Scenario Week 1

October 28, 2016

Introduction

The goal of the week is to get as far as you can with setting up a web application to run on a VM-based server using Amazon Web Services (AWS). Ideally as much as possible will be automated (i.e., controlled by scripts or programs).

The following are the key goals, try to achieve as many as you can:

- You have a server set up to run your application "in production", meaning it is available for general use. A web server will be needed.
- You should use git version control to manage your source code and other assets. Git should be used properly with all members of the team contributing to the same repository with regular commits, branches (e.g. for experimental work), and full commenting.
- As much as you can manage, the deployment of the application should be automated (i.e., a program or script does the work). Deployment should fetch the application from version control, configure it, and start it up. Deployment is controlled from one of the development machines.
- The application should restart automatically if the server restarts - you are aiming for maximum availability without human intervention.
- The application will store its data in a database, which should be backed up on another server on a regular basis.
- It should be possible to restore the application to the state of one of the backed up databases.
- The back up should occur frequently to ensure that in the event of the loss of the production server (e.g., everything wiped), then as little data is lost as possible. The actual data loss will depend on when the last backup took place, and you need to balance backing up with storage space. Also how do you backup the database when the application is running, as data might not be in a consistent state at the moment of the backup? This means that backup may not be as simple as copying the database, maybe you need to read and transfer data to another database on another machine?

Amazon Web Services (AWS)

Each member of your team will need to sign up for a Amazon Web Services **starter account**. To obtain an AWS starter account head to:

https://aws.amazon.com/education/awseducate/

and click on "Get Your Student Account". Then fill in your details making sure to use your UCL e-mail address and make sure that you select the option to sign up for a start account.

Version Control - Git

You should make full use of version control to store your source code, scripts, and other files you create (don't store binary code or programs in version control). As mentioned above, you should use best practices with all team members using the same repository and commits being regularly made.

You can use either GitHub or Bitbucket to host your shared git repository. Alternatively, you could use the department's own git service at http://gitlab.cs.ucl.ac.uk. Also visit the git project website (https://git-scm.com) for further information.

There are many resources to learn about git, one is the UCL Lynda.com course: Git Essential Training with Kevin Skoglund http://www.lynda.com/Git-tutorials/Git-Essential-Training/100222-2.html?org=ucl.ac.uk

While on Lynda.com have a look for other relevant courses. Each course is divided into a number of fairly short videos so you can focus on what you need without having to watch everything all the way through.

Servers

In your group each member will have a VM server to contribute to the development work. Servers are usually run headless, without a GUI, so get used to using the command line and configuring servers over ssh connections.

Production Server

One server should be designated the Production Server. This is where the working version of the web application should be deployed to for general use. By the end of the week your aim is to have a method of deploying the application to the server, automating as much as you can. Ideally the server should be stripped down to the essential software needed.

Staging Server

Another server should be the Staging Server, which is used to practice and test deployments. In a real life situation you would never deploy to a production machine without full testing on the staging machine first. The staging server needs the same software versions as the production server but can be set up to do more logging, experiments and so on.

Database Backup

A third server should be the database backup server. You need to explore how backups are stored. Are they simply timestamped copies of a database? If so can they be restored to use? Or do you need to run a database server which manages separate local databases, with the data copied using queries? We'll assume that the backup server is in a separate location to the production server, and is itself backed up in some way.

Development Machine

To be used for development. You can use your own machine or a lab machine (but note that lab access is not guaranteed).

Web Application Specification

The purpose of developing a web application is to provide an application to deploy, not to develop a large or complex application in its own right. The specification is as follows:

- The application implements a straightforward citation/reference list manager where information sources can be recorded along with any relevant links or notes.
- A user can create an account and then create, edit, display and delete references.

- A reference contains at minimum a title, link, and note. A reference list is the collection of all references held on the database.
- The lists and user accounts are stored in a database.
- A user has a username/email address and password, as is very familiar on many websites.

You can add to this if you want, but bear in mind that the purpose is not to create an elaborate application.

It is recommended that you use one of the well known web application frameworks:

- Django (https://www.djangoproject.com) this is implemented in Python language.
- Rails (http://rubyonrails.org) this is implemented in the Ruby language.
- Grails (http://grails.asia) this is implemented in the Groovy language.

All three have the advantage that they use object-relational mapping meaning that you model the data using objects and classes, which are automatically mapped to a database, such as MySQL (http://www.mysql.com) or PostgreSQL (http://www.postgresql.org). This means that you don't have to separately design and implement a database.

If you don't feel able to learn one of these frameworks, then there are many others out there. Probably the most common option, though, is PHP (https://secure.php.net).

Scripting

One of the key aspects of this project is the automation of setup and deployment using scripting. AWS provides many different means of achieving scripted automation (i.e. AWS Command Line Interface, AWS Elastic Beanstalk, etc.) and some resources to help get you started:

- Amazon Web Services in Action, Michael Wittig and Andreas Wittig (Available online through Safari ProQuest)
- Up and Running with Bash Scripting with Scott Simpson (http://www.lynda.com/Bash-tutorials/Up-Running-Bash-Scripting/142989-2.html?org=ucl.ac.uk)
- Amazon Web Services Essential Training (https://www.lynda.com/AWS-tutorials/Amazon-Web-Services-Essential-Training/163929-2.html)

Questions?

On Tuesday, Wednesday, Thursdays and Friday there are timetabled sessions where you can get feedback and help (see the timetable on Moodle). Also make use of the COMP204P forum on Moodle.

Marking

The Scenario is marked out of 100. Marking will be done on Friday. Each group should book a marking slot, where the marking will be done by a member of staff or TA. They will mark against a tick-list of achievements (as listed at the start of this document), and on the overall quality of what you produce. The tick list will determine 70% of the mark.

In addition, each group will have 100 points to divide amongst the group members, based on contribution and team spirit. The points **cannot** be divided evenly, and no two members can have the same points. The team must decide on how the division is done. The points will determine 30% of the mark.

Each team member will receive a final mark based on the tick list and points.