# Terminology

* VPC – your own private data center in the cloud

# AWS

In Amazon: create a burner account for training: access.amazon.com

https://access.amazon.com/aws/burner/get\_console\_access\_url?account\_id=043163097029

# Identity Access Management (IAM)

## Terms

* Users – End users (people)
* Groups – a collection of users under one set of permissions. For example: HR, Marketing etc. each of them requiring different types of permissions.
* Roles – you create roles (e.g. S3-ReadOnlyAccess) and then you can assign them to users or other AWS resources (e.g. EC2 ).
  + Roles are controlled by policies
  + They are much more secure than using (secret) access keys.
  + You can change a policy on a role and it will take immediate effect
  + You can attach and detach roles to running EC2 instances without having to stop and restart the instances.
* Policies – A document that defines one (or more) permissions. The policy can then be attached to a user/group/role. Readable json format.

## What is IAM

Allow you to manage users and their level of access to the AWS console.

IAM gives you:

* Centralized control of your AWS account
* Shared access to your AWS account
* Granular Permissions
* Identity Federation (allowing users to login to your site using their Active Directory, Facebook or LinkedIn etc accounts)
* Multifactor Authentication
* Provides temporary access for users/devices and services, as necessary.  
  For example, allowing users of your app to retrieve data from your S2 database.
* Allows you to set up you own password rotations policy
* Integrated with many AWS services
* Supports PCI DSS Compliance for applications associated with the payments industry.
* IAM is universal. It does not apply to specific regions.

## IAM Lab

1. Create a new AWS account: <https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/>
2. Create user accounts. Use your root account only when you absolutely don’t have another choice. For everything else – use a user account.  
   NOTE: When you create the account, this is the only time that you can see the user’s Secret Access Key which is required to run CLI commands. So, Download the .csv and keep it safe. Otherwise, you won’t be able to access this account anymore!! Or even better, **save it in keypass!!**
   1. User & paswd – required for logging into the Console
   2. Access Key ID & Secret Access Key – required for using the CLI
   3. New users don’t have any permissions. You need to give them permissions.
   4. Notes:
      1. If you lost your access (secret) key, you can generate a new one and then you’ll need to re-run >> aws configure to use the cli again
      2. Never save the (secret) access keys in code!!
      3. Never share the same access keys between multiple developers.
      4. Do NOT USE access (secret) keys to control the access to your services! It is too insecure and risky. Instead, **use roles** to control them (see below)
3. Create Roles: for example: if we want to allow EC2 instances to write to our database (S3 buckets).
   1. After creating the role, you can go to the EC2 instance and choose Action-> Instance setting -> attach role to connect the role to the target.

# Elastic Compute Cloud - EC2

* Secure, resizable compute capacity in the cloud.
* Like a virtual machine in the cloud (hosted in AWS)
* Allow super-easy scaling
* The capacity you want when you need it
* You have complete access to your instances (root)
* Pay only for what you use, when you use it.
* You can select your capacity and then grow/shrink it as you need.
* Pricing models:
  + On demand – you pay by the hour/second depending on the type of instance you run
    - Good for application that are short-term, spiky or unpredictable workloads that cannot be interrupted.
    - For testing the waters – new application in development
  + Reserved Instance (RI) – you reserve the capacity for yourself for 1-3 years. This is regional and allow you to get up to 72% discount (as long as you’re in the same region)
    - Suitable when you know in advance that you need capacity X long term (years)
    - Require you to pay up front
    - Standard RI – a static capacity (you can’t change it) – up to 72% off
    - Convertible RIs – you can change the capacity for equal/greater value – up to 54%
    - Scheduled RIs – when you don’t need the capacity all the time but instead only X days every month so you can pay only for these re-occuring times.
  + Spot – Purchase unused capacity at discounts of up to 90% (fluctuate according to supply and demand). Once the capacity drop below your ask or the price above it, you will use your instance.
    - Suitable when you need to run something occasionally and you can spin it up from scratch easily (no long installations etc needed) and when it’s not time-critical.
  + Dedicated – a physical EC2 server dedicated for your use. This is the most expensive option.
    - This is suitable for when you have licenses that are tied to a physical HW or you have some regulations that forces you to use this option.

## EC2 Instance Types

* The HW of the underlying server
* Capabilities: different compute, memory, storage etc
* Different instance types are optimised for different types of applications

## EC2 Lab

* Compute->EC2->scroll to Launch instance-> Select Amazon Linux 2 AMI
* You can filter the instances types
* The ‘Type’ is the instance names. The number after the letter (e.g. t2.nano) is the generation. In this case, the 2nd generation of the nano isntances.
* Greyed out options mean that they are not supported for our choosed OS
* Main types:
  + Micro instances are a low-cost option for very small/low throughput applications.
  + Compute optimized – high-processing applications that don’t need that much storage
  + FPGA – for processing on HW. For highly paralalised processing on HW
  + GPU – for graphics and apps that needs parallel processing
  + Machine learning – use custom-built ASIC for machine learning
  + Memory optimised – a lot of memory. Database applications, distributed caches etc.
  + Storage optimized
* Configure Instance Details – important notes:
  + You can select Spot instance and set the price limit
  + Austo-assign public IP – the public IP to use in order to access your app
  + Shutdown behaviour – can choose either Stop or Terminate
  + Enable termination protection – very important to production systems so that they can be accidently terminated.
  + Monitoring – CloudWatch monitor your services at 5 minutes intervals bey default for free. If you want a more detailed monitoring in higher resolution (upto 1 min), you need to set it up and pay extra.
  + Advanced Details – User data – this is where you can add bootstrap batch files to run whenever the system boots up. We can use this for installing updates/applications etc.
  + Add storage – by default you get one EBS volume on which the linux will be installed.
    - Volume type – for most applications you should choose the default (General Purpose SSD (gp2)).
    - Encryption – you can encrypt your volume with KMS when you create it.
    - You can add additional Volumes. For example, if you need to have your application on a different volume that the OS.
  + Tags – completely user-defined. It’s a great way to organise your EC2s and other resources. You can add ‘name’ , ‘team’ etc.
  + Security Groups – a virtual firewall
    - SSH – allow to access it with SSH
    - HTTP – for web apps. Port should be 80
    - Source:
      * For HTTP – should be anywhere so that everyone can access my site. For SSH – should be ‘My IP’ or ‘Custom IP’
      * 0.0.0.0/0 – means all IPV4 addresses will have access ; ::/0 – means all IPV6 addresses will have access.
  + Select an existing key pair or create a new key:  
    Create a new key pair and download it. This will create a private-public key pair that will be used for accessing your EC2 instance. Only users that have the private key will be able to access the EC2 instance.
    - After the ke y-pair downloads, change its permissions:  
      >> chmod 400 my-key-pair.pem  
      otherwise, it won’t work
  + Launch your EC2 instance
* To ssh into your EC2 instance:  
  ssh ec2-user@{Public DNS} -i {private-key file}  
  for example:  
  ssh ec2-user@ec2-54-160-117-176.compute-1.amazonaws.com -i MyTrainingKeyPair.pem
  + Note: in order to ssh from a windows machine, use’ll need to use Putty and PuttyKeyGen to convert your .pem key to .ppk file.
* Move to su account:  
  >> sudo su
* Install all latest OS updates:  
  >> yum update -y
* Install apache:  
  >> yum install httpd -y
* Start apache:  
  >> systemctl start httpd
* Start apache automatically on boot:  
  >> systemctl enable httpd
* Verify that it’s running:   
  >> systemctl status httpd
* Create a minimal website:

1. Browse to /var/www/html
2. Create a minimal index.html file with nano editor:  
   >> nano index.html
3. Now you should be able to access your website using the ‘IPV4 Public IP’ on your instance’s Description page (On your EC2 Dashboard->Volumes->choose your volume)

## Elastic Block Store - EBS

Storage volumes that you can attach to your EC2 instances (like a hard disk). You will use them as you would any physical HD.

* Mission Critical. EBS are:
  + Designed for production workloads
  + High Availability: redundancy: automatically replicated within a single availability zone to protect against HW failures.
  + Scalable: dynamically increate capability and change the type volume with no downtime or performance impact to your live system.
* Types of EBS:
  + **IOPS** – IO operations per second. Important for quick database transactions, low latency apps.
  + General Purpose SSD (gp2) – a balance of price and performance.
    - good for boot volumes or development and test applications which are not latency sensitive.
    - Max 16,000 IOPS per volume.
  + Provisioned IOPS SSD (io1)
    - The highest performance and the most expensive one
    - If you need more then 16,000 IOPS
    - Up to 64,000 IOPS per volume
    - IO intensive applications, large databases
    - Suitable for OLTP (Online Transaction Processing) and Latency-sensitive applications.
  + Throughput Optimized HDD (st1):
    - Low cost HDD volume
    - Max **throughput** 500 MB/s per volume. Important for large datasets, large IO sizes, complex queries. Effec the ability to deal with large datasets.
    - When you need to store huge amounts of data and access it frequently, throughput-intensive workloads
    - Big Data, data warehouses, ETL, log processing
    - Cannot be a boot volume
  + Cold HDD (sc1)
    - Lowest cost option
      * Max throughput 250 MB/s per volume
      * A good choice for colder data requiring fewer scans per day
      * Where performance is not a factor
      * Cannot be a boot volume.
* To create a new volume, we can create it from the EC2 Dashboard:
  + Notice the availability zone of your EC2 instance since it has to be the same for your volume
  + Snapshot ID – allows you to use existing snapshot (image). If you choose a snapshot, it will also copy the same encryption.
  + To attach, choose the volume and in actions-> Choose attach

## EC2 CLI

<https://docs.aws.amazon.com/cli/latest/reference/s3/index.html>

In SSH session:

>> aws configure

And copy the Access Key ID and the Sercret access key from the user’s details.

* Clean your configuration on the EC2:  
  >> cd ~/.aws  
  >> rm config  
  >> rm credentials
* Create an s3 bucket:  
  >> aws s3 mb s3://{my bucket name} [optional: --region {region name}]  
  for example: >> aws s3 mb s3://my-training-bucket1234-rk –region us-west-1
* See your s3 buckets:  
  >> aws s3 ls
* See all the files in your bucket:  
   >> aws s3 ls {bucket name}
* Move files into your s3 bucket:  
  >> aws s3 cp {file name} {bucket name}  
  for example:  
  >> aws s3 cp hello.txt s3://my-training-bucket1234-rk

# Elastic Load Balancers

* Allow us to balance the load across different servers.
* Application Load Balancer – works in the OSI Level 7 (application level) – http/https level. Can make very clever decisions according to the application layer data. So for example, they can route the traffic to different servers based on the sub-domains of the calls (e.g. all site/marketing/… URLs will be routed to the marketing servers etc).
* Network Load Balancer – works in OSI Level 4 (transport – TCP/UDP) – optimised to be super-fast (extreme performance). This is what you would usually use in production systems and when latency is critical.
* Classic Load Balancer – they are legacy only and not recommended for new applications. You can load balance with it both layer 4 or layer 7 (x-forwarded and sticky sessions). It’s not as good as the new load balancers.

## Classic Load Balancer

* Classic Load Balancer Error – if your application stops responding, the ELB responds with a 504 error (gateway timeout):
  + This is not the ELB failing but a failing of the application
  + To troubleshoot it -> troubleshoot the application
  + This could be either at the Web Server Layer or at the Database Layer.
  + Identify where the application is failing and scale it up or out where possible
* X-Forwarded-For Header:  
  

Your application will not see the user’s IP. Instead, it will only see the ELB private IP (e.g. 10.0.0.23). In order to see the end user’s IPV4 look for the X-Forwarded-For header.

## Route 53

Route53 is Amazon’s DNS Service

It allows us to map our domain names to:

* EC2 instances
* Load Balancers
* S3 Buckets

## Lab

1. Services->route 53->DNS Management
2. If you don’t have a domain, you can register (buy) a domain there. Once your domain is registered, you can route it.
3. Create a load balancer:
   1. Go to your EC2 dashboard -> Load Balancers -> Create Load Balancer
   2. Choose application load balancer
   3. Listeners – on which ports will the load balancer listen. Can choose http, https etc.
   4. Availability Zones – choose all for maximum availability
   5. Configure Security Groups – you can choose the security group we created before.
   6. Configure Routing – create/choose your target group – what will be the target to which the routing rule will apply (when you configure the route 53 mapping)
      1. Target type – what will be the type of the target: Instance (EC2), IP or Lambda Function
      2. Which protocol and port to use for the target
      3. Health checks – only route if these checks are ok (?)
   7. Register Targets: choose your target (e.g. EC2 instance)
   8. Wait until you see that the:
      1. Load Balancer State is healthy
      2. In your EC2 -> Status Checks, the status is healthy (what we defined in the target group).
4. Go to your Route 53 dashboard -> hosted zones -> mark your domain -> Go to records set. This is where we create our DNS records
   1. A record: map a DNS
   2. Create Record Set and choose:
      1. Name: leave is as naked domain name (also called Apex Record which means that it doesn’t have the www. Prefix)
      2. Type: A-IPV4 address
      3. Alias: Yes. Choose your EC2 instance
   3. Now, when you browse to your new domain name (e.g. iamacloudguru.com), you’ll be automatically routed to your EC2 instance.

# Databases

## Terms

### Data Warehousing

Used for business intelligence.

Tools like Cognos, Jaspersoft, SQL Server Reporting Services, Oracle Hyperion, and SAP NetWeaver.

Used to pull in very large and complex data sets. Usually used by management to do complex queries on data (such as current performance vs targets etc).

### OLTP vs OLAP

* Online Transaction Processing (OLTP) – simple transactions that happens very frequently. For example: get/write an order data into a database.
* Online Analytics Processing (OLAP) – complex queries that happen very infrequently. For example, the net profit for some sort of product. This will need pulling a lot of different data and doing calculations on it.

### ElastiCache

A web service that makes it easy to deploy, operate and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.

ElastiCache supports 2 open-source in-memory caching engines:

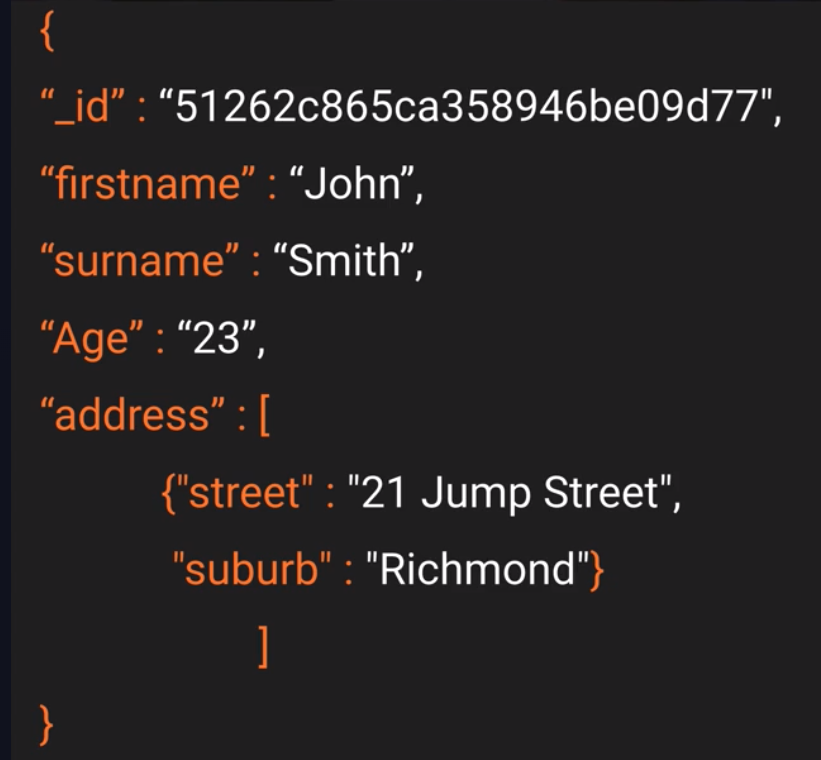
* Memcached
* Redis

## Relational Database Service (RDS) - OLTP

* Think spreadsheets:
  + Database with
  + Tables each table has
  + Rows and
  + Fields (columns)
* RDS Types in Amazon:
  + SQL Server
  + Oracle
  + MySQL Server
  + PostgeSQL
  + Aurora – Amazon flagship SQL database
  + MariaDB

## Non Relational Database

Contains:

* Database with
  + Collections (=tables)
  + Documents (=rows) with
  + Key Value Pairs (=fields)
* Advantages over relational databases:
  + You do not need to define your fields (columns) in advance like in the RDs
* Examples:
  + JSON/NoSQL:  
    
* In Amazon: DynamoDB – No SQL

## Redshift - OLAP

# For the exam

* See the Exam Blueprint pdf
* I don’t need to memorise the EC2 instance types for the exam

# Lab