Show Submission Credentials

P0. Linux warmup Linux Warmup

- ✔ Essential Skills in Linux
- ✔ Installing Software and Configuring Services
- ✔ File Ownership and Permissions
- ✔ Disk Management in Linux (Optional)
- ✔ Conclusion

Essential Skills in Linux

Essential Skills in Linux

In this course, you will be working with various cloud resources that will be provisioned on-demand, and accessed remotely through SSH or putty. We will primarily be using linux-based instances, most of which run some version of Ubuntu. Therefore, before getting into the projects, we have assembled a simple primer to get you started on some of the various fundamental concepts and skills which will help you tremendously in tackling the various projects in this course.

Scripting Languages and References

Typically we would interact with the cloud instances through the command line interface (usually bash on linux instances). In order to perform and/or automate more complex tasks using the command line, a good knowledge of scripting languages and techniques (bash, python, etc.) is required.

You will need to focus on the following topics:

1. Scripting syntax, local variables and environment variables.
2. Input and output redirection using unix pipes.
3. Using tools such as `grep`, `awk`, `sed`, etc. to perform simple text manipulation inline or within bash scripts.

We have prepared several self-study, interactive notebooks for you to practice on the topics above, which will help you to get prepared for the course. Please finish the Jupyter Notebook primer before you move on!

Installing Software and Configuring Services

Installing Software on Ubuntu and Debian-based Linux environments

As you are working on the various projects in this course, we will supply you with a task-specific virtual machine image which contains most of the necessary software, tools and data for you to complete the specified task. In any case, you should also know the commands necessary to install software on these instances on your own. We will mainly be working with Ubuntu-based instances in this course, so you should familiarize yourself with the workings of the Advanced Packaging Tool (APT), using the command `apt-get`. `apt-get` lets you install software packages on Ubuntu and other Debian-based linux environments.

`apt-get` is invoked using the following command:

```
sudo apt-get install
```

For example, to install GNU `emacs` on an Ubuntu machine:

```
sudo apt-get install emacs
```

To install `vim`:

```
sudo apt-get install vim
```

Information

Installing Software on Red-Hat based distros

The equivalent command to install software on Red-Hat based Linux distros (Fedora,CentOS etc.) is:

```
sudo yum install
```

We use `sudo` to execute `apt-get` commands because this requires root access to install packages in the system. In addition, here are some useful `apt-get` commands:

`sudo apt-get update` will update the package index from their sources. This is typically used to refresh the package index with the latest packages and their versions.

`sudo apt-get upgrade` will update all the existing packages on the system with their latest versions.

Configuring Services

There are situations in which you must configure an application to run in the background as a service. Usually, this is the case with web and other types of server processes. For software that is packaged for your Linux distribution and has already been installed, please refer to the man page or software instructions for more details. In a nutshell, services must be installed in the `/etc/init.d` directory in your Linux instance. This allows these scripts to run on startup, and for you to use the service command to start or stop them. For example, if mysql scripts are present in the `/etc/init.d` directory, you can issue the following command:

```
service mysql start
```

`service` can be used to start, stop and restart services.

For more information on creating your own init.d startup scripts, please refer to the TLDP guide on Starting your own software automatically on Boot (http://www.tldp.org/HOWTO/HighQuality-Apps-HOWTO/boot.html).

File Ownership and Permissions

File Ownership and Permissions

Often, you will need to change file ownership and permission for various reasons. For example, SSH key-pair files must always have read-only access for the current user to keep credentials safe from tampering.

Similarly, certain programs and services may require root ownership to execute correctly, or must be owned by a non-root user, and executed in non-root mode for security purposes.

Linux file permissions are defined for three entities: `user`, `group` and `others`. In this context, the `user` permission corresponds to the owner of the file. Similarly, there are 3 types of permissions that can be defined for each of the entities: read (`r`), write (`w`) and execute (`x`). They can also be defined numerically as follows:

#	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-

1	execute only	--x
0	none	---

When file permissions are expressed as numbers, they are in 3 digits, each denoting `user`, `group` and `others` respectively.

For example, to give permissions to anyone (user, group, and others) to read, write and execute a file, the permission can be expressed numerically as `777` (which means rwxrwxrwx permissions for user, group and others). If you would like to give the owner of the file permissions to read and write, while the group and others get only read, then the permission will be `644` (rw-r--r--), and so on.

`chmod` is the command used in Linux to change file permissions.

It can be used numerically:

```
chmod 777 filename.txt
```

(provide read, write and execute permissions for anyone for filename.txt)

Or it can be used to denote specific permissions:

```
chmod u+x filename.txt
```

(Add execute permission for the user for filename.txt)

For more details, please refer to the `chmod` man page.

Similarly, `chown` is used to change the ownership for a file. As an example, using

```
chown ubuntu:web filename.txt
```

Changes the ownership of `filename.txt` to the user `ubuntu` and group `web`.

For more details, please refer to the `chown` man page.

Disk Management in Linux (Optional)

Disk Operations in Linux (Optional)

In this section, we will cover some of the important commands to perform various types of disk operations in linux. This will come in handy when dealing with the various types of virtual storage options available for Amazon EC2, such as Amazon's Elastic Block Store (EBS).

Managing Partitions

A number of applications can be used to manage disk partitions in linux, including `parted` and `fdisk`. `fdisk` is the traditional partition management software but it has been largely replaced by `parted` as it has support for GUID Partition Tables (GPT) and drives that are larger than 2 TB. For the purposes of this tutorial, we will be using `parted`.

To list all of the partitions on the system:

```
parted -l
```

Partitions will be listed in order as devices under `/dev`. On Amazon EC2 instances, the devices are `/dev/xvd**`. `/dev/xvda1` is the OS partition, while `/dev/xvdb` is the instance store partitions. Please note that RAMDISKs will not show in the output of `parted`. `parted` can also be used to create partitions and set the partition type or file system of each partition (Linux has a variety of file systems available, ext4 is commonly used nowadays):

```
parted /dev/xvdX mklabel gpt

parted /dev/xvdX mkpart db ext4 0% 10G
```

Can be used to label a device as using the GUID Partition Table (GPT) and label a 10 GB partition on the device to use the ext4 file system. This partition is now available to format with the file system specified.

In addition, `parted` can be run on its own without any parameters to start it in interactive mode.

Once a partition has been created using `parted`, you can format it using the `mkfs` command:

```
mkfs.ext4 /dev/xvdX1
```

Mounting Volumes

In order to finally use a disk after partitioning and formatting, it must be mounted to some mount-point on the Linux virtual file system. Typically mount points in Linux are created under the directory `/mnt` or `/mount`. Removable media such as USB/CDROM etc. are mounted under `/media`. You are free to mount these partitions in any directory you like.

```
mkdir /storage/mountpoint
mount /dev/yourdevice /storage/mountpoint
```

Conclusion

Conclusion

This primer is just the beginning! Try running the `man` command for any unix tools that you want to learn more about (i.e. `man bash`).