Show Submission Credentials

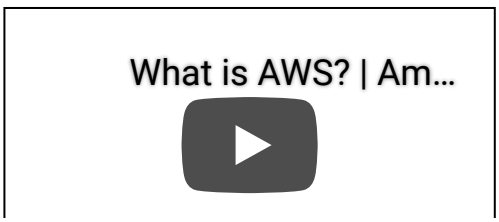
P0. Amazon Web Services Intro Amazon Web Services

- ✔ Amazon Web Services
- ✔ Elastic Compute Cloud (EC2)
- ✔ Simple Storage Service (S3)
- ✔ AWS Command-Line Tool
- ✔ CloudWatch

Amazon Web Services

Amazon Web Services

Amazon Web Services (AWS) is one of the most popular cloud service providers and is widely credited with the introduction of innovative concepts in cloud computing. AWS offers various services such as Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2). For the projects in this course, you will work on instances provisioned on Amazon EC2 and utilize S3 for persistent storage. We will also be working with other AWS technologies such as Elastic Block Store (EBS) and CloudWatch, among others. For more information about AWS, please refer to the AWS website (http://aws.amazon.com/).



Video 1: Introduction to AWS

AWS Authentication

Apart from your AWS account username/password, programmatic API access to AWS services can be granted using a **AWS Access Key ID** and **AWS Secret Key** pair. These are unique credentials that can be generated by the user and used within scripts and programs that interact with AWS services.

The process for creating IAM users is outlined in the AWS IAM documentation (http://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html). To ensure you are following best security practices, create an IAM user which has access to the AWS console and required APIs:

1. Sign in to the AWS Console (https://console.aws.amazon.com)
2. Select the IAM (https://console.aws.amazon.com/iam/home) service under ***Security, Identity & Compliance***

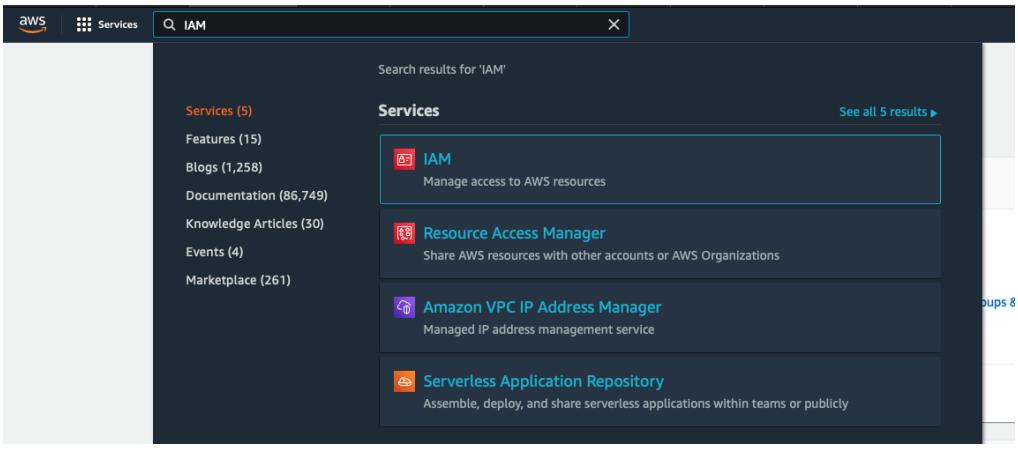


Figure 1: AWS Console, IAM access

3. Select ***Users*** from the IAM console and click ***Add User***.
4. In the first step "Set user details", provide a valid username and select ***Programmatic access***.

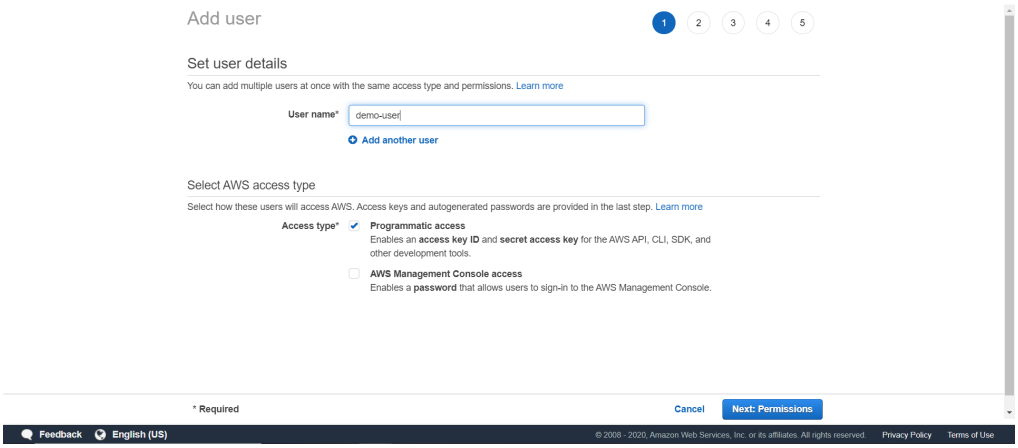


Figure 2: New IAM User details

5. In step 2, select ***Attach existing policies directly***. For example, `AmazonS3ReadOnlyAccess` provides read-only access to all buckets, `AmazonEC2FullAccess` provides full access to Amazon EC2. When you add IAM policies, the standard security advice in industry is to grant least privilege, or to grant only the permissions required to perform a task. That said, fine-tuning role-based access control is an advanced skill which requires experience to troubleshoot subtle permission errors. To avoid unnecessary permission issues, please attach `AdministratorAccess` for this course.

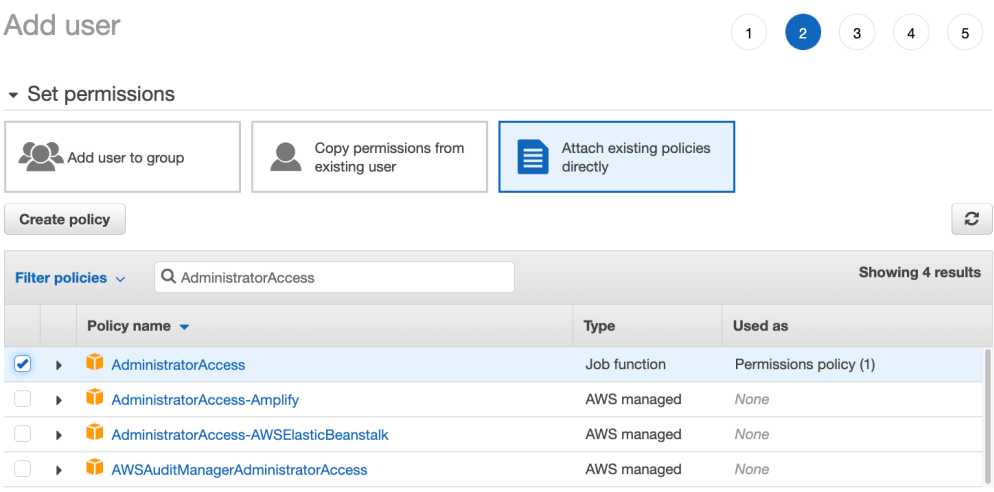


Figure 3: Attach existing policies for services the User needs access to

6. In step 3, add a Tag with the key **`project`** and the value **`getting-started-with-cloud-computing`** , Then move to step 4 and review the User's policies and click ***Create user***

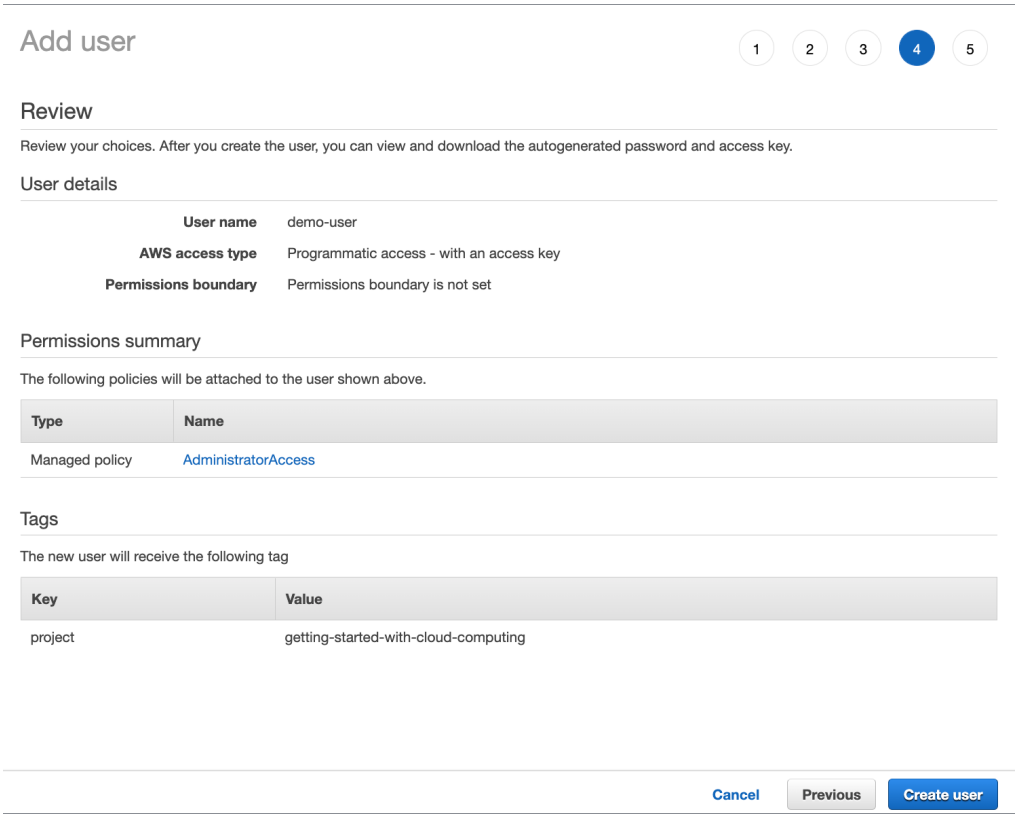


Figure 4: Review the User that is about to be created

7. In step 4, take note of the Users ***Access key ID*** and ***Secret access key*** as this is the only time they are accessible.

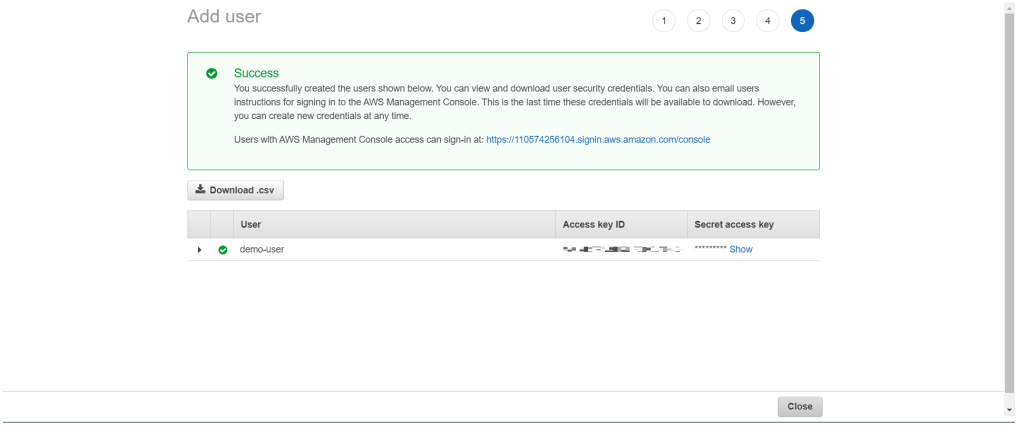


Figure 5: Download the credentials or use them to configure the AWS CLI

Danger

Keeping your Keys Secure

Your AWS keys have control over your AWS account, including the ability to launch, destroy and modify all types of AWS resources. The keys **must** be kept safe, preferably encrypted by password on your PC.

Here are a few important guidelines:

1. **Do not** share the AWS keys with anyone else.
2. **Do not** leave your keys in plain text in your source code.
3. **Do not** commit code to any repositories with your AWS keys in the source code.
4. IAM Users and credentials should have the most limited level of API access as possible.

The safest method to reference the keys in your source code is to define them as environment variables:

```
export AWS_ACCESS_KEY_ID="YOUR_ACCESS_KEY_HERE"
export AWS_SECRET_KEY="YOUR_SECRET_KEY_HERE"
```

You can then access them from within your code, for example, using `System.getenv("AWS_ACCESS_KEY_ID")` in Java, `os.environ['AWS_ACCESS_KEY_ID']` in Python, or `${AWS_ACCESS_KEY_ID}` in Bash.

AWS Management Console

The following video will provide a short overview of AWS:

Video 2: AWS Management Console

To make sure that we do not suffer from excess expenditure, we have suggested budgets for each project module. For this project module, the budget is outlined below:

Danger

Grading Penalties

The following table outlines the violations of the project rules and their corresponding grade penalties for the first week of the course. The cost limits include both Project `getting-started-with-cloud-computing` and the Primers.

These rules apply for the week when Project `getting-started-with-cloud-computing` is active.

Violation	Penalty of the project grade
Using more than $5 of AWS resources (on Project `getting-started-with-cloud-computing` and Primer combined)	Warning email
Using more than $10	Possible penalty on future projects
Not tagging any of your resources	Warning email
Using any project tags apart from `project` : `getting-started-with-cloud-computing`	Warning email

Elastic Compute Cloud (EC2)

Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a service in AWS that provides resizable computing capacity in the AWS cloud. Amazon EC2 makes virtual servers (also known as instances) available to you as a service, thereby eliminating your need to invest in hardware up front and enabling you to develop and deploy applications faster. You can use the web service interfaces or API calls to launch as many or as few instances as you need, configure security and networking, and manage storage.

As you get started with Amazon EC2, you should understand the key concepts of the environment. The following video outlines the basic concepts and ideas behind EC2:

Video 3: EC2 Basics

Information

EC2 Terminology

Some terms and concepts that you should know:

EC2 Instance: EC2 instances are virtual servers that you can configure and launch on Amazon EC2. It can be thought of as a copy of a software image, Amazon Machine Image (AMI), that is actively running on the Amazon EC2 cloud. Instances run on host computers in Amazon's data center, but this is typically transparent to the user. Instances have many parameters including:

1. **Amazon Machine Image (AMI)**: The template that contains a complete software image (operating system, applications, libraries and data) of the system. You launch instances using specific AMIs, which are copies of the AMI running as virtual servers in the cloud. You can launch multiple instances using the same AMI.

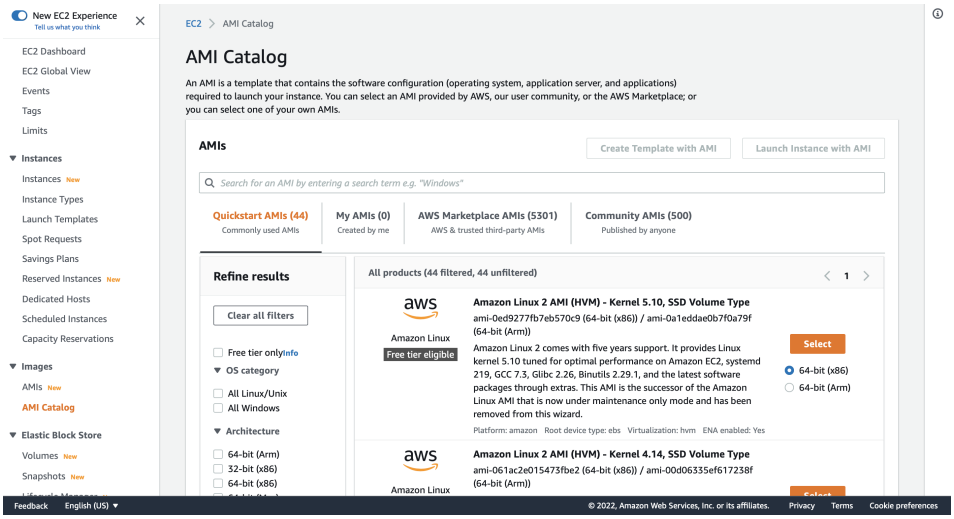


Figure 6: Amazon Machine Image (AMI)

2. **Instance Type**: Amazon offers different instance types (https://aws.amazon.com/ec2/instance-types/) which have varying amounts of compute and storage available to them. They range from t2.nano to x1.32xlarge and have different on-demand/reserved and spot prices.

3. **Regions and Availability Zones**: Instances can be launched across various AWS regions, which are distributed geographically (Virginia/Singapore etc.). Each region has a number of availability zones, which are distinct locations within a data center and are engineered to be isolated from other availability zones in the same region. This allows users to spawn instances in the same region but across availability zones and protect applications from the potential failure of a single availability zone.

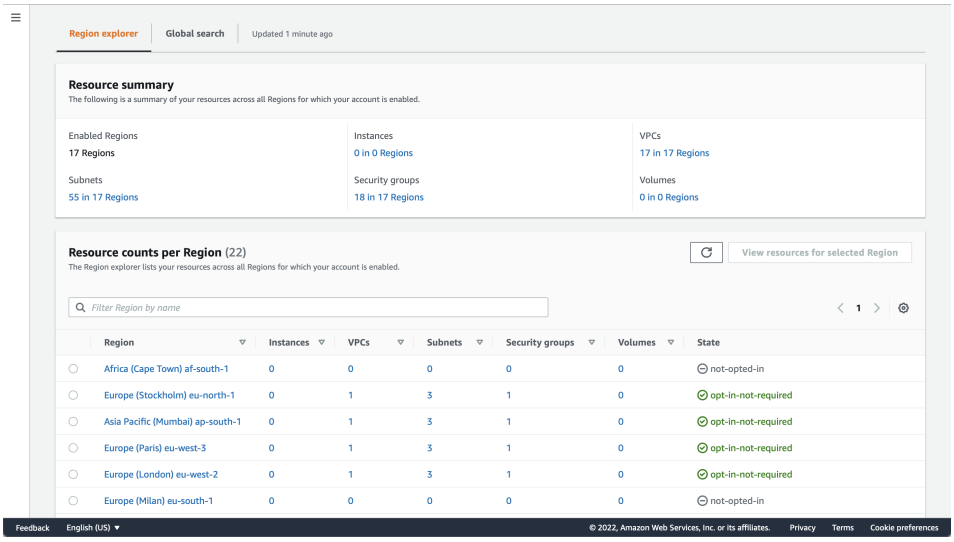


Figure 7: AWS Regions

4. **Pricing**: Instances on the shared tenancy (i.e., multiple customers will share the same physical resources) can be launched using three different pricing models:
    1. The on-demand pricing is a fixed rate that you pay for the instance. It can range from a few cents to a few dollars per hour depending on the instance type.
    2. If you want to use an instance for a fixed amount of time, you can purchase reserved instances (which are typically calculated yearly).
    3. Additionally, there are spot instances which are described below.
5. **Spot Instances**: Spot instances are special instances which allow users to bid for unused computing capacity. You can specify an hourly rate that you are willing to pay to use an instance. In addition, for each instance type under each availability zone, Amazon maintains a spot price which reflects the current demand for the instance type. If your bidding price is more than the spot price, an instance will be launched at the current spot price and will continue to run until the spot price exceeds your bid price. Spot instances are volatile but very useful for using instances for a few hours at prices that are typically much lower than the on-demand prices.
    1. **Warning: please manually tag the EC2 instances launched by Spot requests.** Spot requests behave differently than other AWS resources. There is no guarantee that a Spot Instance exists to fulfill the request at any given point in time. Due to the API design by AWS, there is **NO** built-in AWS API to map the instance by the identity of a Spot Instance request. Hence, the tags of spot requests will not apply to instances. For example, if you launch an EC2 instance from the web console with the "Launch Instance" button, if you selected "Request Spot Instances" at the Step "Configure Instance Details", the tags you add at the step "Add Tags" will only tag the spot request, and you must manually tag the provisioned instance after.
6. **Instance Limits**: AWS has preset limits on the number of simultaneous instances, per instance type, that an account can provision. You can check what the instance limits are for your account by checking the limits under the EC2 dashboard.

Amazon EC2 Storage

Amazon offers multiple options for storage on EC2 instances:

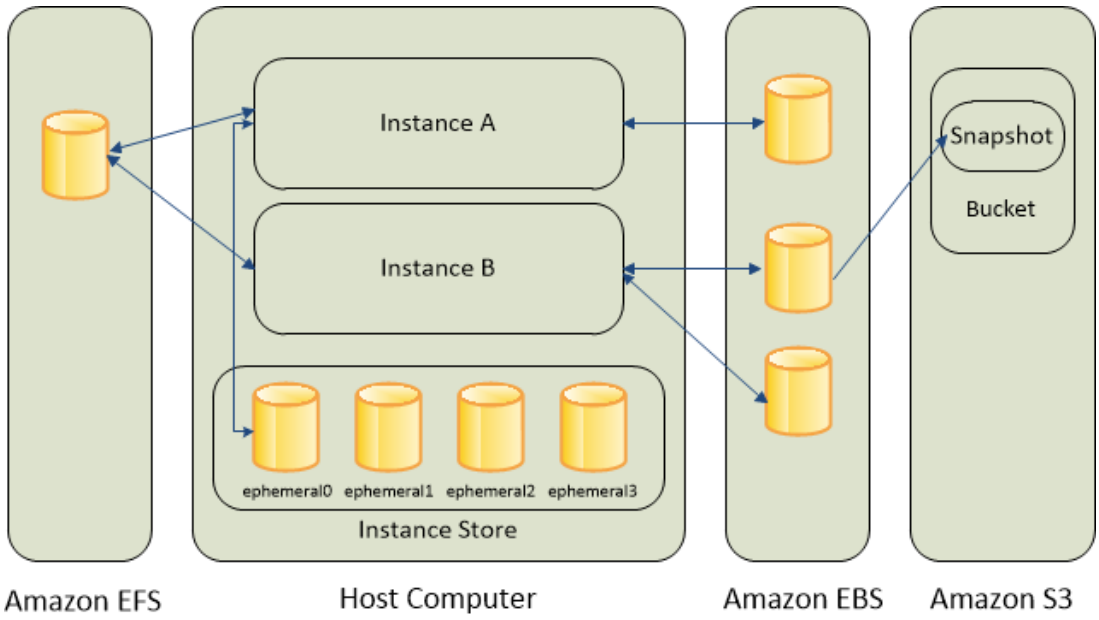


Figure 8: Storage options for EC2 instances. source (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Storage.html)

Amazon EC2 Instance Store: Instance stores are storage volumes that are present on the host computer that the instances are running on. Instance stores are temporary (ephemeral), block level storage. Instance store data is cleared when an instance is stopped or terminated.

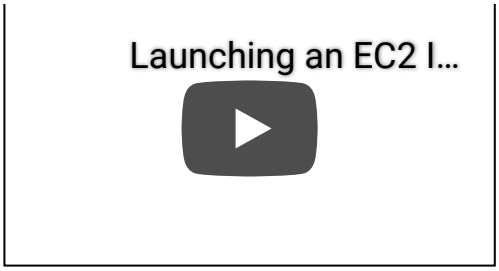
Amazon Elastic Block Store (EBS): EBS is a SAN-style storage system that can be used with EC2 instances. EBS presents volumes to the user that can be created independently of an instance and attached to instances as needed. EBS volumes are persistent and flexible. Multiple EBS volumes can be attached to an instance, and an EBS volume can be detached from an instance and attached to another. EBS incurs additional charges (GB/month) in addition to the EC2 instance charges. EBS volumes can also be backed up by creating a snapshot, which is stored in Amazon S3.

Amazon Simple Storage Service (S3): Amazon S3 is an object storage service which has a web services interface to store and retrieve data. Instances can access data directly on S3 using the web services interface. Amazon S3 is also used to store snapshots of EBS volumes.

Amazon Elastic File System (EFS): Amazon EFS provides scalable file storage for use with Amazon EC2. You can use an EFS file system as a common data source for workloads and applications running on multiple instances.

Launching and connecting to your instance

The following video will cover the procedures required to launch an EC2 instance. Please note that when you follow the video and practice, you should use up-to-date VM image (e.g., Ubuntu 20.04) and VM instance type (e.g., `t3.micro`), and the correct tag (`project : getting-started-with-cloud-computing`) instead of the example ones in the demo video.



Video 4: Launching an EC2 Instance

To launch an instance, log on to your AWS account and use the AWS Management Console to go to the EC2 Dashboard. From there, click on Instances and Launch Instance. You can choose the classic wizard and specify an AMI, Instance Type, region and availability zone, as well as storage options and select a key pair so to connect to your instance.

Once you have completed configuring your instance, you can see your instance listed in the EC2 Dashboard, and you should be able to see its state momentarily being shown as "pending".

Once the instance has been launched, the state should change to "running", with a Public DNS address being assigned to it:

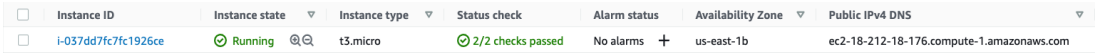
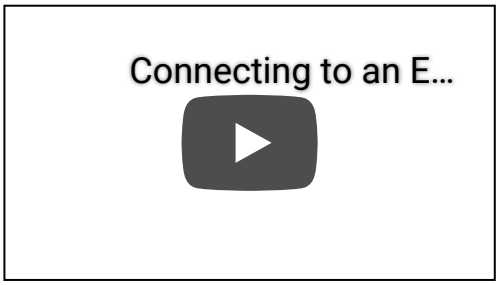


Figure 9: An Example Instance launched on EC2

You can now connect to this running instance over SSH. The following video should show you how to connect to an EC2 instance:



Video 5: Connecting to an EC2 Instance

To recap, the command in Linux to connect to an instance is:

```
ssh -i key_file.pem username@<instance-dns>
```

and the `pem` file should have as narrow permissions as possible:

```
chmod 400 key_file.pem
```

Information

Tip

If you are planning to use PuTTY in Windows, you will need to convert the PEM key into PPK (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html) format using the PuTTYgen program and specify it as the key when connecting to your instance.

We would also like to highly recommend XShell (https://www.netsarang.com/en/xshell/), an advanced Terminal software which is free for home and school use. You can directly import PEM keys into XShell for use with your SSH sessions.

Managing Security Groups

Instances launched on EC2 are publicly accessible using their Public DNS address. In order to better secure instances, EC2 opens only port 22 by default. This is controlled using Security Group definitions in the EC2 Dashboard. You can configure inbound and outbound ports and destinations.

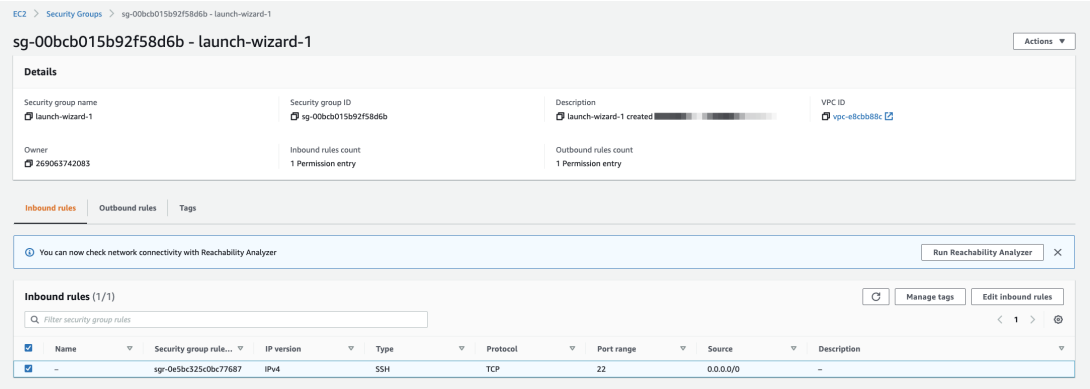


Figure 10: An Example EC2 Security Groups

Information

Tasks to try

You should get comfortable doing the following tasks:

1. Configure and launch a `t3.micro` EC2 Instance using either the Ubuntu or Amazon Linux AMIs.
2. Connect to the EC2 Instance using SSH from your computer.
3. SFTP/SCP a small file from your computer to the EC2 instance and vice-versa.
4. Use any text editor on the instance to create a small script or program (such as a bash or python script) and run it on the instance. This program could simply echo a string to `stdout` or filter words from an input text file.
5. Attach, format and mount an EBS device to this running EC2 instance. Refer to the EC2 documentation (http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-attaching-volume.html) for specific instructions.
6. Terminate the EC2 instance when done and delete any volumes.

Simple Storage Service (S3)

Simple Storage Service (S3)

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. Amazon S3 provides interfaces to write, read, and delete files (also known as objects in Amazon's parlance) of sizes 1 byte to 5 terabytes of data each. The number of objects that you can store is unlimited. Each object is stored in a bucket and retrieved via a unique, developer-assigned key.

When working with S3, one logical difference compared to a traditional file system is that S3 does not have the notion of folders or subfolders. The only logical container is a bucket, but files (objects) can have names with `/` which can be used to preserve the hierarchy of existing files when uploading and downloading. The following video provides an overview of S3:

Video 6: Amazon S3

Working with S3

The following video will introduce and demonstrate S3 functionality through the AWS Console:

Video 7: Amazon S3 Usage

Using the AWS Management Console, you can visit the S3 Management Console, which gives you an interface to create and manage buckets and files.

When you create a bucket on S3, you must specify a bucket name that is unique among all the buckets currently in S3 (similar to choosing a username for a web portal). The S3 bucket namespace is shared by all users of the S3 system. There are also additional restrictions (http://docs.amazonwebservices.com/AmazonS3/latest/dev/BucketRestrictions.html) on bucket names.

Any object stored on S3 is also accessible via HTTP. You should be able to access your bucket from a browser using the URL: `http://s3.amazonaws.com/<bucket-name>/<object-name>`. Please note that you will need the appropriate permissions enabled for your bucket and object to allow for anonymous HTTP access.

Information

Tasks to try

You should get comfortable doing the following tasks:

1. Create a bucket on S3.
2. Store and retrieve a small file on S3 using both the web interface and the command-line tool (covered in the next section).
3. Set permissions to be able to access an object on S3 using an HTTP link.

AWS Command-Line Tool

AWS Command Line Tool (aws-cli)

So far, you have seen how to use two services in AWS, namely EC2 and S3. You have also learned how to access these services manually via the AWS Management Console.

In order to fully leverage the power of cloud services, you should be able to **dynamically** manage AWS resources in an automated fashion. AWS makes all of its cloud service accessible via Tools and Application Programming Interfaces (APIs), which allow you to provision, configure, manage and deploy AWS resources. One of the tools that is extremely useful to access AWS resources is the AWS Command Line Interface (https://aws.amazon.com/cli/) Tool, also known as `awscli`.

Warning

AWS CLI version 2 v.s. AWS CLI version 1

The AWS CLI version 1 is the original AWS CLI. The AWS CLI version 2 is the most recent major version of the AWS CLI officially released on February 10, 2020.

Note: You are required to only use the AWS CLI version 1 to complete this course. For example, most of your work in the future projects will rely on a VM image pre-configured by the course staff, and the AWS CLI version on the VM image is AWS CLI version 1.

The following video covers the installation and usage of the `awscli` tool.

Video 8: AWS CLI Tool

Documentation for installing and working with the AWS CLI may be found in the AWS user guide (https://docs.aws.amazon.com/cli/v1/userguide).

To install the `awscli` tool:

1. Use `pip` to install the most update to date package

```
$ pip install awscli --upgrade --user
```

2. Alternatively use `apt-get` or `yum` in all modern linux distributions (although these repositories may not have the latest package versions)

```
$ sudo apt-get install awscli
```

To configure the `awscli` tool to use your AWS credentials, you can use the following command:

```
$ aws configure
AWS Access Key ID [None]: ********************
AWS Secret Access Key [None]: ********************
Default region name [None]: us-east-1
Default output format [None]: json
```

You can now use `aws` to access AWS resources from the command line. Refer to the `aws` man page for additional information.

To find more help information, refer to the online AWS CLI Command Reference (http://docs.aws.amazon.com/cli/latest/index.html), the AWS CLI user guide (https://docs.aws.amazon.com/cli/v1/userguide), or via the shell by running the following commands:

```
$ aws help
...
$ aws s3 help
...
$ aws s3 cp help
...
```

An example of using the AWS CLI to launch EC2 instances

To launch instance, you should first configure the AWS CLI if you have not done so:

```
$ aws configure
AWS Access Key ID [None]: YOUR AWS ACCESS KEY
AWS Secret Access Key [None]: YOUR AWS SECRET ACCESS KEY
Default region name [None]: us-east-1
Default output format [None]: json
```

Next, use following command to create a Security Group, Key Pair, and Role for the EC2 Instance

```
$ aws ec2 create-security-group --group-name devenv-sg --description "security g
roup for development environment in EC2"

$ aws ec2 authorize-security-group-ingress --group-name devenv-sg --protocol tcp
--port 22 --cidr 0.0.0.0/0

$ aws ec2 create-key-pair --key-name devenv-key --query 'KeyMaterial' --output t
ext > devenv-key.pem

$ chmod 400 devenv-key.pem
```

Finally, you are ready to launch an instance and connect to it. Note that you need to provide AMI_ID (e.g., `ami-xxxxx`) and the ID of your defined security group (e.g. `sg-xxxxx`).

```
$ aws ec2 run-instances \
    --image-id YOUR_AMI_ID \
    --security-group-ids SECURITY_GROUP_ID \
    --count 1 \
    --instance-type t2.micro \
    --key-name devenv-key \
    --query 'Instances[0].InstanceId'
```

CloudWatch

CloudWatch

When working with cloud resources, you will want to keep tabs on their performance, costs as well as any sort of warning signs that point to possible errors or issues.

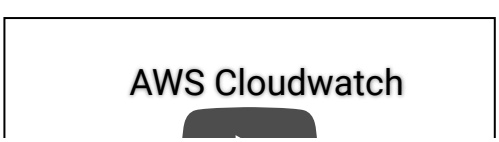
Amazon CloudWatch enables developers to monitor various facets of their AWS resources. Developers can use it to collect and track metrics from various AWS resources that are running on the AWS Cloud. Using APIs, CloudWatch also allows you to programmatically retrieve monitoring data, which can be used to track resources, spot trends and take automated action based on the state of your cloud resources on AWS. CloudWatch also allows you to set alarms, which constitute a set of instructions to be followed in case of an event that is tracked by CloudWatch is triggered.

CloudWatch can be used to monitor various types of AWS resources including:

- EC2 instances
- EBS volumes
- EMR job flows etc
- ELB Loads

For EC2 instances, CloudWatch allows you to monitor CPU, memory and disk utilization.

The video below describes the various features and usage of CloudWatch.



Video 9: Amazon CloudWatch

For more information on CloudWatch please refer to the Amazon CloudWatch documentation. (http://aws.amazon.com/documentation/cloudwatch/)