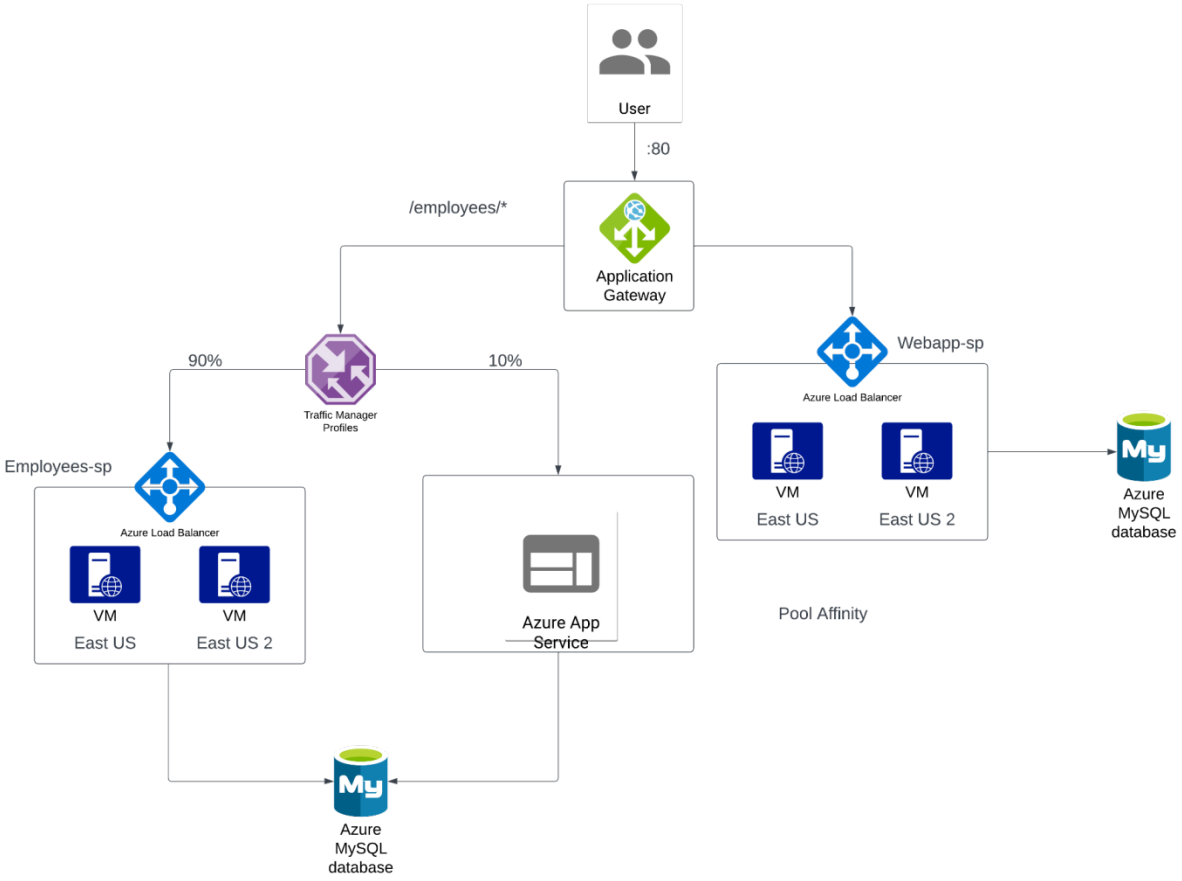
Application Modernization on Cloud

The Strangler Fig Pattern refers to an application modernization strategy where we incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services. As features from the legacy system are replaced, the new system eventually replaces all of the old system's features, strangling the old system and allowing you to decommission it.

Problem Statement



Setup the above architecture on Azure using the Strangler fig pattern. The following parameters should be covered in this implementation

1) Lift and Shift scenario
2) Canary release principles
3) Public/Private load balancers
4) Avoid session affinity for better scaling
5) Applied elasticity
6) Serverless deployment

Deploy the below script to the virtual machines created as part of the architecture. The VMs must be running Ubuntu to be able to run the given script.

Steps to perform this exercise:

1) Create a virtual network along with 3 subnets. You can use the below CIDRs for this
    a) Virtual Network - 10.0.0.0/16
    b) Subnet 1 - 10.0.1.0/24
    c) Subnet 2 - 10.0.2.0/24
    d) Subnet 3 - 10.0.3.0/24

2) Create an Azure SQL deployment based on MySQL. The flexible server option is recommended here
    a) Note down the following values while creating the database
        i) Endpoint URL
        ii) Database server username
        iii) Database password
        iv) Database name
    b) Ensure that the firewall for the database is open for public connections from all sources and can be accessed by other Azure services

3) Deploy a Virtual Machine Scale Set (VMSS) into Subnets 1 and 2.
    a) Ensure that they have attached load balancers with public IP addresses.
    b) Deploy the below bootscript into the custom-init field for the webapp-sp scale set in the architecture diagram

Note : For simplicity, this application does not use the database, but it could.

```
#!/bin/bash
APP_NAME=LiftShift-Application
apt update -y && apt -y install python3-pip zip
cd /opt
wget https://d6opu47qoi4ee.cloudfront.net/loadbalancer/simuapp-v1.zip
unzip simuapp-v1.zip
rm -f simuapp-v1.zip
sed -i "s=MOD_APPLICATION_NAME=$APP_NAME=g" templates/index.html
pip3 install -r requirements.txt
nohup python3 simu_app.py >> application.log 2>&1 &
```

c) Deploy the below bootscript into the custom-init field for the Employees-sp scale set in the architecture diagram. Ensure that the relevant database variables are edited in the script.

```
#!/bin/bash

# Define variables
DB_USER=myuser
DB_PASS=mypassword
DB_NAME=mydatabase
URL=url

# Update packages and install required software
apt-get update
apt-get install -y apache2 php mysql-client

cd /opt
mkdir crud
cd /opt/crud

# Download the PHP application code
wget https://d6opu47qoi4ee.cloudfront.net/labs/option3/index.php

# Update the database connection details in the application configuration file
sed -i "s/DB_USER/$DB_USER/g" index.php
sed -i "s/DB_PASS/$DB_PASS/g" index.php
sed -i "s/DB_NAME/$DB_NAME/g" index.php
sed -i "s/DBServer/$URL/g" index.php

cp index.php /var/www/html

# Create the database tables
wget https://d6opu47qoi4ee.cloudfront.net/employees.sql
mysql -h $URL -u $DB_USER -p$DB_PASS $DB_NAME < employees.sql

# Update Apache configuration
sed -i 's/DirectoryIndex index.html/DirectoryIndex index.php index.html/' /etc/apache2/mods-enabled/dir.conf

# Restart Apache
systemctl restart apache2
```

d) Ensure that port 80 is open for all incoming connections in the Network Security Groups of each of the scale sets.

e) Test the deployment on each of the scale sets using the public IP address of the load balancer

4) Deploy a web application using Azure App service
   a) Runtime stack: Select PHP 8.0.
   b) Operating system: Select Linux.
   c) Region: East US
   d) App Service Plan: Create an app service plan named myAppServicePlan.
   e) Use the code package provided with this exercise and import it into the above app service using any of the supported methods and repositories.
   f) Ensure that the database variables are edited in the index.php file in the above repository

5) Deploy a Traffic Manager profile
   a) Use the Weighted Routing Method
   b) Add the Employees-sp scale set as an endpoint with a weight of 90
   c) Add the App Service plan created above as an endpoint with a weight of 10

6) Deploy an Application Gateway into Subnet 3 created above
   a) Create a new public IP address as part of the FrontEnd configuration
   b) Create a new BackEnd Pool for the webapp-sp scale set
   c) Create a new BackEnd Pool for the Traffic Manager Profile created above using the DNS name of the profile. Remember to remove the **http://** prefix from the DNS name.
   d) Add a new routing rule using the following configuration
      i) Add a new listener using the port 8080
      ii) Under BackEnd Targets, select the BackEnd Pool created from the scale set marked as webapp-sp in the architecture diagram
      iii) Create a new HTTP setting using the default values
      iv) Under Path-based routing, select Add multiple targets to create a path-based rule.
         (1) For Path, type /employees/*.
         (2) For Target name, type employees.
         (3) For HTTP setting, select the HTTP setting created above
         (4) For Backend targets, select the BackEnd Pool created from the Traffic Manager profile

7) Test the deployment by accessing the public IP of the Application Gateway in the web browser. Also append /employees/123 to the IP to access the other deployments.

Reference Links
1) Load Balancers in Azure
2) Path based routing in Azure

3) Weighted Traffic Routing method
4) Strangler Fig Pattern
5) Strangler Fig Pattern on Azure