







NBS 7.0 Installation Guide

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

1. Get the documentation from NBS Central ( [NBS Central](#))
 - a. System admin guide (this document)
 - b. Release notes
 - c. User guide
2. Download configuration (Terraform & Helm) packages
 - a.  [Releases · CDCgov/NEDSS-Infrastructure](#)
 - b. Details:
 - i. download the zip file: [nbs-infrastructure](#)
 - c. [Release · CDCgov/NEDSS-Helm](#)
 - d. Details:
 - i. download the zip file: [nbs-helm](#)
3. Unzip nbs-infrastructure-v1.0.0.zip.
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)
5. cd <example environment>
6. Make sure you are authenticated using the following [link](#) .
7. confirm access to the intended account using

```
1 aws sts get-caller-identity
```

8. Configure the Terraform files
 - a. collect information to update the config files **inputs.tfvars**, and  [terraform.tf](#)
 - b. For information about how to configure these two files see  [Today I Learned for programmers](#) copy in the <examplesite> directory
 - i. legacy-vpc-id = this is the VPC where the existing NBS Classic application application resides e.g. [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 the database is attached (We assume RDS instance is on private subnets with corresponding route table) [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 is attached (We assume these are on on "public" subnets with corresponding route table) [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.
 - c. NOTE: Do not edit files in the individual module folders under the "app-infrastructure" folder. For your reference, the configuration values for each TF module are documented in the readme files in each repository.
9. update the config files **inputs.tfvars**, and  [terraform.tf](#)
10. Add/Verify that the CIDR range for modern-cidr is allowed to connect to the existing RDS instance through the existing security group.
11. Run the Terraform in the account containing your existing NBS 6 application
 - a. Start the Terraform command line. Provide credentials to authenticate to the AWS CLI
 - b. cd to the account configuration directory, i.e. the one containing inputs.tfvars, and  [terraform.tf](#) .
 - c. Run

```
1 terraform init
```

d. Run "terraform plan"

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

e. Run "terraform apply"

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

f. If terraform apply generates errors, rerun step e.

12. Verify that Terraform was applied as expected

a. Examine logs. Expect no errors.

i. One expected warning.

```
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
```

This is due to a bug in a Hashicorp EKS module.

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

13. 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 authenticated using the following [link](#) .

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 cdc-nbs-sandbox
```

Note: 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-v1.0.0 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 --
```

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, lets encrypt will used as a certificate issuer for getting certificates for NIFI and modernization-api services. This step will set up cluster issuer for lets encrypt production certificate issuer.

ii. Use the manifests provided as part of nbs-helm-v1.0.0 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](#).

14. 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. Deploy the helm charts in the following order:

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.

Amazon EFS > File systems

File systems (1) Refresh View details Delete Create file system

Filter by property values

	Name	File system ID	Encryption	Total size	Size in Standard / One Zone	Size in Standard-IA / One Zone-IA	Provisioned Throughput (MiB/s)	File system state	Creation time
<input type="radio"/>	cdc-nbs-modern-vpc	fs-06eda9a0759eed406	✔ Encrypted	40.48 GiB	40.48 GiB	0 Bytes	-	✔ Available	Wed 2023-10-20 20:00 GMT

- 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: v1.0.0
```

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: v1.0.0
```

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=DBPassword"
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. 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: v1.0.0
```

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.

```
ii. 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/oiz7u2g7/cdc-nbs-modernization/nbs-gateway"
3   tag: v1.0.0
```

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

15. Start NIFI Patient Search Ingestion:

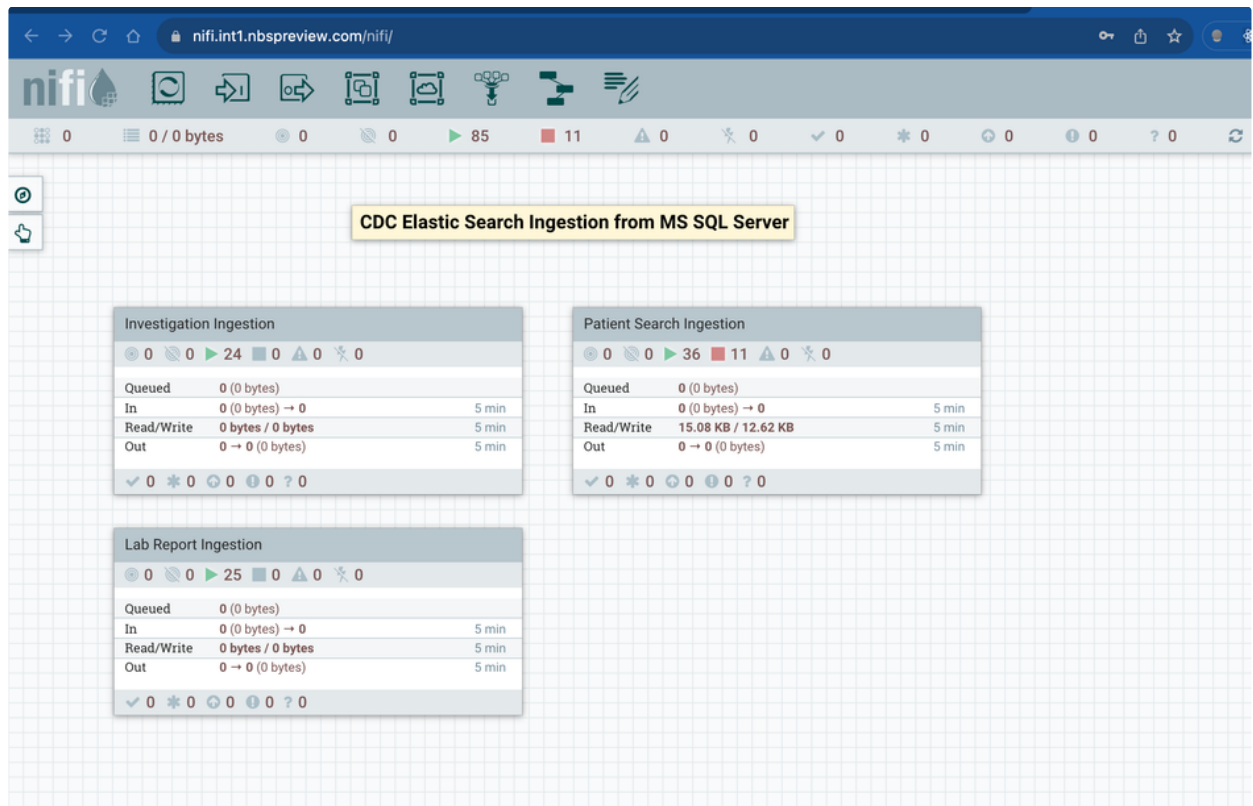
a. NIFI is used to populate Elasticsearch indices from the NBS transactional database. A specific NIFI job has been intentionally stopped by default and made manual to allow STLTs to start this resource-intensive operation. The following are the steps to turn on patient search ingestion:

b. Note: Please note that this step is time and resource-intensive. It is highly recommended to start this during off-peak hours.

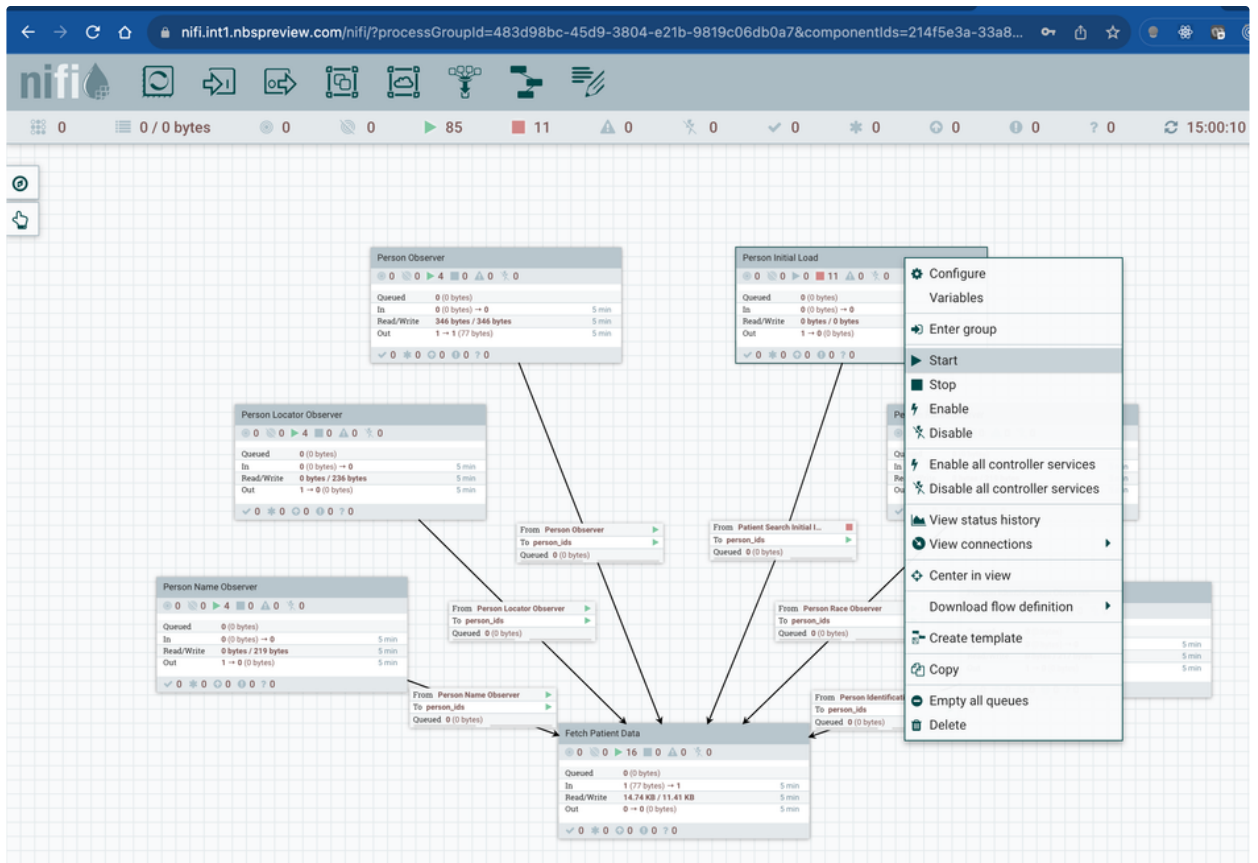
i. Login into nifi admin console using the following URL: nifi.example.com/nifi

ii. Username should be "admin" and password is what was set in the NIFI helm values file for singleUserCredentialsPassword.

iii. As soon as you login you should see the following screen.



iv. Double-click on the Patient Search Ingestion. you should see the following screen.



v. Right-click on the Person Initial Load and click on Start. This will kick start the patient records ingestion process.