AWS:

- Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments.
- The technology allows subscribers to have at their virtual cluster

- of computers, available all the time, through the Internet.

- Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.
- AWS provides instances in the following regions:

    - US East (N. Virginia, Ohio)
    - US West (N. California, Oregon)
    - Asia Pacific (Mumbai, Seoul, Singapore, Sydney, Tokyo, Osaka-Local)
    - Canada (Central)
    - China (Beijing, Ningxia)
    - Europe (Frankfurt, Ireland, London, Paris)
    - South America (Sao Paulo)

- AWS is the number 1 cloud computing company worldwide.

EC2

- Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud.

- It is designed to make web-scale cloud computing easier for developers.

- Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction.

- It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment.

- Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.
- Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use.

- Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

- EC2 offers many types operating systems including:

- ALI (Amazon Linux distribution), RHEL (Red hat enterprise linux), SUSE, Ubuntu and Windows.

- EC2 offers many machines types with different CPU, Storage and networking capabilities

- Pricing can go as high as $24.48 per Hour

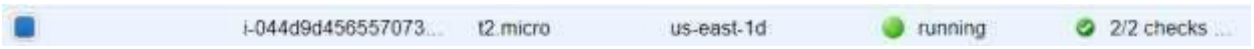
- For our usage we will use instances which are offered for 1 free year.
  https://aws.amazon.com/ec2/

Launching an AWS EC2 instance

- Go to EC2 page (Services → EC2) → Instances (on the left menu panel).
- Press Launch instance button→ Choose Amazon Linux 2 AMI, which is free for first year → keep t2.micro which is free and launch instance by pressing Review and launch button and launch
- In order to have the ability to SSH into our server, we will need to
- create a key, just choose: Create a new key pair → give the key a name → press Download Key Pair

- Make sure to keep this key in a secure location, without this key you will not be able to connect to your server!
- Press launch instances scroll down and press view instances.

- Wait a few minutes until you see this:



https://aws.amazon.com/ec2/getting-started/

Connecting to an AWS EC2 instance

- To connect to our EC2 instance, we will need to use the key (.pem) file, we created earlier.
- Find the EC2 instance public IP address or public DNS in the EC2 console:

- Open CMD / Terminal and navigate to the folder containing the
- .pem file (using CD)
- Type the following command:

```
$ ssh -i <pem file name> <public IP / DNS>
```

- For example:

```
$ ssh -i ec2.pem ec2-user@18.205.46.80
```

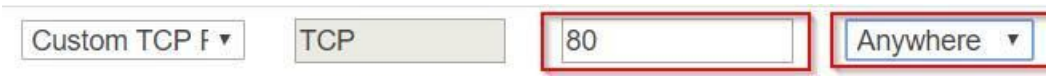
- You will then be asked:

```
Are you sure you want to continue connecting (yes/no)?
```

- Obviously answer - yes

Cloud deployment

- Once we are logged in, let's install Docker:
- $ sudo yum install docker
- Start Docker service by typing:
  $ sudo systemctl start docker
- Let's run Nginx image on port 80 –
- $ docker run --name docker-nginx -p 80:80 -d nginx
- Open a browser on the instance public address... it will not work...
- The reason for that, is because we didn't exposed the instance port 80 to the world...
- To fix the issue, go to Security groups → Choose one of the launch-wizards
- Scroll to inbound and press Edit → press add rule → Choose port 80 and source anywhere → Save



- Go back to your instance → press Actions drop down → Networking → Change security groups → Choose the wizard you used earlier → Assign security groups
- Refresh the browser on your public address...

WE ARE ONLINE

Elastic IP (EIP)

- Go to EC2 page (Services → EC2) → Instances (on the left menu panel).
- When an instance is launched into EC2, Amazon will randomly assign it a public IP address.
- Amazon has a pool of public IP addresses that's been reserved for use within EC2.
- An Elastic IP (EIP) is an IP address that you can reserve from AWS for your account.
- Once you've created an Elastic IP, you can assign it to any instance of your choice.
- Once you reserve an Elastic IP, nobody else can use that IP address.
- There is no cost for Elastic IP addresses while in use.
- You only pay for the Elastic IP when it's not attached to an instance.
- To associate an EIP, we will first need to allocate an IP address by
- going to Elastic Ips under Network & Security sub-menu → Allocate new address → Allocate.
- Go to Actions → Associate address → choose your instance and private IP and Associate
- Once, you finished with instance, you should disassociate the IP through Actions menu

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html

IAM (Identity and Access Management)

- When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account.
- This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account.
- IAM works with groups and users.
- A group is a collection of users who have similar responsibilities.
- For example, one group is for administrators (it's called Admins), another one will be for Developers group (called dev).
- Each group has multiple users.
- Each user can be in more than one group.
- Policies are used to grant permissions to groups.

https://aws.amazon.com/iam/

IAM roles

- An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS.

- However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

- Also, a role does not have standard long-term credentials (password or access keys) associated with it.

- Instead, if a user assumes a role, temporary security credentials are created dynamically and provided to the user.

- We can think of IAM Roles as capabilities. You give an IAM User capabilities (e.g. "can create Lambda function", "can upload to S3").

IAM setup

- To create a security group, go to Services → IAM

- Enter groups (left panel) → Create New Group→ give it a name. It this example we will grant permissions to EC2, to do so, find

- AmazonEC2FullAccess → press next step → Create group Next we will create a user to add to our new group.

- Enter Users (left panel) → Add user → Give it a name → choose AWS Management Console access which will allow users to

- sign-in to the AWS Management Console.
- Choose a password / auto generated password → Choose the

- group we created earlier → Review → Create user.

- When logging in with the new user you will be using the new credentials and the IAM alias number, which is showing in the IAM page.

AWS CLI

- The AWS Command Line Interface (CLI) is a unified tool to manage your AWS services.
- With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
- The AWS Command Line Interface User Guide walks you through installing and configuring the tool.
- After that, you can begin making calls to your AWS services from the command line.

```
$ aws ec2 describe-instances


$ aws ec2 start-instances --instance-ids i-1348636c


$ aws sns publish --topic-arn arn:aws:sns:us-east-1:546419318123:OperationsError --message "Script
 Failure"


$ aws sqs receive-message --queue-url https://queue.amazonaws.com/546419318123/Test
```

AWS CLI Installation

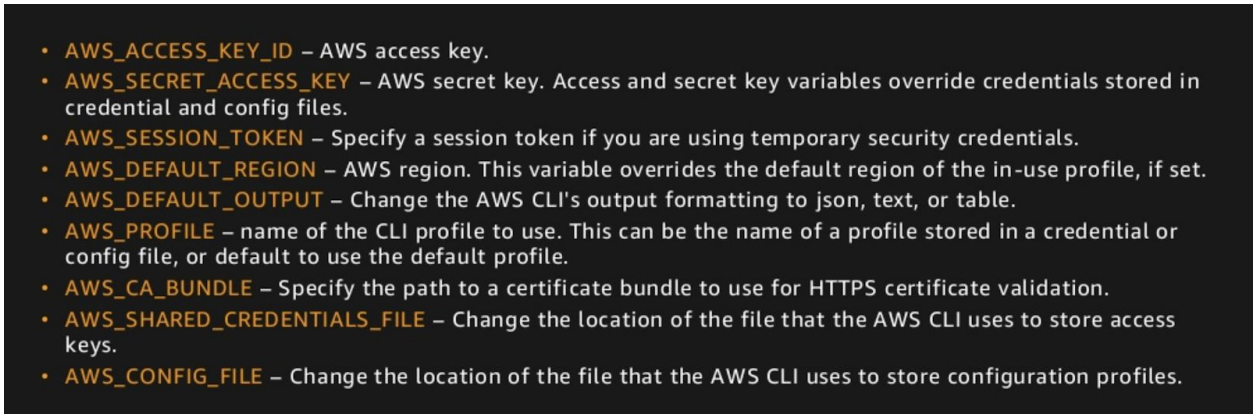
1. Create access key and secret key
    a. Login to AWS -> Go to IAM service -> Under Users find your user
    b. Go to Security Credentials -> Click Generate Access key and save the output.
2. Install AWS CLI
    a. pip install awscli
    b. Aws configure
    c. Enter the access keys credentials we created.

AWS CLI Files

- ~/.aws/credentials - Contains our secret and access key for different profiles
- ~/.aws/config - General AWS CLI configurations such as region and preferred output (json or text lines).

AWS Command Structure:

- At any time we can type aws <service name> <operation> help and discover more about how to use the command properly.
- If we want to override the credentials stored in ~/.aws/credentials we can always send AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY as environment variables.
- If we want to specify a profile name in the awscli command we can add to the command --profile <profile name>
- If we want to set the output for a specific command we can use --output <output type: text/json/table>



- --filter
  - Done on the server side
  - Only available for certain operations
  - Use this when expecting a large answer
  - Also support --page-size for paging.
- --query
  - Done on the client side
  - Available on all commands
  - Used to show only specific columns

Filtering Example:

```
aws ec2 describe-instances --filter Name=instance-type,Values=t2.micro
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "ec2-34-228-84-177.compute-1.amazonaws.com",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "EbsOptimized": false,
```

Querying Example:

```
$ aws iam list-users --query 'Users[*].[UserName, CreateDate, PasswordLastUsed, Arn]'
[
    [
        "leozh",
        "2013-08-04T17:24:01Z",
        "2017-11-07T00:15:03Z",
        "arn:aws:iam::179442201234:user/leozh"
    ],
    [
        "Billing",
        "2015-03-17T03:31:45Z",
        "2015-03-17T18:47:44Z",
        "arn:aws:iam::179442201234:user/billing"
    ]
```

- AWS Aliases - Contains common AWS CLI aliases and shortcuts for easier use.
- https://github.com/awslabs/awscli-aliases

```
$ aws whoami
{
    "Account": "179442201234",
    "UserId": "AIDAIXV2V6G4AEXAMPLE",
    "Arn": "arn:aws:iam::179442201234:user/leozh"
}

$ aws amazon-linux-amis
ami-6057e21a amzn-ami-hvm-2017.09.1.20171103-x86_64-gp2 Amazon Linux AMI 2017.09.1.20171103 x86_64 HVM
ami-8c1be5f6 amzn-ami-hvm-2017.09.0.20170930-x86_64-gp2 Amazon Linux AMI 2017.09.0.20170930 x86_64 HVM
ami-5e8c9625 amzn-ami-hvm-2017.09.rc-0.20170913-x86_64-gp2 Amazon Linux AMI 2017.09.rc-0.20170913 x86_
GP2
ami-4fffc834 amzn-ami-hvm-2017.03.1.20170812-x86_64-gp2 Amazon Linux AMI 2017.03.1.20170812 x86_64 HVM
```