

AWS Cloud Baseline Security Implementation

Complete Project Documentation

Ibrahim Ayomide Fayomi

Introduction

This documentation provides comprehensive detail on the implementation of AWS baseline security controls across core cloud services. The project establishes a secure cloud infrastructure foundation by configuring identity and access management, deploying protected storage resources, managing compute instances with policy-based controls, and implementing automated monitoring and alerting systems.

The lab demonstrates practical application of AWS security best practices including least privilege access, encryption at rest, comprehensive audit logging, and event-driven security monitoring. Each configuration step addresses specific security requirements while maintaining operational functionality. The implementation follows industry standards for cloud security architecture and serves as a reference for building production-grade secure environments.

This work validates hands-on competency in AWS security implementation and demonstrates the ability to architect, deploy, and manage secure cloud infrastructure. The project covers identity management, data protection, network security, logging and monitoring, and automated incident response.

Environment Setup

The project operates within a standard AWS account environment. Initial access is performed using **root** account credentials to establish the baseline **IAM** configuration. The root account is used exclusively for creating the **first administrative user** and is not used for ongoing operations. All subsequent configuration work is performed through the **administrative IAM** user account following **AWS security best practices**.

The environment requires no additional software installations or client-side tools. All configuration is performed through the AWS Management Console web interface. Services are deployed in a single AWS region using default VPC networking. The setup assumes basic familiarity with AWS console navigation and service access patterns.

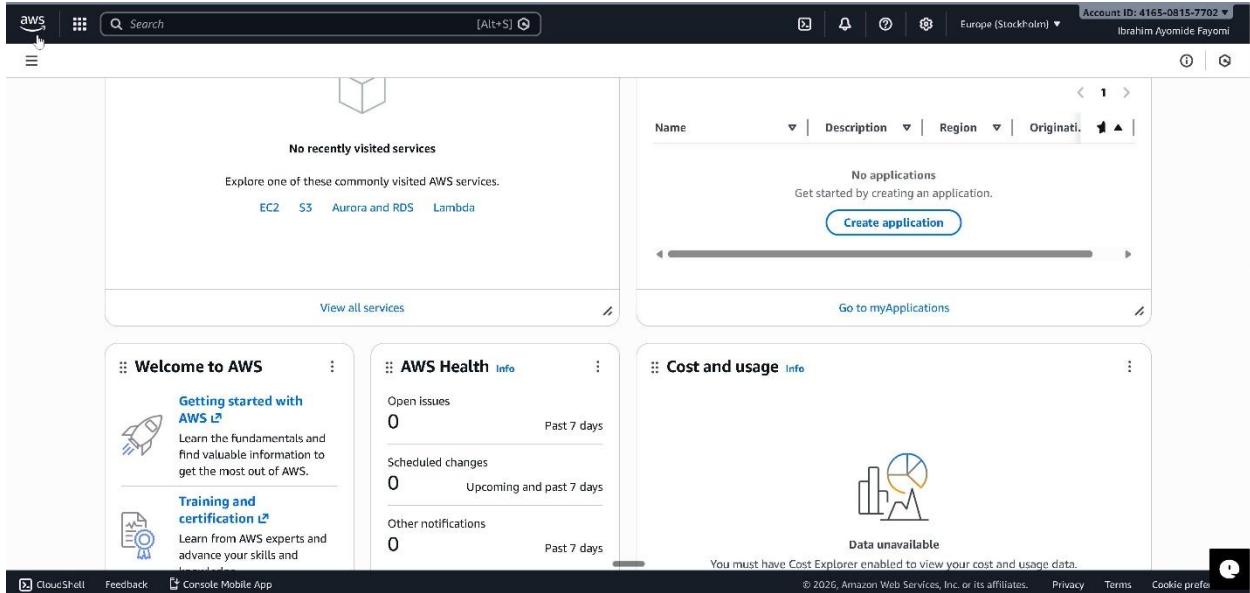
Step-by-Step Configuration

This section documents each configuration step in the exact order performed. Each screenshot is explained with the action taken and the security purpose behind the configuration.

IAM SERVICE

Identity and Access Management (IAM)

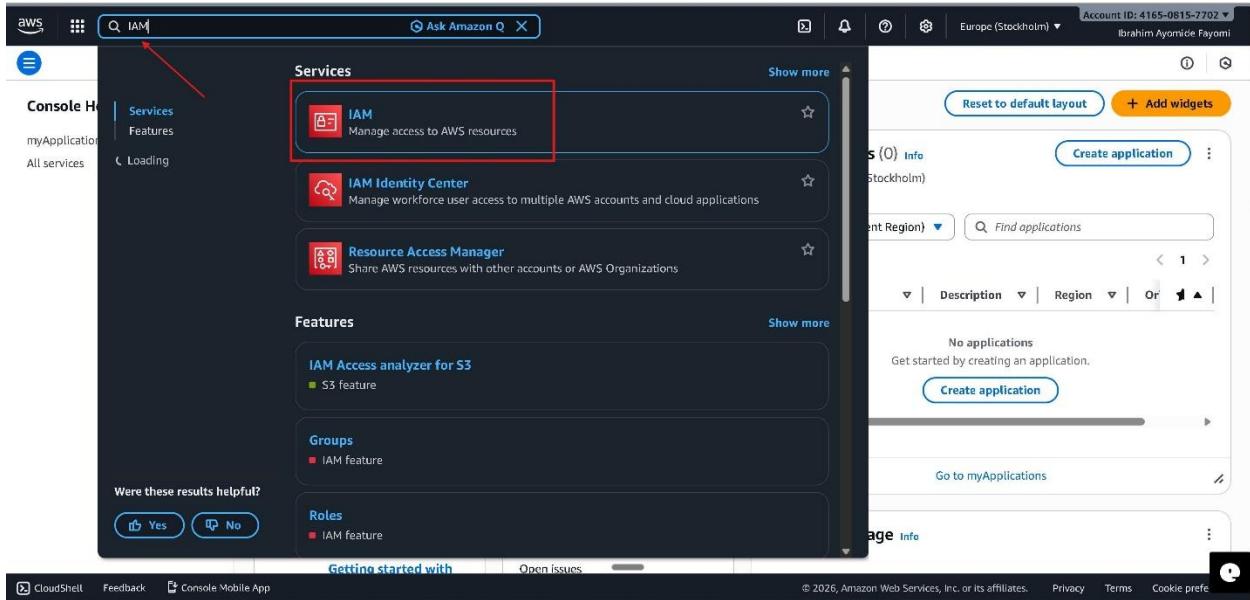
AWS Root Account Dashboard



The screenshot displays the AWS root account dashboard after successful login. The console shows the primary AWS services interface with the region selector visible in the top navigation bar. The root account provides unrestricted access to all AWS resources and billing information. This is the starting point for IAM configuration but should not be used for routine operations.

Using the root account for daily tasks violates AWS security best practices. The root account should be secured with multi-factor authentication and used only for tasks that specifically require root credentials such as closing the account or changing support plans.

Searching for IAM Service from Root Console



The screenshot shows the AWS console search bar with IAM entered as the search term.

The search interface displays IAM service as the first result. This demonstrates how to quickly navigate to specific AWS services from the main console. The search function provides faster access than navigating through service categories.

IAM Dashboard

The screenshot shows the AWS IAM Dashboard. On the left, a navigation menu includes 'Identity and Access Management (IAM)', 'Access Management' (with options like User groups, Users, Roles, Policies, Identity providers, and Account settings), 'Access reports' (with options like Access Analyzer, Resource analysis, and Unused access), and 'AWS CloudTrail' (with a 'New' button). The main content area has a blue header bar with a message about new access analyzers and a 'Create new analyzer' button. Below this is the 'IAM Dashboard' section, which includes a 'Security recommendations' panel with two items: 'Root user has MFA' (green checkmark) and 'Root user has no active access keys' (green checkmark). It also features a 'IAM resources' summary with counts: 0 User groups, 0 Users, 3 Roles, 0 Policies, and 0 Identity providers. A 'Quick Links' panel provides links to 'My security credentials' and other management tools. The top right corner shows account information: Account ID: 4165-0815-7702, Account Alias: Create, and a sign-in URL: https://416508157702.signin.aws.amazon.com/console.

The IAM dashboard presents an overview of the current identity configuration. The security recommendations panel shows items requiring attention such as enabling MFA for root account and creating IAM users. The dashboard displays counts for users, groups, roles, and policies. The left navigation menu provides access to all IAM features including users, groups, roles, policies, and account settings.

The dashboard serves as the central control point for all identity and access management activities. The security recommendations help identify configuration gaps that could expose the account to unauthorized access.

Creating Admin User Under IAM Console

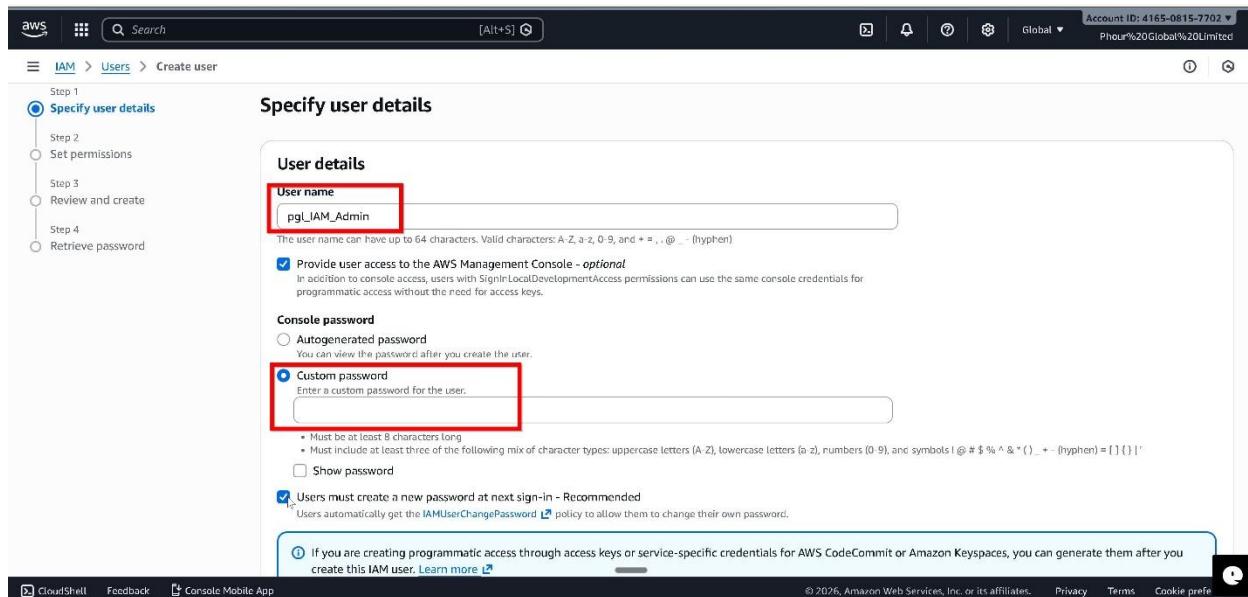
The screenshot shows the AWS IAM Dashboard. On the left, a sidebar menu is open under 'Access Management', with 'Users' highlighted by a red box. The main content area is titled 'IAM Dashboard' and includes sections for 'Security recommendations' (with two items: 'Root user has MFA' and 'Root user has no active access keys') and 'IAM resources' (listing 0 User groups, 0 Users, 3 Roles, 0 Policies, and 0 Identity providers). A blue banner at the top right says 'New access analyzers available'. On the right side, there's an 'AWS Account' summary and a 'Quick Links' section.

The screenshot shows the 'Users' page within the IAM console. The left sidebar shows the 'Access Management' section with 'Users' selected. The main area is titled 'Users (0)' and contains a message: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' Below this is a search bar and a table header with columns: User name, Path, Group, Last activity, MFA, Password age, and Console last sign-in. A red box highlights the 'Create user' button in the top right corner. The table body shows 'No resources to display'.

These screenshots show the IAM user creation wizard. The interface prompts for user details and access type selection. The admin user is being created with AWS Management Console access, which requires setting a password. The option to require password reset at first login is visible but not selected in this configuration.

Creating an administrative IAM user separate from the root account follows AWS security best practices. This user will handle all administrative tasks going forward, leaving the root account secured and unused except for specific root-only operations. The administrative user provides full access while maintaining audit trails of actions performed.

Admin User Named and Passworded



The screenshot displays the username and password configuration for the admin user. The user is named with a clear administrative designation. Console password is set with options for custom password entry. This step establishes the authentication credentials that will be used to access the AWS console as this user.

Adding Permission Directly to Admin User

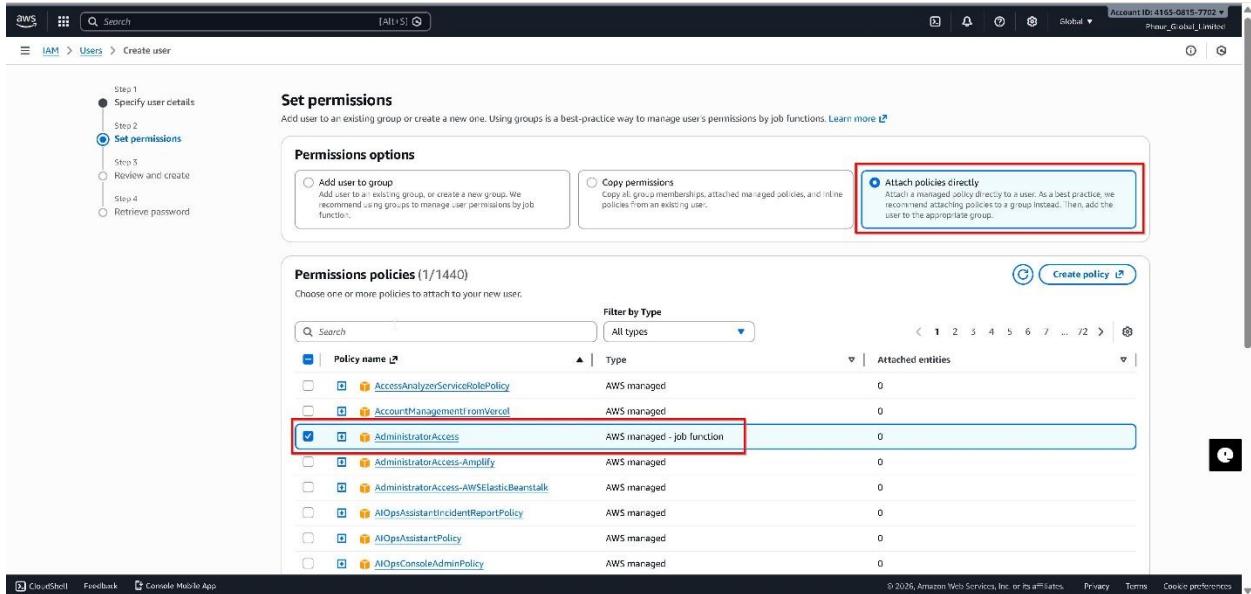
The screenshot shows the AWS IAM 'Create user' wizard at Step 2: Set permissions. The 'Attach policies directly' option is selected and highlighted with a red box. Below it, a table lists three policy types:

Policy name	Type	Attached entities
AccessAnalyzerServiceRolePolicy	AWS managed	0
AccountManagementFromVercel	AWS managed	0
AdministratorAccess	AWS managed - job function	0

The permissions configuration screen shows three methods for assigning permissions: adding user to group, copying permissions from existing user, or attaching policies directly. The direct policy attachment option is selected. This demonstrates different approaches to permission assignment with varying levels of management overhead and scalability.

Direct policy attachment is appropriate for unique administrative users but group-based permissions are preferred for standard users to simplify management. The chosen method depends on the specific access requirements and organizational structure.

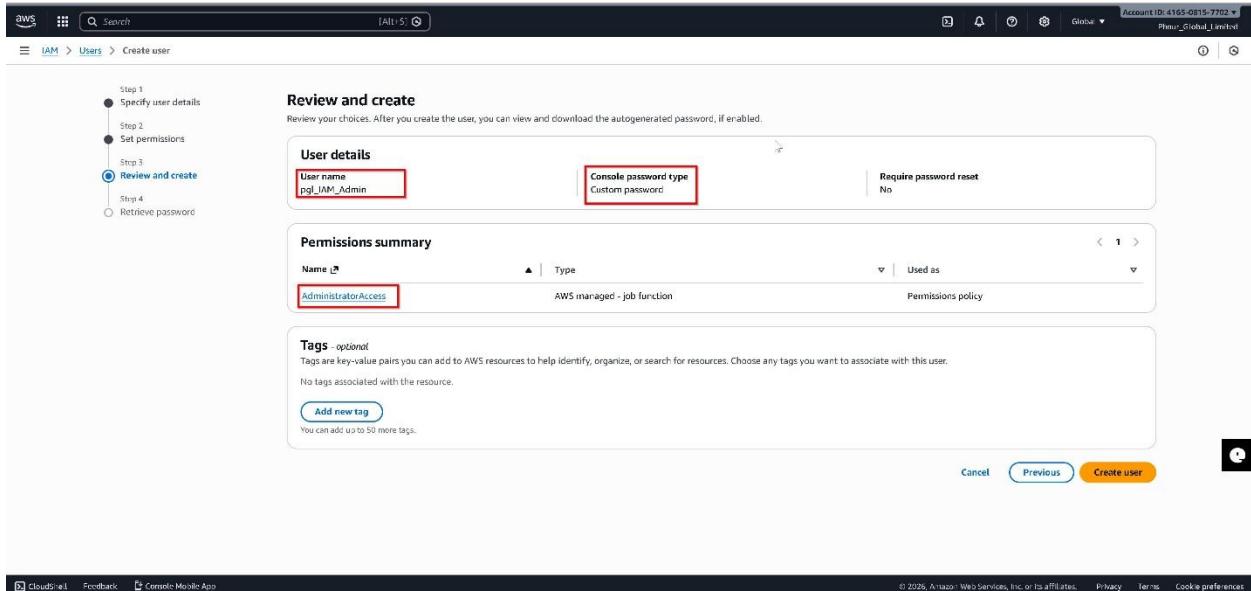
Granting Admin Access to Admin User Created



The screenshot shows the policy selection interface with AdministratorAccess policy selected. This AWS managed policy grants full access to all AWS services and resources. The search filter shows how to locate specific policies from the extensive policy list. The policy attachment will give this user equivalent permissions to root account for service operations.

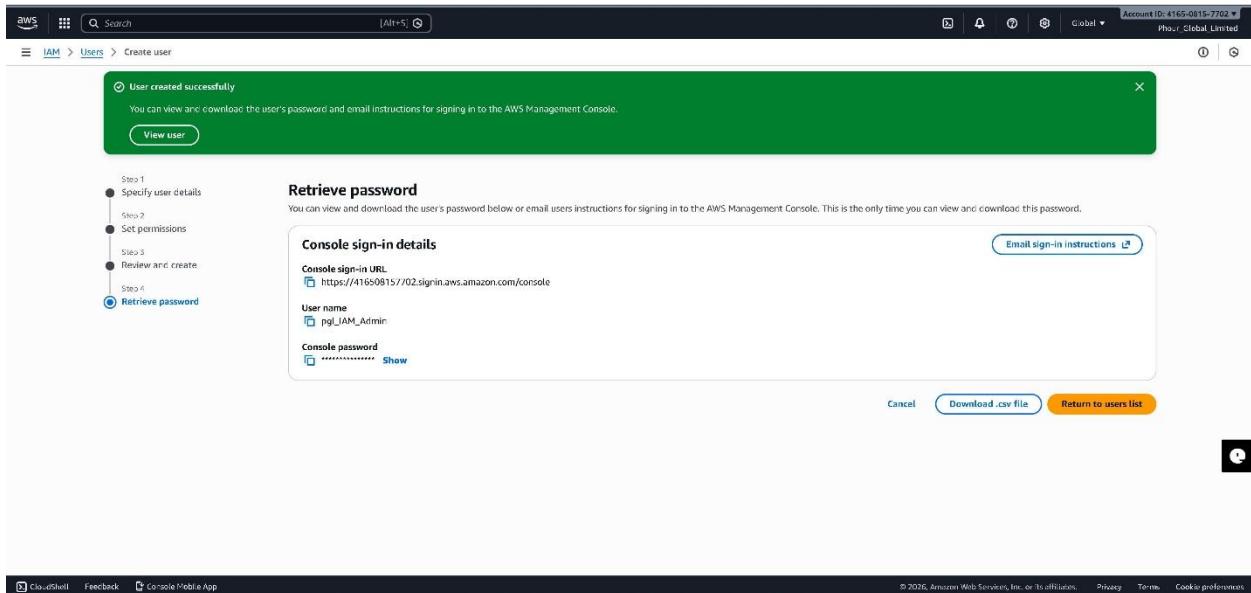
Administrative Access policy provides unrestricted service access while still maintaining separate identity for audit purposes. This allows full administrative control while avoiding direct use of root credentials for operational tasks.

Admin User Created Summary



The user creation summary displays all configured settings before final confirmation. The review shows username, console access enabled, password configuration, and attached Administrative Access policy. This final review step allows verification of all settings before the user is created. Any errors can be corrected by navigating back through the wizard.

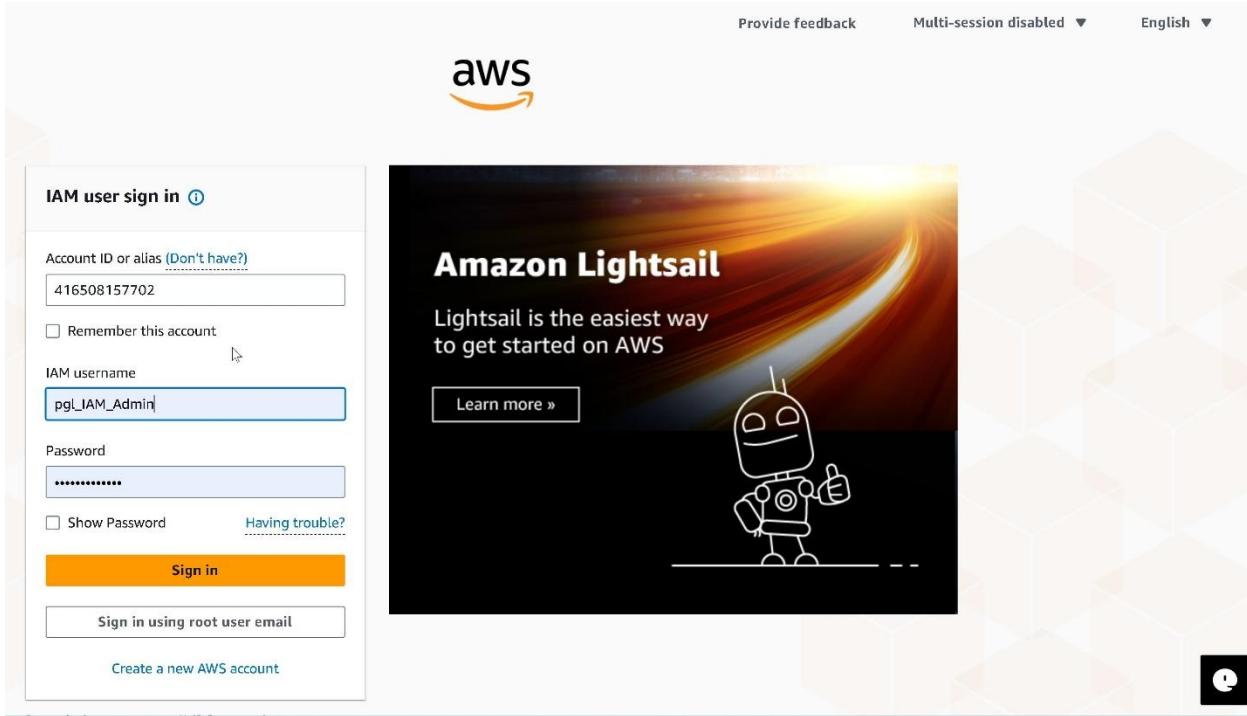
Admin User Created



The success screen confirms the admin user has been created. The console displays the sign-in URL for IAM users along with the username. This URL is specific to the AWS account and is used by all IAM users to access the console. The credentials can be downloaded or sent to the user. The account ID is embedded in the sign-in URL.

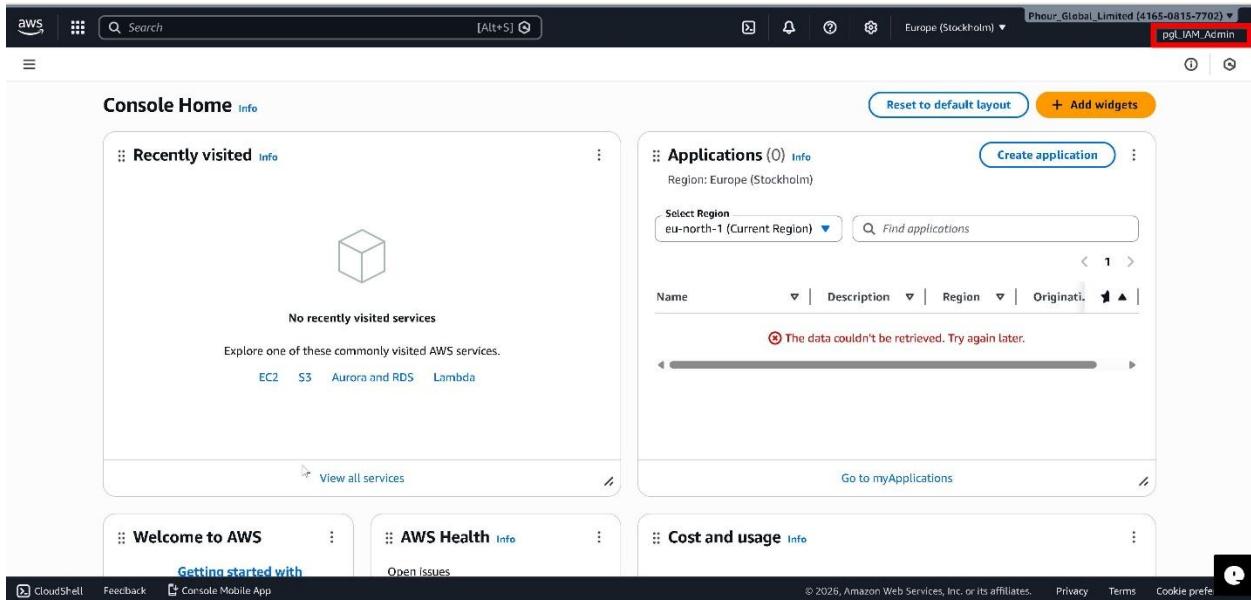
The IAM user sign-in URL differs from the root account login page. IAM users authenticate using account ID or alias plus username and password rather than email address. This separation provides clear distinction between root and IAM user access.

Logging in as IAM Admin User



The IAM user sign-in page displays fields for account ID, username, and password. The account ID field may show as account alias if configured. This login interface is distinct from root account login and can only be used by IAM users. The URL contains the account identifier which routes authentication to the correct AWS account.

Logged in as Admin User (pgl_IAM_Admin)



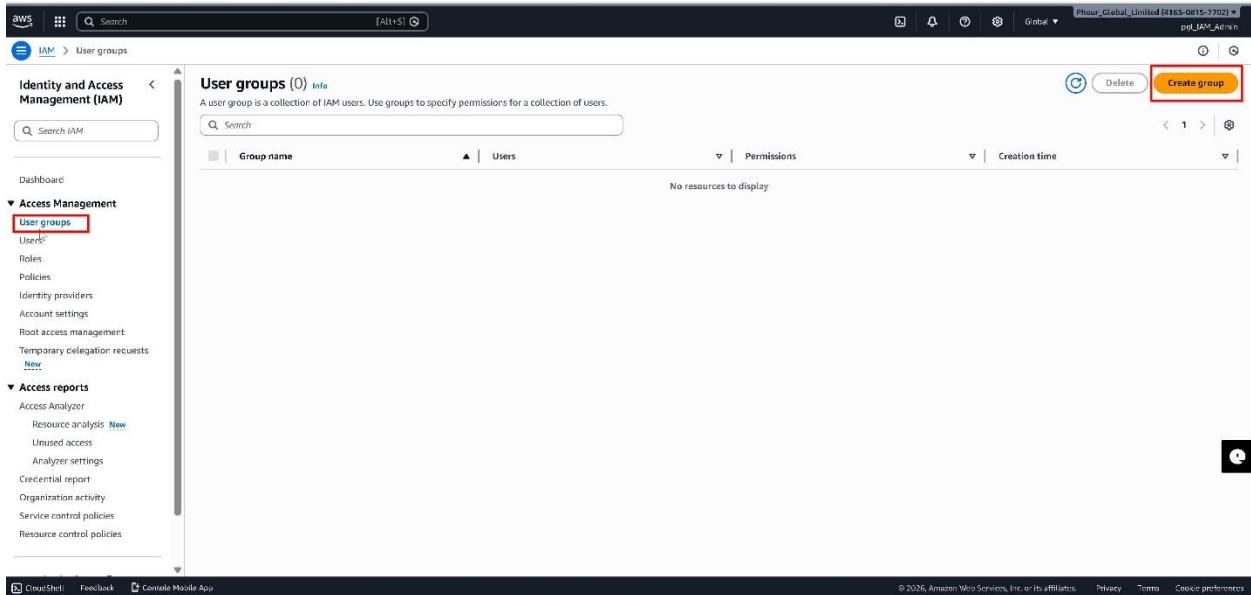
The console displays successful authentication as the admin IAM user. The top navigation bar shows the username indicating current session identity. All subsequent actions will be performed under this user identity and logged to CloudTrail with the IAM user as the principal. The console interface remains the same but operations now execute with IAM user credentials rather than root.

Using the IAM admin user for all operations provides audit trails showing which administrative user performed each action. This accountability is not possible when using shared root credentials.

Creating User Groups (under Admin user console)

The screenshot shows the AWS IAM Admin user console. On the left, there's a sidebar with links like 'Features', 'Documentation', 'Knowledge articles', etc. The main area has a 'Services' section with three cards: 'IAM' (highlighted with a red box), 'IAM Identity Center', and 'Resource Access Manager'. Below that is a 'Features' section with 'IAM Access analyzer for \$3' (highlighted with a red box), 'Groups', and 'Roles'. A message at the bottom asks if results were helpful.

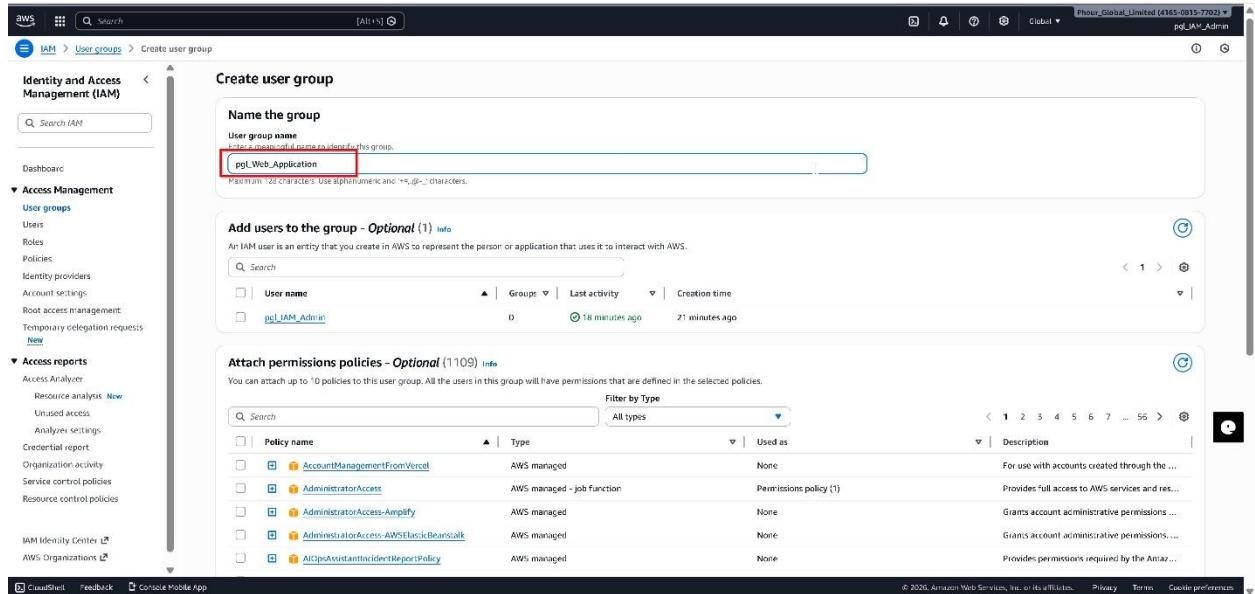
The screenshot shows the IAM Dashboard. The left sidebar has sections for 'Access Management' (with 'User groups' highlighted with a red box) and 'Access reports'. The main dashboard includes sections for 'Security recommendations', 'AWS Account', 'IAM resources' (showing 0 user groups, 1 user, 3 roles, 0 policies, 0 identity providers), 'What's new', and 'Tools'.



These screenshots show the user group creation process. The IAM console shows the groups section with the option to create new group. The group creation wizard prompts for group name and policy attachments. User groups provide a scalable way to manage permissions for multiple users performing similar functions.

Groups simplify permission management by applying policies to the group rather than individual users. When a user joins a group they automatically inherit all group permissions. This reduces administrative overhead and ensures consistent permission application across team members.

User Group Named (under Admin user console)



The screenshot shows a user group being created with a descriptive name such as Sales or Development. The group name should clearly indicate the function or department the group serves. Group names must be unique within the AWS account. Naming conventions help maintain organization when managing multiple groups across different teams.

User Group Created

The screenshot shows the AWS IAM User Groups page. At the top, a green banner displays the message "pgl_Web_Application user group created." Below this, the title "User groups (1) Info" is shown. A sub-banner below the main title says "A user group is a collection of IAM users. Use groups to specify permissions for a collection of users." The main content area is titled "User groups (1)" and contains a table with one row. The table has columns for "Group name", "Users", "Permissions", and "Creation time". The single row in the table is highlighted with a red border. The "Group name" column shows "pgl_Web_Application", the "Users" column shows "0", the "Permissions" column shows "not defined", and the "Creation time" column shows "Now". The left sidebar shows navigation options like "Identity and Access Management (IAM)", "Access Management", "Access reports", and "AWS Organizations". The bottom of the page includes standard AWS footer links.

The groups list now shows the newly created group. The interface displays group name, creation date, and path. The group currently has no users assigned. Policies can be attached to the group before or after users are added. The group structure is now ready to receive user assignments and policy attachments.

Multiple User Groups Created

The screenshot shows the 'Create user group' interface in the AWS IAM console. The 'Name the group' section has 'User group name' set to 'pg1_AI_Project'. The 'Add users to the group - Optional (1)' section lists 'pg1_IAM_Admin' as a member. The 'Attach permissions policies - Optional (1109)' section shows three policies attached: 'AccountManagementFromVercel' (AWS managed), 'AdministratorAccess' (AWS managed - job function), and 'AdministratorAccess-Amplify' (AWS managed). A success message at the top says 'pg1_Web_Application user group created.'

The screenshot shows the 'User groups' page in the AWS IAM console. It lists two groups: 'pg1_AI_Project' and 'pg1_Web_Application'. Both groups have '0' users and '0' permissions assigned. The 'pg1_AI_Project' group was created 'Now' and the 'pg1_Web_Application' group was created '5 minutes ago'. A success message at the top says 'pg1_AI_Project user group created.'

The screenshots show multiple user groups have been created for different organizational functions. Each group will receive appropriate permissions based on the access requirements for that role. Multiple groups allow fine-grained access control aligned with business functions. Users can be members of multiple groups if their role requires combined permissions.

Creating Standard Users

The screenshot shows the AWS IAM User Groups page. A success message at the top indicates that 'pgl_sales user group created.' The main table lists three user groups: 'pgl_AI_Project' (created 2 hours ago), 'pgl_sales' (created now), and 'pgl_Web_Application' (created 3 hours ago). The 'Users' column shows 0 users assigned to each group. The 'Permissions' column shows 'Not defined' for all groups. The 'Creation time' column shows the respective times of creation.

The screenshot shows the 'Specify user details' step of the 'Create user' wizard. The user name is set to 'pgl_IAM_Solante_Hope'. The 'Provide user access to the AWS Management Console - optional' checkbox is checked. Under 'Console password', the 'Custom password' option is selected, and a password '*****' is entered. The 'Users must create a new password at next sign-in - Recommended' checkbox is checked. At the bottom, there is a note about generating programmatic access keys.

The user creation process is initiated for standard non-administrative users. The wizard follows the same flow as admin user creation but will assign different permissions. Standard users receive only the permissions necessary for their specific job functions following the principle of least privilege. These users will be assigned to appropriate groups or given direct permissions based on requirements.

Permission Assignment Scenarios

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more [?](#)

Permissions options

- Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly

Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

User groups (1/3)

Group name	Users	Attached policies	Created
pgl AI Project	0	-	2026-01-13 (3 hours ago)
pgl_talent	0	-	2026-01-13 ('1 minute ago)
pgl_Web_Application	0	-	2026-01-13 (3 hours ago)

Set permissions boundary - optional

Cancel Previous Next

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more [?](#)

Permissions options

- Add user to group

Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions

Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly

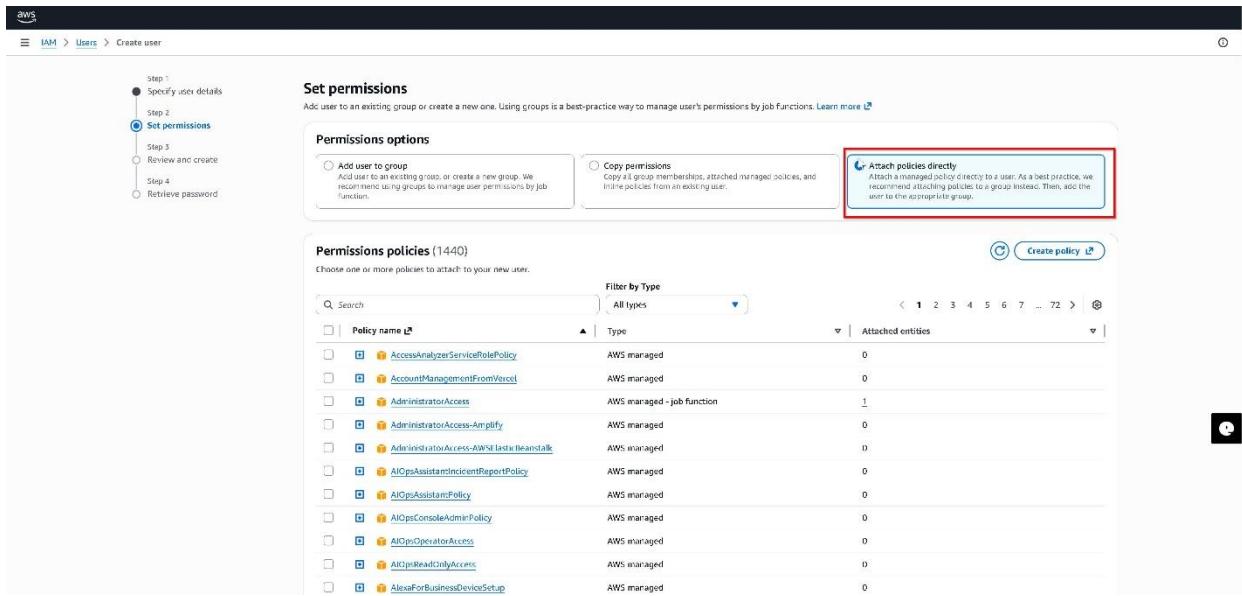
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Users (1/1)

User name	Groups	Attached policies
pgl_IAM_Admin	-	AdministratorAccess

Set permissions boundary - optional

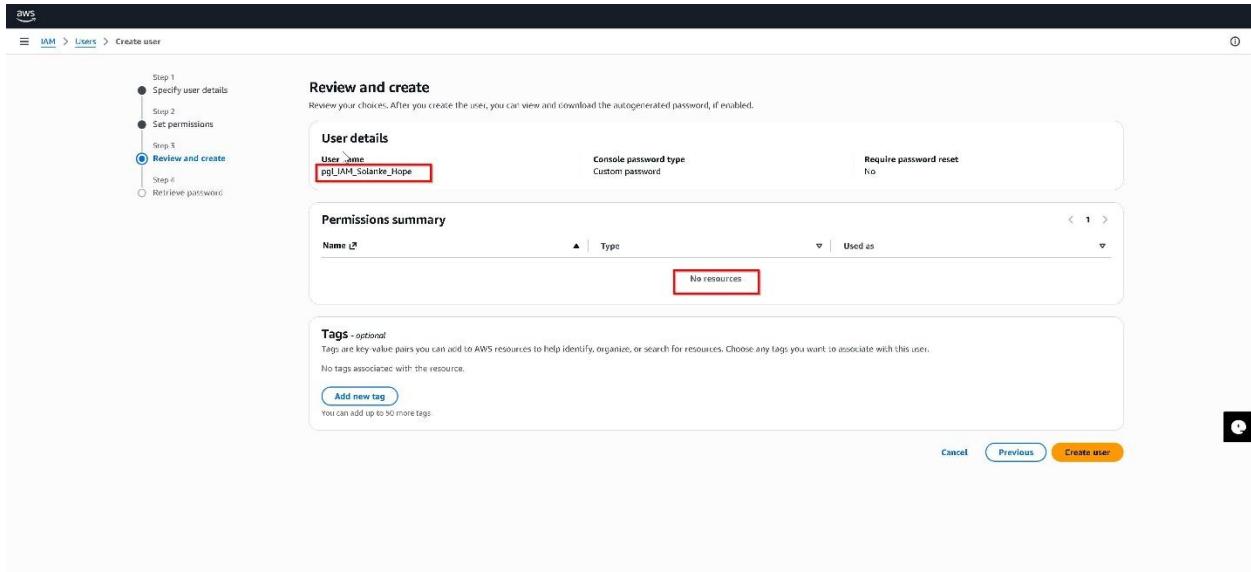
Cancel Previous Next



These screenshots demonstrate three different methods for assigning permissions to users. Scenario one shows adding a user directly to the Sales group, which gives the user all permissions attached to that group. Scenario two displays copying permissions from an existing user, useful when creating accounts for users with similar roles. Scenario three shows attaching policies directly to the user without using groups.

Each permission assignment method has appropriate use cases. Group membership is preferred for standard users as it simplifies management. Direct policy attachment works for unique cases or temporary permission grants. Copying from existing users speeds up provisioning when multiple users need identical permissions.

New User Created Without Permissions



The screenshot shows a user created without any permission assignments. The user exists in the account but cannot access any AWS resources or perform any actions. Permissions can be added after user creation by adding the user to groups or attaching policies. Creating users without initial permissions allows staged provisioning where access is granted only when needed.

Adding User to Group from User Groups Section

The screenshot shows the AWS IAM User Groups page for the 'pgl_sales' group. The left sidebar shows navigation options like Dashboard, Access Management (User groups, Roles, Policies, Identity providers, Account settings, Root access management, Temporary delegation requests), and Access reports (Access Analyzer, Resource analysis, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies, Resource control policies). The main content area displays the 'pgl_sales' group's summary, creation time (January 13, 2026, 16:27 UTC+01:00), ARN (arn:aws:iam:416508157702:group/pgl_sales), and a list of users in the group. A red box highlights the 'User group name' field ('pgl_sales') and the 'Add users' button.

The screenshot shows the 'Add users to pgl_sales' dialog box. It lists 'Other users in this account (1/3)' with three users: 'pgl_IAM_Admin', 'pgl_IAM_Ibrahim_Fayomi', and 'pgl_IAM_Solanke_Hope'. The 'pgl_IAM_Solanke_Hope' user is selected (indicated by a checked checkbox) and highlighted with a red box. The 'Add users' button at the bottom right is also highlighted with a red box.

User name	Group	Last activity	Creation time
pgl_IAM_Admin	0	7 minutes ago	4 hours ago
pgl_IAM_Ibrahim_Fayomi	0	-	33 minutes ago
pgl_IAM_Solanke_Hope	0	-	34 minutes ago

The screenshot shows the AWS IAM User groups page. The left sidebar has sections for Dashboard, Access Management (User groups, Roles, Policies, Identity providers, Account settings, Root access management, Temporary delegation requests), and Access reports (Access Analyzer, Resource analysis, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies, Resource control policies). The main content area shows 'User groups (3) Info' with a note: 'A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.' A search bar is at the top. Below is a table with columns: Group name, Users, Permissions, and Creation time.

Group name	Users	Permissions	Creation time
pg_AI_Project	<input type="checkbox"/> pg_sales	0 Not defined 1 Not defined	4 hours ago 1 hour ago
pg_Web_Application	<input type="checkbox"/>	0 Not defined	4 hours ago

The URL in the address bar is <https://us-east-1.console.aws.amazon.com/iam/home?region=eu-north-1#/groups>.

These screenshots show the process of adding an existing user to a group from the groups management interface. The group details page displays current members and provides the option to add users. Selecting users from the account list adds them to the group. The group membership count updates to reflect the new addition. Users immediately inherit all permissions attached to the group upon joining.

Searching for IAM Service

The screenshot shows the AWS Identity and Access Management (IAM) service interface. The left sidebar has a search bar at the top labeled "Search IAM". Below it, under "Access Management", the "Users" option is selected and highlighted with a red box. The main content area is titled "Users (3)" and contains a table with three rows, each representing a user. The users listed are "pgl_IAM_Admin", "pgl_IAM_Ibrahim_Fayomi", and "pgl_IAM_Solarice_Hope". The "pgl_IAM_Ibrahim_Fayomi" and "pgl_IAM_Solarice_Hope" rows are also highlighted with red boxes. The table includes columns for User name, Path, Groups, Last activity, MFA, Password age, Console last sign-in, Access key ID, Active key age, and Access. At the bottom right of the main area, there is a "Chat with" button.

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access
pgl_IAM_Admin	/	0	11 minutes ago	-	4 hours	11 minutes ago	-	-	-
pgl_IAM_Ibrahim_Fayomi	/	1	-	-	36 minutes	-	-	-	-
pgl_IAM_Solarice_Hope	/	1	-	-	38 minutes	-	-	-	-

The screenshot demonstrates accessing IAM service through the console search. This shows efficient navigation between AWS services during configuration work. The search bar provides quick access to any service without memorizing menu locations. IAM remains a frequently accessed service during security configuration tasks

S3 BUCKET

Amazon S3 Bucket Configuration

Adding S3 Bucket to IAM Admin Console

FA	Password age	Console last sign-in	Access key ID	Active key age	Actions
4 hours	11 minutes ago	-	-	-	
36 minutes	-	-	-	-	
38 minutes	-	-	-	-	

The console shows navigation to the S3 service. S3 provides object storage with high durability and availability. The service is accessed through the console search or services menu. S3 will be used to demonstrate storage security controls including encryption and access policies.

S3 Bucket Dashboard

The screenshot shows the 'Create bucket' wizard in the AWS S3 console. In the 'General configuration' section, the 'Bucket name' field contains 'jmnzr-s3-demo-bucket'. Under 'Object Ownership', 'ACLs disabled (recommended)' is selected. The 'Object Ownership' status is shown as 'Bucket owner enforced'.

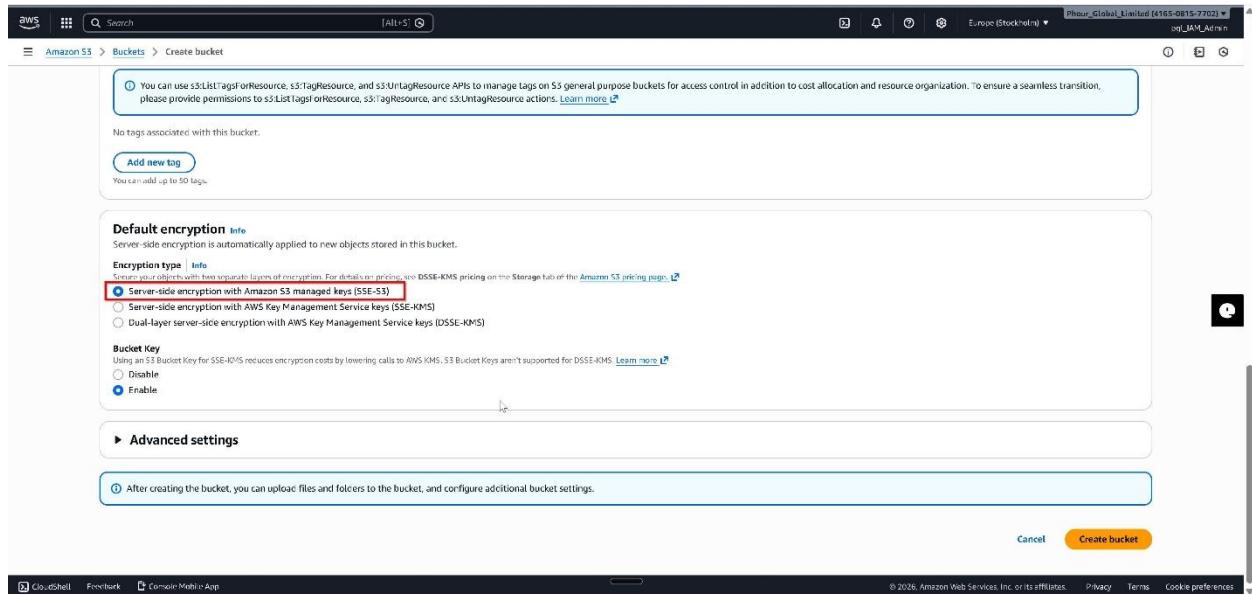
The S3 dashboard displays existing buckets if any and provides options to create new buckets. The interface shows bucket names, regions, and access settings. The create bucket button initiates the bucket creation wizard. S3 bucket names must be globally unique across all AWS accounts.

Naming S3 Bucket

The screenshot shows the 'Create bucket' wizard in the AWS S3 console. The 'Bucket name' field is highlighted with a red box and contains 'pgl-internal-documents'. Under 'Object Ownership', 'ACLs disabled (recommended)' is selected. The 'Object Ownership' status is shown as 'Bucket owner enforced'.

The bucket creation wizard prompts for bucket name and region selection. Bucket names must follow DNS naming conventions using lowercase letters, numbers, and hyphens. The name must be globally unique across all AWS accounts and regions. Region selection affects data residency and latency. Selecting a region close to users improves performance.

Using Server-Side Encryption with Amazon S3 Managed Keys (SSE-S3)



The encryption configuration section shows server-side encryption options. SSE-S3 is selected which uses AES-256 encryption with keys managed by AWS. This provides encryption at rest for all objects stored in the bucket. The encryption is transparent to users and requires no additional configuration or key management. Objects are automatically encrypted when uploaded and decrypted when downloaded.

Encryption at rest protects data from unauthorized access if storage media is compromised. SSE-S3 provides strong encryption without the overhead of customer-managed keys. This encryption option is appropriate for most use cases and meets compliance requirements for data protection.

S3 Bucket Created

The screenshot shows the AWS S3 Buckets page. At the top, a green banner indicates "Successfully created bucket 'pgl-internal-documents'". Below this, the "General purpose buckets" section lists one item: "pgl-internal-documents". The bucket details show it was created on "January 14, 2026, 15:06:34 (UTC+01:00)". To the right, there are two cards: "Account snapshot" and "External access summary".

The bucket list now displays the newly created bucket. The interface shows bucket name, region, and access indicators. The bucket is ready to receive objects. Default settings include all public access blocked which prevents accidental exposure of data. The bucket configuration can be modified after creation if requirements change.

Uploading Documents to S3 Bucket

The screenshot shows the AWS S3 Object Details page for the "pgl-internal-documents" bucket. The "Objects" tab is selected, showing "0" objects. A message states "No objects" and "You don't have any objects in this bucket.". At the bottom, there is a prominent blue "Upload" button with a camera icon. The top navigation bar shows the path "Amazon S3 > Buckets > pgl-internal-documents".

The screenshot shows the AWS S3 console's upload interface. At the top, the navigation bar includes 'Amazon S3 > Buckets > pg1-internal-documents > Upload'. The main area is titled 'Upload' with a 'Info' link. A note says: 'Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. Learn more.' Below this is a dashed-line box for dragging files or choosing them. The 'Files and folders (0)' section has a 'Find by name' search bar and columns for 'Name', 'Folder', 'Type', and 'Size'. A message at the bottom says 'No files or folders' and 'You have not chosen any files or folders to upload.' On the right, there are 'Remove', 'Add files' (which is highlighted with a red box), and 'Add folder' buttons. Below this is the 'Destination' section with a 'Destination' dropdown set to 's3://pg1-internal-documents'. The 'Destination details' section contains a note about bucket settings. At the bottom left are 'Permissions' and 'Properties' links, and at the bottom right are 'CloudWatch', 'Feedback', and 'Console Mobile App' links.

These screenshots show the upload interface within the bucket. The upload button opens a dialog to select files from the local system. Multiple files can be selected and uploaded simultaneously. The interface shows upload progress and confirms successful transfers. Uploaded objects inherit the bucket encryption settings automatically.

File Uploaded to S3 Bucket

This screenshot shows the same AWS S3 upload interface after a file has been selected. The 'Files and folders (1 total, 50.9 MB)' section now lists 'KARAMOT NOUN PROJECT.pdf' with its details: type 'application/pdf' and size '50.9 MB'. The 'Add files' button is still highlighted with a red box. The rest of the interface remains the same, including the destination and other settings.

The screenshot shows files listed within the bucket after upload. Object details include name, size, last modified date, and storage class. Clicking an object displays additional metadata and provides options to download, delete, or modify properties. Objects are encrypted at rest using the bucket encryption configuration.

File Uploading in Progress

The screenshot captures the AWS S3 console interface during a file upload. At the top, a prominent progress bar indicates the upload is 93.3% complete, with a total remaining time of 5 minutes. Below the progress bar, the 'Upload: status' section shows a summary: 1 file has been successfully uploaded (Succeeded) and 0 files have failed (Failed). The 'Files and folders' section lists a single file, 'KARAMOT NOUN PROJECT.pdf', which is 50.9 MB in size and currently in progress (7%). The table includes columns for Name, Folder, Type, Size, Status, and Error. The 'Status' column for the file shows '(In progress (7%))'. The bottom of the screen features standard AWS navigation links like CloudWatch, Feedback, and Console Mobile App, along with legal notices and cookie preferences.

Name	Folder	Type	Size	Status	Error
KARAMOT NOUN PROJECT.pdf	-	application/pdf	50.9 MB	(In progress (7%))	-

The upload progress interface displays during file transfer. Progress bars show upload status for each file. Large files may take time to transfer depending on connection speed. The interface allows canceling uploads if needed. Failed uploads can be retried without affecting successfully uploaded objects.

File Upload to S3 Bucket Successful

The screenshot shows the Amazon S3 console interface. At the top, there's a green success message box containing the text "Upload succeeded" and "For more information, see the Files and folders table." Below this, a header bar includes the AWS logo, a search bar, and navigation links like "Amazon S3". The main content area has a title "Upload: status" with a "Close" button. A note says "After you navigate away from this page, the following information is no longer available." Below this is a "Summary" section with two boxes: "Succeeded" (containing "1 file, 50.9 MB (100.00%)") and "Failed" (containing "0 files, 0 B (0%)"). There are tabs for "Files and folders" and "Configuration", with "Files and folders" selected. A sub-section titled "Files and folders (1 total, 50.9 MB)" shows a table with one item: "KARAMOT NOUN PROJECT.pdf" (application/pdf, 50.9 MB, Status: Succeeded). The entire "Files and folders" table is highlighted with a red border. At the bottom of the page, there are links for "CloudShell", "Feedback", and "Console Mobile App", along with copyright information and links for "Privacy", "Terms", and "Cookie preferences".

The success confirmation shows uploads completed. Objects are now stored in S3 with configured encryption and durability guarantees. The objects can be accessed by authorized users based on bucket policies and IAM permissions. S3 provides eleven nines of durability through redundant storage across multiple facilities.

Assigning S3 Full Access Policy to User

The screenshot shows the AWS IAM 'Users' page. The left sidebar has 'Access Management' expanded, with 'Users' selected and highlighted with a red box. The main table lists three users:

User Name	Path	Groups	Last Activity	MFA	Password Age	Console Last Sign-in	Access Key ID	Active Key Age	Access Type
pgl_IAM_Admin	/	0	3 hours ago	-	Yesterday	3 hours ago	-	-	-
pgl_IAM_Ibrahim_Fayomi	/	1	-	-	22 hours	-	-	-	-
pgl_IAM_Solarike_Ilope	/	1	-	-	22 hours	-	-	-	-

The screenshot shows the AWS IAM 'User: pgl_IAM_Ibrahim_Fayomi' details page. The left sidebar has 'Access Management' expanded, with 'Users' selected and highlighted with a red box. The 'Permissions' tab is active. The 'Add permissions' button is highlighted with a red box.

Summary

- ARN: arn:aws:iam::416508157702:user/pgl_IAM_Ibrahim_Fayomi
- Console access: Enabled without MFA
- Created: January 13, 2026, 17:19 (UTC-01:00)
- Last console sign-in: Never
- Access key 1: Create access key

Permissions

Permissions policies (0): Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

No resources to display

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

Generate policy

Identity and Access Management (IAM)

Summary

ARN: arn:aws:iam::416508157702:user/pgl_IAM_Ibrahim_Fayomi

Created: January 13, 2026, 17:19 (U/C: 01:00)

Console access: Enabled without MFA

Last console sign-in: Never

Access key 1: Create access key

Permissions | Groups (0) | Tags | Security credentials | Last Accessed

Permissions policies (0)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

No resources to display

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

Add permissions

Add inline policy

https://us-east-1.console.aws.amazon.com/iam/home?region=eu-north-1#users/details/pgl_IAM_Ibrahim_Fayomi/add-permissions

Permissions options

Add user to group: Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions: Copy all group membership, attached managed policies, inline policies, and any existing permissions boundaries from an existing user.

Attach policies directly: Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

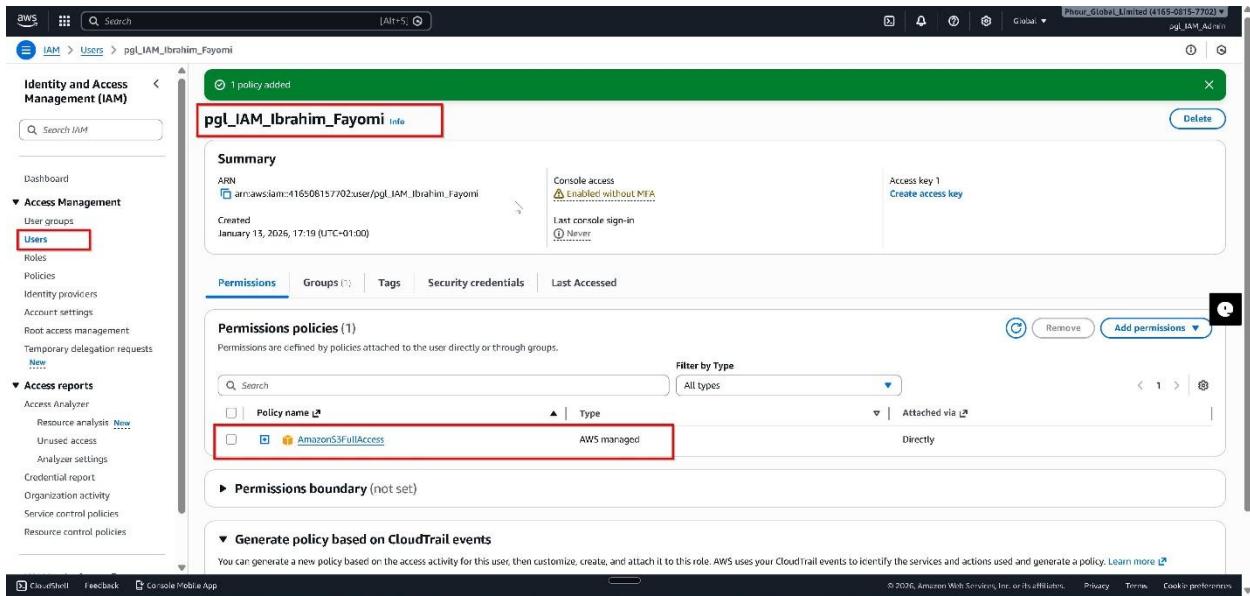
Permissions policies (1/1440)

Filter by Type: All types

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	0
AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	0
AmazonS3OutpostsFullAccess	AWS managed	0
AmazonS3OutpostsReadOnlyAccess	AWS managed	0
AmazonS3ReadOnlyAccess	AWS managed	0
AmazonS3TableFullAccess	AWS managed	0
AmazonS3TableLikeFormationServiceRole	AWS managed	0
AmazonS3TableLikeReadonlyAccess	AWS managed	0
AWSBackupServiceRolePolicyForS3Backup	AWS managed	0
AWSBackupServiceRolePolicyForS3Restore	AWS managed	0
AWSQuickSetupSSMDeploymentS3BucketRolePolicy	AWS managed	0

[CloudShell](#) [Feedback](#) [Console Mobile App](#)

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



These screenshots document assigning S3 permissions to a user. The IAM user configuration page shows the permissions tab where policies can be attached. The AmazonS3FullAccess managed policy is selected from the policy list. This policy grants complete access to all S3 buckets and objects in the account including create, read, update, and delete operations.

The policy attachment confirms the user now has full S3 access. This user can perform any S3 operation including creating buckets, uploading objects, modifying permissions, and deleting resources. Full access is appropriate for administrators but standard users should receive more restrictive permissions based on their specific needs.

Assigning S3 Read Only Access Policy to User

The screenshot shows the AWS IAM User Details page for the user 'pgl_IAM_Solanke_Hope'. The 'Permissions' tab is selected. At the top right of the permissions section, there is a button labeled 'Add permissions' with a red box around it.

The screenshot shows the 'Add permissions' step in the AWS IAM User Details page. The 'Attach policies directly' option is selected and highlighted with a red box. In the list of available policies, the 'AmazonS3ReadOnlyAccess' policy is selected and highlighted with a red box.

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, the navigation menu is visible with sections like 'Identity and Access Management (IAM)', 'Access Management', 'Access reports', and 'Access Analyzer'. The main content area is titled 'pgl_IAM_Solanke_Hope' and displays the user's ARN (arn:aws:iam::416508157702:user/pgl_IAM_Solanke_Hope), console access status (Enabled without MFA), and last sign-in information (January 13, 2026, 17:18 UTC-01:00). The 'Permissions' tab is selected, showing a table of attached policies. One policy, 'AmazonS3ReadOnlyAccess', is listed and highlighted with a red border. Other tabs include 'Groups', 'Tags', 'Security credentials', and 'Last Accessed'. At the bottom, there are links for CloudWatch, Feedback, Console/Tablet App, and a copyright notice.

The screenshots show attaching AmazonS3ReadOnlyAccess policy to a different user.

This managed policy allows listing buckets and reading objects but prevents any modifications.

The user can download files and view bucket configurations but cannot upload, delete, or change settings. Read-only access is appropriate for users who need to access data but should not modify it.

Restricting users to read-only access prevents accidental or malicious data deletion. This permission model supports compliance requirements for separation of duties where some users manage data while others only consume it.

Logged in as Read-Only User

The screenshot shows the AWS Console Home page. At the top right, the navigation bar displays the user's name "Pior_Global_Limited (4165-0815-7702)" and the region "Europe (Stockholm)". Below the navigation bar, the "Applications" section shows a message: "Access denied to servicelistCatalog:listApplications". The "Cost and usage" section also shows several "Access denied" messages under "Current month", "Forecasted month end", and "Savings opportunities". The left sidebar includes sections for "Recently visited", "Welcome to AWS", "AWS Health", and "Cost and usage". The bottom of the page includes standard links like CloudWatch, Feedback, and Console Mobile App.

The console shows successful login as the read-only user. The username in the navigation bar confirms the active session identity. This user has the AmazonS3ReadOnlyAccess policy attached and can view S3 resources but cannot make changes.

Verifying Read-Only Permission to S3 Bucket

The screenshot shows the AWS S3 Bucket Properties page for the bucket "pgl-internal-documents". The object "KARAMOT NOUN PROJECT.pdf" is selected. In the "Permissions" tab, the "Access control list (ACL)" section indicates that the bucket owner has enforced setting applied for Object Ownership. The "Grantee" table shows the following permissions:

Grantee	Object	Object ACL
Object owner (your AWS account)	Read	Read, Write
Everyone (public access)	-	-
Authenticated users group (anyone with an AWS account)	-	-

The bottom of the page includes standard links like CloudWatch, Feedback, and Console Mobile App.

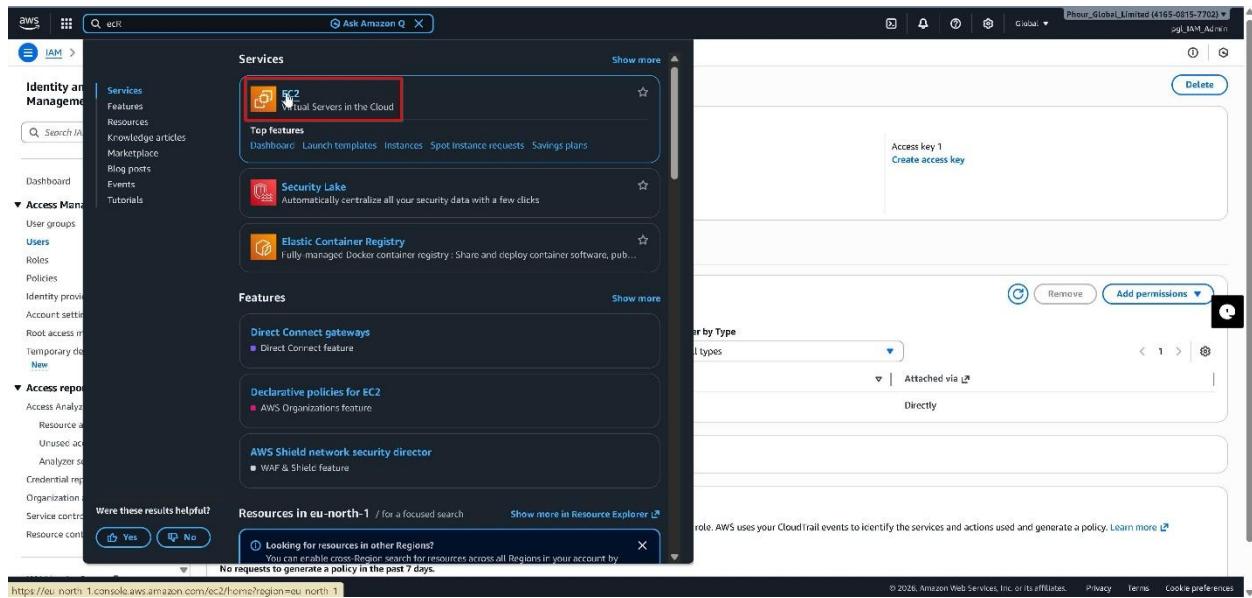
The screenshot shows the read-only user accessing the S3 bucket. The user can see bucket contents and download objects. The upload button is either disabled or will fail if clicked due to insufficient permissions. This demonstrates that the read-only policy is correctly enforced. Attempting to upload, delete, or modify objects will result in access denied errors.

Permission validation confirms that access controls work as intended. Testing actual operations verifies that policy statements produce the expected behavior. This validation step is critical to ensure security configurations function correctly.

EC2 INSTANCES

EC2 Instance Configuration

Accessing EC2 Service



The console shows navigation to EC2 service. EC2 provides resizable compute capacity in the cloud through virtual servers called instances. The service supports various instance types optimized for different workloads. EC2 instances can run multiple operating systems and are the foundation for many cloud applications.

EC2 Dashboard

The screenshot shows the AWS EC2 Dashboard for the Europe (Stockholm) Region. The left sidebar includes links for Dashboard, Instances, Images, Elastic Block Store, and Network & Security. The main area displays the following sections:

- Resources:** Shows 0 instances (running), 0 Auto Scaling Groups, 0 Capacity Reservations, 0 Dedicated Hosts, 0 Elastic IPs, 0 Instances, 0 Key pairs, 0 Load balancers, 0 Placement groups, 1 Security groups, and 0 Snapshots.
- Launch instance:** A button to start a new instance.
- Service health:** Shows the region as Europe (Stockholm) with a status of "This service is operating normally".
- Zones:** Lists availability zones: eu-north-1a (eu-n1-az1), eu-north-1b (eu-n1-az2), and eu-north-1c (eu-n1-az3).
- Instance alarms:** Shows 0 in alarm, 0 OK, and 0 insufficient data.
- Scheduled events:** Shows Europe (Stockholm) with no scheduled events.
- EC2 cost:** Shows a date range of "Past 6 months", a region of "Global", and credits remaining of "\$100 USD". Days remaining are "172 (July 12, 2026)".
- Account attributes:** Includes settings for Default VPC (vpc-0fb20a382fb87d381), Data protection, Allowed AMIs, and EC2 Serial Console.
- Explore AWS:** Promotes saving up to 90% on EC2 with Spot Instances and combining EC2 purchase options.

The EC2 dashboard displays an overview of compute resources. The interface shows running instances, available resources, and service health. Navigation options include instances, AMIs, security groups, key pairs, and elastic IPs. The launch instance button starts the instance creation wizard.

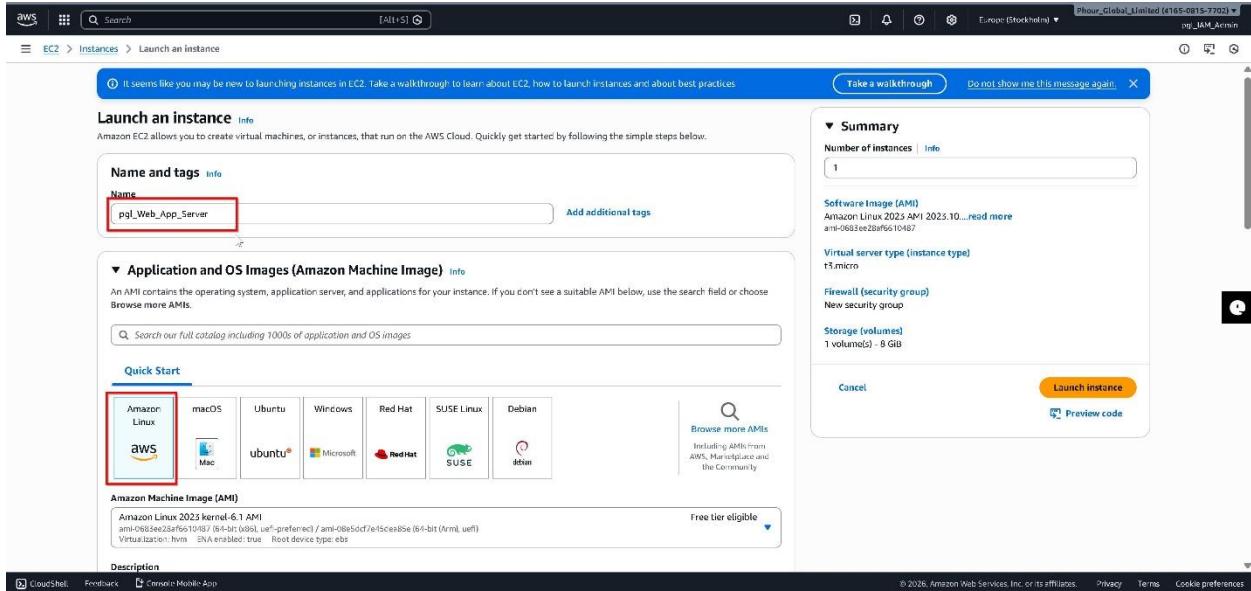
Creating EC2 Instance

The screenshot shows the AWS EC2 Instances page. On the left, a navigation sidebar lists various EC2 services: Dashboard, Instances (highlighted with a red box), Images, Elastic Block Store, Network & Security, and more. The main content area is titled 'Resources' and shows usage statistics for the Europe (Stockholm) Region. It includes sections for Launch instance, Service health, Zones, and Account attributes. A large central box is titled 'Launch instance' with the sub-instruction 'To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.' It features two prominent buttons: 'Launch instance' and 'Migrate a server'. Below these are sections for 'Instance alarms' and 'Scheduled events'. The bottom right of the main area contains a 'Explore AWS' section with a link to 'Save up to 90% on EC2 with Spot Instances'. At the very bottom of the page, there's a footer with links to 'CloudShell', 'Feedback', 'Console Mobile App', and copyright information.

This screenshot shows the same EC2 Instances page as above, but with a different URL in the address bar: <https://eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#instances>. The main content area now displays a message 'No instances' with the sub-instruction 'You do not have any instances in this region'. A large button labeled 'Launch instances' is highlighted with a red box. Below this, a section titled 'Select an instance' is visible. The rest of the interface and sidebar are identical to the first screenshot.

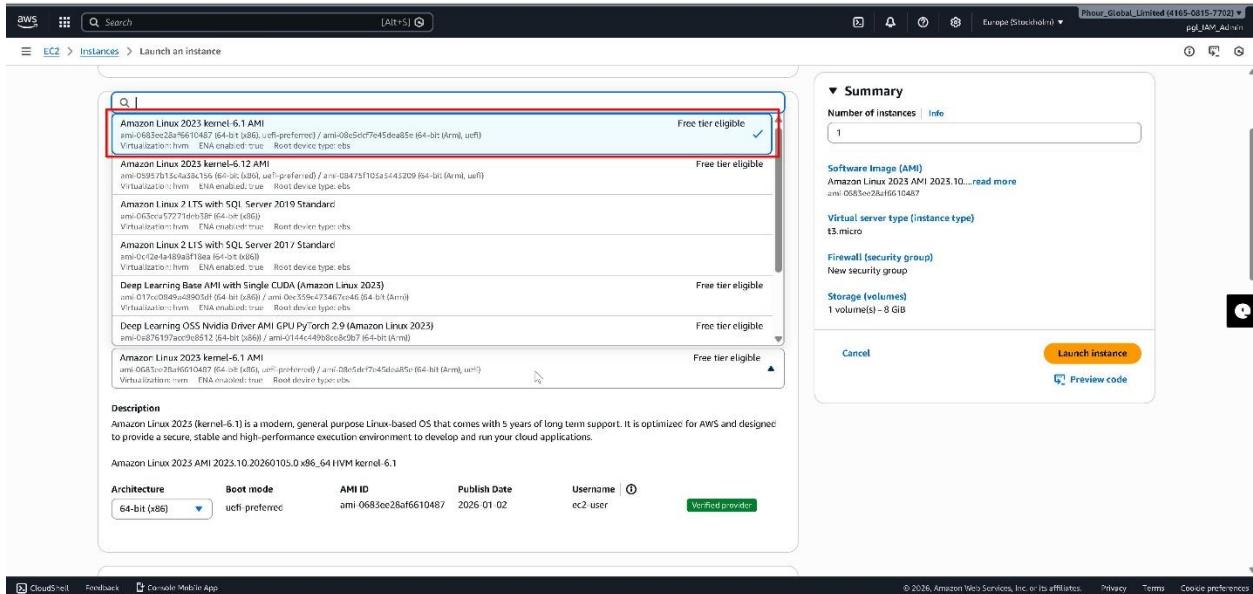
The launch instance wizard begins with instance configuration options. The interface provides a streamlined path to launch instances with common configurations. Advanced options allow detailed customization of network, storage, and security settings. The wizard guides through all required configuration steps before launch.

Naming Instance and Selecting Operating System



The screenshot shows naming the instance with a descriptive identifier such as Web App Server. Instance names help identify resources in the console and billing reports. The name is applied as a tag and does not affect technical configuration. Clear naming conventions simplify resource management in environments with many instances.

Choosing AMI and OS Version



The AMI selection screen displays available machine images. Amazon Linux 2 is selected which provides a secure, stable, and high-performance execution environment. The AMI includes AWS integration and optimization. Amazon Linux receives security updates and patches from AWS. The OS selection determines the software environment available on the instance.

Choosing Instance Type

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Instance type' section, the 't2.micro' configuration is selected and highlighted with a red box. This configuration includes 1 vCPU, 1 GB Memory, and Current Generation true. It is marked as 'Free tier eligible'. Other configurations listed include 't3.micro', 't3.small', and 'c7i.Rio large'. In the 'Summary' panel on the right, it shows the selected AMI (Amazon Linux 2023.10), Virtual server type (t3.micro), Firewall (New security group), and Storage (1 volume(s) - 8 GiB). The 'Launch instance' button is visible at the bottom.

The instance type selection shows available configurations with varying CPU, memory, storage, and network capacity. The t2.micro instance type is selected which provides 1 vCPU and 1 GB of memory. This instance type is eligible for AWS free tier. Instance type selection should match workload requirements balancing performance needs with cost.

Creating Key Pair for Encryption

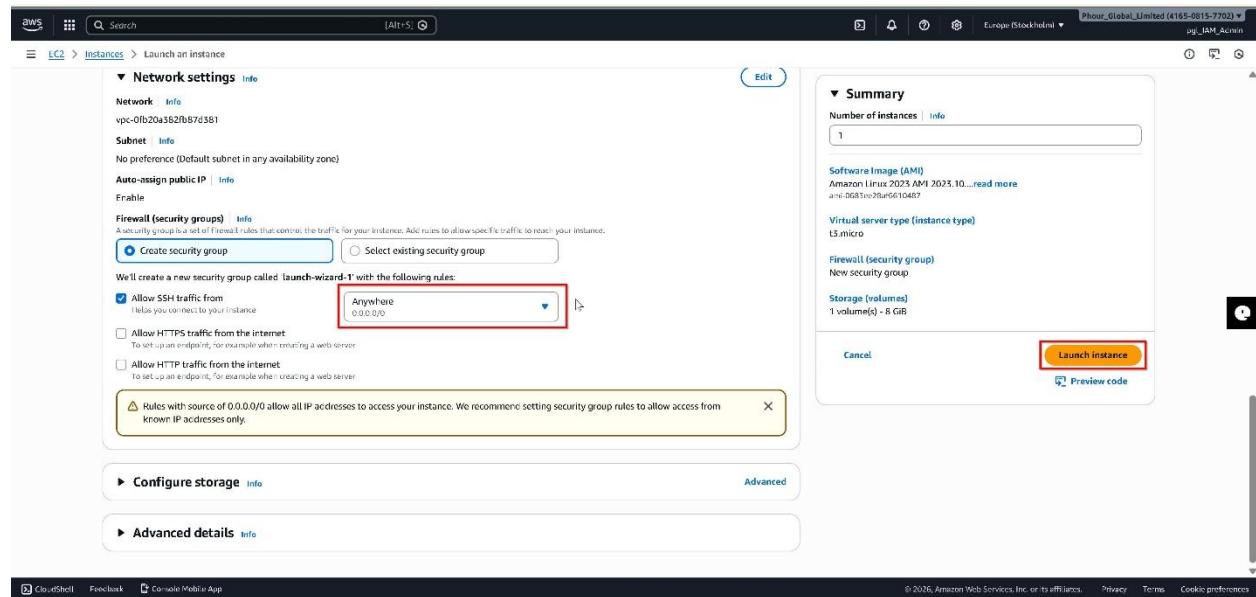
The screenshot shows the 'Launch instance' wizard in the AWS EC2 console. The current step is 'Key pair (login)'. It displays a dropdown menu for selecting an existing key pair and a prominent red-bordered button labeled 'Create new key pair'. Other sections visible include 'Instance type' (t3.micro), 'Network settings', and 'Summary' where the instance is set to launch.

This screenshot shows the 'Create key pair' dialog box overlaid on the 'Launch instance' wizard. The 'Key pair name' field contains 'pgl_WebApp_Keys'. The 'Key pair type' section has 'RSA' selected, indicated by a red border. Below it, 'Private key file format' offers 'pem' (selected) and 'ppk' options. A warning message at the bottom of the dialog states: 'When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. Learn more'.

The key pair configuration prompts for creating or selecting an SSH key pair. A new key pair is created with a descriptive name. The RSA encryption type is selected which provides asymmetric encryption for SSH authentication. The private key file is downloaded and must be stored securely as it cannot be retrieved later. The public key is stored in AWS and installed on the instance.

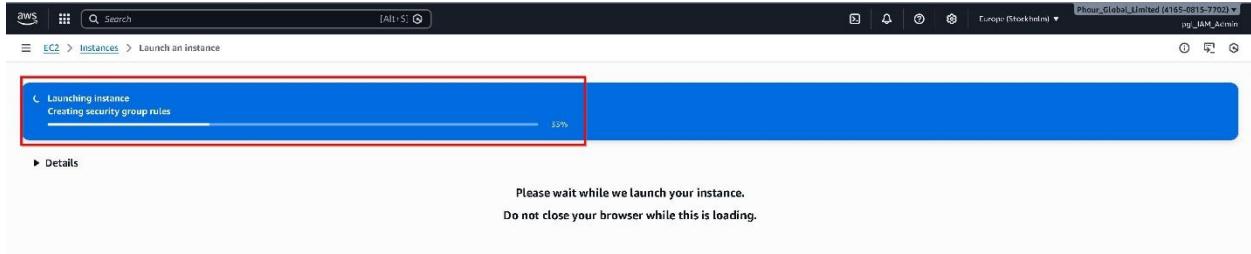
Key pair authentication provides more secure access than password authentication. The private key must be protected as anyone with access to it can connect to the instance. Lost private keys cannot be recovered and require creating a new key pair and updating instance access.

Choosing Network Access and Launching Instance



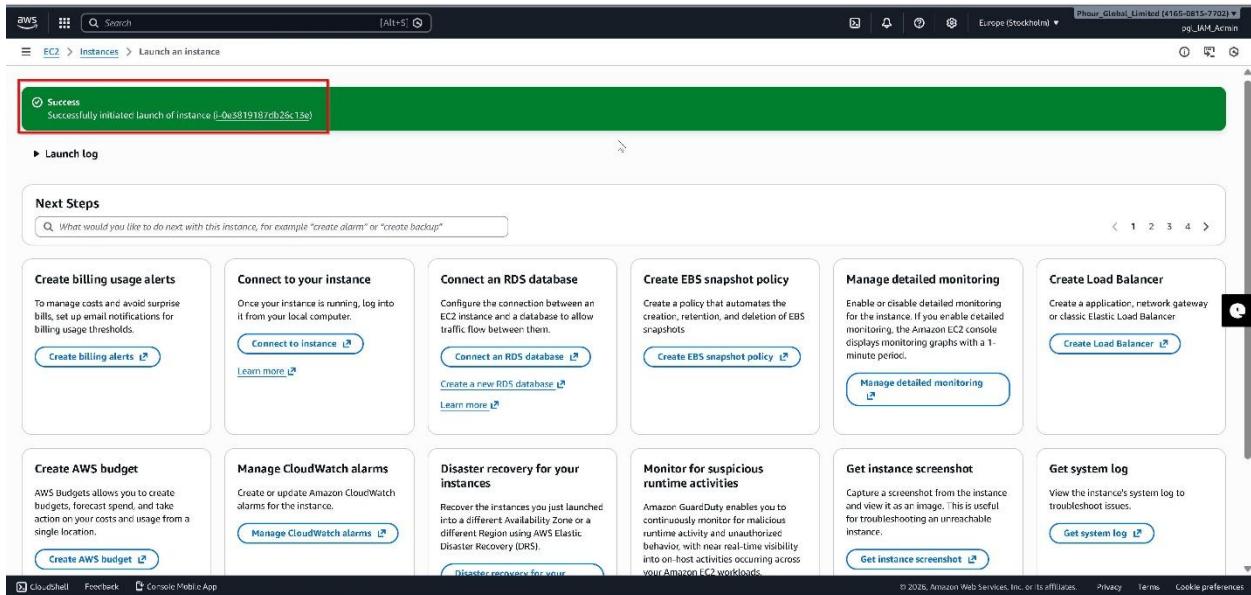
The network settings section configures security group rules controlling inbound traffic. The option to allow SSH access from anywhere is selected which permits connections from any IP address. This provides maximum accessibility but reduces security. Production instances should restrict SSH access to specific IP ranges or use bastion hosts. The launch instance button initiates instance creation with all configured settings.

Instance Launch Loading



The progress indicator shows instance launch in progress. AWS allocates compute resources, attaches networking, and initializes the operating system. Launch typically completes within a few minutes. The instance will appear in the instances list once initialization finishes.

Instance Launch Successful



The success message confirms the instance has been launched. The instance ID is displayed which uniquely identifies this compute resource. The console provides a link to view the instance details. The instance enters a running state after passing status checks.

Instance Running

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, AWS Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, and VPCs. The main area is titled 'Instances (1) Info' and shows a table with one row. The row contains the following information: Name (pgl_Web_App...), Instance ID (i-0e3819187db26c13e), Instance state (Running), Instance type (t3.micro), Status check (3/3 checks passed), Alarm status (View alarms), Availability Zone (eu-north-1b), Public IPv4 DNS (ec2-13-60-174-110.eu...), and Public IPv4 address (13.60.174.110). A red box highlights the entire row.

This screenshot shows the same EC2 Instances page, but the instance row is now selected. The main area displays detailed information for the instance 'i-0e3819187db26c13e (pgl_Web_App_Server)'. Under the 'Details' tab, several fields are highlighted with red boxes: Instance ID (i-0e3819187db26c13e), Public IPv4 address (13.60.174.110), Private IP address (172.31.44.15), Instance state (Running), Private IP DNS name (ip-172-31-44-15.eu-north-1.compute.internal), and VPC ID (vpc-0fb20a382fb87d361). Other tabs include Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The bottom of the page includes links for CloudWatch, Feedback, and Console Media App, along with copyright and legal information.

The instances view shows the Web App Server instance in running state. Instance details display public and private IP addresses, instance type, security groups, and monitoring status. The instance is now accessible via SSH using the private key. The public IP address allows connection from the internet while the private IP is used for internal VPC communication.

Creating Custom EC2 Policy for User Ibrahim

The screenshot shows the AWS IAM User Details page for a user named 'pgI_AM_Ibrahim_Fayomi'. The 'Permissions' tab is active. In the top right corner of the 'Permissions policies' section, there is a red box highlighting the 'Add permissions' button. This button is part of a dropdown menu that includes options like 'Add permissions' and 'Create inline policy'.

The screenshot shows starting the process to create custom permissions for user Ibrahim.

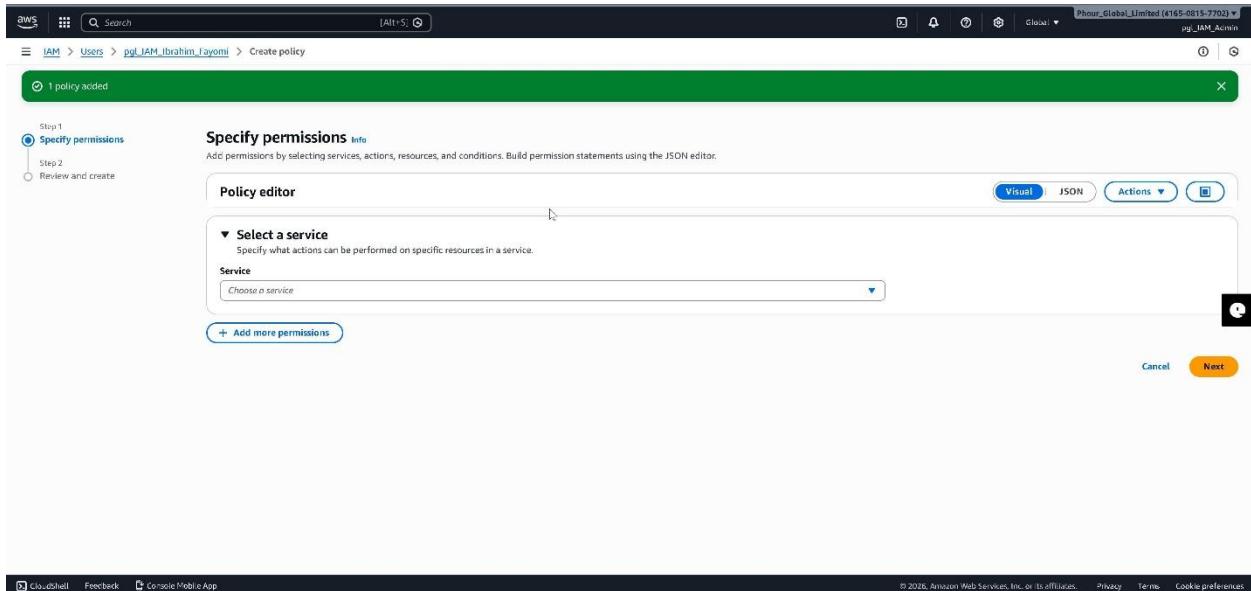
Rather than using AWS managed policies, a custom policy will be created to grant specific EC2 permissions. This demonstrates fine-grained access control allowing only necessary actions on particular resources.

Examining Predefined EC2 Permissions

The screenshot shows the AWS IAM 'Add permissions' interface. The top navigation bar includes 'Search' and 'Global' dropdowns. The main title is 'Add permissions' under 'Step 1'. Below it, there are three options: 'Add user to group' (selected), 'Copy permissions', and 'Attach policies directly' (highlighted with a red box). The 'Permissions policies' section lists three policies: 'AmazonEC2ReadOnlyAccess' (selected and highlighted with a red box), 'AWSLambdaManagedEC2ResourceOperator', and 'DeclarativePoliciesEC2Report'. The 'AmazonEC2ReadOnlyAccess' row has a checked checkbox and a 'Details' button. A 'Filter by Type' dropdown shows 'All types' with '3 matches'. At the bottom right are 'Cancel' and 'Next' buttons.

The interface displays AWS managed policies for EC2 showing the level of access each policy provides. These predefined policies cover common use cases but may grant more permissions than needed. Examining managed policies helps understand available options before creating custom policies for specific requirements.

Creating Policy from Scratch



The policy creation screen offers options to build policies using visual editor or JSON.

The JSON option is selected to use the AWS Policy Generator tool. Custom policies allow precise control over which actions are allowed on which resources. Policy statements define service, actions, resources, and conditions.

Accessing AWS Policy Generator

The screenshot shows the AWS Policy Generator interface. It consists of three main sections:

- Step 1: Select policy type**: A dropdown menu titled "Type of Policy" is open, showing "IAM Policy" as the selected option.
- Step 2: Add statement(s)**: This section includes fields for "Effect" (set to "Allow"), "AWS" (with a checkbox for "All Services (*)" and a dropdown menu for "Select Service"), and "Add conditions (optional)". A button labeled "Add Statement" is at the bottom.
- Step 3: Generate policy**: A large, empty text area where the generated JSON policy will be displayed.

The AWS Policy Generator tool provides a structured interface for creating policy JSON.

The tool guides through selecting service, actions, and resources to build policy statements.

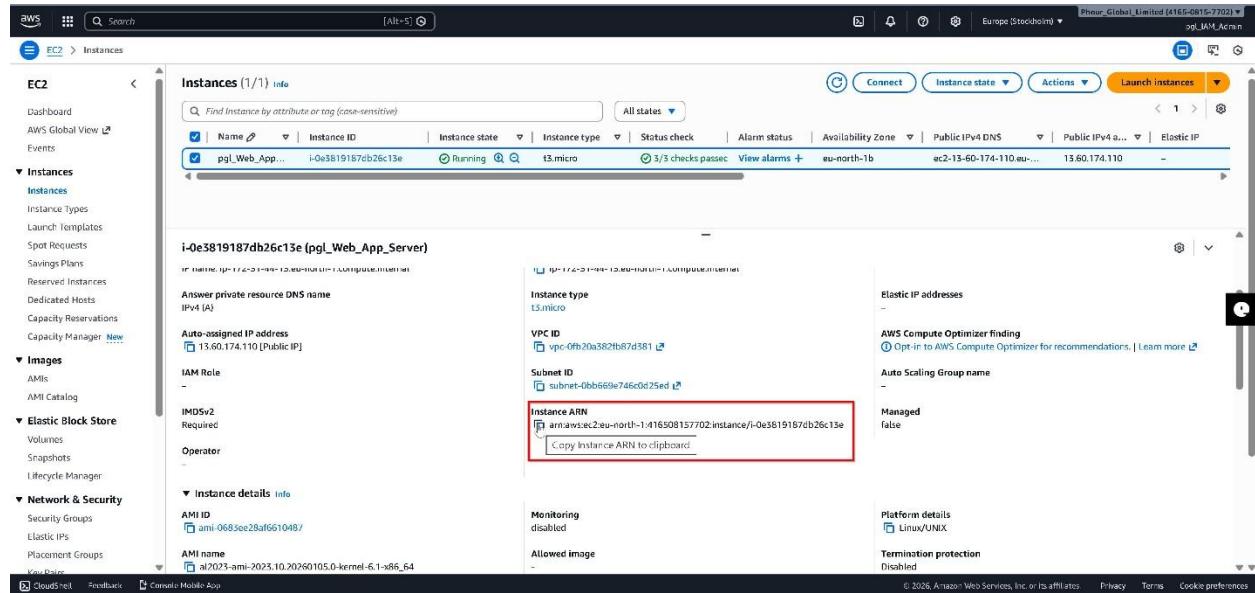
Generated JSON can be copied into IAM console for policy creation. The generator prevents syntax errors and ensures valid policy format.

Configuring Policy in AWS Policy Generator

The screenshot shows the AWS Policy Generator interface. In Step 1: Select policy type, 'IAM Policy' is selected. In Step 2: Add statement(s), the 'Effect' is set to 'Deny'. The 'AWS' dropdown shows 'Amazon EC2' selected. The 'Actions' dropdown lists several EC2 actions: 'DeleteKeyPair', 'TerminateInstances', 'StopInstances', and 'CreateKeyPair'. These four actions are highlighted with a red border.

The policy generator form shows selecting Amazon EC2 as the service. The effect field determines whether actions are allowed or denied. Actions dropdown lists all available EC2 operations. Multiple actions can be selected for a single policy statement. The configuration builds the policy logic that will be converted to JSON.

Getting ARN from EC2 Instance



The instance details display the Amazon Resource Name (ARN) which uniquely identifies the EC2 instance. The ARN format includes account ID, region, and instance ID. Copying the instance ARN allows creating policies that apply only to this specific instance. Resource-specific policies implement least privilege by limiting actions to particular resources rather than all resources of a type.

Completing Policy Configuration with ARN

The screenshot shows the AWS Policy Generator interface. At the top, there is a 'Deny' button. Below it, under 'AWS', 'Amazon EC2' is selected from a dropdown menu. Under 'Actions', 'DeleteKeyPair', 'TerminateInstances', 'StopInstances', and 'CreateKeyPair' are selected. In the 'Amazon Resource Name (ARN)' section, 'arn:aws:ec2:eu-north-1:416508157702:instance/i-0e3819187db26c13e' is entered into a text input field. A note below the ARN field specifies the ARN format: 'ARN should follow the following format: arn:aws:ec2:<Region>:<Account>:resourceType/<ResourceName>. Use a comma to separate multiple values.' At the bottom of the configuration area, there is a 'Generate Policy' button.

Step 3: Generate policy
A policy is a document (written in the Access Policy Language [\(A\)](#)) that acts as a container for one or more statements.

©2026, Amazon Web Services or its affiliates. All rights reserved.

The ARN field in the policy generator is populated with the instance ARN. This restricts the policy to apply only to the specified instance. Actions like Describe Instances, Stop Instances, and Terminate Instances are selected to give Ibrahim specific control over this instance. The policy will not allow actions on other EC2 instances.

Statement Added to Policy

The screenshot shows the AWS Policy Generator interface at Step 3: Generate policy. The page title is "awspolicygen.s3.amazonaws.com/policygen.html". The main content area displays a table titled "Statements added (1)". The table has columns: Effect, Action, Resource(s), Condition(s), and Remove. There is one row with the following data:

Effect	Action	Resource(s)	Condition(s)	Remove
Deny	ec2:DeleteKeyPair ec2:TerminateInstances ec2:StopInstances ec2>CreateKeyPair	arn:aws:ec2:eu-north-1:416508157702:instance/i-0e3819187db26c13e	None	Remove

Below the table, there is a section titled "Step 3: Generate policy" with the sub-instruction: "A policy is a document (written in the Access Policy Language [↗](#)) that acts as a container for one or more statements." A yellow "Generate Policy" button is visible. At the bottom of the page, there is a legal notice and a copyright notice: "This AWS Policy Generator is provided for informational purposes only. You are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided AS IS without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies." and "©2026, Amazon Web Services or its affiliates. All rights reserved."

The policy generator displays the completed statement before generating JSON. The statement shows the configured service, actions, resource ARN, and effect. Multiple statements can be added to create comprehensive policies. Each statement addresses a specific permission requirement.

Generated JSON Policy

The screenshot shows the AWS Policy Generator interface. On the left, there's a sidebar with options like 'Deny', 'All Services ("")', 'Select Service...', and 'Add conditions (optional)'. Below that is a section titled 'Statements added (1)' with a note: 'You added the following statements. Click the table header to sort by Effect or Action.' The table shows one statement:

Effect	Action
Deny	ec2:DeleteKeyPair ec2:TerminateInstances ec2:StopInstances ec2:CreateKeyPair

At the bottom of the sidebar is a 'Generate Policy' button.

The main area is titled 'Policy JSON Document' and contains the following JSON code:

```
1 * []
2 "Version": "2012-10-17",
3 * "Statement": [
4 *   {
5     "Sid": "Statement1",
6     "Effect": "Deny",
7     "Action": [
8       "ec2:DeleteKeyPair",
9       "ec2:TerminateInstances",
10      "ec2:StopInstances",
11      "ec2:CreateKeyPair"
12    ],
13    "Resource": "arn:aws:ec2:eu-north-1:416508157702:instance/*-0e3019187db26c13e"
14  }
15 ]
16 }
```

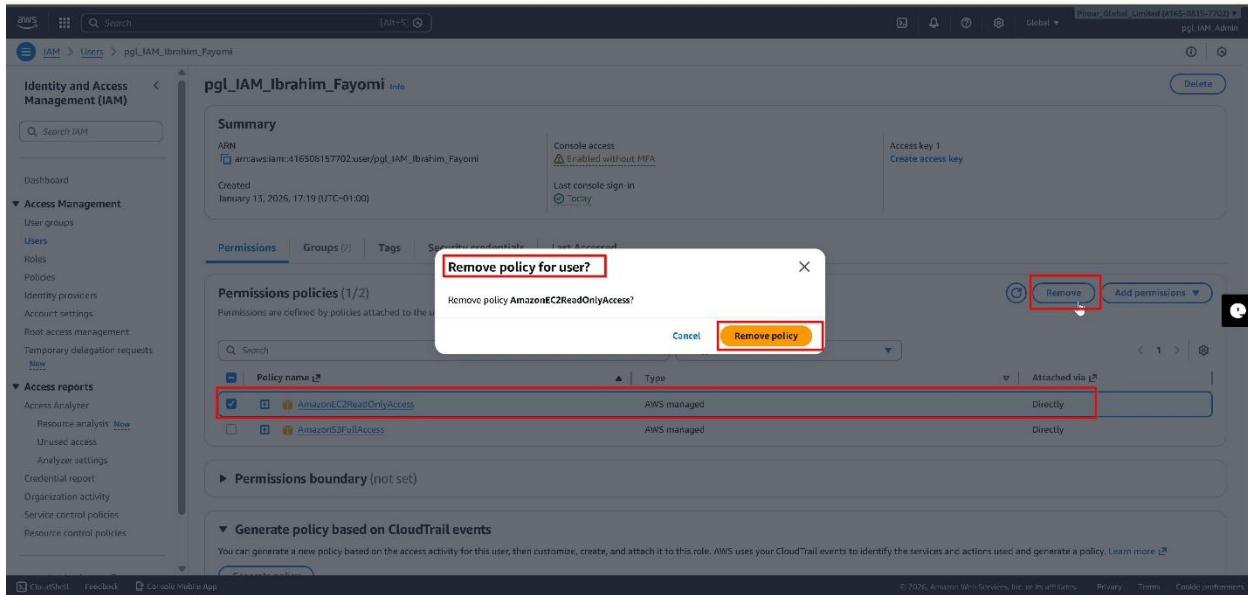
A red box highlights the JSON code. To the right of the code is a '1:1 JSON' button. At the bottom of the main area is a note: 'This AWS Policy Generator is provided for informational purposes only; you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided as is without warranty of any kind, whether express, implied or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.'

At the very bottom are 'Close' and 'Copy Policy' buttons.

The policy generator produces properly formatted JSON containing the policy statement.

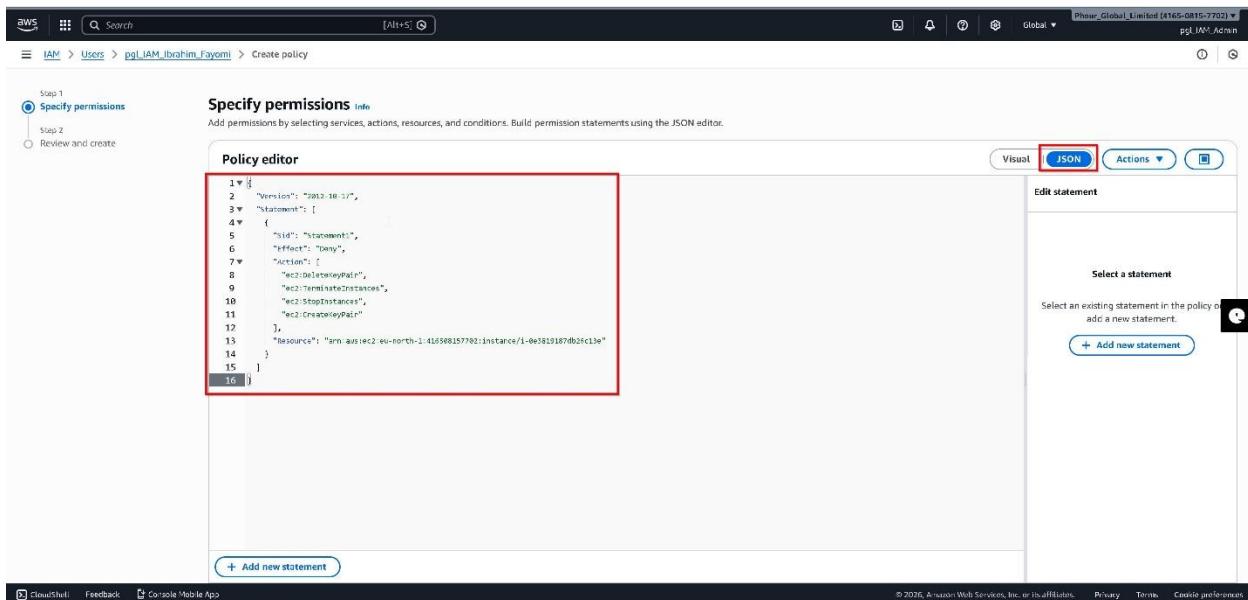
The JSON includes version, statement array, effect, actions list, and resource ARN. This JSON can be copied directly into IAM policy creation. The generated format follows IAM policy syntax requirements ensuring the policy will be accepted.

Deleting Predefined Policy from User



The screenshot shows removing any existing EC2 managed policies from Ibrahim before adding the custom policy. This ensures the user has only the intended permissions defined in the custom policy. Removing excess permissions maintains least privilege access.

Pasting Generated Policy JSON



The JSON editor in IAM shows the generated policy pasted from the AWS Policy Generator. The JSON is validated to ensure correct syntax. The policy defines specific EC2 actions that Ibrahim can perform on the designated instance. Any syntax errors are flagged before policy creation proceeds.

Naming and Creating Custom Policy

Policy details

Policy name
Enter a meaningful name to identify this policy.
pgl_Ibrahim_EC2_policy

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose Show remaining.

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Service	Access level	Resource	Request condition
EC2	Limited: Write	region string like [eu-north-1, InstanceID string like] 0e3819187db26c13e	None

Allow (0 of 461 services)

Service	Access level	Resource	Request condition
No resources to display			

Show remaining 460 services

Cancel Previous **Create policy**

The policy name and description fields are populated with clear identifiers. The policy name should indicate its purpose such as Ibrahim-EC2-InstanceControl. Description explains what permissions the policy grants. Clear naming and documentation help maintain policies as requirements evolve. The create policy button finalizes the custom policy.

Policy Created and Added to User

The screenshot shows the AWS IAM 'Users' section. A user named 'pgl_IAM_Ibrahim_Fayomi' is selected. A green callout box highlights the message 'Policy pgl_Ibrahim_EC2_policy created.' above the user's name. The 'Permissions' tab is selected, showing a list of policies attached to the user. Two policies are listed: 'AmazonS3FullAccess' (AWS managed) and 'pgl_ibrahim_EC2_policy' (Customer inline). The 'Customer inline' policy is highlighted with a red box.

The user permissions list now shows the newly created custom policy attached to Ibrahim. The policy is active and will be enforced immediately. Ibrahim can now perform the allowed EC2 actions on the specified instance but has no other EC2 permissions. The custom policy provides precisely the required access without excess privileges.

Signed in as User Ibrahim with Policy Enforced

The screenshot shows the AWS EC2 'Instances' section. A user named 'pgl_IAM_Ibrahim_Fayomi' is signed in. A red box highlights a denied message: 'You are not authorized to perform this operation. User: arn:aws:iam:416508157702:user/pgl_IAM_Ibrahim_Fayomi is not authorized to perform: ec2:DescribeInstances because no identity-based policy allows the ec2:DescribeInstances action'. This message is displayed in a modal window.

The console displays successful authentication as user Ibrahim. The username confirmation shows the active session. Ibrahim now operates under the constraints of the custom EC2 policy. Any actions outside the policy scope will be denied with insufficient permissions errors.

Stopping EC2 Instance

The screenshot shows the AWS EC2 Instances page. A single instance, "pgl_Web_App..." (instance ID: i-0e3819187db26c15e), is listed. The instance is running and is of type t3.micro. The Actions menu is open, and the "Stop instance" option is highlighted with a red box.

The screenshot shows the "Stop instance" confirmation dialog box. It displays the instance ID "i-0e3819187db26c15e (pgl_Web_App_Server)". The "Stop protection" dropdown is set to "Disabled". A warning message states: "You will be billed for associated resources. After you stop the instance, you are no longer charged usage or data transfer fees for it. However, you will still be billed for associated Elastic IP addresses and EBS volumes." The "Stop" button is highlighted with a red box.

The screenshots show Ibrahim accessing the EC2 console and stopping the instance. The custom policy allows this action on the specified instance. The stop action is selected from the instance state menu. Stopping releases compute resources while preserving instance configuration and attached storage. The instance state transitions from running to stopping to stopped.

Terminating EC2 Instance

This screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with various EC2-related options like Dashboard, Instance Types, Launch Templates, and Images. The main area shows a table of instances. One instance, named 'pgl_Web_App...', is selected and highlighted with a red box. A context menu is open over this instance, with the 'Actions' dropdown expanded. The 'Terminate (delete) instance' option is highlighted with a red box. A tooltip for this option states: 'This action is available when the instance is running, pending, or stopping.' Below the table, the details for the selected instance are shown, including its ID, state (Stopped), and network information.

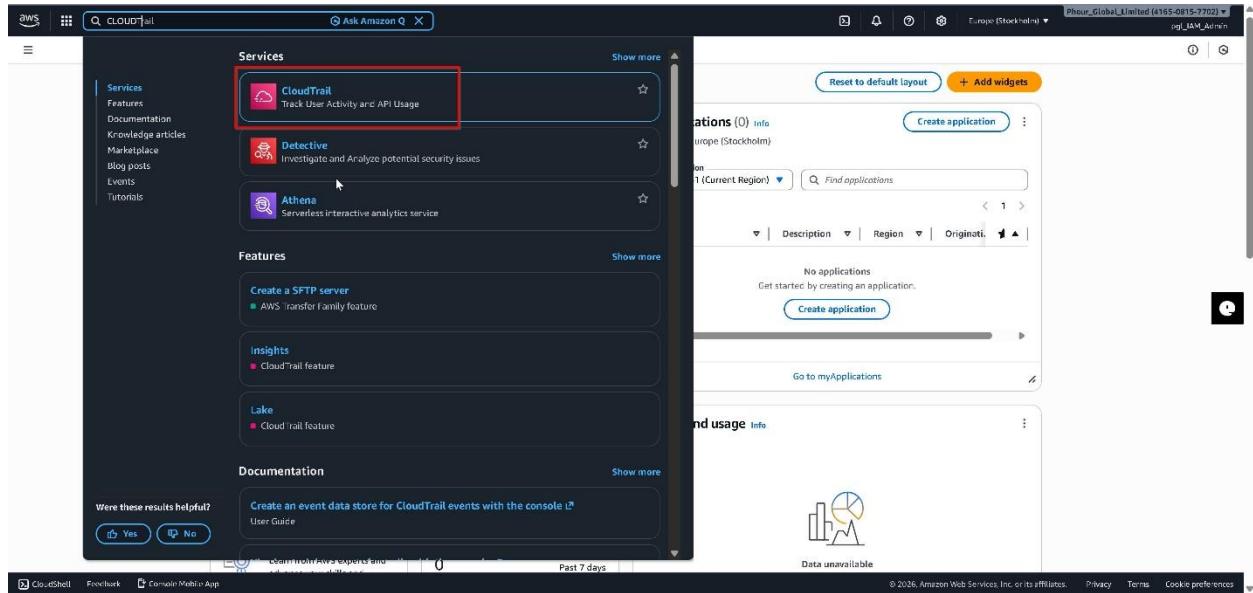
This screenshot shows a confirmation dialog titled 'Terminate (delete) instance'. It contains a warning message: 'On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.' Below this, it asks 'Are you sure you want to terminate these instances?'. It shows the instance ID 'i-0e3819187db26c13e (pgl_Web_App_Server)' and the 'Termination protection' status, which is 'Disabled'. There's also a note about skipping OS shutdown. At the bottom right of the dialog, the 'Terminate (delete)' button is highlighted with a red box.

The instance termination process is initiated. Terminating permanently deletes the instance and its associated resources. The custom policy allows Ibrahim to perform this destructive action on the designated instance. A confirmation prompt prevents accidental termination. Once terminated, the instance cannot be recovered and will be removed from the instance list.

CLOUDTRAIL SERVICES

CloudTrail Configuration

Accessing CloudTrail Service



The console shows navigation to CloudTrail service. CloudTrail records AWS API calls and related events for the account. The service provides audit trails for compliance, security analysis, and troubleshooting. CloudTrail logs capture who made requests, what services were accessed, when actions occurred, and what resources were affected.

Event History and Trails

The screenshot shows the AWS CloudTrail dashboard. On the left, there's a navigation sidebar with options like Dashboard, Event history, Insights, Lake, Dashboards, Query, Event data stores, Integrations, and Trails. The 'Trails' option is highlighted with a red box. The main content area has three main sections: 'Query results history' (which is currently empty), 'CloudTrail Insights' (also currently empty), and 'Event history'. The 'Event history' section displays two recent events:

Event name	Event time	Event source
TerminateInstances	January 21, 2026, 16:49:44 (UTC...)	ec2.amazonaws.com
TerminateInstances	January 21, 2026, 16:47:55 (UTC...)	ec2.amazonaws.com

The CloudTrail dashboard displays event history and configured trails. Event history shows the last 90 days of management events at no additional cost. Trails provide ongoing logging with extended retention by storing logs in S3. The interface shows options to view events and create trails.

Event History Showing All Activities

The screenshot shows the AWS CloudTrail console with the 'Event history' tab selected. The main content area displays a table of 29 events. The columns are: Event name, Event time, User name, Event source, Resource type, and Resource name. The events listed include various API calls such as 'TerminateInstances', 'StopInstances', 'Console.Login', and 'CreateSecurityGroup'. The table has a red border around it. At the bottom left, it says '0 / 5 events selected'. At the bottom right, there are links for 'https://eu-north-1.console.aws.amazon.com/cloudtrailv2/home?region=eu-north-1/events', '© 2026, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Event name	Event time	User name	Event source	Resource type	Resource name
TerminateInstances	January 21, 2026, 15:49:44 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:Instance	i-0e3819187db25c13e
TerminateInstances	January 21, 2026, 15:47:35 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:Instance	i-0e3819187db25c13e
StopInstances	January 21, 2026, 15:43:07 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:Instance	i-0e3819187db25c13e
Console.Login	January 21, 2026, 15:38:58 (UTC...)	pgl_IAM_Admin	signin.amazonaws.com	-	-
RegisterManagedInst...	January 21, 2026, 15:01:11 (UTC...)	i-0e3819187db26c...	ssm.amazonaws.com	-	-
SharedSnapshotVolu...	January 21, 2026, 15:01:00 (UTC...)	-	ec2.amazonaws.com	-	-
RunInstances	January 21, 2026, 15:00:58 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:VPC, AWS:E...	vpc-0fb20a382fb87c381, ami-0683ee28af6610467, eni-0073532e880824523, i...
AuthorizeSecurityGro...	January 21, 2026, 15:00:56 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:SecurityGroup	sg-0f9a31fbaca802eb4
CreateSecurityGroup	January 21, 2026, 15:00:54 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:VPC, AWS:E...	vpc-0fb20a382fb87c381, launch-wizard-1, sg-0f9a31fbaca802eb4
CreateKeyPair	January 21, 2026, 14:26:51 (UTC...)	pgl_IAM_Admin	ec2.amazonaws.com	AWS:EC2:KeyPair	pgl_WebApp_Keys
Console.Login	January 21, 2026, 12:55:07 (UTC...)	pgl_IAM_Admin	signin.amazonaws.com	-	-
Console.Login	January 21, 2026, 12:48:49 (UTC...)	pgl_IAM_Ibrahim_...	signin.amazonaws.com	-	-

The event history view displays a chronological list of API calls. Each event shows event name, user identity, event time, and resource name. Events can be filtered by various attributes such as user name, resource type, or event name. The detail view for each event provides complete request and response information. This audit trail supports security investigations and compliance reporting.

Creating CloudTrail Trail

The screenshot shows the AWS CloudTrail Dashboard. On the left, there's a sidebar with links like Dashboard, Insights, Lake, Pricing, Documentation, Forums, and FAQs. The main area has sections for 'Query results history' (with a 'Create a new query' button) and 'CloudTrail Insights' (with a 'Create a trail' button). Below these is the 'Event history' section, which lists two events: 'TerminateInstances' on January 21, 2026, at 16:49:44 UTC from ec2.amazonaws.com. At the bottom right of the dashboard, there's a 'Create trail' button.

This screenshot shows the 'Choose trail attributes' step of the 'Create trail' wizard. It includes sections for 'General details' (trail name: 'pgl_centralised_logs'), 'Storage location' (option to 'Create new S3 bucket' selected), 'Trail log bucket and folder' (log prefix: 'aws_pgl_cLOUDTRAIL_log_416508157202_21e8810'), 'Log file SSE-KMS encryption' (Enabled checked), and 'AWS KMS alias' ('pgl_trail_encryption' selected). The 'Create new S3 bucket' field is highlighted with a red box.

Additional settings

- Log file validation** Info
Enabled
- SNS notification delivery** Info
Disabled

CloudWatch Logs - optional

Configure CloudWatch Logs to monitor your trail logs and notify you when specific activity occurs. Standard CloudWatch and CloudWatch Logs charges apply. Learn more [?!](#)

CloudWatch Logs Info
 Enabled

Log group info
 New
 Existing
View

Log group name
aws-cloudtrail-logs-416508157702-dac0faa3
1-512 characters. Only letters, numbers, dashes, underscores, forward slashes, and periods are allowed.

IAM Role info
AWS Cloud Trail assumes this role to send Cloud Trail events to your CloudWatch Log log group.
 New
 Existing
View

Role name
CloudTrailRoleForCloudWatchLogs_[trail-name]

Policy document

Step 1
 Choose trail attributes

Step 2
 Choose log events

Step 3
 Review and create

Choose log events

Events Info
 Record API activity for individual resources, or for all current and future resources in AWS account. Additional charges apply [?!](#)

Event type
 Choose the type of events that you want to log:
 Management events
 Capture management operations performed on your AWS resources

Data events
 Log the resource operations performed on or within a resource.

Insights events
 Identify unusual activity, errors, or user behavior in your account.

Network activity events
 Network activity events provide information about resource operations performed on a resource within a virtual private cloud endpoint.

Management events Info
 Management events show information about management operations performed on resources in your AWS account.

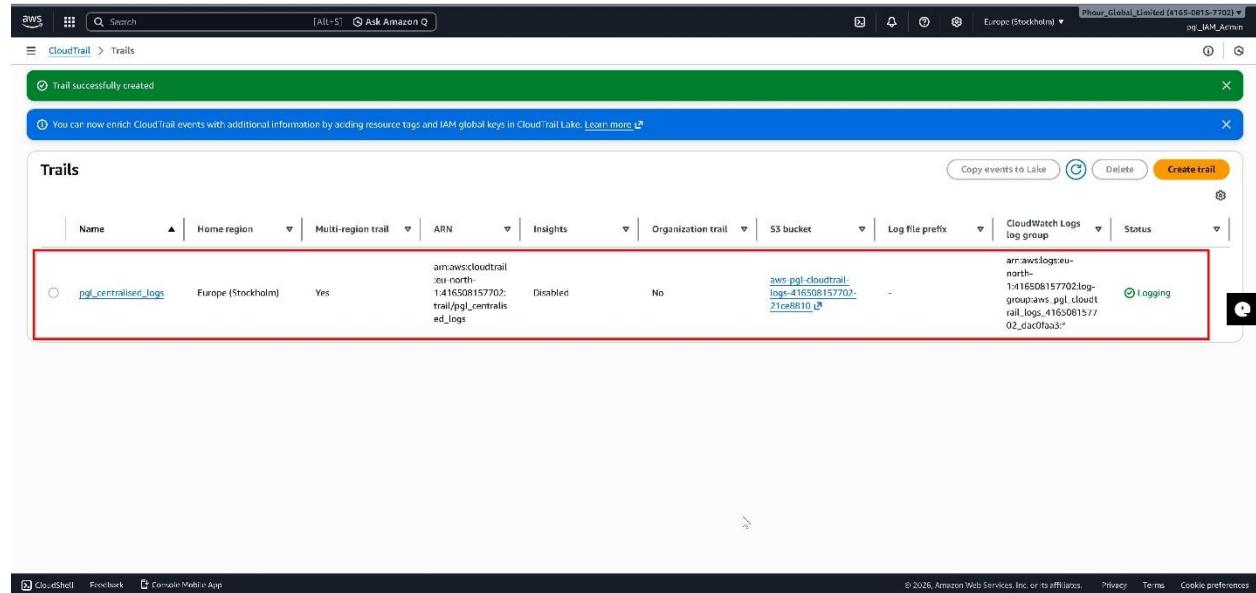
No additional charges apply to log management events on this trail because this is your first copy of management events.

API activity
 Choose the actions you want to log:
 Read
 Write
 Exclude AWS KMS events
 Exclude Amazon RDS Data API events

The trail creation wizard guides through configuration options. Trail name is specified along with log storage settings. The trail can apply to all regions or a single region. Management events record control plane operations like creating resources. Data events record resource operations like S3 object access. A new S3 bucket is created automatically to store trail logs. Log file validation ensures log integrity by detecting tampering.

Creating a trail enables continuous monitoring of account activity beyond the 90-day event history window. Logs stored in S3 can be retained indefinitely and analyzed using various tools. Trail configuration determines what events are logged and where they are stored.

Trail Created Successfully



The screenshot shows the AWS CloudTrail Trails page. At the top, there is a green banner stating "Trail successfully created". Below it, a blue banner says "You can now enrich CloudTrail events with additional information by adding resource tags and IAM global keys in CloudTrail Lake. Learn more." The main table lists a single trail named "pgl_centralised_logs". The table columns include Name, Home region, Multi-region trail, ARN, Insights, Organization trail, S3 bucket, Log file prefix, CloudWatch Logs log group, and Status. The "Status" column shows "Yes" for the trail. The "Log file prefix" column contains the value "aws-cloudtrail-log-416508157702-21ce8810". The "CloudWatch Logs log group" column contains the value "arn:aws:logs:eu-north-1:116508157702:log-group:aws.pgl.cloudtrail.logs.416508157702_dua0fa3.". A red box highlights the "CloudWatch Logs log group" value. On the right side of the table, there are buttons for "Copy events to Lake", "Delete", and "Create trail".

The trails list displays the newly created trail in active status. The trail is now logging events to the specified S3 bucket. CloudTrail typically delivers log files within 15 minutes of API calls. The trail can be modified or deleted as needed. Multiple trails can be configured to separate different types of events or send logs to different destinations.

CloudTrail S3 Bucket

The screenshot shows the AWS S3 console interface. The left sidebar is collapsed, and the main area displays the contents of the S3 bucket '416508157702/'. The bucket contains two objects: 'CloudTrail-Digest/' and 'CloudTrail/'. The 'CloudTrail-Digest/' folder is highlighted with a red box. The table header includes columns for Name, Type, Last modified, Size, and Storage class. The 'Actions' button is visible at the top right of the object list.

Name	Type	Last modified	Size	Storage class
CloudTrail-Digest/	Folder	-	-	-
CloudTrail/	Folder	-	-	-

The S3 bucket list shows the CloudTrail bucket created automatically during trail setup.

The bucket name follows CloudTrail naming convention including account ID and region.

CloudTrail log files are organized by date and region within the bucket. Access to this bucket should be restricted to prevent log tampering or deletion.

CloudTrail Dashboard After Configuration

The screenshot shows the AWS CloudTrail Dashboard. On the left, a navigation sidebar lists 'CloudTrail' (selected), 'Event history' (highlighted with a red box), 'Trails' (selected), and other options like 'Lake', 'Dashboards', 'Query', 'Event data stores', 'Integrations', 'Pricing', 'Documentation', 'Forums', and 'FAQs'. The main content area has three sections: 'Query results history' (empty), 'Trails info' (showing a single trail named 'pgsql_centralised_logs' in 'Logging' status), and 'CloudTrail Insights' (disabled). The 'Event history' section is highlighted with a red box and contains a table with four rows of event logs:

Event name	Event time	Event source
CreateLogStream	January 21, 2026, 17:32:25 (UTC...)	logs.amazonaws.com
CreateLogStream	January 21, 2026, 17:27:53 (UTC...)	logs.amazonaws.com
StartLogging	January 21, 2026, 17:22:35 (UTC...)	cloudtrail.amazonaws.com
UpdateTrail	January 21, 2026, 17:22:34 (UTC...)	cloudtrail.amazonaws.com

The CloudTrail dashboard shows the active trail and recent events. The interface confirms logging is operational. Events are being captured and stored as configured. The dashboard provides quick access to event history and trail management. CloudTrail is now providing continuous audit logging for the account.

Confirming Read-Only Access Restriction

The screenshot shows the Amazon S3 'Upload: status' page. At the top, a red banner displays the message 'Upload failed' with a link to 'Diagnose with Amazon Q'. Below this, a summary table shows 'Succeeded' (0 files, 0 B (0%)) and 'Failed' (1 file, 1.2 MB (100.00%)). The 'Failed' row corresponds to the uploaded file 'IBRA CA CONSENT SCAN.pdf'. A detailed table below lists the file's name, folder, type, size, status, error, and audit logs. The 'Status' column shows 'Failed' and the 'Error' column shows 'Access denied', both highlighted with red boxes. The bottom of the page includes standard AWS navigation links like CloudWatch, Feedback, and Mobile App.

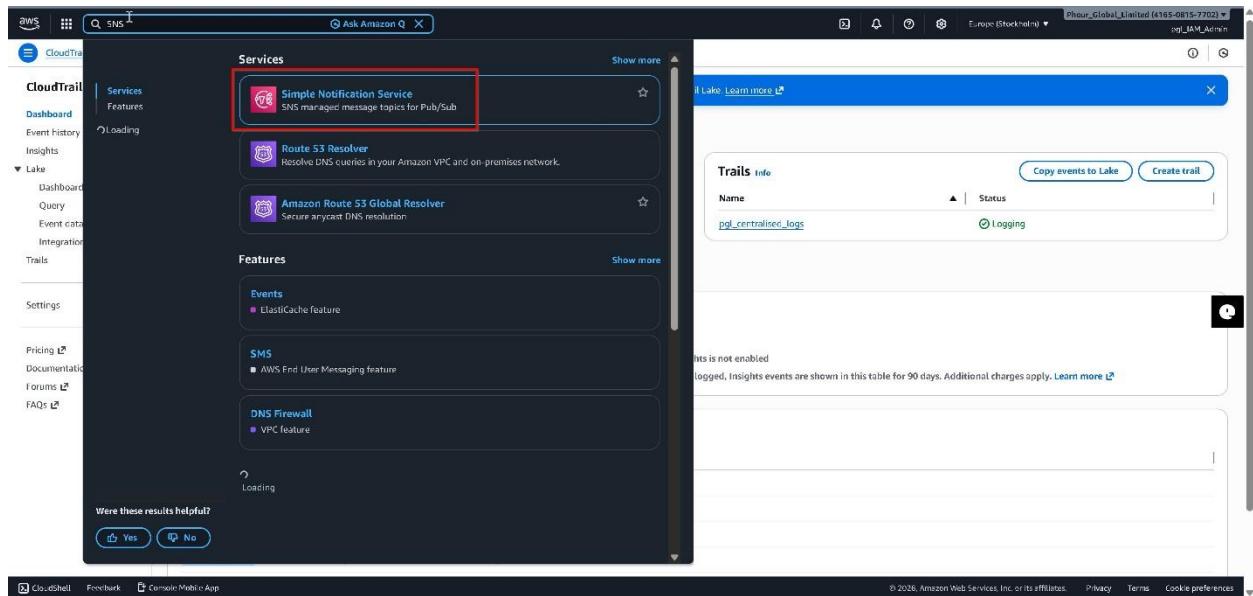
Name	Folder	Type	Size	Status	Error
IBRA CA CONSENT SCAN.pdf	-	application/pdf	1.2 MB	Failed	Access denied

This screenshot verifies the read-only S3 user cannot upload files. The upload attempt fails with insufficient permissions error. This confirms the AmazonS3ReadOnlyAccess policy is working correctly. The error demonstrates proper access control enforcement. All denied actions are logged to CloudTrail for security monitoring.

CREATING NOTIFICATION SERVICE

Simple Notification Service (SNS)

Accessing SNS Service



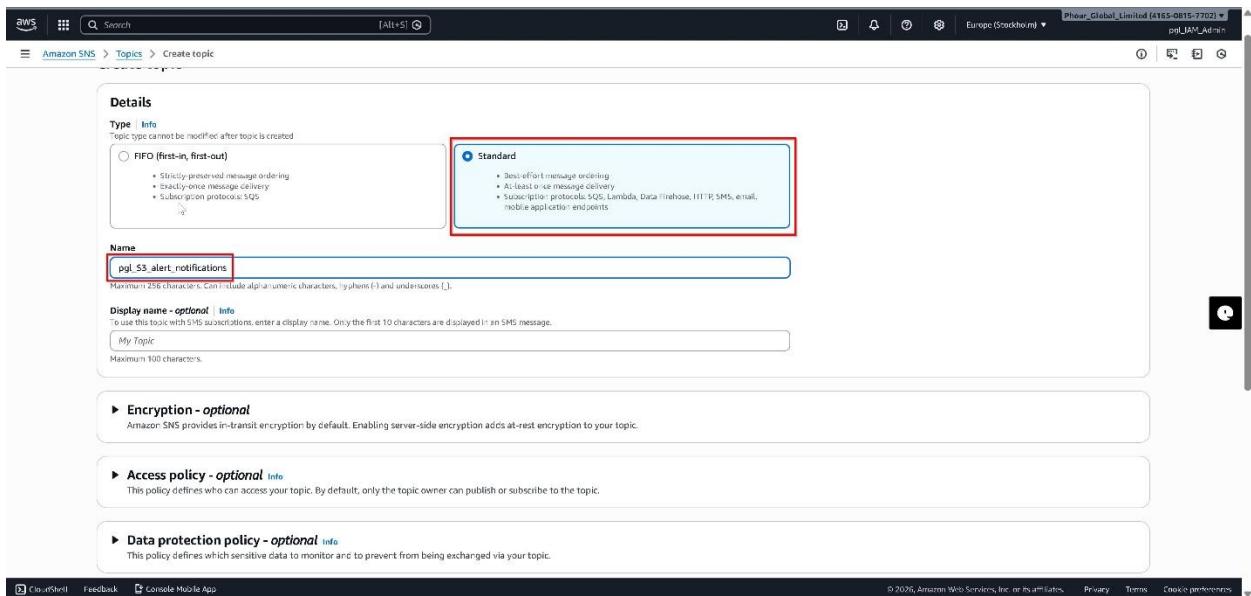
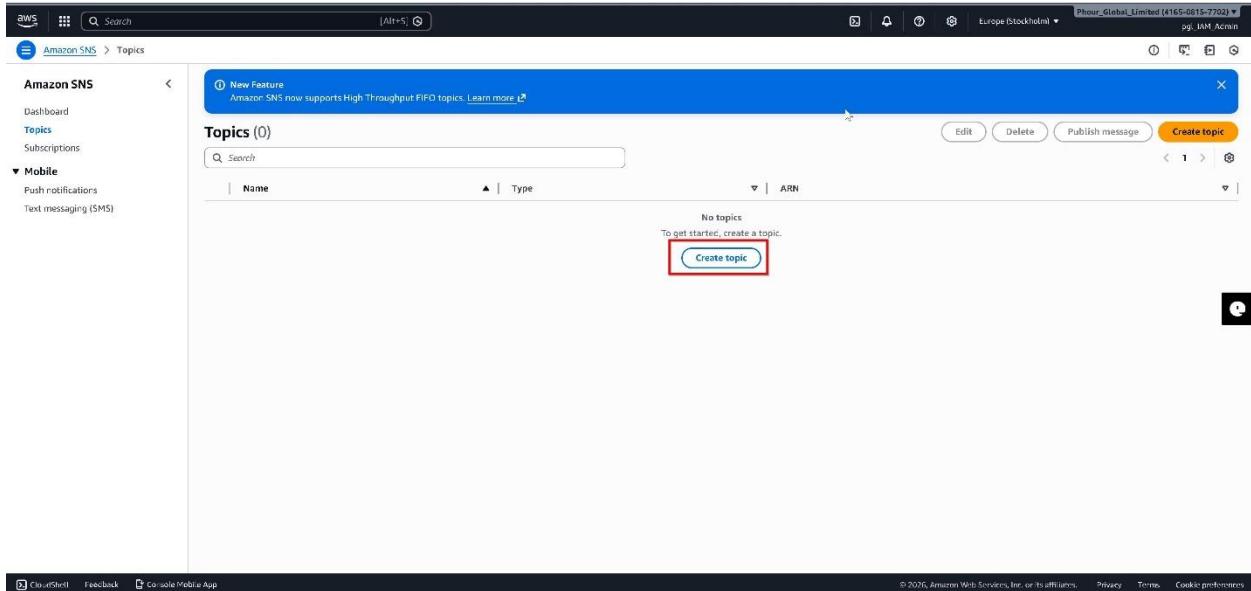
The console displays navigation to Simple Notification Service. SNS coordinates message delivery to subscribers through topics. The service supports multiple protocols including email, SMS, HTTP endpoints, and AWS Lambda. SNS enables building event-driven architectures and notification systems.

SNS Dashboard

The screenshot shows the Amazon SNS Dashboard. At the top, there's a blue banner with the text "Amazon SNS now supports High Throughput FIFO topics. Learn more". Below the banner, the main area is titled "Dashboard" and shows "Resources for eu-north-1". It has three sections: "Topics" (0), "Platform applications" (0), and "Subscriptions" (0). The "Topics" section contains a diagram of the "Application-to-application (A2A)" flow. It starts with a "Publisher" icon, followed by an arrow pointing to a central "Amazon SNS Fully managed Pub/Sub messaging and event delivery computing service" box. From this box, arrows point to two paths: one leading to a "Dead-letter queue" box (described as holding messages for analysis or reprocessing if an endpoint is unavailable) and another leading to an "SMS topic" box. From the "SMS topic" box, an arrow points to a "Message filtering and fanout" box, which then branches out to various "Subscribers" (AWS Lambda, Amazon SQS, Amazon Kinesis Data Firehose, HTTP/HTTPS, and Email). Below this diagram, there's a section for "Application-to-person (A2P)" with a note that it lets you send push notifications to mobile apps. The bottom of the screen includes standard AWS navigation links like "Console", "Feedback", "Console Mobile App", and copyright information.

The SNS dashboard shows topics, subscriptions, and service metrics. Topics are communication channels that subscribers can receive messages from. The create topic button starts the topic configuration wizard. The dashboard provides access to topic management and subscription configuration.

Creating SNS Topic



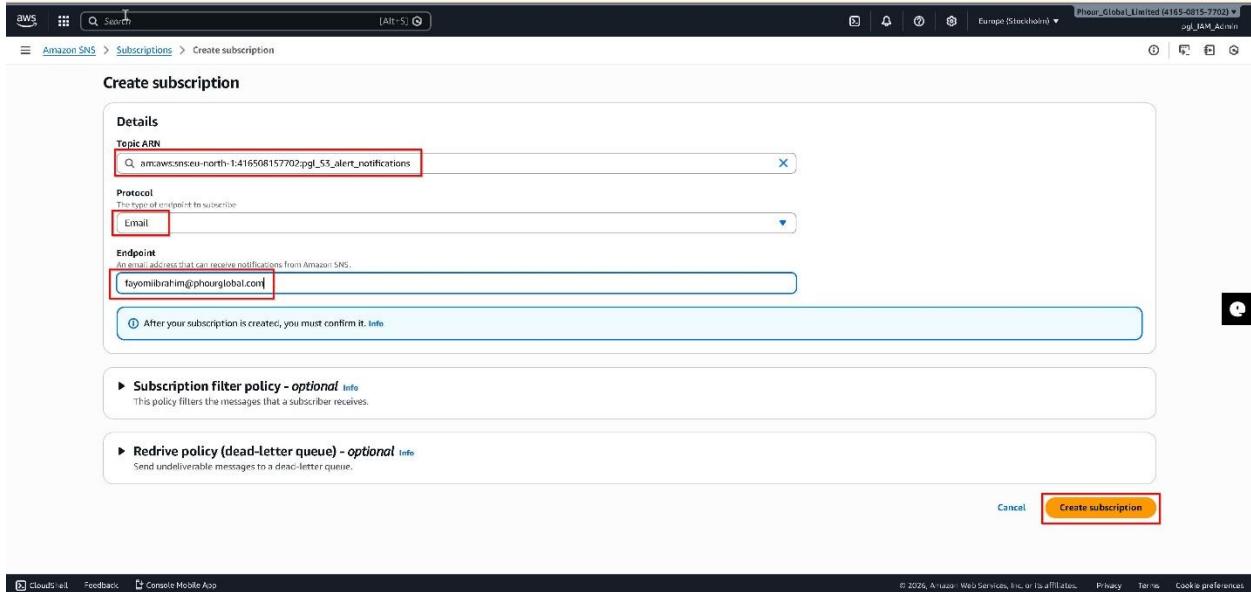
The topic creation wizard prompts for topic configuration. Standard topic type is selected which provides high throughput message delivery. The topic name clearly identifies its purpose such as Security-Alerts. Topic display name is used in certain notification protocols. Access policy determines who can publish and subscribe to the topic.

Topic Created and Creating Subscription

The topic details page shows the newly created topic. The create subscription button adds subscribers who will receive messages published to this topic. Subscriptions define how and where messages are delivered. Multiple subscriptions can be attached to a single topic to notify different endpoints or users.

The topic details page shows the newly created topic. The create subscription button adds subscribers who will receive messages published to this topic. Subscriptions define how and where messages are delivered. Multiple subscriptions can be attached to a single topic to notify different endpoints or users.

Creating Email Subscription



The subscription configuration shows email protocol selected. The endpoint field contains the email address that will receive notifications. Email subscriptions require confirmation to prevent spam. After creation, AWS sends a confirmation message to the specified address. The subscription remains in pending status until confirmed.

Subscription Created

The screenshot shows the AWS SNS console. In the top navigation bar, there is a search bar and a dropdown menu showing "Phour_Global_Limited (4165-0015-7702) Phour_IAM_Admin". Below the navigation bar, the URL is "Amazon SNS > Topics > pgl_53_alert_notifications > Subscription: 200ec8b7-268b-409a-901f-973b7ba935c5". On the left, there is a sidebar with "Amazon SNS" and sections for "Dashboard", "Topics", "Subscriptions", "Mobile", "Push notifications", and "Text messaging (SMS)". The main content area is titled "Subscription: 200ec8b7-268b-409a-901f-973b7ba935c5". It shows the following details:

- ARN:** arn:aws:sns:eu-north-1:416508157702:pgl_53_alert_notifications:200ec8b7-268b-409a-901f-973b7ba935c5
- Endpoint:** fayomilrahim@gmail.com
- Topic:** pgl_53_alert_notifications
- Subscription Principal:** arn:aws:iam::416508157702:user/pgl_IAM_Admin
- Status:** Pending confirmation
- Protocol:** EMAIL

Below the details, there is a section for "Subscription filter policy" with the note "No filter policy configured for this subscription. To apply a filter policy, edit this subscription." There is an "Edit" button.

The subscriptions list displays the new email subscription in pending confirmation status.

The subscription will not receive messages until confirmed. The status indicator helps track which subscriptions are active. Unconfirmed subscriptions are automatically deleted after three days if not confirmed.

Subscription Confirmed from Email

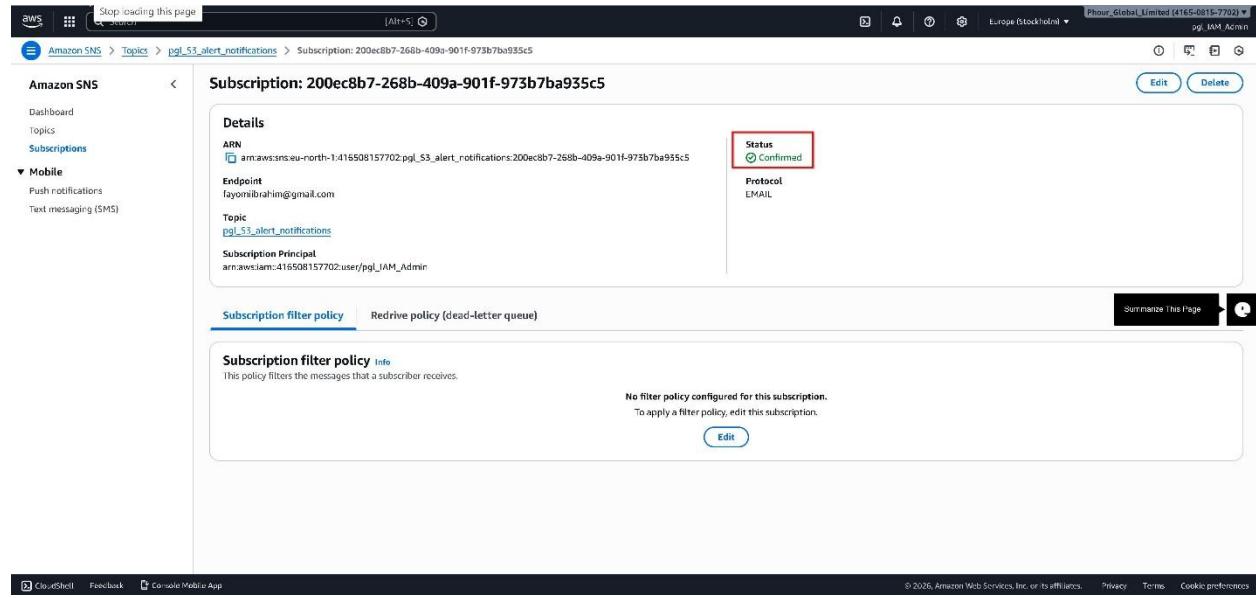
The screenshot shows a confirmation email from "Simple Notification Service". The subject line is "Subscription confirmed!". The body of the email contains the following text:

You have successfully subscribed.
Your subscription's ID is:
arn:aws:sns:eu-north-1:416508157702:pgl_53_alert_notifications:200ec8b7-268b-409a-901f-973b7ba935c5
If it was not your intention to subscribe, [click here to unsubscribe](#).

The confirmation email contains a link to activate the subscription. Clicking the confirmation link opens a web page confirming successful subscription. This verification step

ensures the email owner consents to receiving notifications. The confirmation process prevents unauthorized subscription of email addresses.

Subscription Confirmed in Console

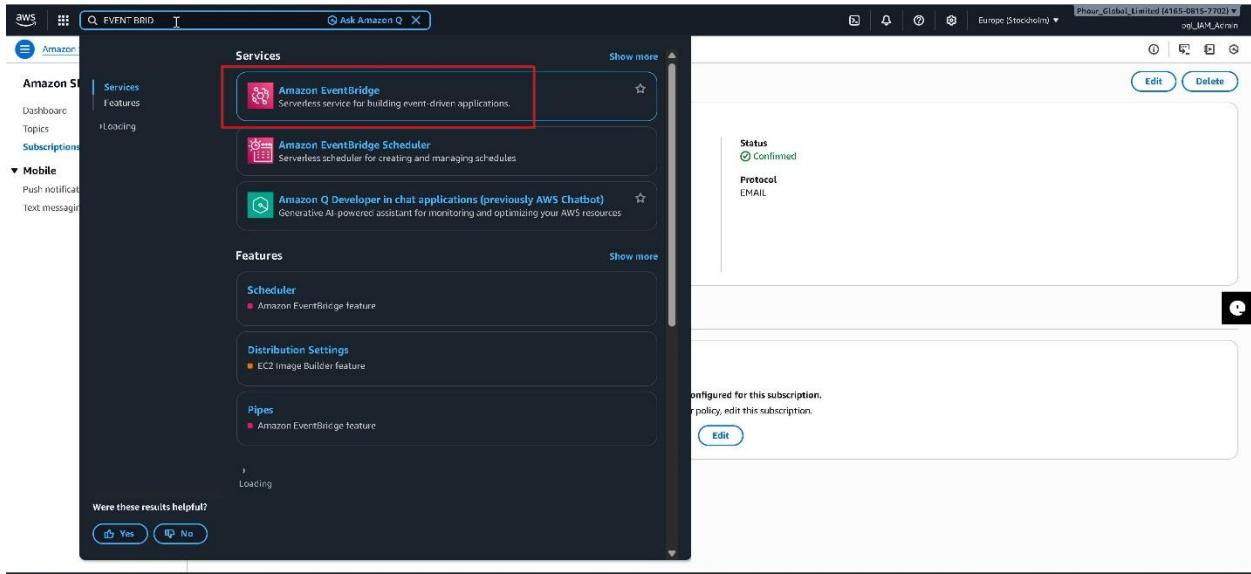


The screenshot shows the AWS SNS console interface. On the left, there's a navigation sidebar with options like Dashboard, Topics, Subscriptions, Mobile, Push notifications, and Text messaging (SMS). The main content area is titled "Subscription: 200ec8b7-268b-409a-901f-973b7ba935c5". It displays the ARN (arn:aws:sns:eu-north-1:416508157702:pgl_S3_alert_notifications:200ec8b7-268b-409a-901f-973b7ba935c5), Endpoint (fayonmrahim@gmail.com), Topic (pgl_S3_alert_notifications), and Subscription Principal (arn:aws:iam::416508157702:user/pgl_IAM_Admin). A red box highlights the "Status" field, which is set to "Confirmed". Below this, the "Protocol" is listed as "EMAIL". Under "Subscription filter policy", it says "No filter policy configured for this subscription. To apply a filter policy, edit this subscription." There are "Edit" and "Delete" buttons at the top right of the main content area. At the bottom, there are links for CloudShell, Feedback, and Console Mobile App, along with standard footer links for Privacy, Terms, and Cookie preferences.

The subscription status changes to confirmed after email verification. The subscription is now active and will receive all messages published to the topic. The email address will receive formatted notifications based on the message content. Multiple subscribers can be added to create distribution lists for alerts.

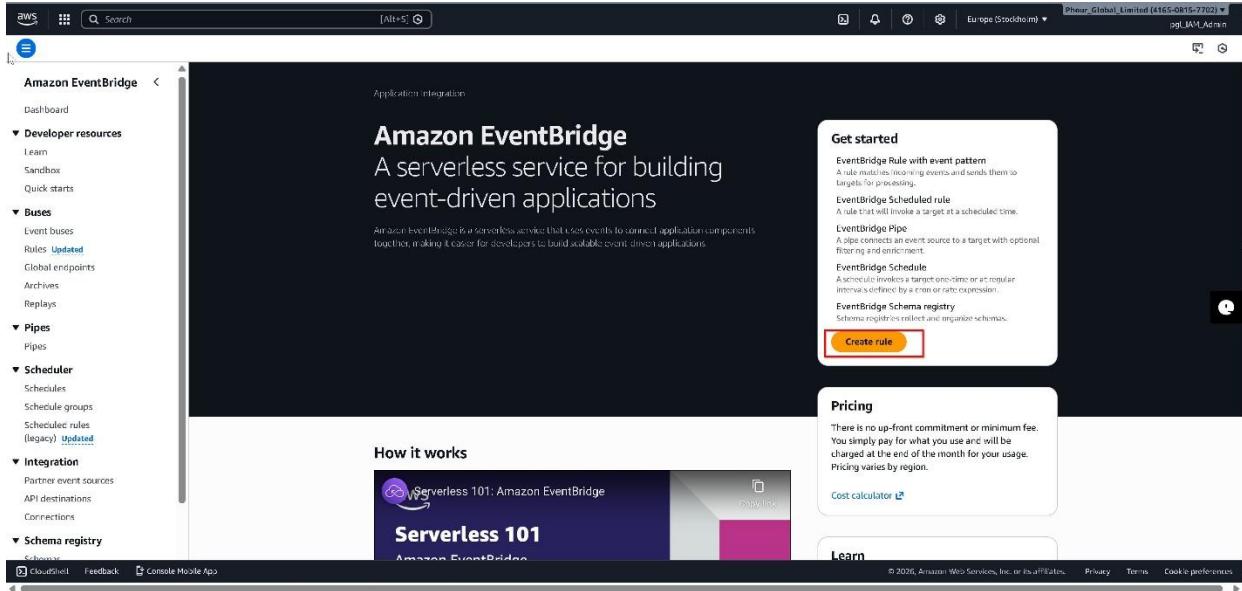
AMAZON EVENT BRIDGE

Accessing EventBridge Service



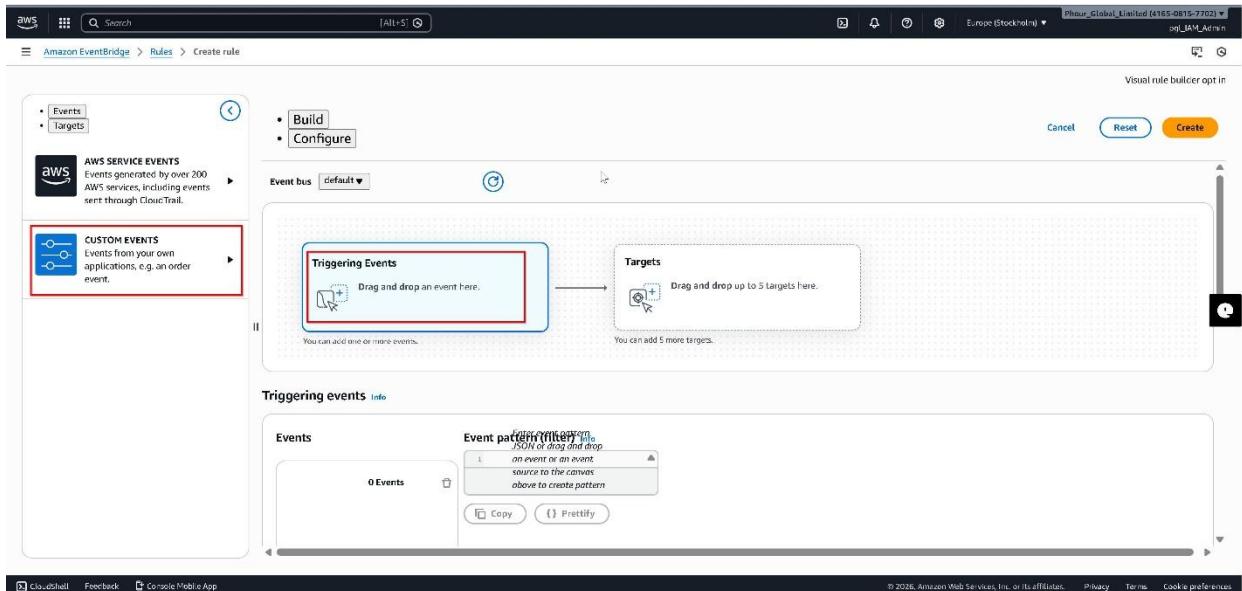
The console shows navigation to Amazon EventBridge. EventBridge is an event bus service that connects applications using events from AWS services, SaaS applications, and custom sources. Rules match incoming events and route them to targets for processing. EventBridge enables event-driven architectures and automated response to system changes.

Creating EventBridge Rule



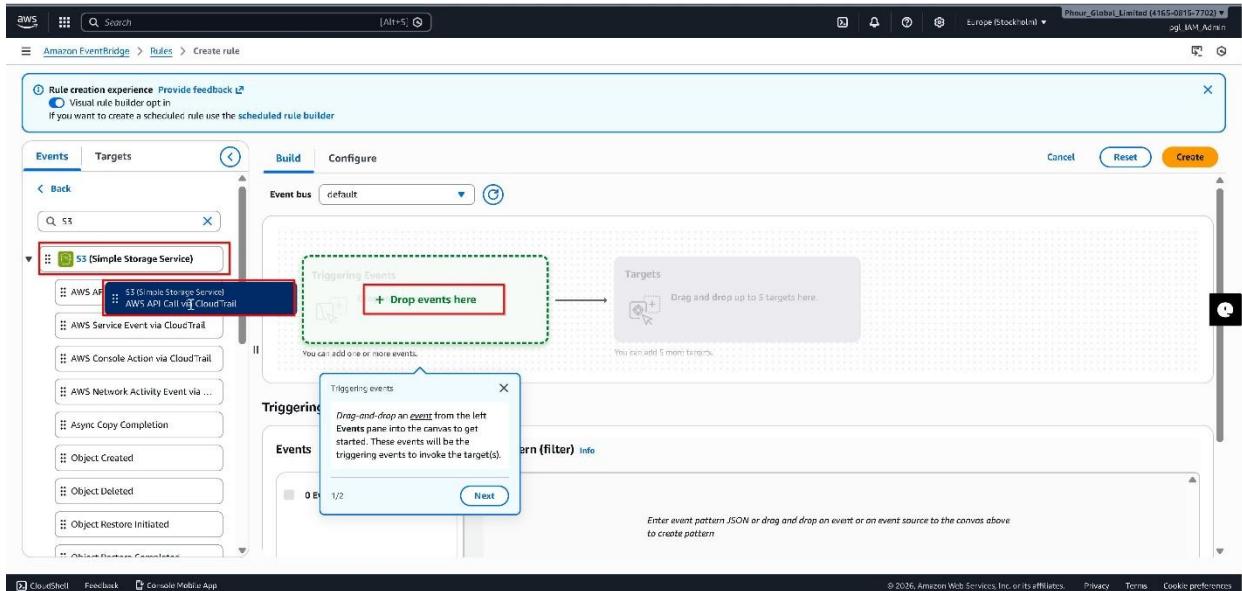
The rules page displays options to create new rules. Rules define event patterns that trigger actions when matched. The create rule button starts the rule configuration wizard. Rules can respond to specific AWS service events, schedules, or custom events.

Configuring Rule Trigger Event



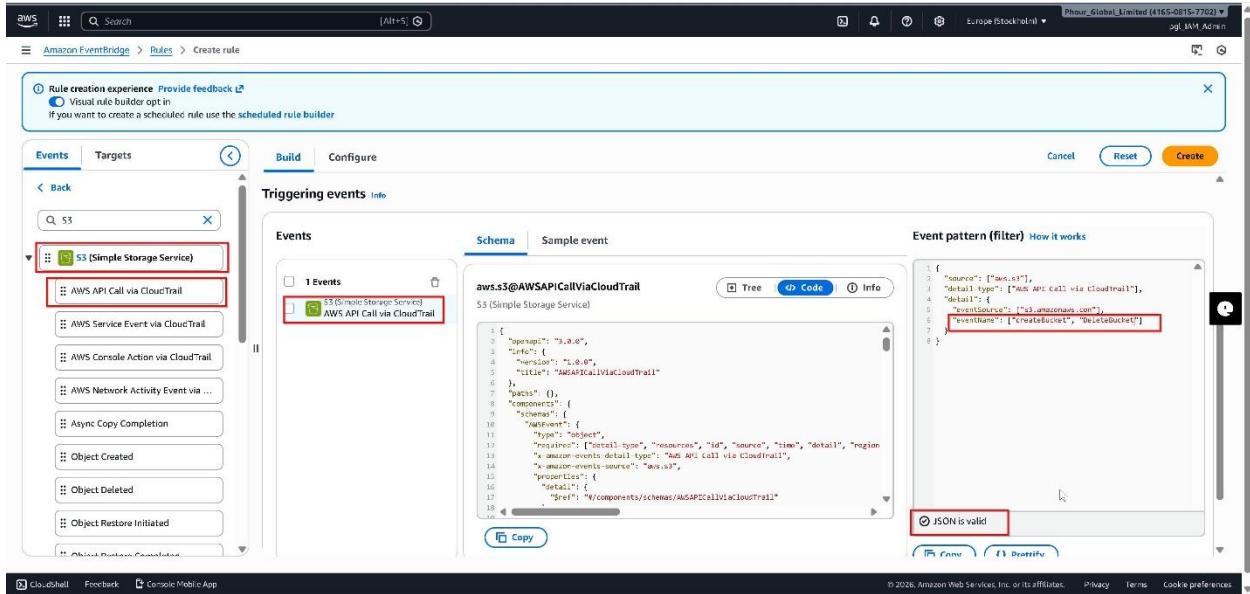
The rule configuration shows event source selection. AWS events option is selected to monitor AWS service actions. The event pattern determines which events trigger the rule. Event patterns can match specific services, event types, or detailed parameters within events.

Selecting API Call via CloudTrail as Event Source



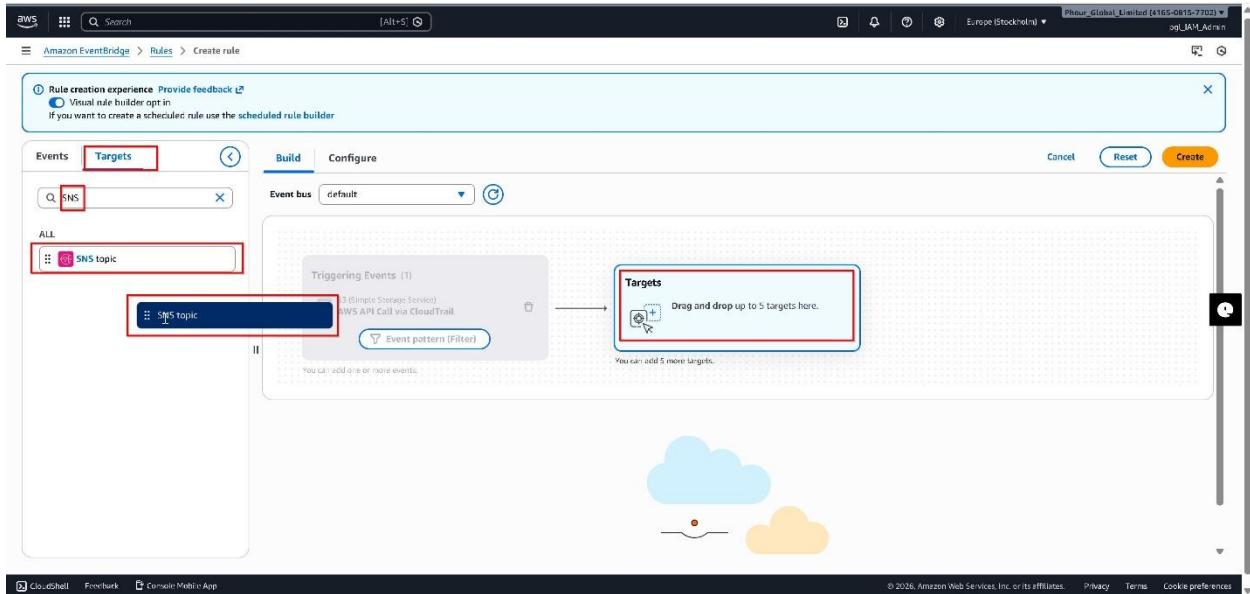
The event pattern builder shows CloudTrail selected as the event source. CloudTrail integration allows EventBridge to respond to any AWS API call. This enables monitoring specific actions like resource creation, modification, or deletion. The pattern can filter by AWS service and specific API operations.

Editing Event Pattern in JSON



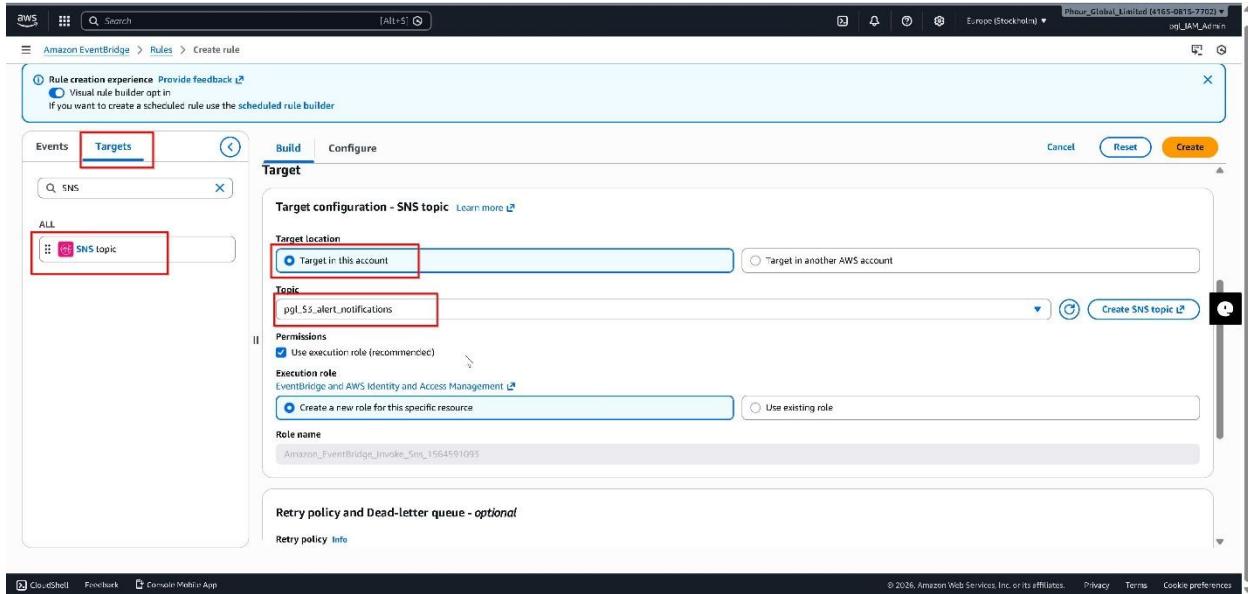
The JSON editor displays the event pattern code. The pattern specifies source as aws.s3 and detail-type as AWS API Call via CloudTrail. The eventName field is added to match specifically CreateBucket events. This pattern will trigger whenever a new S3 bucket is created in the account. JSON editing provides precise control over event matching beyond the visual builder options.

Creating SNS Target



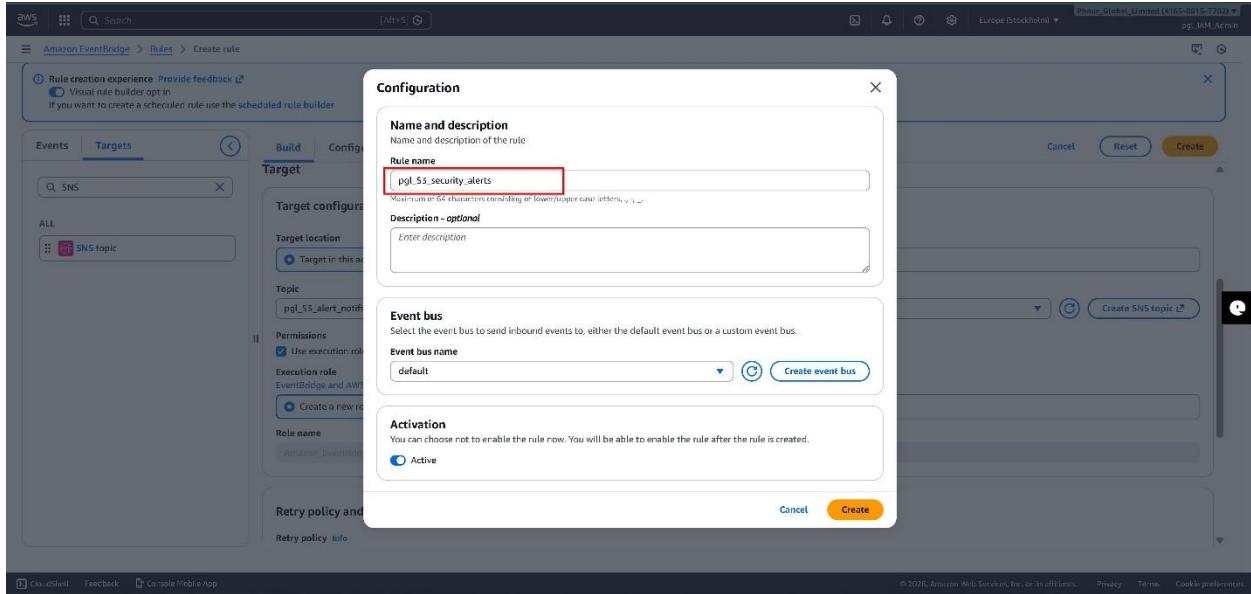
The target configuration shows SNS topic selected as the action to perform when the rule matches. Targets define what happens when events match the rule pattern. Multiple targets can be added to a single rule to trigger different actions. SNS target will publish a message to the specified topic which then notifies all subscribers.

Configuring SNS Topic as Target



The SNS configuration shows selecting the Security-Alerts topic created earlier. The message input determines what content is sent to subscribers. The default setting sends the complete event details in JSON format. Custom messages can be configured to format the notification. EventBridge will automatically create necessary permissions for the rule to publish to the SNS topic.

Naming Rule and Creating



The rule details section prompts for rule name and description. The name clearly identifies the monitoring purpose such as S3-Bucket-Creation-Alert. Description explains what events trigger the rule and what actions result. The create rule button finalizes the configuration and activates monitoring.

Rule Created Successfully

The screenshot shows the AWS EventBridge console. In the left sidebar, under 'Developer resources' > 'Rules', there is a red box around the 'pgl_S3_security_alerts' rule. The main panel displays the rule details: 'Status' is 'Enabled', 'Event bus name' is 'awslogs', and 'Type' is 'Standard'. Below this, the 'Event pattern' tab is selected, showing a JSON configuration:

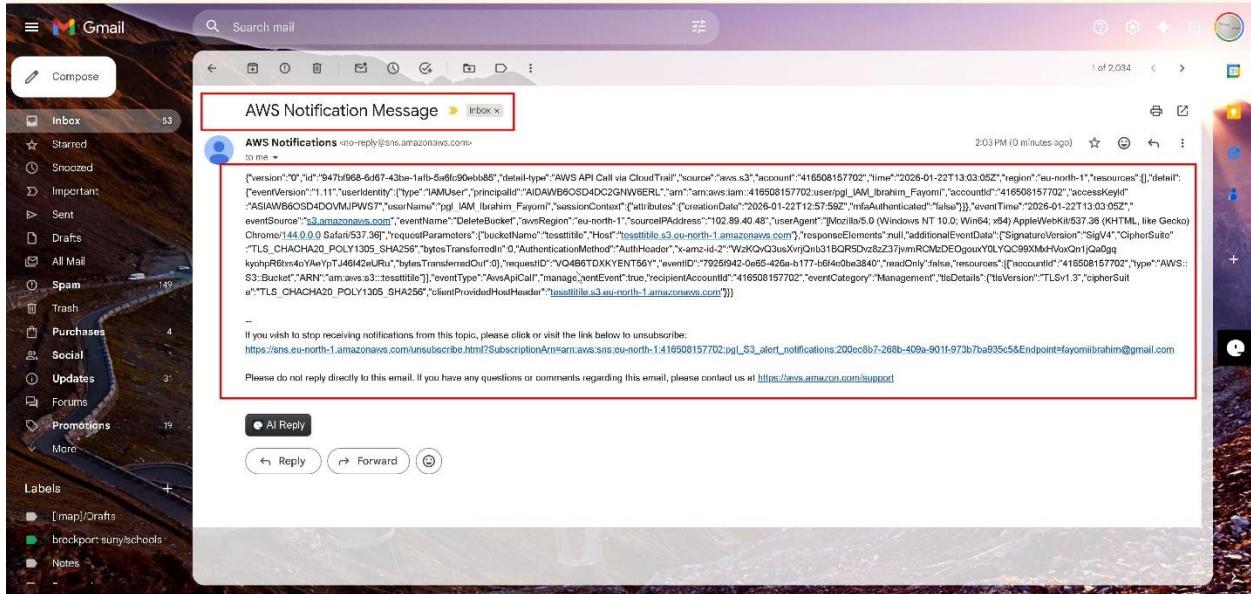
```

1 {
2   "source": ["aws.s3"],
3   "detail-type": ["AWS API call via CloudTrail"],
4   "detail": {
5     "eventSource": ["*.*.amazonaws.com"],
6     "eventName": ["CreateBucket", "DeleteBucket"]
7   }
8 }
  
```

At the bottom of the event pattern, there is a 'Copy' button.

The rules list displays the newly created rule in enabled status. The rule is now actively monitoring for S3 bucket creation events. EventBridge evaluates all CloudTrail events against the rule pattern. When a match occurs the SNS topic receives a notification which is forwarded to email subscribers. The rule provides automated real-time alerting for specified security events.

Alert Received After Creating New Bucket



The email notification shows the alert triggered by creating a new S3 bucket. The message contains event details including event name, time, user identity, and affected resources. The notification arrived within minutes of the bucket creation action. This validates that the entire monitoring pipeline is functional from CloudTrail event capture through EventBridge rule matching to SNS notification delivery.

The successful alert demonstrates event-driven security monitoring in action. The system automatically detected a specific API call and notified designated personnel without manual intervention. This automated alerting enables rapid response to security events and unauthorized actions.

Security Controls and Policies

This project implements comprehensive security controls across multiple layers of AWS infrastructure. Identity and access management provides authentication and authorization

through IAM users, groups, and policies. Root account usage is limited to initial administrative user creation following AWS security best practices.

Access control is enforced through multiple mechanisms. User groups organize personnel by role and apply appropriate permissions collectively. AWS managed policies provide predefined permission sets for common use cases. Custom policies created via AWS Policy Generator implement least privilege by granting only specific actions on designated resources. Permission validation testing confirms that policies work as intended.

Data protection uses encryption at rest through S3 server-side encryption with AWS managed keys. All objects stored in S3 buckets are automatically encrypted and decrypted transparently. Bucket policies and IAM permissions control who can access stored data. Public access is blocked by default preventing accidental data exposure.

Compute security applies EC2 instance access controls through security groups and key pair authentication. Security group rules define allowed inbound and outbound traffic. RSA key pairs provide cryptographic authentication stronger than passwords. Custom IAM policies restrict instance management actions to authorized users.

Comprehensive audit logging captures all API activity through CloudTrail. Management events record control plane operations like creating users or launching instances. CloudTrail logs are stored in S3 with log file validation to detect tampering. Event history provides 90 days of readily accessible logs while trails enable extended retention.

Automated monitoring uses EventBridge to detect specific security events in real-time. Rules match event patterns from CloudTrail and trigger notifications through SNS. Email

subscriptions ensure designated personnel receive alerts immediately. The monitoring system operates continuously without manual oversight required.

Validation and Testing

All security configurations were validated through practical testing. **IAM** permissions were verified by attempting actions both allowed and denied by policies. Users with full **S3** access successfully uploaded and modified objects. Users with read-only access could download objects but received permission errors when attempting uploads or deletions.

Custom **EC2** policies were tested by logging in as the target user and performing allowed operations like stopping instances. Attempting actions outside the policy scope produced access denied errors confirming proper enforcement. Instance ARN restrictions ensured the policy applied only to specified resources.

CloudTrail logging was verified by reviewing event history showing captured API calls with complete details. Trail functionality was confirmed by checking S3 bucket contents for delivered log files. Log entries matched performed actions demonstrating accurate event capture.

SNS notifications were tested through email subscription confirmation and delivery verification. EventBridge rule effectiveness was validated by performing the monitored action and receiving email notification within minutes. The notification content matched the triggered event details.

Encryption implementation was confirmed through S3 bucket configuration review showing SSE-S3 enabled. All uploaded objects inherited encryption settings automatically. The transparent encryption and decryption process required no additional user actions.

Challenges and Solutions

Policy syntax errors during custom policy creation required careful JSON formatting. The AWS Policy Generator tool prevented most syntax issues by producing valid JSON. Manual JSON editing required attention to bracket matching, comma placement, and quotation marks. The IAM console validator flagged syntax errors before policy creation preventing invalid policies from being saved.

ARN formatting for resource-specific policies required understanding ARN structure and obtaining correct resource identifiers. The EC2 instance details page provided the full ARN for copy-paste into policy documents. ARN errors in policies prevented proper resource restriction until corrected with exact format.

SNS subscription confirmation delay could cause confusion if email arrives slowly. Checking spam folders and waiting several minutes resolved most delayed confirmation issues. The pending confirmation status in console indicated subscriptions awaiting activation.

EventBridge rule configuration required understanding event structure from CloudTrail. The JSON pattern needed exact matches for event source and detail fields. Testing rules by performing target actions and verifying notifications helped confirm correct configuration. Rule patterns that were too broad generated excessive alerts while patterns too specific missed relevant events.

Permission testing sometimes produced cached results masking policy changes. Waiting a few seconds for IAM policy propagation and refreshing console sessions ensured current permissions were evaluated. CloudTrail logs provided definitive record of what permissions were actually enforced during operations.

Conclusion

This project successfully implements AWS baseline security controls across core cloud services. The configuration establishes secure identity management with properly scoped permissions, protects data with encryption and access controls, manages compute resources with secure authentication, captures comprehensive audit logs, and monitors security events with automated alerting.

The implementation demonstrates practical application of AWS security best practices including least privilege access, defense in depth, comprehensive logging, and automated monitoring. Each component serves a specific security function while integrating with other components to provide layered protection.

The project validates hands-on competency in architecting, deploying, and managing secure cloud infrastructure. The work shows understanding of identity management, data protection, network security, audit logging, and event-driven automation. These skills are directly applicable to building and maintaining production cloud environments that meet security and compliance requirements.

The documented configurations serve as a reference implementation for AWS security baseline controls. The lab environment can be expanded with additional services such as GuardDuty for threat detection, Config for compliance monitoring, VPC flow logs for network analysis, and multi-region deployments for high availability scenarios. The foundation established here supports growth into more complex security architectures.

