**Installation of MicroK8s**

1) Configure Repo to enable SNAP for MicroK8s installation.

sudo dnf –y install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm>

**Purpose:** This installs the EPEL (Extra Packages for Enterprise Linux) repository, which provides additional packages for your system.

sudo subscription-manager repos --enable "rhel-\*-optional-rpms" --enable "rhel-\*-extras-rpms"

**Purpose:** Enables optional and extras repositories and upgrades your system packages.

sudo dnf -y upgrade

2) Install SNAPD.

sudo yum install snapd

**Purpose:** Installs snapd, a package manager that lets you install and manage software packages (snaps).

3) Enable systemd unit of snapd communication

sudo systemctl enable --now snapd.socket

**Purpose:** Activates the snapd service so it can run and manage snaps

4) Enable classic snap

sudo ln -s /var/lib/snapd/snap /snap

**Purpose:** Creates a symbolic link to enable classic snap support.

5) Install MicroK8s

sudo snap install microk8s --classic

**Purpose:** Installs MicroK8s, a lightweight Kubernetes distribution

6). Linking microk8s binary for accessibility.

ln -s /var/lib/snapd/snap/bin/microk8s /usr/local/bin/

**Purpose:** Creates a link to the MicroK8s executable, making it easier to run from the command line.

7) Enable kubernetes operators.

microk8s enable dashboard ingress registry storage cert-manager kube-ovn

**Purpose:** Activates useful Kubernetes add-ons like the dashboard, ingress controller, registry, storage, certificate manager, and network plugin.

8) Enable kubernetes networking.

microk8s enable kube-ovn --force

**Purpose:** Sets up the Kubernetes network overlay using Kube-OVN.

9) Defining alias for kubernetes commands.

echo “alias helm=microk8s.helm” >> /etc/bash\_profile

echo “alias kubelet=microk8s.kubelet” >> /etc/bash\_profile

**Purpose:** Creates shortcuts (aliases) for helm and kubelet commands, making them easier to use.

10) Yum -y install git

**Purpose:** Installs Git, a version control system useful for managing your code.

11) Open Firewall Ports for MicroK8s:

sudo firewall-cmd --add-port=16443/tcp --permanent

sudo firewall-cmd --add-port=80/tcp --permanent

sudo firewall-cmd --add-port=443/tcp --permanent

sudo firewall-cmd –reload

**Purpose:** Opens necessary ports for MicroK8s communication and services, specifically port 16443 for the Kubernetes API server, and ports 80 and 443 for web traffic.

12) Set Permissions for /mnt/storage:

sudo chmod 777 /mnt/storage

**Purpose:** Grants full read, write, and execute permissions to all users for the /mnt/storage directory.

13) AWX Configuration:

**Configure Hosts:**

* Create an inventory file for AWX.

nano /etc/awx/hosts

Add your hosts to this file.

 **Add Projects:**

* In the AWX web UI, go to Projects.
* Click the "+" button to add a new project.
* Fill in the necessary details, such as Name, Organization, and SCM URL if using a source control system like Git.

 **Create Templates:**

* In the AWX web UI, go to Templates.
* Click the "+" button to create a new job template.
* Fill in the details such as Name, Job Type, Inventory, Project, and Playbook.

**How to Install Ansible AWX on Kubernetes Cluster**

[Ansible AWX](https://github.com/ansible/awx) is a powerful open-source tool for managing and automating IT infrastructure. AWX provides a graphical user interface for Ansible, allowing you to easily create, schedule, and run Ansible playbooks.

Kubernetes, on the other hand, is a popular container orchestration platform that is widely used for deploying and managing containerized applications.

**Prerequisites:**

* Kubernetes cluster
* Kubectl
* A regular user with sudo rights and cluster admin rights
* Internet connectivity

**Step :1 Install helm:**

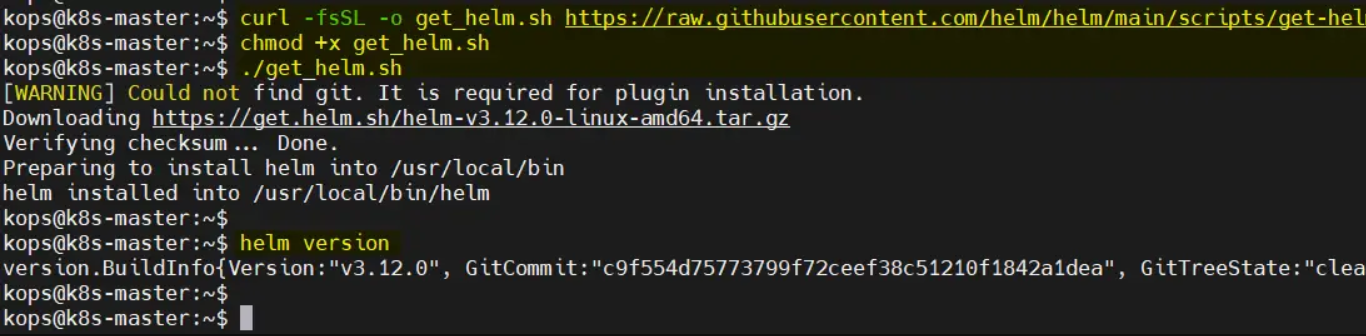
In case you, [helm](https://www.linuxtechi.com/install-use-helm-in-kubernetes/) is installed on your system then run beneath commands to install,

$ curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

$ chmod +x get\_helm.sh

$ ./get\_helm.sh

$ helm version



**Step 2: Install the AWX chart:**

The easiest way to install AWX on Kubernetes is by using the AWX Helm chart. So, to install AWX via chart, first add its repository using following helm command.

$ helm repo add awx-operator https://ansible.github.io/awx-operator/

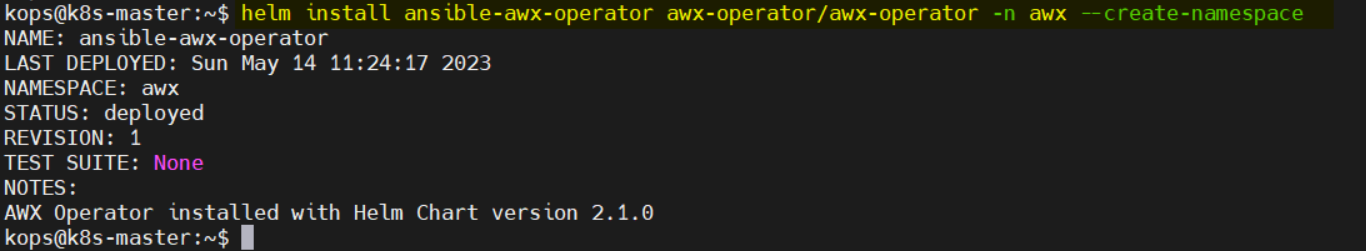
"awx-operator" has been added to your repositories

$

$ helm repo update

To install awx-operator via chart, run

$ helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace

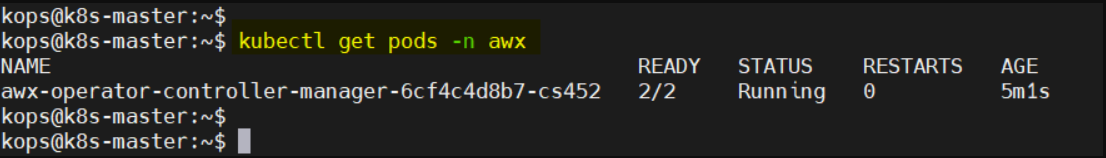
****

This will download the AWX chart and install it on your Kubernetes cluster in awx namespace.The installation process may take a few minutes,

**Step 3: Verify AWX operator installation:**

After the successful installation, you can verify AWX operator status by running below command

$ sudo kubectl get pods -n awx

****

**Step:4 Create PV, PVC and deploy AWX yaml file:**

AWX requires persistent volume for postgres pod. So, let’s first create a storage class for local volume, I am using local file system as persistent volume.

$ vi local-storage-class.yaml

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

  name: local-storage

 namespace: awx

provisioner: kubernetes.io/no-provisioner

volumeBindingMode: WaitForFirstConsumer

Save and close the file and then run

$ kubectl create -f local-storage-class.yaml

$ kubectl get sc -n awx

NAME            PROVISIONER                    RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION

local-storage   kubernetes.io/no-provisioner   Delete          WaitForFirstConsumer   false

$

Next create persistent volume(pv) using following pv.yaml file,

$ vi pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

  name: postgres-pv

  namespace: awx

spec:

  capacity:

    storage: 10Gi

  volumeMode: Filesystem

  accessModes:

  - ReadWriteOnce

  persistentVolumeReclaimPolicy: Delete

  storageClassName: local-storage

  local:

    path: /mnt/storage

  nodeAffinity:

    required:

      nodeSelectorTerms:

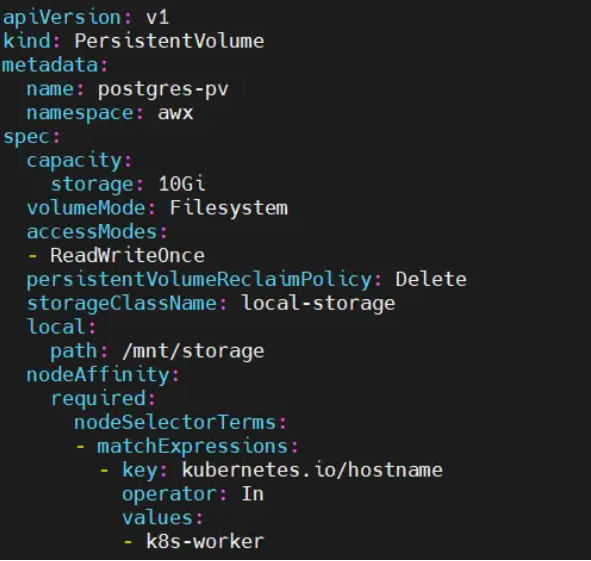
      - matchExpressions:

        - key: kubernetes.io/hostname

          operator: In

          values:

          - k8s-worker



Make sure folder “/mnt/storage” exists on worker node, if it does not exist then create it using mkdir command on worker node. In our case worker node is “k8s-worker”

Execute the beneath command to create postgres-pv in awx namespace.

$ kubectl create -f pv.yaml

Once pv is created successfully then create persistentvolumecliam using pvc.yaml file,

$ vi  pvc.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: postgres-13-ansible-awx-postgres-13-0

  namespace: awx

spec:

  storageClassName: local-storage

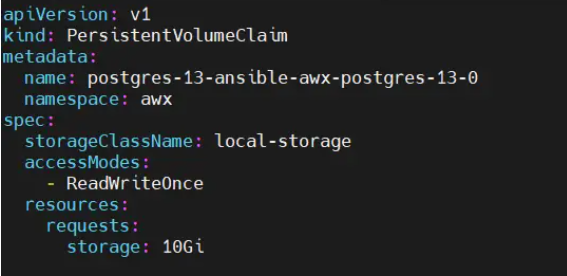
  accessModes:

    - ReadWriteOnce

  resources:

    requests:

      storage: 10Gi



To create pvc, run following kubectl command

$ kubectl create -f pvc.yaml

Verify the status of pv and pvc using beneath command

$ kubectl get pv,pvc -n awx

Now, we are all set to deploy AWX instance. Create an ansible-awx.yaml file with following content

$ vi ansible-awx.yaml

---

apiVersion: awx.ansible.com/v1beta1

kind: AWX

metadata:

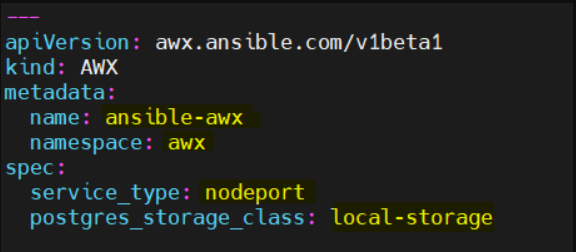
  name: ansible-awx

  namespace: awx

spec:

  service\_type: nodeport

  postgres\_storage\_class: local-storage



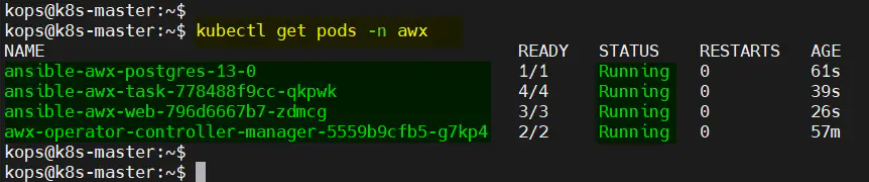
save and close the file.

Execute following kubectl command to deploy awx instance,

kubectl create -f ansible-awx.yaml

Wait for couple of minutes and then check pods status in awx namespace

kubectl get pods -n awx



**Step 5: Access AWX Web Interface:**

To access the AWX web interface, you need to create a service that exposes the awx-web deployment:

kubectl expose deployment ansible-awx-web --name ansible-awx-web-svc --type NodePort -n awx

This command will create a NodePort service that maps the AWX web container’s port to a port on the Kubernetes node. You can find the port number by running:

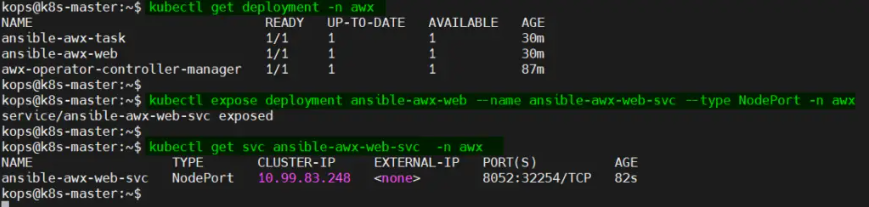
kubectl get svc ansible-awx-web-svc  -n awx

This will output something like this:

NAME              TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE

ansible-awx-web-svc   NodePort   10.99.83.248   <none>        8052:32254/TCP   82s

the web service is available on port 32254.



By default, the admin user is admin for web interface and the password is available in the <resourcename>-admin-password secret. To retrieve the admin password, run

$ kubectl get secrets -n awx | grep -i admin-password

ansible-awx-admin-password        Opaque               1      109m

$

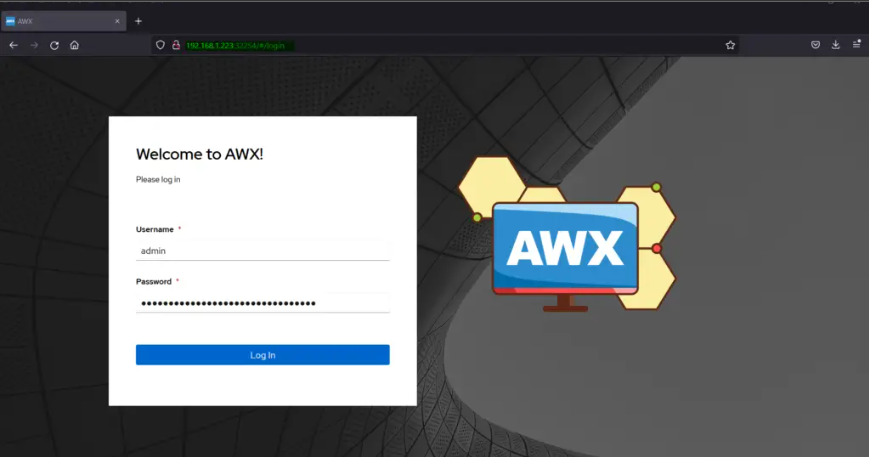
$ kubectl get secret ansible-awx-admin-password -o jsonpath="{.data.password}" -n awx | base64 --decode ; echo

l9mWcIOXQhSKnzZQyQQ9LZf3awDV0YMJ

$

You can now access the AWX web interface by opening a web browser and navigating to `http://<node-ip>:<node-port>/`. In the example above, the URL would be

<http://192.168.1.223:3225>



Click on Log In after entering the credentials.

