

Migrated On Premise Application to AWS

AWS services

Amazon CloudFront, AWS Aurora, RDS, Amazon Simple Storage Service (Amazon S3), IAM Roles, VPC, Internet gateway, Security Group, etc.

Description

Here we have an on-premise web server (in this project its hosted on EC2), static content was first transferred to S3 bucket and then content was delivered via CloudFront. Database was shifted from single instance to highly available and fault tolerant serverless database (Aurora).

#1 - VPC Setup

[Created VPC](#)

[Created Subnets](#)

[Securing Private Subnets](#)

[Adding a new public route](#)

[Create an IAM Role](#)

#2 - EC2 Servers

[Created SSH Certificate](#)

[Created E2 Instance](#)

[Access our Site](#)

[Adding Tags](#)

#3 - S3 &

[CloudFront](#)

[Objectives](#)

[Storing images on S3 Bucket](#)

[Copied images and updated to the server.](#)

[Serving using CloudFront](#)

#4 - VPC Setup

[Create a Database subnet group](#)

[Create a private security group](#)

[Added Aurora Database](#)

[Use secret manager](#)

[Gave server access to Secrets Manager](#)

[Moved the existing data and Update the server](#)

#1 - VPC Setup

In this site cloud network is being setup for resources used in entire project.

Created VPC

Created VPC named **tsgallery.vpc** which has IPv4 CIDR block **10.11.0.0/16**

We need to ensure checkbox for **DNS hostnames** is enabled.

vpc-0aeee17a512a307dd / tsgallery.vpc

Actions ▼

Details Info

VPC ID
vpc-0aeee17a512a307dd

Tenancy
Default

Default VPC
No

Network Address Usage metrics
Disabled

State
Available

DHCP option set
dopt-0c43908b65d09943e

IPv4 CIDR
10.11.0.0/16

Route 53 Resolver DNS Firewall rule groups
-

DNS hostnames
Enabled

Main route table
rtb-0121f83e48d89e66a

IPv6 pool
-

Owner ID
281985069075

DNS resolution
Enabled

Main network ACL
acl-05366133ee0bbc6a3

IPv6 CIDR (Network border group)
-

VPC details

VPC ID
vpc-0aeee17a512a307dd

Name
tsgallery.vpc

DHCP settings

DHCP option set Info
dopt-0c43908b65d09943e ▼

DNS settings

☒ Enable DNS resolution Info

☒ Enable DNS hostnames Info

Network Address Usage metrics settings

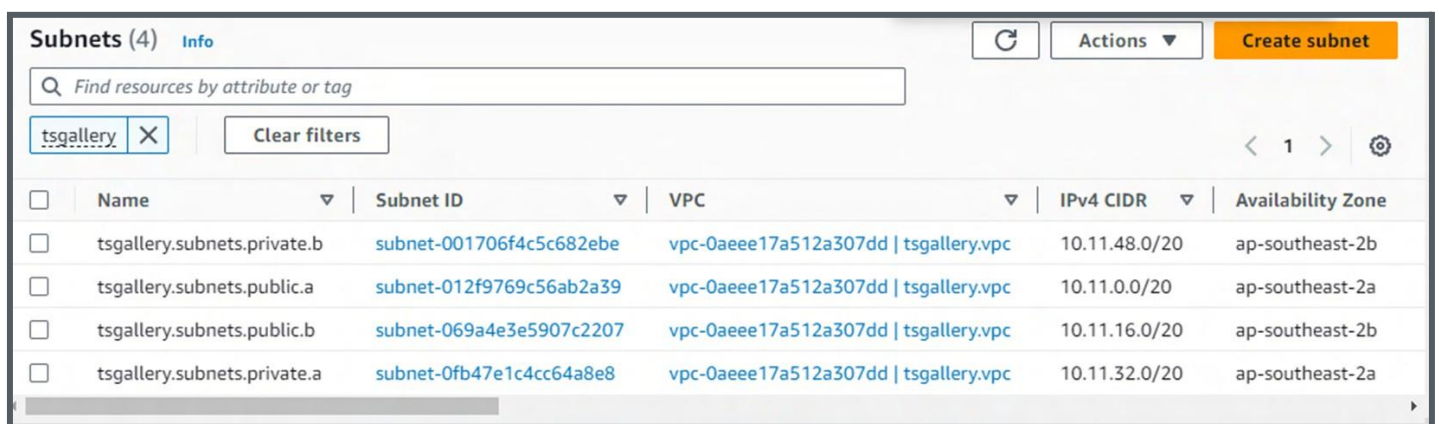
☐ Enable Network Address Usage metrics Info

The DNS hostnames attribute determines whether instances launched in the VPC receive public DNS hostnames that correspond to their public IP addresses.

Created Subnets

Created 4 new subnets, 2 were private and 2 public.
All subnets were associated with VPC created.

Subnet name	IPv4 CIDR Block	Availability Zone
tsgallery.subnets.public.a	10.11.0.0/20	*a
tsgallery.subnets.public.b	10.11.16.0/20	*b
tsgallery.subnets.private.a	10.11.32.0/20	*a
tsgallery.subnets.private.b	10.11.48.0/20	*b



The screenshot shows the AWS Management Console interface for the 'Subnets (4)' page. At the top, there's a search bar with the text 'Find resources by attribute or tag' and a filter tag 'tsgallery'. Below the search bar, there's a table with 6 columns: Name, Subnet ID, VPC, IPv4 CIDR, and Availability Zone. The table lists four subnets: tsgallery.subnets.private.b, tsgallery.subnets.public.a, tsgallery.subnets.public.b, and tsgallery.subnets.private.a. Each row includes a checkbox for selection, the subnet name, its ID, the associated VPC (vpc-0aeee17a512a307dd | tsgallery.vpc), its IPv4 CIDR block, and its availability zone.

<input type="checkbox"/>	Name	Subnet ID	VPC	IPv4 CIDR	Availability Zone
<input type="checkbox"/>	tsgallery.subnets.private.b	subnet-001706f4c5c682ebe	vpc-0aeee17a512a307dd tsgallery.vpc	10.11.48.0/20	ap-southeast-2b
<input type="checkbox"/>	tsgallery.subnets.public.a	subnet-012f9769c56ab2a39	vpc-0aeee17a512a307dd tsgallery.vpc	10.11.0.0/20	ap-southeast-2a
<input type="checkbox"/>	tsgallery.subnets.public.b	subnet-069a4e3e5907c2207	vpc-0aeee17a512a307dd tsgallery.vpc	10.11.16.0/20	ap-southeast-2b
<input type="checkbox"/>	tsgallery.subnets.private.a	subnet-0fb47e1c4cc64a8e8	vpc-0aeee17a512a307dd tsgallery.vpc	10.11.32.0/20	ap-southeast-2a

Both Public subnets needs to be assigned IP address
So Enabled **AUTO ASSIGN PUBLIC IP Address** for both public subnets.

VPC > Subnets > subnet-012f9769c56ab2a39 > Edit subnet settings

Edit subnet settings [Info](#)

Subnet

Subnet ID	Name
subnet-012f9769c56ab2a39	tsgallery.subnets.public.a

Auto-assign IP settings [Info](#)

Enable the auto-assign IP settings to automatically request a public IPv4 or IPv6 address for a new network interface in this subnet.

☒ **Enable auto-assign public IPv4 address** [Info](#)

☐ Enable auto-assign customer-owned IPv4 address [Info](#)
Option disabled because no customer owned pools found.

Created Internet Gateway

Created Internet gateway and attached to VPC - tsgallery.internetgateway

Details

Internet gateway ID igw-0535b3fc937ac153b	State Attached	VPC ID vpc-0aeee17a512a307dd tsgallery.vpc
--	-------------------	--

Securing Private Subnets

The default route table is renamed as Private, No subnet explicitly mentioned meaning it will assign this MAIN Default route table to newly created subnets in this VPC.

Route tables (2) [Info](#) Actions [Create route table](#)

tsgallery

< 1 >

<input type="checkbox"/>	Name	Route table ID	Explicit subnet associati...	Edge associations	Main
<input type="checkbox"/>	tsgallery.routetable.private	rtb-0121f83e48d89e66a	-	-	Yes
<input type="checkbox"/>	tsgallery.routetable.public	rtb-0429b41b4da5b6c...	2 subnets	-	No

Adding a new public route

In the Route Editor table provided 0.0.0.0/0 as Destination, selected Internet Gateway we created from the Target dropdown.

Associated both public subnets to the newly created public route table.

VPC > Route tables > rtb-0429b41b4da5b6c3b > Edit subnet associations

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (2/4)

Filter subnet associations

	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input type="checkbox"/>	tsgallery.subnets.private.b	subnet-001706f4c5c682ebe	10.11.48.0/20	-	Main (rtb-0121f83e48d89e66a)
<input checked="" type="checkbox"/>	tsgallery.subnets.public.a	subnet-012f9769c56ab2a39	10.11.0.0/20	-	Main (rtb-0121f83e48d89e66a)
<input checked="" type="checkbox"/>	tsgallery.subnets.public.b	subnet-069a4e3e5907c2207	10.11.16.0/20	-	Main (rtb-0121f83e48d89e66a)
<input type="checkbox"/>	tsgallery.subnets.private.a	subnet-0fb47e1c4cc64a8e8	10.11.32.0/20	-	Main (rtb-0121f83e48d89e66a)

Selected subnets

subnet-012f9769c56ab2a39 / tsgallery.subnets.public.a X subnet-069a4e3e5907c2207 / tsgallery.subnets.public.b X

Cancel Save associations

Create an IAM Role

This IAM role will be used by your application to interact with AWS Services.

Role is for EC2, Policies of **AmazonS3FullAccess** and **AmazonSSMManagedInstanceCore** is provided to Role.

Step 1: Select trusted entities

Trust policy

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "sts:AssumeRole"
8       ],
9       "Principal": {
10        "Service": [
11          "ec2.amazonaws.com"
12        ]
13      }
14    }
15  ]
16 }

```

Edit

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy
AmazonSSMManagedInstanceCore	AWS managed	Permissions policy

Edit

The SSMManagedInstanceCore is required to allow Systems Manager to open sessions directly on an EC2 instance and for its Patch Manager to apply security updates automatically for you

#2 - EC2 Servers

In this lab I created a new Amazon EC2 instance to host a web server.
 Installed web server, database and other support tools on that web server.
 Then test if it is working before migrating to AWS.

Created SSH Certificate

Created a new **KeyPair** for this EC2, Key pair is secure and convenient way to connect to EC2s.

Key pairs (1) [Info](#)

Search

Refresh

Actions

Create key pair

<input type="checkbox"/>	Name	Type	Created	Fingerprint	ID
<input type="checkbox"/>	tsa-ap-southeast-2	rsa	2023/09/17 18:28 GMT+5:30	7e:5d:64:78:cb:00:b3:bd:48:bf:af:8e:06:...	key-038b27b7ad2654aca

Created E2 Instance

Created webserver on EC2 Instance with instance type **m6g.medium**

Associated the key pair created in the previous step.

This on premise webserver is in the same VPC we created, where a public subnet is provided.

Public subnet associated here is **tsgallery.subnets.public.a**

Firewall Security Group associated had permission to allow HTTP requests access to the webserver from everywhere.

Associated **IAM Role** "tsgallery.roles.serverroles"

Added user data this script runs when the instance boots up for the first time.

```
#!/bin/bash
echo ==== Starting UserData Script ====
curl -k -o /root/setup.sh
http://labassets.awstechshift.cloud/prod/migrate/2/setup.sh
chmod +x /root/setup.sh
sudo -i /root/setup.sh
echo ==== Finished UserData Script ====
```

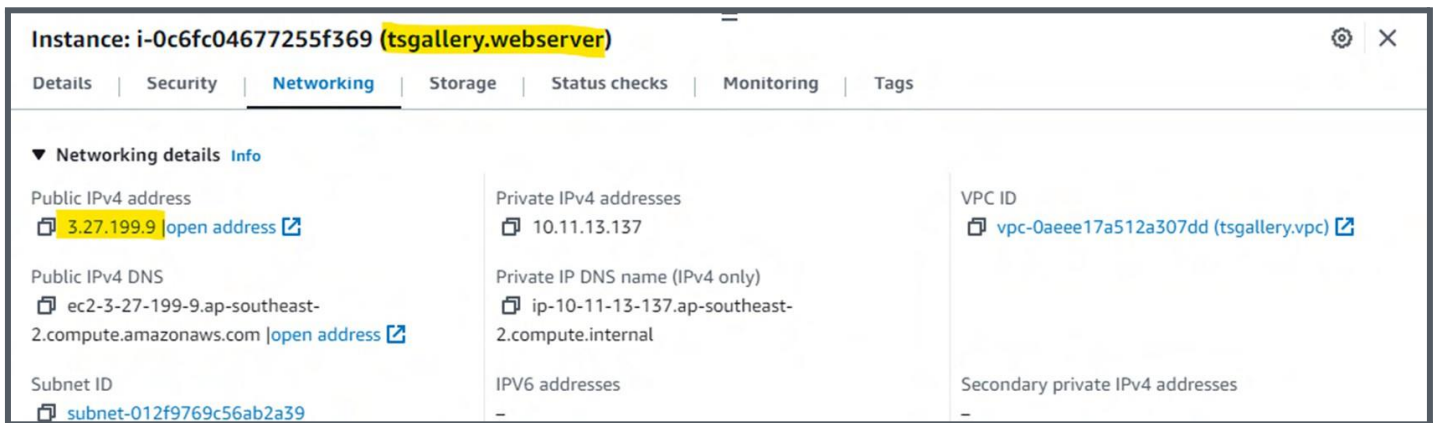
The setup script itself does the following:

- Update the server
- Download the files needed for your server
- Install MySQL and run the initialization SQL script
- Download and configure NodeJS
- Download Forever utility
- Install and configure NGinx
- Copy all the server files to the correct locations
- Start the server

Access our Site

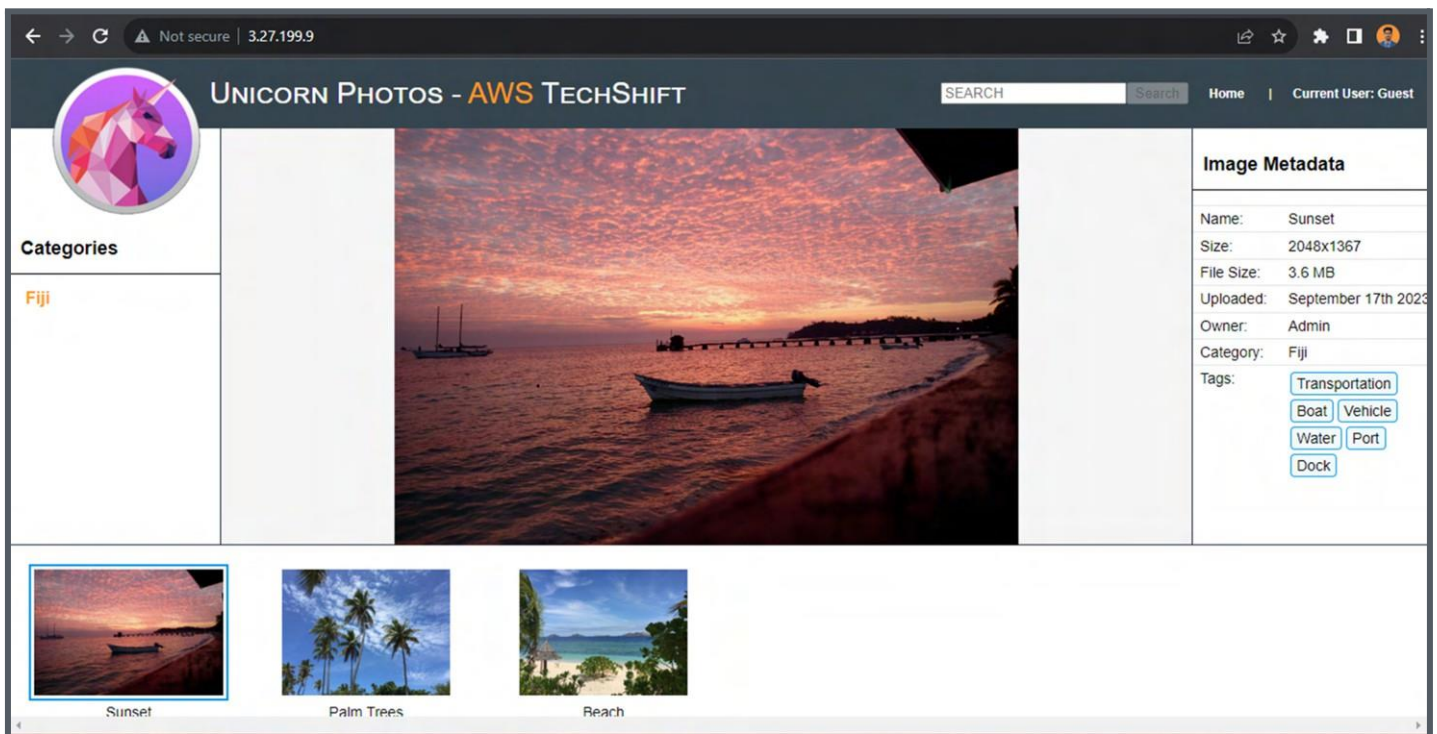
Site is accessible after the instance state is changed to RUNNING and the server completes running the User Data script.

Currently the firewall is configured to access via HTTP protocol so URI with HTTPS wont work here.



Here we can see a photo gallery application is running.

Administration page is accessible from the top right corner by clicking on **Current User: Guest**. Username is **admin** and the password is **awstechshift**



Adding a Tag to our Security Group

Tagging our resources is the best practice. There is no tags to the security group because it was created using the instance wizard template.

Tag added to security group here is **tsgallery.securitygroups.webserver**

search: tsgallery X		Clear filters	
<input type="checkbox"/>	Name ▾	Security group ID ▾	Security group name ▾
<input type="checkbox"/>	tsgallery.securitygr...	sg-01e1a9ac124658f1b	tsgallery.securitygroups.webserver
			vpc-0aeee17a512a307dd ↗

#3 - S3 & Cloudfront

Objectives

Here I moved the static portions of the web application from the WebServer to a S3 bucket which was served using Amazon CloudFront for distribution.

Storing images on S3 Bucket

Created S3 bucket in the same region where EC2 web server is running.

Buckets (1) Info		↻	Copy ARN	Empty	Delete	Create bucket
Buckets are containers for data stored in S3. Learn more ↗						
<input type="text" value="Find buckets by name"/>						
	Name ▲	AWS Region ▾	Access ▾	Creation date ▾		
<input type="radio"/>	tsgallery.281985069075	Asia Pacific (Sydney) ap-southeast-2	Bucket and objects not public	September 17, 2023, 23:50:54 (UTC+05:30)		

Copied images and updated to the server.

SSH session was initiated for EC2 web server, this was the step: Systems Manager > Session Manager > Start Session.

Below script is used to update the server.

```
sudo su -
cd /opt/ts_gallery
sh update.sh
```

Update Script does this:

- Copy the images to S3

Akash Sahu

LinkedIn : <https://www.linkedin.com/in/akashsahu97/>

- Remove the images from the server
- Update the server config to reference S3 bucket
- Update the filesystem server code to use S3
- Restart the server

```
[root@ip-10-11-13-137 ts_gallery]# cd /opt/ts_gallery
[root@ip-10-11-13-137 ts_gallery]# sh update.sh

=====
TechShift Migrate
=====

1. Update S3
7. Exit
Enter choice [1 to 7]: 1
-----
Starting S3 Migration
-----
Enter your bucket name and press <ENTER>: tsgallery.281985069075

s3_migrate: ==== Validating access to tsgallery.281985069075 ====
s3_migrate: Bucket s3://tsgallery.281985069075 found and permissions appears ok
s3_migrate: Lock file created
s3_migrate: ==== Moving images to tsgallery.281985069075 ====
s3_migrate: Backup images
s3_migrate: Copy images to S3

Images have now been deleted from the server. To confirm the files are not on the server,
return to the browser window with the photo gallery open and refresh it. You should see
the site load with broken images.

Press any key to continue updating server . . .
s3_migrate: ==== Update the server ====
s3_migrate: Add the AWS-SDK package
```

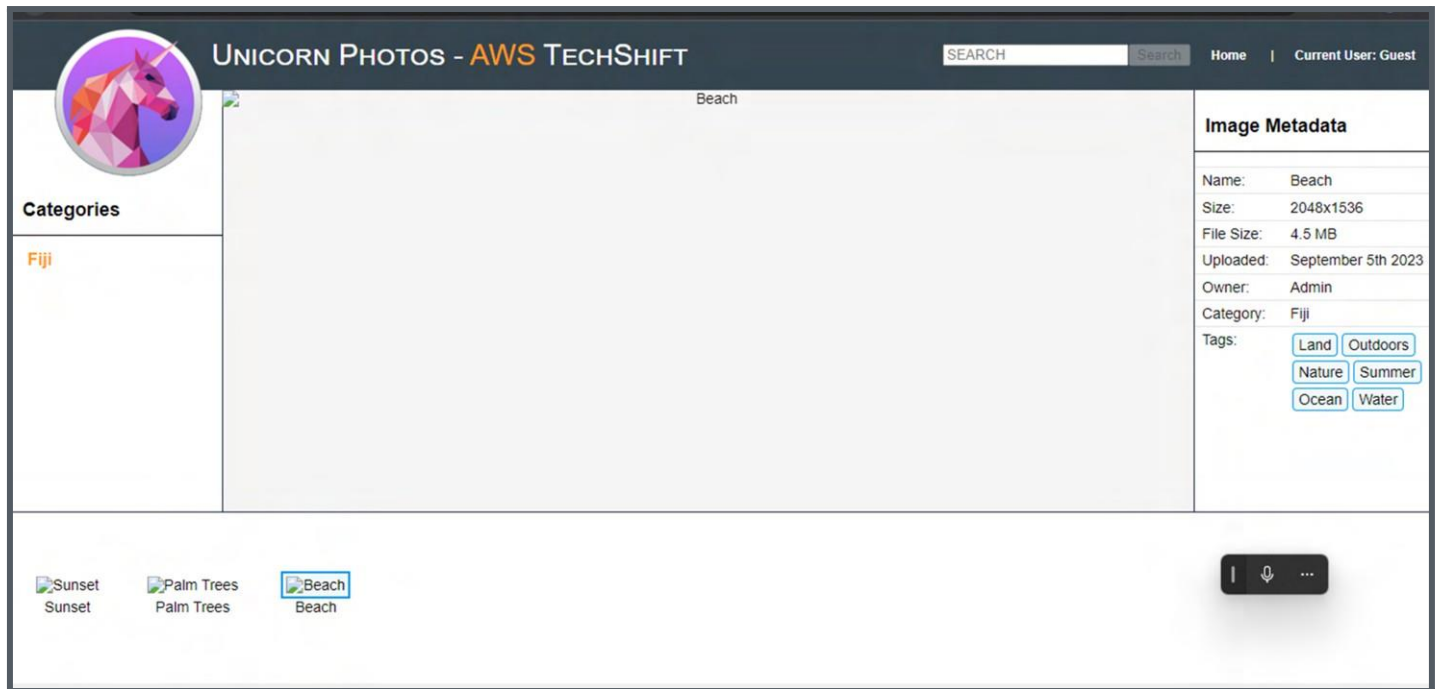
```
Enter choice [1 to 7]: 1
-----
Starting S3 Migration
-----
Enter your bucket name and press <ENTER>: tsgallery.281985069075

s3_migrate: ==== Validating access to tsgallery.281985069075 ====
s3_migrate: Bucket s3://tsgallery.281985069075 found and permissions appears ok
s3_migrate: Lock file created
s3_migrate: ==== Moving images to tsgallery.281985069075 ====
s3_migrate: Backup images
s3_migrate: Copy images to S3

Images have now been deleted from the server. To confirm the files are not on the server,
return to the browser window with the photo gallery open and refresh it. You should see
the site load with broken images.

Press any key to continue updating server . . .
s3_migrate: ==== Update the server ====
s3_migrate: Add the AWS-SDK package
s3_migrate: Backup config to config.s3.bak.js
s3_migrate: Add region and secret name to the aws block in config.js
s3_migrate: ==== Update the server code to use S3 ====
s3_migrate: Backup www to www.s3.bak.js
s3_migrate: Update dependancy injected file system with s3 version
s3_migrate: Restarting server
s3_migrate: Server update done
-----
S3 Migration Finished
-----
```

At this step if we refresh the web page we should see the broken images as seen in the below snap.



Serving using CloudFront

Here I did set up a CloudFront distribution to serve the static assets and route the API requests to the server.

In the Origin Settings section entered the website URI (EC2 instance's Public IPv4 DNS), into the Origin Domain and Enable Origin Shield was set as No.

In the **Default cache behavior** section, under Allowed HTTP Methods selected GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE. Ensure Cache policy and origin request policy is selected and selected Caching Disabled from the Cache Policy dropdown.

CloudFront > Distributions > Create

Create distribution

Origin

Origin domain
Choose an AWS origin, or enter your origin's domain name.

Q ec2-3-27-199-9.ap-southeast-2.compute.amazonaws.com X

Protocol **Info**

☐ HTTP only

☒ HTTPS only

☐ Match viewer

HTTP port
Enter your origin's HTTP port. The default is port 80.

80

HTTPS port
Enter your origin's HTTPS port. The default is port 443.

443

In the Default root object text box, specify **index.html** and click Create Distribution.

At this stage, the distribution only has the server in it. we need to add the images from S3.

With the proper distribution ID, we will create Origin.

For Origin Domain selected the S3 bucket.

In the Origin access section > Legacy access identities. Create a new OIA by clicking on **Create new OIA**. In the Bucket policy section, selected **Yes, update the bucket policy**.

Here now created new behavior that links to new origin. Clicked the **Behaviors tab**, then click **Create Behavior**.

Entered images/uploads/*.* as the Path Pattern and selected S3 bucket for the Origin or Origin Group.

Now we need to get the new cloudfront URI for our photo gallery. If we receive an error stating that our connection is not private, or get an AccessDenied message, S3 is still updating its internal DNS. Our setup is correct, we need to wait until the update completes.

#4 - VPC Setup

Here I moved the database off the single instance to highly available and fault tolerant serverless database.

- Created an Aurora Serverless database
- Exported data from your single instance
- Imported data into Aurora Server-less
- Updated the app code to use the new database

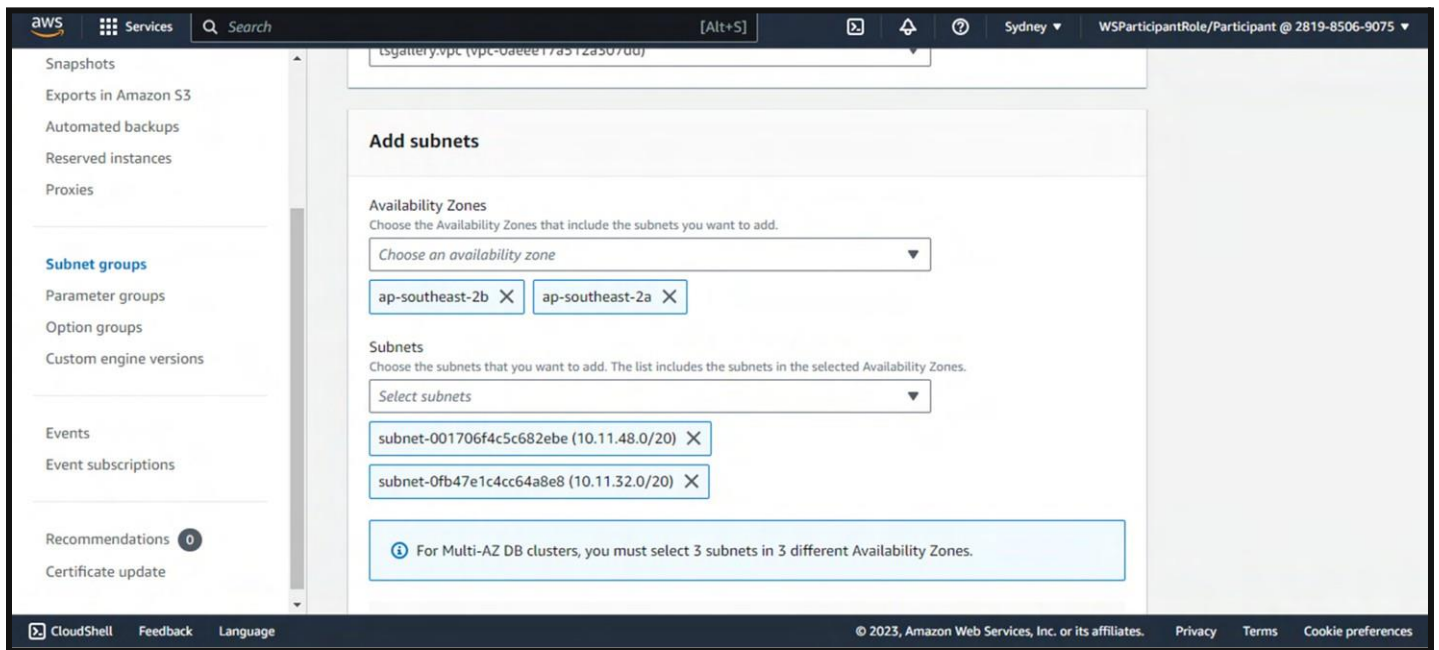
Create a Database subnet group

An RDS database needs to be located in a database subnet group. A database subnet group is simply a set of VPC subnets that the database will use to host the various nodes and endpoints.

Navigated to **RDS Service > Subnet Groups > Create DB Subnet Group**
I named it as tsgallery.dbsubnetgroup

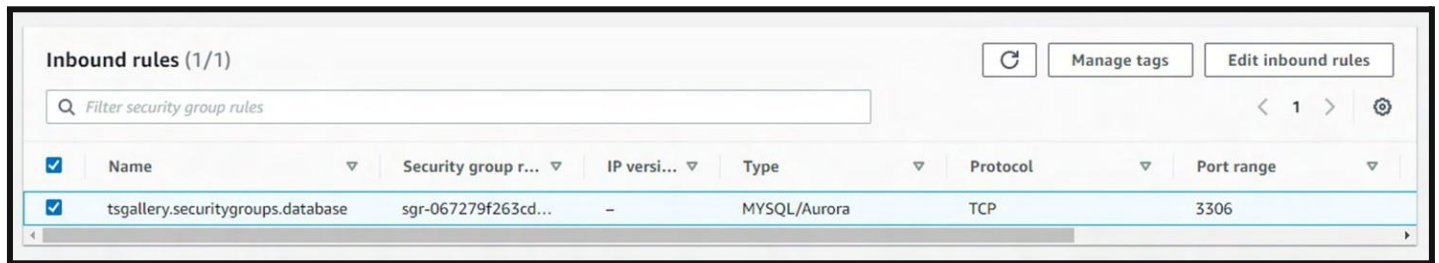
The screenshot shows the AWS Management Console interface for creating a DB subnet group. The left sidebar contains navigation links for various AWS services. The main content area is titled 'Create DB subnet group' and includes a 'Subnet group details' section. The 'Name' field is filled with 'tsgallery.dbsubnetgroup', the 'Description' field is filled with 'TSGallery DB Subnet Group', and the 'VPC' dropdown is set to 'tsgallery.vpc (vpc-0aeee17a512a307dd)'. The page also includes a search bar at the top and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

Here I associated the both private subnets I had created in the initial stage.



Create a private security group

Created a new private Security Group to accept inbound requests of type MYSQL/Aurora.



Added Aurora Database

- Navigated to RDS Service > Databases > **Create Database**.
- **Standard Create** needs to be selected with **Amazon Aurora** as the engine type.
- Then **Amazon Aurora with MySQL compatibility**

Templates

Choose a sample template to meet your use case.

☐ Production
Use defaults for high availability and fast, consistent performance.

☒ Dev/Test
This instance is intended for development use outside of a production environment.

Settings

DB cluster identifier [Info](#)

Enter a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.

The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

Master username [Info](#)

Type a login ID for the master user of your DB instance.

1 to 32 alphanumeric characters. The first character must be a letter.

☒ **Auto generate a password**
Amazon RDS can generate a password for you, or you can specify your own password.

- **DB cluster identifier - tsgallery-dbcluster**
- Checked the **Auto generate a password** option
- In the Instance Configuration section, select the **Burstable classess** instance and leave the default value of **db.t3.small** as the instance type.
- VPC here will be **tsgallery.vpc** and DB Subnet group is set to **tsgallery.dbsubnetgroup**

☒ Don't connect to an EC2 compute resource
 Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

☐ Connect to an EC2 compute resource
 Set up a connection to an EC2 compute resource for this database.

Network type [Info](#)
 To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

☒ IPv4
 Your resources can communicate only over the IPv4 addressing protocol.

☐ Dual-stack mode
 Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) [Info](#)
 Choose the VPC. The VPC defines the virtual networking environment for this DB cluster.

tsgallery.vpc (vpc-0ae17a512a307dd)
 4 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

DB subnet group [Info](#)
 Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB cluster can use in the VPC that you selected.

tsgallery.dbsubnetgroup
 2 Subnets, 2 Availability Zones

Public access [Info](#)
☐ Yes
 RDS assigns a public IP address to the cluster. Amazon EC2 instances and other resources outside of the VPC can connect to your database instance.

- Security group should be **tsgallery.securitygroups.database**, then deselected the **Default** security group.
- Under Monitoring, unchecked the **Enable Enhanced monitoring**.
- Unchecked the **Enable deletion protection**.
- Finally here hit the **Create database**.
- In the **View credential details** button we will get the auto generated credentials.
- Cluster details can be found in **DB Identifier**. Endpoint and port values can be seen in **Connectivity & security** tab.

RDS > Databases > tsgallery-dbcluster

tsgallery-dbcluster

Modify Actions

Related

Filter by databases

DB identifier	Status	Role	Engine	Region & AZ	Size
<input checked="" type="radio"/> tsgallery-dbcluster	Available	Regional cluster	Aurora MySQL	ap-southeast-2	1 instance
<input type="radio"/> tsgallery-dbcluster-instance-1	Available	Writer instance	Aurora MySQL	ap-southeast-2a	db.t3.small

Use secret manager

- Navigated to **Secrets Manager > Store a new secret.**
- Option **Credentials for other database** is selected and enter the **User name and Password.**
- Selected **MySQL** for the **Select which database this secret will access** option.
- Provided the **Endpoint** and **Port** of Aurora database.
- Database name as **tsgallerydata.**
- Enter **tsgallery.secrets.dbcluster** as the Secret name and the Description.

aws/secretsmanager

Add new key

Database Info

☐ MariaDB

☒ MySQL

☐ PostgreSQL

☐ Oracle Database

☐ SQL Server

Server address: tsgallery-dbcluster-insta

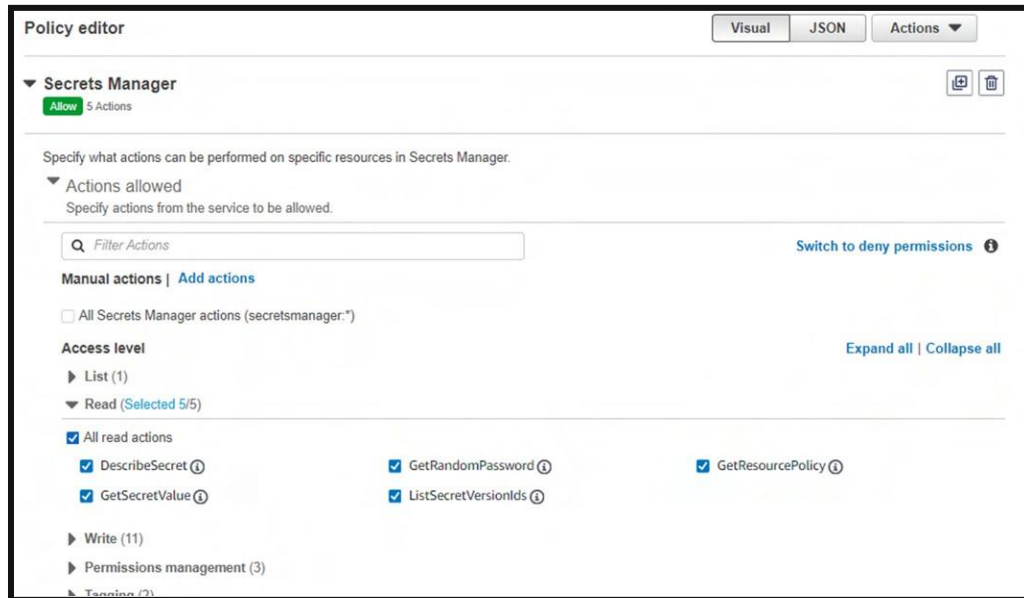
Database name: tsgallerydata

Port: 3306

Gave server access to Secrets Manager

Adding access so that server can read secrets, this is done by updated IAM role with new permissions.

- Selected **Add permissions** and then option of **Create inline policy**.
- Selected **Secrets Manager** Service with **Read** checkbox and **All resources** needs to be selected.
- This new policy is named as **tsgallery.policies.readsecrets**



Moved the existing data and Update the server

Update script is used to migrate data and update the server.

```
sudo su -  
cd /opt/ts_gallery  
sh update.sh
```

The update.sh script will do the following:

- Setup new schema in Aurora
- Copy data from local MySQL server
- Update the server config to reference the secret and remove hard coded database credentials
- Update the filesystem server code to use secret to get credentials
- Restart the server
- Shutdown MySQL server on local host as it's no longer needed

```

TechShift Migrate
=====

2. Rollback S3
3. Update RDS
7. Exit
Enter choice [1 to 7]: 3
-----
Starting RDS Migration
-----
Enter your secret name and press <ENTER>: tsgallery.secrets.dbcluster

rds_migrate: Retrieve secret tsgallery.secrets.dbcluster from ap-southeast-2
rds_migrate: ==== Moving data to tsgallery-dbcluster-instance-1.cb0gku2mg9ez.ap-southeast-2.rds.amazonaws.com ====
rds_migrate: Lock file created
rds_migrate: Create new schema tsgallerydata on tsgallery-dbcluster-instance-1.cb0gku2mg9ez.ap-southeast-2.rds.amazonaws.com
rds_migrate: Export data from MySQL localhost
rds_migrate: Push data to tsgallery-dbcluster-instance-1.cb0gku2mg9ez.ap-southeast-2.rds.amazonaws.com
rds_migrate: ==== Update the server config ====
rds_migrate: Backup config to config.rds.bak.js
rds_migrate: Add secret name to the aws block in config.js
rds_migrate: Removing un-needed db configuration block from config.js
rds_migrate: ==== Update the server code to retrieve database credentials before creating database pool ====
rds_migrate: Backup app to app.rds.bak.js
rds_migrate: Removing existing database configuration block
rds_migrate: Adding SecretsManager enabled database configuration block
rds_migrate: Restarting server
rds_migrate: Disable MySQL currently running on server
rds_migrate: Removing lock files
-----
RDS Migration Finished
-----
[root@ip-10-11-13-137 ts_gallery]#

```

```

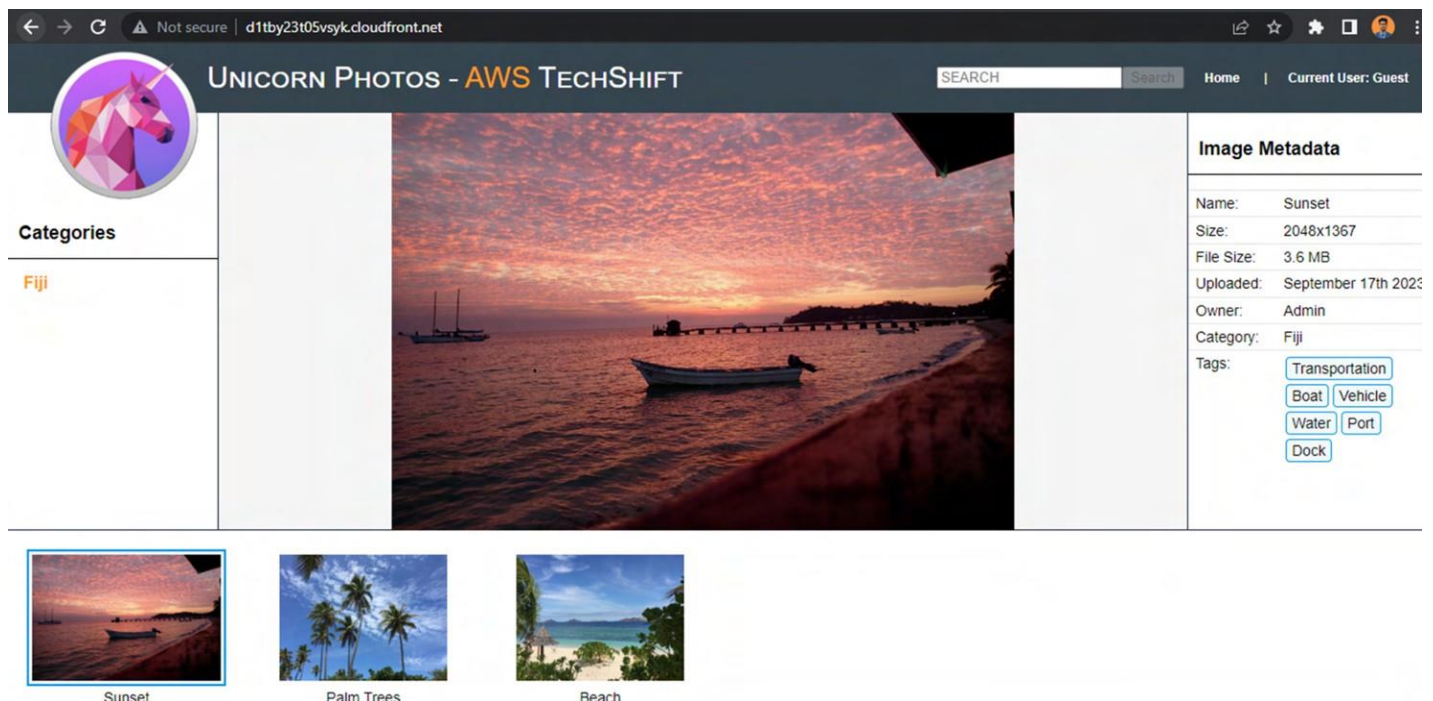
TechShift Migrate
=====

4. Rollback RDS
5. Update Elastic Beanstalk
7. Exit
Enter choice [1 to 7]: 5
-----
Starting Elastic Beanstalk Setup
-----
eb_migrate: ==== Validating access to tsgallery.281985069075 ====
eb_migrate: Bucket s3://tsgallery.281985069075 found and permissions appears ok
eb_migrate: Lock file created
eb_migrate: Create Procfile to control NodeJS startup
eb_migrate: Create .ebextensions folder
eb_migrate: Create disable-npm.config to prevent 'npm rebuild'
eb_migrate: Zip server files for deployment
.
.....
.....
.

```



```
.  
.  
  
eb_migrate: Upload deployment package to S3  
eb_migrate: Removing lock files  
-----  
Elastic Beanstalk Setup Finished  
-----  
[root@ip-10-11-13-137 ts gallery]#
```



So now we have setup a serverless database cluster, migrated our data into the new cluster and updated the server to use the new cluster.