# Graphical user interface, text, email Description automatically generatedExploring Azure Container Apps - Exercises

John Ruzick – Principal Consultant – Improving

February 11, 2022

### Overview

This document will walk you through the steps to create a very simple application that leverages the features of Azure Container Apps (Preview).

We will use a combination of the Azure Portal, Azure CLI, Bicep, and YAML configurations to build out this solution in order to help lock-in the key points. In a true production scenario you would want to have this entirely automated and integrated with CI/CD processes.

Code and other resources for these exercises can be found on GitHub at: [johnnyruz/ContainerAppsDemo (github.com)](https://github.com/johnnyruz/ContainerAppsDemo)

### Pre-requisites

1. Azure Subscription – Free Trial or Pay-as-You Go Subscription
2. Azure CLI Installed ([How to install the Azure CLI | Microsoft Docs](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli))
3. Source Code downloaded from the above GitHub repository

### What we’re building

Diagram

Description automatically generated

### Activity 1: Create an Azure Resource Group

1. Open the Azure Portal (<https://portal.azure.com>)
2. Login with a Microsoft account that has a Free trial, pay-as-you-go subscription, or other Azure credits
3. Graphical user interface, application, Teams

   Description automatically generatedSelect Resource Groups from the list of services. If you do not see this on your dashboard, search for Resource Groups in the search box at the top of the screen.
4. Enter the following information:
   1. Subscription – Select your desired subscription
   2. Resource Group – Give your Resource Group a name (you will need this later)
   3. Graphical user interface, text, application, email

      Description automatically generatedSelect a Region – Can be any Region
5. Click Review + Create, then Create

### Activity 2: Create and Configure Azure Service Bus

1. Select your Resource Group from the list in the Portal
2. Click Create 
3. In the search box, type **Service Bus**, then click create
4. Enter details about your Service Bus configuration
   1. Subscription – same as Activity 1
   2. Resource Group – select the one created in Activity 1
   3. Namespace Name – anything you want but must be unique across Azure
   4. Location – select **Canada Central**
   5. Pricing Tier – select **Standard**
5. Click Review + Create, then Create (Service Bus deployment takes about 60-90 seconds)
6. Once completed, click “Go to resource”
7. In the left-hand menu, select Shared access policies 
8. Select the RootManageSharedAccessKey policy
9. Copy the “Primary Connection String” value and **SAVE THIS FOR LATER!**
10. In the left-hand menu, select Topics 
11. Create a new topic named: **messagepublishtopic** (this name is **CRITICAL** to the rest of the demo)
    1. Select 5GB as the topic size then click Create
12. Create a second topic name: **messageresponsetopic** (this name is **CRITICAL** to the rest of the demo)
    1. Select 5GB as the topic size then click Create

### Activity 3: Deploy Container App Dependencies (Log Analytics, App Insights, Container App Environment)

1. Using PowerShell or Windows Terminal, navigate to the directory where you downloaded the files from GitHub
2. In the Root of that directory, there is a folder called **DeployDependencies**, navigate to this folder using your console
3. Run the command **az login**
   1. This command should launch your browser and you can authenticate with the credentials you used to login to the Azure Portal
   2. Run the command **az account show** to validate you’re on the current subscription. If you need to change your selected subscription, you can run the command **az account set --subscription “[Subscription Name]”**
4. Execute the Deploy Dependencies script by running **./deploy\_dependencies.ps1** and enter the following information when prompted:
   1. Resource Group – enter the name of the resource group created in Activity 1
   2. Resource Prefix – short, all lower-case string that will prefix your resources. (i.e. **containerappdemo**)
5. The script should run to completion and you should see several resources now in the Portal
6. The script will also output two values, your Environment ID and App Insights Key. **SAVE THESE FOR THE NEXT ACTIVITY!!**

### Activity 4: Deploy Container Apps

1. Using a terminal or Windows explorer, navigate to the directory where you saved the source code and course materials.

**Frontend**

1. Navigate to the directory **\ContainerAppsDemo\FrontendWebMVC\Deployment**
2. Open the file **deploy.yaml** in your preferred text editor
3. Input the following values from the 3 values saved earlier
   1. KUBE\_ENVIRONMENT\_ID – The Environment ID from the output of Activity 3
   2. APP\_INSIGHTS\_KEY – The App Insights Key from the output of Activity 3
   3. SERVICE\_BUS\_CONNECTION\_STRING – The connection string saved in step 9 of Activity 2

**Backend**

1. Navigate to the directory **\ContainerAppsDemo\BackendProcessor\Deployment**
2. Open the file **deploy.yaml** in your preferred text editor
3. Input the following values from the 3 values saved earlier
   1. KUBE\_ENVIRONMENT\_ID – The Environment ID from the output of Activity 3
   2. APP\_INSIGHTS\_KEY – The App Insights Key from the output of Activity 3
   3. SERVICE\_BUS\_CONNECTION\_STRING – The connection string saved in step 9 of Activity 2
4. In your PowerShell or Terminal window, ensure you are at the root of the source code files (directory should be named **ContainerAppsDemo**).
5. Execute the following 2 commands:
   1. az containerapp create -g [YOUR RESOURCE GROUP NAME] -n [YOUR RESOURCE PREFIX]-frontend --yaml FrontendWebMVC/Deployment/deploy.yaml
   2. az containerapp create -g [YOUR RESOURCE GROUP NAME] -n [YOUR RESOURCE PREFIX]-backend --yaml BackendProcessor/Deployment/deploy.yaml
6. Your container apps should successfully deploy to your environment

### Activity 5: Browsing your application

1. In the Azure Portal, refresh the list of resources within your Resource Group from Activity 1
2. Graphical user interface, application

   Description automatically generated with medium confidenceYou should see a list of resources that look like this:
3. Select the Container App with “frontend” in the name
4. On the Overview page in the top-right, you should see a property of **Application URL**. Click on that link.
5. The website should open to a page that looks like this:
6. Clicking the buttons should kick off the process of publishing a message to the backend, and then having the backend process it and send a response to the frontend
7. The frontend UI should reflect the number of messages that have been published vs. those that have been processed. **NOTE: There is code in place to explicitly slow down message publishing and processing for demo purposes!**

### Activity 6: Viewing Scaling

1. Let your app sit idle for at least 10 minutes to allow for scale-down of containers
2. In the Portal, under your Resource Group, select the Container app with “backend” in its name
3. Navigate to the link for Revision Management, then click on the Active revision (there should only be 1)
4. Graphical user interface, text, application

   Description automatically generatedObserve the Replica Count metric on the Revision Overview page. If the app has been idle long enough, you should notice that it has scaled to 0 running instances.
5. In your Web Browser, navigate to the URL for the frontend and select “Publish 100 Messages”. **NOTE: It may take some time before messages start being processed as the first instance of the backend service is started. This delay is expected when your app has scaled to 0 running instances.**
6. Graphical user interface, application

   Description automatically generatedRefresh the Revision Overview page and observe the Replica Count to ensure scaling is working based on number of messages in Azure Service Bus.

### Activity 6: Logging and Monitoring

1. In the Azure Portal, under your Resource Group from Activity 1, select the resource with “analytics” in the name.
2. In the left-hand menu, select “Logs” 
3. Close the Query Selection window that pops up
4. In the left side of the Query Builder window, open up the **Custom Logs** section and notice a custom log section has been created for **ContainerAppConsoleLogs\_CL**
5. In the query window enter the following query:

ContainerAppConsoleLogs\_CL |

where TimeGenerated > ago(10m) |

order by TimeGenerated desc

1. You can see the logs from inside your containers as they are published to Log Analytics

Graphical user interface, text, application, email

Description automatically generated

1. In the Azure Portal, under your Resource Group from Activity 1, select the resource with “app-insights” in the name.
2. Advanced usage of App Insights is beyond the scope of this course, but using tools you can observe the message flows, actions, and telemetry captured from the application.

### Activity 7: Destroy All Resources (Optional to save on billing costs)

1. In the portal, select your Resource Group created in step #1
2. Click on the button Delete Resource Group
3. Enter your Resource Group Name to confirm and press **Delete**