

## Introduction to Amazon Web Services – a LITA 2011 Preconference

This tutorial takes you through the process of selecting, starting, customizing and managing a server on the Amazon EC2 platform. The goal of this tutorial is to give you hands-on experience with cloud computing using "infrastructure" level services. This means that we will be working with raw computing resources - Disks, cpu/memory, server software. The secondary goal of this worksheet is to take workshop participants through the processes that are typical of server administration (e.g. select, install, configure, backup, monitor, restore). In doing this, participants will have a solid grounding in technical expertise that will allow you to examine cloud computing issues in more depth.

The following table of contents shows each step in the service configuration process from getting registered to removing your server and disk resources. This tutorial only explores a small portion of the Amazon cloud system and does not include other providers however, with a base knowledge of what these resources can help you do you will be able to examine other options.

<b>1. REGISTER FOR AMAZON WEB SERVICES AND CREATE YOUR PRIVATE KEY .....</b>	<b>2</b>
<b>2. LAUNCH YOUR FIRST INSTANCE AND CONNECT .....</b>	<b>4</b>
<b>3. SERVER CONFIGURATION .....</b>	<b>9</b>
<b>4. MANAGING DISKS – ADDING, RESIZING AND TAKING SNAPSHOTS .....</b>	<b>14</b>
<b>5. BACKING UP YOUR SERVER .....</b>	<b>19</b>
<b>6. RECOVERING YOUR SERVER.....</b>	<b>26</b>
<b>7. SERVER MONITORING .....</b>	<b>29</b>
<b>8. AUTOMATIC SERVER MANAGEMENT USING PUPPET .....</b>	<b>32</b>
<b>9. USING CLIENT-BASED TOOLS TO MANAGE AWS SERVICES .....</b>	<b>36</b>
<b>10. CLEANING UP – MAKING SURE YOU DON'T LEAVE DATA / SERVERS.....</b>	<b>37</b>
<b>11. MORE FUN THINGS TO TRY .....</b>	<b>38</b>

# 1. Register for Amazon web services and create your private key

---

Lets start today by registering for Amazon web services. Registering for AWS will involve giving them a credit card and use of these services during this workshop may involve some charges but given the short term of this workshop the charges should be less than a few dollars (if that).

## Registration

---

1. Register for Amazon. Go to <http://aws.amazon.com>
2. Click Sign Up Now and complete the registration process
3. Once you have completed the registration process visit <http://aws.amazon.com> again and sign in.
4. Click on the EC2 tab and click on the Sign up for Amazon EC2.
  - a. To complete this process you will have to enter a Credit Card
  - b. This process may also include an Identity Verification step via Phone.
5. Signing up for EC2 should also get your registration complete for S3

## Getting your security credentials

---

After we get our account setup you need to generate and download your Key Pairs so you can login to instances that we create. Your keypair includes both a private and public key. The private key sits on your laptop and is used to connect to your server instance. You should not share your private key with anyone.

6. Go to <https://console.aws.amazon.com/ec2/home>
7. On the left hand side of the screen find the link to Key Pairs
8. Click the Create Key Pair Button
9. Enter a Key Pair Name
10. Download and save your private key (it does not matter where at the moment)
11. We will use this key pair to connect to our instances

## Setting up your key

---

### Mac / Linux Users

All you need to do to use your private key is to move it to a secure location and set permissions on it. On OSX (Mac) /Linux this process is similar.

12. Open the program Terminal (command spacebar to open search and type in terminal)
13. Change to your home directory (`cd ~`)
14. Check to see if you have a `.ssh` directory (`ls -al`)
15. If you do not, create it (`mkdir .ssh`)
16. Set permissions on that directory if necessary (`chmod 700 .ssh`)

17. Move the private key that you downloaded from amazon to this folder
  - a. `(mv ~/Downloads/MYKEYNAME.pem ~/.ssh/)`
18. Set the permissions on this file in your .ssh directory (`chmod 600 mykey.pem`)
19. You might as well leave terminal open – you will be using it quite a bit today!

## Windows Users

You may need to convert your pem key file to a format that putty or another program can support. These instructions are based on using putty as your ssh program.

20. Download and install putty and puttygen  
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
21. Launch PuttyGen
22. Load the private key (.pem) file you downloaded
23. Save your new private key (ppk file format). Saving it in your user documents somewhere is probably a good idea.
24. Launch putty
25. Click on SSH >> Authority >> Browse for the ppk key you created.
26. You can leave putty running but background it. In the next tutorial we will use this key to connect to our server.

In this tutorial we created an Amazon AWS account, signed up for Amazon Elastic Computing Cloud (EC2) and generated our Private access Key. In our next tutorial we will launch our first instance and connect to it.

## 2. Launch your first instance and connect

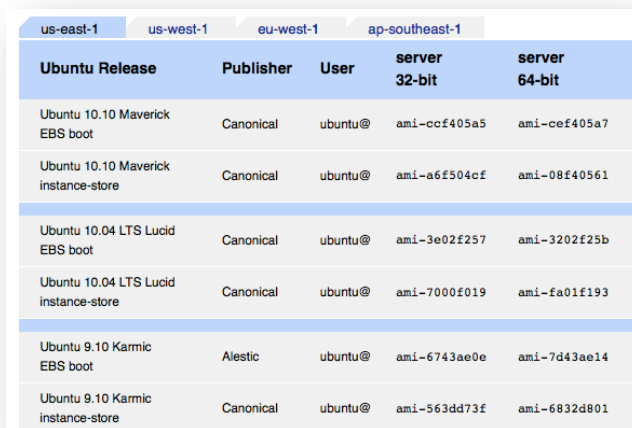
This series of tutorials uses the Alestic Ubuntu server images. When running a server on Amazon EC2 you can select one of a number of machine image sources including:

- a. Images maintained by organizations (e.g. Alestic, Rightscale, Microsoft)
- b. Images maintained by Amazon (e.g. Amazon Linux)
- c. Images maintained by individuals (e.g. Jim-bob)

Clearly, images created and maintained by an organization are preferable. Many machine images, particularly linux images, are available without cost. Other images including Windows, Red Hat Linux and SUSE Linux come with additional hourly CPU charges on top of the base CPU charges Amazon charges. This can have a significant impact on the cost of your EC2 environment. You access these pre-built images using unique AMI ids. For this tutorial we will get our AMI ID from a company called Alestic. We will then find the image that corresponds to this AMI by searching for the ID in the Amazon community instance database. Once we launch this instance however the server belongs to us and when we take an image of it later we will create a new AMI ID.

### Launch an instance on Amazon

1. Open a new tab in your web browser and visit the Alestic website (<http://alestic.com/>) and click on the us-east-1 tab. We are going to use a pre-built AMI (Amazon Machine Image) from Alestic as the base for our server.



The screenshot shows a table of Ubuntu AMIs in the us-east-1 region. The table has columns for Ubuntu Release, Publisher, User, server 32-bit, and server 64-bit. The first row is highlighted in blue, corresponding to the instruction in step 2.

Ubuntu Release	Publisher	User	server 32-bit	server 64-bit
Ubuntu 10.10 Maverick EBS boot	Canonical	ubuntu@	ami-ccf405a5	ami-cef405a7
Ubuntu 10.10 Maverick instance-store	Canonical	ubuntu@	ami-a6f504cf	ami-08f40561
Ubuntu 10.04 LTS Lucid EBS boot	Canonical	ubuntu@	ami-3e02f257	ami-3202f25b
Ubuntu 10.04 LTS Lucid instance-store	Canonical	ubuntu@	ami-7000f019	ami-fa01f193
Ubuntu 9.10 Karmic EBS boot	Alestic	ubuntu@	ami-6743ae0e	ami-7d43ae14
Ubuntu 9.10 Karmic instance-store	Canonical	ubuntu@	ami-563dd73f	ami-6832d801

2. Highlight and copy the EBS Boot 32-bit server ami for the most recent Ubuntu image

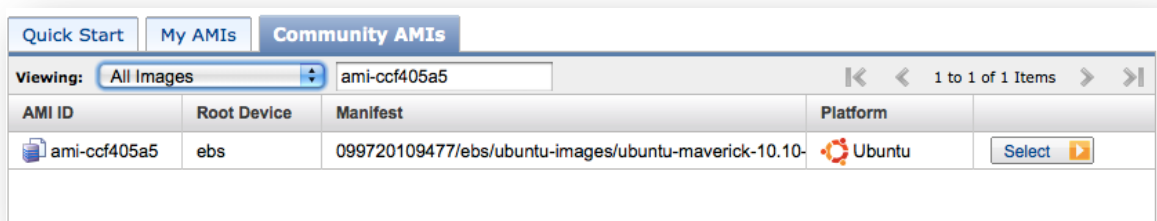
### EBS versus Instance Store.

Amazon offers two type of instance foundations – EBS-based images that use the Amazon Elastic Block Store system and Instance-Store images. The primary difference between EBS and Instance store is that you can stop an EBS instance and retain your server while stopping an instance store server means that the entire server and all of the data on it goes away. Typically you should use EBS machine images unless instance termination is not an issue

- Go back to your Amazon web console and follow the link to instances (<https://console.aws.amazon.com/ec2/home?s=Instances>)
- Click on the Launch Instance button

You will see a screen that allows you to select an instance. We want to head over to community AMIs. Before you go there, take note of the Amazon Linux AMIs that are available. This platform is part of the base Amazon offering and includes regular security updates. If you are looking for a solid Linux system to base your images on this could be a good route to go.

- Click on Community AMIs and paste the ami-id that you copied into the search box. Hit enter to tell the system to make it search.



- Click Select to begin the image launching process.

Over the course of the next few screens we will make decisions about the size, location, and security settings for our instance

- Select US-East-1B for your instance (you can choose any region but you need to remember this region for when we begin to add disk resources in later tutorials)
- Leave your server size at small (if you want to really save \$ you can use micro). Amazon also supports 64 Bit instances that use a different AMI.
- Make sure Launch Instances is selected and click Continue
- Accept the defaults on the Instance Details Screen (Kernel ID and Ram Disk ID) and click continue
- On the Tagging screen, assign a name to your instance. This will help you keep track of your instances. You can name it anything you like. The Key column contains the label "Name." Leave that alone. Under the Value column enter the actual name of your instance.

## Tags

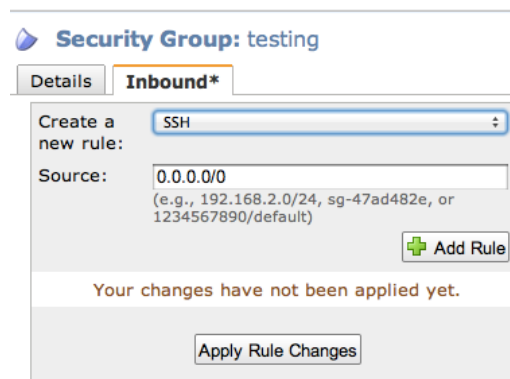
You can add as many tags to your instance as you would like. Name is particularly useful as your instances and EBS volumes grow. It can be helpful to adopt a consistent naming strategy that you will use on your Instance Name Tags, your EBS Name Tags, your Snapshots, and Server Images. You can assign tags to instances, volumes (disks), snapshots (backups), and various other AWS items. If you want to assign another tag, come up with a tag label for the first column (date) and a tag value for the second column (today's date)

12. On the Create Key Pair screen, select “Choose from your existing Key Pair” and select the key pair we created in our previous tutorial. Click Continue
13. On the Configure Firewall Tab click on “Create New Security Group.” On the next screen in item 1 you can name your security group anything you like. You can even create multiple security groups if you want to set different access levels for different servers.

## Security Groups

Security Groups are Amazon EC2 level firewalls that live outside of your server. These firewalls are a good first defense against intrusion and should be used to keep your server safe. In this tutorial we will open up two ports 22, and 80 so that we can ssh to our server and run a web service on it. In later tutorials we will add more complex rules to enable other services. Amazon Security Group rules follow the CIDR notation. For more information on using CIDR to limit access to your campus or other IP range, you can read [http://en.wikipedia.org/wiki/Classless\\_Inter-Domain\\_Routing](http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing).

14. Select your security group and click on the Inbound tab at the bottom of the screen. Using the drop down menu at the bottom of the screen add rules that are open to the whole internet.
15. For this tutorial we want to enable HTTP (80) and SSH (22) to “all internet” (0.0.0.0/0). If you are feeling courageous, figure out what the external IP address of your client machine is and restrict SSH access to that range (e.g. 152.18.0.0/16). Note that in a production environment you would not want SSH (22) open to the whole internet!
16. When ready to apply your rules, click on Apply Rule Changes.
17. Wait for your instance to launch. You can use the Refresh button on the Instances screen to update your view of the screen. When you instance shows the green “running” button, you are ready to connect!



## Connect to your instance

The first time we connect to our instance it will be using the key pair. During our first connection we will simply verify that we are connected and that we can become the root user (the master administrator on your machine). In the following tutorial we will do some basic setup on our server to get it ready for an application.

18. Right click on your instance name in the AWS console, choose “Connect”
19. Copy the Amazon public DNS name (e.g. ec2-50-16-179-191.compute-1.amazonaws.com) – see figure below

#### 4. Connect to your instance using instance's public DNS [ec2-50-16-179-191.compute-1.amazonaws.com].

### Special instructions OSX / Linux Users ( Windows users see below)

20. Launch terminal and enter the following command:

- a. `ssh -i ~/.ssh/myprivatekey.pem ubuntu@SERVERNAME`

### What does that command mean?

In the above command myprivatekey.pem is the name of the key you downloaded and SERVERNAME is the public DNS name of your server. The `-i` parameter tells ssh to use your private key. You will notice that we switched the user to ubuntu from root. On Ubuntu systems this is required because remote login as root is disabled by default. On other Linux systems your mileage may vary.

### Special instructions Windows Users

21. Using your popular ssh application (e.g. putty), add the server name to your connection window and ubuntu as your username. Remember that private key you setup back in the first tutorial? You still need that. If you did not save the key file from the previous tutorial add it in again here. If you are prompted for a username type in "ubuntu."

### Linux / Windows / OSX users

22. You will be asked to confirm the RSA keyprint of the server. This will add the keyprint to a known\_hosts file on your client computer.
23. Become root (`sudo su -`). Congratulations, you are connected!

### Sudo?

Sudo is a process by which you can grant elevated access to users without giving them access to specific user accounts. For the most part we will use sudo in these tutorials to become root or use root's credentials.

### Server updates

---

Let's do one simple process, server updating. Updating an Ubuntu server is done by utilizing the apt-get package management process.

24. After becoming root (`sudo su -`), run the update process
  - a. `apt-get update`
  - b. `apt-get upgrade`

25. These two commands gather updates from the Ubuntu update servers and prompt you to install them.  
The process typically goes without issue but should only be done in production environments if you have a good backup / restore procedure

After updating you may need to restart your server. Lets get this process out of the way.

### apt-get? update? upgrade?

Most linux systems have a set of programs that help install and manage programs. On Ubuntu apt-get is one of way of doing updates. The advantage of installing programs this way is that when updates are posted to the apt servers and you run apt-get update /apt-get upgrade those packages get updated. It is important to note that this can be good and bad. Sometimes a package update breaks something that works! Lots more on Ubuntu management can be found at <http://ubuntu.com>

26. Head back to the Amazon AWS console and right click on your instance. Select reboot. You will notice that you lose your terminal session.
27. Watch the AWS console until your server switches from “rebooting” to “running”
28. Login again to make sure that everything is ok. How you do this depends on your platform. On Mac / OSX you can return to your terminal window. On Windows you should re-launch putty & connect.

### Get a static IP address

---

Before we go much further we should go ahead and allocate and assign a static IP address for our server. Static IP addresses are an excellent way to identify your “production” box and are easily reassigned. You are charged for each static IP that you allocate but do not assign to a server so keeping multiple IP addresses in reserve can cost. In a production environment, once you allocated this static IP address you would want to communicate it to your campus IT and other cloud-based service providers to enable access to and from this machine.

29. You might as well log out and re-connect to your instance as assigning an IP address will change both your internal and external IP addresses
30. In the Amazon EC2 web console
31. Click on the Elastic IPs link on the left hand side
32. Click the “Allocate new Address” button
33. Once the address is allocated, right click and choose “Associate Address”
34. Choose your running instance ID and click “Associate”
35. Reconnect to your server using the external IP address you just allocated. Note, your old connection string will not work.
  - a. Mac users `ssh -i ~/.ssh/KEY.pem ubuntu@new.ip.address.now`
36. You will also have to accept your RSA key fingerprint again.

In this tutorial we have launched, connected to, and updated our first server. In our next exercise we will do some basic server setup and take an image of our server.

## 3. Server configuration

---

In this tutorial we will complete some basic server configuration and prepare our server for taking a base image. This will include setting up an internal firewall, installing apache, php and mysql, adding a user, and modifying ssh parameters to allow password-based login. If you are game you can also install vufind and wordpress as some sample applications.

### Tour your server

---

Before we get started lets take a quick tour. If you are new to linux you might want to take a few minutes and read an introduction (<http://vic.gedris.org/Manual-ShellIntro/1.2/ShellIntro.pdf> ). We will not do too much with the command line but it will be confusing if it is foreign.

1. type `df -h`. You will see the disk devices connected to your machine and their capacity.

Amazon advertises each server with a certain capacity. On small and large servers that capacity tends to be connected to the server as `/dev/sda2` and mounted as `/mnt` (at least for Alestic servers). You will notice no other mounted disks other than `sda1` on your micro server.

2. If linux is new, here are a few interesting commands:
  - a. `ls -l` – how you look at a directory
  - b. `cd` – change directory
  - c. `pwd` – what directory am I in?
  - d. `top` – shows you a process list
  - e. `ps -a` – show all processes
  - f. `man (command)` e.g. `man ps` – gives a how to on commands
  - g. `du -h` – disk usage in human readable format
  - h. `df -h` – disk information in human readable format
  - i. `uptime` – server load
  - j. `who` – who is on your box
  - k. `netstat` – who is connected over the network to your box
3. Look around for configuration and log files. Our Ubuntu server is very standard:
  - a. Server configs tend to be in `/etc/`
  - b. Restarting services is done via `/etc/init.d/service name start|stop|restart`
  - c. Logs are in `/var/log`
  - d. Apache document root is `/var/www`

### Installing your firewall – UFW the “Uncomplicated Fire Wall”

---

Lets start by setting up a server based firewall. Server firewalls are good additions to your Amazon AWS firewall as a second level of security. Unlike the Amazon firewall which will always rely on external IP addresses, a server level firewall may also need to be programmed to allow IP addresses internal to the Amazon EC2 system.

4. Connect to your server using the command from the previous tutorial.

5. Become root (sudo su -)
6. apt-get install ufw
7. ufw status - this shows us what the current status of the firewall is. It should say inactive
8. ufw default deny (deny all connections by default)
9. ufw logging on (enable logging)
10. ufw allow 80 (lets enable the webserver)
11. ufw allow 22 (ssh to our box is important!)
  - a. feeling courageous? ufw allow from EXTERNALIPRANGE/16 to any port 22
12. ufw enable
13. ufw status – if you don't see port 22 here you will not be able to ssh to your box
14. Launch a new ssh session (do not kill your old one yet!) and connect to the server - did it work?
15. Check out the logs for ufw - tail -f /var/log/syslog (this looks at the tail end of a file Ctrl-C to get out of it)
16. Lets play around by deleting access to port 22
  - a. ufw delete allow 22
  - b. ufw reload
  - c. ufw status
  - d. tail -f /var/log/syslog (this looks at the tail end of a file Ctrl-C to get out of it)
17. try to connect on ssh with a new terminal window - what happens in syslog?
18. Be sure to re-enable port 22 before moving on!

## Setup a basic webserver

---

In order to get us ready for an application install, lets get our base system setup. In order to keep us focused on Amazon issues we will use the simplest approach possible.

19. Run tasksel to setup your server (tasksel --section server)
20. Choose LAMP server from the menu
21. Run through the automated installers for Mysql, Apache and PHP. Remember your passwords!
22. Go to a web-browser and hit your IP address. Do you get a stock Apache "It works!" screen?
23. Keep going – lets install phpmyadmin (apt-get install phpmyadmin)
24. Once phpmyadmin is installed it your webserver <http://myaddress/phpmyadmin>
  - e. configure phpmyadmin to work with apache

## Adding a user, updating sudo & enabling password based access

---

You may or may not want to add additional users and enable password access to your server. There are numerous opinions about the best way to approach user management and access on your server (for example you might view added users as unnecessary complications and passwords as insecure).

25. In Ubuntu you can use the adduser command (as root, adduser username)
26. Once you have the user added you can add them to a group that has sudo access.

## More on sudo?

Sudo is a method by which we can grant the ability to act as a user without knowing their password. You will use sudo regularly on your system to become the user root. Lets take a quick look at our sudoers file (more /etc/sudoers). Your file should contain the following line: %admin ALL=(ALL) ALL. Without getting into the particulars of sudo, this file controls how sudo works on your machine. You can edit the file with a specific program (visudo). The line referenced above tells us that members of the admin group can run any program on the server.

27. Lets give our new user the ability to do this by adding them to the admin group
  - a. `usermod -a -G admin username`

## Linux editors – vi and nano

### **nano – a friendly editor**

Nano has a simple graphical interface. menu options are at the bottom of the screen. If you are not familiar with linux at all, use nano.

### **vi: a lightweight editor**

Vi is a simple text editor that you can use to edit files easily. It has no graphical interface and instead relies on keyboard commands. Below is quick list of keyboard commands for you to use:

#### **Navigation mode**

j = up

k = down

h = left

l = right

#### **Editing mode**

i = start inserting text at the cursor

a = start inserting text after the cursor

d = go into delete mode. After you hit d, you can use the space bar to delete a character

[shift] d = delete the line beginning from the cursor

[esc] escape an editing mode or command

#### **Saving and exiting**

: = go into save or exit mode

x = save and exit (e.g. :x)

q! = exit without saving

w = write but stay in the editor

Now that our user has access, we should enable password access to the server

28. Edit the sshd\_config file (`vi /etc/ssh/sshd_config`)
29. Find the line “PasswordAuthentication no”, change no to yes
30. Find the line “PermitRootLogin” change yes to no
31. Exit vi ( `ESC : x` )

32. restart the sshd service
33. /etc/init.d/ssh restart
34. Without stopping your current ssh session, open a new session and connect with your new user

## Install a web application

---

Lets install a few applications on our server. Feel free to do as many or as few as you like!

### Wordpress

Wordpress is a simple blog / CMS platform. Installing wordpress will give us a good web service to work with as we proceed through out exercise. You can setup wordpress using the instructions at [wordpress.org](http://wordpress.org) [http://codex.wordpress.org/Installing\\_WordPress#Famous\\_5-Minute\\_Install](http://codex.wordpress.org/Installing_WordPress#Famous_5-Minute_Install)

35. Lets start by getting our files
  - a. `cd /tmp`
  - b. `wget http://wordpress.org/latest.tar.gz`
  - c. `gunzip latest.tar.gz`
  - d. `tar -xvf latest.tar`
36. Now lets move the wordpress folder into our web directory
  - e. `mv wordpress /var/www`
37. Now lets create our database in pypmyadmin – follow the instructions at [http://codex.wordpress.org/Installing\\_WordPress#Using\\_phpMyAdmin](http://codex.wordpress.org/Installing_WordPress#Using_phpMyAdmin)
38. Now update your wp-config.php file
  - f. `cd /var/www/wordpress`
  - g. `cp wp-config-sample.php wp-config.php`
  - h. `vi (or nano!) wp-config.php`
    - i. Change the database name, username and password to match what you put into phpmyadmin
  - i. save and exit
39. Lets run the installer!
  - j. Go to your ip address/wordpress in a browser
40. Once you have gone through the setup instructions and made sure that your blog works you can move on to the next section.

## Taking an image of your server

---

Amazon allows you to create server images that are snapshots of all of the resources connected to a server. To take a snapshot we will have to be prepared to have our server reboot. Depending on the size of your server and the time it has been since your last snapshot, taking an image could take some time (e.g. 30 minutes). During this time your server is unavailable.

41. Go to the AWS console, click on Instances
42. Right click on your image and select “Create Image EBS AMI.”
43. Enter a descriptive image name (e.g. 020411\_testingserver\_baseimage) and click “Create this Image”

44. Your server will reboot and an image will be created. You can monitor the image creation process on the AMIs tab in the Amazon Web Console
45. Once the server image says “available” you can login to your server again

In this tutorial we installed a firewall, setup a LAMP environment, added a user and enabled password access. We finished up by taking a server image. In our next tutorial we will begin working with external EBS volumes.

## 4. Managing disks – adding, resizing and taking snapshots

---

In this tutorial we will begin working with external EBS volumes on our Amazon server. EBS volumes are externally mounted network disks that have a number of advantages over locally used storage. They persist outside of server instances, can be easily backed up through snapshots, and can be attached, detached, and re-attached easily. In this tutorial we will use external disks for three purposes 1) resizing the root partition, 2) creating an application disk space outside of our server and 3) moving our mysql data files so that they are not tied to a specific server.

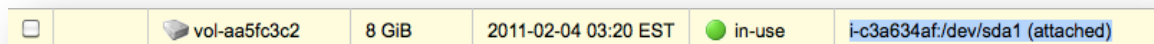
One of the challenges or opportunities in working with Amazon EC2 servers is that you need to be prepared to migrate your system to a new server instance. We will cover this in detail in the coming tutorials but for the time being it is worthwhile pointing out that regular snapshots of data stored on external EBS volumes provides the fastest way to recover a server when something goes wrong. The alternative, keeping data on an instance-store or using the built-in storage with an EBS server makes backup and recovery more complicated and less fault tolerant.

### Resizing your root partition

---

We are going to begin this process by taking a snapshot of our root partition. We will then create a new volume from that snapshot (with a larger size) and connect it to our server. These instructions rely on the web console but if you want to use command line tools there is an excellent tutorial at Alestic (<http://alestic.com/2010/02/ec2-resize-running-ebs-root>)

1. Take a snapshot of your root partition by first navigating to your instance window in the Amazon Web Console (<http://aws.amazon.com>).
2. Take note of your instance ID
3. Click on the Volumes Tab on the left hand side of your console.
4. Find the volume that is attached to this instance as /dev/sda1



5. Right click on the instance and choose “Create Snapshot”
6. Be sure to give your snapshot a descriptive name!
7. Head over to the Snapshots tab in your console and watch the snapshot create.

Now that we have a snapshot of our root partition we will stop our server, create a new volume from the snapshot and swap it with our existing device attached as /dev/sda1.

8. Stop your server (Instances >> instance >> stop server)
9. Create a new volume from your snapshot
10. Click on snapshots, find your root partition snapshot, right click and choose “Create volume from snapshot.”
11. Make sure that the availability zone matches your server!
12. Add in a new size for your partition. Once you size up you cannot go down so add space judiciously!

13. Go to volumes and watch your snapshot create
14. Once it is created, find your old volume that is currently attached as /dev/sda1 and detach it from your server (right click in volumes >> Detach Volume)
15. Attach your new volume (right click, choose attach, select your instance and attach it as /dev/sda1)



16. Start your server and connect to it. Note – When you stop your server you lose your IP address association. Visit the static IP address list and re-associate it with your server.

The last thing that we need to do here is resize our root partition. This is done with the command `resize2fs`.

17. As root or sudo run the command `resize2fs /dev/xvdf1`
  - a. Note – although your disk is mounted as /dev/sda1. When you look at your /dev/ folder on your server you will see that the disks are mounted with a different name (xdf..)
18. Verify that you have a larger root partition (`df -h`)

## Adding an application volume

After resizing root, the addition of an application volume should seem pretty simple. The goal behind this is to add an EBS volume that you will use to store all of your added applications. In this exercise, we will create a new volume, add it to the server, format and mount it and then configure Apache to use it as its home directory.

Running applications on external volumes means that you can easily disconnect / reconnect them to different servers if needed. It also means that you can easily snapshot them as standalone units.

19. Create a new volume (In AWS Volumes >> Create Volume)
20. Set a size (perhaps 10 GB) and an availability zone that matches your server (e.g. us-east-1b). You do not need a snapshot.
21. Watch the disk create

22. Connect the disk to your server (Right click >> Attach Volume), select your proper instance and assign a



device name (start with /dev/sdf)

23. Login to your server and look at your attached disks (df -h). You probably don't see /dev/sdf yet!

24. Lets format our new disk with the ext4 file system (mkfs -t ext4 /dev/sdf)

25. Lets mount our new disk by editing fstab

- a. vi /etc/fstab
- b. add the following line:
- c. /dev/sdf /m1 auto defaults 0 0
- d. Make the directory you just referred to (mkdir /m1)
- e. make sure you have fstab right by running mount -a
  - i. If you are running a micro instance you may get a /dev/sda2 device not found.

Now that we have a new disk mounted to our system we need to adjust Apache to work with it. We have two methods of doing this. One way would be to use symbolic links from /var/www to /m1/website (our library website directory). The second is to adjust Apache configurations to address that area directly. These instructions focus on option 2.

26. Make a directory on /m1 to hold your website (mkdir /m1/website)

27. Relocate the index.html file from /var/www to this space (mv /var/www/\*.html /m1/website)

28. Edit apache configuration to point to this new location

- a. cd /etc/apache2/sites-enabled
- b. vi 000-default
- c. replace "DocumentRoot /var/www" with "DocumentRoot /m1/website"

29. We need to set permissions on this folder in /m1 chown -R www-data:www-data website)

30. Lets restart apache and see how we did (as root or sudo /etc/init.d/apache2 restart)

Now that we have our application volume we can add new apps to it while updating central configuration files as necessary. Lets test our server configuration by rebooting the whole server. Did your disk reconnect properly? Does Apache still work?

## Moving MySQL

---

31. Running MySQL on your root partition can be risky. MySQL contains frequently updated data. The fact that MySQL can be difficult to snapshot while it is running means that keeping it on a core disk can create issues with taking regular snapshots. To move MySQL we will create and attach another volume, stop mysql, move mysql folders to this new disk, create entries in fstab to redirect mysql and then start and test.
32. Start by creating a new volume about 10GB large in the same availability zone
33. Attach the drive to your server as (/dev/sdg), format and mount it. as /m2. Create an entry in /etc/fstab
34. Stop mysql (/etc/init.d/mysql stop)
35. Create directories in /m2 to hold lib, etc and log directories for mysql
  - a. mkdir /m2/etc, mkdir /m2/lib, mkdir /m2/log
36. Move MySQL directories to their new locations
  - b. mv /etc/mysql /m2/etc
  - c. mv /var/lib/mysql /m2/lib
  - d. mv /var/log/mysql /m2/log
37. Recreate those directories
  - e. mkdir /etc/mysql
  - f. mkdir /var/lib/mysql
  - g. mkdir /var/log/mysql
38. Add entries to fstab for these volumes
  - h. /m2/etc/mysql /etc/mysql none bind
  - i. /m2/lib/mysql /var/lib/mysql none bind
  - j. /m2/log/mysql /var/log/mysql none bind
39. Mount these entries
  - k. mount /etc/mysql
  - l. mount /var/lib/mysql
  - m. mount /var/log/mysql
40. start mysql
  - n. /etc/init.d/mysql start

By moving MySQL to a new disk you get many of the benefits of an abstracted database service (e.g. Amazon's RDS) without the added expense. Amazon RDS is a good option for organizations who want automated backup of MySQL or who want to approach cloud computing from a SAAS/PAAS approach rather than an IAAS approach.

Finishing up

Lets finish this tutorial by restarting our server, making sure that everything works and taking a snapshot.

41. reboot your box, test it
  - a. Did it come online? Is apache running? Is MySQL up?

42. Take a server images

- b. Note that the image now contains all of the disks that we have mounted

Ultimately, how you decide to allocate disk space is up to you. There is a tradeoff between disk complexity, server management comfort/time and needed redundancy. I have found that at the least, making sure that my applications, database and root partition reside on separate disks makes system recovery simple.

## 5. Backing up your server

---

This tutorial covers how to setup a quick and easy backup routine using a program built on top of the Amazon ec2 server tools. In this tutorial we are going to explore three different backup methods. Two methods utilize the AWS toolkit and the third method utilizes a product called Zmanda. Deciding how to approach backups for a cloud based system has some points in common with traditional IT systems (e.g. backup rotation, frequency, fall-back plan, fall-forward plan) but also has some things that are completely different. For example, in a virtualized environment machine or disk images may be preferable or acceptable backups for daily or weekly backups. Likewise, a fall-back or fall-forward plan may not involve restoring from backup at all but may involve swapping virtual disks or launching new servers. Before we get into those issues however, lets look at the nuts and bolts of backups.

### Automated image creation

---

Using automated image creation and disk snapshots with regular pruning (e.g. removal of old backups) is an easy to configure and maintain way of keeping backups. While the simple version of these scripts run the risk of not holding onto data long enough or holding onto disk images that you do not need, it is easy to configure, easy to restore and parsimonious in backup space usage. Disk snapshots in Amazon are incremental – meaning that the first snapshot takes lots of space but each subsequent snapshot only stores changed bits.

### Overview

---

This program iterates through all of your instances, lists data about them and creates machine images.

### Dependencies and configuration

---

In order to get this script running we are going to use a python library called boto (<http://code.google.com/p/boto/>). Python boto takes the Amazon AWS API and implements it in Python. While there are also PHP, Java and other implementations, we choose Python in this case for its simplicity.

Lets make sure that we have the correct versions of python and boto on our server.

1. `sudo apt-get update`
2. `sudo apt-get upgrade`
3. `python --version` – It should say 2.7.x

Now lets find our version of boto.

4. `python`
5. at the python prompt (`>>>`) type `import boto`
6. type `boto.Version`

If you have boto version 1.9 or above things should work fine. If not we need to remove the apt version of boto and install our own:

7. `sudo apt-get remove python-boto`
8. `cd /tmp`
9. `wget http://boto.googlecode.com/files/boto-2.0.tar.gz`
10. `gunzip boto-2.0.tar.gz`

11. tar -xvf boto-2.0.tar.gz
12. cd boto-2.0.tar.gz
13. python setup.py install

There are three components to this service, a backup python script, a shell script to run specific backups and a cron job to have the job run automatically. In the section below we will briefly review the function of each script before we create it. In general, the cron job calls the ksh script which calls the python script, passing the volume ids to, number of backups to retain, and backup description text. The python script creates a new snapshot for each volume specified in the ksh script, gets all previous snapshots, sorts them by date and then deletes all but the newest n snapshots as defined in the ksh script.

### Create a place for backup scripts to live

Lets keep this simple – create a new directory in the root directory and make ubuntu the owner. As you will see in later exercises you may want a special backup user or root user to own these files so that access to them can be limited. How you configure this depends in part on your view of system security and who else has access to this machine.

14. sudo mkdir /backups
15. sudo chwon ubuntu:ubuntu /backups
16. cd /backups

### Create the python script

The python script creates a new snapshot for each volume specified in the ksh script, gets all previous snapshots, sorts them by date and then deletes all but the newest n snapshots as defined in the ksh script. Crontab configuration as root to run in the morning. In order to write this script we need to get our access credentials.

17. Visit <http://aws.amazon.com>
18. Select “Sign in to the console”
19. After you login, under your name, select “security credentials”
20. Login again
21. Go to your access credentials and copy your access key and secret access key (click on show)
22. Paste these into a text editor for now so you will have them as you write your program

Lets begin experimenting with the python-boto library. Use your favorite editor (vi, nano, etc) and create a basic script that lists all of your running instances. First, lets talk about the structure of a python script. Python is a language that depends on spacing and indentation as opposed to end of line characters. If your program is not working properly, particularly in for loops pay attention to indentation and the : at the end of the for loop definition line. Call the script listinstances.py

23. vi listinstances.py
24. Enter the following code into your file:

```
#Code adapted from saltycrane.com/blog/2010/03 -  
from boto.ec2.connection import EC2Connection  
from pprint import pprint  
# Substitute your access key and secret key here
```

```

#My AWS Keys
aws_access_key = 'yourkey
aws_secret_key = 'yoursecretkey'
conn = EC2Connection(aws_access_key, aws_secret_key)
reservations = conn.get_all_instances()
instances = [i for r in reservations for i in r.instances]
for i in instances:
    pprint(i.__dict__)

```

Now lets run this program and see our instance output. The program simply prints all of the data returned about your instances

#### 25. Python listinstances.py

You should see all of your instance attributes listed on the screen. Notice that you can access lots of information this way including instance ids, state, ip addresses, and attached volumes. Now that we have a basic understanding working lets use this platform to create a snapshot of our system. Return to the script and edit it to match the code below. You will notice that we are calling the create\_image method of the connection object. We pass the image id (i.id), an image title ("image for" + i.id), and an image description (description = "backup image").

#### 26. vi listinstance.py

```

#Code adapted from saltycrane.com/blog/2010/03 -
from boto.ec2.connection import EC2Connection
from pprint import pprint
# Substitute your access key and secret key here
#mitcheet account gmail
aws_access_key = 'AKIAIOJL5X7743DQSZLQ'
aws_secret_key = 'YBRD+kCCTNTxLu3x5SR8ISnFRQQ72nB7FzPkojbi'
description = "

conn = EC2Connection(aws_access_key, aws_secret_key)
reservations = conn.get_all_instances()
instances = [i for r in reservations for i in r.instances]
for i in instances:
    pprint(i.__dict__)
    conn.create_image(i.id, 'image for ' + i.id, description='Backup image')
    print "Creating image for instance " + i.id

```

Run your script again. You will notice that your box reboots and a full image is taken. While the image is taken be patient. While you wait check the status of the ami in your Amazon web console.

While box reboots are good practice, you can avoid them by adding a fourth parameter into your script (no\_reboot=True). Try adding this parameter to the script and running the program again.

#### 27. Python listimage.py

You probably found that the program balked when it tried to create an image because the imagename was already in use. We can handle that by adding a date/time stamp onto our image name. This takes a bit more code. Review the code below and add the appropriate lines

```
#Code adapted from saltycrane.com/blog/2010/03 -
from boto.ec2.connection import EC2Connection
from pprint import pprint
from datetime import datetime

# Substitute your access key and secret key here
#mitcheet account gmail
aws_access_key = '...'
aws_secret_key = ...
dt = datetime.now()
imagedate = dt.strftime("%Y/%m/%d-%H-%M-%S")
conn = EC2Connection(aws_access_key, aws_secret_key)
reservations = conn.get_all_instances()
instances = [i for r in reservations for i in r.instances]
for i in instances:
    #pprint(i.__dict__)
    conn.create_image(i.id, 'Backup on ' + str(imagedate) + ' for ' + i.id, description='testing',
no_reboot=True)
    print "Creating image for instance " + i.id
```

This will give us an image with a specific date and time and will not reboot the box when taking images. This by itself could be a good approach to large-scale image creation for your systems. A useful strategy would be to cron this process to run weekly or monthly as you see fit to ensure long term image creation and maintenance.

## Use Python-Boto to create disk snapshots

Lets move on to a different script that will create and manage snapshots of all of your attached volumes. The job is setup in the root users cron uses the boto-python library along with AWS tools to take snapshots. These snapshots can be used in a number of manners and are our primary method for backing up servers. The script we are writing today is fairly simple and removes all snapshots except the newest n snapshots specified in your script. A more sophisticated script that maintained weekly, monthly and yearly backups would be a good second step for this basic script.

28. Lets create a new script called `manage_snapshots.py` in our backup folder. Follow the code below to create your script.

```
# manage_snapshots.py
# Author - Vaibhav Puranik
# Adapted by Erik Mitchell for specific EC2 snapshots
#
# Every time this script is executed, it creates a snapshot of the given
# volume. It then checks number of snapshots available for
# the given volume. If the total number of snapshots exceed
# value of 'keep', then it deletes the older snapshots. It ensures
# that 'keep' number of newest snapshots are preserved.
#
# If you specify 0 as keep value, it will delete all the snapshots including
# the one it just created. Thus this script can be used to delete all the snapshots.
# Please use this script at your own risk. Although this code has been
# tested in production, I will not be liable for any damage it causes.
```

```

from boto.ec2.connection import EC2Connection
from datetime import datetime
import sys
import os

# Substitute your access key and secret key here
aws_access_key = 'YOURACCESSKEYHERE'
aws_secret_key = 'YOURSECRETACCESSKEYHERE'

# Check to make sure that we have three entries on the command line
if len(sys.argv) < 3:
    print "Usage: python manage_snapshots.py volume_id number_of_snapshots_to_keep description"
    print "volume id and number of snapshots to keep are required. description is optional"
    sys.exit(1)

# Import our variables from the command line
vol_id = sys.argv[1]
keep = int(sys.argv[2])
description = sys.argv[3]

# Open a new EC2 connection and get the volume object for a given ID
conn = EC2Connection(aws_access_key, aws_secret_key)
volumes = conn.get_all_volumes([vol_id])
volume = volumes[0]

# Try to create a snapshot for the volume. If it is successful return some text to the screen
if volume.create_snapshot(description):
    print 'Snapshot created with description: ' + description

# Get a List of snapshots associated with a volume
snapshots = volume.snapshots()

# A function that sorts the creation dates of snapshots
def date_compare(snap1, snap2):
    if snap1.start_time < snap2.start_time:
        return -1
    elif snap1.start_time == snap2.start_time:
        return 0
    return 1

# Sort the snapshots into date/time stamps
snapshots.sort(date_compare)

# Determine how many snapshots to keep
delta = len(snapshots) - keep

# Remove old snapshots
for i in range(delta):
    print "Found an old snapshot, Deleting snapshot for volume " + vol_id
    snapshots[i].delete()
for volume in vol-b99a3830 vol-39ca3b32 more volumes listed here
do
    python /backups/manage_snapshots.py $volume 90 "Backup of $volume on $(date +%F-%H:%m)"
done

```

You will notice some different python structures here – for, if / elif. This script is a bit more complex in logic but uses the same methods as our script above. Try running the script on a command line:

```
29. python manage_snapshots.py volume_id number_of_snapshots_to_keep description
```

## Database exports

---

Database exports are general good practice in addition to disk snapshots. One issue with taking disk snapshots for databases is that most databases need to flush database memory to disk or lock database files prior to backup. There are multiple approaches to this including specific Amazon tools such as `ec2_consistent_snapshot`, open source tools such as ZManda and database specific tools (e.g. `pg_dump` for postgres or `mysqldump`).

For this workshop we will look at the simple method of database specific dumps. For more systematic approaches I highly recommend Amanda, Zmanda or EC2 consistent snapshot. Previously we installed wordpress and explored wordpress. Now we will focus on ensuring that these systems are backed up properly

Lets create a new backup script in our backup folder for our simple mysql backup script. We are going to use the `mysqldump` program (`mysqldump -u username -ppassword database_name > dump.sql`). We will start by making a directory for our database backups

```
30. mkdir /backups/database_backups
```

And a shell script to run the backups. We will keep it to a single line. Be sure to substitute your password and database name in the script.

```
#!/bin/sh
mysqldump -u root -ppassword database-name > /backups/database_backups/my_backup_$(date
+%F-%H:%m).sql
```

```
31. Make the script executable
```

```
a. chmod 700 mysqlexport.sh
```

```
32. Run the script
```

```
a. ./mysqlexport.sh
```

When you run the script on the command line it should return nothing but will create a sql file with the backup of your database. You can more (`more my_backup....`) to see the sql syntax.

There are many (and better) ways of handling database exports. I recommend programs that help you manage database export rotation and compression automatically. Zmanda is an excellent solution for this. If you want to try out `mysql-zrm` (Zmanda) you can follow the tutorial at [http://wiki.zmanda.com/index.php/Quick\\_Start\\_Example\\_-\\_Ubuntu\\_Configuration](http://wiki.zmanda.com/index.php/Quick_Start_Example_-_Ubuntu_Configuration).

Another option that we will discuss briefly is to use the Amazon AWS provided relational database service. The RDS service provides continuous data protection and offloads all database work from your server.

## Setting up Cron for your backups

---

Before we move on lets make sure we are acquainted with setting up cron jobs to make our life easier. Because we have created a script to backup our volumes and a script to export our sql data we want to add both of those to cron. For simplicity lets do this as root. The first number of cron is the minute of the hour. The second number is the hour. Try tuning these to match a minute in the near future to make sure your jobs run without error.

### 33. crontab -e

```
0 4 * * * /backups/database_backups/mysqllexport.sh | mail -s "Subject of mail" user
15 4 * * * /backups/backups.sh | mail -s "Subject of mail" user
```

## 6.Recovering your server

---

One of the issues that we run into with Amazon EC2 servers is that when they reboot and fail to start we have to be creative with how we recover. You have a few ways of debugging a server issue in Amazon but the easiest way if it does not boot is to look at the console (right click on the instance and select show console). If it is something you can fix, you can stop the server, move the disk that contains the offending configuration to a different server and fix it. For larger problems snapshots are the best way to recover from server failures. This means that snapshots of your root partition before you do system updates and it means being prepared to mount your root volume on a recovery server if needed.

The following tutorial involves messing up your server enough so that it will not boot and fixing it.

### Fixing small problems

---

While preparing for this tutorial I mis-typed a command in fstab and failed to test it before rebooting. This means that when the server restarted I got the following message:

```
cloud-init start running: Fri, 04 Feb 2011 11:17:23 +0000. up 3.33 seconds
found data source: DataSourceEc2
mountall: Event failed
init: ureadahead-other main process (601) terminated with status 4

An error occurred while mounting /m2
Press S to skip mounting or M for manual recovery
mount: unknown filesystem type 'audo'
mountall: mount /m2 [602] terminated with status 32
mountall: Filesystem could not be mounted: /m2
```

I found this message by right clicking on my instance in the Amazon AWS Web Console and choosing "Get System Log."

Lets replicate this error

1. In /etc/fstab, edit the line mounting /m2 to have a clear error (e.g audo instead of auto)
2. reboot – fail.
3. Stop your server
4. Detach your volume
5. Lets use our backup from our first image to serve as a quick instance to fix this issue:
6. Go to AMIs in The Amazon web console, find your server image, right click and choose launch.
7. Set your preferences and wait for the server to launch.
8. When the server is running, connect your root partition as /dev/sdh
9. connect to your recovery server and make a directory (e.g. mkdir /recovery)
10. mount /dev/sdh /recovery
11. Edit your fstab in /recovery/etc
12. Save, exit, unmount, shut down your server, disconnect the drive, reconnect the drive to your old server (as /dev/sda1), start your old server. Remember to associate your IP address again

This approach could be used in conjunction with a number of other backup mechanisms including Bacula, Amanda, rsync, svn, etc. The important thing would be to have a server that had access to your other backup routines that you could use to help fix the issues in the file system.

## Recovering from disk snapshots (massive failure)

---

After going through the previous exercise this may seem pretty simple. To recover by rolling back to a previous snapshot you can “hot swap” devices or do it by stopping your server. The main risk of trying a hot-swap is that you position yourself for failures the next time you reboot your system.

This approach is particularly important (especially cold-swapping) if your server bricks following a system update. Having a recent snapshot of your root partition is the easiest way to avoid significant downtime in this type of environment. In many ways this structure is even faster than using a server management system like Puppet or CFEngine..

### 13. Hot-swapping volumes

- a. Create a new volume from a recent snapshot
- b. Attach it to your server
- c. update fstab to point to your new device
- d. run `mount -a`
- e. Verify that things work correctly
- f. Detach your old device from the server

### 14. Cold-swap volumes

- g. Create a new volume from a recent snapshot
- h. Stop your server
- i. Detach your old volume
- j. Attach your new volume using the same information
- k. Start your server
- l. Verify that everything works correctly

## Snapshot recovery exercise

---

Want to try it out? After taking a successful series of system snapshots using the program developed in the first part of this tutorial we are ready to really mess things up.

15. As root `cd to /`
16. execute `“rm -r *”`
17. Wait for your server to fail.
18. Reboot – fail.
19. Follow the cold-swap procedures outlined above

## Recovery using Amanda

---

I mention using a backup tool like Amanda (<http://amanda.org>) here because Amanda is an effective backup strategy for Amazon based systems. Amanda supports writing directly to S3 (Simple Storage Service) . While these tutorials do not replicate the Amanda instructions for setting up Amanda to write to S3, there is an excellent white paper hosted at Amanda that discusses this process.

The benefit of Amanda is that it is capable of performing very detailed data backup and includes automatic mechanisms such as backup rotation, full vs incremental backups etc. These can be very important for server redundancy, and archiving. In my experience, the downside of Amanda on the EC2 platform is that the tools are slower and more complicated to work with than volume and server snapshots. For that reason this tutorial focuses on automating volume snapshots instead of backing up with Amanda.

# 7. Server monitoring

---

The power behind the AWS platform is that you can create rather large servers that are capable of acting as if they were sitting in your building. The risk of this is that you find yourself replicating the same over-built IT infrastructure that you are used to having in-house. This tutorial covers some ways to consider changing your approach to essential services such as system monitoring, notifications, email services and suggests potential paths for even more aggressive refactoring of system architecture.

## Using a hosted monitoring service

---

There are many good software as a service tools that can be used to help monitor systems. The guide at <http://www.datacenterknowledge.com/archives/2010/02/04/cloud-monitoring-services-a-resource-guide/> and <http://mashable.com/2010/04/09/free-uptime-monitoring/> are good places to start for recommendations.

Today we are going to explore a hosted site called TagBeep.com. TagBeep offers free site checking for up to 50 websites at 1 minute intervals.

1. Visit <http://tagbeep.com>
2. Signup for the service
3. Add a simple monitor for your site (we installed apache earlier!)
4. When you add your site monitor be sure to type something wrong
5. Watch for your email alert!
6. Go back to tag beep and fix the check, you should get an OK email

## Using content checks to verify dynamic service uptime

---

Now that we have a basic service setup (to verify that apache is running), lets setup a dynamic service check. This will allow us to check to make sure that our dynamic systems are working. To make this work we will use a content check of the text that returns on a sample page.

Add a new alert and make the alert match the content of the screen shot below. This will check a specific page in the ZSR library vufind catalog to make sure that the title of the work that should be on the page is returned. If vufind is down or having other issues, the string will not be on the page and an alert will go out.

[my checks](#) / add uptime check

The screenshot shows the configuration page for a monitoring check named "Vufind service check". The "Webpage Url" is set to "http://cloud.library.vufind.org/Record/12073". The "Resolution" is set to "1 minute" and the "Max load time" is set to "20 seconds". Under "Content checks", there is a "With string" field containing "The Informer" and a "Without string" field which is empty. A "Not empty" checkbox is checked, indicating the page should not be blank. There are "hide" and "Custom settings" links at the bottom right of the content checks section.

Now lets mimic the site being down by changing the expected text on the page to something that is not there (like LITA2011). Save your alert and wait for the system downtime email.

## Using the Amazon Monitoring platform

Many IT organizations rely on Nagios to perform system monitoring. Nagios is an invaluable tool that notifies sysadmins about service status, disk space, cpu load and any number of other items. While Nagios may not be replaceable for many of these services, Amazon is beginning to offer monitoring services that could suit simple applications.

By default Amazon includes basic server monitoring with each instance. This data can be accessed via the instances tab in the EC2 console. Simply select a running instance, select the monitoring tab at the bottom of the screen and you can see disk read/writes, cpu load, and network utilization. Advanced monitoring checks server status every minute (as opposed to every 5 minutes) and enables longer logging (2 months).

Both advanced and basic monitoring include the ability to setup alert notifications. In the following exercise we will setup alarms for our services running on our server.

1. In the AWS console, go to the CloudWatch Tab
2. Browse around the service, take a look a the monitoring data that is available. When ready, lets setup a few alarms our system including CPU Load, disk utilization, and service availability
3. First lets setup a CPU Load alarm
  - a. Click on “Create Alarm”
  - b. In the window, select EC2: Instance Metrics from the dropdown
  - c. Find your running instance and the CPU Utilization metric
  - d. Click Continue
  - e. Assign a description and set a low CPU Utilization alarm

The screenshot shows the 'Create Alarm Wizard' in the AWS console. The current step is 'DEFINE ALARM'. The 'Name' field contains 'TestingBox\_CPUUtilization' and the 'Description' field contains 'Just testing'. Under the 'Define Alarm Threshold' section, the threshold is set to '>= 1' for '1' minutes. A line graph titled 'CPUUtilization (Percent)' shows the metric over time, with a red horizontal line at the 1% threshold. The graph shows several peaks above the threshold. At the bottom, there are 'Back' and 'Continue' buttons.

- f. Select “create new topic” from the action dropdown and enter a topic and destination email addresses

When Alarm state is	Take action	Action details	
ALARM	Send Notification	Topic: <input type="text" value="SERVERLOAD!"/> Email(s): <input type="text" value="mitchee@wfu.edu"/>	<input type="button" value="ADD ACTION"/>
<small>A topic is a communication channel that can be reused across Send Notification actions. Please enter a new topic name and a list of comma-separated email addresses.</small>			

- g. After you setup your subscription you can monitor it on your SNS (Simple Notification Service) tab. Check your email to approve the subscription.

It appears that this service is not setup by default to check specific services (e.g. MySQL, Apache) or applications. Deciding whether or not using this service to monitor CPU and I/O levels and another hosted service to monitor specific applications as opposed to using a separate server running Nagios comes down to local preferences (needed capacity, comfort level, other services that could run on a monitoring system). Some options for web-server level monitoring include <http://browsermob.com>, <http://siteuptime.com>, and <http://mon.itor.us>.

If you are looking for a more customizable local solution nagios (<http://nagios.org>) is an ideal linux based open source platform. Want to try installing nagios on your primary or secondary server? Try out the instructions at <http://cloud.lib.wfu.edu/blog/tech/2010/09/29/nagios-installation-and-configuration/>

## 8. Automatic server management using puppet

---

There are a number of ways to approach server configuration and management. The approach we have been working toward up to this point is the 'golden-egg' approach. Using this method you carefully build and document your server, taking images along the way. When you need to change your configuration you take an image of your server, launch a new one, upgrade /change it, test it and deploy it.

The golden egg approach works well and having the ability to move volumes easily, swap IP addresses at will and avoid upgrade mistakes by using short-term test boxes will typically keep you out of trouble there is another approach that focuses more on systematic documentation of your production environment. In this approach you use a server configuration tool such as Puppet, Chef or CFEngine to configure your server, check server settings regularly to ensure stability and customize new server deployments.

In this exercise we will explore Puppet, a server configuration tool and use it to automatically configure a new webserver. To keep things simple we will not repeat instructions that we have been through before (e.g. server launching). Refer to earlier portions of the tutorial when needed.

### Launch a puppet server

---

In this exercise we are going to launch two new servers to setup our puppet system. One server will be out puppetmaster server (e.g. the configuration server). The second server will be out puppet client (e.g. the managed server). If you are really successful you could start pushing configuration directives to your already running wordpress server!

1. First, lets launch two new servers. Use the Ubuntu image from alestic. Make them small servers. Be sure to put them both in the same availability zone and make sure they share the same security group. Name one puppetmaster and the other puppetclient.
2. Connect to both machines. After you connect you need to get the DNS name of the machine and the internal IP address. You can get the DNS name using the command `cat /etc/hostname`. You can get the internal IP address by pinging your DNS name.
  - a. Keep this information in a text editor.
3. Lets keep this simple by being root for this entire exercise.
  - a. `Sudo su -`
4. Update your boxes. Run `apt-get update` and `apt-get upgrade` on both machines
5. We need to update `/etc/hosts` on both machines so that they can get to each other without consulting DNS. We also need to have the machine name resolvable for our security certificates to work .
  - a. On puppetclient edit `/etc/hosts` and add a line pointing to the IP address of puppet master and the domain name (e.g. `10.192.215.33 domU-12-31-39-0E-D4-D3`).
6. Now lets install puppetmaster and puppet client
  - a. On Puppetmaster – `apt-get install puppetmaster`
  - b. On Puppet client – `apt-get install puppet`

## Configuring your puppet master

---

Our puppet master configuration includes the definition of nodes (e.g. servers that we manage) and modules/classes (e.g. applications, files or settings we apply to these servers). We are going to start simple and configure our new server to run apache.

7. In `/etc/puppet/manifests` create a file called `sites.pp`
8. The file `sites.pp` will do double duty. It will both tell puppetmaster where to find the classes of things to configure and what the default server configuration should look like. Make your `site.pp` file look like:

```
#/etc/puppet/manifests/site.pp
import "classes/*"
node default {
  include apache
}
```

This file imports the folder `classes` and includes the `apache` class in the default server configuration.

9. Save the file and create a `classes` directory
10. Inside of that `classes` directory, create a new file called `apache.pp`. Make that file include the following information.

```
class apache {
  package { apache2: ensure => installed}
  service { apache2:
    ensure => running,
    require => Package[apache2],
  }
  group { apache: gid => 1001 }
  user { apache: gid => 1001 }
}
```

This file tells puppet to make sure that the package `apache2` is installed, that it is running and that the `apache` user belongs to the `apache` group.

11. Now lets turn to our puppet client. Go to the puppet client server and edit `puppet.conf`. Include the directive `server=DNSNAME.compute-1.internal` (NOTE: This proved difficult to diagnose. Your FQDN may be `ec2.internal`. The best way to proceed is to check the certificate name on puppetmaster in `/var/lib/puppet/ssl/private_keys`).
12. Save and close the file
13. Now attempt to test you puppet configuration

14. Run the command `puppet agent -test`

```
root@domU-12-31-39-0B-04-E3:/etc/puppet# puppet agent --test
info: Caching certificate_revocation_list for ca
info: Caching catalog for domu-12-31-39-0b-04-e3.compute-1.internal
info: Applying configuration version '1316716181'
notice: /Stage[main]/Apache/Package[apache2]/ensure: ensure changed 'purged' to
'present'
notice: /Stage[main]/Apache/Group[apache]/ensure: created
notice: /Stage[main]/Apache/User[apache]/ensure: created
info: Creating state file /var/lib/puppet/state/state.yaml
notice: Finished catalog run in 14.04 seconds
```

You should see some output similar to the screenshot above. If this command works you are ready to move forward! Lets have our puppet client request a real certificate

15. On puppet client issue the command `puppetd -waitforcert 60`

Now go to puppetmaster and check the certificate authority. Puppet uses its own certificate authority to provide security. This simplifies the process of connecting the two servers and keeps unwanted servers out.

16. On puppetmaster issue the command `puppetca -list -all`

You should see an outstanding request from your client. Approve the certificate request.

17. `puppetca -sign DNSnameOfRequest`
18. Check to make sure that your certificate is signed by running `puppetca -list -all` again

Lets return to our puppet client and run another test to see how things are going.

19. Run `puppet agent -test`

You should see a test that shows apache being installed. If not it is time to start diagnosing why not. If you have a successful test, lets apply our configuration. Ensure that you do not have apache installed by trying to `cd` to `/var/www` or by trying to go to your machine name in a web-browser.

Force puppet to refresh its configuration

20. Run the command `puppetd -vt` on the puppet client

## Test to make sure puppet is working

---

Try to hit the web again. It should work! This process involved connecting to our puppetmaster, scanning the configuration files, finding the classes that are applied to our puppet machine and implementing the necessary changes. Lets prove that this works by automatically stopping our apache server using puppet. Return to puppetmaster find your `apache.pp` file and changed "ensure => running," to "ensure => stopped,".

21. Check to make sure that apache is running on your puppet client (`ps afx | grep apache`).
22. Force a reload of your configuration by running `puppet -vt`.

## Serving files with puppet

---

Watch apache stop! In order to effectively configure our server we also need to be able to deliver files to the new server. Lets do this by turning on fileserving and sending a customized index.html file

23. On puppetmaster return to /etc/puppet and edit fileserver.conf
24. Under the files directive add an allow \*.internal (e.g. all internal dns names). This is not very secure and you would tune this more in a production environment. Notice that our files are served out of /etc/puppet/files
25. Create the files directory (/etc/puppet/files) and an apache directory underneath that.
26. Add a new index.html file with some custom content in it.
27. Return to site.pp in /etc/puppet/manifests and add a new include for apachefiles (e.g. include apachefiles) under the include apache directive. Save and exit
28. Change directories into classes and create a new file called apachefiles.pp. Make the file look like:

```
class apachefiles {
  file { ["/var/www/index.html":
    owner => "apache", group => "apache",
    source => "puppet:///files/apachefiles/index.html",
    notify => Service["apache2"],
    require => Package["apache2"],
  ]
}
```

This file tells puppet to deploy a new index.html file and set permissions on it only if the apache service is installed.. Note that the URI in source points to a file folder. This location is specified as a mountpoint in /etc/puppet/fileserver.conf.

With this in mind, lets think about what it would take to configure wordpress in a production environment. You would have to:

- install mysql
- install phpmyadmin
- install php
- install wordpress
- deploy wordpress customized files

This would take some time to debug and develop. Luckily there are puppet recipes ([http://projects.puppetlabs.com/projects/puppet/wiki/Puppet\\_Patterns](http://projects.puppetlabs.com/projects/puppet/wiki/Puppet_Patterns)) to work from for many situations. If you flew through this demo and still have time start trying to implement each service. Think about the role that a distributed server architecture could play (e.g. a separate database server) in helping or hurting this process.

## 9. Using client-based tools to manage AWS services

---

The AWS community has developed a number of client-based tools that facilitate access to and management of the AWS platform. These tools include mobile phone based management applications, web-browser plug-ins to manage EC2, S3 and EBS resources and stand-alone clients that integrate with S3 as a remote data storage service. This guide documents a few of the applications that might be of interest.

### Mobile management applications

---

1. Decaf
  - a. Android - <http://www.androidguys.com/2010/02/11/manage-amazon-ec2-cloud-decaf/>
2. iAWSManager (Lite):
  - a. iOS – Launch, manage, terminate instances on AWS/EC2/S3
3. S3 Cloud
  - a. iOS - browse, manage, download files located in Amazon S3

### Browser plug-in tools

---

1. Amazon S3 Organizer
  - a. <https://addons.mozilla.org/en-US/firefox/addon/amazon-s3-organizers3fox/>
2. ElasticFox firefox EC2 extension
  - a. [http://aws.amazon.com/developertools/609?\\_encoding=UTF8&jiveRedirect=1](http://aws.amazon.com/developertools/609?_encoding=UTF8&jiveRedirect=1)
  - b.

### Stand-alone clients

---

1. Jungledisk
  - a. <http://jungledisk.com> - Operating System Plug in that facilitates syncing of data from local computers to Amazon S3
2. s3browser
  - a. <http://s3browser.com> - freeware that enables user management and ftp operations for s3

### Tool directories

---

1. <http://www.aboutonlinetips.com/amazon-s3-tools/>
2. <http://www.datacenterknowledge.com/archives/2010/02/04/cloud-monitoring-services-a-resource-guide/>

## 10. Cleaning Up – Making sure you don't leave data / servers

---

Although Amazon AWS is fairly inexpensive you are charged for servers that are running and for all data that exists on the S3 and EBS systems. Luckily, cleaning up is fairly easy –

1. Terminate all instances
2. Delete all images
3. Delete all snapshots
4. Remove any remaining EBS volumes
5. Remove any service alerts

That should cover everything we did in this set of tutorials.

# 11. More fun things to try

---

## **Puppet configuration of EC2**

---

Using Puppet to configure your EC2 systems. I have only just started working on this. If the first set of tutorials was easy and you are wondering what to do with the rest of your time you could try to setup an EC2 server running puppet. Puppet is an automatic server configuration platform. There are some interesting websites that talk about deploying Puppet on the web.

## **Using Beanstalk to deploy web applications**

---

Beanstalk is a new platform offered by Amazon that uses AWS (EC2, S3, ElasticCloud) resources to automatically deploy and scale Tomcat based applications. The service is somewhat limited at the moment but offers some interesting opportunities to run applications without the server management overhead

## **Running databases using the RDS service**

---

Amazon RDS is a relational database service (MySQL only) that supports continuous data protection, automatic full/incremental backups and other database management features. RDS offers you the ability to reduce the services hosted on your amazon EC2 systems. With proper planning, you could move your permanent resources out of EC2 (via RDS and EBS), making your AWS platform more resistant to server issues.

## **Installing other server applications**

---

We only touched the tip of the iceberg of server configuration this morning. There is a lot of configuration and software installation to take care of if you want to see how suitable the AWS platform is for general computing.