Deploying on Amazon EC2

Introduction

Amazon Elastic Compute Cloud (Amazon EC2) is part of Amazon Web Services. It provides "resizable compute capacity in the cloud"---in other words, it allows you to run what are essentially virtual private servers of various sizes and capabilities. It is relatively straightforward to set up a WebObjects application server on an EC2 instance. There are several flavours of Linux available as base images to customize.

Automated Deployment on Amazon EC2

The process is not complex, but it does contain a number of steps which are contained in public examples and you'll need at least a peripheral understanding of EC2, S3 and UNIX tools. The steps taken by these example scripts are outlined as follows:

1. Launch an instance of Amazon Linux AMI of whatever Type (size) you require
2. Install required packages (httpd, mod-ssl etc)
3. Download, install and configure WO deployment tools (JavaMonitor / wotaskd)
4. Download, build and install the WO Apache adaptor
5. Download and install dependancies (e.g. ImageMagick, http-devel)
6. Download and install custom httpd.conf/ssl.conf and JavaMonitor configurations
7. Download, install and boot your apps

This process starts with a clean AMI and within a few minutes, you'll see a completely setup and fully running instance of WO/Wonder. Using your own AWS_ACCESS_KEY_ID and your own AWS_SECRET_ACCESS_KEY you will be able to modify the example scripts and rapidly build a fully complete and reliable deployment of your own.

> To find out the Access Key and Secret Key, check this blog post.

> **Dependencies**
>
> - You need to have an Amazon Web Services account https://console.aws.amazon.com
> - The community scripts depend heavily upon "aws" (http://timkay.com/aws/), a command line utility for interacting with amazon ec2 and s3.
> - The community scripts are built to support Amazon Linux (http://aws.amazon.com/amazon-linux-ami/). This process should work with most any flavor of UNIX, but is tested and reliable on Amazon Linux.

Hello World Walkthrough

In this walkthrough example we'll build a brand new Amazon Linux instance, running WOnder's AjaxExample and AjaxExample2 apps. To get started, grab these scripts and pop them in a fresh directory on your Mac:

http://webobjects.s3.amazonaws.com/launch.sh

http://webobjects.s3.amazonaws.com/helloworld-userdata.sh

- chmod 775 both files
- Edit both and put in your AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.
- Edit launch.sh and comment in the appropriate line to launch in the region and AMI of choice, here is an example using a micro instance in the us-east-1 region:

```
aws run-instance ami-38c33651 -i t1.micro -a public -n 1 -g Basic -k aws -f
./helloworld-userdata.sh --region=us-east-1
```

- Finally, run the launch.sh script

With no changes outside adding your own access and secrete keys, this script will run for a minute or two and produce a complete and running instance visible in your Amazon EC2 Management Console. In your Amazon EC2 dashboard you'll see the new instance, and you'll be able to find it's public DNS record and you can connect to the running WO/Wonder Apps and the configured JavaMonitor, using your browser.

Your own scripts

The best place to start with building your own scripts is to download everything used in the hello world example to see how it all works. It's not rocket science, but a useful place to build from. I cannot stress enough that you need to change everything to operate from your own private s3 buckets - unless of course you do want the world to be able to boot instances of your applications!

Uploading your own apps to S3 from Hudson

In the above example we pull the demo apps down from a public web server using wget. When you come to automate the deployment of your own apps you will want to serve them up from a private bucket or server. Amazon's S3 service using the aws command line tool is an easy way to move your own app.

Here is how we do it:

1. Create a Hudson job that consolidates all of the build products that we want to shift to s3 into a single directory. We do this using the "copyarchiver" plugin.
2. Install the aws command line tool on your Hudson server. Remember to install it as root so it is not user-specific.
3. A Hudson job does the uploading, simply executes a shell with the following command:

```
AWS_ACCESS_KEY_ID=[YOUR_KEY]; export AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY=[YOUR_SECRET_KEY]; export AWS_SECRET_ACCESS_KEY
cd /location/of/your/products
for i in *.tar.gz; do s3put your-bucket-name $i ; done;
```

Elastic Load Balancing enables you to balance incoming requests between an array of EC2 instances. ELB supports SSL termination and sticky http/https sessions. The most reliable way to enable sticky sessions with WO is to store session id's in cookies, then choose "application server generated cookies" as your preferred method for sticky sessions and have the ELB monitor your wosid cookie.

Other resources

You can get more information about Amazon EC2 in the presentation Ubermind made about it at WOWODC West 2009.

WOlastic is no longer supported by Übermind

In a post to the WebObjects development mailing list in February 2011, Joe Kramer from Übermind notes that "WOlastic is no longer supported by Ubermind." From the engineer behind WOlastic:

> *The original intent of WOlastic was to quickly deploy secure WO applications into the cloud. – Public AMI's that were prepackaged were probably great for the time but now it doesn't seem as relevant as you could probably script the entire process with official AMI's that are maintained by the major Linux distros. I'll probably take down the Public AMI's later this week that I initially put up for WOlastic. I did a couple updates to the AMI's, they were barely used, so I stopped supporting them.*

Also see Deploying via Chef.