em User Manual

Table of Contents

Introduction 3

Installation 3

Pre-Requisites 3

Java SDK 3

Amazon EC2 Command Line Interface Tools 3

Set the Amazon EC2 API Access Keys 4

Set your Amazon EC2 Region 5

Instance Access Key Pairs 5

em Installation 7

Download and Extract the Installation Archive 7

Setup the Environment Variables 7

Install the Instance Provisioning Private Key File 8

Test the Installation 8

Building a Basic Environment 10

Basic Environment Overview 10

The NFS File Server 10

Other Instances 10

Building the NFS Server 11

Building a Basic Instance 12

Starting a Basic Instance 14

Logging Into a Basic Instance 14

Stopping a Basic Instance 14

Terminating a Basic Instance 14

Building a Java-enabled Instance 15

Building an AMQ Cluster 16

Testing the Installation 16

Building a JBoss EAP Instance 17

Running Arbitrary Shell Commands Across the Environment 18

Miscellaneous 19

# Introduction

em (Environment Manager) is a set of utilities for creating and using environments in Amazon EC2.

# 

# Installation

## Pre-Requisites

### Java SDK

em requires access to Java SDK, version 1.7.

Download and install the latest JDK 1.7 following instructions specific to your system from Oracle web site: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Once JDK has been installed on your system, export JAVA\_HOME in your environment and add ${JAVA\_HOME}/bin to your PATH.

### Amazon EC2 Command Line Interface Tools

em requires access to Amazon EC2 command line interface tools, version1.7.3.0 or higher. Amazon EC2 command line interface tools can be downloaded from Amazon web site: <http://aws.amazon.com/developertools/351>.

Installation instructions are available here: <http://docs.aws.amazon.com/AWSEC2/latest/CommandLineReference/set-up-ec2-cli-linux.html>

After installation, Amazon EC2 command line interface tools must be made available to em. Set the EC2\_HOME environment variable to point to the installation directory and adjust the PATH environment variable as follows:

PATH=${PATH}:${EC2\_HOME}/bin

After reloading your environment, you can check whether the installation was performed correctly by executing:

ec2-version

The command should return 1.7.3.0 or higher.

### Set the Amazon EC2 API Access Keys

Each Amazon EC2 IAM user has a set of API access keys. These keys are needed when the user attempts to make programmatic calls to AWS or EC2, using Amazon EC2 CLI tools. The user can create, modify, view and rotate these access keys.

Provision the access keys for your account. Once provisioned, set them in the environment as follows:

export AWS\_ACCESS\_KEY=your-aws-access-key-id

export AWS\_SECRET\_KEY=your-aws-secret-key

Test whether the access key installation worked by executing:

ec2-describe-regions

The command should output something similar to:

nombp1:doc ovidiu$ ec2-describe-regions

REGION eu-central-1 ec2.eu-central-1.amazonaws.com

REGION sa-east-1 ec2.sa-east-1.amazonaws.com

REGION ap-northeast-1 ec2.ap-northeast-1.amazonaws.com

REGION eu-west-1 ec2.eu-west-1.amazonaws.com

REGION us-east-1 ec2.us-east-1.amazonaws.com

REGION us-west-1 ec2.us-west-1.amazonaws.com

REGION us-west-2 ec2.us-west-2.amazonaws.com

REGION ap-southeast-2 ec2.ap-southeast-2.amazonaws.com

REGION ap-southeast-1 ec2.ap-southeast-1.amazonaws.com

### Set your Amazon EC2 Region

Pick the appropriate Amazon EC2 region from the list returned by the ec2-describe-regions command and set the following environment variable:

export EC2\_URL=https://<service\_endpoint>

where the <service\_endpoint> value should come from the third column of the ec2-describe-regions command output.

Example:

export EC2\_URL=https://ec2.us-west-2.amazonaws.com

### Instance Access Key Pairs

Amazon AWS uses public-key cryptography to secure the login to instances. The instance has no password - you use a key pair to access your instance securely. The key pairs are provisioned either via the web interface or with Amazon CLI tools. During the provisioning process, the keys pairs are named.

When the instance is created, you need to specify the name of the key pair to use to protect access to it: Amazon will install the public certificate in ~/.ssh/authorized\_keys when creating the instance, and then you need to provide the private key of the pair to access the instance.

em uses at least two key pairs for access: one key pair for access during instance provisioning and one or more key pairs for routine access.

#### Instance Provisioning Key Pair

You need to create and register with the Amazon EC2 management facilities a provisioning key pair.

This key pair will be used for access every time a new instance is created and configured. Once the configuration (initial overlay) phase is over, the key is removed and you are free to install as many regular access keys as you wish.

We recommend you name of the installation provisioning key pair “em-provisioning-key-pair”, but this is not required.

Details on how to create key pairs are available here: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>

#### Instance Access Key Pairs

You must install at least one instance access key pair.

**IMPORTANT:** If you don’t register at least one instance access key pair, the instance will be inaccessible for login, because em removes temporary provisioning access credentials at the end of the instance provisioning process.

Is not required to register with Amazon EC2 management facilities your regular instance access keys. em will install and manage them.

## em Installation

### Download and Extract the Installation Archive

Download the latest release from <https://github.com/NovaOrdis/em/releases>

You will get a zip file that contains everything em needs in order to build and interact with Amazon EC2 environments.

Extract the content of the em installation zip into the directory conventionally used for external utilities. /opt or /usr/local are common choices. For the remainder of this document we will assume that the external utility directory is /opt.

TODO: put in place the install script that can be used to 1) install em locally 2) install -–zip-only. Currently I am creating the zip by hand. Once the procedure is established, update the Development Manual.

The extraction process will create an “em-<version>” top-level directory.

Assuming that the top level directory is em-2.5, link to it with a generic link “em”, as follows:

cd /opt

ln –s./em-2.5 ./em

### Setup the Environment Variables

Set the EM\_HOME environment variable to point to the generic “em” link. This way, you will be able to upgrade by simply unzipping a new version and re-linking, without any environment modification.

Also adjust the PATH environment variable as follows:

export EM\_HOME=/opt/em  
export PATH=${PATH}:${EM\_HOME}/bin:~/.em/bin

~/.em/bin is the directory where em creates “direct login” links, where executing a remote environment instance name as a command logs the user into that instance.

### Install the Instance Provisioning Private Key File

The Amazon EC2 instance provisioning key creation process generates a private key file in PEM format that is automatically downloaded by your. For more details, see “Instance Provisioning Key Pair”.

You will need to make this key accessible to em.

Conventionally, the private key file generated by Amazon is moved under a new name (em-provisioning-private-key.pem) under the em account ~/.ssh directory and the appropriate permissions are set as follows:

mv <browser\_download\_dir>/em-provisioning-key-pair.pem \

~/.ssh/em-provisioning-private-key.pem

chmod go-rwx ~/.ssh/em-provisioning-private-key.pem

For more details about how em uses the access keys, see “Instance Access Key Pairs” section.

### Test the Installation

Reload your environment to make sure the path is updated, and then execute:

em version

and then:

em status

If no instance were previously created with em, the command should still succeed and produce something similar with:

If the environment was previously used, you will get the list of instances already created in the environment:

Run and find out

em status

name state id public-ip private-ip

f01 stopped i-db0ab82d 172.31.25.44

b01 stopped i-55397ea3 172.31.16.215

b02 stopped i-fe014408 172.31.30.67

# Building a Basic Environment

## Basic Environment Overview

### The NFS File Server

A basic environment consists in an NFS fileserver, which will serve as environment configuration keeper and shared file keeper, and other instances.

At the time of this writing, any environment *needs* a file server. In the future, we may add support for HTTP-only based environments.

The fileserver does not need significant resources. We routinely use reasonably large environments – tens of instances – served by a t2.micro NFS server, provided that the client instances prefer “local” storage to the shared file space.

The shared file space should only be used for installation kits and configuration, and it should be mostly read by the client instances.

The NFS file server should be kept around (i.e. not terminated) for the life of the environment. It is the only non-expendable – and this just for the useful life of the environment – instance of the environment. Provided that key state on its file system has been backed up, the NFS file server can be terminated, the restored, though we don’t recommend this approach.

### Other Instances

An environment can have an arbitrary number of expendable instances, which can be created and then terminated arbitrarily.

The typical use for “ephemeral environments” is large instance-count load tests. An environment comprising hundreds of instances can be created, and then load can be generated and applied within the environment, data collected and finally, all instances discarded.

Provided that Amazon EC2 bills for storage space even if the instance is stopped, and instances need at least some local storage, this will result in significant bill savings. If you don’t care about this part, environments can be kept around in stopped state – or even in running state – for as long as you wish.

## Building the NFS Server

em create nfs01

Conventionally, the instances of an environment are named using letters and digits that give a hint regarding their usage. Following this convention, the only NFS file server of an environment could be named “nfs01”.

To discover more options available to you when running the “create” command, execute:

em –h create

ec2-run-instances ami-4dbf9e7d --instance-count 1 --key installation\_access --group sg-bb3222de --instance-type t2.micro --availability-zone us-west-2b --tenancy default --subnet subnet-53993c24 --instance-initiated-shutdown-behavior stop --associate-public-ip-address true --block-device-mapping /dev/sda1=:10

A few seconds after creation, the NFS server non-configured instance should be available in the “em status” query:

nombp1:~ ovidiu$ em status

name state id public-ip private-ip

...

nfs01 running i-1086b9e6 52.25.189.47 172.31.16.246

...

At this point, the instance is not yet accessible for log in by the end-users, because it was configured by the Amazon EC2 runtime with a key that can be only used for installation. To be made accessible, it has to be configured as an NFS server:

em overlay nfs-server

When this step completes successfully, the NFS server should be available for login:

em sync

nfs01

The authenticity of host '52.25.188.17 (52.25.188.17)' can't be established.

RSA key fingerprint is 9e:96:48:53:93:c8:0d:41:c3:dc:69:84:4a:7c:32:6c.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '52.25.188.17' (RSA) to the list of known hosts.

Last login: Mon Jun 8 13:00:55 2015 from 207.114.215.130

ec2-user@nfs01>

All logins are executed by default as ec2-user.

## Building a Basic Instance

That the environment’s NFS server must be up and running when a new instance is being built.

em create test01 --type --security

em --dry-run create test01 --type --security

em sync

em overlay basic test01

Successful basic configuration output:

em overlay basic test01

applying overlay basic to test01 ...

The authenticity of host '52.24.230.169 (52.24.230.169)' can't be established.

RSA key fingerprint is ec:67:2b:27:bf:aa:f0:48:e7:2f:09:a5:a9:13:7c:e1.

Are you sure you want to continue connecting (yes/no)? yes

hostname successfully set to test01

installing package zip ...

installing package unzip ...

cursor configured in /root/.bashrc

alias h='history' configured in /root/.bashrc

alias r='sudo su -' configured in /root/.bashrc

alias nfs='cd /nfs' configured in /root/.bashrc

environment configured for ec2-api-tools

cursor configured in /home/ec2-user/.bashrc

alias h='history' configured in /home/ec2-user/.bashrc

alias r='sudo su -' configured in /home/ec2-user/.bashrc

alias nfs='cd /nfs' configured in /home/ec2-user/.bashrc

environment configured for ec2-api-tools

The authenticity of host '172.31.25.44 (172.31.25.44)' can't be established.

ECDSA key fingerprint is e5:80:d2:65:23:4a:fd:7e:b7:b7:24:4f:aa:01:b8:e1.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '172.31.25.44' (ECDSA) to the list of known hosts.

NFS server identity successfully configured

installing package nfs-utils ...

created NFS mount point /nfs

NFS client successfully configured

installed ec2-user-environment-identity key into ec2-user's authorized\_keys file

installed ovidu\_default key into ec2-user's authorized\_keys file

removed requiretty from /etc/sudoers

Connection to 52.24.230.169 closed.

## Starting a Basic Instance

em start test01

## Logging Into a Basic Instance

em sync

test01

The authenticity of host '52.24.69.9 (52.24.69.9)' can't be established.

RSA key fingerprint is 9e:96:48:53:93:c8:0d:41:c3:dc:69:84:4a:7c:32:6c.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '52.24.69.9' (RSA) to the list of known hosts.

Last login: Sat Jun 13 00:25:05 2015 from 104-50-201-81.lightspeed.sntcca.sbcglobal.net

## Stopping a Basic Instance

em stop test01

## Terminating a Basic Instance

em terminate test01

Note that you don’t need to stop an instance to terminate it. An instance can be terminated directly from a running state.

# Building a Java-enabled Instance

A java-enabled instance can be built from scratch, or a basic instance can be upgraded to “java-enabled”. In both cases, all that is needed is to apply the “java” overlay (the “java” overlay depends on the “basic” overlay, so all configuration applied there is also applied to a “java” instance):

em overlay java test01

When creating the instance, make sure to allocate resources appropriate to the future load (memory, CPUs) by choosing the appropriate instance type.

Successful java configuration output:

em overlay java test01

applying overlay java to test01 ...

environment configured for ec2-api-tools

environment configured for ec2-api-tools

NFS client already setup

installing java ...

installed java into /opt/jdk1.7.0\_75

linked /opt/jdk1.7.0\_75 to /opt/java

java environment configured for user root

java environment configured for user ec2-user

Connection to 52.25.199.219 closed.

# Building an AMQ Cluster

Clustered JBoss AMQ brokers can be built by applying “amq-broker” overlay. Each EC2 instance will run a single broker instances, but those broker instances can be connected via network bridges that are automatically setup by the overlay.

When creating the instance, make sure to allocate resources appropriate to the future load (memory, CPUs) by choosing the appropriate instance type.

em create --instance-type c4.2xlarge \

--group sg-f8257e9d --storage-size 15 b01 b02

em sync

# update the amq-broker configuration (memory, sizes, and especially

# the cluster membership)

em overlay amq-broker b01 b02

# update /etc/hosts of all hosts that were created first

em stop

em start

The “amq-broker” overlay configures the instance to start the AMQ broker at boot .

## Testing the Installation

Start all instances and make sure the bridges connect to each other.

# Building a JBoss EAP Instance

Clustered JBoss EAP instances can be built by applying “jboss-eap” overlay. Each EC2 instance will run a single application server instances, but those application server instances can be clustered if you choose so.

# Running Arbitrary Shell Commands Across the Environment

em run p001 p002 p003 -- uptime

# Miscellaneous

em –h|--help <command>

em –v|--verbose <command>