

AWS - Homework 5

Mehmet Kadri GOFRALILAR, 22301421

May 15, 2024

1 Services and Utilities Provided by AWS

1.1 EC2

1.1.1 Launching Process of an EC2 Instance

In order to launch an EC2 instance, a key pair should be created first. Afterwards, launch configurations should be done from "EC2/Instances/Launch an instance". There are couple of choices provided in this page for multiple aspects given below. After entering the name for the instance, the AMI is chosen according to need, which is a template that includes OS, software etc. Then, instance type is chosen, which has a variety of options that have different prices according to the resources they provide (e.g. vCPU, Memory). Afterwards, the key pair created in the beginning is chosen to provide connection securely. Then, network settings are filled according to need. This field is really important, since there are many crucial configurations here. For example, security groups and allowed traffic is provided here. And finally (excluding additional settings), storage is provided. Both the size and the type is chosen here and they vary according to need. For example, while Provisioned IOPS SSD's are used for frequent I/O procedures and higher durability, general purpose SSD's are used for handling most workloads and should be preferred if sufficient since it is cheaper. Once these inputs are provided, instance can be launched and will be ready to connect.

1.1.2 Pricing Models of EC2

There are three different pricing models in EC2. These are:

1. On-Demand

On-demand Instance are the best option for applications with unpredictable short-term workloads since the pricing for used compute is by second. Also, this model doesn't require commitment.

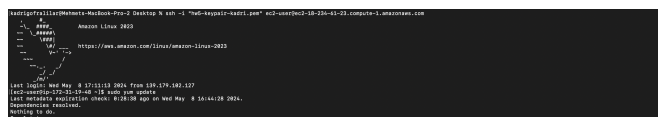
2. Reserved Instances

If the application has long-term and predictable workloads, this model is the second best choice. There are two options; 1-year commitment or 3-years commitment. 3-years commitment has a cheaper rate. This model also provides committed usage.

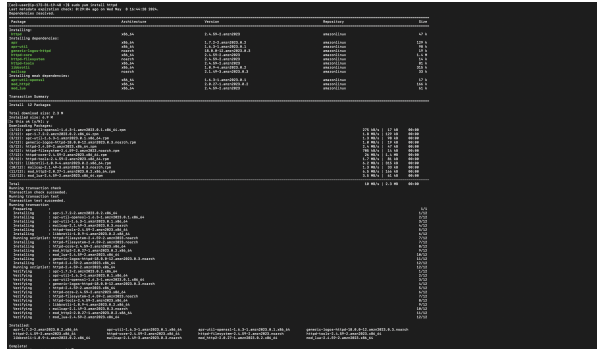
3. Spot Instances

Finally, spot instance model option has the price (saving up to 90%) that changes depending on supply and demand on AWS. However, it should be considered that this model works with bidding and if the price exceeds the bid, AWS can shut the instance down with short notice. Therefore, risk tolerance of the application must be evaluated and this model should be chosen only if the application can tolerate terminations.

1.1.3 Connection and Management to an EC2 Instance with SSH



(a) Update of the OS



(a) Installation of "httpd"

1.2 Databases

1.2.1 Comparison of DB Services in AWS

1. Amazon Aurora

Amazon Aurora is highly efficient and designed for applications that require high performance, reliability, and scalability. It is ideal for demanding workloads like e-commerce platforms or financial systems.

2. Amazon Relational Database Service

Amazon RDS is the go-to managed database service. It supports various database engines like MySQL, PostgreSQL, Oracle, SQL Server, and Amazon Aurora. RDS takes care of routine database tasks such as provisioning, backups, and scaling, helping to focus on application development.

3. Amazon DynamoDB

DynamoDB is a NoSQL database service. It is built for applications that need scalability, low-latency, and high availability. It also is perfect for use cases like gaming, shopping cart, and real-time analytics.

4. Amazon DocumentDB

Amazon DocumentDB is a document database service compatible with MongoDB. It is designed for applications requiring flexibility in schema design and JSON-based document storage. It is ideal for content management systems and mobile apps.

5. Amazon ElastiCache

Amazon ElastiCache is a caching service. It helps speed up applications by storing frequently accessed data in memory. It is great for reducing latency and improving performance.

6. Amazon MemoryDB

Amazon MemoryDB is Redis-compatible, in-memory database service. It is designed for applications requiring extremely low latency and high throughput. It is ideal for real-time analytics and caching.

7. Amazon Neptune

Amazon Neptune is a graph database service. It is optimized for applications that need to model complex relationships, like fraud detection, social networks or recommendation engines.

8. Amazon Timestream

Amazon Timestream is a time-series database service. It is built for analyzing time-series data at scale, making it perfect for IoT applications and event tracking.

9. Amazon QLDB

Amazon QLDB is a ledger database service. It provides a transparent, immutable, and verifiable transaction log, making it suitable for applications requiring trust and transparency in transaction history in areas such as healthcare and finance.

10. Amazon Keyspaces

Amazon Keyspaces is a Apache Cassandra-compatible database service. It's designed for applications needing scalability, availability, and durability while preserving the flexibility and power of Apache Cassandra.

1.2.2 Creation of an Amazon RDS Instance

In order to create an Amazon RDS instance, a DB subnet group should be created first. Afterwards, create configurations should be done from "RDS/Create database". Steps are as follows:

1. Choose "Standard create"
2. Your choice of engine option (MySQL in our case) and version (8.0.34) is chosen
3. Template is chosen (Free Tier does not include multi-AZ as a deployment option)
4. Name and credentials are entered. These will be used to connect.
5. Instance class is chosen. This option lets you to decide on the specifications of the resources, such as vCPU, RAM and Network.
6. Storage type and size are chosen. Options are same as the ones in EC2 configuration.
7. Connectivity has many fields to fill, including where we choose the db subnet group we created in the first place. This allows connections according to the subnet group and whether to open it to public access or not. Also database port is provided here under additional configuration.
8. Authentication is chosen according to need.
9. Enhanced Monitoring can be enabled to observe the CPU uses of threads and processes. This might help for further optimizations.
10. Under additional configuration, there might be important fields. For example, an initial database can be created, deletion protection can be activated or backup configurations (such as retention period, backup window or replication) can be done.

1.2.3 Connection to an Amazon RDS Instance

To connect to an Amazon RDS instance, the endpoint and port of the instance should be noted first. Afterwards, EC2 connection should be made. After installing the command-line client needed for the connection, "mysql -h <endpoint>-P 3306 -u <admin>-p" command is entered. Then, the password should be provided. Now, you are connected to the instance.

"SHOW DATABASES;" shows the databases in MySQL.

"CREATE DATABASE IF NOT EXISTS <database_name>;" creates a database if it does not exist already.

Then, "USE <database_name>;" should be used to run queries in a specific database.

For update, "UPDATE <table_name>SET <column_name>WHERE <condition>" is used.

For insert, "INSERT INTO <table_name>VALUES <values>" is used.

For delete, "DELETE FROM <table_name>WHERE <condition>" is used.

1.3 Security

1.3.1 AWS Shared Responsibility Model

In AWS Shared Responsibility Model, AWS is responsible for the security of the cloud (including the infrastructure, hardware, software, and networking); while the customer is responsible for security in the cloud (including data, identity and access management, configurations, and applications).

AWS resources can be protected by configuring security groups and network access control lists to control inbound and outbound traffic to EC2 instances and encryption can be used to protect the data in databases.

1.3.2 Configuration of AWS Identity and Access Management

To configure AWS Identity and Access Management, and IAM user should be created from "IAM/Users/Create user". Here it will be decided which permissions this user gets (e.g. AWS Management Console access). Afterwards, it is important to keep observing the permissions and protocols given to the user and what they have done. If there is an abuse, or any threat, permissions can be removed by visiting the "IAM/Users" page and clicking on the specific user name. Also new permissions can be added there if needed as well.

1.3.3 AWS Security Groups

AWS Security Groups play a crucial part in security by controlling inbound and outbound traffic, allowing you to define rules that permit or deny them based on source, destination, port, and protocol. This means you can set up virtual versions of safe* databases that are like on premise databases or virtual instances for example.

¹

¹Safe* does not mean entirely foolproof, however security is mostly provided in a way that does not allow unwanted connections.

1.4 Containers

1.4.1 Containerization

Containerization is the process of enabling the packaging and deployment of software; which increases the scalability, resource efficiency, isolation and portability. These aspects make it a perfect concept to use in cloud.

1.4.2 Deployment of a Docker Container using Amazon ECS

Deploying a docker container requires following several steps. First, the application needs to be dockerized. Then, the docker image is built and tested locally to make sure it is working correctly. Afterwards, an ECR repository is created from "Amazon ECR/Private registry/Repositories" and the docker image is pushed to the repository. Then, a task definition needs to be created from "Amazon Elastic Container Service/Create new task definition". This is where container image details, CPU and memory requirements, and task execution roles are specified. Next, an ECS cluster is created from "Amazon Elastic Container Service/Create cluster" to run the application. In this step, launch type and specifications are filled according to need. Then, an ECS service is created to manage the tasks running in the ECS cluster by configuring service settings (e.g. task definition, deployment options, load balancer settings). Finally, the containerized application needs to be deployed.

1.4.3 Differences between Amazon ECS and Amazon EKS

Making a choice between Amazon ECS and Amazon EKS depends on the application requirements and preferences. However, since both work based on containerized applications, choosing one is not a final decision. There is always a chance to go the other way. And for deciding; simplicity, flexibility, cost, compatibility and familiarity are important factors. For example while EKS is fully compatible with the Kubernetes ecosystem, ECS has its own ecosystem of tools and might have issues in terms of compatibility with some Kubernetes-based tools and applications. Or while calculating cost, ECS has easier pricing models and EKS might have additional cost to take into account depending on the workload. Also, it is reported by customers that ECS stands out with the simplicity it provides, meanwhile EKS provides more flexibility.

1.5 Monitoring

1.5.1 Monitoring and Logging Services

AWS offers several monitoring and logging services, each designed to help monitor, analyze, and optimize an AWS environment for performance, security, and compliance.

1. Cloudwatch: CloudWatch provides monitoring for AWS resources and applications while it can be used to collect and track metrics (such as CPU utilization, network traffic, and storage usage). Also CloudWatch Alarms allow you to set thresholds on metrics and trigger notifications or automated actions when thresholds are breached, helping you proactively respond to operational issues. Meanwhile CloudWatch Dashboards enable you to create custom dashboards to visualize and analyze metrics from multiple AWS resources in a single view. And finally, CloudWatch Logs allows you to collect, store, and analyze log data from your applications and AWS services. You can monitor logs in real-time, set up log metric filters, and create log-based metrics for deeper insights. This can also be used for debug prints.

2. CloudTrail: CloudTrail provides an audit trail of API activity in an AWS infrastructure. It records API calls made by users, services, and AWS Management Console actions, including the identity of the caller, timestamp, and source IP address. It is mostly used for monitoring inside the architecture.

3. AWS X-Ray: AWS X-Ray provides application performance monitoring for applications running on AWS. It helps identifying performance bottlenecks, diagnose errors, and optimize application performance.

1.5.2 Setup Process of CloudWatch Alarms

Steps on how to set up Cloudwatch alarms to monitor the health and performance of an EC2 instance is given below:

1. First, a metric to monitor should be chosen from "CloudWatch/Metrics". In this step, the AWS service to monitor is chosen (e.g. EC2). Then, the specific metric to monitor is chosen (e.g. CPUUtilization) and "Create alarm" button is clicked on top of the console.

2. Now, the conditions for the alarm are chosen. After clicking next, the notification and action (e.g. Lambda, Auto scaling or EC2) configurations are done, which will happen when the conditions set in the previous step are met.

3. After alarm name and description are provided, a preview is shown and editing is enabled for the previous steps. If the preview is as desired, creation is done by clicking "create".

1.5.3 Importance of Monitoring and Logging

Monitoring and logging are crucial procedures for troubleshooting and taking proactive measures. These procedures can help decreasing the possibility of crashes due to high traffic, network issues, and bottlenecks. Also observing the general performance of the application and finding the source of problems if there are any are great benefits. And when it comes to analyzing access logs, audit trails, and system events, it helps identifying suspicious activities, unauthorized access attempts, or security breaches and taking action to mitigate risks and strengthen security controls.

1.6 Storage

1.6.1 Storage Options in AWS

There are several options provided by AWS for storage purposes, which are Amazon EBS (Elastic Block Store), Amazon EFS (Elastic File System), and Amazon S3 (Simple Storage Service).

1. Amazon EBS: Amazon Elastic Block Store operates based on block storage, which is similar to hard drives. While this service provides consistent and low-latency performance for I/O-intensive workloads (e.g. databases, transactional applications, and high-performance computing), it is also cost-effective, secure and easy to use. This service offers six different volume types; which are General Purpose SSDs (2 types), Provisioned IOPS SSDs (2 types), Throughput Optimized HDDs and Cold HDDs. While General Purpose SSDs can handle most workloads, Provisioned IOPS SSDs have higher volume size and MaxIOPS/volume, four times the General Purpose SSDs offer (with higher cost, of course). Meanwhile, Throughput Optimized HDDs are ideal for frequently accessed, throughput-intensive workloads. And finally, Cold HDDs are ideal for less frequently accessed workloads with large cold datasets with the lowest cost among all types of EBS volume types.

2. Amazon EFS: Amazon Elastic File System operates based on shared file system, which allows multiple EC2 instances to access files simultaneously. This service provides simple, scalable, and serverless shared storage. EFS offers moderate performance for general-purpose workloads and is ideal for use cases that require shared file storage across multiple EC2 instances, such as content management systems.

3. Amazon S3: Amazon Simple Storage Service operates based on object storage, which can be used storing and retrieving any type of data, including files, images, videos, and backups. This service offers extremely high durability and availability and is commonly used for data backup/archiving and static website hosting.

1.6.2 Use Cases and Characteristics of Storage Services

In certain scenarios, some storage services surpass others. For example, while Amazon S3 is ideal for storing large volumes of unstructured data, such as media files, backups, and log files; Amazon EFS is better for applications requiring shared file storage across multiple EC2 instances, such as content management systems and Amazon EBS is better for hosting databases or running high-performance computing workloads requiring low-latency I/O operations.

1.6.3 Usage of Amazon S3 Bucket

Amazon S3 is a fairly simple service to use. First, a bucket should be created if there are none from "Amazon S3/Buckets/Create bucket", like given in Figure 1. Then, the bucket created clicked and "Upload" button is clicked to upload files. After the desired files are uploaded (via drag-drop or choose files options) as shown in Figure 2, "Upload" button is clicked. In order to download any file, the desired file is chosen among others in the bucket and "Download" button is clicked as shown in Figure 2. And finally, the files in the bucket can be managed by selecting the file and choosing the desired action (e.g. delete, copy or move to another bucket) as it is shown in Figure 3.

Create bucket

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East 1N, Virginia us-east-1

Bucket type [Info](#)

- General purpose** (Selected)
Recommended for most use cases and access patterns. General-purpose buckets are the default S3 bucket type. They allow a mix of storage classes that automatically move objects to save costs when they're not accessed often.
- Directory - New**
Not supported for low latency use cases. These buckets use only the S3 Express file-based storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)
homework5-s3bucket-kadri

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket [Optional](#)
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Object Ownership [Info](#)
Control ownership of objects within this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can modify access to objects.

- ACLs disabled (Recommended)**
All objects in this bucket are owned by the bucket owner. Access to this bucket and its objects is specified using only policies.
- ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner: root

Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access is not granted to the bucket or the objects in the bucket, turn on Block all public access. These settings apply only to this bucket and its access points. **Important:** We recommend that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work properly without public access. To help validate your level of public access to the bucket or objects within, you can configure the individual settings below to test your specific design use case. [Learn more](#)

☒ **Block all public access**
Turning this setting on enforces turning on all four settings below. Each of the following settings are independent of one another.

- ☒ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 and Amazon S3 Select access permissions specified for many bucket and object ACLs, and prevent the creation of new public access points for buckets and objects. This setting doesn't change any existing permissions that allow public access to data resources.
- ☒ **Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 and Amazon S3 Select access permissions specified for many bucket and object ACLs, and prevent the creation of new public access points for buckets and objects. This setting doesn't change any existing permissions that allow public access to data resources.
- ☒ **Block public access to buckets and objects granted through any public bucket or access point policies**
S3 and Amazon S3 Select access permissions specified for many bucket and object ACLs, and prevent the creation of new public access points for buckets and objects. This setting doesn't change any existing permissions that allow public access to data resources.
- ☒ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 and Amazon S3 Select access permissions specified for many bucket and object ACLs, and prevent the creation of new public access points for buckets and objects. This setting doesn't change any existing permissions that allow public access to data resources.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in any Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

☒ **Disable**

☐ **Enable**

Tags - optional [Info](#)
You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

[Add tag](#)

Default encryption [Info](#)
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

- ☒ **Server-side encryption with Amazon S3 managed keys (SSE-S3)**
- ☐ **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**
- ☐ **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing page. [Learn more](#)

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

☐ **Disable**

☒ **Enable**

Advanced settings

☒ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

Figure 1: Creation of an S3 Bucket

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose [Add files](#) or [Add folder](#).

Files and folders (1 Total, 471.6 KB)

All files and folders in this table will be uploaded.

[Remove](#) [Add files](#) [Add folder](#)

<input type="checkbox"/>	Name	Folder	Type
<input type="checkbox"/>	s3_create_bucket.png	-	image/png

Destination [Info](#)

Destination
s3://homework5-s3bucket-kadri

(a) Upload Image File

homework5-s3bucket-kadri [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (1) [Info](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, [grant permissions](#).

<input checked="" type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input checked="" type="checkbox"/>	s3_create_bucket.png	png	May 13, 2024, 16:22:12 (UTC+03:00)	471.6 KB	Standard

Actions [Download as](#) [Share with a presigned URL](#) [Calculate total size](#) [Copy](#) [Move](#) [Initiate restore](#) [Query with S3 Select](#) [Edit actions](#) [Rename object](#) [Edit storage class](#) [Edit server-side encryption](#) [Edit metadata](#) [Edit tags](#)

(b) Download Image File

Figure 2: Upload and Download Process to an S3 Bucket

Objects (1) [Info](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, [grant permissions](#).

<input checked="" type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input checked="" type="checkbox"/>	s3_create_bucket.png	png	May 13, 2024, 16:22:12 (UTC+03:00)	471.6 KB	Standard

Actions [Download as](#) [Share with a presigned URL](#) [Calculate total size](#) [Copy](#) [Move](#) [Initiate restore](#) [Query with S3 Select](#) [Edit actions](#) [Rename object](#) [Edit storage class](#) [Edit server-side encryption](#) [Edit metadata](#) [Edit tags](#)

Figure 3: Managing Files in an S3 Bucket

1.7 API Gateway

1.7.1 Purpose and Functionality of AWS API Gateway

Amazon API Gateway is a service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. It acts as a front door for applications and can be used to expose AWS resources as APIs, enabling seamless integration with applications, mobile apps, or third-party services. RESTful APIs and WebSocket APIs are supported protocols by this service. Defining RESTful API endpoints and methods is possible by using the API Gateway console. Then, integration with the back end services (e.g. Lambda functions or other AWS services) needs to be configured. Then, the API can be deployed.

1.7.2 Creation, Configuration and Testing of an AWS API Gateway Endpoint

Creating, configuring and testing an AWS API Gateway endpoint can be done by following these steps:

1. A new API Gateway endpoint is created from "API Gateway/APIs/Create API". This is done by choosing the API type first. We go with REST API. Then, name and description is provided and "Create API" button is clicked as shown in Figure 4. Now, we have an endpoint created.

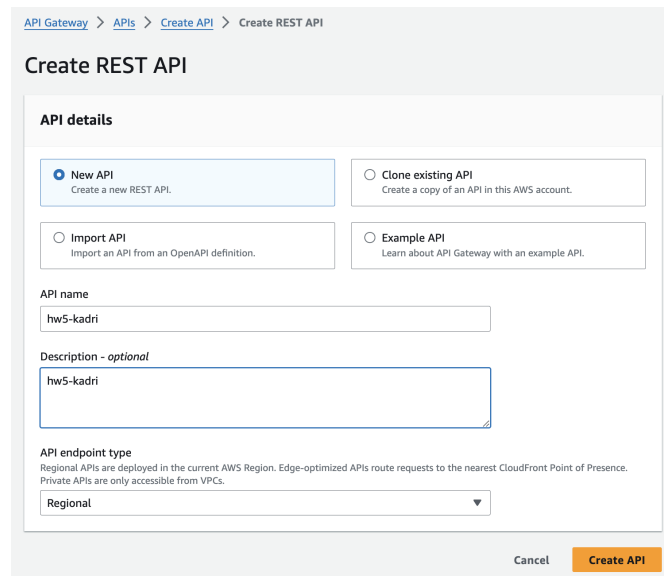
The screenshot shows the 'Create REST API' page in the AWS API Gateway console. The breadcrumb trail at the top is 'API Gateway > APIs > Create API > Create REST API'. The main heading is 'Create REST API'. Under 'API details', there are four radio button options: 'New API' (selected), 'Clone existing API', 'Import API', and 'Example API'. Below these, there is a text input for 'API name' with the value 'hws-kadri' and a larger text area for 'Description - optional' also containing 'hws-kadri'. Further down, under 'API endpoint type', there is a dropdown menu with 'Regional' selected. At the bottom right, there are 'Cancel' and 'Create API' buttons.

Figure 4: Managing Files in an S3 Bucket

2. After creation, the page is redirected to "Resources" page. Here, new resources can be created by create resource button. Then, name is entered. To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for the API. But before that, CORS option should be chosen here as shown in Figure 5. Now, we have our first resource.

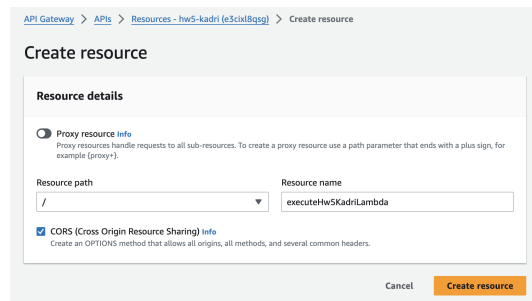
The screenshot shows the 'Create resource' page in the AWS API Gateway console. The breadcrumb trail is 'API Gateway > APIs > Resources - hws-kadri (e3chl8gg) > Create resource'. The main heading is 'Create resource'. Under 'Resource details', there is a radio button for 'Proxy resource info' which is selected. Below this, there is a 'Resource path' dropdown menu with '/' selected and a 'Resource name' text input with the value 'executeHwsKadriLambda'. At the bottom, there is a checkbox for 'CORS (Cross Origin Resource Sharing)' which is checked. At the bottom right, there are 'Cancel' and 'Create resource' buttons.

Figure 5: Managing Files in an S3 Bucket

3. After resource is created, a service instance to integrate to the method we will create needs to be created. We go with Lambda, which can be created from "Lambda/Functions/Create function" as desired.

4. After the lambda is also created, methods can be created. After clicking on the desired resource, "create method" button is clicked as seen in Figure 6. Then, Method type, integration type and the instance are chosen. Here, we choose the Lambda function we created earlier as seen in Figure 6. There are detailed configuration options below, which are not needed for this example.

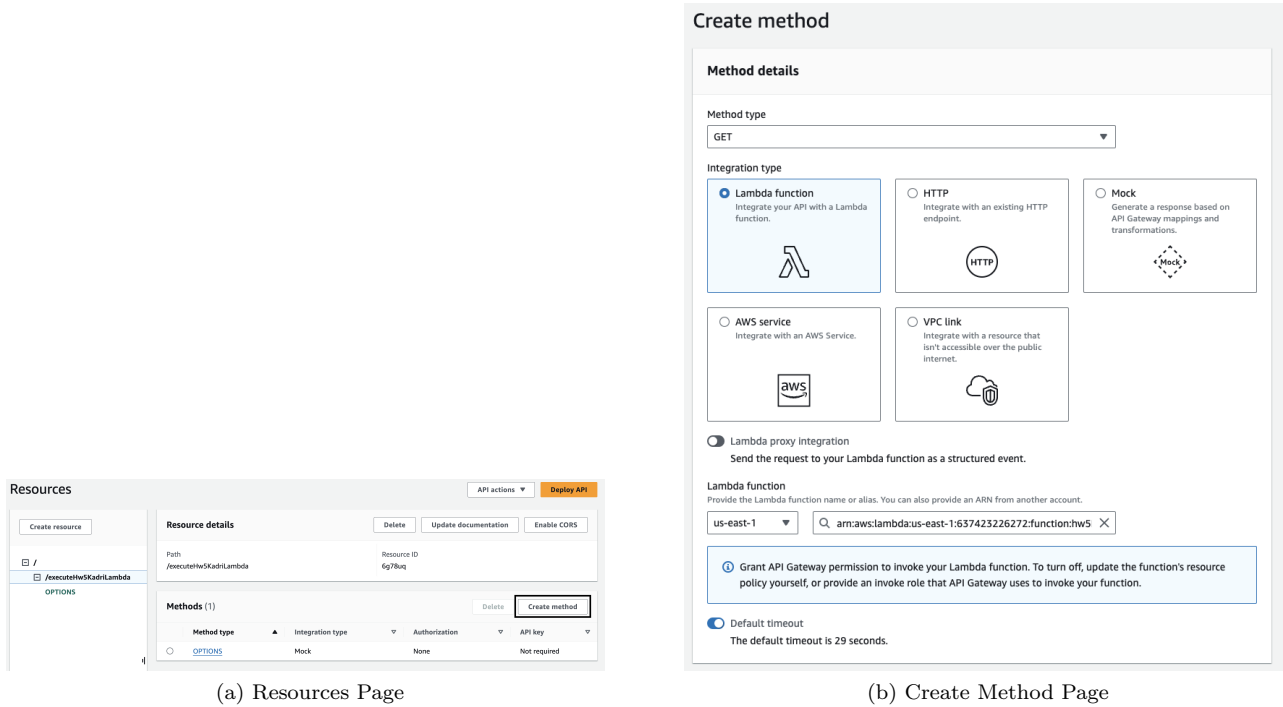


Figure 6: Steps for method creation

5. Now, we can see that the API Gateway is added as a trigger to the Lambda as shown in Figure 7. After checking if the integration has any issues, we can proceed with deploying the API. But before that, CORS configuration needs to be done. After clicking on the resource again, "Enable CORS" is clicked. Here, all Gateway responses and Access-Control-Allow-Methods given are chosen as shown in Figure 7 and "Save" is clicked.

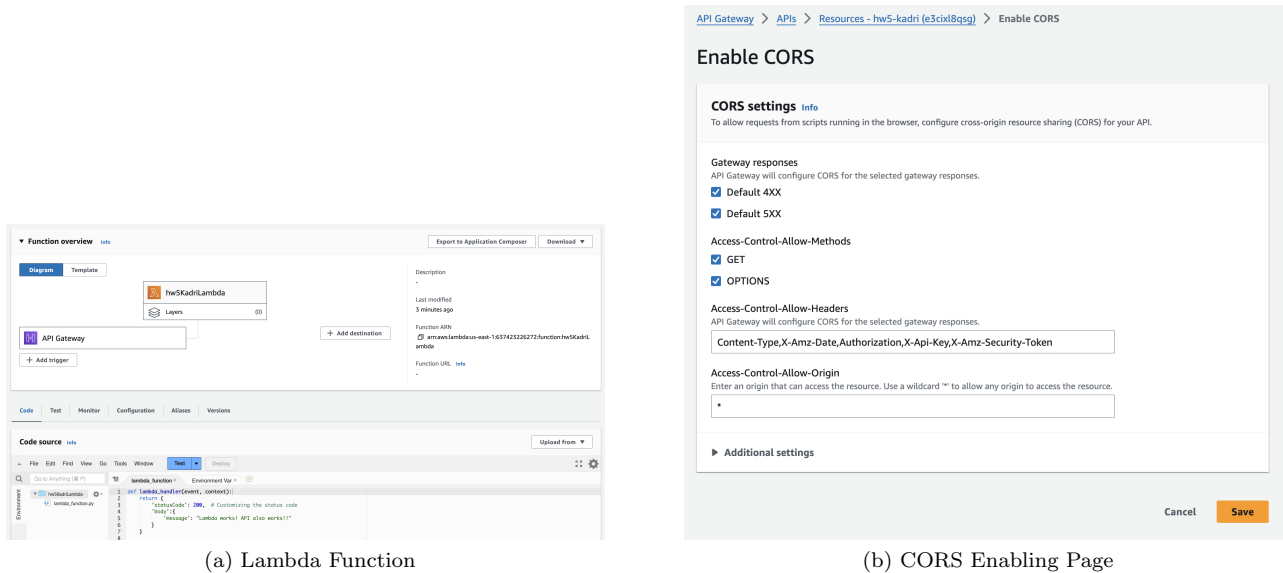


Figure 7: Last checks before deployment

6. The API is allowing requests from scripts running in the browser now. So, we can deploy and test the API. For deploying, click on "Deploy API" button. Then, choose the stage the API will be deployed, or create one if there aren't any as shown in Figure 8.

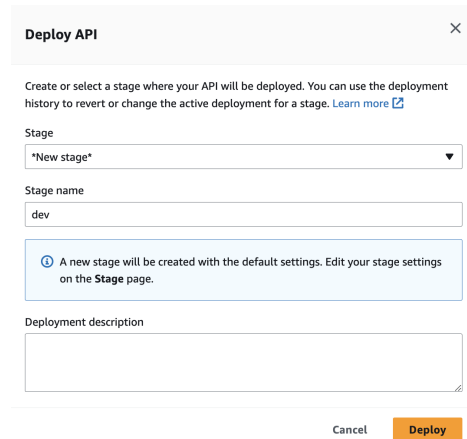


Figure 8: API Deployment

7. Now, the page is redirected to Stages page and here, we can copy the invoke URL of the method we created as shown in Figure 9.

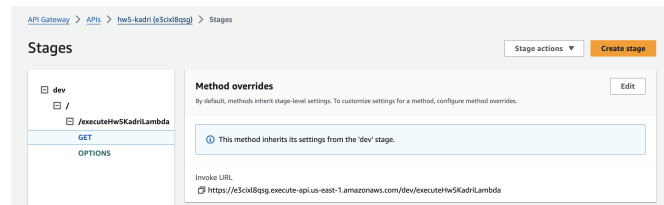


Figure 9: API Method Invoke URL

8. As for the testing, the URL can be testing using any browser or any API platform such as Postman as shown in Figure 10.

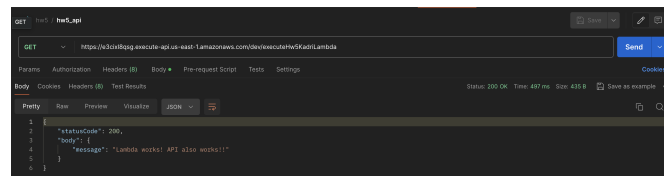


Figure 10: API Testing

1.7.3 Benefits of AWS API Gateway

Using API Gateway offers several benefits that can significantly improve the overall API management experience. This service has some key benefits, including authentication, throttling, and monitoring.

For example, by implementing authentication and authorization at the API Gateway level, centralizing security controls and protecting the backend resources from unauthorized access is possible.

Also by implementing throttling and rate limiting policies, it is possible to control the rate of incoming requests to the APIs. The limits can be set on the number of requests per second, per minute, or per hour for each API key or client IP address. This helps prevent abuse, denial-of-service (DoS) attacks, and excessive usage of API resources. It also helps protect backend services from being overwhelmed by sudden increase in traffic.

And finally, API Gateway provides detailed access logs that capture information about incoming requests, response codes, request parameters, and execution logs for backend integrations. This helps troubleshooting issues, analyzing usage patterns, and optimizing API performance.

1.8 Networking

1.8.1 Importance of Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (VPC) is a service that enables launching AWS resources in a custom created logically isolated virtual network. It is possible to define IP address ranges, subnets, route tables, and network access control lists (ACLs) to segment and isolate resources within the VPC. VPC provides a secure and isolated environment for hosting AWS resources and scales seamlessly, expanding the network infrastructure as the workload grows.

1.8.2 Creation of Amazon Virtual Private Cloud

For creating and configuring an Amazon VPC, the steps to follow are given below:

1. First, a VPC needs to be created from "VPC/Your VPCs/Create VPC". Here, the name of the VPC, the IPv4 CIDR block, and tenancy option (default or dedicated) are given.
2. Second, a subnet needs to be created from "VPC/Subnets/Create subnet". Here, the "Create subnet" button needs to be clicked. Then, the subnet name is given. After, VPC we created and the IPv4 CIDR block for the subnet are chosen. Then, the availability zone for the subnet is chosen and the "Create" button is clicked to create the subnet. This step can be repeated if multiple subnets needed.
3. Thirdly, a route table needs to be created from "VPC/Route tables/Create route table". Here, the name of the route table is entered and the VPC created before is chosen and the "Create" button is clicked.
4. Now, the route table will be edited to add routes from "VPC/Route tables/{route_table_id}/Edit routes". By clicking "Add route", destination and target are entered and "Save changes" is clicked.
5. Later, a new network ACL needs to be created from "VPC/Network ACLs/Create network ACL". Here, the name of the ACL is entered and the VPC created before is chosen and the "Create" button is clicked.
6. Now, the inbound or outbound rules can be edited for allowing or denying traffic.
7. Finally, the subnet needs to be linked to the network ACL and Route table created. To do this:
 - a. From route tables page, subnet associations is clicked and the subnet is selected.
 - b. From subnets page, Network ACL is clicked and the Network ACL is selected.Configuration is finally done.

1.8.3 Entegration of Amazon Virtual Private Cloud

The connection process of an EC2 instance and an RDS instance is given below in Figure 11. This is done while creating the RDS.

Connectivity Info

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

☐ Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

☒ Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

EC2 instance Info
Choose the EC2 instance to add as the compute resource for this database. A VPC security group is added to this EC2 instance. A VPC security group is also added to the database with an inbound rule that allows the EC2 instance to access the database.

i-08fae759cce167a36
hw5-kadri-vpc-ec2

Some VPC settings can't be changed when a compute resource is added
Adding an EC2 compute resource automatically selects the VPC, DB subnet group, and public access settings for this database. To allow the EC2 instance to access the database, a VPC security group rds-ec2-X is added to the database and another called ec2-rds-X to the EC2 instance. You can remove the new security group for the database only by removing the compute resource.

Virtual private cloud (VPC) Info
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

hw5-kadri-vpc (vpc-033b7d29d9aaf728)
1 Subnets, 1 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

Figure 11: Connecting an RDS instance to an EC2 instance during creation

1.9 Boto3 Library

1.9.1 Purpose and Functionality of Boto3 Library

Boto3 library is the AWS SDK for Python and it is used for easily interacting with AWS Services and simplifying their integration.

1.9.2 Usage of Boto3 Library

Boto3 makes it easy to interact with AWS services programmatically easily. For example managing an S3 bucket can be easily done as it is shown in Figure 12.

[illegible]

Figure 12: S3 Bucket management via Boto3

1.9.3 Code Example Using Boto3

The code used above is given here:

```
import boto3

s3_client = boto3.client('s3')
s3_resource = boto3.resource('s3')

response = s3_client.list_buckets()

for bucket in response['Buckets']:
    print(bucket['Name'])

my_bucket = s3_resource.Bucket('homework5-s3bucket-kadri')

for my_bucket_object in my_bucket.objects.all():
    print(my_bucket_object)

s3_client.delete_object(Bucket='homework5-s3bucket-kadri', Key='Boto3 1.34.104 documentation.pdf')

for my_bucket_object in my_bucket.objects.all():
    print(my_bucket_object)

with open('Boto3 1.34.104 documentation.pdf', 'rb') as data:
    s3_resource.Bucket('homework5-s3bucket-kadri').put_object(Key='Boto3 1.34.104 documentation.pdf', Body=data)

for my_bucket_object in my_bucket.objects.all():
    print(my_bucket_object)
```

2 Conclusion

This homework taught me that understanding AWS services in detail is crucial for efficient deployment, management, and optimization of cloud-based applications. Additionally, making informed decisions regarding instance types, storage options, security measures, and network configurations is a must for maximizing the benefits of AWS.