**Exercise: Migrating Web Applications to Azure Web App**

The book review app is a simple HTML and jquery application with no other external dependency.

For this exercise, you will be migrating/deploying the application to Azure Web App after creating all the required Azure resources.

1. Download the web app code [here](https://video.udacity-data.com/topher/2020/June/5edfe399_web/web.zip)
2. Browse the index.html page locally
3. Create a resource group with the following name: book-xxx-rg (xxx represent a random number of your choice)
4. Create the App Service Plan with the following settings:
   * Name: book-xxx-asp (xxx here should match the number of your resource group to keep things organized)
   * OS: Windows
   * Region: region closest to you
   * Pricing Tier: Free F1
5. Create the Azure Web App resource with the following settings:
   * name: book-xxx (xxx here should match the number of your resource group to keep things organized)
   * Publish from code (not docker container)
   * Runtime stack: ASP.NET V4.7
   * Region: same region as the app service plan
   * App Service Plan: select the plan created above
6. Deploy the web app to Azure via VS Code
7. Cleanup and delete resources
   * Be sure to clean up and delete resources to avoid recurring charge

## Exercise: Lift and Shift Migration with Virtual Machine Scale Sets (VMSS)

Create a Virtual Machine Scale using Cloud-init to bootstrap a hello world app. As we learned, creating custom images is the recommended path to deploy applications in VM scale set in a production environment. However, the purpose of this exercise is to have a hands-on experience with all the services required to create a Virtual Machine scale set.

1. Download the cloud-init.txt file [here](https://video.udacity-data.com/topher/2020/June/5eebce49_starter-files/starter-files.zip)
2. Create a resource group with the following settings:
   * name: scaleset-xxx-rg (replace xxx with random numbers)
   * location: closest region to you
3. Create a virtual machine scale set with the following requirements:
   * name: scaleset-xxx (xxx here should match the numbers used in your resource group for organization purposes)
   * image: UbuntuLTS
   * VM Size: costing less than $10 or sku: Standard\_B1ls if pricing details is not available in your region
   * OS Disk Type: Premium SSD
   * Load Balancer Type: Azure Load Balancer
   * Admin username: azureuser
   * Custom-data: Use the content in cloud-init.text
4. Update the Azure load balancer with rule to allow web traffic on port 80 for both backend and frontend
5. Find the IP Address and visit the address using your web browser to verify your scale set is configured
6. Download the starter files [here](https://video.udacity-data.com/topher/2020/June/5eebcefc_starter-files/starter-files.zip)
7. Create the Azure Resource Group
   * name: vm-xxx-rg (where xxx is a random number)
   * location: closest region
8. Virtual Network is shared between the 2 VMs and application gateway:
   * A virtual network with address prefix: 10.0.0.0/16
   * 2 subnet prefix:
     + application gateway subnet address prefix: 10.0.1.0/24
     + backend for VM subnet address prefix: 10.0.2.0/24
9. Create 2 virtual machines with the following conditions:
   * names: vm1 and vm2
   * image: UbuntuLTS
   * admin username: book\_admin
   * Use the content in cloud-init.txt for the cloud init custom data which will install prerequistes
   * VM Size: costing less than $10 or sku Standard\_B1ls
   * Each VM will have a public IP address to allow ssh communication for deployment
10. For each Virtual Machine, deploy content to the Website using SCP at following directory /home/book\_admin/web:

e.g: scp -r ./web/ <\*\*adminUsername\*\*>@<\*\*vmIpAddress\*\*>:/home/<\*\*adminUsername\*\*>/web

1. Create an application gateway:
   * Name: ag-xxx (xxx should match the number set in your resource group to stay organized)
   * Tier: Standard\_v2
   * Minimum Scale unit: 2
   * Create a public IP address
   * Use the Virtual Network created for the VMs
   * On the frontend settings, create a public IP
   * On the backend settings, add the 2 virtual machines created in Step 2
   * Configure routing rules to allow HTTP/port 80 communication to the backend
2. Get the Application Gateway IP Address to browse the application

Migrating to Azure SQL

In this exercise, you are given a SQL migration script. You will be migrating to Azure SQL and running the SQL script to migrate the schema and data.

Download the SQL migration script here

Create a resource group

Name: sql-xxx-rg (where xxx is a random number)

Region: closest region

Create SQL Server

Name: db-server-xxx (xxx should match the number set in your resource group)

username: sql\_admin

password: P@ssword

Allow public connectivity

Enable Access to all Ip Addresses

Create Database Server

Name: MyDatabase

Run the schema-data-script.sql by connecting to the Database using one of these options:

the Azure portal: Query Editor

Visual Studio Code SQL Server extension

Microsoft SQL Server Management Studio

**Manual Migration with Backup and Restore**

In this exercise, you will be creating a local environment with a schema and some data in a database. Afterward, you will create a back up of the database that you will later use to restore in Azure.

**Local Database Environment**

1. Install and configure the Postgres database locally:
   * If you have not already installed **Postgres**, you can find install instructions [here](https://www.postgresql.org/download/)
2. Log in to PGadmin portal and create a database:
   * Name: bookreviewdb
3. Add a Table with the following definitions:
   * Table Name: review
   * Columns:
     + id: integer, primary key, identity with auto increment, not null
     + name: text, not null
     + email: text, not null
     + job\_position: text, not null
     + company: text, not null
     + review: text, not null
4. Manually add 3 reviews to the database via the admin portal:
   * Review 1:
     + name: Enoch Josh
     + email: [enoch@josh.com](mailto:enoch@josh.com)
     + job\_position: Medical Chief Officer
     + company: Mouelet Medical Center
     + review: I recommend this book to all my medical students because lessons, stories, advice from this artistic work applies both in engineering as well as in the medical field.
   * Review 2:
     + name: Lily Michele
     + email: [lily@michele.com](mailto:lily@michele.com)
     + job\_position: Chief Data Scientist
     + company: TOKO LLC
     + review: I wish I had a role model like her to outline my goals and dreams in order to avoid the common mistakes of a young and enthusiastic engineer. Buy, read, and offer a copy to someone younger than you!
   * Review 3:
     + name: Laidry Arian
     + email: [laidry@arian.com](mailto:laidry@arian.com)
     + job\_position: CEO
     + company: MabsInsvestment
     + review: This a classic GPS for this generation; it outlines what to do, what not to do, when to do it, how to do it, and why taking the risk is the greatest legacy you can ever give yourself.
5. Create a backup of the reviewdb database and save it locally

**Azure Environment**

1. Create an Azure resource group
2. Create an Azure Postgres Database Service:
   * Server Type: Single Server Postgres
   * Server Version: select the latest
   * Compute and Storage: Basic, with 1 vcore, and 5GB of storage
   * Allow access to all IP Address range
3. Restore the recently created backup from reviewdb to the Azure Postgres database

**Create an Azure Service Bus and Storage Queue**

In this exercise, you will be creating two of the most popular services used in background task architecture.

**Service Bus**

1. Create a Resource Group
2. Create An Azure Service Bus
   * **Pricing Tier**: Basic
3. Create an Azure Service Bus Queue
   * **Queue Name**: reviewqueue
4. Find and locate your Service Bus namespace primary connection string.

**Storage Queue**

1. Create an Azure Storage Account in the same resource group:
   * **Performance**: Standard
   * **Replication**: Locally Redundant
   * **Networking**: all networks allowed
2. Create an Azure Storage Queue
   * **Name**: orderqueue
3. Find and locate your storage connection string

## Create an Azure WebJob

In this exercise, you'll create an Azure web app resource then you will set up a Triggered WebJob for the web app that will fetch the Azure Announcements RSS feed every 30 sec and store the 5 newest announcements to later display to the user.

1. Download WebJob file [here](https://video.udacity-data.com/topher/2020/June/5eebb268_starter-files/starter-files.zip)
2. Create an Azure Web App:
   * OS: Windows OS
   * Runtime: .Net 4.7 or higher
   * Pricing Tier: Free tier app service plan
3. Upload the web job to the Azure Web App
   * The web job zip file is located: in webjob\AzureAnnouncer.zip
   * This background task fetched the Azure Announcements RSS feed every 30 sec then store the 5 newest announcements to later display to the user
   * Web Job Type: Triggered
   * Triggers: Scheduled
   * CRON Expression: 0/30 \* \* \* \* \*

**Create an Azure Function**

In this exercise, you will be creating an Azure function triggered by an HTTP.

1. Create a Function App named ReviewFunctionApp with one function Review
2. Update the \_\_init\_\_.py file with the contents of this [file](https://video.udacity-data.com/topher/2020/June/5eebd037_function-code/function-code.zip).
3. Run the function application locally and ensure no errors
4. Deploy the function to Azure
   * Function App: myreviewapp
   * Language: Python
   * Hosting Plan: Consumption Plan
5. Test your Azure function with the Azure URL on the browser

## Migration using Azure Batch

In this exercise, you will gain experience in deploying applications and starting tasks in Azure Batch via the portal.

1. Download the StockRecommender.zip file [here](https://video.udacity-data.com/topher/2020/June/5eebb966_stockrecommender/stockrecommender.zip)
2. Create a new Azure Batch Account with a resource group
   * **Resource Group Name**: c3-lesson4-exercise-4
   * **Batch Account Name**: stockrecommenderxxx
   * Connect the Batch Account to an existing or new Storage
3. Upload the StockRecommender.zip file an Azure Batch application.
   * **Note** - Before uploading the zip, ensure the storage account is linked
   * **Name**: stockrecommender
   * **Version**: 1.0.0
4. Create an Azure Batch Pool
   * **Pool ID**: regular-vm
   * **Image Type**: Market Place
   * **Publisher**: canonical
   * **Offer**: ubuntuserver
   * **SKU**: the latest
   * **VM Size**: Standard A1
   * **Target Dedicated Nodes**: 1
   * **Application Package**: the recently added StockRecommender application
5. Create Batch Job
   * **Name**: stock-job
   * **Pool**: regular-vm created above
6. Create a task on the newly created job
   * **Task ID/Name**: stockrecommendertask1
   * /bin/sh -c $AZ\_BATCH\_APP\_PACKAGE\_stockrecommender\_1\_0\_0/StockRecommender
7. Review the stdout.txt output of the task to view the recommended stock