

## VPC- Challenge1

Use Case: Setting up Transit Gateway and VPC Endpoints for a Multi-VPC Architecture

### **Scenario:**

A large organization is migrating its on-premises infrastructure to the AWS cloud. The organization's architecture involves multiple VPCs for different departments and applications, each requiring secure communication with centralized services and external resources. The IT team needs to design and implement a scalable and efficient network architecture to accommodate the organization's growth and ensure robust connectivity between VPCs and external services.

### **Objectives:**

- Design and deploy a scalable network architecture using AWS Transit Gateway to simplify network connectivity between multiple VPCs.
- Configure VPC endpoints to securely access AWS services without internet gateways or NAT gateways, ensuring data privacy and minimizing exposure to external threats.

# VPC- Challenge1

## Objective

- To design a **centralized, scalable, and secure network architecture** that simplifies connectivity between multiple VPCs using **AWS Transit Gateway**, reducing operational complexity and enabling easy future expansion.

## Design Components:

- Transit Gateway
- VPC Attachments
- Transit Gateway Route Tables
- VPC Route Tables

## Deployment Steps

Firstly, create VPC , subnets and Route tables in 3 regions and CIDR should not overlap(Regions:-Jakarta, Malaysia, Melbourne)

The screenshot displays the AWS VPC dashboard with two separate VPC configurations side-by-side.

**VPC-A (Asia Pacific (Jakarta)):**

- Subnets:** ap-southeast-3a, VPC-A-subnet-public1-ap-southeast-3a
- Route tables:** VPC-A-rtb-public, rtb-072f6d1b50bb6c1fb
- Network Connectivity:** VPC-A-igw

**VPC-B (Asia Pacific (Malaysia)):**

- Subnets:** ap-southeast-5a, VPC-B-subnet-private1-ap-southeast-5a
- Route tables:** rtb-04275dffcc12410d81, VPC-B-rtb-private1-ap-southeast-5a

# VPC- Challenge1

The screenshot shows the AWS VPC dashboard with the following details:

- VPC dashboard**: Shows basic VPC settings like IPv6 CIDR (Disabled), Network Address Usage metrics (Disabled), and Route 53 Resolver DNS Firewall rule groups (None).
- Virtual private cloud**: Your VPCs section lists subnets, route tables, internet gateways, egress-only internet gateways, DHCP option sets, elastic IPs, managed prefix lists, endpoints, and endpoint services.
- Resource map**: A diagram showing the VPC structure. It includes:
  - VPC**: Your AWS virtual network (VPC-C-vpc).
  - Subnets (1)**: Subnets within this VPC (ap-southeast-4a). One subnet is highlighted: VPC-C-subnet-private1-ap-southeast-4a.
  - Route tables (2)**: Route network traffic to resources (rtb-0c6c042c955a5ae64 and VPC-C-rtb-private1-ap-southeast-4a).

## Step 1: Create a Transit Gateway

- Enable DNS support if required
- Disable auto-accept (recommended for security)

The screenshot shows the AWS Transit gateway creation page for Melbourne:

- Transit gateways**: Shows one transit gateway named 'TGW-Mel' with ID 'tgw-0e1df611f9755d849' and Owner ID '814588432081'. State is Pending.
- Actions**: Includes a 'Create transit gateway' button.

The screenshot shows the AWS Transit gateway creation page for Jakarta:

- Transit gateways**: Shows one transit gateway named 'TGW-Jakarta' with ID 'tgw-010c0c9e3fbcac2d4' and Owner ID '814588432081'. State is Pending.
- Actions**: Includes a 'Create transit gateway' button.

# VPC- Challenge1

The screenshot shows the AWS VPC Transit gateways console. A blue banner at the top right says "Introducing new feature: Metering Policy for Transit Gateway (TGW)". It explains that transit gateway data usage was metered solely to the source attachment owner, but now you can create custom metering policies to allocate data usage to source, destination, or central TGW accounts. A "Create metering policy" button is visible. Below the banner, the "Transit gateways (1/1)" section is shown with a table. One row is selected for "TGW-Mal" with ID "tgw-0b89b9dd9a5b36ae4". The "State" column shows "Available". At the bottom of the page, there are links for "Feedback", "Console Mobile App", and copyright information.

## Step 2: Create VPC Attachments

- Attach each VPC to the TGW
- Select private subnets only
- One attachment per VPC

The screenshot shows the AWS VPC Transit gateway attachments console. A green success message at the top says "You successfully created VPC attachment tgw-attach-0a66a523c0d00a776 / TG\_attach\_Mal." Below it, the "Transit gateway attachments (1)" section is shown with a table. One row is selected for "TG\_attach\_Mal" with ID "tgw-attach-0a66a523c0d00a776". The "State" column shows "Pending". The "Resource Type" column shows "VPC" and the "Region" column shows "vpc-f". At the bottom of the page, there are links for "Feedback", "Console Mobile App", and copyright information.

The screenshot shows the AWS VPC Transit gateway attachments console in the Asia Pacific (Melbourne) region. A green success message at the top says "You successfully created VPC attachment tgw-attach-0c55c0ad1cd0939c0 / TG\_attach\_Mel." Below it, the "Transit gateway attachments (1)" section is shown with a table. One row is selected for "TG\_attach\_Mel" with ID "tgw-attach-0c55c0ad1cd0939c0". The "State" column shows "Pending". The "Resource Type" column shows "VPC" and the "Region" column shows "vpc-f". At the bottom of the page, there are links for "CloudShell", "Feedback", "Console Mobile App", and copyright information.

# VPC- Challenge1

You successfully created VPC attachment tgw-attach-0ac9742ec291bc2cf / TG\_attach\_jakarta.

Name	Transit gateway attachment ID	Transit gateway ID	State	Resource Type	Region
TG_attach_jakarta	tgw-attach-0ac9742ec291bc2cf	tgw-010c0c9e3fbcac2d4	Pending	VPC	vpc-1

## Step 3: Configure Transit Gateway Route Tables

Transit gateway route tables (1/1)

Name	Transit gateway route table ID	Transit gateway ID	State	Default association route table
TGW_RT_J	tgw-rtb-05029982b5e6b46b8	tgw-010c0c9e3fbcac2d4	Available	Yes

Transit gateway route tables: tgw-rtb-05029982b5e6b46b8 / TGW\_RT\_J

Details			
Transit gateway route table ID	State	Default association route table	Default propagation route table
tgw-rtb-05029982b5e6b46b8	Available	Yes	Yes
Transit gateway ID			

Transit gateway route tables (1/1)

Name	Transit gateway route table ID	Transit gateway ID	State	Default association route table
TGW_RT_Mal	tgw-rtb-018a5fd3c17c82f53	tgw-0b89b9dd9a5b3ae4	Available	Yes

Transit gateway route tables: tgw-rtb-018a5fd3c17c82f53 / TGW\_RT\_Mal

Details			
Transit gateway route table ID	State	Default association route table	Default propagation route table
tgw-rtb-018a5fd3c17c82f53	Available	Yes	Yes

# VPC- Challenge1

The screenshot shows the AWS VPC console with the 'Transit gateway route tables' page. On the left, there's a navigation sidebar for 'Virtual private cloud' with various options like 'Your VPCs', 'Subnets', 'Route tables', etc. The main content area has a title 'Transit gateway route tables (1/1)'. A table lists one entry: 'TG\_RT\_melb' with 'tgw-rtb-09ef84589ad21b5bf' as its ID. The table includes columns for Name, Transit gateway route table ID, Transit gateway ID, State, and Default association route table. Below the table, a detailed view for 'Transit gateway route tables: tgw-rtb-09ef84589ad21b5bf / TG\_RT\_melb' is shown with tabs for Details, Associations, Propagations, Prefix list references, Routes, and Tags. The 'Details' tab is selected, showing the transit gateway route table ID, state (Available), and default association route table (Yes).

## Step 4: Update VPC Route Tables

Add routes in private subnet route tables:

Destination: 10.0.1.0/16 (Remote VPC)

Target: Transit Gateway

The screenshot shows the AWS VPC console with the 'Transit gateway route tables' page. The left sidebar shows 'AWS Verified Access' and 'Transit gateways' sections. The main content area shows a table for 'Transit gateway route tables (1/1)' with one entry: 'TGW\_RT\_J' with 'tgw-rtb-05029982b5e6b46b8' as its ID. The table includes columns for Name, Transit gateway route table ID, Transit gateway ID, State, and Default association route table. Below the table, a detailed view for 'Transit gateway route tables: tgw-rtb-05029982b5e6b46b8 / TGW\_RT\_J' is shown with a table of routes. The table has columns for CIDR, Attachment ID, Resource ID, Resource type, Route type, and Route status. It contains two entries: a propagated route for 10.0.0.0/16 via attachment 'tgw-attach-0ac9742ec291bc2cf' and a static route for 11.0.0.0/16 via attachment 'tgw-attach-001e1dd7d5d059b85'.

# VPC- Challenge1

The screenshot shows the AWS VPC console in the Asia Pacific (Melbourne) region. The left sidebar is expanded to show the 'Virtual private cloud' section. In the main content area, the 'Transit gateway route tables' page is displayed. A single transit gateway route table named 'TG\_RT\_melb' is listed. It has a Transit gateway route table ID of 'tgw-rtb-09ef84589ad21b5bf' and a Transit gateway ID of 'tgw-0e1df611f9755d849'. The table is in an 'Available' state and has 'Yes' as its default association route. Below the table, the routes section shows two static routes: one for CIDR 11.0.0.0/16 pointing to attachment 'tgw-attach-070f05ec06353deb9' (Peering, Static, Active) and another for CIDR 12.0.0.0/16 pointing to attachment 'tgw-attach-0c55c0ad1cd0939c0' (VPC, Propagated, Active).

The screenshot shows the AWS VPC console in the Asia Pacific (Malaysia) region. The left sidebar is expanded to show the 'Virtual private cloud' section. In the main content area, the 'Transit gateway route tables' page is displayed. A single transit gateway route table named 'TGW\_RT\_Mal' is listed. It has a Transit gateway route table ID of 'tgw-rtb-018a5fd3c17c82f53' and a Transit gateway ID of 'tgw-0b89b9dd9a5b36ae4'. The table is in an 'Available' state and has 'Yes' as its default association route. Below the table, the routes section shows three static routes: one for CIDR 10.0.0.0/16 pointing to attachment 'tgw-attach-001e1dd7d5d059b85' (Peering, Static, Active), one for CIDR 11.0.0.0/16 pointing to attachment 'tgw-attach-0a66a523c0d00a776' (VPC, Propagated, Active), and one for CIDR 12.0.0.0/16 pointing to attachment 'tgw-attach-070f05ec06353deb9' (Peering, Static, Active).

## Step 5: Security Configuration

- Use **Security Groups** and **NACLs**(follow these rules for all 3 regions)
- Restrict TGW routes to required CIDRs

The screenshot shows the AWS VPC console in the Asia Pacific (Malaysia) region, specifically the 'Edit inbound rules' page for a security group named 'sg-0c955e9bf02a2a3bd'. The page lists three existing inbound rules:

- A rule for SSH (Protocol TCP, Port range 22, Source 0.0.0.0/0) with a description 'Delete'.
- A rule for All ICMP - IPv4 (Protocol ICMP, Port range All, Source 0.0.0.0/0) with a description 'Delete'.
- A rule for All traffic (Protocol All, Port range All, Source sg-0c955e9bf02a2a3bd) with a description 'Delete'.

A 'Delete' button is also present at the bottom right of the page.

# VPC- Challenge1

The screenshot shows the AWS VPC Network ACL inbound rules configuration. It displays two rules:

Rule number	Type	Protocol	Port range	Source	Action
100	All traffic	All	All	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Buttons at the bottom include 'Add new rule', 'Sort by rule number', 'Cancel', 'Preview changes', and 'Save changes'.

## Step 6: High Availability

- TGW is regionally highly available
- Use multiple subnets (AZs) per attachment
- For multi-region:
  - Use TGW Peering

The screenshot shows the AWS Transit gateway attachments page. It lists two attachments:

Name	Transit gateway attachment ID	Transit gateway ID	State	Resource Type
TGA_peerJakarta	tgw-attach-001e1dd7d5d059b85	tgw-010c0c9e3fbcac2d4	Available	Peering
TG_attach_jakarta	tgw-attach-0ac9742ec291bc2f	tgw-010c0c9e3fbcac2d4	Available	VPC

A modal window titled 'Select a transit gateway attachment' is open, showing the same list of attachments.

# VPC- Challenge1

The screenshot shows the AWS VPC console with the 'Transit gateway attachments' section selected. The left sidebar includes options like Site-to-Site VPN, Concentrators, Site-to-Site VPN connections, Client VPN endpoints, Transit gateways, Traffic Mirroring, and Feedback. The main area displays a table of transit gateway attachments:

Name	Transit gateway attachment ID	Transit gateway ID	State	Resource Type	Region
Peer_jak-mal	tgw-attach-001e1dd7d5d059b85	tgw-0b89b9dd9a5b36ae4	Available	Peering	tgw-l
TGA-peerMalaysia	tgw-attach-070f05ec06353deb9	tgw-0b89b9dd9a5b36ae4	Available	Peering	tgw-l
TG_attach_Mal	tgw-attach-0a66a523c0d00a776	tgw-0b89b9dd9a5b36ae4	Available	VPC	vpc-f

Below the table, a section titled 'Select a transit gateway attachment' is visible.

This screenshot shows the same 'Transit gateway attachments' section in the AWS VPC console, but for the region Asia Pacific (Melbourne). The table data is identical to the previous screenshot:

Name	Transit gateway attachment ID	Transit gateway ID	State	Resource Type	Region
TGA_peerMel	tgw-attach-070f05ec06353deb9	tgw-0e1df611f9755d849	Available	Peering	tgw-l
TG_attach_Mel	tgw-attach-0c55c0ad1cd0939c0	tgw-0e1df611f9755d849	Available	VPC	vpc-f

## Create EC2 instance for checking connectivity.

The screenshot shows the AWS EC2 console with the 'Instances' section selected. The left sidebar includes options like Dashboard, AWS Global View, Events, Instances, Images, and CloudWatch Metrics. The main area displays a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
EC2-jakarta	i-07caf15af3726e4e2	Running	t3.micro	Initializing	View alarms +	ap-southeast-1

Below the table, a detailed view of the instance 'i-07caf15af3726e4e2 (EC2-jakarta)' is shown, with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under Details, it shows the Public IPv4 address (16.78.77.248) and Private IPv4 addresses (10.0.5.169).

# VPC- Challenge1

The screenshot shows the AWS EC2 Instances page. At the top, there is a green success message: "Successfully initiated starting of i-075b8faa5920c9b24". Below this, the "Instances (1/1)" section is displayed. The instance details are as follows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
EC2-Malaysia	i-075b8faa5920c9b24	Running	t3.micro	Initializing	View alarms +	ap-southeast-1

The instance summary table shows:

Instance ID	Public IPv4 address	Private IPv4 addresses
i-075b8faa5920c9b24	-	11.0.139.60

At the bottom of the page, there are links for "Privacy", "Terms", and "Cookie preferences".

The screenshot shows the AWS EC2 Instances page. At the top, there is a green success message: "Successfully initiated starting of i-07dda7623e82107e7". Below this, the "Instances (1/1)" section is displayed. The instance details are as follows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
EC2-Melb	i-07dda7623e82107e7	Running	t3.micro	Initializing	View alarms +	ap-southeast-1

The instance summary table shows:

Instance ID	Public IPv4 address	Private IPv4 addresses
i-07dda7623e82107e7	-	12.0.132.250

At the bottom of the page, there are links for "Privacy", "Terms", and "Cookie preferences".

# VPC- Challenge1

## Connection: To Verify

Ec2 instance for Jakarta region

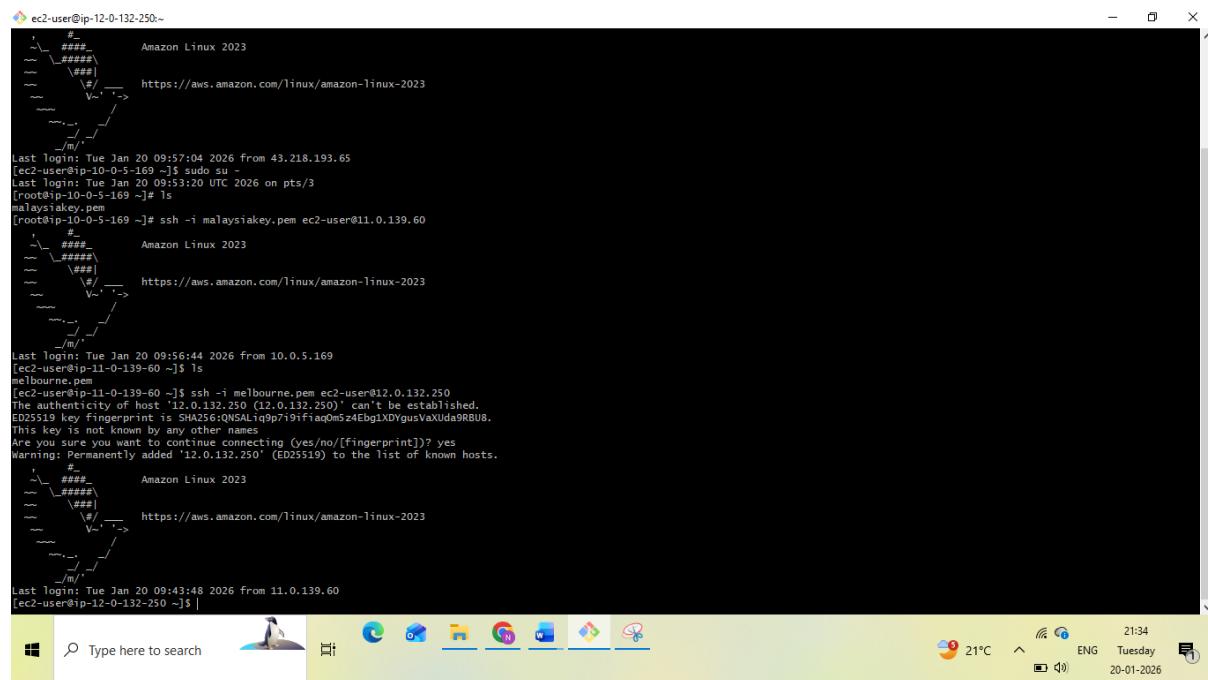
```
ec2-user@ip-10-0-5-169:~  
user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)  
$ ssh -i "Jakarta_keypair.pem" ec2-user@ec2-16-78-77-248.ap-southeast-3.compute.amazonaws.com  
The authenticity of host 'ec2-16-78-77-248.ap-southeast-3.compute.amazonaws.com (16.78.77.248)' can't be established.  
ED25519 key fingerprint is SHA256:P5EhVyr1bNcgE/VUFRNHXOorqB4s5yt5mRPX119yr+Y.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'ec2-16-78-77-248.ap-southeast-3.compute.amazonaws.com' (ED25519) to the list of known hosts.  
, #  
~\_\_ ##### Amazon Linux 2023  
~~ \_\#\#\#\\  
~~ \#\#\|  
~~ \#/ __ https://aws.amazon.com/linux/amazon-linux-2023  
~~ V~, '-->  
~~ /  
~~ .--/  
~~ /--/  
~~ /m/  
Last login: Tue Jan 20 09:57:04 2026 from 43.218.193.65  
[ec2-user@ip-10-0-5-169 ~]$ |
```

```
[ec2-user@ip-10-0-5-169 ~]$ sudo su -  
Last login: Tue Jan 20 09:53:20 UTC 2026 on pts/3  
[root@ip-10-0-5-169 ~]# ls  
malaysiakey.pem  
[root@ip-10-0-5-169 ~]# ssh -i malaysiakey.pem ec2-user@11.0.139.60  
, #  
~\_\_ ##### Amazon Linux 2023  
~~ \_\#\#\#\\  
~~ \#\#\|  
~~ \#/ __ https://aws.amazon.com/linux/amazon-linux-2023  
~~ V~, '-->  
~~ /  
~~ .--/  
~~ /--/  
~~ /m/  
Last login: Tue Jan 20 09:56:44 2026 from 10.0.5.169  
[ec2-user@ip-11-0-139-60 ~]$ |
```

# VPC- Challenge1

```
[ec2-user@ip-11-0-139-60 ~]$ ls
melbourne.pem
[ec2-user@ip-11-0-139-60 ~]$ ssh -i melbourne.pem ec2-user@12.0.132.250
The authenticity of host '12.0.132.250 (12.0.132.250)' can't be established.
ED25519 key fingerprint is SHA256:QNSALiq9p7i9ifiaq0m5z4Ebg1XDYgusVaXUda9RBU8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '12.0.132.250' (ED25519) to the list of known hosts.

      #_
      ~\_ #####_          Amazon Linux 2023
      ~~ \_\#\#\#\`_
      ~~ \#\#\#|_
      ~~ \#/`___.-->
      ~~ V~`_____
      ~~ .-.`_/
      ~~ /`_/
      ~~ /m/`_
Last login: Tue Jan 20 09:43:48 2026 from 11.0.139.60
[ec2-user@ip-12-0-132-250 ~]$ |
```



```
ec2-user@ip-12-0-132-250:~$ ls
      #_
      ~\_ #####_          Amazon Linux 2023
      ~~ \_\#\#\#\`_
      ~~ \#\#\#|_
      ~~ \#/`___.-->
      ~~ V~`_____
      ~~ .-.`_/
      ~~ /`_/
      ~~ /m/`_
Last Login: Tue Jan 20 09:57:04 2026 from 43.218.193.65
[ec2-user@ip-10-0-5-169 ~]$ sudo su -
Last login: Tue Jan 20 09:53:20 UTC 2026 on pts/3
[root@ip-10-0-5-169 ~]$ ls
malaysiakey.pem
[root@ip-10-0-5-169 ~]$ ssh -i malaysiakey.pem ec2-user@11.0.139.60
      #_
      ~\_ #####_          Amazon Linux 2023
      ~~ \_\#\#\#\`_
      ~~ \#\#\#|_
      ~~ \#/`___.-->
      ~~ V~`_____
      ~~ .-.`_/
      ~~ /`_/
      ~~ /m/`_
Last Login: Tue Jan 20 09:56:44 2026 from 10.0.5.169
[ec2-user@ip-11-0-139-60 ~]$ ls
melbourne.pem
[ec2-user@ip-11-0-139-60 ~]$ ssh -i melbourne.pem ec2-user@12.0.132.250
The authenticity of host '12.0.132.250 (12.0.132.250)' can't be established.
ED25519 key fingerprint is SHA256:QNSALiq9p7i9ifiaq0m5z4Ebg1XDYgusVaXUda9RBU8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '12.0.132.250' (ED25519) to the list of known hosts.

      #_
      ~\_ #####_          Amazon Linux 2023
      ~~ \_\#\#\#\`_
      ~~ \#\#\#|_
      ~~ \#/`___.-->
      ~~ V~`_____
      ~~ .-.`_/
      ~~ /`_/
      ~~ /m/`_
Last Login: Tue Jan 20 09:43:48 2026 from 11.0.139.60
[ec2-user@ip-12-0-132-250 ~]$ |
```

## Conclusion

- By using **AWS Transit Gateway**, we can design a **scalable, secure, and centralized network architecture** that simplifies VPC connectivity, reduces operational overhead, and supports future growth without re-architecting the network.

# VPC- Challenge1

- Configure VPC endpoints to securely access AWS services without internet gateways or NAT gateways, ensuring data privacy and minimizing exposure to external threats.

## Objective:

- Implemented **VPC Gateway and Interface Endpoints (PrivateLink)** to enable secure, private access to AWS services without Internet or NAT Gateways.
- Ensured **data privacy and reduced attack surface** by keeping all service traffic on the AWS private backbone using Private DNS and endpoint policies.
- Optimized **security and cost** by eliminating public IP dependencies and enforcing least-privilege access controls.

```
MINGW64:/c/Users/user/Downloads
user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ aws --version
aws-cli/2.32.26 Python/3.13.11 Windows/10 exe/AMD64

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ aws configure
AWS Access Key ID [*****MTIY]: AKIA33KKBB3I6K456467
AWS Secret Access Key [*****FtU5]: 5wTufgg9XNqt6cEUFN4PDdaITFXP8QbysW
Yey1PQ
Default region name [ap-south-1]: us-west-2
Default output format [json]: json

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ neelimaranis3
bash: neelimaranis3: command not found

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ aws sts get-caller-identity

{
    "UserId": "AIDA33KKBB3IWEUYMZHID",
    "Account": "814588432081",
    "Arn": "arn:aws:iam::814588432081:user/Neelima_DevOps"
}

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ |

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ |
```

```
user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ vi Hello

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ aws s3 cp Hello s3://neelimaranis3/
upload: .\Hello to s3://neelimaranis3/Hello

user@DESKTOP-3KH1IRE MINGW64 ~/Downloads (master)
$ |
```