AWS Serverless SFTP Solution + Monitoring Solution

Requirements:

The overall requirement is an SFTP server with an AWS Cloud-native solution (without altering the external system user experience, in case of migration from on-premise SFTP servers).

High-level design

The following capabilities are included in addition to the AWS Transfer Family to reduce the operational maintenance & support (and to adhere to the PCI standards).

Part 1 : Base Solution

1. Core Module - AWS Transfer Family Integration with Secret Manager via API Gateway & Lambda Function. [An AWS serverless managed service for FTPS/SFTP file transfer. The user credentials are securely stored in SecretManager].
2. User, Group, GroupAdmin, and SuperAdmin hierarchy:- Enables the multiple business units to use the same underneath infra with complete data isolation.

Part 2 : Enhanced Monitoring Solution for reduced Operational Overhead.

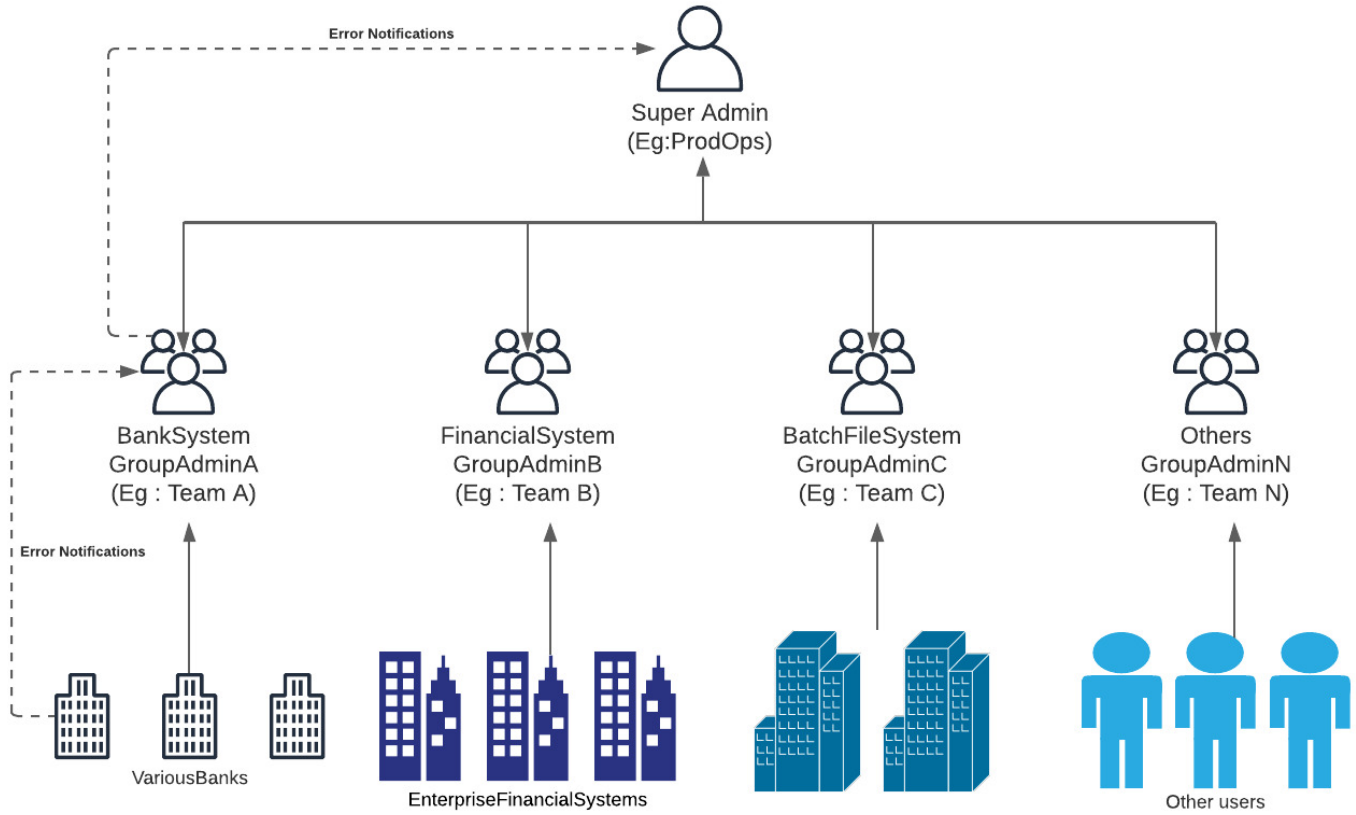
1. Secret Validation Module: A solution that constantly monitors the end user's / external system's information in the secret manager and sends notifications to GroupAdmin or Super Admin in case of discrepancies.
2. SSHKey Expiry Notification Module: An automated monitoring solution to notify external users to rotate keys every year. (PCI Standard Security)
3. Automated IPv4/IPv6 IP Whitelisting Module: A module that automatically keeps the prefix list up to date based on user information in the secret manager.

***Note: Kindly complete the "Deployment Prerequisites" provided in the document before proceeding with deploying the solution in any new AWS account. ** **Note: Enable CloudTrail for the Part2 of the solution to work. The part2 solution depends upon the CloudTrail logs for execution and hence it's impreative to enable the CloudTrail for the monitoring solution to operate as expected. ***

Users, GroupAdmin and SuperAdmin Hierarchy

The below diagram explains the hierarchy of Users, GroupAdmin's and SuperAdmin, and the error notification flow. In case of a user configuration issue, an error notification will be sent to the user's corresponding GroupAdmin. In the case of GroupAdmin's configuration issue, error notifications will be sent to SuperAdmin.

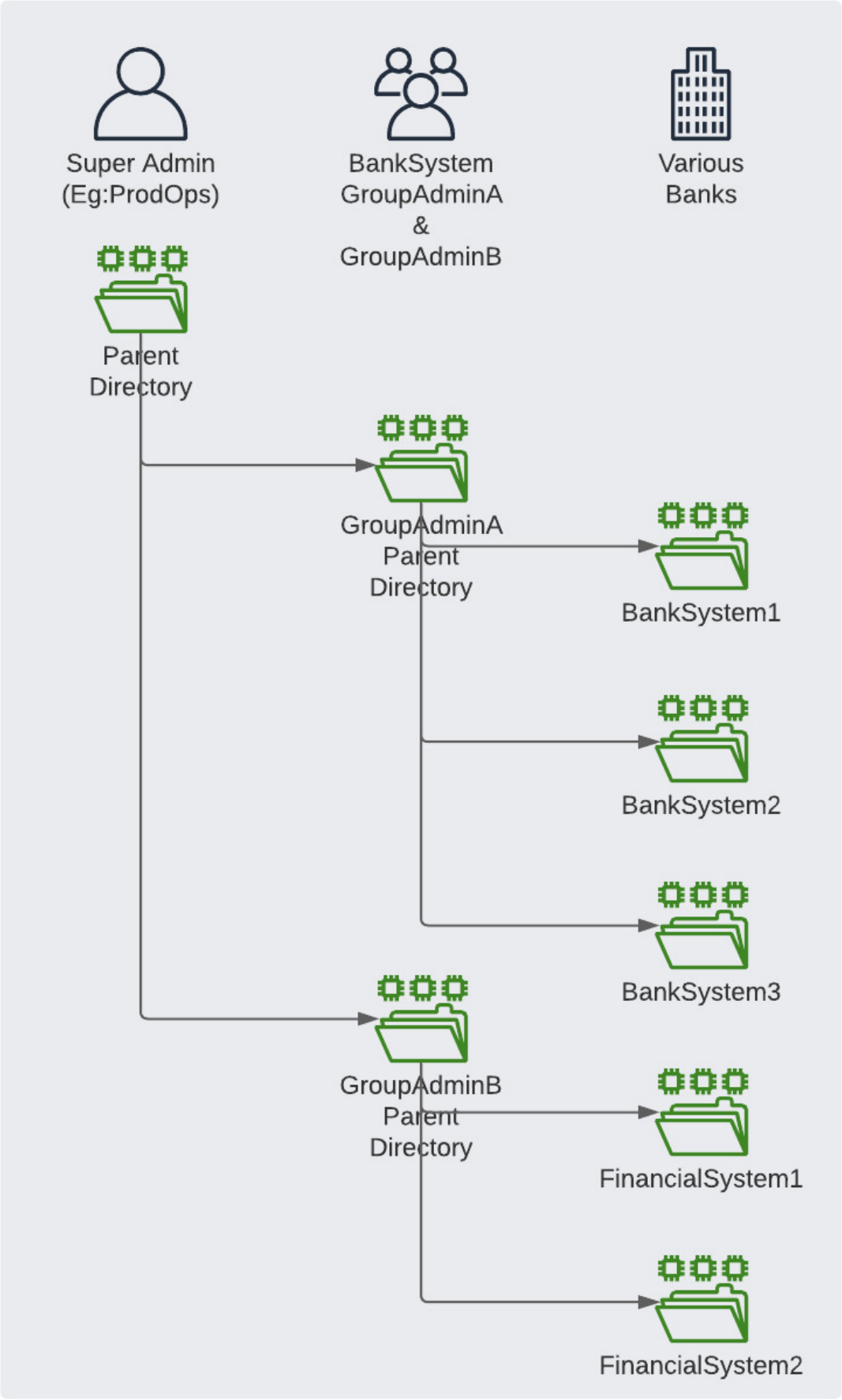
Note: All config information for the SuperAdmin user is expected to be correct.



Files and Folder Hierarchy

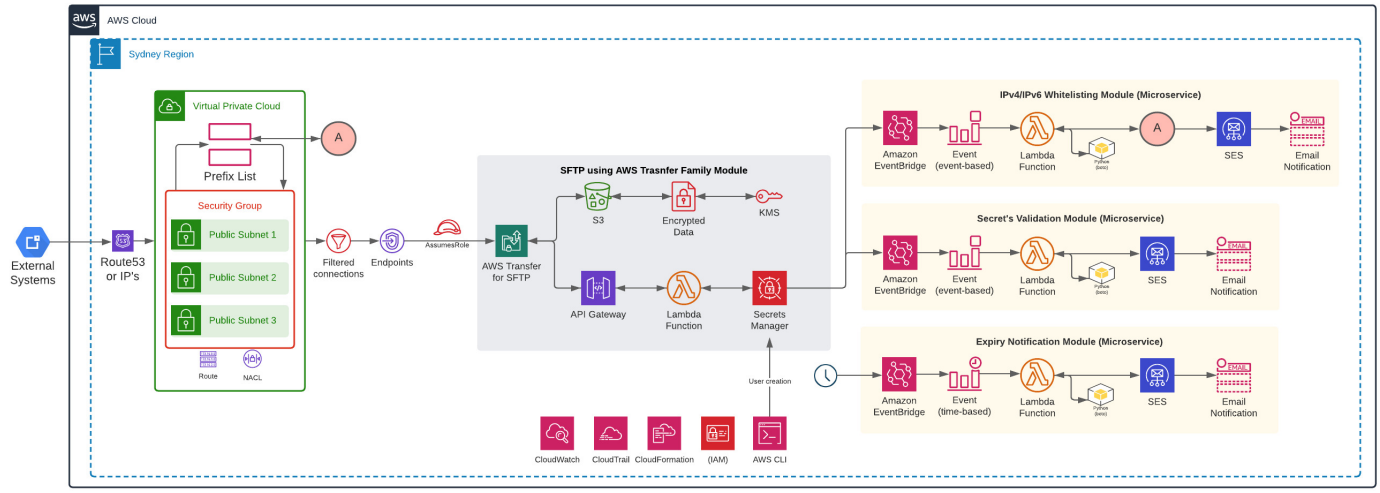
The below flow diagram explains the folder structures and User access to all the folders for Users, GroupAdmin and SuperAdmin.

- SuperAdmin has full access to all folders and sub-folders.
- GroupAdmin will have access to a sub-folder, which will act as GroupAdmin's parent directory and will have access to all further sub-folders underneath it.
- A user (bank user or financial system user) will only have to access it to its dedicated folder.
- All users will be explicitly denied to navigate to other users' folders and thereby providing granular access to all the users.



Architecture diagram (Core module, Secret Validation, Expiry Notification & IP Whitelisting modules)

The Cevo Serverless AWS SFTP server module contains the modules supported by the listed components. The overall architecture is explained in 2 parts.



Modules

1. SFTP server (AWS Transfer Family)
2. Secret Validation module (Lambda)
3. SSHKey Expiry Notificaiton (Lambda)
4. IPv4 Whitelisting (Lambda)
5. IPv6 Whitelisting (Lambda)

Components

All the above-mentioned modules works based on the below AWS and other components or software.

1. AWS Transfer Family - Acts as SFTP server for inbound connection.
2. S3 buckets - Storage system for AWS Transfer Family. (S3 can be replaced with EFS)
3. Secret Manager - Secure storage of user credentials
4. API Gateway - Integrates AWS Transfer Family with Secret Manager
5. Lambda - Integrates API Gateway with Secret Manager.
6. Lambda - Secret validation, SSHKey Expiry Notification, IPv4 and IPv6 whitelisting functions.
7. VPC, Subnet, Security Groups, VPC endpoint - Enables externals to securely access AWS Transfer family via a virtual private network.
8. Prefix list - A solution to store and easily manage CIDR which will be further referred by the security group for secured inbound connection.
9. EventBridge - Provides capability for events-based lambda trigger based on insert/update/delete in secret manager. Also provides the capability for scheduled lambda triggers based on date and time.
10. Cloudwatch - Stores all lambda execution logs, AWS Transfer Family access logs, Secret Manager's access logs, and API Gateway execution logs.
11. CloudTrail - Stores all AWS API access logs.
12. Cloudformation - Deploy & manage components using the AWS Cloudformation template.
13. SAM or Server Application Model - Deploy serverless applications like Lambda using the SAM template, which then converts to a CloudFormation template internally within AWS before deployment.
14. Docker - Runtime environment to validate the cloud formation template. (If Docker Compose is used)
15. Github/BitBucket & Buildkite/Jenkins - CI/CD pipeline for automated deployment.

A standard VPC and its subcomponents are created using the cloud formation template. The VPC has been created to provide a VPC endpoint and security group that can be integrated with AWS Transfer Family to send filtered traffic based on entries in the SecurityGroup / Prefix-list.

1. VPC
2. Subnet
3. Security groups
4. Route table
5. InternetGateway
6. NetworkAccessList
7. VPC endpoints
8. Prefix-list

AWS Transfer Family is a managed serverless service offered by AWS for SFTP/FTP file transfer. The AWS Transfer family can be integrated with either EFS/S3 for storage. The user credentials can be either stored directly on the AWS Transfer Family for users to access the SFTP. The other option is to integrate it with SecretManager to store the user credentials using API Gateway and Lambda. AWS provides this integrated module as a packaged solution.

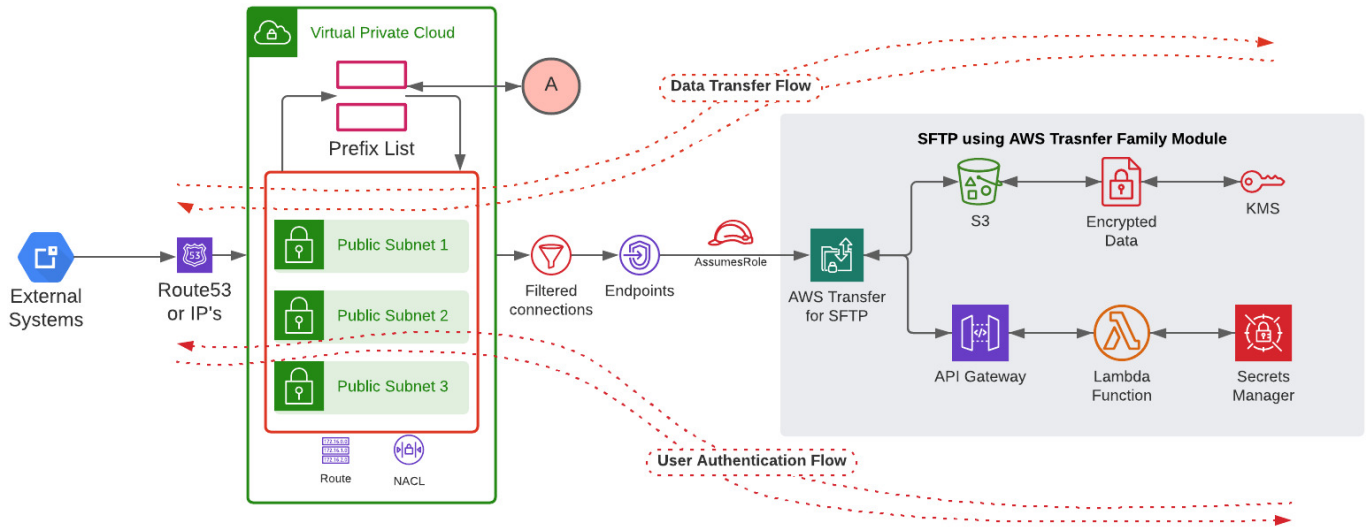
1. AWS Transfer family

2. API Gateway
3. Lambda
4. Secret Manager

The serverless application module is used to package and deploy the Lambda function and its respective code. AWS converts the serverless application module to a cloud formation template internally and stores a copy in the S3 bucket for deployment. Time & Event based lambda functions using EventBridge and Roles for Lambda functions are built using SAM template.

1. SAM
2. Event-based Lambda
3. Time-based Lambda
4. Roles for Lambda execution

DataTransferFlow and UserAuthenticationFlow



Data Transfer Flow:

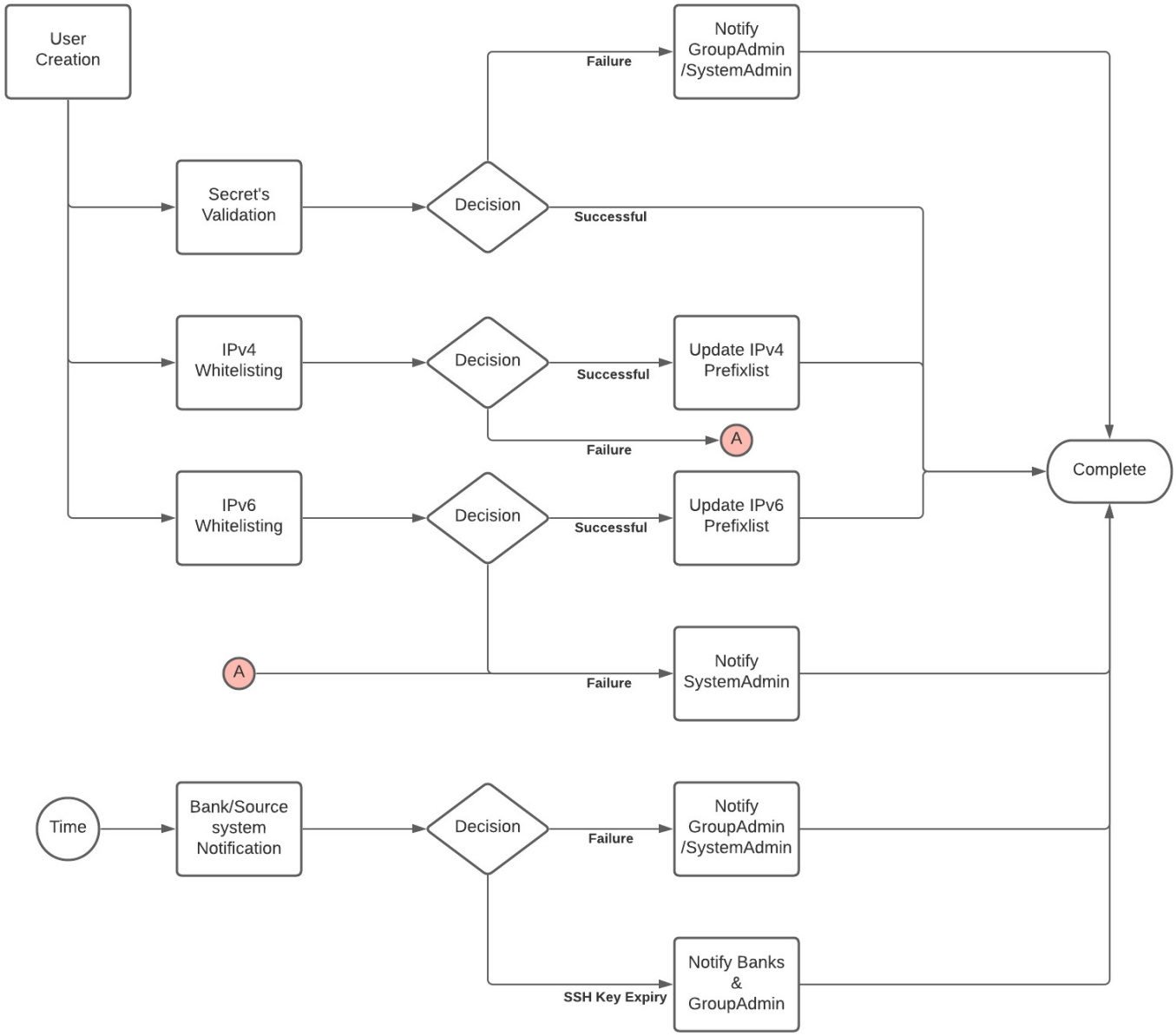
- A user enters the SFTP module via a dedicated Domain name or IP.
- Depending on the IP whitelisting on the Security Group, the user will be allowed further to access the Transfer Family or will receive connection rejection.
- In case of successful network authentication, the user assumes a role based on the user configuration in the Secret Manager and accesses the S3 bucket for placing/accessing/updating/removing files in the user's dedicated landing directory.

User Authentication Flow:

- A user enters the SFTP module via a dedicated Domain name or IP.
- Depending on the IP whitelisting on the Security Group, the user will be allowed further to access the Transfer Family or will receive connection rejection.
- In case of successful network authentication, the user accesses the Secret Manager via API Gateway and Lambda function with the username as a reference to retrieve information PublicKey, Role, and LandingDirecrtory for successful authentication and authorization to access the storage.
- Detailed AWS documentation for reference: https://aws.amazon.com/blogs/storage/enable-password-authentication-for-aws-transfer-for-sftp-using-aws-secrets-manager/

DataTransferFlow and UserAuthenticationFlow

The end-to-end functional flow diagram explains the order in which the microservices are triggered in case of a new user creation or update/delete on the existing users configured in the SecretManager. The microservice modules are executed either based on events or based on time based on the EventBridge configuration.



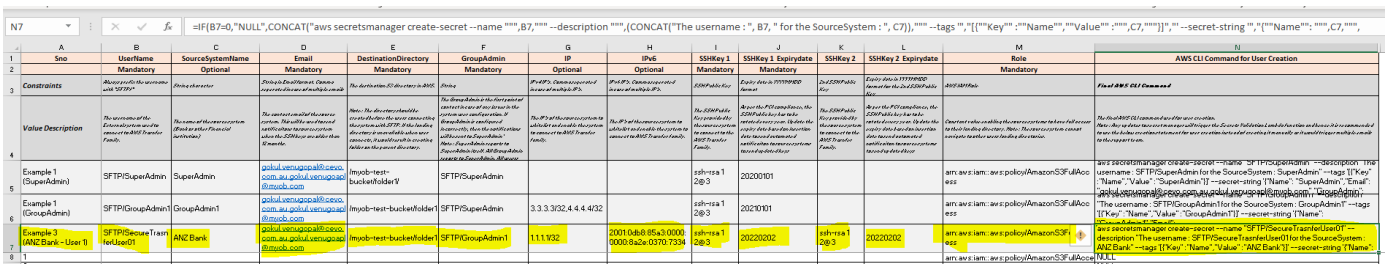
Functional Design

Step 1:

The AWSTrasnferFamily-AutomatedUserSetup excel will be used to create the AWS CLI commands to create the users in the Secret Manager. All the below user information needs to be updated in the relevant columns that generate the comprehensive CLI command.

Document for User Creation & Status Tracking: To be provided with this PDF document.

- Username - The username with which the external system connects to the CUSTOMER system.
- Name - The name of the external system
- GroupAdmin - The GroupAdmin is the Admin for one more user group. All configuration error notifications will be sent to the corresponding GroupAdmin.
- Email - The email contact of the external system (Comma-separated)
- Landing Directory - The S3 landing path for the respective user account.
- Role - The default S3 full access role. (Note: The user's access and its data access will be isolated for each user).



Step 2:

Create an S3 bucket folder for the user account before logging in for the first time. The AWSTrasnferFamily-AutomatedUserSetup excel provides AWS CLI command for S3 bucket creation based on the landing directory details.

Step 3:

Log in to AWS console using Single Sign-on user. Open AWS CloudShell and execute the above generated CLI commands to create an S3 bucket & subfolders and users in the secret manager. Note: If the AWS CloudShell is not avaliable in the specified region, then execute the commands using AWS CLI. Note: The user should be created in the same region, where the AWS Transfer Family is deployed.

Step 4:

Even-based lambda functions will be triggered to validate the user's secret values automatically. Error/warning notifications will be sent to GroupAdmin/SuperAdmin users if the values in the secret manager for the respective users are not aligned with the standard format.

Step 5:

Even-based lambda functions will be triggered to append/remove the IPv4 and IPv6 CIDR from the prefix list based on the user's secret values automatically. Error/warning notifications will be sent to SuperAdmin users if the system is unable to append/remove the CIDR to the prefix list. *Note: The info will be sent to SuperAdmin's because this will be a system /Technical issue that needs to be managed by the system admin.*

Step 6:

Time-based lambda functions will be triggered once a week to re-validate the secret values of all the users and will send emails to GroupAdmin/SuperAdmin if values in the secret manager for the respective users are not aligned with the standard format. Email notifications will be sent to SourceSystems & corresponding GroupAdmins if the SSH Key Expiry date is greater than 10 months notifying external systems to provide new SSH Public keys to meet the PCI standards. *Note: Notifications to banks and GroupAdmins will be sent constantly every week until the new keys are placed in the secret manager for the respective user and the existing keys are removed. It becomes the joint responsibility of external systems/source systems and application teams / GroupAdmin to remove the old keys from the secret manager to be compliant with PCI standards and hence notifications will be sent to all parties involved.*

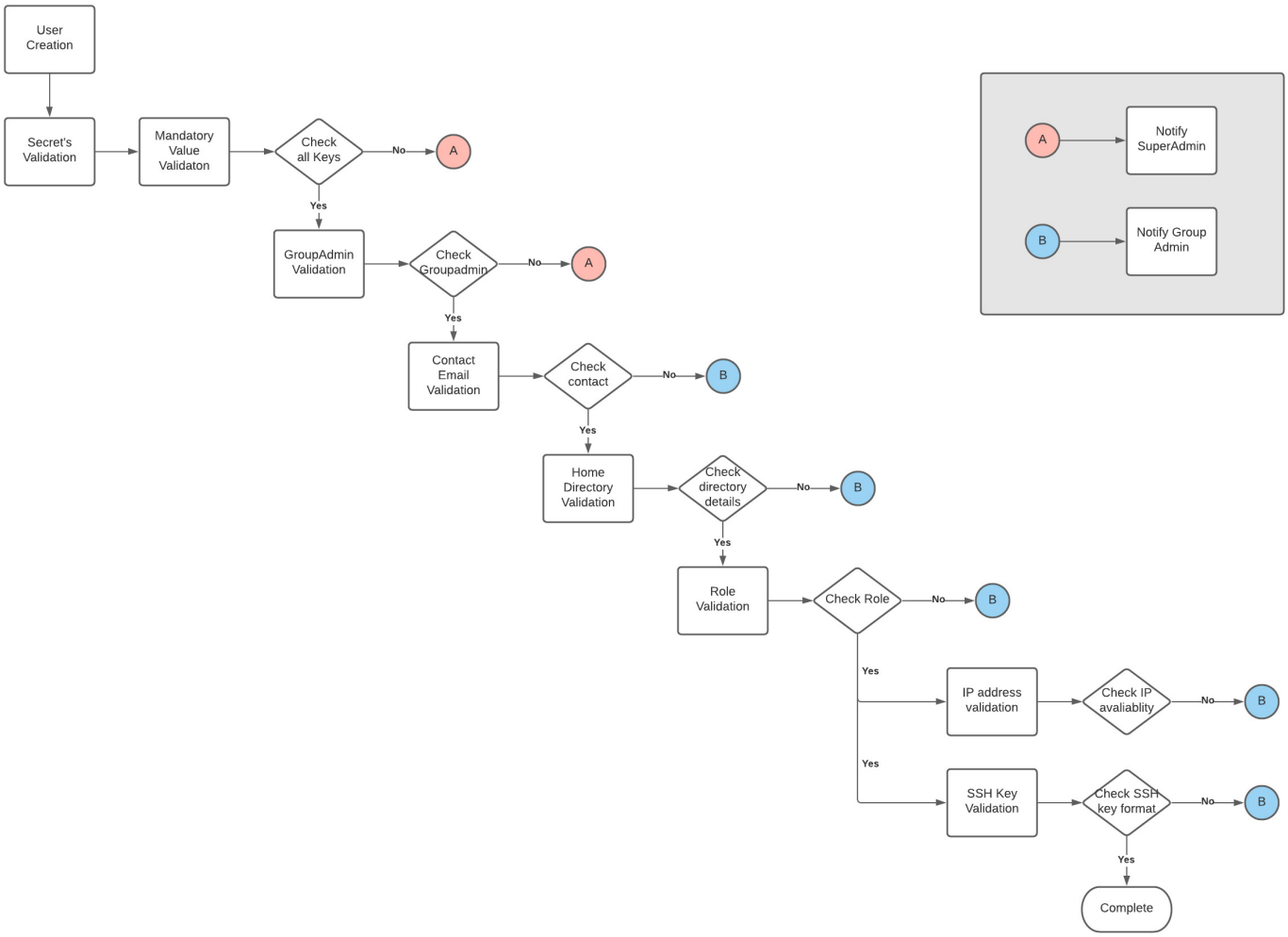
Technical detailed design

Step 1:User Setup and Configuration

Follow steps 1 to 3 in the "Functional Design" to configure new users in the secret manager and to create landing directories for the users.

Step 2: Secret Validation Module.

Even-based lambda functions will be triggered to validate the user's secret values automatically. The following sub-modules will be executed in the below order to validate the secret values.



Step 2.1: Collect Secret List

This module collects all the user accounts from the Secret Manager that has the prefix "SFTP/". This module collects all users like SuperAdmin, GroupAdmin, and Source System users. The user names will be passed as parameters in the loop on the following sub-modules. The secret Value Validation module treats all users alike and performs similar validation for all the user accounts.

Step 2.2: Collect Admin List

This module collects all the user accounts from the Secret Manager that has the keywords "admin", "Admin", "GroupAdmin" and "SuperAdmin". All admin users should have the admin keyword in the user name to be identified by the solution as an admin user. Note: This is necessary because the GroupAdmin value configured on the system user will be validated against this list to make sure that an admin user is configured as GroupAdmin.

Step 2.3: Collect User's Metadata (SecretString)

Based on the information collected by module 2.1, the name of users will be iterated and the secret string metadata will be collected for all the SFTP users from the SecretManager and will be stored in a variable (SSMmetadata). The values in this will be iterated in a loop on the below submodules.

Step 2.4: Mandatory Value Validation

This module validates the "Keys" in the Secret Manager Secret String for a given user. All users should have the following keys and values are either mandatory/optional depending on the nature of the key.

1. Name
2. Email
3. PublicKey
4. HomeDirectoryType
5. HomeDirectoryDetails
6. GroupAdmin
7. Role
8. IP or IPv6

If a user is missing the above-mentioned key, an email notification will be triggered to GroupAdmin (refer to "GroupAdmin Email Notification module" for more details). The Values for these Keys will be validated separately on the below modules and this module is responsible for only validating the Keys.

Step 2.5: GroupAdmin Value Validation

In this module, the value of the GroupAdmin will be extracted and will be compared against the list of the GroupAdmin users extracted as part of step 2.2. If the value in the GroupAdmin is not part of the list of GroupAdmin's in the system, an email notification will be sent to SuperAdmin. (refer to "SuperAdmin Email Notification module" for more details)

Step 2.6: Contact Email Delimiter Validation

This module validates the delimiter used in the contact email of the user. These email ids will be used to send emails notification to source systems in case of SSH Keys expiry and hence it is mandatory to configure the emails with the right delimiter for the system to recognize. A list of common mistakes or mistyped delimiters are configured and the contact email value is validated against this check. Note: If there is a new pattern of typos identified in the future, this module can be updated accordingly. If an error delimiter is identified on the contact email, then an email will be sent to GroupAdmins.

Step 2.7: Contact Email Value Validation

This module validates the emails id against the standard email pattern using regex. These email ids will be used to send emails notification to source systems in case of SSH Keys expiry and hence it is mandatory to configure the proper emails for the system to recognize and send notifications without any errors. If an email is identified on the contact email that doesn't follow the standard email pattern, then an email will be sent to GroupAdmins.

Step 2.8: Home Directory Value Validation

This module has 2 parts. Setting up a landing directory for individual users requires 2 key-values pairs to be added to the secret string of the user. Out of those 2 values, 1st key-value pair is a constant value and the second one varies based on the landing directory of the user.

- *Step 2.8.1*: Part 1 of this module checks the constant value. This module ensures the below value is present in the secret string of a user in Secret Manager. In case of a typo or missing values, an email will be sent to the GroupAdmin team. Key-Value pair *HomeDirectoryType = LOGICAL*
- *Step 2.8.2*: Part 2 of this module checks the HomeDirectoryDetails value, which is different for all users in the system. The module checks if the values are populated in the pattern that can be recognized by the system and sends notifications to GroupAdmin in case of pattern error. Expected pattern : *HomeDirectoryDetails : [{"Entry": "/", "Target": "DESTINATION-S3-FOLDER-PATH"}]*

Step 2.9: Role Value Validation

This module validates the value of the key-value pair "Role". All the users are expected to have a constant AWS-managed policy, that provides full access to the users for their dedicated landing directory. The module checks the below constant value for all the users and sends notifications to GroupAdmin in case of incorrect or missing configuration.

Role = arn:aws:iam:AccountNumber:aws:role/TransferS3AccessRole

Step 2.10: IP Value Validation

This module checks the existence of "Key" IP or IPv6 in the secret string of a user. This module also checks the CIDR patterns for IPv4 and IPv6 using regex. However, the pattern validation is commented out currently due to reducing the complexity and can be enabled anytime.

Note: It is not mandatory to have CIDR values for the IP or IPV6 Key-Pair. However, it is imperative to have the Key IP or IPv6 configured on the secret string and the value column can be left empty. In case of no CIDR or incorrect CIDR, the source system will not be able to connect to the AWS SFTP module of the CUSTOMER for file transfer.

Step 2.11: SSH Key Structure Validation

This module has multiple submodules. The values in these modules are crucial for the source systems to connect to the AWS SFTP to transfer files. Also, the second part of the values is used to send a notification to the source system regarding the SSH Public Key expiry to comply with PCI standards.

- *Step 2.11.1*: PublicKey Value availability check The value for the PublicKey Key-Value part in the secret string of a user cannot be empty. This module checks SSH key avaliablity. If the user doesn't have an SSH key, then a dummy entry as follows needs to be populated in the value section.

ssh-rsa 1 2@3 Updated(YYYYMMDD)=20220202

Note: This enables the system to perform similar checks for all the users without any exceptions. A valid SSH key will be validated and in case of no SSH key, a dummy key will be validated similar to an actual key.

- *Step 2.11.2*: PublicKey count check The SSH public key needs to be comma-separated and AWS Transfer can handle 2 public for a given user. This module checks, if there are only 2 commas separated by Public Keys configured in the secret string for a user in Secret Manager.
- *Step 2.11.3*: PublicKey and Updated timestamp position check An SSH PublicKey has 4 parts.

1. ssh-rsa is the first standard part
2. The second part is the encrypted ssh key from the user.
3. The third part of the user and server name.
4. The last part is UPDATED(YYYYMMDD)=DATE, a custom value provided by the support team used for source system expiry notification.

This module ensures that all SSH PublicKey has all 4 parts in them.

- *Step 2.11.4*: SSH Key validation (Commented out) This module decrypts the encrypted SSH key and validates if the keys are decryptable and in the right encryption format. This part of the code is commented on based on the CUSTOMER's request.
- *Step 2.11.5*: Updated timestamp Key-value check This module verifies if the timestamp provided on the 4th section of the SSH Public Key is in YearMonthDate (YYYYMMDD) format. The data provided in this section is crucial for sending notifications to SourceSystems.

In case of any submodules failure, an email will be sent to the GroupAdmin team notifying the issue and issue description.

- *Step 2.11.5*: Public Key Expiry calculation & notification This module will be executed, if and only all the above modules and submodules are successful for a given user. This module extracts the updated timestamp provided in the PublicKey and compares it against the current date to define the age of the PublicKey. If a PublicKey is older than 10 months, then this module will send out notification emails requesting for new PublicKey to SourceSystem with GroupAdmin's in a loop. Note: If the GroupAdmin associated with a user doesn't have email configured, then SuperAdmin email IDs will be looped in CC while sending emails to SourceSystems. This is to ensure at least one part of the support team is in the loop while sending emails to the external systems. (this is only enabled for the SourceSystem notification module and no notification will be sent to external systems as part of the secret validation module and it's controlled by the Enable_Bank_Notification parameter. The values can be True and False)

Note: Notifications will be sent once every week to SourceSystem until the old keys are removed from the user's secret string in the Secret Manager.

Step 2.12: GroupAdmin Email Notificaiton Module

This module has an email template to send emails to GroupAdmin teams. This module has 3 input parameters Email_Subject, Email_Body, and Username. The Email Subject, Email Body, and Username for which the validation is performed can be passed in as parameters and emails will be sent with this information included.

```
def Function_SupportTeam_sendemail(Email_Subject, Email_Body, Username):
```

Note: If the email id's in the GroupAdmin are empty, then SuperAdmin email IDs will be used.

Step 2.13: SuperAdmin Email Notification Module

This module has an email template to send emails to SuperAdmin teams. This module has 3 input parameters Email_Subject, Email_Body, and Username. The Email Subject, Email Body, and Username for which the validation is performed can be passed in as parameters and emails will be sent with this information included.

```
def Function_SuperAdminTeam_sendemail(Email_Subject, Email_Body, Username):
```

Note: A valid email address is expected to be populated for SuperAdmin users.

Step 2.14: SourceSystem Email Notification Module

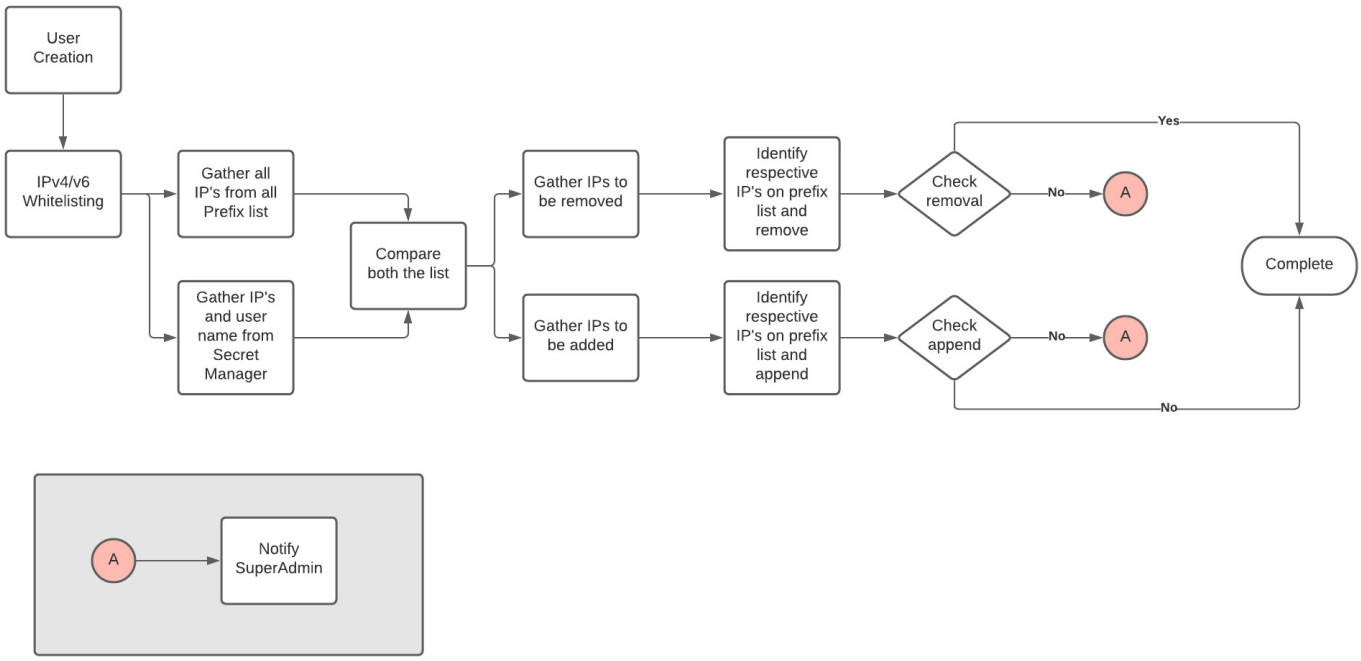
This module has an email template to send emails to SourceSystem. This module will be invoked for a user only if all secret string validation has passed and if the updated date configured in the PublicKey is older than 10 months from the current date.

```
def Function_BANK_sendemail(Email_Subject, Email_Body, Username):
```

Note: GroupAdmin emails will be looped in CC when sending emails to SourceSystems. If the GroupAdmin email section doesn't have values, then SuperAdmin emails will be used instead of GroupAdmin emails. Special Note: A valid source email is hardcoded in the Python script on both the Secret Validation module and SSH Key expiry notification module. This source email or domain has to be verified by SES in the same region.

Step 3: IPv4/IPv6 Whitelisting Module

Even-based lambda functions will be triggered to validate the user's secret values automatically. The following sub-modules will be executed in the below order to validate the secret values.



Step 3.1: Collect all the available prefix-list

This module collects all the Prefix lists on the AWS region based on matching values in the name of the prefix list (*AUTOMATED_PREFIX*).

Step 3.2: Collect all the IP's from the prefix-list

The prefix list collected from the previous step is iterated in a loop to fetch all the CIDR values stored in the prefix list.

Step 3.3: Collect all the Users and their secret string metadata from the Secret Manager

This module collects all the user accounts from the Secret Manager that has the prefix "SFTP/". This module collects all users like SuperAdmin, GroupAdmin, and Source System users. The user names will be passed as parameters to fetch all the secret sting values of the user.

Step 3.4: Collect all the CIDR's for all users

This module extracts the CIDR's of IPv4 or IPv6 from the gathered secret string values of all users. All IPs are stored in an IPv4Prefixlist_Entries_finallist variable to be compared against the overall CIDR's in the prefix list.

Step 3.5: Compare the prefix-list CIDRs with User CIDRs to identify Append and remove list

The CIDRs extracted from the prefix list and Secret manager are compared against each other to identify the list of CIDRs that need to be added to the prefix list and the list of CIDRs that need to be removed from the prefix-list.

- 1. Prefix-list CIDRs - SecretManager CIDR's = CIDRs to be appended
- 2. SecretManager CIDRs - Prefix-list CIDR's = CIDRs to be removed

Step 3.6: Remove the IPs from all the prefix-list based on the removal list.

The identified CIDR's that needs to be removed from the prefix-list is passed to IPV4_Remove_Function. This module compares the overall list with all prefix-list in a loop and removes the identified CIDR's from the respective prefix-list in bulk. An email notification will be sent to SuperAdmin in case of failure.

Step 3.7: Append the IPs to the prefix-list based on the append list.

This module gets CIDR's & comments (the username from the SecretManager) as input. The CIDR's are iterated in a loop and added to the prefix-list one at a time. While adding the entries the module checks the max entries that a prefix-list can accommodate and count of current entries in the prefix list. If a prefix list is deemed to be fully occupied based on the above check, then the next prefix list will be picked up for adding the CIDRs. If all prefix list is full and if there are no further prefix-list available to accommodate the CIDR, an email notification will be sent to the SuperAdmin team. Multiple Note: Email notifications will be sent in case of an issue with adding multiple CIDRs to the prefix list.

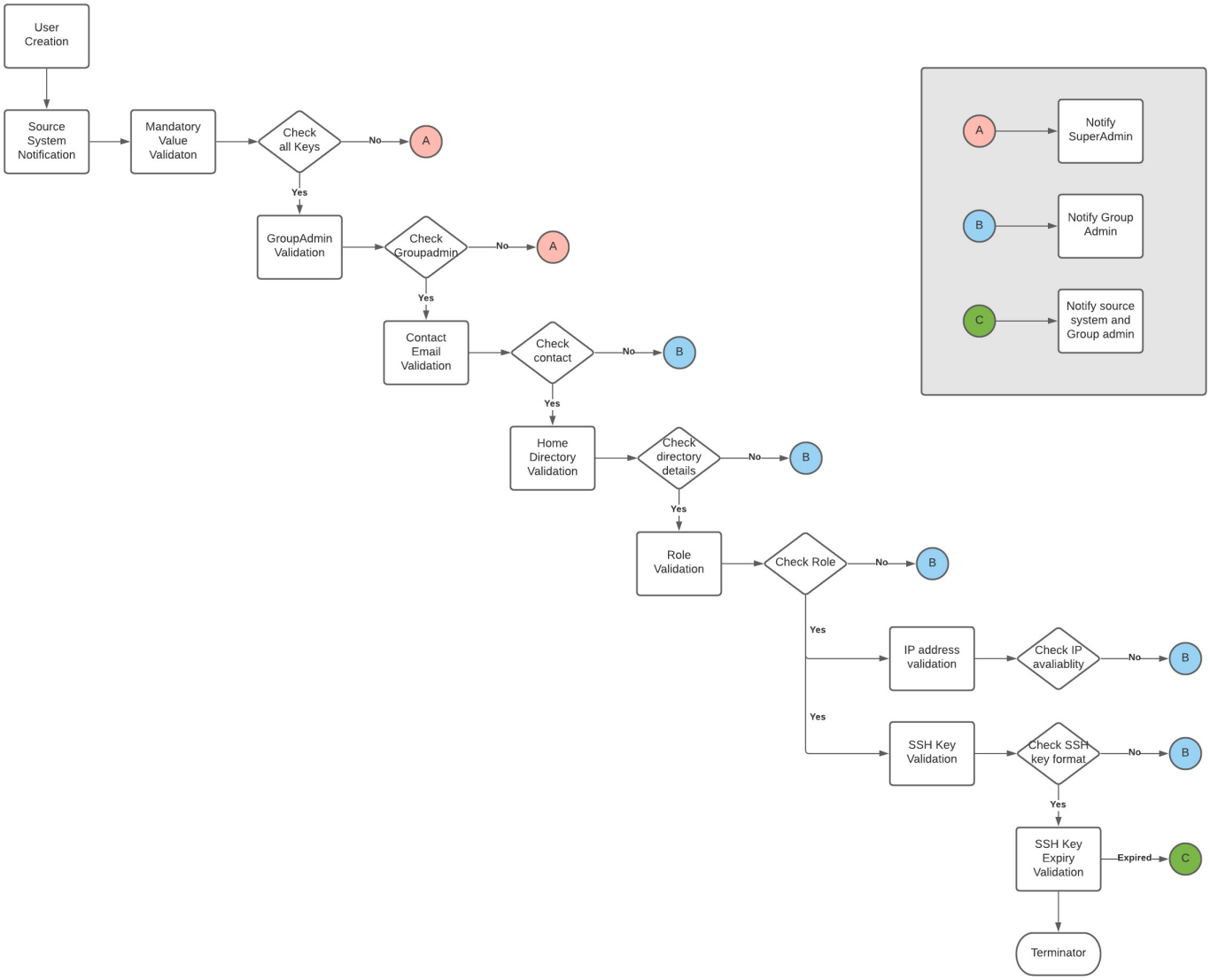
Step 3.8: Notify SuperAdmin in case of append failure or no space to append the IPs in the prefix list.

This module has an email template to send emails to SuperAdmin teams. This module has 3 input parameters Email_Subject, Email_Body, and Username. The Email Subject, Email Body, and Username for which the validation is performed can be passed in as parameters and emails will be sent with this information included.

Note: A valid email address is expected to be populated for SuperAdmin users.

Step 4: Source system SSH key Expiry Notification Module

Email notifications will be sent to SourceSystems & corresponding GroupAdmins if the SSH Key Expiry date is greater than 10 months notifying banks to provide new SSH Public keys to meet the PCI standards.



Step 4.1:

This module executes all the sub-modules available on secret value validation checks and performs the below sub-module in addition to sending notifications to banks.

Step 4.2: SSH Key Expiry date Validation (Applicable for SSHKey Expiry Notification Module only)

"Enable_Bank_Notification = True"

This module will be executed, if and only all the above modules and submodules are successful for a given user. This module extracts the updated timestamp provided in the PublicKey and compares it against the current date to define the age of the PublicKey. If a PublicKey is older than 10 months, then this module will send out notification emails requesting for new PublicKey to SourceSystem with GroupAdmin's in a loop. Note: If the GroupAdmin associated with a user doesn't have email configured, then SuperAdmin email IDs will be looped in CC while sending emails to SourceSystems. This is to ensure at least one part of the support team is in the loop while sending emails to the external systems. (this is only enabled for the SourceSystem notification module and no notification will be sent to external systems as part of the secret validation module and it's controlled by Enable_Bank_Notification parameter. The values can be True and False)

Notifications will be sent once every week to SourceSystem until the old keys are removed from the user's secret string in the Secret Manager.

Configurable Parameters

Source Email-ID for Email notifications

The Source Email has to be a verified email id. For example, the domain sftptrasnfer.CUSTOMER.com has used the source email and the complete MAIL FROM id is notification@partner.CUSTOMER.com. While requesting domain verification and custom mail from the address, the AWS SES requires the below DKIM CNAME, TXT, and MX records to be added to Route53.

Since the domain partner.CUSTOMER.com is maintained in a separate AWS account, this has to be added manually.

Type	Name	Value
CNAME	5ag5bitbyg6yrc6avfeykgcc4lrzcs24._domainkey.partner.CUSTOMER.com	5ag5bitbyg6yrc6avfeykgcc4lrzcs24.dkim.amazonses.com
CNAME	ggsx4dyzhdeh2drhilupeqsz7kddisun._domainkey.partner.CUSTOMER.com	ggsx4dyzhdeh2drhilupeqsz7kddisun.dkim.amazonses.com
CNAME	7eo6agncc7o5li4hef6ugzusblqfahtg._domainkey.partner.CUSTOMER.com	7eo6agncc7o5li4hef6ugzusblqfahtg.dkim.amazonses.com

Type	Name	Value
MX	notification.partner.CUSTOMER.com	10 feedback-smtp.ap-southeast-2.amazonses.com
TXT	notification.partner.CUSTOMER.com	v=spf1 include:amazonses.com ~all

Use the custom "MAIL FROM" entry in the below scripts as the source email id.

- Notify.py
- Validate.py
- IPv4_Whitelist.py
- Pv4_Whitelist.py

Cloudformation template parameters

Below are the configurable parameters from cloud formation templates.

1. Option A: Add more Security Groups and Prefix list in the CloudFormation template.
2. Option A: Raise a request with AWS on the Service Quota to increase entries that can be added to a security group from 60 to 1000. The max value should not be altered in the parameter file unless the service quota limit increase is approved by the AWS team.

```
- Env: "dev"
- VpcCidr: "10.0.0.0/16"
- PublicSubnet1Cidr: "10.0.1.0/24"
- PublicSubnet2Cidr: "10.0.2.0/24"
- PublicSubnet3Cidr: "10.0.3.0/24"
- EnableVpcFlowLogs: "true"
- FlowLogsRentionPeriod: "3"
- HostedZone: "/hostedzone/ZZQAW82MHZA5Q"
- DomainName: "dev-securetransfer.partner.CUSTOMER.com"
- IPv4PrefixlistCidrlimit: "30"
- IPv6PrefixlistCidrlimit: "30"
```

Note: The max limit of IPv4PrefixlistCidrlimit or IPv6PrefixlistCidrlimit is 60 as these prefix lists are attached to the security group and the maximum entry that a security group can accommodate is 60. If the requirement is to store more than 60 CIDRs then consider one of the following options.

Configurable parameter in Python Scripts on Lamda funcitons

```
1. SourceEmailid= "XXXXXXXX@CUSTOMER.com"
2. Notification_Exclusion_list= ["Sample"]
3. Enable_Bank_Notification = True | False
```

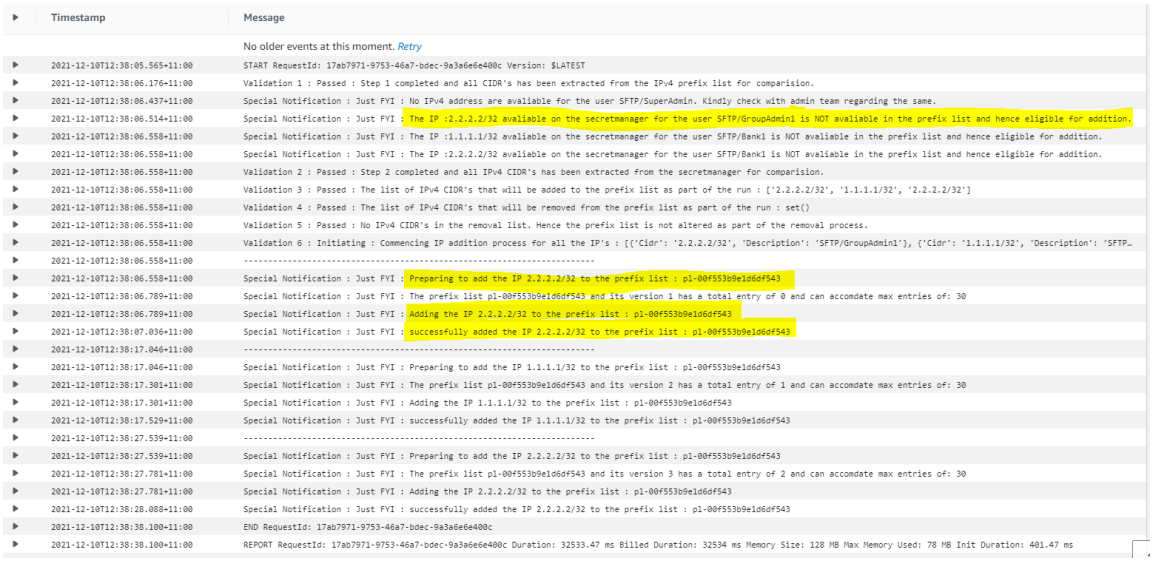
- The Source email-id or domain has to be a verified identity. The verification has to be on the SES service in the same region.
- The Notification Exclusion list enables the support team to exclude certain users from the checks. This could be one of the special cases.
- Enable_Bank_Notification parameter decides whether notification is to be triggered to a source system or not. For the SSH Key Expiry Notification module, this has to be set to True.

Logging

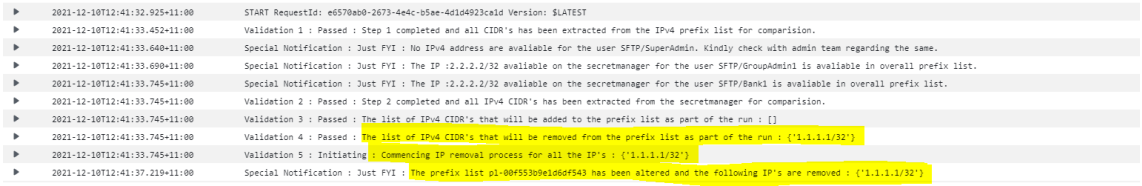
The Application and its microservices are configured to send logs to Cloudwatch at each step of the execution. It is recommended to follow the validation order and formatting while adding new logs or while appending the existing logging info for readability.

User Creation - Secret Validaton Module

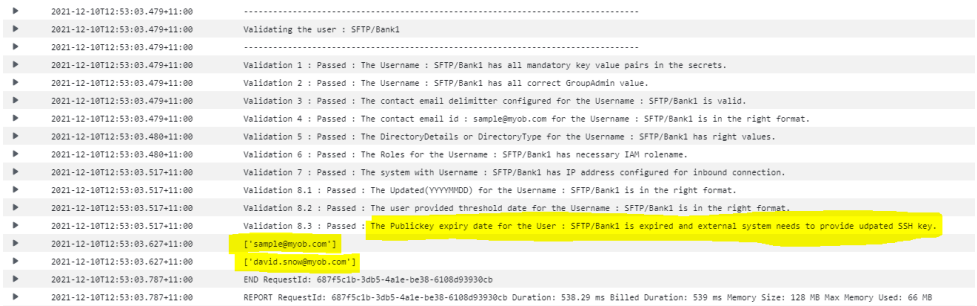
IPv4/V6 Whitelisting Module - Appending IP logs



IPv4/V6 Whitelisting Module - Removing IP logs

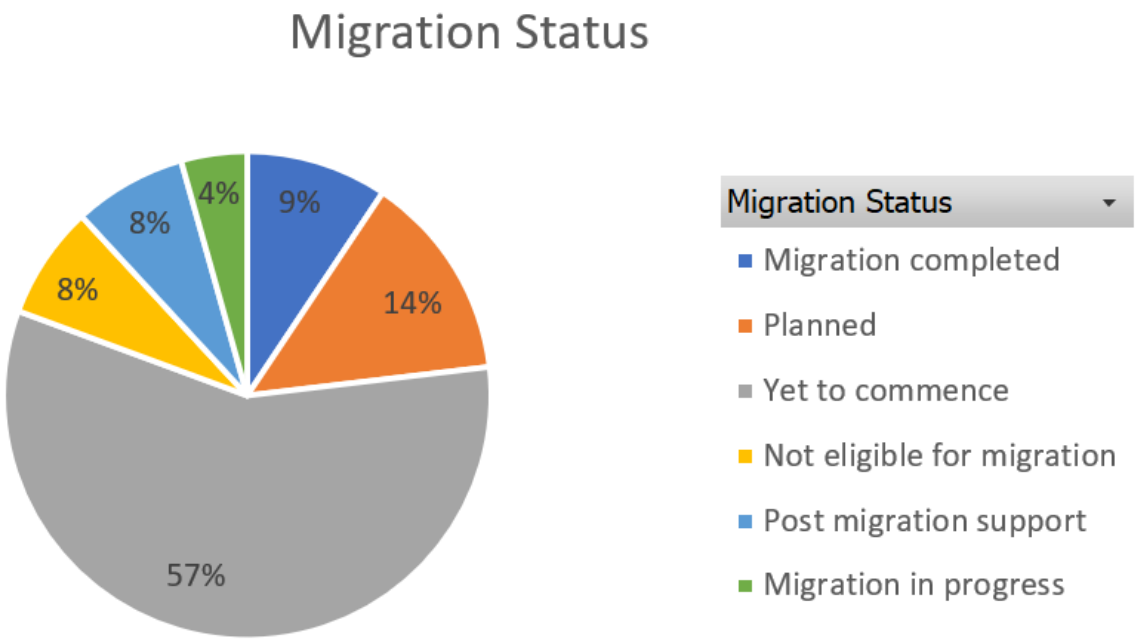


SSH Key Expiry - Source System Notification Module



Migration Tracker

The StatusTracker worksheet on the AWSTrasnferFamilt-UserSetupAutomation sheet can be used to track and report the status of user migration from on-premise to cloud-based SFTP solutions. Macro's are enabled to auto-update the pie chart based on the status update of each user.



Summary :

Modules	Lines of Code
Cloudformation for Network Layout	1000
Python for Microservices (800*2)	800
Serverless Application Model for App	200
Shell Scripts for deployment	100
Unique lines of code	**~2100**

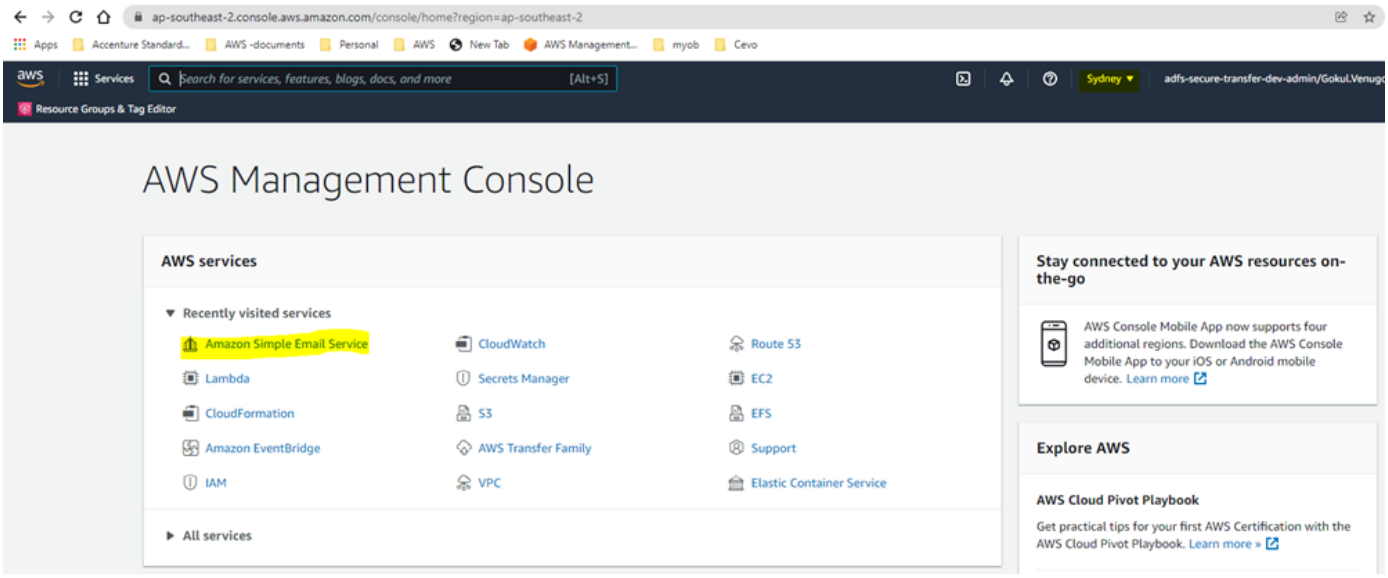
Deployment Prerequisites :

- Enabling Production Status for SES. This enables the SES to send emails to all external email id's (Bank systems for example)
- Increasing the Service Quota limit of inbound and outbound entries for security groups. This enables the storage of almost 1000 CIDR's per security group.
- Verifying an email address or domain to use it as "From ID" for emails triggered from SES.
- Enable CloudTrail for the Part2 of the solution to work. The part2 solution depends upon the CloudTrail logs for execution and hence it's impreative to enable the CloudTrail for the monitoring solution to operate as expected.

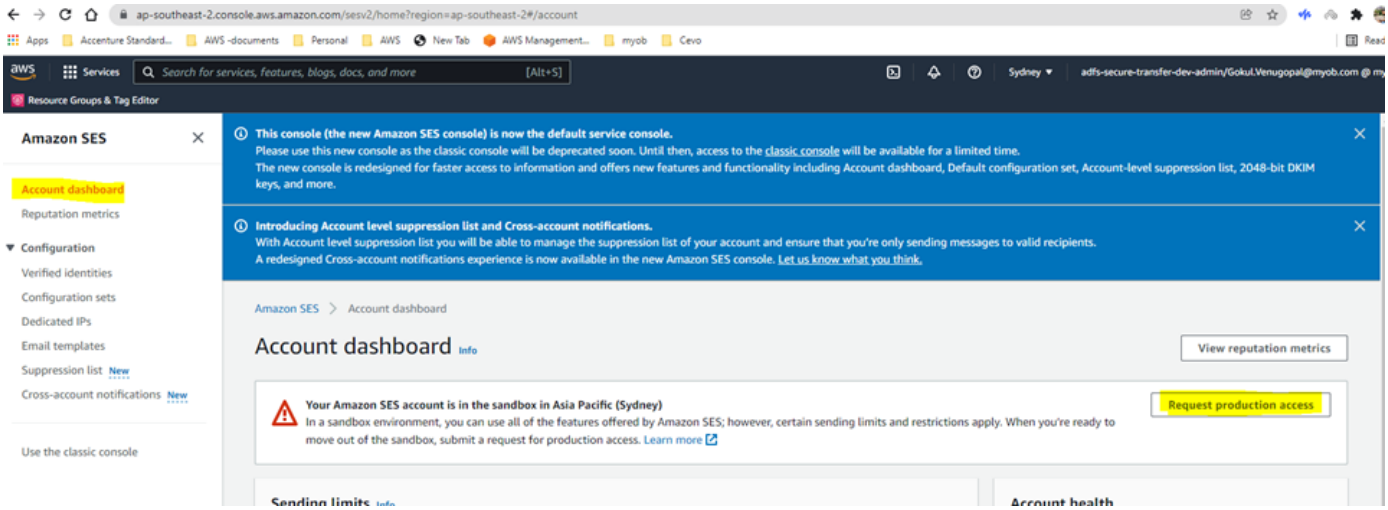
1. Enabling Production Status for SES

Reason: By default, the SES (Simple Email Service) in a region will not be in production status. This limits the AWS SES to send emails to only verified identities. To verify an identity, the email address verification has to be requested, followed which AWS will send an email to the target email address asking for confirmation to be part of the verified identity of a particular AWS account. If the user approves the request, it enables the AWS account to send emails outbound using the SES service to the verified identity. If the SES status is updated to Production Status, SES can send emails to any email ids without the verification process. (For example: Marketing emails)

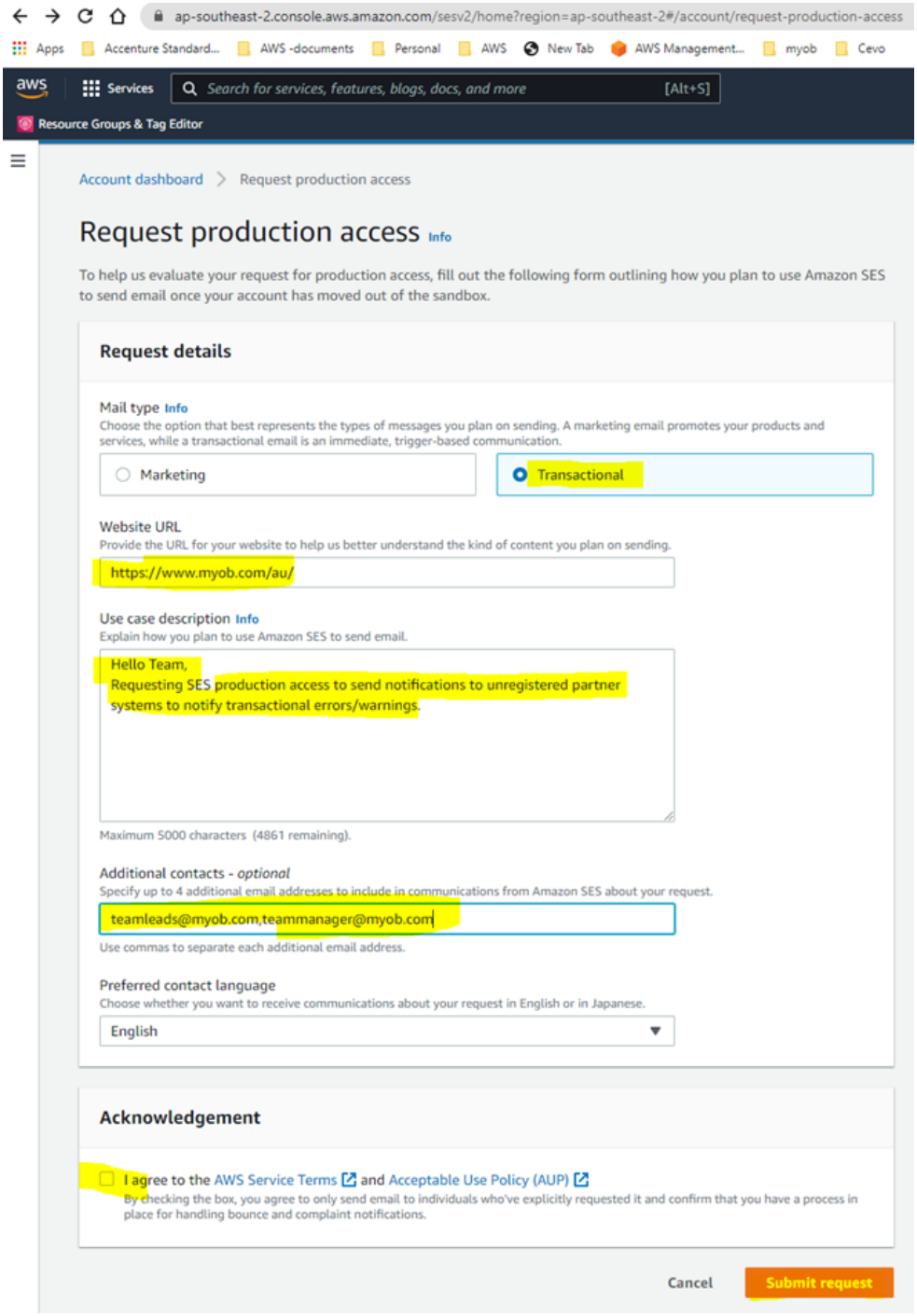
Step 1: Search for Amazon Simple Email Service in the Search bar in the AWS console.



Step 2: Under the Account Dashboard in the Amazon Simple Email Service console and select the "Request Production Status".



Step 3: Provide the Business Justification for requesting Production Access and submit the request.

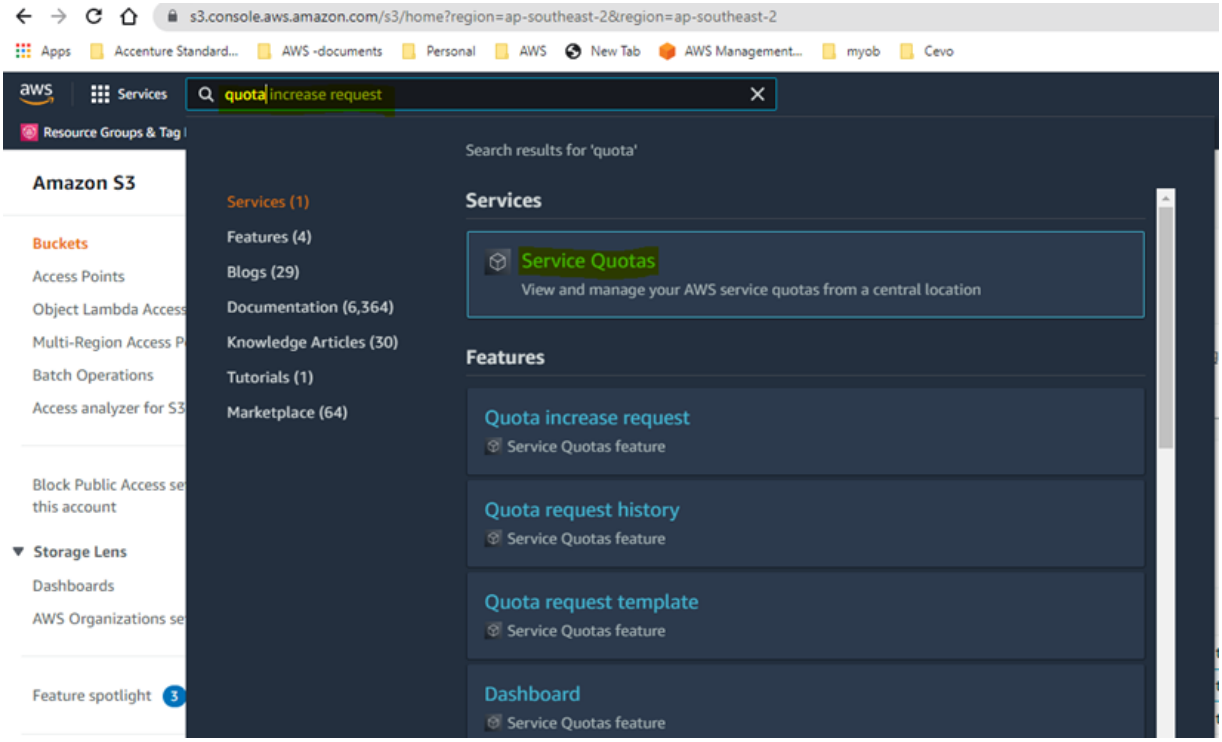


2. Increasing the Service Quota limit

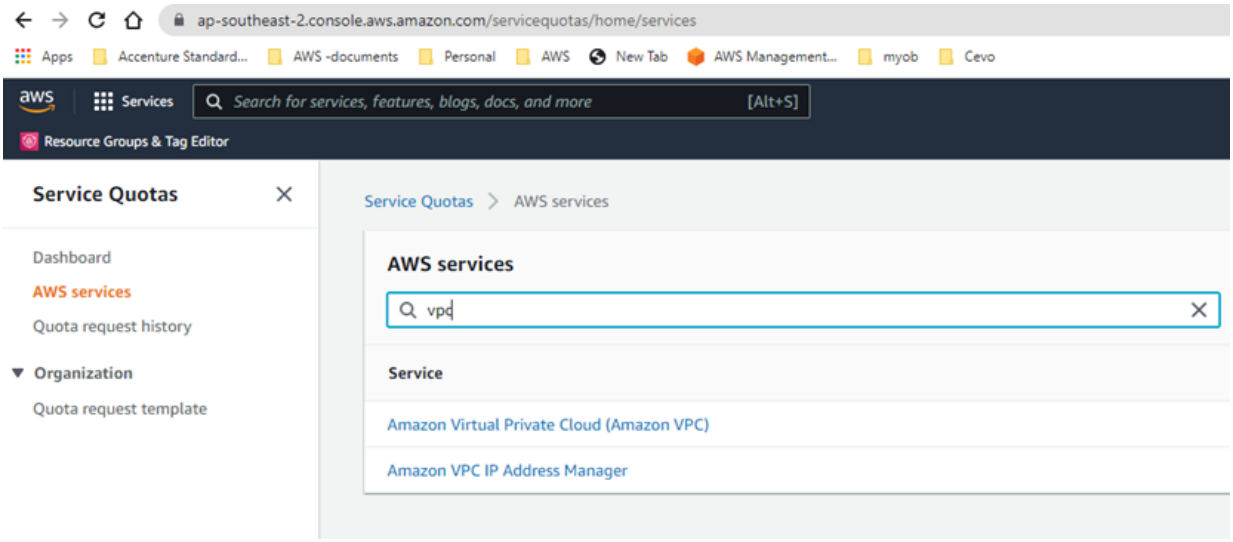
Increasing the Service Quota to accommodate more than 60 entries per prefix-list, that will be attached to security groups.

Reason: By default a security group can accommodate only 60 entries maximum. When attaching a prefix-list to the security group, the prefix list inherits the max entry count from the corresponding security group to which it has been attached. Hence only 60 CIDR entries can be added to the prefix-list, if the default value is unchanged. By increasing the service quota limit of security groups max inbound and outbound entries, the count of CIDRs that can be added to a prefix can be increased. **Limitation:** A prefix list cannot be altered once its created. So it is recommended to alter the max count of entries to its maximum (1000) before deploying the solution. Else the stack has to be redeployed.

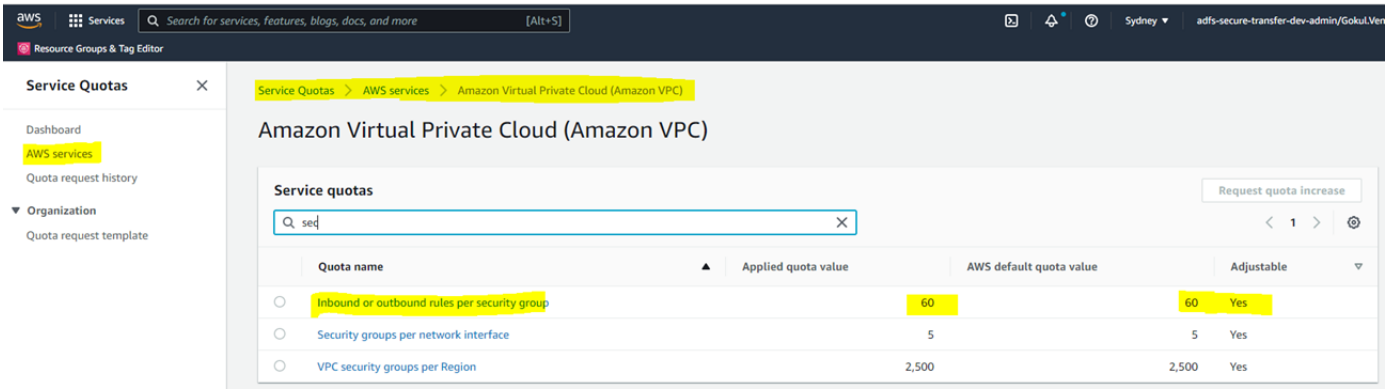
Step 1 : Search for Service Quota in the Search bar in AWS console.



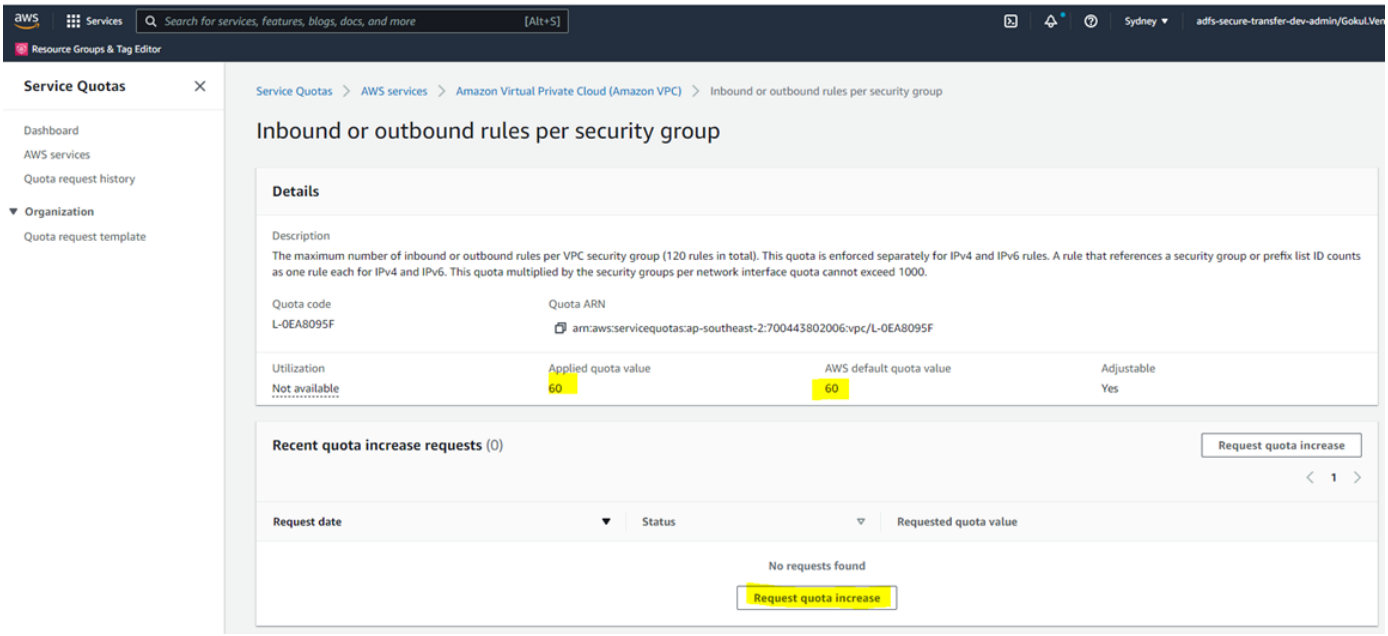
Step 2 : Choose VPC.



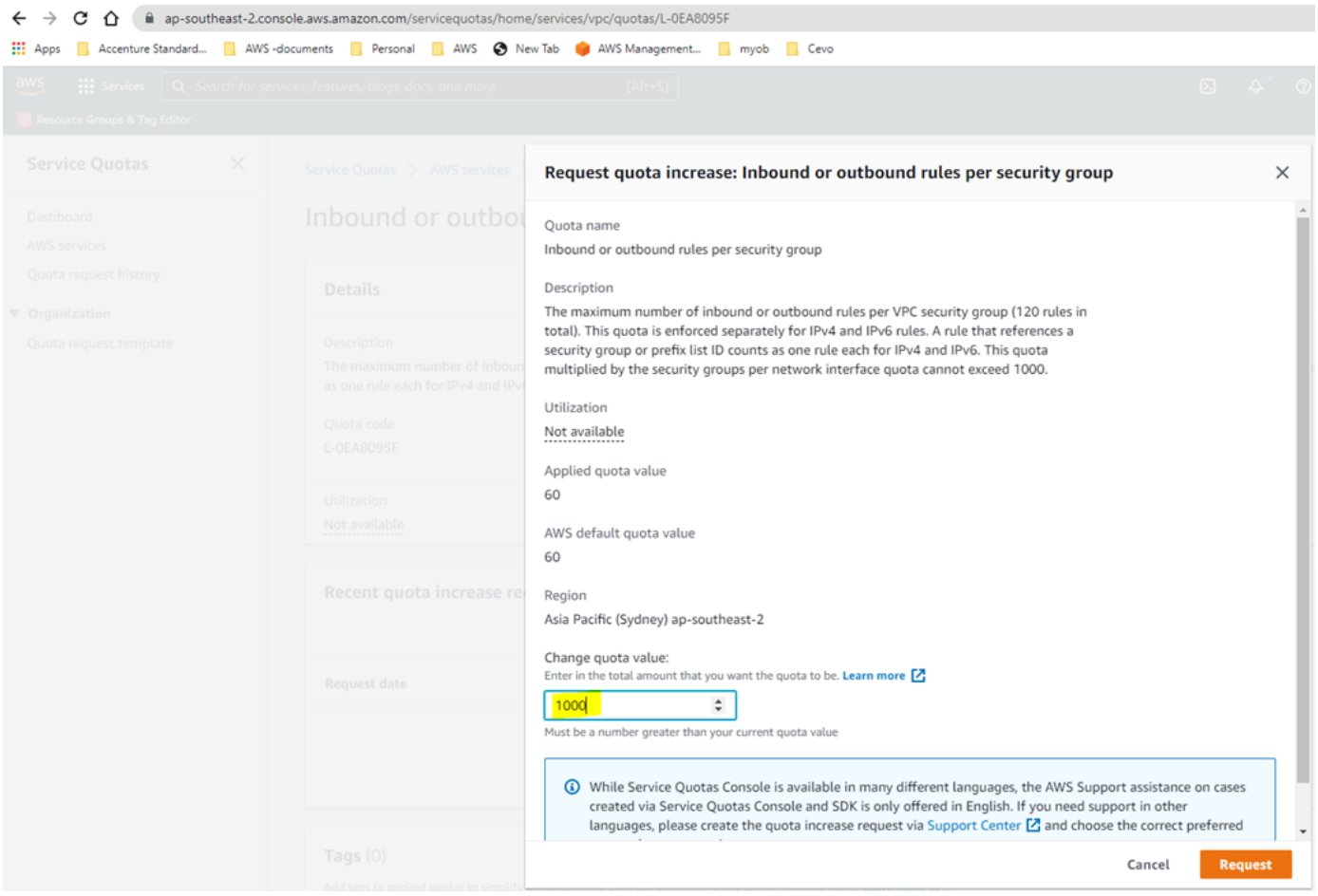
Step 3 : Search for Security Groups under VPC



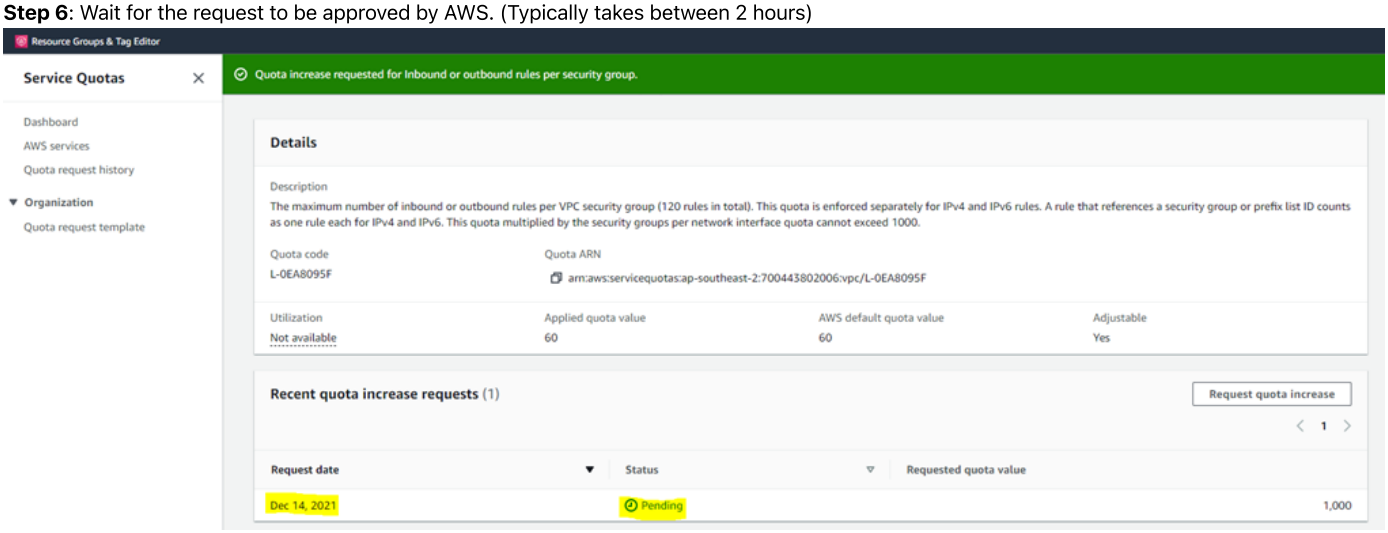
Step 4 : “Request Quota increase” for the Security Group.



Step 5 : Request for the max count (1000)



Step 6: Wait for the request to be approved by AWS. (Typically takes between 2 hours)

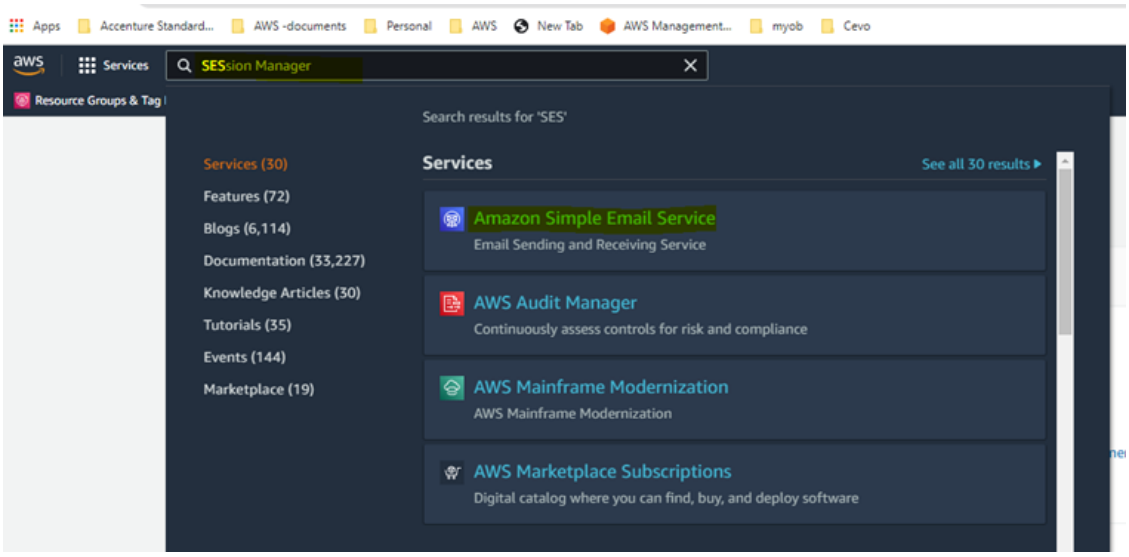


3. Verifying an email address or domain to use it as "From ID" for emails triggered from SES

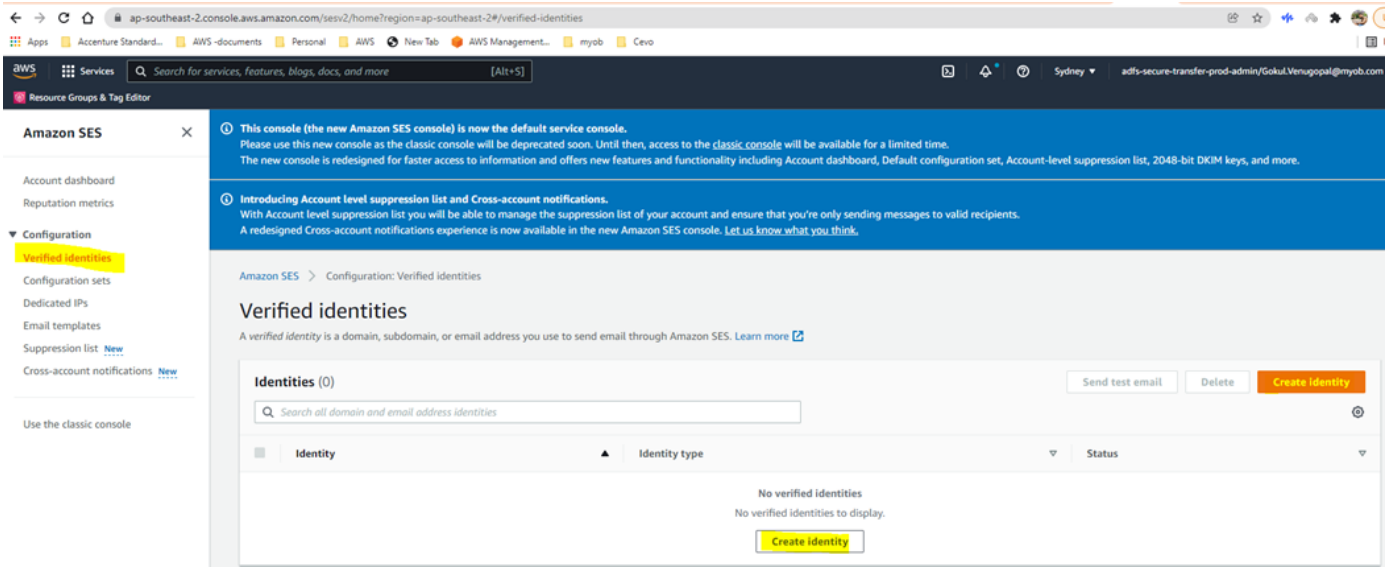
The Source Email has to be a verified email id or domain in SES. In this case, the domain partner.CUSTOMER.com has used the source email and the complete MAIL FROM id is notification@partner.CUSTOMER.com. While requesting domain verification and custom mail from the address, the AWS SES requires the below DKIM CNAME, TXT, and MX records to be added to Route53. Also, refer to the contents from "Source Email-ID for Email notification" for reference.

Below are the steps for verifying identity in Simple Email Service to use it as “FROM Email ID” for all the notifications.

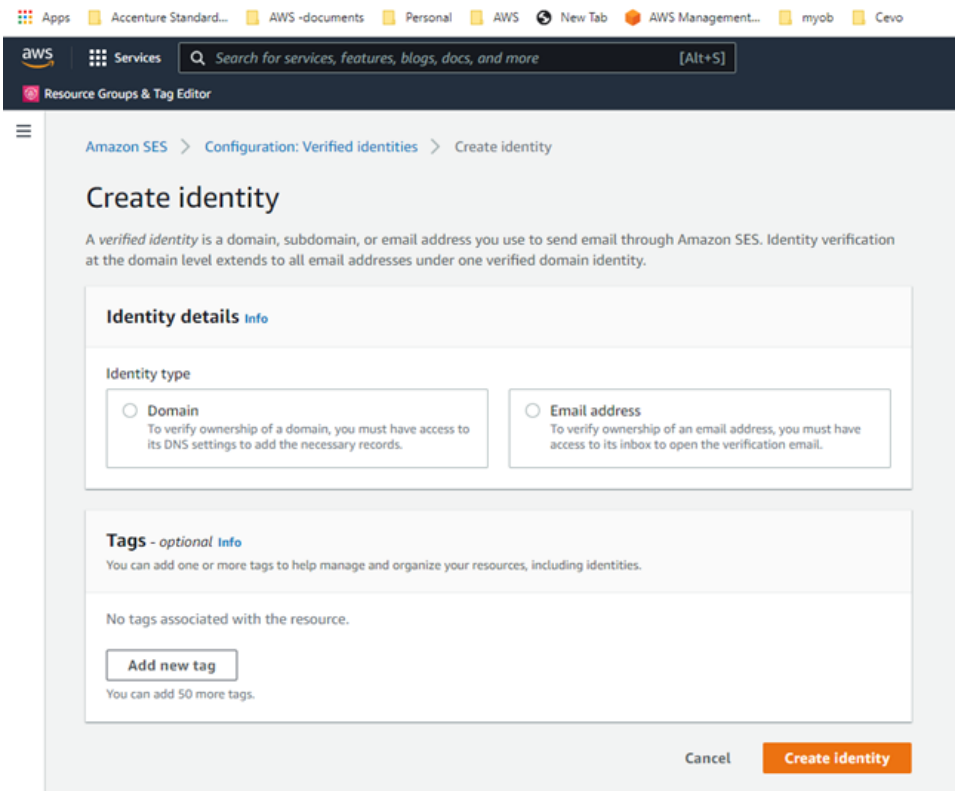
Step 1 : Search for Simple Email Service in AWS Console.



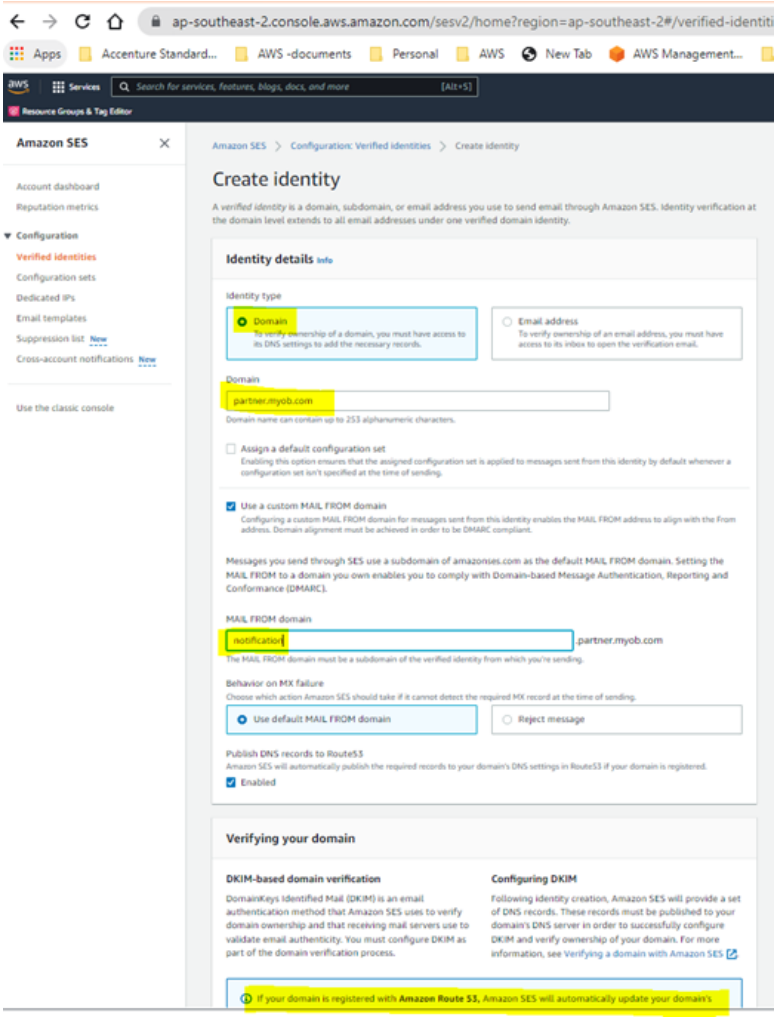
Step 2 : Click on Verified Identities under the SES.



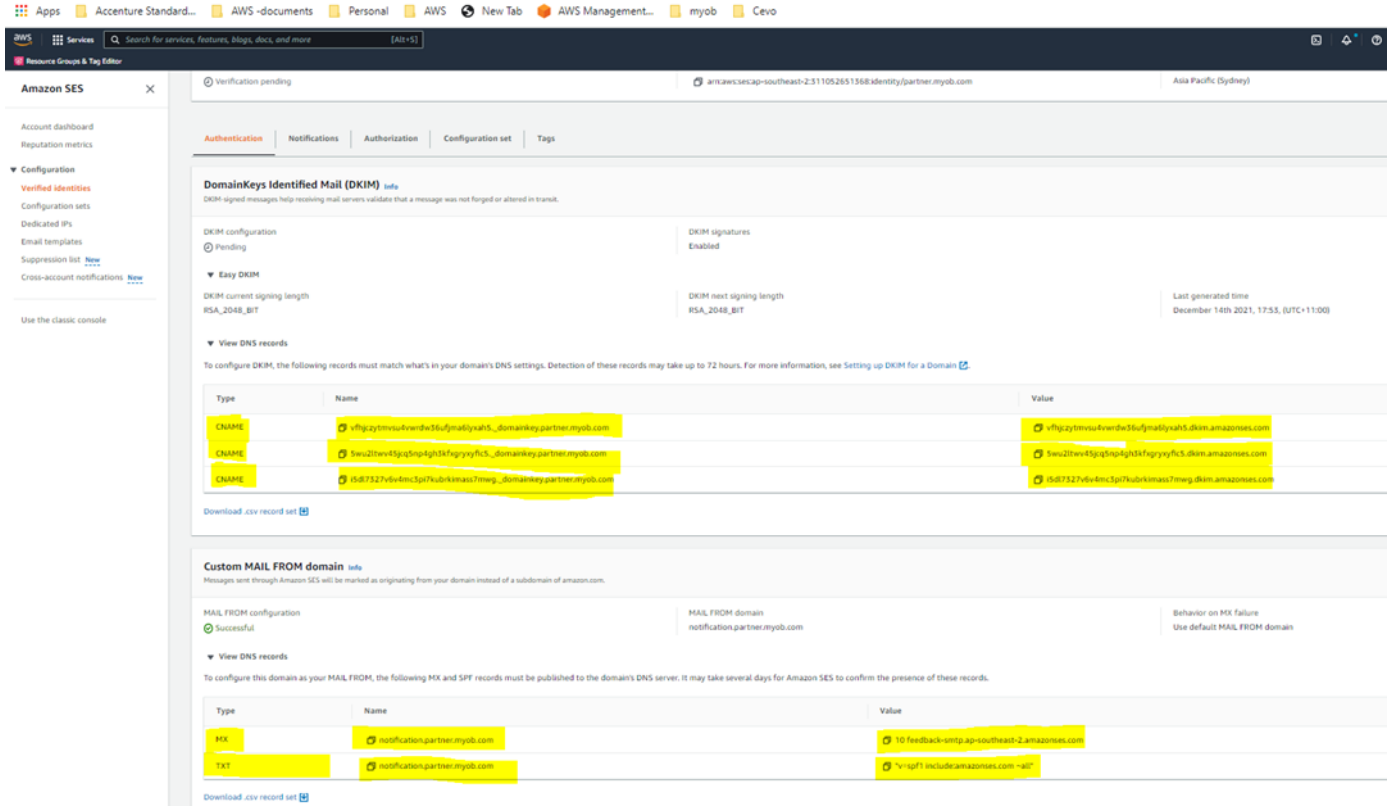
Step 3 : Click on create identity.



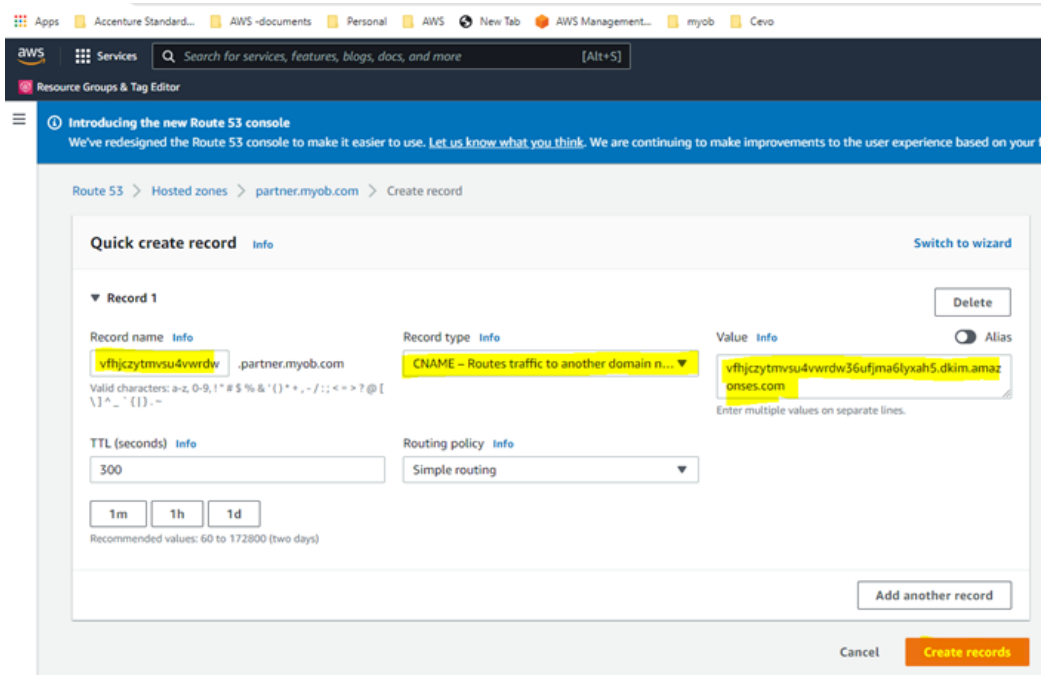
Step 4 : Populate the domain and Mail From Domain to proceed with next steps.



Step 5 : AWS provides a set of name-value entries (CName, TXT, and MX) that need to be added to the Route53 where the domain is maintained.



Step 6 : Update the values in Route53.



4. Make sure the CloudTrail is enabled on the Region