CS3226

WEB PROGRAMMING AND APPLICATIONS

LECTURE 11 - WEB SERVER

DR STEVEN HALIM

STEVENHALIM@GMAIL.COM

OUTLINE

- Preliminaries
- Setup
- Maintenance

PRELIMINARIES

TARGET AUDIENCE: SOME OF YOU

Some of you may never apply the techniques in this lecture as you do not have (or do not want to pay to have) access to a root (Administrator) account of a web server

But some others probably already have (or will have) such access, so this short lecture may contain some interesting knowledge for this group of students

REASON 1: CS3226-1 VM LIMITATION

The CS3226-1 Virtual Machine (VM) web server is a *shared machine* for 57 of you in this module

Your account will be removed once this module is over and none of your web application files (or any other files that you have in the VM) will survive beyond this semester

How to make web applications that you build in CS3226 live for longer period beyond this semester?
(similarly for your *future* potential web application(s))

One valid answer: Set up and maintain *your own* web server
(Note: Not just about using a web hosting service out there)

REASON 2: PERFORMANCE

Some of the Performance and/or Scalability tips mentioned in the previous lecture can only be applied *if* you have access to the root (Administrator) account of a web server

Otherwise you are basically tied to whatever hardware specifications and/or software libraries that are available in that web server, especially the server-side technologies...

GETTING YOUR OWN WEB SERVER

There are two ways:

1. Sacrifice (one of) *your own* personal computer (usually a dedicated *desktop** computer), setup and run a web server on it 24/7, pay a huge electricity and Internet bill...
2. You can *purchase** cloud-based web (hosting) service
    1. Digital Ocean (the basis of this lecture note *for now*)
    2. Amazon Web Services (I haven't try, but promising)
    3. Google App Engine (I haven't try, but promising)
    4. IBM Bluemix (I haven't try)
    5. There are many others, search the Internet for more options (and more confusion)...

WHAT'S NEXT?

Once you have access to your own root (Administrator) account of a web server, you can set up the web server, upload your web application files and setup its settings, and let the web application run 24/7

But once a while, you have to maintain your web server as no one else will (or can) do it for you, failing which your web server (and thus your web application) performance will gets slower and slower and its security will get more and more compromised

WEB SERVER SETUP

FOCUS ON LAMP STACK

LAMP = Linux, Apache, MySQL, and PHP

For the preliminary version of this lecture note (started 2016), we will not digress too much and talk about many other web server options out there that I have not tried (WA/IMP*, MEAN, etc)...

DIGITAL OCEAN* DROPLET

Not-really-live demonstration

I will not actually create the droplet live in front of you...

Things to be considered:

1. Choice of Linux distro (Ubuntu, Debian, CentOS, Fedora, etc...) or from a software bundle (this will save you time from installing the "AMP" component yourself)
2. How much do you want to pay per month (or per hour)
3. Data center region (where your main* web application visitors are geographically located)...

WHAT'S NEXT?

You can now host your web application by copying/uploading your web application files to a certain designated document directory (depends on your Apache setting, the default for Ubuntu 14.04 LTS is `/var/www/html`)

The index.html (or index.php) stored in that directory is is the one that will be served when you entered the IP address of your Digital Ocean's droplet in a web browser, e.g. visit http://128.199.88.135 (we will talk about Domain Name registration in the next lecture)

MAINTENANCE - LINUX SPECIFIC

Every year, there are various (new) Linux distro released to public, e.g. the 2016 version

Upgrading the OS (the "L" in the LAMP stack) will likely entail the need of upgrading everything else (the "AMP"), so do such upgrade when there is a (critical) security upgrade...

As root, you can perform these tasks (not exhaustive):

1. Set up owner of PHP (or other server-side) files to "apache" user so that you can disable the read access from group or other (the counter of this)

APACHE SPECIFIC

Likely the no 1 web server *software* as of 2016

As root, you can perform these tasks (not exhaustive):

1. Remove big/old access/error log or change log setting
2. Turn on/off per-directory setting (.htaccess)
3. Set Virtual Host: 1 web server, multiple websites/apps
4. Set file caching options (especially for static files)

MYSQL SPECIFIC

As a root user, you have a root MySQL account that you should not directly use in your PHP script

With that MySQL root account, you can perform the following tasks (not exhaustive):

1. Create other MySQL account(s) and granting only specific priviliges for that account(s)

PHP SPECIFIC

As root, you can perform these tasks (not exhaustive):

1. Edit PHP configuration file (php.ini), e.g. display_errors, file upload settings

UPGRADING THE LAMP STACK

Sooner or later, the Linux distro, the Apache version, the PHP version, and/or the MySQL version that you have will get outdated

When a new version appears, you have to decide whether to upgrade and when to do so...

THE END

This is a short lecture (started March 2016) and will be upgraded over time