


Base Application Installation Guide

This guide sets out the detailed steps to installing NBS 7, end to end.

1. Make sure you're using the latest documentation from NBS Central ( [NBS Central](#))

- a. System Administrator Guide (this document)
- b. User Guide
- c. Release Notes

2. Download the Terraform and Helm configuration packages from GitHub

- a.  [Releases · CDCgov/NEDSS-Infrastructure](#)
- b. Details:
 - i. download the zip file: [nbs-infrastructure-v1.0.3.zip](#)
- c. [Release v7.1.0 · CDCgov/NEDSS-Helm](#)
- d. Details:
 - i. download the zip file: [nbs-helm-v1.1.0.zip](#)

3. Unzip nbs-infrastructure-v1.0.3.zip (run in bash/mac/cloudshell)

```
1 $ mkdir nbs-infrastructure && mv nbs-infrastructure-v1.0.3.zip nbs-infrastructure
2 $ cd nbs-infrastructure
3 $ unzip nbs-infrastructure-v1.0.3.zip
4 Archive:  nbs-infrastructure-v1.0.3.zip
5   inflating: app-infrastructure/.gitignore
6   inflating: app-infrastructure/aws-prometheus-grafana/.gitignore
7   inflating: app-infrastructure/aws-prometheus-grafana/data.tf
8   inflating: app-infrastructure/aws-prometheus-grafana/main.tf
9 [...]
```

4. Copy samples/NBS7_standard to a new directory <example_environment> (e.g. nbs7-testing, this is a name for you easily identify this new environment, arbitrary), and change into the <example_environment> directory

```
1 cp -r samples/NBS7_standard nbs7-testing
2 cd nbs7-testing
```

5. Make sure you are authenticated to AWS. Confirm access to the intended account using the following command. (More information about authenticating to AWS can be found at the following [link](#).)

```
1 $ aws sts get-caller-identity
2 {
3     "UserId": "AIDBZM0Z03E7R88J3DBTZ",
4     "Account": "123456789012",
5     "Arn": "arn:aws:iam::123456789012:user/lincolna"
6 }
```

6. Customize the Terraform configuration files **inputs.tfvars**, and **terraform.tf**

- a. For information about how to configure these two files, reference the comments within the files and the README.md file in the same directory. Collect the following information from your existing environment **before** editing the files.
 - i. **legacy-vpc-id**: This is the VPC where the existing NBS Classic application application resides. Expect a VPC ID with the format "vpc-1234567890123456". [Link to list of VPCs in the AWS console \(chose appropriate region\)](#)
 - ii. **legacy_vpc_private_route_table_id**: This is the route table used by the the subnets to which the database is attached. Expect a Route table ID with the format "rtb-1234567890abcdef". (We assume that the RDS instance is on private subnets with corresponding route tables). [Link to list of route tables - chose the one attached to private/RDS subnets](#)
 - iii. **legacy_vpc_public_route_table_id**: This is the route table used by the the subnets the application server(s) and/or the application load balancer are attached to. (We assume these are on on "public" subnets with corresponding route table). Expect a Route table ID with the format "rtb-fedcba0987654321". [Chose the one attached to "public" subnets](#)

iv. **tags section:** There are four recommended tags. You are free to add more based on your organization's policies.

v. **aws_admin_role_name:** This is the role your IAM/SSO user assumes when logged in. This value is AWS Account/Admin Account specific, Run: `aws sts get-caller-identity` to get the unique portion of the role.

vi. **modern-cidr:** When the terraform is applied later, it will create a new VPC for the required resources. Assign a CIDR range for the "modern" VPC using local addressing conventions with room for at least **4 x /24 subnets**. The existing inputs.tfvars uses 10.x.0.0/16, you may change the second octet or renumber completely using the same subnetting scheme.

b. Edit file **inputs.tfvars** (reference comments for everything that needs to change, some variables can be left blank if they are blank in the original)

c. Edit file **terraform.tf**. Change the bucket name and SITE_NAME in the file. Suggested bucket name cdc-nbs-terraform-<account number> (ignore naming in comments if conflicting)

d. create s3 bucket specified in the terraform.tf file

e. NOTE: Do not edit files in the individual modules under the "app-infrastructure" folder. For your reference, the configuration values for each TF module are documented in the readme files in each repository.

7. Verify that the allocated CIDR range for the VPC hosting NBS 7 (modern-cidr) is allowed to connect to the existing RDS instance through the security groups attached to the instance. If necessary, update your rds security group configuration to enable this access.

8. Terraform stores its state in an S3 bucket. The commands below assume that you are running Terraform authenticated to the same AWS account that contains your existing NBS 6 application. Please adjust accordingly if this does not match your setup.

a. Start the shell you will use to run terraform. Provide credentials to authenticate to the AWS CLI

b. Change directory to the account configuration directory, i.e. the one containing inputs.tfvars, and terraform.tf.

c. Initialize Terraform by running:

```
1 terraform init
```

d. Run "terraform plan" to enable it to calculate the set of changes that need to be applied:

```
1 terraform plan -var-file=inputs.tfvars
```

e. Review the changes carefully to make sure that they 1) match your intention, and 2) do not unintentionally disturb other configuration on which you depend. Then run "terraform apply":

```
1 terraform apply -var-file=inputs.tfvars
```

f. **If terraform apply generates errors, review and resolve the errors, and then rerun step e.**

9. Verify that Terraform was applied as expected

a. Examine the logs. Expect no errors.

i. One warning is expected (due to a bug in the Hashicorp EKS modul):

```
ii. 1 Warning: Argument is deprecated
2
3   with module.eks_nbs.module.eks.aws_eks_addon.this["aws-ebs-csi-driver"],
4   on .terraform/modules/eks_nbs.eks/main.tf line 392, in resource "aws_eks_addon" "this":
5   392:   resolve_conflicts      = try(each.value.resolve_conflicts, "OVERWRITE")
6
7   The "resolve_conflicts" attribute can't be set to "PRESERVE" on initial resource creation. Use "resolve_
```

b. Inspect the newly created infrastructure via the AWS console.

i. [Existing and newly created VPC, subnets](#)

ii. [EKS Kubernetes cluster](#) select the cluster and inspect Resources->Pods, Compute (expect 30+ pods at this point, and 3-5 compute nodes depending on setup in inputs.tfvars)

10. Deploy Kubernetes (K8s) support services in the Kubernetes cluster

a. Now that the infrastructure is created using terraform, the following steps detail deployment of a few prerequisites on the Kubernetes environment.

b. Start the Terminal/command line:

- i. Make sure you are still authenticated with AWS (reference the following [Configuration and credential file settings](#)).
- ii. Next step would be to authenticate into the Kubernetes cluster(EKS) using the following command (Replace the cluster name if it is different. By default it should be cdc-nbs-sandbox):

```
1 aws eks --region us-east-1 update-kubeconfig --name <clustername> # e.g. cdc-nbs-sandbox
```

Note: You should see a line "Added new context ...". If the above command errors out, there could be an issue with the AWS CLI installation or make sure you are setting the latest AWS environment variables from the previous step or the cluster name from the previous command could be wrong.

c. Run the following command to check if you are able to run commands to interact with the Kubernetes objects and the cluster.

```
1 kubectl get pods --namespace=cert-manager
```

The above command should return 3 pods

```
1 kubectl get nodes
```

The above command should list 3 worker nodes for the cluster.

d. Deploy ingress: In this step, we will install nginx-ingress using helm charts.

- i. This step will deploy NGINX ingress controller on the kubernetes cluster.
 1. Please use the values file supplied as part of nbs-helm-v1.1.0 zip file . Use this [link](#) to download the zip file. The values.yaml file should be under charts\nginx-ingress\values.yaml . The values file contains configurations for Prometheus to scrape metrics and also to instruct NGINX controller to create AWS network load balancer instead of classic load balancer in front of the k8 NGINX ingress.
 2. Unzip the release zip file and using your terminal go into the charts directory within the folder and run the following command. The result of the command below should create NGINX controller within kubernetes.

```
1 cd <HELM_DIR>/charts
2 helm upgrade --install ingress-nginx ingress-nginx --repo https://kubernetes.github.io/ingress-nginx --f
```

Note: All helm commands needs to be executed from the charts directory:

3. Monitor the status of the nginx deployment

```
1 kubectl --namespace ingress-nginx get services -o wide -w ingress-nginx-controller
```

a. Note: Use Ctrl+c to exit if the command is still running.

4. You may also check that a network load balancer was created in the AWS web console and the target groups are pointing to the EKS cluster [List of load balancers](#)

5. Run the following command to see if the NGINX controller pods are running.

```
1 kubectl get pods -n=ingress-nginx
```

e. Configure Cert manager Cluster Issuer

- i. Cert-manager is a tool to manage certificates within the Kubernetes cluster. By default, [LetsEncrypt](#) will be used as a certificate authority/issuer for certificates for NIFI and modernization-api services. This step will set up cluster issuer for LetsEncrypt production certificate issuer.
- ii. Use the manifests provided as part of nbs-helm-v1.1.0 zip file. The yaml manifests should be under k8-manifests\cert-manager\cluster-issuer-prod.yaml. Within the yaml file, ensure the sample email is replaced with your valid operations email address. This email address will be used to notify any certificate expirations that are due. In your terminal, change your directory to k8-manifests\cert-manager\ and run the following command.

```
1 cd <HELM_DIR>/k8-manifests/cert-manager
2 kubectl apply -f cluster-issuer-prod.yaml
```

iii. Run the following command to make sure the cluster issuer is deployed properly and is in ready state.

```
1 kubectl get clusterissuer
```

iv. Note: By default, lets encrypt is used as an issuer/CA. If you have manual certificates, please skip step(e) and store your certificates manually in Kubernetes secrets and reference them in your configuration by following this [link](#).

11. Deploy microservices

a. Note: Please ensure that the following DNS entries are manually created and pointed to the network load balancer in front of your Kubernetes cluster(make sure this is the ACTIVE NLB just provisioned via nginx-ingress). This should be done in your DNS service(For eg: Route 53). Please replace [Example Domain](#) with the appropriate domain name.

i. Modernized NBS Application - Eg: [app.example.com](#)

ii. NIFI application - Eg: [nifi.example.com](#)

b. We will use HELM CLI to deploy NBS microservices into Kubernetes cluster. **Please deploy the helm charts in the following order.** Verify that each microservice has started successfully before moving on to the next service.

i. elasticsearch-efs

ii. modernization-api

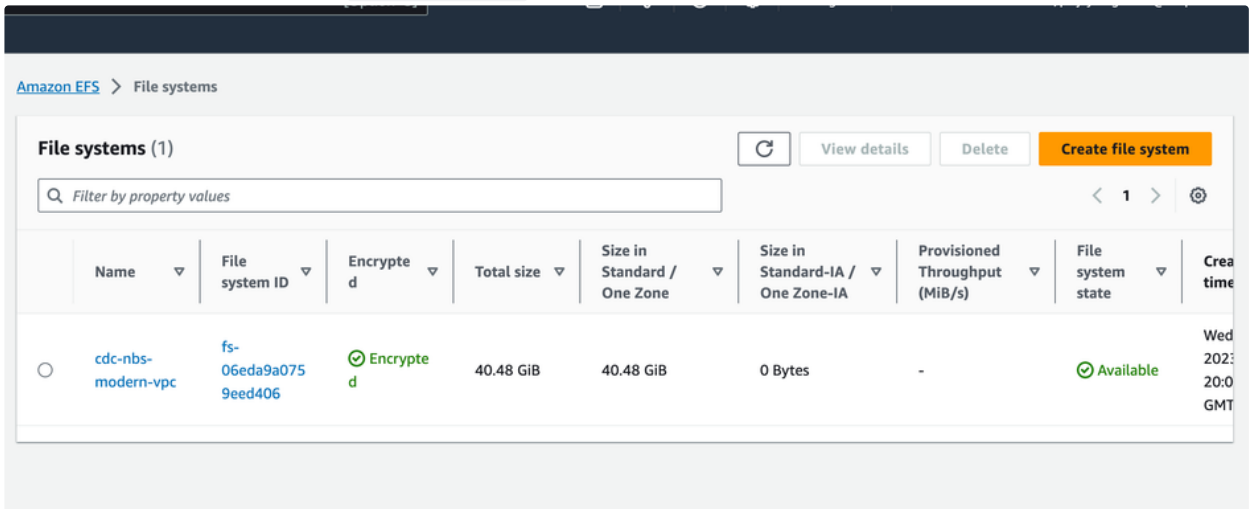
iii. nifi

iv. nbs-gateway

c. Deploy elasticsearch-efs helm chart

i. The helm chart for elasticsearch-efs should be available under charts/elasticsearch-efs.

ii. Update the values.yaml to populate `efsFileSystemId` which is the EFS file system id from the AWS console. See image below.



iii. Make sure the correct image repository and tags are populated before executing the following helm install command. The image repository and tag in the values file should point to the following:

```
1 image:
2 repository: "public.ecr.aws/o1z7u2g7/cdc-nbs-modernization/elasticsearch"
3 tag: <tag matching zip file release number e.g. v1.1.0>
```

After updating the values file, Run the following command to install Elasticsearch.

```
1 helm install elasticsearch -f ./elasticsearch-efs/values.yaml elasticsearch-efs
```

iv. Note: Check to see if the pod is running before proceeding with the next deployment using `kubectl get pods`

d. Deploy Modernization API helm chart

i. In the values.yaml file, Replace all occurrences of [app.example.com](#) with your domain name. Populate jdbc with the correct connection string details. Ensure the image repository and tags are populated with the following:

```
1 image:
2 repository: "public.ecr.aws/o1z7u2g7/cdc-nbs-modernization/modernization-api"
```

```
3 tag: <tag matching zip file release number e.g. v1.1.0>
```

Populate the jdbc section in the values file in the following format

```
1 jdbc:
2   connectionString: "jdbc:sqlserver://servername:1433;databaseName=NBS_ODSE;user=DBUsername;password=DBPa
3   user: "DBUsername"
4   password: "DBPassword"
```

1. After updating the values file, Run the following command to install modernization API.

```
1 helm install modernization-api -f ./modernization-api/values.yaml modernization-api
```

- Note: Check to see if the pod is running before proceeding with the next deployment using `kubectl get pods`

e. Deploy NiFi helm chart

i. **Important Note: The NIFI ingress is intentionally disabled by default to enhance security. Consequently, the NIFI admin UI is not accessible out of the box. To enable access to the NIFI admin interface, it is necessary to activate the ingress feature by setting "ingress:enabled: true" within the values.yaml file before initiating the installation process using the Helm chart.**

For the sake of heightened security, it is strongly advised to establish a privately accessible domain name instead of a public one, considering the presence of known security vulnerabilities within NIFI.

ii. In the values.yaml file, Replace all occurrences of [nifi.example.com](#) with your domain name. Populate jdbc connection string with the correct information which includes the username and password. Populate singleUserCredentialsPassword with the password for the NIFI admin UI. This can be any password of your choice. Ensure the image repository and tags are populated with the following :

```
1 image:
2   repository: "public.ecr.aws/o1z7u2g7/cdc-nbs-modernization/nifi"
3   tag: <tag matching zip file release number e.g. v1.1.0>
```

For the jdbcConnectionString value, get the RDS server URL and update the database connection string including database credentials in the following format :

```
1 jdbcConnectionString: "jdbc:sqlserver://servername:1433;databaseName=NBS_ODSE;user=DBUser;password=DBpass
```

1. After updating the values file, Run the following command to install nifi.

iii.

```
1 helm install nifi -f ./nifi/values.yaml nifi
```

- Note: Check to see if the pod is running before proceeding with the next deployment using `kubectl get pods`

f. Deploy nbs-gateway helm chart

i. In the values.yaml file, Make sure nbsExternalName is pointing to the correct legacy NBS domain. Ensure the image repository and tags are populated with the following :

```
1 image:
2   repository: "public.ecr.aws/o1z7u2g7/cdc-nbs-modernization/nbs-gateway"
3   tag: <tag matching zip file release number e.g. v1.1.0>
```

1. After updating the values file, Run the following command to install nbs-gateway.

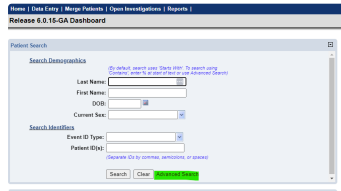
```
1 helm install nbs-gateway -f ./nbs-gateway/values.yaml nbs-gateway
```

- Note: Check to see if the pod is running before proceeding with the next deployment using `kubectl get pods`

Smoke Test (manual)

Test end-to-end functionality using the following steps:

1. Login to the new NBS system using the URL you set up above: <https://app.<exampledomain>/nbs/login> (e.g. [NBS](#)).
2. Select advanced search:



3. View some patient records. One way to do this is to select "Male" from the the "Sex" pull down and click search:



4. Results should appear in the pane on the right:



When you are able to perform these steps and get results back, it validates the following portions of the system are functioning properly:

- Name resolution is working
- Routing requests and traffic between the NBS 6.x portions of the system and NBS 7 is working properly
- Routing from the NBS 7 components to the (shared) database is working
- The search indices have been created and populated, and are available, thereby validating Elasticsearch, NiFi, and the Modernization-API

API Smoke Test (scripted)

nbs-test-api.sh script is included in the infrastructure zip file (scripts/observability/nbs-test-api.sh)

It will

- create a patient
- search for the patient
- delete that patient (note record still exists but is inactive)

It is a bash script that can be run via cloudshell if NBS is hosted in

AWS or by any system with bash installed. It requires a user in the database that can run api calls.

Curl is the only other dependency.

USAGE

`nbs-test-api.sh [-h] [-?] [-d] [-D] [-P] [-B BASE_URL] [-U USER] [-c count]`

For the initial smoke test run

`nbs-test-api.sh -B https://app.<site>.example.com -U <apiuser> -c 10`

This will verify api functionality AND populate the observability dashboards with some initial traffic