AppScale Tools Documentation

Release 1.0

AppScale

February 14, 2014

Contents

1 Getting Started **3**

1.1 Installing VirtualBox . . . 3

1.2 Installing Vagrant . . . 3

1.3 Clone the AppScale Tools repository . . . 3

1.4 Configure AWS credentials . . . 4

1.5 Launch a development VM . . . 4

1.6 Confirm EC2 Tools are working . . . 4

1.7 Create an EC2 SSH key . . . 5

2 First Run **7**

2.1 Create an AppScale Cluster . . . 7

3 Development **9**

4 Packaging **11**

5 Indices and tables **13**

Contents:

CHAPTER 1

Getting Started

The quickest way to get started using AppScale Tools is using VirtualBox and Vagrant.

Note, while the process below should work on any system that VirtualBox and Vagrant run on, this documentation was written while following the steps below on a Mac OS X system.

1.1 Installing VirtualBox

Please visit the VirtualBox website for more details.

1.2 Installing Vagrant

Make sure you have Ruby and Rubygems support. Then, from the command line, run:

```
gem install vagrant
```

1.2.1 Downloading the base box

Vagrant uses a base box to create VMs. Run the command below to download an Ubuntu 10.04 Lucid 64-bit box:

```
vagrant box add lucid64 http://files.vagrantup.com/lucid64.box
```

1.3 Clone the AppScale Tools repository

Vagrant is configured to automatically mount `~/Appscale` as `/srv/appscale` on the VM, and look for the `appscale-tools` repo at `/srv/appscale/repo/appscale-tools`. At the moment there are some limitations with the Puppet manifest script that require hard-coded absolute paths.

Clone the repository in `~/Appscale/repo` on your computer:

```
mkdir -p ~/Appscale/repo
cd ~/Appscale/repo
git clone https://github.com/AppScale/appscale-tools.git
cd appscale-tools
```

Currently this documentation reflects the changes in the `auto_appscale_install` branch, so switch to that branch:

```
git checkout auto_appscale_install
```

1.4 Configure AWS credentials

Create the AppScale Tools configuration directory:

```
mkdir ~/.appscale-tools
```

Next, create the file `~/.appscale-tools/appscale_config.sh` with the following content (filling in your personal information, of course. Do not include curly braces):

```
export EC2_ACCESS_KEY='{{ your key here }}'
export EC2_SECRET_KEY='{{ your key here }}'
export EC2_USER_ID='{{ your id here }}'
export EC2_MY_SSH_KEY=~/.appscale-tools/id_{{ your username }}
```

Finally, download your account's certificate and private key, or ask your administrator for IAM credentials, and copy the AWS certificate and private key:

```
cp <path to cert-*.pem> ~/.appscale-tools/cert.pem
cp <path to pk-*.pem> ~/.appscale-tools/pk.pem
```

1.5 Launch a development VM

Once the prerequisites are installed and credentials configured, run the command `vagrant up` to setup the environment, followed by `vagrant ssh` to connect to the VM. The repository is mounted inside the VM at `/srv/appscale/repo/appscale-tools`.

1.5.1 The `repo` command

AppScale Tools includes a `repo` command which makes it easy to switch to repositories located in `/srv/appscale/repo`. Type `repo`, followed by a space, and enter the `tab` key. You will then see a list of available repositories in the `/srv/appscale/repo` directory. Please see the Wikipedia article on tab completion for more information.

1.6 Confirm EC2 Tools are working

Once you SSH into the VM (`vagrant ssh`), confirm your personal information is displayed when using the command:

```
echo $EC2_ACCESS_KEY
```

Confirm EC2 Tools are working by running the command:

```
ec2-describe-instances
```

1.7 Create an EC2 SSH key

With a working EC2 environment, create an SSH key that will be used to SSH into EC2 instances. Run the following commands substituting your name below:

```
ec2-add-keypair <your name> >> ~/.appscale-tools/id_<your username>
chmod 0600 ~/.appscale-tools/id_<your username>
```

For example:

```
ec2-add-keypair berto >> ~/.appscale-tools/id_berto
chmod 0600 ~/.appscale-tools/id_berto
```

Your system is now configured! Note that even if you destroy this VM, you will not need to re-configure the system. These configuration files are kept in your host machine's home directory under `~/.appscale-tools`. You can run `vagrant destroy`, followed by `vagrant up` and you'll be ready to run instantly.

CHAPTER 2

First Run

With a configured EC2 Tools environment, now it's time to create an AppScale image.

If not already there, SSH into the Vagrant box:

```
vagrant ssh
```

Then go into the appscale-tools repository:

```
cd /srv/appscale/repo/appscale-tools
```

From here you can bootstrap the image creation process using the command:

```
./bin/appscale-bootstrap
```

For more information on this command simply run it with the `-h` flag:

```
./bin/appscale-bootstrap -h
```

This command will take a while to run. Once it's complete, you will see in the last few lines of the log the instance id and the image id that was created.

NOTE: The bootstrap script does not currently terminate the instance that is created in order to make an AppScale image. Please terminate it once you are done.

2.1 Create an AppScale Cluster

`appscale-run-instances` [with flags] - fires up a cluster

CHAPTER 3

Development

There are a couple tools configured for developing AppScale Tools with Python: `nose` and `pylint`. Nose is a test runner that is invoked using the `nosetests` command. Nose will automatically find and run any tests it can find. Initial tests have been placed in the `tests` directory.

Pylint is a "linter", which helps keep the code free of syntax and formatting convention errors. Before pushing code upstream, make sure to run pylint in order to catch errors and keep code consistent.

Pylint has been re-configured with the following changes from the default:

- the messages in the report include their respective IDs. they are useful for configuring pylint
- the max line width was changed from 80 to 120.
- the string module is not in the deprecated list because there are useful functions in there not anywhere else.
- there is a `lint` package in the project's root that contains a module that defines a helper module for defining objects in the `sh` module that are used in the code, but are dynamic so pylint, by default, counts them as errors.

To make it easier to run there is a `make` target: pylint. Run it like so to lint the `appscaletools` package:

```
make pylint
```

Or to run it on another file, like `bin/appscale-boostrap`:

```
make pylint f=bin/appscale-tools
```

CHAPTER 4

Packaging

AppScale Tools is designed to run on Ubuntu 10.04 LTS. To build a .deb package, simply run the command `make` from the project root.

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*