Securing and Monitoring Resources with AWS

Table of contents

Project overview and objectives
The lab environment and monitoring your budget
AWS service restrictions
Scenario
Solution requirements
Lab project tips
Approach
Phase 1: Securing data in Amazon S3
    Task 1.1: Create a bucket, apply a bucket policy, and test access
    Task 1.2: Enable versioning and object-level logging on a bucket
    Task 1.3: Implement the S3 Inventory feature on a bucket
    Task 1.4: Confirm that versioning works as intended
    Task 1.5: Confirm object-level logging and query the access logs by using Athena
    Cost assessment to secure Amazon S3
Phase 2: Securing VPCs
    Task 2.1: Review LabVPC and its associated resources
    Task 2.2: Create a VPC flow log
    Task 2.3: Access the WebServer instance from the internet and review VPC flow logs in CloudWatch
    Task 2.4: Configure route table and security group settings
    Task 2.5: Secure the WebServerSubnet with a network ACL
    Task 2.6: Review NetworkFirewallVPC and its associated resources
    Diagramming activity
    Task 2.7: Create a network firewall
    Task 2.8: Create route tables
    Task 2.9: Configure logging for the network firewall
    Task 2.10: Configure the firewall policy and test access
    Cost estimate to secure a VPC with a network firewall
Phase 3: Securing AWS resources by using AWS KMS
    Task 3.1: Create a customer managed key and configure key rotation
    Task 3.2: Update the AWS KMS key policy and analyze an IAM policy

Task 3.3: Use AWS KMS to encrypt data in Amazon S3
Task 3.4: Use AWS KMS to encrypt the root volume of an EC2 instance
Task 3.5: Use AWS KMS envelope encryption to encrypt data in place
Task 3.6: Use AWS KMS to encrypt a Secrets Manager secret
Cost assessment for using AWS KMS
Phase 4: Monitoring and logging
Task 4.1: Use CloudTrail to record Amazon S3 API calls
Task 4.2: Use CloudWatch Logs to monitor secure logs
Task 4.3: Create a CloudWatch alarm to send notifications for security incidents
Task 4.4: Configure AWS Config to assess security settings and remediate the configuration of AWS resources
Cost assessment for monitoring and logging

Project overview and objectives

Return to table of contents

In this project, you're challenged to use familiar AWS services, as well as AWS services that might be new to you, to create resources in AWS and to implement security on them. Throughout various AWS Academy courses, you have completed hands-on labs. You have used different AWS services and features to build a variety of solutions.

In many parts of the project, step-by-step guidance is *not* provided. This is intentional. These specific sections of the project are meant to challenge you to practice skills that you have acquired throughout your learning experiences prior to this project. In some cases, you might be challenged to use resources to independently learn new skills.

By the end of this project, you should be able to do the following:

- Secure access to objects in an Amazon Simple Storage Service (Amazon S3) bucket.
- Secure network access to your virtual network.
- Encrypt data at rest by using AWS Key Management Service (AWS KMS) on an Amazon Elastic Block Store (Amazon EBS) volume.
- Manage encryption keys by using AWS KMS.
- Create a monitoring and incident response system by using Amazon CloudWatch and AWS Config.

The lab environment and monitoring your budget

Return to table of contents

This environment is long lived. When the session timer runs to 0:00, the session will end, but any data and resources that you created in the AWS account will be retained. If you later launch a new session (for example, the next day), your work will still be in the lab environment.

At any point before the session timer reaches 0:00, you can choose **Start Lab** again to extend the lab session time.

❗ **Important:** Monitor your lab budget in the lab interface. When you have an active lab session, the latest known remaining budget information displays at the top of this screen. The remaining budget that you see might not reflect your most recent account activity. If you exceed your lab budget, your lab account will be disabled, and all progress and resources will be lost. If you exceed the budget, contact your educator for assistance.

If you ever want to reset the lab environment to the original state, before you started the lab, use the **Reset** option above these instructions. Note that this won't reset your budget. ⚠️ **Warning:** If you reset the environment, you will *permanently delete* everything that you have created or stored in this AWS account.

AWS service restrictions

Return to table of contents

In this lab environment, access to AWS services and service actions are restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

Return to table of contents

In this project, you work as a cloud security specialist at AnyCompany Financial bank.

The company has locations throughout the country. The company provides checking and savings accounts, credit cards, loans, and investment products. All products require the use of personally identifiable information (PII), such as account numbers, contact information, and personal IDs. Your manager, the Director of IT, has tasked you to ensure the security of the company's resources in the AWS Cloud.

Key areas of the AWS infrastructure that need to be secured include S3 buckets, the network that hosts web servers, and keys that are used to encrypt data. Your manager also wants you to monitor and analyze access to these resources so that the bank can improve its security posture as changes and potential security incidents occur.

Your task within this project is to secure these AWS resources based on AWS best practices that are associated with the AWS Well-Architected Framework, the principle of least privilege, and internal company IT standards.

Throughout the project, you will test access to resources as two test users, *Paulo* and *Mary*, who are members of the Account Manager Group. In some cases, Paulo has more privileged access than Mary, to certain resources stored in the AWS account. You will compare his access to Mary's access. By comparing their access, you will be able to verify if specific resources have been secured by using mechanisms beyond AWS Identity and Access Management (IAM) policies.

In the AWS KMS phase of the project, you will use a test user named *Sofia*, who is a member of the Financial Advisor Group.

Solution requirements

Return to table of contents

This project is split into different phases, and each phase is designed to address one or more solution requirements. The solution must meet the following requirements:

- R1 - Design (phase 2)
- R2 - Optimize cost (phase 2)
- R3 - Restrict access (phases 1, 2, and 3)
- R4 - Enforce compliance (phases 3 and 4)

- R5 - Encrypt data (phase 3)
- R6 - Withstand penetration testing (phase 2)
- R7 - Monitor and log user activity (phases 1 and 4)
- R8 - Generate alert and change management notifications (phase 4)

Lab project tips

Return to table of contents

You’re encouraged to be resourceful as you complete this project. For example, reference the AWS Documentation or a search engine if you need an answer to a technical question.

As you work through the project, you will find tips to help you complete specific tasks.

Approach

Return to table of contents

The following table provides an overview of the phases of this project.

PHASE	DETAIL	SOLUTION REQUIREMENT(S)
1	Securing data in Amazon S3 - The solution you build will implement security features including bucket policies, object versioning, inventory logging, and object-level logging.	R3, R7
2	Securing VPCs - The solution you build will implement security features including VPC flow logs, route table configurations, network ACLs, and network firewalls.	R1, R2, R3, R6

PHASE	DETAIL	SOLUTION REQUIREMENT(S)
3	Securing AWS resources by using AWS KMS - The solution you build will implement security features including using a KMS customer managed key to encrypt data stored in S3, to encrypt an EC2 instance EBS volume, to encrypt data using KMS envelope encryption, and to encrypt data stored in Secrets Manager.	R3, R5
4	Monitoring and logging - The solution you build will implement security features available in CloudTrail, CloudWatch, and AWS Config to log and monitor activity in your account.	R4, R7, R8

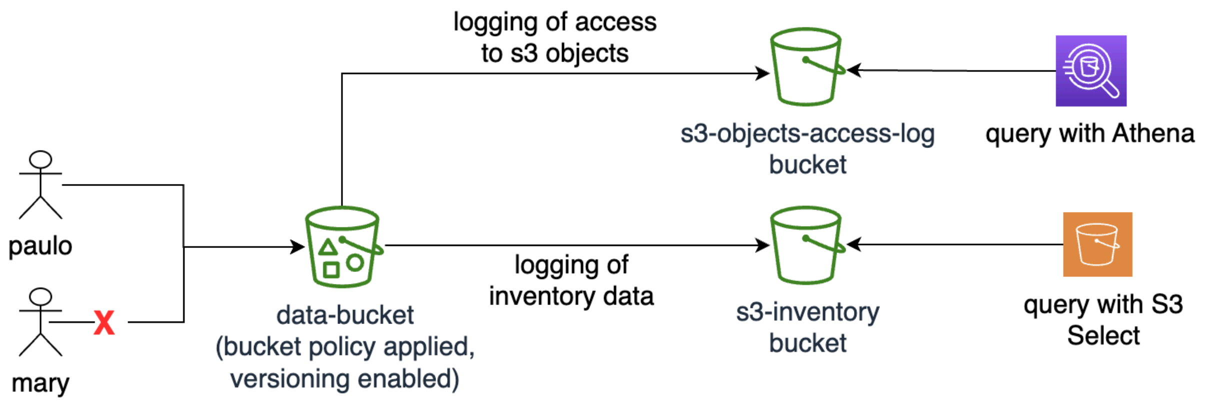
Phase 1: Securing data in Amazon S3

Return to table of contents

In this phase, you are challenged to begin implementing security settings in the AWS account. You have been asked to secure customer PII data that is stored in Amazon S3. The leadership team of AnyCompany Financial has heard about recent data breaches at other companies and wants to protect customer data from unauthorized access. The company wants to do the following:

- Limit access to buckets to certain account managers, who are in the Account Manager group.
- Enable versioning on S3 buckets and all objects in them.
- Enable object logging in all S3 buckets.
- Encrypt all buckets by using server-side encryption with Amazon S3 managed keys (SSE-S3).
- Implement Amazon S3 Inventory to keep a running inventory of all files that are stored in Amazon S3.

By the end of this phase, you will have the architecture that is shown in the following diagram:



Notice that when you are done configuring security on the bucket, the Paulo test user should have access to it, but Mary should not.

Task 1.1: Create a bucket, apply a bucket policy, and test access

Return to table of contents

In this task, you are challenged to create a bucket, apply a bucket policy to it, and then test whether Paulo and Mary can access the bucket.

1. After you connect to the AWS Management Console, notice that you're logged in with the *voclabs* IAM role. You can see this in the top-right corner of the console.

   Tip: For most tasks and steps, you will perform actions with this *voclabs* role. However, in some steps, you will log in as a test user (Mary, Paulo, or Sofia). When you aren't sure which user you are logged in as, verify the name that is displayed in this area of the console.

2. In the Amazon S3 console, notice that a few buckets have already been created for you. Copy the unique ID from the bucket names (the same unique ID is appended to the end of each bucket name).

3. Create a new bucket named `data-bucket-<unique-ID>` in *us-east-1*, where `<unique-ID>` is the value that you copied from the name of the existing bucket. Accept all the default bucket settings.

   Note: When you create the bucket, notice that server-side encryption is always *enabled*. The default is server-side encryption with Amazon S3 managed keys (SSE-S3).

Important: All future references to bucket names in these instructions will use the short name, without the unique ID. For example, `data-bucket-<unique-ID>` will be referred to as `data-bucket`, and you will be expected to add the unique ID to match your actual bucket name.

4. Upload an object to the *data-bucket*.

   For example, create a file named *myfile.txt*, write "hello world" in the file, save it, and then upload it.

5. Author a bucket policy for the *data-bucket* and apply it to the bucket. The bucket policy should include two statements:

   Note: Be careful as one common mistake that can cause significant issues during resource cleanup is applying a blanket **deny-all** policy on an S3 bucket. Once applied, this policy will block access to the bucket and its contents for all users and services. If this happens a new bucket may need to be created.

   - The first statement should do the following:
     - *Allow* all Amazon S3 service actions for *all* principals for *data-bucket* and all objects in it.
     - Include a condition that the ARN equals the ARN for the *voclabs* IAM role, the *paulo* IAM user, or the *sofia* IAM user.

       Tip: In the condition, use *aws:PrincipleArn*.
   - The second statement should do the following:
     - *Deny* all Amazon S3 service actions for *all* principals for *data-bucket* and all objects in it.
     - Include a condition that the ARN *doesn't* equal the ARN for the *voclabs* IAM role, the *paulo* IAM user, or the *sofia* IAM user.

       Tip: To generate the bucket policy, you can use the AWS Policy Generator or the JSON or visual policy editors in the IAM console. An example of how to use the visual policy editor in IAM is available in the IAM lab for the AWS Academy Cloud Architecting course. Alternatively, you could reference the AWS Documentation.

6. Verify the S3 access level of both the paulo and mary user logins. To do this:

    - In an incognito or private browser window, log in to the AWS Management Console as the *paulo* IAM user.

      Tip: You can find the IAM user console sign-in link in the IAM console while you're logged in as the *voclabs* role. To find the password, choose **AWS Details** above these lab instructions.

    - Test the *paulo* user's access to Amazon S3:

        - Verify he can access the *data-bucket* and download objects from it.
        - Verify he can't access the contents of any other bucket in the account.

          Analysis: The *paulo* user should be able to access the *data-bucket* and the objects inside of it. However, the user shouldn't be able to access objects in the other buckets because they don't have sufficient permissions.

          Troubleshooting tip: If the *paulo* user's access is different than described, return to the browser tab where you are logged in as the *voclabs* role and modify the bucket policy settings. Then, return to the browser tab where you're logged in as the *paulo* user and retest. Continue this process until you achieve the desired level of access for the *paulo* user.

    - Log the *paulo* user out of the console. Then use the same incognito or private browser window to test the *mary* user's access to Amazon S3.
    - Test the *Mary* user's access to Amazon S3:

        - Verify she sees that four of the buckets show Access status *Buckets and objects not public*, however the **data-bucket** shows Access status *Error*.
        - Verify that when she accesses any of the buckets, she sees an "Insufficient permissions to list objects" error.

          Questions for thought: Why does the mary user see the **Insufficient permissions to list objects** error? Can you explain why the mary and paulo users are experiencing different levels of access even though they have the same S3 permissions granted to them in the *IAM policy* attached to the AccountManagerGroup IAM group that they are both members of?

- When you are finished testing, log the *mary* user out of the console.

Task 1.2: Enable versioning and object-level logging on a bucket

Return to table of contents

In this task, you are challenged to enable versioning and object-level logging on the *data-bucket*. With versioning enabled, you can track all changes to objects that are stored in the bucket and revert any object to a previous version if needed. Object-level logging creates a detailed audit trail of the objects that are stored in a bucket, which helps you to detect security incidents quickly.

1. Ensure that you are using the browser session where you're logged in to the console as the *voclabs* IAM role.

2. Enable versioning on the *data-bucket*.

   Tip: You might have practiced this in the AWS Academy Cloud Architecting course.

3. Enable server access logging on the *data-bucket*

   Configure the logs to be written to the *s3-objects-access-log* bucket.

   Add `/data-bucket` as a prefix to the end of the **Target bucket** path (it doesn't need to match the actual *data-bucket* name).

4. Verify that the bucket policy for the *s3-objects-access-log* bucket now includes a statement to allow the Amazon S3 logging service to write objects to the bucket.

   Tip: At this point, although you might want to test versioning and object-level logging, we recommend that you wait until you have enabled S3 Inventory in the next task. By doing so, you can test all new configurations at the same time.

Task 1.3: Implement the S3 Inventory feature on a bucket

Return to table of contents

In this task, you are challenged to enable the S3 Inventory feature to monitor changes to objects that are stored in an S3 bucket. S3 Inventory provides a scheduled report of the metadata and object-level changes to your S3 objects and buckets. By using the feature, you can track changes to the stored objects and detect potential security incidents.

1. Enable the S3 Inventory feature on the *data-bucket*.

   Tip: You can do this in the bucket's **Management** tab.

   Use the following settings:

   - Use `Inventory` as the inventory configuration name.
   - Include all object versions.
   - Set the *s3-inventory* bucket as the destination for report details.
   - Set *Apache Parquet* as the output format.
   - Select all six additional metadata *Object* fields.

   Note: It can take up to 48 hours for AWS to deliver the first inventory report.

Task 1.4: Confirm that versioning works as intended

Return to table of contents

In this task, you are challenged to access the AWS account as the *paulo* user and upload an object to the *data-bucket*. Then, you are challenged to confirm that versioning is enabled on the object. You are also challenged to test access as the *mary* user. In the next task, you will analyze the object-level logs to see the actions that you took as different users in this task.

1. On your computer, create a new file named `customers.csv`. Then, copy the following text to the file and save the changes.

   ```
   CustomerID,First Name,Last Name,Join Date,Street
   Address,City,State,Phone

   1,Alejandro,Rosalez,12/12/2013,123 Main St.,Any Town,MD,301-555-0158

   2,Jane,Doe,10/5/2014,456 State St.,Anywhere,WA,360-555-0163
   ```

2. Log in to the AWS account as the *paulo* user and upload the *customers.csv* file to the *data-bucket*.

   After you upload the file, check the server-side encryption settings that are applied to the object. It should be encrypted with Amazon S3 managed keys (SSE-S3).

3. Analyze how many versions of customer.csv exist by navigating to the *customers.csv* details page and choosing the **Versions** tab.

4. On your computer, edit the *customers.csv* file and add more data to it. For example, add the following two rows of data at the bottom of the file.

   ```
   xxxxxxxxxx

   3,John,Stiles,9/20/2016,1980 8th St.,Nowhere,NY,914-555-0122

   4,Li,Juan,6/29/2011,1323 22nd Ave.,Anytown,NY,914-555-0149
   ```

5. In the browser window where you're logged in as the *paulo* user, upload the new version of the *customers.csv* file to the *data-bucket*.

   After you upload the file, notice the versions that are available.

6. From the Amazon S3 console, open the *current version* of the *customers.csv* file to confirm that it contains all four rows of records.

   Also, test opening the prior version to confirm that it contains only two rows of records.

   Note: Be sure to test opening both versions. These actions help generate log data.

7. Log in as the *mary* user and take action in the Amazon S3 console to generate log events.

   - Log the *paulo* user out of the console, and then log in as the *mary* user.
   - Try to access objects in the *data-bucket*.

As you saw earlier, the *mary* user cannot access this bucket or its contents.

- ❗ **Important:** Log the *mary* user out of the console, and close the incognito or private window.

Task 1.5: Confirm object-level logging and query the access logs by using Athena

Return to table of contents

In this task, you are challenged to confirm the S3 object-level logging you enabled earlier is successfully writing log data to S3. You will also use Athena to query these logs.

1. In the browser session where you're logged in to the console as the *voclabs* role, observe the objects that are stored in the *s3-objects-access-log* bucket. Download one of the objects, open it, and review the contents.

   Note: Access log records are delivered on a best effort basis. You might need to wait up to 15 minutes before the log entries appear. For more information, see Logging Requests Using Server Access Logging in the *Amazon S3 User Guide*.

   Analysis: For information about the log format, see Amazon S3 Server Access Log Format in the *Amazon S3 User Guide*. A separate object is created for every logged action, so the logs aren't easy to read through. AnyCompany financial would like you to come up with a solution that makes it easier to read the logs. In the next steps, you will see how to run queries against all the accumulated logs by using Amazon Athena queries.

2. Create an Athena table from the access logs. To do this:

   - Create an S3 bucket named `athena-results-<random-number>` where `<random-number>` is some random number. Accept the default bucket settings.
   - In the Athena console open the query editor.
   - Before running queries, set the athena-results bucket to be the location for query results.

     Note: consult the Amazon Athena documentation for details as needed.
   - Then, in the **Editor** tab paste the following query into the query area:

```
xxxxxxxxxx

CREATE EXTERNAL TABLE `default.bucket_logs`(

  `bucketowner` STRING,

  `bucket_name` STRING,

  `requestdatetime` STRING,

  `remoteip` STRING,

  `requester` STRING,

  `requestid` STRING,

  `operation` STRING,

  `key` STRING,

  `request_uri` STRING,

  `httpstatus` STRING,

  `errorcode` STRING,

  `bytessent` BIGINT,

  `objectsize` BIGINT,

  `totaltime` STRING,

  `turnaroundtime` STRING,

  `referrer` STRING,

  `useragent` STRING,

  `versionid` STRING,

  `hostid` STRING,

  `sigv` STRING,

  `ciphersuite` STRING,

  `authtype` STRING,

  `endpoint` STRING,

  `tlsversion` STRING,

  `accesspointarn` STRING,

  `aclrequired` STRING)

ROW FORMAT SERDE
```

```
  'org.apache.hadoop.hive.serde2.RegexSerDe'

WITH SERDEPROPERTIES (

  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*)
([^ ]*) ([^ ]*) ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*)
([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) (\"[^\"]*\"|-) ([^
]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^
]*) ([^ ]*))?.*$')

STORED AS INPUTFORMAT

  'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT


 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
'

LOCATION

  's3://s3-objects-access-log-<UNIQUE-ID>/'
```

Note: This query is from Querying Access Logs for Requests Using Amazon Athena in the *Amazon S3 User Guide*.

- Before you run the query, replace the `<UNIQUE-ID>` placeholder on the last line with the unique ID from the name of your *s3-objects-access-log* bucket.
- Run the query.

  The **Query results** tab indicates that the query was successful. A table named *bucket_logs* is now available in the **Tables** area on the left.

3. Query the table to discover access details.

   - Preview the contents of the **bucket_logs** table.

     Notice that rows of data from the S3 object-level logs appear. If data doesn't appear, you might need to wait longer before running the query.
   - Run the following query.

```
xxxxxxxxxx

SELECT requester, operation, key, httpstatus

 FROM "default"."bucket_logs"

 WHERE requester LIKE 'arn:aws:iam%';
```

Analysis: The WHERE clause in this query filters the results so that only logs that are related to IAM users are included. Actions that you took as the *voclabs* IAM role are filtered out. Notice that multiple entries were recorded for actions taken as the *paulo* user because that user can access the bucket and its objects. Those actions have an *httpstatus* of 200 because the requests were successful. Actions taken by the *mary* user also appear. However, those actions result in an *httpstatus* of 403 because she doesn't have permissions to access the bucket.

Important: If you don't see any results when you run the query, remember that S3 server access logging records are delivered on a best effort basis. As a result, it might take time for the actions that you took to appear in the logs. Plan accordingly. You could skip ahead and return to this task later. Also, remember that this lab environment is long-lived, so work that you completed will be preserved and available when you return to this lab environment later.

Cost assessment to secure Amazon S3

Return to table of contents

In this phase, you created a bucket policy and applied it to an S3 bucket. You also enabled versioning, service access logging, and S3 Inventory on a bucket. Finally, you used Athenat to analyze access logs.

Question: In this lab, does your use of these Amazon S3 security features result in additional cost in the account? If so, how much do you estimate that it will cost?

Tip: Use the Amazon S3 Pricing page and Amazon Athena Pricing page to help you answer the question.

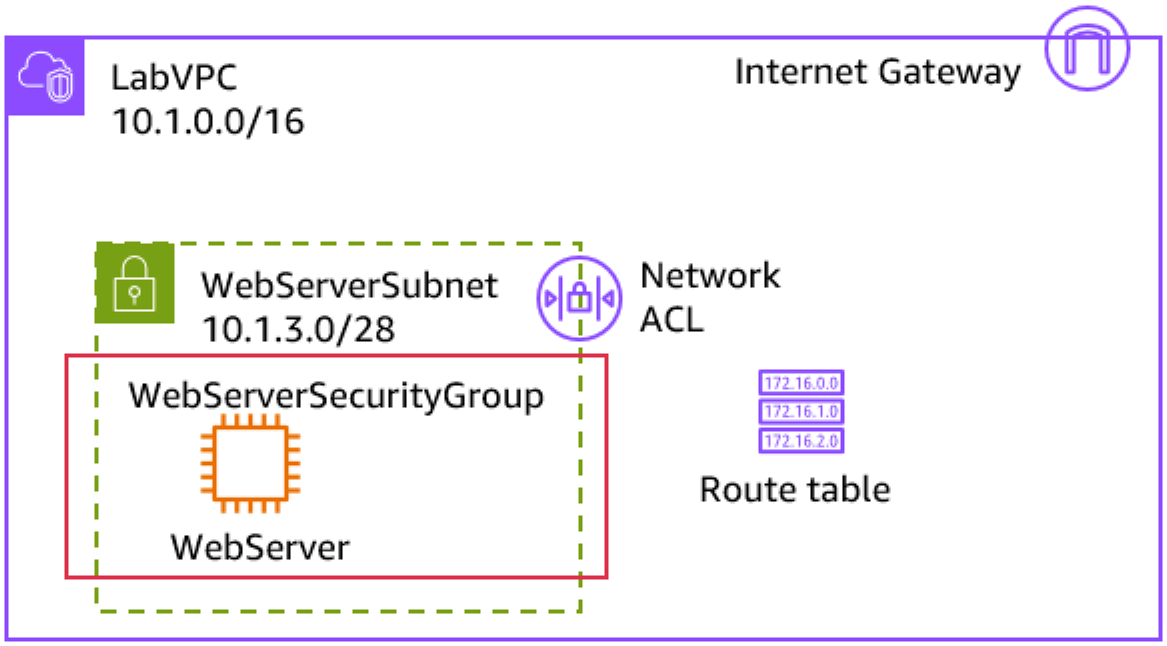
Phase 2: Securing VPCs

Return to table of contents

After you secure the data in Amazon S3, the leadership team for AnyCompany Financial wants you to focus on securing the *network* in the AWS Cloud that hosts the company's critical applications. They are aware of recent network security incidents and want to ensure that their network is protected from unauthorized access and attacks. Your task is to secure the virtual private clouds (VPCs) for the company's web servers.

An inexperienced employee of AnyCompany Financial created the LabVPC and the WebServer instance that pre-exist in your lab project environment. The employee made some mistakes and the result is that the network is not properly configured. In tasks 2.1 through 2.4 you will analyze the existing configurations and make updates to correct the network configuration.

The following diagram shows the resources that already exist in the lab environment and which you will use for the first five tasks in this phase:



Task 2.1: Review LabVPC and its associated resources

Return to table of contents

In this task, you will familiarize yourself with resources that already exist in the lab environment.

1. In the Amazon VPC console, verify that you are observing the *us-east-1* Region, and analyze the resource map of the existing *LabVPC*.

   Notice that it contains the following resources:

   - A subnet named *WebServerSubnet*.
   - A route table. This is the main route table that is created by default for any VPC.
   - An internet gateway.

   Notice that the subnet isn't currently routed to the internet gateway.

2. In the IAM console, locate the *VPCFlowLogsRole* IAM role. Review the permissions that are granted to this role.

   Analysis: You will use this policy in the next task. The policy will allow the VPC Flow Logs feature to write CloudWatch logs to a log group when events occur in the VPC, such as when someone accesses a web server over HTTP.

3. In the Amazon EC2 console, observe the details for the *WebServer* instance.

   Notice the following:

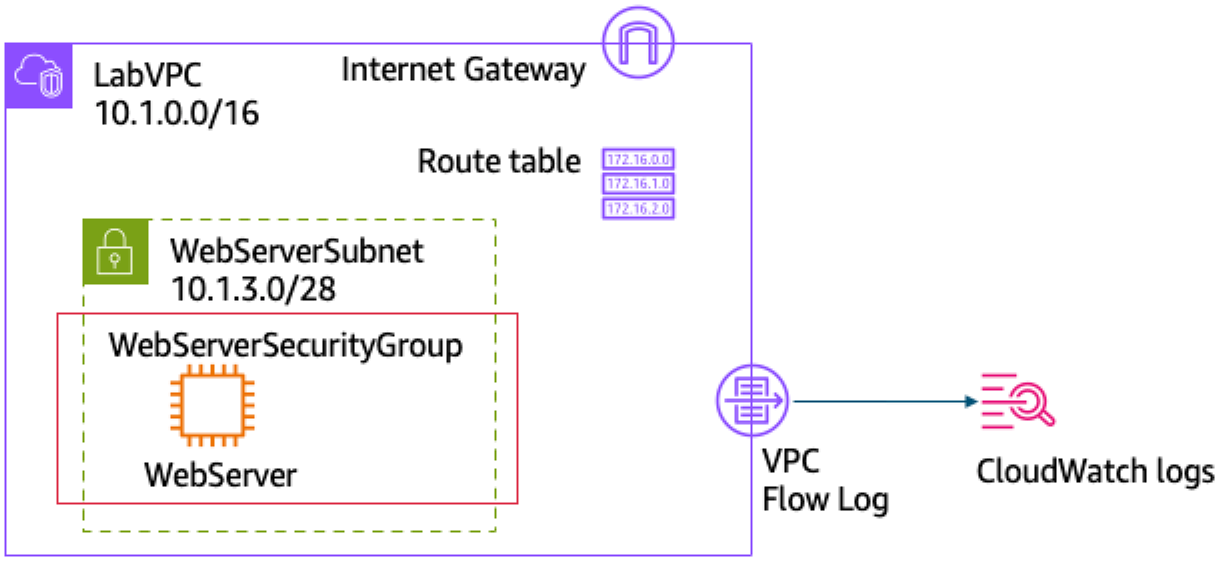
   - A public IPv4 address is assigned.
   - An IAM role is attached.
   - A security group named *Webserver Security Group* is associated.
   - The instance runs in a subnet named *WebServer Subnet*, which is in *LabVPC*.

Task 2.2: Create a VPC flow log

Return to table of contents

In this task, you are challenged to create a VPC flow log for *LabVPC*. The VPC Flow Logs feature can help you understand how inbound and outbound traffic flows through the VPC. The feature also provides monitoring information that will provide insights for how to secure the web server, subnets, and VPC in later tasks.

By the end of this task, you will have configured the architecture that is shown in the following diagram:



1. Create a VPC flow log for *LabVPC*.
    - Name the flow log `LabVPCFlowLogs`.
    - Configure it to capture all types of traffic, and send all log data to CloudWatch Logs with a maximum aggregation interval of **1 minute**.
    - When you create the flow log, define a new destination log group named `LabVPCFlowLogs`. Have it use the IAM role that you observed in the previous task.

Task 2.3: Access the WebServer instance from the internet and review VPC flow logs in CloudWatch

Return to table of contents

In this task, you are challenged to use your web browser to test access to the *WebServer* EC2 instance over port 80 (HTTP). You are also challenged to test access to port 22 (SSH) by using the *netcat* command, which will test whether inbound traffic is allowed. Then, you will be challenged to review the VPC flow log to see how these attempts were recorded.

1. In a new browser tab, try to load the WebServer instance's public IP address at `http://<WebServer-public-IP>`

    The page doesn't load. This is expected. Recall that an inexperienced employee of AnyCompany Financial made some mistakes and the result is that the network is not properly configured.

2. Use the AWS Cloud9 **Cloud9Instance** IDE to test access to the *WebServer* instance.

   Note: Because you are testing access to the *public* IPv4 address, your test is effectively happening *from* the internet.

   - To try to access the instance over the standard HTTP port (80), run the following command, replacing the placeholder with the correct value:

     ```
     xxxxxxxxxx

     nc -vz <WebServer-public-IP> 80
     ```

     After some time passes, the connection fails or times out.

     Tip: To stop the command before the connection fails or times out, press Ctrl+C.

   - Next, try to access the same instance on standard SSH port (22), again using the `nc` command.

     This connection also fails or times out.

3. In the CloudWatch console, look at the log stream for the *LabVPCFlowLogs* log group to verify logs were generated by your actions in the previous step.

   Notice that the *Message* field for all entries indicates "REJECT".

   Tip: The flow log service typically delivers logs to CloudWatch Logs within 5 minutes. However, log delivery is on a best effort basis, so it might take longer for the netcat actions to appear in the log.

4. Filter the log entries to display only the events that originated from the public IP address of the AWS Cloud9 instance that you used to run commands earlier in the task.

   Tip: To discover the public IP address of the AWS Cloud9 instance, run the following command in the AWS Cloud9 terminal:

   ```
   xxxxxxxxxx

   curl http://169.254.169.254/latest/meta-data/public-ipv4
   ```

Analysis: The list of log events is filtered to show the netcat scans that you ran against the *WebServer* instance from the AWS Cloud9 terminal. Each log entry indicates the port number (for example, 80 or 22) that the attempt was made on. For information about the log format, see Flow Log Record Examples in the *Amazon VPC User Guide*.

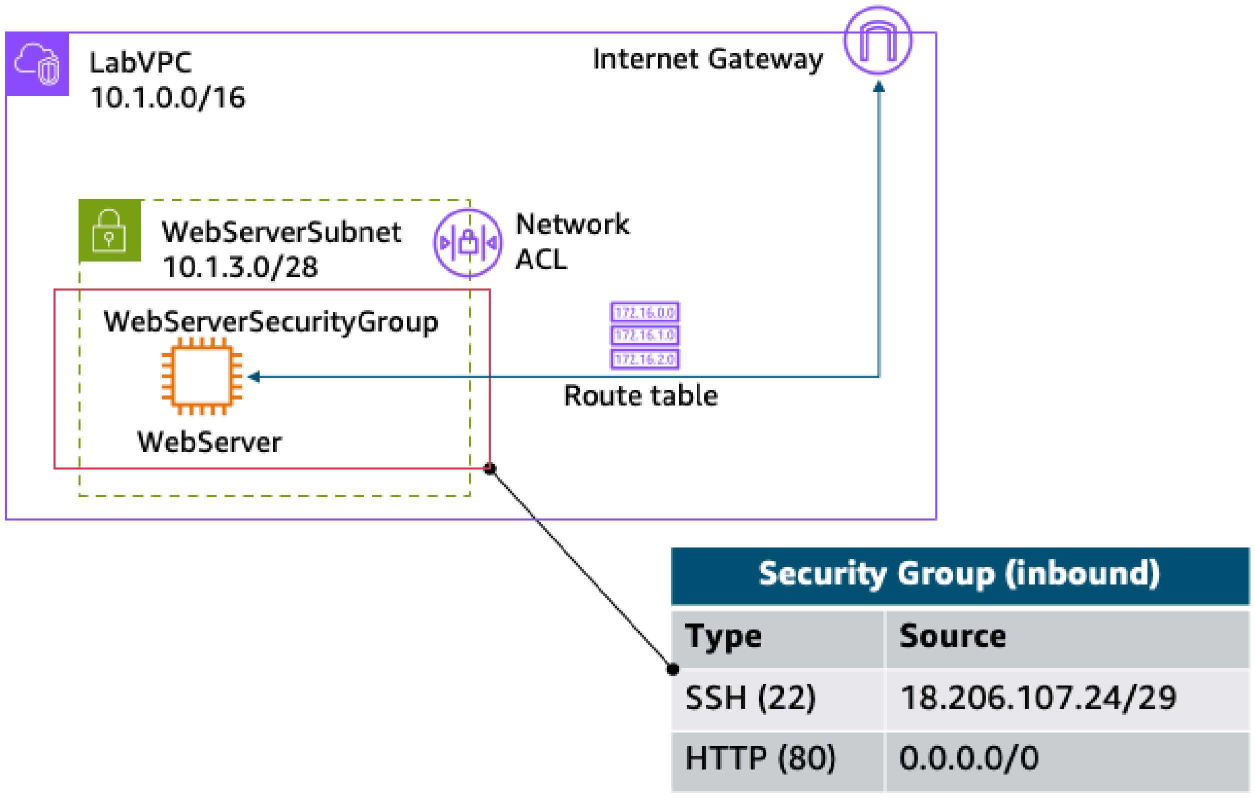
Optional: Filter the log entries by the IP address of your computer. You can find your IP address on a website such as www.whatismyip.com. If you filter by this IP address, you will see entries that are related to your attempt to load the webpage that is hosted on the *WebServer* instance.

Task 2.4: Configure route table and security group settings

Return to table of contents

In this task, you are challenged to create a route for traffic from the internet to access the *WebServerSubnet* through an internet gateway. This will allow inbound HTTP traffic to be directed to the *WebServer* instance. You will also be challenged to modify the security group that is associated with the *WebServer* instance to allow inbound traffic on ports 22 (SSH) and 80 (HTTP).

By the end of this task, you will have configured the architecture that is shown in the following diagram:



Security Group (inbound)	
Type	**Source**
SSH (22)	18.206.107.24/29
HTTP (80)	0.0.0.0/0

1. Modify the route table that is associated with the *WebServerSubnet*. Add a second route to the route table so that traffic to and from the internet (0.0.0.0/0) is routed to the existing *LabVPCIG* internet gateway.

2. Test access from port 80 to the *WebServer* instance again. You can use the `nc` command in the AWS Cloud9 IDE or try to load the following address in a web browser: `http://<WebServer-public-IP>`

   The connection times out or fails.

   Analysis: Although a route now exists from the VPC to the internet gateway, a configuration step is missing to allow inbound internet traffic to the *WebServer* instance. In the next step, you will modify the security group settings to allow this traffic.

3. Modify the inbound rules for the security group that is associated with the *WebServer* instance so that the instance's webpage will be accessible from the internet and so that certain SSH connections to it will be permitted. To do this:

   - Delete any inbound rules that already exist, unless they allow traffic to ports 22 or 80.

- Add an inbound rule for HTTP traffic. The rule should allow traffic from *anywhere* to *TCP port 80*.
- Add an inbound rule to allow SSH access to the *WebServer* instance from the AWS Cloud9 instance. Set the *source* for the rule to match the public IP address of the AWS Cloud9 instance. (You retrieved this IP address in an earlier task.)

  Important: Append `/32` to the public IP address of the AWS Cloud9 instance. `/32` sets the CIDR range to exactly one IP address.

  Tip: A security best practice is to *not* allow access from anywhere through port 22. You should only allow traffic through this port from sources that you plan to use.
- Add an inbound rule to allow SSH access to the *WebServer* instance so that you can use EC2 Instance Connect to connect to the instance.

  To discover the IP address range to use as the *source* for this rule, run the following commands in your AWS Cloud9 IDE:

  ```
  xxxxxxxxxx
  
  sudo yum install -y jq
  
  curl https://ip-ranges.amazonaws.com/ip-ranges.json| jq -r
  '.prefixes[] | select(.region=="us-east-1") |
  select(.service=="EC2_INSTANCE_CONNECT") | .ip_prefix'
  ```

  Note: AWS publishes a list of AWS IP address ranges. The first command that you ran from the previous code block installed the jq utility on your AWS Cloud9 instance. This utility is used to transform JSON data into a more readable format and print it to standard output. The second command read in the *ip-ranges.json* file and found the IP address ranges that EC2 Instance Connect uses in the us-east-1 Region, where the *WebServer* instance is running.

  You should now have three inbound rules defined for this security group.

4. Use the `nc` command to test access from port 22 to the *WebServer* instance again.

   The test should be successful.

5. Use your web browser to test access from port 80 to the webpage on the *WebServer* instance.

   The test should be successful. The message "Hello world from WebServer!" displays.

6. Use EC2 Instance Connect to access the *WebServer* instance.

   After you connect, run the following command: `ping -c 3 www.amazon.com`

   This command tests whether the instance can access the internet. (In this case, you test whether the instance can access the public Amazon.com website.). If you receive 64 bytes of data in return *three* times, then the test was successful, and the *WebServer* instance can access the internet.

   Congratulations! You were successfully able to troubleshoot network connections in this task.

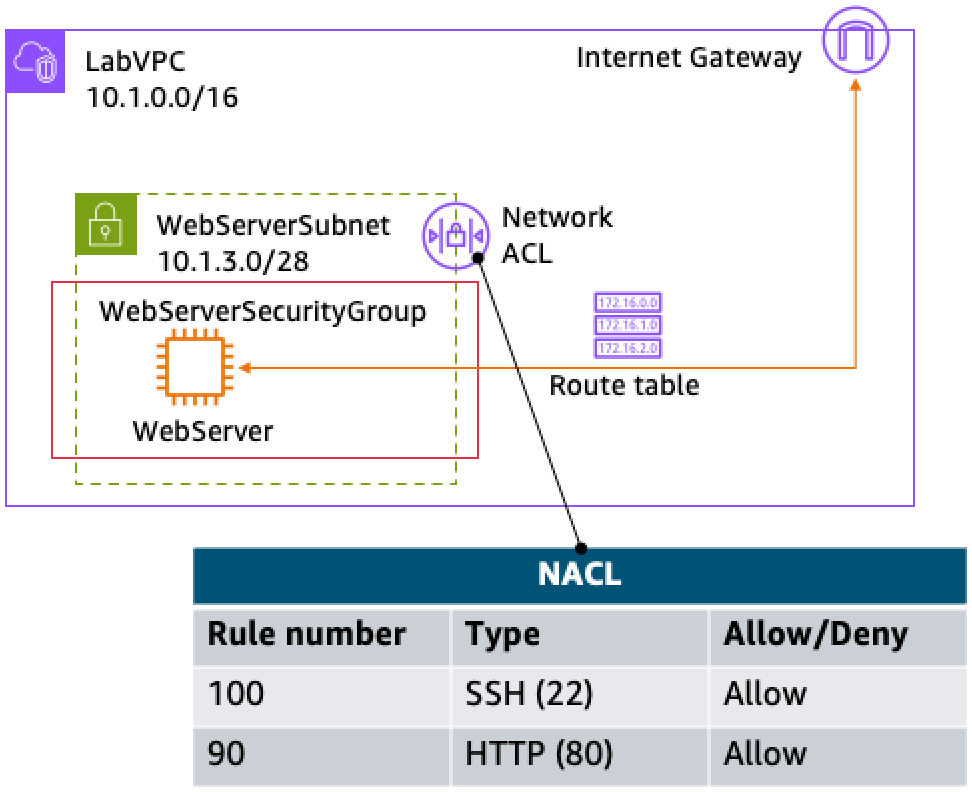
   What were the essential configurations you made that enabled internet access from the instance but also secured it? List all the actions you have performed so far to make the web server instance more secure.

Task 2.5: Secure the WebServerSubnet with a network ACL

Return to table of contents

In this task, you are challenged to configure a network access control list (ACL) to secure the subnet where the web server is running. The network ACL will provide an additional layer of security beyond the security group that you already configured.

By the end of this task, you will have configured the architecture that is shown in the following diagram:



NACL

Rule number	Type	Allow/Deny
100	SSH (22)	Allow
90	HTTP (80)	Allow

1. In the Amazon VPC console, navigate to the details page for the network ACL that is associated with the *WebServerSubnet* in *LabVPC*.

   ❗ **Important**: The existing inbound rule with number 100 indicates that inbound network traffic is allowed to the WebServerSubnet associated with this network ACL. This is the default setting.

2. To test whether you can access port 22 on the *WebServer* instance, modify the *100* rule from *Allow* to *Deny*. Then, run the `nc` command.

   The connection fails or times out, which indicates that access is no longer allowed. Recall that this test *was successful* in step 4 of the previous task.

   Analysis: Network ACL rules can block network access even when security group inbound rules allow access. Security groups can block network traffic at the instance level, however Network ACLs can block traffic at the subnet level and thus offer an additional layer of security.

3. Test whether you can access the webpage at `http://<WebServer-public-IP>` where `<WebServer-public-IP>` is the public IPv4 address of the *WebServer* instance.

It should fail for the same reason.

4. Modify rule number 100 to *allow* connections on port 22 only. Then, use netcat to confirm whether you can access port 22.

   The test should be successful.

5. Edit the inbound rules for the network ACL again. Add a *new* rule that allows *HTTP* traffic from *anywhere*. Use *90* as the rule number.

   Then, confirm whether HTTP traffic is allowed by trying to access the webpage at `http://<WebServer-public-IP>` where `<WebServer-public-IP>` is the public IPv4 address of the *WebServer* instance.

   You should see the "Hello world from WebServer!" message.

Task 2.6: Review NetworkFirewallVPC and its associated resources

Return to table of contents

In this phase so far, you worked to secure *LabVPC* by updating a route table, network ACL, and security group.

In the remaining tasks in this phase, you are challenged to work to secure a different VPC, named *NetworkFirewallVPC*. You will be challenged to secure the VPC by using AWS Network Firewall. AWS Network Firewall features provide you another tool for securing your network. In this project you will be challenged to use it to achieve a result similar to how you secured *LabVPC* using security groups and NACLs.

⚠ **Warning:** The AWS Network Firewall can incur a substantial cost. It's recommended to complete **Tasks 2.6-2.10** promptly.

1. Observe the existing NetworkFirewallVPC resources and configurations.
   - In the VPC console select **NetworkFirewallVPC**, and choose the **Resource map** tab. Notice that the VPC contains the following:
     - A *WebServer2Subnet* subnet. (*LabVPC* contained a subnet named *WebServerSubnet*.)

    - A second subnet, named *FirewallSubnet*. (*LabVPC* had only one subnet).
    - An internet gateway, named *NetworkFirewallIG*. The default (main) route table is already configured to route traffic between *WebServer2Subnet* and the internet.
- Still in the Amazon VPC console, observe the network ACL settings for *NetworkFirewallVPC*.

  The default inbound rules exist. Rule 100 allows all traffic from all sources.

- In the Amazon EC2 console, observe the settings for the *WebServer2* instance and copy the public IPv4 address to use in the next step.

  Observe that this instance runs in *WebServer2Subnet*.

  Notice that inbound traffic is allowed from anywhere (0.0.0.0/0) to ports 80 (HTTP), 22 (SSH), and 8080.

2. Confirm access to the WebServer2 instance on ports 80 and 22.
    - Verify HTTP access using a browser.

      You should see the "Hello world from WebServer2!" message.

    - Verify SSH access by using the `nc` command in the AWS Cloud9 terminal.

      The attempt should be successful.

3. Start an additional website that runs on *WebServer2* port 8080 and test access.
    - Use EC2 Instance Connect to connect to the *WebServer2* instance.
    - After you connect, run the following command.

      ```
      xxxxxxxxxx

      python3 -m http.server 8080 &
      ```

    - After running the command, press Enter to return to the prompt.

      Keep the EC2 Instance Connect session open.

- In a separate browser tab, try to load `http://<WebServer2-public-IP>:8080`

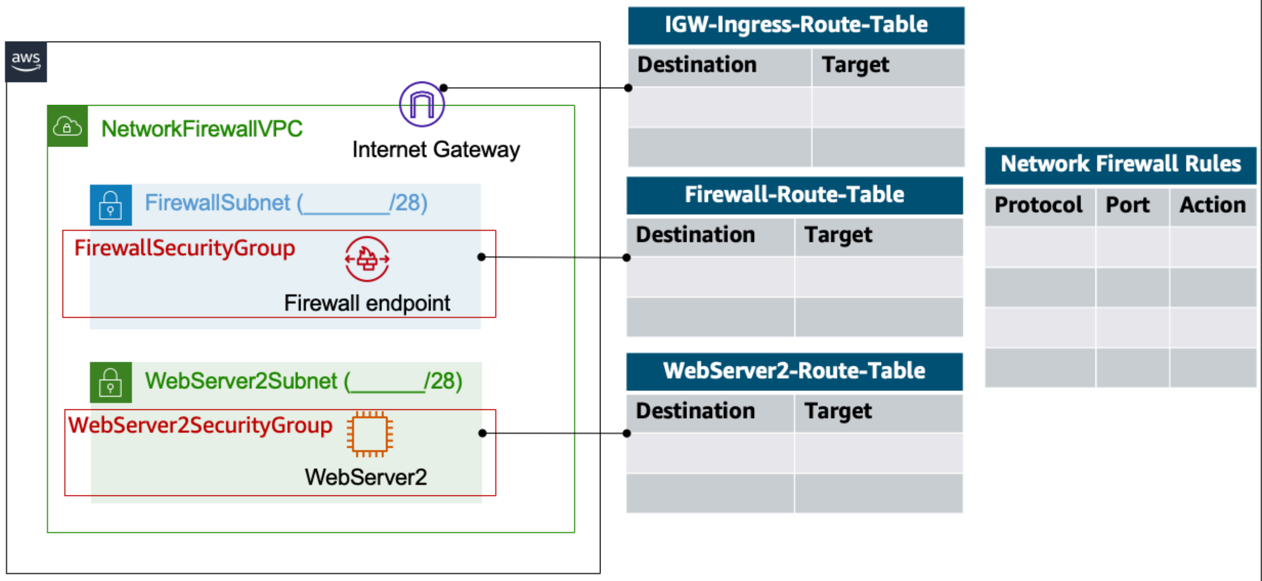
  The attempt should be successful, and you should see the "Hello world from WebServer2 port 8080!" message.

Diagramming activity

Return to table of contents

The following diagram is missing detail, but it shows what you will build in the remaining tasks in this phase.

- In task 2.7, you will create the firewall endpoint.
- In task 2.8, you create all the route tables that are needed to route traffic to and from the internet through the firewall endpoint.
- After you configure network firewall logging in task 2.9, you will configure network firewall rules in task 2.10.



Your challenge is to fill in the missing configuration details (including destinations, targets, protocols, ports, actions, and CIDR blocks for the subnets) in the diagram. Be sure to fill in all four tables and the missing CIDR block information for the two subnets. You can download a copy of the diagram to fill in.

Note: requires that you have access to Microsoft PowerPoint.

Be prepared to explain how the network routing configuration works, in terms of traffic from the internet being routed to *WebServer2* and traffic from *WebServer2* being routed to the internet.

Task 2.7: Create a network firewall

Return to table of contents

In this task, you are challenged to create a network firewall for the *NetworkFirewallVPC*, which you haven't yet used in this project.

1. In the Amazon VPC console, create a firewall named `NetworkFirewall`.
    - Associate it with *NetworkFirewallVPC*. Create the firewall in the *us-east-1a* Availability Zone for *FirewallSubnet* and IP address type *IPv4*.
    - Name the firewall policy `FirewallPolicy`
    - Under **Delete protection** remove the check for **Enable**. Also for **Subnet change protection** remove the check for **Enable**.
    - Wait for the firewall to attain the status of *Ready* before you continue to the next step. The process takes approximately 3 minutes to complete.

      Tip: You might want to refresh the browser tab periodically to know the status more quickly.

Task 2.8: Create route tables

Return to table of contents

In this task, you are challenged to create and configure three new route tables, including one for each subnet in the *NetworkFirewallVPC* and one to handle inbound (ingress) traffic for the internet gateway in *NetworkFirewallVPC*.

1. Create a new route table named `IGW-Ingress-Route-Table` in the *NetworkFirewallVPC*.

2. Edit the route table to add a new route.
    - Set the *destination* to be the CIDR block of the subnet that the *WebServer2* instance runs in.
    - Set the *target* to the only *Gateway Load Balancer Endpoint* that is available.

3. Add an edge association to the *IGW-Ingress-Route-Table* so that the *NetworkFirewallIG* internet gateway is associated with the route table.

4. Create another route table in *NetworkFirewallVPC* for the *FirewallSubnet*.
    - Name the route table `Firewall-Route-Table`
    - Add a route so that `0.0.0.0/0` traffic is routed to the *NetworkFirewallIG*.
    - Create an explicit association between the *FirewallSubnet* subnet and the route table.

5. Create another route table in *NetworkFirewallVPC* for the *Webserver2Subnet*.
    - Name the route table `WebServer2-Route-Table`
    - Add a route so that `0.0.0.0/0` traffic is routed to the *Gateway Load Balancer Endpoint*.
    - Create an explicit association between the *Webserver2Subnet* subnet and the route table.

Task 2.9: Configure logging for the network firewall

Return to table of contents

In this task, you are challenged to configure logging for the network firewall so that you can analyze details of network traffic requests.

1. Create a CloudWatch log group named `NetworkFirewallVPCLogs` with a retention setting of *6 months*.

2. In the settings for the *NetworkFirewall* that you created, browse to the **Firewall details** area, and configure both **Alert** and **Flow** type logging. Set the log destination for both types of logging to use the *NetworkFirewallVPCLogs* CloudWatch log group.

3. Test the logging configuration by attempting to load *WebServer2* instance's website on port 80 in a new browser tab.

   Important: The browser will time out when you attempt to access *WebServer2* at the public IPv4 address. This is expected because the network firewall policy isn't yet configured to allow HTTP traffic on port 80. Recall that you could access the website at the start of task 2.6; however, you then created the network firewall in task 2.7.

4. In the CloudWatch console, look for log events in the *NetworkFirewallVPCLogs* log group. Filter the log events by your public IP address to find the log entries that resulted from the test that you ran in the previous step.

   Note: The timing of network firewall log delivery varies and averages 3–6 minutes. For more information, see Logging Network Traffic from AWS Network Firewall in the *AWS Network Firewall Developer Guide*.

   Tip: If you don't know your public IP address, you can use a site such as https://www.whatismyip.com to discover it.

   Analysis: The log entries from your search should show your public IP address as either the *src_ip* (for the inbound request log entry) or the *dest_ip* (for the outbound response log entry). The inbound request log entry should also show your attempt to load the website at a *dest_port* of port 80 and the outbound response log entry should show a *src_port* of port 80.

Task 2.10: Configure the firewall policy and test access

Return to table of contents

When you first created the network firewall, you defined an empty policy. By default, the empty policy blocks all traffic, which is why your attempt to load the *WebServer2* website in the previous task failed.

In this task, you are challenged to define and add a stateful rule group to the network firewall's policy. With this policy, traffic to and from the internet and the *NetworkFirewallVPC* will be monitored and managed. Access over specific ports will be allowed, and access over other ports will be specifically denied.

1. Navigate to the details page for the *NetworkFirewall* and begin to create the rule group.
    - In the navigation pane of the Amazon VPC console, choose **Firewalls.**
    - Choose the name link for **NetworkFirewall**.
    - In the **Overview** section at the top of the page, under **Step 2**, choose **Add rule groups** > **Create stateful rule group**.

2. Configure the stateful rule group with the name `NetworkFirewallVPCRuleGroup` and a capacity of *100*. Use the other default settings.
    - Configure the rule group to have five rules with the following requirements:
        - First rule: Use the *TCP* protocol. Set the destination port to *8080*, and set the action to *Drop*.

          Important: Note that the *action* for the first rule is different from the other rules.
        - Second rule: Use the *TCP* protocol. Set the destination port to *80*, and set the action to *Pass*.
        - Third rule: Use the *TCP* protocol. Set the destination port to *22*, and set the action to *Pass*.
        - Fourth rule: Use the *TCP* protocol. Set the destination port to *443*, and set the action to *Pass*.
        - Fifth rule: Use the *ICMP* protocol. Set the destination port to *any*, and set the action to *Pass*.

          Important: Note that the *protocol* for the fifth rule is different from the other rules.
    - After you configure the rules, choose **Create and add to policy**.

3. Test the firewall rules.
    - In a new browser tab, load the webpage hosted on WebServer2 using the public IP address.

The attempt should succeed because the firewall policy now contains a rule that allows traffic on TCP port 80.

- In the AWS Cloud9 IDE, use the netcat command to test SSH (port 22) access to *WebServer2*.

  The attempt should succeed.

- Use EC2 Instance Connect to connect to *WebServer2*. The attempt should succeed. After you connect, run the following commands in the session:

  ```
  xxxxxxxxxx

  ping -c 3 www.amazon.com

  sudo netstat -tulpn | grep -i listen
  ```

  Analysis: The results from the ping command confirm that access out to the internet is working and that responses are received over the ICMP protocol, which ping uses. The results of the netstat command show which TCP ports have running processes that are actively listening on them on the instance. For example, the *sshd* process is listening on port 22, and the *httpd* process is listening on port 80. You should also see that python3 is still listening on port 8080.

  Troubleshooting: If you *don't* see a line for port 8080 in the results from the netstat command, you might need to start the web server that runs on that port again by running `python3 -m http.server 8080 &`. To return to the prompt, press Enter. Then, run `sudo netstat -tulpn | grep -i listen` again to confirm that a process is listening on port 8080.

- In a new browser tab, load the webpage at `http://<WebServer2-public-IP>:8080`

  Analysis: The attempt should *fail* because of the *deny* rule that you authored in the network firewall policy. A web server is still running on port 8080, which you confirmed by using the netstat command. However, you cannot reach the web server on that port because of the deny rule in the firewall policy.

To prove this, you *could* temporarily change the firewall rule for port 8080 from *deny* to *pass* and try to load `http://<WebServer2-public-ip>:8080`. If you change the firewall rule, be sure to change it back after testing.

4. In the CloudWatch console, observe the network firewall log entries that were created by your tests in the previous step.

   Congratulations! You successfully implemented AWS Network Firewall to secure a VPC!

   Note: Make sure to remove all resources associated with this section.

   - Remove edge association on IGW-Ingress-RouteTable
   - Delete IGW-Ingress-RouteTable
   - Remove subnet association on Firewall-Route-Table
   - Delete Firewall-Route-Table
   - Remove subnet association on WebServer2-Route-Table
   - Delete WebServer2-Route-Table
   - Remove logging config from the NetworkFirewall
   - Delete NetworkFirewall
   - Delete firewall policy
   - Delete firewall rule group
   - Delete CloudWatch logs (optional)

Cost estimate to secure a VPC with a network firewall

Return to table of contents

AnyCompany Financial has asked you to provide an estimate of the cost to implement a network firewall in the us-east-1 Region. The company has provided the following information about the planned usage.

CRITERIA	ESTIMATE PER MONTH
EC2 - WebServer2	1 On-demand (Shared Instance, 100% usage)
VPC - NetworkFirewallVPC	1 IPAM
Inbound data transfer from (internet free)	100 GB

CRITERIA	ESTIMATE PER MONTH
Outbound data transfer to (US East, N. Virginia) - outbound not Intra-Region	1 TB
Network firewall - number of endpoints	1
Network firewall - usage per endpoint	30 days
Network firewall - data processed per month	2.0 TB

Use the AWS Pricing Calculator and the information from the previous table to create a detailed pricing estimate for the network firewall.

After you create the estimate, export it:

- Choose **Export**, and then choose to export as a .csv or .pdf file.
- When prompted, choose **OK**, and download the file.
- If required by your educator, add information about the cost estimate to your presentation.

Phase 3: Securing AWS resources by using AWS KMS

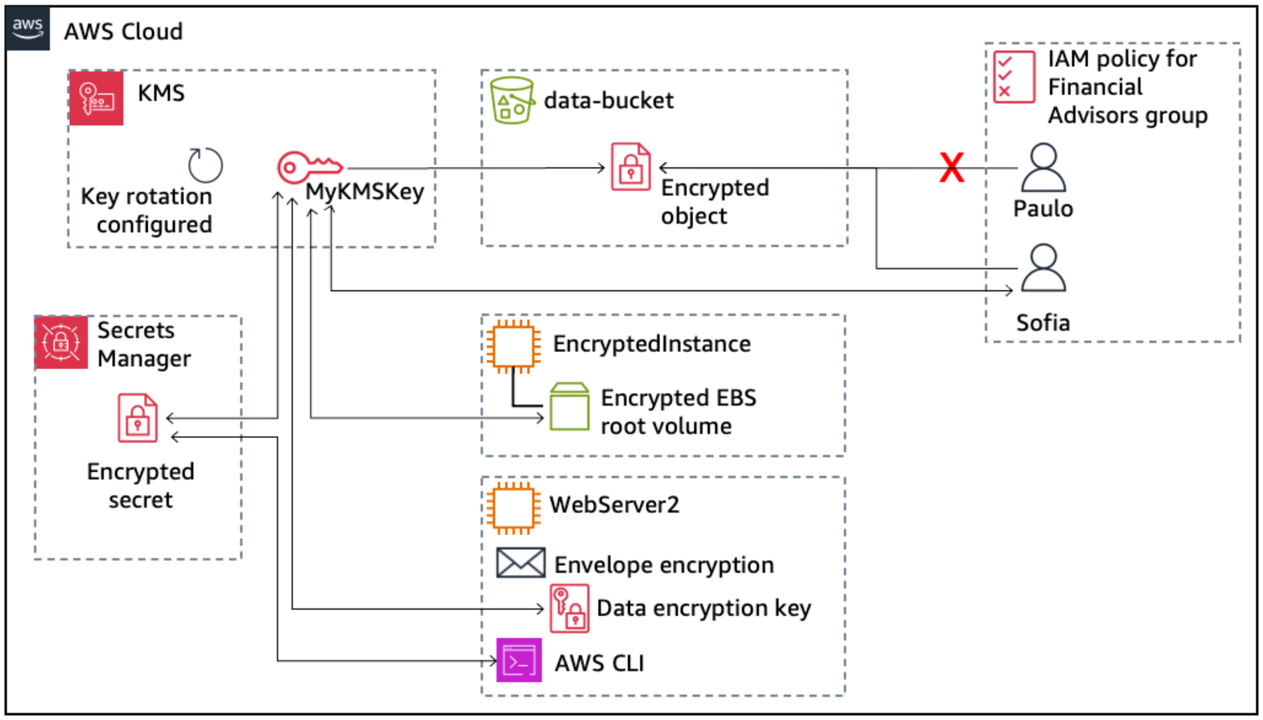
Return to table of contents

The legal department at AnyCompany Financial received notice from the U.S. Federal Deposit Insurance Corporation (FDIC) that the company needs to encrypt sensitive information, such as PII, credit card numbers, and social security numbers. The legal department contacted your manager, the director of IT, and they tasked you to implement encryption and tokenization to meet regulatory compliance standards.

Specifically, the company wants to do the following:

- Implement a mechanism to create and manage AWS KMS customer managed keys.
- Use the managed key solution to encrypt data that is stored in Amazon S3 and on the EBS root volume of an EC2 instance.
- Use the customer managed keys to encrypt secrets that are stored in Secrets Manager.

By the end of this phase, you will have created the architecture that is shown in the following diagram.



Task 3.1: Create a customer managed key and configure key rotation

Return to table of contents

In this task, you will create an AWS KMS customer managed key. You will then configure automatic key rotation on the key. In later tasks, you will be challenged to use this key to protect data that is stored in your account.

1. Create an AWS KMS customer managed key with the alias (name) of `MyKMSKey`. Keep the default settings *except* grant the *Key administrator* and *Key user* permissions to the *voclabs* role.

2. Configure AWS KMS key rotation on the new key so that it is automatically rotated every year.

Task 3.2: Update the AWS KMS key policy and analyze an IAM policy

Return to table of contents

In this task, you are challenged to modify the policy of the AWS KMS key that you created so that the *sofia* user will be authorized to use the key. You are also challenged to analyze the IAM policy that controls what the *sofia* user can do in the AWS account.

1. Modify the key policy for *MyKMSKey* so that it allows the *sofia* IAM user to use the key.

   Tip: The existing key policy contains a statement with a statement ID (SID) value of *Allow use of the key*. Add an additional principal to this statement so that the principal allows both the *voclabs* IAM role and the *sofia* IAM user to use the key. After you modify the key policy, the code *inside* the `Principal {},` element should match the following (where ACCOUNT-NUMBER is the actual account number):

   ```
   xxxxxxxxxx
   
   "AWS": [
   
     "arn:aws:iam::ACCOUNT-NUMBER:role/voclabs",
   
     "arn:aws:iam::ACCOUNT-NUMBER:user/sofia"
   
   ]
   ```

2. Analyze the *PolicyForFinancialAdvisors* IAM policy that exists in your account.

   Analysis: The policy allows full control of all S3 buckets within the account and grants the ability to encrypt and decrypt objects. This policy is attached to the *FinancialAdvisorGroup* IAM group, which the *sofia* user is a member of.

Task 3.3: Use AWS KMS to encrypt data in Amazon S3

Return to table of contents

In this task, you are challenged to use the AWS KMS key that you created to encrypt an object in the *data-bucket* S3 bucket. You will then test access to the object.

1. Modify the encryption settings on the *data-bucket* S3 bucket that you created in phase 1 so that the bucket uses *SSE-KMS* encryption.

2. On your computer, create a new file named `loan-data.csv` that you can use to test encryption. After creating the file in a text editor, paste the following data into the file and save the changes.

   ```
   xxxxxxxxxx

   date,description,amount,principal,interest

   2023-01-14,payment,1000.00,845.52,154.48

   2022-12-22,payment,1021.52,742.80,278.72

   2022-11-15,payment,1000.00,855.27,144.73
   ```

3. In an incognito or private browser window, log in to the AWS Management Console as the *sofia* IAM user.

   Tip: You can find the IAM user console sign-in link in the IAM console while you're logged in as the *voclabs* role.

   - To find the password, choose **AWS Details** above these lab instructions.
   - After you log in, upload the **loan-data.csv** file to the *data-bucket*.

     The upload should be successful.

     Analysis: The upload was successful because sofia has permissions to upload objects, which was granted through the bucket policy that is applied to the *data-bucket*. She also has permissions through an IAM policy to use AWS KMS keys. Finally, the specific AWS KMS key that the *data-bucket* uses to encrypt data that is stored in it has a policy attached that allows the *sofia* user to use the key. The combination of all three security settings allows the user to complete the action.

4. In the Amazon S3 console, choose the link for the *loan-data.csv* object, and analyze the server-side encryption settings that are applied to it.

The encryption type for the object is SSE-KMS.

5. While logged in as the *sofia* user, try to open or download the *loan-data.csv* file.

   The attempt should succeed.

6. Sign the *sofia* user out of the console, and then sign in as the *paulo* user. Then, try to open or download the *loan-data.csv* object.

   The attempt should fail.

   - When done testing, log out and exit the incognito browser tab.

   Analysis: This demonstrates how AWS KMS encryption provides an additional layer of security beyond the security that S3 bucket policies and IAM policies provide. Sofia has permissions to open the file, but Paulo doesn't. Paulo *does* have permission to access the bucket and download other objects from the bucket, granted to him in the *S3 bucket policy and* in the *IAM policy* attached to the IAM group of which he is a member. However, the IAM policy doesn't grant him the AWS KMS service access that Sofia has. In task 3.2, you added Sofia to the *KMS key policy* used by the data-bucket, but you didn't add Paulo to it. Because of these combined security settings paulo can't view the encrypted file.

   Question: Can you imagine a scenario where the security policies you applied would be useful to AnyCompany Financial?

Task 3.4: Use AWS KMS to encrypt the root volume of an EC2 instance

Return to table of contents

In this task, you are challenged to use the AWS KMS key again, but now you will use it to encrypt the root volume of a new EC2 instance.

1. While logged in with the *voclabs* role, create a new EC2 instance. Keep the default settings except for the following:

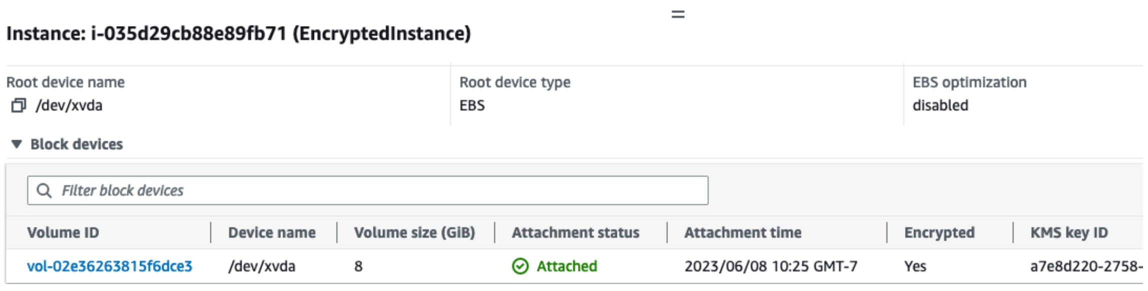
   - Name the instance `EncryptedInstance`

- Choose *Amazon Linux 2 AMI* as the AMI and *t2.micro* as the instance type.

  Important: You must choose Amazon Linux 2 (not Amazon Linux 2023) as the AMI type.
- For key pair, choose *vockey*.
- Edit the network settings and deploy it to the *NetworkFirewallVPC* in *WebServerSubnet2*. Use the existing *WebServer2SecurityGroup*.
- In the *Configure storage* settings, choose *Advanced* and choose to *encrypt* the *AMI root* volume by using *MyKMSKey*.
- In the *Advanced details*, assign *WebServerInstanceProfile* as the IAM instance profile.

2. On the details page for *EncryptedInstance*, choose the **Storage** tab and verify that the instance root volume is encrypted.

   The tab should look similar to the following image:



Instance: i-035d29cb88e89fb71 (EncryptedInstance)

Root device name	Root device type	EBS optimization
/dev/xvda	EBS	disabled

Block devices

Filter block devices

Volume ID	Device name	Volume size (GiB)	Attachment status	Attachment time	Encrypted	KMS key ID
vol-02e36263815f6dce3	/dev/xvda	8	Attached	2023/06/08 10:25 GMT-7	Yes	a7e8d220-2758-

Task 3.5: Use AWS KMS envelope encryption to encrypt data in place

Return to table of contents

In this task, you are challenged to use the AWS Command Line Interface (AWS CLI) to encrypt data in place by using the AWS KMS key. You are also challenged to see how to decrypt the encrypted data. For convenience, you will use the *WebServer2* instance from phase 2 to complete this task.

1. Use EC2 Instance Connect to connect to *WebServer2*.

2. To create a file that you will later try to encrypt, run the following commands in the EC2 Instance Connect session:

   ```
   xxxxxxxxxx

   echo "Let's encrypt these file contents. Sensitive data here." >
   data_unencrypted.txt

   cat data_unencrypted.txt
   ```

3. Use the AWS CLI to generate a data key from the AWS KMS key.

   Note: Amazon Linux 2023 instances, such as *WebServer2*, have the AWS CLI client software preinstalled. In addition, the instance profile (*WebServerRole* IAM role) that is attached to the *WebServer2* instance grants the AWS KMS permissions that are needed to complete this task. Finally, one of the stateful rules that you defined in the network firewall allows communication over port 443, including responses, which is required for AWS CLI access.

   - In the *WebServer2* EC2 Instance Connect session, run the following command to test access to AWS KMS:

     ```
     xxxxxxxxxx

     aws kms list-keys
     ```

     The command returns a list of AWS KMS keys in the account.

     Tip: If you don't receive a response, verify that you properly configured a rule in *NetworkFirewallVPCRuleGroup* that allows traffic over port 443 (as mentioned in the prior phase), which the AWS CLI needs.

   - Next, run the following commands to generate a *data key* for the MyKMSKey, save it to a bash variable, and then echo the data key details in pretty JSON format:

     ```
     xxxxxxxxxx

     result=$(aws kms generate-data-key --key-id alias/MyKMSKey -
     -key-spec AES_256)

     echo $result | python3 -m json.tool
     ```

Analysis: *CiphertextBlob* is a data key that you just generated. It is encrypted with the customer managed *MyKMSKey*, which you created in task 3.1. You will want to store this data key persistently so that you can decrypt and encrypt files with it. *Plaintext* is the data key in unencrypted format.

4. Save the data key to disk.

- To export the *CiphertextBlob* value from the result and save it to a text file in base64-encoded format, run the following commands:

```
xxxxxxxxxx

dk_cipher=$(echo $result| jq '.CiphertextBlob' | cut -d '"'
-f2)

echo $dk_cipher

echo $dk_cipher | base64 --decode > data_key_ciphertext
```

- To see that the data key is stored in encrypted (not human-readable) format, run the following command:

```
xxxxxxxxxx

cat data_key_ciphertext
```

Tip: You can regenerate the plaintext version of the data key from the base64-encoded ciphertext at anytime, as demonstrated by running the following command:

```
xxxxxxxxxx

aws kms decrypt --ciphertext-blob
fileb://./data_key_ciphertext --query Plaintext --output
text
```

- Run the previous command again but also save the result to a file in base64-encoded format:

```
xxxxxxxxxx

aws kms decrypt --ciphertext-blob
fileb://./data_key_ciphertext --query Plaintext --output
text | base64 --decode > data_key_plaintext_encrypted
```

5. Use the data key to encrypt the file that you created earlier.
    - To encrypt the *data_unencrypted.txt* file, which you created in the previous step, run the following command:

      ```
      xxxxxxxxxx

      openssl enc -aes-256-cbc -salt -pbkdf2 -in
      data_unencrypted.txt -out data_encrypted -pass
      file:data_key_plaintext_encrypted
      ```

    - To try to read the file, run the following command:

      ```
      xxxxxxxxxx

      cat data_encrypted
      ```

      It's not human-readable, unfortunately.
    - To delete the unencrypted version of the file, run the following command:

      ```
      xxxxxxxxxx

      rm data_unencrypted.txt
      ```

      Analysis: So far in this task, you generated a data key from your KMS key and stored it in an encrypted format. You then used the data key to encrypt an unencrypted file (data_unencrypted.txt). Finally, you deleted the unencrypted file. Next, you will see how to decrypt the file that you encrypted.

6. Decrypt the file to prove that the data is retrievable.

- To decrypt the data and save it as a new file named *data_decrypted*, run the following command:

```
xxxxxxxxxx

openssl enc -d -aes-256-cbc -pbkdf2 -in data_encrypted -out
data_decrypted.txt -pass file:./data_key_plaintext_encrypted
```

- To print the results to ensure the data is now readable, run the following command:

```
xxxxxxxxxx

cat data_decrypted.txt
```

  You should see the text from the original *data_unencrypted.txt* file.

  You have now seen how it's possible to use envelope encryption to encrypt data at rest.

Task 3.6: Use AWS KMS to encrypt a Secrets Manager secret

Return to table of contents

In this task, you are challenged to create a key-value pair (a *secret*), which you will encrypt with your AWS KMS key and store in Secrets Manager. You are then challenged to verify that you can retrieve the secret by using the AWS CLI.

1. In the Secrets Manager console, create a new secret of type *Other type of secret*.
   - Give it a key of `secret` with the value `my secret data`.
   - Encrypt it with your *MyKMSKey* and name the secret `mysecret`.

     Tip: Notice that the console provides sample code that you could use to access this secret later by using a programming language.

2. Use EC2 Instance Connect to connect to the *WebServer2* instance, and then use the AWS CLI to retrieve the secret.

   Tip: Start by invoking the `aws secretsmanager list-secrets` command. Then, use the `aws secretsmanager get-secret-value` command to actually retrieve the secret.

   You should see the secret's key-value pair data returned in the results.

   Note: In a real scenario, you would store information that is more sensitive, such as database credentials, in Secrets Manager. If you were running a web application on an EC2 instance, you could implement code in the application to access and decrypt the secret, and then use the decrypted secret to connect to the database. In this way, if your application code was ever comprised, the database credentials would remain secure.

Cost assessment for using AWS KMS

Return to table of contents

In this phase, you created a customer managed key in AWS KMS, configured key rotation, configured an S3 bucket to use the AWS KMS key to encrypt objects, encrypted the EBS root volume of an EC2 instance, used AWS KMS envelope encryption, and used AWS KMS with Secrets Manager.

Question: In this lab, does your use of AWS KMS result in additional cost in the account? If so, what usage would you be charged for?

Tip: Use the AWS Key Management Service Pricing page to help you answer the question.

Phase 4: Monitoring and logging

Return to table of contents

The leadership team at AnyCompany Financial received reports of a new security breach at one of its biggest competitors. The company wants to ensure that it's prepared to detect and respond to any future security incidents. The director of IT has asked you to implement a monitoring and logging solution to detect security incidents so that the company can respond promptly.

The solution needs to do the following:

- Track all API calls to S3 buckets.
- Monitor application logs.
- Notify team members in case of security incidents.
- Monitor AWS resource configurations and automatically modify configurations that are out of compliance.

The following table describes the tasks that you will implement in this phase.

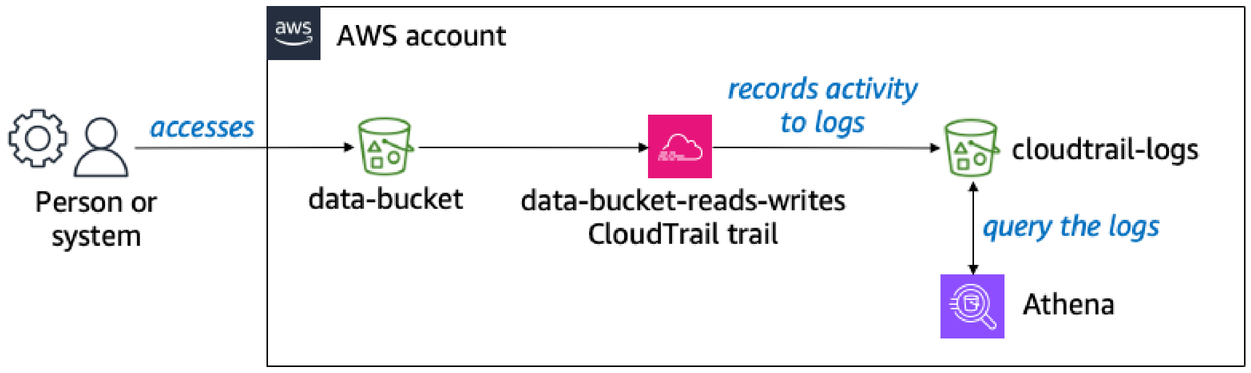
NUMBERED TASK	DETAIL
1	Create a trail in AWS CloudTrail to record Amazon S3 API calls.
2	Configure CloudWatch Logs to monitor the *WebServer* instance's authentication logs.
3	Create a CloudWatch alarm to notify team members when access to the *WebServer* instance is attempted.
4	Use AWS Config to ensure that S3 buckets have object logging enabled when they are created.

Task 4.1: Use CloudTrail to record Amazon S3 API calls

Return to table of contents

In this task, you are challenged to use CloudTrail to record API calls that are made to Amazon S3 buckets. This information will provide an audit trail to track when S3 objects are created, modified, or read.

By the end of this task, you will have configured the architecture that is shown in the following diagram:



1. Create a CloudTrail trail with the following characteristics:

   Important: After you load the CloudTrail console, to open the navigation pane, choose the menu icon (☰). Then, choose **Trails** and then choose **Create trail**. Don't choose the option to create a trail from the main CloudTrail console page. This option takes you to the *Quick trail create* wizard, which won't provide you an opportunity to specify some of the configurations you will want.

   - Name the trail `data-bucket-reads-writes`
   - Store the logs in the existing *cloudtrail-logs* S3 bucket.
   - *Disable* SSM-KMS encryption.
   - Record both management events and data events in the trail.
   - For data events, log all S3 events.

2. On your computer, create a file named `customer-data.csv`. Then, in a text editor, paste the following data into the file and save the changes.

```
xxxxxxxxxx

CustomerID,First Name,Last Name,Join Date,Street
Address,City,State,Phone

1,Alejandro,Rosalez,12/12/2013,123 Main St.,Any Town,MD,301-555-0158

2,Jane,Doe,10/5/2014,456 State St.,Anywhere,WA,360-555-0163

3,John,Stiles,9/20/2016,1980 8th St.,Nowhere,NY,914-555-0122

4,Li,Juan,6/29/2011,1323 22nd Ave.,Anytown,NY,914-555-0149
```

3. Upload the *customer-data.csv* file to *data-bucket*.

After you upload the file, open the file in the Amazon S3 console.

Note: Your actions from this step will create entries in the CloudTrail trail, which will be important for the next steps.

4. Use the CloudTrail console to create an Athena table that describes the format of the data in the *cloudtrail-logs* S3 bucket.
    - Consult the CloudTrail documentation for how to accomplish this.
    - Before creating the table, be sure to configure the **Storage location** to be the **cloudtrail-logs** bucket.

      Analysis: A *CREATE EXTERNAL TABLE* table query is generated in Athena. When you run the query in the next step, the table will be created. The table will define the structure of the data that is being written to the *cloudtrail-logs* S3 bucket so that you can later query it. You will use Athena in this task to analyze CloudTrail logs, in the same way that you used Athena to analyze S3 object-level logging in phase 1.

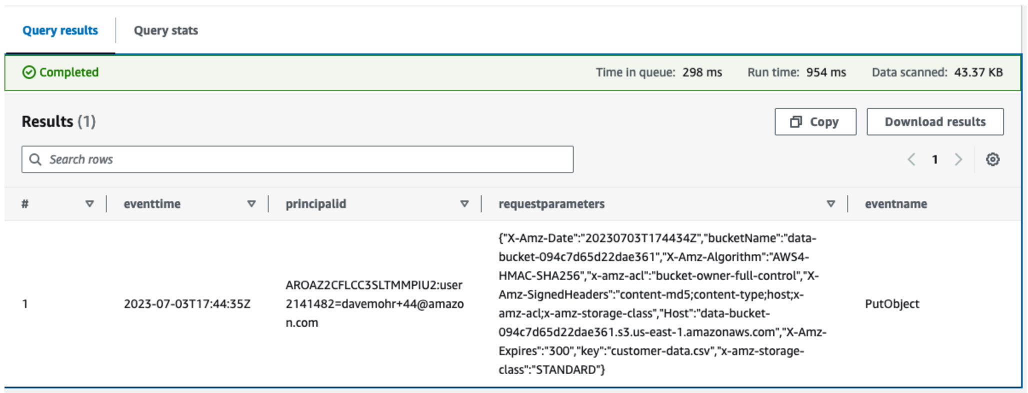
5. Create and run an Athena query to retrieve the CloudTrail event log data for when you uploaded the *customer-data.csv* file to Amazon S3.
    - In the Athena console, *preview* the cloudtrail_logs table that should now exist since you created it in the previous step.

      Verify that 10 rows of data appear in the **Results** area.

      Important: Typically, CloudTrail delivers events within 5 minutes of an API call. You might need to wait a few minutes and then try to preview the table again before it returns results. Verify that data is returned in the preview before continuing to the next step.
    - Run query shown below into a new query tab after replacing `<table name>` with the correct table name:

```
xxxxxxxxxx

SELECT eventtime, useridentity.principalid,
requestparameters, eventname

FROM <table name>

WHERE

    eventname in ('PutObject') AND

    requestparameters LIKE '%customer-data.csv%'

limit 10;
```

The following image provides an example of the results:



#	eventtime	principalid	requestparameters	eventname
1	2023-07-03T17:44:35Z	AROAZ2CFLCC3SLTMMPIU2:user2141482=davemohr+44@amazon.com	{"X-Amz-Date":"20230703T174434Z","bucketName":"data-bucket-094c7d65d22dae361","X-Amz-Algorithm":"AWS4-HMAC-SHA256","x-amz-acl":"bucket-owner-full-control","X-Amz-SignedHeaders":"content-md5;content-type;host;x-amz-acl;x-amz-storage-class","Host":"data-bucket-094c7d65d22dae361.s3.us-east-1.amazonaws.com","X-Amz-Expires":"300","key":"customer-data.csv","x-amz-storage-class":"STANDARD"}	PutObject

Important: You might need to wait a few minutes and then run the query again before it returns evidence of the actions that you took in Amazon S3 earlier in the task.

Analysis: Notice that the user who performed the PutObject action is identified by the **principalid** column in the query results. The date and time that the file was written to Amazon S3 are also included in the results. The **requestparameters** column contains many details, such as the bucket name and the name of the object that was uploaded. A security specialist could use this information to understand more about the event. Such detailed audit information will help AnyCompany Financial with monitoring and compliance.

6. **Challenge:** Create a similar Athena query to retrieve the CloudTrail log information for when you opened (or downloaded) the *customer-data.csv* file. In the results, include the IP address of the principal who retrieved the file and the browser that was used.

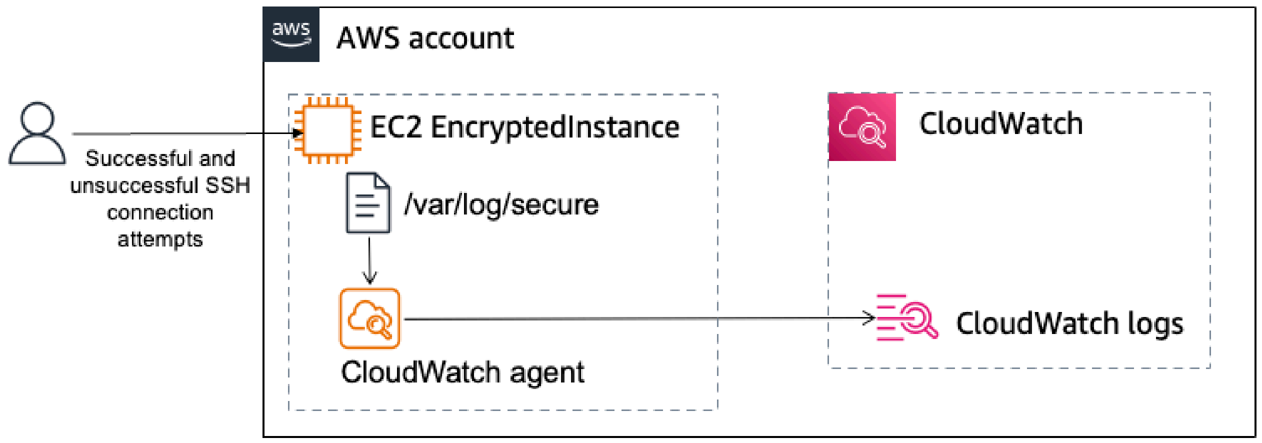
Task 4.2: Use CloudWatch Logs to monitor secure logs

Return to table of contents

AnyCompany Financial has been working with a vendor to design and manage their website. The vendor has requested SSH access to design and manage the website. Company policy doesn't allow direct access through SSH to the production web servers, but access can be provided to a development web server named *EncryptedInstance*.

You have been asked to create a solution to monitor access to *EncryptedInstance*. In this task, you will configure CloudWatch Logs to monitor SSH access to the instance so that your company can understand who accesses the server, where they access it from, when they access it, and what actions they take.

By the end of this task, you will have configured the architecture that is shown in the following diagram.



1. Create a CloudWatch log group named `EncryptedInstanceSecureLogs` with all default settings.

2. Use EC2 Instance Connect to connect to *EncryptedInstance*.

   After you connect, run the following commands to install the CloudWatch agent and a Linux daemon named collectd, which the CloudWatch agent will use:

```
xxxxxxxxxx

sudo yum install -y amazon-cloudwatch-agent

sudo amazon-linux-extras install -y collectd
```

The agent and the collectd software should be successfully installed.

3. Download and configure a JSON file that provides configuration details for the CloudWatch agent.
    - To download the template file, run the following command in the EC2 Instance Connect session:

      ```
      xxxxxxxxxx

      sudo wget https://aws-tc-largeobjects.s3.us-west-
      2.amazonaws.com/CUR-TF-200-ACCAP6-91948/capstone-6-
      security/s3/config.json -P /opt/aws/amazon-cloudwatch-
      agent/bin/
      ```

    - To print out the file template so that you can see what it specifies, run the following command:

      ```
      xxxxxxxxxx

      sudo cat /opt/aws/amazon-cloudwatch-agent/bin/config.json
      ```

      Analysis: The file instructs the CloudWatch agent to collect log entries from the standard Linux *secure* log, which can be found in the /var/log folder on the *EncryptedInstance* instance. Logs that are found there will be forwarded to the *EncryptedInstanceSecureLogs* CloudWatch log group. As entries are collected, they will be written to new log streams.

4. Start the CloudWatch agent, and confirm that it is running.
    - To start the agent, run the following command:

```
xxxxxxxxxx

sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-
agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-
cloudwatch-agent/bin/config.json
```

- To verify the CloudWatch agent status, run the following command:

  ```
  xxxxxxxxxx

  sudo service amazon-cloudwatch-agent status
  ```

  The output should indicate that the agent is *active* and *running*.

- To confirm that the CloudWatch agent is able to reach the CloudWatch service in your AWS account, run the following command:

  ```
  xxxxxxxxxx

  sudo cat /opt/aws/amazon-cloudwatch-agent/logs/amazon-
  cloudwatch-agent.log
  ```

  The output should include a line toward the end that says "[logagent] piping log from EncryptedInstanceSecureLogs/EncryptedInstanceSecureLogs-... (/var/log/secure) to cloudwatchlogs...".

- To actively tail the */var/log/secure* file so that you can monitor the logs that will be created in the next step, run the following command:

  ```
  xxxxxxxxxx

  sudo tail -f /var/log/secure
  ```

- Keep the EC2 Instance Connect tab open, and continue to the next step.

5. Create some security logs by successfully connecting and then failing to connect to the *EncryptedInstance* over SSH from your AWS Cloud9 IDE.

   - Download the PEM file from the **AWS Details** link above these lab instructions.

- Upload the PEM file from your computer to your AWS Cloud9 IDE by using the **File** > **Upload Local Files** option.

  Tip: Before you run the next command, you might want to arrange your browser tabs so that you can see the EC2 Instance Connect session that you still have open from the previous step.

- In the AWS Cloud9 bash terminal, run the following commands. Replace `EncryptedInstance-public-IP` with the public IPv4 address of the *EncryptedInstance*:

  ```
  xxxxxxxxxx

  chmod 400 labsuser.pem

  ssh -i labsuser.pem ec2-user@EncryptedInstance-public-IP
  ```

- When prompted if you are sure that you want to connect, enter `yes`

  The connection should succeed. In the EC2 Instance Connect session, you should see that lines were recorded in the */var/log/secure* file when you ran the previous commands.

- In the AWS Cloud9 bash terminal, run the following commands to disconnect from the SSH session and then try to connect with a username that isn't valid. Replace `EncryptedInstance-public-IP` with the public IPv4 address of the *EncryptedInstance*:

  ```
  xxxxxxxxxx

  exit

  ssh -i labsuser.pem ubuntu@EncryptedInstance-public-IP
  ```

  The connection attempt fails with a "Permission denied" message.

  Analysis: You tried to connect to the instance with the username *ubuntu*, which doesn't exist on this Amazon Linux instance. This results in a failed connection attempt, which is also recorded in */var/log/secure*. (You should be able to see this where you are running the *tail* command in the EC2 Instance Connect session for *EncryptedInstance*.) Recall that the CloudWatch agent should be forwarding the secure file logs to the CloudWatch log group. You will confirm that next.

6. Verify that the secure logs are being written to the CloudWatch log group.
    - Locate the *EncryptedInstanceSecureLogs* CloudWatch log group.
    - Open the latest log stream.

      You should find that the SSH actions that you took have been logged. For example, you should see log events similar to "Accepted publickey for ec2-user from..." and "Invalid user ubuntu from...". Expand the entries to view details.

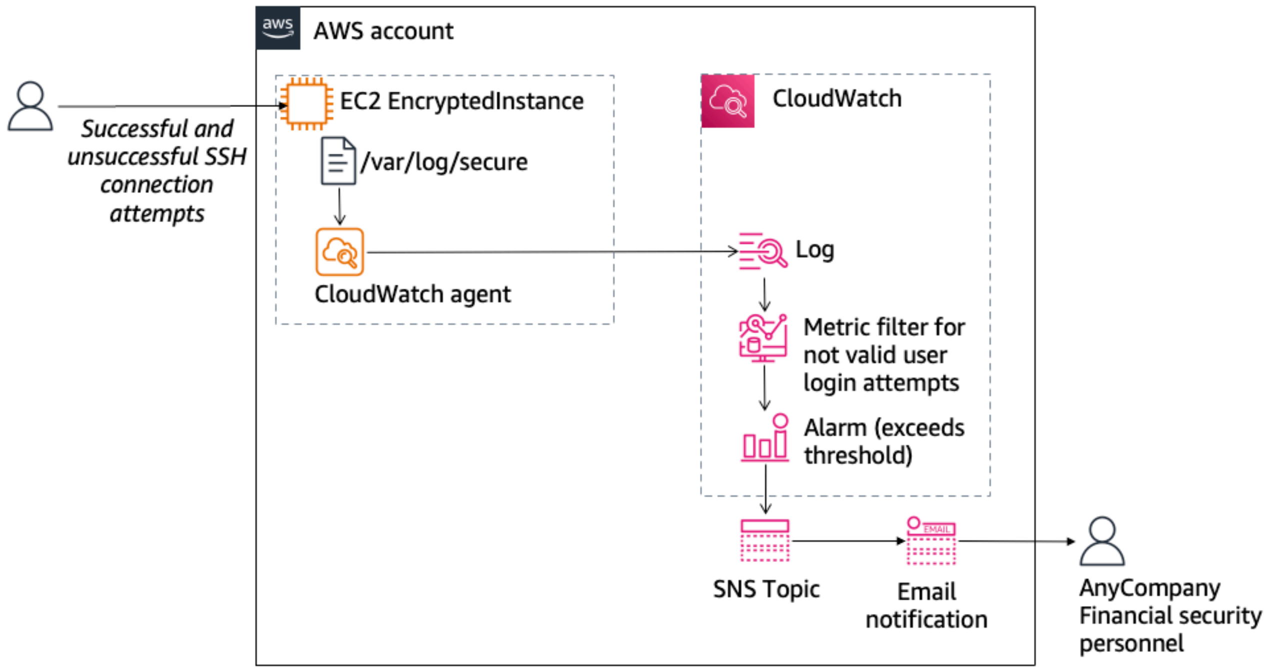
      Note: The result set that you see may also include many log entries of failed log attempts not initiated by you. This is not uncommon.

Task 4.3: Create a CloudWatch alarm to send notifications for security incidents

Return to table of contents

AnyCompany Financial wants security team members in IT to be notified when attempts to access the *EncryptedInstance* through SSH are denied. In this task, you will create a CloudWatch alarm to notify these team members when such an incident occurs.

By the end of this task, you will have configured the architecture that is shown in the following diagram:


AWS account
EC2 EncryptedInstance
/var/log/secure
CloudWatch agent
Successful and unsuccessful SSH connection attempts
CloudWatch
Log
Metric filter for not valid user login attempts
Alarm (exceeds threshold)
SNS Topic
EMAIL
Email notification
AnyCompany Financial security personnel

1. Go to the *EncryptedInstanceSecureLogs* CloudWatch log group, and create a metric filter with the following characteristics:
    - Filter pattern: `"Invalid user"` (Be sure to include the quotation marks.)
    - Filter name: `Not valid users`
    - Metric namespace: `secure`
    - Metric name: `NotValidUsers`
    - Metric value: `1`
    - Default value: `0`
    - Unit: **Count**

2. Create a CloudWatch alarm from the metric filter that you just created.
    - On the details page for the *EncryptedInstanceSecureLogs* log group, on the **Metric filters** tab, select the check box for the **Not valid users** metric filter.
    - Create a CloudWatch alarm with the following characteristics:
        - Period: *1 day*
        - Condition: Whenever NotValidUsers is *greater than or equal to 5*.
        - Notification: Create a new Amazon Simple Notification Service (Amazon SNS) topic named `Not_valid_users_exceeding_limit` and have notifications emailed to you. (Use an email address that you have access to.)
    - Name the alarm `Not valid users exceeding limit on EncryptedInstance` with the following description: `Not valid access attempts over SSH to the EncryptedInstance server have exceeded 4 in the last 24 hours.`

      Important: When you first create the alarm state will show *Insufficient data*, and the **Actions** column will show a warning. The warning appears because the alarm sends a message to an Amazon Simple Notification Service (Amazon SNS) topic with an endpoint (the email address) that hasn't yet been confirmed. The alarm won't work as expected until the endpoint is confirmed.

3. Go to the inbox for the email address that you entered in the previous step. You should see an email from AWS Notifications. In the email, choose the **Confirm subscription** link.

4. To test the alarm, return to the AWS Cloud9 IDE. In the bash terminal, run at least five invalid SSH access attempts to connect to the *EncryptedInstance* public IP address over SSH.

5. In the CloudWatch console, in the latest log stream for the *EncryptedInstanceSecureLogs* log group, filter the log events for `Invalid user`.

   Verify that at least five events are returned and that they all occurred within the past 24 hours. This indicates that the alarm threshold has been met.

6. In the Alarms area of the CloudWatch console, confirm that the *Invalid users* alarm has a state of *In alarm*.

   Check your email to confirm that you received a notification that the alarm threshold was crossed.

   Note: It could take a couple minutes before the alarm state changes and before you receive the email.

Task 4.4: Configure AWS Config to assess security settings and remediate the configuration of AWS resources

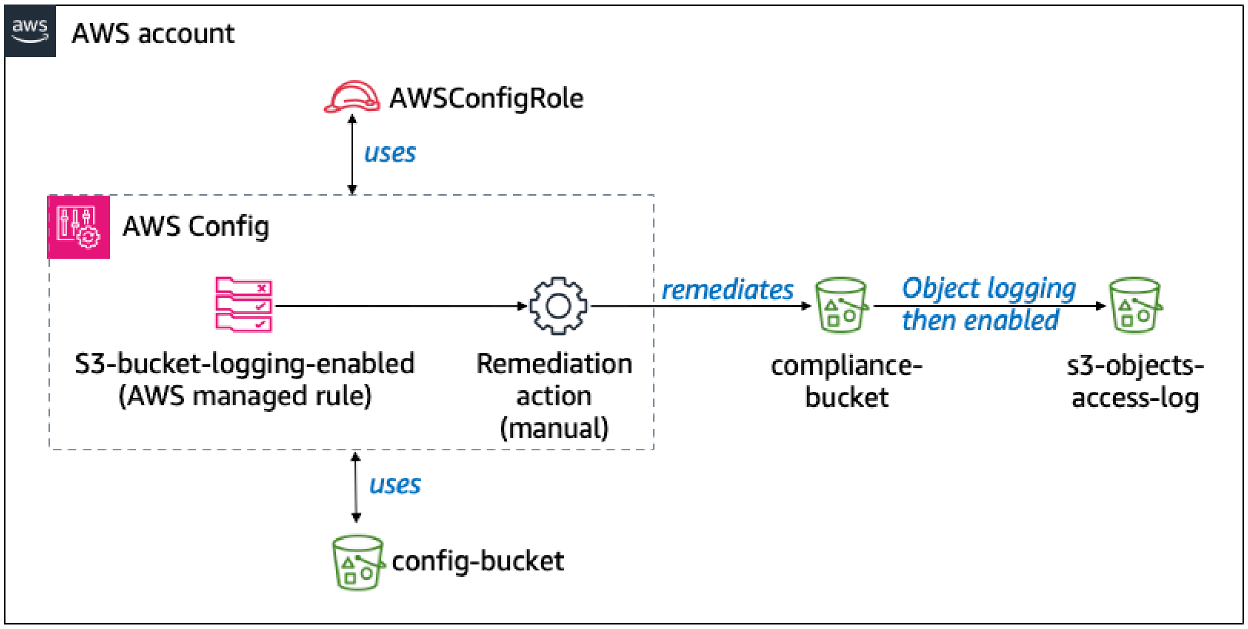
Return to table of contents

AnyCompany Financial would like to have a mechanism to automatically configure AWS resources based on company standards. One standard is to have object logging enabled on S3 buckets.

In this task, you are challenged to use AWS Config to report on whether object logging is configured on the S3 buckets in the AnyCompany Financial account. You will also be challenged to configure an automation script to remediate noncompliance.

This solution will help AnyCompany Financial to remain compliant to standards that have been set for their IT infrastructure while reducing the administrative burden that is associated with ensuring compliance to these standards.

By the end of this task, you will have the architecture that is shown in the following diagram:



1. Review the existing resources that you will use in this task:
    - In the IAM console, review the following roles:
        - **AWSConfigRole:** This role was created for you in this lab environment. The role grants permissions that you will need to set up AWS Config.
        - **SSMAutomationRole:** This is the role that AWS Systems Manager will use when it performs remediation actions with an AWS Config managed rule that you will use.
    - In the Amazon S3 console, notice the **aws-config** bucket. AWS Config will use this bucket for logging purposes.

2. Create a new S3 bucket named `compliance-bucket-unique-ID` where `unique-ID` is the unique ID that all the other buckets in the account contain. *Enable ACLs* on the bucket, and create it in the *us-east-1* Region.

3. Enable object ownership on the *s3-objects-access-log* bucket.

    - Navigate to the Amazon S3 console.
    - In the buckets list, choose the link for the **s3-objects-access-log** bucket.
    - On the **Permissions** tab, scroll down to the **Object Ownership** section, and then choose **Edit**.
    - Choose **ACLs enabled**.
    - Select the check box for the statement that says "I acknowledge that ACLs will be restored," and then choose **Save changes**.

      Note: This change was needed so that the remediation action that you will perform later will succeed.

4. Navigate to the AWS Config console, and choose **Get started**.

    Set up AWS Config with the following characteristics:

    - Record all current and future resources that are supported in the *us-east-1* Region.
    - For AWS Config role, use the existing *AWSConfigRole* role.
    - For delivery method, specify the existing *aws-config* bucket.
    - Don't select any managed rules.

5. In the AWS Config console, add an AWS managed rule.

    When you define the rule, search for and select the rule named `s3-bucket-logging-enabled`. Keep all other default settings and save it.

6. Confirm that the *s3-bucket-logging-enabled* rule that you defined is finding resources that are in scope. Also, confirm that the *compliance-bucket* is flagged as noncompliant.

    - Open the details page for the *s3-bucket-logging-enabled* rule.
    - Wait until you see S3 buckets listed in the **Resources in scope** section of the page.

      Note: If you don't see the bucket listed, you might need to wait up to 15 minutes before it appears. Refresh the page periodically until you see it before continuing.

- Notice that the *compliance-bucket* is flagged as *Noncompliant*.
- Choose the **ID** link for the *compliance-bucket*, and then choose **Manage Resource**. On the **Properties** tab, verify that server access logging is disabled.

  This explains why the bucket was flagged as noncompliant.

  Note: After the remediation action is enabled and you invoke it on this bucket, server access logging will be enabled.

7. Configure manual remediation settings for the *s3-bucket-logging-enable* rule.

- Return to the browser tab where the AWS Config console is open.
- In the navigation pane, choose **Rules**.
- Choose the link for the **s3-bucket-logging-enable** rule.
- Choose **Actions** > **Manage remediation** and configure the following:
- Select remediation method: **Manual remediation**
- Choose remediation action: Search for and select **AWS-ConfigureS3BucketLogging**

  Note: When you select a remediation action, you are choosing a Systems Manager Automation document that will be used to update resources. In this case, server access logging will be enabled on S3 buckets that you choose to manually remediate.

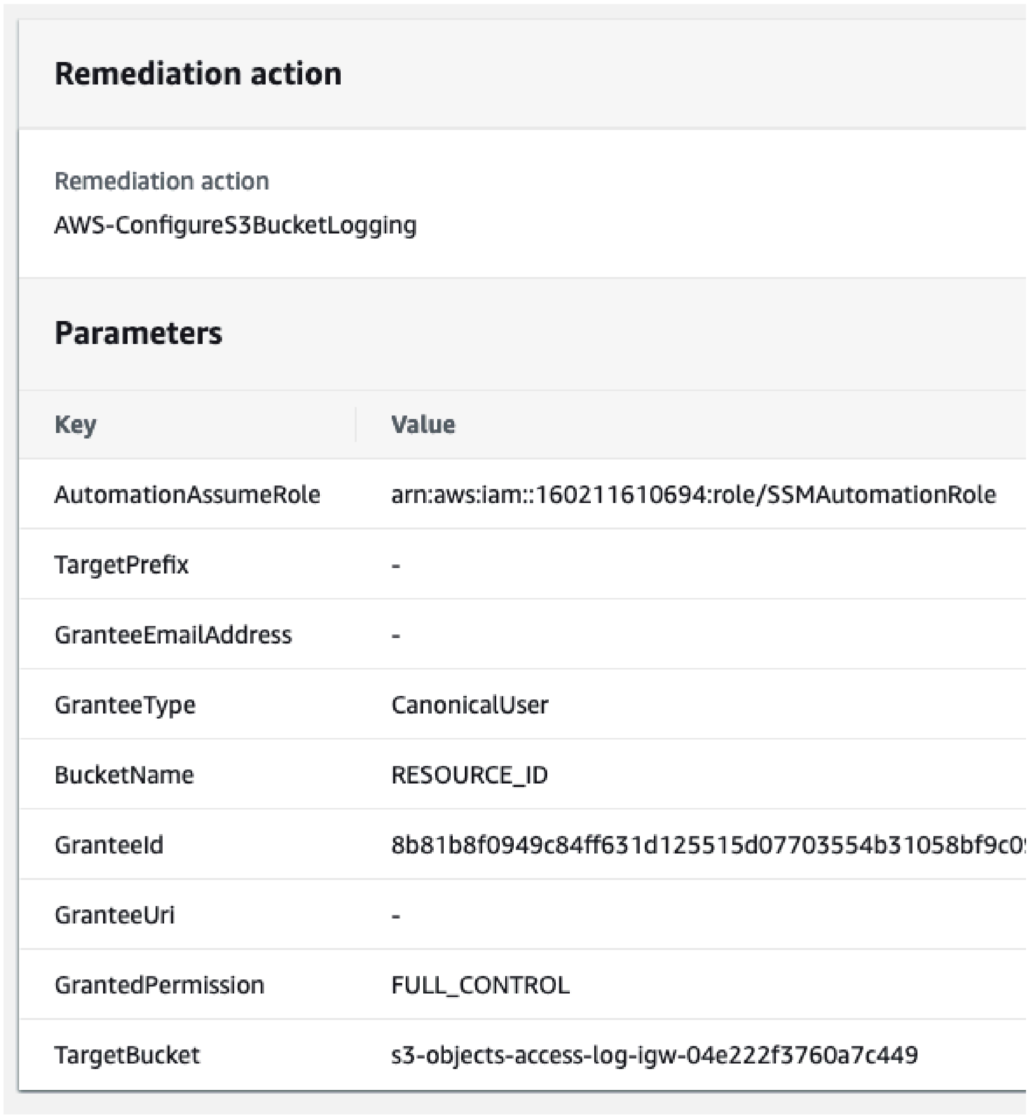
  Tip: For more information about this automation, including the parameters that are used in it, see AWS-ConfigureS3BucketLogging in the *AWS Systems Manager Automation Runbook Reference*.
- Resource ID parameter: **BucketName**
- GrantedPermission: `FULL_CONTROL`
- GranteeType: `CanonicalUser`

  Note: *CanonicalUser* refers to your AWS account.
- TargetBucket: `s3-objects-access-log-uniqueID` (where `uniqueID` is the unique ID that appears in the name of the bucket)

  Tip: *TargetBucket* specifies where you want the server access logs to be stored. You will use the same bucket for logging that you used in phase 1 when you enabled logging on the *data-bucket*.

- GranteeId: Enter the canonical ID for your AWS account. To find this value:
    - Open the Amazon S3 console in a separate browser tab.
    - Choose the link for the **compliance-bucket**.
    - On the **Permissions** tab, scroll down to the **Access control list (ACL)**.
    - Find and copy the **Canonical ID** value.
- AutomationAssumeRole: Enter the ARN for the *SSMAutomationRole*. To find this value:
    - Open the IAM console in a separate browser tab.
    - In the navigation pane, choose **Roles**.
    - Search for and choose the **SSMAutomationRole**.
    - In the **Summary** area of the page, copy the ARN value. It is similar to *arn:aws:iam::ACCOUNT-ID:role/SSMAutomationRole*.
- Choose **Save changes**.

  The configuration for the rule should look similar to the following image:



Remediation action

Remediation action
AWS-ConfigureS3BucketLogging

Parameters

Key	Value
AutomationAssumeRole	arn:aws:iam::160211610694:role/SSMAutomationRole
TargetPrefix	-
GranteeEmailAddress	-
GranteeType	CanonicalUser
BucketName	RESOURCE_ID
GranteeId	8b81b8f0949c84ff631d125515d07703554b31058bf9c0
GranteeUri	-
GrantedPermission	FULL_CONTROL
TargetBucket	s3-objects-access-log-igw-04e222f3760a7c449

8. Invoke the AWS Config remediation action so that object logging is enabled on the *compliance-bucket*.
    - On the details page for the **s3-bucket-logging-enabled** rule, scroll down to the **Resources in scope** section.
    - Choose the radio button for the **compliance-bucket**, and then choose **Remediate**.

      The status changes to *Action execution queued*. Wait a few seconds, and then choose the refresh icon. The status should soon change to *Action executed successfully*.

      Troubleshooting tip: If the action isn't successful, you can find relevant logs in the Systems Manager console. In the navigation pane, choose **Automation**. Choose the **Execution ID** link for the most recent automation job. (Note that the status is likely *Failed* if you are troubleshooting.) In the **Executed steps** section, choose the link for the first step ID that has a status of *Failed*. The execution detail page can provide insight into what you need to configure differently for the AWS Config remediation job to run correctly.

9. Confirm that server access logging is now enabled on the *compliance-bucket*.

   It was enabled by the remediation rule that you just ran.

Cost assessment for monitoring and logging

Return to table of contents

In this phase, you created a CloudTrail trail to create an audit trail of activity on an S3 bucket. You configured CloudWatch Logs and CloudWatch agent to monitor SSH access to an EC2 instance, and you created a CloudWatch alarm to send notifications about security incidents. Finally, you configured AWS Config to monitor S3 buckets.

Question: In this lab, does your use of CloudTrail, CloudWatch, and AWS Config security features result in additional cost in the account? If so, how much do you estimate that it will cost?

Tip: Use the AWS CloudTrail Pricing page, Amazon CloudWatch Pricing page, and AWS Config Pricing page to help you answer the question.

Ending your session

Reminder: This is a long-lived lab environment. Data is retained until you either use the allocated budget or the course end date is reached (whichever occurs first).

To preserve your budget when you are finished for the day, or when you are finished actively working on the assignment for the time being, do the following:

1. At the top of this page, choose **End Lab**, and then choose Yes to confirm that you want to end the lab.

   A message panel indicates that the lab is terminating.

   Note: Choosing **End Lab** in this environment will *not* delete the resource you have created. They will still be there the next time you choose Start lab (for example, on another day).

2. To close the panel, choose **Close** in the upper-right corner.