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.

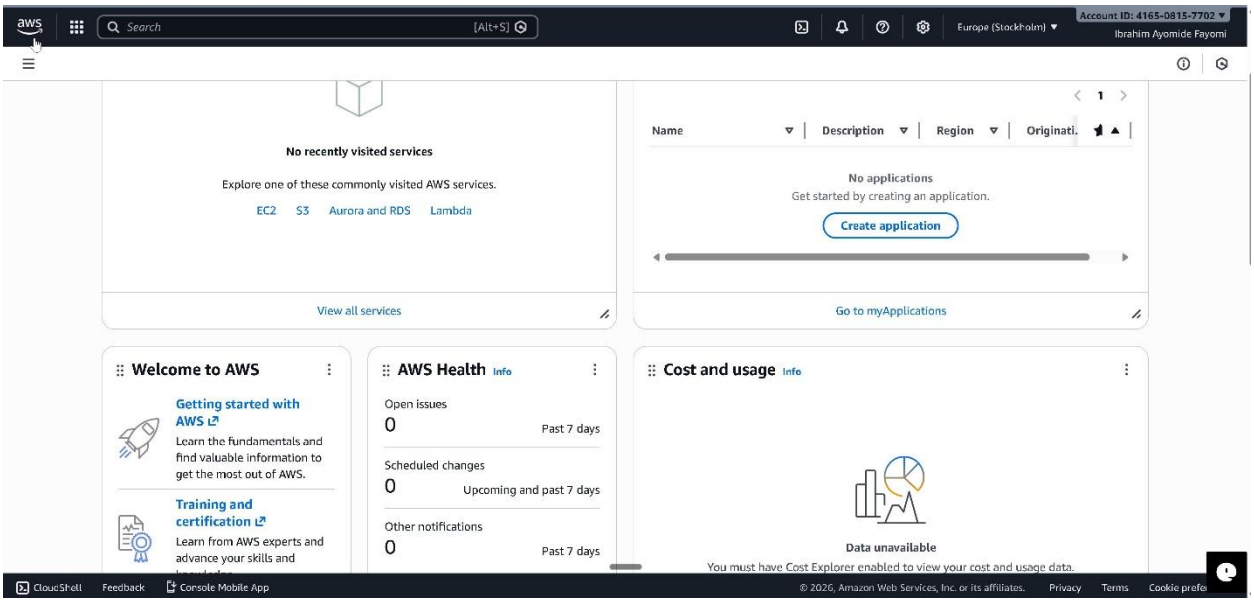
Step-by-Step Configuration

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)

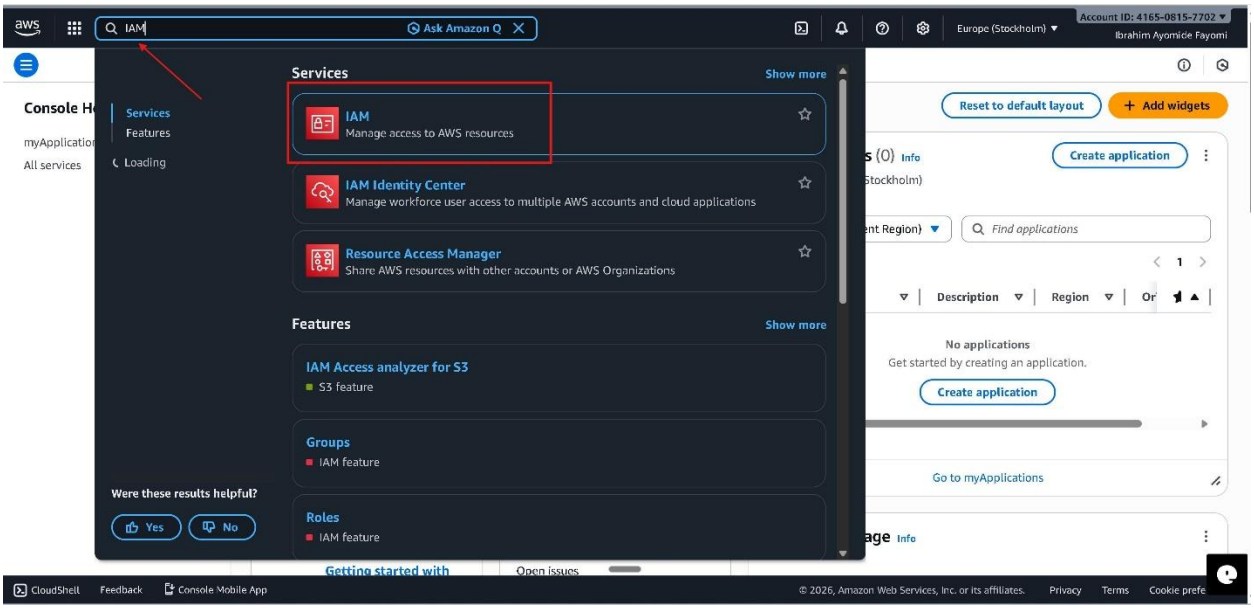
AWS Root Account Dashboard



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.

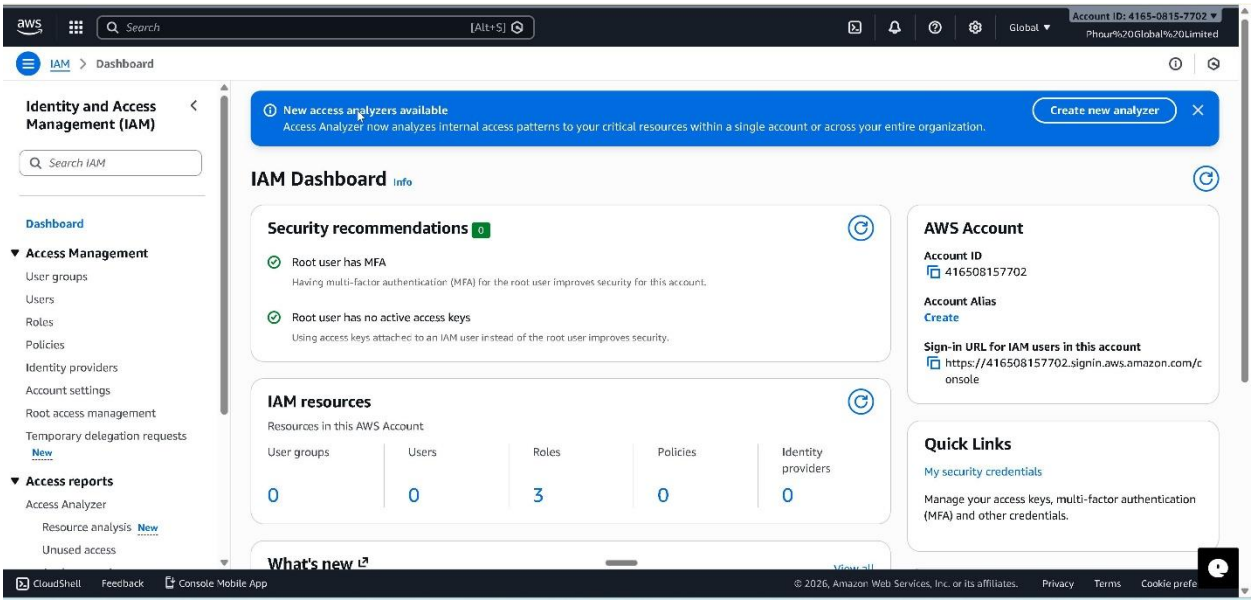
Using the root account for daily tasks violates AWS security best practices. The root account should be secured with multi-factor authentication and used only for tasks that specifically require root credentials such as closing the account or changing support plans.

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.

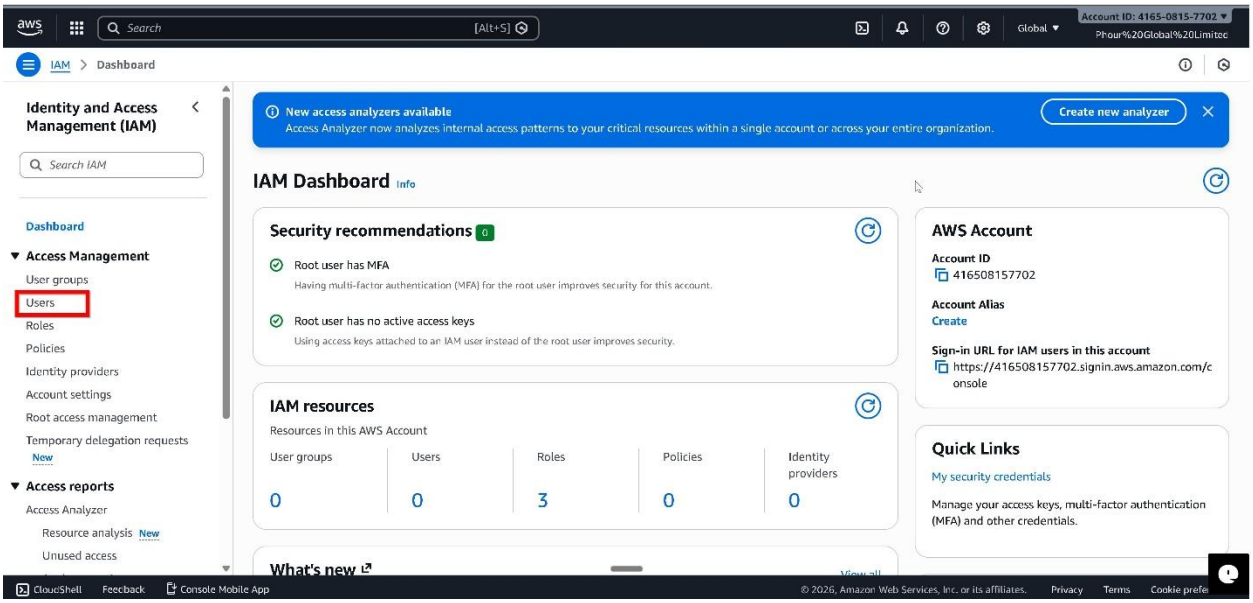
IAM Dashboard

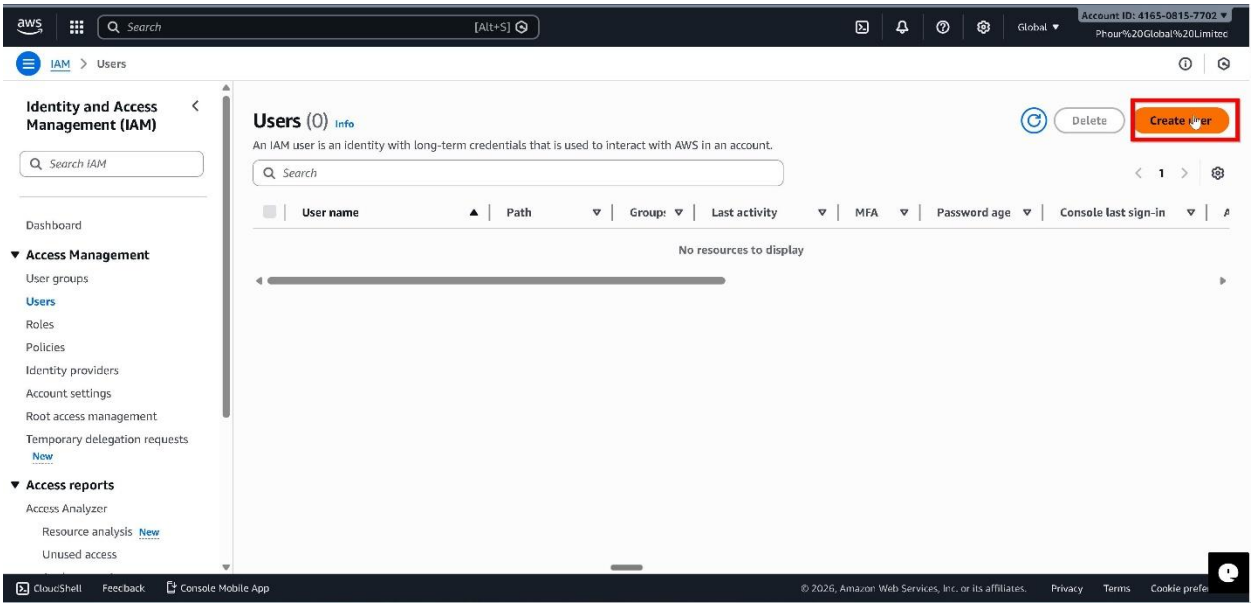


The IAM dashboard presents an overview of the current identity configuration. The security recommendations panel shows items requiring attention such as enabling MFA for root account and creating IAM users. The dashboard displays counts for users, groups, roles, and policies. The left navigation menu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

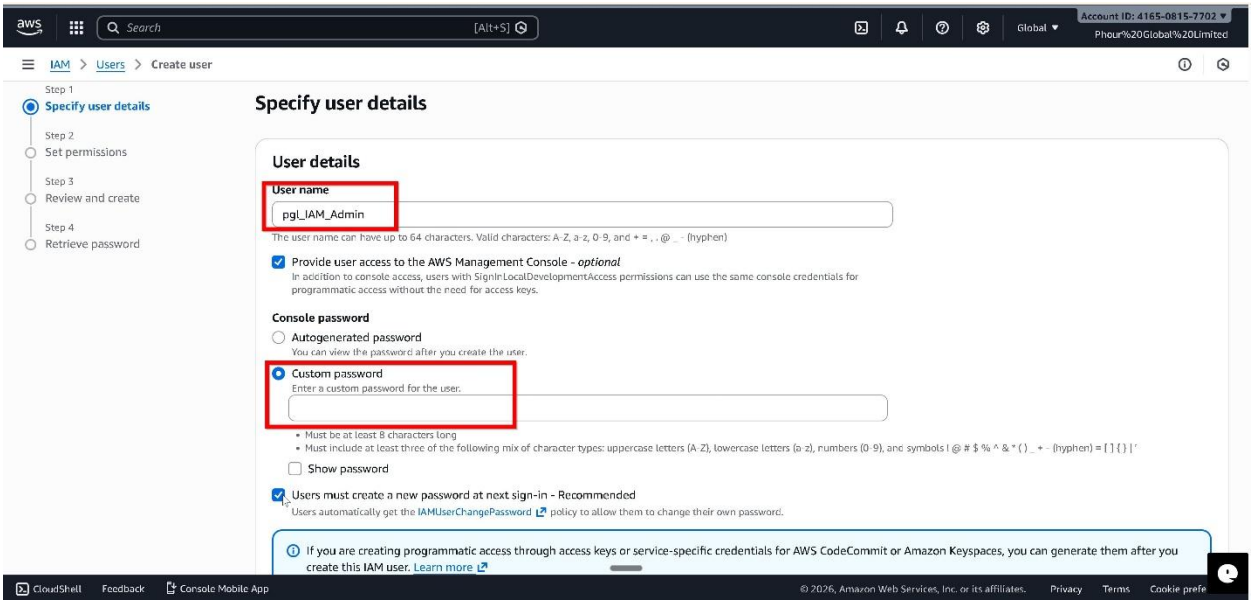




These screenshots show the IAM user creation wizard. The interface prompts for user details and access type selection. The admin user is being created with AWS Management Console access, which requires setting a password. The option to require password reset at first login is visible but not selected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

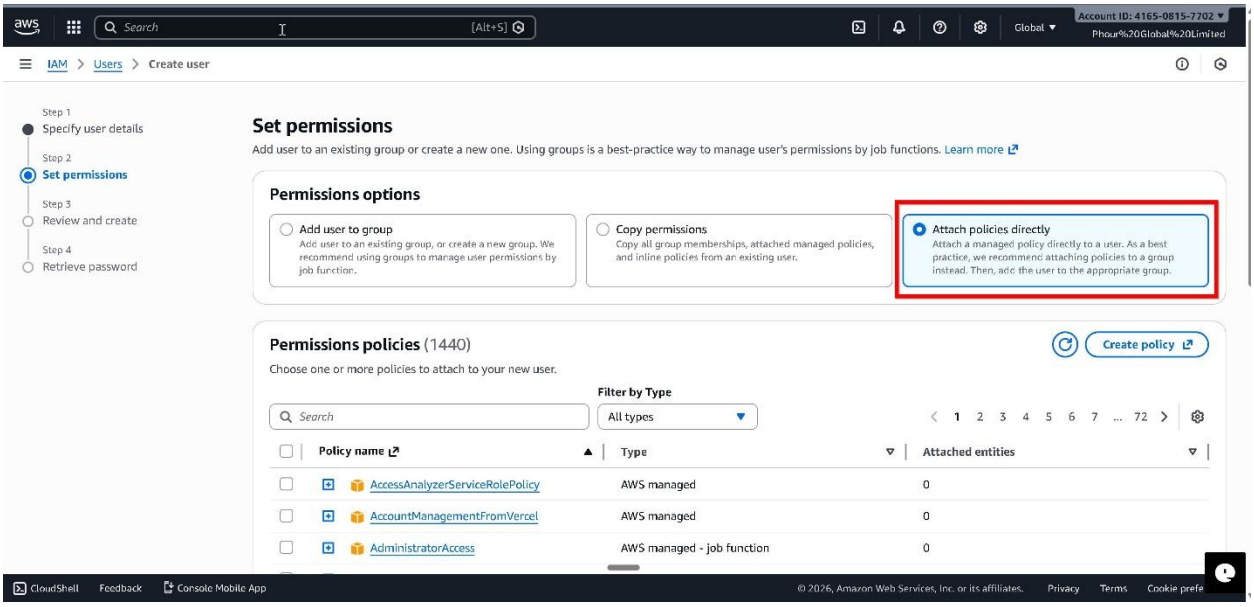
Creating an administrative IAM user separate from the root account follows AWS security best practices. This user will handle all administrative tasks going forward, leaving the root account secured and unused except for specific root-only operations. The administrative user provides full access while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

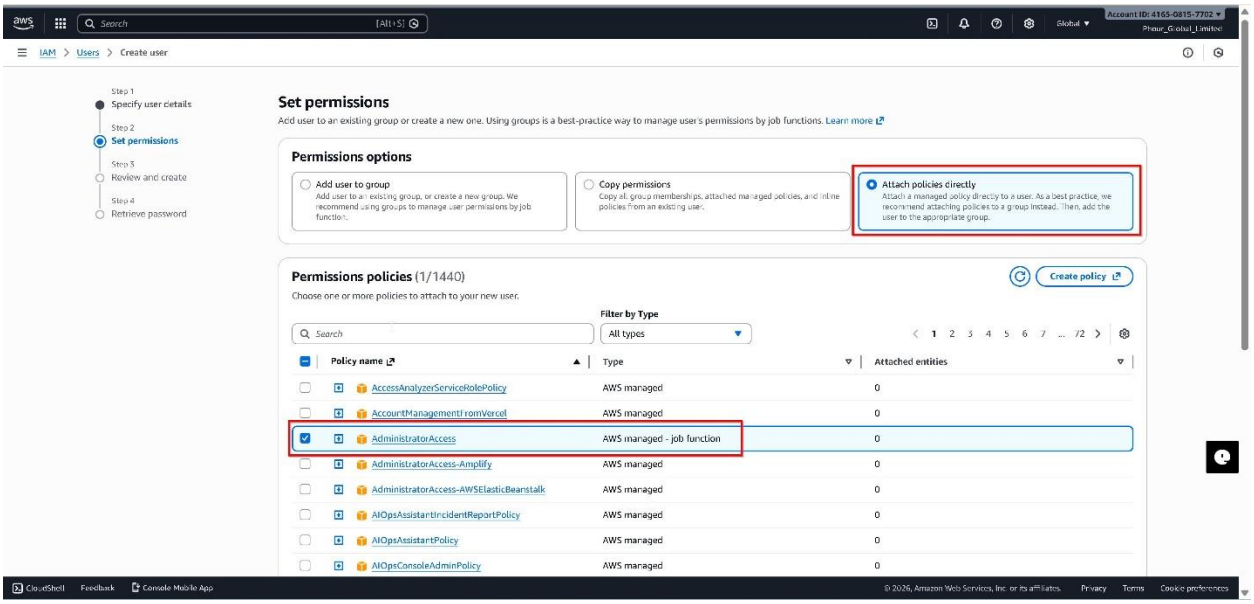
Adding Permission Directly to Admin User



The 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.

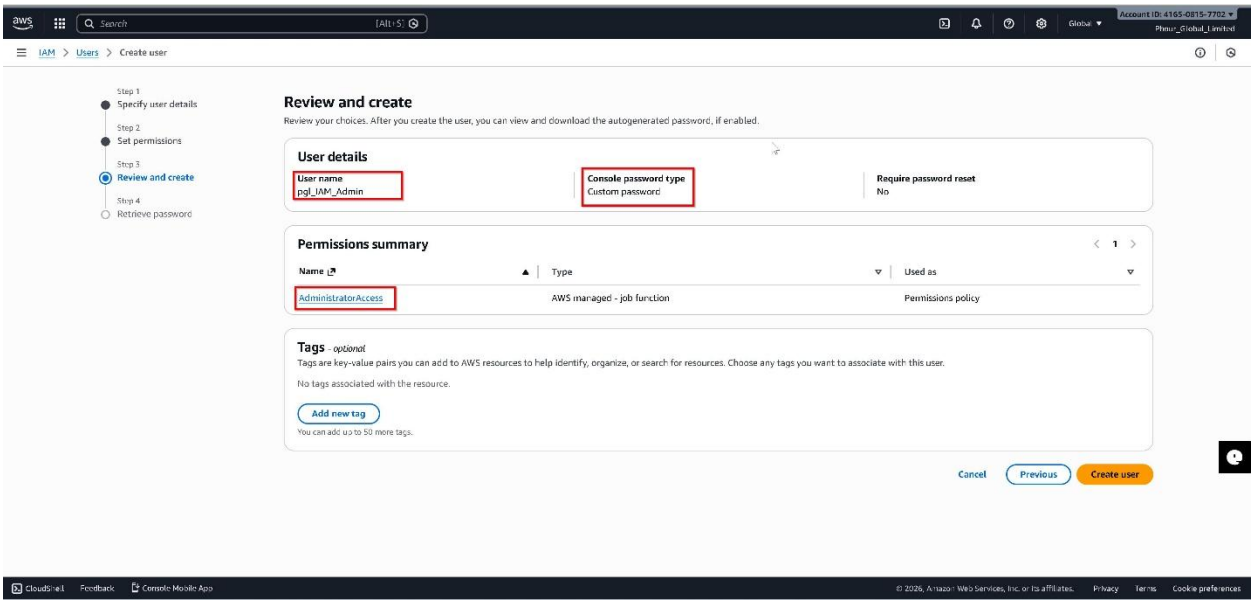
Granting Admin Access to Admin User Created



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.

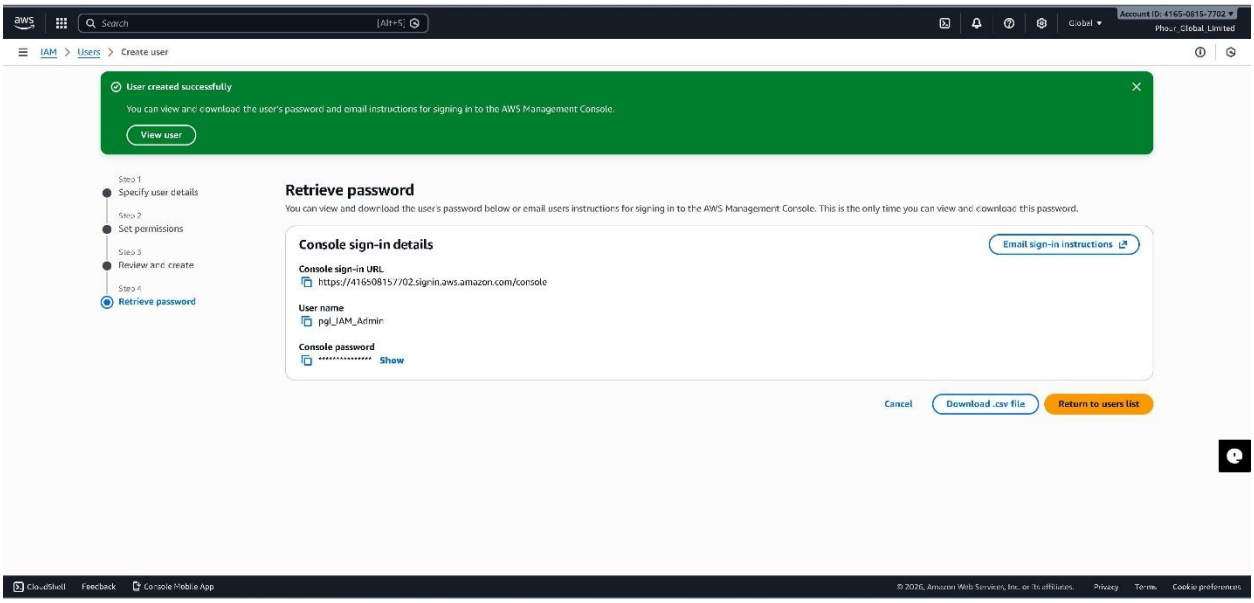
Administrative Access policy provides unrestricted service access while still maintaining separate identity for audit purposes. This allows full administrative control while avoiding direct use of root credentials for operational tasks.

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.

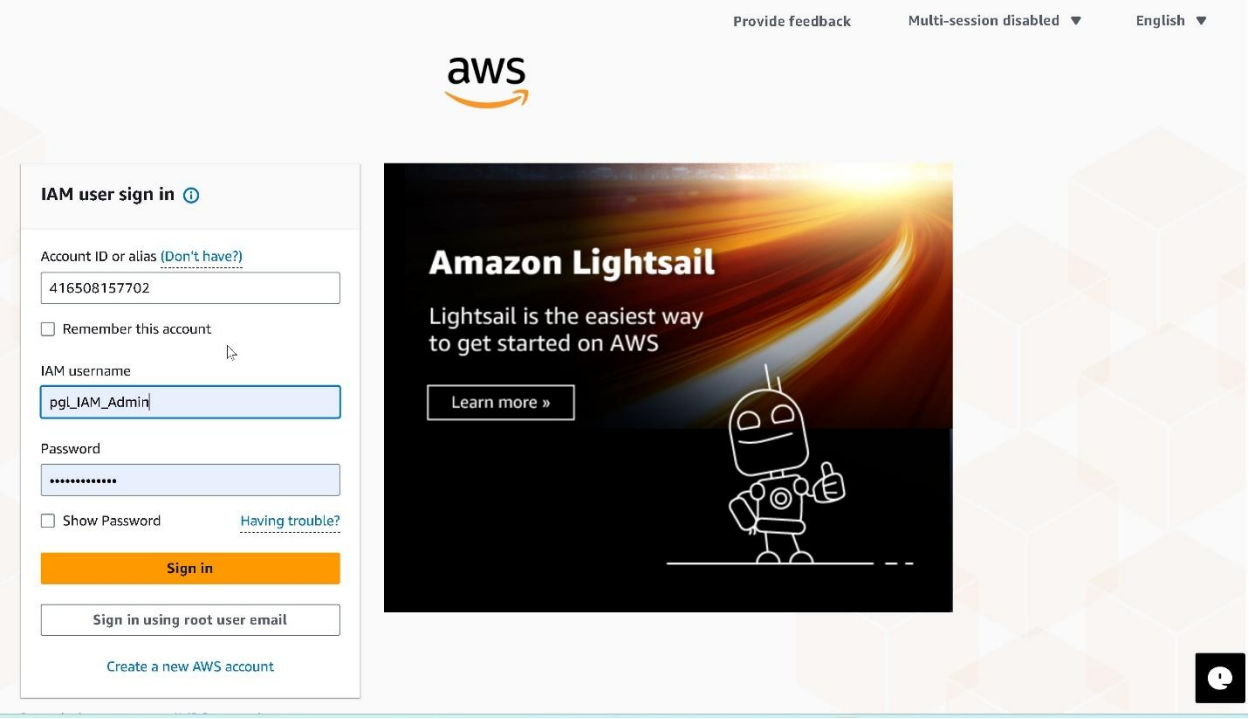
Admin User Created



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.

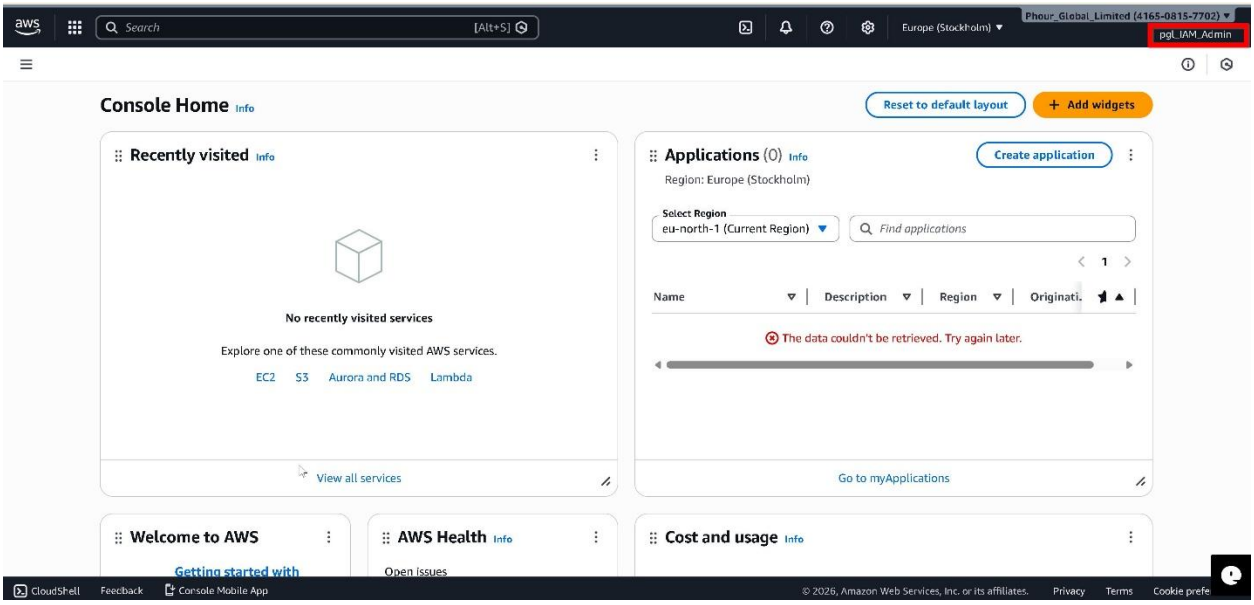
The IAM user sign-in URL differs from the root account login page. IAM users authenticate using account ID or alias plus username and password rather than email address. This separation provides clear distinction between root and IAM user access.

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.

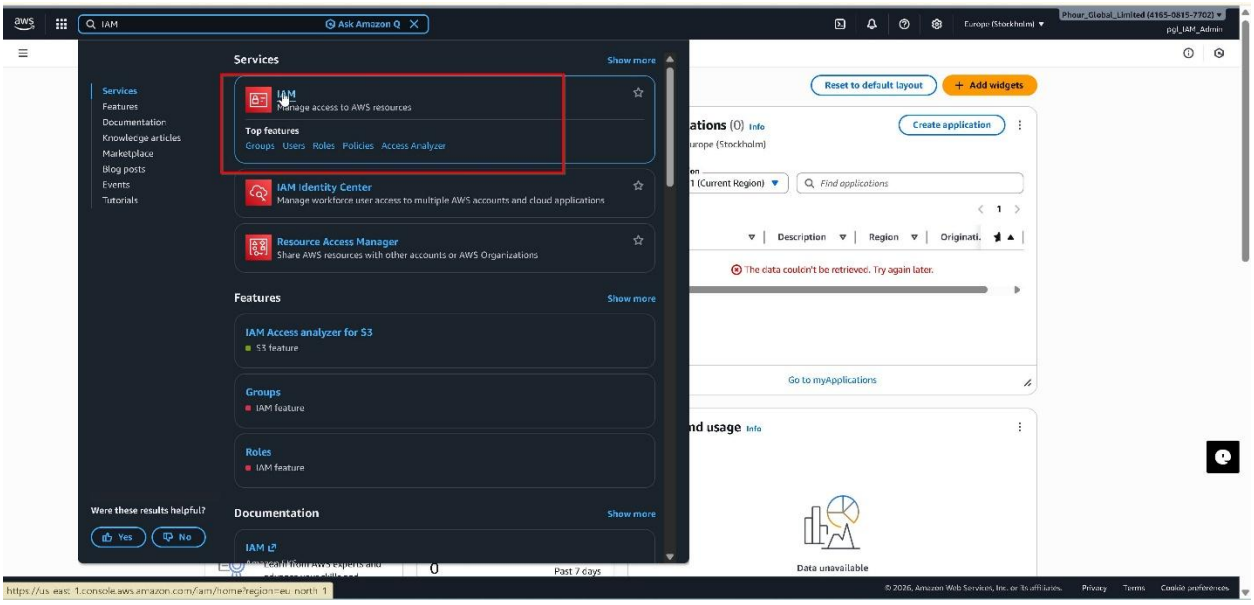
Logged in as Admin User (pgl_IAM_Admin)

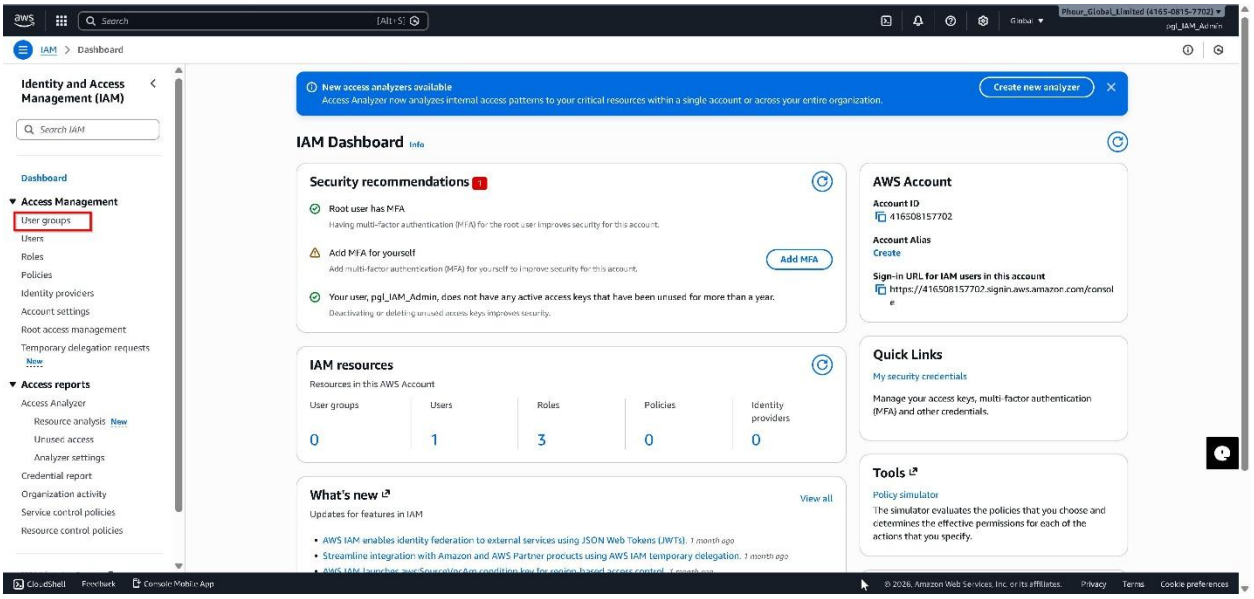


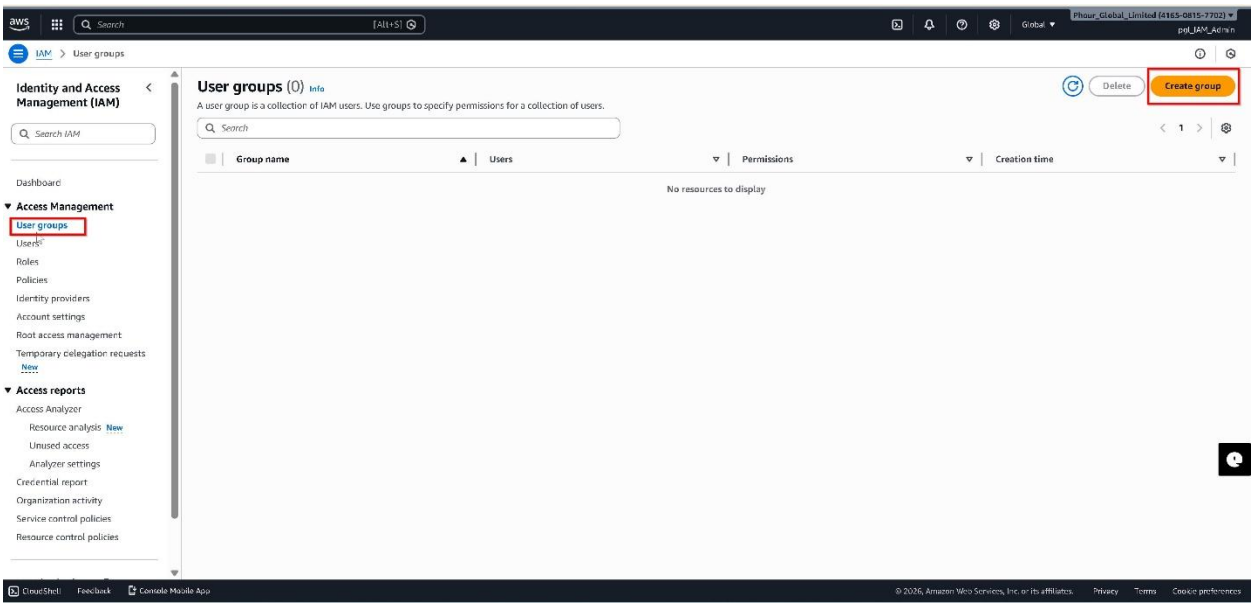
The console displays successful authentication as the admin IAM user. The top navigation bar shows the username indicating current session identity. All subsequent actions will be performed under this user identity and logged to CloudTrail with the IAM user as the principal. The console interface remains 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)



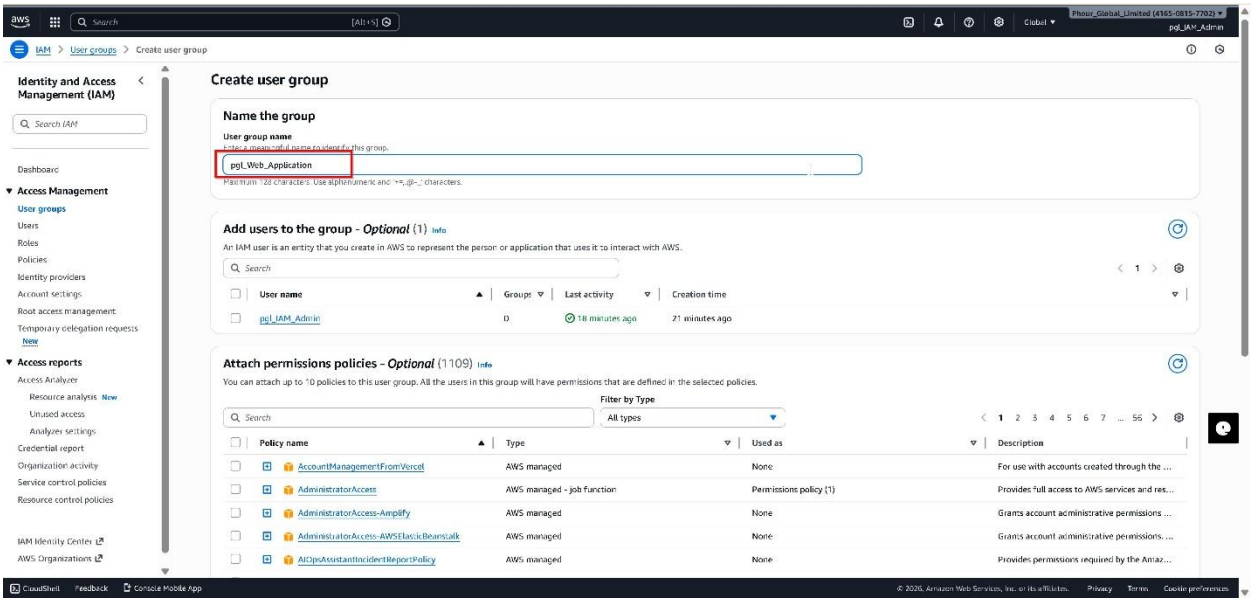




These screenshots show the user group creation process. The IAM console shows the groups section with the option to create new group. The group creation wizard prompts for group name and policy attachments. User groups provide a scalable way to manage permissions for multiple users performing similar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

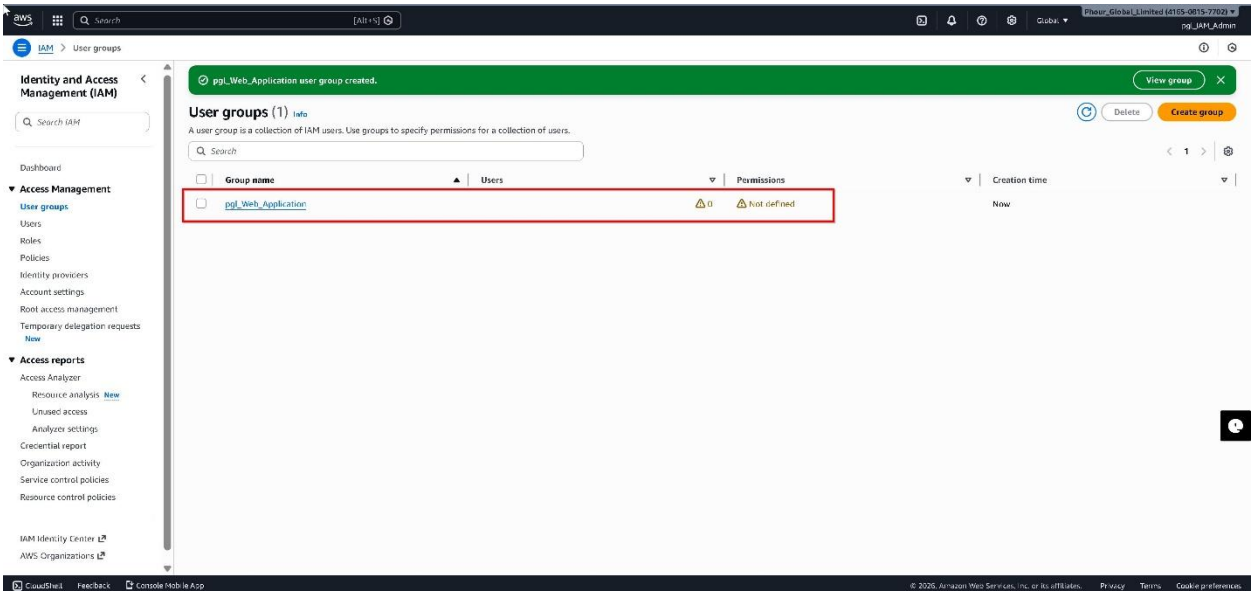
Groups simplify permission management by applying policies to the group rather than individual users. When a user joins a group they automatically inherit all group permissions. This reduces administrative overhead and ensures consistent permission application across team members.

User Group Named (under Admin user console)



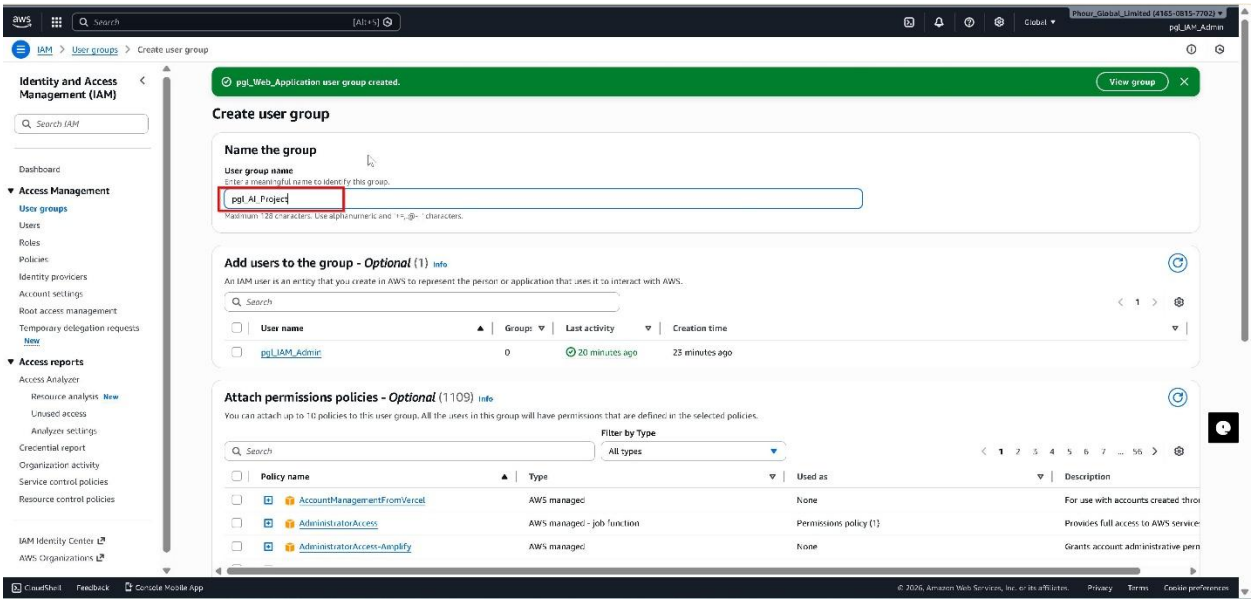
The screenshot shows a user group being created with a descriptive name such as Sales or Development. The group name should clearly indicate the function or department the group serves. Group names must be unique within the AWS account. Naming conventions help maintain organization when managing multiple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

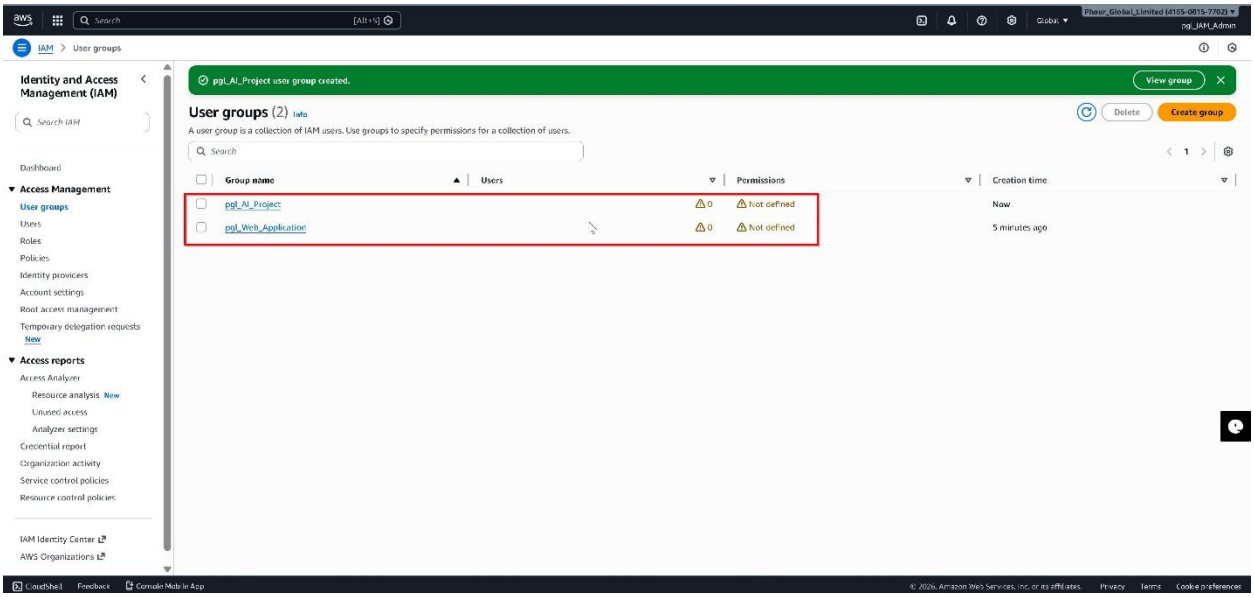
User Group Created



The 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.

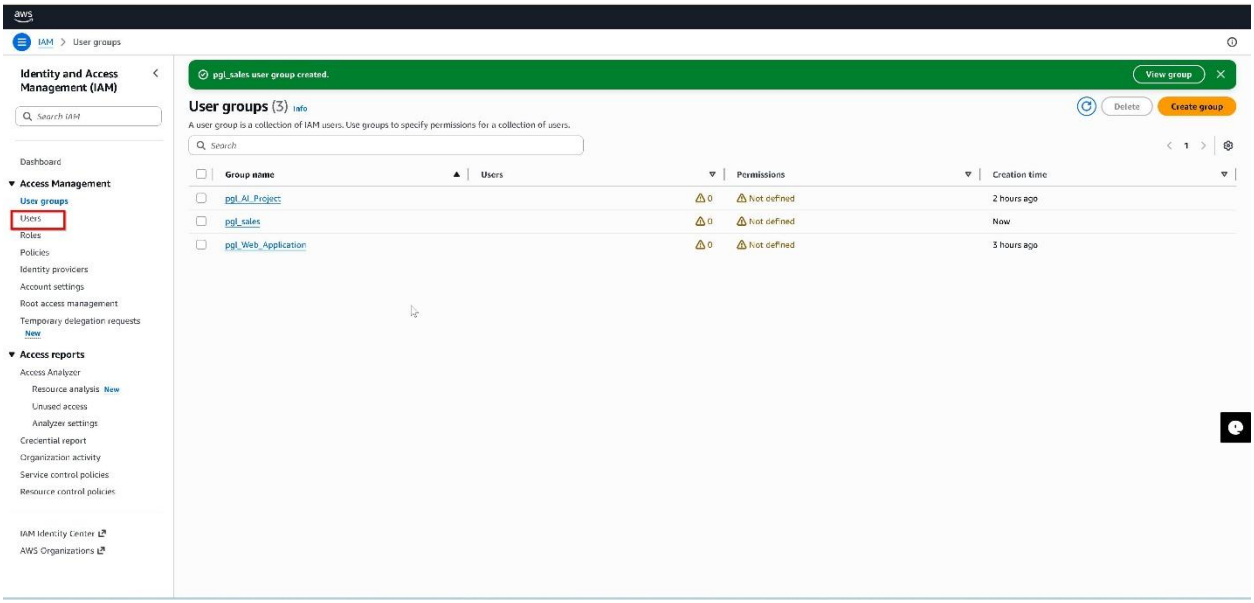
Multiple User Groups Created

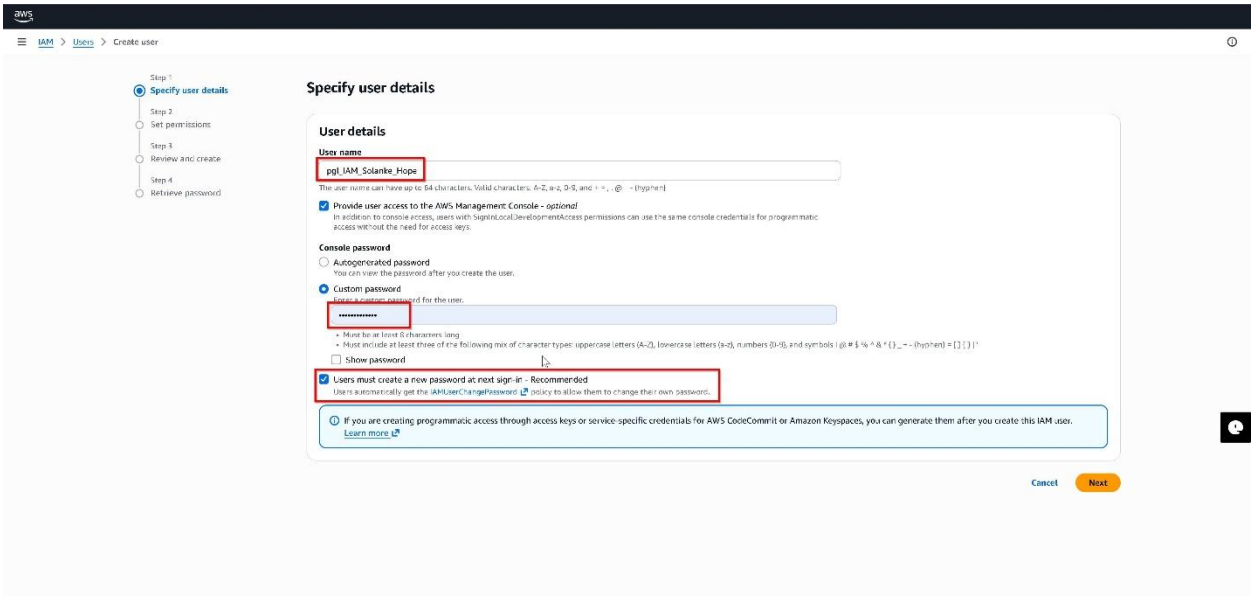




The screenshots show multiple user groups have been created for different organizational functions. Each group will receive appropriate permissions based on the access requirements for that role. Multiple groups allow fine-grained access control aligned with business functions. Users can be members of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

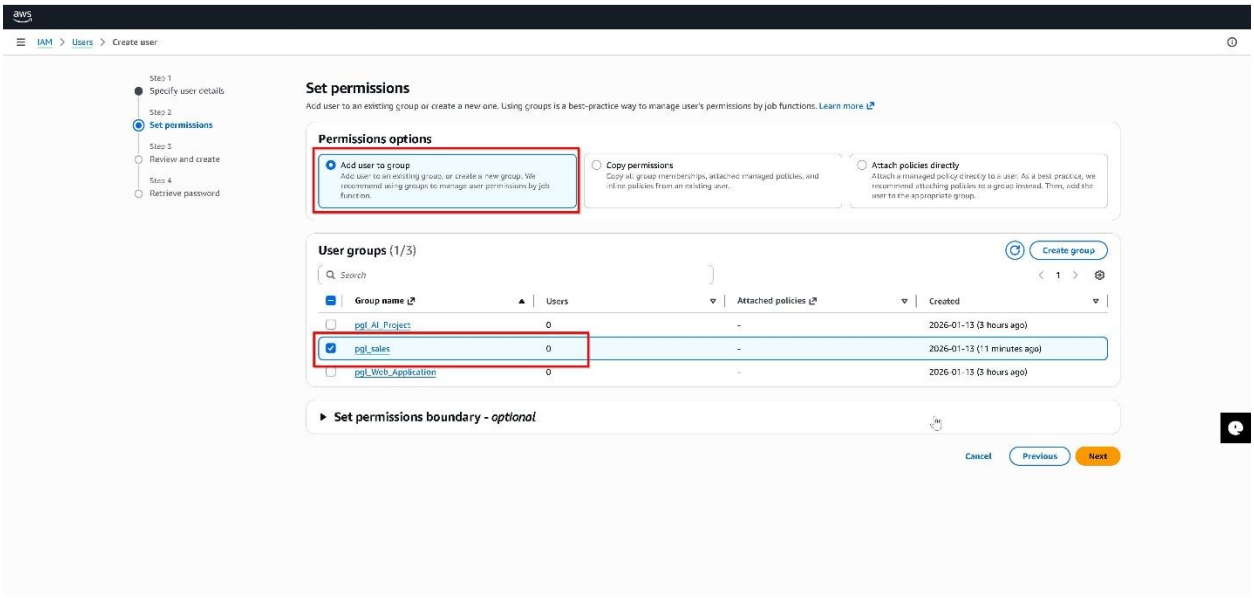
Creating Standard Users

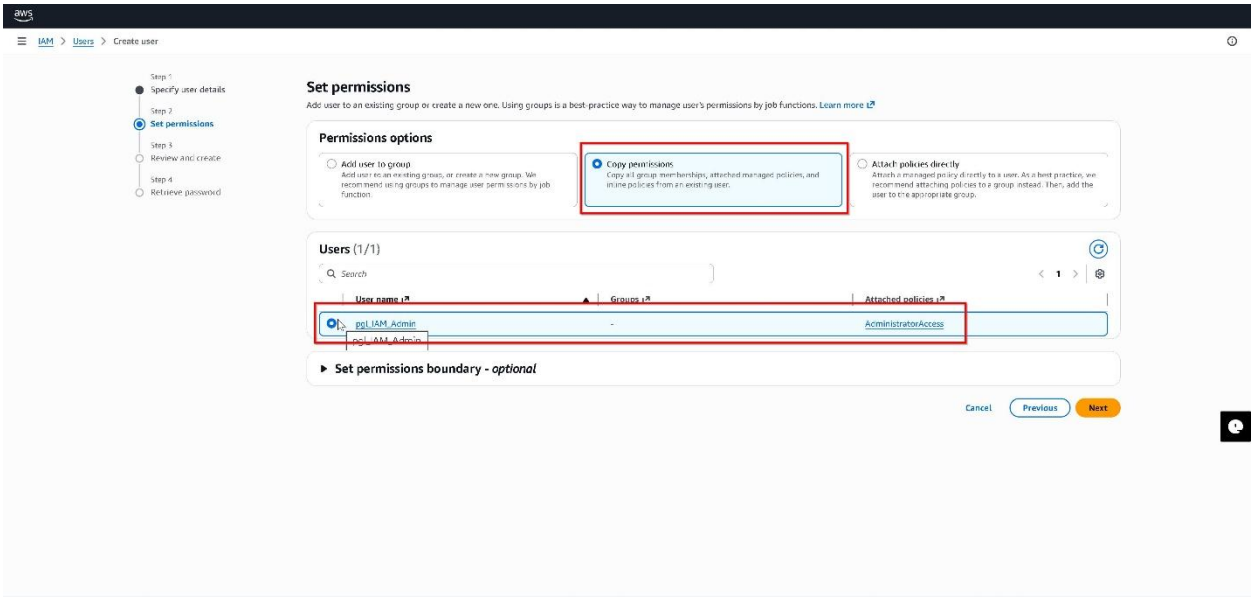


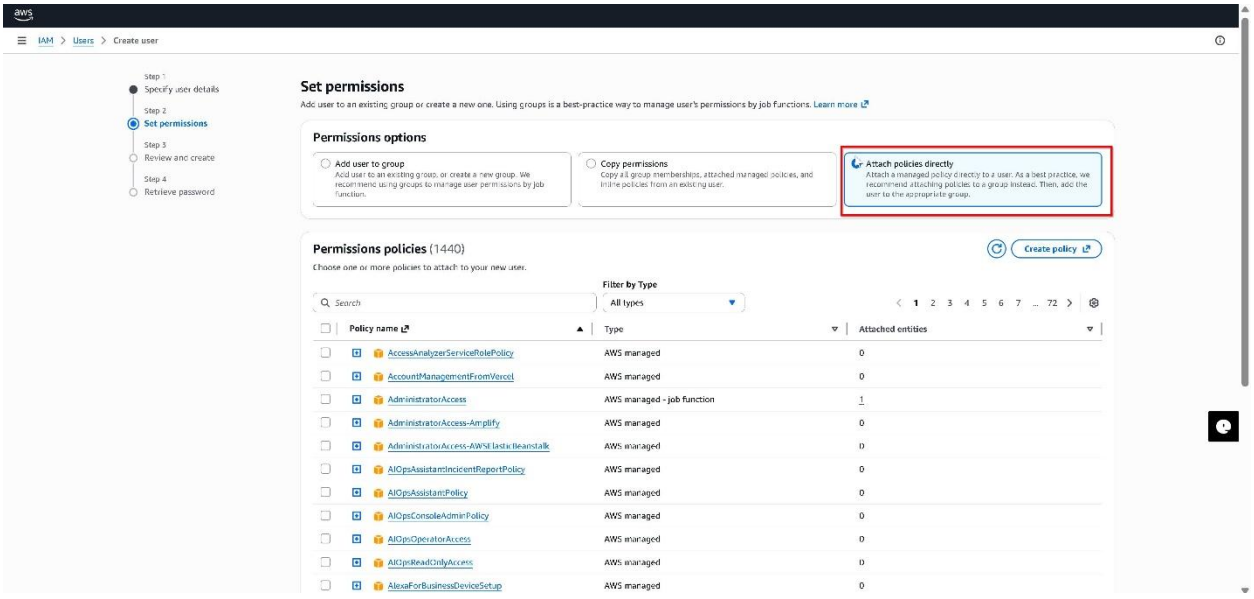


The user creation process is initiated for standard non-administrative users. The wizard follows the same flow as admin user creation but will assign different permissions. Standard users receive only the permissions necessary for their specific job functions following the principle of least privilege.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

Permission Assignment Scenarios



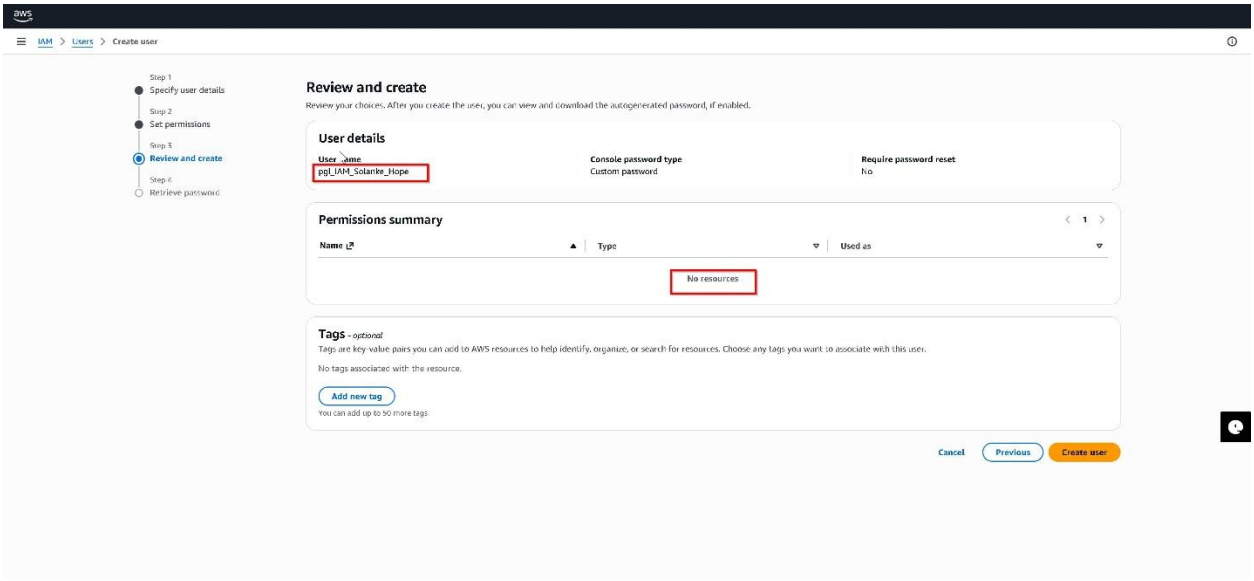




These screenshots demonstrate three different methods for assigning permissions to users. Scenario one shows adding a user directly to the Sales group, which gives the user all permissions attached to that group. Scenario two displays copying permissions from an existing user, useful when creating accounts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

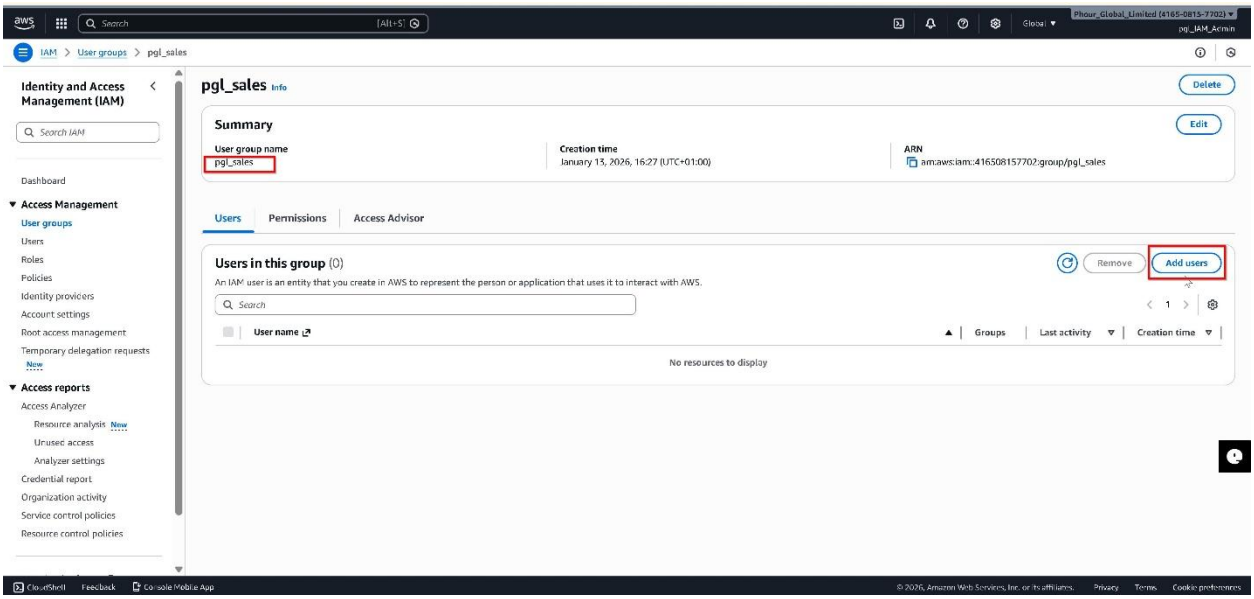
Each permission assignment method has appropriate use cases. Group membership is preferred for standard users as it simplifies management. Direct policy attachment works for unique cases or temporary permission grants. Copying from existing users speeds up provisioning when multiple users need identical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

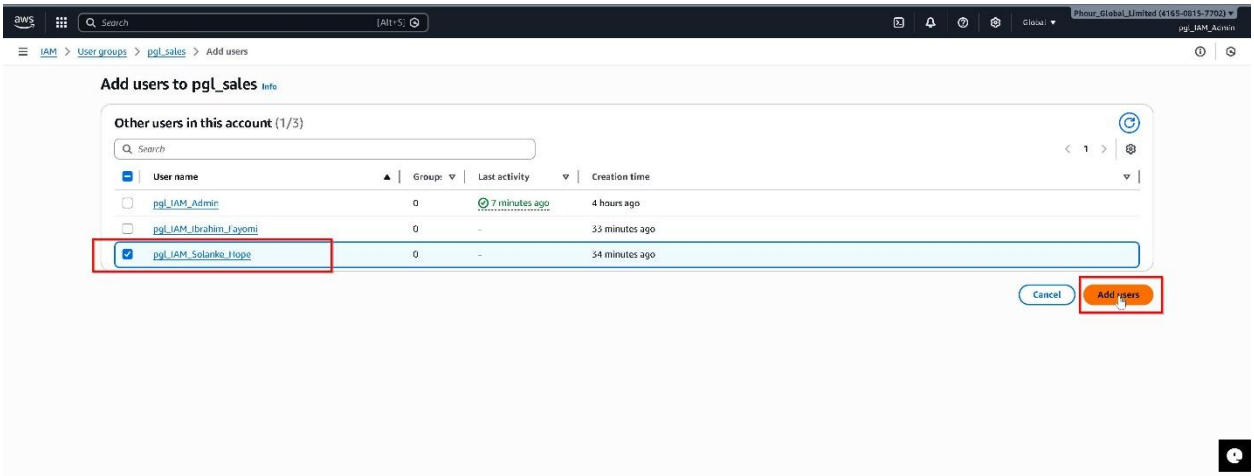
New User Created Without Permissions

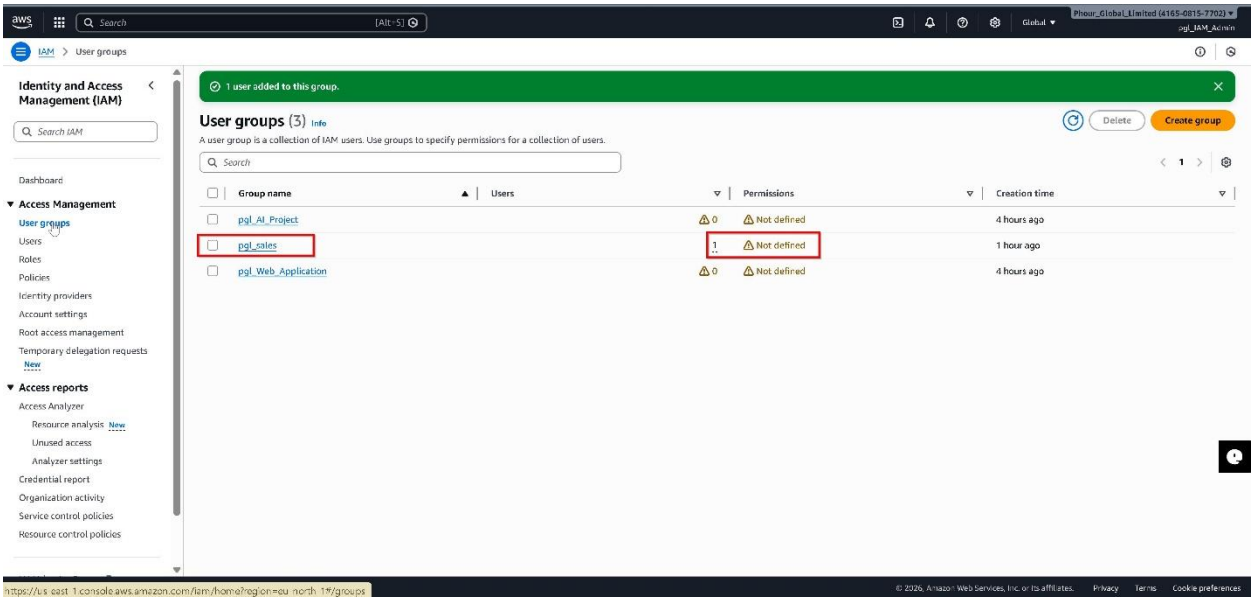


The screenshot shows a user created without any permission assignments. The user exists in the account but cannot access any AWS resources or perform any actions. Permissions can be added after user creation by adding the user to groups or attaching policies. Creating users without initial permissions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

Adding User to Group from User Groups Section

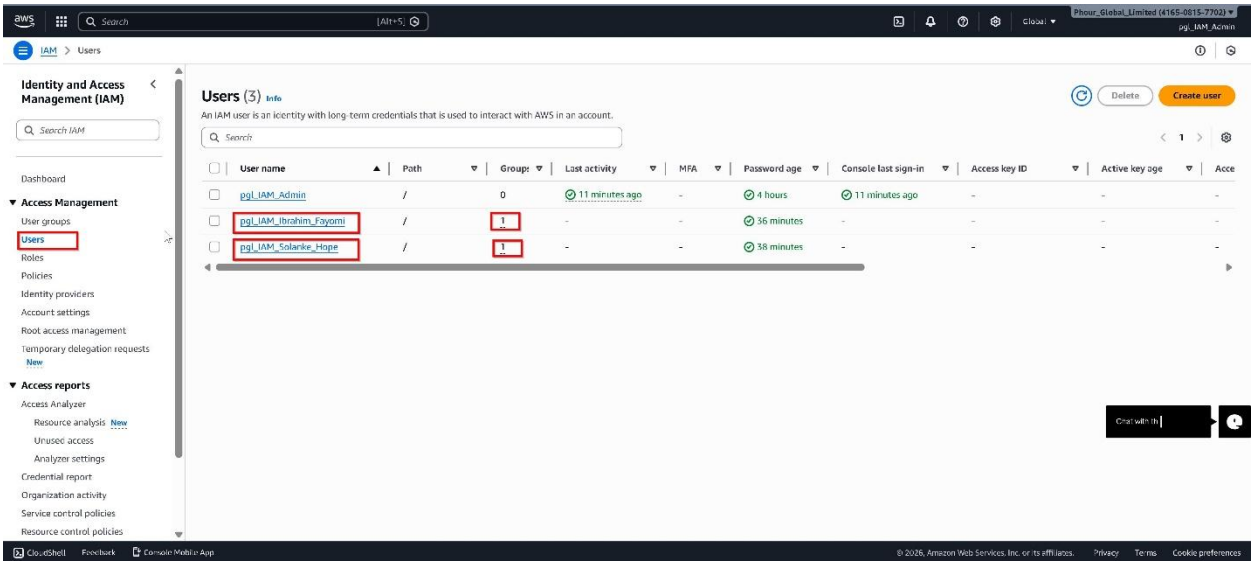






These screenshots show the process of adding an existing user to a group from the groups management interface. The group details page displays current members and provides the option to add users. Selecting users from the account list adds them to the group. The group membership count updates to reflect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

Searching for IAM Service

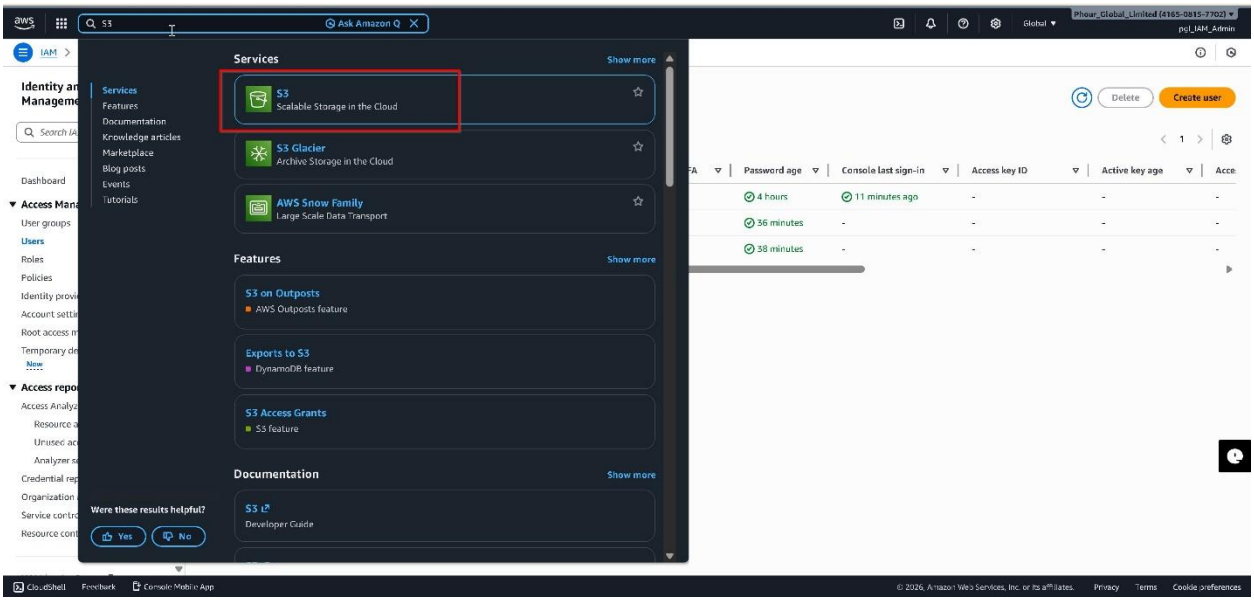


The screenshot demonstrates accessing IAM service through the console search. This shows efficient navigation between AWS services during configuration work. The search bar provides quick access to any service without memorizing menu locations. IAM remains a frequently accessed service during security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

S3 BUCKET

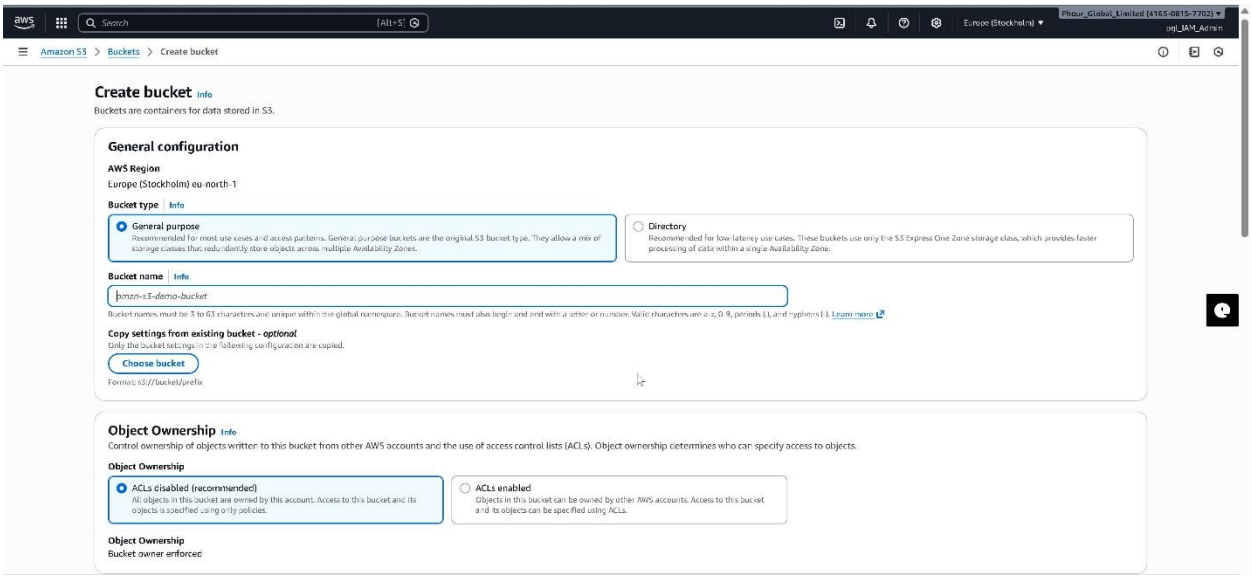
Amazon S3 Bucket Configuration

Adding S3 Bucket to IAM Admin Console



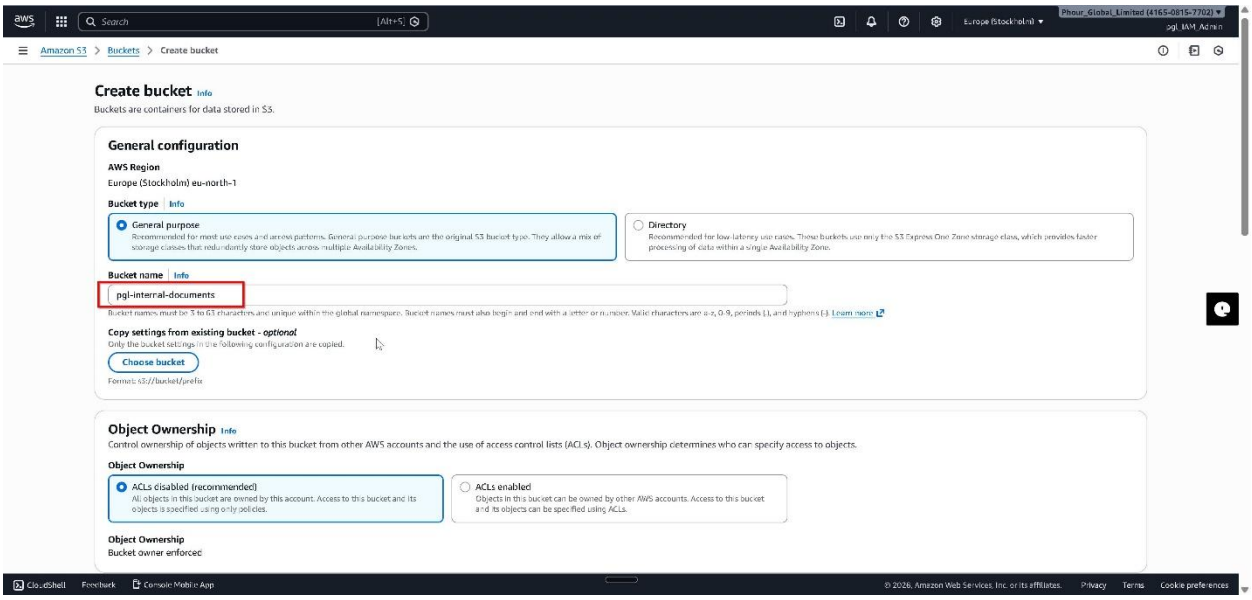
The console shows navigation to the S3 service. S3 provides object storage with high durability and availability. The service is accessed through the console search or services menu. S3 will be used to demonstrate storage security controls including encryption and access policies.

S3 Bucket Dashboard



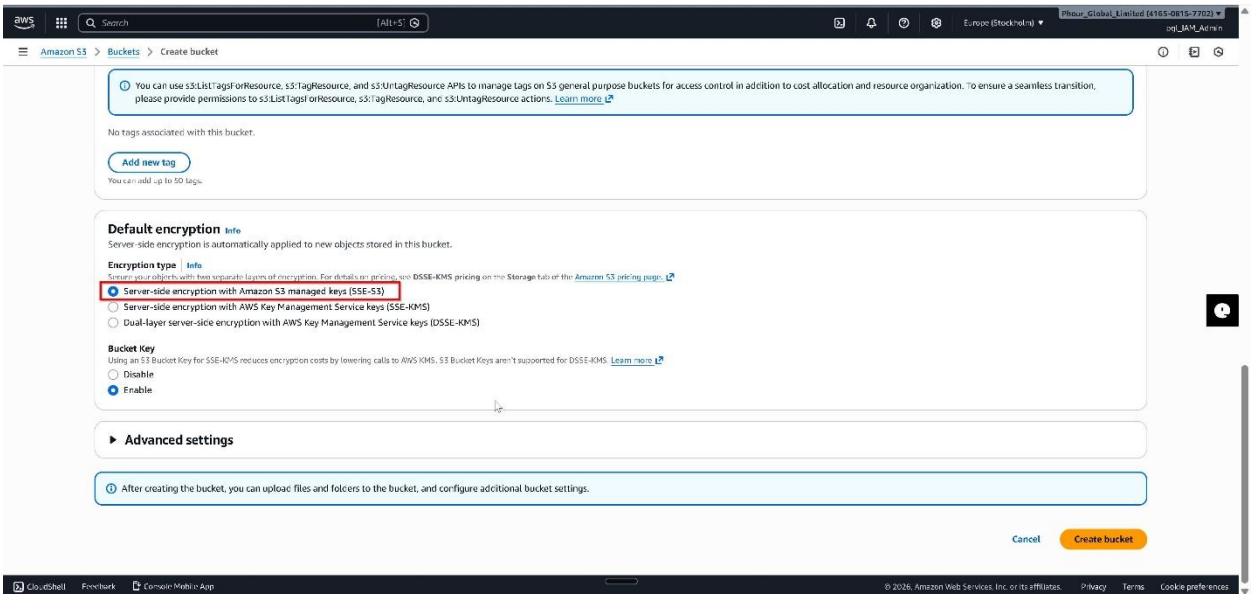
The S3 dashboard displays existing buckets if any and provides options to create new buckets. The interface shows bucket names, regions, and access settings. The create bucket button initiates the bucket creation wizard. S3 bucket names must be globally unique across all AWS accounts.

Naming S3 Bucket



The 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.

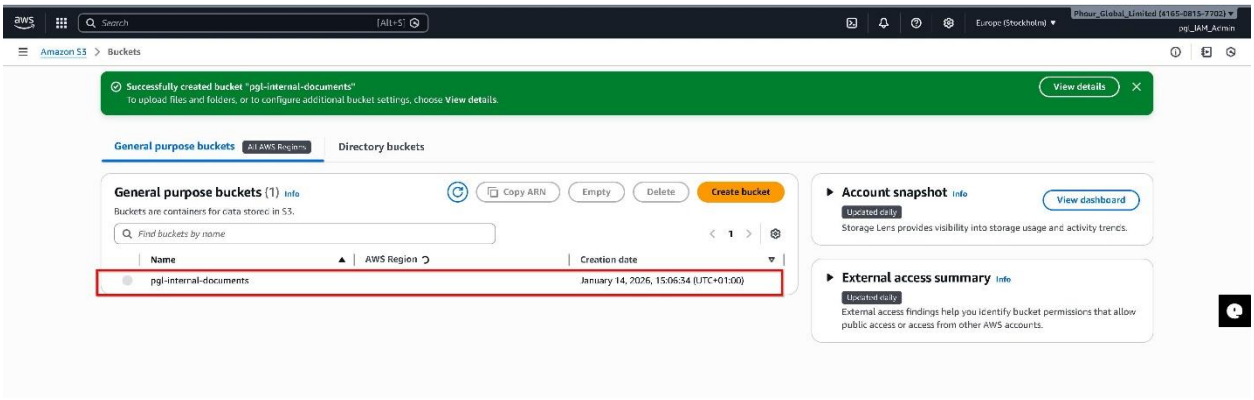
Using Server-Side Encryption with Amazon S3 Managed Keys (SSE-S3)



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.

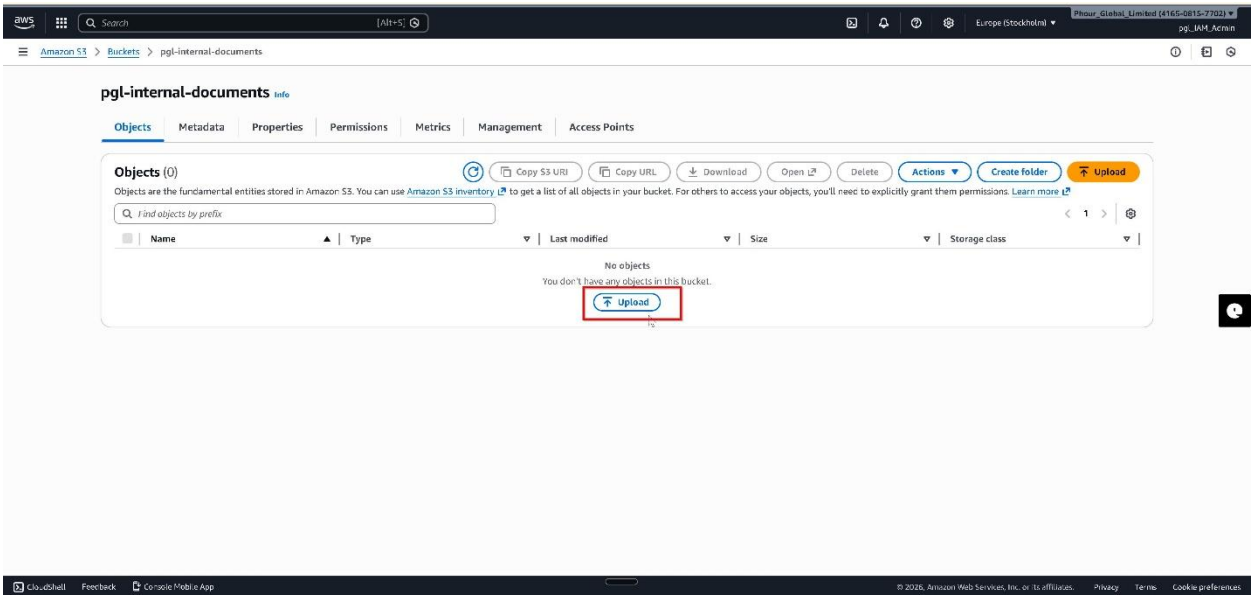
Encryption at rest protects data from unauthorized access if storage media is compromised. SSE-S3 provides strong encryption without the overhead of customer-managed keys. This encryption option is appropriate for most use cases and meets compliance requirements for data protection.

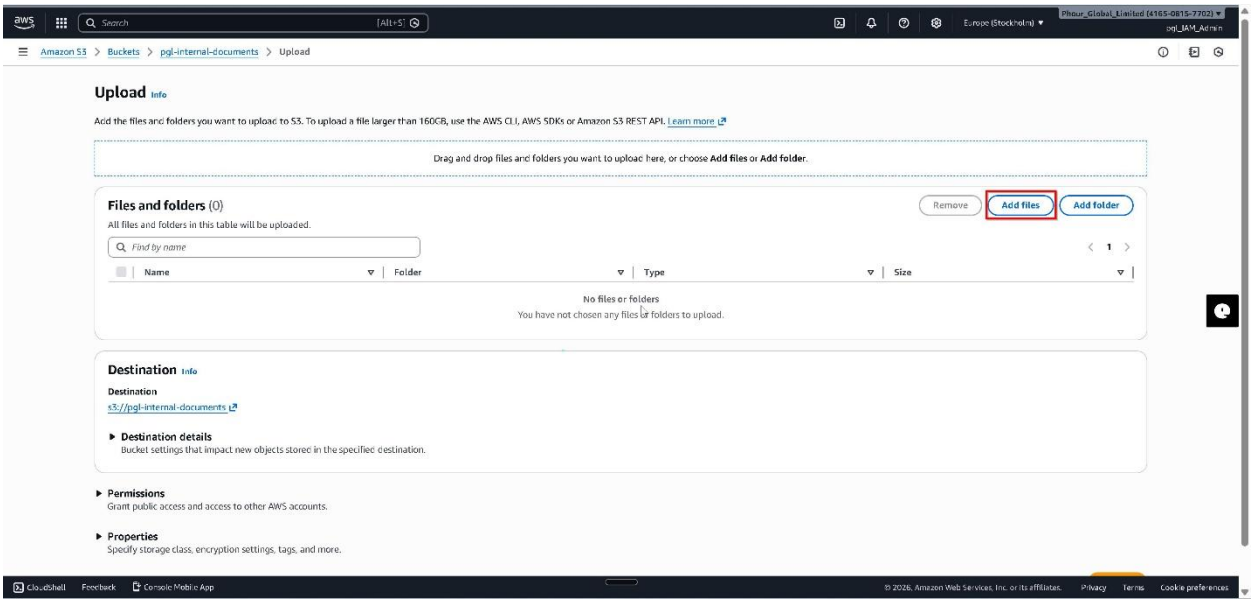
S3 Bucket Created



The 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.

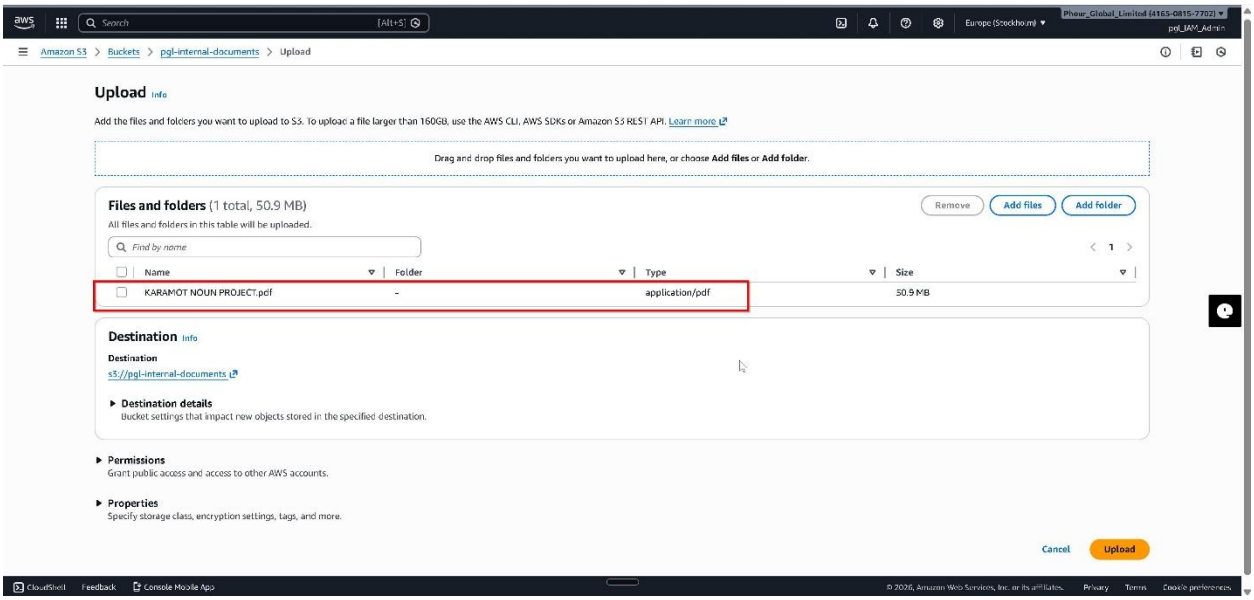
Uploading Documents to S3 Bucket





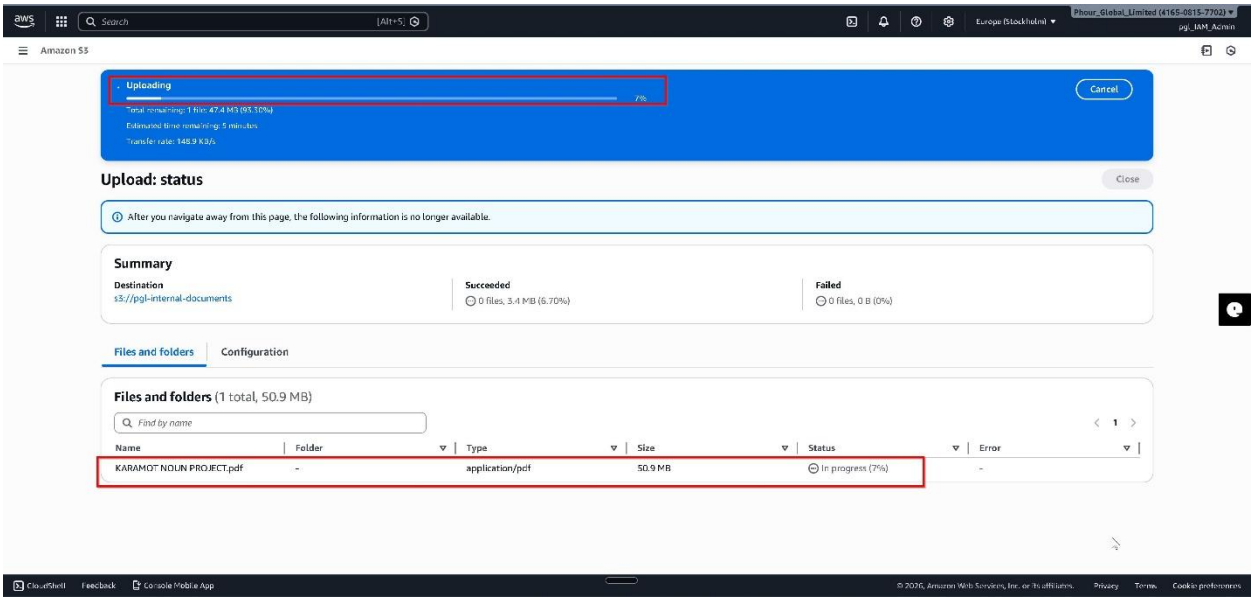
These screenshots show the upload interface within the bucket. The upload button opens a dialog to select files from the local system. Multiple files can be selected and uploaded simultaneously. The interface shows upload progress and confirms successful transfers. Uploaded objects inherit the bucket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

File Uploaded to S3 Bucket



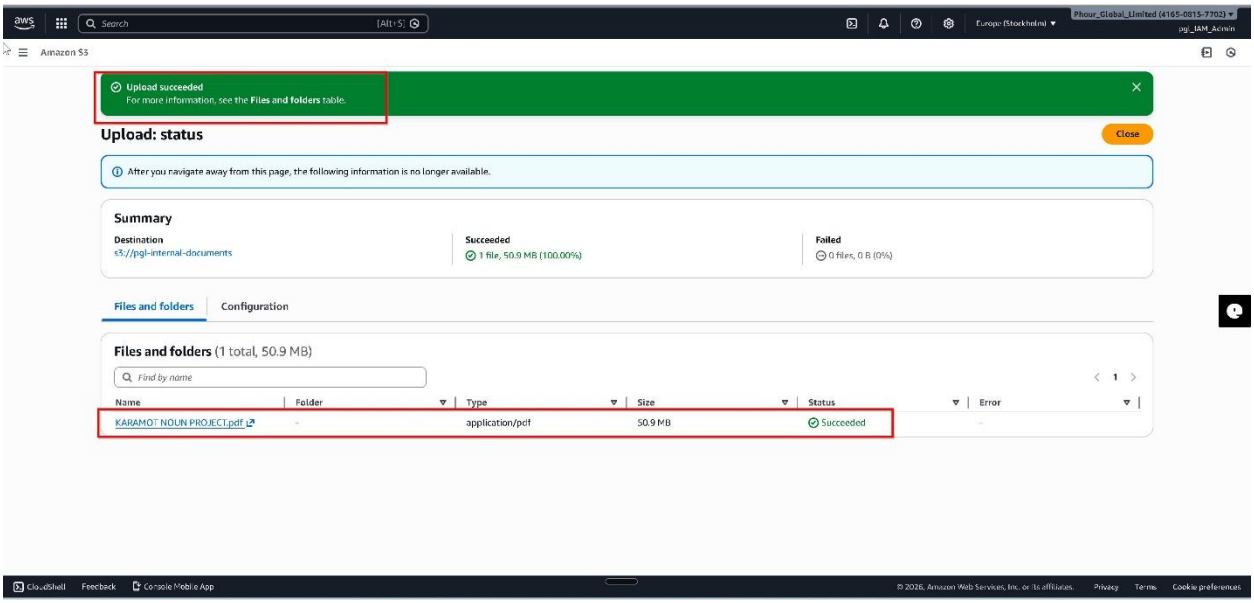
The screenshot shows files listed within the bucket after upload. Object details include name, size, last modified date, and storage class. Clicking an object displays additional metadata and provides options to download, delete, or modify properties. Objects are encrypted at rest using the bucket encryption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

File Uploading in Progress



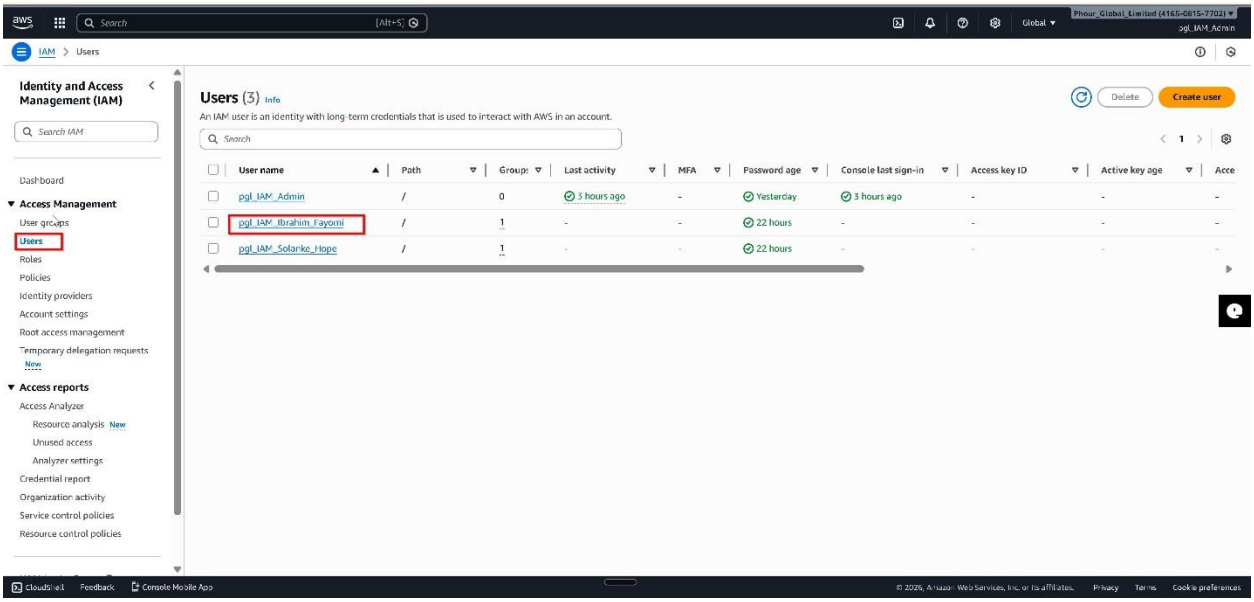
The upload progress interface displays during file transfer. Progress bars show upload status for each file. Large files may take time to transfer depending on connection speed. The interface allows canceling uploads if needed. Failed uploads can be retried without affecting successfully uploaded objects.

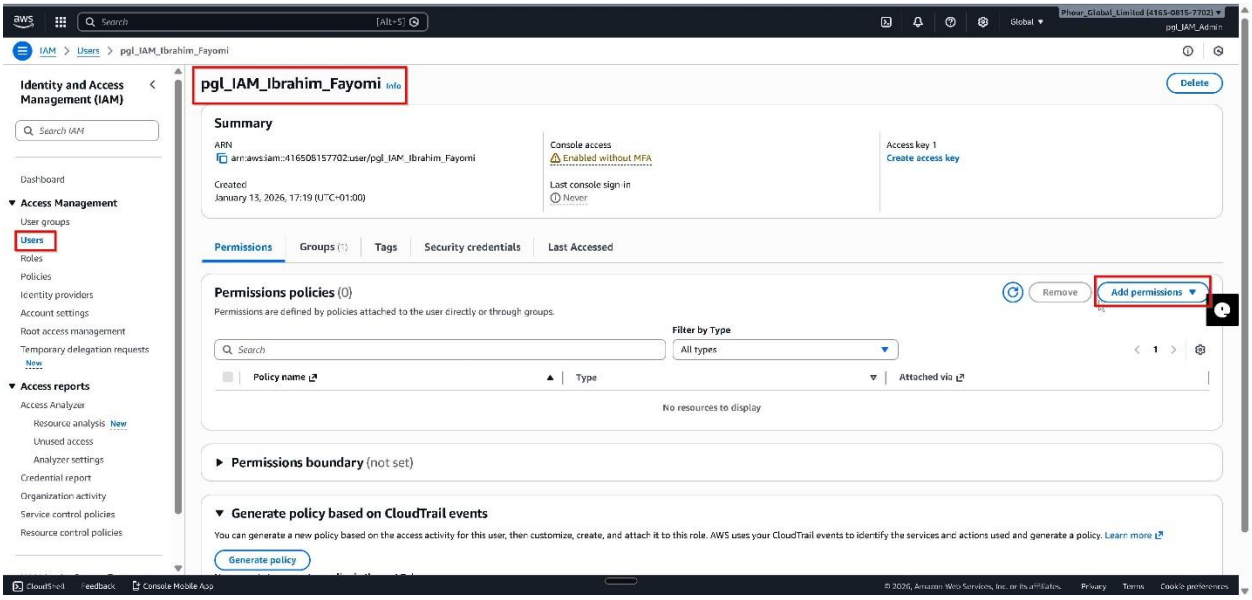
File Upload to S3 Bucket Successful

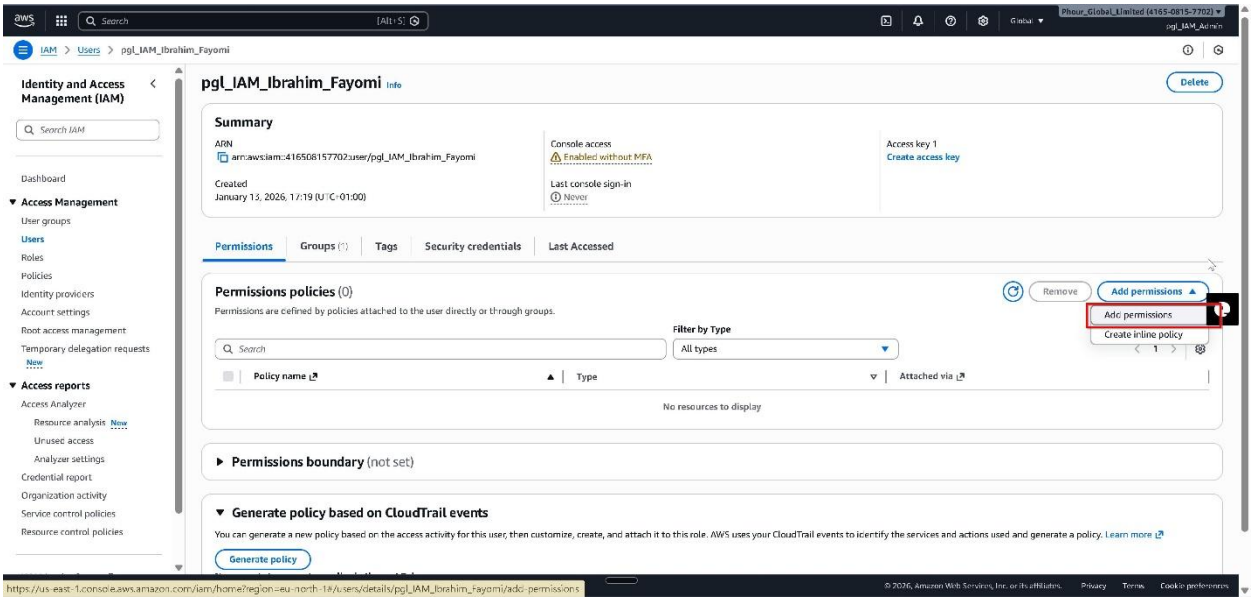


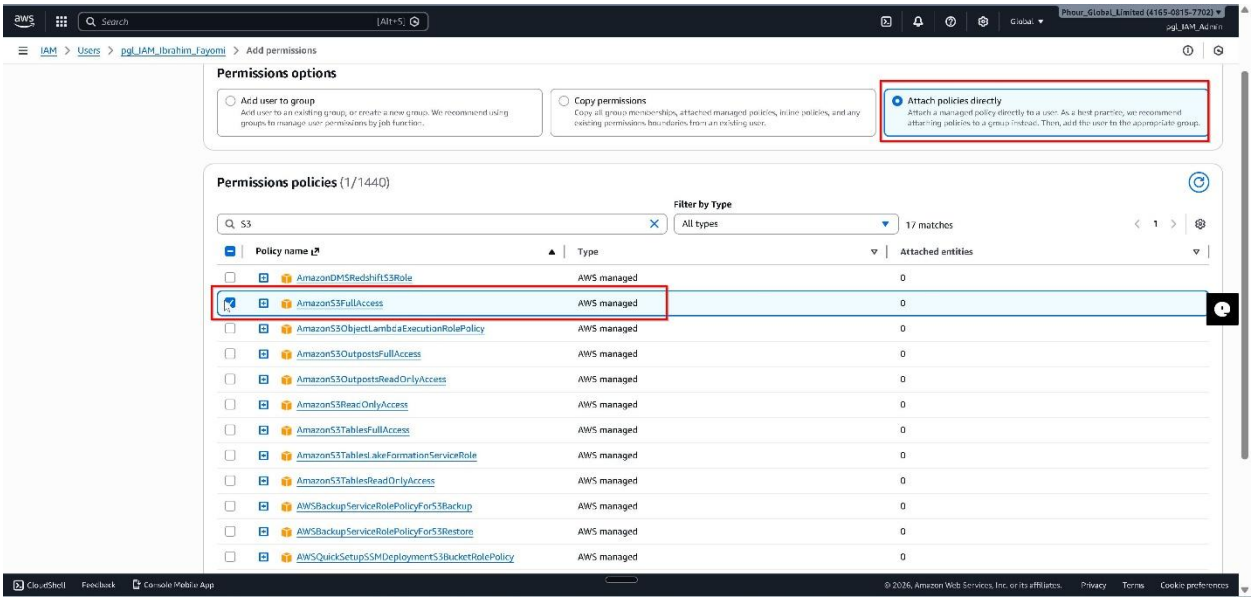
The success confirmation shows uploads completed. Objects are now stored in S3 with configured encryption and durability guarantees. The objects can be accessed by authorized users based on bucket policies and IAM permissions. S3 provides eleven nines of durability through redundant storage across multiple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

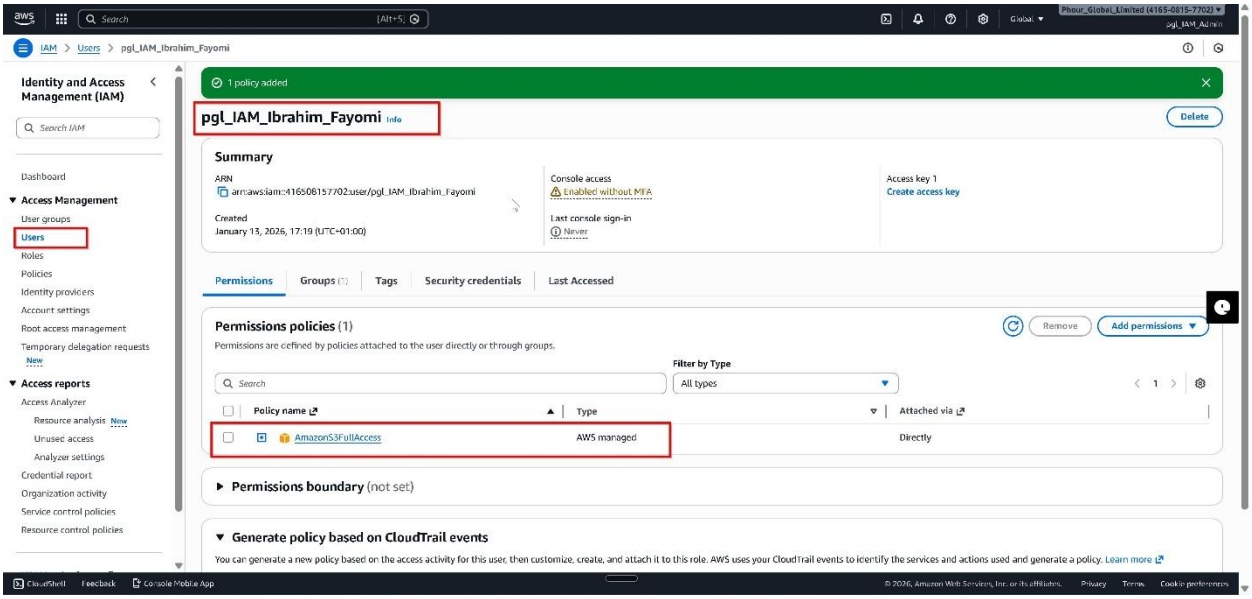
Assigning S3 Full Access Policy to User






IAM > Users > pgl_IAM_Ibrahim_Fayomi
Identity and Access Management (IAM)
pgl_IAM_Ibrahim_Fayomi Info
Delete
Summary
ARN
arn:aws:iam::416508157702:user/pgl_IAM_Ibrahim_Fayomi
Console access
Enabled without MFA
Access key 1
Create access key
Created
January 16, 2026, 17:19 (UTC-01:00)
Last console sign-in
Never
Permissions
Groups (1)
Tags
Security credentials
Last Accessed
Permissions policies (0)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Add permissions
Create inline policy
Filter by Type
Search
All types
Policy name
Type
Attached via
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

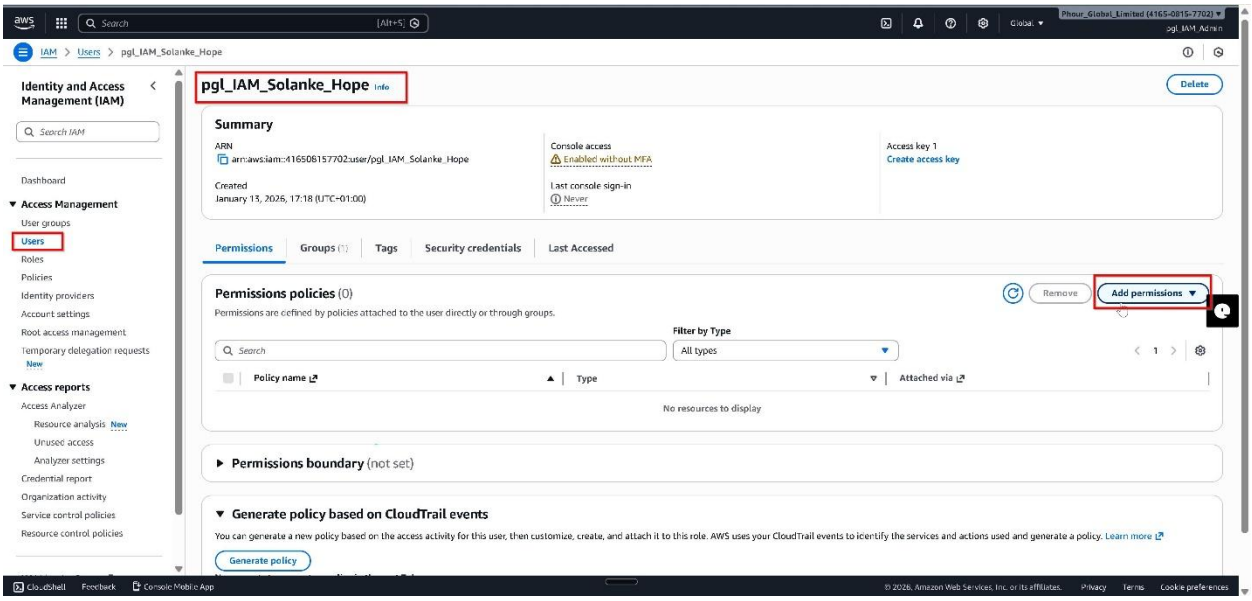

IAM > Users > pgl_IAM_Ibrahim_Fayomi > Add permissions
Permissions options
Add user to group
Copy permissions
Attach policies directly
Permissions policies (1/1440)
Filter by Type
S3
All types
17 matches
Policy name
Type
Attached entities
AmazonDMSRedshiftS3Role
AmazonS3FullAccess
AmazonS3ObjectLambdaExecutionRolePolicy
AmazonS3OutpostsFullAccess
AmazonS3OutpostsReadOnlyAccess
AmazonS3ReadOnlyAccess
AmazonS3TablesFullAccess
AmazonS3TablesLakeFormationServiceRole
AmazonS3TablesReadOnlyAccess
AWSBackupServiceRolePolicyForS3Backup
AWSBackupServiceRolePolicyForS3Restore
AWSQuickSetupSSMDeploymentS3BucketRolePolicy
AWS managed
0
CloudShell
Feedback
Console Mobile App

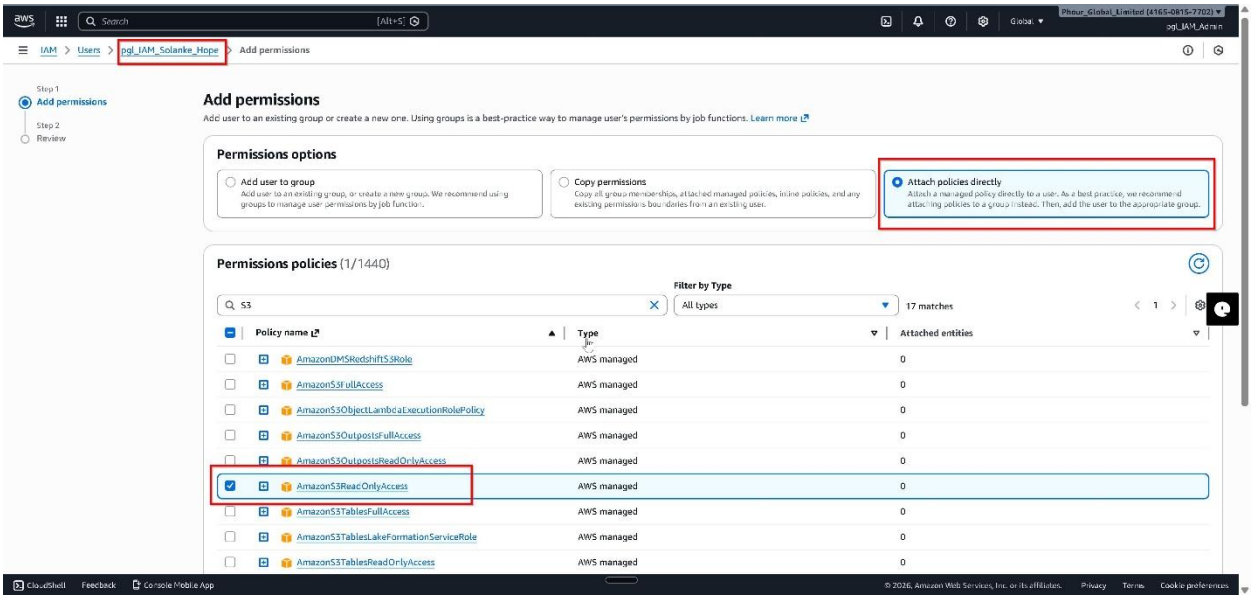


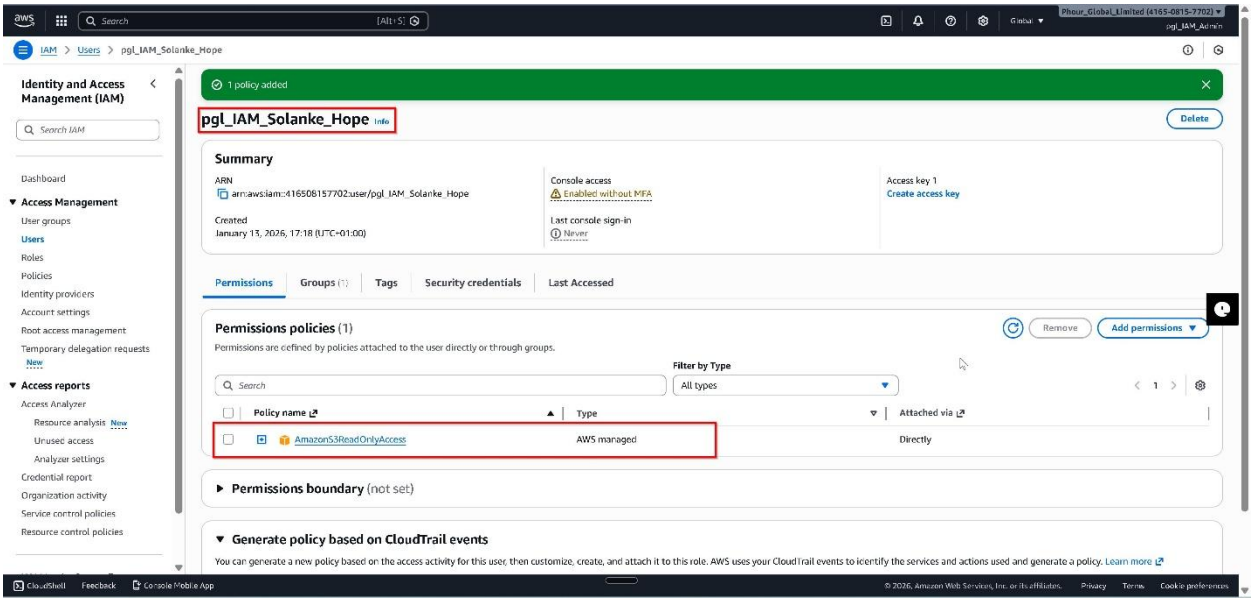
These screenshots document assigning S3 permissions to a user. The IAM user configuration page shows the permissions tab where policies can be attached. The AmazonS3FullAccess managed policy is selected from the policy list. This policy grants complete access to all S3 buckets and objects in the account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



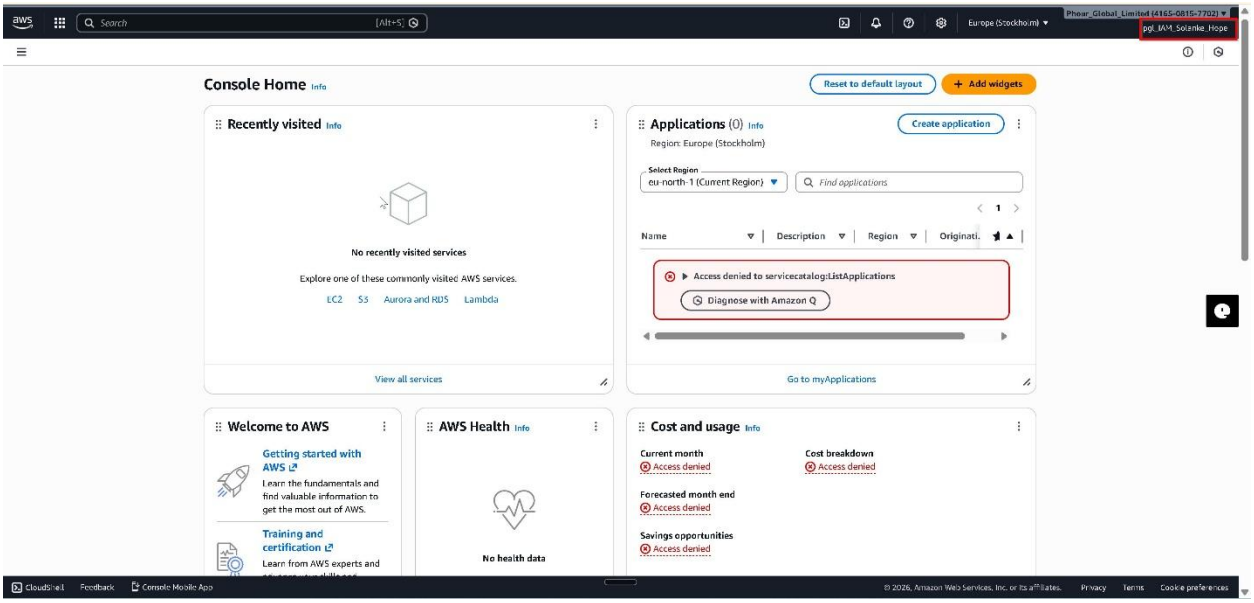




The screenshots show attaching AmazonS3ReadOnlyAccess policy to a different user. This managed policy allows listing buckets and reading objects but prevents any modifications. The user can download files and view bucket configurations but cannot upload, delete, or change settings. Read-only access is appropriate for users who need to access data but should not modify it.

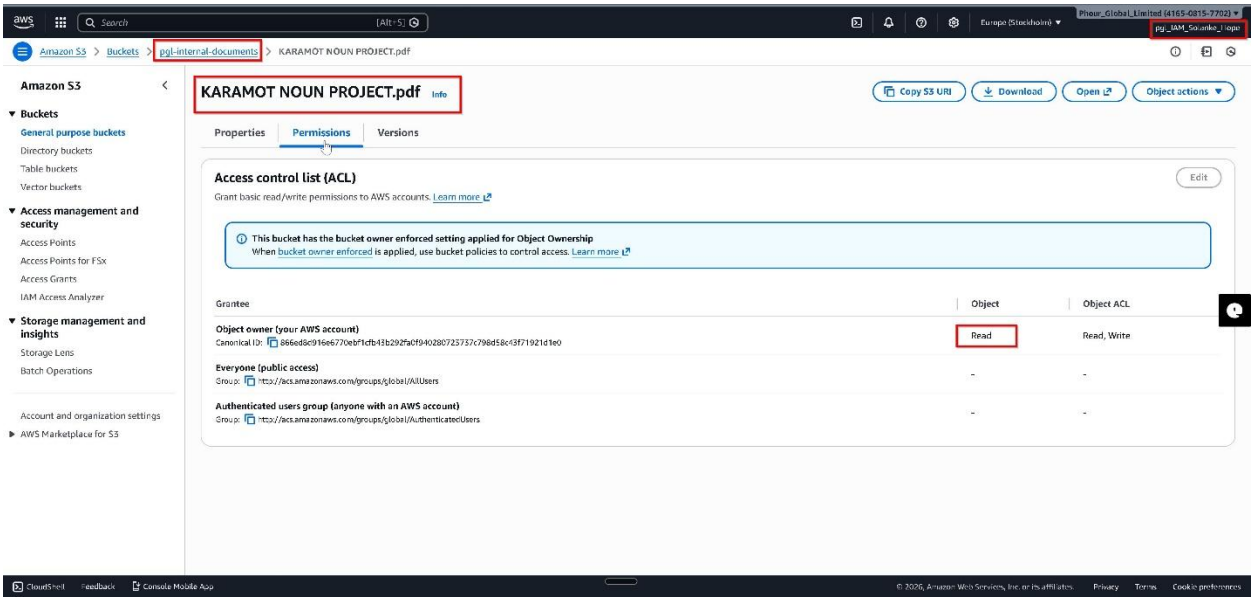
Restricting users to read-only access prevents accidental or malicious data deletion. This permission model supports compliance requirements for separation of duties where some users manage data while others only consume it.

Logged in as Read-Only User



The 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



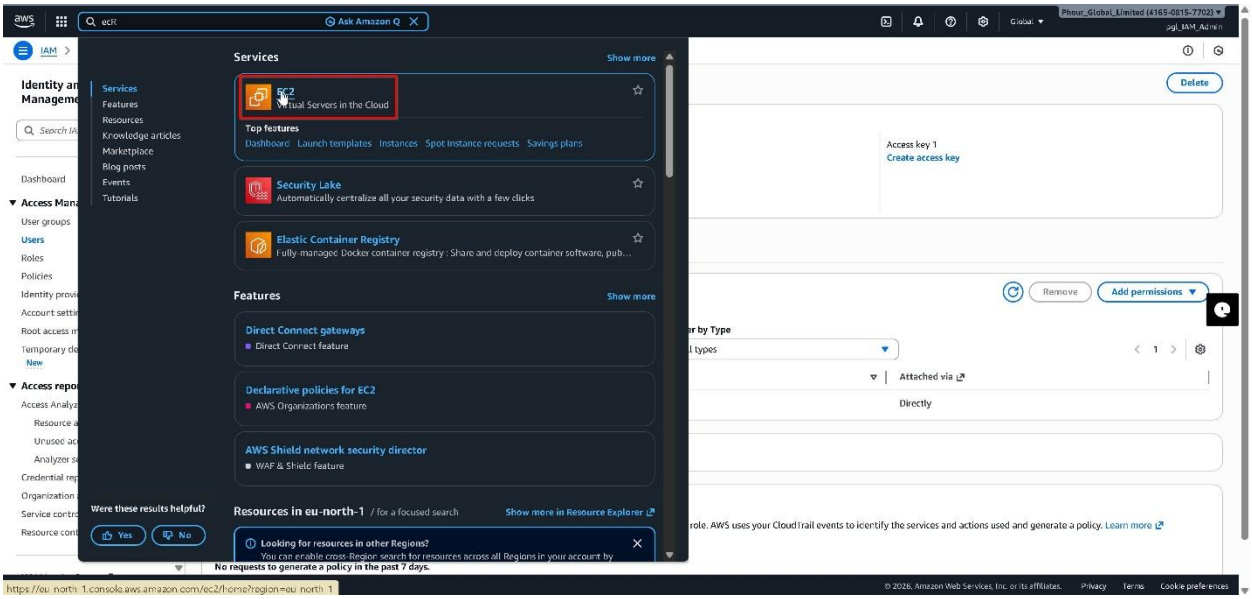
The screenshot shows the read-only user accessing the S3 bucket. The user can see bucket contents and download objects. The upload button is either disabled or will fail if clicked due to insufficient permissions. This demonstrates that the read-only policy is correctly enforced. Attempting to upload,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

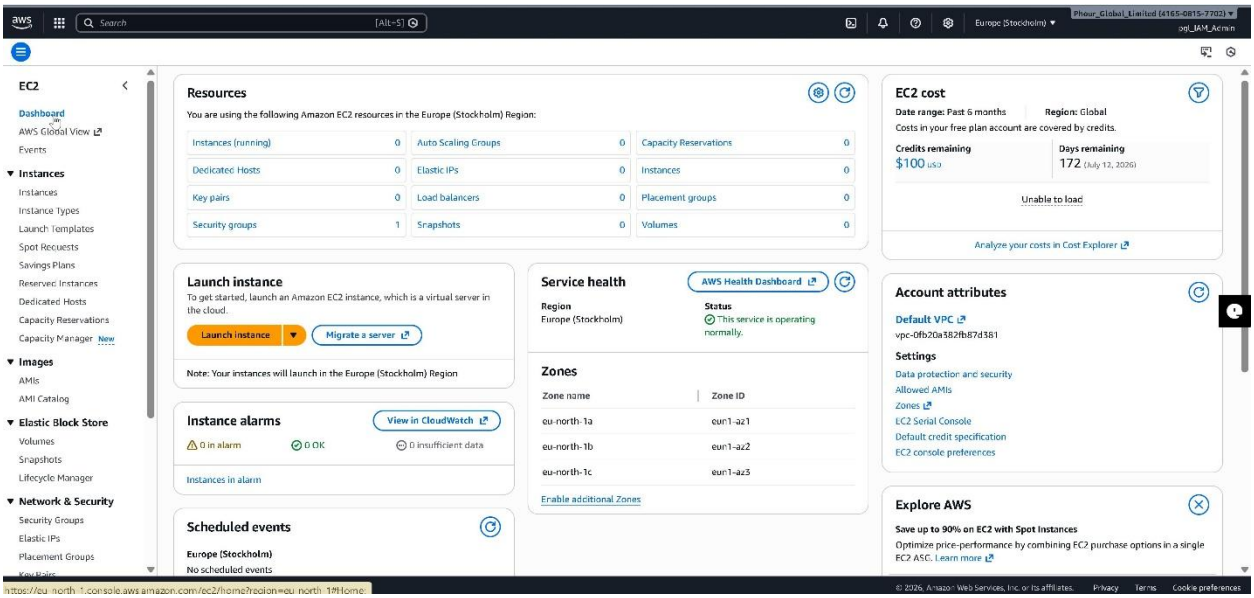
EC2 Instance Configuration

Accessing EC2 Service



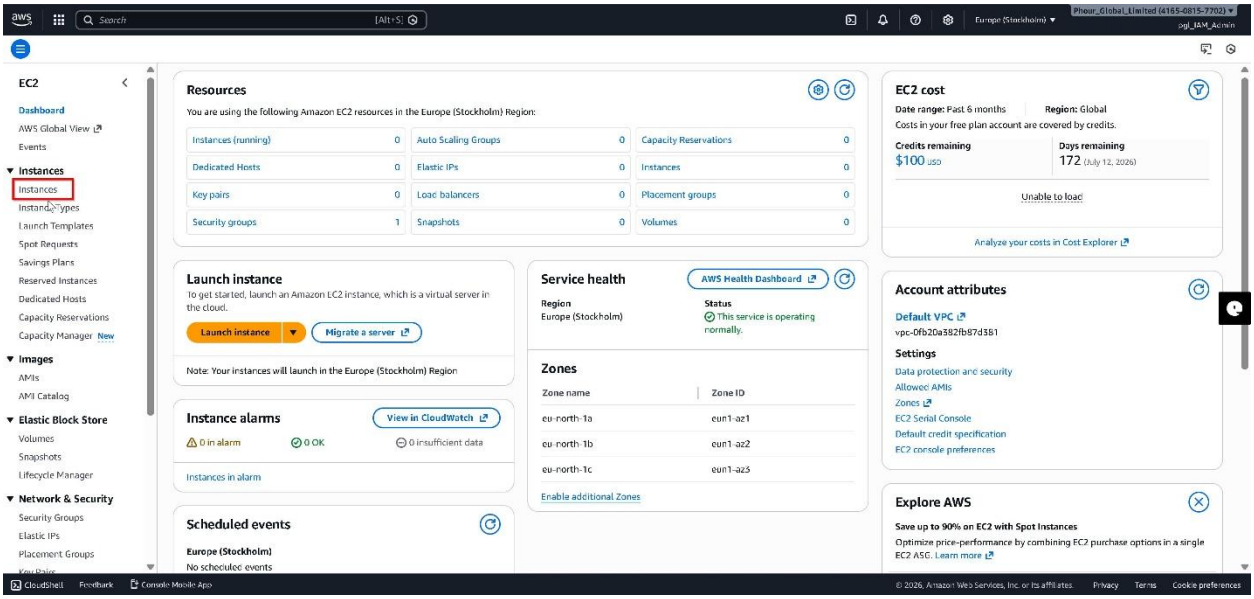
The console shows navigation to EC2 service. EC2 provides resizable compute capacity in the cloud through virtual servers called instances. The service supports various instance types optimized for different workloads. EC2 instances can run multiple operating systems and are the foundation for many cloud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

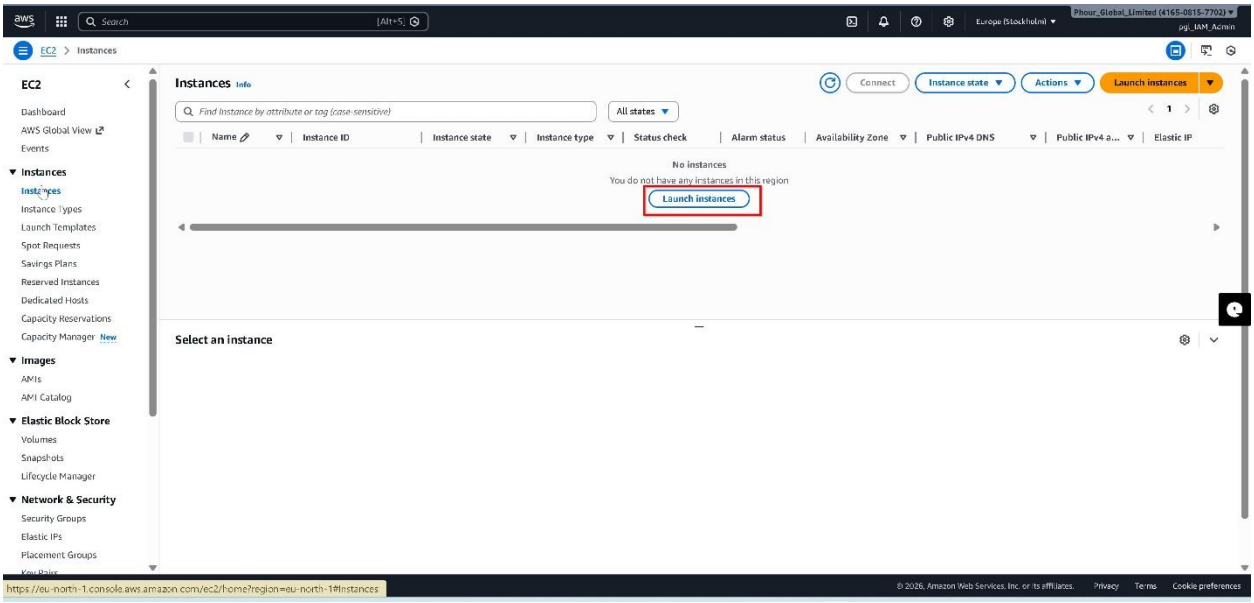
EC2 Dashboard



The 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.

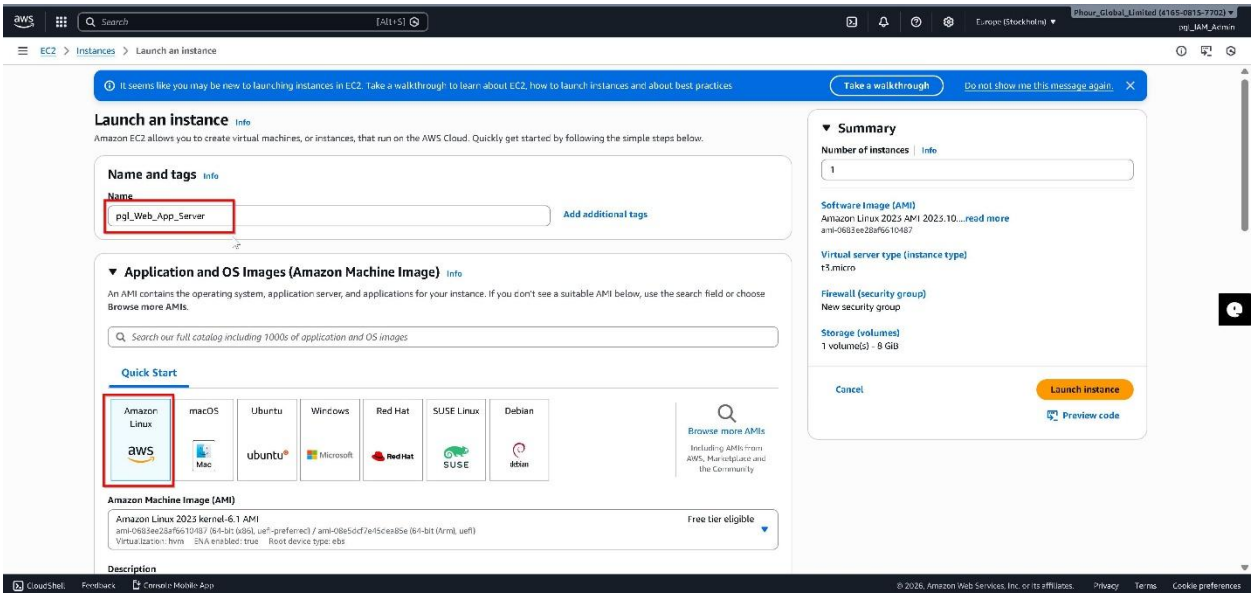
Creating EC2 Instance





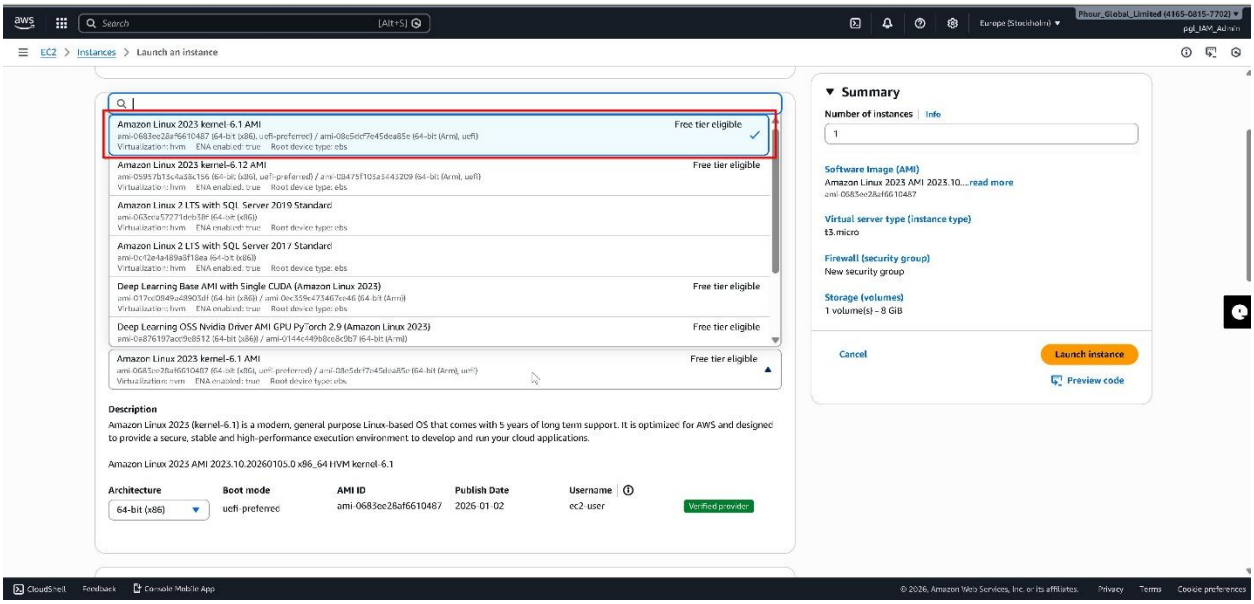
The launch instance wizard begins with instance configuration options. The interface provides a streamlined path to launch instances with common configurations. Advanced options allow detailed customization of network, storage, and security settings. The wizard guides through all required configuration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

Naming Instance and Selecting Operating System



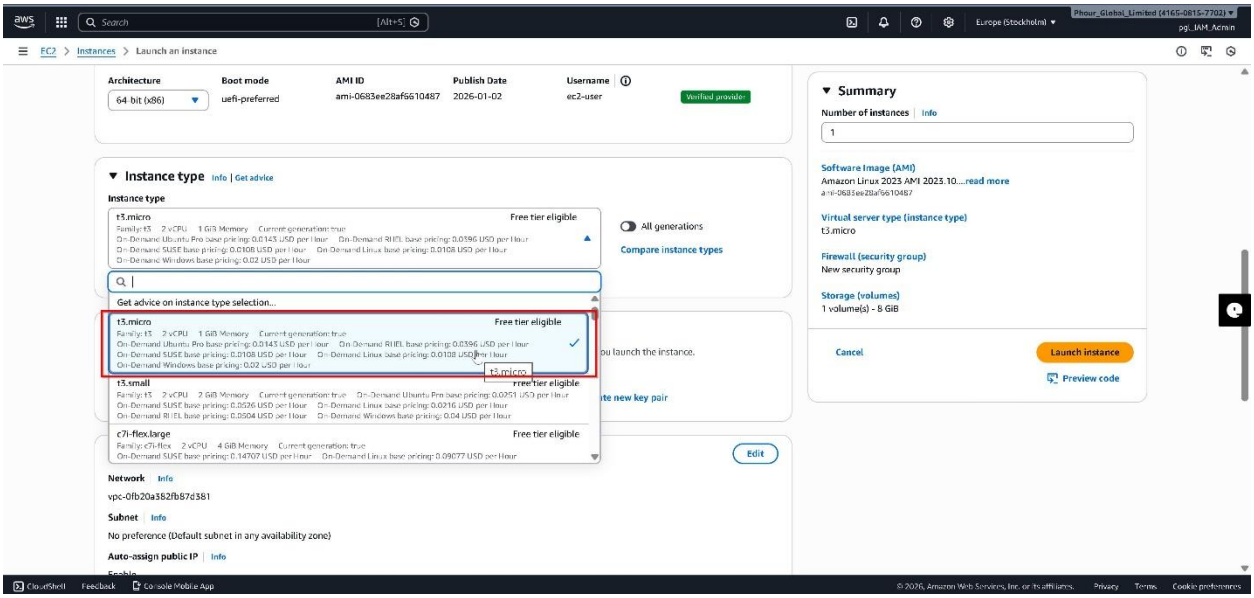
The screenshot shows naming the instance with a descriptive identifier such as Web App Server. Instance names help identify resources in the console and billing reports. The name is applied as a tag and does not affect technical configuration. Clear naming conventions simplify resource management in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

Choosing AMI and OS Version



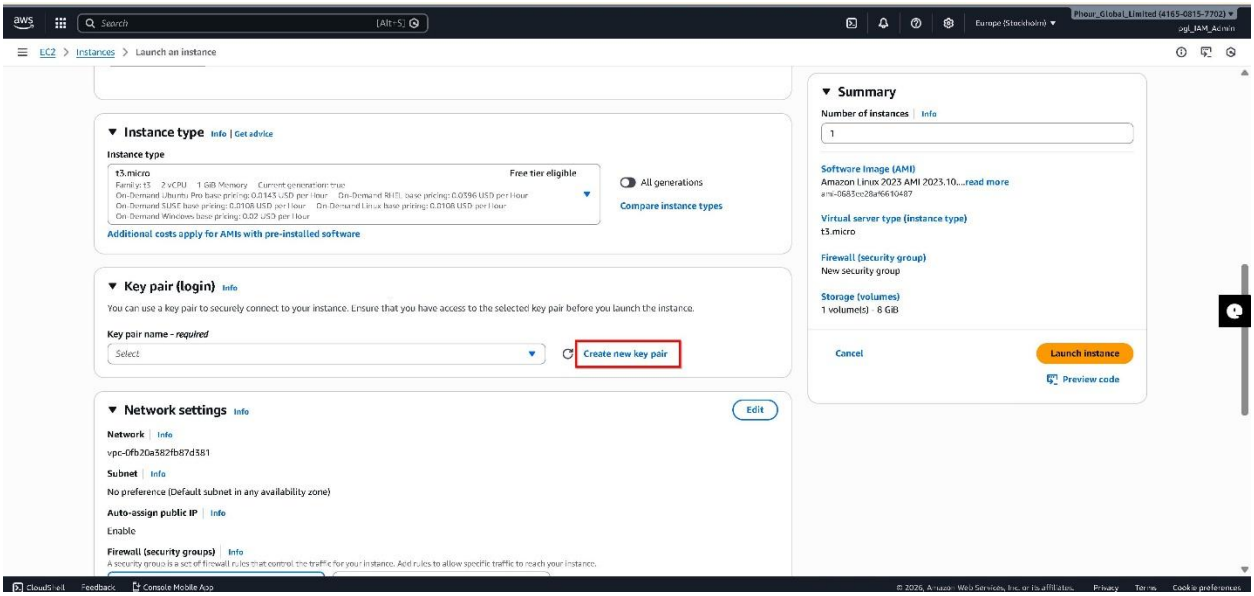
The AMI selection screen displays available machine images. Amazon Linux 2 is selected which provides a secure, stable, and high-performance execution environment. The AMI includes AWS integration and optimization. Amazon Linux receives security updates and patches from AWS. The OS selection determines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

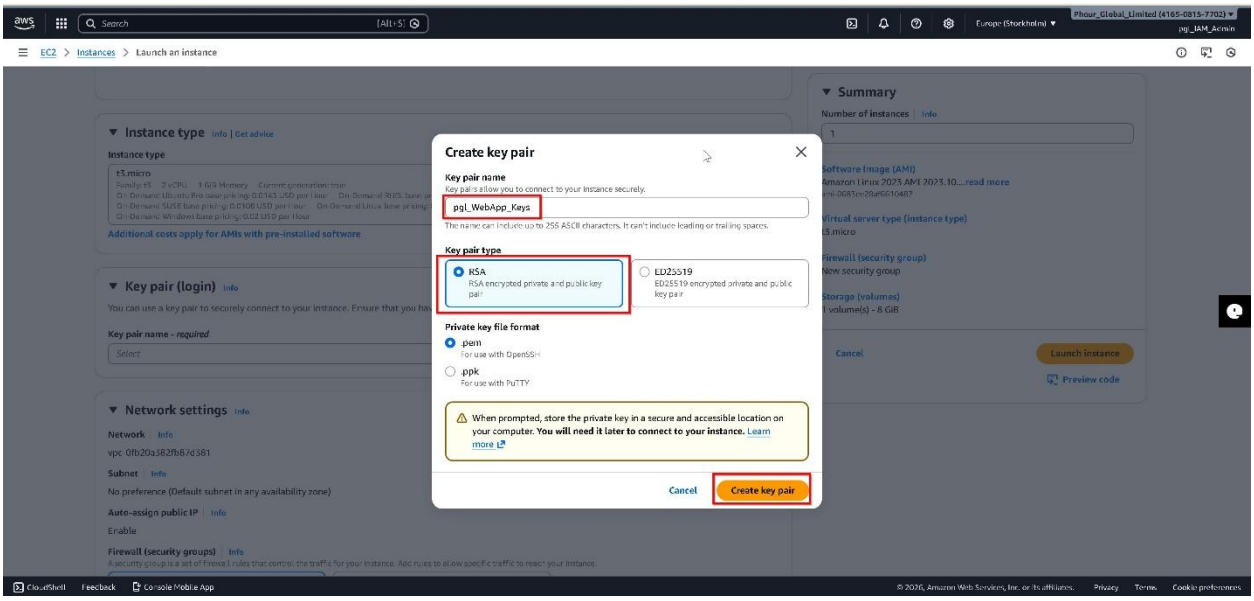
Choosing Instance Type



The 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

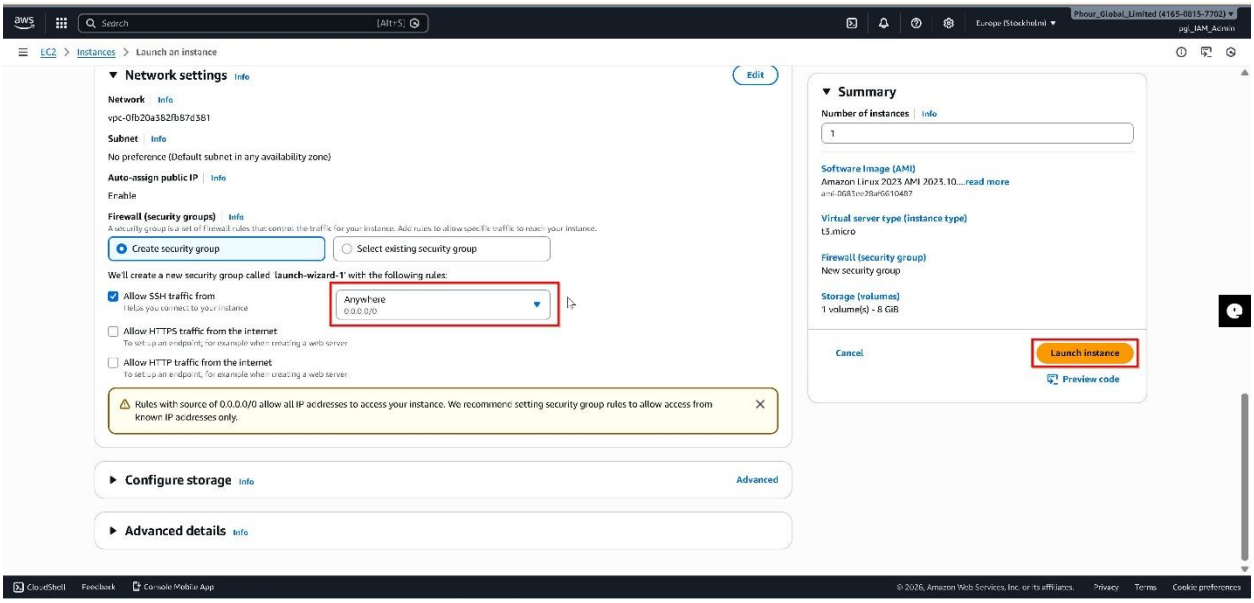




The key pair configuration prompts for creating or selecting an SSH key pair. A new key pair is created with a descriptive name. The RSA encryption type is selected which provides asymmetric encryption for SSH authentication. The private key file is downloaded and must be stored securely as it cannot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

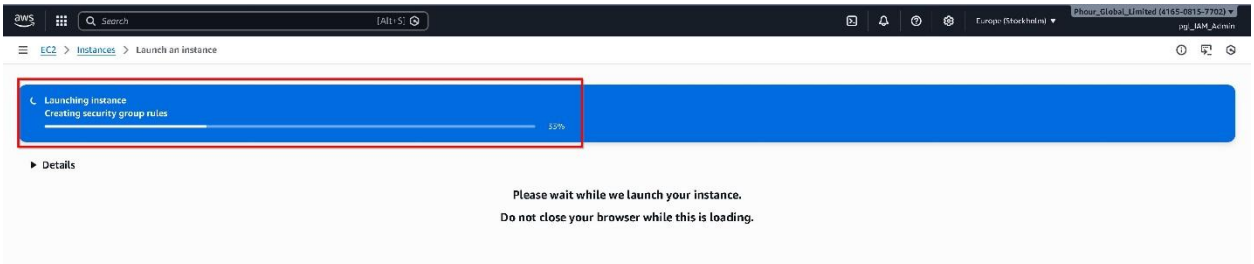
Key pair authentication provides more secure access than password authentication. The private key must be protected as anyone with access to it can connect to the instance. Lost private keys cannot be recovered and require creating a new key pair and updating instance access.

Choosing Network Access and Launching Instance



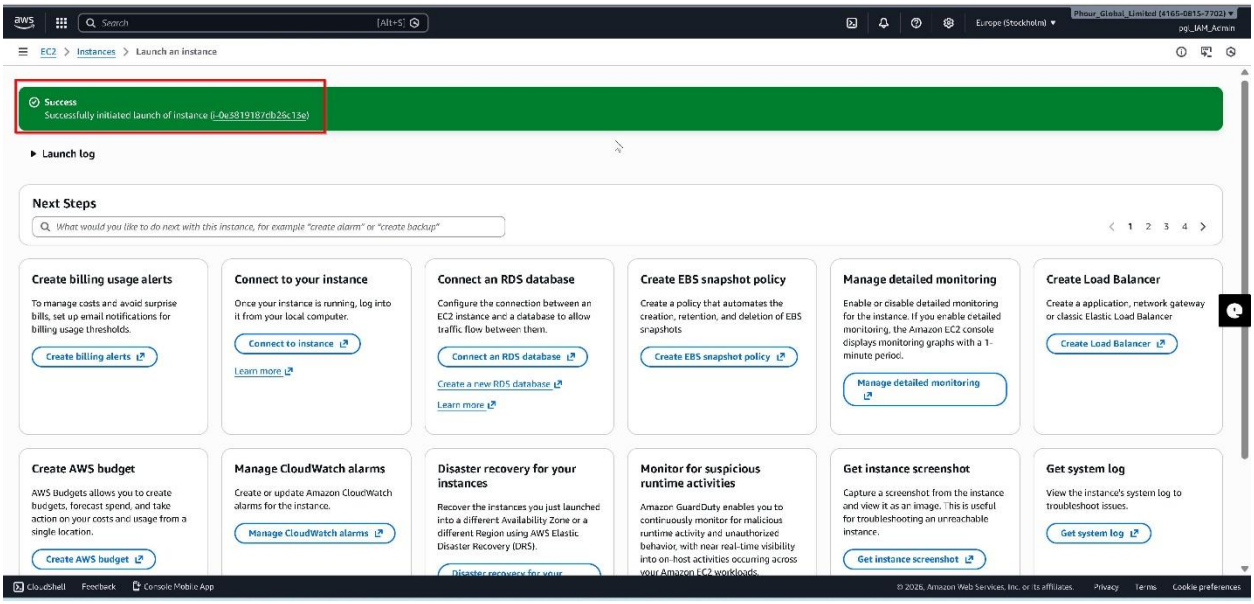
The network settings section configures security group rules controlling inbound traffic. The option to allow SSH access from anywhere is selected which permits connections from any IP address. This provides maximum accessibility but reduces security. Production instances should restrict SSH access to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

Instance Launch Loading



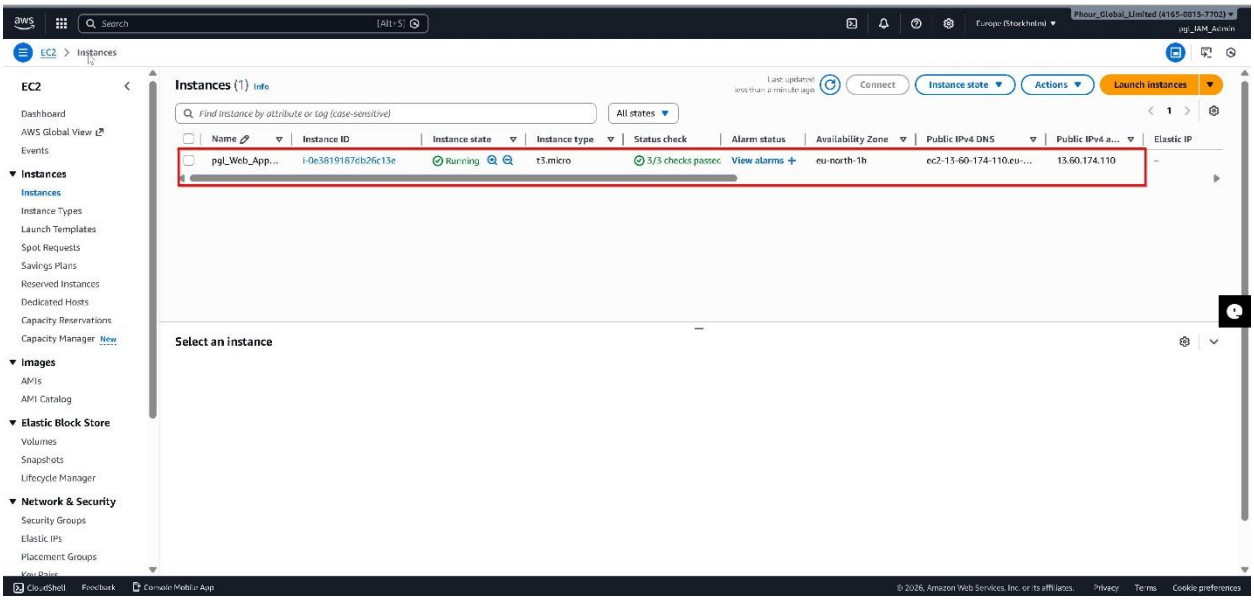
The progress indicator shows instance launch in progress. AWS allocates compute resources, attaches networking, and initializes the operating system. Launch typically completes within a few minutes. The instance will appear in the instances list once initialization finishes.

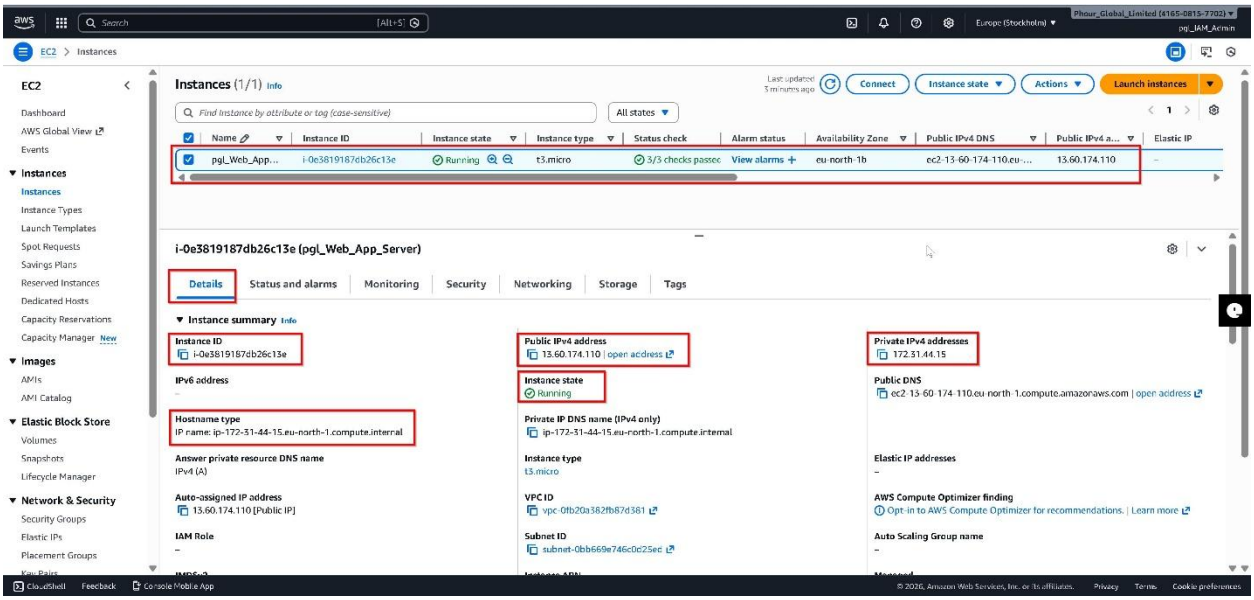
Instance Launch Successful



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.

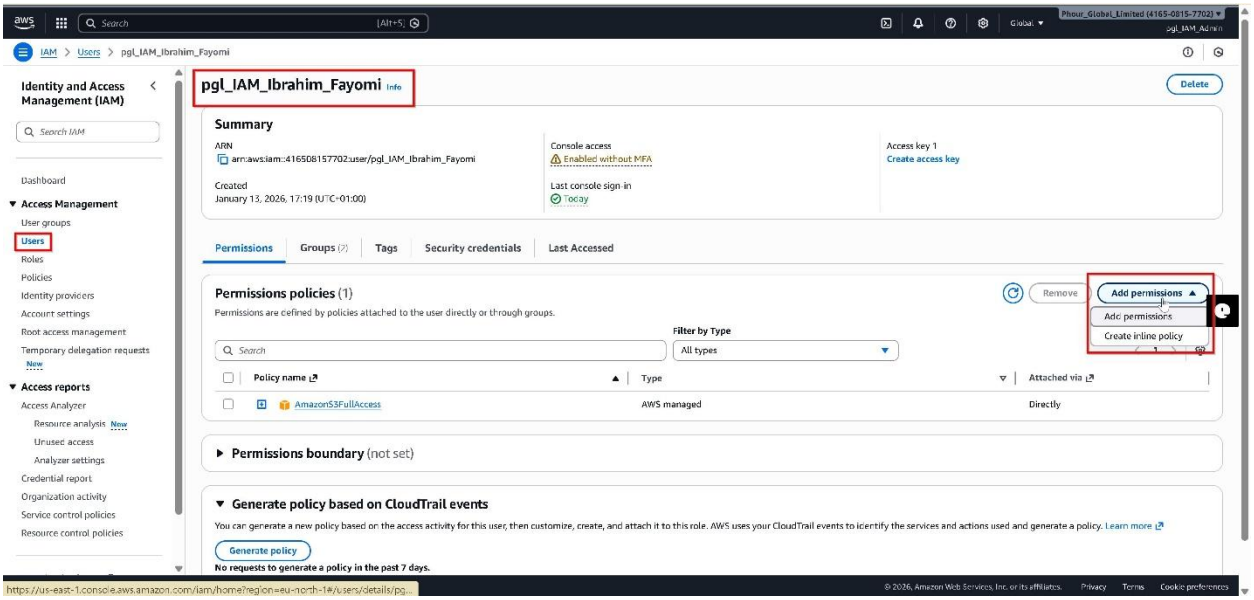
Instance Running





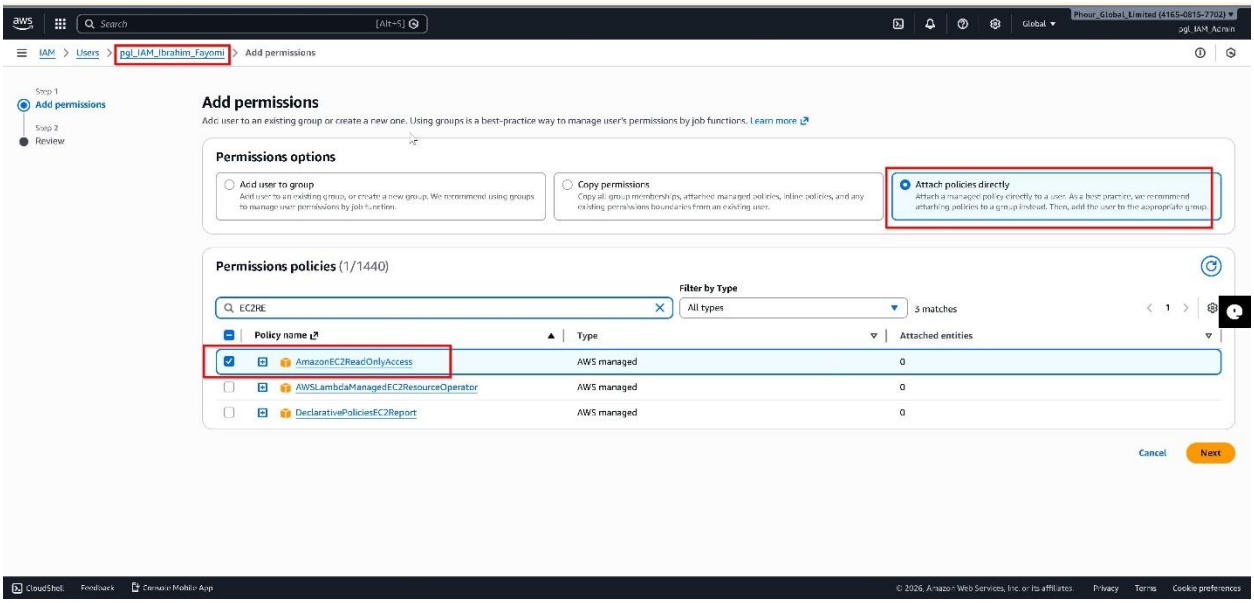
The instances view shows the Web App Server instance in running state. Instance details display public and private IP addresses, instance type, security groups, and monitoring status. The instance is now accessible via SSH using the private key. The public IP address allows connection from the internet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

Creating Custom EC2 Policy for User Ibrahim



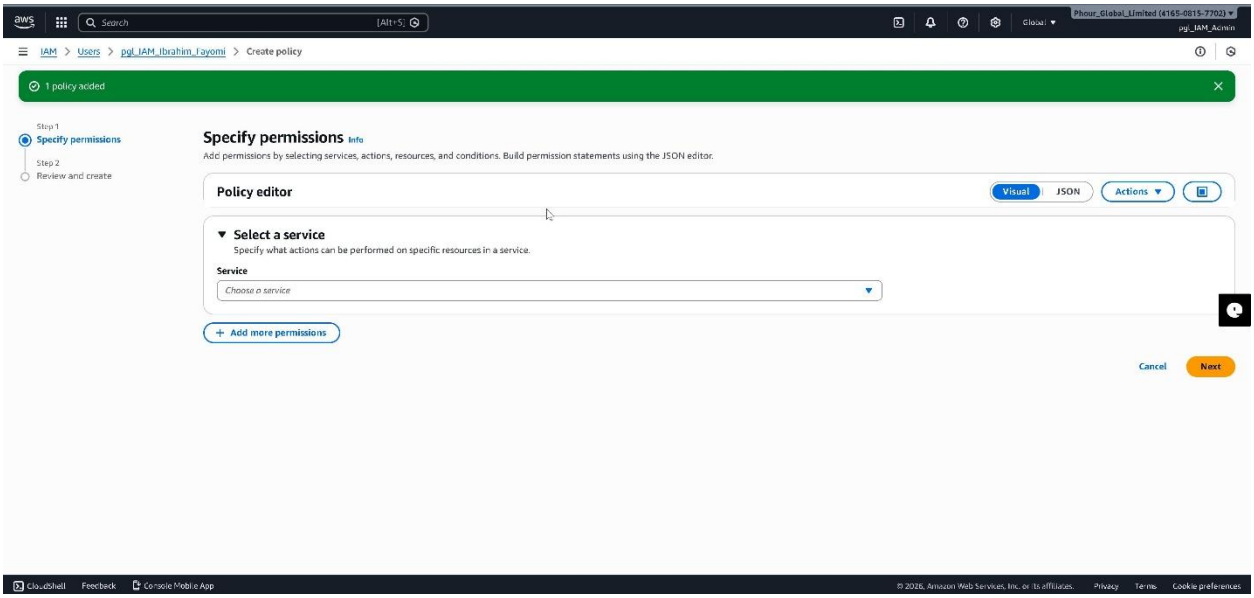
The screenshot shows starting the process to create custom permissions for user Ibrahim. Rather than using AWS managed policies, a custom policy will be created to grant specific EC2 permissions. This demonstrates fine-grained access control allowing only necessary actions on particular resources.

Examining Predefined EC2 Permissions



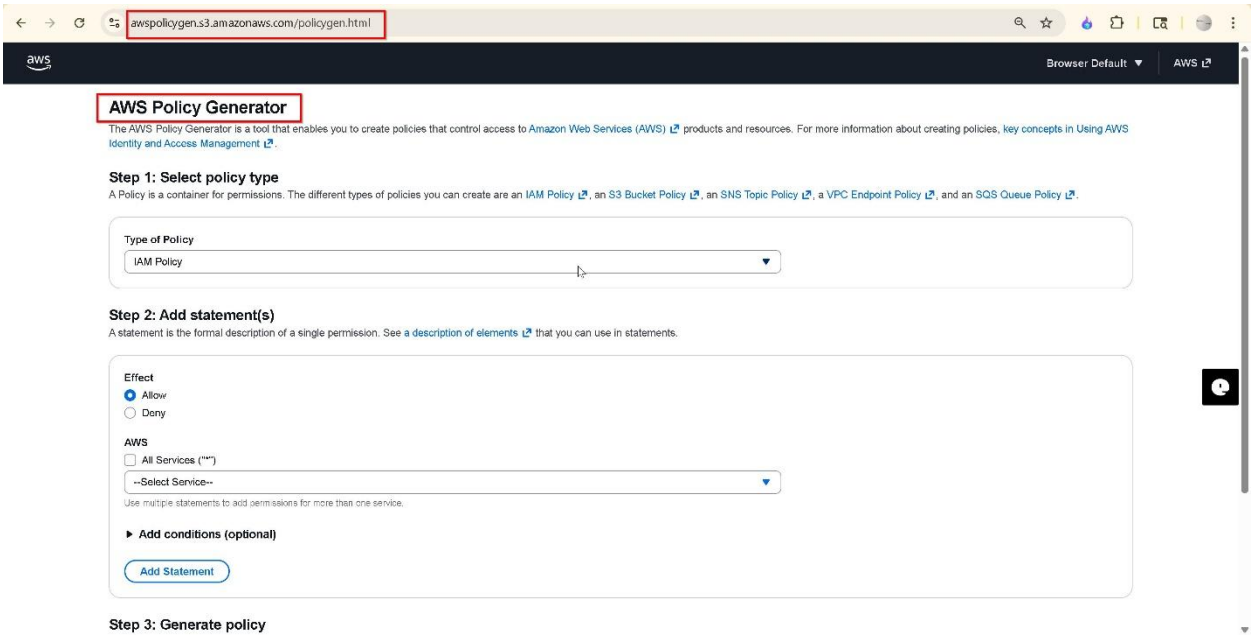
The interface displays AWS managed policies for EC2 showing the level of access each policy provides. These predefined policies cover common use cases but may grant more permissions than needed. Examining managed policies helps understand available options before creating custom policies for specific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

Creating Policy from Scratch



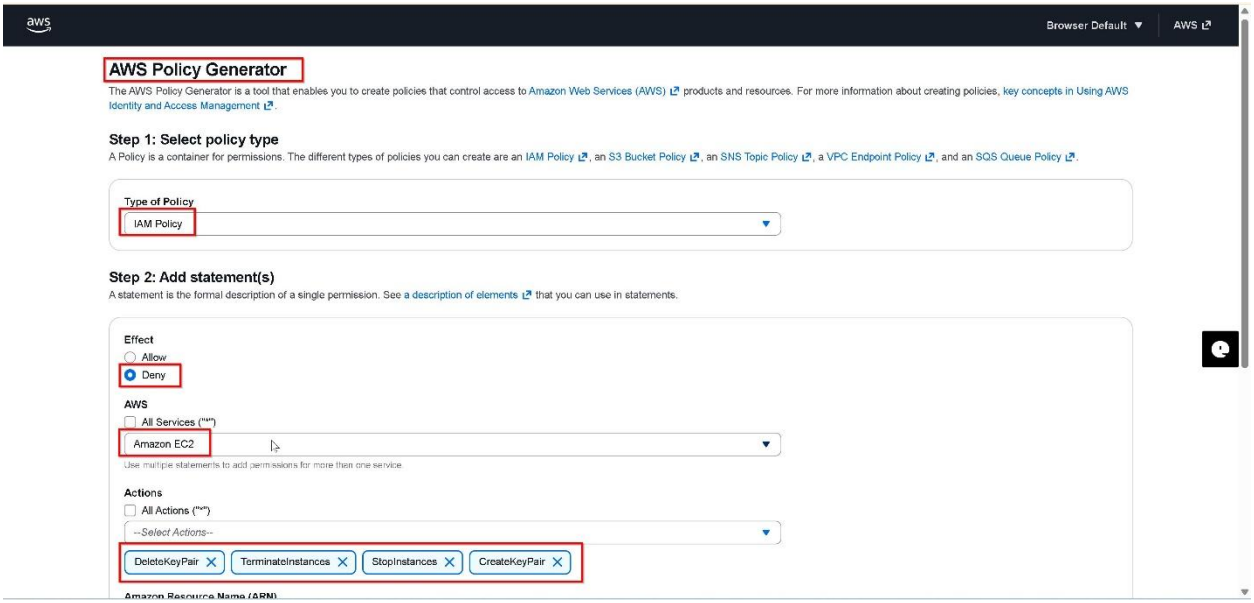
The policy creation screen offers options to build policies using visual editor or JSON. The JSON option is selected to use the AWS Policy Generator tool. Custom policies allow precise control over which actions are allowed on which resources. Policy statements define service, actions, resources, and conditions.

Accessing AWS Policy Generator



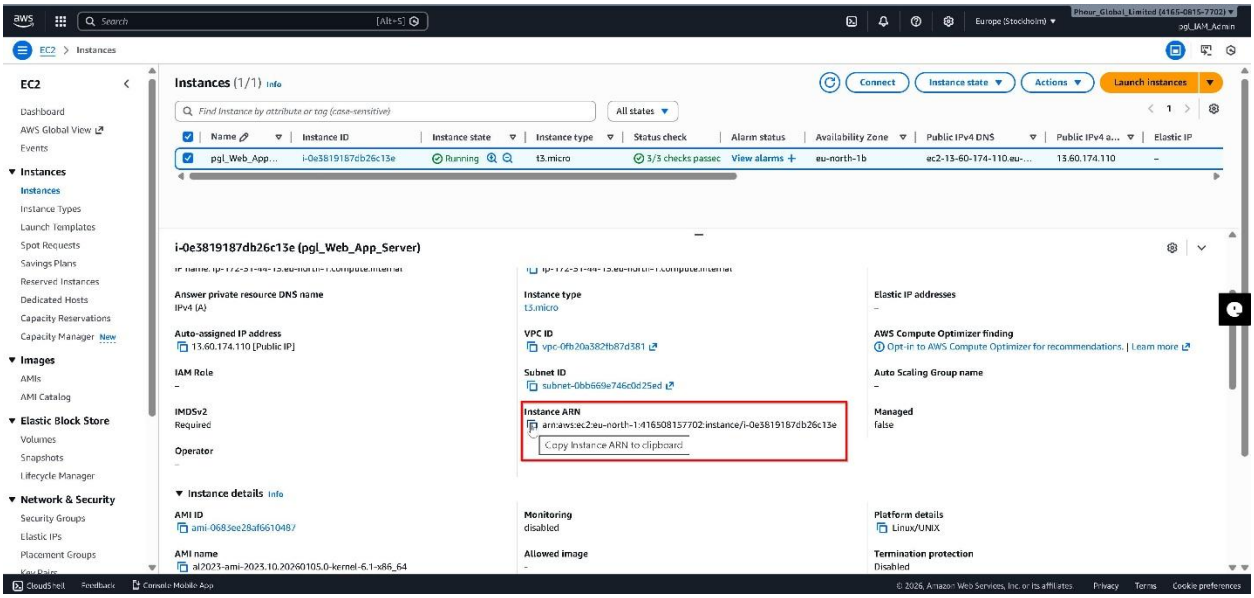
The AWS Policy Generator tool provides a structured interface for creating policy JSON. The tool guides through selecting service, actions, and resources to build policy statements. Generated JSON can be copied into IAM console for policy creation. The generator prevents syntax errors and ensures valid policy format.

Configuring Policy in AWS Policy Generator



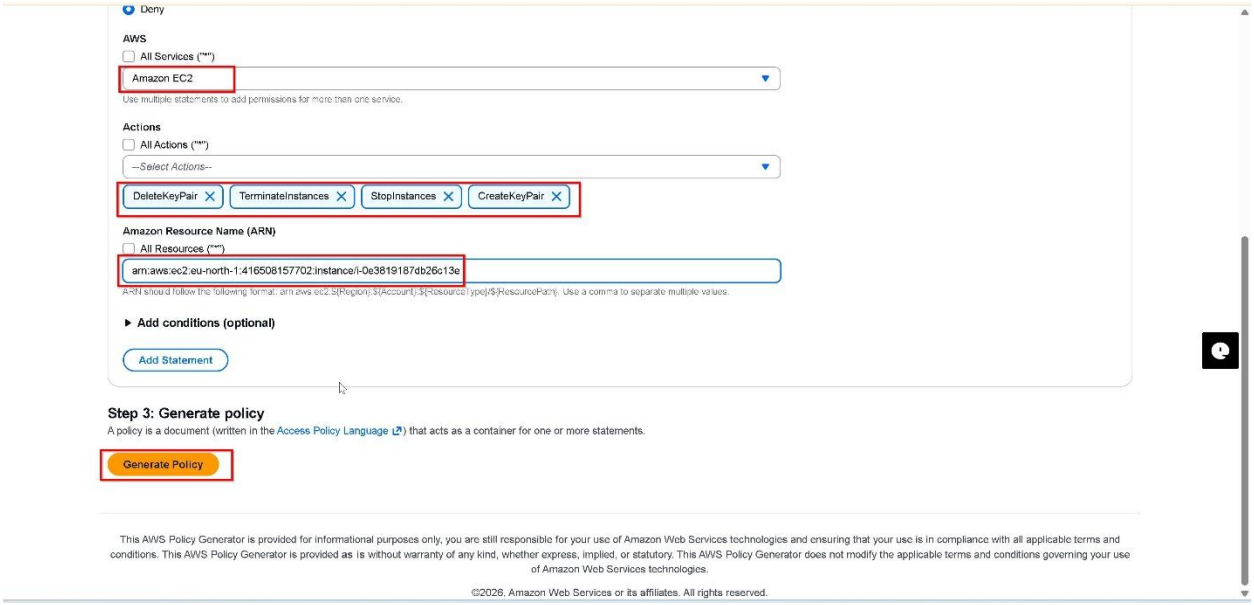
The policy generator form shows selecting Amazon EC2 as the service. The effect field determines whether actions are allowed or denied. Actions dropdown lists all available EC2 operations. Multiple actions can be selected for a single policy statement. The configuration builds the policy logic that will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

Getting ARN from EC2 Instance



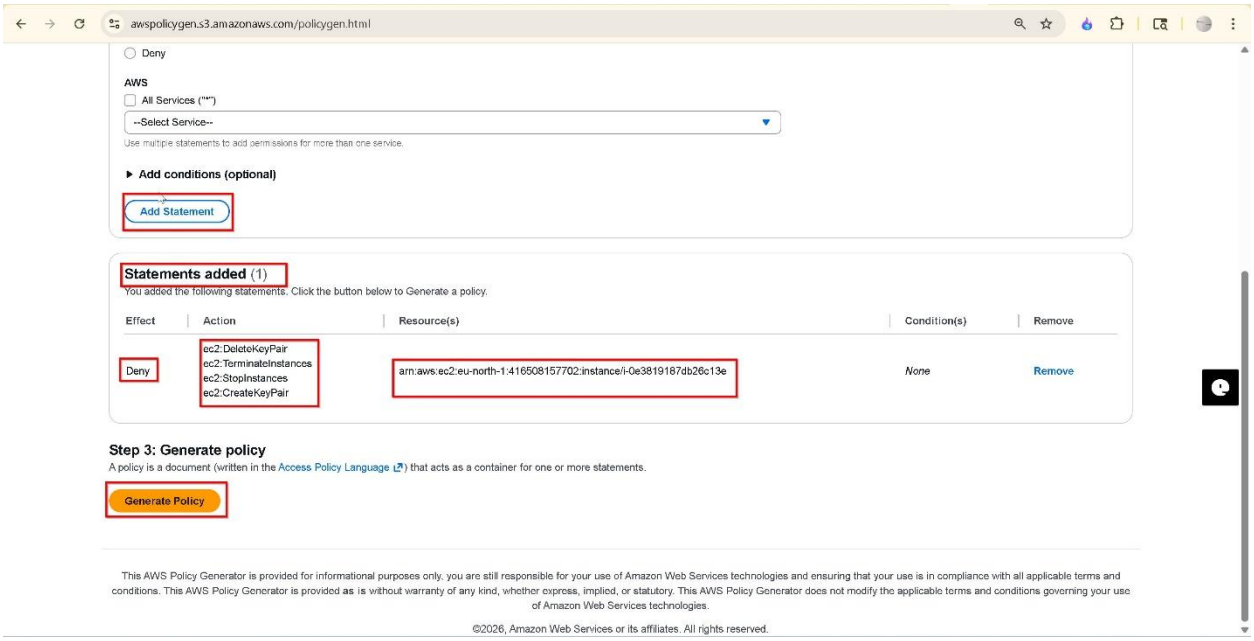
The instance details display the Amazon Resource Name (ARN) which uniquely identifies the EC2 instance. The ARN format includes account ID, region, and instance ID. Copying the instance ARN allows creating policies that apply only to this specific instance. Resource-specific policies implement least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

Completing Policy Configuration with ARN



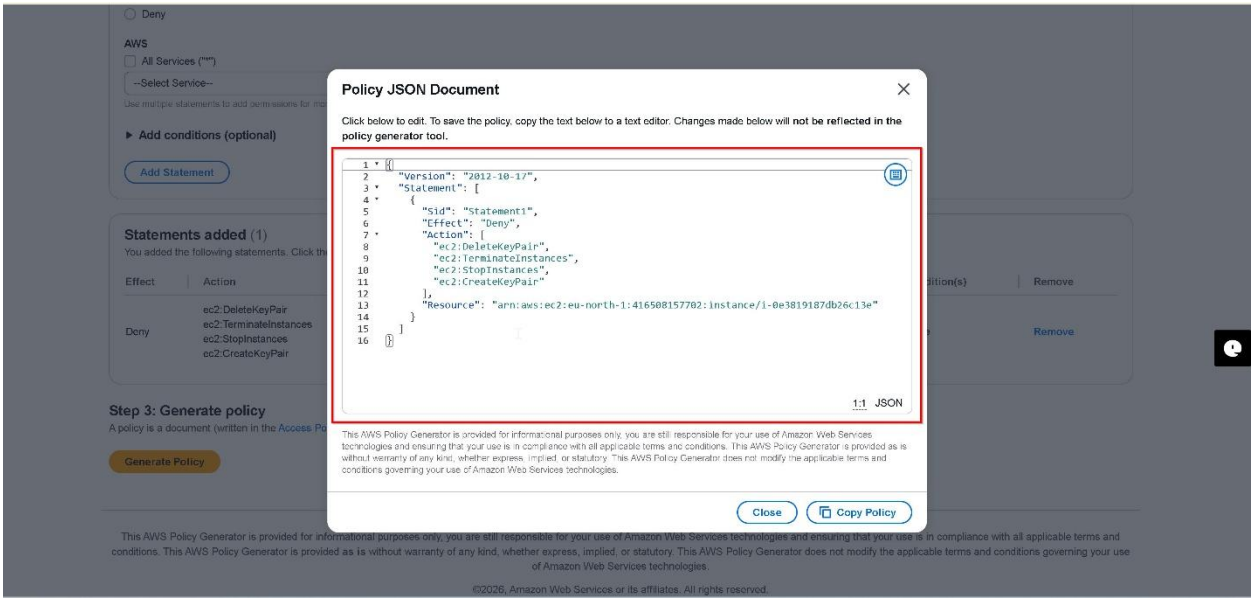
The ARN field in the policy generator is populated with the instance ARN. This restricts the policy to apply only to the specified instance. Actions like Describe Instances, Stop Instances, and Terminate Instances are selected to give Ibrahim specific control over this instance. The policy will not allow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

Statement Added to Policy



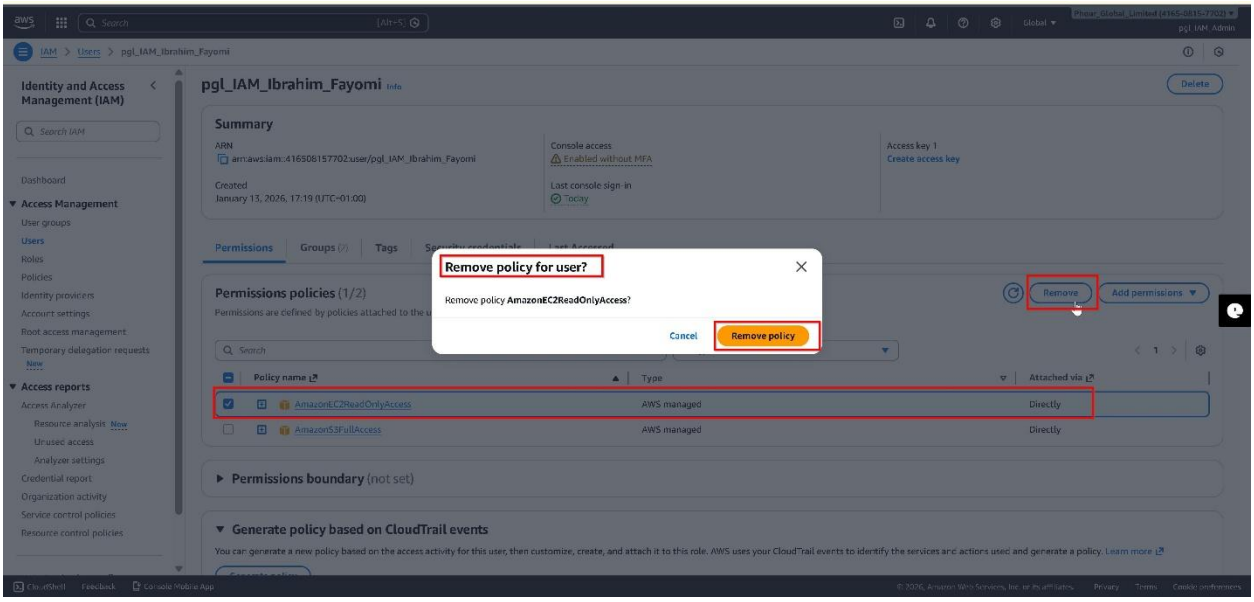
The policy generator displays the completed statement before generating JSON. The statement shows the configured service, actions, resource ARN, and effect. Multiple statements can be added to create comprehensive policies. Each statement addresses a specific permission requirement.

Generated JSON Policy



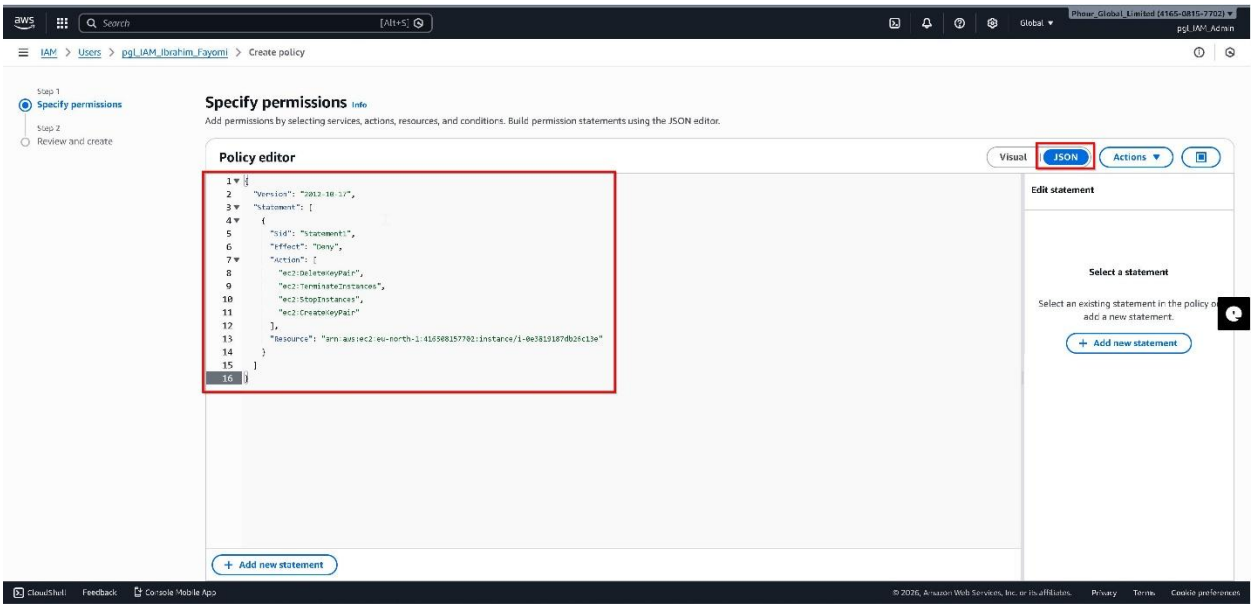
The policy generator produces properly formatted JSON containing the policy statement. The JSON includes version, statement array, effect, actions list, and resource ARN. This JSON can be copied directly into IAM policy creation. The generated format follows IAM policy syntax requirements ensuring the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

Deleting Predefined Policy from User



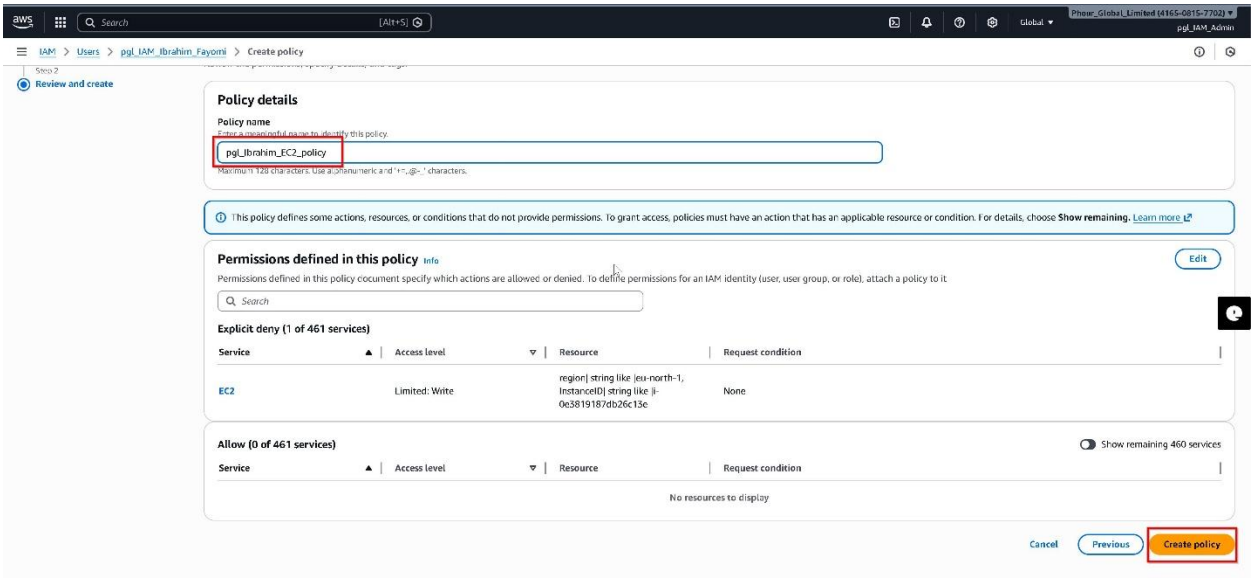
The screenshot shows removing any existing EC2 managed policies from Ibrahim before adding the custom policy. This ensures the user has only the intended permissions defined in the custom policy. Removing excess permissions maintains least privilege access.

Pasting Generated Policy JSON



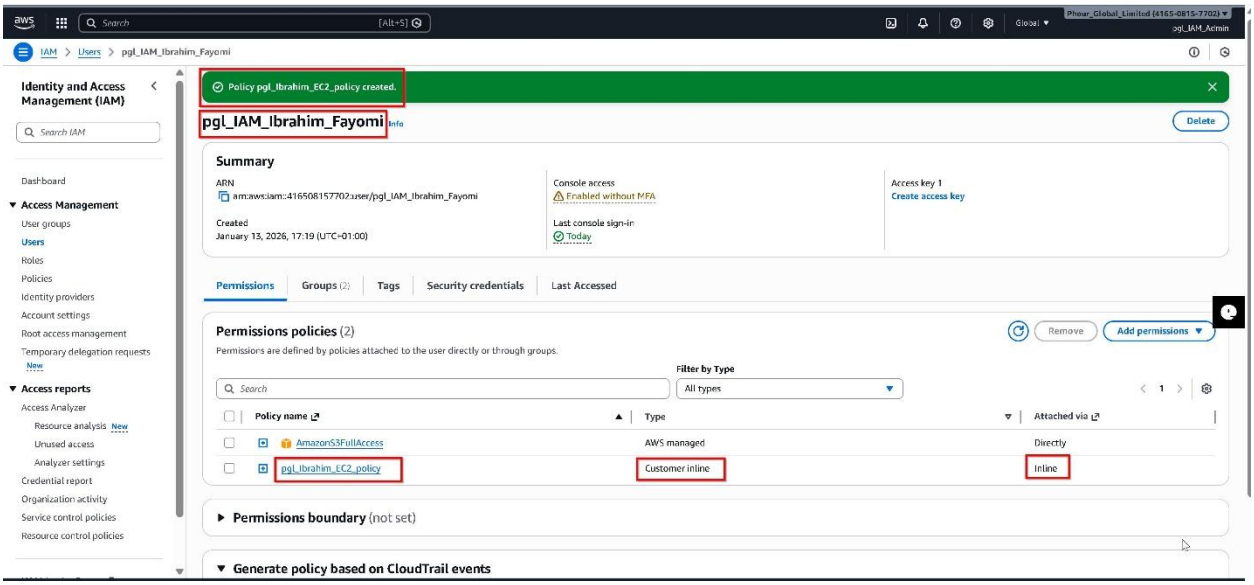
The JSON editor in IAM shows the generated policy pasted from the AWS Policy Generator. The JSON is validated to ensure correct syntax. The policy defines specific EC2 actions that Ibrahim can perform on the designated instance. Any syntax errors are flagged before policy creation proceeds.

Naming and Creating Custom Policy



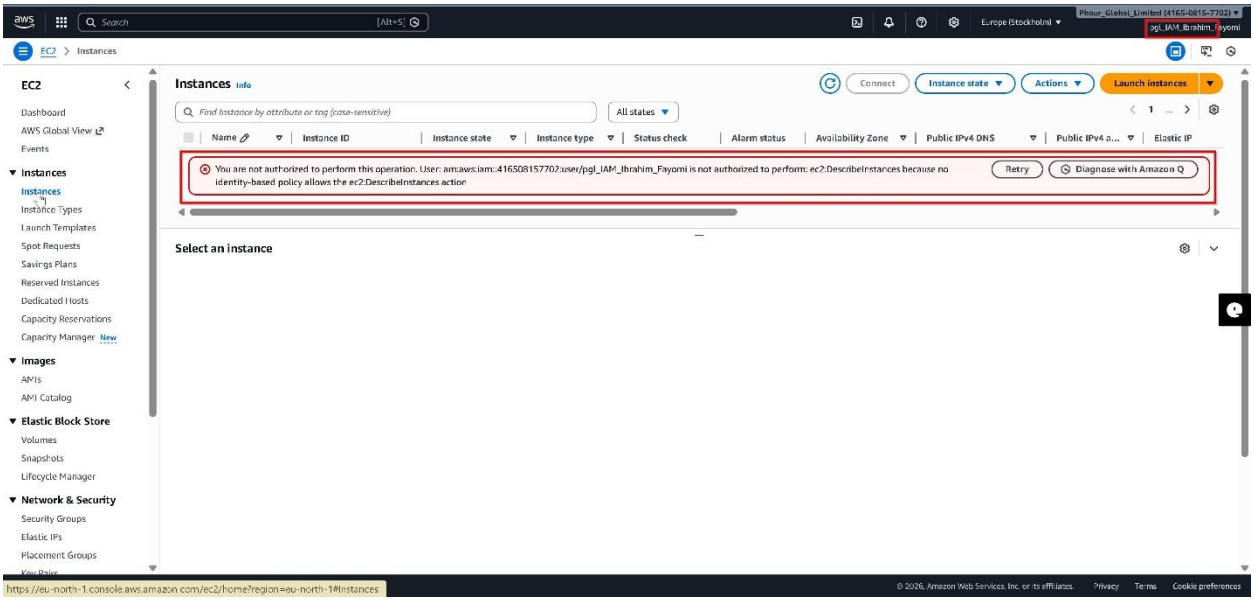
The policy name and description fields are populated with clear identifiers. The policy name should indicate its purpose such as Ibrahim-EC2-InstanceControl. Description explains what permissions the policy grants. Clear naming and documentation help maintain policies as requirements evolve. The create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

Policy Created and Added to User



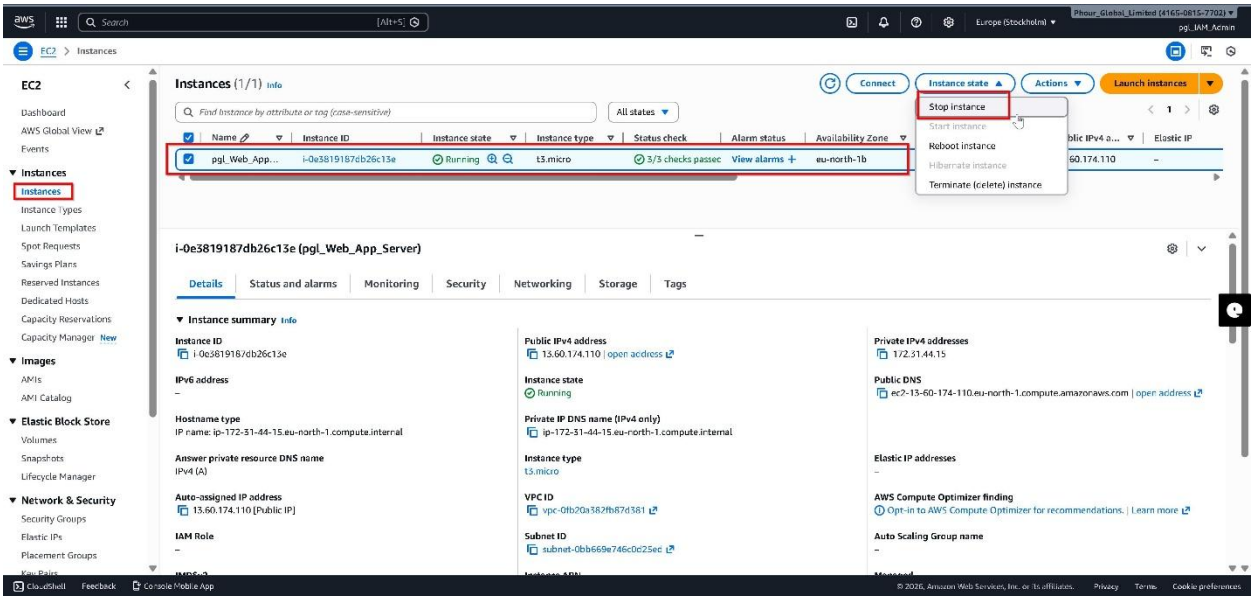
The user permissions list now shows the newly created custom policy attached to Ibrahim. The policy is active and will be enforced immediately. Ibrahim can now perform the allowed EC2 actions on the specified instance but has no other EC2 permissions. The custom policy provides precisely the required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

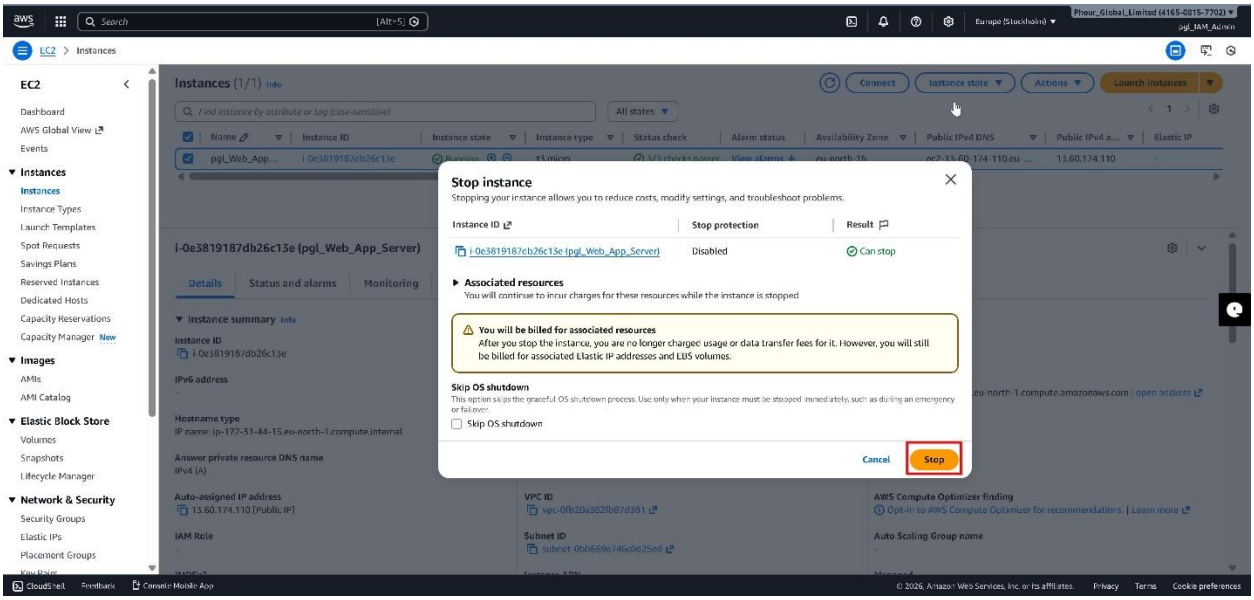
Signed in as User Ibrahim with Policy Enforced



The 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.

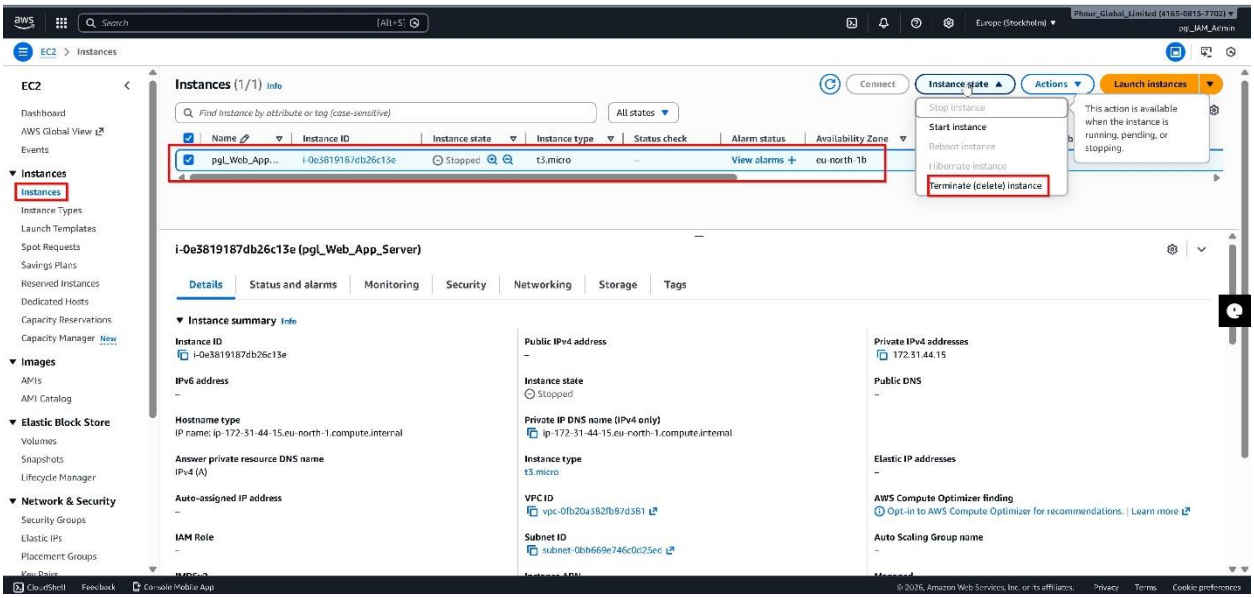
Stopping EC2 Instance

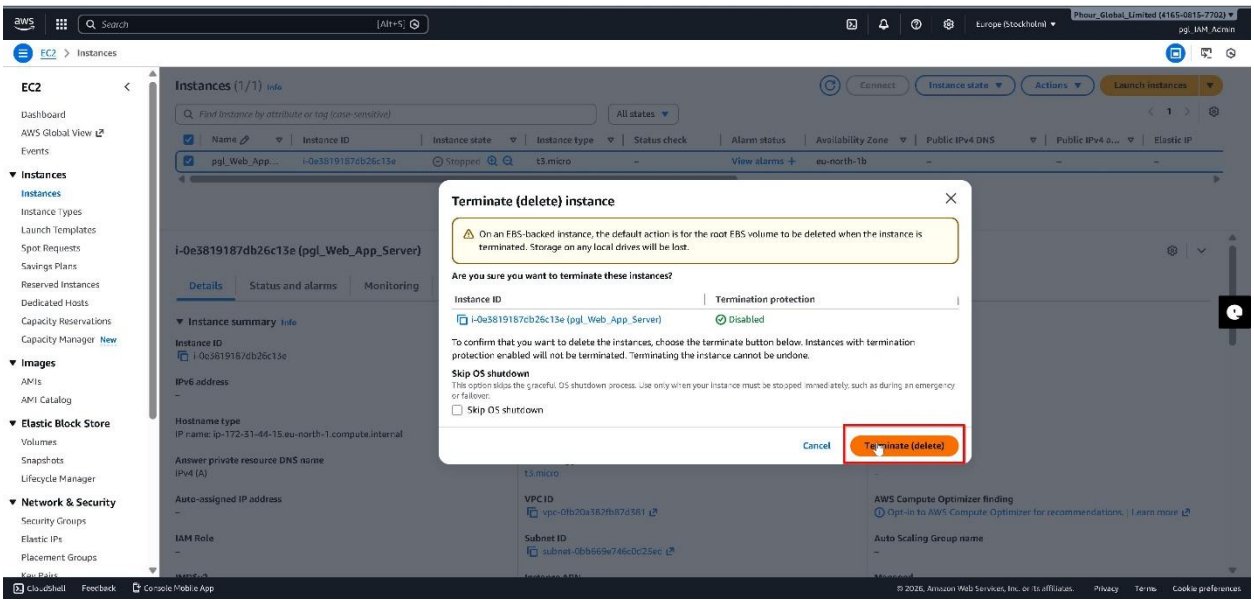




The screenshots show Ibrahim accessing the EC2 console and stopping the instance. The custom policy allows this action on the specified instance. The stop action is selected from the instance state menu. Stopping releases compute resources while preserving instance configuration and attached storage.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

Terminating EC2 Instance



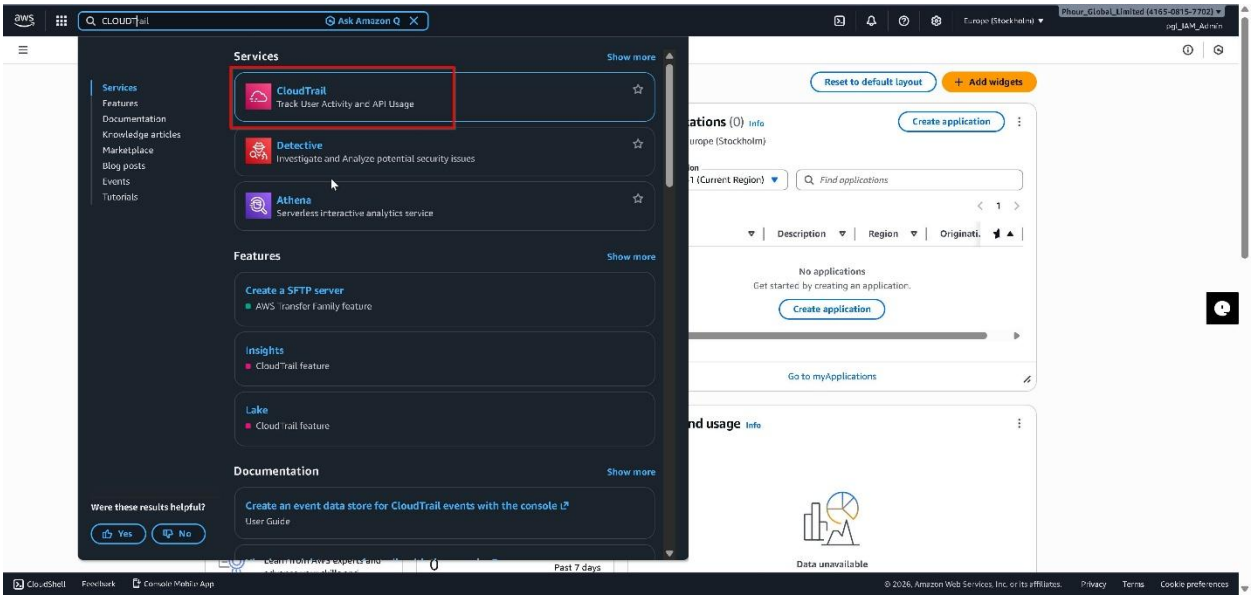


The instance termination process is initiated. Terminating permanently deletes the instance and its associated resources. The custom policy allows Ibrahim to perform this destructive action on the designated instance. A confirmation prompt prevents accidental termination. Once terminated, the instance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

CLOUDTRAIL SERVICES

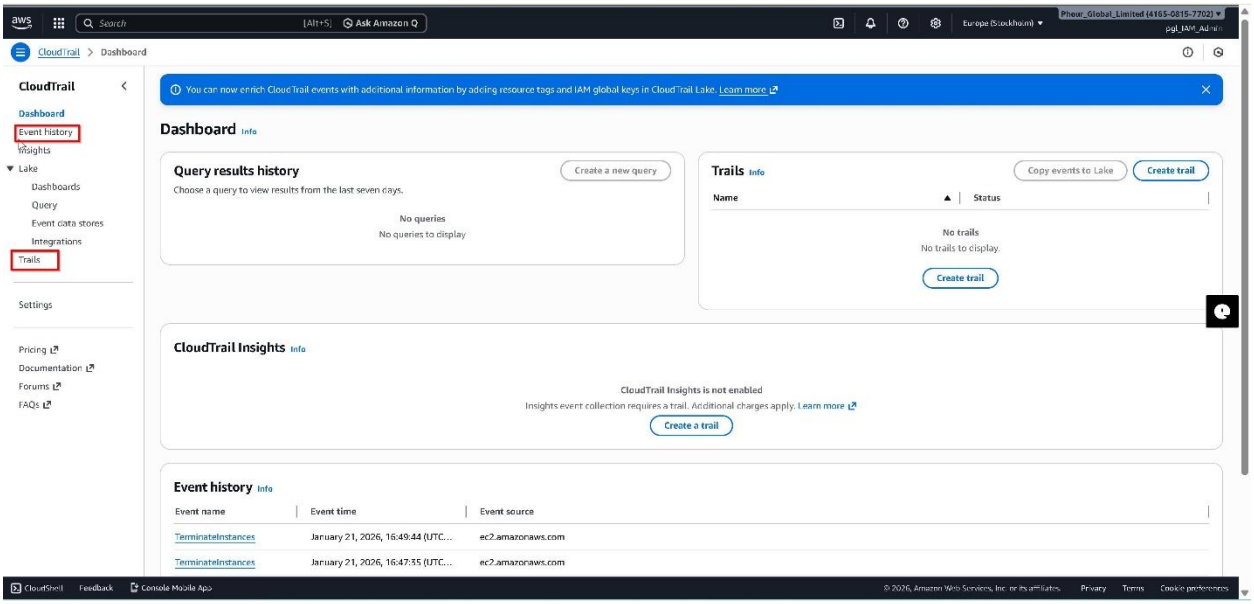
CloudTrail Configuration

Accessing CloudTrail Service



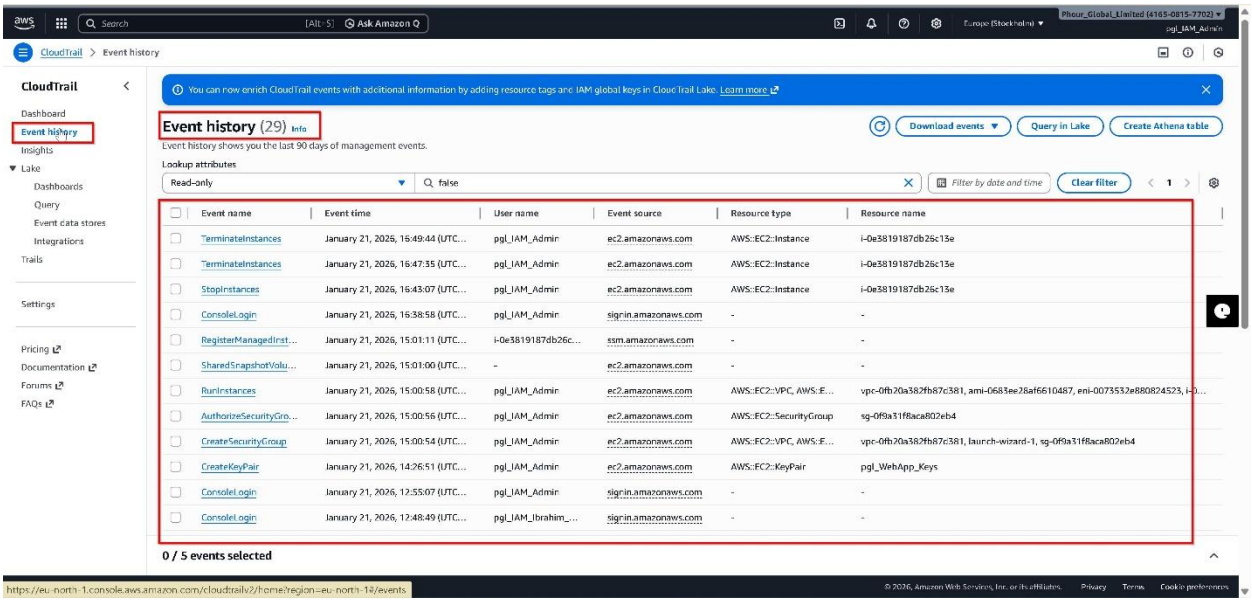
The console shows navigation to CloudTrail service. CloudTrail records AWS API calls and related events for the account. The service provides audit trails for compliance, security analysis, and troubleshooting. CloudTrail logs capture who made requests, what services were accessed, when actions occurred,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

Event History and Trails



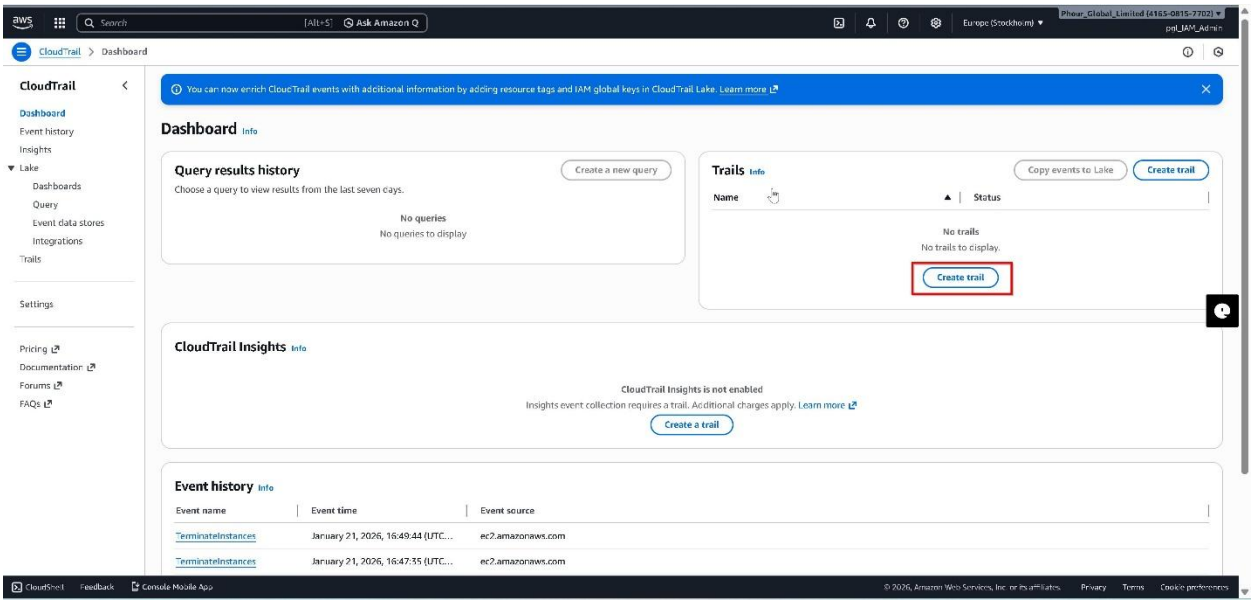
The CloudTrail dashboard displays event history and configured trails. Event history shows the last 90 days of management events at no additional cost. Trails provide ongoing logging with extended retention by storing logs in S3. The interface shows options to view events and create trails.

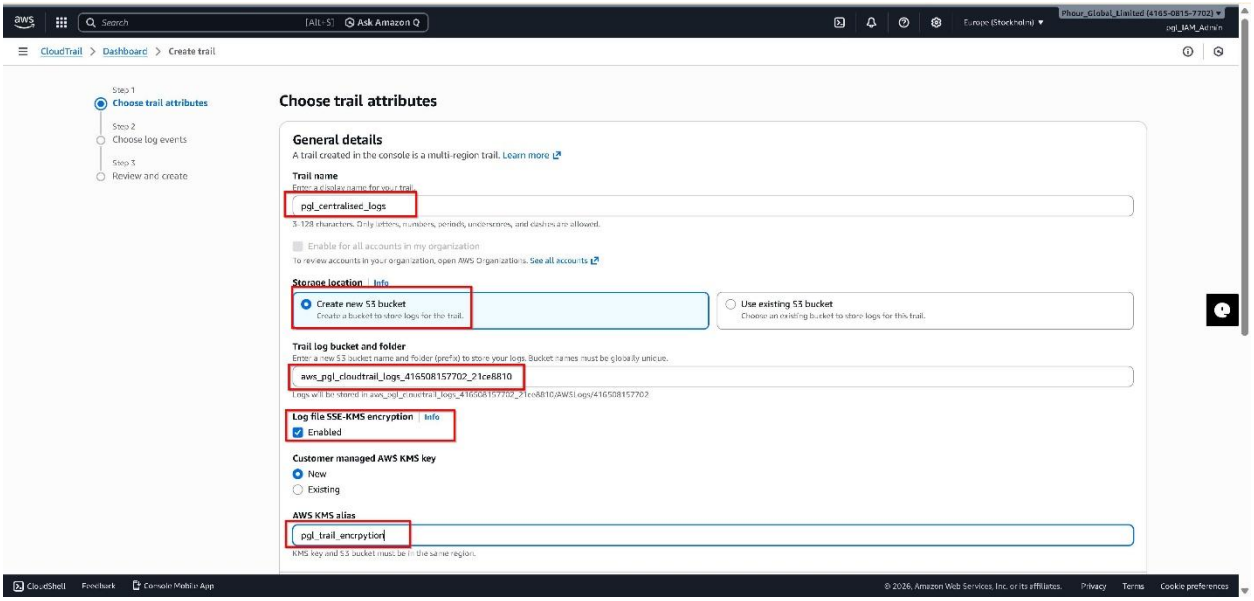
Event History Showing All Activities

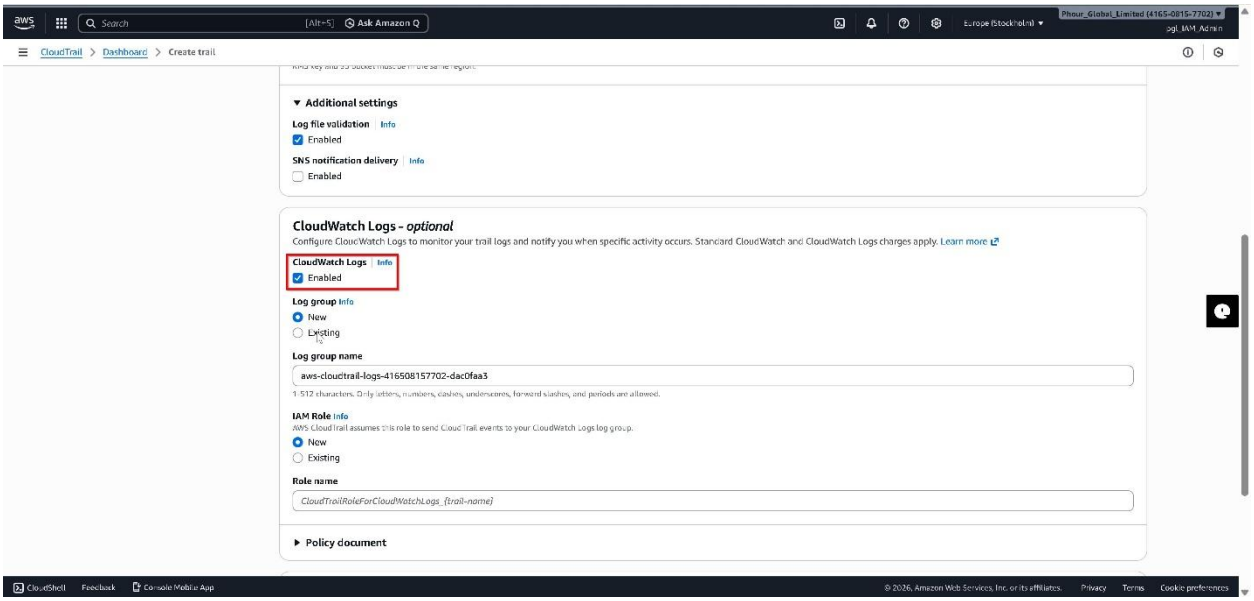


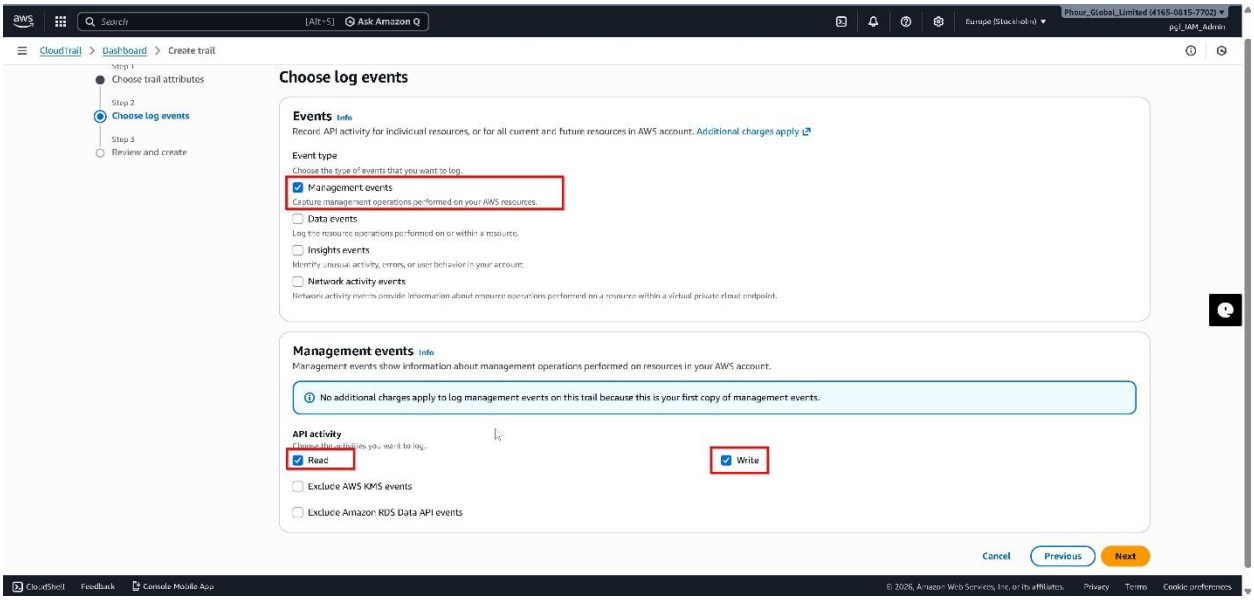
The event history view displays a chronological list of API calls. Each event shows event name, user identity, event time, and resource name. Events can be filtered by various attributes such as user name, resource type, or event name. The detail view for each event provides complete request and response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

Creating CloudTrail Trail





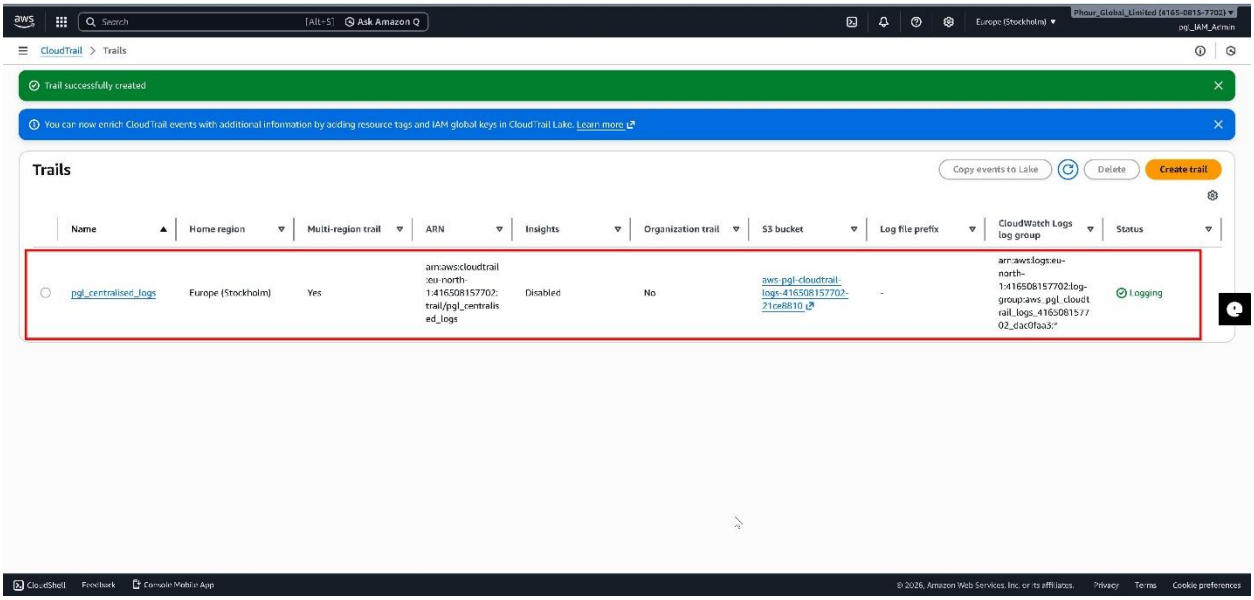




The trail creation wizard guides through configuration options. Trail name is specified along with log storage settings. The trail can apply to all regions or a single region. Management events record control plane operations like creating resources. Data events record resource operations like S3 object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

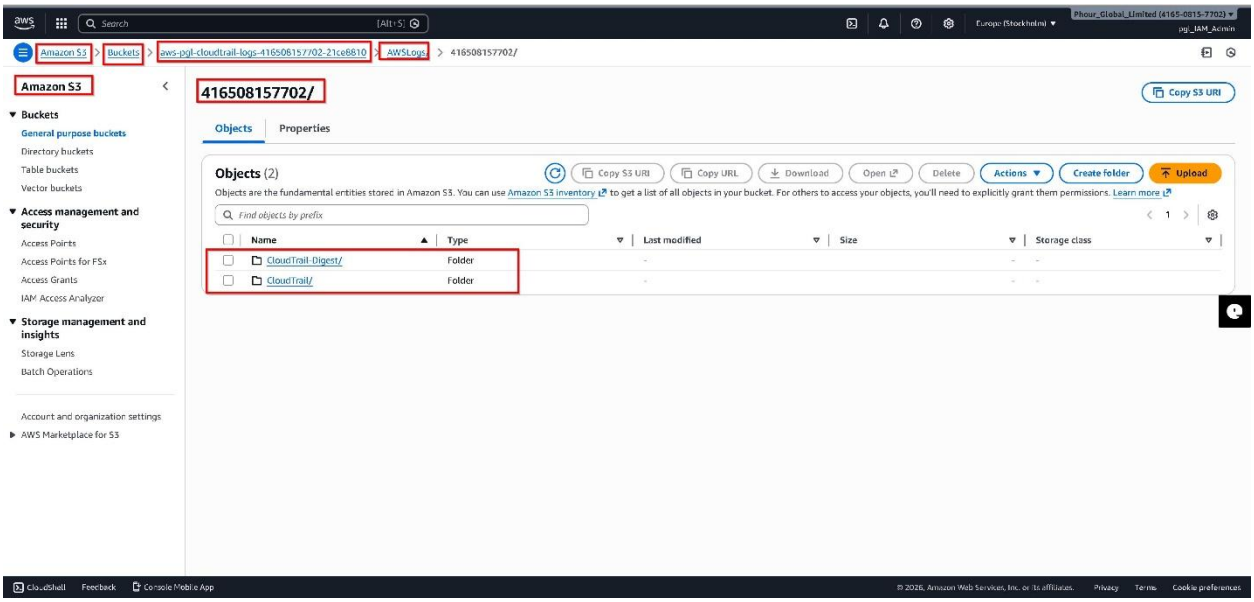
Creating a trail enables continuous monitoring of account activity beyond the 90-day event history window. Logs stored in S3 can be retained indefinitely and analyzed using various tools. Trail configuration determines what events are logged and where they are stored.

Trail Created Successfully



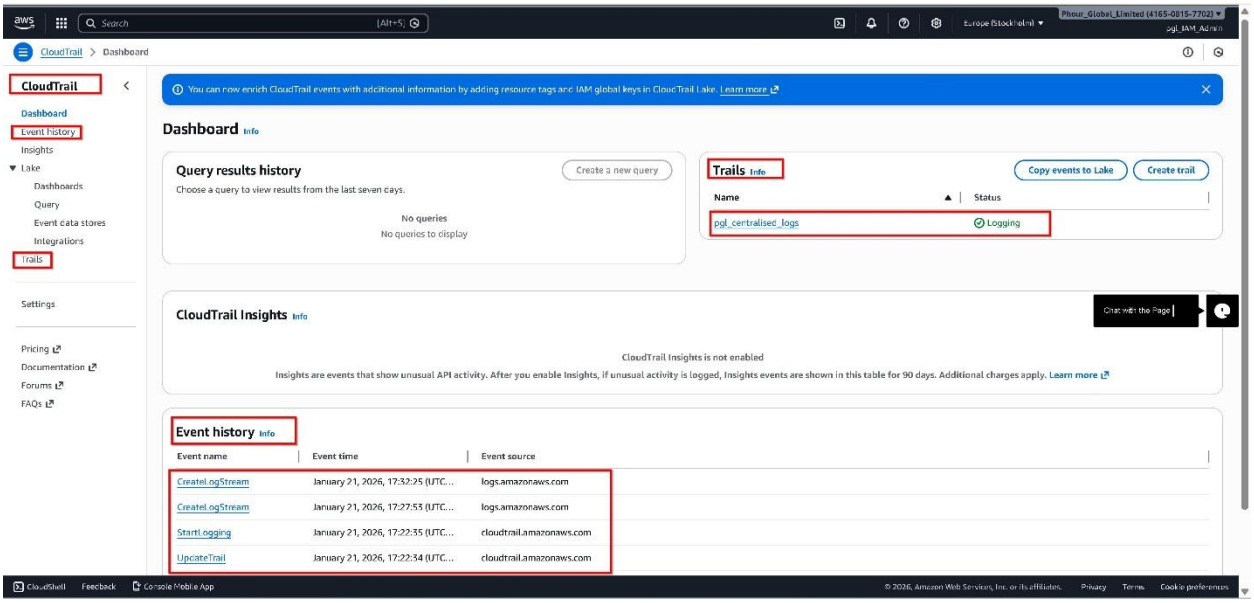
The trails list displays the newly created trail in active status. The trail is now logging events to the specified S3 bucket. CloudTrail typically delivers log files within 15 minutes of API calls. The trail can be modified or deleted as needed. Multiple trails can be configured to separate different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

CloudTrail S3 Bucket



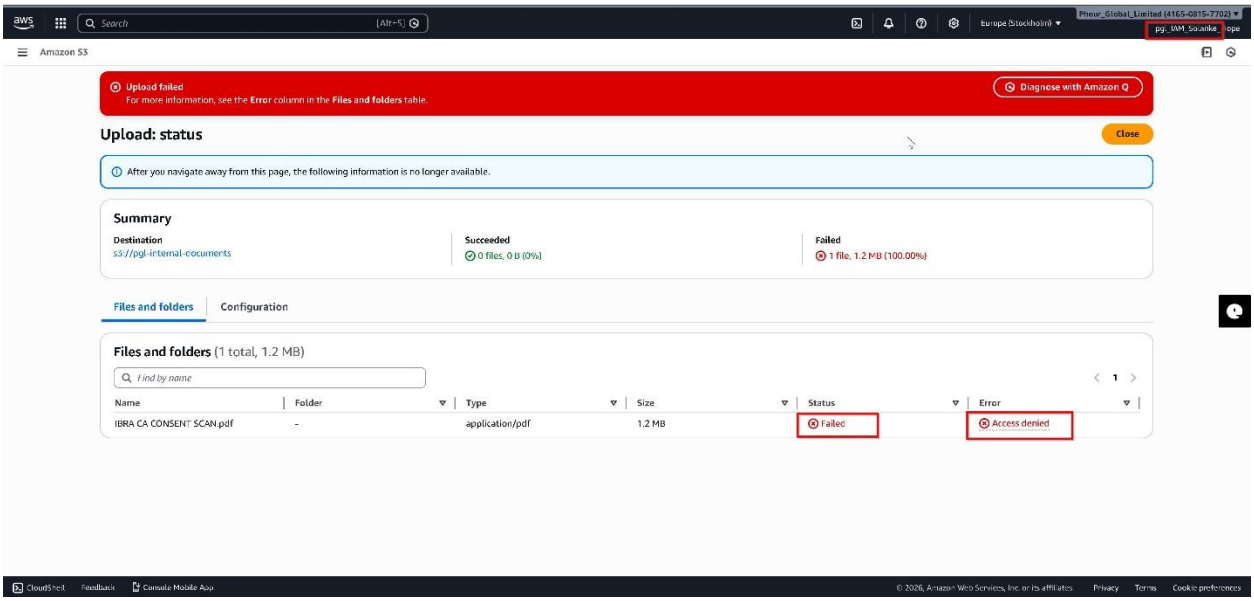
The S3 bucket list shows the CloudTrail bucket created automatically during trail setup. The bucket name follows CloudTrail naming convention including account ID and region. CloudTrail log files are organized by date and region within the bucket. Access to this bucket should be restricted to prevent log tampering or deletion.

CloudTrail Dashboard After Configuration



The 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

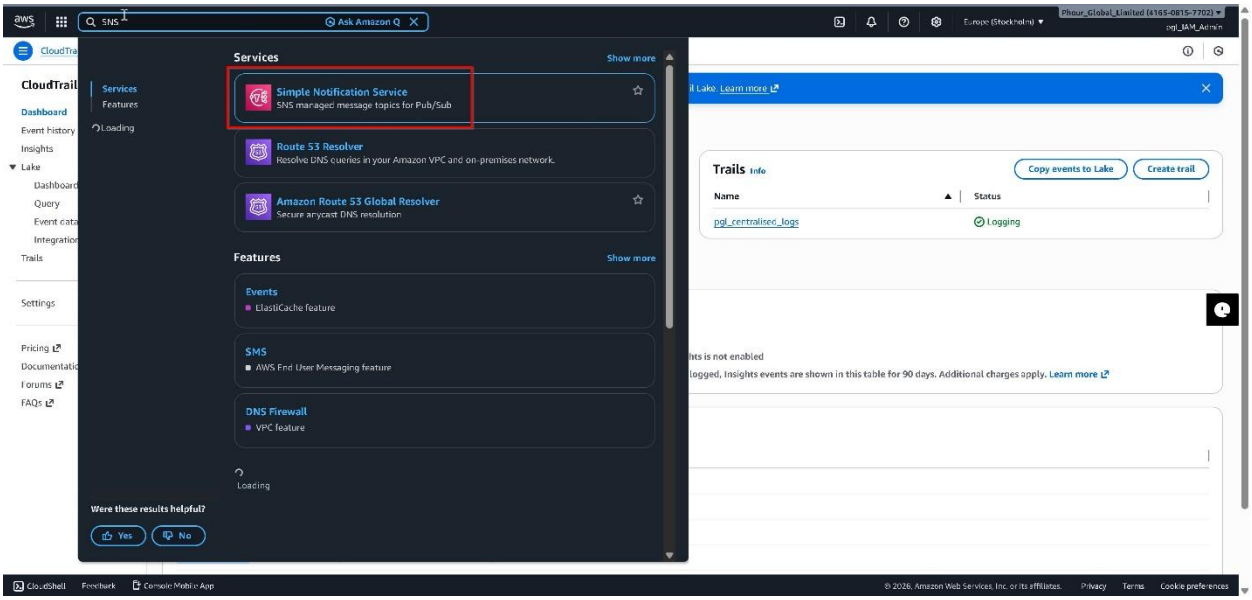


This screenshot verifies the read-only S3 user cannot upload files. The upload attempt fails with insufficient permissions error. This confirms the AmazonS3ReadOnlyAccess policy is working correctly. The error demonstrates proper access control enforcement. All denied actions are logged to CloudTrail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

CREATING NOTIFICATION SERVICE

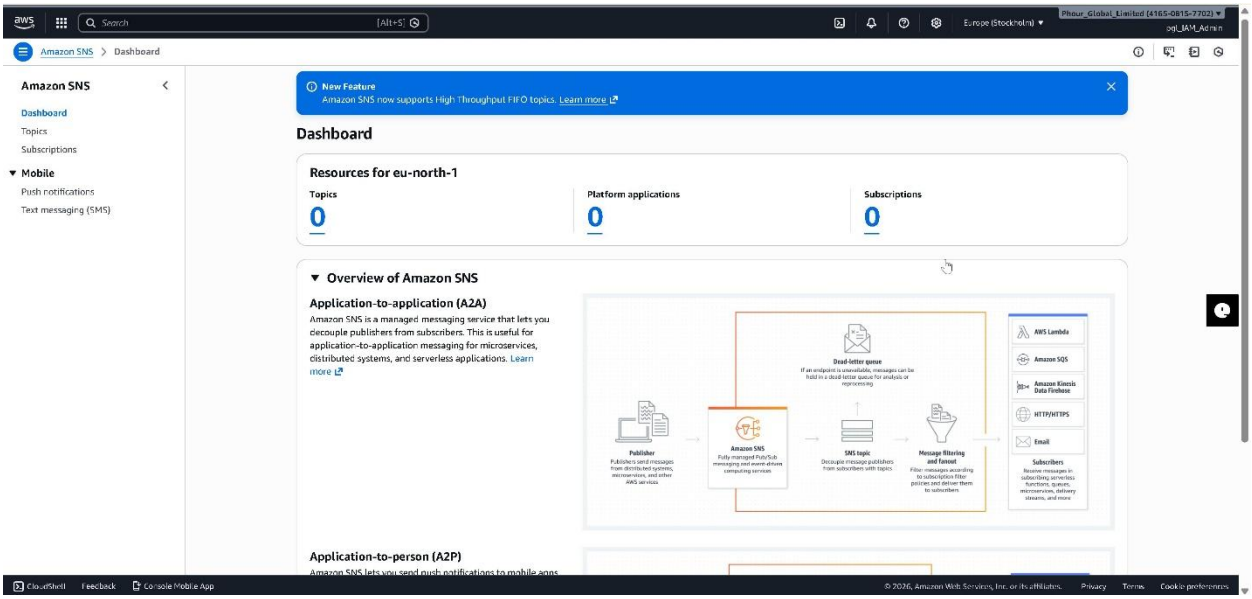
Simple Notification Service (SNS)

Accessing SNS Service



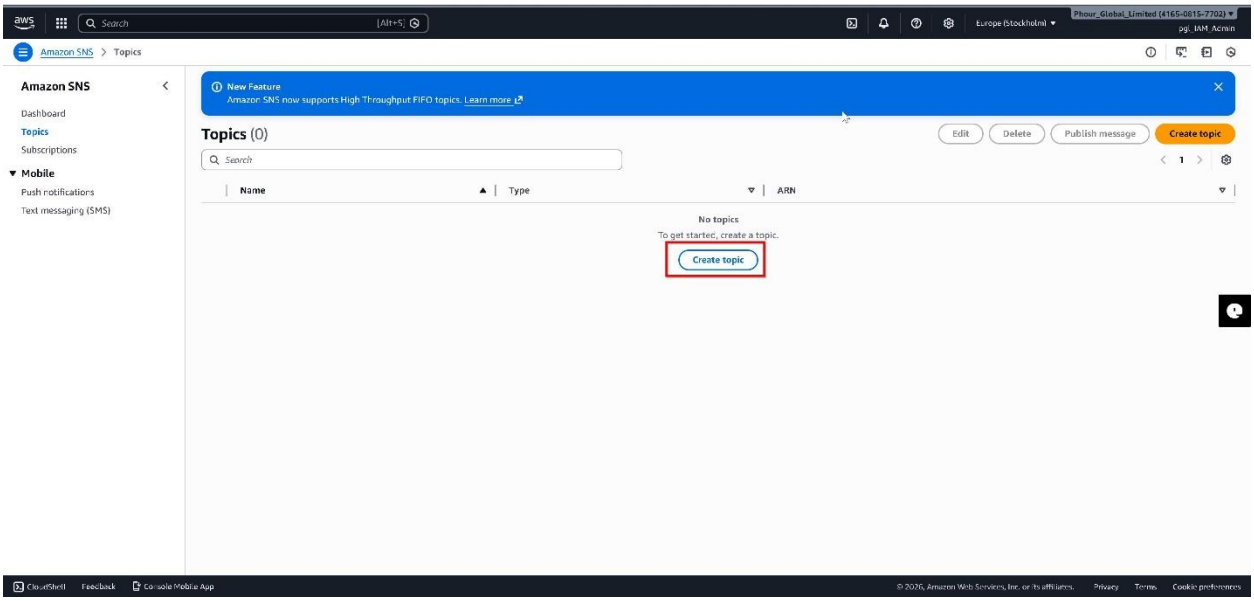
The console displays navigation to Simple Notification Service. SNS coordinates message delivery to subscribers through topics. The service supports multiple protocols including email, SMS, HTTP endpoints, and AWS Lambda. SNS enables building event-driven architectures and notification systems.

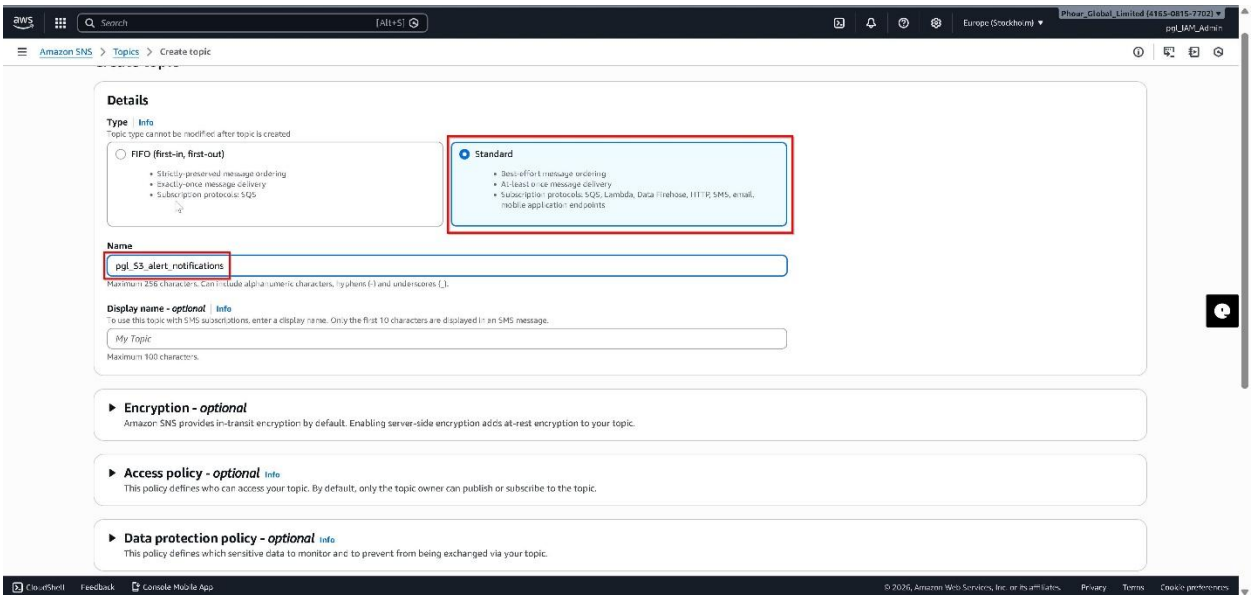
SNS Dashboard



The 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.

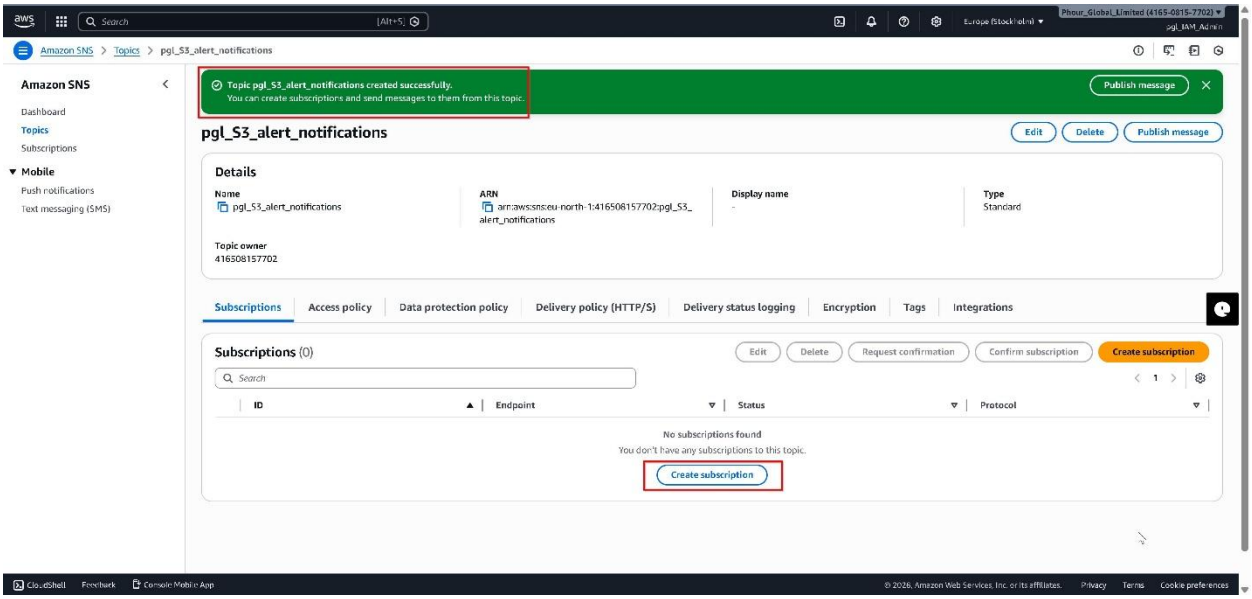
Creating SNS Topic





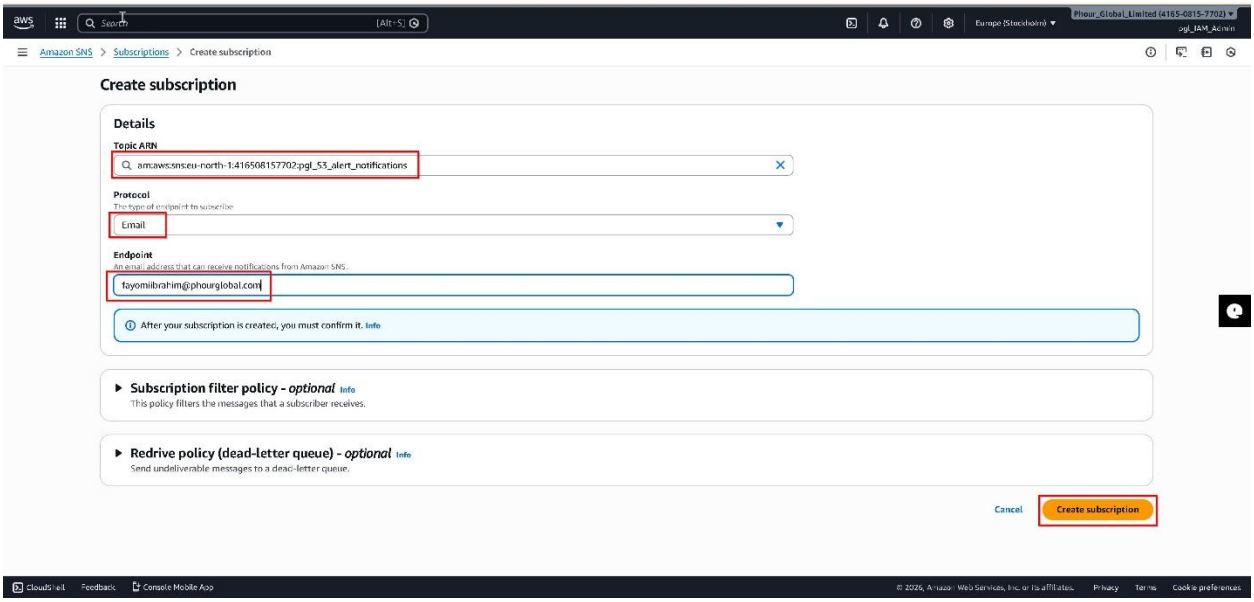
The topic creation wizard prompts for topic configuration. Standard topic type is selected which provides high throughput message delivery. The topic name clearly identifies its purpose such as Security-Alerts. Topic display name is used in certain notification protocols. Access policy determines who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

Topic Created and Creating Subscription



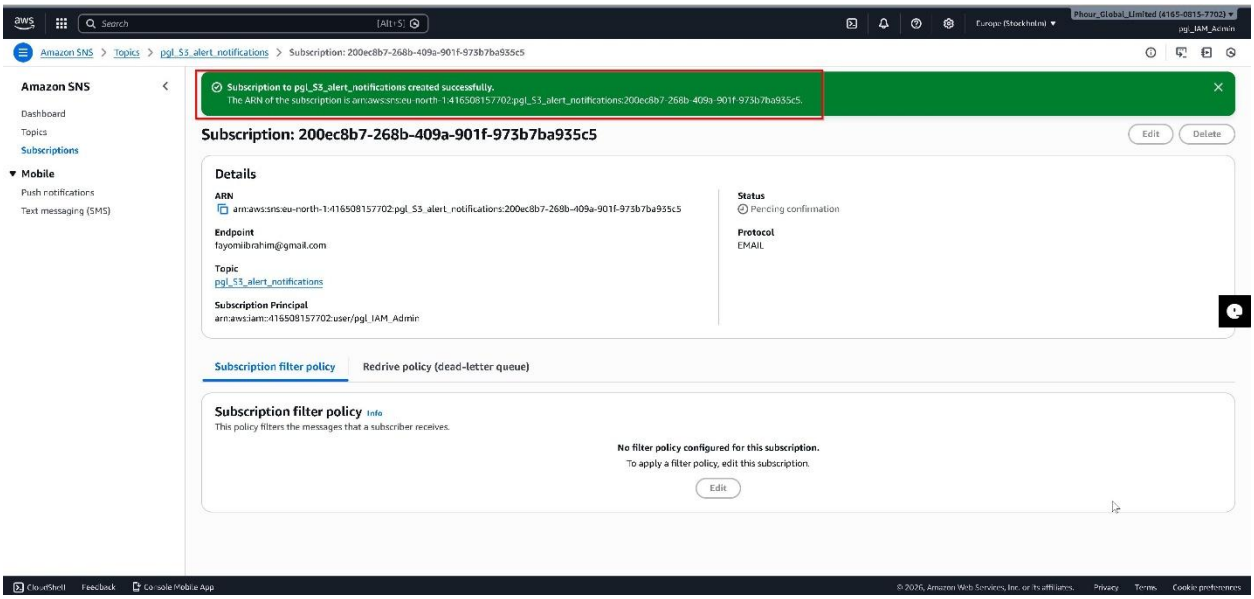
The topic details page shows the newly created topic. The create subscription button adds subscribers who will receive messages published to this topic. Subscriptions define how and where messages are delivered. Multiple subscriptions can be attached to a single topic to notify different endpoints or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

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



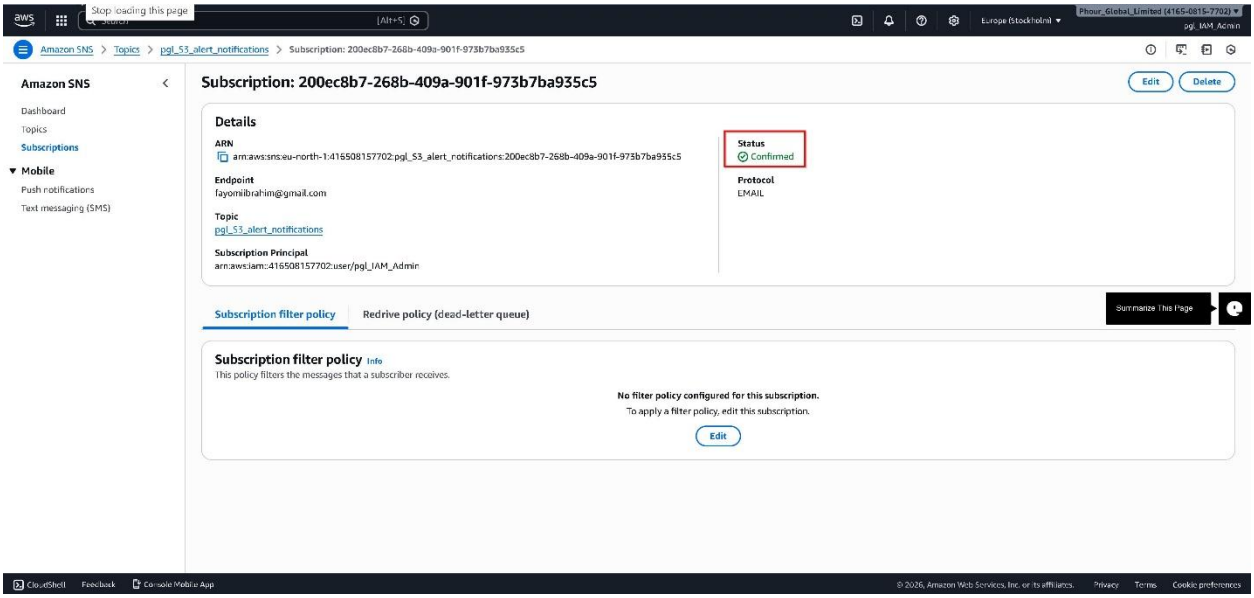
The subscriptions list displays the new email subscription in pending confirmation status. The subscription will not receive messages until confirmed. The status indicator helps track which subscriptions are active. Unconfirmed subscriptions are automatically deleted after three days if not confirmed.

Subscription Confirmed from Email

The 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.

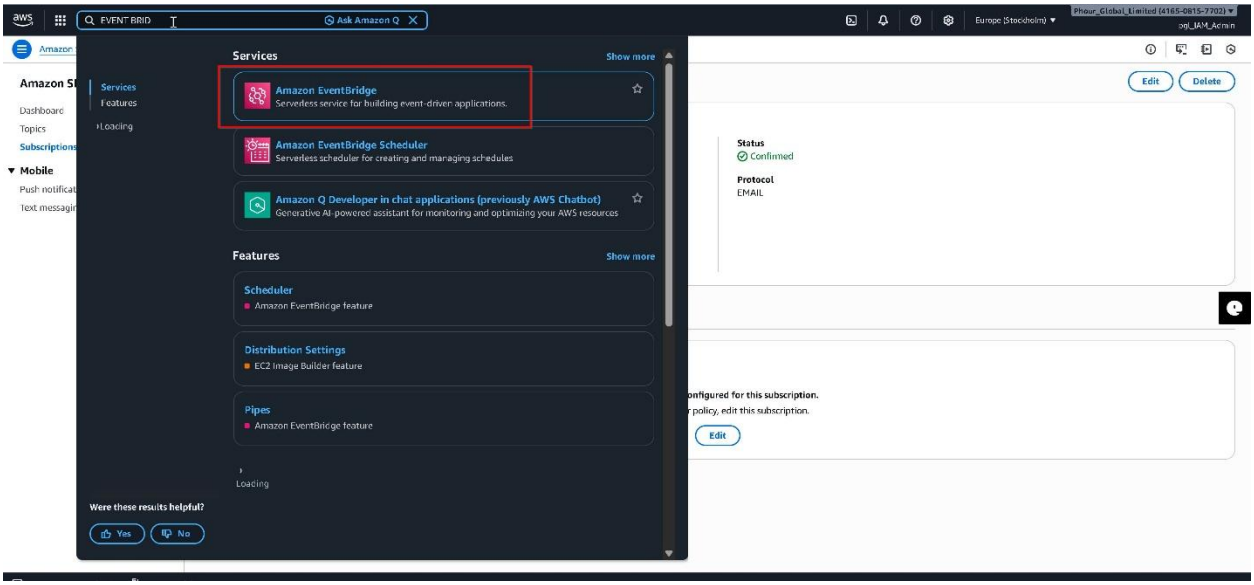
Subscription Confirmed in Console



The 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.

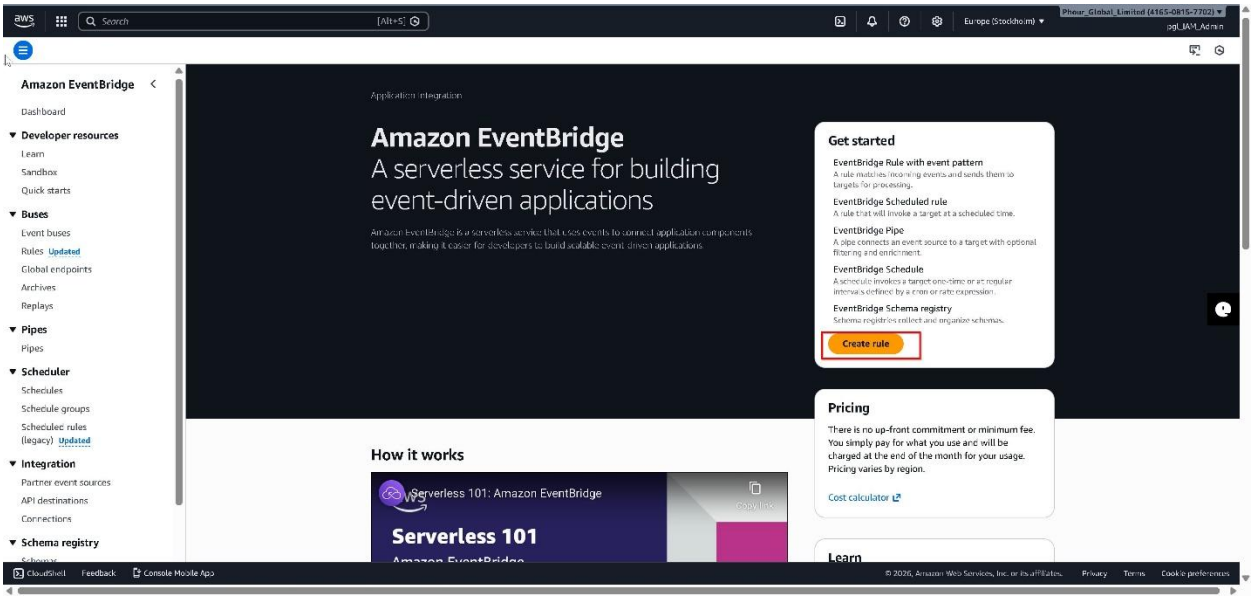
AMAZON EVENT BRIDGE

Accessing EventBridge Service



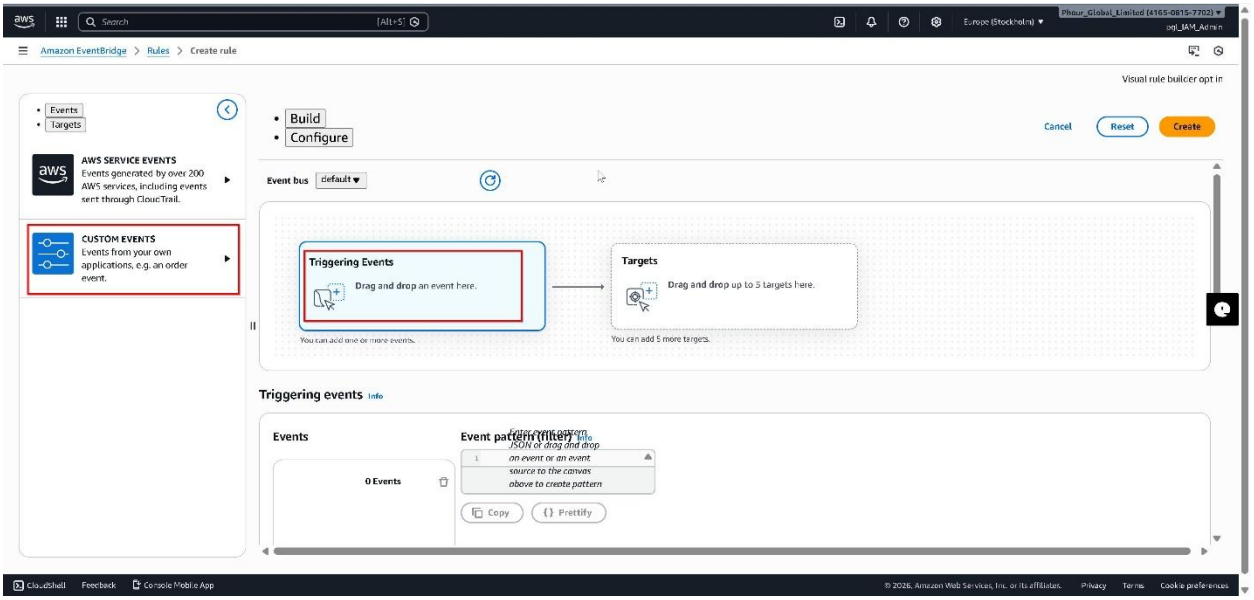
The console shows navigation to Amazon EventBridge. EventBridge is an event bus service that connects applications using events from AWS services, SaaS applications, and custom sources. Rules match incoming events and route them to targets for processing. EventBridge enables event-driven architectures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

Creating EventBridge Rule



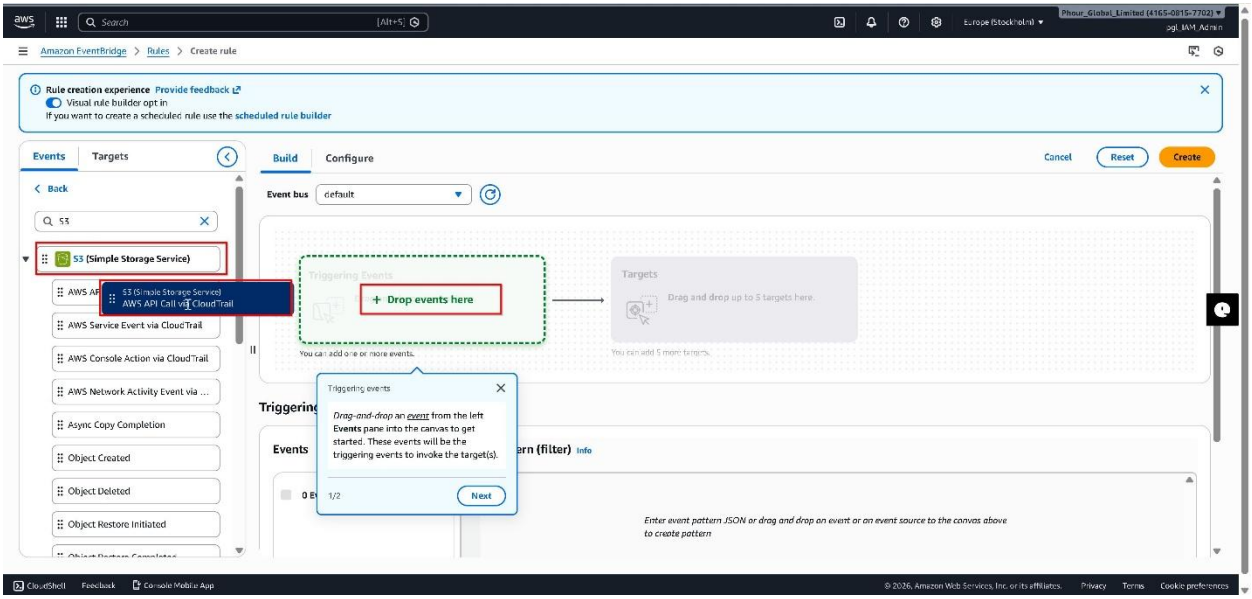
The rules page displays options to create new rules. Rules define event patterns that trigger actions when matched. The create rule button starts the rule configuration wizard. Rules can respond to specific AWS service events, schedules, or custom events.

Configuring Rule Trigger Event



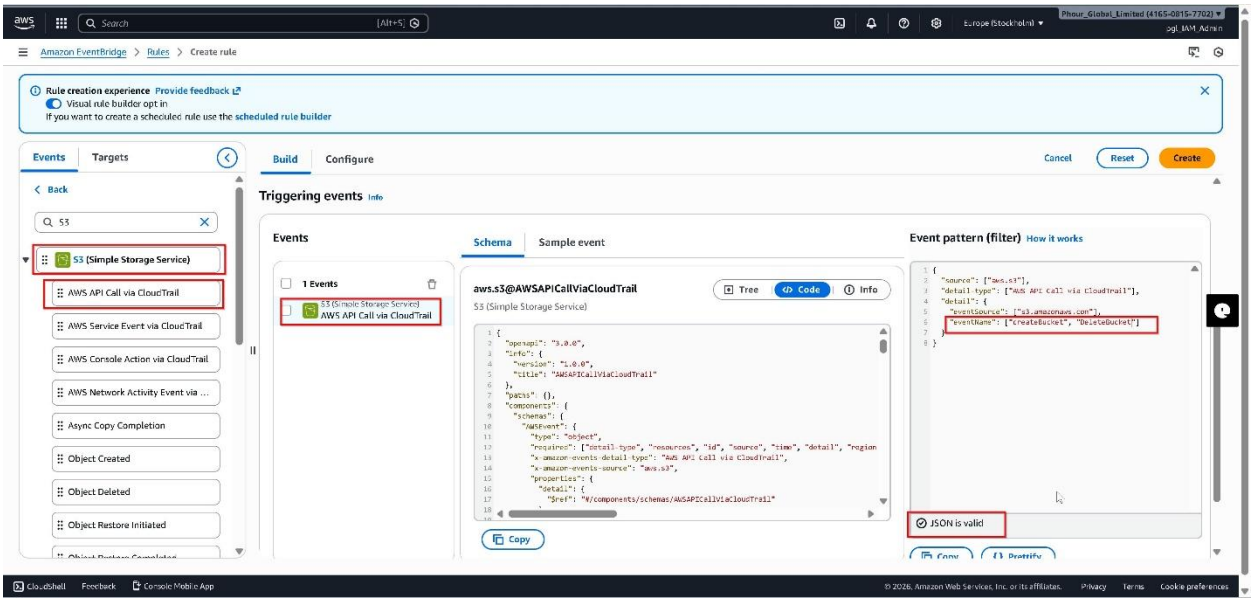
The rule configuration shows event source selection. AWS events option is selected to monitor AWS service actions. The event pattern determines which events trigger the rule. Event patterns can match specific services, event types, or detailed parameters within events.

Selecting API Call via CloudTrail as Event Source



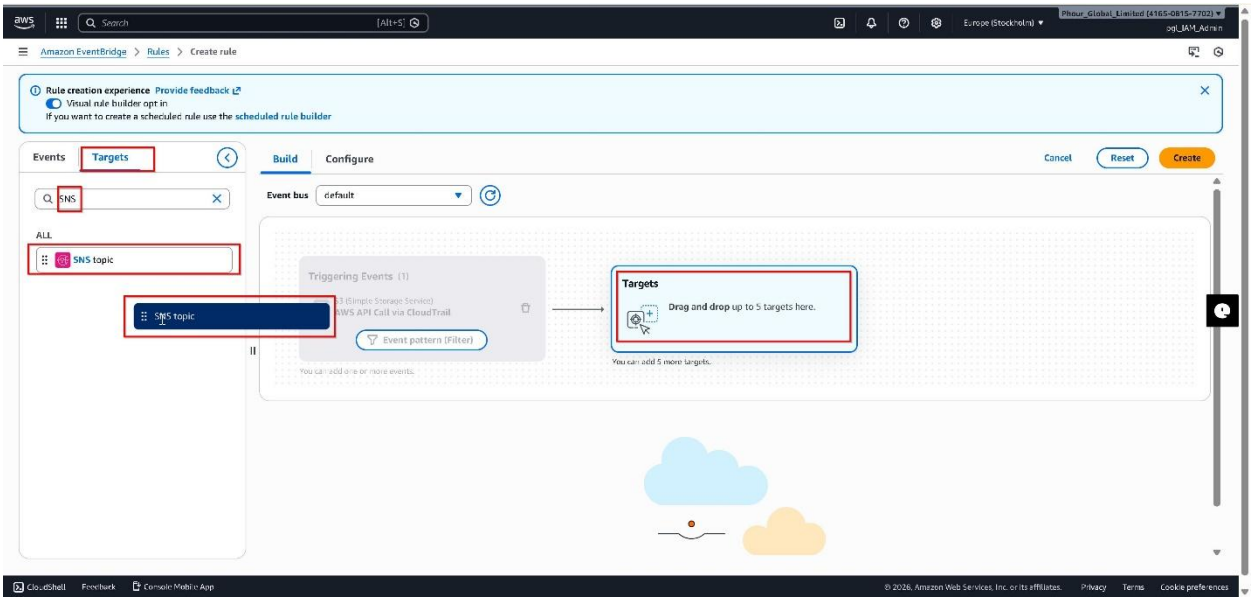
The event pattern builder shows CloudTrail selected as the event source. CloudTrail integration allows EventBridge to respond to any AWS API call. This enables monitoring specific actions like resource creation, modification, or deletion. The pattern can filter by AWS service and specific API operations.

Editing Event Pattern in JSON



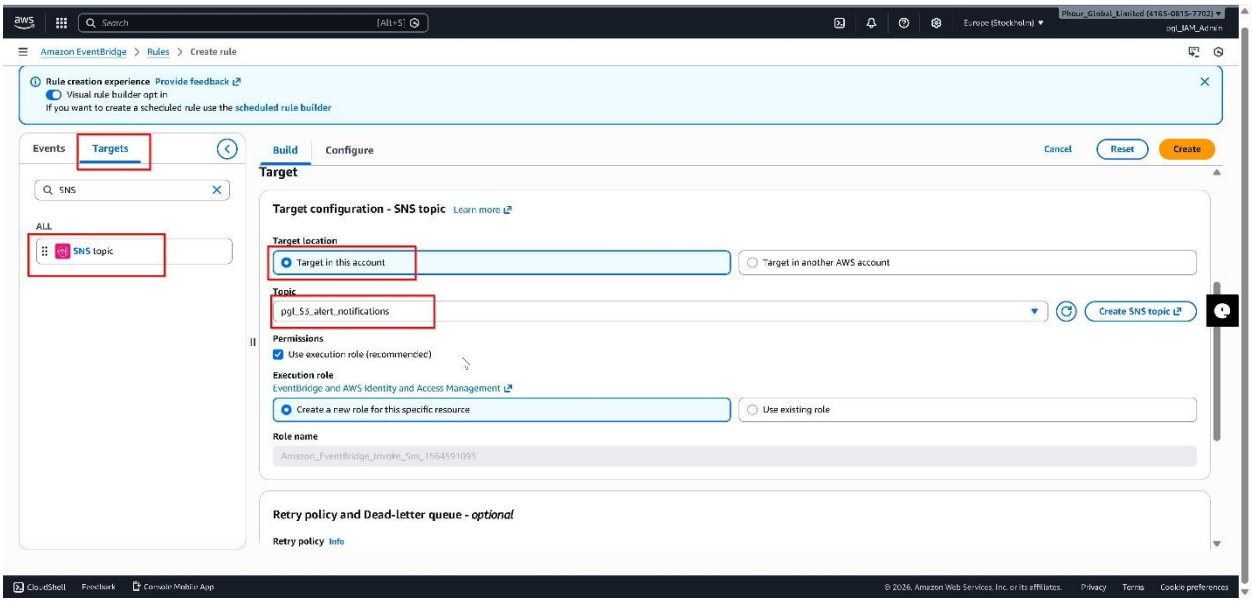
The JSON editor displays the event pattern code. The pattern specifies source as aws.s3 and detail-type as AWS API Call via CloudTrail. The eventName field is added to match specifically CreateBucket events. This pattern will trigger whenever a new S3 bucket is created in the account. JSON editing provides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

Creating SNS Target



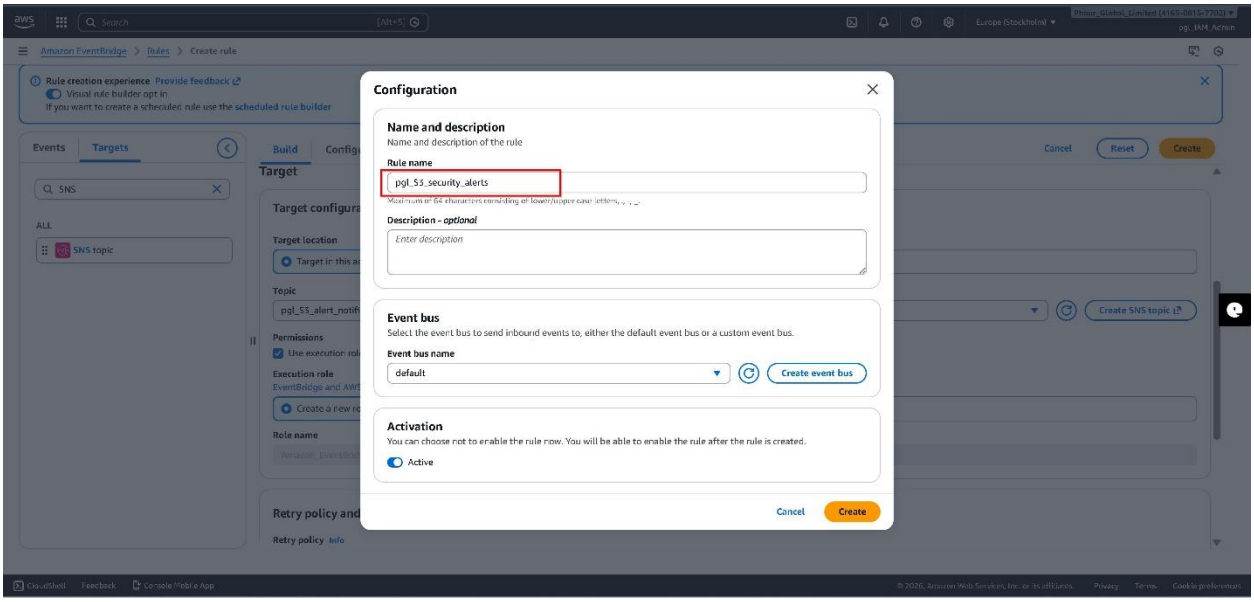
The target configuration shows SNS topic selected as the action to perform when the rule matches. Targets define what happens when events match the rule pattern. Multiple targets can be added to a single rule to trigger different actions. SNS target will publish a message to the specified topic which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

Configuring SNS Topic as Target



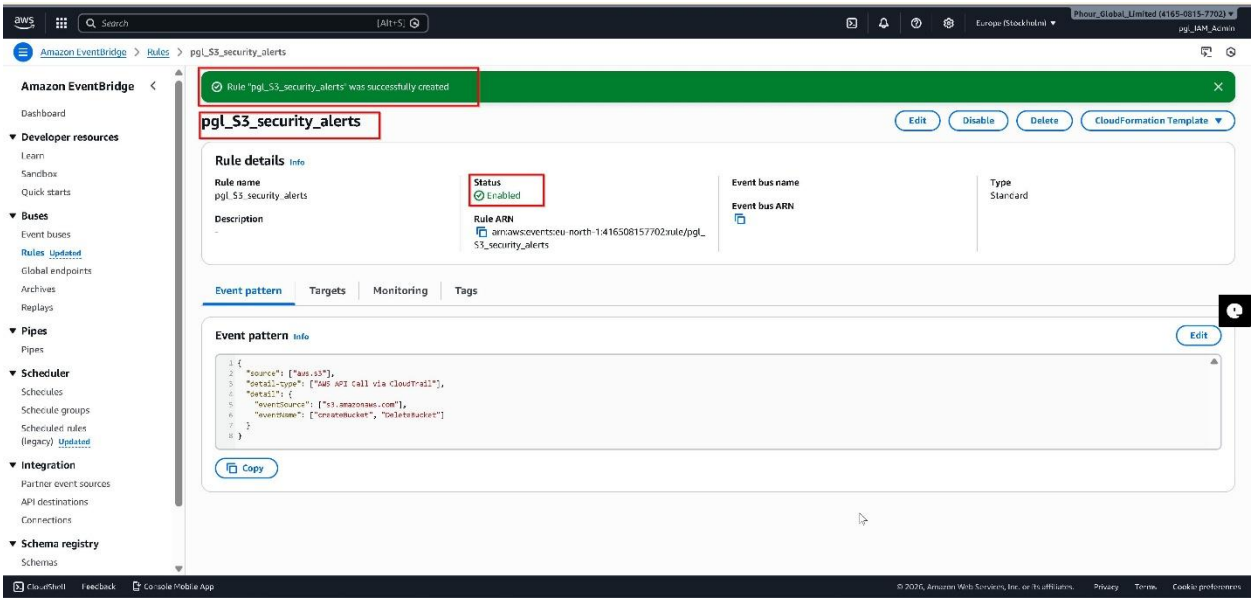
The SNS configuration shows selecting the Security-Alerts topic created earlier. The message input determines what content is sent to subscribers. The default setting sends the complete event details in JSON format. Custom messages can be configured to format the notification. EventBridge will automatically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

Naming Rule and Creating



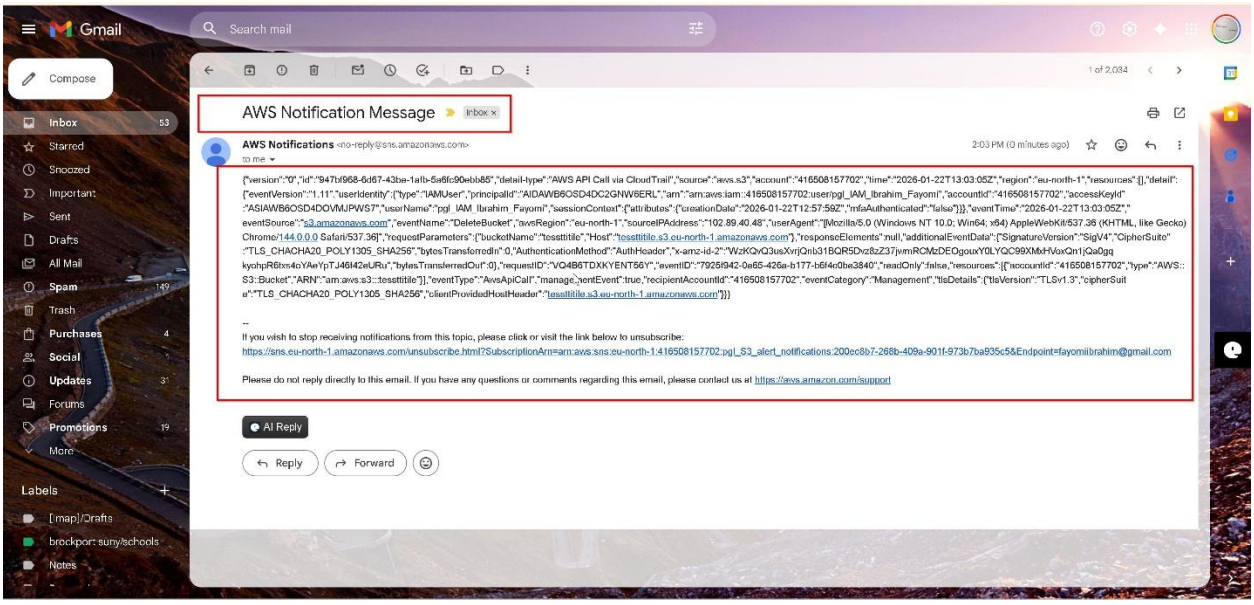
The rule details section prompts for rule name and description. The name clearly identifies the monitoring purpose such as S3-Bucket-Creation-Alert. Description explains what events trigger the rule and what actions result. The create rule button finalizes the configuration and activates monitoring.

Rule Created Successfully



The 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.