**Azure Service Fabric** is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable micro-services and containers.

Service Fabric is platform for building and managing cloud-scale applications running in containers.

Service Fabric: is alternative to Docker Swarm, DC/OS, Kubernetes and other containers technologies to develop scalable services.

Service Fabric enables you to build applications that consist of micro services. Stateless micro services (such as Web APIs, protocol gateways and web proxies) do not maintain a mutable state outside a request and its response from the service.

State-full micro services (such as Web Applications, user accounts, databases, devices, shopping carts, and queues) maintain a mutable, authoritative state beyond the request and its response.

Many existing Microsoft Azure cloud services run on top of Service fabric like document db, SQL server, power Bi, Skype for Business etc.

Deploy different versions of the same application side by side, and upgrade each application independently, with hundreds or thousands of applications or containers per machine.

Manage the lifecycle of your applications without any downtime, including breaking and nonbreaking upgrades.

Scale out or scale in the number of nodes in a cluster. As you scale nodes, your applications automatically scale.

**Service Fabrics applications programing models**: there are there general approaches

1. Reliable services are of two types (easiest one like console app, )
   1. Stateless: console applications, **Web API** get fit here.
   2. State-full: with its own transections storage (can store information)
2. Reliable Actors: is build on top of State-full reliable services
3. Guest Executables: can run any application written in any language without any code change.

For further **documentation**, samples, how to guides and tutorials: s

<https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-overview>

**Application Vs Reliable Service (which is written using Service Fabric):**

Both are similar except it only runs on x64 bit OS, little learning curve. Designed to survive the services outage, while running on multiple machines.

Service Fabric is tailored to create cloud native services that can start small, as needed, and grow to massive scale with hundreds or thousands of machines.

Cluster = Nodes

Node = normal computer with OS and Service Fabric runtime. Each node can host multiple applications.

**Stateful reliable service can sstore some data in the application**:

Service Fabric allows you to consistently and reliably store your data right inside your service by using reliable collections. Reliable collections are a set of highly available and reliable collection classes that are familiar to anyone who has used C# collections.

**SF Deployment**: Need MS Build & PowerShell

Another option is Azure CLI

MSBuild Path: C:\Windows\Microsoft.NET\Framework\v4.0.30319

Configure MSBuild path in System Properties => Environment Variables => Path => add

**Generate the Package**: Run power shell as admin and go to the solution folder and just type MsBuild

<https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-tutorial-create-dotnet-app>

**Service Fabric Debug/Run on local machine**:

**To run and debug Service Fabric locality**: Start Service Fabric local cluster (near clock) => Switch cluster node to 1.

**Right click Service Fabric** project in VS => Publish => Switch cluster node to 1 =>

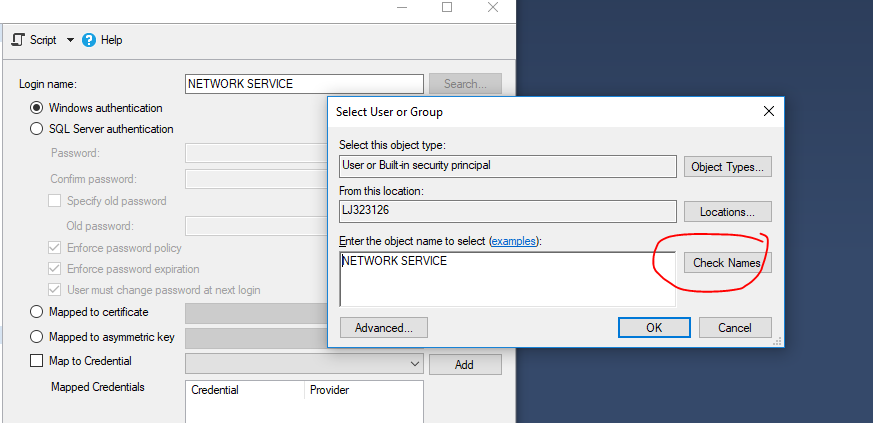
Right click Service Fabric local cluster => Click Manage Local Cluster (In browser) => go till the last **node\_Node\_o** to get the url.

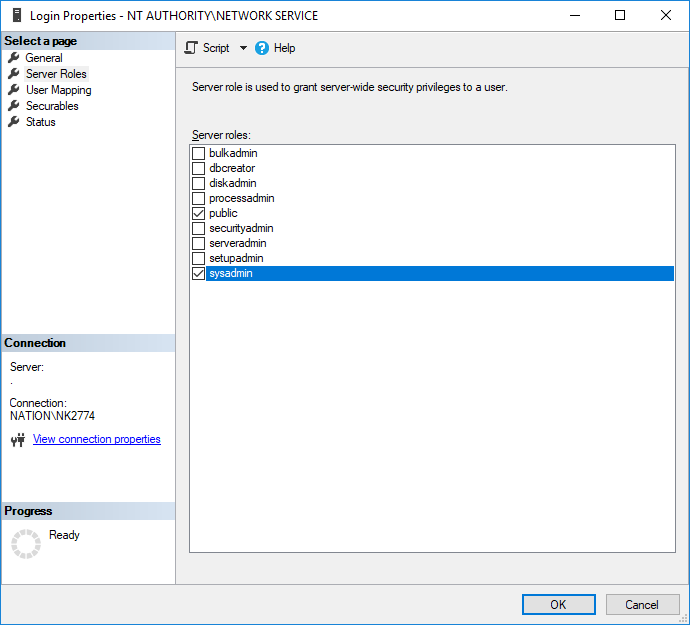
Debugging with VS: Debug Menu => Attach process => click check box show processes form all users => find the project name and attach.

**For Db ConnectionStrings Integrated Security = true** (VIP for Local Cluster)

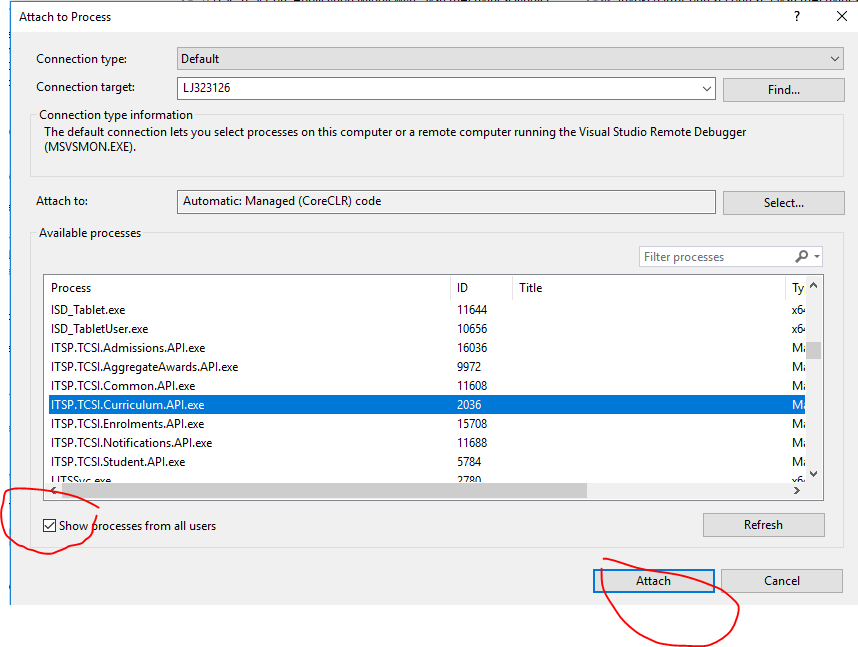
Go to SQL Server Management Studio => Security => Logins => New Login

=>type Login Name: “**NETWORK SERVICE**” (click on Search) => Server Roles (left side tab)=> **sysadmin**





Debug local cluster in VS



+++++++++++++++++++++++

**Service Fabric Cluster** (use local cluster):

Service Fabric Cluster provides a sophisticated, lightweight runtime to build distributed, scalable, stateless, and state-full micro services running in containers.

We need to setup cluster at azure (Service Fabric cluster in Azure using the Azure portal) before publishing Service Fabric app.

The local Service Fabric cluster on a developer workstation version is just like the real cluster in cloud/production system as there is no difference in between both.

Service Fabric only runs on 64-bit OS and can use VS 2017 community Edition.

For development environment we need:

* Install Service Fabric tooling
* Install Service Fabric SDK (with includes Service Fabric Cluster)

Install certificates (Install-Certificates and Install-Certificates\_User) from O:\HEIMS\O Developer\

++++++++++++++++++

**Service Fabric project**: Usually there is only on Service Fabric project in whole solution. It contains how to description how to deploy the application, what are the application parameters, and which services it consists of.

**Service Lifecycle**: both state-full and stateless service have identical sequence of events.

state-full services have slightly more complicated life.

To get a complete understanding of how ASP.NET Core integrates with Service Fabric, we strongly recommend reading through the: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-communication-aspnetcore>

For Stateless reliable services we can manage state in an external store such as Azure Document-Db or SQL Db or otherwise your app service has no persistent state.

Service Fabric with Azure API Management: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-api-management-overview>

A Service Fabric cluster is a network-connected set of virtual or physical machines into which your microservices are deployed and managed.

Create a test cluster in Azure

Tips:

If Git is not working use TFS. <https://stackoverflow.com/questions/44637305/vs-2017-cannot-add-project-to-tfs-source-control>

You can right click the service fabric project and publish to local cluster (node) or cloud.

.\sign.ps1; .\ Deploy-FabricApplication.ps1

note: get sign.ps1 file from :C:\Developer\Source\repos\its-apps-code

Install the Microsoft Azure Service Fabric SDK (the latest to keep the local cluster updated)

Create a Windows Service Fabric cluster on Azure, by following this tutorial: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-tutorial-create-vnet-and-windows-cluster> (Deploy a Service Fabric cluster running Windows into an Azure virtual network, by using PowerShell and a template). This tutorial describes a production scenario.

For testing only create a Test Service Fabric cluster: <https://docs.microsoft.com/en-us/azure/service-fabric/scripts/service-fabric-powershell-create-secure-cluster-cert>

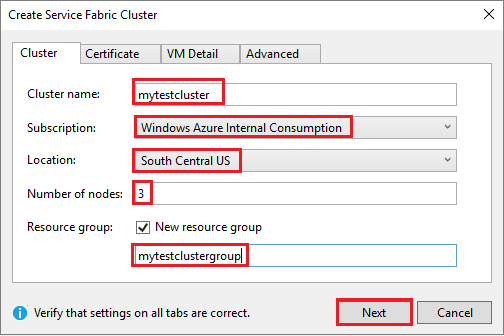
Go to <https://portal.azure.com> otherwise you can create form Visual Studio.

Create a test cluster in Azure using VS: <https://docs.microsoft.com/en-au/azure/service-fabric/service-fabric-tutorial-deploy-app-to-party-cluster> s

In **Solution Explorer**, right-click on your project and select **Publish**.

In **Connection Endpoint**, select **Create New Cluster**. If you're deploying to an existing cluster, select the cluster endpoint from the list. The Create Service Fabric Cluster dialog opens.

In the **Cluster tab**, enter the Cluster name (for example, "mytestcluster"), **select your subscription**, select a region for the cluster (such as South Central US), **enter the number of cluster nodes** (we recommend three nodes for a test cluster), and enter a resource group (such as "mytestclustergroup"). Click Next.



Go to <https://portal.azure.com> Click Create a Resource => find Service Fabric Cluster => click create

Enter name “my-api-test-cluster” => Default VM credentials “NaveeKhan” => password “7-1aZ” => new resource group “my-test-cluster-resource-group” => node select one or 2 => Virtual machine size (sort by price and pick) => click ok => Security select 'Basic' => configure Key vault =>

**Create a new vault** => name: my-test-key-vault-00 => all network can access => create

**Certificate** name “certificate01-for-fabric-cluster” => ok => summary => create (you can also download template and parameter)

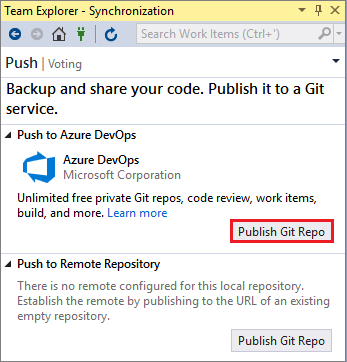
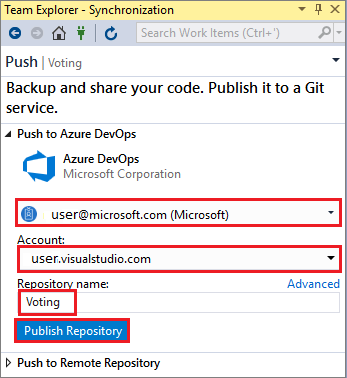
sIf required in Ports, enter your project's service endpoint (for example, 8080). When the cluster is created, these application ports are opened in the Azure load balancer to forward traffic to the cluster.

**Deploy an application with CI/CD to a Service Fabric cluster**: <https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-tutorial-deploy-app-with-cicd-vsts>

First of all we need a **Git Repo in Azure DevOps** and to generate **builds** (CI).

Create a new local Git repo for your project by selecting **Add to Source Control** => Git on the status bar in the lower right-hand corner of Visual Studio. Otherwise right click the solution in VS and click “**Add solution to source code**”.

In the Push view in Team Explorer (Sync), select the Publish Git Repo button under “**Push to Azure DevOps**” => “publish Git Repo” => “Publish Repository”. You can also do “git push --set-upstream origin master”

Go to <https://dev.azure.com> => Sign in to either your organization in Azure DevOps ([https://dev.azure.com/{yourorganization}](https://dev.azure.com/%7byourorganization%7d))

**Create a build pipeline**: Select the **Pipelines** tab, then **Builds**, then **click New Pipeline**.

Select Azure Repos Git or TFS as source.

Select a template, select the **Azure Service Fabric application template** and **click Apply**.

In Tasks, enter "Hosted VS2017" as the Agent pool.

Under Triggers, enable continuous integration by **checking Enable continuous integration**. Within Branch filters, the Branch specification defaults to **master**. Select Save and queue to manually **start a build**.

Builds also trigger upon push or check-in.

Once you verify that the build executes successfully, define a release pipeline that deploys your application to a cluster.

You can **Edit** build by going to **Pipelines** tab, then **Builds** and **Edit** (right top corner).

**Create a release pipeline**:

Select the **Pipelines tab**, then **Releases**, then **+ New pipeline**. In Select a template, select **the Azure Service Fabric Deployment** template from the list and then **Apply**.

Select **Tasks** => **Environment 1**/Dev/ Stage 1/Test/Staging and then **+New** to add **a new cluster connection**.

In the Add new Service Fabric Connection view select **Certificate Based** or Azure Active Directory authentication. Specify a connection name of "mysftestcluster" and a cluster endpoint of "tcp://mysftestcluster.southcentralus.cloudapp.azure.com:19000" (or the endpoint of the cluster you are deploying to).

find you cluster in azure go to <https://portal.azure.com> => Resource groups => select Resource group for cluster => find/click on Type "Service Fabric cluster" => select the cluster

Find Client connection endpoint e.g. tcp://my-api-test-cluster.australiaeast.cloudapp.azure.com:19000

For certificate-based authentication, add the Server certificate thumbprint of the server certificate used to create the cluster.

To find certificate thumbprint inside the Azure portal select the cluster => **Security** => under **Cluster certificates** => **Thumbprint** copy e.g. 6E7F5BBC7C69C228812687873DE4A21618897A00

In **Client certificate**, add the base-64 encoding of the client certificate file. See the help pop-up on that field for info on how to get that base-64 encoded representation of the certificate. Also add the Password for the certificate. **You can use the cluster or server certificate if you don't have a separate client certificate**.

Tip: download the certificate (Client certificate) by selecting corresponding **KeyVault** in azure portal => **Secrets** > Select **certificate** => **Download in PFX/PEM format**

Generate the Base-64 encoded string using power shell. Open **Power shell ISE** copy past bellow command. It will convert the .pfx to Base-64 encoded string in a txt file.

$fileContentBytes = get-content 'C:\temp\Certificate\my -key-vault-certificate.pfx' -Encoding Byte

[System.Convert]::ToBase64String($fileContentBytes) | Out-File 'C:\temp\Certificate\pfx-bytes.txt'

Adding client certificates via portal: Navigate to the Security section, and select the '+ Add' button on top of the security section.

For Azure Active Directory credentials, add the Server certificate thumbprint of the server certificate used to create the cluster and the credentials you want to use to connect to the cluster in the Username and Password fields.

Next, **add a build artifact** to the pipeline so the release pipeline can find the output from the build:

Select **Pipeline** => Artifacts => +Add. In Source (Build definition), select the build pipeline you created previously. **Click Add** to save the build artifact.

**Enable a continuous deployment trigger** so that a release is automatically created when the build completes. Click the **lightning icon** in the **artifact**, enable the trigger, and **click Save** to save the release pipeline.

**Manually create a release**: Select + **Release** => **Create a Release** (top right corner) => select the Stages (Dev/PreProd) => enter some description => click to see the progrees

You can **monitor the release progress** in the Releases tab.

If not deployed, click on Stages (Dev/PrePord) => enter some comments and Deploy.

If deployment failed, go to Release => click Edit (top right corner) to fix the release => Tasks => select the stage/Dev/PreProd => Step causing error and fix.

**Verify that the deployment succeeded** and the application is running in the cluster. Open a web browser and navigate to http://mysftestcluster.southcentralus.cloudapp.azure.com:19080/Explorer/. Note the application version, in this example it is "1.0.0.20170616.3". <https://my-api-test-cluster.australiaeast.cloudapp.azure.com:19080/Explorer>

**Service Fabric cluster security scenarios**:

<https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-cluster-security>

Cluster exposes management endpoints to the public internet, anonymous users can connect to it. Unsecured clusters are not supported for production workloads.

**Node-to-node security** helps secure communication between the VMs or computers in a cluster.

Service Fabric uses X.509 server certificates that you specify as part of the node-type configuration when you create a cluster.

**Client-to-node security** authenticates clients and helps secure communication between a client and individual nodes in the cluster. This type of security helps ensure that only authorized users can access the cluster and the applications that are deployed on the cluster. Clients are uniquely identified through either their Windows security credentials or their certificate security credentials.

Personal access tokens (PATs) are alternate passwords that you can use to authenticate into Azure DevOps. <https://docs.microsoft.com/en-au/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?toc=%2Fvsts%2Fsecurity%2Ftoc.json&bc=%2Fvsts%2Fsecurity%2Fbreadcrumb%2Ftoc.json&view=azure-devops#using-pats>

Good place to start understanding Service Fabric: <https://docs.microsoft.com/en-au/azure/service-fabric/>

Service Fabric Getting Started Sample (VS 2017 mix of API, Exe and MVC/UI): <https://azure.microsoft.com/en-au/resources/samples/service-fabric-dotnet-getting-started/>

**State API**: Azure Service Fabric has it’s state api for each service type which is isolated from each other. It’s much faster than talking to the external database etc. as it stores state on the local machine on the same disk. There is no need to open network connections, login to the databases, handshaking and encryption etc. Every time some data is written on it is replicated to few other multiple copies and no data is lost if one fails. So if primery copy gets fails, it makes another copy active.

The simplest way to store anything in a service is by using built-in **ReliableStateManager**. It’s like a normal dictionary collection.

All the operation in Service Fabric requires Transections just like database transections.

Quorum is set of one primary and few Replica/inactive copies of a service. Size of the Quorum is 3 nodes (I active/primary and 2 Replica).

Communication between the services we can use Service Remoting (is built-in and is default and preferred way), other options are WCF and Http (like web api). For Service Remoting we need to add nugget package Services.Remoting.

**Azure Service Fabric Runtime**: it runs your exe, registers the service type and service is ready.

It creates service instance and then ask to create listeners (**ServiceInstanceListener**), after that open listener loop (time to listen incoming requests). After this RunAsync (CancellationToken ).

Before Service Fabric shutdowns the service it cancels the Cancellation Token passed in RunAsync. After this it closes all the listeners and finally destroy class instance. We should care about the Cancellation Token in case of state-full service.

**Actor Model Support**: is a framework built on top of Reliable Services. It’s most popular. Actor = code (Reliable Services) + state (e.g. Reliable State) + mailbox + can send messages (e.g. using service remoting).

Message sent and received are like method calls. In mail box messages are queued and it can only process one message at a time and returns the results.

Multiple actors can run parallel, but one actor processes messages sequentially.

In VS add nugget package ServiceFabric.Actors.