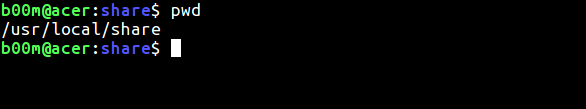
**Day 6 Assignment**

### LINUX COMMANDS

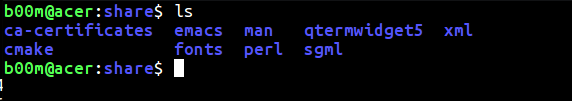
### 1. pwd command

This command prints the location of your current working directory. It’s important to know actually where you’re before going to a parent or sub directories.



### 2. ls command

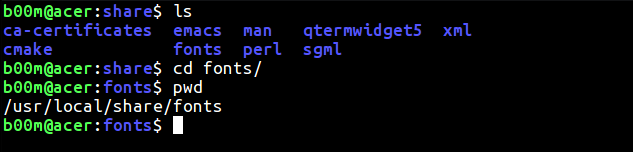
**ls** is one of the most used basic linux commands, used to **print** contents of a directory, by default it lists contents of current working directory(**pwd**).



Example, use ls /usr/bin to list contents of the **/usr/bin** folder.

### 3. cd command

After knowing your **pwd** and getting an overview with the **ls**, it’s time to move around with **cd**command. Clarification, assume you’re on your **Home**directory, you need to go to the **/usr/local/share/fonts** directory, use cd /usr/local/share/fonts.



There’s three shortcut, if you need to move one directory up, use cd .. and go straight to your Home folder with cd, and use cd - to go back to your last working directory.

### 4. cat command

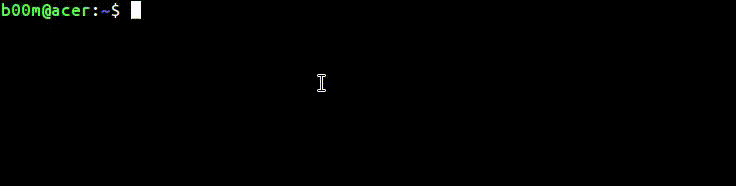
It’s used to print the contents of a file to the screen(**stdout** more precisely), really useful when you want to have a quick look on contents of a file. As example, use cat a\_text\_file to get the inside contents of that file in your screen.

### 5. cp command

**cp** , You can copy files and directories with this command. Typical usage is like cp file\_a file\_1\_copy or cp directory\_a dir\_a\_copy Also don't forget to use proper path when you're coping something to different location.

### 6. mv command

The mv command is used to **move** or **rename** directories and files. To rename a file use mv old\_name new\_name, more details about mv [**here**](https://www.pcsuggest.com/rename-file-linux-command-line/) and [**here**](https://www.pcsuggest.com/rename-directory-linux/).



### 7. rm command

The rm command is used to [remove directory](https://www.pcsuggest.com/remove-directory-linux/) or files. Like use rm -r /tmp/backup to remove everything that folder. Of course you've to be careful before removing anything.

### 8. mkdir command

**mkdir**, it’s used to make a new directory in linux. Example, use mkdir my\_new\_dir to make a new directory named my\_new\_directory. The -pargument is useful, when you don't want to make parent directories manually.

### 9. rmdir command

**rmdir**, if you need to remove a directory, use this command. As example, use rmdir my\_dir to remove that specific directory. More details about the rmdir command [**here**](https://www.pcsuggest.com/remove-directory-linux/).

### 10. touch command

**touch**, It’s the equivalent command of mkdir for files. You can create a blank file with touch command. As example, use touch ~/Public/index.html to create a blank index.html file under the Public directory.

### 11. ln command

This command is used to make link between files and directories. As example, you need to make a symbolic link of the /var/www directory to the /tmpdirectory.

ln -s /var/www/ /tmp/

To un-link that symlink, use

unlink /tmp/www

You’ve to be extra careful with complete path and trailing slashes while linking and un-linking.

### 12. sudo command

**sudo** , that’s an essential yet potentially dangerous command. Whenever you’re getting a Permission denied, Authorization failed or something like that use sudo.

As example, the /var/www directory is not writable by the normal user. So to create a blank **index.html** file under the **/var/www** directory use sudo touch /var/www/index.html

### 13. head command

If you need to print first few lines of a file(any type) then you can use head command. A nice practical example w’d be

head -20 /var/log/syslog

This will print the first 20 lines of the **rsyslogd** log to the stdout. By default head command prints first 10 lines.

### 14. tail command

It’s similar to the head command, but the function is opposite, prints last 10 lines of any file by default. Here’s an example, how to print last 30 lines of the kernel log.

tail -30 /var/log/kern.log

### 15. chmod command

It’s also a very important command, used to change file and directory permission. As the chmod command is a very long topic, so here I’ll explain it in brief.

Basically there’s three type of permission, read, write and execute. Each of them denoted by a number.

* 4 for **read** permission
* 2 for **write** permission
* 1 for **execute** permission

So if you need to set universal read/write permission to a file, you can use

chmod 666 my\_file\_name

Assume you need to make a script executable, you can use

chmod +x my\_script\_name

There’ll be a full chmod tutorial very soon, to explain you in detail.

### 16. md5sum command

You may often need to check if a file tempered with or not. However md5sum is not the safest, but no doubt one of the most used.

An easy example could be finding the checksum of a ISO file

user@host:~$ md5sum ~/OS/slitaz-5.0-rc3.iso  
0d685551f8b0b0bd9caa3a4e66d61a3e ~/OS/slitaz-5.0-rc3.iso

The long string of numbers and digits is the md5 hash of that particular file, just match first and last two characters, that’s enough.

### 17. locate command

The basic command to find files and directories in Linux. As it’s a database driven command, so for the first time you need to build the database, run sudo updatedb and wait for few minutes.

A typical example to locate something could be like below.

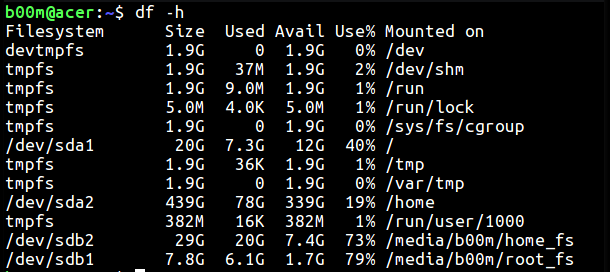
locate -i \*chromium\*

It supports wildcards, and use the -i option to ignore upper/lower case.

### 18. df command

This command is used to check disk space usage on a linux system. The most common usage is like below, used along with the -h flag.

df -h



### 19. du command

If you need to quickly check disk space usage of a file or directory, the ducommand is here.

For a single file, a nice example could be like below,

du -sh /boot/vmlinuz-4.10.10

Or could be like below for a entire directory and it’s contents.

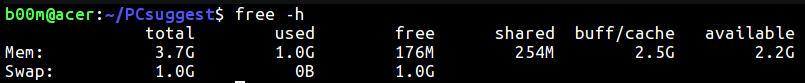
du -sh /opt/google/chrome/

The -s flag is used to suppress unnecessary clutter and -h flag is to make the output more human readable.

### 20. free command

The free command is used to display amount of free and used RAM in the system, also prints the swap space stats.

free -h



Again, the -h flag is used to make the output easier to read by humans.

**Introduction to AWS**

Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow. Explore how millions of customers are currently leveraging AWS cloud products and solution to build sophisticated applications with increased flexibility, scalability and reliability.

AWS covers a broad range and increasing spectrum of cloud services. In a nutshell Amazon delivers cloud-based computing services and resources and like any other cloud services and resource provider this model is based on a pay-as-you-use basis.  It is widely believed that Amazon started the cloud revolution and today it’s a leader in web services.

Amazon is experienced in running large-scale networks and cloud services. Amazon built a large scalable network and infrastructure for themselves and then realised that they did such a great job at it that Amazon could scale and resell the network and services to the world.  Consequently by utilising the AWS offering you as an individual or organisation are afforded the advantages of Amazon’s expertise, experience, geographic resilience and scalability in the area of web services and the comprehensive benefits of cloud computing. With exceptional cost benefits due to the large economies of scale and volumes (hundreds of thousands) of organisations and users globally that Amazon now support and provision the subscription benefits are substantial.

**Some Amazon Services**

The broad products offered by AWS include the following:

**Compute Services**

* Amazon Elastic Compute Cloud (Amazon EC2)- cloud computing and database storage requirements
* Amazon (EC2) Container service-container management service
* Auto scaling-flexible scaling for (EC2)
* AWS Lambda-compute service
* Elastic Load Balancing- end user load management

**Storage and Content Delivery**

* Amazon Simple Storage Service (Amazon S3)- static file cloud storage
* Amazon Glacier-low cost storage for archiving and backup
* AWS Storage Gateway-connecting onsite with cloud-based storage
* Amazon Elastic Block Store (Amazon EBS)-block level storage volumes
* AWS import and export-moving large data volumes in and out of AWS
* Amazon CloudFront- web content delivery service

**Content and Networking Delivery**

* Amazon Redshift
* Amazon ElastiCache
* Amazon Virtual Private Network (VPC)-provision a private cloud
* Amazon Route 53
* Amazon Direct connect

**Database Services**

* Amazon DynamoDM- NoSQL database service
* Amazon RDS- relational database service

**Analytics**

* Amazon Elastic MapReduce (Amazon EMR)-processing data and indexing and search functionalities
* Amazon Kinesis-real time data streaming and processing

**Application Services**

* Amazon AppStream-streaming of applications
* Amazon Cloudsearch
* Amazon SWF
* Amazon SES-email service
* Amazon SNS-notification service
* Amazon SQS- queue service for messages
* Amazon Elastic Transcoder- cloud media transcoding

**History of AWS**

The AWS platform was launched in July 2002,  in the beginning, the platform consisted of only a few disparate tools and services. Then in late 2003, the AWS concept was publicly reformulated when Chris Pinkham and Benjamin Black presented a paper describing a vision for Amazon’s retail computing infrastructure that was completely standardized, completely automated, and would rely extensively on web services for services such as storage and would draw on internal work already underway. Near the end of their paper, they mentioned the possibility of selling access to virtual servers as a service, proposing the company could generate revenue from the new infrastructure investment. In November 2004, the first AWS service launched for public usage: Simple Queue Service (SQS). Thereafter Pinkham and lead developer Christopher Brown developed the Amazon EC2 service, with a team in Cape Town, South Africa.

Amazon Web Services was officially re-launched on March 14, 2006, combining the three initial service offerings of Amazon S3 cloud storage, SQS, and EC2. The AWS platform finally provided an integrated suite of core online services, as Chris Pinkham and Benjamin Black had proposed back in 2003, as a service offered to other developers, web sites, client-side applications, and companies. Andy Jassy, AWS founder and vice president in 2006, said at the time that Amazon S3 (one of the first and most scalable elements of AWS) “helps free developers from worrying about where they are going to store data, whether it will be safe and secure, if it will be available when they need it, the costs associated with server maintenance, or whether they have enough storage available. Amazon S3 enables developers to focus on innovating with data, rather than figuring out how to store it.”. In 2016 Jassy was promoted to CEO of the division. Reflecting the success of AWS, his annual compensation in 2017 hit nearly $36 million.

To support industry-wide training and skills standardization, AWS began offering a certification program for computer engineers, on April 30, 2013, to highlight expertise in cloud computing.

James Hamilton, an AWS engineer, wrote a retrospective article in 2016 to highlight the ten-year history of the online service from 2006 to 2016. As an early fan and outspoken proponent of the technology, he had joined the AWS engineering team in 2008.

In 2016 AWS partnered with Digital Currency Group to create a laboratory environment allowing companies to experiment with blockchain technologies.

In January 2018, Amazon launched an autoscaling  service on AWS.

**Why AWS**

There are a number of cloud providers today in the technology space; however Amazon Web Services continues to rule the roost regardless of the highly publicized cloud rivalry among AWS, Microsoft, Oracle and Google. AWS made number one on the list of Top 10 Cloud Computing Providers at SearchCloudComputing.com for 2010, 2011 and 2012 consecutively. AWS has not only been providing cloud services to small, medium and large businesses but it has also extended its services to another planet. NASA’s Mars Mission was fueled by Amazon Web Services to manage the flow of information and imagery that transmissions to and from the Earth carried. Moreover, AWS recently won a $600 Million cloud deal when CIA chose Amazon Web Services over IBM to develop a version of its public cloud that runs inside the CIA’s data center.

some of the reasons one might choose AWS over an alternate provider:

**1.Pricing Model**

The pay-as-you-use model expands to all the AWS products.  During times of reduced usage you can benefit from reduced cost, which is not achievable when utilising onsite or alternative hardware.  Amazon provision large volumes of users and systems to support these users and thus is able to provide the services at competitive pricing.  With the cost+pricing model and fixed margin, reduced pricing is also achievable.

**2.Free Tier Model**

For users who are less knowledgeable of AWS and are doubtful of jumping in without the knowledge of the workings of AWS, Amazon provides a free tier model to get users started and learning without involving cost.

**3.Performance**

AWS offers speed and reliability that surpasses most global private datacentres.  The performance is noted across the AWS services but that of the storage service is especially great.

**4.Flexibility**

The superb flexibility is probably one of the most appreciative features of AWS. The compatibility of all the services to communicate and work together to achieve the best computing performance is evident. The services are unassuming, robust and scalable.

**5.Speed of Deployment**

Amazon provides rapid deployment (provisioning of servers), which could traditionally take days to achieve, is achievable in minutes with AWS.

**6.Knowledge of Consumer requirement**

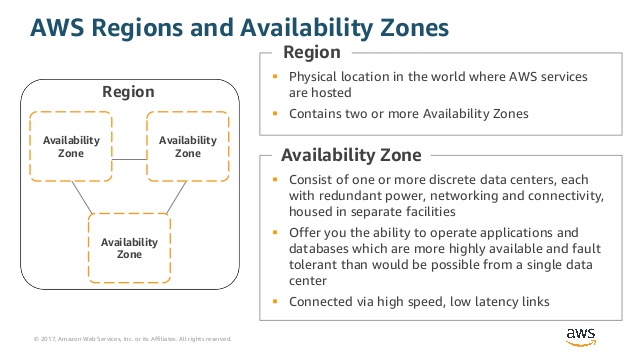
Amazon has the expertise and insight into consumer service requirements. Amazon has constructed AWS in a way that it’s relatively easy to use and inexpensive.  Amazon has the expertise to realise where emphasis should be placed on services, such a services for developers, and by staying up-to-date they can easily maintain a lead in the area of cloud computing.

**7.Innovation and speed to evolve**

Amazon has an engineering culture with long-term roadmap, research and Amazon is future thinking, which allows for evolution and the release of features at a speed that is difficult to equal.

**Region and Availability Zones in AWS**

The AWS Global infrastructure is built around Regions and Availability Zones (AZs). AWS Regions provide multiple, physically separated and isolated Availability Zones which are connected with low latency, high throughput, and highly redundant networking. These Availability Zones offer AWS customers an easier and more effective way to design and operate applications and databases, making them more highly available, fault tolerant, and scalable than traditional single datacenter infrastructures or multi-datacenter infrastructures. For customers who specifically need to replicate their data or applications over greater geographic distances, there are AWS Local Regions. An AWS Local Region is a single datacenter designed to complement an existing AWS Region. Like all AWS Regions, AWS Local Regions are completely isolated from other AWS Regions. The AWS Cloud spans 60 Availability Zones within 20 geographic regions around the world.



**Configuring EC2 instances**

Each virtual machine, called an “instance”, functions as a virtual private server. Amazon sizes instances based on “Elastic Compute Units”.

As of February 2018, the following instance types were offered:[[](https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud" \l "cite_note-18)

* General Purpose: M5, M4, T2
* Compute Optimized: C5, C4
* Memory Optimized: X1e, X1, R4
* Accelerated Computing: P3, P2, G3, F1
* Storage Optimized: H1, I3, D2

As of April 2018, the following paying method for instance were offered:[[](https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud" \l "cite_note-19)

* On-demand: pay by the hour without commitment.
* Reserved: rent instances with one-time payment receiving discounts on the hourly charge.
* Spot: bid-based service: runs the jobs only if the spot price is below the bid specified by bidder. The spot price is claimed to be supply-demand based, however a 2011 study concluded that the price was generally not set to clear the market, but was dominated by an undisclosed reserve price.

**AWS Security Groups**

AWS security groups (SGs) are associated with EC2 instances and provide security at the protocol and port access level. Each security group –  working much the same way as a firewall – contains a set of rules that filter traffic coming into and out of an EC2 instance. There are no ‘Deny’ rules. Rather, if there is no rule that explicitly permits a particular data packet, it will be dropped.

**AWS Security Groups: rules**

The actual rule set that filters traffic is made up of two tables: ‘Inbound’ and ‘Outbound’. AWS Security groups are stateful, meaning you do not need the same rules for both outbound traffic and inbound. Therefore any rule that allows traffic **into** an EC2 instance, will allow responses to pass back **out** without an explicit rule in the Outbound rule set.

Each rule is comprised of four fields: ‘Type’, ‘Protocol’, ‘Port Range’, and ‘Source’. This applies for both ‘Inbound’ and ‘Outbound’ rules.

* **Type.** The drop down list allows you to select common protocols like SSH, RDP, or HTTP. You can also choose custom protocols.
* **Protocol.** This is typically greyed out, as it’s covered by most ‘Type’ choices. However, if you create a custom rule, you can specify your protocol (TCP/UDP etc.) here.
* **Port Range.** This value will also usually be pre-filled, reflecting the default port or port range for your chosen protocol. However, there might be times when you prefer to use custom ports.
* **Source.** This can be a Network Subnet range, a specific IP address, or another AWS security group. You can also leave access open to the entire Internet using the ‘Anywhere (0.0.0.0/0)’ value.

**Amazon EC2 Elastic IP’s**

Every AWS user would have an AWS account and can launch EC2 instances under this account for various computing purposes. These instances can be accessed using public IP or elastic IP.

When a new EC2 instance is started, it will be automatically assigned a public IP to access that instance. This public IP address can be used to access the EC2 instance from outside.

Elastic IPs (EIPs) are static IPs allocated to a specific AWS account. They are public IPv4 addresses, that can be manually assigned to EC2 instances to access them from outside.

The public IP is a dynamic one and changes with every reboot. As a result, services in that instance that use DNS, can face downtime during this change in IP.

Whereas, the elastic IPs are static in nature and does not change with an instance restart. Their static nature makes EIPs a more preferable option for services in EC2 instances.

**Benefits of using EC2 Elastic IPs**

Many internet services require DNS. While using public IPs, it is very difficult to switch the DNS each time an IP change occurs, and it involves a downtime owing to the DNS propagation.

Here are some benefits of using elastic IPs, which prompts us to prefer them over public IPs for our customers’ EC2 instances.

1. **Static and public** – Being static in nature, we use the Elastic IPs for public services that require static IPs – such as web server, dns or email server.
2. **Flexibility** – As Elastic IPs are allotted to each AWS account and are not specific to an EC2 instance, we allocate an Elastic IP to any EC2 instance as and when a requirement arise.
3. **Load balancing** – When one EC2 instance fails or gets overloaded with traffic, we allocate its associated EIP to another EC2 instance, by configuring a failover setup in the system. This helps us to ensure uninterrupted service for the users.
4. **Fail-over setup** – Having multiple Elastic IPs for an AWS account enables us to quickly map a service from one IP to another, in case of issues such as IP blacklisting or IP changes.
5. **Uptime** – With our load balancing and fail-over setup with the help of elastic IPs, we are able to prevent downtime in services that can happen due to maintenance or DNS propagation.