# Transact-BOHA Web Application Framework

## Introduction

This document will provide you with the instructions to implement Transact BOHA web application.

The Transact BOHA application consists of the components listed below

* SQL Server Cloud Database
* Azure AD B2C
* ASP.NET Core Web API
* Angular 13 Web Application

In addition to these components, we will build a Kubernetes controller, an image repository, and an operating environment (server farm). Operations management, redundancy, scaling up and down and automatic restart are managed by Kubernetes and native Azure features. The application code for the Web Application and the Web API is located at the git address specified below:

<https://symphonize.git.beanstalkapp.com/transact.git>

## Prerequisites

The required components are a Windows and/or Mac desktop and an Azure account.

The document assumes that your desktop machine is already on the latest version of the following components:

1. .Net Framework 5
2. Docker Desktop for Windows
3. Azure CLI

.NET Framework 5: <https://dotnet.microsoft.com/en-us/download/dotnet/5.0>

Docker Desktop for Windows: <https://docs.docker.com/desktop/windows/install/>

Azure CLI: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>

In addition to the above components, please make sure the Cloud SQL database and the AzureAD B2C are installed and configured in the same resource group (Transact)

The Cloud SQL Connection string and the Azure AD B2C values need to be updated in the appSetings.json file of the Transact.API project. Please check the screenshot below.

Graphical user interface, text, application, Word

Description automatically generated

## Step 1 – One Time Infrastructure Build

In this step you will construct the permanent infrastructure to deploy and operate the application service. There is no additional set up required for Docker.

1. From a Windows or Mac CLI system, log in to the appropriate Azure account

|  |
| --- |
| % AZ login |
| The default web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser fails to open, use device code flow with `az login --use-device-code`.  [  {  "cloudName": "AzureCloud",  "homeTenantId": "35gfsgs…a32",  "id": "frgsdfg333….475d",  "isDefault": true,  "managedByTenants": [],  "name": "Azure subscription 1",  "state": "Enabled",  "tenantId": "34556….ca32",  "user": {  "name": "Joe.Mueller@symphonize.com",  "type": "user"  }  }  ]  % |

1. Define a Resource Group

All the application components will reside in a resource group. Select the location based on the geography of your customers. Multiple server farms can be located throughout the world.

|  |
| --- |
| % az group create --location eastus --name Transact |
| {  "id": "/subscriptions/gt....7h/resourceGroups/Transact",  "location": "eastus",  "managedBy": null,  "name": "Transact",  "properties": {  "provisioningState": "Succeeded"  },  "tags": null,  "type": "Microsoft.Resources/resourceGroups"  } |

1. Create an Azure Container Registry (ACR)

This Azure component will manage your image repositories. The –sku Basic option is for selecting a basic computing platform. Depending on your activity against the repositories, you may want to increase this to Standard or greater and add additional repositories.

|  |
| --- |
| % az acr create -n transactweb -g Transact --admin-enabled false --sku Basic |
| {  "adminUserEnabled": false,  "anonymousPullEnabled": false,  "creationDate": "2022-01-13T21:29:25.580608+00:00",  "dataEndpointEnabled": false,  "dataEndpointHostNames": [],  "encryption": {  "keyVaultProperties": null,  "status": "disabled"  },  "id": "/subscriptions/7c2183b3-c273-4f3e-a0be-26f5c9a2a193/resourceGroups/Transact/providers/Microsoft.ContainerRegistry/registries/bohaACR",  "identity": null,  "location": "eastus",  "loginServer": "bohaacr.azurecr.io",  "name": "bohaACR",  "networkRuleBypassOptions": "AzureServices",  "networkRuleSet": null,  "policies": {  "exportPolicy": {  "status": "enabled"  },  "quarantinePolicy": {  "status": "disabled"  },  "retentionPolicy": {  "days": 7,  "lastUpdatedTime": "2022-01-13T21:29:32.356163+00:00",  "status": "disabled"  },  "trustPolicy": {  "status": "disabled",  "type": "Notary"  }  },  "privateEndpointConnections": [],  "provisioningState": "Succeeded",  "publicNetworkAccess": "Enabled",  "resourceGroup": "Transact",  "sku": {  "name": "Basic",  "tier": "Basic"  },  "status": null,  "systemData": {  "createdAt": "2022-01-13T21:29:25.580608+00:00",  "createdBy": "ashok@symphonize.com",  "createdByType": "User",  "lastModifiedAt": "2022-01-13T21:29:25.580608+00:00",  "lastModifiedBy": "ashok@symphonize.com",  "lastModifiedByType": "User"  },  "tags": {},  "type": "Microsoft.ContainerRegistry/registries",  "zoneRedundancy": "Disabled"  } |

1. Create an Azure Cluster

This component allows Kubernetes to manage the deployments and run time environments. It connects the Kubernetes cluster, defined here, with the Azure Container Registry (ACR).

The “--node-count” should be adjusted based on your recovery and capacity requirements.

|  |
| --- |
| % az aks create -n bohaAKS -g Transact --attach-acr bohaACR --node-count 1 --enable-managed-identity –generate-ssh-keys |
| AAD role propagation done[############################################] 100.0000%{  "aadProfile": null,  "addonProfiles": null,  "agentPoolProfiles": [  {  "availabilityZones": null,  "count": 1,  "enableAutoScaling": false,  "enableEncryptionAtHost": false,  "enableFips": false,  "enableNodePublicIp": false,  "enableUltraSsd": false,  "gpuInstanceProfile": null,  "kubeletConfig": null,  "kubeletDiskType": "OS",  "linuxOsConfig": null,  "maxCount": null,  "maxPods": 110,  "minCount": null,  "mode": "System",  "name": "nodepool1",  "nodeImageVersion": "AKSUbuntu-1804gen2containerd-2022.01.07",  "nodeLabels": null,  "nodePublicIpPrefixId": null,  "nodeTaints": null,  "orchestratorVersion": "1.21.7",  "osDiskSizeGb": 128,  "osDiskType": "Managed",  "osSku": "Ubuntu",  "osType": "Linux",  "podSubnetId": null,  "powerState": {  "code": "Running"  },  "provisioningState": "Succeeded",  "proximityPlacementGroupId": null,  "scaleDownMode": null,  "scaleSetEvictionPolicy": null,  "scaleSetPriority": null,  "spotMaxPrice": null,  "tags": null,  "type": "VirtualMachineScaleSets",  "upgradeSettings": null,  "vmSize": "Standard\_DS2\_v2",  "vnetSubnetId": null  }  ],  "apiServerAccessProfile": null,  "autoScalerProfile": null,  "autoUpgradeProfile": null,  "azurePortalFqdn": "bohaaks-transact-7c2183-e156a577.portal.hcp.eastus.azmk8s.io",  "disableLocalAccounts": false,  "diskEncryptionSetId": null,  "dnsPrefix": "bohaAKS-Transact-7c2183",  "enablePodSecurityPolicy": null,  "enableRbac": true,  "extendedLocation": null,  "fqdn": "bohaaks-transact-7c2183-e156a577.hcp.eastus.azmk8s.io",  "fqdnSubdomain": null,  "httpProxyConfig": null,  "id": "/subscriptions/7c2183b3-c273-4f3e-a0be-26f5c9a2a193/resourcegroups/Transact/providers/Microsoft.ContainerService/managedClusters/bohaAKS",  "identity": {  "principalId": "480a4b6f-5fb3-400e-9ef5-3ed4a5aed411",  "tenantId": "44d8b453-bc45-44ea-ae2b-5c48de48fdf9",  "type": "SystemAssigned",  "userAssignedIdentities": null  },  "identityProfile": {  "kubeletidentity": {  "clientId": "0285fc0d-54a2-4e49-ad4c-240b14f59517",  "objectId": "065ba0ad-3620-4cda-b778-ce0fbeafdb2c",  "resourceId": "/subscriptions/7c2183b3-c273-4f3e-a0be-26f5c9a2a193/resourcegroups/MC\_Transact\_bohaAKS\_eastus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/bohaAKS-agentpool"  }  },  "kubernetesVersion": "1.21.7",  "linuxProfile": {  "adminUsername": "azureuser",  "ssh": {  "publicKeys": [  {  "keyData": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCkvK52QFbhm42toiH7RSyb+2z7EqE+JyxHzGoXRN1g1tgM8KtvfxWHdhV6MW63oO1LMFcP+dNwLU+1xsiu8llXBsaWXC5F/JvUgCFQjmCzIhJ6aHR1ra2erNmbmCbECdn1P2/AF1tdibS3q8fI5PWBrpj+EwrzoywStD0Kkdm8rWtV2acwpuEktDviihYg2pm/6cPQoXm8KviD8OInXK4LNg+vnEnpF9xDQg2b/ArIT6gzTa8CUdVRZ2yLcT2lDg9U/VLjtSmcHACwFHl5uPbnQ6LX9iyhu50QMs5ieBswkEiUYdQ4Ybzv3+xtzxr3Iyuq7loouBQep9pnhO2zjCQ7"  }  ]  }  },  "location": "eastus",  "maxAgentPools": 100,  "name": "bohaAKS",  "networkProfile": {  "dnsServiceIp": "10.0.0.10",  "dockerBridgeCidr": "172.17.0.1/16",  "loadBalancerProfile": {  "allocatedOutboundPorts": null,  "effectiveOutboundIPs": [  {  "id": "/subscriptions/7c2183b3-c273-4f3e-a0be-26f5c9a2a193/resourceGroups/MC\_Transact\_bohaAKS\_eastus/providers/Microsoft.Network/publicIPAddresses/c1b8f4ff-5fed-4c4b-8daa-0922e6d3cb57",  "resourceGroup": "MC\_Transact\_bohaAKS\_eastus"  }  ],  "idleTimeoutInMinutes": null,  "managedOutboundIPs": {  "count": 1  },  "outboundIPs": null,  "outboundIpPrefixes": null  },  "loadBalancerSku": "Standard",  "natGatewayProfile": null,  "networkMode": null,  "networkPlugin": "kubenet",  "networkPolicy": null,  "outboundType": "loadBalancer",  "podCidr": "10.244.0.0/16",  "serviceCidr": "10.0.0.0/16"  },  "nodeResourceGroup": "MC\_Transact\_bohaAKS\_eastus",  "podIdentityProfile": null,  "powerState": {  "code": "Running"  },  "privateFqdn": null,  "privateLinkResources": null,  "provisioningState": "Succeeded",  "resourceGroup": "Transact",  "securityProfile": null,  "servicePrincipalProfile": {  "clientId": "msi",  "secret": null  },  "sku": {  "name": "Basic",  "tier": "Free"  },  "tags": null,  "type": "Microsoft.ContainerService/ManagedClusters",  "windowsProfile": null  } |

1. Get Registry Credentials

This step updates the deployment server’s (your desktop) credentials from Azure. This allows Kubernetes access to oversee the build and deploy environment from your machine.

|  |
| --- |
| % az aks get-credentials -n bohaAKS -g Transact |

At this point your permanent Azure infrastructure is in place. The Kubernetes cluster (AKS) and the Azure Container Registry (ACR) are in place and connected. You can log into Azure and see the results of your work.

## Step 2 – API Deployment

API deployment consists of using an application repository (git) to construct a Docker image (application + operating system), testing the build then moving the image to Azure where rollout and operations will be managed by Kubernetes.

1. Create Application Directory

On you desktop machine, cd to the desired location for the application code.

|  |
| --- |
| % mkdir transact  % cd transact |
| transact % |

1. Download the application project from the git repo

The clone/download is the repo with our deliverables.

|  |
| --- |
| % git clone https://symphonize.git.beanstalkapp.com/transact.git |
| transact % git clone https://symphonize.git.beanstalkapp.com/transact.git  Cloning into 'transact'...  remote: Enumerating objects: 1458, done.  remote: Counting objects: 100% (1458/1458), done.  remote: Compressing objects: 100% (1355/1355), done.  remote: Total 1458 (delta 492), reused 772 (delta 72)  Receiving objects: 100% (1458/1458), 7.96 MiB | 4.31 MiB/s, done.  Resolving deltas: 100% (492/492), done.  transact % |

1. Log in to the appropriate Azure account (output details shown above)

|  |
| --- |
| % az login |

1. Log in to the Azure Container Registry (ACR)

|  |
| --- |
| % az acr login --name bohaACR |
| Login Succeeded |

1. Build the Docker Image

This step takes a file called “Dockerfile” and the source from the current directory and delivers an image (OS+app) to the Desktop Docker.

|  |
| --- |
| % docker build . -t transact.api:v1 |
| [+] Building 0.8s (19/19) FINISHED  => [internal] load build definition from Dockerfile 0.0s  => => transferring dockerfile: 678B 0.0s  => [internal] load .dockerignore 0.0s  => => transferring context: 34B 0.0s  => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:5.0 0.6s  => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:5.0 0.6s  => [internal] load build context 0.1s  => => transferring context: 31.44kB 0.0s  => [build 1/10] FROM mcr.microsoft.com/dotnet/sdk:5.0@sha256:18fac152a33c218c8e405039084eab219b65cfaf4ec2c507dded81127e61f4dc 0.0s  => [runtime 1/3] FROM mcr.microsoft.com/dotnet/aspnet:5.0@sha256:3c975b9137429026029b292b712965f29719d02b9b62796fed4d78b5d18ea048 0.0s  => CACHED [runtime 2/3] WORKDIR /app 0.0s  => CACHED [build 2/10] WORKDIR /app 0.0s  => CACHED [build 3/10] COPY \*.sln . 0.0s  => CACHED [build 4/10] COPY Transact.Data/\*.csproj ./Transact.Data/ 0.0s  => CACHED [build 5/10] COPY Transact.API/\*.csproj ./Transact.API/ 0.0s  => CACHED [build 6/10] RUN dotnet restore /app/Transact.API/Transact.API.csproj 0.0s  => CACHED [build 7/10] COPY Transact.API/. ./Transact.API/ 0.0s  => CACHED [build 8/10] COPY Transact.Data/. ./Transact.Data/ 0.0s  => CACHED [build 9/10] WORKDIR /app/Transact.API 0.0s  => CACHED [build 10/10] RUN dotnet publish -c release -o out 0.0s  => CACHED [runtime 3/3] COPY --from=build /app/Transact.API/out ./ 0.0s  => exporting to image 0.0s  => => exporting layers 0.0s  => => writing image sha256:81d1115d0f4d44bbe1f6d87ffd31bc38ff4c9d09d6cdfaa73210a40770e51156 0.0s  => => naming to docker.io/library/transact.api:v1 0.0s  Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them |

1. Desktop Test Image

At this point you can follow Docker Instructions[[1]](#footnote-2) to start the image, connect to it and run the app. Keep in mind Azure services will not be present on your build machine.

1. Tag Image

Tagging the image makes it accessible by the repository (ACR). There is no response.

|  |
| --- |
| % docker tag transact.api:v1 bohaacr.azurecr.io/transact.api:v1 |
| No output indicates successful tagging |

1. Copy Image to Azure

This step copies the tagged image into a repository connected to the Azure Container Registry (ACR)

|  |
| --- |
| % docker push bohaacr.azurecr.io/transact.api:v1 |
| The push refers to repository [bohaacr.azurecr.io/transact.api]  8d736aba052a: Pushed  b349c4dfe564: Pushed  653b4358e58a: Pushed  2c360e0ff640: Pushed  5171dec7715f: Pushed  493105d869be: Pushed  ad6b69b54919: Pushed  v1: digest: sha256:53937fce63d9ed58c6dbca40ca115bbbed08c05f301ff1ed19c2d18bf1ec4308 size: 1790 |

1. Locate Kubernetes context

Kubernetes stores environment information, including which cluster to use, in a file called “context”. This command will list the contexts. Adjusting which context Kubernetes controls which cluster you will use in delivering the application to Azure.

|  |
| --- |
| % kubectl config get-contexts |
| CURRENT NAME CLUSTER AUTHINFO NAMESPACE  \* bohaAKS bohaAKS clusterUser\_Transact\_bohaAKS |

1. Adjust Kubernetes context

This command instructs Kubernetes to use the cluster called “transact-aks”.

|  |
| --- |
| % kubectl config use-context bohaAKS |
| Switched to context "bohaAKS". |

1. Define and Activate the Application

In this step we provide kubectl with a yaml file[[2]](#footnote-3) that defines the deployment including service connections, infrastructure requirements and other operating parameters.

|  |
| --- |
| % kubectl apply -f pod.yaml  % kubectl apply -f service.yaml |
| pod/transact.api created |
|  |

1. Identify External IP Address[[3]](#footnote-4)

We can requesting kubectl to provide us with the IP address that has been associated with our service. In the output below, the IP address is xx.xx.xxx.xxx

|  |
| --- |
| % kubectl get svc -w |
| NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.0.0.1 <none> 443/TCP 32m  transact-api LoadBalancer 10.0.248.216 52.226.201.84 80:32210/TCP 10m |

1. Access

The application is accessible at the IP address which could be seen in the output above. A successful load of the Swagger API page suggests that the deployment is successful

Graphical user interface

Description automatically generated with medium confidence

1. Conclusion

This application is secure, fault tolerant and performant. It can be updated from repo to live in minutes. It can be sensitive to business cycles and add compute resources to meet increased demand or remove resources when demand subsides. Cycling servers is automatic and unobtrusive. This is state of the art computing technology that is a powerful tool to support your business goals.

1. <https://docs.docker.com/desktop/windows/>

   https://hub.docker.com/editions/community/docker-ce-desktop-mac [↑](#footnote-ref-2)
2. Commands read in yaml files that contain situation or environment specific parameters. Be sure the view the contents of the file before submitting it to be processed.  [↑](#footnote-ref-3)
3. Actual IP addresses redacted [↑](#footnote-ref-4)