Top of Form

[Logged in as JaswantS](https://nuget.nof1health.com:8443/user/profile) - [Logout](https://nuget.nof1health.com:8443/user/logout)

* [Main Page](https://nuget.nof1health.com:8443/)
* [Categories](https://nuget.nof1health.com:8443/pages/alltags)
* [All pages](https://nuget.nof1health.com:8443/pages/allpages)
* [New page](https://nuget.nof1health.com:8443/pages/new)
* [Manage files](https://nuget.nof1health.com:8443/filemanager)



Bottom of Form

Top of Form

Bottom of Form

N of 1 v2 Deployment Guide - v2.1.0

Deployment Guide v2.1.0

Some of the commands, file names, and other content in this guide contain variables that you must change for your deployment. These variables are indicated by {CurlyBraces}.

0. Initial steps

These steps only need to be performed once. The output of these steps is used in all of the subsequent steps.

**Example**

This guide uses an "example deployment" to illustrate commands and values. Examples will be shown in a box like this one.

0.1. Azure prerequisites

You need to know the subscription that you will be deploying to. This is referred to as {SubscriptionName} in this guide.

The account that is logged in to run the deployment needs to have Contributor access level to this subscription.

0.2. Software prerequisites

You must install PowerShell version 5.x.

You must install the Azure PowerShell commands, version 6.9.0 or later. Instructions for how to install these commands are here:

<https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-6.9.0>

You must install the Azure CLI. Instructions for how to install this are here:

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>

You must install the Azure Powershell Table Storage Management commands. To install this module, run PowerShell as an administrator and execute this command:

Install-Module AzureRmStorageTable

0.3. Establish conventions

N of 1 Azure resources are named using some conventions. These conventions are based on several important values, which need to be established before a deployment.

**Once these values are set, they can't be changed.**

1. Customer Identifier
   * The customer identifier is a short string (8 characters max) that uniquely identifies the customer.
   * The customer identifier should be all lower-case.
   * Example customer identifiers are nof1-v210, ifopa, and nxt.
2. Environment Name
   * The environment name is a short string (4 characters max) that identifies the environment that is being deployed.
   * The environment name should be all lower-case.
   * Example environment names are dev, qa, and prod.
3. Location
   * The location sets the Azure location where the environment will be deployed.
   * The default location is eastus2.
   * To see a complete list of locations, execute the PowerShell command Get-AzureRMLocation. Use the Location property value from the results to populate this parameter.

**Example**

The example deployment has the following values:

* Customer Identifier: example
* Environment Name: dev

0.4. Create parameter file

The EnvironmentSetup directory contains many template files, named using this convention: {ResourceType}-template.json.

Each template has a set of required parameters.

You will need to create a parameter file to supply values for each of these parameters.

The parameter file can be named however you would like. The standard convention is: {CustomerIdentifier}-{EnvironmentName}-parameters.json. However, feel free to adopt a different naming convention if you would like.

The initial content of the parameter file is in the file parameters-template.json. Copy this file and rename it to reflect the environment you are creating.

Fill in the customerIdentifier parameter value with your customer identifier. Fill in the environment parameter value with your environment name. Fill the sponsorName parameter value with human-readable text describing the sponsor.

**Example**

Using the example deployment's parameters (described above), the parameters file for this template would be named example-dev-parameters.json.

The contents of the parameter file are:

{

"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",

"contentVersion": "1.0.0.0",

"parameters": {

"customerIdentifier": { "value": "example" },

"environment": { "value": "dev" }

...other blank properties...

}

}

To change the Azure Location that resources are deployed to, add a location parameter in the parameter file. Otherwise, the default location is "East US 2".

0.5. Prepare to deploy

Open a PowerShell prompt.

Log in to your Azure account, and set the subscription that you are deploying to.

Connect-AzureRmAccount

Set-AzureRmContext -SubscriptionName "{SubscriptionName}"

pushd GenesisNof1\Nof1HealthPlatform\EnvironmentSetup

1. Create the resource groups

In the PowerShell window, execute the following command:

.\Setup-Resource-Group.ps1 -TemplateParameterFile {TemplateParameterFileName}

**Example**

For the example deployment, the command is:

.\Setup-Resource-Group.ps1 -TemplateParameterFile .\example-dev-parameters.json

2. Create the B2C tenant

Unfortunately, this step cannot yet be automated.

In the Azure Portal, navigate to the resource group that you created in step 1. In that resource group, click the "Add" button at the top of the blade. Search for Azure Active Directory B2C. Once you select Azure Active Directory B2C, click the "Create" button at the bottom of the blade.

When you create a directory, you are prompted with two options in the "Create new B2C Tenant or Link to existing Tenant" blade:

* Create a new Azure AD B2C Tenant
* Link an existing Azure AD B2C Tenant to my Azure subscription

Choose the first option.

Set the Organization Name using the following convention:

{CustomerIdentifier}-b2c-{EnvironmentName}

Set the Initial Domain Name to the same value as the Organization Name, without the dashes. Then click "Create". Do not close the "Create new B2C Tenant or Link to existing Tenant" blade.

*Add the Organization Name to your parameters file, as b2cTenantName.value.*

**Example**

For the example deployment, the Organization Name is example-b2c-dev. The Initial Domain Name is exampleb2cdev.

You will receive a notification in the Azure Portal when the tenant has been created. Once you receive this notification, on the "Create new B2C Tenant or Link to existing Tenant" blade, click "Link an existing Azure AD B2C Tenant to my Azure subscription". Select the directory you just created in the "Azure AD B2C Tenant" dropdown. Select the subscription that you chose in step 0.5, then select the resource group you created in step 1. Click "Create". You will receive a notification in the Azure portal when the B2C directory has been added to the resource group.

2.1. Create a B2C application for the portal

1. Go into the Azure AD B2C Settings page for the tenant.
2. On the Settings blade, click Applications.
3. Click +Add at the top of the blade.
4. Enter nof1-portal in the *Name* field.
5. Toggle the *Include Web App/Web API* switch to Yes.
6. Set the Reply URLs.
   * https://{PortalDomainName}/OnAzureAdLogin.aspx
   * https://{PortalDomainName}/OnAzureAdSignUp.aspx
   * https://{PortalDomainName}/OnAzureAdESignature.aspx
7. Set the *App ID URI* as nof1-portal.
8. Click Create to register your application.
9. In the B2C tenant Applications list, click the nof1-portal application.
10. Copy the Application ID and add it to your parameters file, as b2cApplicationId.value.
11. In the blade settings menu on the left, click *Keys*. In the *Keys* blade, click *Generate Key*, then click *Save* to see the key. Copy the key and add it to your parameters file, as b2cApplicationSecret.value.
12. In the blade settings menu, click *Published scopes*. Verify that there is a user\_impersonation scope entry. If not, create a scope entry with the name user\_impersonation and the value https://{CustomerIdentifier}b2c{EnvironmentName}.onmicrosoft.com/nof1-portal/user\_impersonation.
13. In the blade settings menu, click *API access*. In the *API Access* blade, click *Add*. For *Select API*, select nof1-portal. For *Select scopes*, check "Select All". Then click *OK*.

14.

2.2 Create B2C Policies

* Sign-Up Policy
  + Name: signup
  + Identity providers: Email signup
  + Sign-up attributes: Email address
  + Application claims:
    - Email Addresses
    - Identity Provider
    - User is New
    - User's Object ID
  + At this point, click "Create." After the policy has been created, click it, then click "Edit".
  + Token, session & SSO config
    - **Token lifetimes**
      * Access & ID token lifetimes (minutes): 60
      * Refresh token lifetime (days): 1
      * Refresh token sliding window lifetime (days): Bounded, 7
    - **Token compatibility settings**
      * Issuer (iss) claim: https://<domain>/tfp/....
      * Subject (sub) claim: ObjectID
      * Claim representing policy ID: tfp
    - **Session Behavior**
      * Web app session lifetime (minutes): 1440
      * Web app session timeout: Absolute
      * Single sign-on configuration: Tenant
      * Require ID token in logout requests: No
  + Password complexity: Strong
  + Click "OK", then "Save".
* Sign-In Policy
  + Name: signin
  + Identity providers: Local Account Signin
  + Application claims:
    - Email Addresses
    - Identity Provider
    - User's Object ID
  + At this point, click "Create." After the policy has been created, click it, then click "Edit".
  + Token, session & SSO config
    - **Token lifetimes**
      * Access & ID token lifetimes (minutes): 60
      * Refresh token lifetime (days): 1
      * Refresh token sliding window lifetime (days): Bounded, 7
    - **Token compatibility settings**
      * Issuer (iss) claim: https://<domain>/tfp/....
      * Subject (sub) claim: ObjectID
      * Claim representing policy ID: tfp
    - **Session Behavior**
      * Web app session lifetime (minutes): 1440
      * Web app session timeout: Absolute
      * Single sign-on configuration: Tenant
      * Require ID token in logout requests: No
  + Click "OK", then "Save".
* Sign-In Policy for E-Signature
  + Name: esignature
  + Identity providers: Local Account Signin
  + Application claims:
    - Email Addresses
    - Identity Provider
    - User's Object ID
  + At this point, click "Create." After the policy has been created, click it, then click "Edit".
  + Token, session & SSO config
    - **Token lifetimes**
      * Access & ID token lifetimes (minutes): 5
      * Refresh token lifetime (days): 1
      * Refresh token sliding window lifetime (days): Bounded, 1
    - **Token compatibility settings**
      * Issuer (iss) claim: https://<domain>/tfp/....
      * Subject (sub) claim: ObjectID
      * Claim representing policy ID: tfp
    - **Session Behavior**
      * Web app session lifetime (minutes): 15
      * Web app session timeout: Absolute
      * Single sign-on configuration: Disabled
      * Require ID token in logout requests: No
  + Click "OK", then "Save".
* Password Reset Policy
  + Name: reset
  + Identity providers: Reset password using email address
  + Application claims:
    - Email Addresses
    - User's Object ID
  + At this point, click "Create." After the policy has been created, click it, then click "Edit".
  + Token config
    - **Token compatibility settings**
      * Issuer (iss) claim: https://<domain>/tfp/....
      * Subject (sub) claim: ObjectID
      * Claim representing policy ID: tfp
  + Password complexity: Strong

3. Create an Active Directory

In the Azure Portal, create an Active Directory for this *customer*.

**Note**

All of a customer's environments will share the same Active Directory. So the directory only needs to be created for the *first* customer environment. For subsequent environments, use the existing directory.

Click "Create a resource" in the left-hand menu. Search for "Azure Active Directory" and click "Create". The "Organization Name" is the name of the customer. This is *not* the name of the environment; this directory will serve all environments. The "Initial domain name" is the customer identifier (with any unallowed characters removed). Click "Create" and wait for the directory to be created. You will automatically be set as a Global Administrator for the new directory.

Once the directory has been created, switch to the new directory using the menu in the top right of the Azure portal. In the left-hand menu, click "Azure Active Directory". This should show the directory properties page. In the directory properties, click "Properties" in the left-hand menu. Copy the "Directory ID". This is the *Tenant ID* for this directory and is required to create the Key Vault.

Copy the tenant ID and add it to your parameters file, as activeDirectoryTenantId.value.

4. Deploy the Key Vault

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\keyvault-template.json -TemplateParameterFile {TemplateParameterFileName}

4.1. Create an Active Directory application

An Active Directory application is needed in order to authenticate to the Key Vault. You will need the Tenant ID for the Active Directory that you created above.

Set up PowerShell as per the steps in "Prepare to deploy".

This script requires the Azure CLI. Install the Azure CLI by following the instructions here: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>

In the PowerShell console, execute this command:

.\Setup-KeyVaultAccess.ps1 -TemplateParameterFile {TemplateParameterFile} -KeyVaultSubscription {SubscriptionName}

This script will prompt you to log in to Azure *twice* the first time you run it. This is the expected behavior.

This script generates a self-signed certificate to allow applications to authenticate to the KeyVault. The script automatically adds the certificate to the current user's local certificate store.

The script prints the certificate password and thumbprint to the console. It also populates the parameter file with the certificate password, thumbprint, and a base64-encoded version of the certificate content. These parameters are used to add the certificate to the Web API.

5. Deploy the Background App

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\background-app-template.json -TemplateParameterFile {TemplateParameterFileName}

Once the app has been created, you can modify its speed (and price) in the Azure portal.

6. Deploy the API App

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute these commands:

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\webapi-template.json -TemplateParameterFile {TemplateParameterFileName}

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\application-insights-template.json -TemplateParameterFile {TemplateParameterFileName}

Once the app has been created, you can modify its speed (and price) in the Azure portal.

7. Deploy the Storage Accounts

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute these commands:

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\storage-template.json -TemplateParameterFile {TemplateParameterFileName}

.\Setup-StorageAccount.ps1 -TemplateParameterFile {TemplateParameterFileName}

8. Deploy the Service Bus

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\servicebus-template.json -TemplateParameterFile {TemplateParameterFileName}

9. Deploy Cosmos DB

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\cosmosdb-template.json -TemplateParameterFile {TemplateParameterFileName}

This command creates the CosmosDB account.

Now, we need to create the CosmosDB collections. We have two standard collections:

* collection
* phi

You can create the collections on the Azure Portal, or through the Azure CLI.

Install the Azure CLI by following the instructions here: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest>

Once you've installed the Azure CLI, execute the PowerShell script to create the standard collections:

.\Setup-CosmosDb-Collections.ps1 -SubscriptionName "{SubscriptionName}" -TemplateParameterFile {TemplateParameterFileName}

This command will prompt you to log in.

10. Deploy the SQL Databases

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Setup-Sql-Database.ps1 -TemplateParameterFile {TemplateParameterFileName}

This script will create a username and password for the database, then populate the sqlAdministratorUsername and sqlAdministratorPassword parameters in the parameters file.

11. Set up Solr

Set up PowerShell as per the steps in "Prepare to deploy".

Obtain the host name, user name, and password for the Solr VM.

In the PowerShell console, execute this command:

.\Setup-Solr.ps1 -TemplateParameterFile {TemplateParameterFileName} -SolrVMHost {SolrVMHost} -SolrVMUsername {SolrVMUsername} -SolrVMPassword {SolrVMPassword}

This script will set up a Solr instance and deploy the standard configuration to that instance. The script will determine the port to use based on the existing instance(s).

This script will prompt you to log in to Azure again. This is because the Solr VM may be using the "Classic" deployment model, which does not share a login with the newer "Resource Manager" model. The script logs in using the "Classic" model because our existing Solr VM hosts (as of 2018) are using the Classic model. The script checks to see if the VM is using the Classic model, and if so, the script creates a network endpoint to allow Solr traffic for this instance through the firewall.

The endpoint creation only needs to be done once. If you need to re-run the script for any reason, the endpoint doesn't need to be created again, so you can avoid the login prompt by adding the switch -SkipEndpointCreation to the command.

12. Deploy the Portal Virtual Machine

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute these commands, in order:

.\Setup-PortalVM.ps1 -TemplateParameterFile {TemplateParameterFileName}

.\Setup-DNN-Database.ps1 -TemplateParameterFile {TemplateParameterFileName}

.\Setup-PortalVM-DNNConnectionString.ps1 -TemplateParameterFile {TemplateParameterFileName}

Ensure you have enabled IIS http to https redirect on portal VM. It automatically redirect http request to https response.

13. Import KeyVault Secrets

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Import-KeyVault-Secrets.ps1 -TemplateParameterFile {TemplateParameterFileName}

This script takes the parameterized values in secrets-template.json, fills the parameters with the values from the parameters file, and stores the secrets in the Key Vault.

To export KeyVault secrets, issue this PowerShell command:

$secrets = @{}; Get-AzureKeyVaultSecret -VaultName '{KeyVaultName}' | % { $secret = Get-AzureKeyVaultSecret -VaultName '{KeyVaultName}' -Name $\_.Name ; $secrets[$secret.Name] = $secret.SecretValueText }; $secrets | ConvertTo-Json | Out-File .\secrets.json -Force

This will write the secret values to a file named secrets.json. This JSON file can be the basis for a secrets template.

14. Import Configuration Settings

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute this command:

.\Import-ConfigurationTable-Settings.ps1 -TemplateParameterFile {TemplateParameterFileName}

15. Set up customer solution

First Time Setup

Before you can run the customer solution generator, there are some initial setup steps.

1. Open a command prompt in the directory GenesisNof1\Nof1HealthPlatform\generator-nof1-customer-project.
2. Execute this command: npm install
3. Once this command executes successfully, execute this command: npm link

These steps will set up the generator and make it available globally.

Create a directory for your customer solution. Open that directory in a PowerShell console.

Execute this command:

yo nof1-customer-project {TemplateParameterFileName}

In this command, TemplateParameterFileName must be the *full path* to the parameters file.

Answer the prompts from the yo command. This command will generate a customer-specific solution with the parameters from the parameter file.

16. Install the DNN Modules

Build the DNN portal modules.

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute these commands:

.\Install-Portal-Modules.ps1 -TemplateParameterFile {TemplateParameterFileName}

.\Finalize-Portal-VM-Configuration.ps1 -TemplateParameterFile {TemplateParameterFileName}

17. B2C function configuration for B2C user account display name and e-mail ID update

Set up PowerShell as per the steps in "Prepare to deploy".

In the PowerShell console, execute these commands:

.\B2CGraphAPIService-AppRegistration.ps1 -TemplateParameterFile {TemplateParameterFileName}

.\Add-Key-Keyvault-StorageTable.ps1 -templateParameterFile {TemplateParameterFileName} -subscription {StorageTableSubscriptionName}

.\Import-Cetificate-AppSetting.ps1 -templateParameterFile {TemplateParameterFileName} -certificateFilePath {CertificateFilePath}

B2CGraphAPIService-AppRegistration.ps1 script will create an Azure AD Application which will be used to update user diaplay name and email ID in B2C AD tenant. Script will populate the azureADB2CGraphAPIAppId and azureADB2CGraphAPIAppPassword parameters in the parameters file.

Add-Key-Keyvault-StorageTable.ps1 script will add B2C-Graph-API, Nof1SharedFunction-B2CGraphAPI-DisplayName-FunctionKey and Nof1SharedFunction-B2CGraphAPI-LoginEmail-FunctionKey keys in keyvault. This script also add SharedFunctionBaseUrl and ApplicationName keys in configuration table on Azure Storage.

.\Import-Cetificate-AppSetting.ps1 script will import .pfx certificate and update WEBSITE\_LOAD\_CERTIFICATES application setting on Nof1-SharedFunctions Azure function.

18. Manage wild card certificates on VM

Set up PowerShell as per the steps in "Prepare to deploy".

Import Certificate

Import new \*.nof1health.com certificate to keyvault.

you need to run this script only when you have new certificate to upload other wise you can skip this step. nof1-shared keyvault is common for all VMs to connect and upload \*.nof1health.com certificate. This script will upload certificate and you should use nof1-shared as common Keyvault for all VMs. Open powershell as admin and Go to following path in source code nof1HealthPlatform\EnvironmentSetup\Certificate.

In the PowerShell console, execute this command:

.\ImportCert.ps1 -CertificateName {Nof1Health-WildCard} -keyVaultName {name of keyVault} -CertificateFilename {physical path of certificate} -CertificatePassword { Certificate Password} -keyVaultAdminTenant {Tenant of vault}

Create Virtual Machine Managed Identity

Create a Managed Identity for VM. VM use this ID to access Certificate on Vault.

Open powershell as admin and Go to following path in source code nof1HealthPlatform\EnvironmentSetup\Certificate. In the PowerShell console, execute this command:

.\createManagedIdentity.ps1 -VMResourceGroup { VM Resource Group } -VMName { VM Name} -VmSubscriptionId { Vm Subscription Id } -VaultName { Vault Name } -VaultSubscriptionId { Vault Subscription Id } -VaultResourceGroupName { Vault Resource Group Name }

Deploy Key vault Extension on VM

Deploy Key vault Extension on VM using powershell.

Open powershell as admin and Go to following path in source code nof1HealthPlatform\EnvironmentSetup\Certificate. In the PowerShell console, execute this command:

.\VMExtension-sharedKV.ps1 -keyVaultName { Vault Name } -portalVmName { VM Name} -VMresourceGroupName { VM Resource Group } -VMLocation { VM Location } -CertificateName { CertificateName } -VmSubscriptionId { VM Subscription Id }

Binding IIS web site with certificate

Certificate will be deployed to VM using key vault extension. This step use VM extension to run powershell script on VM to bind web site with wild card certificate on port 443. You can upload ExtensionSetting.ps1 script on any blob storage and set anonymous read access for blob folder so that VM extension can read ExtensionSetting.ps1 script.

Open powershell as admin and Go to following path in source code nof1HealthPlatform\EnvironmentSetup\Certificate. In the PowerShell console, execute this command:

.\IISBinding.ps1 -VMResourceGroup {VM Resource Group} -VMName {VM Name} -VMLocation { VM Location } -URI {URI of blob storage of ExtensionSetting.ps1 script } -scriptName {ExtensionSetting.ps1}

19. Deploy Data Factory V2

Set up PowerShell as per the steps in "Prepare to deploy".

Deploy Medidata Rave Pipeline

Update the input template parameters as per client specific environment.

.\DataFactory-Deploy-Resource-Template -ResourceGroupName {resource group name } -TemplateFile .\Medidata-Study-Pipeline-template.json -TemplateParameterFile .\data-factory-parameters-template.json

Follow the above two steps for each rave enabled studies.

Deploy Novi Pipeline

Update the input template parameters as per client specific environment.

.\DataFactory-Deploy-Resource-Template -ResourceGroupName {resource group name } -TemplateFile .\Novi-Pipeline-template.json -TemplateParameterFile .\data-factory-parameters-template.json

Deploy Mobile Pipeline

Update the input template parameters as per client specific environment.

.\DataFactory-Deploy-Resource-Template -ResourceGroupName {resource group name } -TemplateFile .\Mobile-Pipeline-template.json -TemplateParameterFile .\data-factory-parameters-template.json

Deploy ODM Data Process Pipeline

This pipeline process the external data from staging table to appropriate tables in DW.

Update the input template parameters as per client specific environment.

.\DataFactory-Deploy-Resource-Template -ResourceGroupName {resource group name } -TemplateFile .\DataFactory-DataInProcessOdm.json -TemplateParameterFile .\data-factory-parameters-template.json

Appendix: Complete Steps, in PowerShell

$SubscriptionName = ""

$TemplateParameterFileName = ""

$SolrVMHost = ""

$SolrVMUsername = ""

$SolrVMPassword = ""

Connect-AzureRmAccount

Set-AzureRmContext -SubscriptionName "$SubscriptionName"

.\Setup-Resource-Group.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\keyvault-template.json -TemplateParameterFile $TemplateParameterFileName

.\Setup-KeyVaultAccess.ps1 -TemplateParameterFile $TemplateParameterFileName -KeyVaultSubscription $SubscriptionName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\background-app-template.json -TemplateParameterFile $TemplateParameterFileName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\webapi-template.json -TemplateParameterFile $TemplateParameterFileName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\application-insights-template.json -TemplateParameterFile $TemplateParameterFileName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\storage-template.json -TemplateParameterFile $TemplateParameterFileName

.\Setup-StorageAccount.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\servicebus-template.json -TemplateParameterFile $TemplateParameterFileName

.\Deploy-Nof1-Resource-Template.ps1 -TemplateFile .\cosmosdb-template.json -TemplateParameterFile $TemplateParameterFileName

.\Setup-CosmosDb-Collections.ps1 -SubscriptionName "$SubscriptionName" -TemplateParameterFile $TemplateParameterFileName

.\Setup-Sql-Database.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Setup-Solr.ps1 -TemplateParameterFile $TemplateParameterFileName -SolrVMHost $SolrVMHost -SolrVMUsername $SolrVMUsername -SolrVMPassword $SolrVMPassword

.\Setup-PortalVM.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Setup-DNN-Database.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Setup-PortalVM-DNNConnectionString.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Import-KeyVault-Secrets.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Import-ConfigurationTable-Settings.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Install-Portal-Modules.ps1 -TemplateParameterFile $TemplateParameterFileName

.\Finalize-Portal-VM-Configuration.ps1 -TemplateParameterFile $TemplateParameterFileName

yo nof1-customer-project $TemplateParameterFileName

Powered by Roadkill Wiki (2.0.275).

This page was last modified 3 months ago.

[View History](https://nuget.nof1health.com:8443/pages/history/84)

[Top](https://nuget.nof1health.com:8443/wiki/84/n-of-1-v2-deployment-guide-v210)