ECGR 4090/5090 Cloud Native Application Architecture
Lab 9 : Amazon Web Services (AWS)

In this lab, you will get hands-on experience with Amazon Cloud, also known as AWS. A public cloud such as AWS offers Infrastructure-as-a-Service (IaaS, for example, virtual machines, storage, network etc), Platform-as-a-Service (PaaS, for example, databases, messaging systems, Kubernetes, etc.), and Software-as-a-Service (SaaS, for example, machine vision, speech recognition etc.). Competing cloud vendors such as Google (GCP), Microsoft (Azure), and IBM (IBM Cloud) also provide similar services.

The material in this document is drawn from the relevant AWS documentation. You are encouraged to consult the extensive documentation for further information.
https://docs.aws.amazon.com/index.html

1. Create a new AWS Account

You will need a credit card. If you stick with the free tier services, there will be no charges. You are eligible for the free tier for one year after you open your AWS account. After a year has passed, you will no longer be eligible for the free tier and will be charged any applicable fees for your AWS usage.
Note 1: AWS account is different from any Amazon shopping account that you may have.
Note 2: Only one team member needs to create a new AWS account for the purpose of the lab. All are welcome to open their own accounts if desired.

To do -

1. Follow steps as listed here to set up your AWS account -
https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/

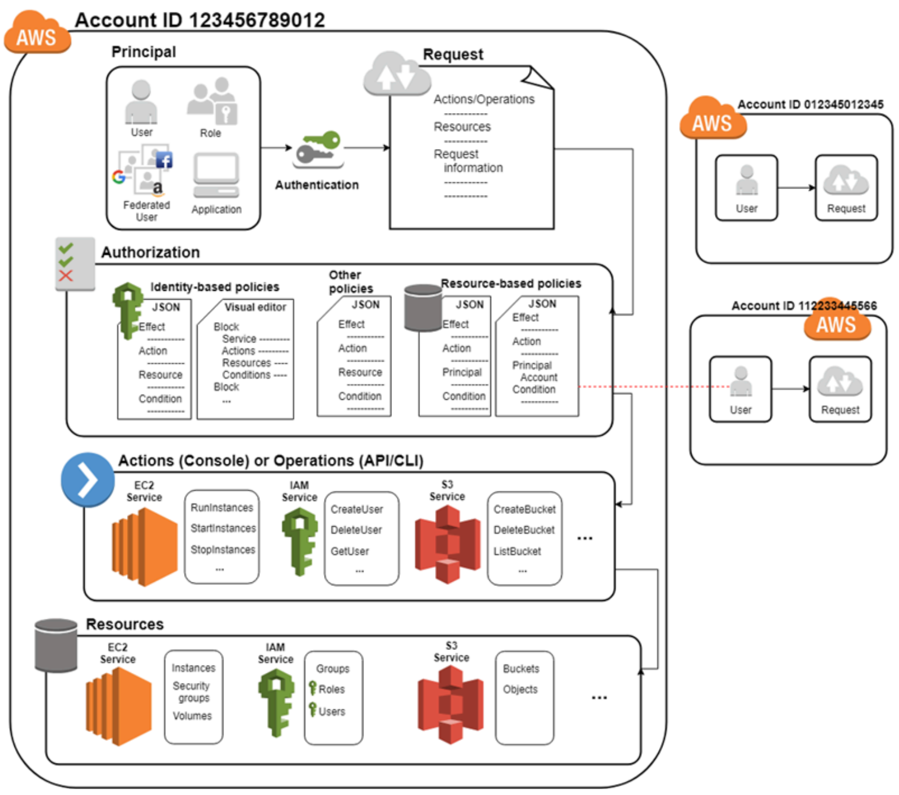
2. Setup AWS Free Tier usage alerts
It is important to monitor your usage using the AWS billing service. You can opt in to or out of the AWS Free Tier usage alerts through the Billing and Cost Management console. Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/

Under Preferences in the navigation pane, choose Billing preferences.
Under Cost Management Preferences, select Receive AWS Free Tier Usage Alerts to opt in to Free Tier usage alerts. To opt out, clear the Receive AWS Free Tier Usage Alerts check box.

2. AWS Identity and Access Management (IAM)

IAM provides the infrastructure necessary to control authentication and authorization for your account. The IAM infrastructure includes the following elements:

Principal, Request, Authentication, Authorization, Actions or operations, Resources



Principal - A *principal* is a person or application that can make a request for an action or operation on an AWS resource. The principal is authenticated as the AWS account root user or an IAM entity to make requests to AWS

Request - When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a request to AWS.

Authentication - A principal must be authenticated (signed in to AWS) using their credentials to send a request to AWS.

Authorization - You must also be authorized (allowed) to complete your request. During authorization, AWS uses values from the request context to check for policies that apply to the request. It then uses the policies to determine whether to allow or deny the request. Most policies are stored in AWS as JSON documents and specify the permissions for principal entities. AWS checks each policy that applies to the context of your request. If a single permissions policy includes a denied action, AWS denies the entire request and stops evaluating. This is called an explicit deny. Because requests are denied by default, AWS authorizes your request only if every part of your request is allowed by the applicable permissions policies.

Operations - operations are defined by a service, and include things that you can do to a resource, such as viewing, creating, editing, and deleting that resource.

Resource -- resource is an object that exists within a service. Examples include an Amazon EC2 instance, an IAM user, and an Amazon S3 bucket. The service defines a set of actions that can be performed on each resource.

IAM Users

When you create an AWS account, you create an AWS account root user identity, which you use to sign in to AWS. You can sign in to the AWS Management Console using this root user identity—that is, the email address and password that you provided when creating the account. This combination of your email address and password is also called your root user credentials.

When you use your root user credentials, you have complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password. This level of access is necessary when you first set up your account. **However, do not use root user credentials for everyday access. And do not share the root credentials with anyone.** Instead of sharing your root user credentials with others, you can create individual IAM users within your account. You can also create an individual access key for each user so that the user can make programmatic requests (say with Go or Python) to work with resources in your account. An IAM user doesn't have to represent an actual person; you can create an IAM user in order to generate an access key for an application.

AWS recommends that you create an IAM user for yourself and then assign yourself administrative permissions for your account. You can then sign in as that user to add more users as needed.

When you create an IAM user, they can't access anything in your account until you give them permission. You give permissions to a user by creating an identity-based policy, which is a policy that is attached to the user or a group to which the user belongs.

To do -

Creating your first IAM admin user and user group

1. Implement steps listed here to create admin user from console - https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-admin-group.html
2. Implement steps to create IAM users for members of your team (optional) https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_create-delegated-user.html

After you create IAM users (with passwords), those users can sign in to the AWS Management Console. To sign in, they need your account ID or alias. See this link on how to set account aliases https://docs.aws.amazon.com/IAM/latest/UserGuide/console_account-alias.html

Note: If your team has one AWS account, then the person who has put in the credit card information, should be the Admin user, and the rest of the team would be power users.

After you create IAM users (with passwords), those users can sign in to the AWS Management Console. To sign in, they need your account ID or alias.

3. Compute - AWS EC2

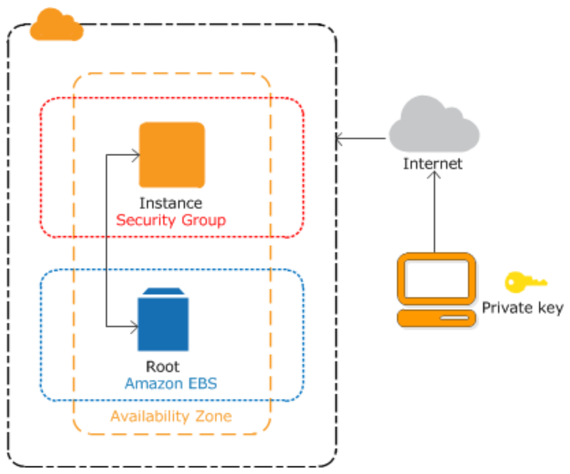
Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud.

Amazon EC2 provides the following features:

- Virtual computing environments, known as instances
- Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the software you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop, hibernate, or terminate your instance, known as instance store volumes
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes
- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources

- Virtual networks you can create that are logically isolated from the rest of the AWS Cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

The figure below shows an EC2 instance, the associated security group, and the AWS EBS storage associated with the instance.



To do -

1. Setup EC2 instance

Note: AWS Services are available in different regions. Please use either us-east-2 (Ohio) or us-east-1 (N. Virginia) for these labs. Stick with that region for the remainder of the lab.

Follow the instructions in the link below to 1) Create a key pair, and 2) Create a security group. Stick with Linux instances.
Note: Navigation bar is on the left on the EC2 console page. See Network & Security tab.
Note: In the security group, Inbound should be your laptop, and specify 22 for port. Outbound could be all.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for-amazon-ec2.html

2. Launch EC2 instance following instructions in the link below to stick with Amazon Linux 2 AMI machine images, and t2.micro instances.
   Note: When configuring instance details, be sure to enable auto-assign public IP. Use the security group created in Step 1.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

3. Connect to the instance from you local computer using SSH

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html

Note: Click on the instance to get the instance summary. You will find the Public IPv4 dns there.
User name is ec2-user
After logging into your ec2 instance, check the processor, memory, and filesystem information for your instance.

```
$ cat /proc/cpuinfo
$ cat /proc/meminfo
$ lsblk -f
```

4. Be sure to clean up the instance (under instance state)

Note: If you do not clean up the instance, you will continue to incur charges both for the EC2 instance, and the associated EBS storage.

4. Object storage - AWS S3

Amazon S3 is an object storage service that stores data as objects within buckets. An object is a file and any metadata that describes the file. A bucket is a container for objects.

To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a key (or key name), which is the unique identifier for the object within the bucket.

Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your Amazon S3 bucket in all AWS Regions. Amazon S3 achieves high availability by replicating data across multiple servers within AWS data centers. If a PUT request is successful, your data is safely stored.

To do -

1. Steps 1 - 5 described in the following links on creating a bucket, uploading an object to a bucket, downloading an object from a bucket, copying an object, and deleting an object from a bucket.

https://docs.aws.amazon.com/AmazonS3/latest/userguide/creating-bucket.html
https://docs.aws.amazon.com/AmazonS3/latest/userguide/uploading-an-object-bucket.html
https://docs.aws.amazon.com/AmazonS3/latest/userguide/copying-an-object.html
https://docs.aws.amazon.com/AmazonS3/latest/userguide/deleting-object-bucket.html

5. Key value storage - DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service. With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic.

DynamoDB stores data in tables. A table is a collection of data. Each table contains zero or more items. An item is a group of attributes that is uniquely identifiable among all of the other items. For example, In a People table, each item represents a person. Each item is composed of one or more attributes. For example, an item in a People table contains attributes called PersonID, LastName, FirstName, and so on.
To do -

1. Step 1 to 5 described in the following links to Create a table, Write data, Read data, Update data, Query data, and finally Clean up. You only need to do it from the AWS console (not CLI).

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-1.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-2.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-3.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-4.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-5.html

6 - Programmatically accessing AWS with Go SDK (Software Development Kit)

Thus far we have accessed AWS services from the AWS console. There are two other ways to access it - (1) from the command line (similar to a shell command) by installing the command line tools, and (2) programmatically from a number of different languages including Go, Python, Java, Javascript, and C++.
If you are interested in the CLI, follow the instructions in this link (optional) -
https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

To do -

1. In this lab, we will access AWS S3 with Go.

Create a lab9 folder. In the lab9 folder,
Follow the instructions in this link (Install the AWS SDK for Go V2, Get your AWS access keys)
https://aws.github.io/aws-sdk-go-v2/docs/getting-started/
Go to your home directory

```
$ cd
```

In your home directory, (note the dot before aws)

```
$ mkdir .aws
```

Use the nano (or any other text editor) to create a credentials file

```
$ nano credentials
```

Copy and paste the following into the credentials file replacing your Access Key ID and Secret access keys with those you created

```
[default]
aws_access_key_id = <YOUR_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_SECRET_ACCESS_KEY>
```

Save and exit.

Back to lab9 folder
Download the following Go code to create an S3 bucket. Edit the code to replace the region to us-east-1 (or the region you have been using throughout the lab).
https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/go/example_code/s3/s3_create_bucket.go
Save as create.go
Run with a unique bucket name
```
$ go run create.go <bucket-name>
```
You should get a successfully created bucket message.

Now upload a file to the bucket.
https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/go/example_code/s3/s3_upload_object.go
Save as upload_bucket.go. Be sure to change the region name in the code.
```
$ go run upload_bucket.go <bucket-name> <file that you want to upload>
```

List objects in the bucket
https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/go/example_code/s3/s3_list_objects.go
Save as list_bucket.go. Be sure to change the region name in the code.
```
$ go run list_bucket.go <bucket-name>
```

Delete all objects in the bucket
https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/go/example_code/s3/s3_delete_objects.go
Save as delete_objects_bucket.go. Be sure to change the region name in the code.
```
$ go run delete_objects_bucket.go <bucket-name>
```

Delete bucket
https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/go/example_code/s3/s3_delete_bucket.go
Save as delete_bucket.go. Be sure to change the region name in the code.
```
$ go run delete_bucket.go <bucket-name>
```

For more about Go SDK
https://aws.github.io/aws-sdk-go-v2/docs/

What next?

AWS has over 200 services, and is growing. Play around with some services. Please note whether they belong to the free tier, and what charges are involved. It is super important to tear down the service as soon as you are done using it, to avoid getting billed. And please stick with one region, since services invoked are region specific. **Don't end up making Amazon richer at your expense!**

Here's a comprehensive list of services -
https://docs.aws.amazon.com/index.html

You can get certified in AWS. A number of different certifications are available. I would recommend getting an AWS Developer Certification if you are interested in a career in cloud computing. You can find more details here -

https://aws.amazon.com/certification/certified-developer-associate/?ch=tile&tile=getstarted