**Step 1: Prepare Development Environment**

Windows:

1. Set Administration Authoriaztion:

**Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force -Scope CurrentUser**

1. PowerShell command:

**& "$ENV:ProgramFiles\Microsoft SDKs\Service Fabric\ClusterSetup\DevClusterSetup.ps1" -CreateOneNodeCluster**

Linux:

1. OS

Ubuntu 16.04

1. Update apt source

Add service fabric to the source list

**sudo sh -c 'echo "deb [arch=amd64] http://apt-mo.trafficmanager.net/repos/servicefabric/ trusty main" > /etc/apt/sources.list.d/servicefabric.list'**

Add dotnet repository to the source list

**sudo sh -c 'echo "deb [arch=amd64] https://apt-mo.trafficmanager.net/repos/dotnet-release/ xenial main" > /etc/apt/sources.list.d/dotnetdev.list'**

Add new GPG security key to apt keyring

**sudo apt-key adv --keyserver apt-mo.trafficmanager.net --recv-keys 417A0893**

**sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 417A0893**

Add Docker office GPG key to apt keyring.

**sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –**

Refresh the package list

**sudo apt-get update**

1. Configure SDK for the container and guest exe.

Install Service Fabric SDK

**sudo apt-get install servicefabricsdkcommon**

the following command to accept the service fabric license

**echo "servicefabric servicefabric/accepted-eula-v1 select true" | debconf-set-selections**

**echo "servicefabricsdkcommon servicefabricsdkcommon/accepted-eula-v1 select true" | debconf-set-selections**

run the SDK installation scripts.

**sudo /opt/microsoft/sdk/servicefabric/common/sdkcommonsetup.sh**

运行安装常用 SDK 包的步骤以后，应该就可以运行 yo azuresfguest 或 yo azuresfcontainer，通过来宾可执行文件或容器服务创建应用。

Set the $NODE\_PATH environment

**export NODE\_PATH=$NODE\_PATH:$HOME/.node/lib/node\_modules**

1. Config the XPlat Service Fabric CLI

Install the Node.js

Clone the github repository.

**git clone** <https://github.com/Azure/azure-xplat-cli.git>

switch to repository and use NPM to install the CLI dependency

**cd azure-xplat-cli**

**npm install**

create the link from bin/azure to /usr/bin/azure

**sudo ln -s $(pwd)/bin/azure /usr/bin/azure**

start to complete the Service Fabric command

**azure --completion >> ~/azure.completion.sh**

**echo 'source ~/azure.completion.sh' >> ~/.bash\_profile**

**source ~/azure.completion.sh**

1. Configure the local cluster

Run the cluster script

**sudo /opt/microsoft/sdk/servicefabric/common/clustersetup/devclustersetup.sh**

open web browers.

<http://localhost:19080/Explorer>

1. Install the Java SDK in Linux

**sudo apt-get install servicefabricsdkjava**

**sudo /opt/microsoft/sdk/servicefabric/java/sdkjavasetup.sh**

Install Eclipse Neon add-in.

请选择“帮助”>“安装新软件...”

在“使用”文本框中，输入 <http://dl.microsoft.com/eclipse>

1. Install the dotnet in Linux

**sudo apt-get install servicefabricsdkcsharp**

**sudo /opt/microsoft/sdk/servicefabric/csharp/sdkcsharpsetup.sh**

**sudo apt-get update**

**sudo apt-get install servicefabric servicefabricsdkcommon servicefabricsdkcsharp servicefabricsdkjava**

Switch node from 5 Node to 1 Node

Powershell command

**& "$ENV:ProgramFiles\Microsoft SDKs\Service Fabric\ClusterSetup\DevClusterSetup.ps1" -CreateOneNodeCluster**

**Task 1:**

1. Close hyper-v service

**bcdedit /set hypervisorlaunchtype off 关闭**

**bcdedit /set hypervisorlaunchtype auto 开启**

1. Set up the service fabric manager in local host

**& "$ENV:ProgramFiles\Microsoft SDKs\Service Fabric\ClusterSetup\DevClusterSetup.ps1"**

1. Import service fabric modules

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

**Download the V1 application: http://aka.ms/servicefabric-wordcountapp**

1. Create directory for deploy the application

**mkdir c:\ServiceFabric\**

**cd c:\ServiceFabric\**

1. Connect the local cluster

**Connect-ServiceFabricCluster localhost:19000**

1. Create a new application using SDKs deployment command with

**Publish-NewServiceFabricApplication -ApplicationPackagePath c:\ServiceFabric\WordCountV1.sfpkg -ApplicationName "fabric:/WordCount"**

1. Review the wordcount application

<http://localhost:8081/wordcount/index.html>

1. Query all deployed application on cluster

**Get-ServiceFabricApplication**

**Get-ServiceFabricService -ApplicationName 'fabric:/WordCount'**

**Get-ServiceFabricPartition 'fabric:/WordCount/WordCountService'**

1. Update realtime

**Download the WordCountV2.sfpkg** <http://aka.ms/servicefabric-wordcountappv2>

**Publish-UpgradedServiceFabricApplication -ApplicationPackagePath C:\ServiceFabric\WordCountV2.sfpkg -ApplicationName "fabric:/WordCount" -UpgradeParameters @{"FailureAction"="Rollback"; "UpgradeReplicaSetCheckTimeout"=1; "Monitored"=$true; "Force"=$true}**

Get-ServiceFabricService -ApplicationName 'fabric:/WordCount'

1. Remove the service fabric

Cleaning up:

**Unpublish-ServiceFabricApplication -ApplicationName "fabric:/WordCount"**

Or click the delete application in Action button.

**Remove-ServiceFabricApplicationType -ApplicationTypeName WordCount -ApplicationTypeVersion 2.0.0**

**Remove-ServiceFabricApplicationType -ApplicationTypeName WordCount -ApplicationTypeVersion 1.0.0**

1. Stop local cluster , Remove local cluster.

**Task 2: Docker Container**

1. Install the container function:

**$Env:Path = "${Env:Path};c:\Program Files (x86)\Git\bin"**

**Enable-WindowsOptionalFeature -Online -FeatureName containers -All**

**Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All**

**Restart-Computer -Force**

1. Install Docker

**Invoke-WebRequest "https://get.docker.com/builds/Windows/x86\_64/docker-17.03.0-ce.zip" -OutFile "$env:TEMP\docker.zip" -UseBasicParsing**

**Expand-Archive -Path "$env:TEMP\docker.zip" -DestinationPath $env:ProgramFiles**

**# Add path to this PowerShell session immediately**

**$env:path += ";$env:ProgramFiles\Docker"**

**# For persistent use after a reboot**

**$existingMachinePath = [Environment]::GetEnvironmentVariable("Path",[System.EnvironmentVariableTarget]::Machine)**

**[Environment]::SetEnvironmentVariable("Path", $existingMachinePath + ";$env:ProgramFiles\Docker", [EnvironmentVariableTarget]::Machine)**

**dockerd --register-service**

Start-Service Docker

When you first run the window container, Docker CE should pull down the basic image of container. To run the following powershell command:

**docker pull microsoft/mssql-server-windows-developer**

**docker pull microsoft/aspnet:4.6.2**

Create the docker virtual machine.

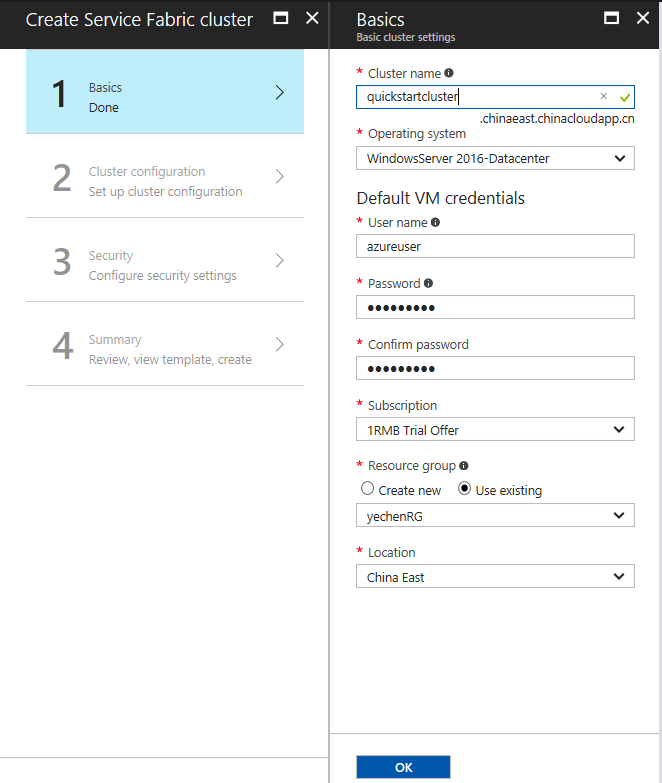
**docker-machine create --driver virtualbox chenyeserver**

**https://linux.cn/article-6113-1.html**

//

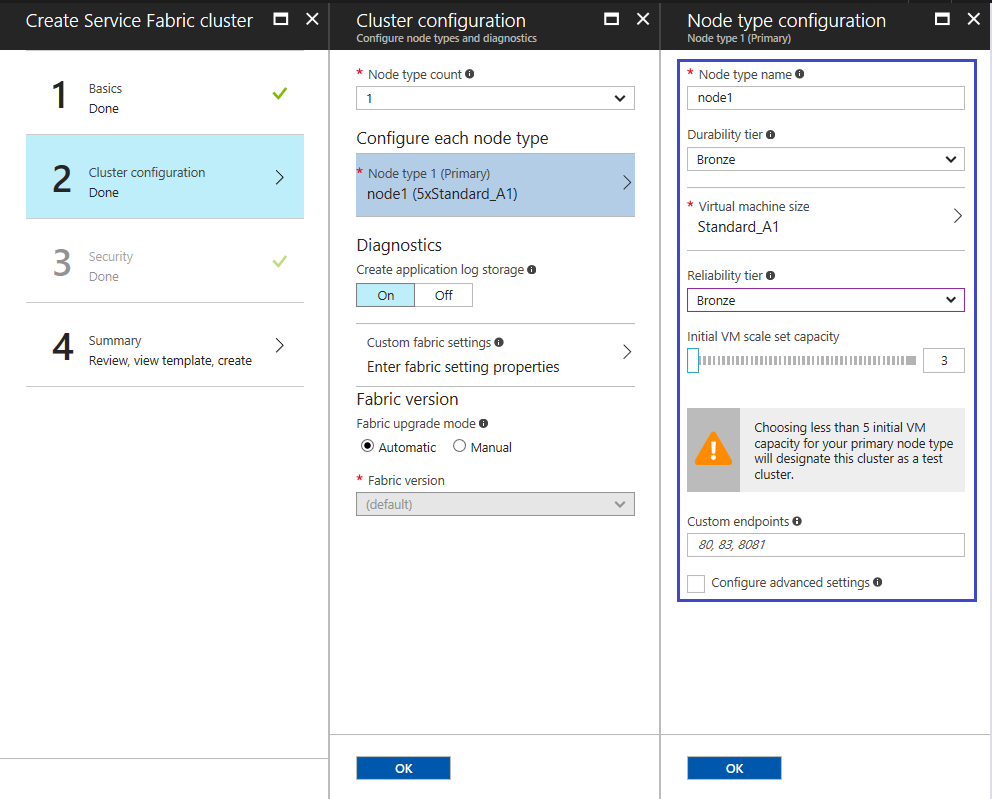
**docker run -d -p 80:80 --name webserver nginx**

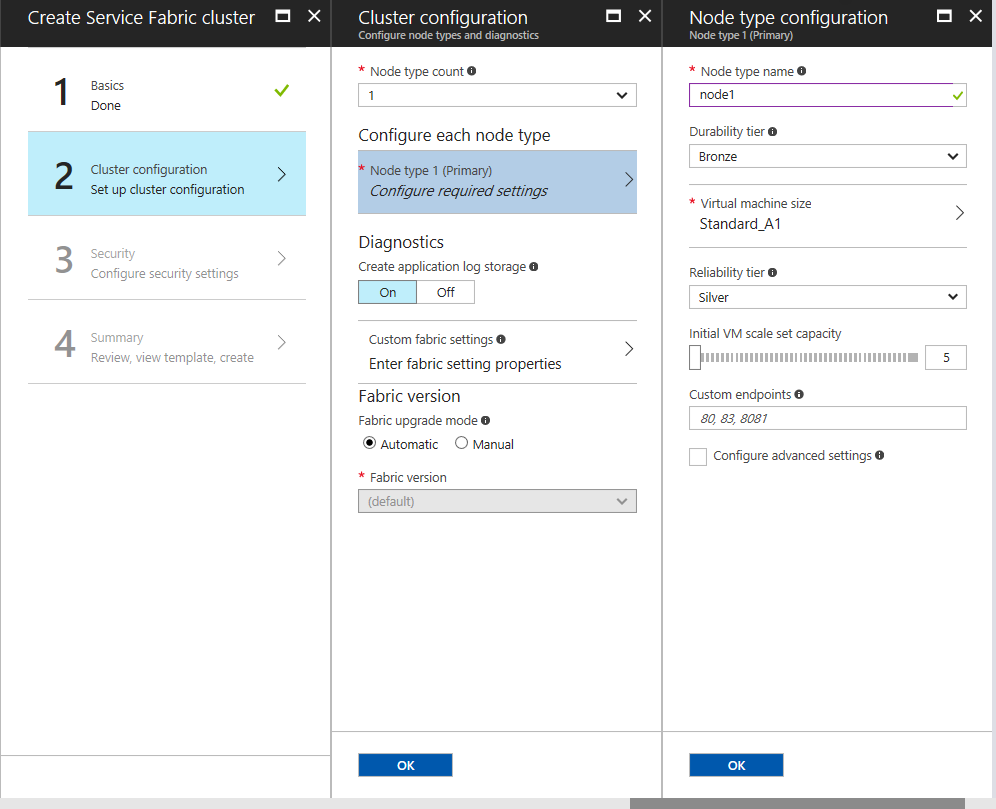
Create the Service Fabric Cluster:

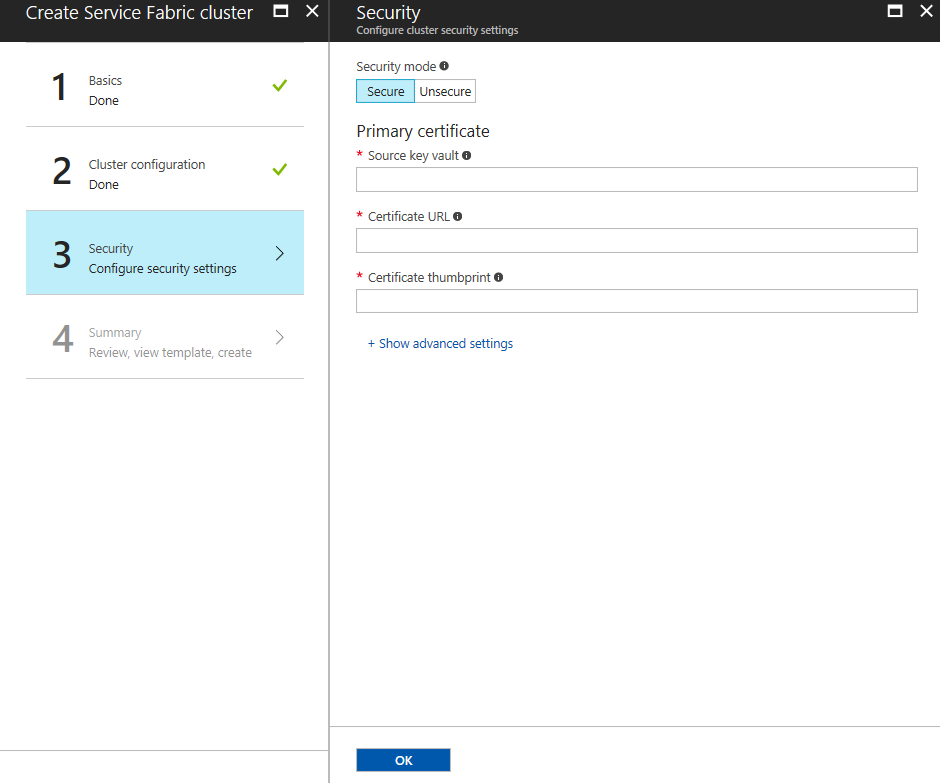


User: azureuser

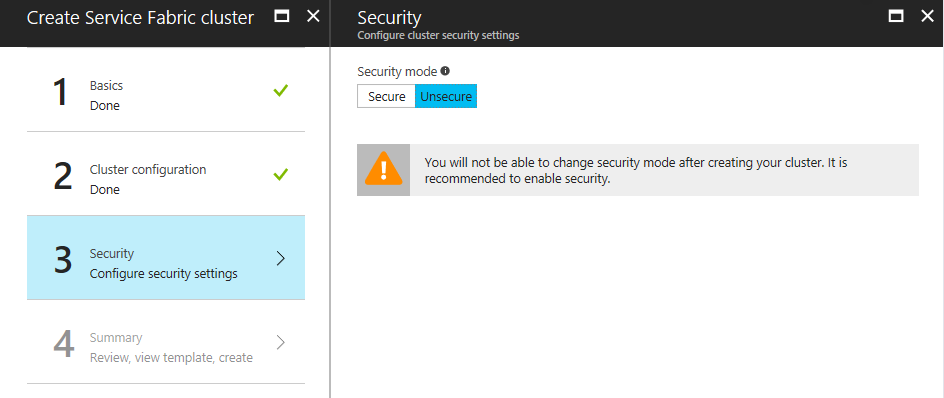
Password: Lindsayc5

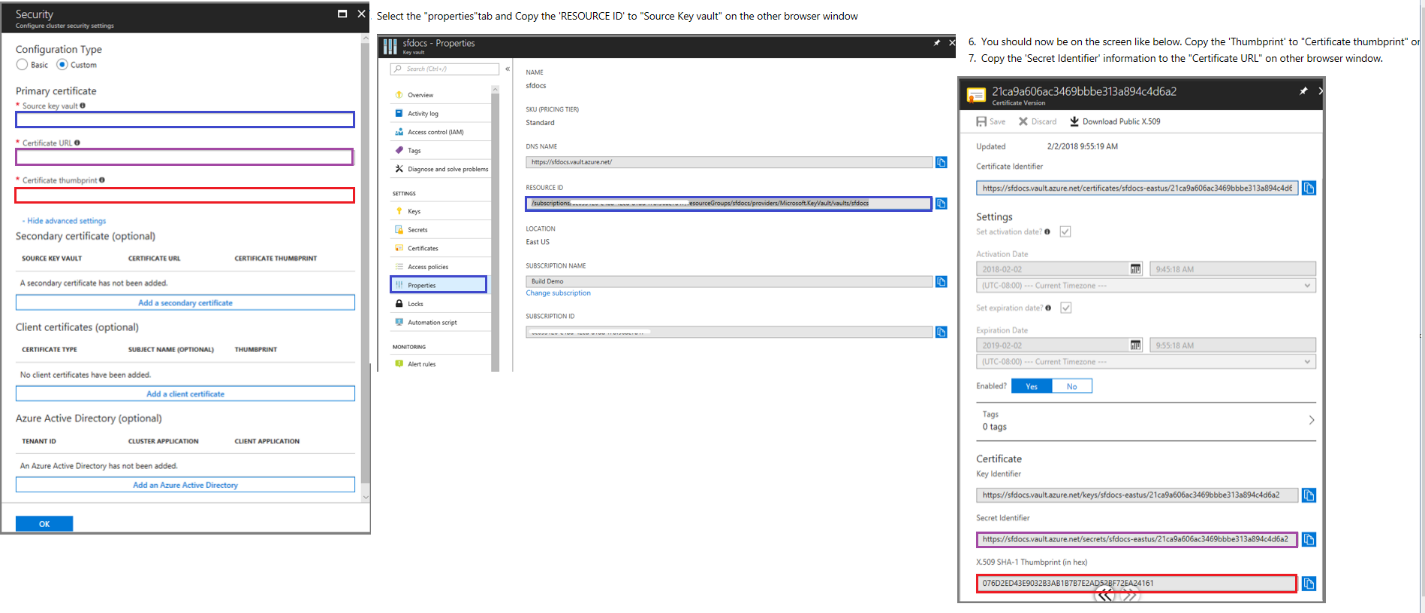


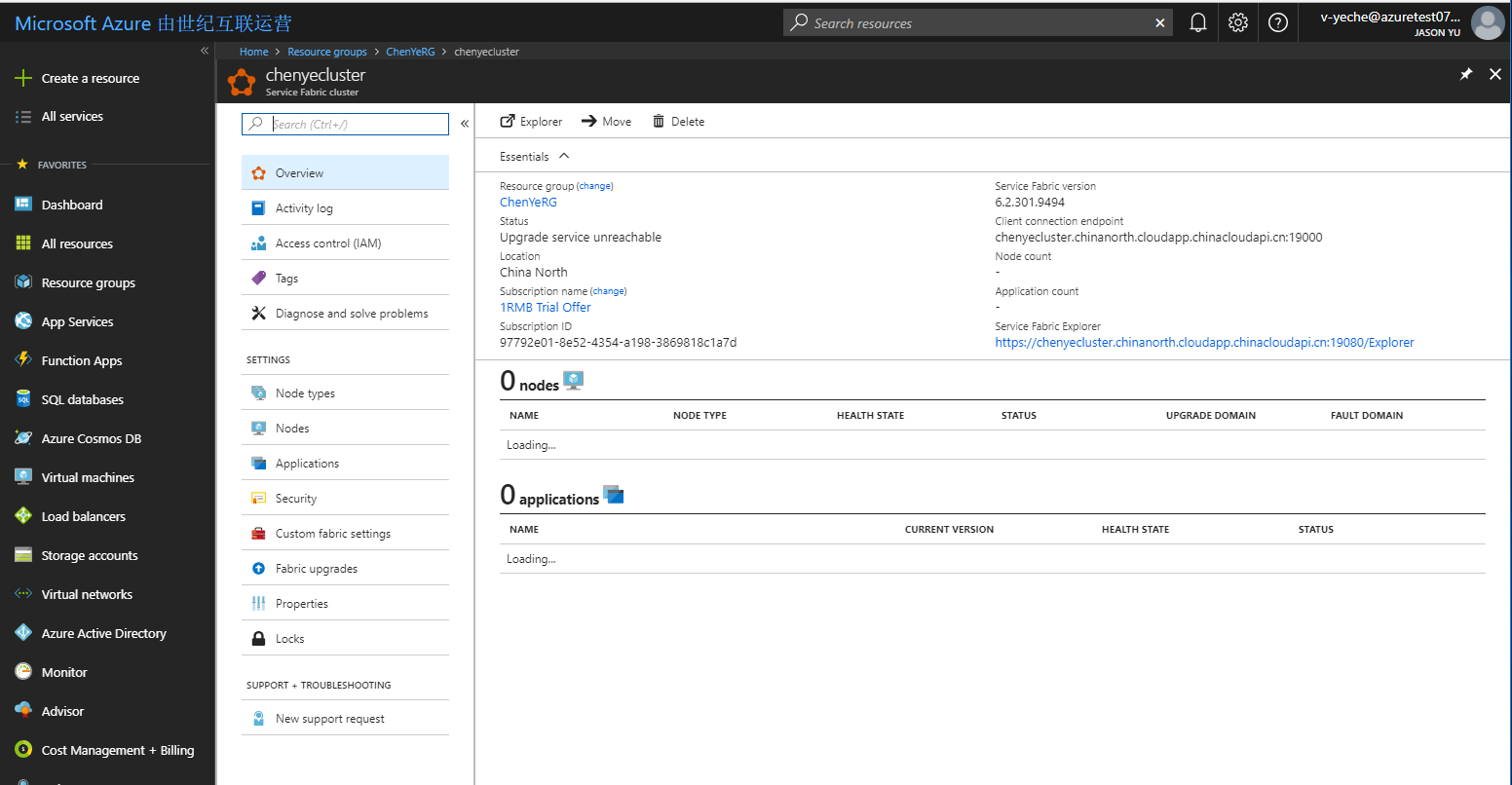




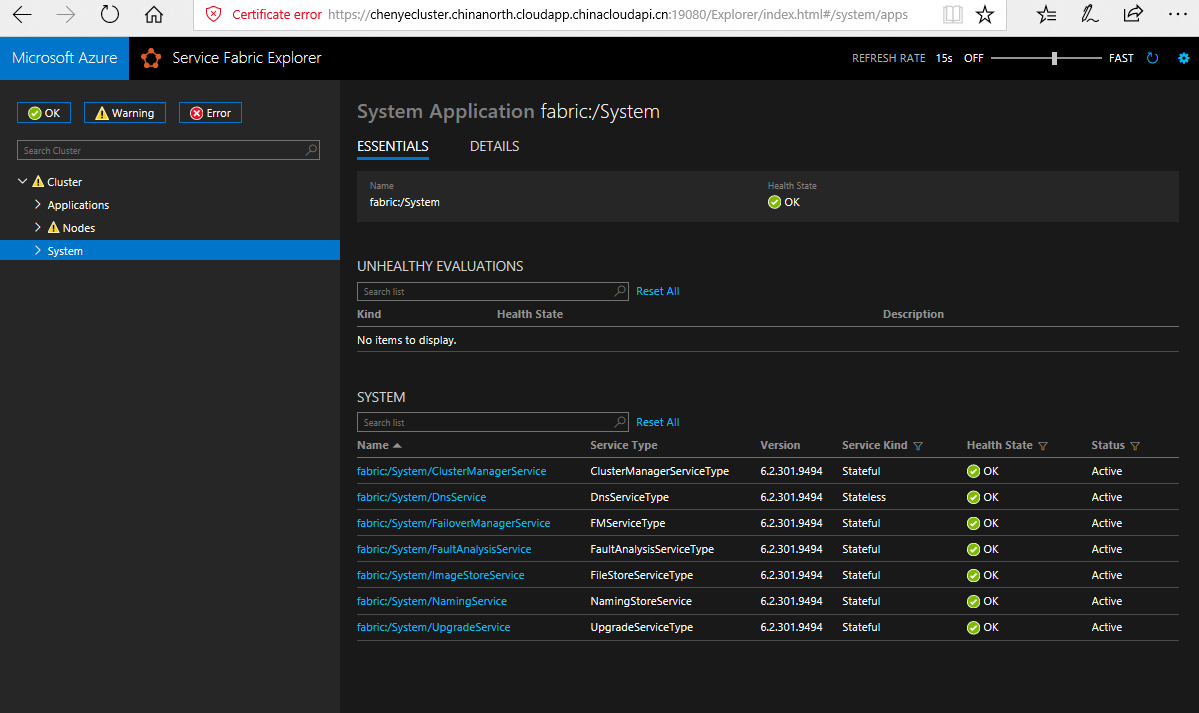
When we do testing, we can use Unsecure condition.







When connect the service fabric explorer, we must connect MSFTVPN and select the corresponding certification.

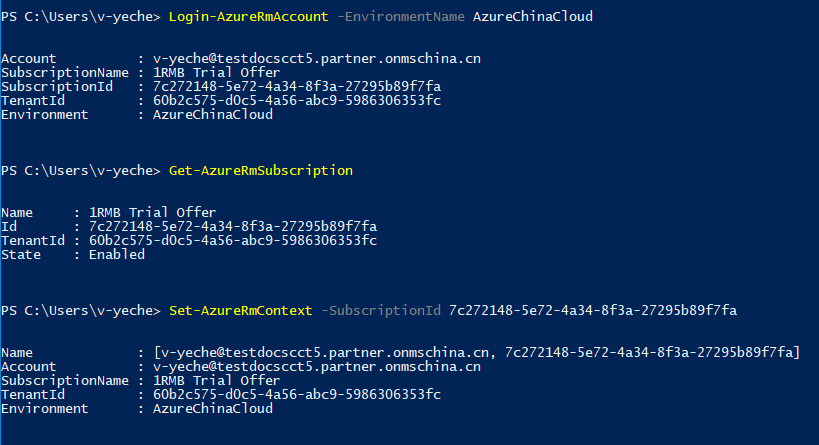


How to set the security service fabric:

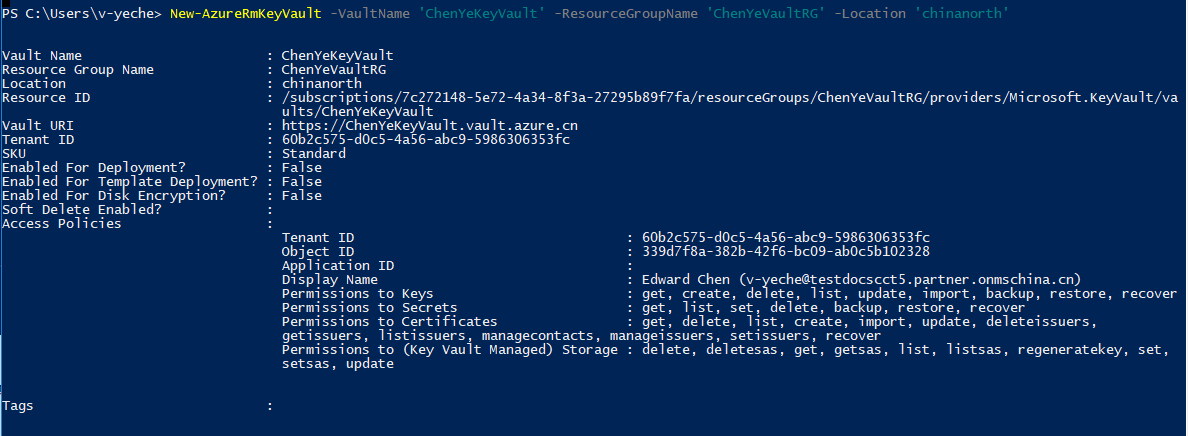
1. **Login-AzureRmAccount -EnvironmentName AzureChinaCloud**
2. **Get-AzureRmSubscription**

When user own multiple subscriptions, and use the specific subscriptions to create Vault.

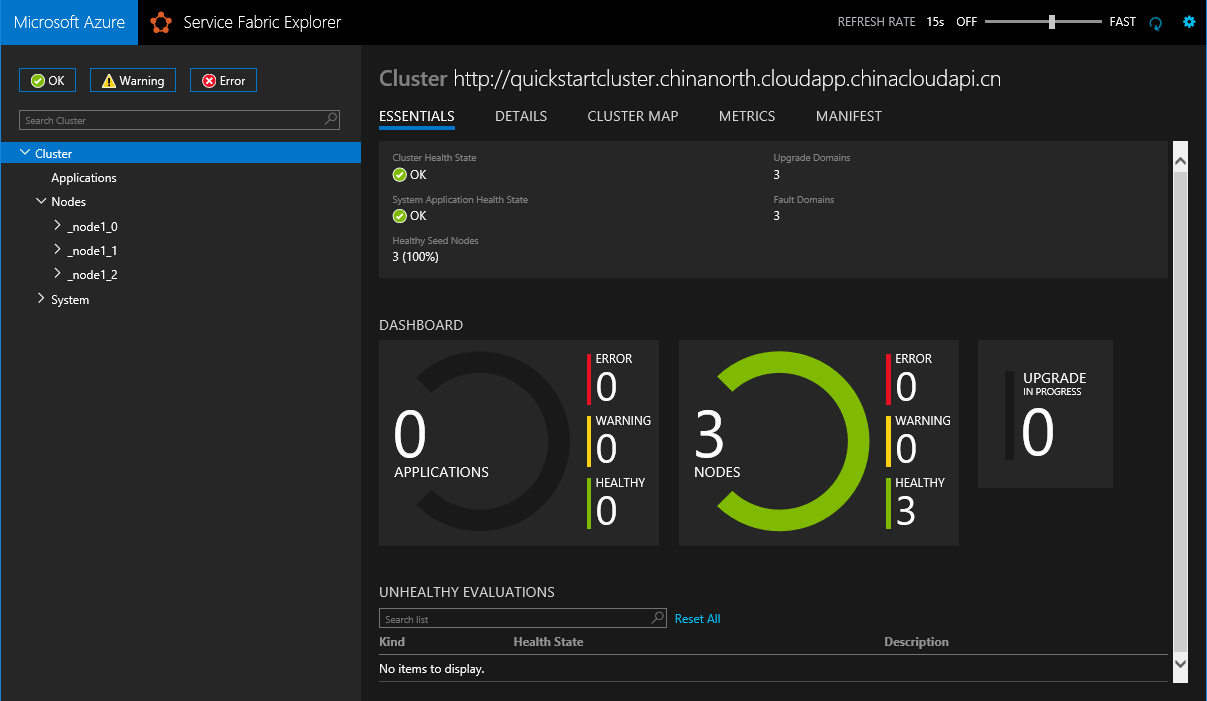
1. **Set-AzureRmContext -SubscriptionId 7c272148-5e72-4a34-8f3a-27295b89f7fa**



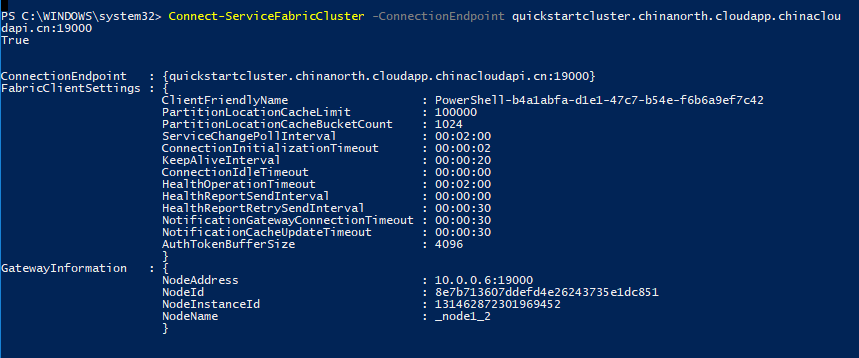
1. **New-AzureRmKeyVault -VaultName 'ChenYeKeyVault' -ResourceGroupName 'ChenYeVaultRG' -Location 'chinanorth'**

****

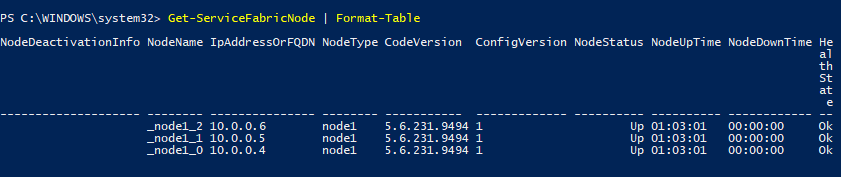
<http://quickstartcluster.chinanorth.cloudapp.chinacloudapi.cn:19080/Explorer/>



PS C:\WINDOWS\system32> **Connect-ServiceFabricCluster -ConnectionEndpoint quickstartcluster.chinanorth.cloudapp.chinacloudapi.cn:19000**

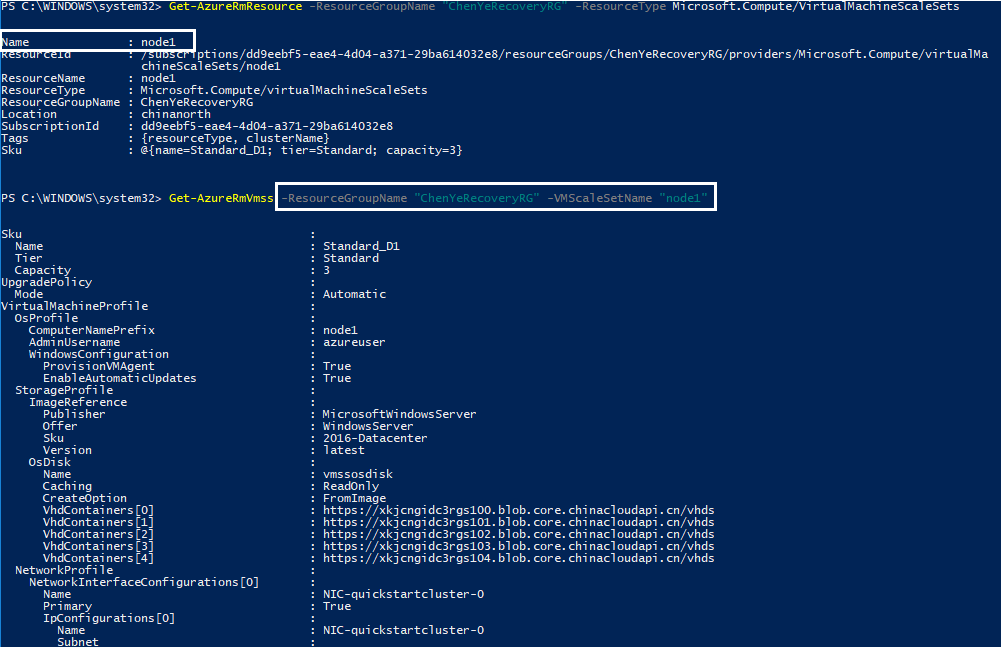


Get-ServiceFabricNode | Format-Table

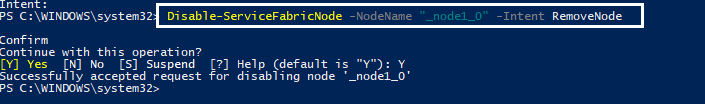


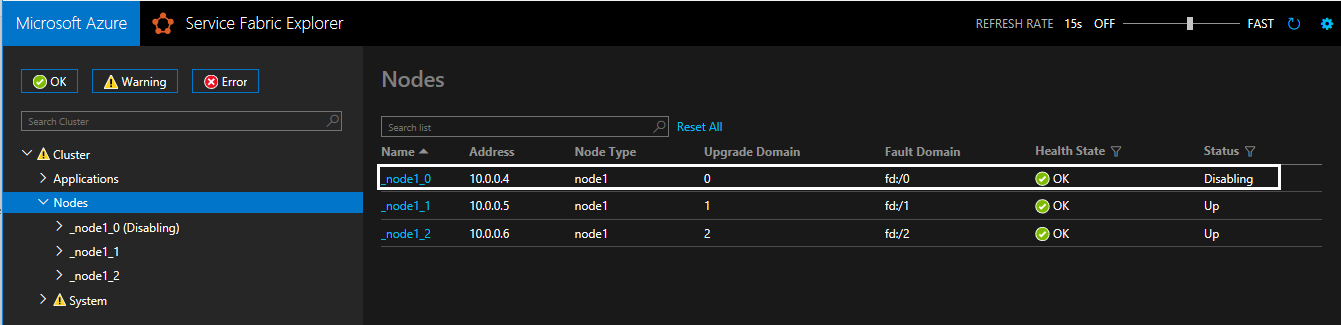
**Get-AzureRmResource -ResourceGroupName "ChenYeRecoveryRG" -ResourceType Microsoft.Compute/VirtualMachineScaleSets**

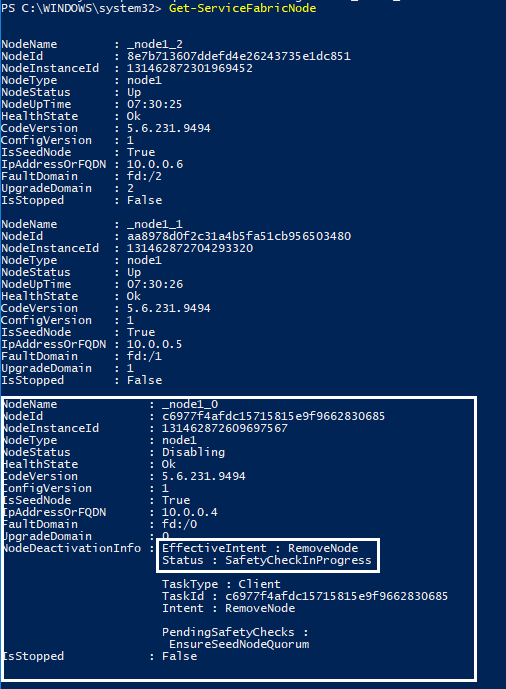
**Get-AzureRmVmss -ResourceGroupName "ChenYeRecoveryRG" -VMScaleSetName "node1"**



**Disable-ServiceFabricNode -NodeName "\_node1\_0" -Intent RemoveNode**





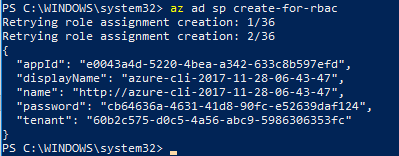


Expand Cluster programmer

az colud -set AzureChinaCloud

az login

az ad sp create-for-rbac



{

"appId": "e0043a4d-5220-4bea-a342-633c8b597efd",

"displayName": "azure-cli-2017-11-28-06-43-47",

"name": "http://azure-cli-2017-11-28-06-43-47",

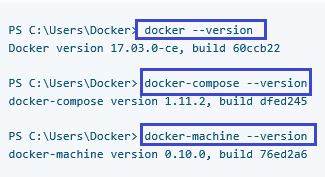
"password": "cb64636a-4631-41d8-90fc-e52639daf124",

"tenant": "60b2c575-d0c5-4a56-abc9-5986306353fc"

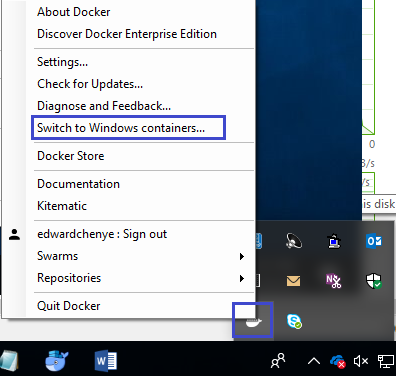
}

**How to run service fabric linux cluster in windows server.**

1. Install the latest Docker application:
2. Check the Docker install successful on not



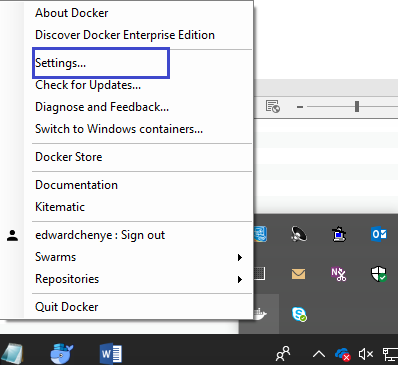
1. Start the Docker and switch to Linux Container.

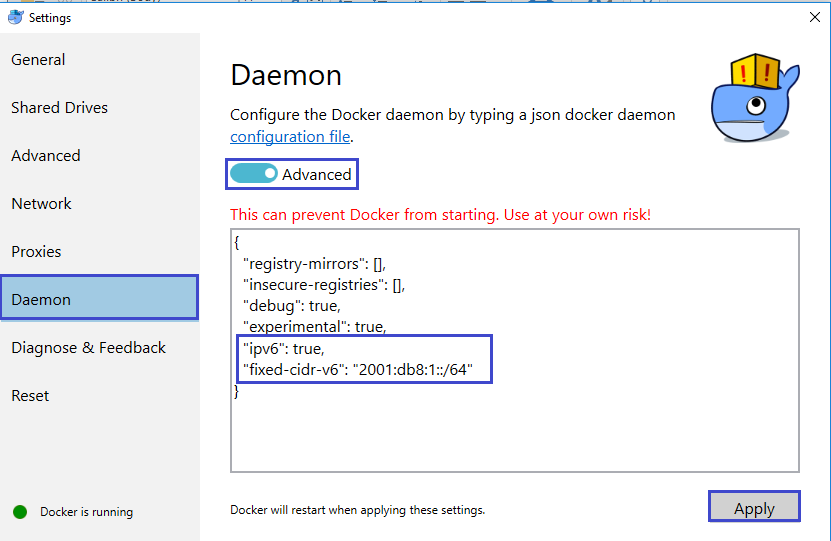


1. Start the PS with administrator , then run the command

docker pull servicefabricoss/service-fabric-onebox

1. Update the setting of Daemon in the setting.





1. Start the Service Fabric One-box container instance with image:

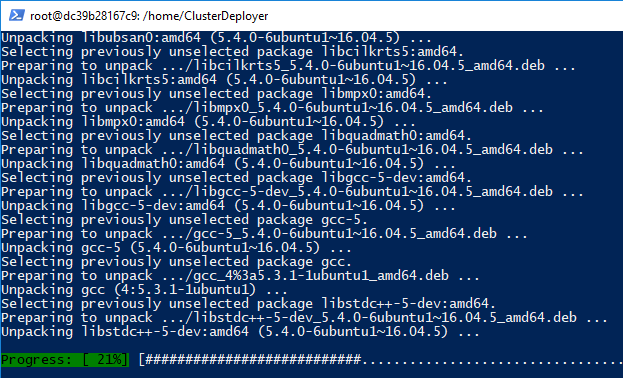
docker run -itd -p 19080:19080 --name sfonebox servicefabricoss/service-fabric-onebox

1. Log in to the Docker container in interactive ssh mode:

docker exec -it sfonebox bash

1. Run the setup script, that will fetch the required dependencies and after that start the cluster on the container

./setup.sh # Fetches and installs the dependencies required for Service Fabric to run

./run.sh # Starts the local cluster

How to prepare the development environment in Linux server

1. Install the Ubuntu 16.04
2. Install the service fabric script

sudo curl -s https://raw.githubusercontent.com/Azure/service-fabric-scripts-and-templates/master/scripts/SetupServiceFabric/SetupServiceFabric.sh | sudo bash

Or install the environment with manual like following:

1. 将 Service Fabric 存储库添加到源列表。

sudo sh -c 'echo "deb [arch=amd64] http://apt-mo.trafficmanager.net/repos/servicefabric/ xenial main" > /etc/apt/sources.list.d/servicefabric.list'

1. 将 dotnet 存储库添加到源列表。

sudo sh -c 'echo "deb [arch=amd64] https://apt-mo.trafficmanager.net/repos/dotnet-release/ xenial main" > /etc/apt/sources.list.d/dotnetdev.list'

1. 向 APT Keyring 添加新的 Gnu 隐私防护（GnuPG 或 GPG）密钥。

sudo apt-key adv --keyserver apt-mo.trafficmanager.net --recv-keys 417A0893

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 417A0893

1. 向 APT Keyring 添加官方的 Docker GPG 密钥。

sudo apt-get install curl

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

1. 设置 Docker 存储库。

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

1. 根据新添加的存储库刷新包列表。

sudo apt-get update

为本地群集设置安装并设置 Service Fabric SDK

1. 源进行更新后，即可安装 SDK。After you have updated your sources, you can install the SDK. 安装 Service Fabric SDK 包，确认安装，并同意许可协议。

sudo apt-get install servicefabricsdkcommon

1. 以下命令自动接受 Service Fabric 包的许可证：

echo "servicefabric servicefabric/accepted-eula-ga select true" | sudo debconf-set-selections

echo "servicefabricsdkcommon servicefabricsdkcommon/accepted-eula-ga select true" | sudo debconf-set-selections

设置本地群集

1. 运行群集安装程序脚本。

sudo /opt/microsoft/sdk/servicefabric/common/clustersetup/devclustersetup.sh

1. 打开 Web 浏览器，转到 [Service Fabric Explorer](http://localhost:19080/Explorer)。Open a web browser and go to [Service Fabric Explorer](http://localhost:19080/Explorer). 如果群集已开始，应看到 Service Fabric Explorer 仪表板。

为容器和来宾可执行文件安装 Yeoman 生成器

1. 在计算机上安装 nodejs 和 NPM

sudo apt-get install npm

sudo apt install nodejs-legacy

1. 通过 NPM 在计算机上安装 [Yeoman](http://yeoman.io/) 模板生成器

sudo npm install -g yo

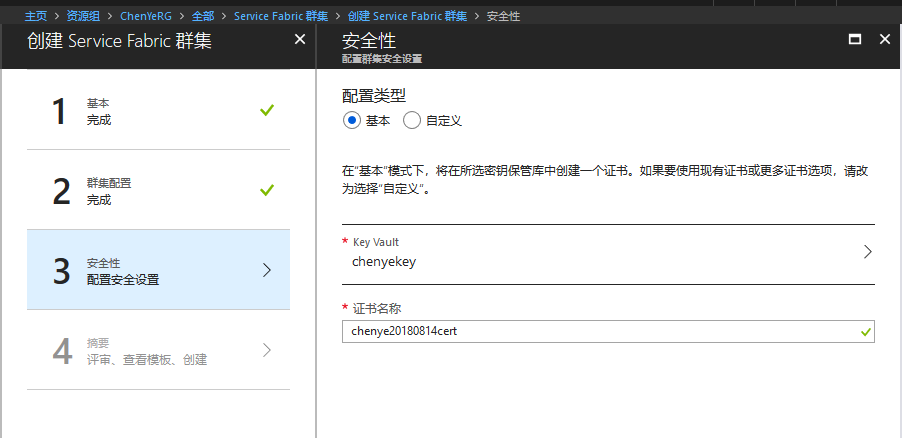
1. 通过 NPM 安装 Service Fabric Yeo 容器生成器和来宾可执行文件生成器

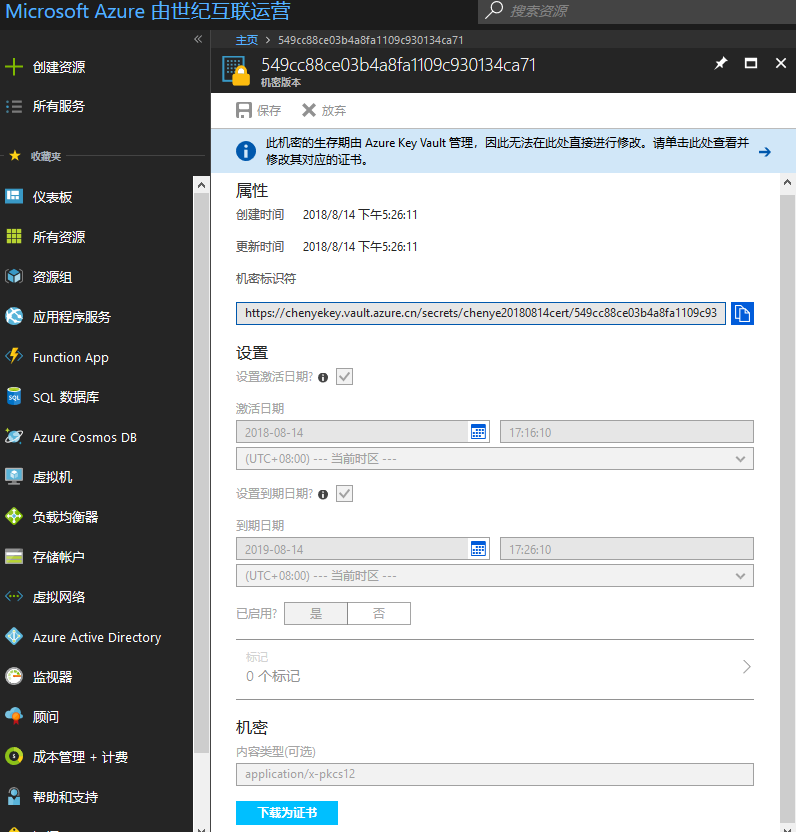
sudo npm install -g generator-azuresfcontainer # for Service Fabric container application

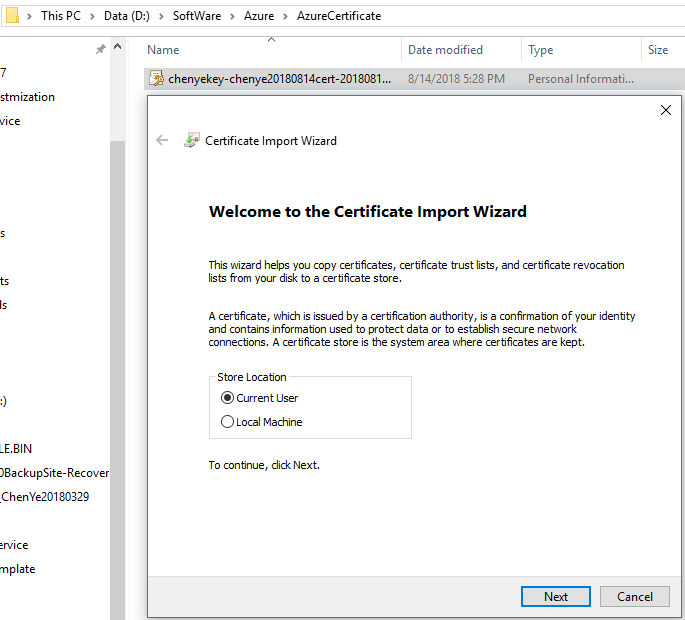
sudo npm install -g generator-azuresfguest # for Service Fabric guest executable application

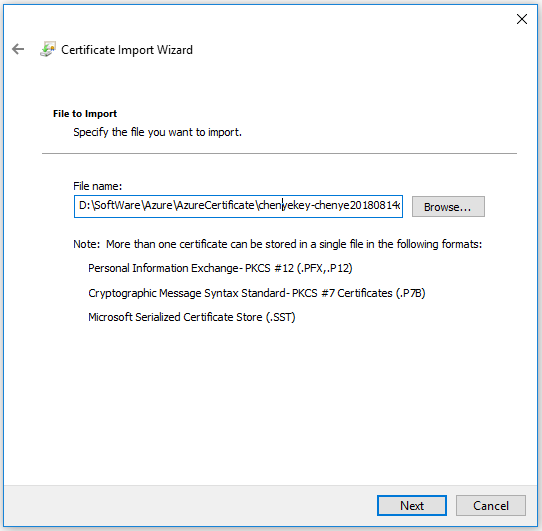
1. 安装生成器后，应该可以分别运行 yo azuresfguest 或 yo azuresfcontainer，创建来宾可执行文件或容器服务。

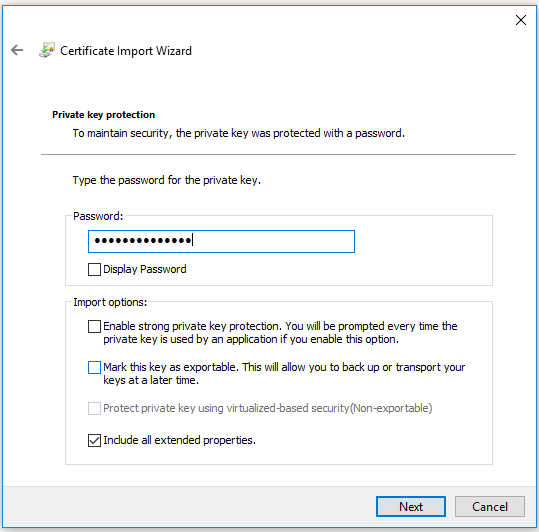
Service Fabric explorer with certificate:



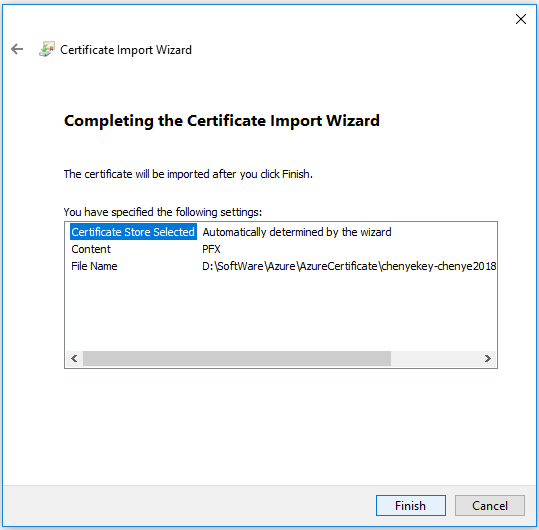
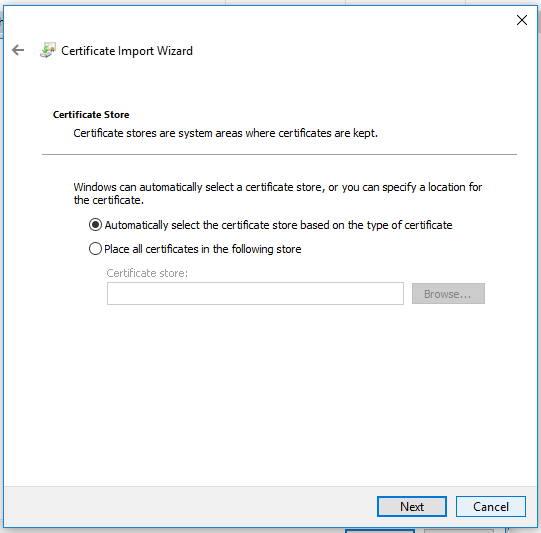




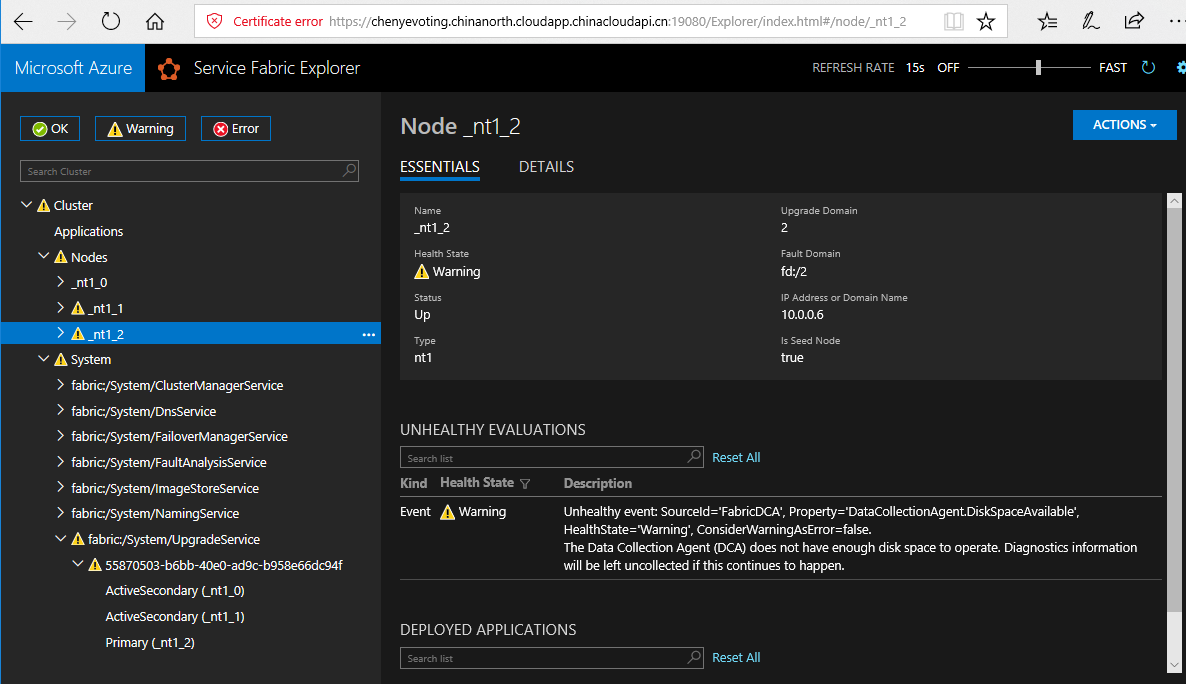


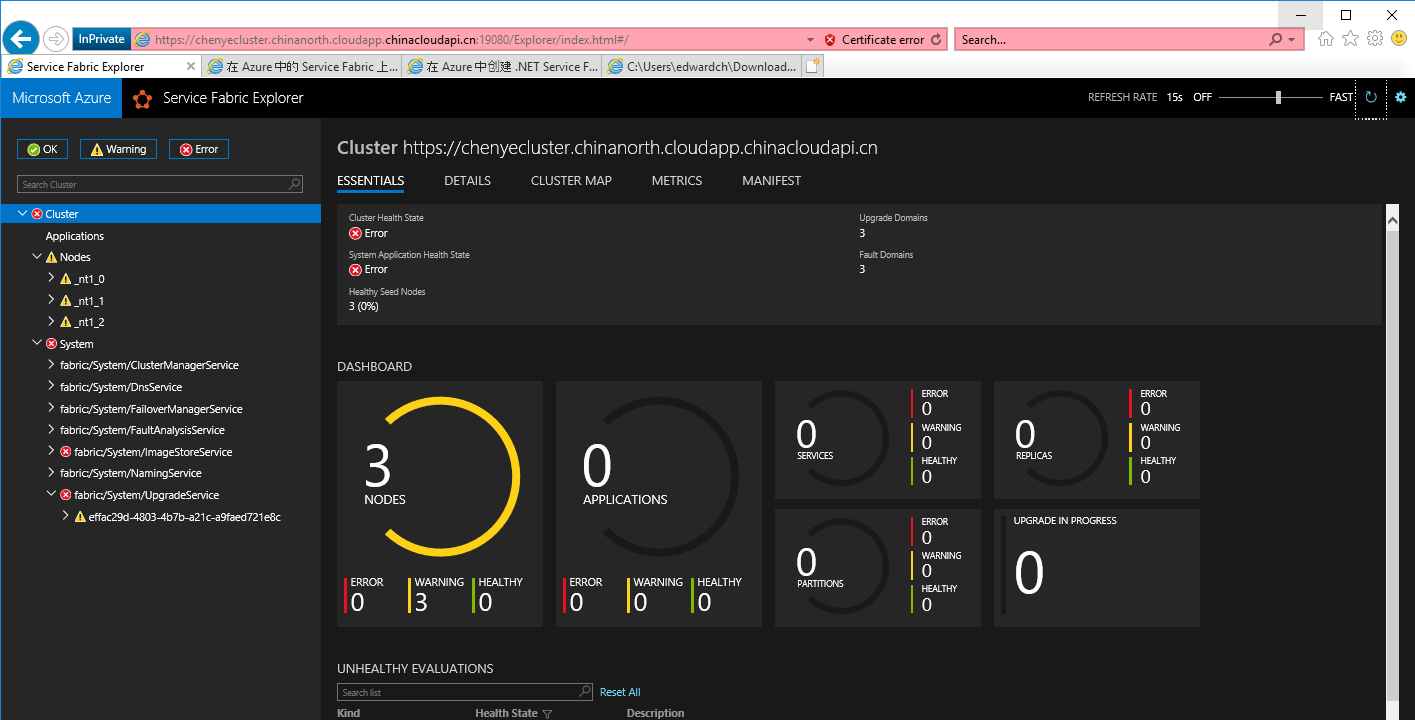


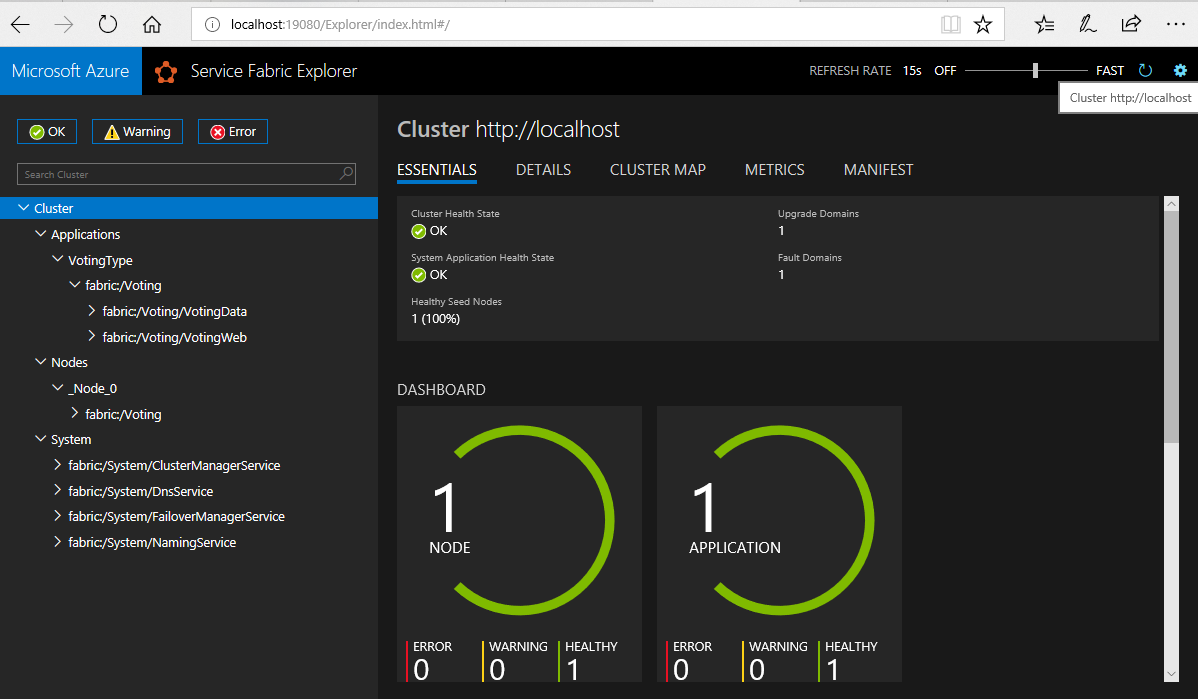
Notice: The Password is not verified, we will keep it blank.

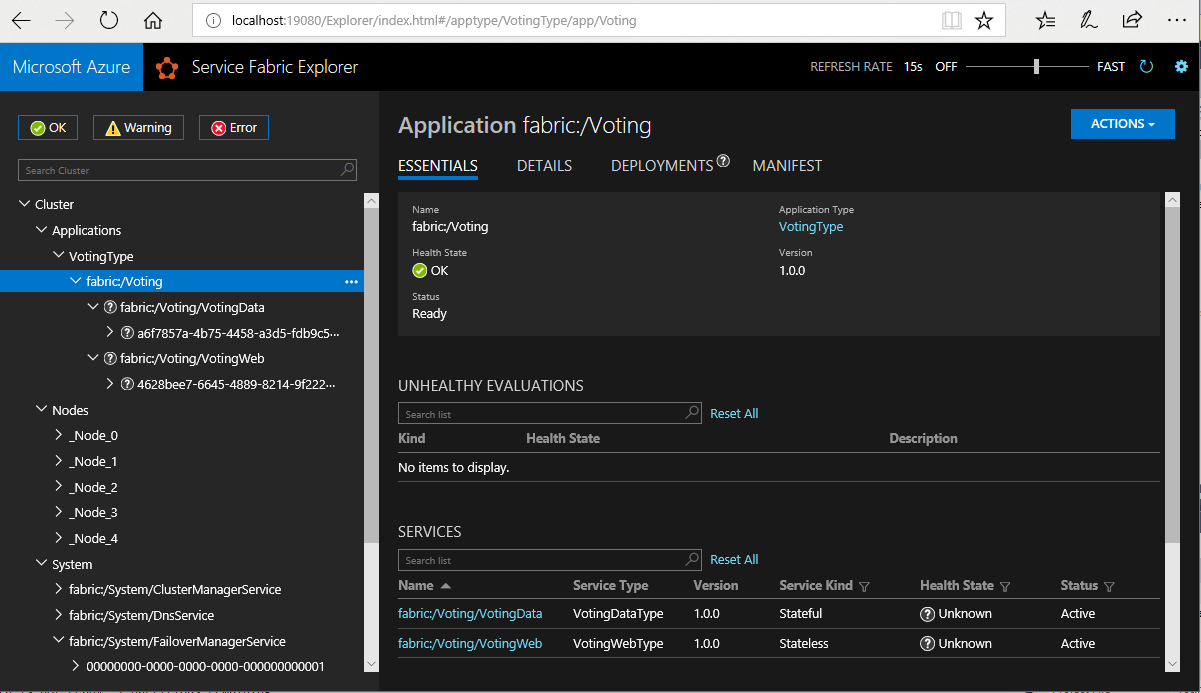




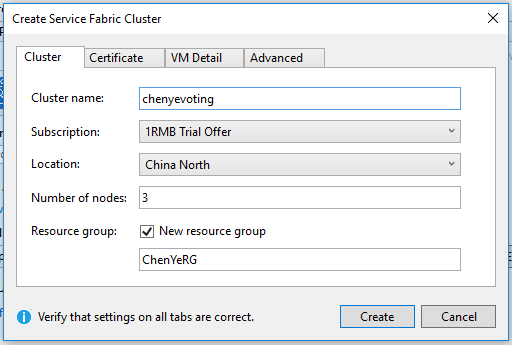


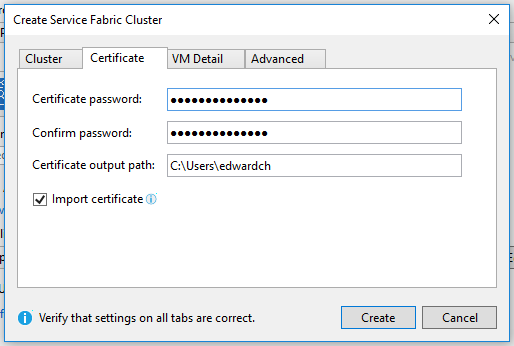


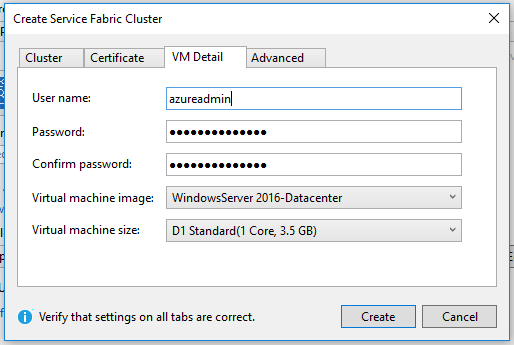


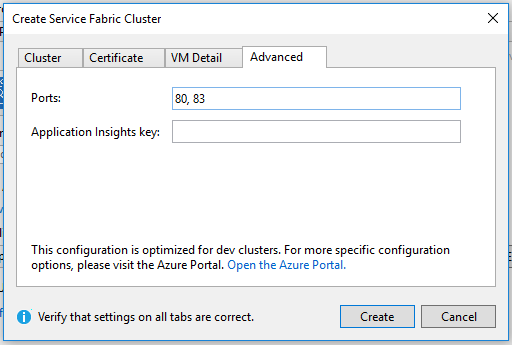


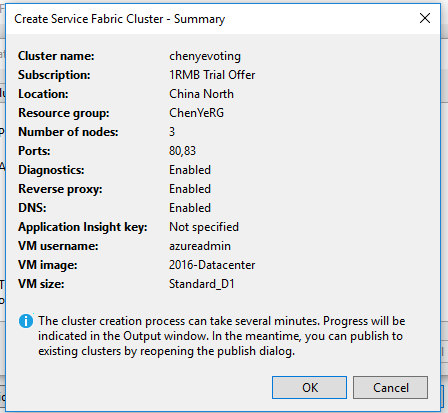
Create Azure Service Fabric cluster via Visual Studio 2017







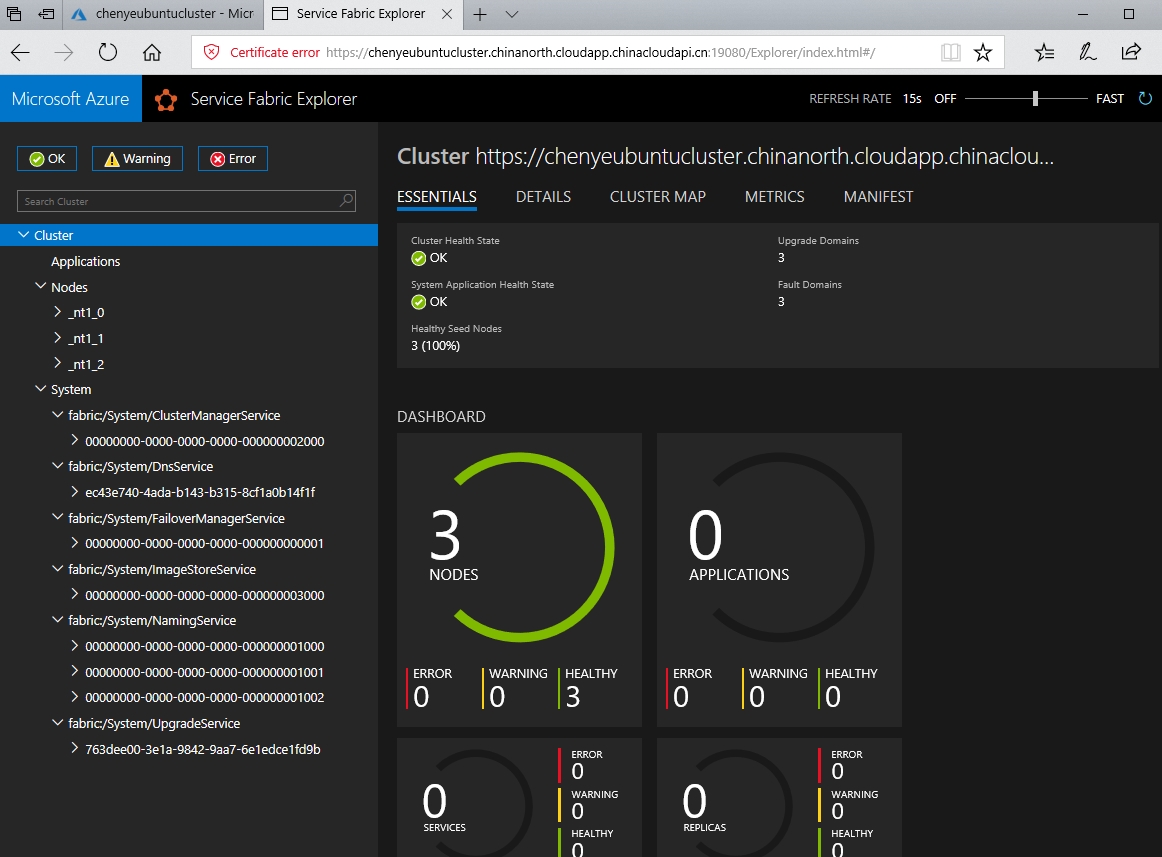




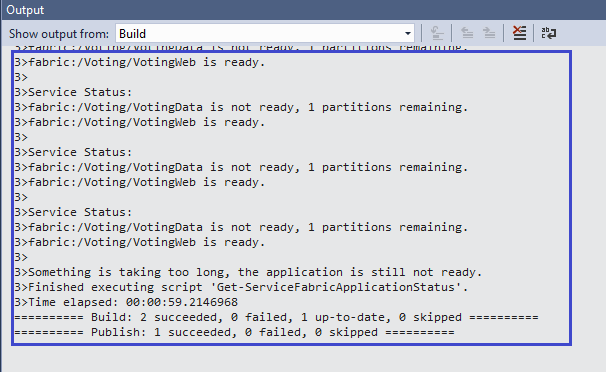
Create the Linux Cluster :

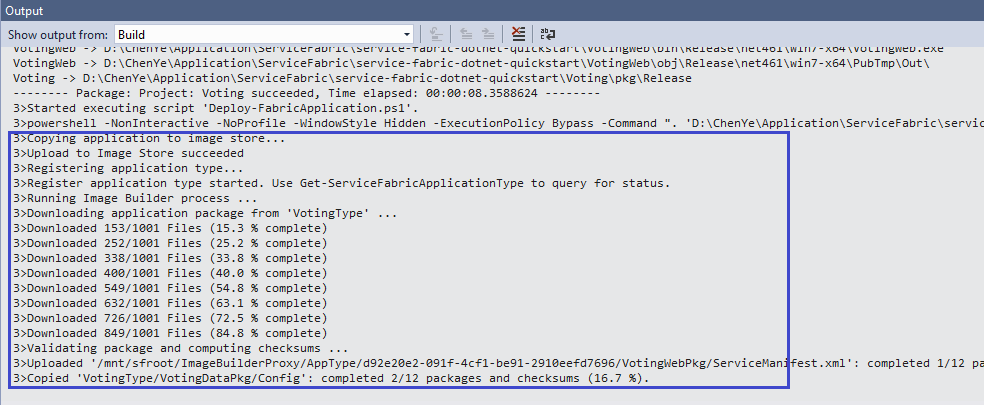
1. First download the \*.cert file and install the certificate on your local PC.
2. Connect the Explorer of cluster.

Notice: It will take less more time when installing the Linux cluster more than Windows cluster.

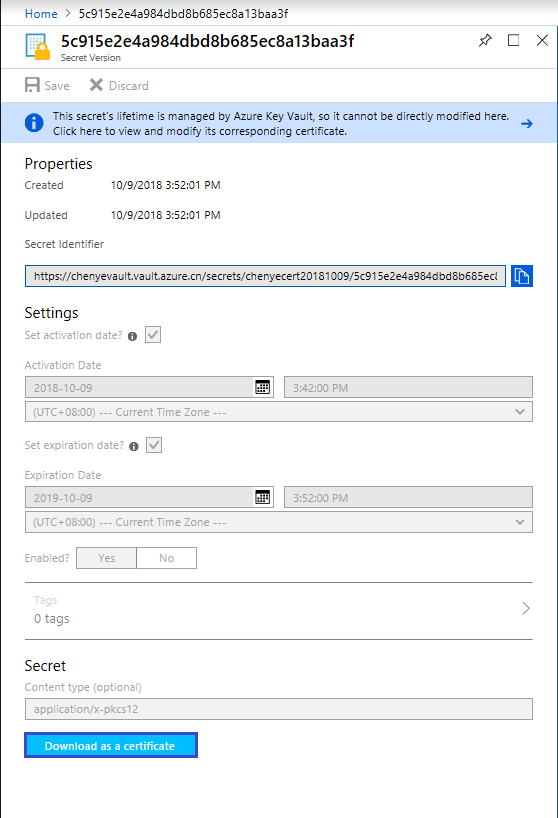


1. How to public the application on Linux cluster.





Download the certification from the portal



PowerShell commandlet.

Create Windows Cluster which have been verified successfully!

$certpwd="Ryan=Lindsayc" | ConvertTo-SecureString -AsPlainText -Force

$certfolder="C:\mycertificates"

$clustername = "mychenyecluster"

$vaultname = "chenyevault"

$vaultgroupname="chenyevaultrg"

$subname="$clustername.$clusterloc.cloudapp.chinacloudapi.cn"

# sign in to your Azure account and select your subscription

Connect-AzureRmAccount -Environment AzureChinaCloud

Get-AzureRmSubscription

Set-AzureRmContext -SubscriptionId 1cb45922-70a7-42a1-8d2f-781244c25530

# Create a new resource group for your deployment and give it a name and a location.

New-AzureRmResourceGroup -Name $groupname -Location $clusterloc

# Create the Service Fabric cluster.

New-AzureRmServiceFabricCluster -ResourceGroupName $groupname -TemplateFile "$templatepath\AzureDeploy\_Windows.json" `

-ParameterFile "$templatepath\AzureDeploy\_Windows.parameters.json" -CertificatePassword $certpwd `

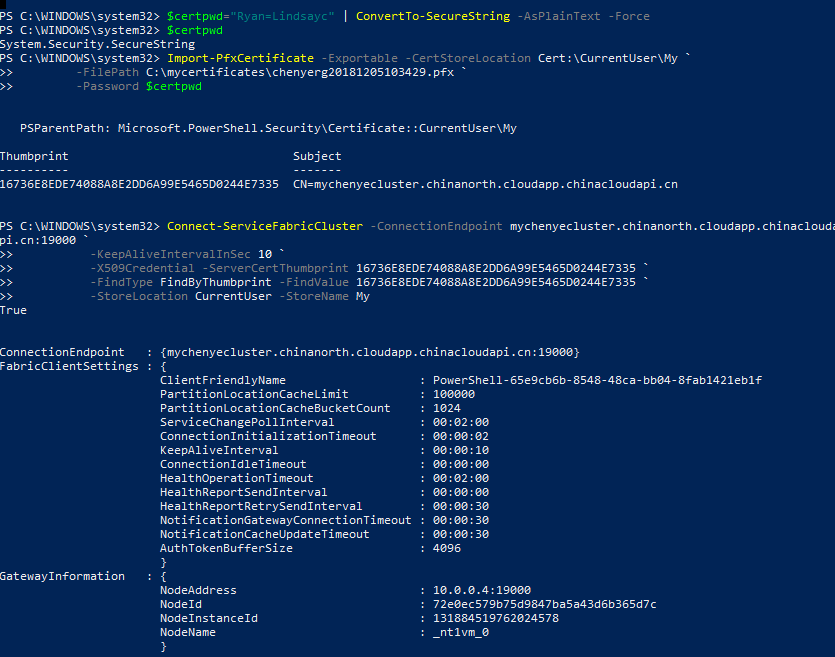
-CertificateOutputFolder $certfolder -KeyVaultName $vaultname -KeyVaultResouceGroupName $vaultgroupname -CertificateSubjectName $subname

$certpwd="Ryan=Lindsayc" | ConvertTo-SecureString -AsPlainText -Force

Import-PfxCertificate -Exportable -CertStoreLocation Cert:\CurrentUser\My `

-FilePath C:\mycertificates\chenyerg20181205103429.pfx `

-Password $certpwd



Connect-ServiceFabricCluster -ConnectionEndpoint mysfcluster123.southcentralus.cloudapp.azure.com:19000 `

-KeepAliveIntervalInSec 10 `

-X509Credential -ServerCertThumbprint C4C1E541AD512B8065280292A8BA6079C3F26F10 `

-FindType FindByThumbprint -FindValue C4C1E541AD512B8065280292A8BA6079C3F26F10 `

-StoreLocation CurrentUser -StoreName My

