AWS Partner: Advanced Migrating to AWS (Technical)

Lab Guide

Version 3.0.0

300-PTADMI-30-EN-LG

Corrections, feedback, or other questions? Contact us at
https://support.aws.amazon.com/#/contacts/aws-training.

Contents

Advanced Migrating to AWS with AWS Cloud Migration Factory

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab Overview

This lab uses an emulated on-premise environment with AWS EC2 servers, configured in a way that mimics a customers' environment, with private and public networks, DNS server, pre-installed applications and a bastion host.

Objectives

After completing this lab, you will be able to:

- Understand how migration projects works.
- Discover servers and applications using AWS Application Discovery Service.
- Group servers as applications using AWS Migration Hub.
- Migrate servers with AWS Application Migration Service.

Prerequisites

This lab requires:

- Google **Chrome** browser. The lab requires the students to use only Chrome as the internet browser because for Fleet Manager RDP, only **Chrome** browser supports bidirectional copying and pasting between RDP sessions and your local machine.
- Access to a computer with Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).
- Be familiar with the AWS console.

- Be familiar with AWS Application Migration Service (MGN).

Terminology table

Term	Description
Bastion Host	Also known as Migration Automation Server. It is a windows server to host and run all the automation scripts.
MGN	Also known as AWS Application Migration Service, a lift and shift solution to rehost servers to EC2.
Migration Factory	An AWS solution to automate and migrate servers to AWS at scale.
Wave	Also known as Migration wave, it is a group of apps and servers with the same migration schedule.

Duration

This lab requires approximately 120 minutes to complete.

Icon key

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A hint, tip, or important guidance.
- ⓘ **Additional information:** Where to find more information.
- ! **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- ⚠ **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).
- **Refresh:** A time when you might need to refresh a web browser page or list to show new information.

Start lab

1. To launch the lab, at the top of the page, choose Start lab.

ⓘ You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose Open Console.

You are automatically signed in to the AWS Management Console in a new web browser tab.

⚠ **Do not change the Region unless instructed.**

Common sign-in errors

Error: You must first sign out

If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose Open Console again.

Error: Choosing Start Lab has no effect

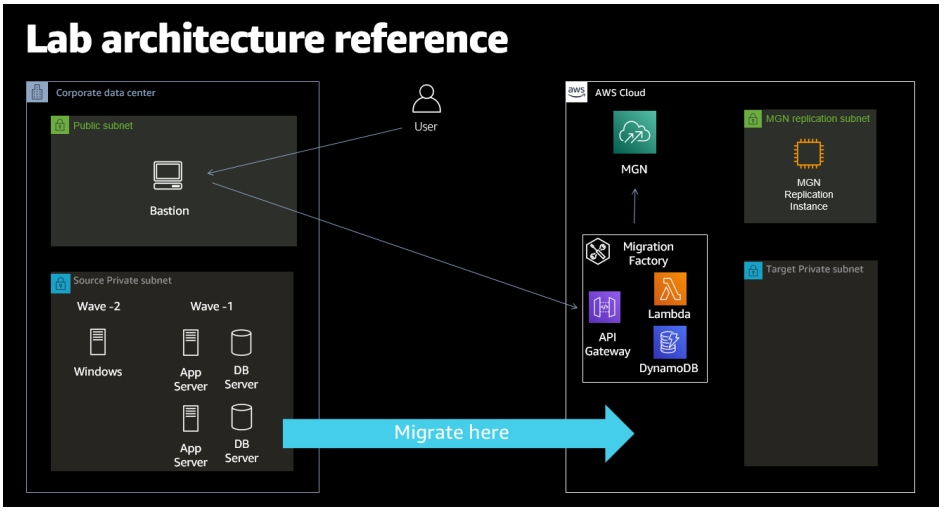
In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

Scenario

In this workshop we will perform migration of an emulated production environment running on a customer datacenter with AWS EC2 servers, configured in a way that mimics a customers' environment, with private and public networks, DNS server, pre-installed applications and a bastion host. All the concepts applied in this migration exercise can also be applied to 10s, 100s or 1000s of server migrations.

The following architecture reference will be used:



- The current production environment running on-premise is comprised of 3 applications and 5 servers as per the following:

Application	Hostname	FQDN	OS	Platform
Wordpress	wordpress-web	wordpress-web.onpremsim.env	Centos7	Apache+PHP
Wordpress	wordpress-d	wordpress-db.onpremsim.env	Centos7	MariaDB
OFBiz	ofbiz-web	ofbiz-web.onpremsim.env	Centos7	Java
OFBiz	ofbiz-db	ofbiz-db.onpremsim.env	Centos7	PostgreSQL
Intranet	windows	windows.onpremsim.env	Windows 2019	IIS

Task 1: Connect to the bastion host

In this task, you will connect to the bastion host instance using AWS Systems Manager Fleet Manager.

! Please make sure that you are using the **Chrome** as the Internet browser because for Fleet Manager RDP, only **Chrome** browser supports bidirectional copying and pasting between RDP sessions and your local machine.

If you are unable to use **RDP** with **Fleet Manager**, you can also connect to your windows instance using a Remote Desktop client.

To connect to the bastion host using RDP with Fleet Manager:

3. To the left of the instructions you are currently reading, choose **Download PEM**.
4. Save the file to the directory of your choice.
5. Get the **BastionInstanceSessionRDP** values from the left side of these instructions.
6. Open **Google Chrome** on your local computer, and access **BastionInstanceSessionRDP** from the browser.
7. For preferred **Authentication type** choose Key pair.
8. For **Key pair content**, choose the following option Browse your local machine to select the key pair file.
9. Select Browse to upload the **PEM** key from your local directory that is associated with your instance.
10. Select Connect .

You will be prompted with a Networks pop-up window asking: **Do you want to allow your PC to be discoverable by other PCs and devices on this network?** Choose **No**.

Your bastion host has all the required software to run this lab and should look like this:

All the commands listed on this guide should be executed from inside the bastion host.

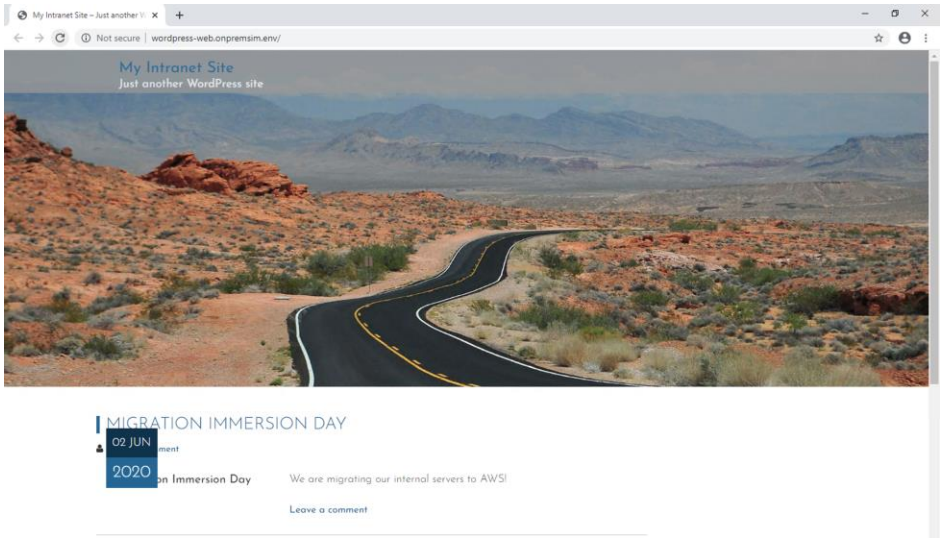
! Fleet Manager RDP connections have a maximum session duration of 60 minutes. When that duration is reached, Fleet Manager disconnects the session. If you run into any issues while interacting with the Fleet Manager RDP, then choose Actions ▼ drop-down list, and then select **Renew session** to restart the duration timer.

11. **Copy/Paste:** Test the applications that we're going to migrate. On the bastion host, open the following URL using Chrome browser.

Application	URL
Wordpress	**http://wordpress-web.onpremsim.env/**
OFBiz ERP	**https://ofbiz-web.onpremsim.env:8443/accounting**
Intranet	**http://windows.onpremsim.env/`**

- This is a simple test, just check if the application's webpage shows up. You should be able to access these 3 web applications:

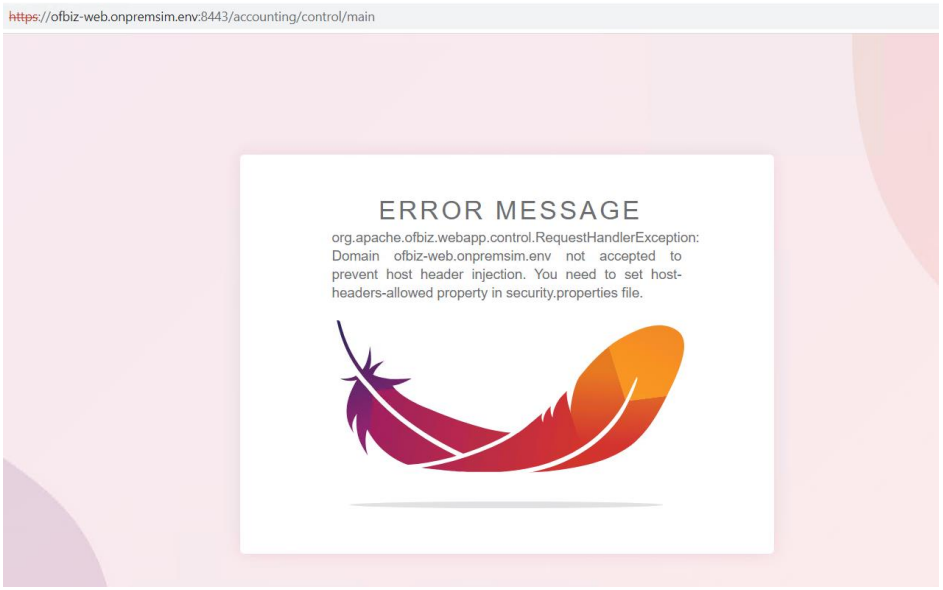
Wordpress



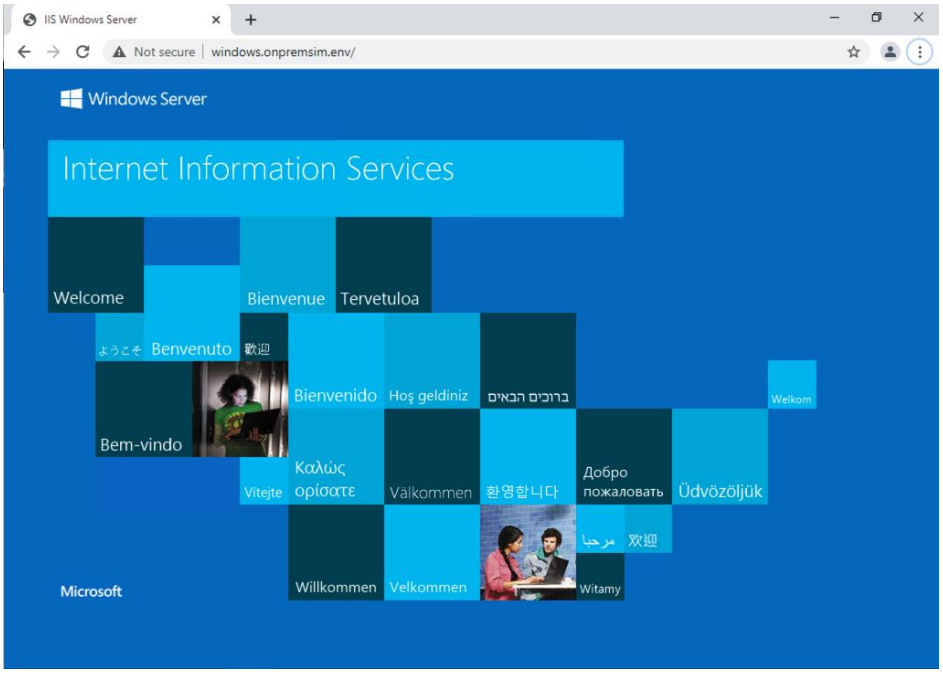
OFBiz

OFBiz application uses a self signed certificate. It is required to add the exception on Chrome to be able to explore the application. In the Chrome browser, select on **Advanced**, then **Proceed to ofbiz-web.onpremsim.env (unsafe)**. If you saw the following **Error** page for

OFBiz app, don't panic, this is because there were some build errors of this lab, you can still continue the lab without any issue.



Intranet



Task 2: Configure migration tools

In this step, we will initialize the AWS Application Migration Service (MGN), configure Migration Factory and import data to the factory. We will use Application Migration Service (MGN) with Migration Factory to migrate your servers, now let's initialize the MGN service and create the replication template.

Task 2.1: Initialize AWS Application Migration Service

12. At the top of the AWS Management Console, to the right of Services menu, in the search bar, search for **`MGN`** and then choose **AWS Application Migration Service** from the list.

! This is the AWS console for your lab account, not your regular AWS account. If you don't have the AWS console open, please refer to the previous step Connect to the lab environment.

13. Choose Get started .

14. In the Set up Application Migration Service page, choose Set up service .

Close the Failed to set up Application Migration Service error message if it appears. The services that you need for this lab have provisioned successfully.

- Every time you add a source server to Application Migration Service, its Replication settings, Launch settings and Post-launch action settings are initialized based on default templates. You can edit the default templates in the Settings page.
- The next step to setting up Application Migration Service is adding your source servers by installing the AWS Replication agent on them.

15. Under ▼ **Settings** in the left-hand navigation menu, select Replication template and then choose Edit .

You might have to choose the menu ☰ icon to expand the left navigation pane.

16. On **Edit replication template** page, under **Replication server configuration**, select ▼ `TargetPublic` as **Staging area subnet**, leave everything else as default value.

The staging area subnet is the subnet within which replication servers and conversion servers are launched. By default, Application Migration Service will use the default subnet on your AWS Account.

17. Choose Save template button at the button of the page.

ⓘ This will spin off a t3.small EC2 instance which is the replication server used by the Application Migration Service.

Task 2.2: Configure Migration Factory

To accelerate a large scale migration project, a key component is establishing a “migration factory” composed of people, tools and processes to streamline the movement of workloads from on-premises to AWS. The migration factory teams work through a prioritized backlog of workloads based on migration patterns identified in the portfolio discovery and planning process. With this approach, you will quickly start to achieve the business benefits of lower operating costs and gaining agility and scalability. In this lab, we will use AWS CloudEndure Migration Factory Solution

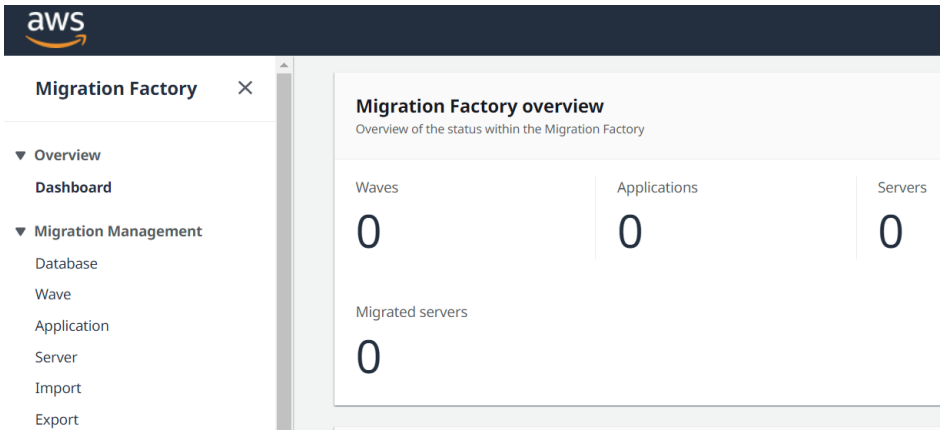
- **Login to the Migration factory console**

18. Get the **Migration Factory URL** and **Migration Factory UserName**, **Migration Factory Password** values from the left side of these instructions.

19. On the bastion host, open **Google Chrome**, and access **Migration Factory URL** from the browser. Enter your factory username and password value from the left side of these instructions to login:

- For Migration Factory **URL**, select **MigrationFactoryURL**.
- For Migration Factory **Username**, select **FactoryUserName**.
- For Migration Factory **Password**, select **FactoryPassword**.
- Choose Login .

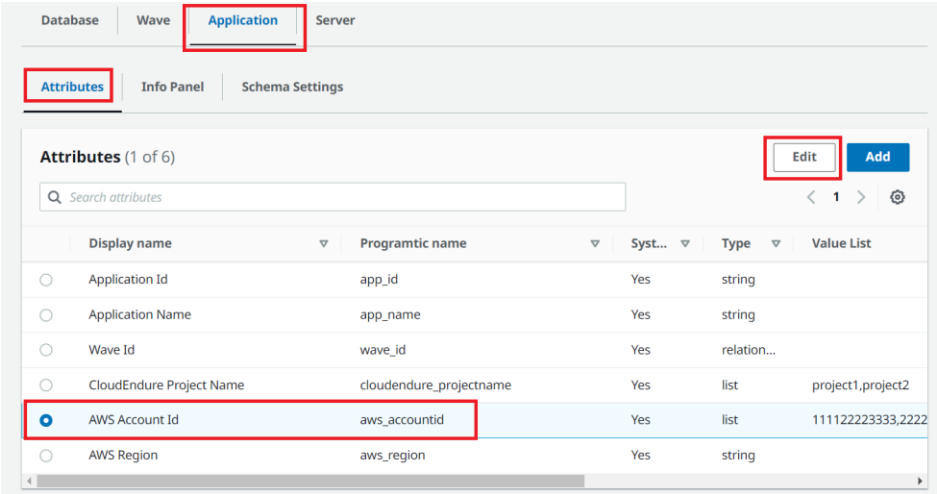
After logging, you should see a factory home page. Now let’s move to the next step to update factory schema.



- **Update Migration Factory schema to add new target AWS account Id**

20. On the left hand side menu, select **Attributes** under ▼ **Administration**.

21. On the **Attributes** Configuration page, switch to **Application** tab. Select **AWS Account Id**, then choose Edit button.

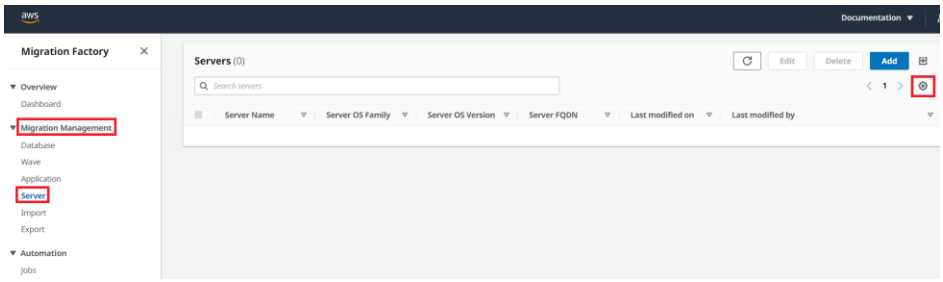


22. On the **Amend attribute** page, replace **Value list** default account Id with **AWSAccountId** value from the left side of these instructions, and choose Save .

! Please use the correct AWS Account Id value from the left side of these instructions, the Id should be 12 digits. For example, 432143214321.

Update server list preferences to show additional columns

23. On the left hand side menu, choose **Server** under ▼ **Migration Management**, then select Settings button (gear) on the right hand side.



24. Enable **Migration Status** and **Replication Status**, then choose Confirm to save the preference.

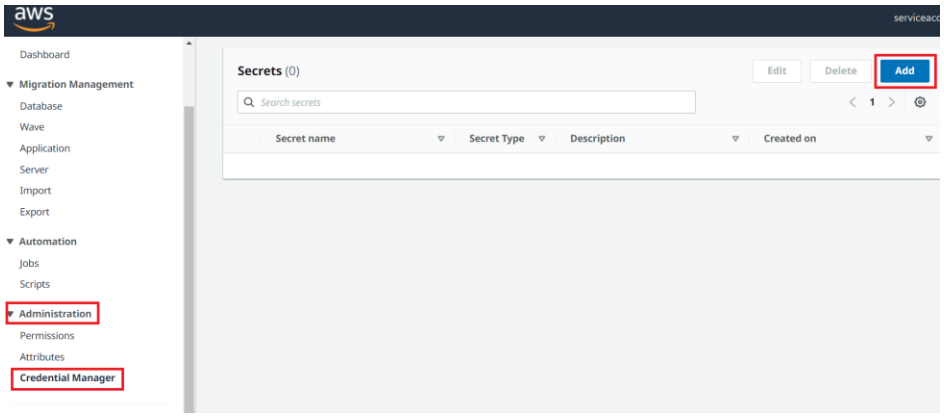
ⓘ On this lab, we're using Migration Factory as a centralized place to organize and track the migration progress. The intention is to have a dashboard which can help to consolidate all the migration information as a single source of truth.

Task 2.3: Configure source server credentials

Part of automation steps, you will need to use the admin user to connect to the source.

- **Create source Linux and Windows servers users**

25. On the factory console, select **Credential Manager** under ▼ **Administration** on the left hand side menu, then choose Add button on the right hand side.



26. In the Create secret page configure the following:

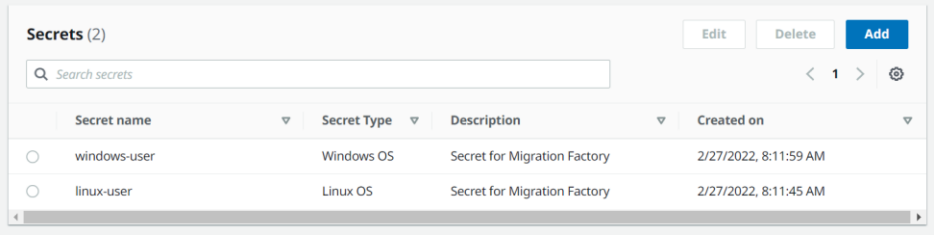
- For **Secret Type**, select **OS Credentials (Username/Password)**
- For **Secret Name**, enter **`linux-user`**
- For **User Name**, enter **`user`**
- For **Password**, enter the **RootPassword** value from the left side of these instructions.
- For **OS Type**, select **Linux**
- Finally choose Save

⚠ Make sure no space in the Secrets Name, otherwise you might get a bad credentials error.

ⓘ **Optional** - If you plan to migrate Wave 2 Windows server, create a Windows credential as well:

- Choose Add button on the right hand side.
- For **Secret Type**, select **OS Credentials (Username/Password)**
- For **Secret Name**, enter **`windows-user`**
- For **User Name**, enter **`Administrator`**
- For **Password**, enter the **RootPassword** value from the left side of these instructions.
- For **OS Type**, select **Windows**
- Finally choose Save

After that, you should see a list of credentials like below:

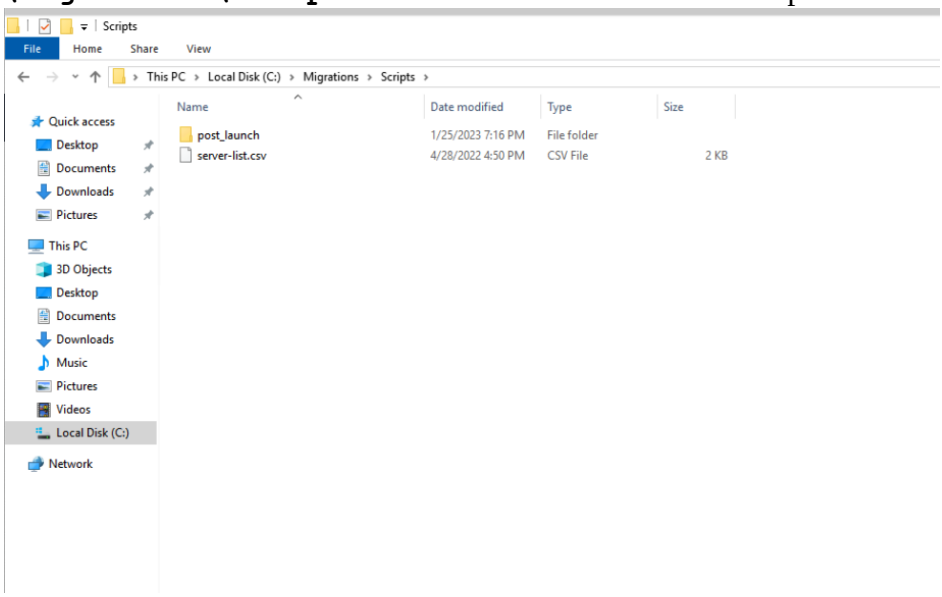


Task 2.4: Import servers to the factory

- **Update server list with target instance values**

! Make sure you are on the **bastion host** not your own laptop.

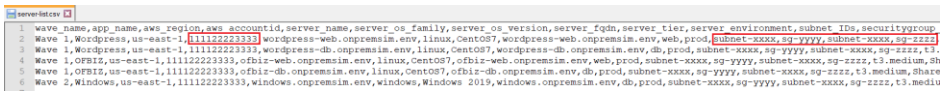
27. Open **`c:\Migrations\Scripts`** folder on the **bastion host** and update **server-list.csv**.



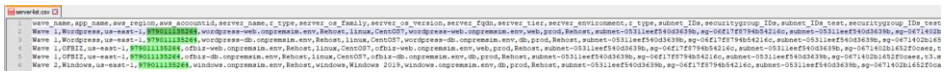
Edit **server-list.csv** using any preferred text editor on the bastion host.

28. Update rows 2 to 6 with the following fields using the information collected from the left side of these instructions.

Server List Form Key	Current Value	AWS Labs dashboard key
aws_accountid	111122223333	AWSAccountId
subnet_IDs	subnet-xxxx	SubnetTargetPrivate
securitygroup_IDs	sg-yyyy	TargetSecurityGroup
subnet_IDs_test	subnet-xxxx	SubnetTargetPrivate
securitygroup_IDs_test	sg-zzzz	TargetSecurityGroupTest



- Repeat the process for all the servers listed in the configuration file.



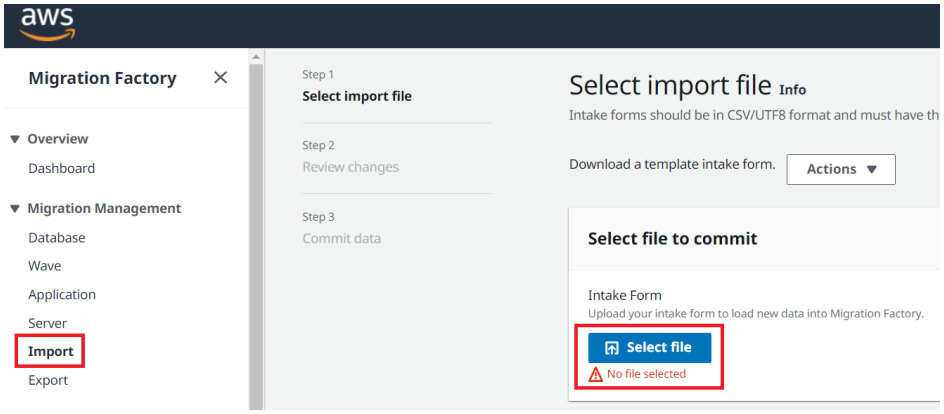
ⓘ If you prefer, try find/replace the following values: 111122223333 by Target AWS Account Id value, subnet-xxxx by SubnetTargetPrivate value, sg-yyyy by TargetSecurityGroup value and sg-zzzz by targetSecurityGroupTest value. We use the same values for subnet_IDs and subnet_IDs_test for the purpose of this lab. For a real migration, it is recommended to have a subnet for the test migration environment and a subnet for the target production environment.

- **Import server list to the migration factory**

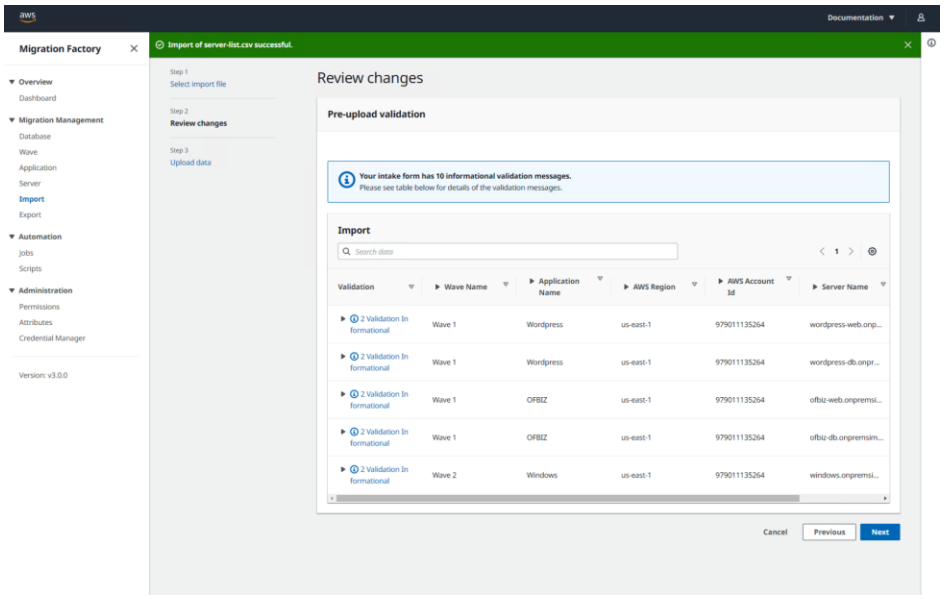
29. Login to the **Migration Factory** web console.

30. Under ▼ **Migration Management** select **Import** on the left hand side, and choose **Select file** button, select the **`c:\Migrations\Scripts\server-list.csv`** file that you updated in the previous step.

31. Choose Next .



32. Review the changes and make sure you do not see any errors (Information message is normal), and choose Next .



33. Finally, choose Upload . You should see a success message **Intake file upload completed successfully**.

ⓘ Group applications in migration waves. That helps to organize the cutover window for multiple applications. On this lab we have 2 migration waves composed by 2 Applications and 5 servers.

Task 3: Automate migration tasks

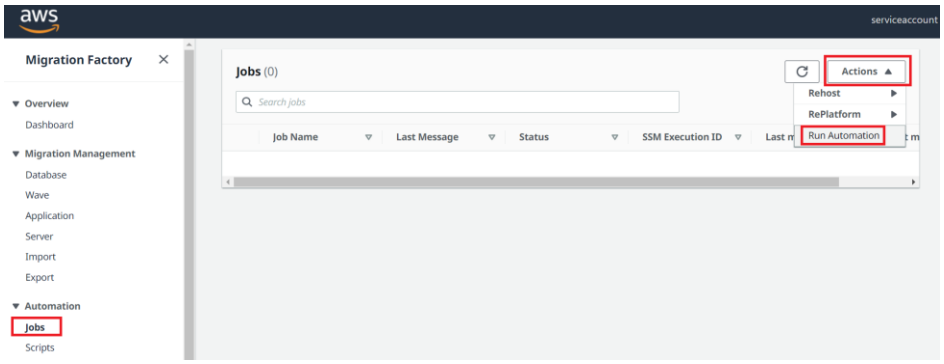
On this task we will automate:

- Check pre-requisites
- Install replication Agents
- Copy post launch scripts
- Verify replication status
- Validate Launch template
- Mark as ready for cutover
- Shutdown source
- Launch cutover instances
- Verify cutover instances status
- Test application

Task 3.1: Check MGN prerequisites

- **Check MGN Prerequisites for all servers in Wave 1**

34. On the **Migration Factory console**, choose **Jobs** on the left hand side menu, and then choose **Actions ▼** and select **Run Automation** on the right hand side.



35. In the **Run Automation Attributes** page configure the following:

    a. For **Job Name**, enter `Check pre-requisites`

    b. For **Automation Server**, select **automation-server.WORKGROUP**

    c. For **Script Name**, select **0-Check MGN Prerequisites**

    d. For **Linux Secret**, select `linux-user`

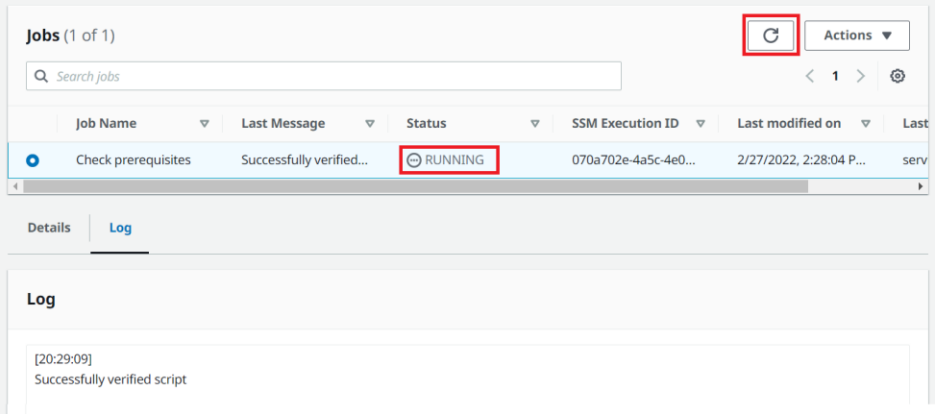
e. For **Replication Server IP**, enter `52.5.16.99`

f. For **Wave Name**, select **Wave 1**

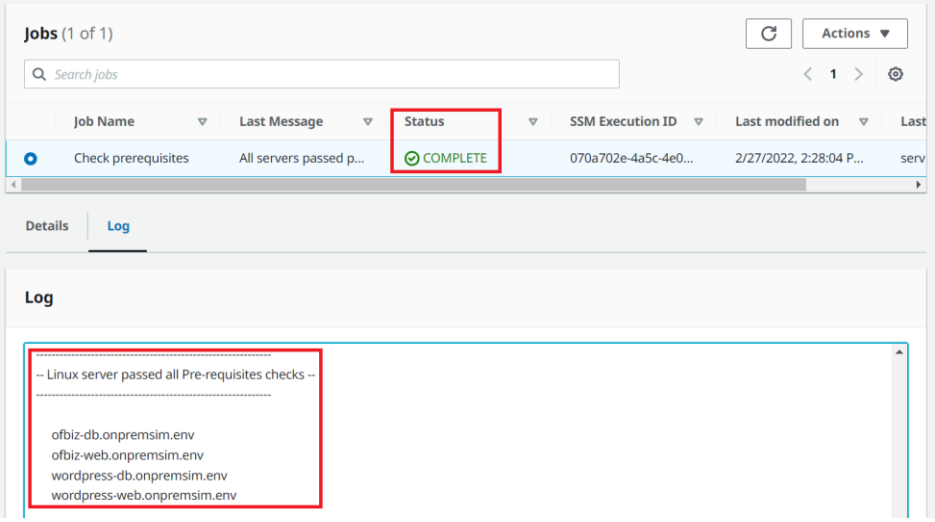
g. Scroll to the bottom of the page and choose **Submit Automation Job**

ⓘ For this **Replication Server IP**, we're providing a temporary server listening on port TCP 1500 just for firewall test purpose. No need to replace the IP address for this lab. On a real migration the IP should be the Replication server IP running in your account.

36. You will be redirected to the Job list page, the job status should be running, and you can select refresh button after about 1 minute to see the status.



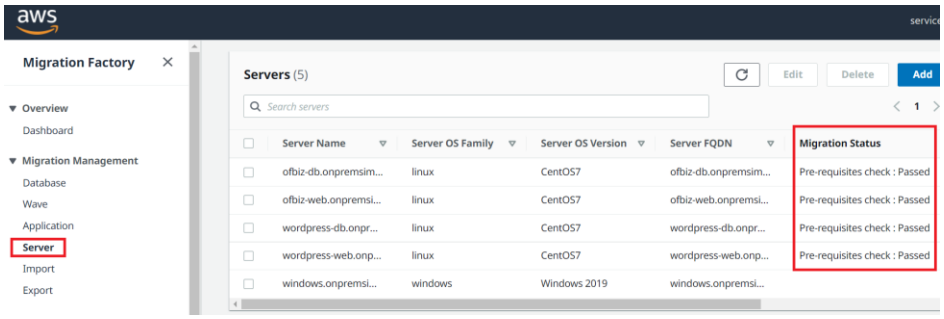
37. Finally, job will be completed. Job status should change to **COMPLETE**, and you can see the details in the Log section.



If the server failed one or more pre-requisites checks, please scroll up to check which one fail or check the detailed error message.

The script will also update migration factory migration status attribute.

Switch to server on the left hand side, you should see the migration status Pre-requisites check : Passed on the right hand side.

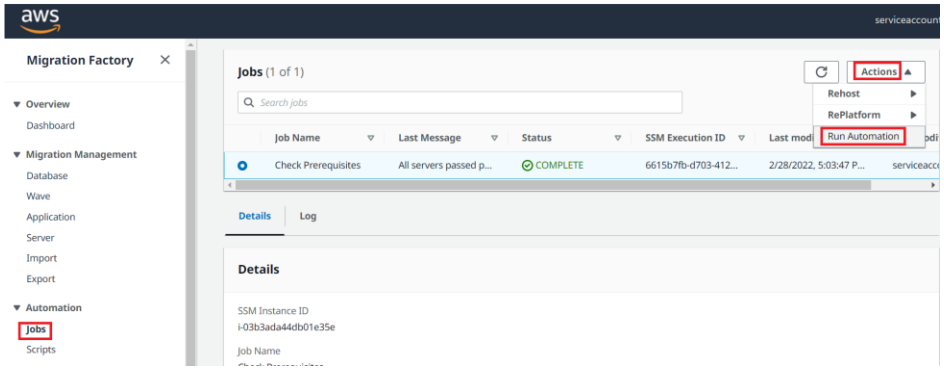


ⓘ There are pre-migration tasks such as installing software dependencies as Python, AWS SSM, checking firewall connectivity, free disk space, SSH or RDP service enabled, etc… make sure you can clear up all your migration dependencies before starting replicating servers

Task 3.2: Install replication agent

- **Install replication agent on all servers in Wave 1**

38. Same as previous step. On the **Migration Factory console**, choose **Jobs** on the left hand side menu, and then choose **Actions ▼** and select **Run Automation** on the right hand side.



39. In the **Run Automation Attributes** page configure the following:

    a. For **Job Name**, enter `Install MGN Agents`

    b. For **Automation Server**, select **automation-server.WORKGROUP**

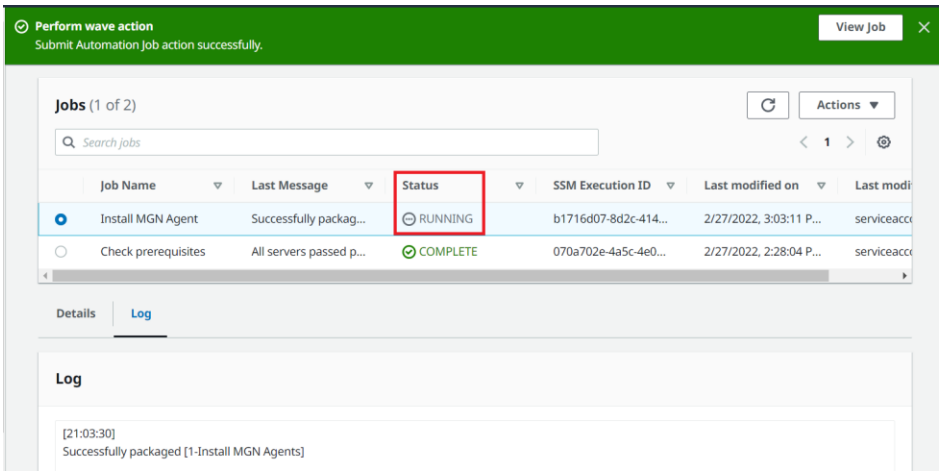
    c. For **Script Name**, select **1-Install MGN Agents**

    d. For **Linux Secret**, select `linux-user`

    e. For **Wave Name**, select **Wave 1**

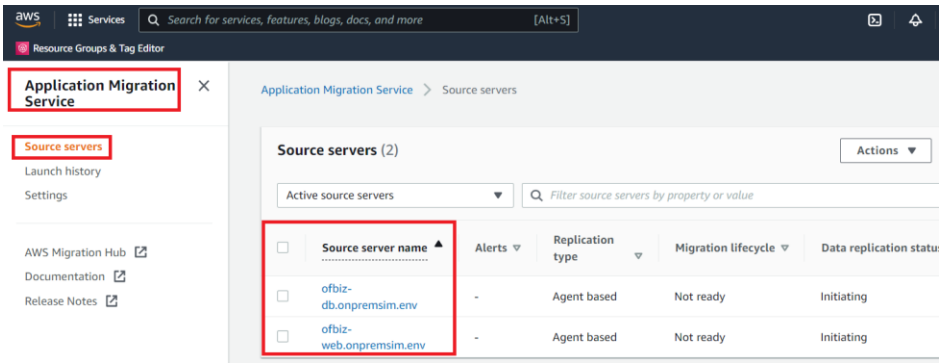
    f. Scroll to the bottom of the page and choose **Submit Automation Job** .

40. You will be redirected to the Job list page, the job status should be running, and you can select refresh button after about 1 minute to see the status.

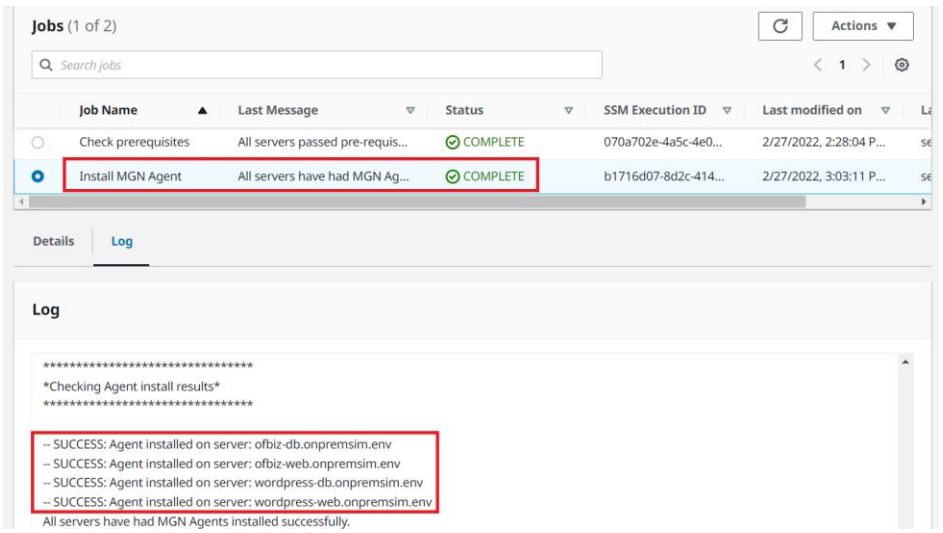


The entire script might take up to 10 minutes to finish agent installation, you might not see status update from the scripts until installation is complete

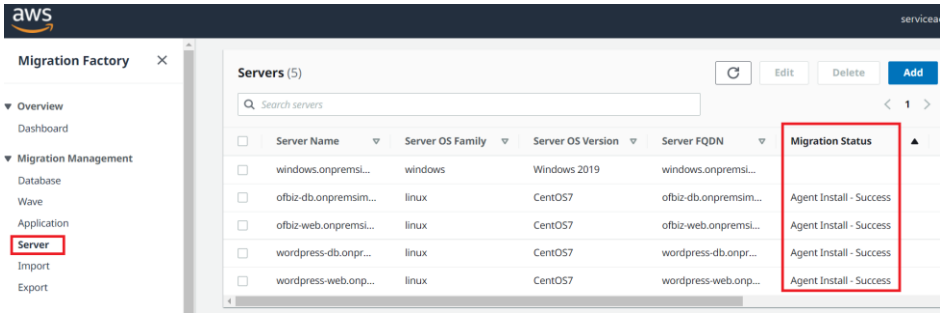
As an alternative to check real time status, you can login to AWS Console, switch to **AWS Application Migration Service**. Select Source Servers on the left hand side, you should see servers being added to the list one by one:



41. Finally, job will be completed. Job status should change to **COMPLETE**, and you can see the details in the Log section.



42. Migration factory should have also received status update from the script. If necessary, refresh screen.



- Right after installing Replication agents, the replication will start immediately.

Batch Replication agent installation aligned with the migration waves. Automate agent installation and give enough time in advance for the servers to finish the initial replication.

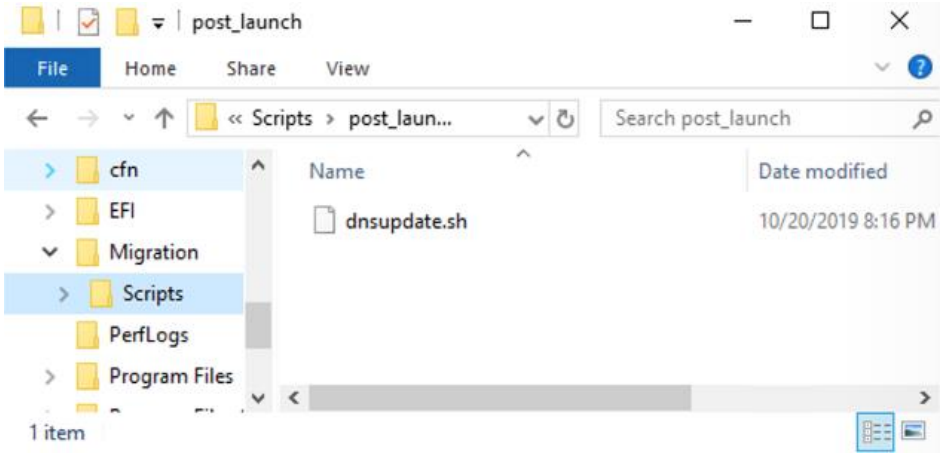
Task 3.3: Copy post launch script to all Servers

⚠ Please wait for MGN agent installation script in the previous step to complete installation before pushing post launch scripts (Data Replication might still in progress, which is Ok). Because the script need to grant permission to MGN service account, it will fail if the service account does not exist.

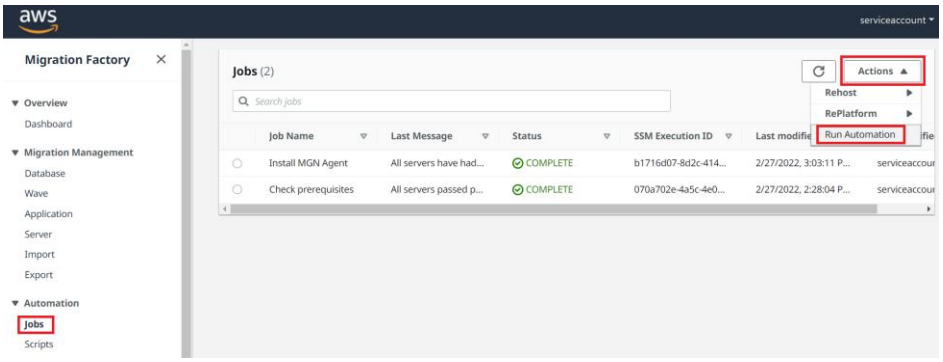
This task will copy the post launch scripts to the destination folder on Linux. The post launch script used on this lab will update DNS server to resolve the new IP once the servers are migrated (Source CIDR is **192.168.0.0/24** and Target AWS CIDR will be **10.0.1.0/24**).

MGN will trigger all the scripts located in the folder **/boot/post_launch** during the launching process.

43. Open the post launch scripts folder **`C:\migrations\Scripts\post_launch`**, make sure the test scripts exist.



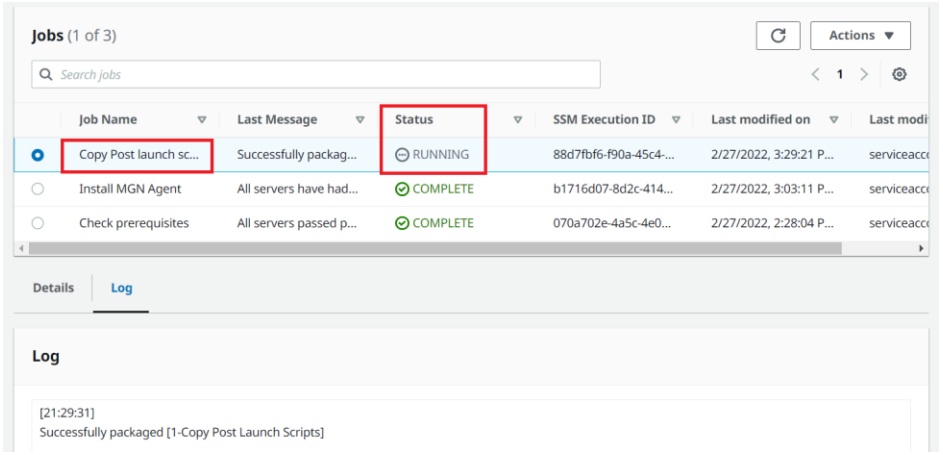
44. Same as previous step. On the **Migration Factory console**, choose **Jobs** on the left hand side menu, and then choose **Actions ▼** and select **Run Automation** on the right hand side.



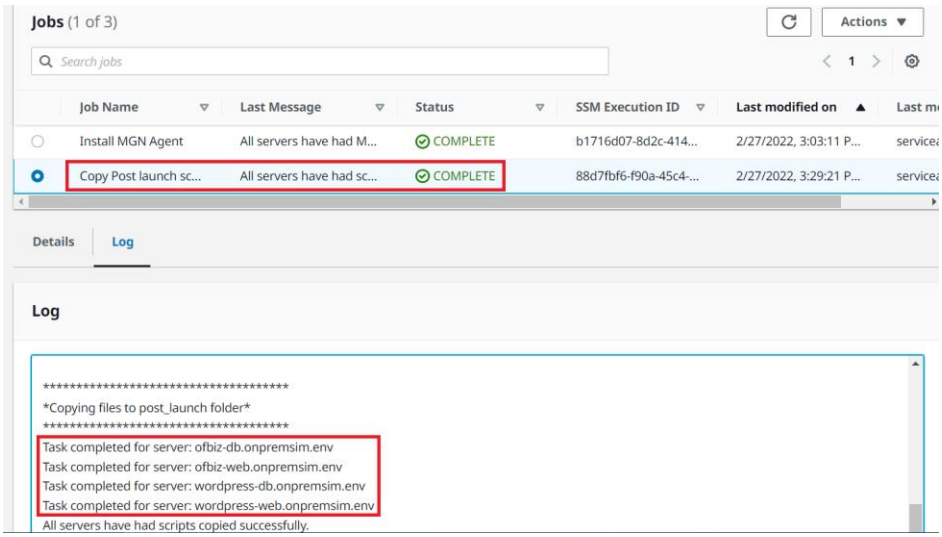
45. In the **Run Automation Attributes** page configure the following:

- For **Job Name**, enter `Copy Post Launch Scripts`
- For **Automation Server**, select **automation-server.WORKGROUP**
- For **Script Name**, select **1-Copy Post Launch Scripts**
- For **Linux Secret**, select `linux-user`
- For **Linux source location**, enter **`C:\migrations\Scripts\post_launch`**
- For **Wave Name**, select **Wave 1**
- Scroll to the bottom of the page and choose **Submit Automation Job**

46. You will be redirected to the Job list page, the job status should be running, and you can select refresh button after about 1 minute to see the status.



47. Finally, job will be completed. Job status should change to **COMPLETE**, and you can see the details in the Log section.

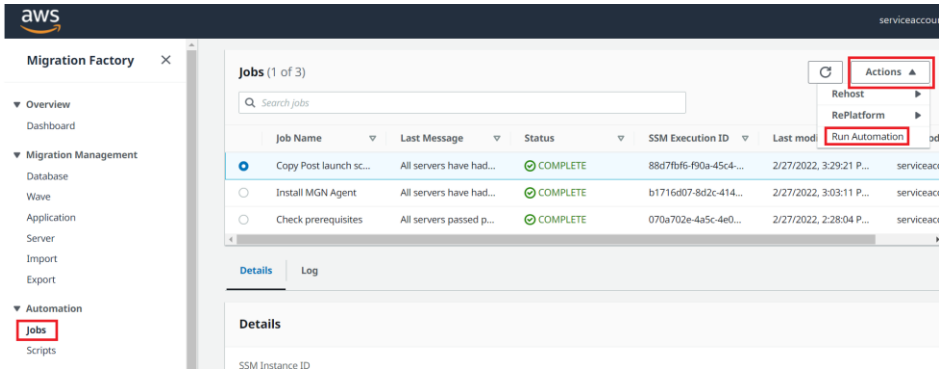


ⓘ Post launch script can be very helpful to automate cutover tasks as for e.g. renaming computers, updating DNS records, joining AD domains, installing or removing software, etc.

Task 3.4: Verify replication status

This task will verify the replication status for all servers in all MGN AWS accounts in a specific wave. Executing the following script will return an output on the log screen and update the replication status attribute in the migration factory as well. The expected replication status must be Healthy before launching servers for testing or cutover.

48. On the **Migration Factory console**, choose **Jobs** on the left hand side menu, and then choose **Actions ▼** and select **Run Automation** on the right hand side.



49. In the **Run Automation Attributes** page configure the following:

    a. For **Job Name**, enter `Verify replication status`

    b. For **Automation Server**, select **automation-server.WORKGROUP**

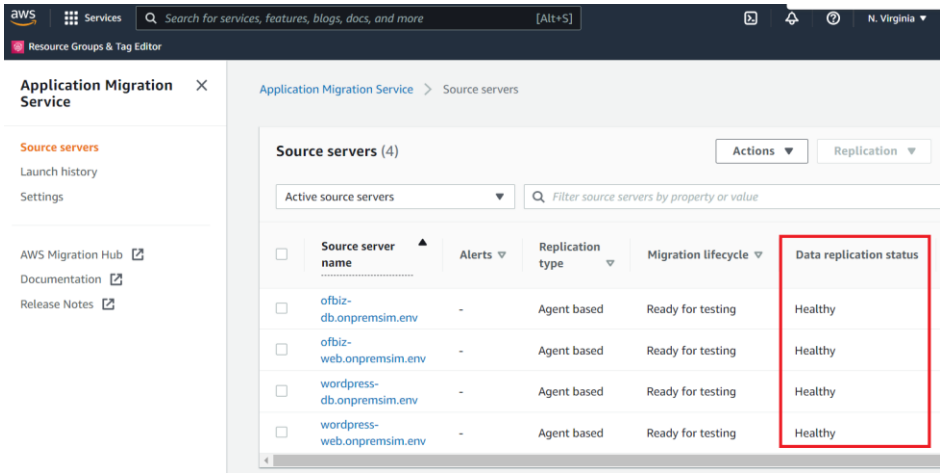
    c. For **Script Name**, select **2-Verify Replication Status**

    d. For **Wave Name**, select **Wave 1**

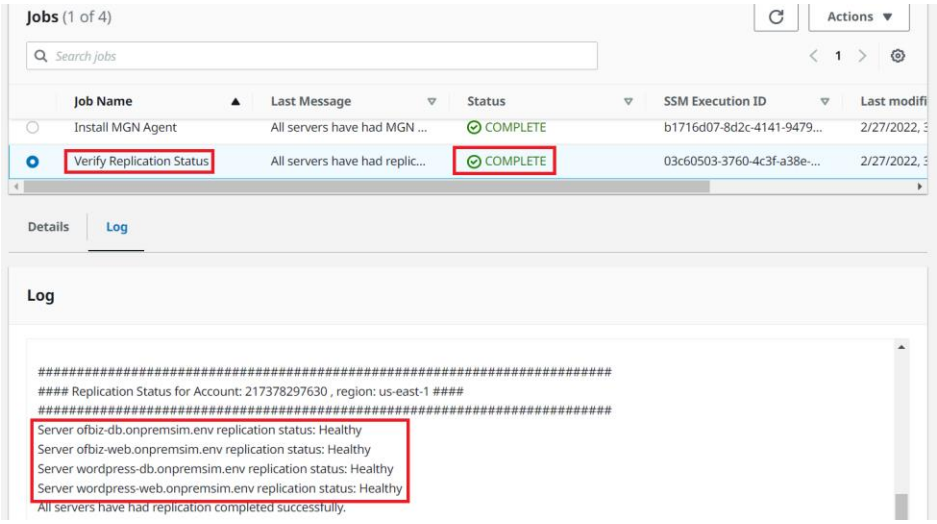
    e. Scroll to the bottom of the page and choose **Submit Automation Job**

The replication might take up to 30 minutes to complete, you might not see status update from the scripts until the replication is complete.The replication status for each server might be different. For example, status Initial sync means server has not finished initial replication, Initiating - (detailed stage) means the initial sync is preparing and has not started.

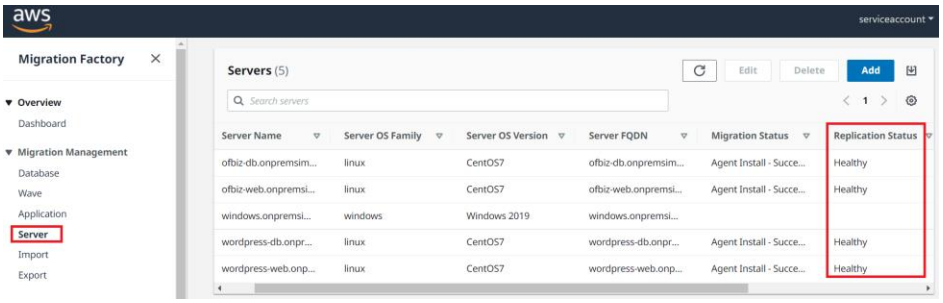
ⓘ As an alternative to check real time status, you can login to AWS Console, switch to AWS Application Migration Service. Select Source Servers on the left hand side, you should see the real time replication status, you need to wait for all servers to change to Healthy:



50. Finally, job will be completed. Job status should change to **COMPLETE**, and you can see the details in the Log section.



51. Migration factory should have also received status update from the script. If necessary, refresh screen.



⚠ It might take up to 30 minutes to replicate all servers, especially the last stage Creating Snapshot. You can proceed with 5-Validate Launch Template while waiting for the replication. But **DO NOT** shutdown the source servers.

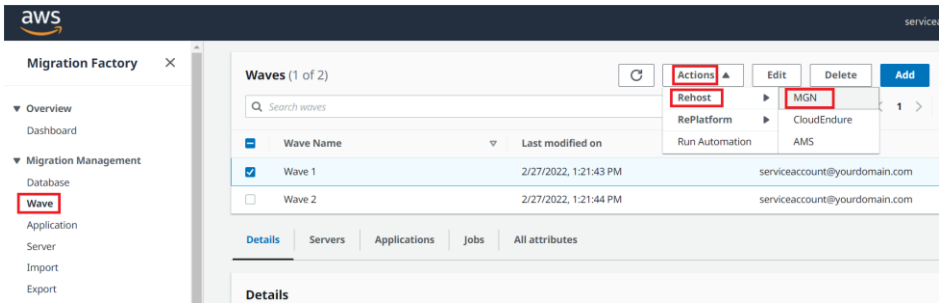
Task 3.5: Validate Launch template

- **Validate launch templates for all servers in Wave 1**

This task will validate the launch templates that will be used when instances are started in AWS. This will check that the supplied subnets, security groups, and other settings are correct.

52. On the **Migration Factory console**, select **Wave** on the left hand side menu.

53. Select **Wave 1**, and choose **Actions ▼** then select **Rehost** and finally select **MGN** .



54. In the **MGN Attributes** page configure the following:

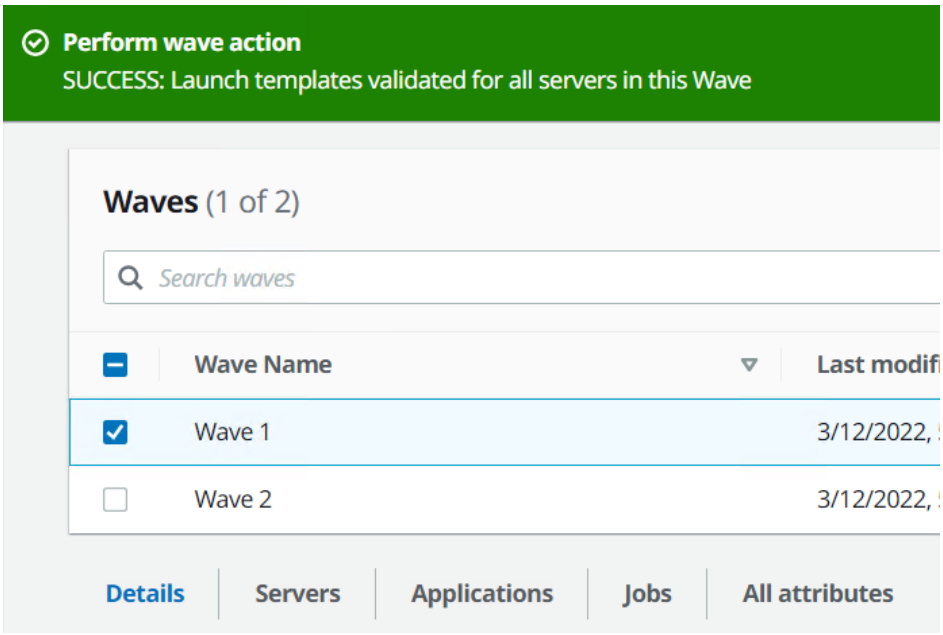
    a. For **Action**, select **Validate Launch Template**

    b. For **Wave Name**, select **Wave 1**

    c. For **Applications**, select **All**

    d. Choose **Submit** .

- After some time the validation will return a successful result.

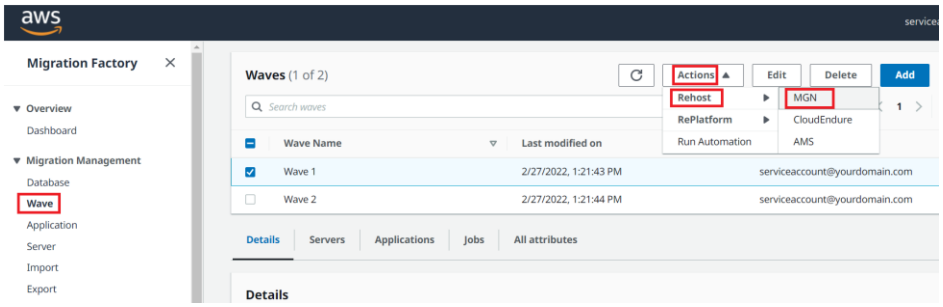


Task 3.6: Mark as ready for cutover

⚠ Please wait for MGN replication to finish before working on this step. Make sure replication status is Healthy for all servers. Replication may need 20 - 30 minutes to complete.

55. On the **Migration Factory console**, select **Wave** on the left hand side menu.

56. Select **Wave 1**, and choose **Actions ▼** then select **Rehost** and finally select **MGN** .



57. In the **MGN Attributes** page configure the following:

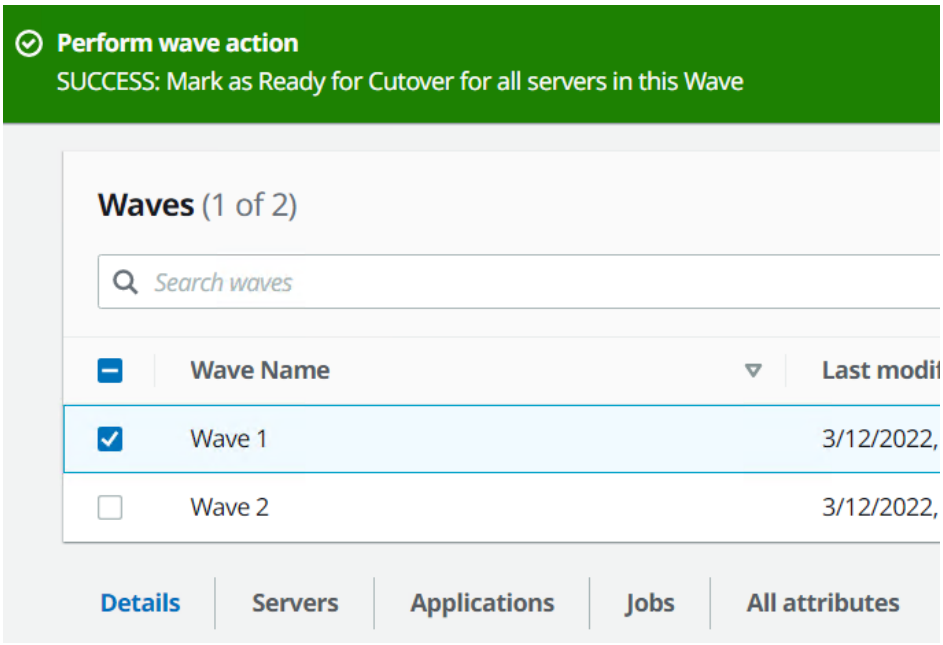
    a. For **Action**, select **Mark as Ready for Cutover**

    b. For **Wave Name**, select **Wave 1**

    c. For **Applications**, select **All**

    d. Choose **Submit** .

- After some time the mark as ready for cutover will return a successful result.



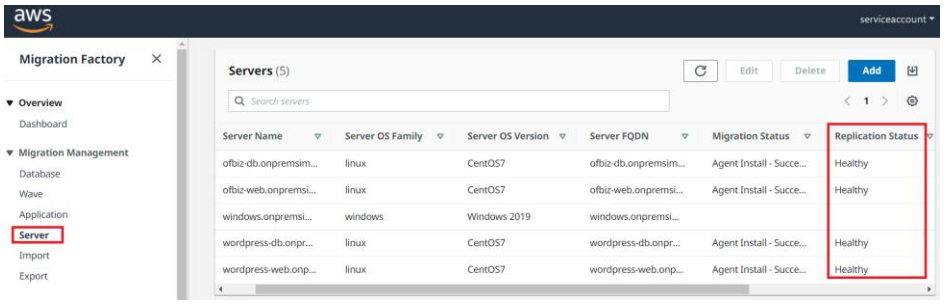
Task 3.7: Shutdown Source

- **Shutdown all servers in Wave 1**

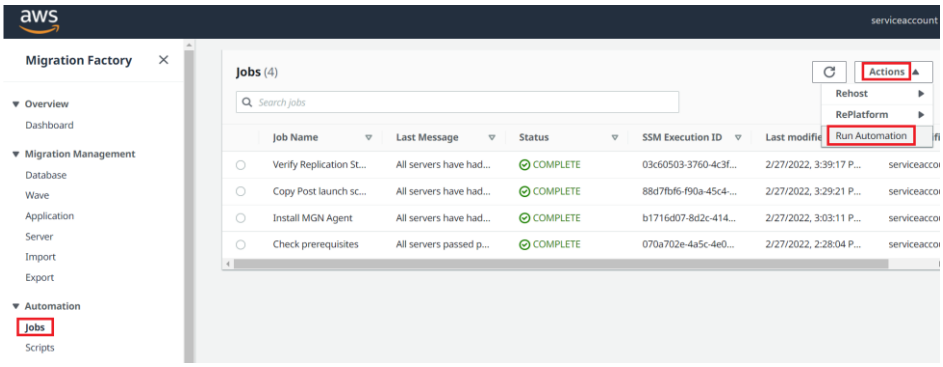
⚠ Please wait for MGN replication to finish before shutting down the servers. Make sure replication status is Healthy for all servers. Replication may need 30 minutes to complete.

This task will shut down the source servers before cutover. We want to make sure that client is not able to connect to the source server during cutover to ensure data consistency.

58. Make sure replication status is Healthy for all servers.



59. On the **Migration Factory console**, choose **Jobs** on the left hand side menu, and then choose **Actions ▼** and select **Run Automation** on the right hand side.



60. In the **Run Automation Attributes** page configure the following:

    a. For **Job Name**, enter `Shutdown All Servers`

    b. For **Automation Server**, select **automation-server.WORKGROUP**

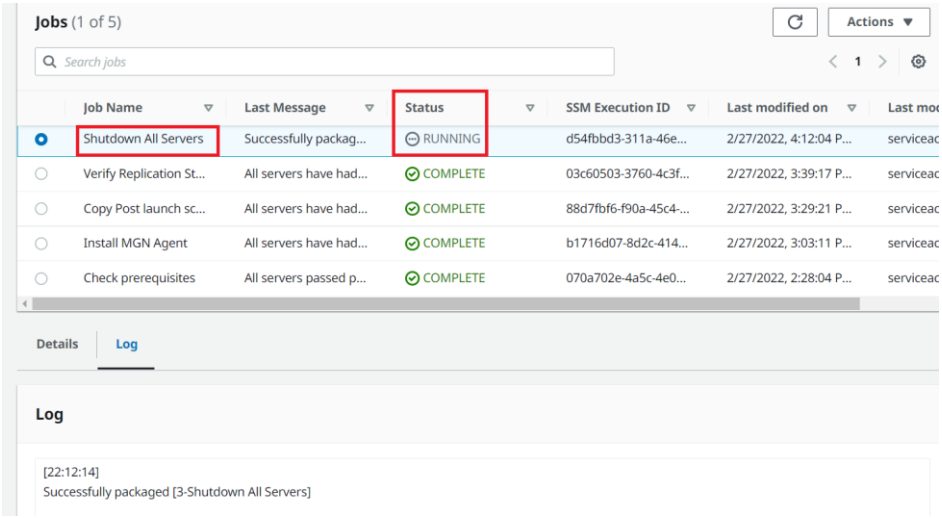
    c. For **Script Name**, select **3-Shutdown All Servers**

    d. For **Linux Secret**, select **`linux-user`**

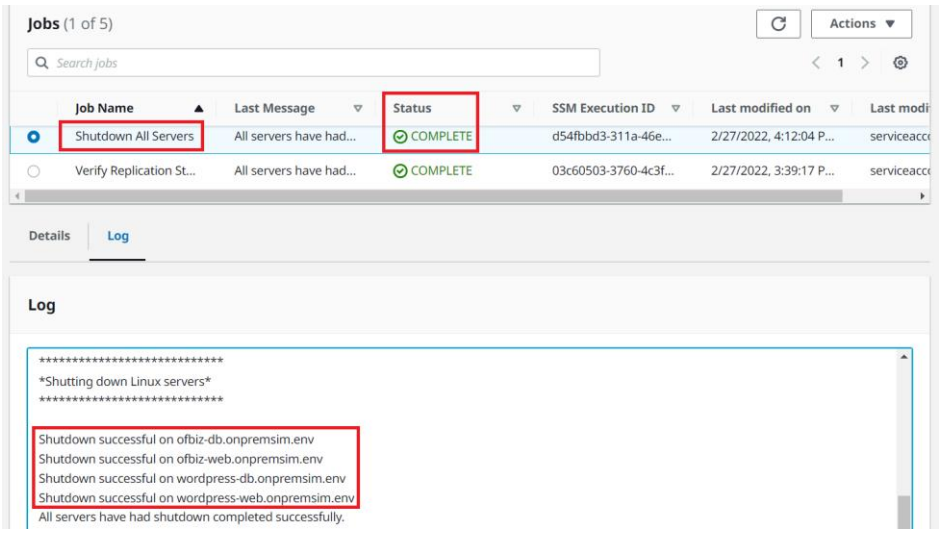
    e. For **Wave Name**, select **Wave 1**

    f. Scroll to the bottom of the page and choose **Submit Automation Job**

61. You will be redirected to the Job list page, the job status should be running, and you can select refresh button after about 1 minute to see the status.



62. Finally, job will be completed. Job status should change to **COMPLETE**, and you can see the details in the Log section.



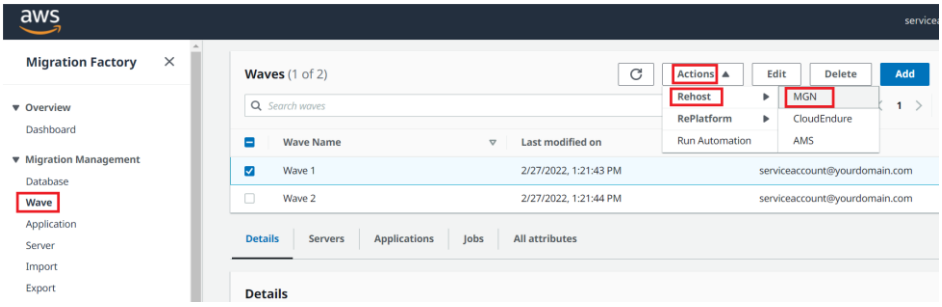
ⓘ Shutdown the server before cutover. Make sure that client is not able to connect to the source server during cutover to ensure data consistency. After shutting down the source servers, the status of source servers on **AWS Application Migration Service** console will change to stalled. This is normal because source server is no longer connecting to replication server, we want to stop the transaction on the source before cutover.

Task 3.8: Launch cutover instances

The following tasks will migrate all the servers in Wave 1 and apply the Cutover subnet and security groups specified in the intake form.

63. On the **Migration Factory console**, select **Wave** on the left hand side menu.

64. Select **Wave 1**, and choose **Actions ▼** then select **Rehost** and finally select **MGN** .



65. In the **MGN Attributes** page configure the following:

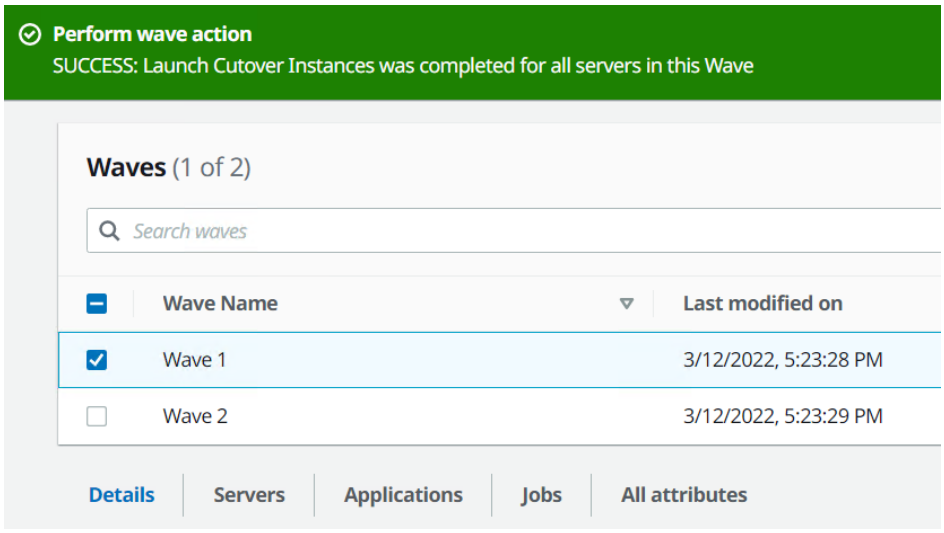
    a. For **Action**, select **Launch Cutover Instances**

    b. For **Wave Name**, select **Wave 1**

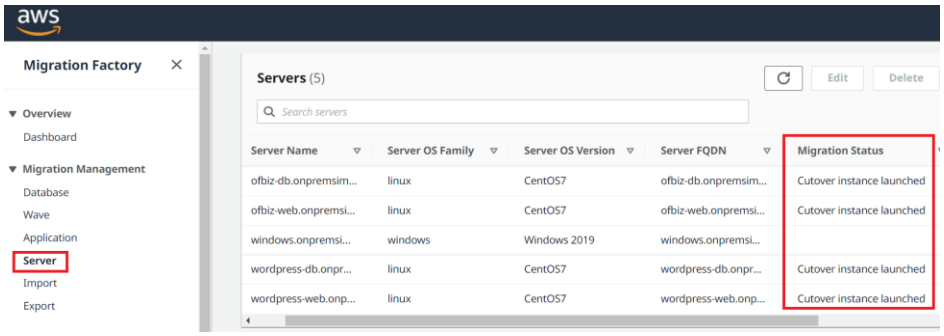
    c. For **Applications**, select **All**

    d. Choose **Submit** to initiate the launch of the instances into the Target subnets and security groups.

- After sometime the launch will return a success message.



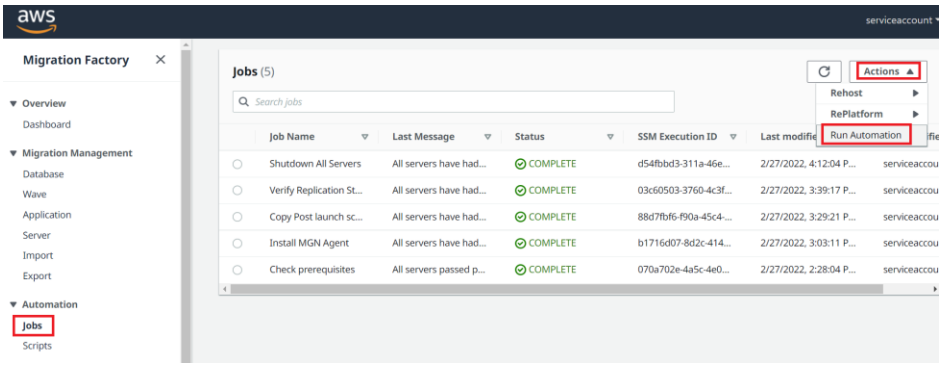
66. The status of the servers will be updated in Migration Factory. The current migration status should be listed on the screen and show Cutover instance launched.



Task 3.9: Verify Cutover Instances Status

Amazon EC2 servers have a control system to determine if the OS is running and responsive. That's called Instance Status Check. The following script can be used to determine if all servers that have been converted to EC2 successfully and have booted and are responsive.

67. On the **Migration Factory console**, choose **Jobs** on the left hand side menu, and then choose **Actions ▼** and select **Run Automation** on the right hand side.



68. In the **Run Automation Attributes** page configure the following:

    a. For **Job Name**, enter `Verify Instance Status`

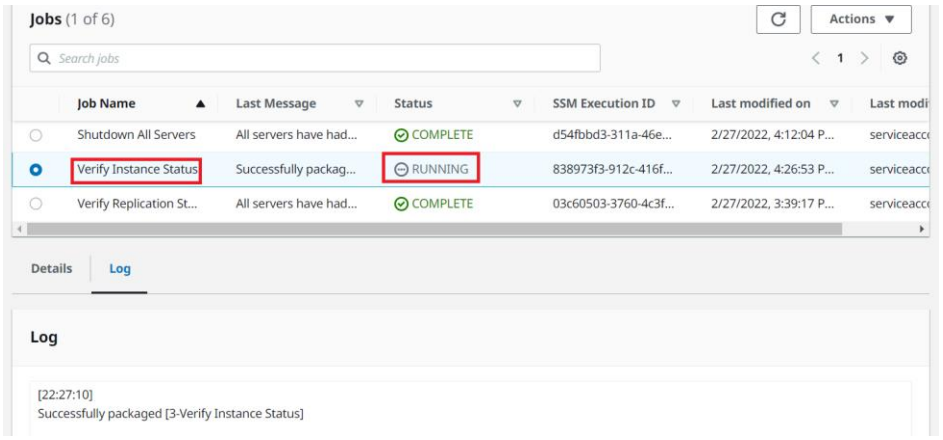
    b. For **Automation Server**, select **automation-server.WORKGROUP**

    c. For **Script Name**, select **3-Verify Instance Status**

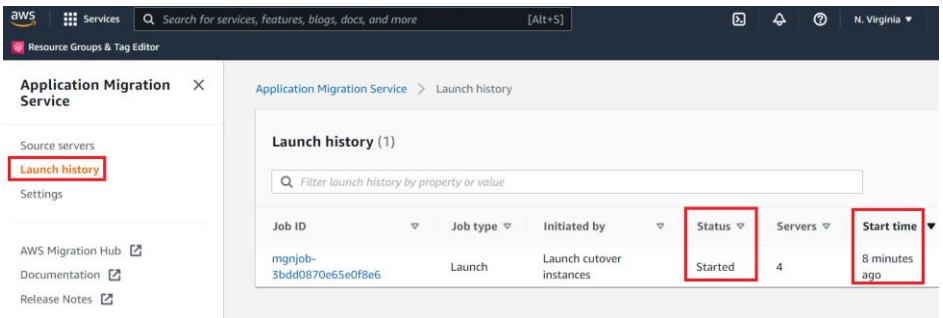
    d. For **Wave Name**, select **Wave 1**

    e. Choose **Submit Automation Job** .

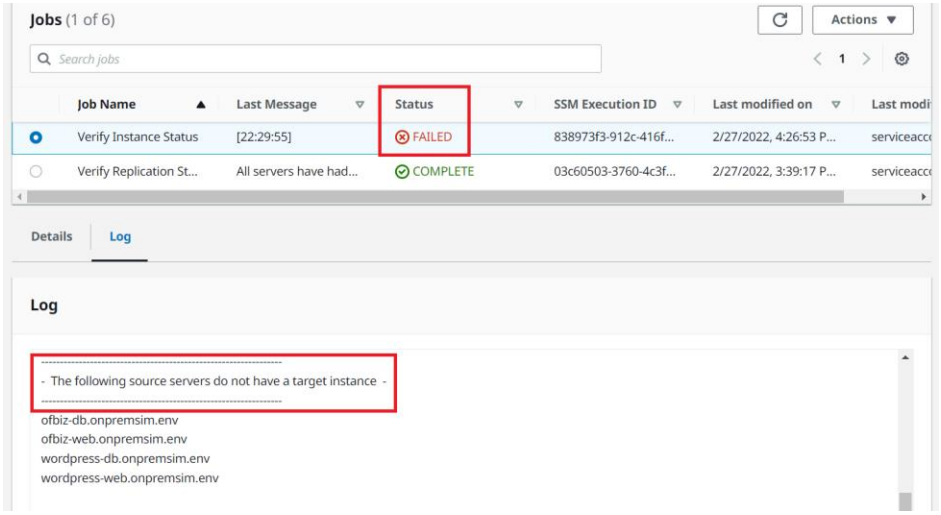
69. You will be redirected to the Job list page, the job status should be running, and you can select refresh button after about 1 minute to see the status.



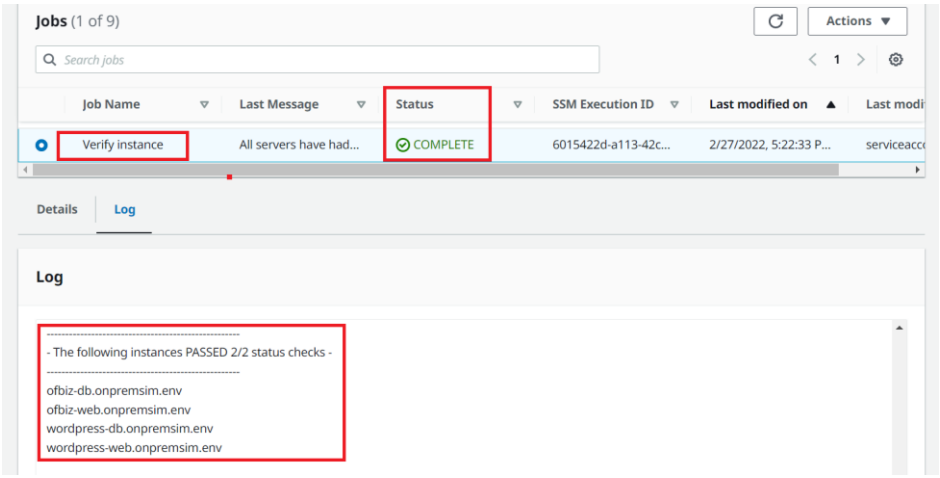
- As an alternative to check real time status, you can login to AWS Console, switch to AWS Application Migration Service. Select Launch History on the left hand side, you can see the real time job status, you need to wait for the status to change to Completed:



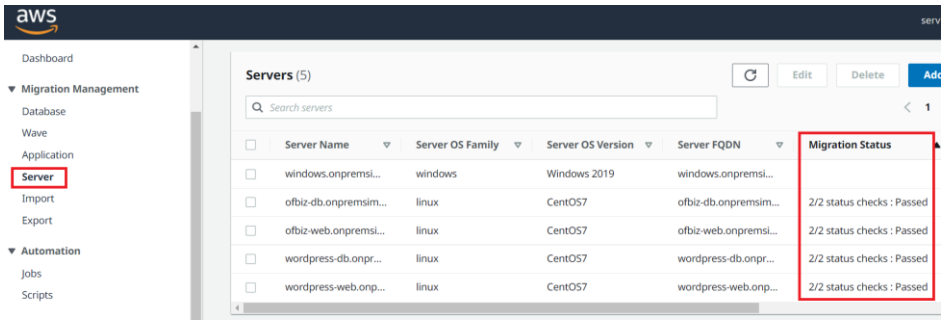
ⓘ The Target instance might take up to 30 minutes to complete conversion and change the status to 2/2 Passed, you might not see status update from the scripts until the replication is complete. If the job failed with do not have a target instance, wait for 5 minutes and try again. This error means MGN job is still in progress, target instance hasn't been created yet.



70. Finally, job will be completed. Job status should change to **COMPLETE**, and you can see the details in the Log section.



71. Migration factory should have also received status update from the script. If necessary, refresh screen.



Task 3.10: Test Application

72. Test the applications post migration. Open the following URL using Chrome from the bastion host.

Application	URL
Wordpress	http://wordpress-web.onpremsim.env/
OFBiz ERP	https://ofbiz-web.onpremsim.env:8443/accounting

- This is a simple test, just check if both application webpages show up.

73. Open CMD.EXE (Windows Command Prompt) and ping all the servers below. Please notice they are located now in the target cloud VPC using the CIDR 10.0.1.0/24

Hostname
wordpress-web
wordpress-db
ofbiz-web
ofbiz-db

- If any of the apps does not work, and you are still getting 192.168.1.x IP address, this is usually because the post launch script not executed successfully. To fix this, use the following steps:

Do the following 4 steps only if the app does not work and IP address is still 192.168.1.x

- Go to EC2 console, find the new 10.0.1.X IP address for the problematic instance.
- SSH to the target instance using Putty on the desktop.
- Login to the target instance with the same Linux user name and password.

Username	Password
user	AWSmid23

- Run the following command to update DNS server:

```
#!/bin/sh
ADDR=`hostname -I`
HOST=`hostname`
sudo touch /tmp/nsupdate.txt
sudo chmod 666 /tmp/nsupdate.txt
echo "server dns.onpremsim.env" > /tmp/nsupdate.txt
echo "update delete $HOST A" >> /tmp/nsupdate.txt
echo "update add $HOST 86400 A $ADDR" >> /tmp/nsupdate.txt
echo "update delete $HOST PTR" >> /tmp/nsupdate.txt
```

```
echo "update add $HOST 86400 PTR $ADDR" >> /tmp/nsupdate.txt
echo "send" >> /tmp/nsupdate.txt
sudo nsupdate /tmp/nsupdate.txt
```

CONGRATULATIONS!

At this point, you have successfully completed all the standard steps in this lab.

End lab

Follow these steps to close the console and end your lab.

74. Return to the **AWS Management Console**.

75. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.

76. Choose End lab and then confirm that you want to end your lab.

Additional Resources

- For more information about how to use Cloud Migration Factory on AWS, see Cloud Migration Factory on AWS Documentation.

Connect to your Windows instance using RDP

- Copy the **BastionInstance** value from the left side of these instructions.
- To the left of the instructions you are currently reading, choose **Download PEM**.
- Save the file to the directory of your choice.
- Open the Amazon EC2 console
- In the navigation pane, select **Instances**. Select the **BastionInstance** instance and then choose **Connect**.
- On the **Connect to instance** page, choose the **RDP client** tab, and then choose **Get password**.
- Choose **Browse** and navigate to the private key (**.pem**) file you created earlier. Select the file and choose **Open** to copy the entire contents of the file to this window.
- Choose **Decrypt Password**. The console displays the default administrator password for the instance under **Password**, replacing the **Get password** link shown previously. Save the password in a safe place. This password is required to connect to the instance.

For your RDP client, use the following details to connect:

- **Bastion Host IP:** Enter the **BastionRDP** value from the left side of these instructions.
- **Username:** Enter **`Administrator`**.
- **Password:** Copy and paste the **password** that you saved previously.

Return to the instructions

For more information about AWS Training and Certification, see *https://aws.amazon.com/training/*.

Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.

Refactoring Legacy Apps to Microservices using AWS Migration Hub Refactor Spaces

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab Overview

This lab demonstrates the process of moving from a monolithic architecture to a microservice architecture. You will explore an existing monolithic application named Unishop and deploy its shopping cart functionality as separate microservices. You'll use AWS Migration Hub Refactor Spaces to facilitate this incremental transition.

Objectives

After completing this lab, you will be able to:

- Objective 1: Explore the monolithic application.
- Objective 2: Use AWS Migration Hub Refactor Spaces to create a refactor environment and define traffic routing.
- Objective 3: Leverage Microservices architecture.
    - Use Amazon DynamoDB for storing and accessing shopping cart information.
    - Use AWS Lambda as compute resource for accessing and manipulating the shopping cart.

Prerequisites

This lab requires:

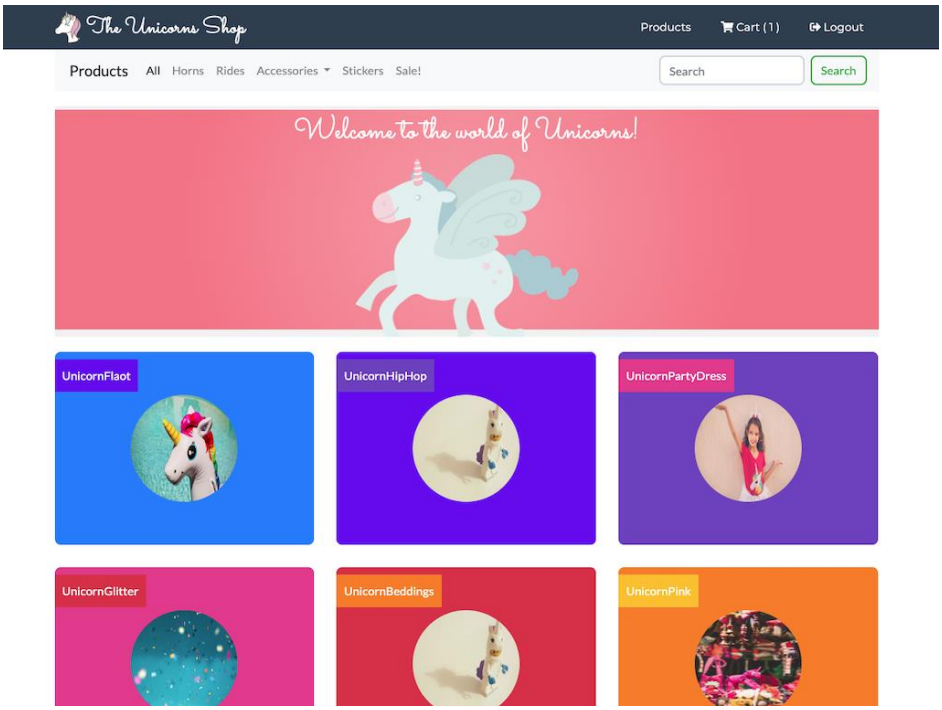
- Access to a computer with Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).

- A modern internet browser such as Chrome or Firefox.
- Be familiar with the AWS console.
- Be familiar with AWS Migration Hub.

Duration

This lab requires approximately 90 minutes to complete.

Scenario



Unishop is THE one-stop-shop for all your Unicorn needs. You can find the best Unicorn selection online at the Unishop and get your Unicorn delivered in less than 24 hours! Its current monolithic architectecture makes it difficult to grow with business expansion. The CTO would like to explore migrating the Unishop to a microservice architecture using the **strangler pattern**.

AWS Migration Hub Refactor Spaces

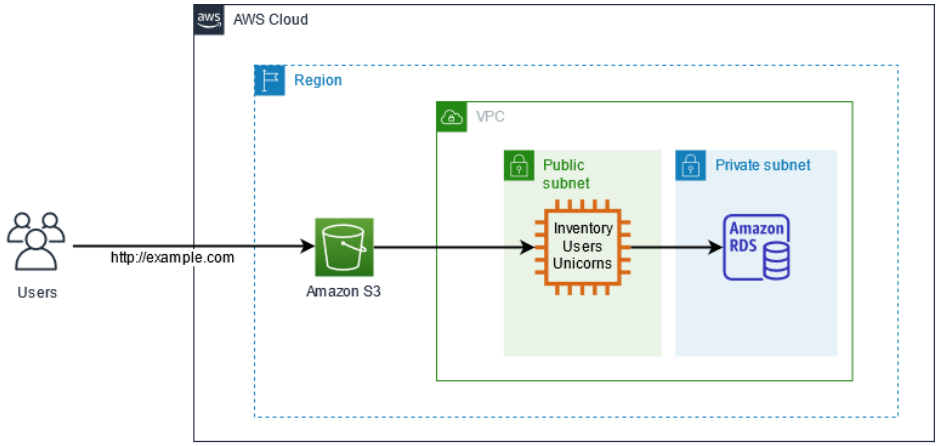
You'll use the **AWS Migration Hub Refactor Spaces** to facilitate incremental transition to a microservices-based architecture. Using **Refactor Spaces**, you can focus on the refactor of their applications, and not the creation and management of the underlying infrastructure.

Refactor Spaces creates a network fabric so that an existing application and new microservices can communicate directly across different AWS accounts. This lab is conducted in a single account, but **Refactor Spaces** simplifies incrementally refactoring using multiple AWS accounts. It does this by orchestrating the following infrastructure/services:

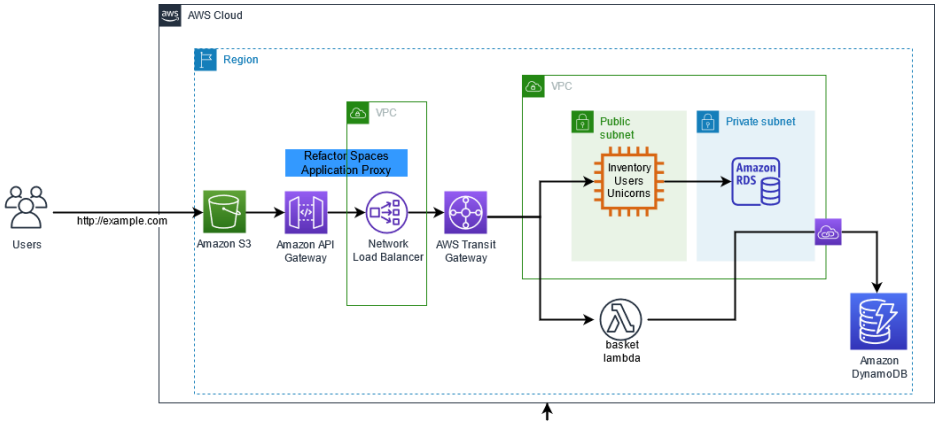
- **Transit Gateway** and **VPCs** to bridge networking across accounts to simplify old and new services communications.
- **API Gateway** and **VPC Link** for external API access to transparently add new services and incrementally route traffic from old to new.

Architecture

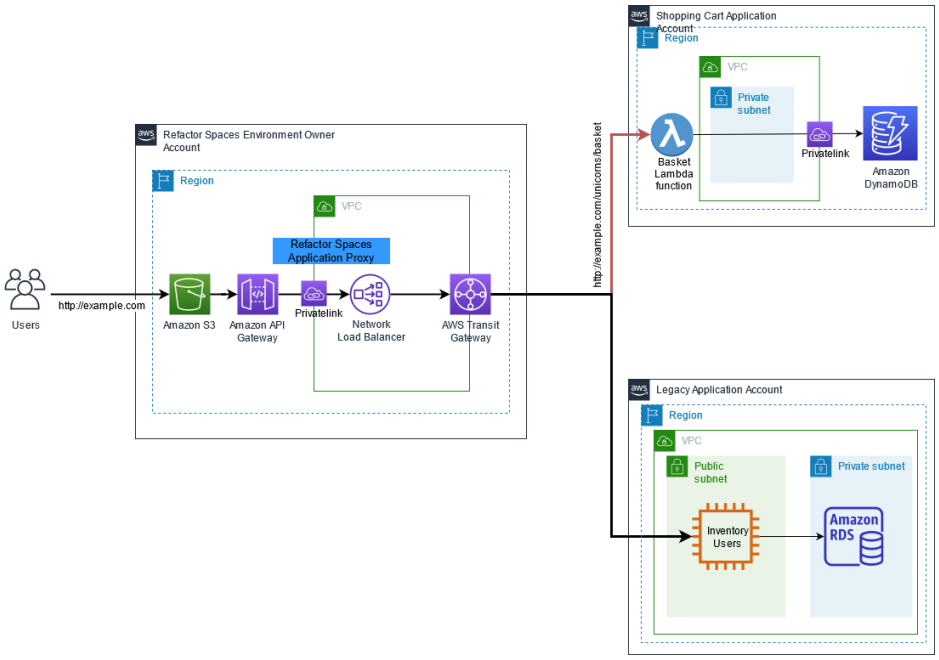
The **AS-IS architecture** leverages a single VPC where an EC2 instance resides within a single availability zone. Likewise, the RDS instance reside within a single availability zone.



The **TO-BE architecture** uses **Refactor Spaces**. Your goal is to use **Refactor Spaces** to modify the route for shopping cart to a new microservice.



As a best practice, the **Refactor Spaces** infrastructure and the microservices are deployed into their own account as follows (see Refactor Spaces documentation how to share with other accounts). However, you'll implement a simpler architecture using single account for this lab.



Icon key

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A hint, tip, or important guidance.
- ⓘ **Additional information:** Where to find more information.
- ! **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- ⚠ **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).

Start lab

1. To launch the lab, at the top of the page, choose Start lab.

ⓘ You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose Open Console.

You are automatically signed in to the AWS Management Console in a new web browser tab.

⚠ **Do not change the Region unless instructed.**

Common sign-in errors

Error: You must first sign out

If you see the message, **You must first log out before logging into a different AWS account:**

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose Open Console again.

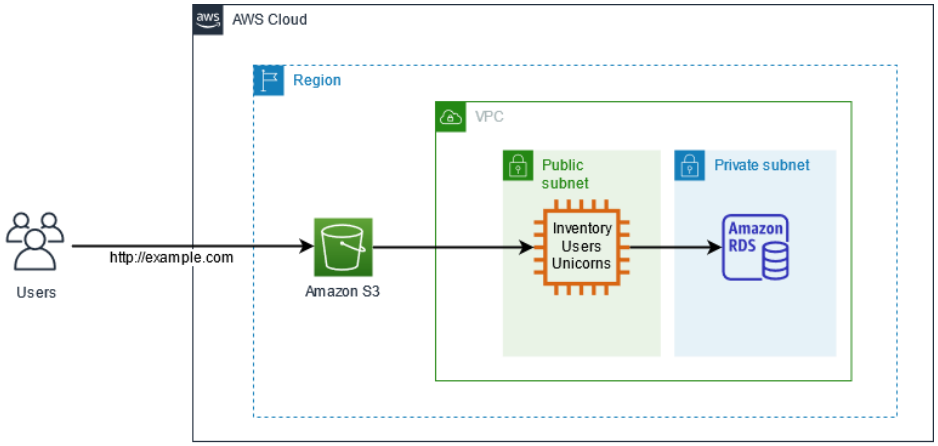
Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

Task 1: Monolith

In this task, your goal is to explore the Unishop legacy application.



The application is a Spring Boot Java application using bootstrap. The frontend website is hosted using Amazon S3 static web hosting. The backend app is deployed on a single EC2 instance connected to a MySQL database.

ⓘ This is not the ideal infrastructure architecture for running highly available production applications but suffices for the purposes of this lab. Also note that the EC2 security groups is configured as “open to the world”, and the Database security group is only open to the EC2 instance.

Task 1.1: Verify Backend

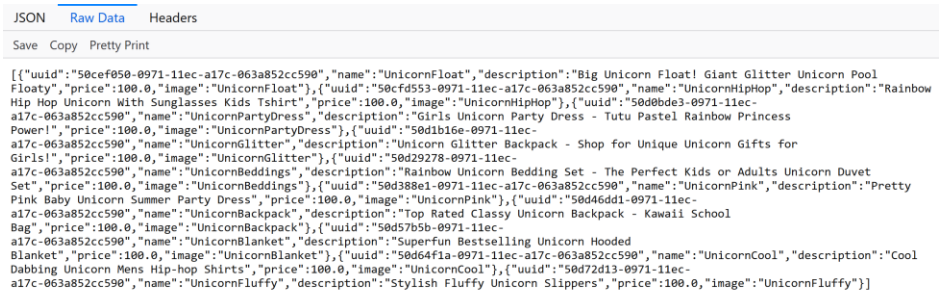
Goal

- Get the backend application URL.
- Verify response.

3. Copy the **PublicDNS** values from the left side of these instructions.
4. This is the DNS name for the EC2 instance that is running your **Unishop** application. You will use that DNS name for configuring Refactor Spaces and API Gateway so take a note of this DNS value.
5. Invoke the Unishop backend application. Open a browser and paste the **PublicDNS** value with path **`/unicorns`** appended (e.g. http://ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com/unicorns). Alternatively, you can use curl command using a command line.

Use HTTP (not HTTPS) for this GET call and don't forget to add /unicorns at the end of the copied URL.

You should see a response similar to the below image:



Now that you have the backend (Java Spring Boot application) verified, let's check the frontend using S3 static website hosting.

Task 1.2: Verify Frontend

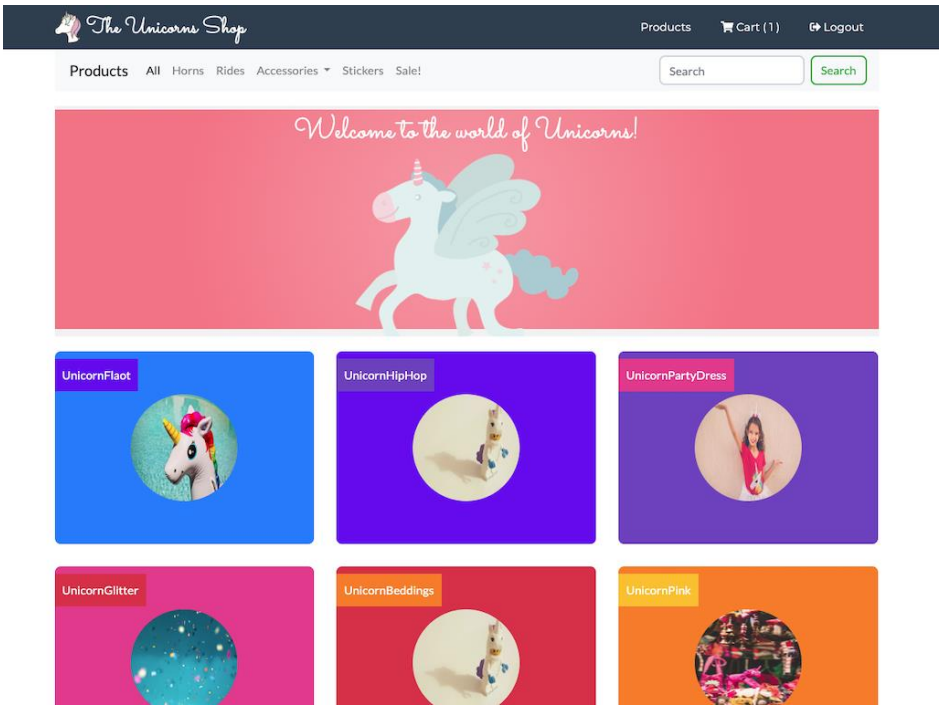
Goal

- Get the website URL.
- Register using email address.
- Add items to shopping cart and see UUID that identifies user.

The frontend website is hosted using Amazon S3 static web hosting. It is a simple yet powerful hosting solution which auto-scales and meets growing needs automatically.

6. Get the **WebsiteURL** values from the left side of these instructions.
7. Open a browser and paste the **WebsiteURL** value (e.g. http://xxxxx-xxxxx-xxxxx-uibucket-xxxxx.s3-website-us-east-1.amazonaws.com).

You should see the Unishop landing page with Unicorns displayed:



8. Sign up yourself as user.

9. Register yourself into the application. You just need to provide an e-mail (doesn't need to be valid email). **Remember this email** as you will use it later in this lab.

10. Check the output in your browser's developer console.

ⓘ You can use your browser developer console as a tool to troubleshoot your actions during this lab (mainly around CORS origin issues). To open the developer console in Google Chrome, open the Chrome Menu in the upper-right-hand corner of the browser window and select More Tools > Developer Tools. You can also use Option + ⌘ + J (on macOS), or Shift + CTRL + J (on Windows/Linux) or check your browser's specific instructions.

You will get something like:

```
{ uuid: "f031e124-f75a-4112-1234-78abbcc9d070", email: "<provided email>" } "User Signed Up"
```

Take note of this UUID as this is going to identify this user and its basket.

11. Log in into the application. You just need to provide the registered email. Check to make sure that the message `[] "Got the cart"` appears in console.

12. Add/remove items to your shopping cart. Check the outputs in the browser's console. Note that you can only add one of each item.

Review

Now that you've successfully verified Unishop monolithic application, you're ready for microservice architecture.

The current Unishop application has **5 Controllers** that uses **3 Tables** in the RDS MySQL database:

Controllers

- **CoreController**
- **BasketController:** basket management
- **UnicornController:** inventory management
- **UserController:** user management, registration, login
- **HealthController:** performs basic health checks

Tables

- **unicorns:** holds the inventory of Unicorns
- **unicorn_basket:** an association table between the Unicorns and the user's selection
- **unicorn_user:** represents the users in the system

Our goal is to deploy a new microservice that can replace the **BasketController** functionality of the legacy app along with the **unicorn_basket** table it uses. The microservice will be implemented using AWS Lambda and DynamoDB as database.

In order to apply the Stranger Fig pattern, you need a way to redirect the shopping cart traffic away from the legacy application to the new microservice. This is where you'll leverage the AWS Migration Hub Refactor Spaces to simplify the routing.

Task 2: Configure Refactor Spaces

In Task 2, you'll use **AWS Migration Hub Refactor Spaces** to create an environment that can replace functionality in the legacy application and route traffic to the replacement microservices (strangler pattern). You'll create the following resources:

- A **Refactor Spaces Environment** is a container of Refactor Spaces Services and Proxies. It's a multi-account network fabric consisting of bridged VPCs, which allows resources within it to interact through private IP addresses. It provides a unified view of networking and services across accounts. Account level isolation of services supports customer process and team alignment.
- **Refactor Spaces Services** are entities that provide business capabilities. These services are reachable through unique endpoints and can interoperate across accounts in a Refactor Spaces Environment. In this example there will be two services:
    - **Legacy service** represents the existing monolith. By default all traffic will be routed to the monolith.
    - **Shopping Cart Service** represents the shopping cart functionality.
- A **Refactor Spaces Proxy** is a container of services and routes, which allows a single endpoint to be used to contact multiple services. All traffic hits the single proxy endpoint and then it's sent to multiple services based on different rules. It's an implementation of Strangler Fig Proxy.

  The strangler fig pattern was introduced by Martin Fowler as a way to manage risk when modernizing or rewriting large, monolithic systems. The pattern is an analogy for a type of plant that begins life as a vine growing alongside an older, established tree. As the vine grows, it spreads to completely consume and ultimately replace the host tree, leaving a new, strangler fig tree in its place.

  In the natural analog, this is when the strangler fig initially sends a vine up the host tree’s trunk. Then, a new service, which is decoupled from the monolith, is created, and the proxy’s implementation is deferred to that new service. In the natural analog, this is when the strangler fig vine wraps around one of the tree’s branches and overtakes it. This pattern of proxying and then swapping out the proxy implementation with a new service continues until all the legacy system’s functions are migrated to new services. At this point, the strangler fig vine completely consumes the tree, and the legacy system can be decommissioned. For more information, see Strangler Fig Proxy.

  Using a Refactor Spaces Proxy you can transparently add new services to a unified endpoint and route traffic to new services, all while keeping the underlying architecture changes transparent to the users of the proxy. A Refactor Spaces Proxy supports routing to all compute platforms with public or private visibility.

Task 2.1: Infrastructure

Goal

- Verify Refactor Spaces Environment and Application.

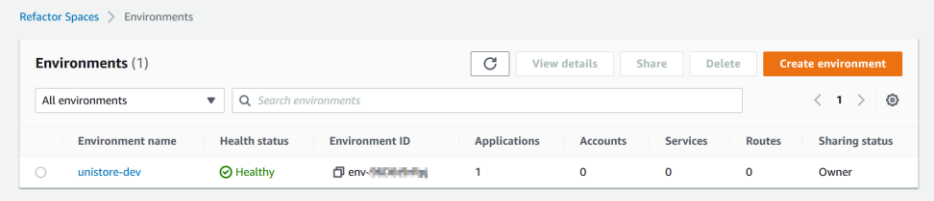
13. At the top of the page, in the unified search bar, search for and choose **`AWS Migration Hub`**.

! This is the AWS console for your lab account, not your regular AWS account. If you don't have the AWS console open, please refer to the previous step to Connect to the lab environment.

Please ignore the following permission error message "**Permission must be granted by your account administrator in order to use Migration Hub**" and select **Refactor Spaces** from the left menu. The permissions that you need for this lab have been granted successfully.

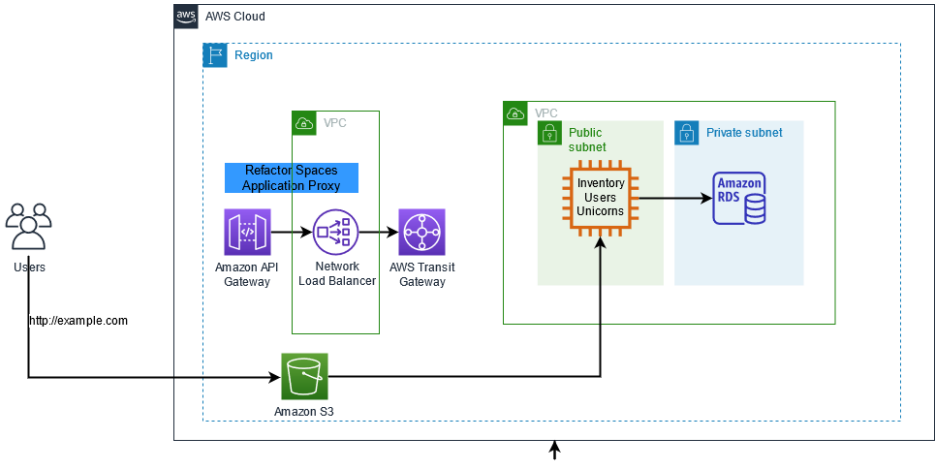
14. Select **Refactor Spaces** from the left menu.

15. Verify the environment named **unistore-dev** exists.



16. Select the environment named **unistore-dev** and notice the application named **unistore** within the environment. The **Health status** of the application should be Healthy.

The current state of the refactor environment resembles the following diagram.



ⓘ There are no user traffic flowing through Refactor Spaces yet at this point.

Task 2.2: Create Services

Goal

- Create Refactor Spaces Service for legacy backend application.
- Route all existing traffic to legacy service through Application Proxy (API Gateway).

Create Refactor Spaces Service for Legacy Application

You'll create a Refactor Spaces service named **legacy** that represents the existing monolith. Your goal is to create the service and configure it using the current monolith's backend endpoint URL (e.g. http://ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com/
). This will be the **PublicDNS** values from the left side of these instructions.

17. **Create Service** - Navigate to **Migration Hub Refactor Spaces** and choose **Create service** from the **Quick actions** menu. Select the Refactor Spaces environment and application you reviewed earlier from the drop down.

18. In the **Create service** page configure the following:

    a. For **Environment**, select **unistore-dev**.

    b. For **Application**, select **unistore**.

    c. For **Service name**, enter **`legacy`**.

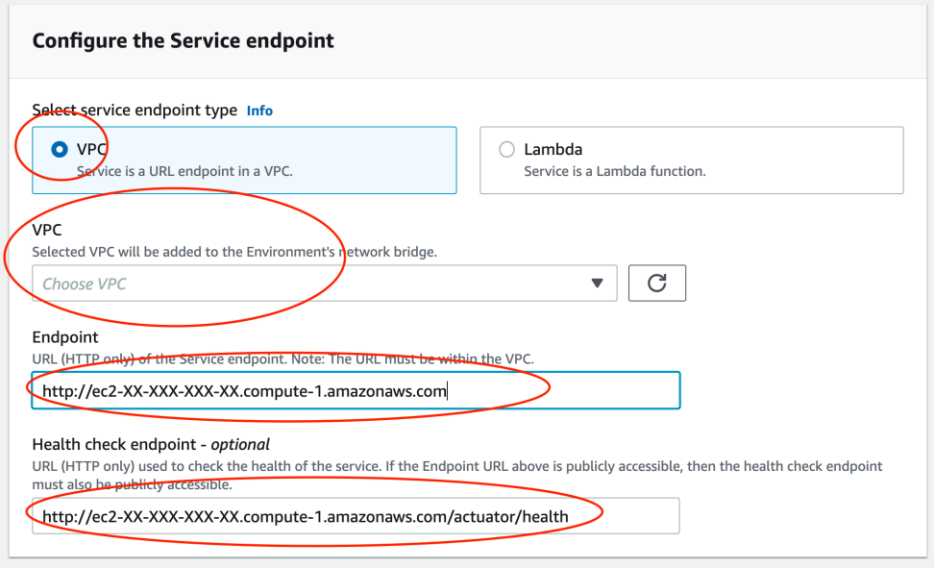
    d. For **Select service endpoint type**, choose **VPC**.

    e. For **VPC**, select the VPC that contains the monolith application from the drop down (named "MonoToMicroVPC"). You can verify the correct VPC using the VPC service in the console. This will be the **VpcId** values from the left side of these instructions.

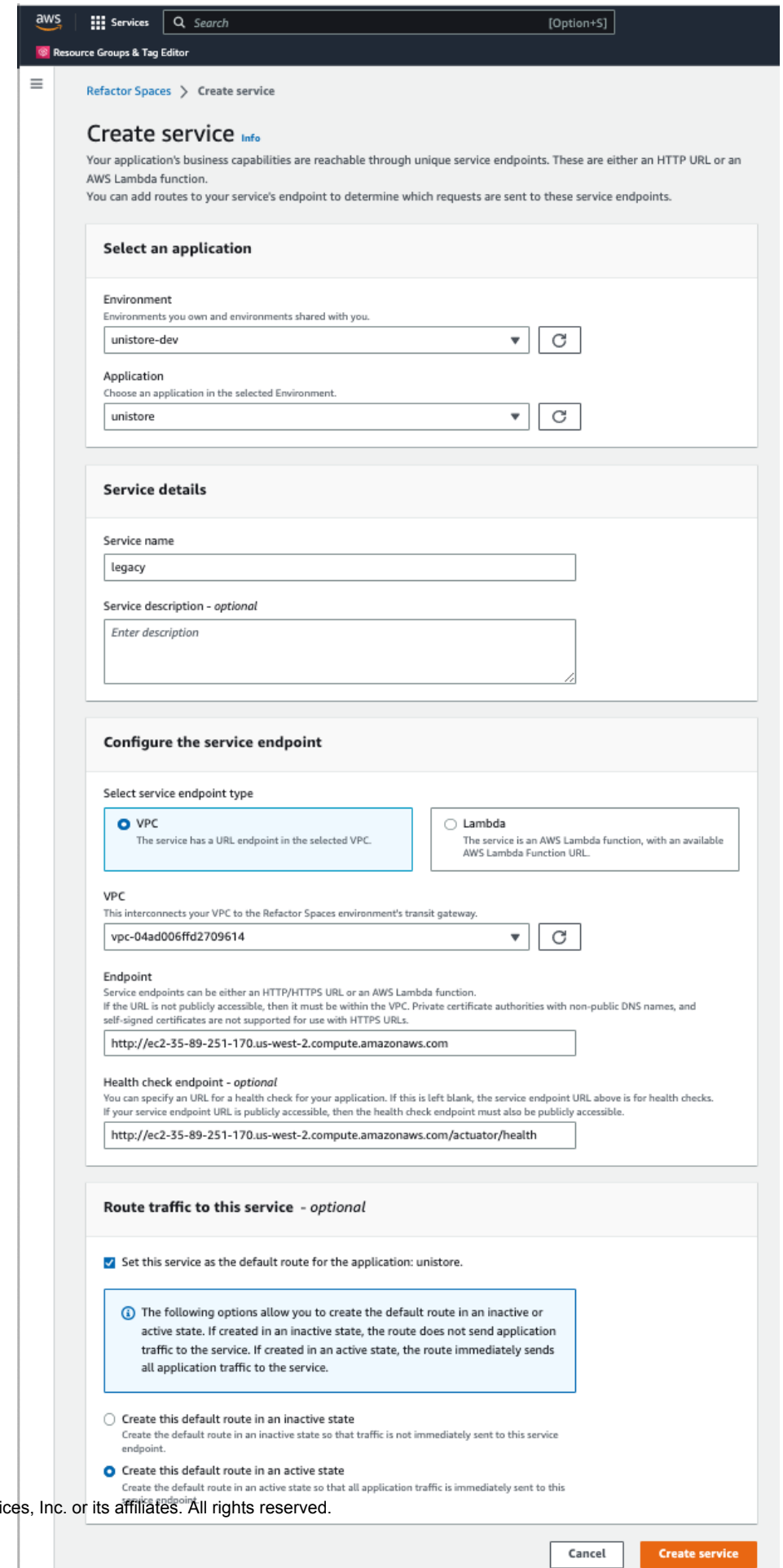
    f. For **Endpoint**, enter the **PublicDNS** values from the left side of these instructions (e.g. http://ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com/).

    g. For **Health check endpoint - optional**, this is same value as URL from previous step with path **`/actuator/health`** appended.

    Enter the **PublicDNS** values from the left side of these instructions with path **`/actuator/health`** appended (e.g. http://ec2-XXX-XXX-XXX-XXX.compute-1.amazonaws.com/actuator/health).



- For **Route traffic to this service**, check the box for **Set this service as the default route for the application**. This sends all traffic through the monolith by default.
- Select **Create this default route in an active state**.



- Choose **Create service**.

⚠ Wait for the service status to change to Healthy and routes status to Active before proceeding to the next task. The Refactor Spaces is now creating the necessary routes and API Gateway resources and stage. These API Gateway resources will be used as a new Application Proxy that manages all traffic to the legacy service and any future microservices.

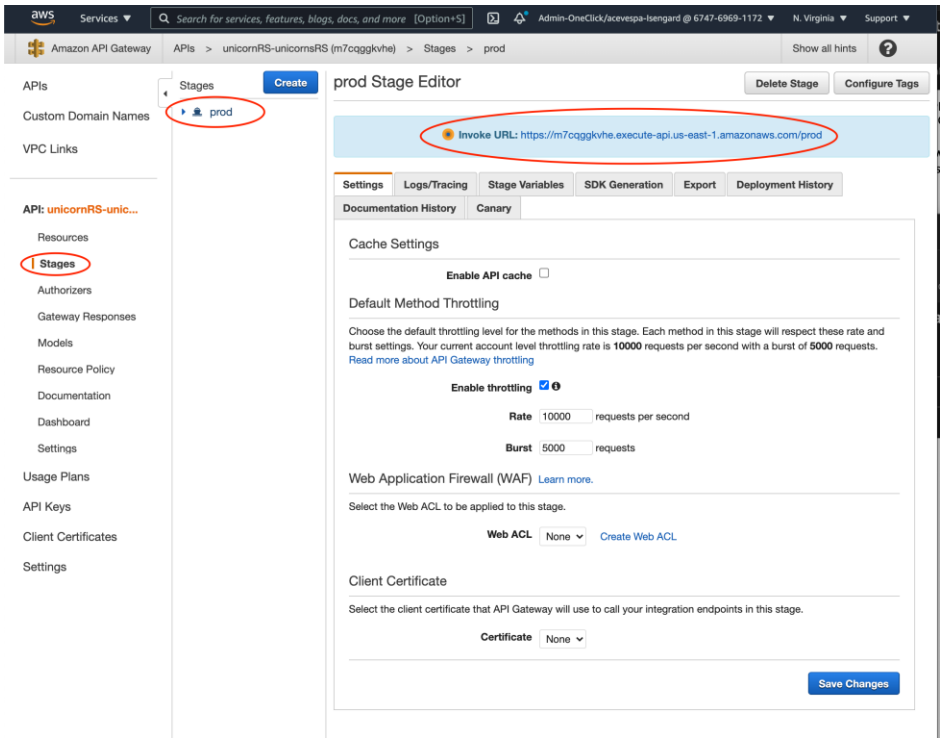
Route all existing traffic to legacy service through Application Proxy (API Gateway)

Now with application proxy (API Gateway) resources available, let's configure the front end to use the new endpoint Refactor Spaces created for the legacy service. In order to do this, you'll update a config.json file of the front end application with the URL of the new endpoint.

19. At the top of the page, in the unified search bar, search for and choose **`API Gateway`**.

20. Select **unistore** API.

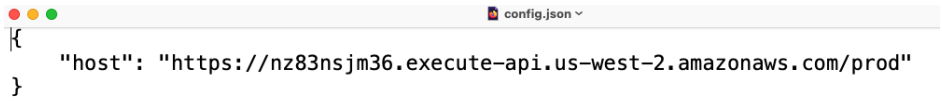
21. Navigate to **Stages** menu and select the **prod** stage. Copy the **Invoke URL**.



Now you need to modify the front end config file to use the new URL for backend. The config file is located in an S3 bucket that is hosting the front end website. Navigate to the Amazon S3 service and open the bucket named values `**UIBucket**` from the left side of these instructions

22. At the top of the page, in the unified search bar, search for and choose **`S3`**.

23. In the Buckets list, select the bucket named values `**UIBucket**` from the left side of these instructions.
24. Download the **config.json** file to your local machine.
25. Open the newly downloaded **config.json** file and replace the host URL with the **API Gateway** stage's **Invoke URL** you copied in the previous step.
26. Replace "host" value and save the file.

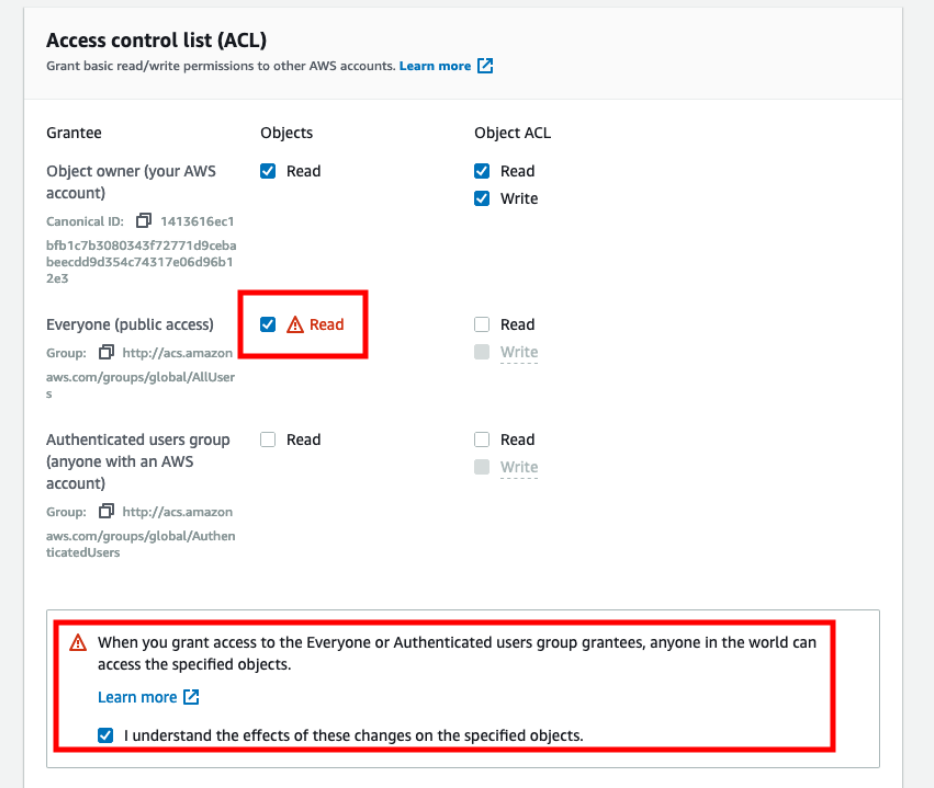


⚠ Make sure the URL is secured using HTTPS.

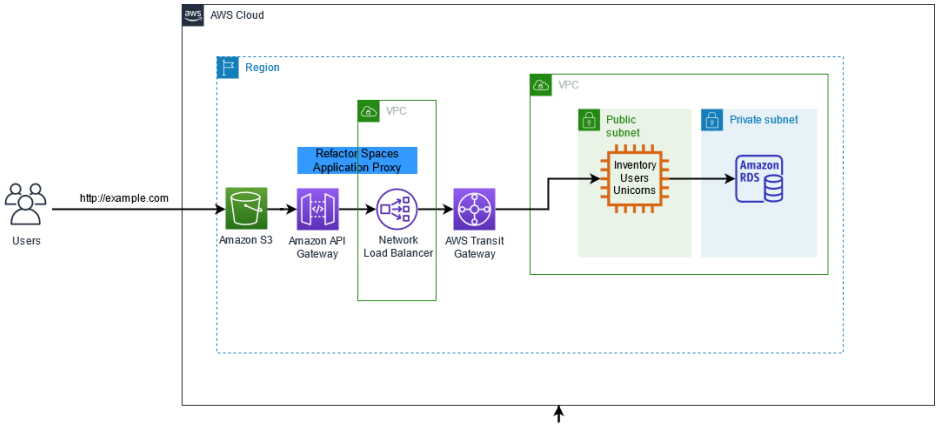
27. To upload the new **config.json** navigate to Amazon S3 bucket console.
28. In the Buckets list, select the value for bucket named **`UIBucket`** from the left side of these instructions.
29. Choose **Upload**.
30. Choose **Add files**, choose files and then select the new **config.json** to upload, and choose **Open**.
31. Choose **Upload**.

Make sure you grant the file with **Read** access for **Everyone** in the Permissions tab by editing the Access control list (**ACL**) for public access. Your console may show option **Grant Public-read access** which is same setting.

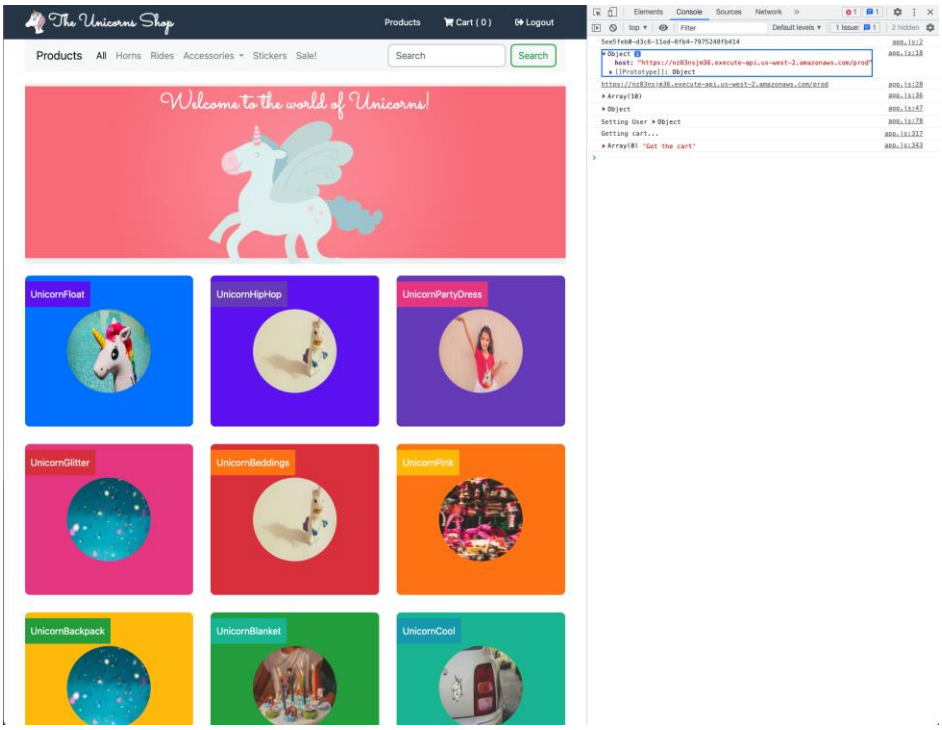
32. In the **Objects** list, choose **config.json**.
33. Choose **Permissions**.
34. Under **Access control list (ACL)**, choose **Edit**.
35. Select the check boxes for the permissions file with **Read** access for **Everyone**, and **I understand the effects of these changes on this object** and then choose **Save changes**.



Now that you've updated the URL in your configuration file the application is now being proxied and looks as follows:



36. **Test Website** - Let's test the front end to make sure the Unishop website still works. Open a browser and navigate to the Unishop URL from Verify Frontend. If you view the browser's developer console, you'll see the new proxy host.



If you cannot see the unicorns after switching to the API Gateway endpoint, perform a hard refresh using the S3 frontend URL on your web browser (response may have been cached).

ⓘ To open the developer console in Google Chrome, open the Chrome Menu in the upper-right-hand corner of the browser window and select More Tools > Developer Tools. You can also use Option + ⌘ + J (on macOS), or Shift + CTRL + J (on Windows/Linux) or check your browser's specific instructions.

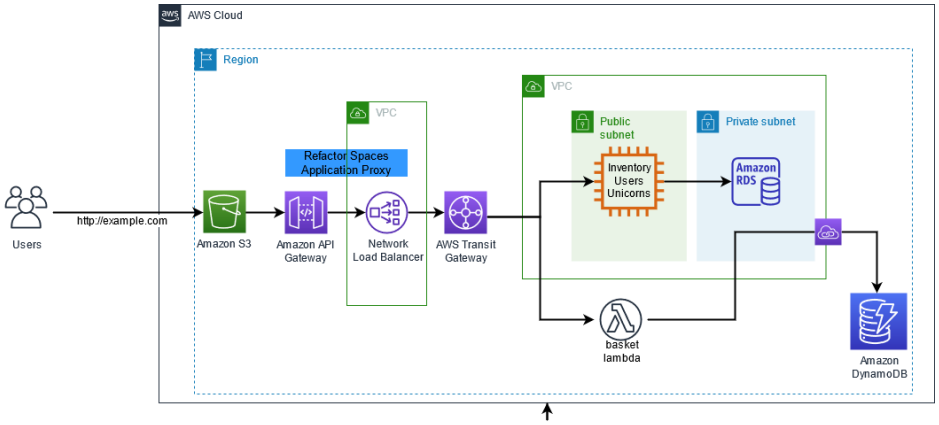
Review

In task 2 you used AWS Migration Hub Refactor Spaces to create an environment that will allow you to manage legacy applications and microservices as a single application with flexible routing control, isolation, and centralized management.

In the next task, you will create a new microservice to replace the shopping cart functionality of the monolith. Then using the Migration Hub Refactor Spaces proxy, you'll redirect traffic for the shopping cart to the new microservice.

Task 3: Microservices

In this task, you are going to build a new microservice to replace the shopping cart functionality using AWS Lambda and AWS DynamoDB. Then you'll configure Refactor Spaces to route the traffic for shopping cart to the new microservice from the monolith. Finally, you'll test the Unishop website to make sure everything works seamlessly using the Strangler Fig pattern.



Task 3.1: DynamoDB

Goal

- Verify DynamoDB for new Lambda service.

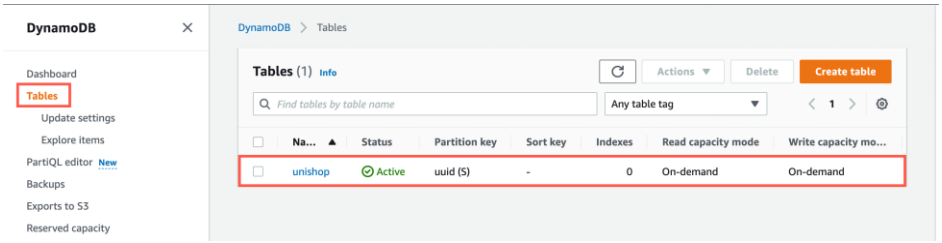
By using microservices architecture, it gives you the freedom to choose different technologies for your implementation. You will use Amazon DynamoDB as your database for your shopping cart microservice. DynamoDB is a fast key/value store which can hold shopping cart information and also enable quick reads and writes.

ⓘ DynamoDB table named **unishop** has been pre-deployed for this lab.

37. At the top of the page, in the unified search bar, search for and choose **`DynamoDB`**.

38. Select **Tables** from the left menu.

39. Verify the DynamoDB table named **unishop**:



Task 3.2: Lambda

Goal

- Verify 3 AWS Lambda Functions for new Shopping Cart microservice.

AWS Lambda is a serverless compute service where you can run code without provisioning or managing infrastructure. You'll use AWS Lambda to build 3 functions that you need for shopping cart service:

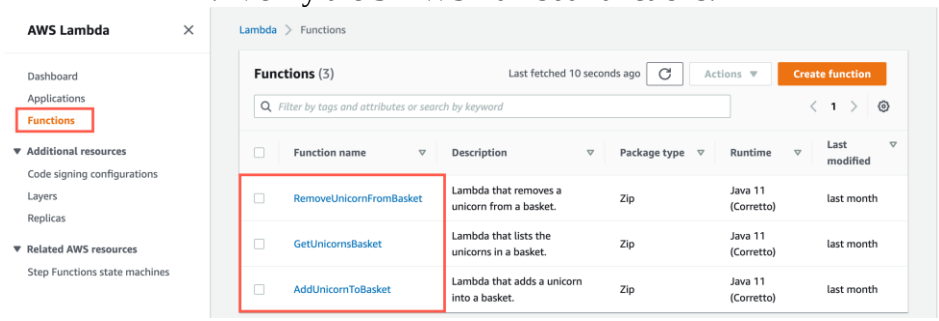
- **AddUnicornToBasket**
- **RemoveUnicornFromBasket**
- **GetUnicornsBasket**

ⓘ 3 AWS Lambdas have been pre-deployed for this lab.

40. At the top of the page, in the unified search bar, search for and choose **`Lambda`**.

41. Select **Functions** from the left menu.

42. Verify the 3 AWS Lambda functions:



Task 3.3: Shopping Cart Services

Now that you have your Lambda functions and DynamoDB deployed, next step is to create the Refactor Spaces services that will enable you to route traffic from the monolith to your new microservices. You will configure the following services with corresponding Lambda functions to route the shopping cart traffic.

Refactor Spaces Service	Lambda	Description
AddToCartService	AddUnicornToBasket	Adding a unicorn to the cart.
RemoveCartService	RemoveUnicornFromBasket	Removing a unicorn from the cart.
GetCartService	GetUnicornsBasket	Getting shopping cart items.

Task 3.4: Add to Cart Service

Goal

- Route traffic for shopping cart's add items to new AddUnicornToBasket Lambda Function.

In order to route traffic to new AddUnicornToBasket Lambda function, you will create a Refactor Spaces service named **AddToCartService**. This Refactor Spaces service will automatically configure the API Gateway to route traffic from monolith to the AddUnicornToBasket Lambda function.

43. At the top of the page, in the unified search bar, search for and choose **`AWS Migration Hub`**.
44. Select **Refactor Spaces** from the left menu.
45. Choose **Create Service** from the **Quick actions** menu. Select the Refactor Spaces environment and application you created earlier from the drop down.
46. Create the **AddToCartService** in **Refactor Spaces** with the following steps:

- For **Environment**, select **unistore-dev**.
- For **Application**, select **unistore**.
- For **Service name**, enter **`AddToCartService`**.
- For **Service description**, enter **`Adding a unicorn to the cart`**.
- For **Select service endpoint type**, choose **Lambda**.
- For **Lambda function**, select **AddUnicornToBasket**.
- For **Source path**, enter **`/unicorns/basket`**.
- Uncheck **Include child paths**.
- For **Verb**, select **POST** (uncheck the **"Match All"**).
- Select **Create this route in an active state**.

47. Choose **Create service**.

Task 3.5: Remove Cart Service

Goal

- Route traffic for shopping cart's deleted items to new RemoveUnicornFromBasket Lambda Function.

In order to route traffic to new RemoveUnicornFromBasket Lambda function, you will create a Refactor Spaces service named **RemoveCartService**. This Refactor Spaces service will automatically configure the API Gateway to route traffic from monolith to the RemoveUnicornFromBasket Lambda function.

48. Navigate back to the **Refactor Spaces** page and select **Create Service** from the **Quick actions** menu.

49. Create the **RemoveCartService** in **Refactor Spaces** with the following steps:

- For **Environment**, select **unistore-dev**.
- For **Application**, select **unistore**.
- For **Service name**, enter **`RemoveCartService`**.
- For **Service description**, enter **`Removing a unicorn from the cart`**.
- For **Select service endpoint type**, choose **Lambda**.
- For **Lambda function**, select **RemoveUnicornFromBasket**.
- For **Source path**, enter **`/unicorns/basket`**.
- Uncheck **Include child paths**.
- For **Verb**, select **DELETE** (uncheck the **"Match All"**).
- Select **Create this route in an active state**.

50. Choose **Create service**.

Task 3.6: Get Cart Service

Goal

- Route traffic for listing items in the shopping cart to GetUnicornsBasket Lambda Function.
- Create API Gateway Model.
- Disable Lambda proxy integration.
- Configure mapping template for GET method.
- Delete redundant GET method.

In order to route traffic for listing shopping cart items to GetUnicornsBasket Lambda function, you will create a Refactor Spaces service named **GetCartService**. The GetCartService needs to map the path variable (uuid) to a JSON object (which is part of the Lambda method signature). So, you will create an object model and configure a mapping template for the Integration Request in API Gateway.

51. Navigate back to the **Refactor Spaces** page and select **Create Service** from the **Quick actions** menu.

52. Create the **GetCartService** in **Refactor Spaces** with the following steps:

- For **Environment**, select **unistore-dev**.
- For **Application**, select **unistore**.
- For **Service name**, enter **`GetCartService`**.
- For **Service description**, enter **`Getting shopping cart items`**.
- For **Select service endpoint type**, choose **Lambda**.
- For **Lambda function**, select **GetUnicornsBasket**.
- For **Source path**, enter **`/unicorns/basket`**.
- For **Include child paths**, keep the box **checked**.
- For **Verb**, select **GET** (uncheck the **"Match All"**).
- Select **Create this route in an active state**.

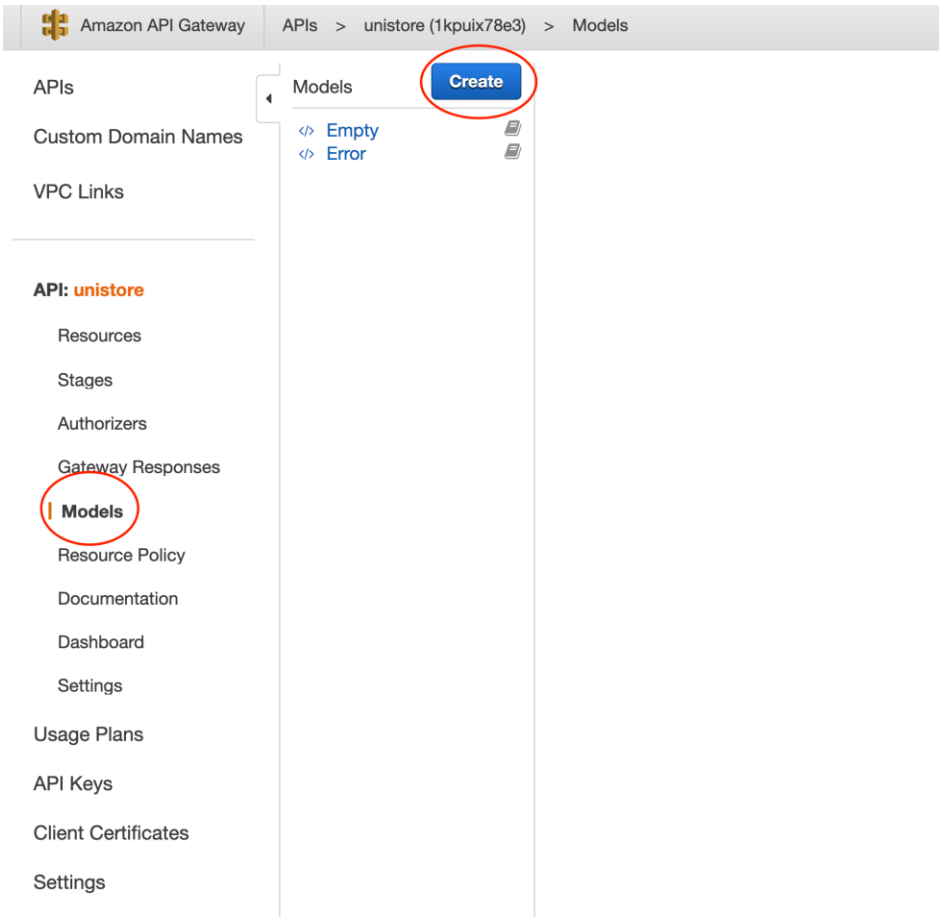
53. Choose **Create service**

Create API Gateway Model

54. At the top of the page, in the unified search bar, search for and choose **`API Gateway`**.

55. Select **unistore** API that Refactor Spaces created for us.

56. Navigate to the **Models** section from the left side menu and choose **Create** and fill in the following values:

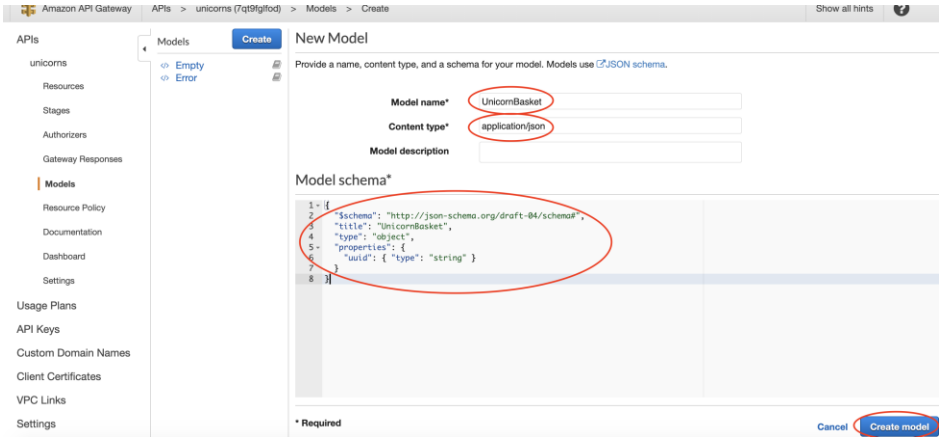


- For **Model name**, enter **`UnicornBasket`**.
- For **Content type**, enter **`application/json`**.
- For **Model schema**, copy and paste the below code:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "UnicornBasket",
  "type": "object",
  "properties": {
    "uuid": { "type": "string" }
  }
}
```

ⓘ Press ESC to navigate away from the code editor.

57. Choose **Create model**.



Disable Lambda Proxy IntegrationHeader anchor link

In this step, you'll disable Lambda Proxy Integration for GET, POST and DELETE methods.

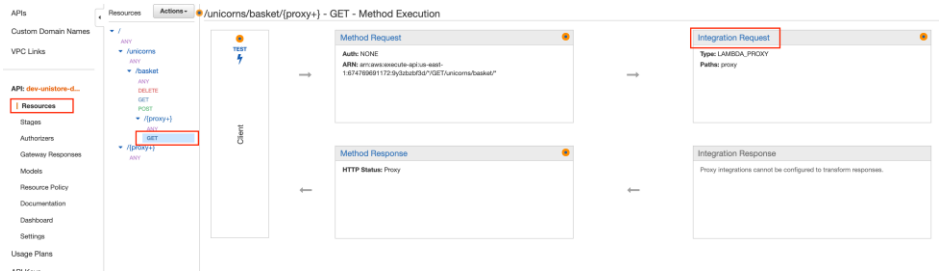
GET method

58. Navigate to **API Gateway** console and select the **unistore** API.

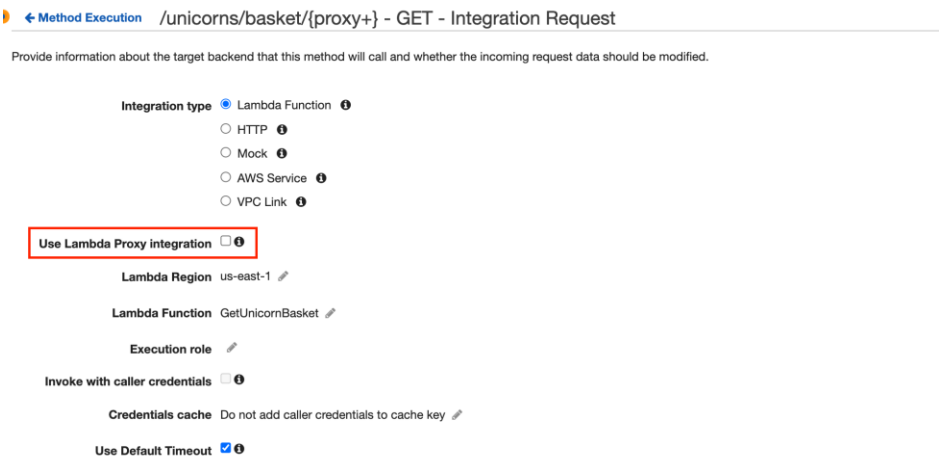
59. Select **Resources** from the API Gateway left menu.

60. From the **Resources** tree select the **GET** method at **/unicorns/basket/{proxy+}**.

61. Select the **Integration Request** title box on the right.



62. Uncheck **Use Lambda Proxy integration** and choose **OK** in both dialogs.



POST method

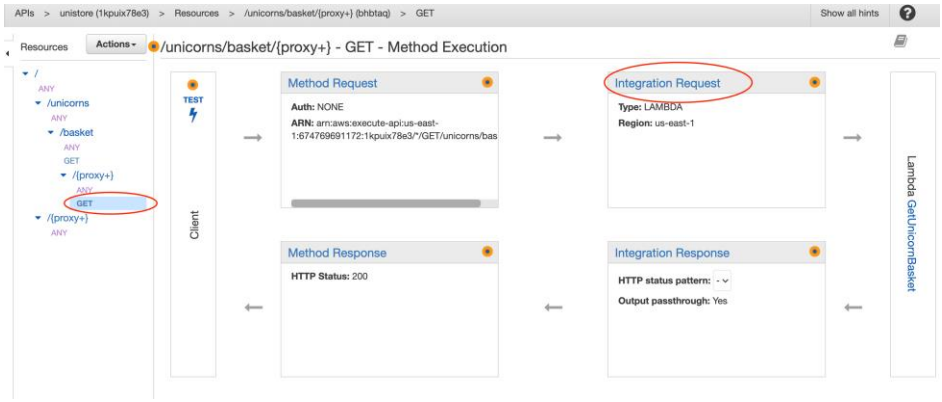
63. Select **Resources** from the left menu.
64. From the Resources tree select the **POST** method at **/unicorns/basket**.
65. Select the **Integration Request** title box on the right.
66. Uncheck **Use Lambda Proxy integration** and choose **OK** in both dialogs.

DELETE method

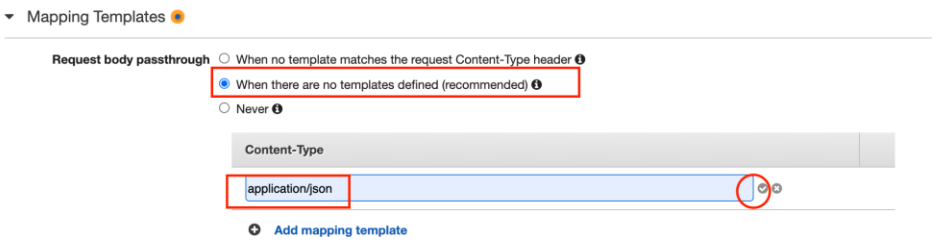
67. Select **Resources** from the left menu.
68. From the Resources tree select the **DELETE** method at **/unicorns/basket**.
69. Select the **Integration Request** title box on the right.
70. Uncheck **Use Lambda Proxy integration** and choose **OK** in both dialogs.

Configure mapping template for GET method

71. Select the **GET** method in the **/unicorns/basket/{proxy+}** resource.
72. Select the **Integration Request** title box on the right.



73. Expand **Mapping Templates**.
74. Select **When there are no templates defined**.
75. Choose **Add mapping template**.
76. Set **Content-Type** to `application/json` in the text field and select the **check** mark.



77. Select from the **Generate template** drop down and choose the model **UnicornBasket**.

78. Replace the text in the template area with the following:

```
#set($inputRoot = $input.path('$'))
{
  "uuid" : "$input.params('proxy')"
}
```

79. Choose **Save**.

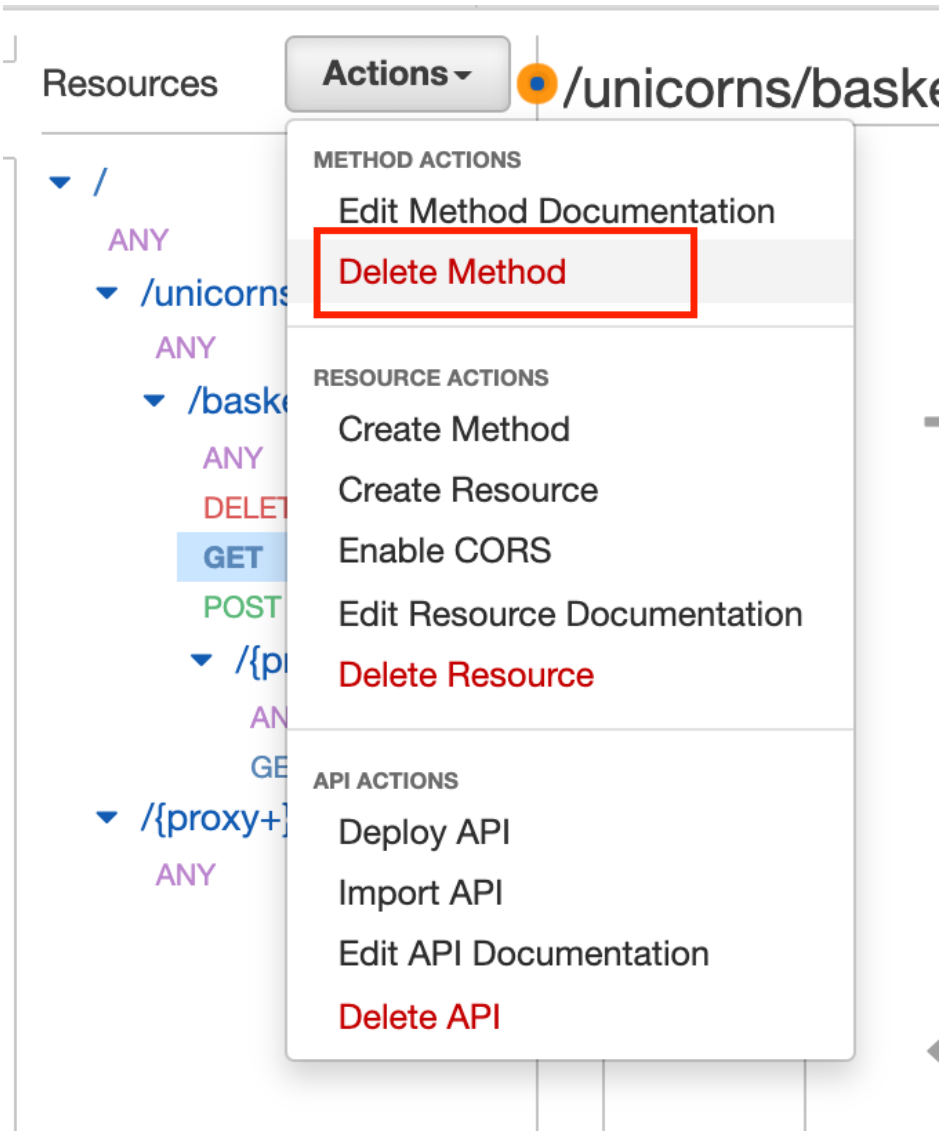
Delete redundant GET method

Since you are only interested in the path variable following **/unicorns/** basket you need to delete the GET method at **/unicorns/basket** and leave only the one at **/unicorns/basket/{proxy+}**.

80. Select the **GET** method on **/unicorns/basket**.

81. Choose **Actions**.

82. Select **Delete Method** and choose **Delete** in the dialog that follows.



Task 3.7: Deploy Services

Goal

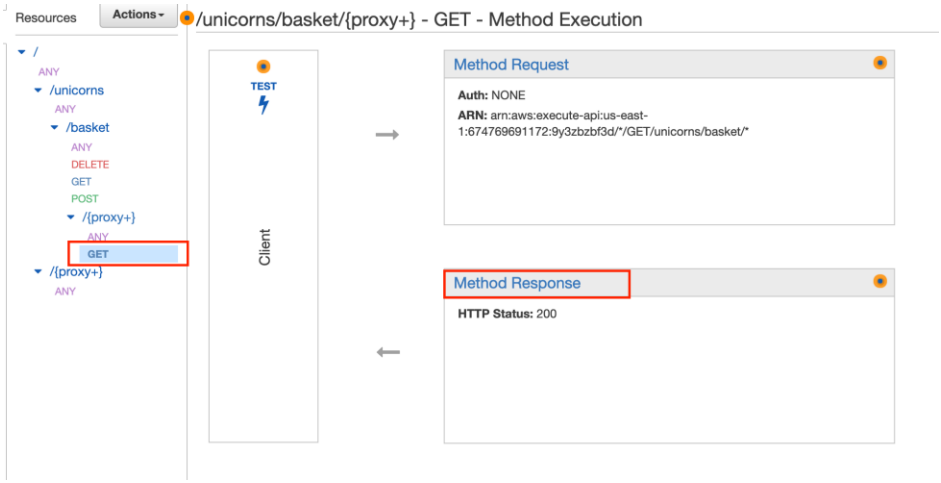
- Configure method response code.
- Enable Cross-Origin Resource Sharing (CORS).
- Deploy API.

Before you deploy the changes, you need to complete a few tasks.

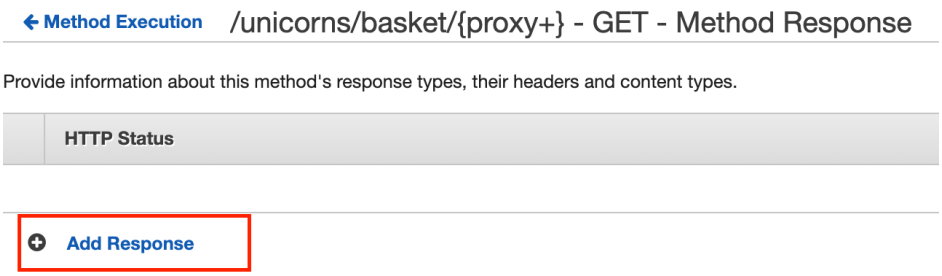
Configure path response for GET, POST and DELETE method

83. From the **Resources** tree, select the **GET** method at **/unicorns/basket/{proxy+}**.

84. Select the **Method Response** title box.



85. Choose **Add Response** and then enter **200** in the HTTP status code text box and choose the check mark.



86. From the **Resources** tree, select the **POST** method at **/unicorns/basket**.

87. Select the **Method Response** title box.

88. Choose **Add Response** and then enter **200** in the HTTP status code text box and choose the check mark.

89. From the **Resources** tree, select the **DELETE** method at **/unicorns/basket**.

90. Select the **Method Response** title box.

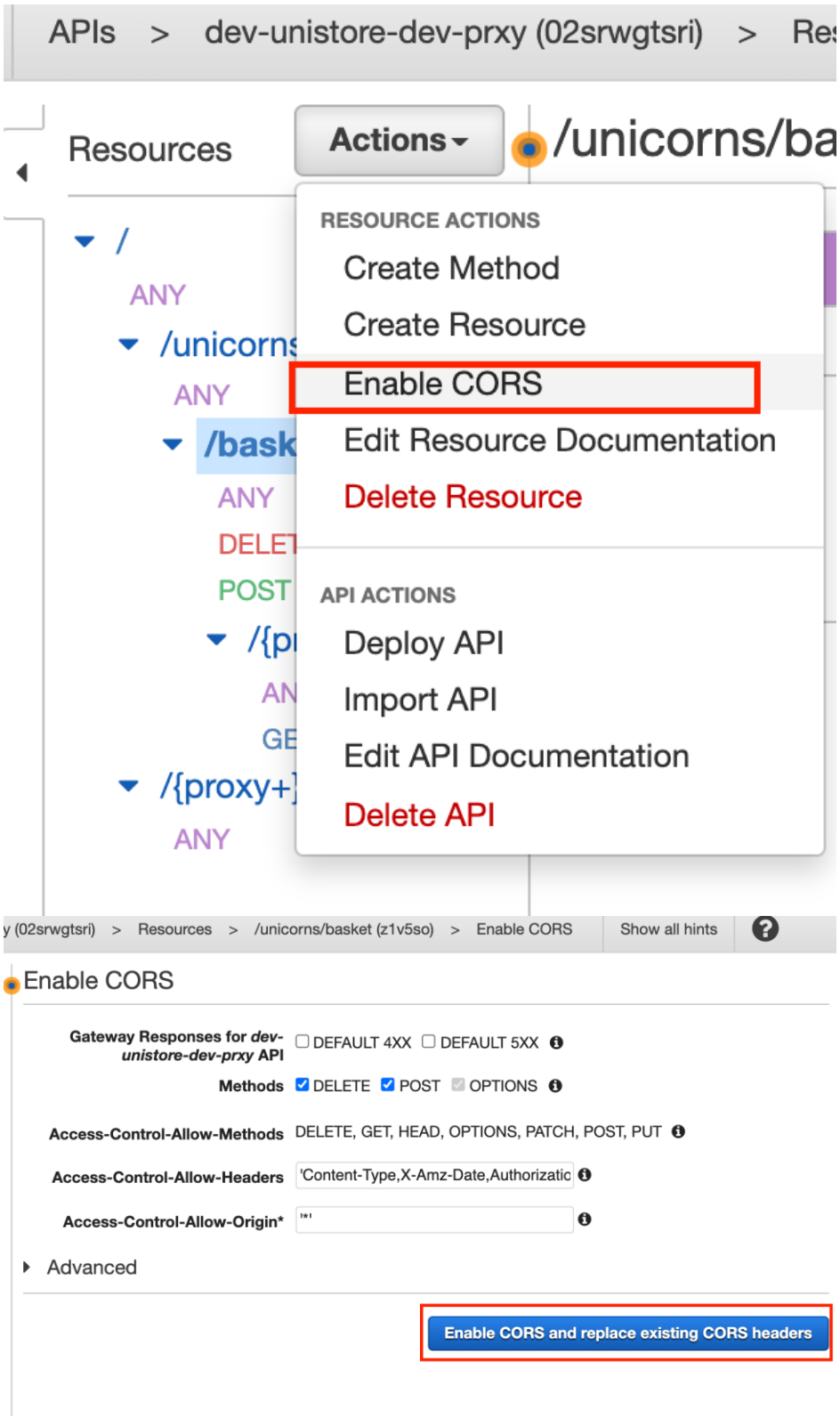
91. Choose **Add Response** and then enter **200** in the HTTP status code text box and choose the check mark.

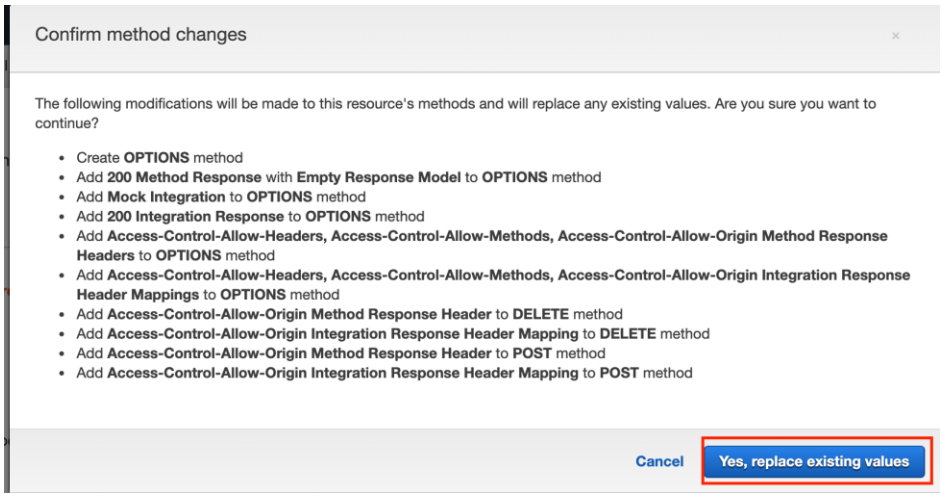
Enable Cross-Origin Resource Sharing (CORS)

One final configuration is to enable **CORS**, because the domains differ between your front-end and API. You'll need to do this on the **/unicorns/basket** and **/unicorns/basket/{proxy+}** resources.

/unicorns/basket resource

92. Select the **/unicorns/basket** resource from tree.
93. Select **Actions** menu and choose **Enable CORS**.
94. Choose **Enable CORS and replace existing CORS headers** and **Yes, replace existing values** button.





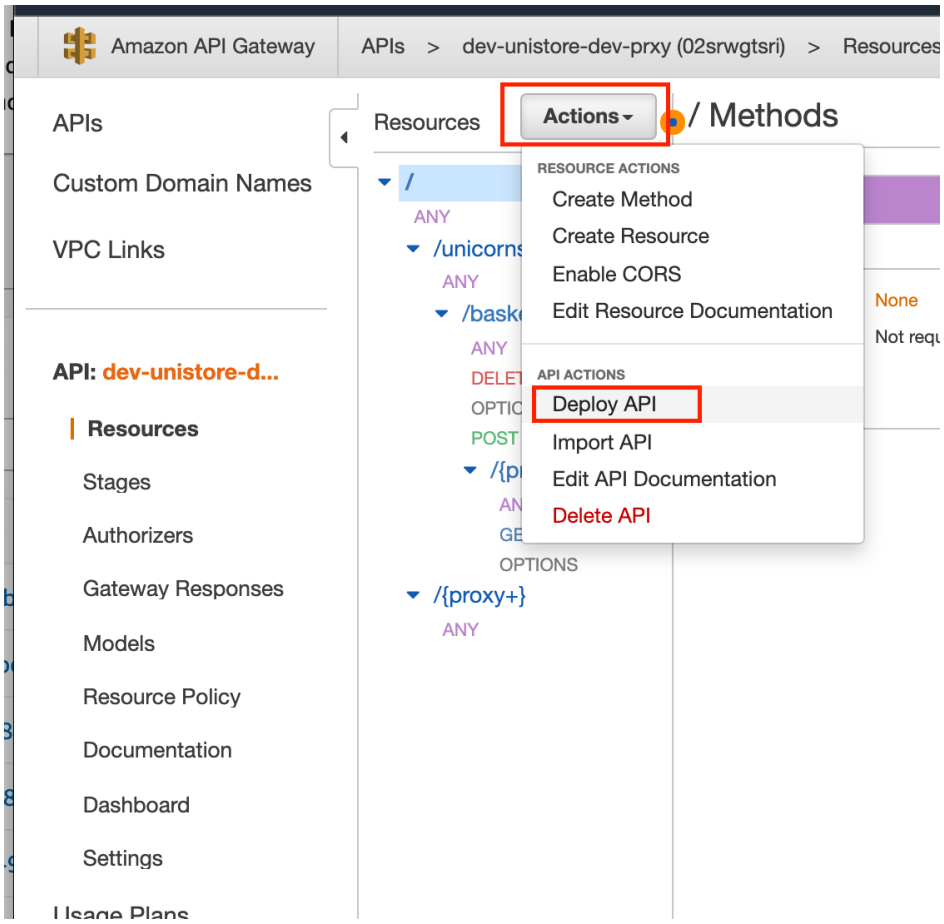
- /unicorns/basket/{proxy+}

95. Select the **/unicorns/basket/{proxy+}** resource from tree.
96. Select **Actions** menu and choose **Enable CORS**.
97. Choose **Enable CORS and replace existing CORS headers** and **Yes, replace existing values** button.

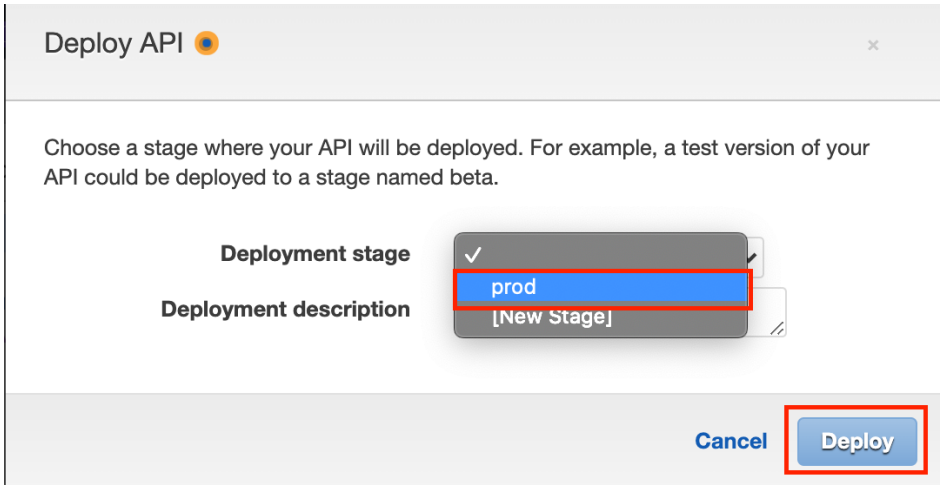
Deploy APIHeader anchor link

The configuration changes you've made, are not live as you need to deploy them.

98. From the **Actions** menu, select **Deploy API**.



99. In the **Deploy API** dialog, select **prod** from the **Deployment stage** drop down and then choose **Deploy**.



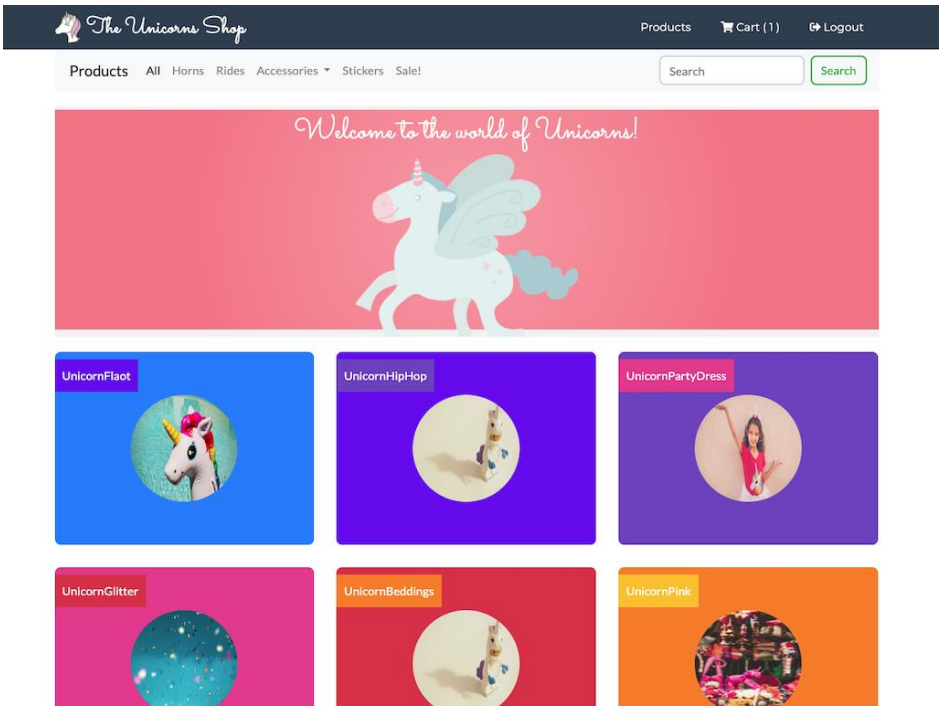
Task 3.8: Testing Services

Goal

- Test Unishop website with new shopping cart services.

- Test shopping cart services directly using API Gateway (optional).

If all is set properly, you should be able to refresh the UI in your browser and see that your Unishop is loading as usual. At this point, your application is running behind a proxy, and all of the requests that represent the shopping cart functionality are being routed to your new microservices that are running in a serverless API Gateway, Lambda, and DynamoDB solution. All other requests continue to be serviced from the legacy monolith application.



Test Unishop Website

100. Log in to the website using the same email address that you used to register.

101. Add and remove items from the cart.

102. Open browser's developer console to explore where the request is going. You should see a new host for the microservice.

ⓘ To open the developer console in Google Chrome, open the Chrome Menu in the upper-right-hand corner of the browser window and select More Tools > Developer Tools. You can also use Option + ⌘ + J (on macOS), or Shift + CTRL + J (on Windows/Linux) or check your browser's specific instructions.

103. Open Lambda dashboard and metrics. You can also open the individual Lambda function and see monitor CloudWatch logs.

104. Open DynamoDB table and explore items to see shopping cart details saved to the database.

Testing services in API Gateway directly

You can also test the shopping cart services directly within API Gateway by making test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

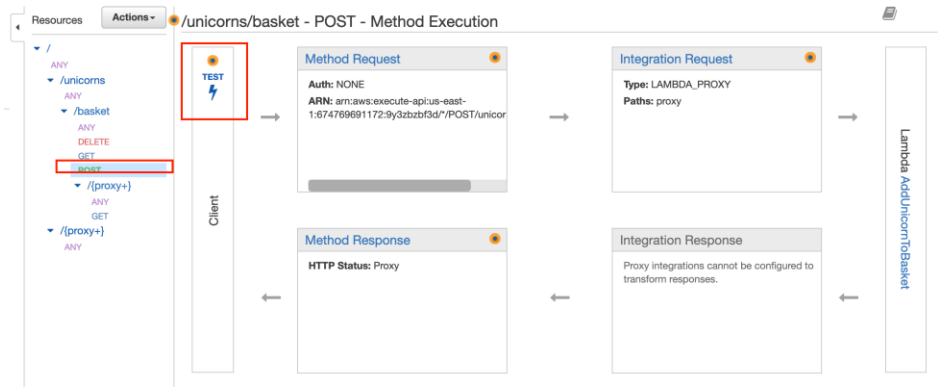
Add a unicorn to the shopping cart

105. Navigate to **API Gateway** console and select the **unistore** API.

106. Select **Resources** from the API Gateway left menu.

107. Navigate to the configuration screen for the **POST** method on the **/unicorns/basket**.

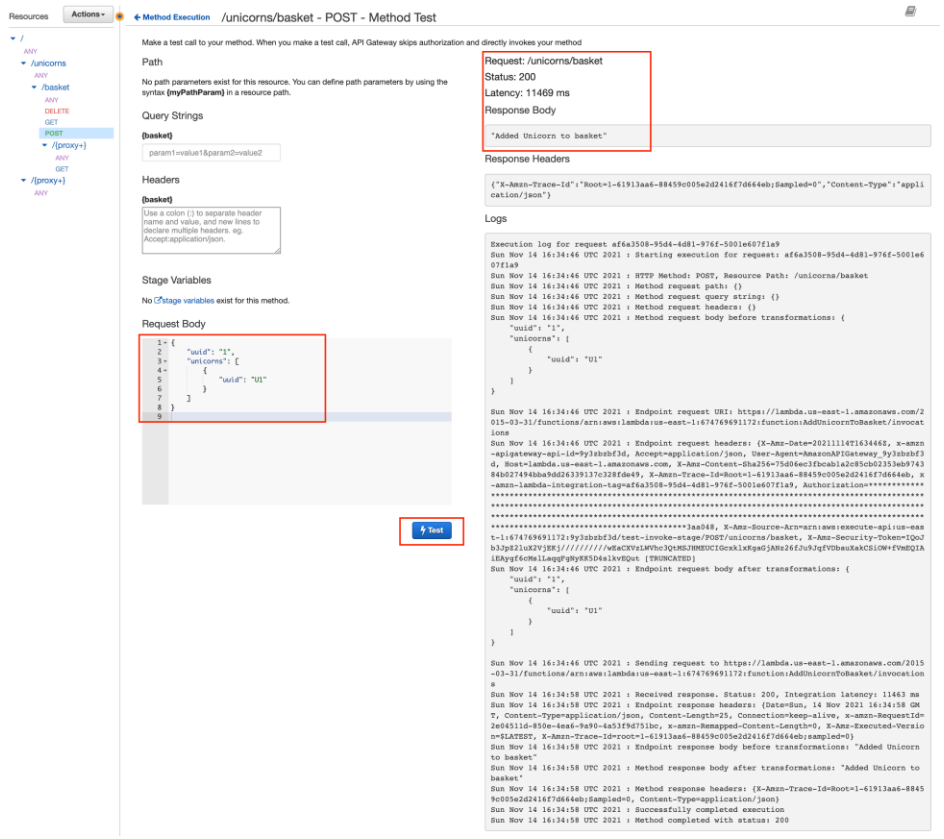
108. Choose **Test** in the Client box on the right.



109. For testing purposes you're going to use a uuid of 1 to represent a logged in user and a unicorn uuid of U1 to represent a unicorn that you want to add to the cart. In the **Request Body** box enter the following text and choose **Test**:

```
{
    "uuid": "1",
    "unicorns": [
        {
            "uuid": "U1"
        }
    ]
}
```

110. You should see a response that the unicorn was added to the cart:

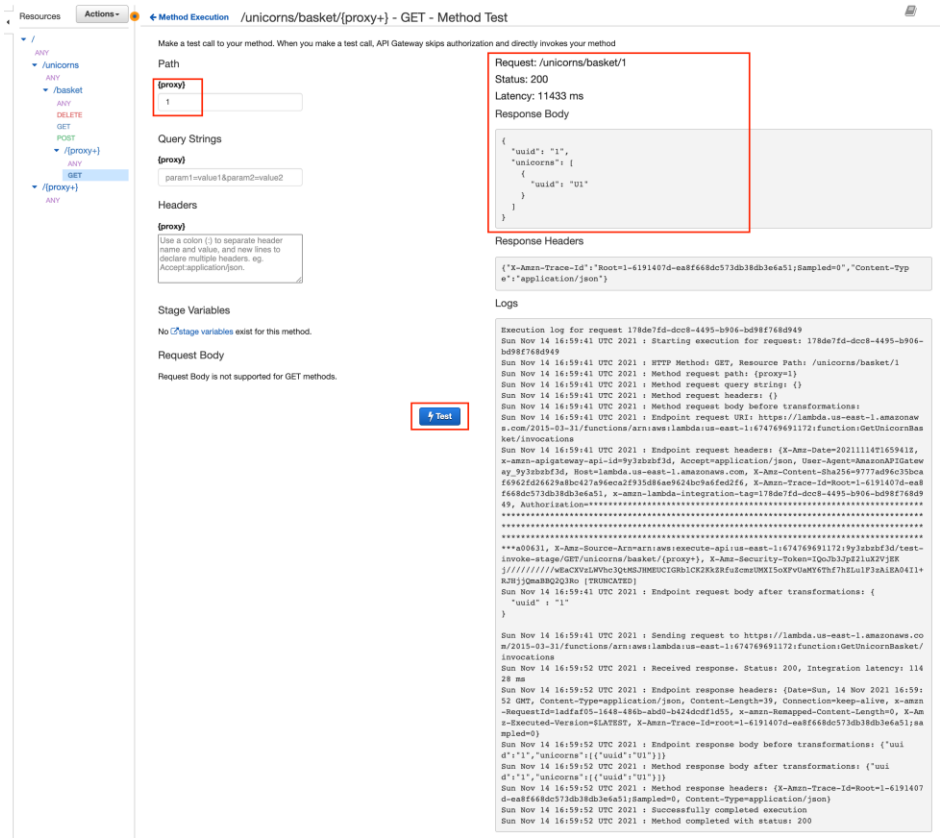


Get list of items in the shopping cart

111. Navigate to the configuration screen for the **GET** method on **/unicorns/basket/{proxy+}**.

112. Choose **Test** in the Client box on the right.

113. Recall in the previous step you used a uuid for a fictitious user of "1". Your test will retrieve the shopping cart for that user. In the text field for “{proxy}” in the “Path” section, enter the value 1 and choose Test.



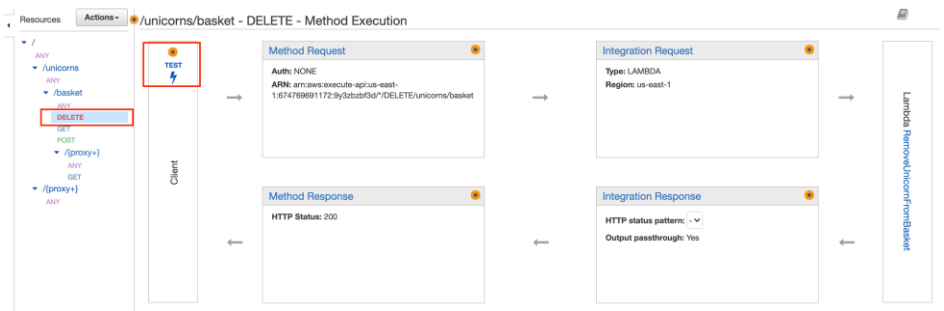
114. You should see the shopping cart retrieved with the following content in the response body:

```
{
  "uuid": "1",
  "unicorns": [
    {
      "uuid": "U1"
    }
  ]
}
```

Delete from the shopping cart

115. Navigate to the configuration screen for the **DELETE** method on **/unicorns/basket**.

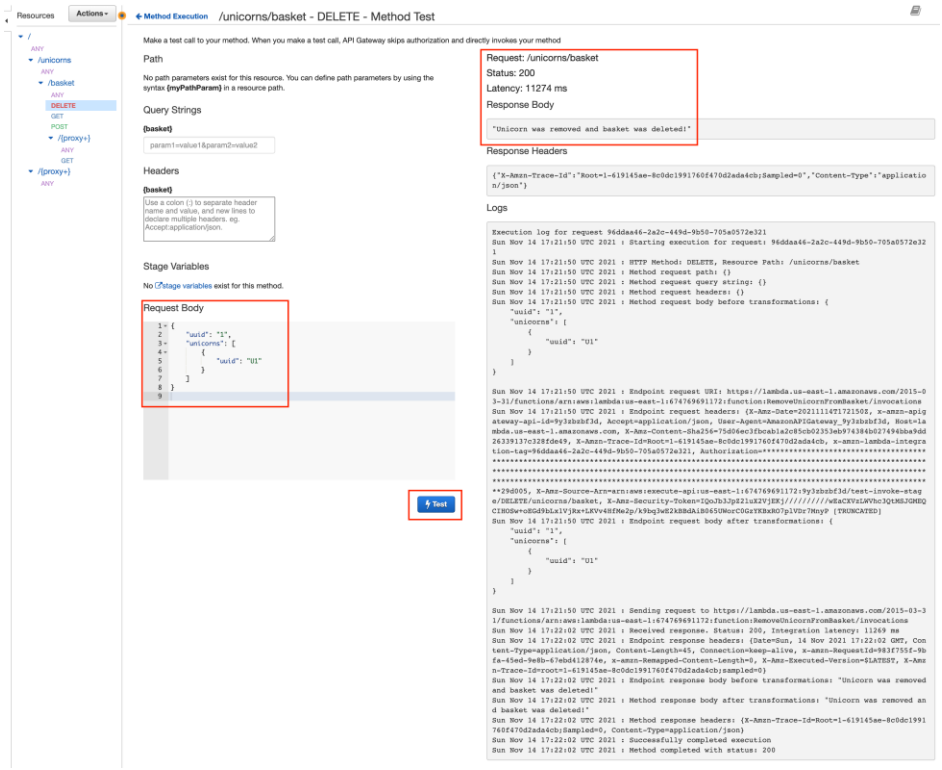
116. Choose **Test** in the Client box on the right.



117. In the **Request Body** box enter the following text and choose **Test**:

```
{
    "uuid": "1",
    "unicorns": [
        {
            "uuid": "U1"
        }
    ]
}
```

118. You should see in the response body that the Unicorn was removed:



Review

In part 3, you added the AWS Migration Hub Refactor Spaces service that represent the get/add/remove functionality of a shopping cart backed by your new Lambda functions that replaced the same functionality in the monolith. Refactor Spaces handled most of the setup for you, leaving only some minor configuration detail specific to your service implementation. You also briefly covered testing in API gateway to confirm your services are functioning as expected.

Modernizing legacy applications is a necessity, there are a few approaches you can follow. We hope that this lab highlighted the benefits of breaking the monolith using the strangler pattern. Whichever approach you decide to follow, we believe that the gradual improvement will benefit your business and increase confidence in delivering application modernization. Good luck with your journey!

End lab

Follow these steps to close the console and end your lab.

119. Return to the **AWS Management Console**.

120. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.

121. Choose End lab and then confirm that you want to end your lab.

Additional Resources

- For more information about how to use AWS Migration Hub, see AWS Migration Hub Documentation.

For more information about AWS Training and Certification, see *https://aws.amazon.com/training/*.

Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.