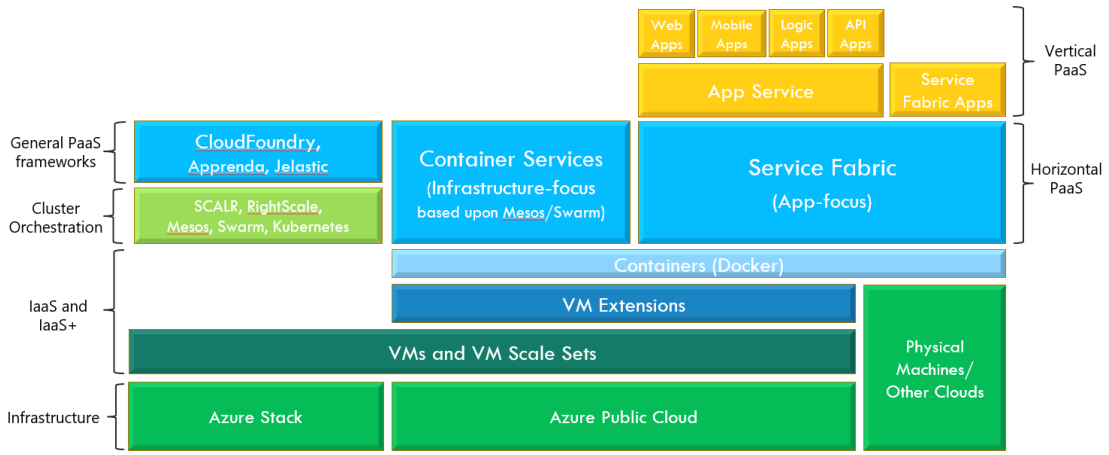
Migrate a Docker Container application to Azure Service Fabric

This article describes migrating a Docker Container application from on premise Docker environment to Azure Service Fabric, including building up cloud repository for Docker images, packaging a container app with specified image and running the app in service fabric cluster.

As a sample, we will start with a Docker image filled with Nginx and migrate it to Azure container registry, then create a container application in service fabric cluster with the uploaded Docker image. Also in this practice, special work in service fabric on national cloud like China Azure will be addressed, including service fabric cluster template and container app manifest settings.

Why Service Fabric?

Regarding how to choose the right platform on azure to host Docker container app, there are several talks and recommendations, like Choosing between Azure Container Service, Azure Service Fabric and Azure Functions and Build Microservice Applications with Microsoft Azure, a widely acknowledged conclusion on Service Fabric VS. Azure Container Service (ACS) is illustrated as below:



Azure Service Fabric is more of App focus while Azure Container Service is more of infrastructure focus.

- Azure Container Service allows to use the open source, industry famous container orchestrators like Docker Swarm and Kubernetes, that gives more openness and flexibility.
- Azure Service Fabric provides its own orchestration, that means Service Fabric provides more integrated, easier to use feature rich model.
- Azure Service Fabric offers several specific programming models, like reliable actor for IoT, reliable service for stateful app, and guest executables/containers for existing apps.
- Service Fabric applications can run on premise, on Azure or even in other cloud platforms.

In brief, Service Fabric is a portable PaaS platform for running micro services including Docker container app, it provides managed orchestration mechanism to guarantee service availability, scalability and reliability, which release developers to focus only on business logistics in application modules.

Create Service Fabric Cluster for Docker Container

Service fabric cluster can be created quickly via serval ways, such as Azure Portal, Visual Studio and ARM template, no matter which way you choose, you will finally get an ARM template created for Azure platform to provision your customized cluster. Generally, for continuous delivery and continuous integration (CI/CD), ARM template is a perfect way to make the customized cluster replicable anytime without duplicate efforts, and the guide of create service fabric cluster and popular service fabric ARM templates are good reference for designing and deploying your Service Fabric cluster.

When following the guide or template to design your service fabric cluster, please be noted that endpoints in Azure clouds are different, see the endpoint mapping.

One more thing for attention is AAD authentication in service fabric cluster, if you choose AAD as secure entry for your service fabric cluster but with older version of service fabric package in China Azure, then in your ARM template file, an extra element "`fabricSettings`" should be added in resource of service fabric, see below, otherwise following login to service fabric explorer will be redirected to global AAD.

```
····{
·····"apiVersion":·"2016-03-01",
·····"type":·"Microsoft.ServiceFabric/clusters",
·····"name":·"[parameters('clusterName')]",
·····"location":·"[parameters('clusterLocation')]",
·····"dependsOn":·[...],
·····"properties":·{
·····"certificate":·{...},
·····"azureActiveDirectory":·{...},
·····"clientCertificateCommonNames":·[],
·····"clientCertificateThumbprints":·[],
·····"clusterState":·"Default",
·····"diagnosticsStorageAccountConfig":·{...},
·····"fabricSettings":·[
·····{
·····"parameters":·[
·····{
·····"name":·"ClusterProtectionLevel",
·····"value":·"[parameters('clusterProtectionLevel')]"
·····},
·····{
·····"name":·"AADLoginEndpoint",
·····"value":·"https://login.partner.microsoftonline.cn"
·····},
·····{
·····"name":·"AADTokenEndpointFormat",
·····"value":·"https://login.partner.microsoftonline.cn/{0}"
·····},
·····{
·····"name":·"AADCertEndpointFormat",
·····"value":·"https://login.partner.microsoftonline.cn/{0}/federationmetadata/2007-06/federationmetadata.xml"
·····}
·····],
·····"name":·"Security"
·····}
·····],
·····"managementEndpoint":·"[concat('https://',reference(concat(variables('lbIPName'),'-','0')).dnsSettings.fqdn,':',variables('nt0fabricHttpGatewayPort'))]",
·····"nodeTypes":·[...],
·····"provisioningState":·"Default",
·····"reliabilityLevel":·"Silver"
·····},
·····"tags":·{...}
····}
```

A complete ARM template for China Azure is shared here, it aims to create a service fabric cluster with 2 node types and 10 nodes in total as well as both cert and AAD authentication.

Publish Container Image to Repository

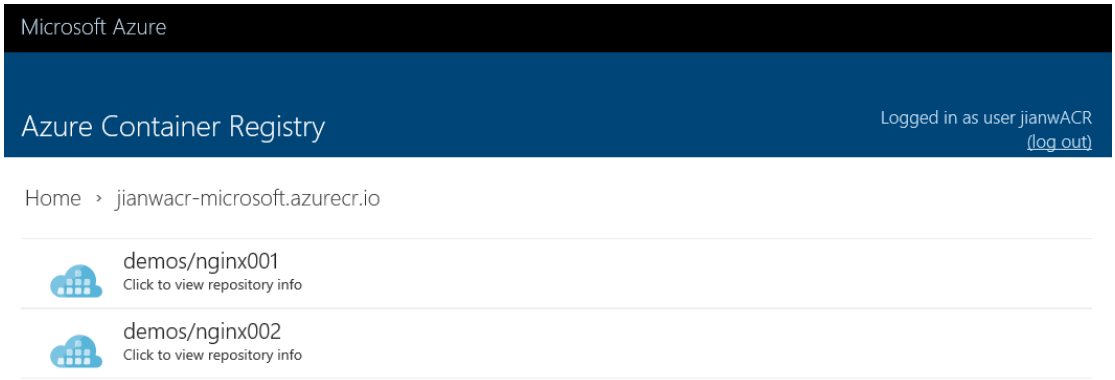
You can either setup your private container repository in cloud for service fabric use or leverage Azure Container Registry as cloud repository, that will be for your use only and not accessible to others per your settings, this guide has detailed info about pulling and pushing images with Azure Container Registry, a quick practice for that is resulted as below:

```
docker -H localhost:2375 images
#download nginx image
sudo docker pull nginx

sudo docker login jianwacr-microsoft.azurecr.io -u jianwACR -p R+d/p/jm/G//[redacted]

docker -H localhost:2375 images
sudo docker tag nginx jianwacr-microsoft.azurecr.io/demos/nginx001
sudo docker push jianwacr-microsoft.azurecr.io/demos/nginx001
```

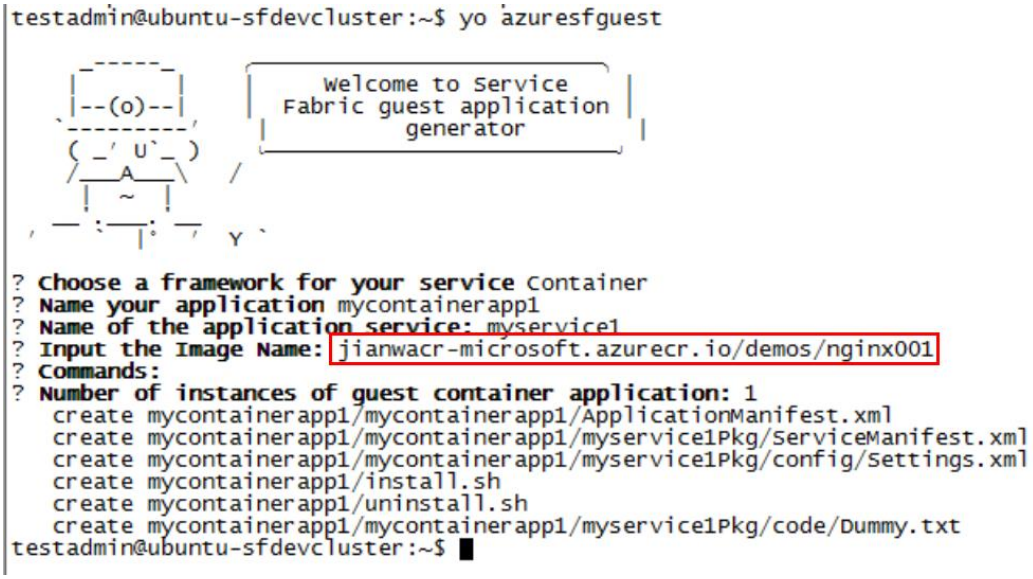
After uploading, you will check out your images in Azure Container Registry by its portal, such as screenshot below, the name *demos/nginx001* will be used to create guest container app in service fabric.



Alternatively, you can also use public Docker Hub as your repository, once your image is ready there like https://hub.docker.com/r/travisyeh/backend/, you will use the image name docker0707/dockersf for creating following micro service.

Build Container application on Service Fabric

This guide is a good tutorial for deploying Docker Container image into service fabric container app, compared with import container image from Docker Hub in the guide, you can also use container image prepared beforehand in your Azure Container Registry, for example:



With the yeoman command, a service fabric container app can be quickly created, right with that, some extra configuration should be added in to service manifest and application manifest, which let the service fabric cluster succeed downloading image from azure container registry and initializing container app then.

See configuration required for the micro service:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceManifest Name="myservice1Pkg" Version="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" >

   <ServiceTypes>
      <StatelessServiceType ServiceTypeName="myservice1Type" UseImplicitHost="true">
   </StatelessServiceType>
   </ServiceTypes>

   <CodePackage Name="code" Version="1.0.0">
      <EntryPoint>
         <ContainerHost>
            <ImageName>jianwacr-microsoft.azurecr.io/demos/nginx001</ImageName>
            <Commands></Commands>
         </ContainerHost>
      </EntryPoint>
      <EnvironmentVariables>
      </EnvironmentVariables>
   </CodePackage>
        <Resources>
```

```
            <Endpoints>
                <Endpoint Name="Endpoint1" UriScheme="http" Port="80" Protocol="http"/>
            </Endpoints>
        </Resources>
 </ServiceManifest>
```

And below are changes needed in application manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<ApplicationManifest  ApplicationTypeName="mycontainerapp1Type" 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">

   <Parameters>
    <Parameter Name="myservice1_InstanceCount" DefaultValue="1" />
   </Parameters>

   <ServiceManifestImport>
      <ServiceManifestRef ServiceManifestName="myservice1Pkg" ServiceManifestVersion="1.0.0"
/>

            <Policies>
                <ContainerHostPolicies CodePackageRef="code">
                    <RepositoryCredentials AccountName="jianwACR"
Password="R+d/p/jm/G//u*******ll0Mq"  PasswordEncrypted="false"/>
                    <PortBinding ContainerPort="80" EndpointRef="Endpoint1"/>
                </ContainerHostPolicies>
            </Policies>

   </ServiceManifestImport>

   <DefaultServices>
      <Service Name="myservice1">
         <StatelessService ServiceTypeName="myservice1Type"
InstanceCount="[myservice1_InstanceCount]">
            <SingletonPartition />
         </StatelessService>
      </Service>
   </DefaultServices>

</ApplicationManifest>
```

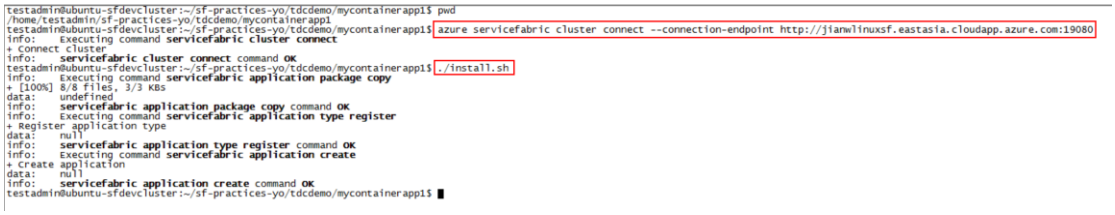
The high lightened part in above manifest files work together to enable the created container

app to:

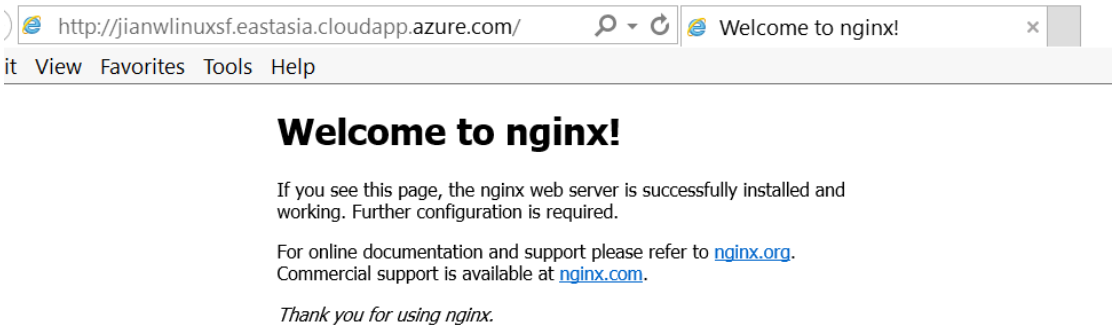
- Authenticate with source Azure Container Registry for downloading target Docker image.
- Expose service in container instance by mapping container port and service port.

Publish Container App in Service Fabric

When above 3 steps of creating service cluster, preparing Docker Image in Azure Container Registry and Building customized service fabric container app are ready, you are almost done to have the app running on the cluster, the final step for you is to publish the app by the provided bash script “install.sh”, see quick deploying below:



The container app running on service fabric can be quickly verified by the configured endpoint and service fabric address.



After the first deployment of container app, later changes or upgrade can be performed as the same as normal service fabric upgrade.

Reference:
https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-linux-overview
https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-azure-cli
https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-deploy-container-linux
https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-docker-cli