# Deploying Your Application On Public Cloud

## With Egle Sigler and Iccha Sethi

**Egle Sigler** is a Principal Architect on a Private Cloud Solutions team at Rackspace. In addition to working with OpenStack and related technologies, Egle is a governing board member for POWER (Professional Organization of Women Empowered at Rackspace), Rackspace's internal employee resource group dedicated to empowering women in technology. In her spare time, Egle enjoys traveling, hiking, snorkeling, and nature photography. Egle holds a master's degree in Computer Science from a University of Texas at San Antonio.
Blog: http://anystacker.com/
Twitter: @eglute

**Iccha Sethi** is a long time core contributor to OpenStack, open source cloud computing project, and has worked on the Cloud Images (Glance) and Cloud Databases (Trove) OpenStack products at Rackspace. She has been involved in several community initiatives including being a mentor for GNOME OPW program and is the founder of Let's Code Blacksburg!

Blog: http://www.icchasethi.com/
Twitter: @IcchaSethi

**All workshop materials and videos will be posted here: http://www.icchasethi.com/ and here: http://anystacker.com/**

## Cloud Vocabulary

**Virtual Machine (VM)**: is an emulation of a particular computer system. Cloud servers are usually virtual machines.
**Cloud Server**: is a compute resource accessible via network, usually a virtual resource or a virtual machine.
**Snapshot**: usually refers to a copy at a particular time of a virtual machine. Different virtualization software implement it differently.
**Image**: a static copy of a virtual machine. Can be used to create a new virtual machine from it or used as a backup.
**Clone**: virtual machine copy, usually used in VMware technology.
**Template (VMware)**: Virtual machine template, very similar to clone.
**Template (Other)**: a way to deploy a virtual machine with software installed and configured.

**NIST (National Institute of Standards and Technology)** provides the following definitions of cloud computing:

**Cloud computing** is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

**Service Models:**

*Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user- specific application configuration settings.

*Platform as a Service (PaaS)*. The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

*Infrastructure as a Service (IaaS).* The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

**Deployment Models:**

*Private cloud.* The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

*Community cloud.* The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

*Public cloud.* The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

*Hybrid cloud*. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

Full publication can be found here: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

## Deploying IPython Notebook Using Orchestration In Rackspace Cloud

1. Log into mycloud.rackspace.com
2. Click on Orchestration
3. Select create a stack
4. Choose a name for your stack
5. Choose the IPython notebook template. Feel free to edit the settings in the template.
6. Click next
7. Choose the OS and size for your server.
8. Click on create stack and wait for your deployment to complete.
9. Use your Ipython Url and password displayed in the control panel to access your IPython notebook!

## Deploying WordPress on EC2 Amazon Cloud

### *Launching an EC2 Instance*

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

2. From the console dashboard, click **Launch Instance**.

3. The **Choose an Amazon Machine Image (AMI)** page displays a list of basic configurations called Amazon Machine Images (AMIs) that serve as templates for your instance. Select the 64-bit Amazon Linux AMI. Notice that this configuration is marked "Free tier eligible."

4. On the **Choose an Instance Type** page, you can select the hardware configuration of your instance. The t2.micro instance type is selected by default. Alternatively, select **All generations** from the filter list, and then select the t1.micro instance type. Note that these are the only instance types eligible for the free tier.

   **Note**

   T2 instances must be launched into a VPC. If your AWS account supports EC2-Classic and you do not have any VPCs, the launch wizard creates a VPC for you. Otherwise, if you have one or more VPCs, click **Next: Configure Instance Details** to select a VPC and subnet.

5. Click **Review and Launch** to let the wizard complete the other configuration settings for you.

6. On the **Review Instance Launch** page, under **Security Groups**, you'll see that the wizard created and selected a security group for you. Instead, select the security group that you created when getting set up using the following steps:

   a. Click **Edit security groups**.

   b. On the **Configure Security Group** page, ensure the **Select an existing security group** option is selected.

   c. Select your security group from the list of existing security groups, and click **Review and Launch**.

7. On the **Review Instance Launch** page, click **Launch**.

8. In the **Select an existing key pair or create a new key pair** dialog box, select **Choose an existing key pair**, then select the key pair you created when getting set up.Alternatively, you can create a new key pair. Select **Create a new key pair**, enter a name for the key pair, and then click **Download Key Pair**. This is the only chance for you to save the private key file, so be sure to download it. Save the private key file in a safe place. You'll need to provide the name of your key pair when you launch an instance and the corresponding private key each time you connect to the instance.A key pair enables you to connect to a Linux instance through SSH. Therefore, don't select the **Proceed without a key pair** option. If you launch your instance without a key pair, then you can't connect to it. When you are ready, select the acknowledgment check box, and then click **Launch Instances**.

9. A confirmation page lets you know that your instance is launching. Click **View Instances** to close the confirmation page and return to the console.

10. On the **Instances** screen, you can view the status of your instance. It takes a short time for an instance to launch. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running, and it receives a public DNS name. (If the **Public DNS** column is hidden, click the **Show/Hide** icon and select **Public DNS**.)

**Connect from Mac/Linux:**

Your Mac or Linux computer most likely includes an SSH client by default. You can check for an SSH client by typing **ssh** at the command line. If your computer doesn't recognize the command, the OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, seehttp://www.openssh.org.
Open your command shell and run the following command:

```
$ ssh -i /path/key_pair.pem ec2-user@public_dns_name
```

This documentation is provided by: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/LaunchingAndUsingInstances.html

## *Install Apache, PHP and MySQL and WordPress*

1. **Install Apache**
   a. yum install httpd
   b. service httpd restart
2. **Install PHP and other packages**
   a. yum install php libmcrypt libmcrypt-devel php-mcrypt php-mbstring
   b. service httpd restart
3. **Install MySQL**
   a. yum install mysql
   b. yum install –y mysql-server
   c. service mysqld start
4. **Set MySQL user and password**
   a. mysqladmin -u root password 'password'
5. **Install PHPMyAdmin**
   a. cd /var/www/html

   b. wget http://sourceforge.net/projects/phpmyadmin/files/phpMyAdmin/3.3.9.1/phpMyAdmin-3.3.9.1-all-languages.tar.gz
   c. tar -xzvf phpMyAdmin-3.3.9.1-all-languages.tar.gz -C /var/www/html
   d. mv phpMyAdmin-3.3.9.1-all-languages phpmyadmin
   e. rm -rf phpMyAdmin-3.3.9.1-all-languages.tar.gz
6. **Set appropriate user and permissions**
   a. chown -R phpmyadmin.apache phpmyadmin/
   b. cd /var/www/html/phpmyadmin/ mkdir config chmod o+rw config cp config.sample.inc.php config/config.inc.php chmod o+w config/config.inc.php
   c. mkdir config
   d. chmod o+rw config
   e. yum -y install php-mysql
   f. service httpd restart
7. **Set up server at:**
   a. http://YOUR_SERVER_IP/phpmyadmin/setup/index.php
8. **Set up database and user privileges at:**
   a. http://YOUR_SERVER_IP/phpmyadmin/index.php
9. **Installing WordPress:**
   a. cd /
   b. wget http://wordpress.org/latest.tar.gz
   c. tar -xzvf latest.tar.gz -C /var/www/html
   d. cp -avr /var/www/html/wordpress/* /var/www/html
   e. rm -rf /var/www/html/wordpress
   f. chown -R apache /var/www/html
   g. chmod -R 755 /var/www/html
   h. service httpd restart
10. **Install WordPress plugin:**
    a. Go to http://YOUR_SERVER_IP/ and fill in your database credentials and follow instructions.
11. **Viola! Your WordPress site is up!**

## Deploying Applications to Cloud Foundry

Cloud Foundry is Platform as a Service. It provides runtimes for common development environments such as Rails, Java, NodeJS, Go, Python, PHP, and others via "buildpacks". Application developer that wants to deploy an application to the cloud needs only to prepare her application to be deployed, create a deployment manifest (http://docs.cloudfoundry.org/devguide/deploy-apps/manifest.html) and simply use command line client to "push" it to the PaaS.

To get started, point Cloud Foundry command line client (CLI) to the Cloud Foundry service:

```
MFC89CDF93:spring-music egle$ cf login -a api.run.pivotal.io
API endpoint: api.run.pivotal.io
Email> eglesi@gmail.com
Password>
```

Check out a sample app:

```
git clone https://github.com/cloudfoundry-samples/spring-music
Build the sample application with a provided gradle script:
$ ./gradlew assemble
```

Once the application is built, it can be pushed to Cloud Foundry:

```
$ cf push
```

Once application is deployed, a URL will be provided:

```
urls: grace-hopper-2014.cfapps.io
     state     since                    cpu    memory          disk
#0   running   2014-10-05 02:59:41 AM   0.0%   430.7M of 512M   131M of 1G
```

Copy the URL and open it in the browser. Your application is up and running!

**Useful links:**

Cloud Foundry documentation: http://docs.cloudfoundry.org/
Cloud Foundry sample applications: https://github.com/cloudfoundry-samples
Cloud Foundry by Pivotal: https://run.pivotal.io/
Cloud Foundry by IBM, BlueMix: https://bluemix.net/
Download Cloud Foundry client: https://github.com/cloudfoundry/cli#downloads