4 - AWX Setup and Configuration

This page provides information on the steps that are required to set up and configure the openIDL Ansible Jobs. Those jobs are essential to set up the openIDL node environment and deploy the different node components on the provisioned infrastructure.

The openIDL ansible playbooks can be used and executed standalone from a CLI. The preferred way though is to use AWX as it provides a powerful engine to execute ansible playbooks through a user-friendly web-based interface or automate using AWX Web APIs.

More information on AWX: https://github.com/ansible/awx

Prerequisites:

- AWS infrastructure is provisioned
- AWX is installed and operational
- Access to AWX with the default Admin user/password
- Configuration is done and available at a private git repository
- Credentials information is defined and available
    - AWS IAM user
    - git private repo deploy key
    - bastion ssh private key
    - HDS db access
    - fabric console user and password
- Bastion machine (gateway) host address is available

The steps:

Step	Notes	User
Create new Organization	Create a new organization with the org_id	admin
Setup new AWX org user	Create a new user specific to the organization, and assign admin permissions to the created organization above. It is a good idea to use a dedicated user for every organization that is deployed through AWX. Note that AWX can be used in a multitenant mode where multiple organizations can be deployed using the same AWX.	admin
Setup org 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: operator-develop Update Revision on Launch: Checked	org user
Setup inventory	Create a new Inventory named with the org name (bastion-org_id). Add host using the bastion machine address Add a group named ansible_provisioners Add the bastion host to the group	org user
Create Credentials	Create the credential types as specified below (see credentials table)	admin
Create AWX job tempaltes	Create the AWX job templates as specified below (see AWX job templates table)	org user

Credentials:

Credential	Description	Definition/Type

aws-git-actions	An AWS credential is used to access AWS APIs. The IAM user is created during the AWS provisioning step. This user usually should have access to AWS resources and the provisioned k8s clusters (HLF and applications k8s). The user is usually named and suffixed with git-actions admin. External AWS id is usually git-actions.
	
	The user is used by the playbooks to perform the deployment and setup actions.
	
	The credential detailed parameters can be found in the terraform state (project <org_id>-<env>-aws-resources; entry: "name": "git_actions_access_key")
		`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`
		
		`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 }}'`

aws-terraform	The terraform AWS credential used to provision some resources in AWS like DNS entries. You may find the credentials of the terraform user in terraform state of your project <org_id>-<env>-iam (entry "user": "terraform_user")	The definition type is the same as defined for aws-git-actions. The definition can be re-used when creating the credential by picking the type as created above.

git-config	Git credentials (used to pull configuration from the private repository)	`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`
		
		`extra_vars:`
		`  ssh_key: '{{ sshkey }}'`
		`  git_configs_repo_url: '{{ repourl }}'`
		`  git_configs_repo_branch: '{{ repobranch }}'`

bastion	Bastion Machine SSH credential. This machine is bootstrapped during the AWS infrastructure provisioning step. It is used as a remote agent for the ansible playbooks. It is the entry point (gateway) to access the AWS infrastructure in order to setup and deploy the network.	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 ansible playbooks don't use it to establish a connection to the HDS and perform operations.
		`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`
		
		`extra_vars:`
		`  hds_host: '{{ hds_host }}'`
		`  hds_port: '{{ hds_port }}'`
		`  hds_dbname: '{{ hds_dbname }}'`
		`  hds_password: '{{ hds_password }}'`
		`  hds_username: '{{ hds_username }}'`

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.
		`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`
		
		`extra_vars:`
		`  console_password: '{{ console_password }}'`
		`  console_username: '{{ console_username }}'`

AWX Job Templates:

Playbook	Template Name	Credential	Description
ansible/environment-setup.yaml	<env_id>-<org_id>-environment-setup	aws-git-actions bastion git-config	Install open source tools on the bastion host. Setup the access to the cloud APIs
ansible/deploy-fabric-ingress.yml	<env_id>-<org_id>-deploy-fabric-ingress	aws-git-actions bastion git-config	Deploy Ingress controllers (classes) and cloud load balancers for the HLF k8s cluster
ansible/dns-zone-config-blk.yml	<env_id>-<org_id>-dns-config-blk	aws-terraform bastion git-config	Creates DNS entries to the defined domain and routes to the deployed load balancers. Specific to the HLF and Vault endpoints

ansible/dns-zone-config-apps.yml	<env_id>-<org_id>-dns-config-apps	aws-terraform bastion git-config	Creates DNS entries to the defined domain and routes to the deployed load balancers. Specific to the openIDL application endpoints
ansible/deploy-vault.yml	<env_id>-<org_id>-deploy-vault	aws-git-actions bastion git-config	Deploy Vault raft cluster for storing HLF identities (application and HLF nodes admins)
ansible/deploy-fabric-operator.yml	<env_id>-<org_id>-deploy-fabric-operator	aws-git-actions bastion git-config	Deploy HLF fabric operator
ansible/deploy-fabric-console.yml	<env_id>-<org_id>-deploy-fabric-console	aws-git-actions bastion git-config fabric-console	Deploy HLF operator console
ansible/deploy-openidl-app-identities.yml	<env_id>-<org_id>-deploy-app-identities	aws-git-actions bastion git-config fabric-console	Registers and enrolls the openidl application identities used to transact on the openidl fabric network
ansible/deploy-openidl-app-ingress.yml	<env_id>-<org_id>-deploy-app-ingress	aws-git-actions bastion git-config	Deploys the application ingress controller and class. Creates the applications load balancers for the applications k8s cluster.
ansible/deploy-mongodb.yml	<env_id>-<org_id>-deploy-mongodb	aws-git-actions bastion git-config	Deploys mongoDB as application database
ansible/deploy-openidl-app-config.yaml	<env_id>-<org_id>-deploy-app-config	aws-git-actions bastion git-config fabric-console hds-access (carrier node)	Generates and deploys the application config as secrets
ansible/deploy-openidl-app.yaml	<env_id>-<org_id>-deploy-app	aws-git-actions bastion git-config	Deploys the openidl applications
ansible/chaincode-private-init.yml	<env_id>-<org_id>-chaincode-init	bastion git-config fabric-console	Calls init method of the chaincode deployed on the carrier/analytics private channel