the directory where you want to store the script folder:

```
$ cd /usr/local
/usr/local/$
```

2. Use wget to download the package

this example uses v1.1, but will work for newer releases

```
/usr/local$ wget
https://github.com/superbDev/flexCLI/archive/v1
.1.tar.gz
```

3. Extract the files

```
/usr/local$ tar -zxvf v1.1.tar.gz
```

4. Now you can execute the script from this directory (or from anywhere using the full path)

```
/usr/local$ cd flexCLI-1.1
/usr/local/flexCLI-1.1$ ./flexCLI.sh help
```

## Commands

All of the flexCLI commands except "list" correspond to an API call. To learn more about what is required, or what a command does check out it out in the API guide:

Command	Description	Corresponding API Call	Returns On Success
build <virtual_machine_id>	builds specified VM	Build Virtual Server	"success"
create	creates a new VM	Add Virtual Server	JSON Virtual Machine Object

- Get VS Network Interfaces
- Rebuild VS Network

## Recipes

- Add Recipe
- Add Recipe Step
- Assign Recipe to Virtual Server
- Delete Recipe
- Delete Recipe Step
- Edit Recipe
- Edit Recipe Step
- Get All Recipes
- Get Recipe Steps
- Get Virtual Server Recipes
- Remove recipe from Virtual Server
- Run Recipe on Multiple Virtual Servers
- Swap Recipe Step Number

## SSH Keys

- Add SSH Key
- Delete SSH Key
- Edit SSH Key
- Get SSH Keys
- Set SSH Keys on VS

## Templates

- Get Templates

## Test Route

## Troubleshooting API Issues

## Viewing Activity Logs

## Virtual Server Operating Systems

### Virtual Servers

- Add Virtual Server
- Billing Statistics
- Build Virtual Server
- Delete Virtual Server
- Edit Virtual Server
- Get CPU Usage Statistics
- Get List of Virtual Machines
  - Get specific VM Details
  - Search Virtual Servers by label
- Get statuses for All VMs
  - Get Specific VS Status
- Reboot Virtual Server
- Reset VS Password
- Shutdown Virtual Server
- Startup a Virtual Server
- Stop Virtual Server

delete <virtual_machine_id>	deletes VM	Delete Virtual Server	"success"
edit <virtual_machine_id>	edits(also used to re-size) specified VM	Edit Virtual Server	JSON Virtual Machine Object
list(ls)	A brief formatted summary of all your machines	N/A	Brief overview of current machines
list(ls)<virtual_machine_id>	A brief formatted summary of a single VM	N/A	Brief overview of single VM
get	returns a json array representing all your machines	Get List of Virtual Machines	JSON Virtual Machine Object Array
get<virtual_machine_id>	returns the json object representation of specified VM	Get specific VM Details	JSON Virtual Machine Object
reboot <virtual_machine_id>	reboot a single VM	Reboot Virtual Server	"success"
search <label>	search for a VM by label	Search Virtual Servers by label	JSON Virtual Machine Object Array
shutdown <virtual_machine_id>	shuts down specified VM	Shutdown Virtual Server	"success"
status	returns a list of all VM statuses	Get statuses for All VMs	JSON Virtual Machine Status Object Array
status <virtual_machine_id>	returns the status of a single VM	Get Specific VS Status	JSON Virtual Machine Status Object
stop <virtual_machine_id>	stops specified VM	Stop Virtual Server	"success"
startup <virtual_machine_id>	starts specified VM	Startup a Virtual Server	"success"
test	tests api connection	Test Route	"success"

## Inputs and values

To specify a VM parameter when creating or editing a VM use --<parameter> followed by the value. The flags -u <account\_id> -p <password> are required for all requests (unless they are defined in a config file):

```
/usr/local/flexCLI-1.1$ ./flexCLI.sh edit
<virtual_machine_id> -u <account_id> -p
<api_key> --label="newlabel"
```

You may also place them in a configuration file, included with --config="flexConfig.ini"

## Using a config file to store your credentials:

As an alternative to explicitly declaring your credentials with each request you can use config file to store them.

When working with multiple Flex Cloud accounts, you can create separate config files to store each account's information.

Bundled with the FlexCLI source you will find an example config file. Open it with your favorite editor replace these lines with your credentials:

## VS Disks

- Add New Disk
- Delete Disk
- Edit Disk
- Get VS Disks

```
user=<account_id>
password=<api_key>
```

should now look something like:

```
user=7654321
password=751baba7726a0227beba2e36bc08f012b52dee
6
```

then save and continue.

You can use the hash (#) to create comments.

## Testing your authentication/connection

The "test" command can be used to test your connection.

```
/usr/local/flexCLI-1.1$ ./flexCLI.sh test
--config="flexConfig.ini"
success
```

## An Example Request

This is an example showing how to edit a VM use --<parameter> followed by the value. Here we are changing the label to "newlabel":

```
/usr/local/flexCLI-1.1$ ./flexCLI.sh edit
<virtual_machine_id> -u <account_id> -p
<api_key> --label="newlabel"
```

Use "ls" to get a quick overview of your servers and their statuses

```
/usr/local/flexCLI-1.1$ ./flexCLI.sh ls
--config="flexConfig.ini"
|--id--|-----label-----|---ip-address---|
--memory--|-storage-|---status---|
| 511|          hostOmatic| 192.168.000.1|
500|          31|      running|
| 561|          emailServer| 192.168.000.1|
1500|          7|      building|
| 562|          superServe| 192.168.000.1|
384|          6|      running|
| 576|          serverTron| 192.168.000.1|
1548|          8|      running|
| 587|          powerServe| 192.168.000.1|
384|          6|      running|
| 589|          hosteria| 192.168.000.1|
512|          6|      shutdown|
```

Use `-f` to format the JSON output, making it more readable from the console.

```
/usr/local/flexCLI-1.1$ ./flexCLI.sh get
--config="flexConfig.ini" -f
[{"virtual_machine":{
    "add_to_marketplace":null,
    "admin_note":"Test Who?",
    "allowed_hot_migrate":true,
    "allowed_swap":true,
    "booted":true,
    "built":true,
    "cores_per_socket":0,
    "cpu_shares":1,
    "cpu_sockets":null,
    "cpu_threads":null,
    "cpu_units":10,
    "cpus":1,

"created_at":"2015-08-27T19:34:49+00:00",
    "customer_network_id":null,
    "deleted_at":null,
    "edge_server_type":null,
    "enable_autoscale":false,
    "enable_monitis":false,
    "firewall_notrack":false,
    "hostname":"lazar",
    "hypervisor_id":7,
    "id":511,
    "identifier":"w8zqm0m9nlf2ts",
    "initial_root_password":"123456Q",

"initial_root_password_encrypted":false,
    .....
```