COMP260 – Distributed programming **Worksheet WK 5**

Remote Hosting

Introduction
In this worksheet, we will look at hosting Python applications on remote (DigitalOcean) servers using PuTTY, ftp clients and servers, and a git client.

Unlike our desktop development environments, DO servers are 'headerless' meaning that they no GUI environment and must be communicated with using remote command line services like PuTTY or other ssh clients.

Your goals for this worksheet align with Assessed Worksheet 2 (MUD as a service & Operations):

1. Establish a connection to your DigitalOcean server
2. Install your MUD server application onto it
3. Run your MUD server reliably

The steps below will help you to achieve these goals. In general, most linux server functionality can be enabled by following DigitalOcean how-to guides and this is a common approach to server management and is nothing to be concerned over.

Establishing connection to your remote server
You each have a separate DigitalOcean server to develop and host your services on. If you run into any issues, you will need to speak with Andy Smith to re-establish your server. Given the intrinsic value of remote servers, please ensure that you maintain security with usernames and passwords.

I will give you your account details in the workshop, please log on using PuTTY (or any other suitable ssh client) and follow the server's instructions for resetting your password.

To connect to your server, I recommend PuTTY (https://www.putty.org/). This can be installed on an external drive and used on your lab PCs. PuTTY will allow you interact with the server machine using a command-line interface. If you are unfamiliar with Linux, this link http://mally.stanford.edu/~sr/computing/basic-unix.html will help get you up to speed. Also, learning material is available on LinkedIn Learn, Pluralsight and Lynda. However, you do not require a deep understanding of Linux command line for the assignment.

Uploading files – 1. Copy and paste
Given that your server applications are likely to be quite small, one approach to uploading files to the server is to copy and paste code from your PC into a file using nano, a linux text editor. To do this, launch nano:

```
Nano <my file>.py
```

copy code from your pc and paste it into the file window. You can quit nano with ctrl+x, nano will ask you to save the file. Once you are back in the command shell you can see your file with text.

Uploading files – 2. Git (or any other version control package)
Whilst copying text from one machine to another will work, it's not a particularly elegant solution. An alternative is to install a git client on your server and the pull your development work from your remote git server (github, bit bucket etc).

DigitalOcean provides a how to for this:
https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-18-04

In addition, you can install a git server on your server to host your own repos. However, do speak to Andy Smith so you know when your server will be decommissioned.

DigitalOcean provides a how to for this:

https://www.digitalocean.com/community/tutorials/how-to-set-up-a-private-git-server-on-a-vps

It's also worth noting that other open source version control systems are available for you to host on your server:

https://help.ubuntu.com/community/Subversion

Uploading files – 3. FTP
Another approach to managing files between client and server is to use FTP (File Transfer Protocol). This allows you to set up a file manager such that you can copy files between client and server, typically using Filezilla (https://filezilla-project.org/) as a client and vsftpd as the host.

Again, DigitalOcean provides a how-to guide for this, though it is more complex than the other 2 methods:

https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-ubuntu-16-04

Installing Python
To use python on the server, you will need to set up Python3. It's worth pointing out that Python on linux is not the same as Python on windows / OS X and Linux generally a few steps behind. You are likely to be running 3.5.x on Linux which may lead to some version issues. Also, Python 3.x applications are run from python3, not python (as that is for 2.x). DigitalOcean provides a how-to guide for this:

https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-programming-environment-on-an-ubuntu-18-04-server

Running remote applications
To run an application on the server, use Python 3, the format is:

```
python3 <filename>.py
```

Running applications this way has the advantage that any output (normal or abnormal program execution) will be echoed to the ssh window the application is running from, therefore it is a good approach to debugging.

By default, your application will stop executing when you close the PuTTY session that you are running your app from. To get around this, you can use the nohup command to keep the application run on shell exit. The format for nohup is:

```
nohup python3 <filename>.py &
```

However, there will be no output while the program is running, it is all collected in nohup.txt and may be read using a text editor in another window.

To stop a app that is running through nohup, use ps to determine the ID of the app and then kill the task with kill <process_id>, look at this website for more detail: https://www.booleanworld.com/kill-process-linux/

Other functionality
Linux will support multiple users; it is common to create non-root (superuser) accounts for performing day-to-day activities (devops).

The server will also run as a www server and mail server if you are prepared to install the correct applications to do so.

Please use your server responsibly.