8 - openIDL Carrier HDS Connector Service Deployment

Deployment on AWS k8s cluster

Prerequisites

- K8s cluster up and running on AWS
- AWS credentials (access key, secret, key, role,user external id). The AWS CLI user should have access and authorization to manage the K8s cluster
- Bastion machine on AWS with SSH access information (IP) and credentials (user name and private key). The current openIDL deployment is tested on bastion machine based on AWS image amzn2-ami-hvm-2.0.20230221.0-x86_64-gp2.
- Carrier HDS is up and running. Access information and credentials are available (the carrier HDS connector service can be re-configured post deployment if changes of the used endpoints are required)
- The carrier Node As a Service is up and running
- The Carrier Node As a Service vendor provides the following information:
  - Hashicorp Vault access information
  - HLF console access information

Configuration (in git private repository)

The specific node settings are stored in a file (or as AWX credentials in case they define sensitive data) and supplied to the openIDL deployment scripts as a resource from a git repository. The configuration file can be created as a copy of "https://github.com/openidl-org/openidl-aais-gitops/blob/develop/ansible/values.yml", configured locally and pushed to the private git repository of the node. The above config file should be pushed to the git repo with path "orgs_config/<org id>/config.yml"

The following parameters must be configured (the rest can stay as is):

```
# Organization ID / HLF MSP ID / Organization name. This is the org setting used to provision the cloud resources

# Example: carr1

org_id: ""



# Environment ID is usually a combination of the org_id and the env used to provision the cloud resources

# Example: carr1-test

env_id: ""



# The AWS account number

aws_account_number: ""



# The AWS region of the deployment

# Example: us-east-2

region:



# The opneIDL application type: analytics or carrier

# Example: carrier

application_node_type: "carrier"
```

The openIDL ansible playbooks use the fabric operator ansible collection that is imported from the private git repository.

The collection is available as archive file in the openIDL repository: https://github.com/senofi/openidl-testnet-config/blob/n107-azure/bin/ibm-blockchain_platform-2.0.0-beta.tar.gz.

The above archive should be downloaded and manually uploaded in the private git repository under the path bin/ ibm-blockchain_platform-2.0.0-beta.tar.gz.

Setup AWX

AWX helps to organize and manage the ansible resources. The ansible playbooks can be also executed from a command line using ansible CLI (https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html). Using AWX is optional. In case ansible CLI is used, the ansible resources should be organized and managed locally (i.e. in config files).

AWX Project

Create a new project named with the org name, use openIDL ansible git URL and the appropriate branch.

Source Control Type: Git

Source Control URL: https://github.com/openidl-org/openidl-aais-gitops.git

Source control Branch: develop

Update Revision on Launch: Checked

AWX Inventory

Create new inventory in AWX named with the org name. Add host using the azure bastion machine address. Add a group named ansible_provisioners. Add the bastion host to the group.

Credentials

The following credential types should be created in AWX before adding the specific credentials resources. The created credentials will be assigned and used by the ansible playbooks to deploy the HDS connector service. In case the ansible playbooks are executed outside AWX by using ansible CLI on a local machine, the credentials should be supplied as extra vars (i.e. as files) to the CLI. The supplied credentials vars should be defined as specified in the Injector Configuration Section below (for example could be separated or merged in multiple or single local files).

Credential	Description	Type Definition in AWX

aws-cli	The AWS credential is used to access AWS APIs. The IAM user should have access to the used k8s cluster.
	
	The user is used by the playbooks to perform the deployment and setup actions.
		Input Configuration Section:
		
		`fields:`
		`  - id: aws_access_key`
		`    type: string`
		`    label: aws_access_key`
		`    secret: true`
		`    help_text: AWS IAM user access key for aws`
		`  - id: aws_secret_key`
		`    type: string`
		`    label: aws_secret_key`
		`    secret: true`
		`    help_text: AWS IAM user secret key for aws`
		`  - id: aws_external_id`
		`    type: string`
		`    label: aws_external_id`
		`  - id: aws_assume_role_arn`
		`    type: string`
		`    label: AWS IAM user role to assume`
		`required:`
		`  - aws.access_key`
		`  - aws.secret_key`
		`  - aws.external_id`
		`  - aws.assume_role_arn`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  aws_access_key: '{{ aws_access_key }}'`
		`  aws_secret_key: '{{ aws_secret_key }}'`
		`  aws_external_id: '{{ aws_external_id }}'`
		`  aws_assume_role_arn: '{{ aws_assume_role_arn }}'`

git-config	Git credentials for access to the private git repository where the configuration file is available.
	
	The ansible playbooks will use the credential to pull configuration from the private repository
		Input Configuration Section:
		
		`fields:`
		`  - id: sshkey`
		`  type: string`
		`  label: Base64 encoded deploy private key string`
		`  secret: true`
		`  - id: repourl`
		`  type: string`
		`  label: GIT repo URL`
		`  - id: repobranch`
		`  type: string`
		`  label: Git repo branch`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  ssh_key: '{{ sshkey }}'`
		
		`git_configs_repo_url: '{{ repourl }}'`
		
		`git_configs_repo_branch: '{{ repobranch }}'`

bastion	Bastion Machine SSH credential. The machine is used as a remote agent for the ansible playbooks. It is the entry point (gateway) to access the AWS K8S cluster in order to setup and deploy the carrier HDS connector service container.	Machine - an existing standard credential in AWX

hds-access	Access information for application HDS DB.
	
	This credential is injected by the playbooks to configure the openIDL applications for access to the local carrier HDS database. The connection to the HDS DB will be established by the carrier HDS connector service at runtime
		Input Configuration Section:
		
		`fields:`
		`  - id: hds_host`
		`  type: string`
		`  label: HDS host`
		`  help_text: HDS host address`
		`  - id: hds_port`
		`  type: string`
		`  label: hds_port`
		`  help_text: HDS port`
		`  - id: hds_username`
		`  type: string`
		`  label: hds_username`
		`  secret: true`
		`  - id: hds_password`
		`  type: string`
		`  label: hds_password`
		`  secret: true`
		`  - id: hds_dbname`
		`  type: string`
		`  label: hds_dbname`
		`required:`
		`  - hds_host`
		`  - hds_port`
		`  - hds_username`
		`  - hds_password`
		`  - hds_dbname`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  hds_host: '{{ hds_host }}'`
		`  hds_port: '{{ hds_port }}'`
		`  hds_dbname: '{{ hds_dbname }}'`
		`  hds_password: '{{ hds_password }}'`
		`  hds_username: '{{ hds_username }}'`

vault-access	The Hashicorp vault is used to store securely the certs and private keys of the users that can access the HLF network by connecting to the HLF peer of the carrier.
	
	Those users are used by the carrier HDS connector service to connect and transact securely on the openIDL network by connecting to the carrier peer.
	
	The vault access credential contains the access information and credentials that is used to connect to the Hashicorp vault to fetch the HLF user credentials.
	
	The vendor of the carrier Node As a Service will provide the credential to the carrier. The credential is a base64 encoded json file.
		Input Configuration Section:
		
		`fields:`
		`  - id:`
		`vault_config_encoded`
		`    type: string`
		`    label:`
		`vault_config_encoded`
		`    help_text: Vault endpoint and access credentials`
		
		Injector Configuration Section:
		
		`extra_vars:`
		
		`vault_config_encoded:`
		`'{{`
		`vault_config_encoded`
		`}}'`

fabric-console	Fabric Operator Console access default user/password.
	
	Used by the playbooks to inject default user and password for the fabric console deployment. Make sure the generate a strong password as it will secure properly the access to the node HLF managed.
	
	The playbooks also use this credential to connect to the console for the purpose of performing operations on the HLF nodes.
	
	Take note of that credential as the provided user and password will be required to log in to the fabric operator console.
		Input Configuration Section:
		
		`fields:`
		`  - id:`
		`console_username`
		`    type: string`
		`    label:`
		`console_username`
		`    help_text: Fabric Operator Console Username`
		`  - id:`
		`console_password`
		`    type: string`
		`    label:`
		`console_password`
		`    secret: true`
		`    help_text: Fabric Operator Console Password`
		`required:`
		`  - console_username`
		`  - console_password`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  console_password:`
		`'{{ console_password }}'`
		`  console_username:`
		`'{{ console_username }}'`

Ansible Playbooks

Playbook	Template Name	Credentials	Description
ansible/environment-setup.yaml	environment-setup	bastion git-config	Installs a few open-source libraries required for ansible playbook runs (i.e. AWS CLI, JQ, etc)
ansible/deploy-mongodb-k8s-native.yml	aws-deploy-mongodb	aws-cli bastion git-config-azure	Installs MongoDB on the k8s cluster. The mongo DB access is stored as k8s secret that is later shared with the HDS connector service container.
ansible/deploy-openidl-app-config-k8s-native.yaml	aws-deploy-carrier-config	aws-cli bastion fabric-console hds-access vault-access git-config	Creates a k8s secret with all configurations needed by the HDS connector service. The created secret content is later injected in the HDS connector service container
ansible/deploy-openidl-app-carrier-k8s-native.yaml	aws-carrier-app-deploy	aws-cli bastion git-config	Deploys the HDS container service container using the openIDL helm chart

Deployment on Azure K8s cluster

Prerequisites

- K8s cluster up and running on Azure
- Azure CLI access and credentials (tenant id, subscription id , user name and password). The Azure CLI user should have access and authorization to manage the K8s cluster
- Bastion machine on Azure with SSH access information (IP) and credentials (user name and private Key). The current openIDL deployment is tested on bastion machine based on ubuntu 22.04 (Azure image: com-ubuntu-server-jammy)
- Carrier HDS is up and running. Access information and credentials are available (the carrier HDS connector service can be re-configured post deployment if changes of the used endpoints are required)
- The carrier Node As a Service is up and running
- The Carrier Node As a Service vendor provides the following information:
  - Hashicorp Vault access information
  - HLF console access information

Configuration (in git private repository)

The specific node settings are stored in a file (or as AWX credentials in case they define sensitive data) and supplied to the openIDL deployment scripts as a resource from a git repository. The configuration file can be created as a copy of "https://github.com/openidl-org/openidl-aais-gitops/blob/develop/ansible/values.yml", configured locally and pushed to the private git repository of the node. The above config file should be pushed to the git repo with path "orgs_config/<org id>/config.yml"

The following parameters must be configured (the rest can stay as is):

```
# The k8s cluster where the carrier HDS connector service container will be deployed
openidl_apps_cluster: ""

# Azure resource group
azure_aks_rg: ""

# Set to azure as a cloud deployment (this is required to overwrite the default value of aws that is used for AWS deployment)
cloud_deployment: "azure"

# Organization ID / HLF MSP ID / Organization name. This is the org setting used to provision the cloud resources
# Example: carr1
org_id: ""

# Environment ID is usually a combination of the org_id and the env used to provision the cloud resources
# Example: carr1-test
env_id: ""

# The opneIDL application type: analytics or carrier
# Example: carrier
application_node_type: "carrier"
```

The openIDL ansible playbooks use the fabric operator ansible collection that is imported from the private git repository.

The collection is available as archive file in the openIDL repository: https://github.com/senofi/openidl-testnet-config/blob/n107-azure/bin/ibm-blockchain_platform-2.0.0-beta.tar.gz.

The above archive should be downloaded and manually uploaded in the private git repository under the path bin/ ibm-blockchain_platform-2.0.0-beta.tar.gz.

Setup AWX

AWX helps to organize and manage the ansible resources. The ansible playbooks can be also executed from a command line using ansible CLI (https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html). Using AWX is optional. In case ansible CLI is used, the ansible resources should be organized and managed locally (i.e. in config files).

AWX Project

Create a new project named with the org name, use openIDL ansible git URL and the appropriate branch.

Source Control Type: Git

Source Control URL: https://github.com/openidl-org/openidl-aais-gitops.git

Source control Branch: develop

Update Revision on Launch: Checked

AWX Inventory

Create new inventory in AWX named with the org name. Add host using the azure bastion machine address. Add a group named ansible_provisioners. Add the bastion host to the group.

Credentials

The following credential types should be created in AWX before adding the specific credentials resources. The created credentials will be assigned and used by the ansible playbooks to deploy the HDS connector service. In case the ansible playbooks are executed outside AWX by using ansible CLI on a local machine, the credentials should be supplied as extra vars (i.e. as files) to the CLI. The supplied credentials vars should be defined as specified in the Injector Configuration Section below (for example could be separated or merged in multiple or single local files).

Credential	Description	Type Definition in AWX

azure-cli	The azure CLI is used to establish connection to the running K8S cluster on Azure.	**Input Configuration Section:**
		`fields:`
		`  - id: azure_cli_user`
		`  type: string`
		`  label: azure_cli_user`
		`  secret: true`
		`  help_text: Azure CLI user name service principal`
		`  - id: azure_cli_pw`
		`  type: string`
		`  label: azure_cli_pw`
		`  secret: true`
		`  help_text: Azure CLI user password service principal`
		`  - id: azure_cli_tenant`
		`  type: string`
		`  label: azure_cli_tenant`
		`  - id: azure_cli_subscription`
		`  type: string`
		`  label: azure_cli_subscription`
		`required:`
		`  - azure_cli_user`
		`  - azure_cli_pw`
		`  - azure_cli_tenant`
		`  - azure_cli_subscription`
		
		Injector Configuration Section:
		`extra_vars:`
		`  azure_cli_pw: '{{ azure_cli_pw }}'`
		`  azure_cli_user: '{{ azure_cli_user }}'`
		`  azure_cli_tenant: '{{ azure_cli_tenant }}'`
		
		`azure_cli_subscription: '{{ azure_cli_subscription }}'`

git-config-azure	Git credentials for access to the private git repository where the configuration file is available.
	
	The ansible playbooks will use the credential to pull configuration from the private repository
		Input Configuration Section:
		
		`fields:`
		`  - id: sshkey`
		`  type: string`
		`  label: Base64 encoded deploy private key string`
		`  secret: true`
		`  - id: repourl`
		`  type: string`
		`  label: GIT repo URL`
		`  - id: repobranch`
		`  type: string`
		`  label: Git repo branch`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  ssh_key: '{{ sshkey }}'`
		`  git_configs_repo_url: '{{ repourl }}'`
		
		`git_configs_repo_branch: '{{ repobranch }}'`

azure-bastion	Bastion Machine SSH credential. The machine is used as a remote agent for the ansible playbooks. It is the entry point (gateway) to access the Azure K8S cluster in order to setup and deploy the carrier HDS connector service container.	Machine - an existing standard credential in AWX

hds-access	Access information for application HDS DB.
	
	This credential is injected by the playbooks to configure the openIDL applications for access to the local carrier HDS database. The connection to the HDS DB will be established by the carrier HDS connector service at runtime
		Input Configuration Section:
		
		`fields:`
		`  - id: hds_host`
		`  type: string`
		`  label: HDS host`
		`  help_text: HDS host address`
		`  - id: hds_port`
		`  type: string`
		`  label: hds_port`
		`  help_text: HDS port`
		`  - id: hds_username`
		`  type: string`
		`  label: hds_username`
		`  secret: true`
		`  - id: hds_password`
		`  type: string`
		`  label: hds_password`
		`  secret: true`
		`  - id: hds_dbname`
		`  type: string`
		`  label: hds_dbname`
		`required:`
		`  - hds_host`
		`  - hds_port`
		`  - hds_username`
		`  - hds_password`
		`  - hds_dbname`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  hds_host: '{{ hds_host }}'`
		`  hds_port: '{{ hds_port }}'`
		`  hds_dbname: '{{ hds_dbname }}'`
		`  hds_password: '{{ hds_password }}'`
		`  hds_username: '{{ hds_username }}'`

vault-access	The Hashicorp vault is used to store securely the certs and private keys of the users that can access the HLF network by connecting to the HLF peer of the carrier.
	
	Those users are used by the carrier HDS connector service to connect and transact securely on the openIDL network by connecting to the carrier peer.
	
	The vault access credential contains the access information and credentials that is used to connect to the Hashicorp vault to fetch the HLF user credentials.
	
	The vendor of the carrier Node As a Service will provide the credential to the carrier. The credential is a base64 encoded json file.
		Input Configuration Section:
		
		`fields:`
		`  - id: vault_config_encoded`
		`    type: string`
		`    label: vault_config_encoded`
		`    help_text: Vault endpoint and access credentials`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  vault_config_encoded: '{{ vault_config_encoded }}'`

fabric-console	Fabric Operator Console access default user/password.
	
	Used by the playbooks to inject default user and password for the fabric console deployment. Make sure the generate a strong password as it will secure properly the access to the node HLF managed.
	
	The playbooks also use this credential to connect to the console for the purpose of performing operations on the HLF nodes.
	
	Take note of that credential as the provided user and password will be required to log in to the fabric operator console.
		Input Configuration Section:
		
		`fields:`
		`  - id: console_username`
		`    type: string`
		`    label: console_username`
		`    help_text: Fabric Operator Console Username`
		`  - id: console_password`
		`    type: string`
		`    label: console_password`
		`    secret: true`
		`    help_text: Fabric Operator Console Password`
		`required:`
		`  - console_username`
		`  - console_password`
		
		Injector Configuration Section:
		
		`extra_vars:`
		`  console_password: '{{ console_password }}'`
		`  console_username: '{{ console_username }}'`

Ansible Playbooks

The following ansible playbooks will setup, configure and deploy the carrier HDS connector service. The playbooks should be executed in the order specified below and using the credentials as defined above (ansible CLI may be used instead of AWX).

When running in AWX, the corresponding templates should be created before executing the jobs (run the playbooks by launching the AWX templates)

Playbook	Template Name	Credentials	Description
ansible/environment-setup-carrier-azure.yaml	environment-setup-carrier-azure	azure-bastion git-config-azure	Installs a few open-source libraries required for ansible playbook runs (i.e. Azure CLI, JQ, etc)
ansible/deploy-mongodb-k8s-native.yml	azure-deploy-mongodb	azure-cli azure-bastion git-config-azure	Installs MongoDB on the k8s cluster. The mongo DB access is stored as k8s secret that is later shared with the HDS connector service container.
ansible/deploy-openidl-app-config-k8s-native.yaml	azure-deploy-carrier-config	azure-cli azure-bastion fabric-console hds-access vault-access git-config-azure	Creates a k8s secret with all configurations needed by the HDS connector service. The created secret content is later injected in the HDS connector service container
ansible/deploy-openidl-app-carrier-k8s-native.yaml	azure-carrier-app-deploy	azure-cli azure-bastion git-config-azure	Deploys the HDS container service container using the openIDL helm chart