***Group Project***

***Group 8***

Harsh Guleria-8959212, Anuj Kumar-8976000, Jatin Mehta-8986070, Payne Jackson-8757065

INFO 2350

Nikola Cedic

5 August 2025

**Task-1-Architecture Diagram(Anuj Kumar)**

**Description**

In this task I worked on the designing of the aws artitecture . And this diagram I created and it is showing the vpc is containing both public and private subnet, NAT Gateway, an Internet Gateway, a Web Server, and Database Servers. And I also include security group with the rules for HTTP, HTTPS, SSH, . And this design also ensure the internt connectivity for the public resources, while the private still remain secure and protected.

**What tool I use.**

I use word to create this diagram and I add all the components which is needed.

**Why I used each component and its meaning and the importance.**

**Vpc-** The fullform of the vpc is virtual private cloud it is a logically isolated section of the aws where we can control network and also used to secure host and manage the resources in one control network.

**Public subnet-** In this w eb server can acceseed from the internet and which is all ows the income ing web traffic .

**Private subnet** – This private subnet hold the sensitive resources, like the database, that should not accessible from the internet.

**Internet gateway**- I use this by because it provide internet access to resources in the public subnet.

**Nat Gateway-** Its allows the resources in the private subnet to access the internet without expose them publicily and which also increase the security.

**Security Group-** The security goup helps us to control the inbound and outbound traffic. And the rules allow only required traffic, which also increase the security**.**

**Web server(Public subnet)-**It handle the user requests and delivers the application to the internet

**Database Server (Private Subnet)-** It storres the application data securely and it will only accessible from the web server

**Screenshot**

Internet

Internet Gateway

VPC 10.0.0:0/16

Nat Gateway

Private Subnet

Private Route Table

Public Route Table

Public Subnet

Security Group

http (80), https (443),

SSH (22)

Web Server

Security Group

Only from Web Server

Database Server

Database Server

This diagaram shows shows the vpc with public and private subnets , and also include NAT gateway, web server, database servers, security groups, and internet connectivity.

**Observation**

And while I was creating this diagram I also have to makes sure that my public subnet can access the internat by the help of the internet gatway. And the private subnet remain isolate and I also use the nat gateway for the external connectivity. And the security group will be configure in the task 2 for the inbound traffic to restrict unwaned access. And this setup and diagram will ensure that or web server is public accessible while our database remain protected.

**Reflection**

This diagram which I created I think will help my group for creating the secure and scable network artitecture in the aws. And I also gained some more experience in placing logical componenmets in the vpc for applying security practices. And this experience also help me to understand the importance of the public and private resources and I show proper cloud infrastructure in the diagram.

**Task-2-Implemenetation( Harsh Guleria)**

**Description**

In this task I implement the aws environment according to the architecture . And I first create the web app-vpc, and then I set up four subntes in which two for the public and same other two for the private . And then I created the internal gateway name webapp-igw and attached that in to vpc. And also created the public route table which I associate with the public subnets. And then I step up the Nat gateway in public subnet only . After that I create the two security group web app as for the webserver which allows http, https and ssh and also created the db server sg for the database which only allow access from the web server. I also launch the two server instance in both public and private subnet but the public subnet include public ips and dns and on the other hand in the database servevr instance in private subnet work without the public ip . And then finally I test the web server in the brower by using public ip and check it is working or not and I do this with the public ip and then connected to the mariadb database via ssh to check the connectivity.

**Preparation**

And in the preparation, first of all I preprared the aws environment by starting the lab and then I review the diagram which anuj created in the task 1 and then I created a checklist which I needed in this part like vpc, subnets, routetable, gateways, security group and ec2 instance. And I also ensure that I understand that everything is connected with each other . So I clearfully configure everthing carefully.

**Screenshot**

A screenshot of a computer

AI-generated content may be incorrect.

This screenshot shows the the WebApp-vpc is successfully created.

A screenshot of a computer

AI-generated content may be incorrect.

This screenshot confirms that four subnets have been created under the "WebApp-VPC-vpc" VPC. Two of them are labeled as public (public1 and public2) and two are private (private1 and private2), distributed across two availability zones (us-east-1a and us-east-1b).

A computer screen with a white background

AI-generated content may be incorrect.

This screenshot shows that I successfully created the internet gateways named WebApp-igw.

A computer screen with a white background

AI-generated content may be incorrect.

The above screenshot shows that the public route table (WebApp-VPC-rtb-public) is successfully created and has been eassigned to two public subnets in the WebApp-VPC-vpc (VPC ID: vpc-080dd61189e062c3). With this association, the subnets can route information via the internet gateway.

A screenshot of a computer

AI-generated content may be incorrect.

In t his screenshot, it is confirmed that two NAT gateways were released in pu blic subnets. The state of both gateways is in the A vailable stage and they are linked to public Elastic IPs. These NAT gateways make the insta nces in the private subnets able to access the inter net either to perform updates or downloads without exposing them to the o utside world.

A screenshot of a computer

AI-generated content may be incorrect.

In this sc rrenshot I created the webappsg security group and set inbound rules like port 22 (SSH), 80 (HTTP), and 443 (HTTPS), which we can see in the screenshot.

A screenshot of a computer

AI-generated content may be incorrect.

In this scrresnhot I created the DBServerssg security group with different SSH (port 22), (port 80), and MySQL/Aurora (port 3306) inbound rules like compare with Webappsg.

A computer screen with a computer screen

AI-generated content may be incorrect.

This screenshot proves that the EC2 in stance was properly launched in a private subnet under the name DBServer. It has no direct a ccess to the inter net and it has no p ublic IPv4 address. The private IP a ddress (10.0.136.125) will be assi gned to the instance and will be use d only to communicate inside the VPC.

A screenshot of a computer

AI-generated content may be incorrect.

The present screenshot proves that the EC2 instance was properly launched in a private subnet under the name DBServer. It has no direct access to the internet and it has no public IPv4 address. The private IP address (10.0.136.125) will be assigned to the instance and will be used only to communicate inside the VPC.

A screenshot of a computer

AI-generated content may be incorrect.

This screen-shot proves that the webserver is up. The webpage is open from browser using public ip 3.228.0.138 and it says “ INFO2350 AWS Project Web Server” that shows that the instance is good and accessible.

A screenshot of a computer

AI-generated content may be incorrect.

This screenshot proves that the connection with MariaDB database hosted on the instance of DBServer worked (private IP: 10.0.136.125). It was connected with MySQL client through SSH on the Web Server instance. The MariaDB prompt is responsive, which means that the database is up and operational.

**Observation**

And by creating this setup I notice that by attaching the internet gateway and associate the public route table is crucial for the web server to work properly over the internet. And I also create the two security group named webapp sg and dbserversg I observe that the first security group webappsg for the web server is allowing the inbound rule and then I obersere and notice that the dbserversg is allowing access only to the webserversg group . And I also deploy ec2 instance in both subnet types. Because I want the web server instance will launch in public subnet in public ip address and dns name, while I set database server it only launch in private subnet without public ip. And which also verify that the web server is accessible from the browser and I I can see that the database is reachable via ssh and from the websever.

**Reflection**

In this task I get good experience with the making of the cloud infrastructure by the help of aws. And I also learn the importance of the routle table , nat gateway setup and subnet configuration. And I see that the route table is used to control how the traffic flows between the public and private subntet and it also ensure the proper communication paths. And the nat gateway I implement because the private instance like database server could access the internet for the updates without being any directy exposed and which also make sure the security. And the subnet configuration help in the separation of the public resource like webeserver from the the private resource like database for the security. And the security group I applied to allow only on the necessary port for the security. And then placing the database in private subnet which ensure that itwill be unaccessible from the internet which will protect our sensitive data. After applying these configuration in my task 2 I was able to create the Aws architecture in which the web server is accessilbly publicly and on the other hand the database remain private but functional. And I also confirm and that my connectivity between web server and database remain secure and I test that the web application is operational via the public ip and I also verify that my infrastructure resources like vpc, subnet, nat gatway, route table and security group are working and connected with each other. This give me the understanding of the aws networking , And secure architecture design and and doing the practical steps I deploy the scure, and functional cloud environment.

**Task-3-Automation(Jatin Mehta)**

**Description:**

In the task, I confirm and tested the deployment on my aws environment . And then I used aws cloud formation for the automation and by the help of the automation. And by this it allow me to create the entire infrastructure by the yaml template and instead of the building each componnents manually. And the yaml file include the vpc , public and private subnets , internet gateway, natgateway , route table , Ec2 webserver , database server and security group. And the cloud formation automatically deploy all the resources same as task and configuration in task 2. And after the deployment I test the setup by connecting to the mariadb database by using mabaxterm and then I verified the web server in the public subnet and then the database in private subnet were work as needed.

**Preparation**

In this task, I preparaed everytihing on the aws envuonment that harsh deployed in the task 2 , I also made sure that the aws resources required for the cloud formation stack ware defined in the yaml file. And then I use the mobaxterm to test ssh connection for the server and use web brower to check the web server accessibility. And then I revewied the cloud formation stack to ensure that all componnets are included like VPC, subnets, NAT Gateway, Internet Gateway, route tables, and EC2 instances before running the deployment.

**Observation**

In this task I observe that the cloud formatiuon is successfully created the entire infrastructure by the one yaml template. And the web server is running propaly with the public subnet with public and private ip, and while the database is unreachable from the public access but will be reachable form the web sevrer. And I observe that the the cloud formation stack is showing all the components .Which proved that automation save the time and reduce the errors compared to the manual setup.

**Screenshot**

A screenshot of a computer

AI-generated content may be incorrect.

This screenshot shows the proper connection with MariaDB server via MobaXterm. The connection was done between the EC2 instance at 10.0.2.85 on the private network to database server at 10.0.2.85 using MySQL client. MariaDB prompt is open, which means that the database is available.

A screenshot of a computer

AI-generated content may be incorrect.

This screenshot confirms that the web server is running and accessible. The message “INFO2350 AWS Project Web Server” is successfully displayed in the browser using the public IP address 3.231.93.9, indicating that connectivity, security group rules, and subnet configurations are functioning correctly.

A computer screen with a white screen

AI-generated content may be incorrect.

This screenshot confirms that the CloudFormation stacks were successfully created. The stack named AUTOTASK3 shows a status of create complete, indicating that the AWS CloudFormation template executed successfully to deploy the web app infrastructure.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

This screenshot confirms that multiple key infrastructure components were successfully created using AWS CloudFormation under the AUTOTASK3 stack. These include the NAT Gateway, Private Route, and Internet Gateway, all showing a status of create complete.. This indicates the stack executed correctly and the network architecture is fully provisioned.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

This screenshot confirms that the PublicSubnet, VPC, and WebInstance resources were successfully created as part of the AUTOTASK3 CloudFormation stack. All resources show a status of creation complete, indicating that the public networking and compute infrastructure have been provisioned correctly.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

This screenshot show the Cloud Vpc is created and and I also successfully configured the public and private sunnet , route table, and internet gatway and nat is working fine and created successfully.

A computer screen with a computer screen

AI-generated content may be incorrect.

In this screenshot shows that the web app instance is successfully running in the public subnet and both public and private ip are working which shows the deployment is succesully working.

A computer screen with a computer screen

AI-generated content may be incorrect.

Here the Db instance is running fine with Private subnet with private ip .

**Reflection**

This task shows me the power of the automation in the cloud infrastructure . And by using the cloudfromation and and yaml file. I can make that I can deploy complex aws environment quickly, and accurately. And it also reduce the human error , and also improves the efficiency and one more thing it also easy the scaling in the future. And I also confirm that my network and security configuration was correct and the public resources was accessible to the public resource which worked as expected and the private resources remain secure. And this experience improve my skill in infrastructure as code and this also gave me the confidence for using the automation as professional cloud deployments.

**Task-4-Security Considerations (Payne Jackson)**

Task-4

Security Considerations for AWS Infrastructure

In designing our AWS environment, we focused on key security best practices. Here are the top three challenges we addressed:

1. Database Protection To prevent attacks, we placed the database in a private subnet with no public IP. Access is restricted, only the web server can connect to it on port 3306. This fully isolates the database from the public internet.

2. Restricted Management Access Instead of leaving SSH open to everyone, we limited it to a specific IP. This prevents brute-force attacks and ensures only trusted users can access the instance.

3. Data Security We secured data in transit and at rest. TLS/SSL was enabled on the web app and encrypted connections were enforced on the database. EBS volumes were also encrypted with AES‑256, meeting compliance standards.

In summary, our architecture ensures minimal exposure, strong access control, and robust data protection.