**High Availability AWX Cluster Setup**

**Node Roles:**

* **Control Plane Node (Node 1):** Runs the Kubernetes control plane components
* **Worker Node (Node 2):** Runs AWX workloads
* **Worker Node (Node 3):** Runs AWX workloads

**Prerequisites**

For a 3-node cluster, you will need:

• 3 servers meeting the system requirements (16GB RAM, 8 CPU cores, 40GB disk each)

• All prerequisites (docker, kubectl, kind) installed on all nodes

• Network connectivity between all nodes

• Shared storage (NFS or similar) for persistent data

**1. Install Prerequisites (Run on ALL Nodes)**

**Install Docker**

* sudo dnf config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>
* sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
* sudo systemctl enable --now docker
* sudo usermod -aG docker $USER
* sudo reboot

**Install kubectl**

* curl -LO https://dl.k8s.io/release/$(curl -L -s <https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl>
* sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

**Install kind**

* [ $(uname -m) = x86\_64 ] && curl -Lo ./kind <https://kind.sigs.k8s.io/dl/v0.27.0/kind-linux-amd64>
* chmod +x ./kind
* sudo mv ./kind /usr/local/bin/kind

**2. Control Plane Node (Node 1) Only**

**Create the Multi-Node KIND Cluster**

Create kind.yaml:

*apiVersion: kind.x-k8s.io/v1alpha4*

*kind: Cluster*

*nodes:*

*- role: control-plane*

*extraPortMappings:*

*- containerPort: 80*

*hostPort: 80*

*protocol: TCP*

*- containerPort: 443*

*hostPort: 443*

*protocol: TCP*

*- role: worker*

*- role: worker*

Then run:

* kind create cluster --config kind.yaml --name awx-cluster

**Get the Kubeconfig for Worker Nodes**

* kind get kubeconfig --name awx-cluster > kubeconfig.yaml

Copy this kubeconfig.yaml to **Node 2 & Node 3**.

**3. Worker Nodes (Node 2 & Node 3) Only**

**Join the KIND Cluster**

On each worker node, place the kubeconfig.yaml and run:

* export KUBECONFIG=./kubeconfig.yaml
* kubectl cluster-info --context kind-awx-cluster

**4. Control Plane Node (Node 1) Only**

**Install NGINX Ingress Controller (HA Mode)**

* wget <https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml>
* sed -i '/nodeSelector:/d' deploy.yaml # Remove nodeSelector for HA

Modify the deploy.yaml to include:

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: ingress-nginx-controller*

*namespace: ingress-nginx*

*spec:*

*replicas: 3*

*selector:*

*matchLabels:*

*app.kubernetes.io/name: ingress-nginx*

*template:*

*spec:*

*affinity:*

*podAntiAffinity:*

*requiredDuringSchedulingIgnoredDuringExecution:*

*- labelSelector:*

*matchExpressions:*

*- key: app.kubernetes.io/name*

*operator: In*

*values:*

*- ingress-nginx*

*topologyKey: kubernetes.io/hostname*

* kubectl apply -f deploy.yaml

**Install AWX Operator**

* kubectl create namespace awx
* kubectl config set-context --current --namespace=awx
* git clone <https://github.com/ansible/awx-operator.git>
* cd awx-operator
* git checkout tags/2.19.1

Create kustomization.yaml:

*apiVersion: kustomize.config.k8s.io/v1beta1*

*kind: Kustomization*

*resources:*

*- github.com/ansible/awx-operator/config/default?ref=2.19.1*

*images:*

*- name: quay.io/ansible/awx-operator*

*newTag: 2.19.1*

*namespace: awx*

Apply:

* kubectl apply -k .

**Deploy AWX in HA Mode**

Create awx-demo.yaml:

*apiVersion: awx.ansible.com/v1beta1*

*kind: AWX*

*metadata:*

*name: awx-demo*

*spec:*

*service\_type: ClusterIP*

*hostname: rocky.local.com*

*ingress\_type: ingress*

*ingress\_tls\_secret: awx-tls-secret*

*replicas: 3 # Runs 3 instances of AWX*

*postgres\_configuration\_secret: awx-postgres-config*

*postgres\_storage\_class: standard*

*postgres\_storage\_requirements:*

*requests:*

*storage: 8Gi*

*postgres\_extra\_args:*

*- "-c"*

*- "max\_connections=1000"*

*postgres\_priority\_class: high-priority*

*web\_resource\_requirements:*

*requests:*

*cpu: 1000m*

*memory: 2Gi*

*limits:*

*cpu: 2000m*

*memory: 4Gi*

*task\_resource\_requirements:*

*requests:*

*cpu: 1000m*

*memory: 2Gi*

*limits:*

*cpu: 2000m*

*memory: 4Gi*

Set Up PostgreSQL for High Availability

Create a separate PostgreSQL cluster:

*apiVersion: awx.ansible.com/v1beta1*

*kind: AWXPostgres*

*metadata:*

*name: awx-postgres*

*spec:*

*replicas: 3*

*storage\_class: standard*

*storage\_requirements:*

*requests:*

*storage: 8Gi*

**Configure Persistent Storage**

Set up a shared storage solution (NFS, Ceph, or cloud storage) and create PersistentVolumeClaims for AWX data.

**Deploy AWX**

* kubectl apply -f awx-demo.yaml

**Generate & Apply TLS Certificates**

* openssl req -x509 -nodes -newkey rsa:4096 -keyout awx.key -out awx.crt
* kubectl create secret tls awx-tls-secret --cert=awx.crt --key=awx.key --namespace=awx

**Load Balancing Considerations**

Configure an external load balancer to distribute traffic across all three nodes. You can use:

•   Cloud provider load balancer

•   HAProxy

•   Keepalived with Nginx

**5. Verify Deployment (Run on Any Node)**

**Check AWX Pods (Should be Running on All Nodes)**

* kubectl get pods -n awx -o wide

**Get Admin Password**

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

**Access AWX**

Open https://rocky.local.com in a browser.

**Summary: Which Commands Run Where?**

| **Step** | **Control Plane (Node 1)** | **Worker Nodes (Node 2 & 3)** |
| --- | --- | --- |
| Prerequisites (Docker, kubectl, kind) | ✅ | ✅ |
| KIND Cluster Setup | ✅ (kind create cluster) | ❌ |
| Kubeconfig Setup | ✅ (kind get kubeconfig) | ✅ (Copy & apply kubeconfig.yaml) |
| NGINX Ingress Install | ✅ | ❌ |
| AWX Operator Install | ✅ | ❌ |
| AWX Deployment | ✅ (kubectl apply -f awx-demo.yaml) | ❌ |
| TLS Setup | ✅ | ❌ |
| Verification | ✅ (kubectl get pods) | ✅ (kubectl get pods) |

This ensures a **highly available AWX cluster** running across 3 nodes.

**Scaling Considerations**

•   Monitor resource usage and adjust replica counts as needed

•   Consider separating the database to dedicated servers for larger deployments

•   Implement horizontal pod autoscaling for AWX components

This HA setup provides redundancy for all critical components and ensures AWX remains available even if one node fails.