Azure Service Fabric



Vision Scope

Prepared for

Microsoft

25-Jan-16

Version 1.0 Draft

Prepared by

**Kristian Nese**

CTO

kristian.nese@lumagate.com

Contributors

**Add Contributors to Doc Properties**

Table of Contents

[1 Before you begin 3](#_Toc441502473)

[2 Lab 1: Deploying Service Fabric 3](#_Toc441502474)

[3 Lab 2: Stateless Microservice 6](#_Toc441502475)

[4 Lab 3: Stateful Microservice 10](#_Toc441502476)

[5 Lab 4: No Downtime Upgrade 15](#_Toc441502477)

1. Before you begin

Preview Documentation: Note that Azure Service Fabric is currently in a public preview, and some of content is subject to change

The objective of this Lab is to learn how to implement Service Fabric and explore some of its capabilities from an operation standpoint, as well as a developer using the Programming Model for stateful and stateless actors and services

* Admin access to an Azure Subscription
* Visual Studio 2015 with the Azure and Service Fabric SDK installed
* Microsoft Azure PowerShell (1.0.2 or higher)

1. Lab 1: Deploying Service Fabric

In this lab you will create and deploy a Service Fabric Cluster in Azure, and then (optionally) locally on your computer.

**Part 1 – Deploy Azure Service Fabric Cluster**

For Part 1, you can choose whether you want to deploy the Service Fabric Cluster to Azure using a prepared ARM template on Github in an automated way, or go through the configuration step-by-step in the Portal. Regardless of what option you choose, you will have to explore the configuration steps in the portal during this exercise.

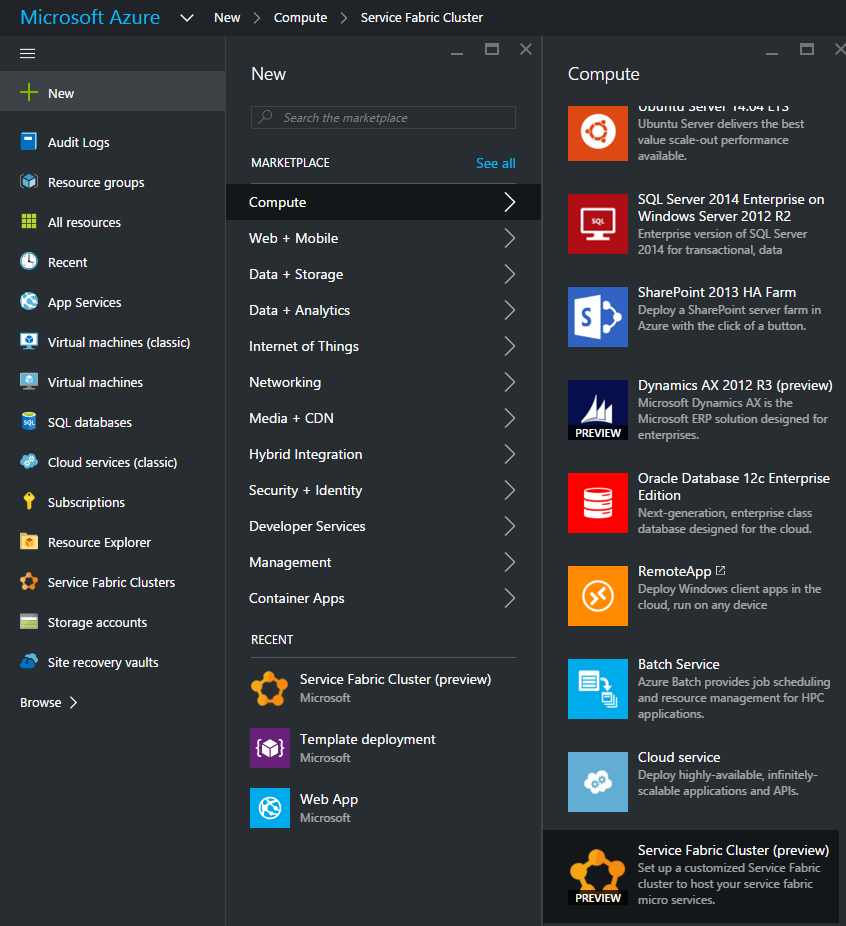
1. From your computer, open PowerShell ISE and login to your Azure subscription with the ‘Login-AzureRM’ cmdlet, and select the subscription using ‘Select-AzureRMSubscription –SubscriptionName’ cmdlet.
2. Run the following cmdlet to provision a new Service Fabric Cluster in a new Azure Resource Group

New-AzureRmResourceGroupDeployment -Name preview -ResourceGroupName (New-AzureRmResourceGroup -Name SF10US -Location "west us").ResourceGroupName -TemplateFile 'https://raw.githubusercontent.com/Lumagate/Azure-PaaS/master/05%20-%20Azure%20Service%20Fabric%20and%20Containers/ARM%20Templates/azuredeploy.json' -clusterLocation 'west us' -clusterName knsfcl01 -adminUserName knadmin -storageAccountType Standard\_LRS -Verbose

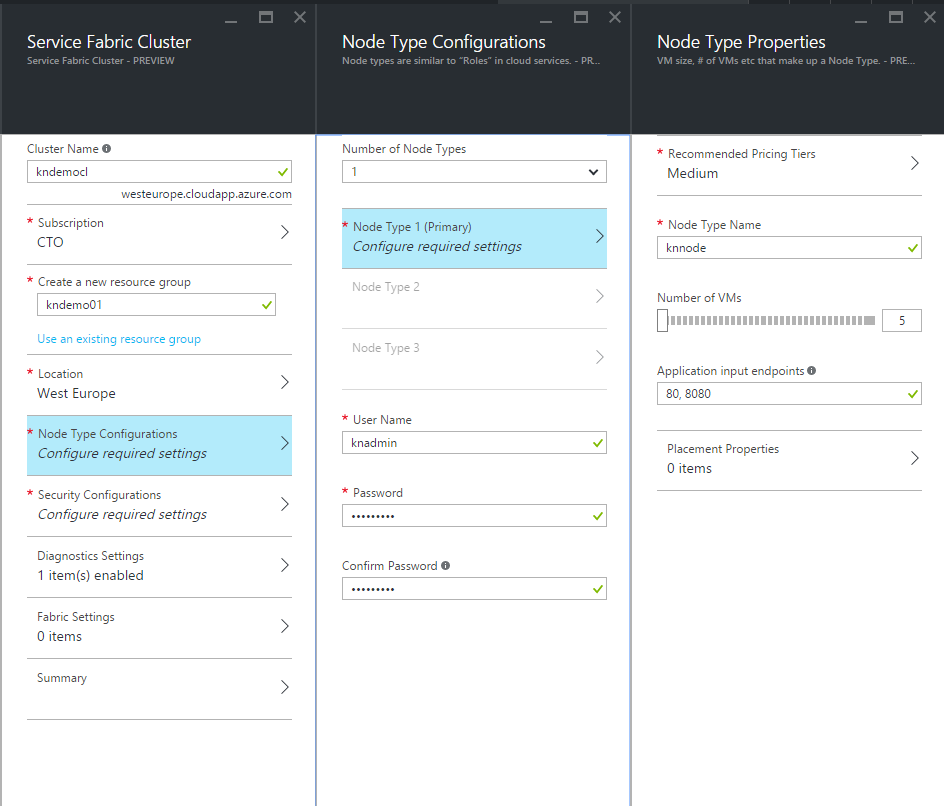
The deployment will take some time. In the meantime, we will explore some of the settings for a Service Fabric deployment in the Azure Portal. If you didn’t deploy using the template in Part 1, you can deploy the Service Fabric Cluster as part of the configuration in Part 2

**Part 2 – Explore Service Fabric Cluster configuration in the Azure Portal**

1. Access <https://portal.azure.com> and login with your credentials
2. Once logged in, click on **New**, **Compute, Service Fabric Cluster (preview)**



This will load the configuration page for the Service Fabric Cluster

1. Assign a cluster name, a new resource group name and click on ‘Node Type Configuration’. 

Assign a Node type name, select pricing tier, number of VMs and application input endpoints. Also verify that Number of Node Types is set to 1, and complete this step by assigning a username and password. Click ‘Ok’ twice once it’s done.

1. Next, click on ‘Security Configuration’, ‘Unsecure’ and ‘Ok’.

Explore the rest of the configuration, but we will leave those as is for now. If you didn’t deploy the Service Fabric Cluster in Part 1, you can now click on ‘Create’ and the cluster will be instantiated in your preferred region (US or Europe) with this configuration.

**Part 3 – Deploy Service Fabric Cluster a on local machine**

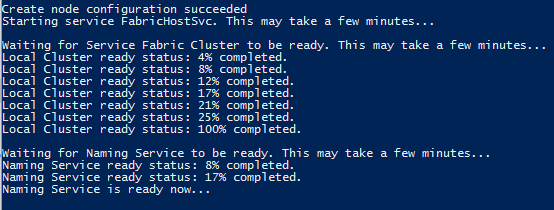
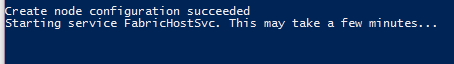
1. On your computer, open PowerShell and navigate to the Service Fabric SDK directory, which is normally located at ‘C:\Program Files\Microsoft SDKs\Service Fabric’

PS C:\Program Files\Microsoft SDKs\Service Fabric>

If this folder is not present, ensure you have successfully installed the Service Fabric SDK for Visual Studio.

1. Next, drill into the ‘Cluster Setup’ folder and invoke the PowerShell script that will install the local cluster

C:\Program Files\Microsoft SDKs\Service Fabric\ClusterSetup> .\DevClusterSetup.ps1 -Verbose

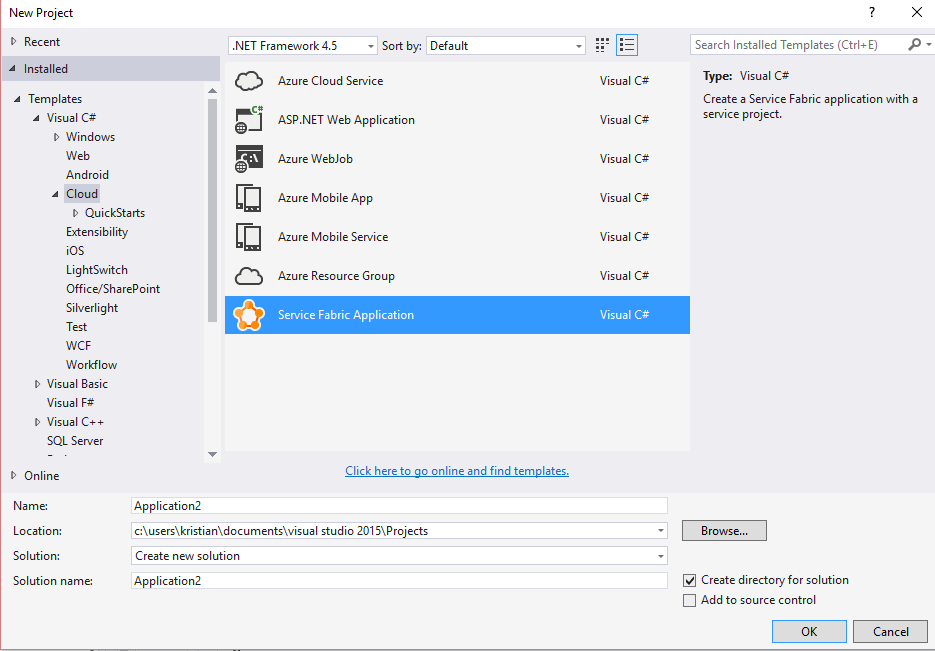
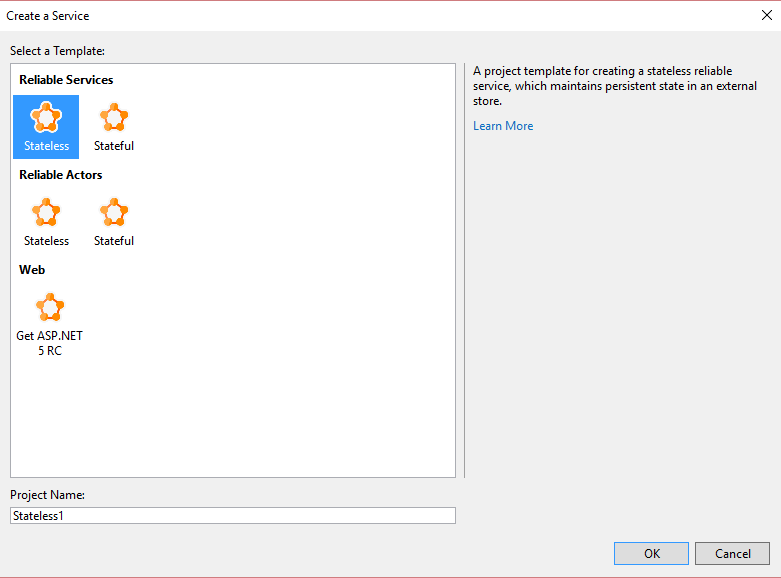
You have now successfully deployed a local Service Fabric Cluster.

1. Lab 2: Stateless Microservice

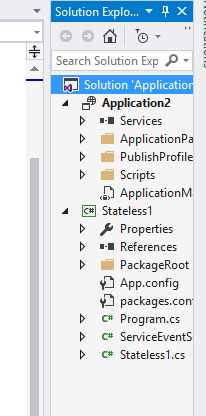
In this lab, you will explore the behavior and runtime of a stateless Microservice in Service Fabric.

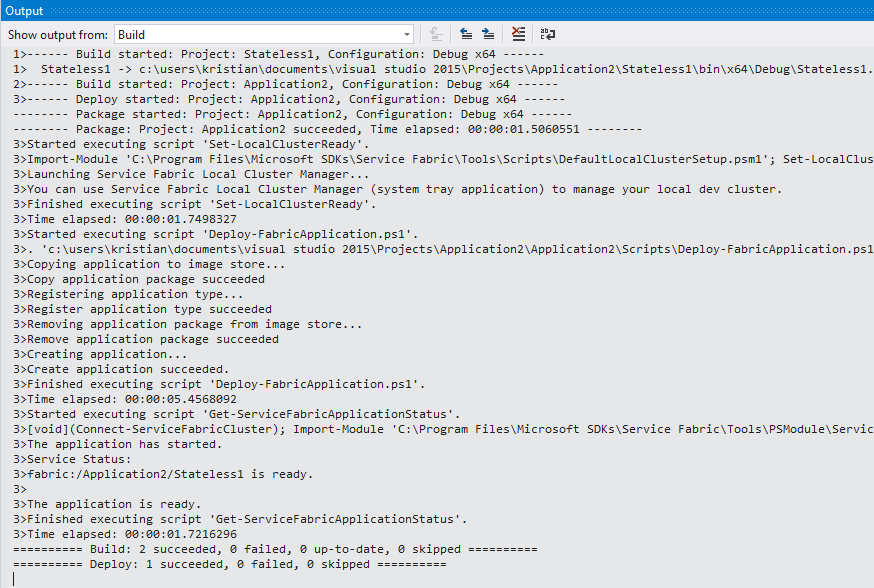
You will deploy the example application both to Azure and locally

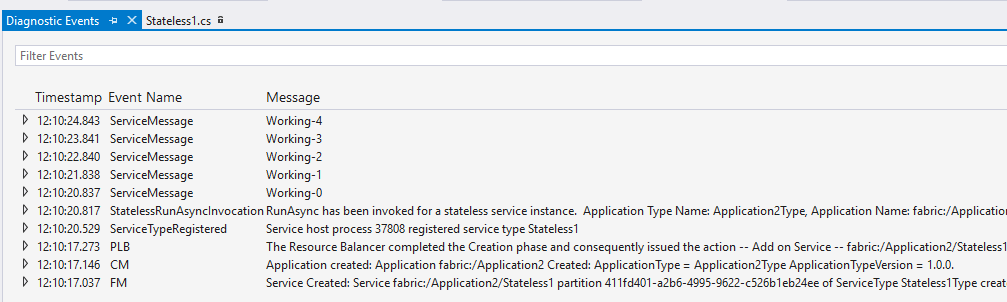
**Part 1 – Deploy Stateless Microservice to local cluster**

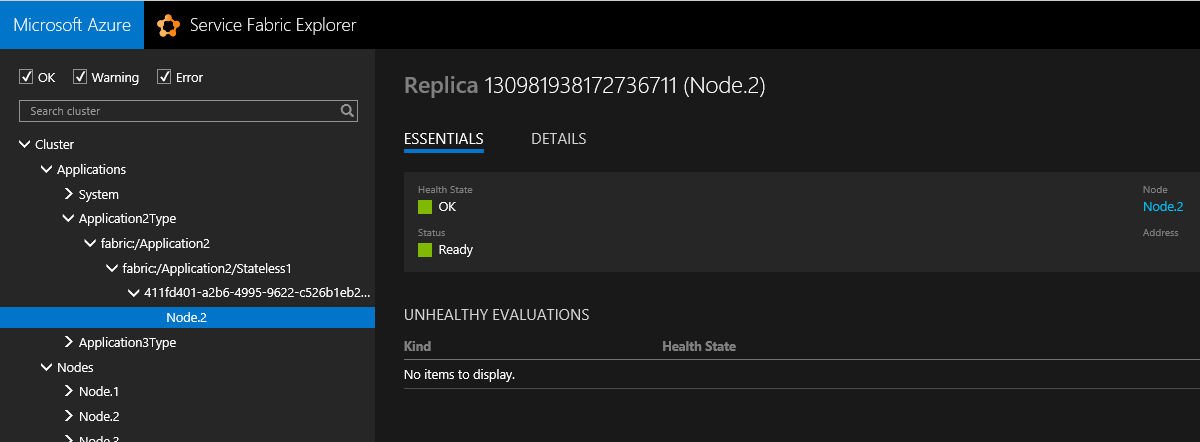
1. On your computer, open Visual Studio 2015 as an administrator, click on **New, Project, Templates, Cloud, Service Fabric Application**. Assign an application name and click ‘Ok’ to create the solution 
2. You will be prompted to select a template. In this exercise, we will create a Stateless Reliable Service. Click ‘Ok’ 
3. When the solution has opened in Visual Studio, explore the current code.

When ready, right-click on **Solution** and click **Deploy solution**

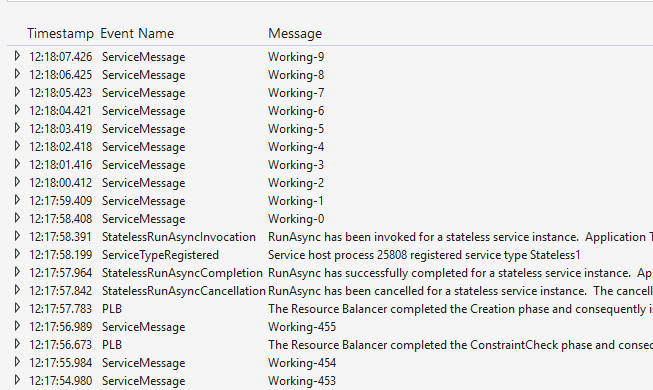


This will deploy the solution locally to your Service Fabric Cluster, and you will be notified that Service Fabric needs your permission to run. 

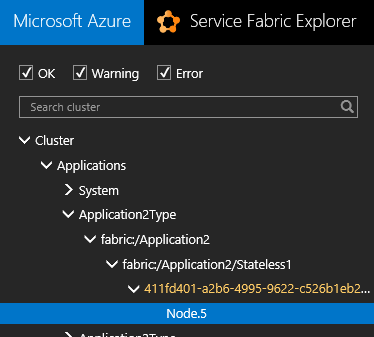
1. You should see the diagnostic pane showing that the application is incrementing the count 
2. While the application is running, open your preferred browser and navigate to <http://localhost:19080/Explorer>

This will launch the Service Fabric Explorer console, which is the web admin tool for your Service Fabric Cluster. Drill into the application you have deployed and notice the node it is currently running on 

1. Once detected which node that is running the application, click on the node from the node list and click **Actions, Deactivate (restart).**
2. Go back to Visual Studio and notice that the state wasn’t captured, meaning that the application has failed over to a different node in the cluster and restarted the counting



1. In Service Fabric Explorer, verify that the application has been moved to a different node in the cluster



1. Lab 3: Stateful Microservice

In this lab you will go through the required steps to deploy a stateful Microservice to a Service Fabric Cluster in Azure, and explore the structure and how it operates when different events occur.

**Part 1 – Deploy Service Fabric Application to Azure**

1. Open PowerShell and login to your Azure subscription
2. Run the following cmdlets to find the management connection endpoint to your newly created Service Fabric Cluster:

$rg = Get-AzureRmResourceGroup -Name SF10us

$sf = Get-AzureRmResource -ResourceGroupName $rg.ResourceGroupName -ResourceName knsfcl01 -ResourceType Microsoft.ServiceFabric/Clusters

$sf.Properties.ManagementEndpoint

PS C:\> $sf.Properties.ManagementEndpoint

http://knsfcl01.westus.cloudapp.azure.com:19080

To deploy to the cluster endpoint, we have to use port 19000. So run the following cmdlets to deploy the stateless application to your newly created Service Fabric Cluster

1. First, connect to the cluster endpoint using the following cmdlet:

Connect-ServiceFabricCluster -ConnectionEndpoint knsfcl01.westus.cloudapp.azure.com:19000 -Verbose

1. Next, we will import the PowerShell module for Service Fabric SDK and download the sample application to a newly created directory with the following cmdlets

# Import PSModule

Import-Module "$ENV:ProgramFiles\Microsoft SDKs\Service Fabric\Tools\PSModule\ServiceFabricSDK\ServiceFabricSDK.psm1"

# Download the sample application

mkdir -Path c:\appdemo\

Invoke-WebRequest -Uri "http://aka.ms/servicefabric-wordcountapp" -OutFile "C:\appdemo\WordCountV1.sfpkg"

1. We will then deploy the sample application to our Service Fabric Cluster in Azure with this cmdlet

Publish-NewServiceFabricApplication -ApplicationPackagePath C:\appdemo\WordCountV1.sfpkg -ApplicationName "fabric:/WordCount" -Verbose

The output should be similar to this:

Copying application to image store...

Copy application package succeeded

Registering application type...

Register application type succeeded

Removing application package from image store...

Remove application package succeeded

Creating application...

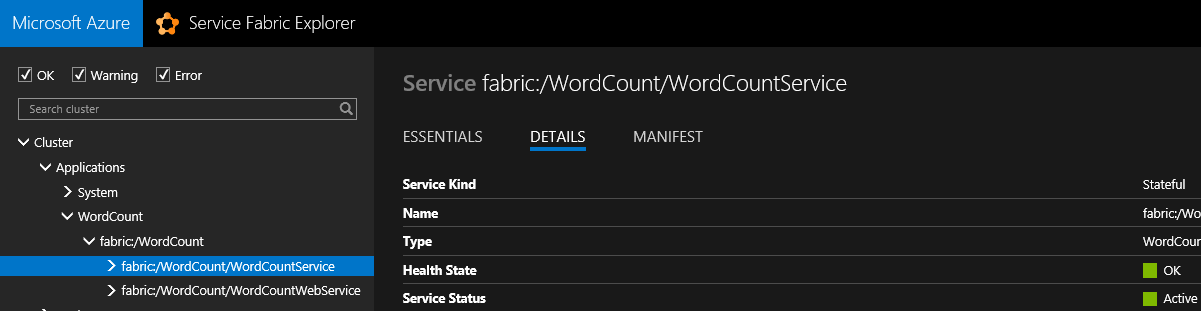
Create application succeeded.

Run the following cmdlet to verify that your application has been successfully deployed to your Azure Service Fabric Cluster

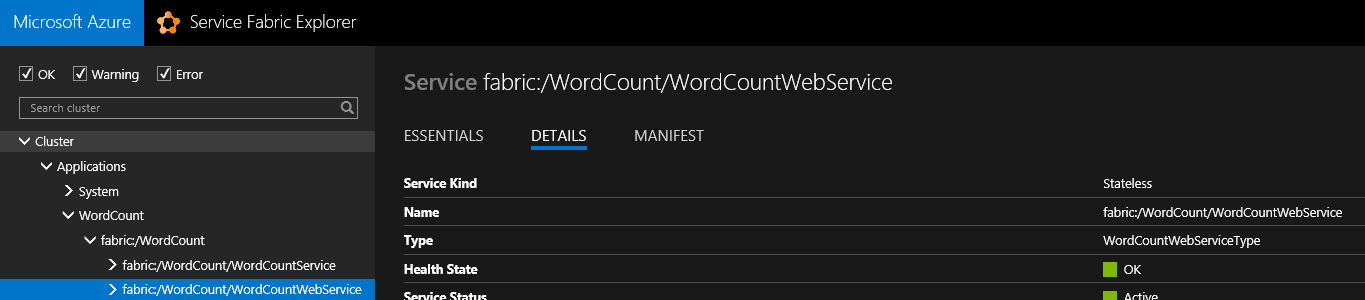
Start-Process http://knsfcl01.westus.cloudapp.azure.com:19080

This will launch your default browser, accessing the Service Fabric Cluster in Azure

1. Navigate to the WordCount application, fabric:/WordCount/WordCountService and click on **Details**. Service Kind show show ‘Stateful’

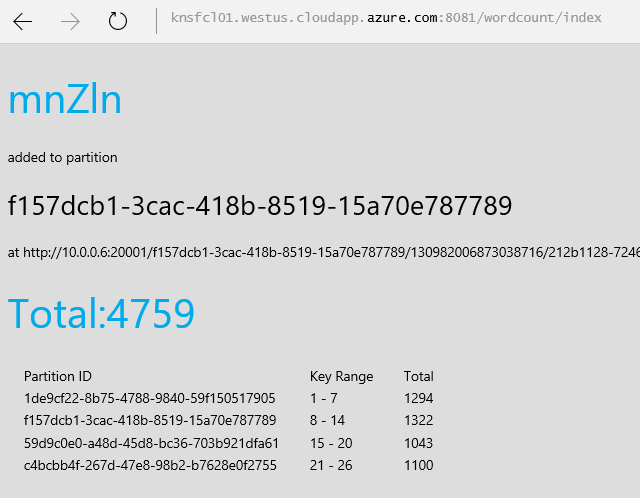


1. Click on fabric:/WordCount/WordCountWebService and see that the **Details** show it is ‘Stateless’

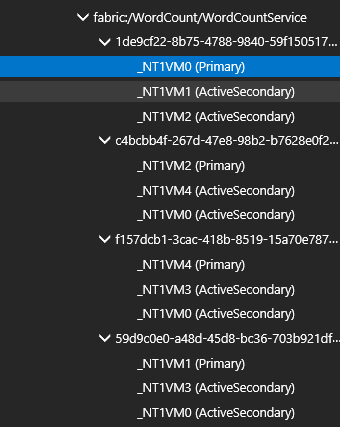


1. Access the application with the following URL (it must reflect your Service Fabric endpoint) <http://knsfcl01.westus.cloudapp.azure.com:8081/wordcount/index>

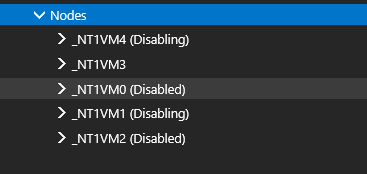
This shows an application that is basically counting random words and dynamically store them into partitions for the reliable collectors in Service Fabric



1. Go back to the Service Fabric Explorer and expand the fabric:/WordCount/WordCountService to see the nodes that are running the application



1. On 3 of those nodes, click on them and select **Actions** and **Deactivate(pause)**



1. on the 4th node, click **Actions** and **Deactivate(restart)**

Monitor the application and verify that the state isn’t lost during these exercises.

Once done, activate the nodes again.

**Part 2 – Verify Service Fabric Application using PowerShell**

1. From PowerShell, run the following cmdlet to get details about the application package

Get-ServiceFabricApplication -ApplicationName 'fabric:/WordCount'

ApplicationName : fabric:/WordCount

ApplicationTypeName : WordCount

ApplicationTypeVersion : 1.0.0

ApplicationStatus : Ready

HealthState : Ok

ApplicationParameters : {}

1. Next, run the following cmdlet to get the details about the service(s) within the application package

Get-ServiceFabricService -ApplicationName 'fabric:/WordCount'

ServiceName : fabric:/WordCount/WordCountService

ServiceKind : Stateful

ServiceTypeName : WordCountServiceType

IsServiceGroup : False

ServiceManifestVersion : 1.0.0

HasPersistedState : True

ServiceStatus : Active

HealthState : Ok

ServiceKind : Stateless

ServiceName : fabric:/WordCount/WordCountWebService

ServiceTypeName : WordCountWebServiceType

ServiceManifestVersion : 1.0.0

HealthState : Ok

ServiceStatus : Active

IsServiceGroup : False

1. Run the following cmdlet to get details about the stateful microservice

Get-ServiceFabricPartition 'fabric:/WordCount/WordCountService'

PartitionId : 1de9cf22-8b75-4788-9840-59f150517905

PartitionKind : Int64Range

PartitionLowKey : 1

PartitionHighKey : 7

PartitionStatus : Ready

LastQuorumLossDuration : 00:00:00

MinReplicaSetSize : 2

TargetReplicaSetSize : 3

HealthState : Ok

DataLossNumber : 130982006863363431

ConfigurationNumber : 8589934592

PartitionId : c4bcbb4f-267d-47e8-98b2-b7628e0f2755

PartitionKind : Int64Range

PartitionLowKey : 21

PartitionHighKey : 26

PartitionStatus : Ready

LastQuorumLossDuration : 00:00:00

MinReplicaSetSize : 2

TargetReplicaSetSize : 3

HealthState : Ok

DataLossNumber : 130982006863363431

ConfigurationNumber : 8589934592

PartitionId : f157dcb1-3cac-418b-8519-15a70e787789

PartitionKind : Int64Range

PartitionLowKey : 8

PartitionHighKey : 14

PartitionStatus : Ready

LastQuorumLossDuration : 00:00:00

MinReplicaSetSize : 2

TargetReplicaSetSize : 3

HealthState : Ok

DataLossNumber : 130982006863363431

ConfigurationNumber : 8589934592

PartitionId : 59d9c0e0-a48d-45d8-bc36-703b921dfa61

PartitionKind : Int64Range

PartitionLowKey : 15

PartitionHighKey : 20

PartitionStatus : Ready

LastQuorumLossDuration : 00:00:00

MinReplicaSetSize : 2

TargetReplicaSetSize : 3

HealthState : Ok

DataLossNumber : 130982006863363431

ConfigurationNumber : 8589934592

1. Run the following cmdlet to get details about the stateless service

Get-ServiceFabricPartition 'fabric:/WordCount/WordCountWebService'

PartitionId : d43293ad-8126-45a9-8d86-b2b12ca2a165

PartitionKind : Singleton

PartitionStatus : Ready

InstanceCount : 1

HealthState : Ok

You have now successfully completed lab 3

1. Lab 4: No Downtime Upgrade

In this lab, you will perform an update on the current Service Fabric application you deployed to your Service Fabric cluster in Azure in lab 3.

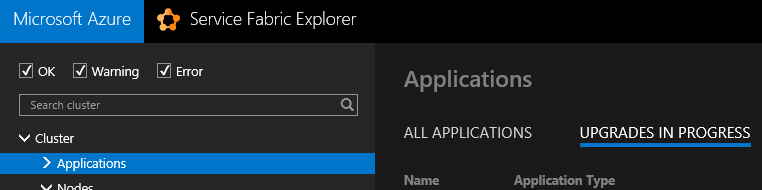
1. Open PowerShell and login to your Azure subscription
2. Download and store the updated Service Fabric Application on your local machine by running these cmdlets

mkdir c:\appupdate

Invoke-WebRequest "http://aka.ms/servicefabric-wordcountappv2" -OutFile "c:\appupdate\WordCountv2.sfpkg"

1. Once completed, you can update the application by running the cmdlet below, and also monitor the process in Service Fabric Explorer in real-time:

Publish-UpgradedServiceFabricApplication -ApplicationPackagePath C:\appupdate\WordCountv2.sfpkg -ApplicationName "fabric:/WordCount" -UpgradeParameters @{"FailureAction"="Rollback"; "UpgradeReplicaSetCheckTimeout"=1; "Monitored"=$true; "Force"=$true}



Copying application package to image store...

Copy application package succeeded

Registering application type...

Register application type succeeded

Start upgrading application...

ApplicationName : fabric:/WordCount

ApplicationParameters : {}

TargetApplicationTypeVersion : 2.0.0

UpgradeKind : Rolling

ForceRestart : False

UpgradeMode : Monitored

UpgradeReplicaSetCheckTimeout : 00:00:01

FailureAction : Rollback

HealthCheckRetryTimeout : 00:10:00

HealthCheckWaitDuration : 00:00:00

UpgradeDomainTimeout : 10675199.02:48:05.4775807

UpgradeTimeout : 10675199.02:48:05.4775807

ConsiderWarningAsError :

MaxPercentUnhealthyPartitionsPerService :

MaxPercentUnhealthyReplicasPerPartition :

MaxPercentUnhealthyServices :

MaxPercentUnhealthyDeployedApplications :

ServiceTypeHealthPolicyMap :

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Waiting for upgrade...

Upgrade completed

The expected behavior in the application now is that the counting is taking more time and that the first partition ends up slightly with the more volume

You have now successfully completed lab 4.