😊 AWS MIGRATION

AWS (Amazon Web Services) migration refers to the process of moving an organization's applications, data, and other business elements to the cloud platform provided by AWS. This migration can involve moving from on-premises data centres, other cloud providers, or a hybrid environment to AWS. A successful migration to AWS can bring various benefits, including scalability, cost-efficiency, agility, and improved security.

Here is a general guide to help you understand the key steps involved in an AWS migration:

1. **Assessment and Planning:**

   a) **Assess Your Current Environment:** Understand your existing infrastructure, applications, and data dependencies. Identify any dependencies and potential challenges.
   b) **Define Objectives:** Clearly define the goals of your migration, such as cost reduction, improved performance, scalability, or modernization.
   c) **Create a Migration Plan:** Develop a detailed plan that outlines tasks, timelines, and resources needed for each phase of the migration.

2. **Designing the AWS Architecture:**

   a) **Select AWS Services:** Choose the AWS services that align with your requirements. AWS provides a wide range of services for computing, storage, databases, networking, and more.
   b) **Architect for Scalability and Resilience:** Design your architecture to take advantage of AWS scalability and ensure high availability and fault tolerance.
   c) **Consider Security:** Implement security best practices, such as encryption, identity and access management (IAM), and network security.

3. **Data Migration:**

   a) **Transfer Data to AWS:** Move your data to AWS using various migration methods, such as AWS Snowball, AWS DataSync, or traditional data transfer methods.
   b) **Database Migration:** If you're migrating databases, use services like AWS Database Migration Service (DMS) for a seamless transition.

4. **Application Migration:**

   a) **Rehost, Refactor, or Rebuild:** Decide whether to lift and shift (rehost), refactor, or rebuild your applications based on your objectives and requirements.
   b) **Use Containers and Serverless:** Consider containerization (using services like Amazon ECS or EKS) or serverless computing (AWS Lambda) for modernization.

5. **Testing:**

   a) **Perform Testing:** Conduct thorough testing of applications and infrastructure in the AWS environment to identify and resolve any issues.

b) **Implement a Rollback Plan:** Have a rollback plan in case any issues arise during or after migration.

6. **Optimization:**

a) **Cost Optimization:** Optimize your AWS resources to ensure cost efficiency. Use services like AWS Cost Explorer to monitor and manage costs.
b) **Performance Optimization:** Fine-tune configurations for better performance based on actual usage patterns.

7. **Training and Documentation:**

a) **Training:** Ensure that your team is trained on AWS services and best practices.
b) **Documentation:** Document the new AWS environment, including configurations, security policies, and procedures.

8. **Monitoring and Management:**

1. **Implement Monitoring:** Set up monitoring tools (e.g., AWS CloudWatch) to track the performance, health, and security of your AWS resources.
2. **Incident Response Plan:** Develop an incident response plan to address and resolve issues promptly.

9. **Go-Live and Post-Migration Support:**

a) **Go-Live:** Execute the migration plan and monitor the performance immediately after the migration.
b) **Provide Post-Migration Support:** Address any issues that may arise and provide support to end-users as needed.

😄 AWS Application Migration Service

AWS Application Migration Service (MGN) is a highly automated lift-and-shift (rehost) solution that simplifies, expedites, and reduces the cost of migrating applications to AWS. It allows companies to lift-and-shift a large number of physical, virtual, or cloud servers without compatibility issues, performance disruption, or long cutover windows. MGN replicates source servers into your AWS account. When you're ready, it automatically converts and launches your servers on AWS so you can quickly benefit from the cost savings, productivity, resilience, and agility of the Cloud. Once your applications are running on AWS, you can leverage AWS services and capabilities to quickly and easily replatform or refactor those applications – which makes lift-and-shift a fast route to modernization.

Here are some AWS services commonly used for application migration:

1. **AWS Server Migration Service (SMS):** AWS SMS is used for migrating on-premises virtualized servers to AWS. It supports the migration of VMware and Microsoft Hyper-V virtual machines. Server replication, automated incremental updates, server grouping, and multi-server application migration.

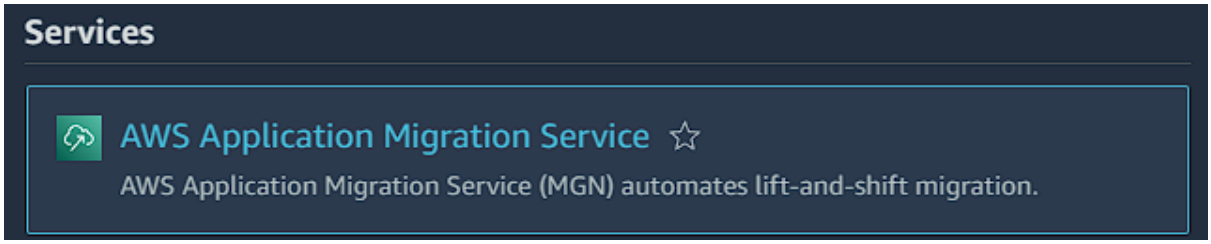
2. **AWS Database Migration Service (DMS):** DMS is specifically designed for database migrations. It supports the migration of various database engines to and from AWS. Schema conversion, ongoing replication, and minimal downtime during migration.
3. **AWS DataSync:** AWS DataSync is a service for fast and secure data transfer between on-premises and AWS. Supports one-time or recurring transfers, provides network optimization, and is suitable for large-scale data migrations.
4. **AWS Data Migration Hub:** This is a centralized location to monitor the progress of application migrations across multiple AWS services. Visibility into the status of migrations, tracking of resource changes, and troubleshooting capabilities.
5. **AWS Migration Hub:** Migration Hub provides a single location to track the progress of application migrations across multiple AWS and partner solutions. Centralized migration tracking, progress reporting, and integration with various migration tools.
6. **AWS Serverless Application Model (SAM):** SAM is an open-source framework for building serverless applications. It simplifies the process of defining serverless applications on AWS. Simplified syntax for defining serverless resources, local testing and debugging, and easy deployment.
7. **AWS CloudEndure Migration:** CloudEndure Migration is an agent-based service that facilitates the migration of physical, virtual, and cloud-based servers to AWS. Continuous data replication, automated machine conversion, and minimal cutover time.

😁 TO BEGIN WITH THE LAB:

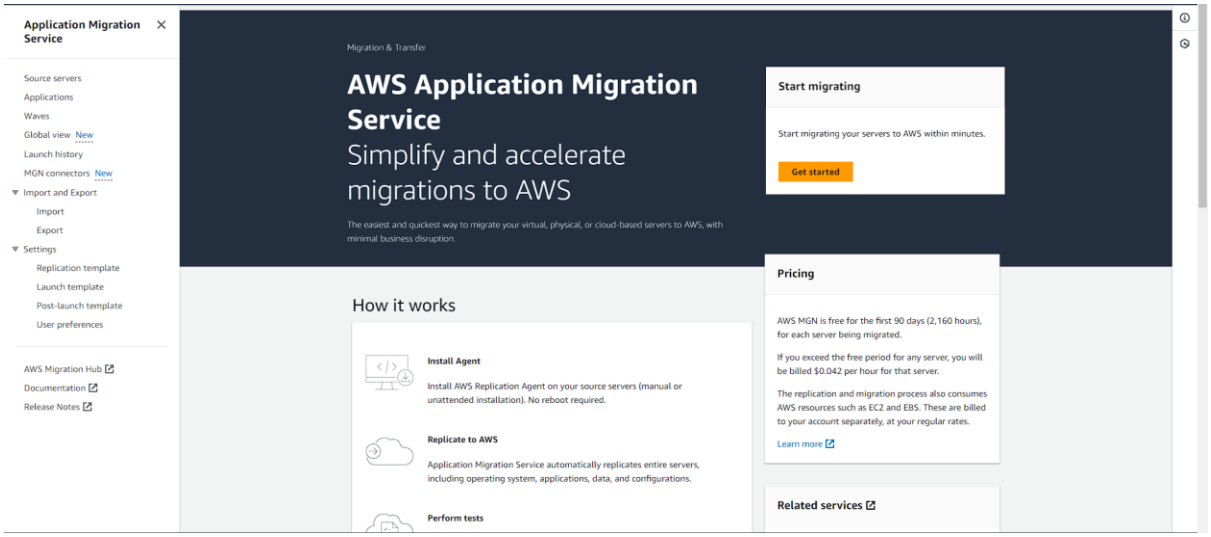
Basically, we need to decide source and destination for data migration.

😎 STEP 1: Create MGN in the Target Region

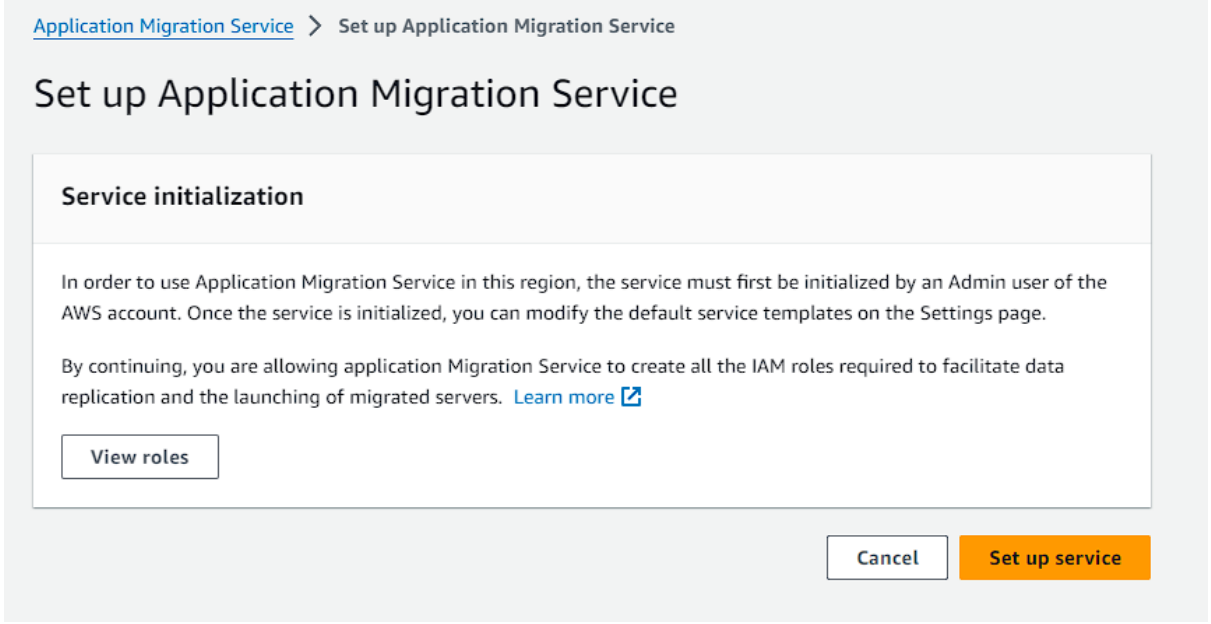
1. Log in to AWS Console. Then search MGN in the search box and choose this service shown below i.e. AWS Application Migration Service (MGN)

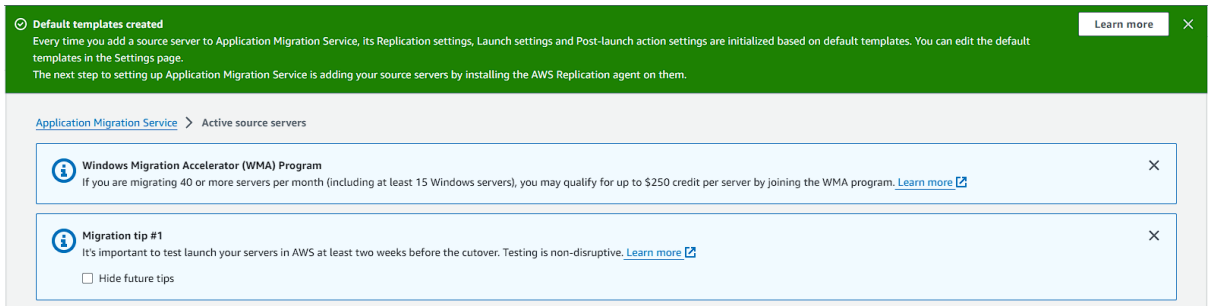


2. To you the console would look like this.

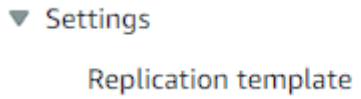


3. Here now you have to click on Get Started to move forward. From here you will start your service.
4. So, once you have clicked on Set up Service it will say that; default templates are created.
5. Now you have to add servers here.



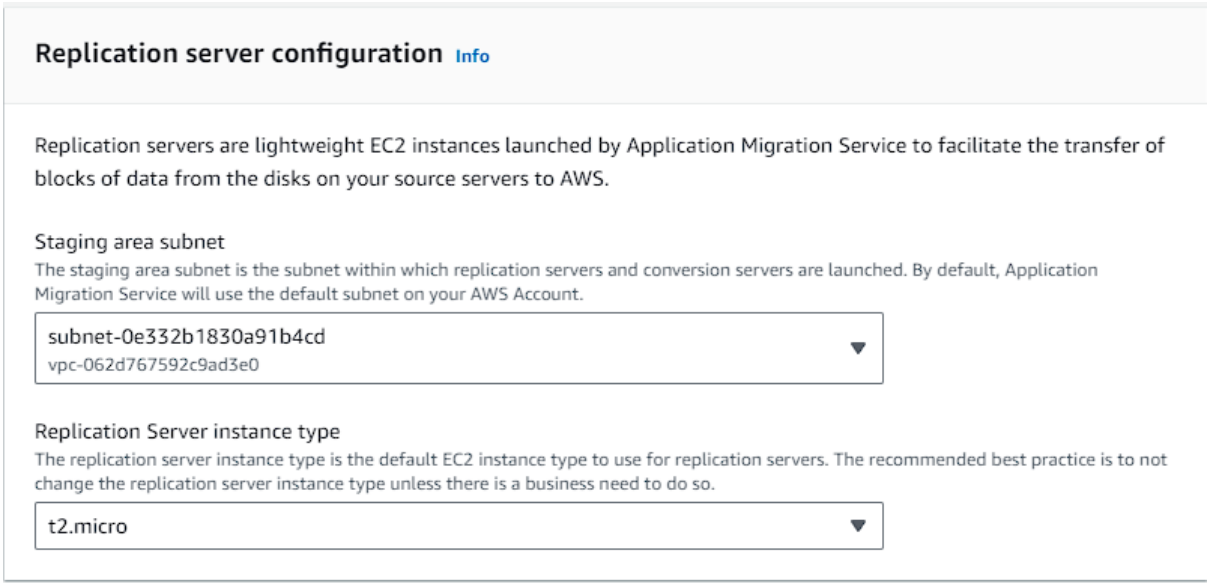


6. After your default templates are created now, click on the hamburger icon and from the left under settings you will find an option as replication template.

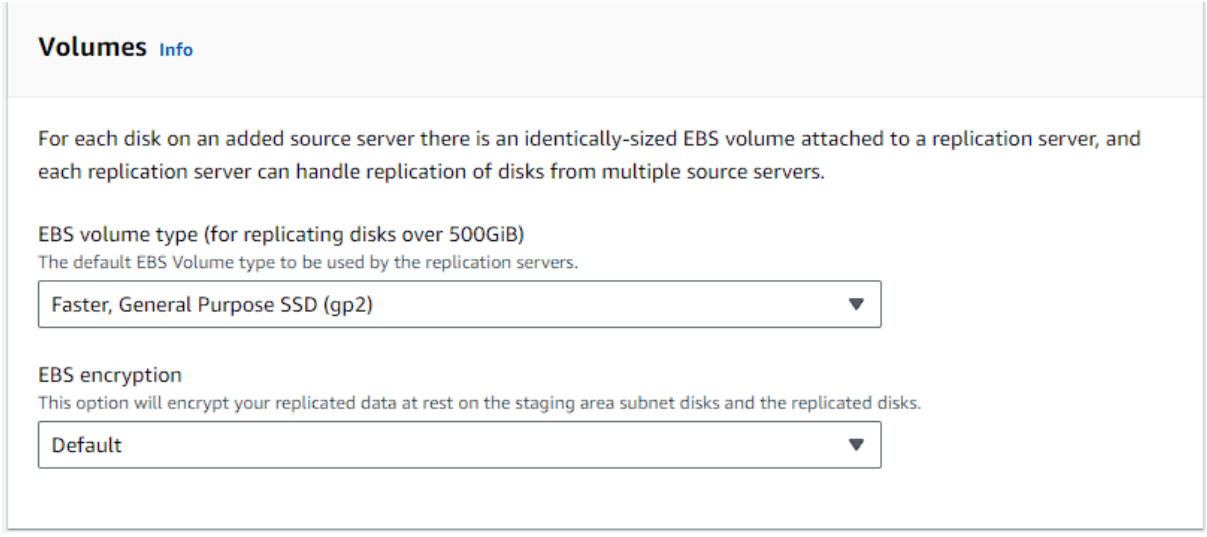


7. Click on it and there you will see that a default template has been created for you by the service itself.
8. But for now, you have to click on edit to change some settings.

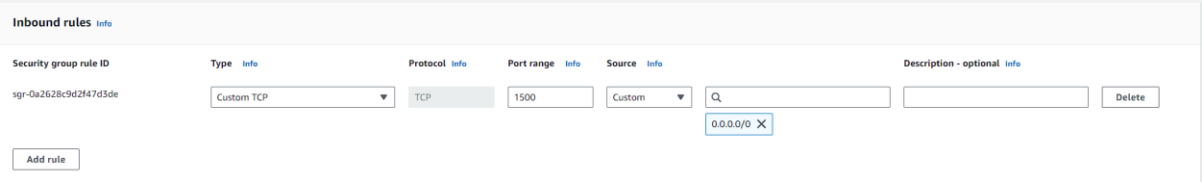
9. Now you need select a subnet of your choice, then select replication server instance type to t2.micro (because it is eligible for free tier)



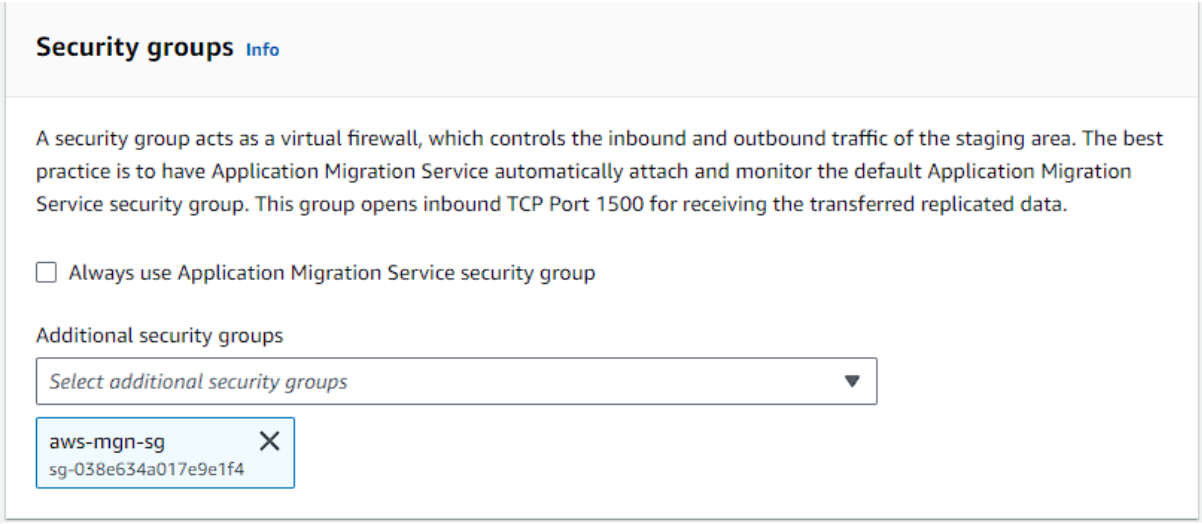
10. Then select EBS volume type to gp2 and keep EBS encryption to default.



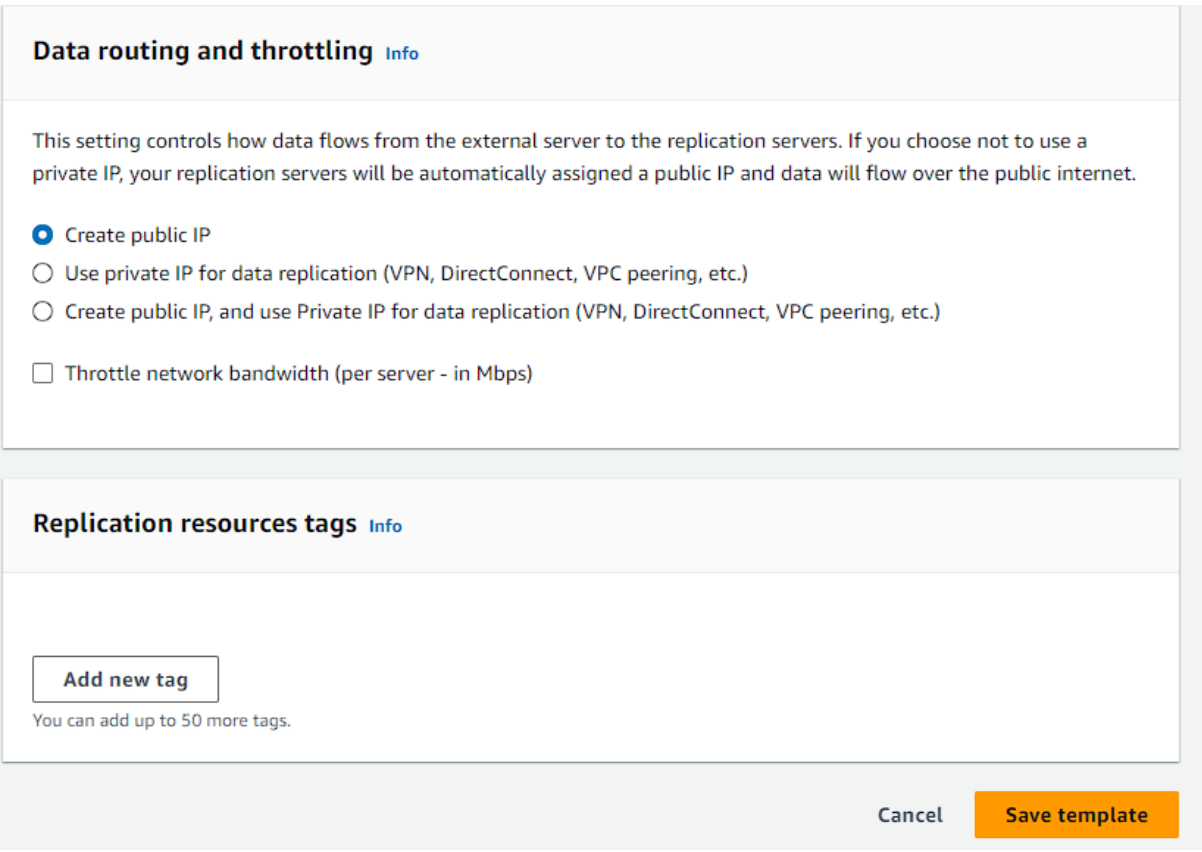
11. For the security group you have to create a new security group with port 1500 open for everywhere in inbound rule.



12. After creating your security group refresh your page again and select your security group.

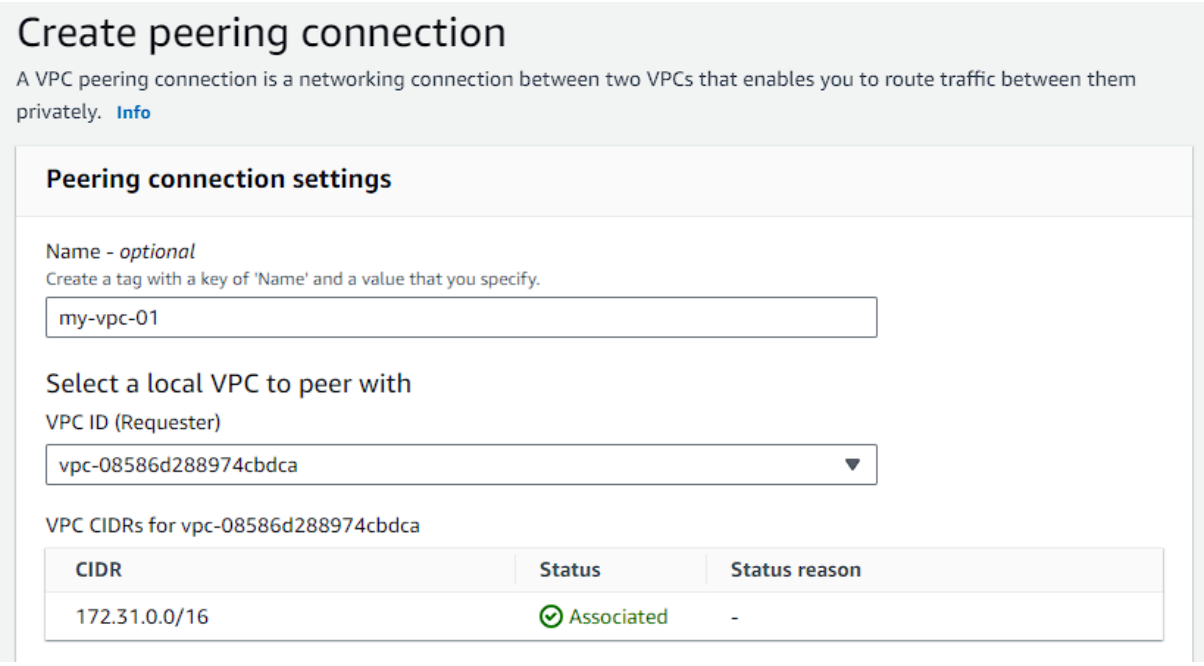


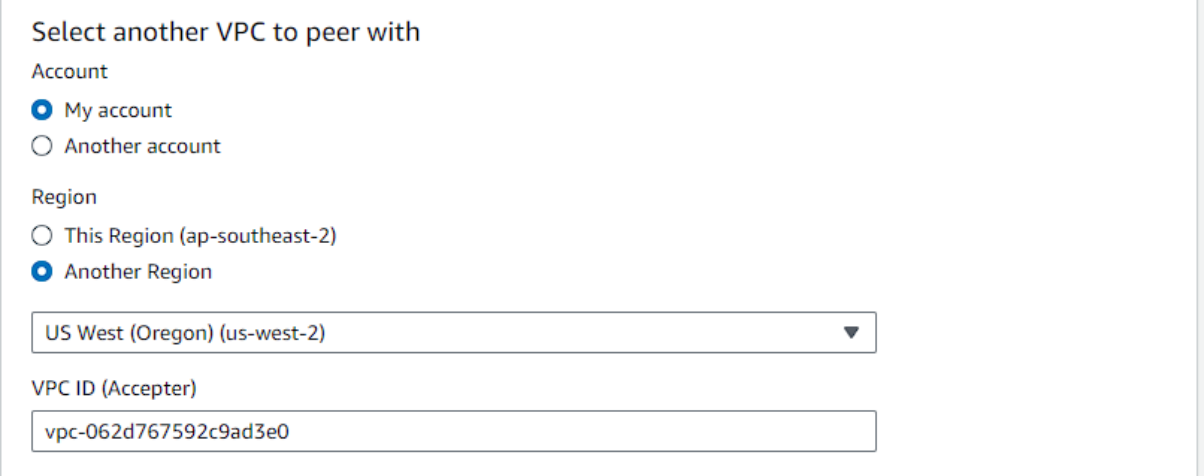
13. For the demo purpose, select create public IP.
14. Now click on save template and your replication template will be edited as per your choice.



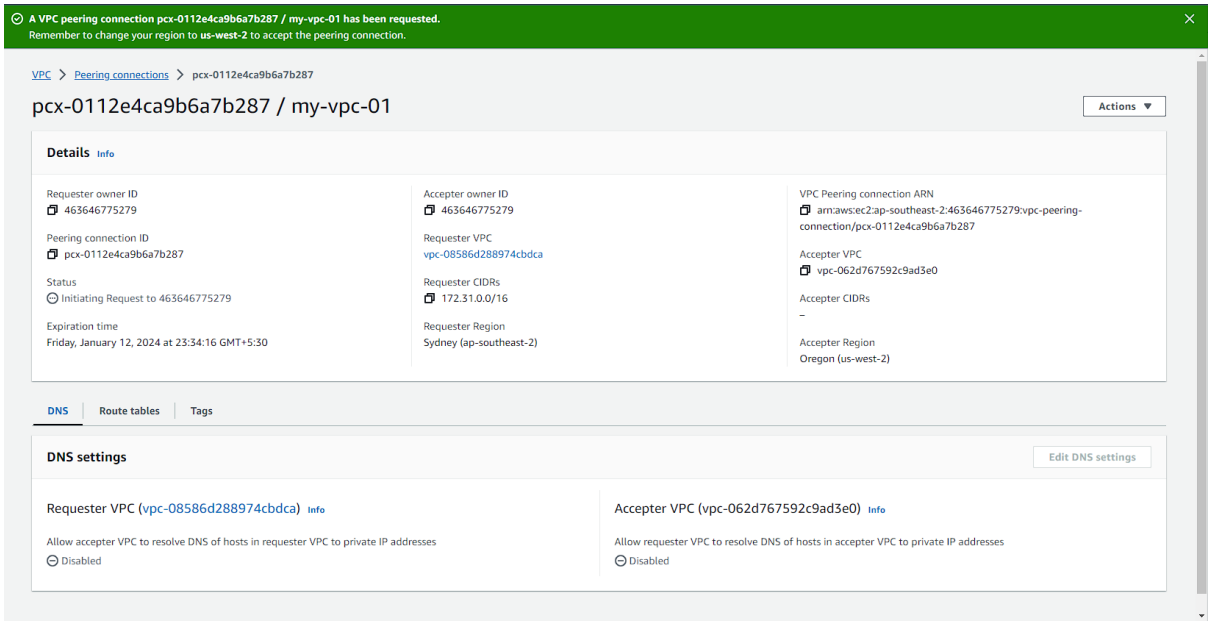
🤓 STEP 2: Create Peering connection

1. Now you have to create VPC Peering connection between the two regions that you have selected for the data migration. (like mine are Sydney and Oregon)
2. Go to peering connection and create a connection from your source between both the region.

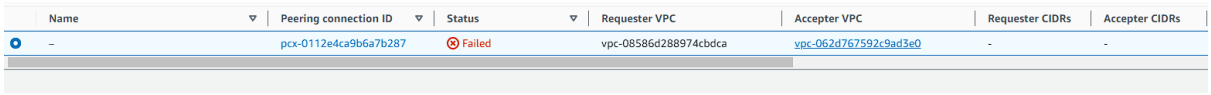


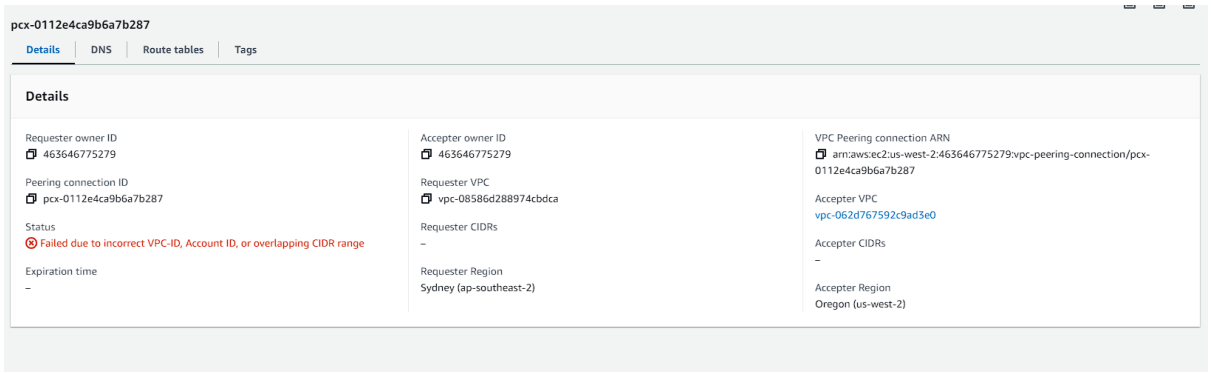


3. Once your connection is created, it will look like this.

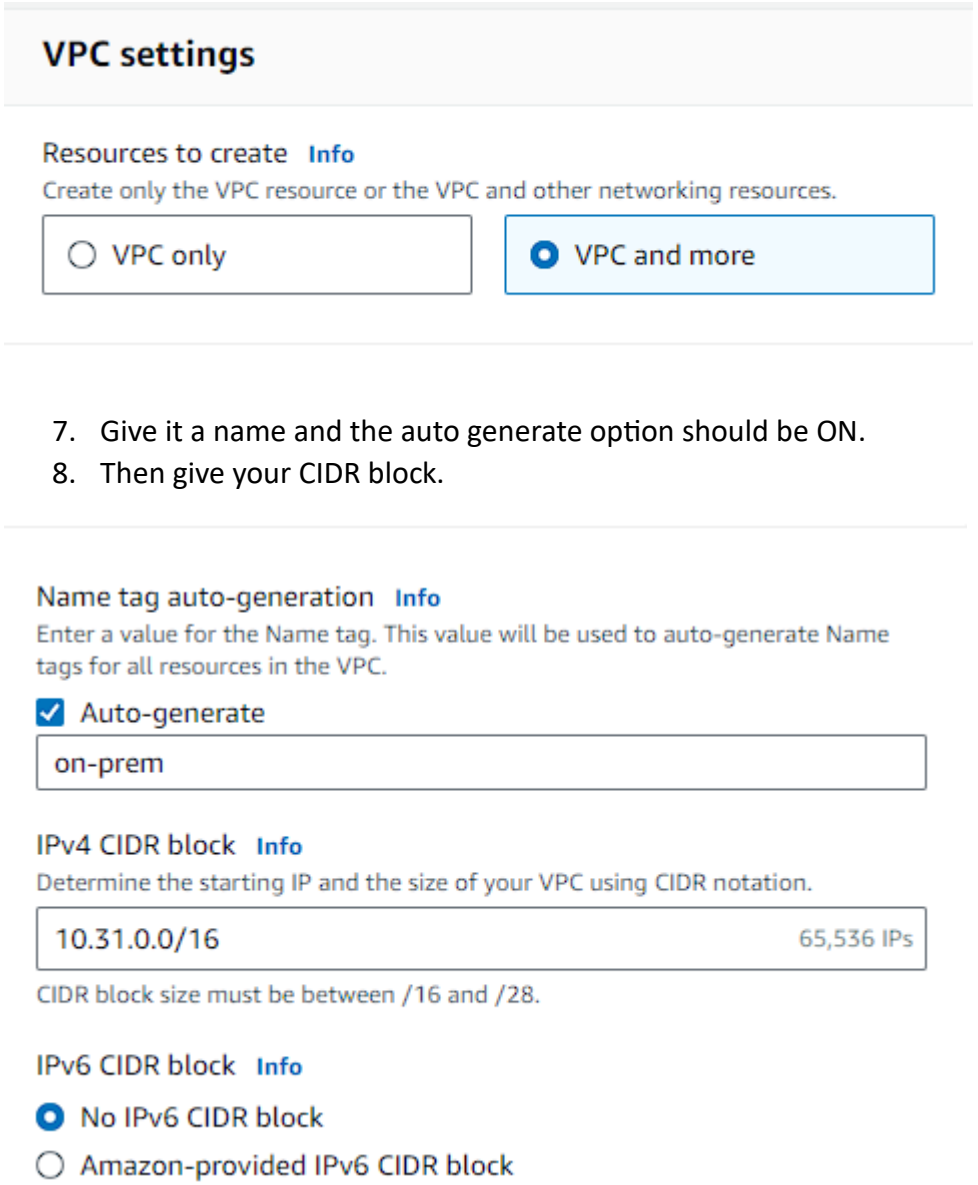


4. **But there is a catch here, now you created peering connection using default VPC, because of it there is a failure on the other side because of the overlapping of the CIDR. That is why you need to create a new VPC and then again create your peering connection.**



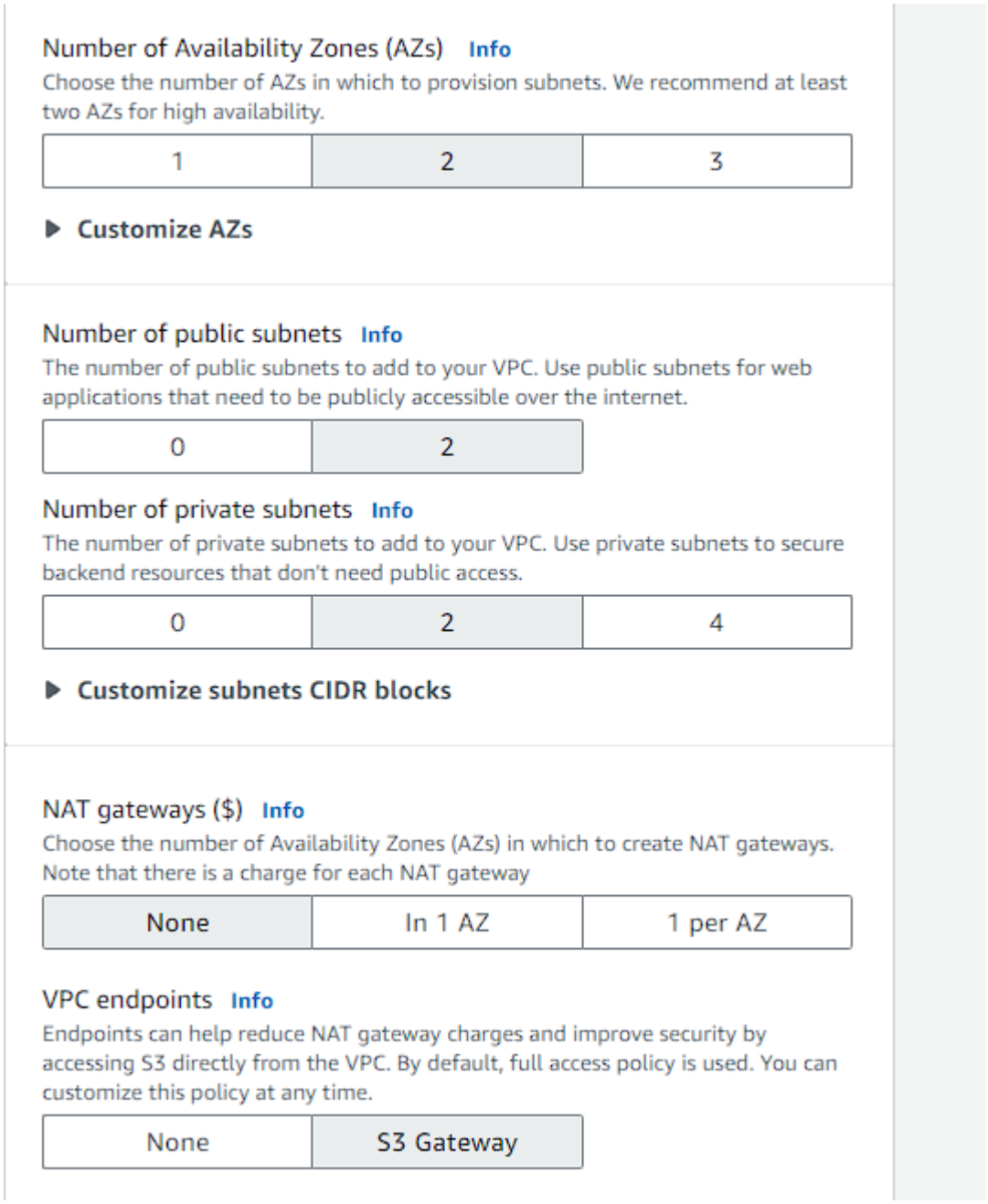


5. Now create new VPC for source region then again create your peering connection from there only.
6. While creating your VPC you need to select VPC and more because is kind of an automation type of mode for creating VPC.

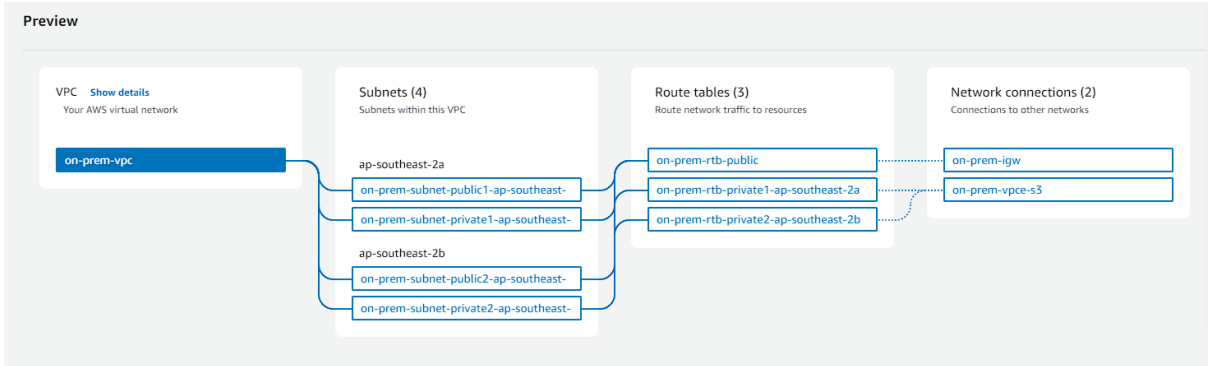


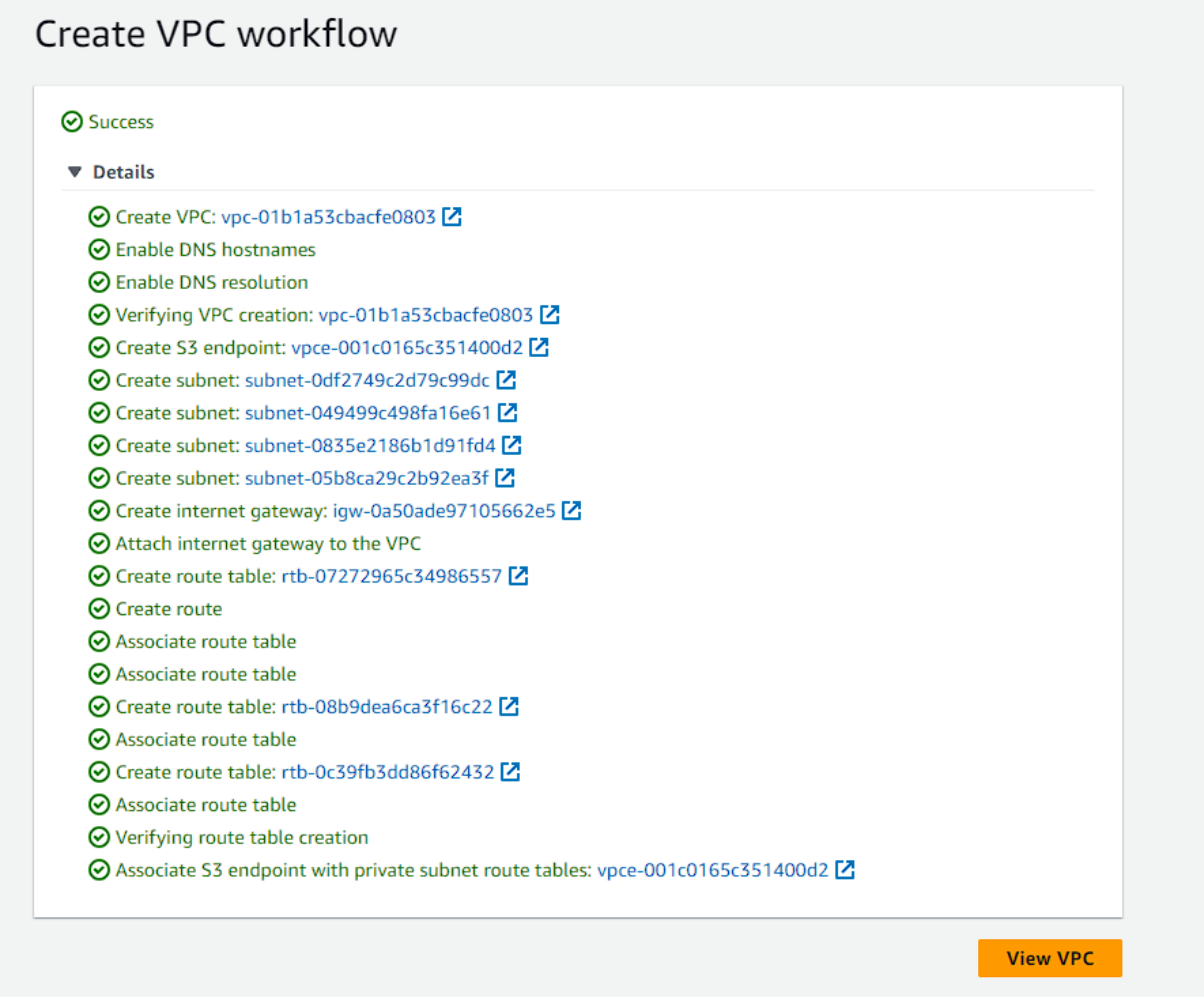
7. Give it a name and the auto generate option should be ON.
8. Then give your CIDR block.

9. Now for the number of availability zones, select it to 2 because you need two zones.
10. Number of public subnets should be 2.
11. Number of private subnets should also be 2.
12. Keep NAT gateways to none.
13. Select S3 for VPC endpoints.

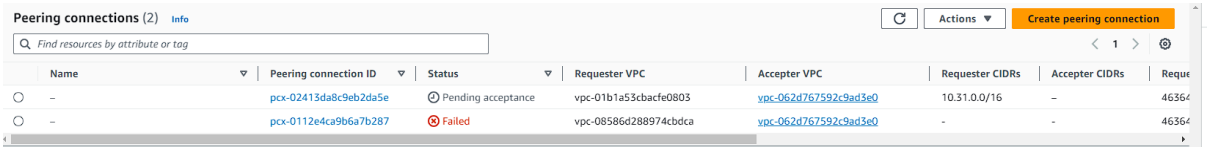


14. As you can see it is giving you the preview for the VPC.
15. Whatever the requirements are, it is creating them itself.
16. Now click on create VPC and you will see your workflow is also been created.

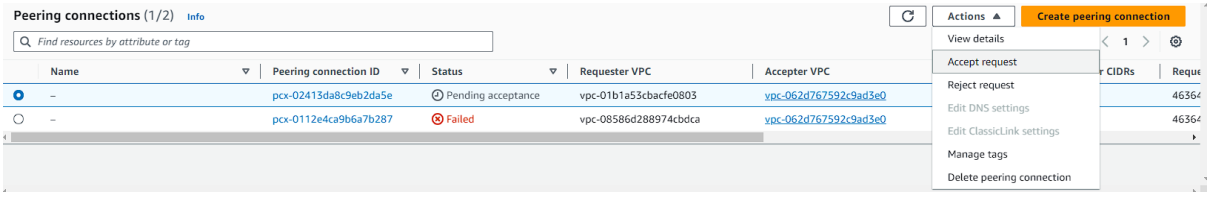




17. Now go back to peering connection and create your connection.
18. Once you have created a new Peering connection, you will see that at your destination it asking you to accept the connection.

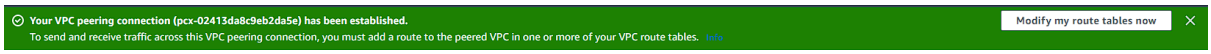


19. So, to accept it, select your connection, then click on actions, now click on accept invite.



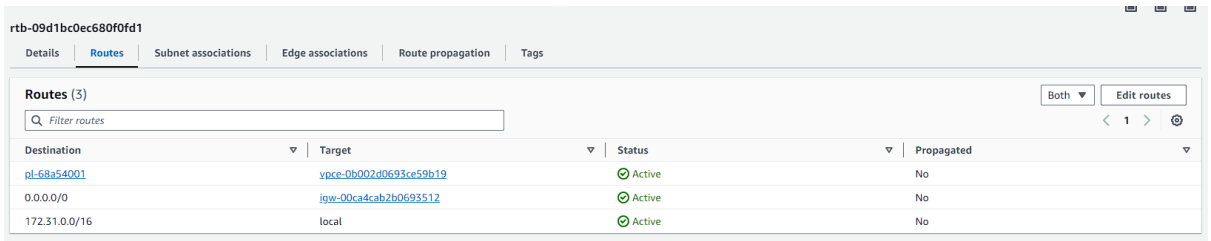
20. Soon after, you will see this window from here accept the request.

21. Once you have accepted the request, you will get this message to fix your route table.

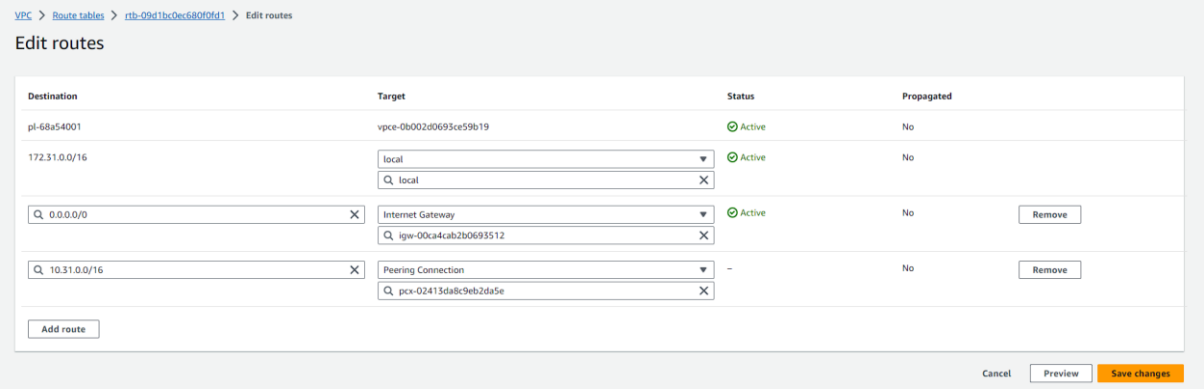


22. To avoid the confusion between the route table, first go to your VPC, then select your VPC and in the details of VPC, you can find this option for Main Route Table. Now select this option to directly go the desired route table.

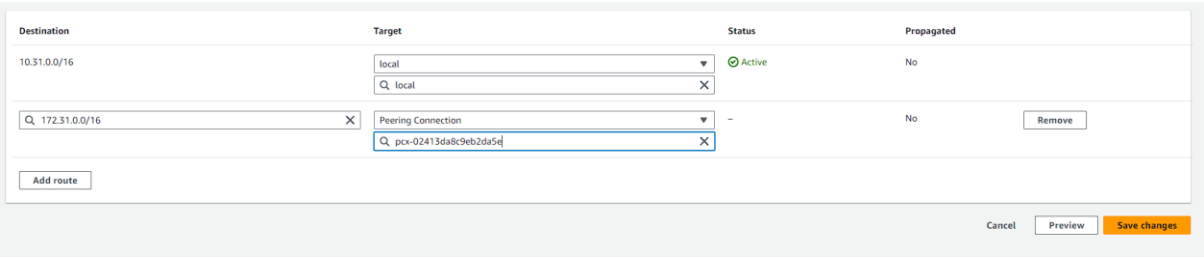
23. For this, now you have to make some changes in your route table. Go to your route table, select your route table then click on routes. Then click on edit routes.



24. Now in the edit route section, click on add route, there you need to write the CIDR of your source region because you are at your destination region right now.
25. Then select Peering connection in target and click on save changes.



26. Once it has been done then go to your source region and so the same thing.



STEP 3: Create EC2 Instances

1. Now navigate to EC2 and create 2 instances in your source region.
2. First create a windows instance. Then create a Linux instance.
3. While creating the windows instance, in the network settings, select the VPC which you created and for the subnet select any public subnet. Disable auto assign public IP.
4. Then create your windows instance.

Name and tags Info

Name

on-prem-win-server

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Microsoft Windows Server 2022 Base — Free tier eligible
ami-015e01fec392ea845 (64-bit (x86))
Virtualization: hvm ENA enabled: true Root device type: ebs

Instance type Info | Get advice

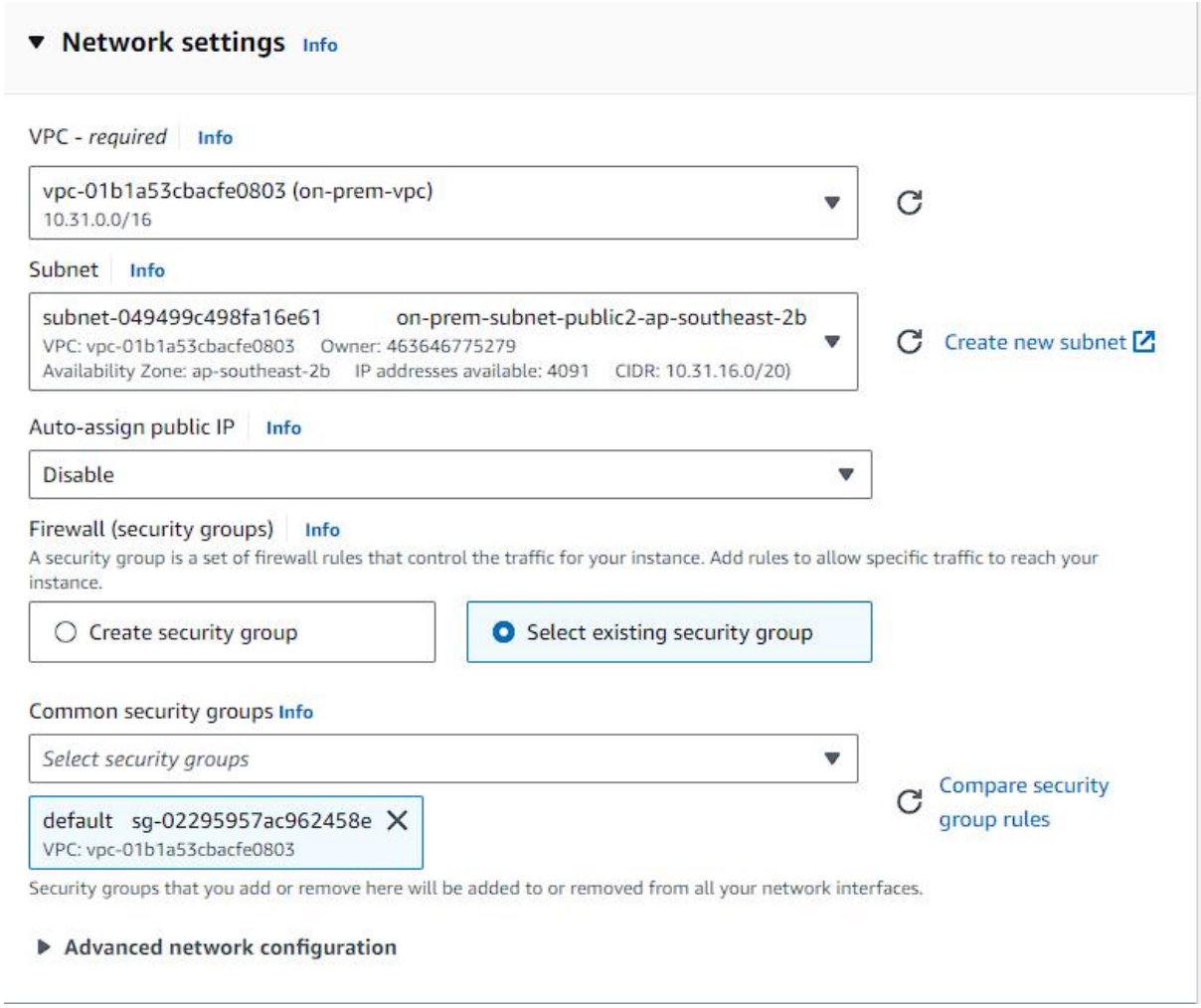
Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand SUSE base pricing: 0.0146 USD per Hour
On-Demand Linux base pricing: 0.0146 USD per Hour
On-Demand Windows base pricing: 0.0192 USD per Hour
On-Demand RHEL base pricing: 0.0746 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software



5. The steps for creating Linux instance are also same. Just remember to select Linux 2 AMI.
6. Then just launch your instance.

Name and tags Info

Name

on-prem-lin-server

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs

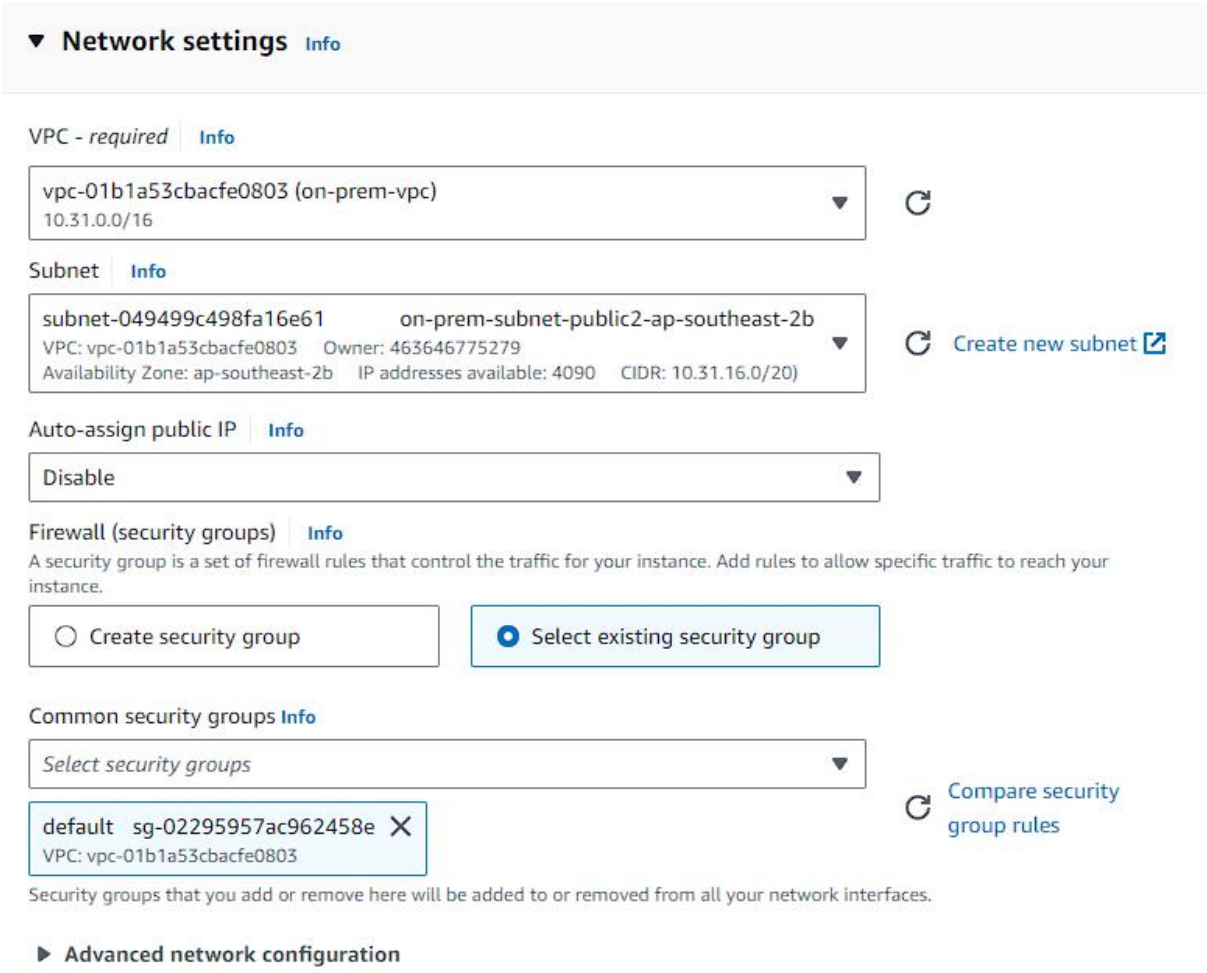
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

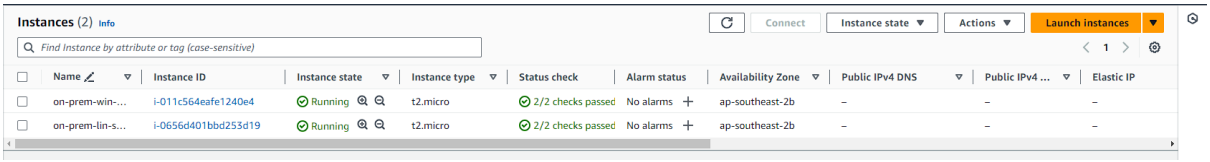
Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible

ami-0d13b3038380b0796 (64-bit (x86)) / ami-0fd0bb835243a6706 (64-bit (Arm))

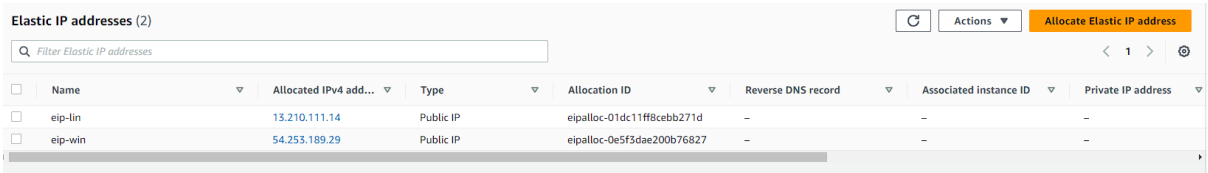
Virtualization: hvm ENA enabled: true Root device type: ebs



7. Once both of your instances are created then allocate elastic IP to both of them.

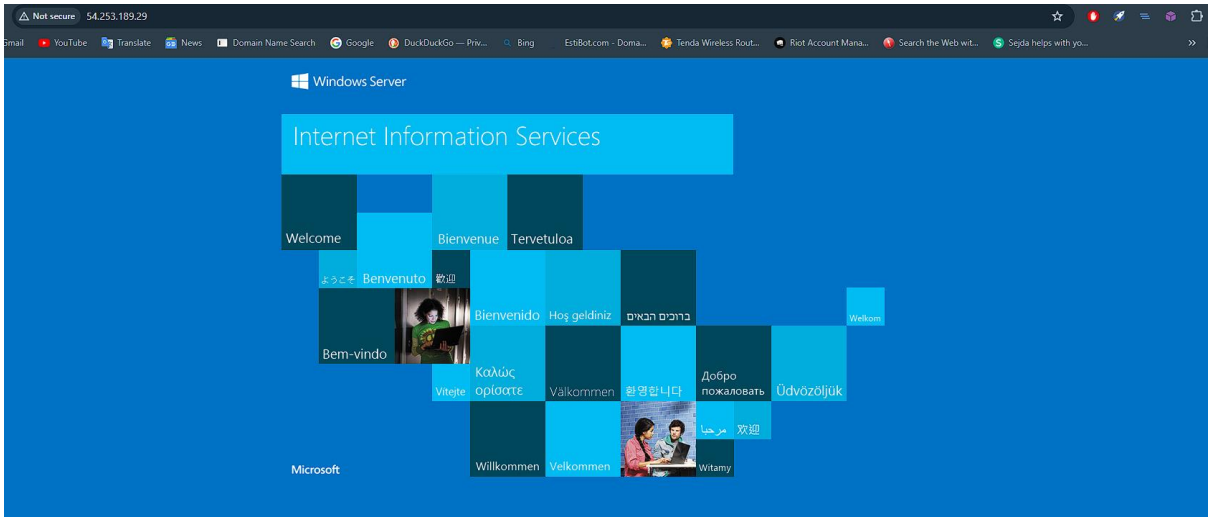


8. Now create two elastic IPs, one for Windows instance and one for Linux instance.



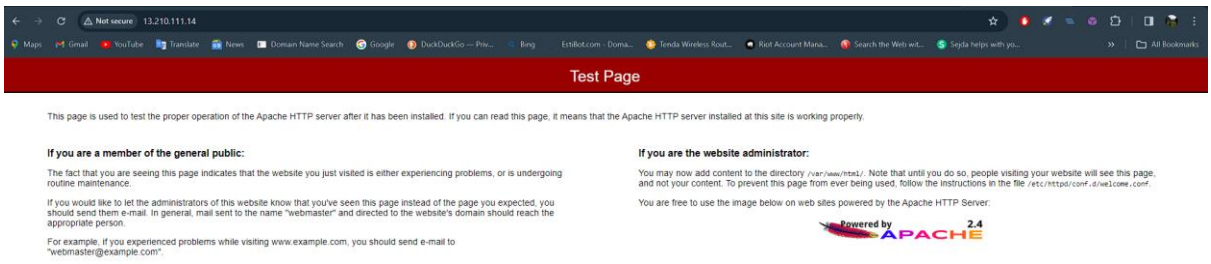
9. Now associate these elastic IPs to the instances.
10. After this, go back to the instances and try to establish a remote desktop connection with windows instance.
11. Before that you need to add port 3389 and port 80 to the inbound rules of the windows instance.
12. There you need to open server manager dashboard in the windows instance and install IIS (Internet Information Service) over there.

13. Once you have installed IIS on windows instance then you need to copy its public IP address and paste it in a new tab. You can see that IIS is successfully installed on it.



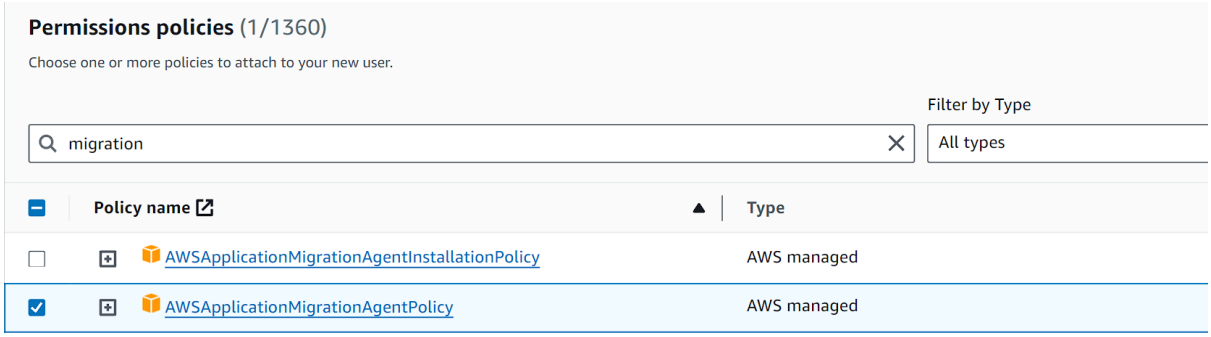
14. Now SSH in to the Linux instance and install HTTPD on it.
15. But before that add port 22 and 80 on Linux instance.
16. Now connect it to the session manager.
17. In there you need to run some commands. Run these commands step by step and it will install the HTTPD or the apache test server on your linux instance.

    sudo -i
    yum update -y
    yum install httpd -y
    systemctl status httpd
    systemctl start httpd
    systemctl enable httpd

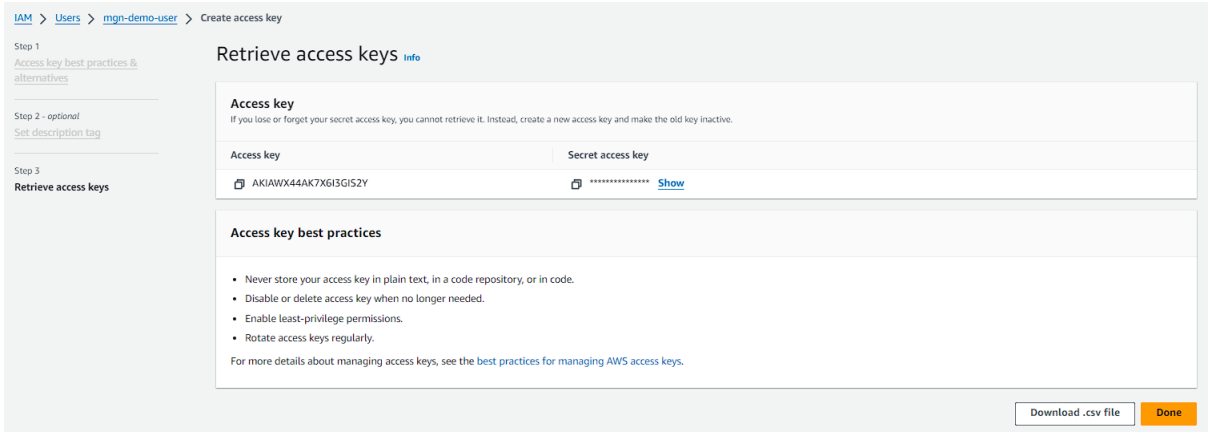


STEP 4: Now go to Target MGN and create Server

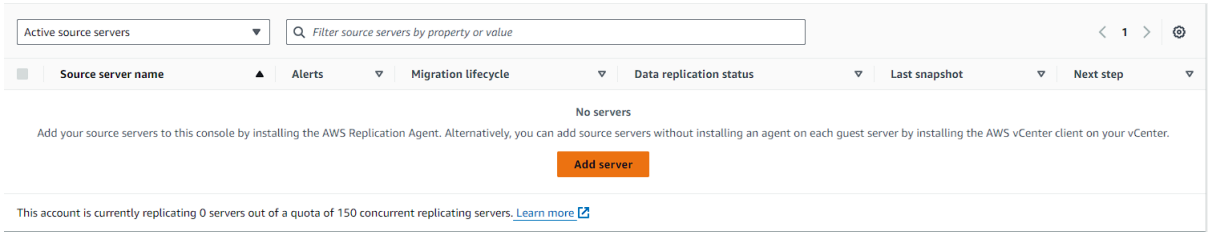
1. Here you have to add the servers but before adding the servers you need to have access keys and secret access keys.
2. For that go to IAM and create a user then download its access and secret keys.
3. Attach this policy to the user and then create your user.



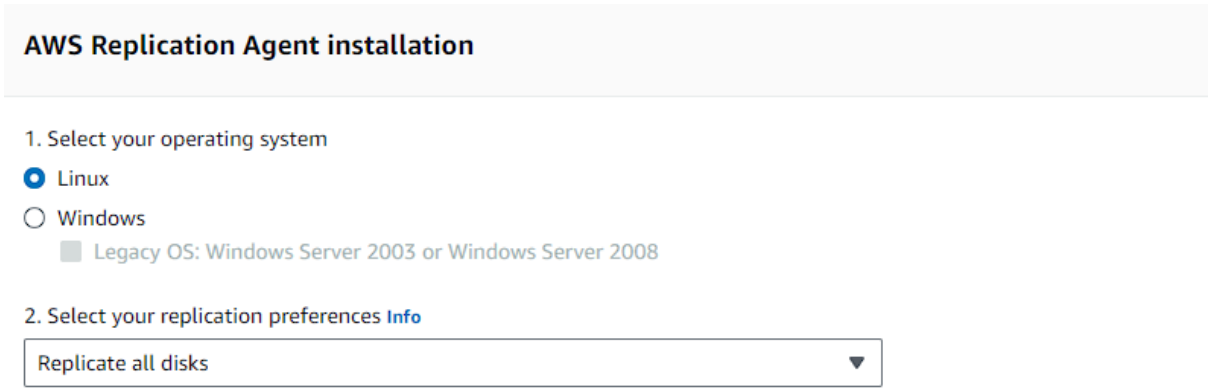
4. Once your user is created now go and create its access and secret keys.



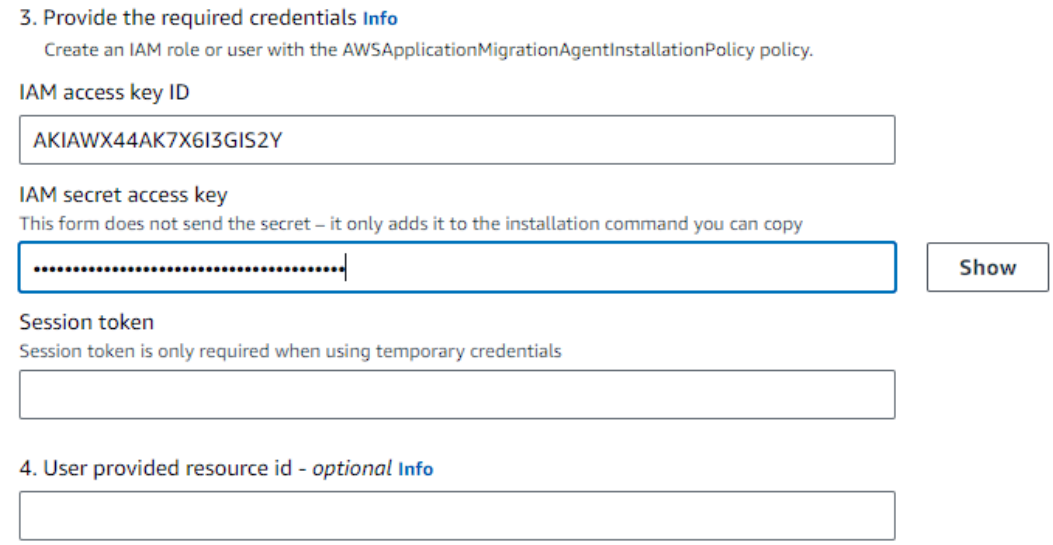
5. After everything is set up, now go back to MGN. There you need to select and go to source server.
6. Now you have add your server.



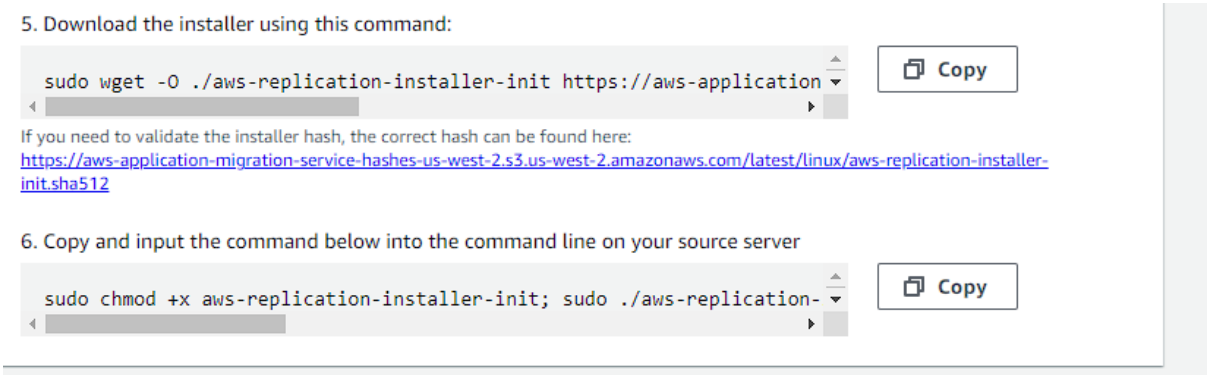
7. In there choose it for Linux.
8. Then select replicate all disks.



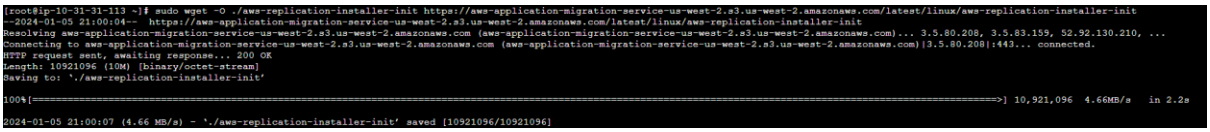
9. Now give your access and secret keys.

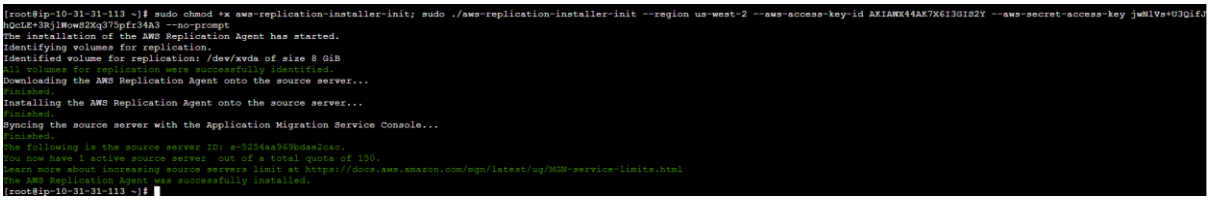


10. Once you have given the keys it will give you a command which you have to use on your Linux instance SSH session client and install it there.

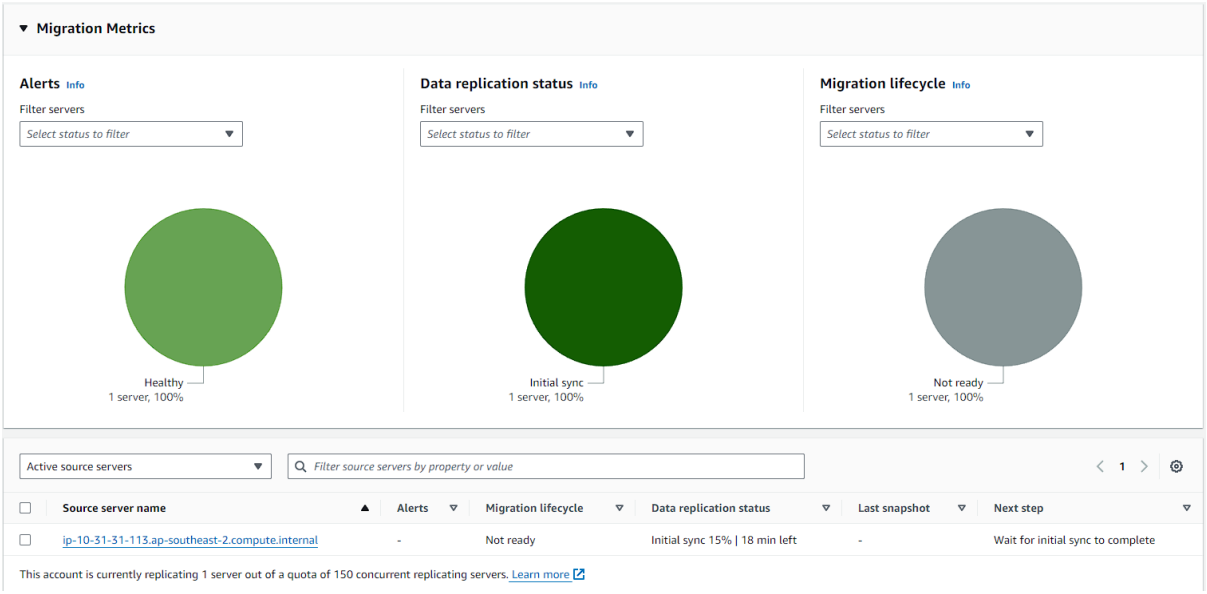


11. Copy these commands there.

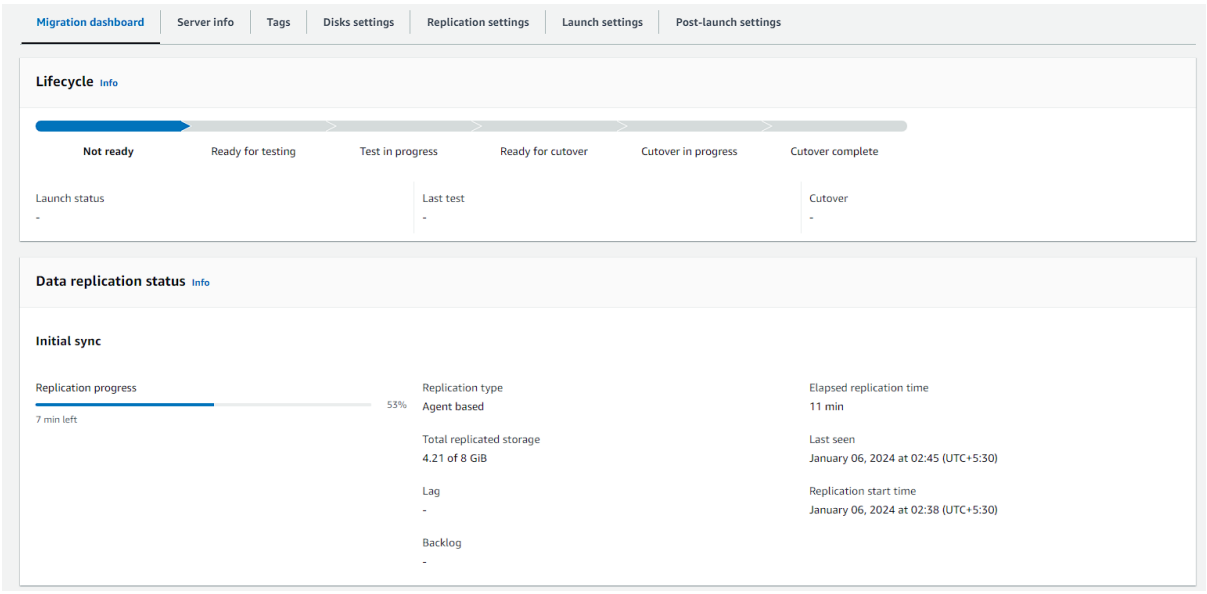




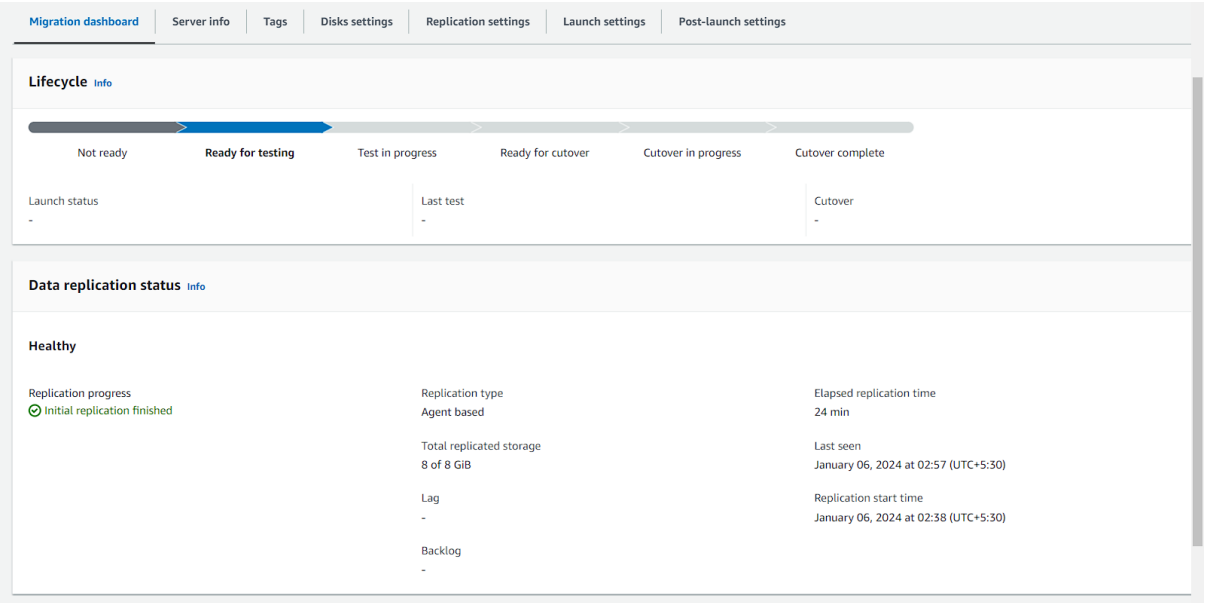
12. Once your commands have run successfully then if you go back to your MGN, then you will see, that a server has been added here.



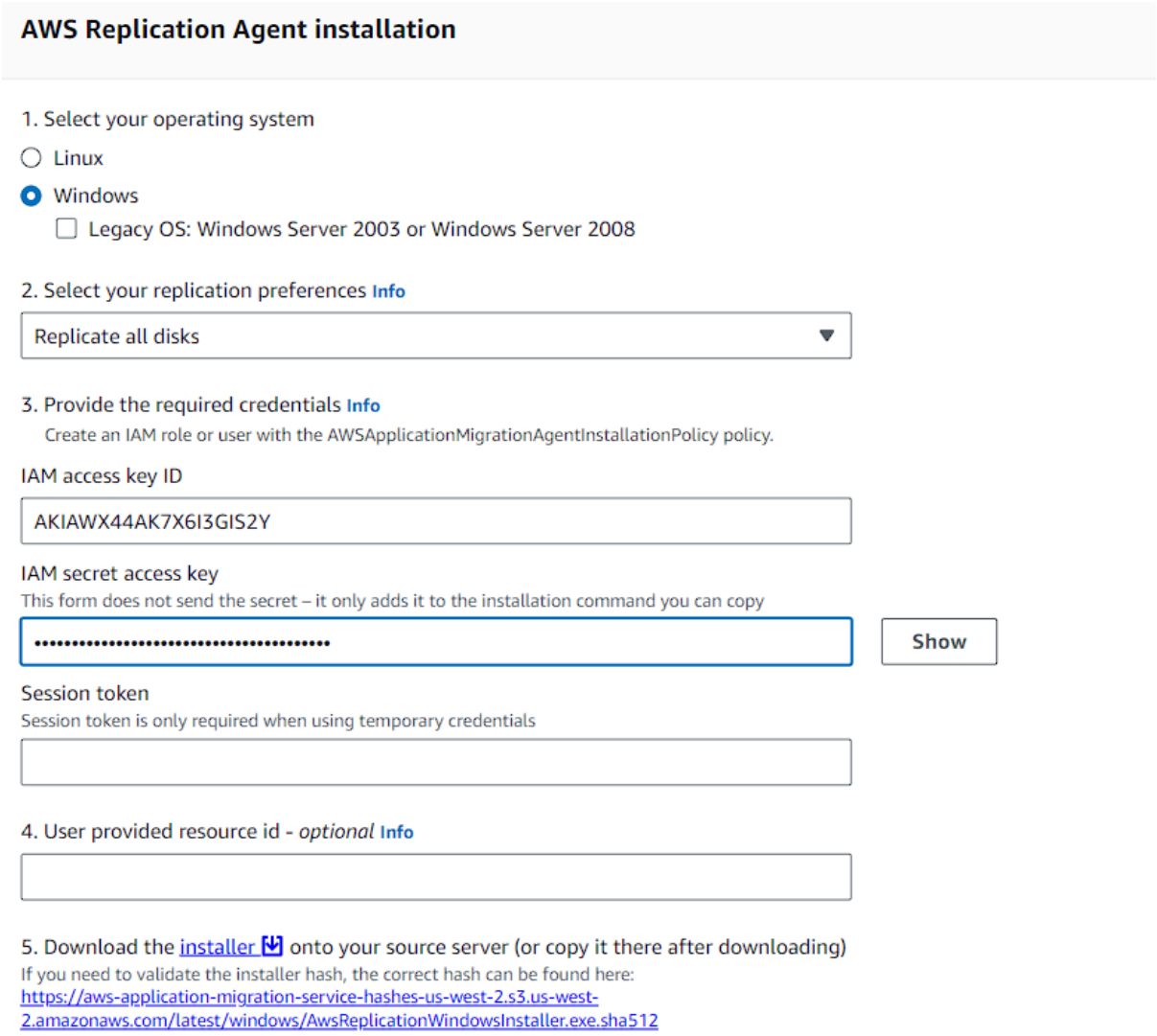
13. And if you will open your server, you can see the data replication status.



14. Now you have to wait for some time, till the process gets complete.

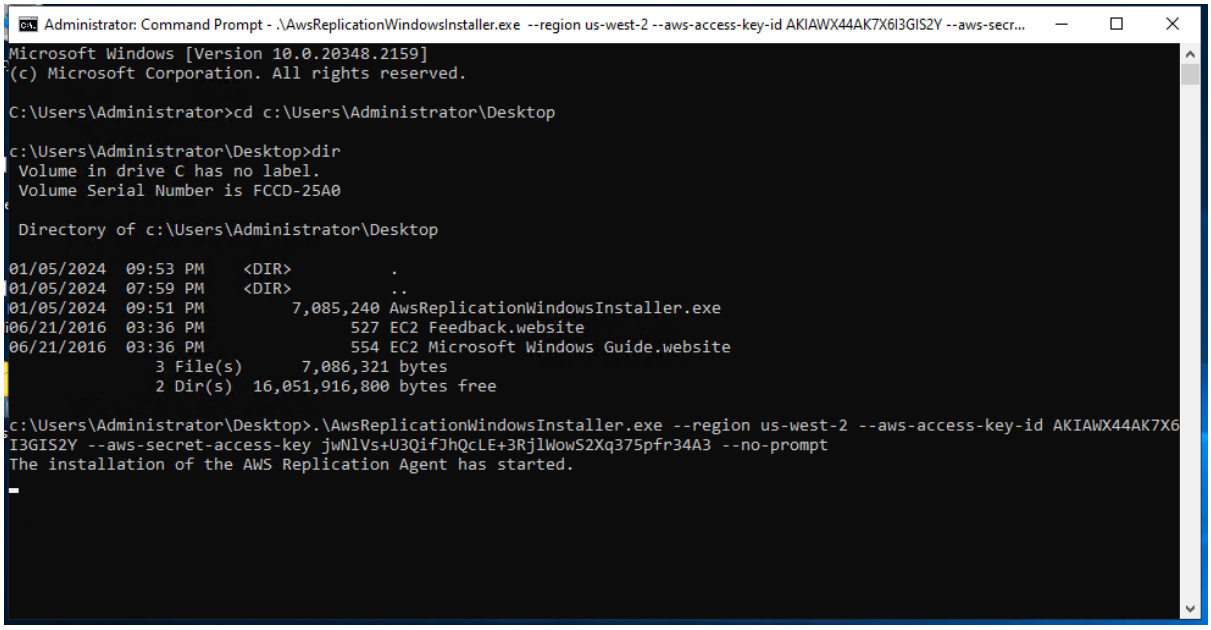


15. Now as Linux data has started migrating, move towards windows data.
16. Go in the Remote session of the windows instance and do the same thing there too.
17. First go to MGN and add a server for windows.
18. The process is almost same, select windows for the OS.
19. Then select replicate all disks.
20. Provide access and secret access keys.
21. Then you need to download an installer then copy it and paste it in the windows server.



22. After downloading the installer paste it in the windows server
23. Now open command prompt in the windows server

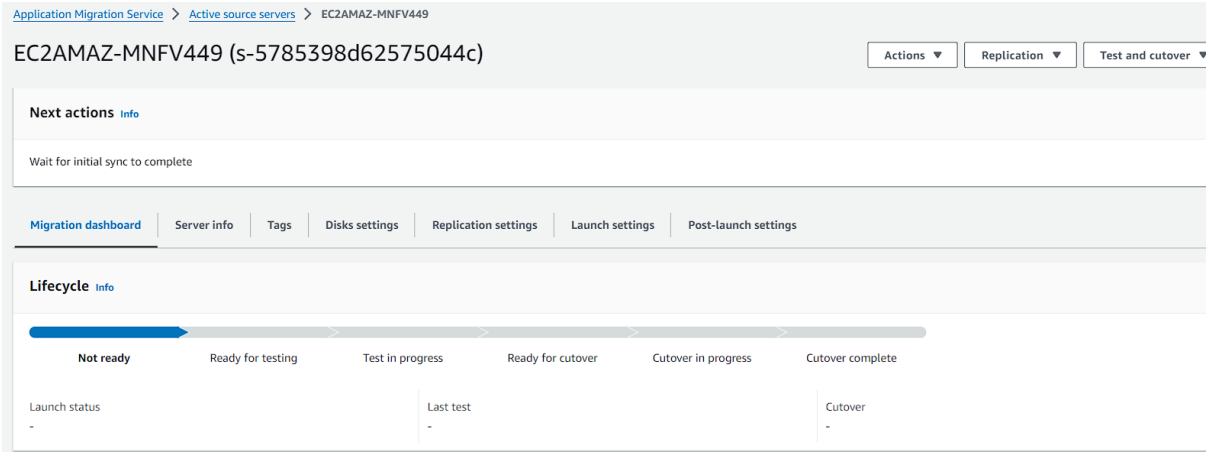
24. Open CMD and go to this path. Say to the desktop or to the path where you will install this AWS replication windows installer.
25. After that copy the code from MGN and paste it in the CMD, it will execute it and create a new server for data replication.



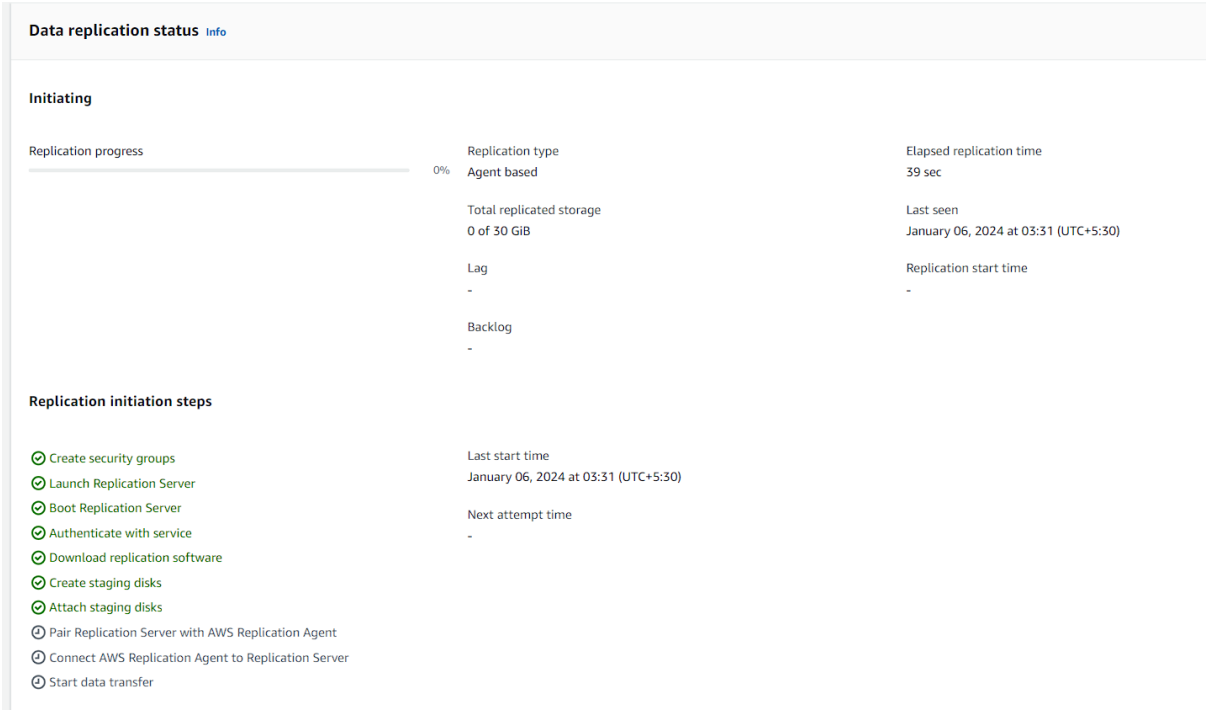
26. As you can see the command has been executed successfully.

```
c:\Users\Administrator\Desktop>.\AwsReplicationWindowsInstaller.exe --region us-west-2 --aws-access-key-id AKIAWX44AK7X6
I3GIS2Y --aws-secret-access-key jwNlVs+U3QifJhQcLE+3RjlWowS2Xq375pfr34A3 --no-prompt
The installation of the AWS Replication Agent has started.
Verifying that the source server has enough free disk space to install the AWS Replication Agent (a minimum of 2 GB of f
ree disk space is required).
Identifying volumes for replication.
Disk to replicate identified: c:0 of size 30 GiB
All volumes for replication were successfully identified.
Downloading the AWS Replication Agent onto the source server...
Finished.
Installing the AWS Replication Agent onto the source server...
Finished.
Syncing the source server with the Application Migration Service Console...
Finished.
The following is the source server ID: s-5785398d62575044c.
You now have 2 active source servers  out of a total quota of 150.
Learn more about increasing source servers limit at https://docs.aws.amazon.com/mgn/latest/ug/MGN-service-limits.html
The AWS Replication Agent was successfully installed.
```

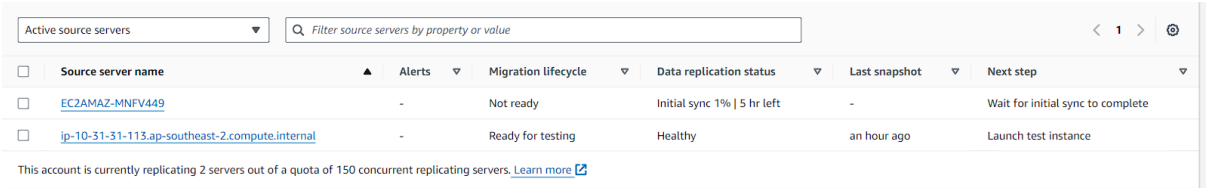
27. Now if you will come back to MGN you can see that a windows server has been created and it up and running to migrate data.



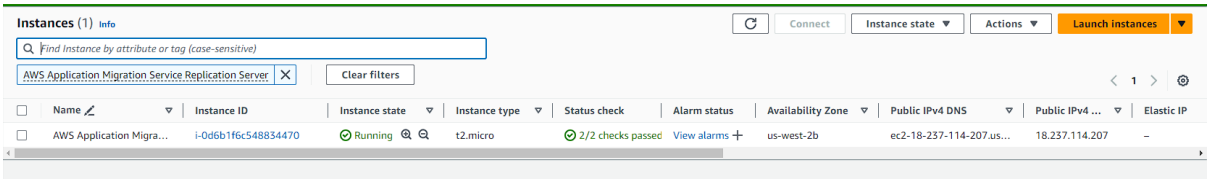
28. Below are the steps and progress.



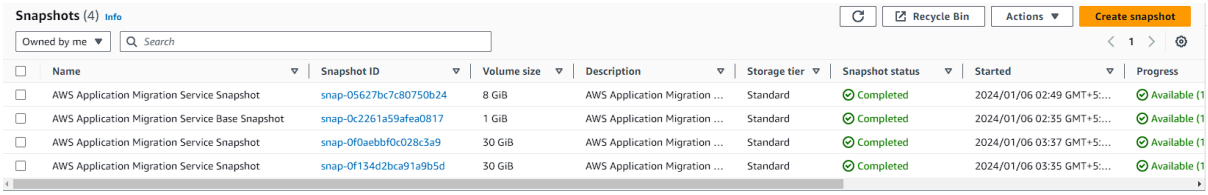
29. These are the two servers. They might take their time to replicate data.



30. Now if you will go to EC2 of the destination region. You will see that a instance is up and running.

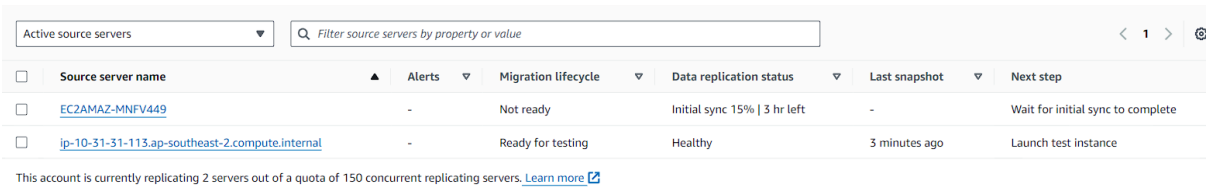


31. And if you visit to the snapshots part of the EC2 under elastic block store (EBS).
32. You will see that snapshots are being created for your replicated data.

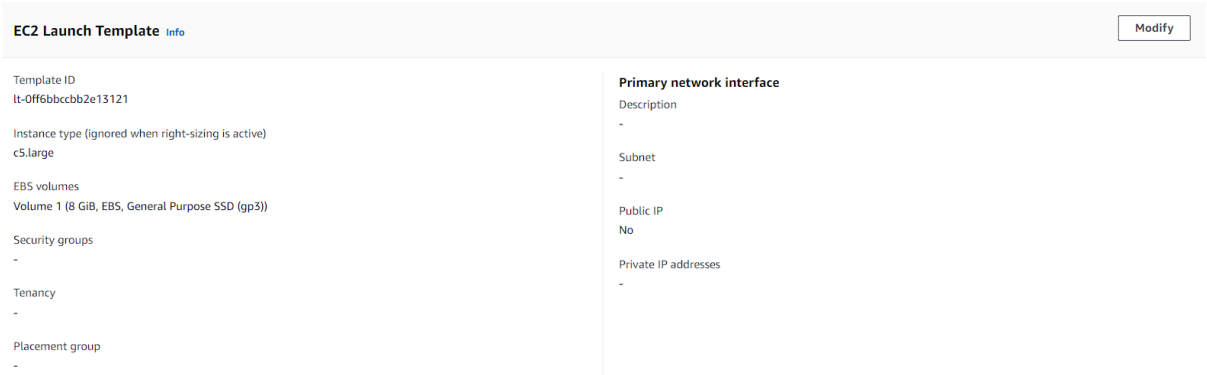


STEP 5: Checking Server

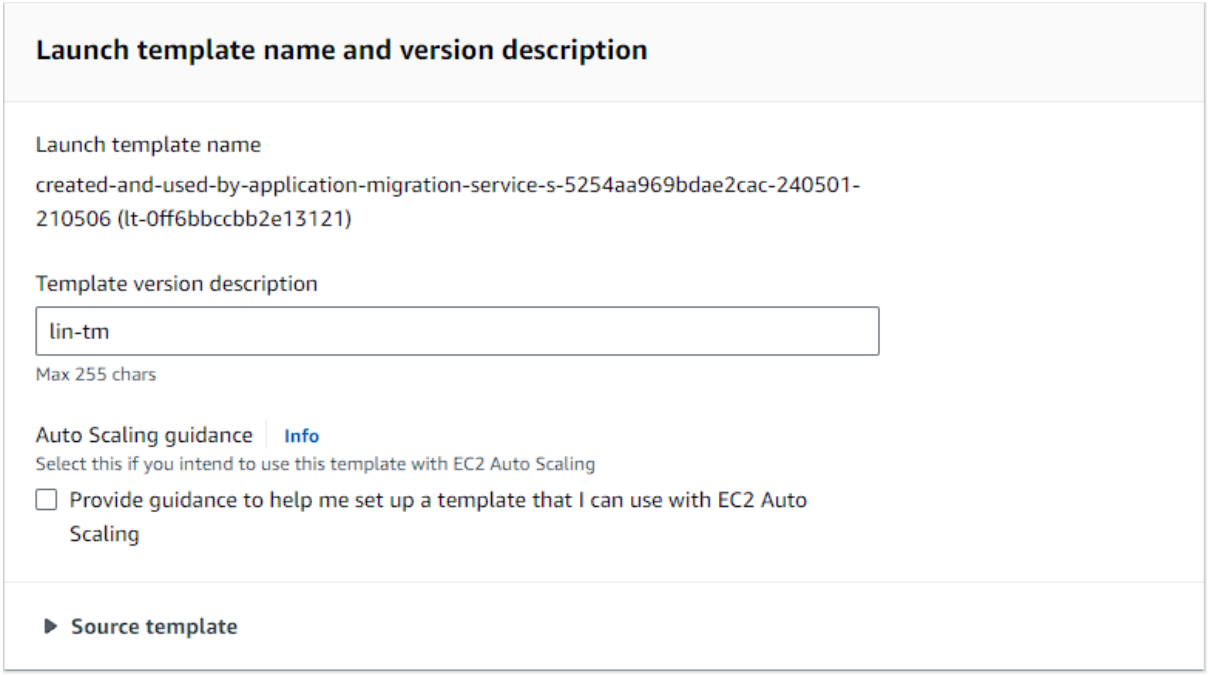
1. Now go back to check if any one of your servers is healthy or not.



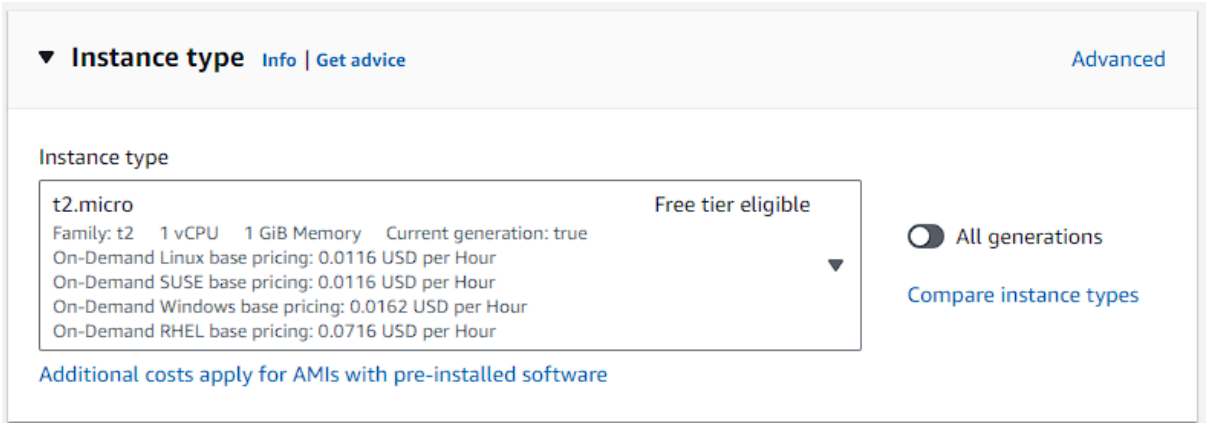
2. As you can see the Linux is the one that is ready to get launched.
3. Now open it. Go to launch settings. Then click on modify EC2 launch template.



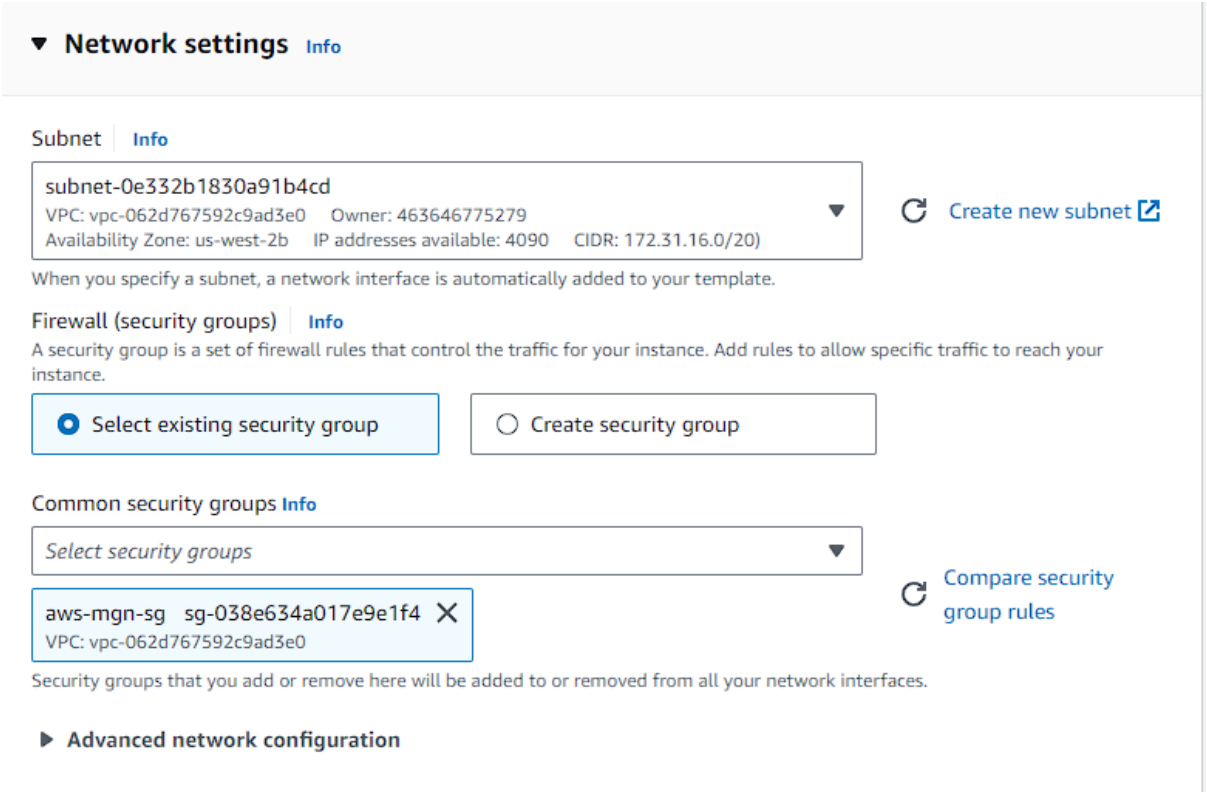
4. In there, first you need to give it a name.



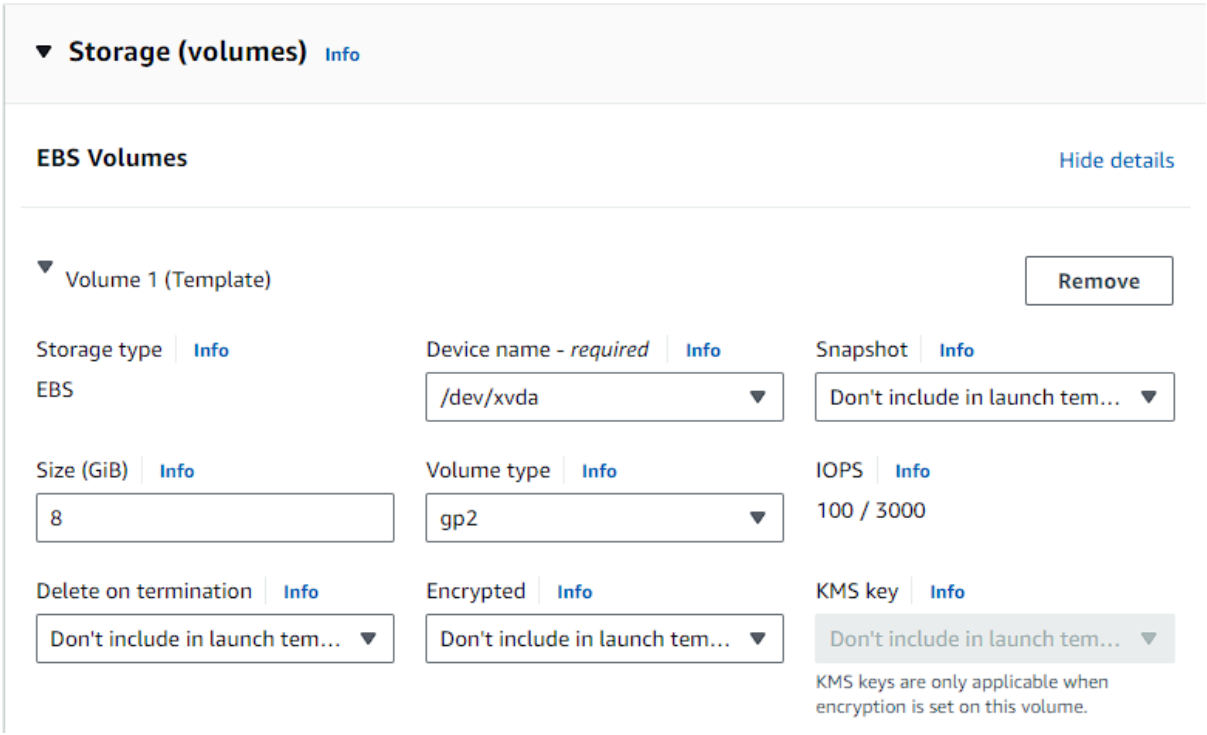
5. Then select instance type to t2.micro



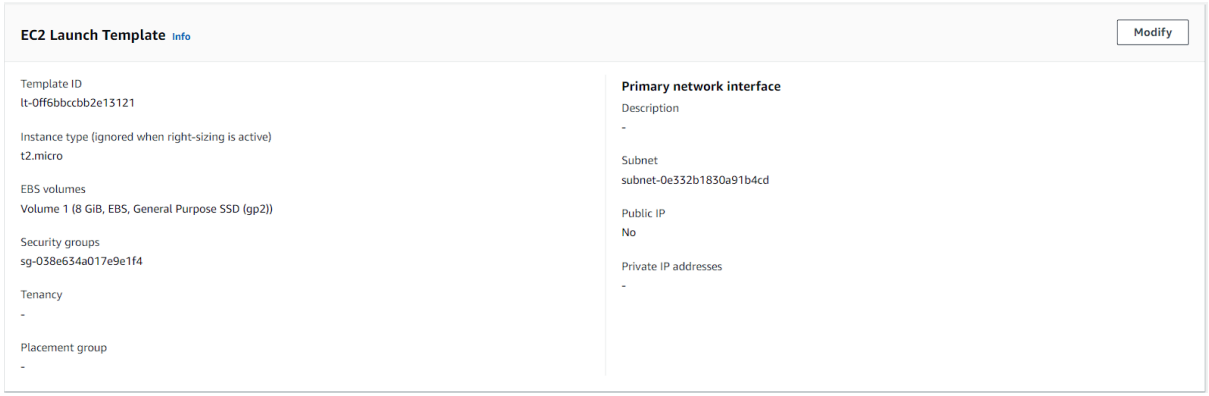
6. In the network settings, select your subnet.
7. Then select your existing security group.



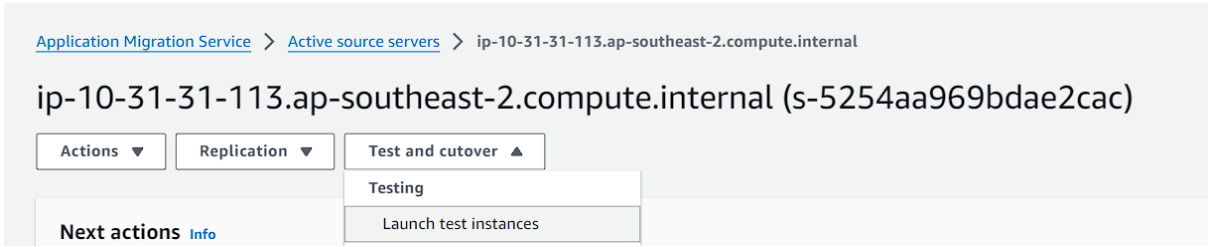
8. In the EBS volumes select volume type to gp2 and you can increase the size from 8gb to whatever you like. Disk Size totally depends on you.



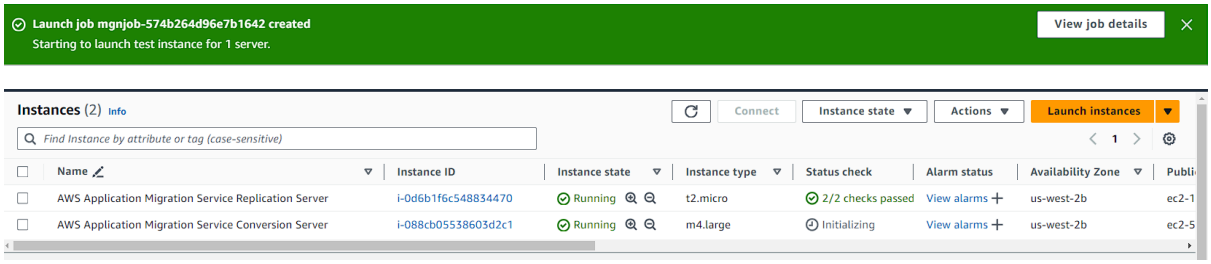
9. Once you have done all the above settings now modify it and you will see that these are the specification that you changed. Before the instance type was c5.large but now it is t2.micro



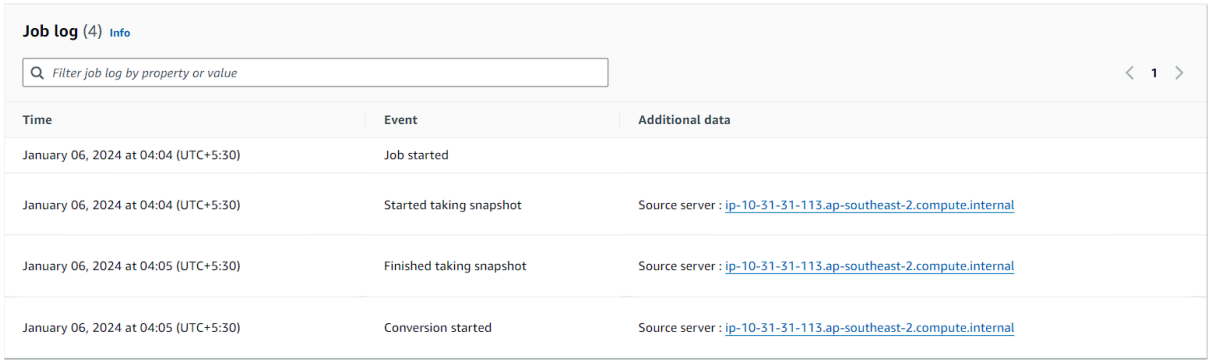
10. Now go back and click on test and cutover, then click on Launch test instance.



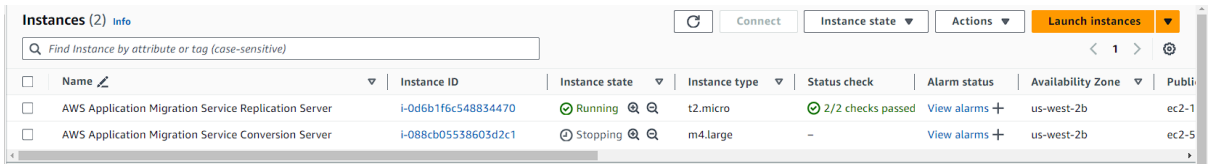
11. Once you have done that you can see that a Conversion Server is also been started as an EC2 instance.



12. And if you come back to MGN you can see that the logs are generating and it is showing you what is going on the back end of the server.



13. Once the conversion is completed you can see that the Conversion server has stopped.



14. In the logs too, it has been stopped.

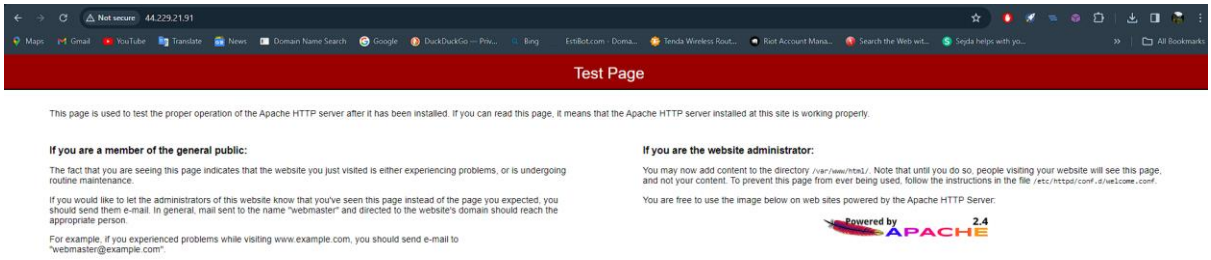
Job log (6) Info

Filter job log by property or value

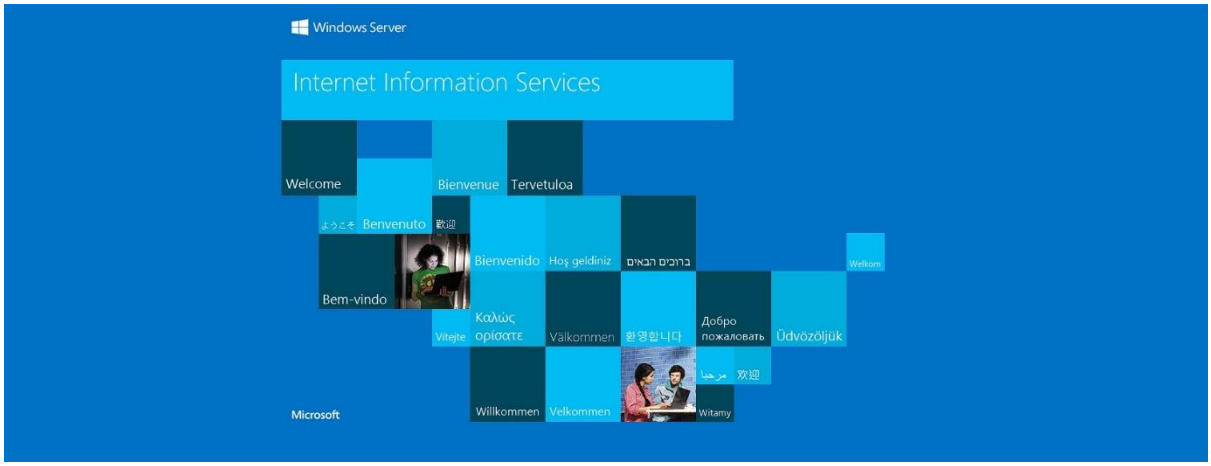
1

Time	Event	Additional data
January 06, 2024 at 04:04 (UTC+5:30)	Job started	
January 06, 2024 at 04:04 (UTC+5:30)	Started taking snapshot	Source server : ip-10-31-31-113.ap-southeast-2.compute.internal
January 06, 2024 at 04:05 (UTC+5:30)	Finished taking snapshot	Source server : ip-10-31-31-113.ap-southeast-2.compute.internal
January 06, 2024 at 04:05 (UTC+5:30)	Conversion started	Source server : ip-10-31-31-113.ap-southeast-2.compute.internal
January 06, 2024 at 04:07 (UTC+5:30)	Conversion ended	Source server : ip-10-31-31-113.ap-southeast-2.compute.internal Conversion Server instance ID: i-088cb05538603d2c1
January 06, 2024 at 04:07 (UTC+5:30)	Started launching test/ cutover EC2 instance	Source server : ip-10-31-31-113.ap-southeast-2.compute.internal

15. And if you go back to your instance which was created on the destination region, and if you copy its public IP after allocating it an elastic IP.
16. So, you can see your test which you created on the source region EC2 instance.
17. This means that our migration is successfully completed.



18. The same thing is going to happen with windows server too.
19. Check it in the MGN that is healthy or not and when it is healthy then you can go and modify its EC2 instance settings then you will see a conversion instance.
20. Once the conversion is complete then if you go its instance and allocate it with an elastic IP.
21. Then copy its public IP and paste it in a new browser, then you will your IIS installed on it too. Which means that it has been migrated successfully.



AFTER COMPLETING YOUR MIGRATION DELETE ALL YOUR INSTACNES AND MGN ACCORDINGLY.

DO NOT FORGET TO DISSASSOCIATE YOUR ELASTIC IP.