Bellatrix Documentation

Release 1.1.9

Adrián Deccico

June 01, 2012

CONTENTS

Bellatrix is a set of (magic) command line tools to automate the management of Amazon Web Services.

CHAPTER

ONE

INTRODUCTION TO BELLATRIX COMMANDS

Bellatrix automates the interaction with EC2 Amazon Web Services. It uses the boto library and others. The goal of Bellatrix is to wrap the underlying libraries in easy to use utilities that are both simple and consistent.

Contents

1.1 Installing Bellatrix

If you already have pip installed, just execute:

```
pip install bellatrix
```

Another option is to go to: http://pypi.python.org/pypi/bellatrix/ download the .tar.gz package and then:

```
tar xvzf bellatrix-[version].tar.gz
cd belatrix-[version]
python setup.py   #this will require admin access
```

1.2 Setting your AWS credentials

Bellatrix only needs three single files in order to help you access your AWS EC2 resources. Not all the tools need all the files. If they are not present Bellatrix will show a nice message explaining what file (and where) you need to

provide.

- **Access key id**
  - Location <your_home>/.bellatrix/**key**:

```
Your 'access key id' will be something like AKIAIU**************) and is part of your AW
If you don't remember it please sign into:
https://aws-portal.amazon.com/gp/aws/developer/account/index.html?action=access-key
```

- **Secret file**
  - Location <your_home>/.bellatrix/**secret**:

```
The file should contain your 'secret access key' (a string with an approximate length of
It is part of your AWS security credentials.
In order to get your secret file please sign into:
https://aws-portal.amazon.com/gp/aws/developer/account/index.html?action=access-key
```

- **Private key**
  - Location <your_home>/.bellatrix/**ec2.pk**:

```
This file is the private key to use in order to connect to your instance.
You need to specify before a key-pair in your AWS account. For more details, please refe
http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/generating-a-keypair.html
The private key file is only used by the 'provision' and 'bewitch' commands.
```

1.3 Displaying your EC2 instances

Since running EC2 instances cost you money, this command will print information about your instances (running or not) and a useful summary at the very end. You only need to execute:

```
bellatrix list
```

The command will display an output similar to this:

```
adrian@adrian-packard-bell:~/b$ bellatrix list
2012-04-01 12:26:02,891 INFO listing...
2012-04-01 12:26:02,913 INFO Getting instances information... (this operation usually takes some seco
Instance:i-d67a61b2 | architecture:x86_64 | dns_name: | hypervisor:xen | image_id:ami-6fa27506 | inst

Instance:i-dc756eb8 | architecture:x86_64 | dns_name:ec2-50-19-190-98.compute-1.amazonaws.com | hyper

Running instances (you pay for them):1
Total instances:2
```

Please remember that you only pay for your **running** instances. Your **stopped** instances will generate a very small cost due to the use of an EBS disk. So far the cost is equal to $0.10 per allocated GB per month (http://aws.amazon.com/ebs/).

1.4 Starting an EC2 instance

In order to start an instance, just type:

```
bellatrix start ami key_name
```

The complete usage, with optional parameters is:

```
bellatrix start [--security_groups [SECURITY_GROUPS]] [--type [type]] [--new_size [NEW_SIZE]] ami key
```

Once this command has finished, it will generate an output file called **start_instance_out** with the instance code and the dns name so you can access it in a browser or through ssh. The format will be similar to this line:

```
i-6d6a1d09,ec2-50-17-116-43.compute-1.amazonaws.com
```

This file can be used to "chain" commands together.

Parameters list

- **ami - Amazon Machine Image. A pre-configured operating system with its applications.**
  - **The code will be something like ami-6ba27502. You can access a large set of AMI's in:**
    - https://aws.amazon.com/amis
    - http://cloud.ubuntu.com/ami/
    - http://thecloudmarket.com/
  - Or you can easily generate your own AMI using Bellatrix's commands.
- **key_name - Name of the ssh key-pair name that will be applied to your instance.**
  - The key name is the name of the two files (public and private keys) that you can use to connect to your EC2 instance.
  - AWS will put the public key file in the instance while you need to use the private key file to connect to your instance:

    ```
    ssh -i private_key_file user@public_dns
    ```

  - In order to generate your 'key name', please refer to: http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/ge a-keypair.html
  - If you are starting a Windows AMI then you would normally use RDP instead of this ssh key.
- **[optional] –security_groups [SECURITY_GROUPS]**
  - Comma separated list (with no spaces) of the security groups that will be applied to the new instance.
  - It can be only one. By default it will be "default"
- **[optional] –type type.**
  - Instance type. The same AMI can be launched with different 'hardware' options.
  - **You can choose between:**
    - m1.small,m1.medium,m1.large,m1.xlarge,t1.micro,m2.xlarge,m2.2xlarge,m2.4xlarge,c1.medium,c1.xlarge,cc1.4
    - By default you will get t1.micro.
  - Please take a look at: http://aws.amazon.com/ec2/instance-types for more information.
- **[optional] –new_size NEW_SIZE (in giga bytes).**
  - An EBS AMI can be started with a larger size just by using this option. If you then save the instance into a new AMI then this will be the disk size of the AMI.

– Providing a larger size doesn't mean you will be able to access it straight away. If the file system is ext4, then you are done. If not, you will need to execute one of this commands:

```
# ext3 root file system (most common)
sudo resize2fs /dev/sda1
#(OR)
sudo resize2fs /dev/xvda1

# XFS root file system (less common):
sudo apt-get update && sudo apt-get install -y xfsprogs
sudo xfs_growfs /

# In the case of Windows, you can use the graphical administration tools.
```

1.5 Provisioning an EC2 instance or any host

Provision an instance means you will execute a set of commands on it. Typically in order to apply some configuration. Your set of commands can be anything you want, even the execution of a Puppet script ;) Bellatrix provides a large set of ready to use commands but it is very easy to use your own. As a suggestion if you are adding a new command, you may want to make it idempotent, which means executing it once should have the same effect that multiple executions, for example using 'mkdir -p dir' instead of 'mkdir dir'.

The **provision** command is generic and can be used with any host, EC2 instances or not. Windows machines with an SSH server can be used too.

Usage example:

```
bellatrix provision [--private_key [PRIVATE_KEY]] configuration user hostname
```

Parameters list

- configuration - Python configuration file. E.g. **ubuntu.py** This is how a configuration command looks like:

```
"""
Simple example of a configuration file for the provisioning command.
"""

#commands library from Bellatrix
#The source file can be found here:     https://bitbucket.org/adeccico/bellatrix/src/tip/bellatr
# and the documentation here: http://bellatrix.readthedocs.org/en/latest/source/ref/bellatrix.li
from bellatrix.lib import cmds
commands = cmds.apt_get_update()
commands = cmds.install_pip()
commands += cmds.pip_install("virtualenv")
commands += ['echo "Adding my own command :)" > test', 'cat test']
```

The previous example can be found here: https://bitbucket.org/deccico/bellatrix_configs/src/tip/bellatrix_configs/simple_provision_test.p
Another, more complex example of a configuration file can be found here: https://bitbucket.org/deccico/bellatrix_configs/src/tip/bellatrix_configs/ubuntu_django_nginx_gunicorn.py Here is the commands documentation: http://bellatrix.readthedocs.org/en/latest/source/ref/bellatrix.lib.html#module-bellatrix.lib.cmds

- user - User used to connect to the host. E.g. **ubuntu**
- hostname - Hostname or simply the ip of the machine.

- [optional] –private_key PRIVATE_KEY - In case we need to specify a private key to connect to the host. This is empty by default. If you are using an EC2 instance you will typically use the default private key located in ~/.bellatrix/ec2.pk

1.6 Saving the state of an instance into a new Amazon AMI

If you want to capture the current state of your EC2 instance into a new AMI, you just need to call this command:

```
bellatrix burn [--wait [{true,false}] instance image_name
```

Parameters list

- instance - Instance name. Something like: "i-b63c98d4". The instance should be running when you invoke this command.
- image_name - This will be the name of your AMI. A time stamp will be added, so you can apply the same name and more importantly, identify when each version was generated.
- **–wait [true, false]**
  - This is **false** by default.
  - Burning a new image usually takes some minutes. If you don't use this option (or you set it to false) this command will show you the AMI code being burned and then finish immediately, but if you use "–wait=true" the **burn** command will finish only when the AMI is ready to be used.

1.7 Copying files to a S3 bucket

This command will copy a file or directory to a S3 bucket. You can imagine S3 as an encripted 'infinite' disk in the AWS cloud. Your files and directory structure will be put into a bucket that you need to create first. After you copy your files you can access them in: https://s3.amazonaws.com/your_bucket This is how you use this command:

```
bellatrix copy2s3 source bucket [key_prefix] [{private,public-read,public-read-write,authenticated-re
```

Paramaters list

- source - Source file or directory in your computer.
- bucket - S3 bucket destination. Please remember to create it first. A bucket needs to be unique.
- key_prefix - This prefix will be added to the source path we copy. It is blank by default.
- **{private,public-read,public-read-write,authenticated-read}**
  - With this option you control who can access your files in the S3 bucket. If you don't specify anything the policy will be **private** by default.

1.8 Setting launch permissions to an AMI

If you choose your AMI to be private (option by default) you can still give permissions to other accounts so they can access it. In order to do so, just execute this command:

```
bellatrix perm2ami ami permissions_file
```

Paramaters list

- ami - AMI name. Something like ami-6ba27502
- permissions_file. Text file with an account number (12 digits number without dashes) on each line.

1.9 Stopping an EC2 instance

With this command, you can stop a given instance or all of them if you pass the "all" argument. Stopping an instance will shut-down the instance but will preserve data on the EBS volume. You won't pay for the more expensive computing capacity. The only charge after you stop an instance is $0.10 per allocated GB per month (http://aws.amazon.com/ebs/). After you **stop** an instance, you can **start** it and access to the same data:

```
bellatrix stop [-h] instance
```

Paramaters list

- instance - Instance id. Something like i-39e2075d. If you pass **all** then all instances will be stopped.

1.10 Terminating an EC2 instance

Terminate a given instance or all of them if you pass the **all** parameter. Terminating an instance will shut it down and delete the data on the EBS volume. Your instance/s won't produce any cost after you terminate them.:

```
bellatrix terminate [-h] instance
```

Paramaters list

- instance - Instance id. Something like i-39e2075d. If you pass **all** then all running instances will be terminated (unless they are explicitly protected by the "disable termination" flag.)

1.11 Bewitching an AMI or how to start, provision and burn with a single command

This is a macro-command, that will:

- Start a new instance,
- apply a configuration (**provision** command) to it
- and if all the commands were successful, it will save the instance into a new ami (**burn** command).

This is how you use it:

```
bellatrix bewitch [-h] configuration
```

Paramaters list

- **configuration - Python configuration file. E.g. ubuntu.py. It is just a simple Python file with some attributes on it like:**
  - **amis - This is the list of ami's that will be executed.**

    * Something like:

      ```
      amis = [
             ["ami-fd589594",  "ubuntu1104-ff36-mysql51-x64"],
             ]
      ```

    * As you can see we define in the sub-list the code, and the configuration name of the new AMI.

– **user - This is the user that is going to be used in order to connect to the instance. In the case of Ubuntu AMIs it v**

    * Example:

      ```
      user = "ubuntu"
      ```

– key_name - Name of the key-pair name that Bellatrix will specify when launching the new instance:

  ```
  key_name = "key"
  ```

– security_groups - Comma separated list (with no spaces) of the security groups that will be applied to the new instance:

  ```
  security_groups = "default"
  ```

– instance_type - Type of instance that Bellatrix will request to AWS. Usually **t1.micro** is enough (not to mention cheap) just for applying some commands. If you need more power, you can find the one that suits you better here: http://aws.amazon.com/ec2/instance-types:

  ```
  instance_type = "t1.micro"
  ```

– **cmds - The list of command that are going to be executed in order to provision the new instance.**

    * **cmds** is just a list of strings. Each element needs to be a regular command that is going to be executed in order.

    * **Bellatrix already provides a set of handy commands ready to use.**

        · The list can be found here: https://bitbucket.org/adeccico/bellatrix/src/tip/bellatrix/lib/cmds.py

        · While the documentation is here: http://bellatrix.readthedocs.org/en/latest/source/ref/bellatrix.lib.html#modu bellatrix.lib.cmds

    * Usage example:

      ```
      from bellatrix.lib import cmds
      commands = cmds.apt_get_update()
      commands = cmds.install_pip()
      commands += cmds.pip_install("virtualenv")
      commands += ['echo "Adding my own command :)" > test', 'cat test']
      ```

Here is an example of a complete configuration file that manages to provision Django, GUnicorn, NGnix and Upstart into a blank Ubuntu AMI.

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *ref-index*
- *search*

CHAPTER

THREE

GENERAL INFORMATION

- Getting Started: http://bellatrix.readthedocs.org/en/latest/commands_use_tut.html
- Main documentation: http://bellatrix.readthedocs.org
- Bugs? Feature request? https://bitbucket.org/adeccico/bellatrix/issues
- Source code: https://bitbucket.org/adeccico/bellatrix
- Configuration examples: https://bitbucket.org/deccico/bellatrix_configs
- New releases can be found here: http://pypi.python.org/pypi/bellatrix
- Contact information: deccico[at]gmail.com

CHAPTER

FOUR

PROVISIONING EXAMPLE

- Installing Django, GUnicorn and Nginx into a blank Ubuntu AMI: https://bitbucket.org/deccico/bellatrix_configs/src/tip/bellatrix_configs/ubuntu_django_nginx_gunicorn.py