Deploying Containerized Application to Azure Service Fabric.

1.Overview

Azure Service Fabric is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices and containers. Service Fabric also addresses the significant challenges in developing and managing cloud-native applications. Developers and administrators can avoid complex infrastructure problems and focus on implementing mission-critical, demanding workloads that are scalable, reliable, and manageable. Service Fabric represents the next-generation platform for building and managing these enterprise-class, tier-1, cloud-scale applications running in containers.

2.Goals for this walkthrough

1. Containerize your application by manually adding the dockerfiles and using the Docker CLI.
2. Creating and accessing Service fabric cluster.
3. Deploying containerized applications to service fabric cluster having Linux nodes.
4. Deploying containerized applications to service fabric cluster having windows nodes.
5. Automating deployment using Jenkins application.

3.Containerizing existing .net application for deployment

Windows Containers should be used as a way to improve deployments to production, development and test environments of existing .NET applications based on .NET Framework technologies like MVC, Web Forms or WCF.

3.1.Choosing a Right image for your application:

To containerize an application, we need a docker image. There are many sources online from where we can pull the docker images. We have the official repository of docker where we can find images of Ubuntu, nginix, tomcat, mango, jetty, php, Jenkins and many more.

But we don't get any windows images in that repository. Microsoft repository is the best and preferred source of Docker images for windows application. Also, there are many private images available online for .net applications.

1. Official Docker registry: https://hub.docker.com/explore/
2. Microsoft docker registry: https://hub.docker.com/u/microsoft/
3. Docker registry by Capgemini: https://hub.docker.com/r/capgemini/

For deploying asp.net MVC application, I used Microsoft's docker image that has iis server installed in it, to deploy ASP.NET applications.

microsoft/iis:10.0.14393.206

3.2.Creating a Docker file to deploy the application in a Docker container:

Docker can build images automatically by reading the instructions from `Dockerfile.` A `Dockerfile` is a text document that contains all the commands a user could call on the command line to assemble an image. Using `'docker build'` users can create an automated build that executes several command-line instructions in succession. Here is the docker file that uses a docker image that has iis server installed in it, that is provided by Microsoft. The comment describes the meaning of that command.

```
FROM microsoft/iis:10.0.14393.206        ----- Use this image to generate the container

SHELL ["powershell"]                     ----- Use Powershell


RUN Install-WindowsFeature NET-Framework-45-ASPNET ;\
Install-WindowsFeature Web-Asp-Net45 ----- Run this command to install .net framework.


COPY HelloWorld HelloWorld               ----- Copy HelloWorld directory to HelloWorld directory in the container.

RUN Remove-WebSite -Name 'Default Web Site'   ----- Remove existing website from IIS server.


RUN New-Website -Name 'helloworld' -Port 80 \ ----- New website to required and straightforward on -Port 80, giving the physical path.
-PhysicalPath 'c:\HelloWorld' -ApplicationPool '.NET v4.5'

EXPOSE 80 ----- Expose port 80 to the internet.


CMD Write-Host IIS Started... ; \

while ($true) { Start-Sleep -Seconds 3600 }
```

3.3.Essential commands to manage the container:

Docker commands and best practices cheat sheets are available online. You can find them by navigating to the websites using the URLs given below.

1. https://afourtech.com/guide-docker-commands-examples/
2. https://zeroturnaround.com/rebellabs/docker-commands-and-best-practices-cheat-sheet/

3.4.Creating an image using a working container:

Build a docker image from docker file using 'docker build' command. For example:

docker build -f iis.dockerfile --tag aanusha/helloworldpublisher.v1.0 .

--tag is the name given to the image.

aanusha - is the username of the repository the image is pushed to

helloworldpublisher - is the name of the image

Note: It is always the best practice to use Azure's Container Registry to work with Azure. We are using docker registry right now, and we are planning to use Azure's container registry in future.

v1.0 - version of the image, which is referred to tag in the repository. This can be anything like Tolatest, :your_name, etc.

3.5.Deploy container to the local machine and test:

Use 'docker run' command to create a container out of the image.

docker run --name deployhelloworld -d -p 80:80 aanusha/helloworldpublisher:v1.0.

Use 'docker inspect' command to get the information of container in JSON format.

docker inspect --format="{{.NetworkSettings.Networks.nat.IPAddress}}" deployhelloworld

3.6.Pushing the Container to Docker hub:

Use 'docker push' command to publish the image to docker hub.

docker push aanusha/helloworldpublisher:v1.0

4.Creating service fabric clusters

4.1.Creating a Linux cluster:

4.1.1. In general:

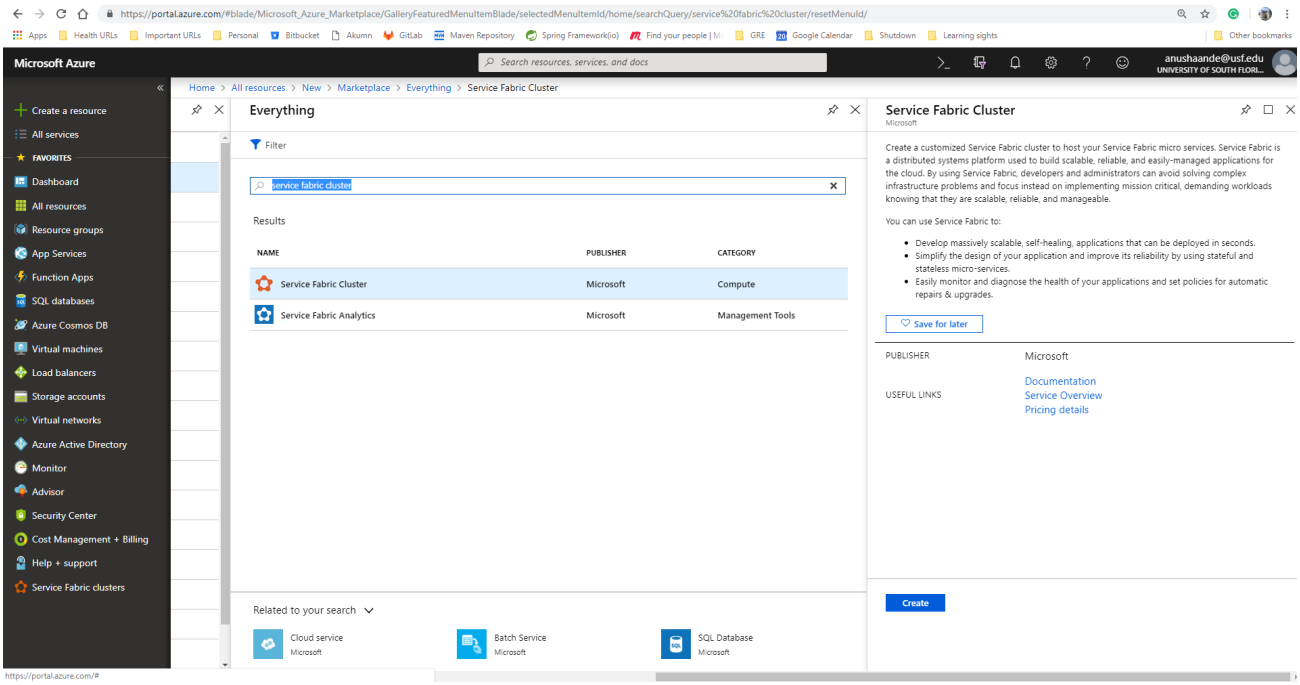
Here is the Microsoft document which gives a clear understanding of creating a service fabric cluster in Azure using the Azure portal.

https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-cluster-creation-via-portal

4.1.2. Specific to USF Health:

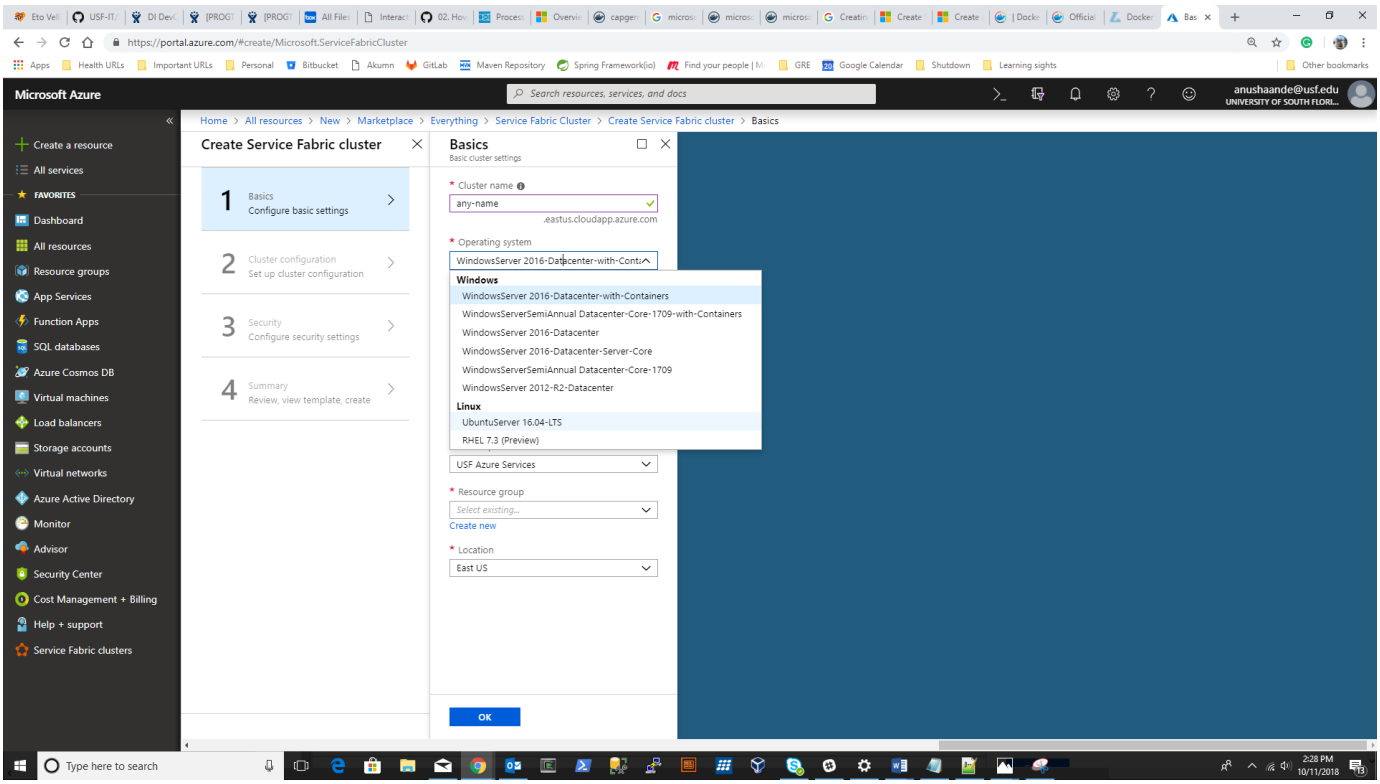
The procedure of creating a service fabric cluster for USF Health is the same as the procedure that is mentioned in the document above. But there are a few options we choose differently, which are mentioned below.

- ***Choose a service fabric cluster resource:***

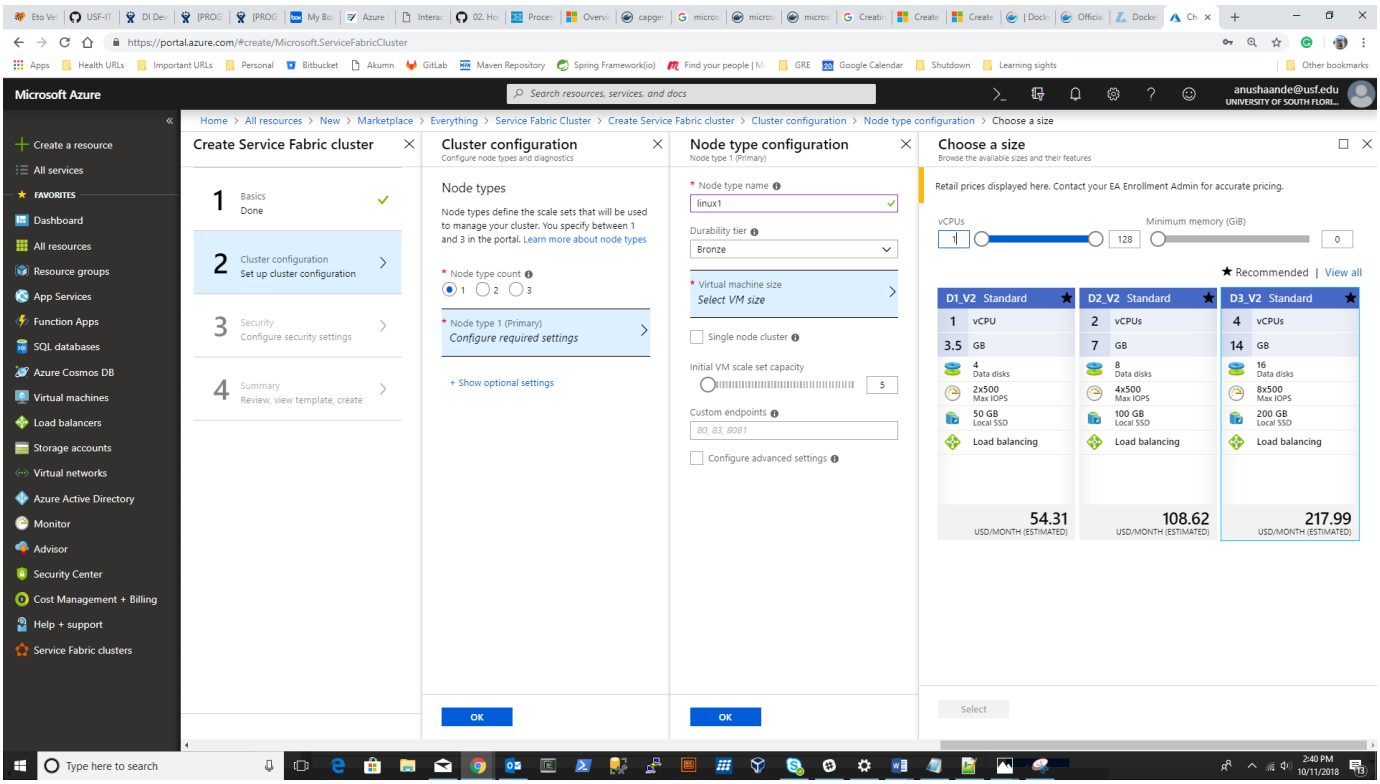


- ***Basics:***

1. Enter the name of your cluster.
2. Enter a Username and Password for Remote Desktop for the VMs.
3. Make sure to select the Subscription that you want your cluster to be deployed to, especially if you have multiple subscriptions.
4. Create a new Resource group, if you have access to. Else select one from existing resource groups.
5. Select the Location in which you want to create the cluster. It's eastus by default.



- **Cluster configuration:**
  - In the Cluster configuration section, the Virtual Machine size is to be **D3_V2 Standard**
  - Leave the custom endpoints blank as we define load balancer rules to open engaging.



- **Security:**

Configure security by following the Microsoft Document.

Download the certificate and click on create the cluster. Once the cluster is established, the next step is to configure the load balancer to open ports, to access the application from the internet.

4.2.Creating windows cluster:

Creating windows cluster is the same as creating a Linux cluster. But the only thing different is choosing a right Operating system, that is found under windows option while setting up the Basics.

4.3.USF Azure Clusters:

In USF Azure, there are currently two types of service fabric clusters. One is a Linux-based cluster, and the other is a Windows-based cluster.

1. Linux-cluster : https://portal.azure.com/#@usfedu.onmicrosoft.com/resource/subscriptions/7fd86781-de49-416d-8708-9ab35d104e5c/resourceGroups/Waleed_LabRG716033/providers/Microsoft.ServiceFabric/clusters/usfdilinux/overview

   This cluster has one application deployed in it, which is Jenkins.

2. Windows-cluster : https://portal.azure.com/#@usfedu.onmicrosoft.com/resource/subscriptions/7fd86781-de49-416d-8708-9ab35d104e5c/resourceGroups/ARG-JenkinsDev/providers/Microsoft.ServiceFabric/clusters/usfsfw/overview

   This cluster is used to deploy windows based containerized applications.

5.Load Balancer Configuration:

There are many documents available online provided by Nginx, Citrix and many other companies that offer load balancers. Below are links for the sample documents picked from those, as I found them to be interesting with simple and required information.

Doc1: This document is provided by Digital Ocean which explains what a load balancer is and its functionality. I felt this document to be easy to understand for someone getting introduced to the load balancers.

https://www.digitalocean.com/community/tutorials/what-is-load-balancing

Doc2: Microsoft provides this document, and it is specific to the Azure load balancer. It describes all the options available in the Azure portal to set up a load balancer.

https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview#publicloadbalancer

Doc3: This document is by a private freelancer developer explaining how to configure a service fabric cluster endpoints and azure load balancer.

https://dajbych.net/service-fabric-endpoints-azure-load-balancer

5.1 Load balancers configured in USF Azure for service fabric:

5.1.1.Load Balancer for the Linux-based cluster :

https://portal.azure.com/#@usfedu.onmicrosoft.com/resource/subscriptions/7fd86781-de49-416d-8708-9ab35d104e5c/resourceGroups/Waleed_LabRG716033/providers/Microsoft.Network/loadBalancers/LB-usfdilinux-usfdilin/overview

5.1.2.Load balancer for the Windows-based cluster:

https://portal.azure.com/#@usfedu.onmicrosoft.com/resource/subscriptions/7fd86781-de49-416d-8708-9ab35d104e5c/resourceGroups/ARG-JenkinsDev/providers/Microsoft.Network/loadBalancers/LB-usfsfw-windows/overview

Linux as well as Windows clusters have load balancing rules set for port 8081 as this is the port open for Jenkins application deployed to Linux cluster and the containerized application that is configured to be deployed to windows cluster.

6. Getting connected to the service fabric cluster:

To get connected to the service fabric cluster, install sfctl. sfctl is command line tool provided to automate the actions required to maintain service fabric cluster should be installed in the system. Follow Instal sfctl

To get connected to service fabric, we need certificates and URLs that can be found in Azure Service Fabric.

1. Download the certificate to your local machine.
2. Copy the entire path where this certificate is downloaded to.
3. Follow Azure Service Fabric document for the remaining steps.

7.Deploying Jenkins Application to Linux cluster:

This document in the link below gives a good understanding of how to set up CI/CD using service fabric cluster. This article explains how to deploy containerized applications using Jenkins installed inside the service fabric cluster or outside the cluster or by just using a plugin.

https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-cicd-your-linux-applications-with-jenkins

Make sure to create a new Storage account everytime you create Jenkins server. Change configurations in setupentrypoint.sh to use the new volume created.

7.1 Jenkins for USF Azure:

https://hub.docker.com/r/aanusha/sfjenkinswithsfctl/

Use this image to install Jenkins. This image has sfctl and supported software to communicate to windows cluster and deploy applications to windows service fabric cluster.

7.2. Access to Jenkins server of USF Azure:

1. Get connected to the Linux service fabric cluster.
2. As the application is deployed to 8081 port, Jenkins can be accessed using http://DNSname:8081
   https://usfdilinux.eastus.cloudapp.azure.com:8081

8. Deploying Containerized application to service fabric cluster:

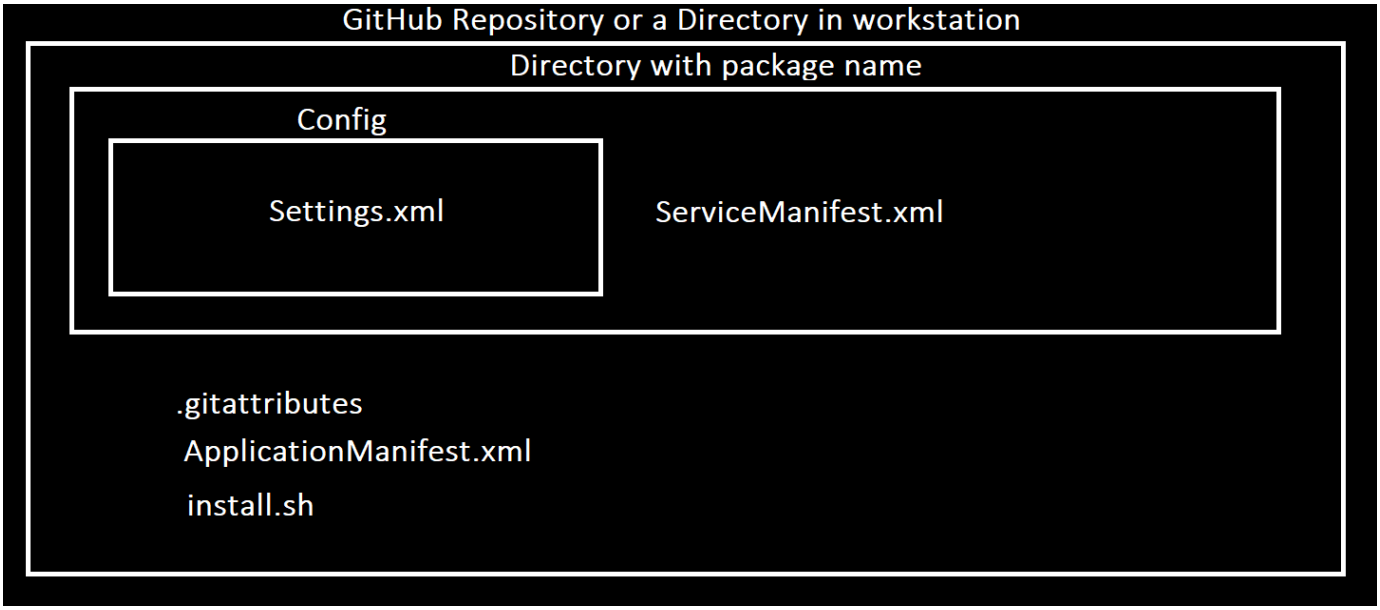
This article provides an overview of the Azure Service Fabric application model and how to define an application and service via manifest files.

This article describes how Service Fabric applications and services are defined and versioned using the ApplicationManifest.xml and ServiceManifest.xml files.

8.1. Directory Structure:

To successfully deploy your containerized application to service fabric cluster, there are few configurations to be taken care of. All these files should be placed in directories in such a way that service fabric cluster knows where to find the configuration file.

https://github.com/anushaande/HelloWorldPublisher_anu



Settings.xml: There are no significant changes to be done in Settings.xml file.

Config: Settings.xml is in the config folder.

ServiceManifest.xml: ServiceManifest configuration document gives a good understanding of configuring HTTP endpoints and overriding endpoints in the servicemanifest.xml file.

HelloPublisherPkg: Name of the package.

ApplicationManifest.xml: All the parameters specific to an application are given here. These parameters override the parameters of the servicemanifest.xml file.

install.sh: This script has all the commands required to deploy the application to service fabric cluster.

HelloWorldPublisher_anu: This is the name of the directory or GitHub repository.

8.2: Configuring manifest files:

Below is the table with attributes which should be configured and that should be same in ApplicationManifest.xml and ServiceManifest.xml documents.

Attribute	Example Value	Explanation	ApplicationManifest.xml	ServiceManifest.xml
ServiceManifest Name.	HelloPublisherPkg	Name of servicemanifest	'***ServiceManifestName***' attribute of '***ServiceManifestRef***' *element under* '***ServiceManifestImport***' *element.*	'**Name**' attribute of '**ServiceManifest**' element.
ServiceManifest Version	1.0.0	Version of service manifest	'**ServiceManifestVersion**' attribute of '***ServiceManifestRef***' *element under* '***ServiceManifestImport***' *element.*	'**Version**' attribute of '**ServiceManifest**' element.
ServiceTypeName	HelloPublisherType	Servicetype Document on servicetypes	'**ServiceTypeName**' attribute of '**StatelessService**' *element under* '***Service***' *element.*	'**ServiceTypeName**' attribute of '**StatelessServiceType**' element under '**ServiceTypes**'
UseImplicitHost	true	The UseImplicitHost attribute indicates this is a guest service	N/A	'**UseImplicitHost**' attribute of '**StatelessServiceType**' element under '**ServiceTypes**'
CodePackage Name	Code		'**CodePackageRef**' attribute of '**ContainerHostPolicies**' element under '**Policies**' under '**ServiceManifestImport**'	'**Name**' attribute of '**CodePackage**' element.
CodePackage Version	1.0.0			'**Version**' attribute of '**CodePackage**' element.
ImageName	aanusha/helloworldpublisher:v1.0	Name of docker image that is to be deployed.	N/A	'**ImageName**' element of '**ContainerHost**' under '**EntryPoint**' under '**CodePackage**' element
EnvironmentVariable Name	HttpGatewayPort	Name of environmental variable.	N/A	'**Name**' attribute of '**EnvironmentVariable**' element under '**EnvironmentVariables**'
EnvironmentVariable Value	19080	Value of the variable.	N/A	'**Value**' attribute of '**EnvironmentVariable**' element under '**EnvironmentVariables**'
ConfigPackage Name	Config	Config package is the contents of the Config directory under PackageRoot that contains an independently-updateable and versioned set of custom configuration settings for your service		'**Name**' attribute of '**ConfigPackage**' element
ConfigPackage Version	1.0.0			

Endpoint Name	HelloPublisherTypeEndpoint	This endpoint is used by the communication listener to obtain the port on which to listen.	'**EndpointRef**' attribute of '**PortBinding**' element under '**ContainerHostPolicies**' under '**Policies**' of '**ServiceManifestImport**' element.	'**Name**' attribute of '**Endpoint**' element under '**Endpoints**' under '**Resources**'.
Endpoint UriScheme	http			'**UriScheme**' attribute of '**Endpoint**' element under '**Endpoints**' under '**Resources**'.
Endpoint Port	8081	This is the port from which application is accessible via the internet. Hence, this port is to be configured in load balancer as given in configuring load balancer section. As the application is redirected to port 80 in docker container, by configuring port binding in ApplicationManifest.xml file, the app is redirected to port 8081 of the node, in which container is deployed.	'**ContainerPort**' attribute of '**PortBinding**' element under '**ContainerHostPolicies**' under '**Policies**' of '**ServiceManifestImport**' element, should be given the value of port to which application is deployed in congtainer.	'**Port**' attribute of '**Endpoint**' element under '**Endpoints**' under '**Resources**'.
Endpoint Protocol	http			'**Protocol**' attribute of '**Endpoint**' element under '**Endpoints**' under '**Resources**'.
CertificateRef Name	wincert	Name of the certificate required to access the service fabric cluster.	'**Name**' attribute fo '**CertificateRef**' element under '**Policies**' under '**ServiceManifestImport**'	
CertificateRef X509FindValue	F4XXXXXXXXXXXXXXXXXXCC7	Certificate footprint	'**X509FindValue**' attribute fo '**CertificateRef**' element under '**Policies**' under '**ServiceManifestImport**'	

8.3: install.sh script:

sfctl application upload --path **HelloWorldPublisher_anu** --show-progress

Path attribute's value should have either GitHub repository value or the root directory in which the application is wrapped. Hence it's always a best practice to give full path of the directory, else execute the script just being outside the directory.

For example:

The application directory is C:/Workstation/githubrepo/HelloWorldPublisher_anu.

Then cd to C:/Workstation/githubrepo and execute ./HelloWorldPublisher_anu/install.sh

sfctl application provision --application-type-build-path **HelloWorldPublisher_anu**

--application-type-build-path is same as --path given above.

sfctl application create --app-name fabric:/HelloWorldPublisher_anu --app-type HelloWorldPublisherType --app-version 1.0.0

--app-type is applicationTypeName mentioned in ApplicationManifest.

--app-version is ApplicationTypeVersion mentioned in ApplicationManifest

--app-name is fabric:/(--path)

```
<ApplicationManifest ApplicationTypeName="HelloWorldPublisherType"
                     ApplicationTypeVersion="1.0.0"
                     xmlns="http://schemas.microsoft.com/2011/01/fabric"
                     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Run the script and check if the application is deployed in 8081 port of your service fabric cluster. The application should be deployed successfully. Else go through the configurations and check if all the configurations are done properly.

9. Automating deployment to Service Fabric using Jenkins:

9.1. Prerequisites to use Jenkins:

1. sfctl should be installed in the server that has Jenkins installed in it.
2. Alternatively, you can use aanusha/sfjenkinswithsfctl image and deploy to service Linux cluster as this has sfctl installed in it.
3. Service fabric cluster certificate should be uploaded to jenkins_home.
4. For Jenkins deployed to service fabric cluster, this certificate can be uploaded to the volume mounted to the cluster.

9.2. Jenkins Job configuration:

Jenkins job configuration is as simple as giving install.sh script to be run to deploy the application. Before that, we need to get connected to the cluster and change directory to the directory in which the app is downloaded.

Give the below commands in build step ---> execute the shell script.

sfctl cluster select --endpoint https://usfdilinux.eastus.cloudapp.azure.com:19080 --pem /var/jenkins_home/usfdisflinux-usfdisflinux-20181001.pem --no-verify

cd /var/jenkins_home/workdir/

./HelloWorldPublisher_anu/install.sh