

DevOps Project Ansible Awx (EKS, AKS, GKE and Multicloud-Multicluster)



By Ritesh Kumar Singh

Email Address: - riteshkumarsingh9559@gmail.com

LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/>

GitHub: - <https://github.com/singhritesh85>



या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाङ्गापहा ॥

DevOps Project Ansible Awx (EKS, AKS, GKE and Multicloud-Multicluster)

AWX is the open-source, community-driven upstream project of the commercial Red Hat Ansible Automation Platform (AAP) priorly known as Ansible Tower. For installation of Ansible Awx in production environment it is suggestable to install on the top of Kubernetes (EKS, AKS or GKE) and use PostgreSQL (RDS, Azure Database for PostgreSQL Flexible Server or Cloud SQL).



Module-1: - DevOps Project Ansible Awx (EKS)

In this module I will explain about installation of Ansible Awx on the top of Private EKS Cluster and RDS (PostgreSQL) had been used as the database. To access the Ansible Awx using the URL I had used Kubernetes Gateway API (while installation of Kubernetes Gateway API, Wildcard SSL Certificate had been used). AWS Route53 Hosted Zone had been created using the Terraform Script present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path `terraform-route53-hosted-zone`.

Commands as written below had been used to create infrastructure using Terraform.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

After running the Terraform Script update your domain provider with the generated Nameserver of the AWS Route53 Hosted Zone.

```

+ name                  = "singhritesh85.com"
+ name_servers          = (known after apply)
+ primary_name_server   = (known after apply)
+ tags                  = {
    + "Environment" = "dev"
}
+ tags_all              = {
    + "Environment" = "dev"
}
+ zone_id               = (known after apply)

}

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ route53_hosted_zone_details = {
    + hosted_zone_id      = (known after apply)
    + hosted_zone_name_servers = (known after apply)
}
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Creating...
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Still creating... [00m10s elapsed]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Still creating... [00m20s elapsed]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Still creating... [00m30s elapsed]
module.route53_hosted_zone.aws_route53_zone.hosted_zone: Creation complete after 32s [id=██████████]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

route53_hosted_zone_details = {
  "hosted_zone_id" = "██████████"
  "hosted_zone_name_servers" = tolist([
    "██████████",
    "██████████",
    "██████████",
    "██████████",
  ])
}

```

Then Private EKS Cluster, a K8S-Management Node and RDS (PostgreSQL) had been created using the Terraform Script present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **terraform-private-eks-cluster-and-rds**.

Commands as written below had been used to create infrastructure using Terraform.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

```

module.eks_cluster.aws_ec2_tag.cluster_security_group: Creation complete after 1s [id=sg-0a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0u1v2w3x4y5z6]
module.eks_cluster.null_resource.kubectl (local-exec): Updated context arn:aws:eks:us-east-1:123456789012:cluster/eks-demo-cluster-dev in /root/.kube/config
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.core_dns: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.core_dns: Creation complete after 14s [id=eks-demo-cluster-dev:coredns]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Creation complete after 45s [id=eks-demo-cluster-dev:metrics-server]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Creation complete after 45s [id=eks-demo-cluster-dev:snapshot-controller]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Creation complete after 46s [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]

Apply complete! Resources: 80 added, 0 changed, 0 destroyed.

Outputs:

ec2_eks_and_karpenter_iam_role_rds_details = {
  "eks_cluster_endpoint" = "https://[REDACTED].us-east-2.eks.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "k8s_management_instance_id" = "i-01234567890abcdef"
  "k8s_management_private_ip" = "10.10.4.195"
  "karpenter_node_spun_iam_role_arn" = "arn:aws:iam::[REDACTED]:role/karpenter-eks-noderole"
  "rds_dbinstance_address" = "dbinstance-1.[REDACTED].us-east-2.rds.amazonaws.com"
  "rds_dbinstance_endpoint" = "dbinstance-1.[REDACTED].us-east-2.rds.amazonaws.com:5432"
}

```

```
[k8s-management@k8s-management ~]$ kubectl get nodes
NAME                               STATUS   ROLES      AGE   VERSION
ip-10-10-1-119.us-east-2.compute.internal   Ready    <none>   1m    v1.33.0-eks-[REDACTED]
ip-10-10-2-97.us-east-2.compute.internal   Ready    <none>   1m    v1.33.0-eks-[REDACTED]
ip-10-10-3-114.us-east-2.compute.internal   Ready    <none>   1m    v1.33.0-eks-[REDACTED]
```

To install karpenter I edited the configmap **aws-auth** in the namespace **kube-system** as shown in the screenshot present below.

```
[k8s-management@k8s-management ~]$ kubectl edit cm aws-auth -n kube-system

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::023456789012:role/eks-nodegroup-role-dev
      groups:
        - system:bootstrappers
        - system:nodes
      username: system:node:{{EC2PrivateDNSName}}
    - rolearn: arn:aws:iam::023456789012:role/karpenter-eks-noderole
      groups:
        - system:bootstrappers
        - system:nodes
      username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2026-01-11T10:00:00Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1"
  uid: [REDACTED]
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT (paste) --
```

I installed the carpenter controller using helm as shown in the screenshot attached below.

```
helm upgrade --install carpenter oci://public.ecr.aws/karpenter/karpenter --version "1.6.5" --namespace "karpenter" --create-namespace --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks.amazonaws.com/role-arn=arn:aws:iam::02XXXXXXXXX6:role/karpenter-controller-role" --set controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set controller.resources.limits.cpu=1 --set controller.resources.limits.memory=1Gi
```

[k8s-management@k8s-management ~]\$ helm upgrade --install carpenter oci://public.ecr.aws/karpenter/karpenter --version "1.6.5" --namespace "karpenter" --create-namespace --set "settings.clusterName=eks-demo-cluster-dev" --set "serviceAccount.annotations.eks.amazonaws.com/role-arn=arn:aws:iam::02XXXXXXXXX6:role/karpenter-controller-role" --set controller.resources.requests.cpu=1 --set controller.resources.requests.memory=1Gi --set controller.resources.limits.cpu=1 --set controller.resources.limits.memory=1Gi

Then ran the kubernetes Manifests file **karpenter.yaml** present at the path **terraform-private-eks-cluster-and-rds** in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git>. For your reference **karpenter.yaml** I had also provided below.

```
[k8s-management@k8s-management ~]$ kubectl apply -f karpenter.yaml
nodepool.karpenter.sh/general-purpose created
ec2nodeclass.karpenter.k8s.aws/default created

[k8s-management@k8s-management ~]$ kubectl get all -n karpenter
NAME                               READY   STATUS    RESTARTS   AGE
pod/karpenter-[REDACTED]           1/1     Running   0          10m
pod/karpenter-[REDACTED]           1/1     Running   0          10m

NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/karpenter   ClusterIP   172.20.100.104  <none>        8080/TCP   10m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/karpenter   2/2     2           2           10m

NAME                           DESIRED   CURRENT   READY   AGE
replicaset.apps/karpenter-[REDACTED]  2         2         2         10m
```

```

cat karpenter.yaml

# This example NodePool will provision general purpose instances

---

apiVersion: karpenter.sh/v1

kind: NodePool

metadata:

  name: general-purpose

  annotations:

    kubernetes.io/description: "General purpose NodePool for generic workloads"

spec:

template:

  spec:

    requirements:

      - key: kubernetes.io/arch

        operator: In

        values: ["amd64"]

      - key: kubernetes.io/os

        operator: In

        values: ["linux"]

      - key: karpenter.sh/capacity-type

        operator: In

        values: ["on-demand"] #["spot"] ### You can select on-demand or spot instance depending
on your requirement.

      - key: karpenter.k8s.aws/instance-category

        operator: In

        values: ["t"] # Interested to launch t series instance      #["c", "m", "r"]

      - key: karpenter.k8s.aws/instance-generation

        operator: Gt

        values: ["1"] # Instances launched will be greater than 1 ###["2"] # Instances launched will
be greater than 2

nodeClassRef:

  group: karpenter.k8s.aws

```

```

kind: EC2NodeClass
name: default

---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: default
  annotations:
    kubernetes.io/description: "General purpose EC2NodeClass for running Amazon Linux 2023 nodes"
spec:
  role: "karpenter-eks-noderole" # replace with your karpenter noderole
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "eks-demo-cluster-dev" # replace with your cluster name
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "eks-demo-cluster-dev" # replace with your cluster name
  amiSelectorTerms:
    - alias: al2023@latest # Amazon Linux 2023

```

Installation of AWX using Helm

```

helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
helm repo update
helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace

```

```

[k8s-management@k8s-management ~]$ helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
"awx-operator" has been added to your repositories
[k8s-management@k8s-management ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "awx-operator" chart repository
Update Complete. ⚡Happy Helm-ing!⚡
[k8s-management@k8s-management ~]$ helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace

```

```
[k8s-management@k8s-management ~]$ cat awx-postgres-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "dbinstance-1.████████.us-east-2.rds.amazonaws.com"
  port: "5432"
  database: "dexter"
  username: "postgres"
  password: "I████████W"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
[k8s-management@k8s-management ~]$ kubectl apply -f awx-postgres-secret.yaml
secret/awx-postgres-configuration created
```

```
[k8s-management@k8s-management ~]$ kubectl get secrets -n awx|grep -i "awx-postgres-configuration"
awx-postgres-configuration          Opaque      7    48s
```

```
[k8s-management@k8s-management ~]$ cat ansible-awx.yaml
apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
[k8s-management@k8s-management ~]$ kubectl apply -f ansible-awx.yaml
awx.awx.ansible.com/ansible-awx created
```

```
[k8s-management@k8s-management ~]$ kubectl get all -n awx
NAME                                         READY   STATUS    RESTARTS   AGE
pod/ansible-awx-migration-24.6.1-████████     0/1     Completed  0          6m25s
pod/ansible-awx-task-████████                  4/4     Running   0          8m
pod/ansible-awx-web-████████                   3/3     Running   0          8m2s
pod/awx-operator-controller-manager-████████     2/2     Running   0          17m

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/ansible-awx-service   ClusterIP   172.20.78.211  <none>       80/TCP    8m7s
service/awx-operator-controller-manager-metrics-service   ClusterIP   172.20.195.101 <none>       8443/TCP  17m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ansible-awx-task   1/1     1           1           8m
deployment.apps/ansible-awx-web    1/1     1           1           8m2s
deployment.apps/awx-operator-controller-manager  1/1     1           1           17m

NAME                           DESIRED  CURRENT  READY   AGE
replicaset.apps/ansible-awx-task-████████     1        1        1        8m
replicaset.apps/ansible-awx-web-████████       1        1        1        8m2s
replicaset.apps/awx-operator-controller-manager-████████  1        1        1        17m

NAME                           STATUS  COMPLETIONS  DURATION   AGE
job.batch/ansible-awx-migration-24.6.1  Complete  1/1        4m51s    6m26s
```

```
cat awx-postgres-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "dbinstance-1.XXXXXXXXXXXXXXX.us-east-2.rds.amazonaws.com"
  port: "5432"
  database: "dexter"
  username: "postgres"
  password: "XXXXXXXXXXXXXXXXXX"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
```

```
cat ansible-awx.yaml

apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
```

Then I created the Gateway API and did the entry of created LoadBalancer Service DNS to create the Record-Set of A-Type with Alias as shown in the screenshot attached below.

```
[k8s-management@k8s-management ~]$ ls
STAR_singhritesh85_com.crt ansible-awx.yaml awx-postgres-secret.yaml karpenter.yaml mykey.key
[k8s-management@k8s-management ~]$ kubectl -n awx create secret tls tls-awx-ingress --cert=STAR_singhritesh85_com.crt --key=mykey.key
secret/tls-awx-ingress created
[k8s-management@k8s-management ~]$ kubectl get secrets -n awx|grep "tls-awx-ingress"
                           kubernetes.io/tls   2      25s
```

Installation of Gateway API as explained below.

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml
```

```
helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway
```

```
[k8s-management@k8s-management ~]$ kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml
customresourcedefinition.apiextensions.k8s.io/backendlspolicies.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/gatewayclasses.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/gateways.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/grpcroutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/httproutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/referencegrants.gateway.networking.k8s.io created
[k8s-management@k8s-management ~]$ helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway
Release "ngf" does not exist. Installing it now.
Pulled: ghcr.io/nginx/charts/nginx-gateway-fabric:2.3.0
Digest: sha256:XXXXXXXXXX
NAME: ngf
LAST DEPLOYED: 2026-01-11T11:26:26Z
NAMESPACE: nginx-gateway
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

```
[k8s-management@k8s-management ~]$ kubectl get all -n nginx-gateway
NAME                                         READY   STATUS    RESTARTS   AGE
pod/ngf-nginx-gateway-fabric-XXXXXXXXXX   1/1     Running   0          68s

NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/ngf-nginx-gateway-fabric   ClusterIP  172.20.214.211 <none>       443/TCP   68s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ngf-nginx-gateway-fabric  1/1     1           1          68s

NAME                               DESIRED  CURRENT   READY   AGE
replicaset.apps/ngf-nginx-gateway-fabric-XXXXXXXXXX  1        1         1        68s
```

```
[k8s-management@k8s-management ~]$ kubectl apply -f gateway-api.yaml
gateway.gateway.networking.k8s.io/awx-gateway created
httproute.gateway.networking.k8s.io/awx-http-to-https-redirect created
httproute.gateway.networking.k8s.io/awx-route-https created
[k8s-management@k8s-management ~]$ kubectl get all -n awx
```

```
[k8s-management@k8s-management ~]$ kubectl get all -n awx
NAME                                         READY   STATUS    RESTARTS   AGE
pod/ansible-awx-migration-24.6.1-XXXXXXXXXX   0/1     Completed   0          27m
pod/ansible-awx-task-XXXXXXXXXX           4/4     Running   0          28m
pod/ansible-awx-web-XXXXXXXXXX           3/3     Running   0          28m
pod/awx-gateway-nginx-XXXXXXXXXX           1/1     Running   0          19s
pod/awx-operator-controller-manager-XXXXXXXXXX   2/2     Running   0          38m

NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP
service/ansible-awx-service          ClusterIP  172.20.78.211 <none>
service/awx-gateway-nginx            LoadBalancer 172.20.124.82 XXXXXXXXXX.us-east-2.elb.amazonaws.com
                                         80:32453/TCP,443:32200/TCP  20s
service/awx-operator-controller-metrics-service ClusterIP  172.20.195.101 <none>

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ansible-awx-task   1/1     1           1          28m
deployment.apps/ansible-awx-web    1/1     1           1          28m
deployment.apps/awx-gateway-nginx  1/1     1           1          20s
deployment.apps/awx-operator-controller-manager  1/1     1           1          38m

NAME                               DESIRED  CURRENT   READY   AGE
replicaset.apps/ansible-awx-task-XXXXXXXXXX  1        1         1        28m
replicaset.apps/ansible-awx-web-XXXXXXXXXX  1        1         1        28m
replicaset.apps/awx-gateway-nginx-XXXXXXXXXX  1        1         1        20s
replicaset.apps/awx-operator-controller-manager-XXXXXXXXXX  1        1         1        38m

NAME          STATUS    COMPLETIONS   DURATION   AGE
job.batch/ansible-awx-migration-24.6.1 Complete  1/1        4m51s   27m
```

```
cat gateway-api.yaml

apiVersion: gateway.networking.k8s.io/v1beta1

kind: Gateway

metadata:

  name: awx-gateway

  namespace: awx

spec:

  gatewayClassName: nginx # e.g., "nginx"

  listeners:

    - name: http

      protocol: HTTP

      port: 80

      hostname: "ansible-awx.singhritesh85.com"

    - name: https

      protocol: HTTPS

      port: 443

      hostname: "ansible-awx.singhritesh85.com"

  tls:

    mode: Terminate

    certificateRefs:

      - kind: Secret

        name: tls-awx-ingress

---

apiVersion: gateway.networking.k8s.io/v1

kind: HTTPRoute

metadata:

  name: awx-http-to-https-redirect

  namespace: awx

spec:

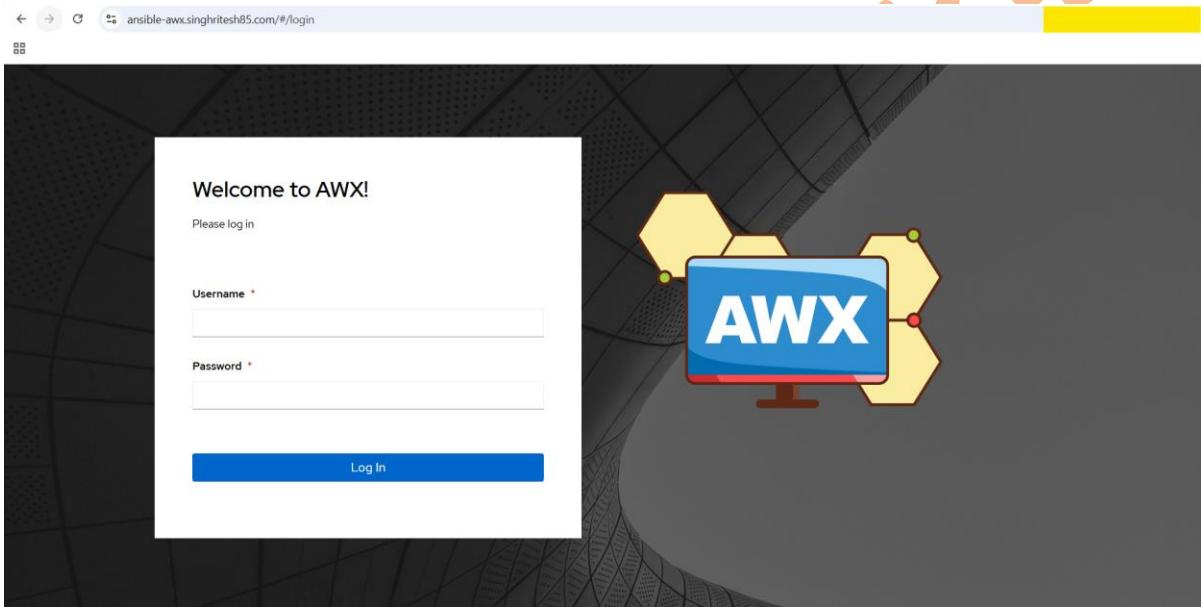
  parentRefs:

    - name: awx-gateway
```

```
namespace: awx
sectionName: http
rules:
- filters:
  - type: RequestRedirect
    requestRedirect:
      scheme: https
      statusCode: 301 # Use 301 for permanent redirect
---
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: awx-route-https
  namespace: awx
spec:
  parentRefs:
  - name: awx-gateway
    namespace: awx
    sectionName: https
  hostnames:
  - "ansible-awx.singhritesh85.com"
  rules:
  - backendRefs:
    - name: ansible-awx-service # Name of the Service created by the Helm chart
      port: 80
```

The screenshot shows the 'Create record' page in the AWS Route 53 console. A new record named 'Record 1' is being created for the domain '.singhritesh85.com'. The record type is set to 'A' (IPv4), which routes traffic to an IP address and some AWS resources. The target is an Application Load Balancer (ALB) named 'dualstack.ansible-awx.us-east-2.elb.amazonaws.com' located in the US East (Ohio) region. The routing policy is set to 'Simple routing'. There is also an option to evaluate target health, which is turned off.

Finally, I was able to access Ansible Awx using the URL as shown in the screenshot attached below.



Now, I will reset the Ansible-Awx **admin** user **password** using the procedure as explained below.

kubectl get pods -n awx

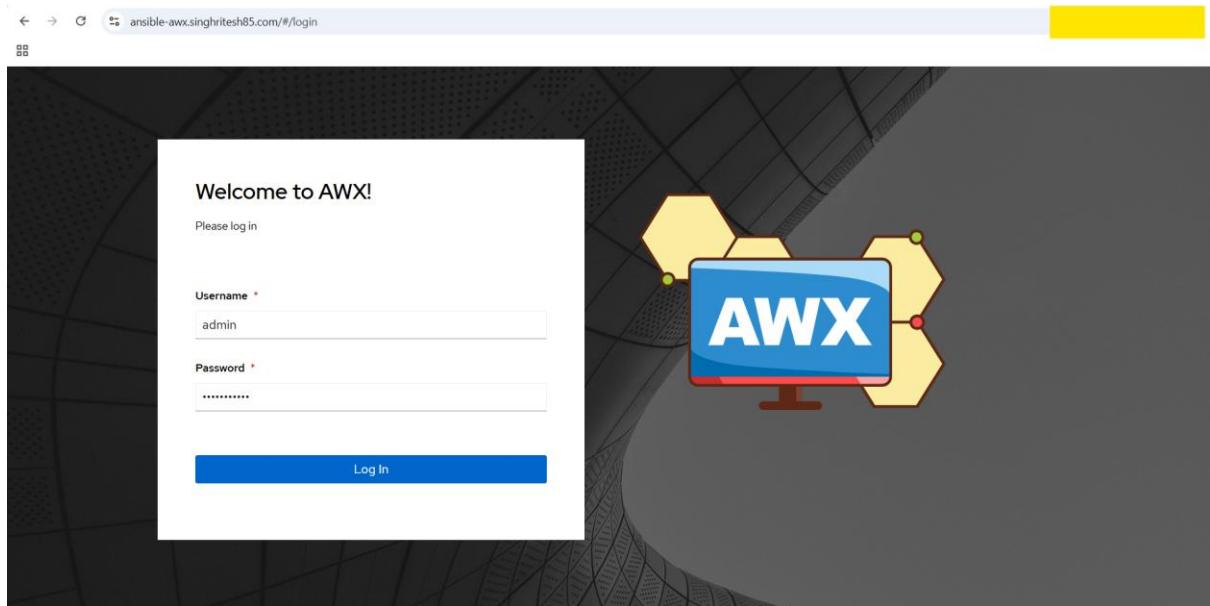
Log-into the ansible-awx-web Pod as shown below

kubectl exec -it <ansible-awx-web pod> -n awx -- bash

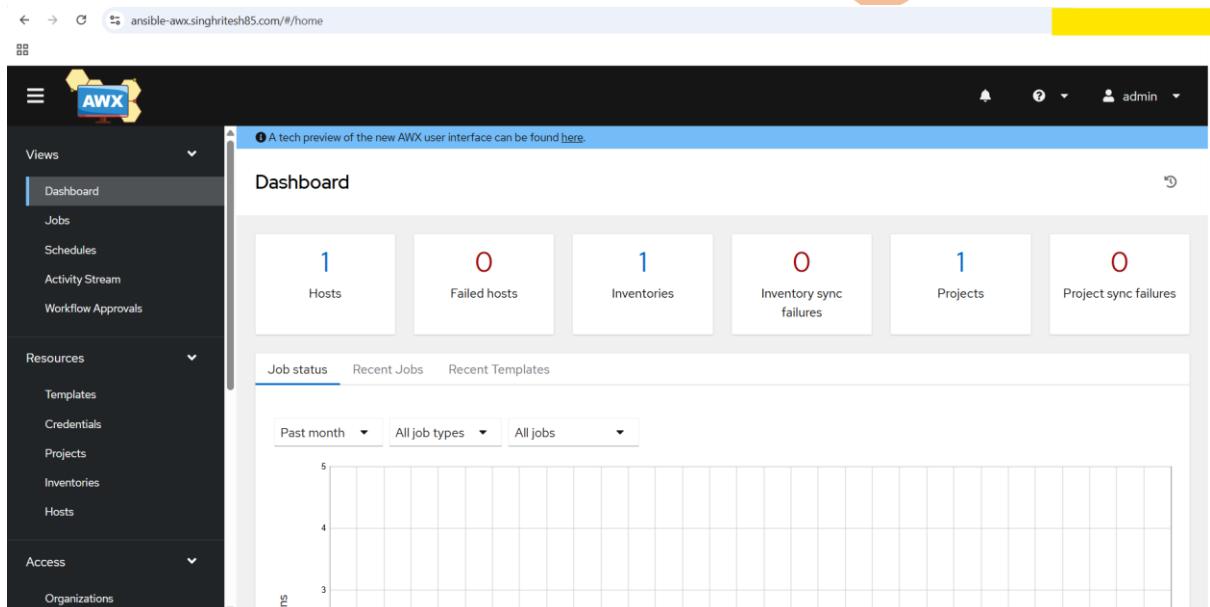
awx-manage update_password --username=admin --password=<provide-your-password-here>

```
[k8s-management@k8s-management ~]$ kubectl get pods -n awx
NAME                               READY   STATUS    RESTARTS   AGE
ansible-awx-migration-24.6.1-[REDACTED]   0/1     Completed  0          99m
ansible-awx-task-[REDACTED]                4/4     Running   0          100m
ansible-awx-web-[REDACTED]                 3/3     Running   0          100m
awx-gateway-nginx-[REDACTED]              1/1     Running   0          72m
awx-operator-controller-manager-[REDACTED]  2/2     Running   0          110m
[k8s-management@k8s-management ~]$ kubectl exec -it ansible-awx-web-[REDACTED] -n awx -- bash
bash-5.1$ awx-manage update_password --username=admin --password=[REDACTED]
Password updated
```

Then, I logged-into the Ansible-Awx using the updated password as shown below.

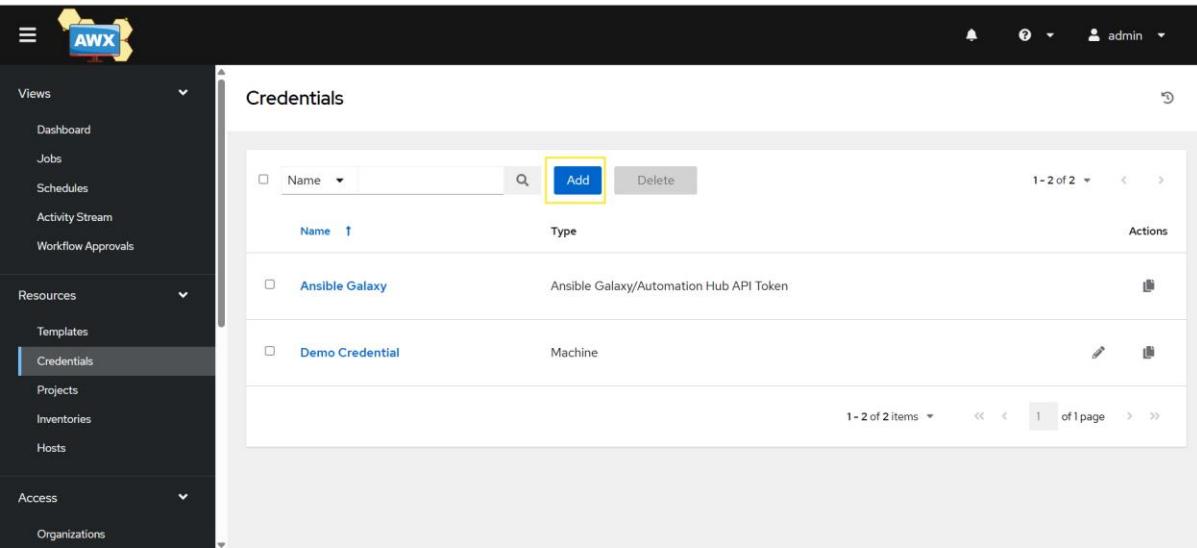


The screenshot shows the Ansible-Awx login interface. The URL in the browser is `ansible-awx.singhritesh85.com/#/login`. The form has 'Username' set to 'admin' and 'Password' set to '*****'. A large blue 'Log In' button is at the bottom. To the right of the form is a decorative graphic featuring the letters 'AWX' on a screen surrounded by yellow hexagons.

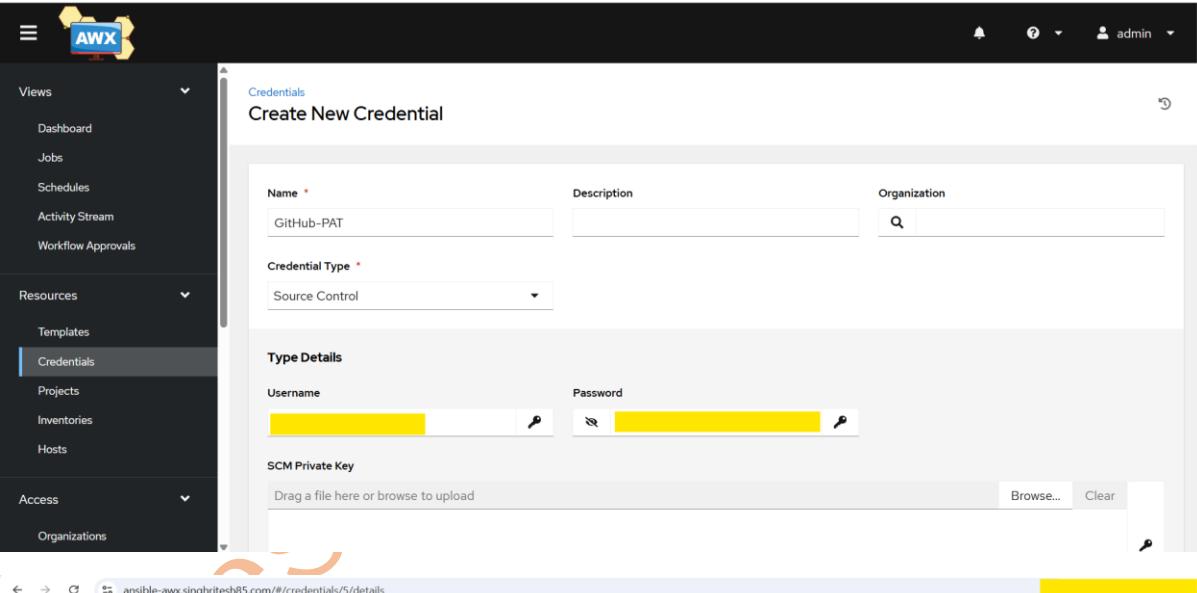


The screenshot shows the Ansible-Awx dashboard. The URL in the browser is `ansible-awx.singhritesh85.com/#/home`. The dashboard includes a sidebar with 'Views' (Dashboard selected), 'Resources' (Templates, Credentials, Projects, Inventories, Hosts), and 'Access' (Organizations). The main area displays a summary of system status: 1 Hosts, 0 Failed hosts, 1 Inventories, 0 Inventory sync failures, 1 Projects, and 0 Project sync failures. Below this is a chart showing job status over the past month. A message at the top says 'A tech preview of the new AWX user interface can be found [here](#)'.

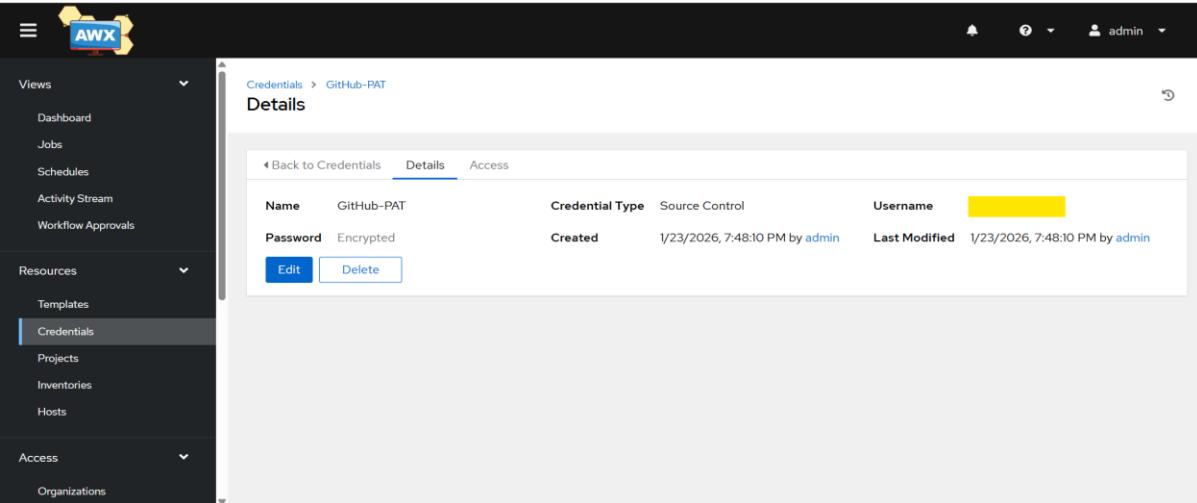
Now, I will run the Ansible Playbook using Ansible-Awx and will try to know the uptime of a server which comes under the group dexter. First, create the Credentials to access GitHub Repo and Login into the Server, inventory and host group and then add the hosts. Then create the project and finally, template as explained below.



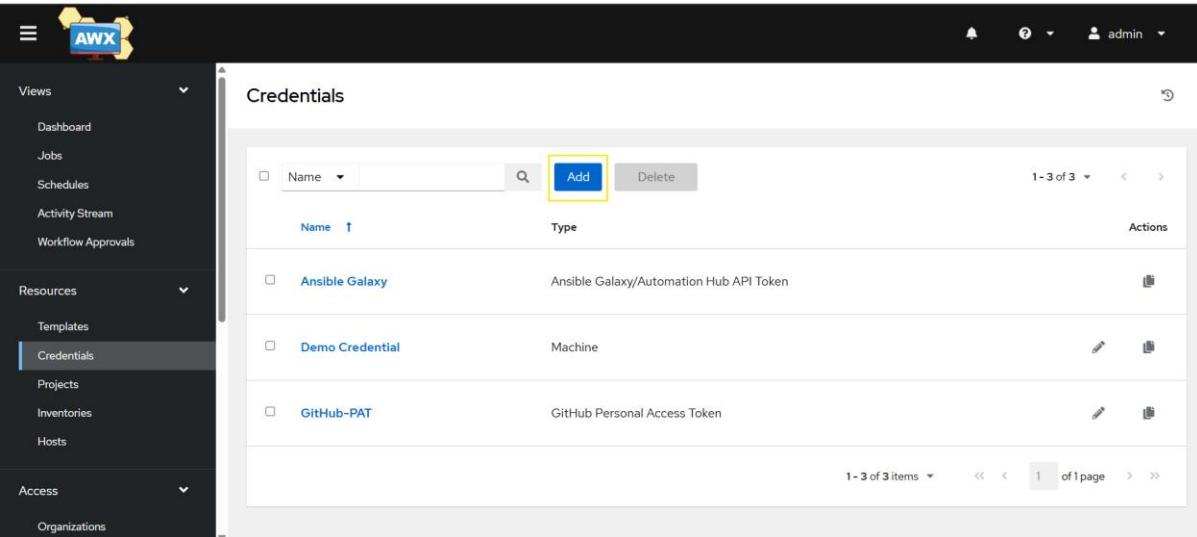
The screenshot shows the AWX interface with the URL ansible-awx.singhritesh85.com/#/credentials. The left sidebar is open, showing 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), 'Resources' (Templates, Credentials, Projects, Inventories, Hosts), and 'Access' (Organizations). The 'Credentials' option under Resources is selected. The main panel displays a table titled 'Credentials' with two items: 'Ansible Galaxy' (Ansible Galaxy/Automation Hub API Token) and 'Demo Credential' (Machine). A yellow box highlights the 'Add' button at the top right of the table header.



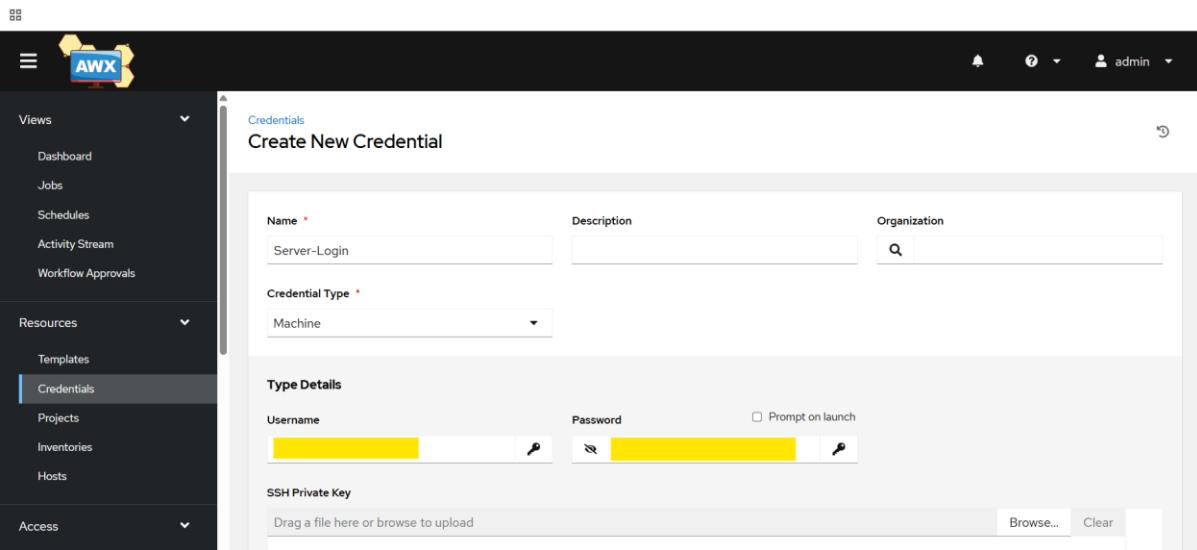
The screenshot shows the AWX interface with the URL ansible-awx.singhritesh85.com/#/credentials/add. The left sidebar is open, showing 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), 'Resources' (Templates, Credentials, Projects, Inventories, Hosts), and 'Access' (Organizations). The 'Credentials' option under Resources is selected. The main panel is titled 'Create New Credential'. It has fields for 'Name' (GitHub-PAT), 'Description', 'Organization', 'Credential Type' (Source Control), 'Username' and 'Password' (both masked), and a section for 'Type Details' with 'SCM Private Key' (input field with placeholder 'Drag a file here or browse to upload' and 'Browse...', 'Clear' buttons).



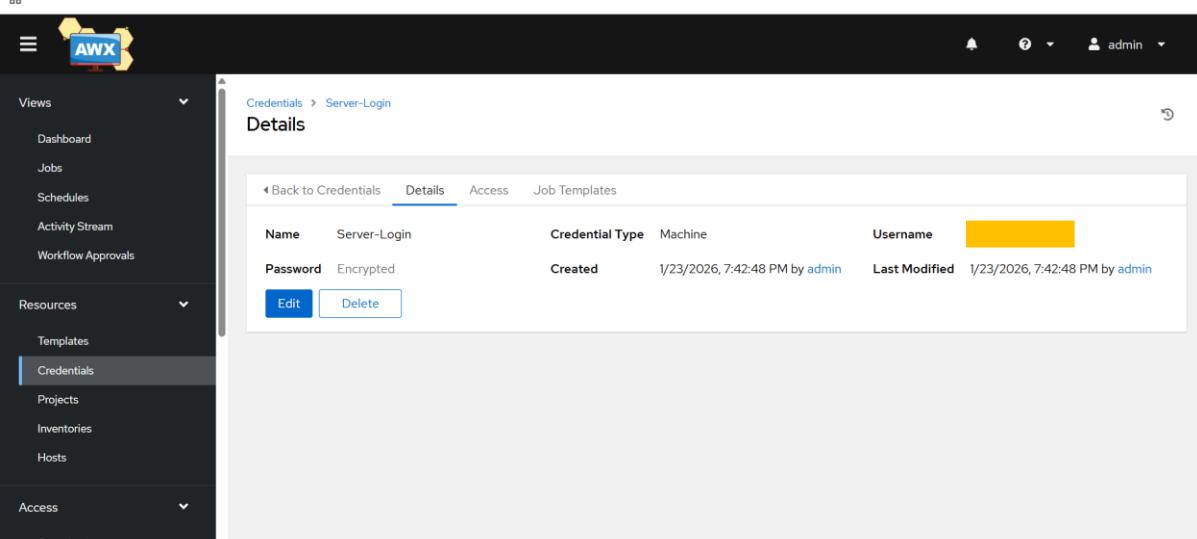
The screenshot shows the AWX interface with the URL ansible-awx.singhritesh85.com/#/credentials/5/details. The left sidebar is open, showing 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), 'Resources' (Templates, Credentials, Projects, Inventories, Hosts), and 'Access' (Organizations). The 'Credentials' option under Resources is selected. The main panel is titled 'Details' for the 'GitHub-PAT' credential. It shows the 'Name' (GitHub-PAT), 'Credential Type' (Source Control), 'Created' (1/23/2026, 7:48:10 PM by admin), and 'Last Modified' (1/23/2026, 7:48:10 PM by admin). There are 'Edit' and 'Delete' buttons at the bottom.



The screenshot shows the AWX web interface with the URL <https://ansible-awx.singhritesh85.com/#/credentials>. The left sidebar is open, showing 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), 'Resources' (Templates, Credentials, Projects, Inventories, Hosts), and 'Access' (Organizations). The 'Credentials' option under Resources is selected. The main panel displays a table titled 'Credentials' with three items: 'Ansible Galaxy' (Ansible Galaxy/Automation Hub API Token), 'Demo Credential' (Machine), and 'GitHub-PAT' (GitHub Personal Access Token). The 'Add' button is highlighted with a yellow box.



The screenshot shows the 'Create New Credential' form with the URL <https://ansible-awx.singhritesh85.com/#/credentials/add>. The left sidebar is identical to the previous screenshot. The main panel has a title 'Create New Credential'. It includes fields for 'Name' (Server-Login), 'Description', 'Organization', 'Credential Type' (Machine), 'Username' (redacted), 'Password' (redacted), and a checkbox for 'Prompt on launch'. Below these are sections for 'Type Details' and 'SSH Private Key' (with a file upload input).



The screenshot shows the 'Details' view for the 'Server-Login' credential with the URL <https://ansible-awx.singhritesh85.com/#/credentials/4/details>. The left sidebar is identical. The main panel shows the credential details: Name (Server-Login), Credential Type (Machine), Username (redacted), Password (Encrypted), Created (1/23/2026, 7:42:48 PM by admin), and Last Modified (1/23/2026, 7:42:48 PM by admin). There are 'Edit' and 'Delete' buttons at the bottom.

The image consists of three vertically stacked screenshots of the AWX web interface, showing the process of creating a new inventory.

Screenshot 1: Inventories List

- The top screenshot shows the 'Inventories' page. A search bar and an 'Add' button are highlighted with yellow boxes. The table lists one item: 'Demo Inventory' (Type: Inventory, Organization: Default).

Screenshot 2: Create New Inventory

- The middle screenshot shows the 'Create new inventory' form. It includes fields for 'Name' (dexter), 'Description', and 'Organization' (Default). Below these are sections for 'Instance Groups' (with a search bar) and 'Labels' (with a dropdown menu). Under 'Options', there is a checkbox for 'Prevent Instance Group Fallback'. The 'Variables' section shows a single entry labeled '1'.

Screenshot 3: Inventory Details - Groups

- The bottom screenshot shows the 'Groups' tab for the 'dexter' inventory. An 'Add' button is highlighted with a yellow box. The message 'No Items Found' is displayed, along with a note: 'Please add Items to populate this list'.

The image consists of three vertically stacked screenshots of the AWX web interface, showing the process of creating a host group and then adding hosts to it.

Screenshot 1: Create new group

- URL:** <https://ansible-awx.singhritesh85.com/#/inventories/inventory/2/groups/add>
- Description:** A modal window titled "Create new group" is open. It has fields for "Name" (set to "dexter") and "Description". Below these are tabs for "Variables" (YAML selected) and "JSON". A large text area contains YAML code:

```
1 ---
```

. At the bottom are "Save" and "Cancel" buttons.

Screenshot 2: Hosts

- URL:** https://ansible-awx.singhritesh85.com/#/inventories/inventory/2/groups/1/nested_hosts
- Description:** A list titled "Hosts" under the "Groups > dexter > dexter" path. It shows a search bar, an "Add" button (highlighted with a yellow box), and a "Run Command" and "Disassociate" button. Below the search bar are buttons for "Add existing host" and "Add new host" (also highlighted with a yellow box). A message "No Hosts Found" is displayed with a note "Please add Hosts to populate this list".

Screenshot 3: Create new host

- URL:** https://ansible-awx.singhritesh85.com/#/inventories/inventory/2/groups/1/nested_hosts/add
- Description:** A modal window titled "Create new host" under the "Groups > dexter > dexter > Hosts" path. It has fields for "Name" (set to "3 [REDACTED] 97") and "Description". Below these are tabs for "Variables" (YAML selected) and "JSON". A large text area contains YAML code:

```
1 ---  
2 ---
```

. At the bottom are "Save" and "Cancel" buttons.

ansible-awx.singhritesh85.com/#/projects

Views

- Dashboard
- Jobs
- Schedules
- Activity Stream
- Workflow Approvals

Resources

- Templates
- Credentials
- Projects**
- Inventories
- Hosts

Access

- Organizations

Projects

Name	Status	Type	Revision	Actions
Demo Project	Successful	Git	347e44f	

1-1 of 1 items << < > >>

ansible-awx.singhritesh85.com/#/projects/add

Views

- Dashboard
- Jobs
- Schedules
- Activity Stream
- Workflow Approvals

Resources

- Templates
- Credentials
- Projects**
- Inventories
- Hosts

Access

- Organizations

Create New Project

Name * project-1 Description Organization * Default

Execution Environment Git Source Control Type * Content Signature Validation Credential

Type Details

Source Control URL * https://github.com/singhritesh85/webserver-u... Source Control Branch/Tag/Commit Source Control Refspec

Source Control Credential GitHub-PAT

Options

ansible-awx.singhritesh85.com/#/templates

Views

- Dashboard
- Jobs
- Schedules
- Activity Stream
- Workflow Approvals

Resources

- Templates**
- Credentials
- Projects
- Inventories
- Hosts

Templates

Name	Type	Organization	Last Ran	Actions
Demo Job Template	Job Template	Default		

1-1 of 1 items << < > >>

Views

- Dashboard
- Jobs
- Schedules
- Activity Stream
- Workflow Approvals

Resources

- Templates**
- Credentials
- Projects
- Inventories
- Hosts

Access

- Organizations

Name * template-1

Description

Job Type * Run

Inventory * dexter

Project * project-1

Execution Environment

Playbook * server-uptime.yaml

Credentials SSH: Server-Login

Labels

Finally, Run the Template as shown below.

Name	Type	Organization	Last Ran	Actions
Demo Job Template	Job Template	Default		
template-1	Job Template	Default		

Jobs > 2 - template-1

Output

template-1 Successful

Plays 1 Tasks 3 Hosts 1 Elapsed 00:02:05

Stdout

```

14 IASK [to know the uptime of server] ****
15 changed: [3.████.97]
16
17 TASK [Print the uptime] ****
18 ok: [3.████.97] => {
19     "msg": "The uptime of 3.████.97 is █████ up 9:31, 6 users, load average: 0.00, 0.00, 0.00"
20 }
21
22 PLAY RECAP ****
23 3.████.97 : ok=3    changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

Module-2: - DevOps Project Ansible Awx (AKS)

In this module I will explain about installation of Ansible Awx on the top of Private AKS Cluster and Azure Database for PostgreSQL Flexible Servers had been used as the database. To access the Ansible Awx using the URL I had used Azure Application Gateway Ingress Controller (for Kubernetes Secrets of tls, Wildcard SSL Certificate had been used). A DNS Zone had been created in Azure DNS Zone using the Terraform Script present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **terraform-azure-dns-zone**.

Commands as written below had been used to create infrastructure using Terraform.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** -----> Finally, Create the resources.

After running the Terraform Script update your domain provider with the generated Nameserver of the DNS Zone of Azure DNS Zone.

```

}
# module.dns.azurerm_resource_group.dns_rg will be created
+ resource "azurerm_resource_group" "dns_rg" {
  + id      = (known after apply)
  + location = "eastus"
  + name    = "dns-resource-group"
}

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ azure_dns_zone_name_and_nameserver = {
  + dns_zone_name = "singhritesh85.com"
  + name_servers = (known after apply)
}
module.dns.azurerm_resource_group.dns_rg: Creating...
module.dns.azurerm_resource_group.dns_rg: Still creating... [00m10s elapsed]
module.dns.azurerm_resource_group.dns_rg: Still creating... [00m20s elapsed]
module.dns.azurerm_resource_group.dns_rg: Creation complete after 23s [id=/subscriptions/[REDACTED]/resourceGroups/dns-resource-group]
module.dns.azurerm_dns_zone.dns_zone: Creating...
module.dns.azurerm_dns_zone.dns_zone: Creation complete after 5s [id=/subscriptions/[REDACTED]/resourceGroups/dns-resource-group/providers/Microsoft.Network/dnsZones/singhritesh85.com]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

azure_dns_zone_name_and_nameserver = {
  "dns_zone_name" = "singhritesh85.com"
  "name_servers" = toset([
    "+ [REDACTED]",
    "+ [REDACTED]",
    "+ [REDACTED]",
    "+ [REDACTED]"
  ])
}

```

Then Private AKS Cluster, a K8S-Management Node and Azure Database for PostgreSQL Server had been created using the Terraform Script present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **terraform-private-aks-cluster-postgresql-flexible-servers**. Cluster Autoscaler is already enabled in AKS Cluster.

Commands as written below had been used to create infrastructure using Terraform.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created.

Ritesh Kumar Singh || Email Address: - riteshkumarsingh9559@gmail.com || LinkedIn: - <https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/> || GitHub: - <https://github.com/singhritesh85>

Then you can run the command **terraform apply -auto-approve** ----> Finally, Create the resources.

```
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [00m40s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [00m40s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [00m50s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [00m50s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [01m00s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [01m00s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [01m10s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [01m10s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [01m20s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [01m30s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [01m30s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [01m40s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Still creating... [01m40s elapsed]
module.aks.azurerm_kubernetes_cluster_node_pool.autoscale_node_pool: Creation complete after 1m49s [id=/subscriptions//resourceGroups/ansible-awx-rg/providers/Microsoft.ContainerService/managedClusters/ansible-awx-cluster/agentPools/userpool]
module.aks.null_resource.kubectl: Creating...
module.aks.null_resource.kubectl: Provisioning with 'local-exec'...
module.aks.null_resource.kubectl [local-exec]: Executing: ["bin/bash" "-c" "az account set --subscription $(az account show --query id|tr -d '\\"') && az aks get-credentials --resource-group ansible-awx-rg --name ansible-awx-cluster --overwrite-existing && chmod 600 ~/.kube/config"]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [01m50s elapsed]
module.aks.null_resource.kubectl: WARNING: Merged "ansible-awx-cluster" as current context in /root/.kube/config
module.aks.null_resource.kubectl: Creation complete after 2s [id=]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Still creating... [02m00s elapsed]
module.aks.azurerm_monitor_metric_alert.alert_rule1: Creation complete after 2m3s [id=/subscriptions//resourceGroups/ansible-awx-rg/providers/Microsoft.Insights/metricAlerts/ansible-awx-alert-rule1]

Apply complete! Resources: 36 added, 0 changed, 0 destroyed.

Outputs:

azurerm_private_ip_and_aks_details_and_application_gateway_name = {
  "aks_id" = "/subscriptions//resourceGroups/ansible-awx-rg/providers/Microsoft.ContainerService/managedClusters/ansible-awx-cluster"
  "aks_name" = "ansible-awx-cluster"
  "application_gateway_name" = "app-gtw-ingress-controller"
  "azurerm_k8s_management_node_privateip" = "10.224.0.10"
}
```

[demo@k8s-management-node-vm ~]\$ kubectl get nodes				
NAME	STATUS	ROLES	AGE	VERSION
aks-agentpool-[REDACTED]-vmss000001	Ready	<none>	3m17s	v1.33.0
aks-userpool-[REDACTED]-vmss000000	Ready	<none>	2m32s	v1.33.0

Installation of AWX using Helm

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

```
helm repo update
```

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

```
[demo@k8s-management-node-vm ~]$ helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
"awx-operator" has been added to your repositories
[demo@k8s-management-node-vm ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "awx-operator" chart repository
Update Complete. ⚡Happy Helm-ing!⚡
[demo@k8s-management-node-vm ~]$ helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace
```

```
[demo@k8s-management-node-vm ~]$ cat awx-postgres-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "ansible-awx-postgresql3.postgres.database.azure.com"
  port: "5432"
  database: "dexter"
  username: "postgres"
  password: "QJlnI^oTwPD_2nTQ"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
[demo@k8s-management-node-vm ~]$ kubectl apply -f awx-postgres-secret.yaml
secret/awx-postgres-configuration created
```

```
[demo@k8s-management-node-vm ~]$ kubectl get secrets -n awx|grep "awx-postgres-configuration"
awx-postgres-configuration          Opaque      7           2m7s
```

```
[demo@k8s-management-node-vm ~]$ cat ansible-awx.yaml
apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
  admin_password_secret: awx-admin-password-secret
[demo@k8s-management-node-vm ~]$ kubectl apply -f ansible-awx.yaml
awx.ansible.com/ansible-awx created
```

```
[demo@k8s-management-node-vm ~]$ kubectl get all -n awx
NAME                                         READY   STATUS    RESTARTS   AGE
pod/ansible-awx-migration-24.6.1-[REDACTED]   0/1     Completed  0          6m51s
pod/ansible-awx-task-[REDACTED]                4/4     Running   0          7m31s
pod/ansible-awx-web-[REDACTED]                 3/3     Running   0          7m33s
pod/awx-operator-controller-manager-[REDACTED]   2/2     Running   0          13m

NAME                           TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
service/ansible-awx-service  ClusterIP  10.0.105.111 <none>       80/TCP    7m36s
service/awx-operator-controller-manager-metrics-service  ClusterIP  10.0.230.233 <none>       8443/TCP  13m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ansible-awx-task  1/1     1           1           7m31s
deployment.apps/ansible-awx-web   1/1     1           1           7m33s
deployment.apps/awx-operator-controller-manager  1/1     1           1           13m

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/ansible-awx-task [REDACTED]  1         1         1         7m31s
replicaset.apps/ansible-awx-web  [REDACTED]  1         1         1         7m33s
replicaset.apps/awx-operator-controller-manager-[REDACTED]  1         1         1         13m

NAME          STATUS    COMPLETIONS   DURATION   AGE
job.batch/ansible-awx-migration-24.6.1  Complete  1/1          4m5s      6m51s
```

```
cat awx-postgres-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "ansible-awx-postgresql3.postgres.database.azure.com"
  port: "5432"
  database: "dexter"
  username: "postgres"
  password: "XXXXXXXXXXXXXXXXXX"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
```

```
cat ansible-awx.yaml

apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
```

Then I created the Ingress rule (to route the traffic through Application Gateway Ingress Controller in AKS) and did the entry of created LoadBalancer Service Public IP Address to create the Record-Set of A-Type in Azure DNS Zone as shown in the screenshot attached below.

```
[demo@k8s-management-node-vm ~]$ ls
STAR_singhritesh85_com.crt ansible-awx.yaml awx-postgres-secret.yaml mykey.key
[demo@k8s-management-node-vm ~]$ kubectl create secret tls awx-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n awx
secret/awx-tls-secret created
[demo@k8s-management-node-vm ~]$ kubectl get secrets -n awx|grep "awx-tls-secret"
                           kubernetes.io/tls      2       32s

[demo@k8s-management-node-vm ~]$ cat ingress-rule-awx-aks.yaml
# kubectl create secret tls awx-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n awx
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ansible-awx-ingress
  namespace: awx
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: awx-tls-secret
  rules:
  - host: ansible-awx.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: ansible-awx-service
            port:
              number: 80

[demo@k8s-management-node-vm ~]$ kubectl apply -f ingress-rule-awx-aks.yaml
ingress.networking.k8s.io/ansible-awx-ingress created
[demo@k8s-management-node-vm ~]$ kubectl get ing -n awx
NAME           CLASS          HOSTS
ansible-awx-ingress   azure-application-gateway  ansible-awx.singhritesh85.com
                      ADDRESS        PORTS      AGE
20.████████.165     80, 443      7s
```

To access the Ansible-Awx using the URL I did the entry of Public IP Address of the Ingress as shown in the screenshot attached above in the Azure DNS Zone to create the A-Type Record Set.

The screenshot shows the 'Add record set' dialog in the Azure portal. The URL is [https://portal.azure.com/#create/DNS-Zone](#). The form fields are as follows:

- Name:** ansible-awx
- Type:** A – IPv4 Address records
- Alias record set:** No
- TTL:** 1
- TTL unit:** Hours
- IP address:**
 - 20.████████.165
 - 0.0.0.0

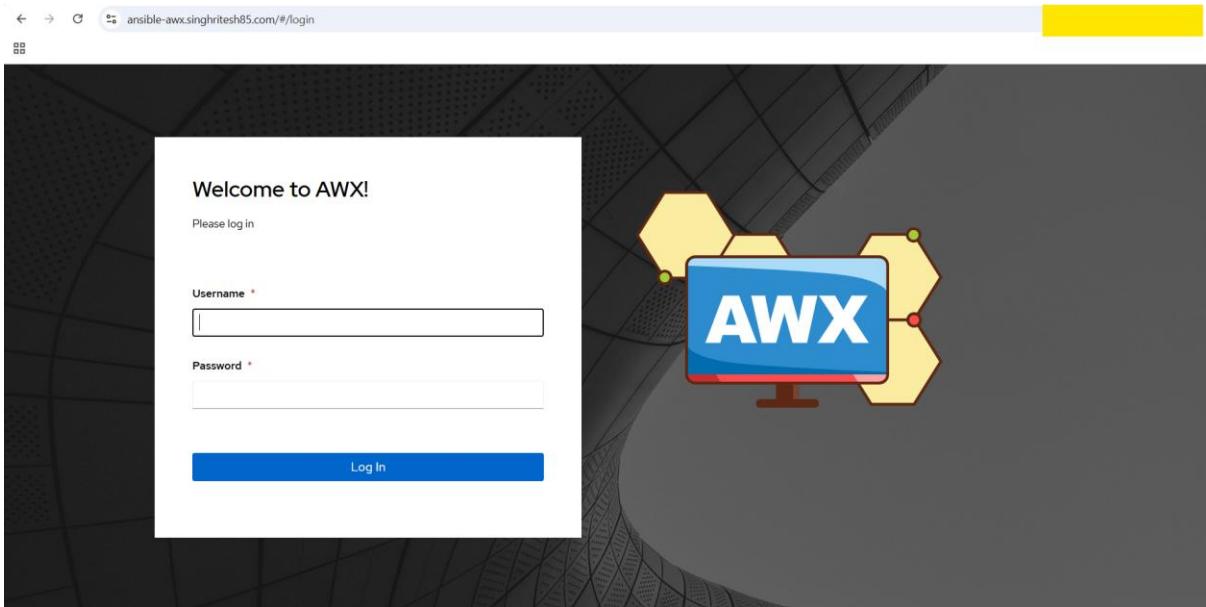
At the bottom, there are buttons for **Add** (highlighted with a yellow box), **Cancel**, and **Give feedback**.

```
cat ingress-rule-awx-aks.yaml

# kubectl create secret tls awx-tls-secret --cert=STAR_singhritesh85_com.crt --key=mykey.key -n awx
---

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ansible-awx-ingress
  namespace: awx
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: awx-tls-secret
  rules:
  - host: ansible-awx.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: ansible-awx-service
            port:
              number: 80
```

Finally, I was able to access the Ansible-Awx using the URL as shown in the screenshot attached below.



Now, I will reset the Ansible-Awx **admin** user **password** using the procedure as explained below.

```
kubectl get pods -n awx
```

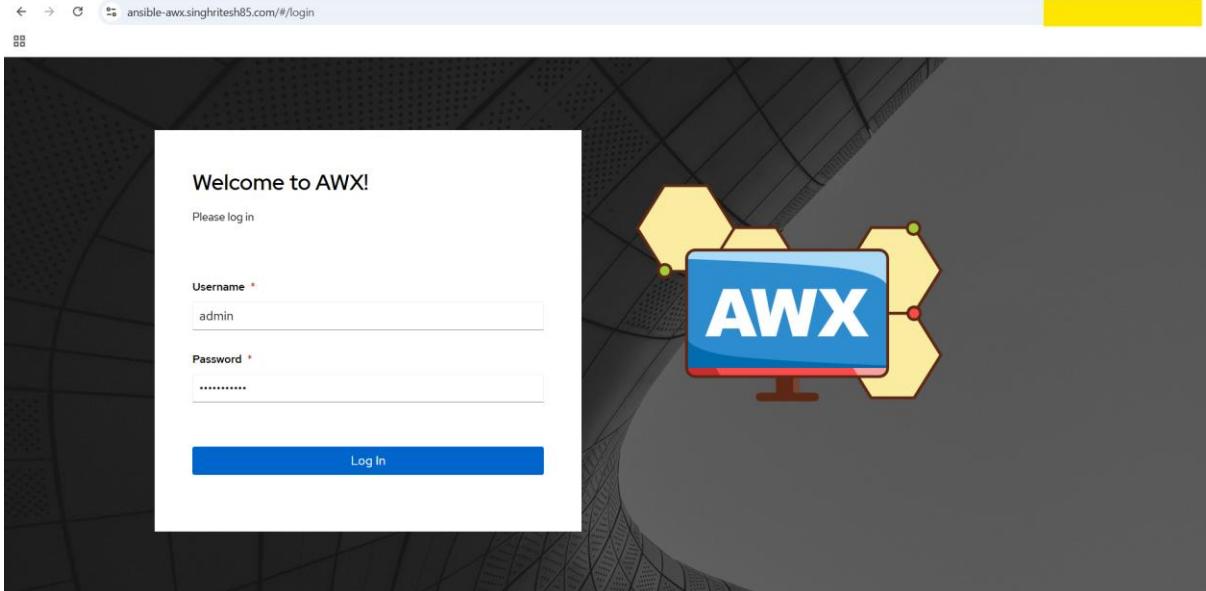
Log-into the ansible-awx-web Pod as shown below

```
kubectl exec -it <ansible-awx-web pod> -n awx -- bash
```

```
awx-manage update_password --username=admin --password=<provide-your-password-here>
```

```
[demo@k8s-management-node-vm ~]$ kubectl exec -it ansible-awx-web-  
bash-5.1$ awx-manage update_password --username=admin --password=  
Password updated
```

Then, I logged-into the Ansible-Awx using the updated password as shown below.



Now, you can proceed further with the steps as I discussed in module-1.

Module-3: - DevOps Project Ansible Awx (GKE)

In this module I will explain about installation of Ansible Awx on the top of Private GKE Standard Cluster and Cloud SQL (PostgreSQL) had been used as the database. To access the Ansible Awx using the URL I had used GKE Ingress for Application Load Balancers (for Kubernetes Secrets of tls, Standard SSL Certificate had been used). A DNS Zone had been created in GCP Cloud DNS using the Terraform Script present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path `terraform-gcp-cloud-dns`.

Commands as written below had been used to create infrastructure using Terraform.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** ----> Finally, Create the resources.

After running the Terraform Script update your domain provider with the generated Nameserver of the Zone of GCP Cloud DNS.

```
+ enable_logging = true
}

+ dnssec_config {
  + kind           = "dns#managedZoneDnsSecConfig"
  + non_existence = (known after apply)
  + state          = "on"

  + default_key_specs (known after apply)
}
}

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ gcp_cloud_dns_name_and_dns_name = {
  + dns_zone_nameservers = (known after apply)
  + zone_dns_name       = "singhritesh85.com."
  + zone_name           = "public-hosted-zone-logging-enabled"
}
module.gcp_cloud_dns.google_dns_managed_zone.cloud_logging_enabled_zone: Creating...
module.gcp_cloud_dns.google_dns_managed_zone.cloud_logging_enabled_zone: Creation complete after 1s [id=projects/[REDACTED]/managedZones/public-hosted-zone-logging-enabled]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

gcp_cloud_dns_name_and_dns_name = {
  "dns_zone_nameservers" = tolist([
    "REDACTED",
    "REDACTED",
    "REDACTED",
    "REDACTED",
    "REDACTED"
  ])
  "zone_dns_name" = "singhritesh85.com."
  "zone_name"     = "public-hosted-zone-logging-enabled"
}
```

Then Private GKE Standard Cluster, a K8S-Management Node and Cloud SQL (PostgreSQL) had been created using the Terraform Script present in the GitHub Repo

<https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **terraform-gke-cluster-cloudsal**. Cluster Autoscaler is already enabled in GKE Standard Cluster.

Commands as written below had been used to create infrastructure using Terraform.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created.

Then you can run the command **terraform apply -auto-approve** ----> Finally, Create the resources.

```
module.gke.google_sql_database_instance.db_instance: Still creating... [14m40s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [14m50s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [15m00s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [15m10s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [15m20s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [15m30s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [15m40s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [15m50s elapsed]
module.gke.google_sql_database_instance.db_instance: Still creating... [16m00s elapsed]
module.gke.google_sql_database_instance.db_instance: Creation complete after 16m19s [id=awx-private-dbinstance-[REDACTED]]
module.gke.google_sql_database.dexter_dbschema: Creating...
module.gke.google_sql_user_db_users: Creating...
module.gke.google_sql_database.dexter_dbschema: Creation complete after 6s [id=projects/[REDACTED]/instances/awx-private-dbinstance-[REDACTED]/database]
module.gke.google_sql_user_db_users: Creation complete after 9s [id=dbadmin//awx-private-dbinstance-[REDACTED]]

Apply complete! Resources: 34 added, 0 changed, 0 destroyed.

Outputs:

gcp_cloud_sql_vm_instance_gke_details = {
  "GCP_Linux_VM_Instance_External_IP" = "34.125.124"
  "GCP_Linux_VM_Instance_Internal_IP" = "10.20.15.200"
  "cloud_sql_connection_name" = "[REDACTED];us-central1:awx-private-dbinstance-[REDACTED]"
  "cloud_sql_db_instance_name" = "awx-private-dbinstance-[REDACTED]"
  "cloud_sql_private_ip_address" = "[REDACTED]"
  "gke_standard_cluster_endpoint" = "[REDACTED]"
  "gke_standard_cluster_name" = "awx-gke-cluster"
  "gke_standard_cluster_node_pool_instance_group_urls" = tolist([
    "https://www.googleapis.com/compute/v1/projects/[REDACTED]/zones/us-central1-a/instanceGroupManagers/gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]-grp",
    "https://www.googleapis.com/compute/v1/projects/[REDACTED]/zones/us-central1-c/instanceGroupManagers/gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]-grp",
    "[REDACTED]"
  ])
  "gke_standard_cluster_node_pool_name" = "awx-linux-nodepool-1"
  "gke_standard_cluster_node_pool_version" = "1.33.5"
}
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get nodes
NAME                      STATUS   ROLES      AGE     VERSION
gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]   Ready    <none>   6m52s   v1.33.5-gke.2172001
gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]   Ready    <none>   6m42s   v1.33.5-gke.2172001
```

Installation of AWX using Helm

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

```
helm repo update
```

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

```
[k8s-management@awx-k8s-management-node ~]$ helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
"awx-operator" has been added to your repositories
[k8s-management@awx-k8s-management-node ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "awx-operator" chart repository
Update Complete. ⚡Happy Helm-ing!⚡
[k8s-management@awx-k8s-management-node ~]$ helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace
```

```
[k8s-management@awx-k8s-management-node ~]$ cat awx-postgres-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "[REDACTED]"
  port: "5432"
  database: "dexter"
  username: "dbadmin"
  password: "[REDACTED]"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f awx-postgres-secret.yaml
secret/awx-postgres-configuration created
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get secrets -n awx|grep "awx-postgres-configuration"
awx-postgres-configuration          Opaque            7           15s
```

```
[k8s-management@awx-k8s-management-node ~]$ cat ansible-awx.yaml
apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
  admin_password_secret: awx-admin-password-secret
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f ansible-awx.yaml
awx.ansible.com/ansible-awx created

[k8s-management@awx-k8s-management-node ~]$ kubectl get all -n awx
NAME                                         READY   STATUS    RESTARTS   AGE
pod/ansible-awx-migration-24.6.1-[REDACTED]  0/1     Completed  0          18m
pod/ansible-awx-task-[REDACTED]                4/4     Running   0          19m
pod/ansible-awx-web-[REDACTED]                 3/3     Running   0          19m
pod/awx-operator-controller-manager-[REDACTED]  2/2     Running   0          31m

NAME                                         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/ansible-awx-service                  ClusterIP   172.19.244.214  <none>       80/TCP    19m
service/awx-operator-controller-manager-metrics-service ClusterIP   172.19.40.235   <none>       8443/TCP  31m

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ansible-awx-task            1/1     1           1           19m
deployment.apps/ansible-awx-web              1/1     1           1           19m
deployment.apps/awx-operator-controller-manager 1/1     1           1           31m

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/ansible-awx-task-[REDACTED]  1         1         1         19m
replicaset.apps/ansible-awx-web-[REDACTED]   1         1         1         19m
replicaset.apps/awx-operator-controller-manager-[REDACTED]  1         1         1         31m

NAME                                         STATUS  COMPLETIONS   DURATION   AGE
job.batch/ansible-awx-migration-24.6.1      Complete 1/1          16m        18m
```

Ritesh Kumar Singh

```
cat awx-postgres-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "XX.XXX.XXX.X"
  port: "5432"
  database: "dexter"
  username: "dbadmin"
  password: "XXXXXXXXXXXXXXX"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
```

```
cat ansible-awx.yaml

apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
```

Then Ingress Rule (to route the traffic through GKE Ingress for Application Load Balancers) had been created and did the entry of created LoadBalancer Service Public IP Address to create the Record-Set of A-Type in GCP Cloud DNS Zone as shown in the screenshot attached below.

```
[k8s-management@awx-k8s-management-node ~]$ ls
ansible-awx.yaml awx-postgres-secret.yaml f_____b.crt server.key
[k8s-management@awx-k8s-management-node ~]$ kubectl create secret tls awx-tls-secret --cert=f_____b.crt --key=server.key -n awx
secret/awx-tls-secret created
[k8s-management@awx-k8s-management-node ~]$ kubectl get secrets -n awx|grep -i "awx-tls-secret"
          kubernetes.io/tls      2        45s
awx-tls-secret

[k8s-management@awx-k8s-management-node ~]$ cat ingress-rule-gke.yaml
# kubectl create secret tls awx-tls-secret --cert=f58ce5a18a182d4b.crt --key=server.key -n awx
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: awx-ingress
  namespace: awx
  annotations:
    kubernetes.io/ingress.class: "gce" # Specify the Ingress class
    kubernetes.io/ingress.allow-http: "false"
#   cloud.google.com/backend-config: '{"default": "bankapp-backendconfig"}'
spec:
  ingressClassName: "gce"
  tls:
  - hosts:
    - ansible-awx.singhritesh85.com
    secretName: awx-tls-secret
  rules:
  - host: "ansible-awx.singhritesh85.com"
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: ansible-awx-service
            port:
              number: 80
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f ingress-rule-gke.yaml
ingress.networking.k8s.io/awx-ingress created
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get ing -n awx
NAME      CLASS   HOSTS           ADDRESS          PORTS      AGE
awx-ingress  gce    ansible-awx.singhritesh85.com  136._____.57  80, 443  2m57s
```

To access the Ansible-Awx using the URL I did the entry of Public IP Address of the Ingress as shown in the screenshot attached above in the GCP Cloud DNS Zone to create the A-Type Record Set.

Network Services / Zones / Zone: public-hosted-zone-logging-enabled / Create record set

Cloud DNS

DNS name: ansible-awx .singhritesh85.com. (②)

Resource record type: A

TTL *: 5

TTL unit: minutes (②)

IPv4 Address ②

IPv4 Address 1 *: 136._____.57 (Select)

+ Add item

Create Cancel

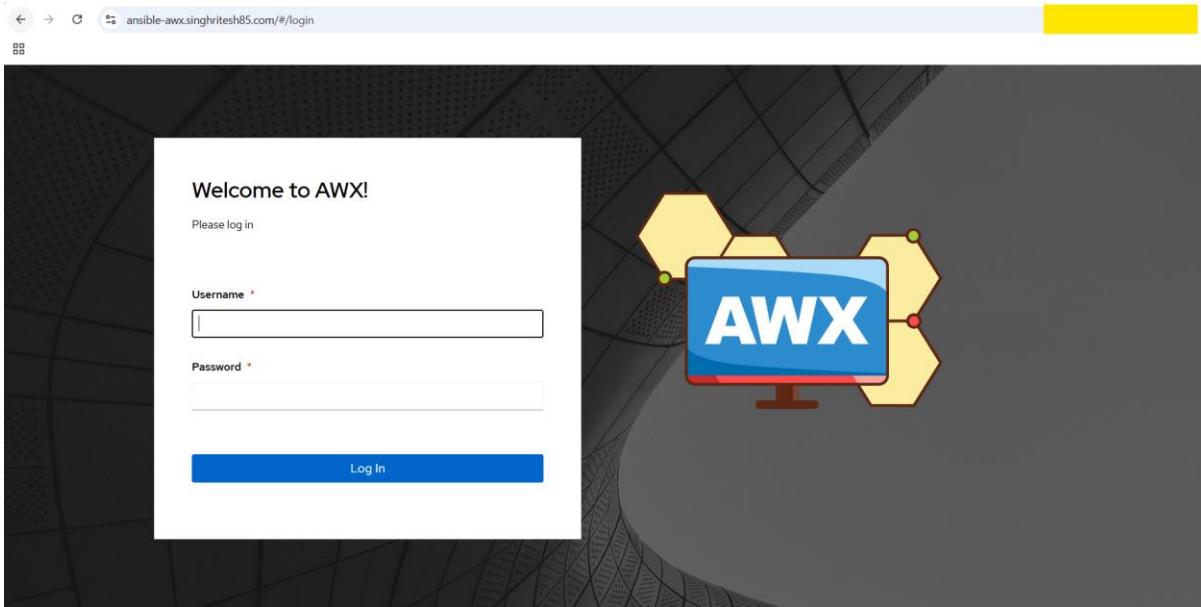
```
cat ingress-rule-gke.yaml

# kubectl create secret tls awx-tls-secret --cert=f58ce5a18a182d4b.crt --key=server.key -n awx

---

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: awx-ingress
  namespace: awx
  annotations:
    kubernetes.io/ingress.class: "gce" # Specify the Ingress class
    kubernetes.io/ingress.allow-http: "false"
#  cloud.google.com/backend-config: '{"default": "bankapp-backendconfig"}'
spec:
  ingressClassName: "gce"
  tls:
    - hosts:
        - ansible-awx.singhritesh85.com
      secretName: awx-tls-secret
  rules:
    - host: "ansible-awx.singhritesh85.com"
      http:
        paths:
          - path: /
            pathType: Prefix
        backend:
          service:
            name: ansible-awx-service
          port:
            number: 80
```

Finally, I was able to access the Ansible-Awx using the URL as shown in the screenshot attached below.



Now, I will reset the Ansible-Awx **admin** user **password** using the procedure as explained below.

```
kubectl get pods -n awx
```

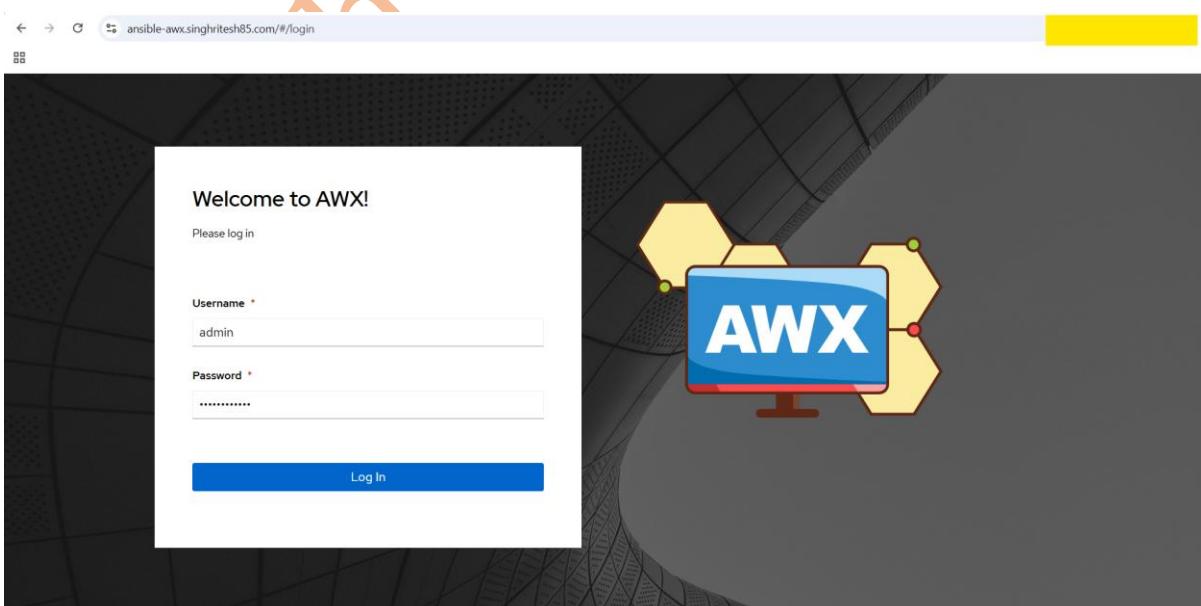
Log-into the ansible-awx-web Pod as shown below

```
kubectl exec -it <ansible-awx-web pod> -n awx -- bash
```

```
awx-manage update_password --username=admin --password=<provide-your-password-here>
```

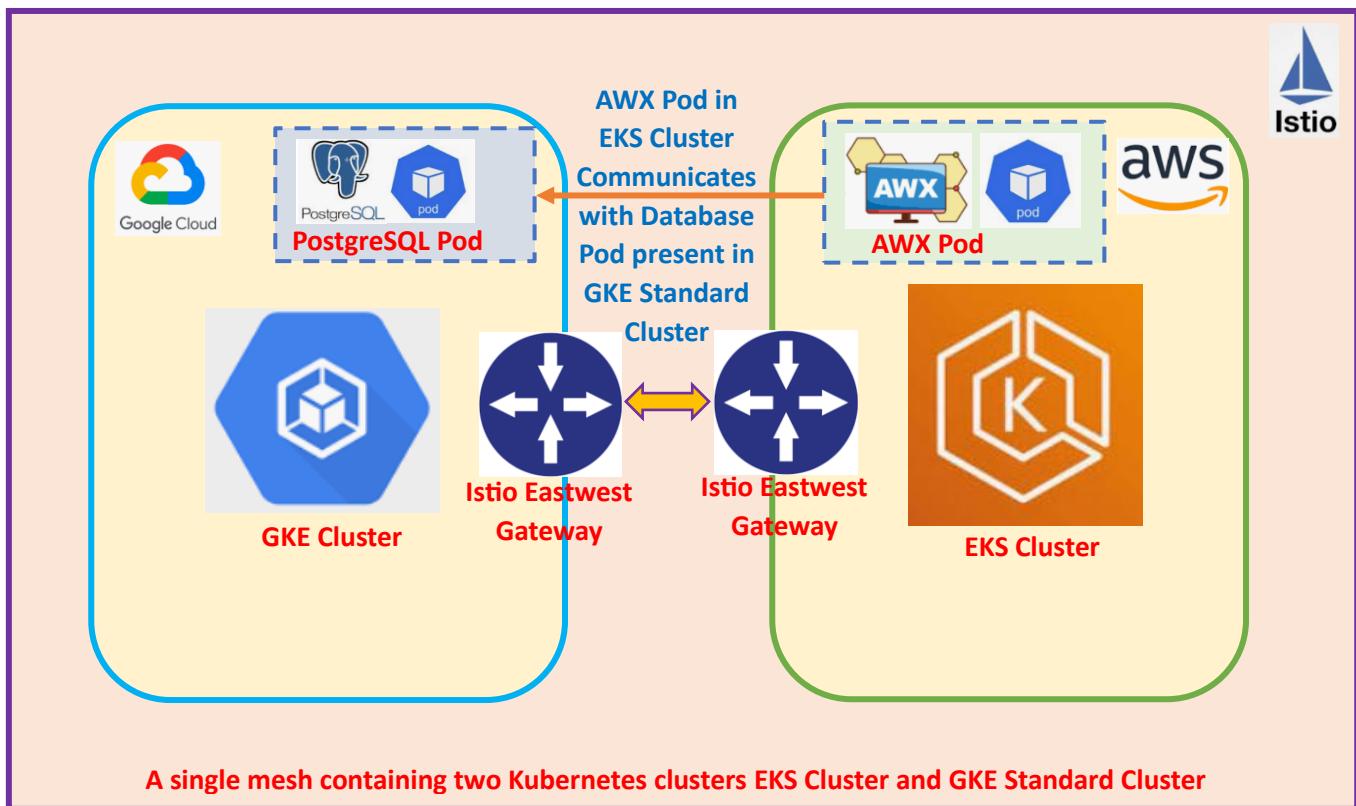
```
[k8s-management@awx-k8s-management-node ~]$ kubectl exec -it ansible-awx-web-[REDACTED] -n awx -- bash  
bash-5.1$ awx-manage update_password --username=admin --password=[REDACTED]  
Password updated
```

Then, I logged-into the Ansible-Awx using the updated password as shown below.



Now, you can proceed further with the steps as I discussed in module-1.

Module-4: - DevOps Project Ansible Awx (using Multicloud Multicluster EKS and GKE)



Kubernetes Multicluster refers to managing and deploying multiple Kubernetes Clusters as a single and unified platform. I had created a single mesh using Istio and two Kubernetes Clusters (GKE Standard Cluster and EKS Cluster) with the help of Istio Eastwest Gateway which is Gateway in Istio for internal cross-cluster communication. I had created GKE (Google Kubernetes Engine) Standard Cluster and EKS (Elastic Kubernetes Services) Cluster using the Terraform Script present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **multicloud-multicloud-eks-cluster-and-gke-cluster**. After creation of the two Kubernetes Cluster (GKE Standard Cluster and EKS Cluster) I installed Istio in each of the Kubernetes Cluster using helm. Before installing the Istio I create the Root CA Certificate and generated intermediate certificate for each cluster. A common root CA certificate is used to establish a shared trust and enable secure communication across clusters. I installed Istio using multi-primary on different networks.

I installed terraform on Rocky Linux GCP VM Instance and State file was kept in the GCP Bucket and state lock had been achieved using the same GCP Bucket. Before running the terraform script I ran the shell script present in the directory **terraform-eks-cluster-and-gke-cluster** as shown in the screenshot attached below. This shell script will install the aws cli.

```
[root@terraform-server terraform-eks-cluster-and-gke-cluster]# pwd
/root/terraform-eks-cluster-and-gke-cluster
[root@terraform-server terraform-eks-cluster-and-gke-cluster]# ls
README.md  initial-setup.sh  karptenter.yaml  main  module

[root@terraform-server terraform-eks-cluster-and-gke-cluster]# ./initial-setup.sh
```

Then I logged-out and login again then ran the command aws configure as shown in the screenshot attached below. As it is an important step to Authenticate and Authorize the user before creating resources using terraform in the AWS Account.

```
[root@terraform-server main]# aws configure
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: [REDACTED]
Default output format [None]: [REDACTED]
```

Then you can run the below commands

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created. Then you can run the command

terraform apply -auto-approve -----> Finally, Create the resources.

```
cations/us-central1/clusters/awx-gke-cluster/nodePools/awx-linux-nodepool1]
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl: Creating...
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl: Provisioning with 'local-exec'...
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec): Executing: ["/bin/sh" "-c" "gcloud container clusters get-credentials awx-gke-cluster --location=us-central1 --project=[REDACTED] && chmod 600 ~/.kube/config"]
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec): WARNING: Accessing a Kubernetes Engine cluster requires the kubernetes commandline
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec): client [kubectl]. To install, run
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec):   $ gcloud components install kubectl

module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec): Fetching cluster endpoint and auth data.
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec): CRITICAL: ACTION REQUIRED: gke-gcloud-auth-plugin, which is needed for continued use of kubectl, was not found or is not executable. Install gke-gcloud-auth-plugin for use with kubectl by following https://cloud.google.com/kubernetes-engine/docs/how-to/cluster-access-for-kubectl#install_plugin
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl (local-exec): kubeconfig entry generated for awx-gke-cluster.
module.eks_cluster_and_standard_gke_cluster.null_resource.gke_kubectl: Creation complete after 4s [id=[REDACTED]]

Apply complete! Resources: 93 added, 0 changed, 0 destroyed.

Outputs:

eks_and_karpenter_iam_role_and_standard_gke_cluster_details = {
  "GCP_Linux_VM_Instance_K8S_Management_External_IP" = "35.[REDACTED].41"
  "GCP_Linux_VM_Instance_K8S_Management_Internal_IP" = "10.[REDACTED].200"
  "eks_cluster_endpoint" = "https://[REDACTED].us-east-2.eks.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "gke_standard_cluster_endpoint" = "34.[REDACTED].60"
  "gke_standard_cluster_name" = "awx-gke-cluster"
  "gke_standard_cluster_node_pool_instance_group_urls" = tolist([
    "https://www.googleapis.com/compute/v1/projects/[REDACTED]/zones/us-central1-a/instanceGroupManagers/gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]-grp",
    "https://www.googleapis.com/compute/v1/projects/[REDACTED]/zones/us-central1-c/instanceGroupManagers/gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]-grp",
  ])
  "gke_standard_cluster_node_pool_name" = "awx-linux-nodepool1"
  "gke_standard_cluster_node_pool_version" = "1.33.5"
  "karpenter_node_spun_iam_role_arn" = "arn:aws:iam::02[REDACTED]6:role/karpenter-eks-noderole"
}
```

After creation of the resources a common kubeconfig file was generated for both the GKE and EKS Cluster.

```
[k8s-management@awx-k8s-management-node ~]$ aws configure
AWS Access Key ID [None]: [REDACTED]
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: [REDACTED]
Default output format [None]: [REDACTED]

[k8s-management@awx-k8s-management-node ~]$ kubectl get nodes --context=gke-[REDACTED]_us-central1_awx-gke-cluster
NAME                               STATUS   ROLES   AGE     VERSION
gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]   Ready    <none>  10m    v1.33.5-gke.2228001
gke-awx-gke-cluster-awx-linux-nodepool-[REDACTED]   Ready    <none>  10m    v1.33.5-gke.2228001
[k8s-management@awx-k8s-management-node ~]$ kubectl get nodes --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
NAME                               STATUS   ROLES   AGE     VERSION
ip-10-10-1-236.us-east-2.compute.internal   Ready    <none>  13m    v1.33.0-eks-802817d
ip-10-10-2-28.us-east-2.compute.internal   Ready    <none>  13m    v1.33.0-eks-802817d
ip-10-10-3-114.us-east-2.compute.internal   Ready    <none>  13m    v1.33.0-eks-802817d
```

Install Istioctl as shown in the screenshot attached below.

```
[k8s-management@awx-k8s-management-node ~]$ curl -L https://istio.io/downloadIstio | sh -
% Total    % Received % Xferd  Average Speed   Time   Time     Current
                                         Dload  Upload   Total   Spent   Left  Speed
100  102  100  102    0      0  772      0  --:--:--  --:--:--  --:--:--  772
100  5124  100  5124    0      0 23290      0  --:--:--  --:--:--  --:--:-- 23290

Downloading istio-1.28.3 from https://github.com/istio/istio/releases/download/1.28.3/istio-1.28.3-linux-amd64.tar.gz ...

Istio 1.28.3 download complete!

The Istio release archive has been downloaded to the istio-1.28.3 directory.

To configure the istioctl client tool for your workstation,
add the /home/k8s-management/istio-1.28.3/bin directory to your environment path variable with:
export PATH="$PATH:/home/k8s-management/istio-1.28.3/bin"

Begin the Istio pre-installation check by running:
  istioctl x precheck

Try Istio in ambient mode
  https://istio.io/latest/docs/ambient/getting-started/
Try Istio in sidecar mode
  https://istio.io/latest/docs/setup/getting-started/
Install guides for ambient mode
  https://istio.io/latest/docs/ambient/install/
Install guides for sidecar mode
  https://istio.io/latest/docs/setup/install/

Need more information? Visit https://istio.io/latest/docs/
```

```
[k8s-management@awx-k8s-management-node ~]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]] ; then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
PATH="$PATH:/home/k8s-management/istio-1.28.3/bin"
```

I created first the **istio-system** namespace in both the clusters (GKE and EKS Cluster) as shown in the screenshot attached below.

```
[k8s-management@awx-k8s-management-node ~]$ kubectl create ns istio-system --context=gke_XXXX-XXXXXXX-2XXXX6_us-central1_awx-gke-cluster
namespace/istio-system created
[k8s-management@awx-k8s-management-node ~]$ kubectl create ns istio-system --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
namespace/istio-system created
```

kubectl create ns istio-system --context=gke_XXXX-XXXXXXX-2XXXX6_us-central1_awx-gke-cluster

kubectl create ns istio-system --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev

To create the Root CA certificate and certificate for each of the clusters I used make command and I install the same as shown in the screenshot attached below.

sudo yum groupinstall 'Development Tools' -y

```
[k8s-management@awx-k8s-management-node ~]$ sudo yum groupinstall 'Development Tools' -y
```

It will install Java-1.8 which you can uninstall later after creating the Root CA certificate and certificate for each of the clusters.

```
[k8s-management@awx-k8s-management-node ~]$ java -version
openjdk version "1.8.0_482"
OpenJDK Runtime Environment (build 1.8.0_482-b08)
OpenJDK 64-Bit Server VM (build 25.482-b08, mixed mode)
```

I used below command to create the Root CA certificate and certificate for each of the clusters.

```
mkdir certs
cd certs/
make -f ..istio-1.28.3/tools/certs/Makefile.selfsigned.mk root-ca
make -f ..istio-1.28.3/tools/certs/Makefile.selfsigned.mk awx-gke-cluster-cacerts
make -f ..istio-1.28.3/tools/certs/Makefile.selfsigned.mk eks-demo-cluster-dev-cacerts
cd
```

```
[k8s-management@awx-k8s-management-node ~]$ mkdir certs
[k8s-management@awx-k8s-management-node ~]$ cd certs/
[k8s-management@awx-k8s-management-node certs]$ make -f ..istio-1.28.3/tools/certs/Makefile.selfsigned.mk root-ca
generating root-key.pem
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
generating root-cert.csr
generating root-cert.pem
Signature ok
subject=O = Istio, CN = Root CA
Getting Private key
[k8s-management@awx-k8s-management-node certs]$ make -f ..istio-1.28.3/tools/certs/Makefile.selfsigned.mk awx-gke-cluster-cacerts
generating awx-gke-cluster/ca-key.pem
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
generating awx-gke-cluster/cluster-ca.csr
generating awx-gke-cluster/ca-cert.pem
Signature ok
subject=O = Istio, CN = Intermediate CA, L = awx-gke-cluster
Getting CA Private Key
generating awx-gke-cluster/cert-chain.pem
Intermediate inputs stored in awx-gke-cluster/
done
rm awx-gke-cluster/cluster-ca.csr awx-gke-cluster/intermediate.conf

[k8s-management@awx-k8s-management-node certs]$ make -f ..istio-1.28.3/tools/certs/Makefile.selfsigned.mk eks-demo-cluster-dev-cacerts
generating eks-demo-cluster-dev/ca-key.pem
Generating RSA private key, 4096 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
generating eks-demo-cluster-dev/cluster-ca.csr
generating eks-demo-cluster-dev/ca-cert.pem
Signature ok
subject=O = Istio, CN = Intermediate CA, L = eks-demo-cluster-dev
Getting CA Private Key
generating eks-demo-cluster-dev/cert-chain.pem
Intermediate inputs stored in eks-demo-cluster-dev/
done
rm eks-demo-cluster-dev/intermediate.conf eks-demo-cluster-dev/cluster-ca.csr
```

Then created the kubernetes secrets **cacerts** in the namespace **istio-system** in each of the kubernetes cluster (GKE Cluster and EKS Cluster) as shown in the screenshot attached below.

```
kubectl create secret generic cacerts --from-file=certs/awx-gke-cluster/ca-cert.pem --from-file=certs/awx-gke-cluster/ca-key.pem --from-file=certs/awx-gke-cluster/root-cert.pem --from-file=certs/awx-gke-cluster/cert-chain.pem -n istio-system --context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster

kubectl create secret generic cacerts --from-file=certs/eks-demo-cluster-dev/ca-cert.pem --from-file=certs/eks-demo-cluster-dev/ca-key.pem --from-file=certs/eks-demo-cluster-dev/root-cert.pem --from-file=certs/eks-demo-cluster-dev/cert-chain.pem -n istio-system --context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl create secret generic cacerts --from-file=certs/awx-gke-cluster/ca-cert.pem --from-file=certs/awx-gke-cluster/root-cert.pem --from-file=certs/awx-gke-cluster/cert-chain.pem -n istio-system --context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster
secret/cacerts created
[k8s-management@awx-k8s-management-node ~]$ kubectl create secret generic cacerts --from-file=certs/eks-demo-cluster-dev/ca-cert.pem --from-file=certs/eks-demo-cluster-dev/ca-key.pem --from-file=certs/eks-demo-cluster-dev/root-cert.pem --from-file=certs/eks-demo-cluster-dev/cert-chain.pem -n istio-system --context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev
secret/cacerts created
```

For Istio installation I installed istio-base, istiod and then istio Eastwest gateway on GKE Standard Cluster as shown in the screenshot attached below.

```
kubectl create secret generic cacerts --from-file=certs/awx-gke-cluster/ca-cert.pem --from-file=certs/awx-gke-cluster/ca-key.pem --from-file=certs/awx-gke-cluster/root-cert.pem --from-file=certs/awx-gke-cluster/cert-chain.pem -n istio-system --context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster

helm repo add istio https://istio-release.storage.googleapis.com/charts

helm repo update

helm install istio-base istio/base -n istio-system --kube-context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster

helm install istiod istio/istiod -n istio-system --set global.meshID=mesh1 --set global.multiCluster.clusterName=awx-gke-cluster --set global.network=network2 --kube-context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster

helm install istio-eastwestgateway istio/gateway -n istio-system --set name=istio-eastwestgateway --set networkGateway=network2 --kube-context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster

kubectl get svc istio-eastwestgateway -n istio-system --context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster
```

I applied the label **topology.istio.io/network=network2** to the namespace **istio-system** in GKE Standard Cluster as shown in the screenshot attached below.

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get namespace istio-system && kubectl label namespace istio-system topology.istio.io/network=network2 --context=gke_XXXX-XXXXXX-2XXXX6_us-central1_awx-gke-cluster
NAME           STATUS   AGE
istio-system   Active   23m
namespace/istio-system labeled
```

```
[k8s-management@awx-k8s-management-node ~]$ helm repo add istio https://istio-release.storage.googleapis.com/charts
"istio" has been added to your repositories
[k8s-management@awx-k8s-management-node ~]$
[k8s-management@awx-k8s-management-node ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "istio" chart repository
Update Complete. Happy Helm-ing!
```

```
[k8s-management@awx-k8s-management-node ~]$ helm install istio-base istio/base -n istio-system --kube-context=gke_[REDACTED]_us-central1_awx-gke-cluster
```

```
[k8s-management@awx-k8s-management-node ~]$ helm install istiod istio/istiod -n istio-system --set global.meshID=mesh1 --set global.multiCluster.clusterName=awx-gke-cluster --set global.network=network2 --kube-context=gke_[REDACTED]_us-central1_awx-gke-cluster
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
istio-eastwestgateway	LoadBalancer	172.19.146.228	34.[REDACTED].199	15021:31715/TCP,15443:31415/TCP,15012:31287/TCP,15017:30280/TCP	5m52s

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get svc istio-eastwestgateway -n istio-system --context=gke_[REDACTED]_us-central1_awx-gke-cluster
apiVersion: networking.istio.io/v1
kind: Gateway
metadata:
  name: cross-network-gateway
spec:
  selector:
    istio: eastwestgateway
  servers:
    - port:
        number: 15443
        name: tls
        protocol: TLS
      tls:
        mode: AUTO_PASSTHROUGH
      hosts:
        - ".local"
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f expose-service.yaml --context=gke_[REDACTED]_us-central1_awx-gke-cluster
gateway.networking.istio.io/cross-network-gateway created
[k8s-management@awx-k8s-management-node ~]$
```

For Istio installation on EKS Cluster I installed istio-base, istiod and then istio Eastwest gateway on as shown in the screenshot attached below.

```
kubectl create secret generic cacerts --from-file=certs/eks-demo-cluster-dev/ca-cert.pem --
from-file=certs/eks-demo-cluster-dev/ca-key.pem --from-file=certs/eks-demo-cluster-
dev/root-cert.pem --from-file=certs/eks-demo-cluster-dev/cert-chain.pem -n istio-system --
context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev

kubectl get namespace istio-system && kubectl label namespace istio-system
topology.istio.io/network=network1 --context=arn:aws:eks:us-east-
2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev

helm repo add istio https://istio-release.storage.googleapis.com/charts

helm repo update

helm install istio-base istio/base -n istio-system --kube-context=arn:aws:eks:us-east-
2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev

helm install istiod istio/istiod -n istio-system --set global.meshID=mesh1 --set
global.multiCluster.clusterName=eks-demo-cluster-dev --set global.network=network1 --kube-
context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev

helm install istio-eastwestgateway istio/gateway -n istio-system --set name=istio-
eastwestgateway --set networkGateway=network1 --kube-context=arn:aws:eks:us-east-
2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev

kubectl get svc istio-eastwestgateway -n istio-system --context=arn:aws:eks:us-east-
2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev
```

I applied the label **topology.istio.io/network=network1** to the namespace **istio-system** in EKS Cluster as shown in the screenshot attached below.

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get namespace istio-system && kubectl label namespace istio-system topology.istio.io/network=network1 --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
NAME          STATUS   AGE
istio-system   Active   48m
namespace/istio-system labeled
[k8s-management@awx-k8s-management-node ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "istio" chart repository
Update Complete. Happy Helming!
[k8s-management@awx-k8s-management-node ~]$ helm install istio-base istio/base -n istio-system --kube-context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev

[k8s-management@awx-k8s-management-node ~]$ helm install istiod istio/istiod -n istio-system --set global.meshID=mesh1 --set global.multiCluster.clusterName=eks-demo-cluster-dev --set global.network=network1 --kube-context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev

[k8s-management@awx-k8s-management-node ~]$ helm install istio-eastwestgateway istio/gateway -n istio-system --set name=istio-eastwestgateway --set networkGateway=network1 --kube-context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev

[k8s-management@awx-k8s-management-node ~]$ kubectl get svc istio-eastwestgateway -n istio-system --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
NAME           TYPE      CLUSTER-IP        EXTERNAL-IP          PORT(S)
istio-eastwestgateway   LoadBalancer   172.20.252.151   a[REDACTED]3.us-east-2.elb.amazonaws.com   15012:32720/TCP,15443:30540/TCP,15017:32555/TCP
AGE
10s
```

cat expose-service.yaml

```
apiVersion: networking.istio.io/v1
```

```
kind: Gateway
```

```
metadata:
```

```
  name: cross-network-gateway
```

```
spec:
```

```
  selector:
```

```
    istio: eastwestgateway
```

```
  servers:
```

```
    - port:
```

```
      number: 15443
```

```
      name: tls
```

```
      protocol: TLS
```

```
    tls:
```

```
      mode: AUTO_PASSTHROUGH
```

```
  hosts:
```

```
    - "*.local"
```

```
[k8s-management@awx-k8s-management-node ~]$ cat expose-service.yaml
apiVersion: networking.istio.io/v1
kind: Gateway
metadata:
  name: cross-network-gateway
spec:
  selector:
    istio: eastwestgateway
  servers:
    - port:
        number: 15443
        name: tls
        protocol: TLS
      tls:
        mode: AUTO_PASSTHROUGH
      hosts:
        - "*.local"
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f expose-service.yaml --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
gateway.networking.istio.io/cross-network-gateway created
```

=====

Enable endpoint discovery in GKE and EKS Cluster

=====

```
istioctl create-remote-secret --name=eks-demo-cluster-dev | kubectl apply -f - --context=gke_XXXX-XXXXXX-XXXXX6_us-central1_awx-gke-cluster
```

```
istioctl create-remote-secret --name=awx-gke-cluster | kubectl apply -f - --context=arn:aws:eks:us-east-2:02XXXXXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ istioctl create-remote-secret --name=awx-gke-cluster | kubectl apply -f - --context=arn:aws:eks:us-east-2:02XXXX-XXXXX6_us-central1_awx-gke-cluster
secret/istio-remote-secret-awx-gke-cluster created
[k8s-management@awx-k8s-management-node ~]$ istioctl create-remote-secret --name=eks-demo-cluster-dev | kubectl apply -f - --context=gke_XXXX-XXXXXX-XXXXX6_us-central1_awx-gke-cluster
secret/istio-remote-secret-eks-demo-cluster-dev created
```

Now verify that istiod can communicate with the remote cluster control plan

```
istioctl remote-clusters --context=arn:aws:eks:us-east-2:02XXXXXXXXXX6:cluster/eks-demo-cluster-dev
```

```
istioctl remote-clusters --context=gke_XXXX-XXXXXX-XXXXX6_us-central1_awx-gke-cluster
```

```
[k8s-management@awx-k8s-management-node ~]$ istioctl remote-clusters --context=arn:aws:eks:us-east-2:02XXXX-XXXXX6:cluster/eks-demo-cluster-dev
NAME           SECRET          STATUS    ISTIOD
eks-demo-cluster-dev      istio-system/istio-remote-secret-awx-gke-cluster   synced    istiod-
awx-gke-cluster
[k8s-management@awx-k8s-management-node ~]$ istioctl remote-clusters --context=gke_XXXX-XXXXXX-XXXXX6_us-central1_awx-gke-cluster
NAME           SECRET          STATUS    ISTIOD
awx-gke-cluster      istio-system/istio-remote-secret-eks-demo-cluster-dev   synced    istiod-
eks-demo-cluster-dev      istio-system/istio-remote-secret-eks-demo-cluster-dev   synced    istiod-
```

Finally, Multicloud Kubernetes Cluster (with GKE and EKS Clusters) had been created using Istio with **multi-primary on different networks**.

As shown in the diagram drawn in Module-4, **PostgreSQL Pods and Service** will be created in the GKE Standard Cluster and **AWX pods and Service along with PostgreSQL Service** will be created in the EKS Cluster.

First, I created the namespace postgresql in the GKE and EKS Cluster and the namespace awx in the EKS Cluster then provided the kubernetes label **istio-injection=enabled** as shown in the screenshot attached below. **PostgreSQL Pods and Service in GKE Standard cluster** had been created using the helm as shown in the screenshot attached below. The helm chart to create the PostgreSQL Pods and Service is present in the GitHub Repo <https://github.com/singhritesh85/helm-repo-for-bitnami-postgresql-ha.git>.

```
[k8s-management@awx-k8s-management-node ~]$ kubectl create ns awx --context=arn:aws:eks:us-east-2:02XXXX-XXXXX6:cluster/eks-demo-cluster-dev
namespace/awx created
[k8s-management@awx-k8s-management-node ~]$ kubectl create ns postgresql --context=gke_XXXX-XXXXXX-XXXXX6_us-central1_awx-gke-cluster
namespace/postgresql created
[k8s-management@awx-k8s-management-node ~]$ kubectl create ns postgresql --context=arn:aws:eks:us-east-2:02XXXX-XXXXX6:cluster/eks-demo-cluster-dev
namespace/postgresql created

[k8s-management@awx-k8s-management-node ~]$ kubectl label ns postgresql istio-injection=enabled --context=arn:aws:eks:us-east-2:02XXXX-XXXXX6:cluster/eks-demo-cluster-dev
namespace/postgresql labeled
[k8s-management@awx-k8s-management-node ~]$ kubectl label ns postgresql istio-injection=enabled --context=gke_XXXX-XXXXXX-XXXXX6_us-central1_awx-gke-cluster
namespace/postgresql labeled
[k8s-management@awx-k8s-management-node ~]$ kubectl label ns awx istio-injection=enabled --context=arn:aws:eks:us-east-2:02XXXX-XXXXX6:cluster/eks-demo-cluster-dev
namespace/awx labeled
```

```
[k8s-management@awx-k8s-management-node ~]$ git clone https://github.com/singhritesh85/helm-repo-for-bitnami-postgresql-ha.git
Cloning into 'helm-repo-for-bitnami-postgresql-ha'...
remote: Enumerating objects: 58, done.
remote: Counting objects: 100% (58/58), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 58 (delta 24), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (58/58), 255.55 KiB | 2.48 MiB/s, done.
Resolving deltas: 100% (24/24), done.
[k8s-management@awx-k8s-management-node ~]$ cd helm-repo-for-bitnami-postgresql-ha/
[k8s-management@awx-k8s-management-node helm-repo-for-bitnami-postgresql-ha]$ helm upgrade --install postgresql bitnami/postgresql-ha/ -f bitnami/postgresql-ha/values.yaml --create-namespace --namespace postgresql --set postgresql.replicaCount=3,persistence.enabled=true,persistence.size=1Gi,global.defaultStorageClass=standard,rwo,service.type=ClusterIP,postgresql.username=postgres,postgresql.password=Dexter123,postgresql.database=dexter --kube-context=gke_us-central1_awx-gke-cluster
[k8s-management@awx-k8s-management-node helm-repo-for-bitnami-postgresql-ha]$ kubectl get pods -n postgresql --context=gke_us-central1_awx-gke-cluster
NAME          READY   STATUS    RESTARTS   AGE
postgresql-postgresql-ha-pgpool-7 [REDACTED] 9/9     Running   0          3m43s
postgresql-postgresql-ha-postgresql-0 1/1     Running   0          3m42s
postgresql-postgresql-ha-postgresql-1 1/1     Running   0          3m42s
postgresql-postgresql-ha-postgresql-2 1/1     Running   0          3m42s
[k8s-management@awx-k8s-management-node ~]$ kubectl get svc -n postgresql --context=gke_us-central1_awx-gke-cluster
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
postgresql-postgresql-ha-pgpool   ClusterIP  172.19.250.144 <none>      5432/TCP  5m24s
postgresql-postgresql-ha-postgresql ClusterIP  172.19.209.136 <none>      5432/TCP  5m24s
postgresql-postgresql-ha-headless  ClusterIP  None        <none>      5432/TCP  5m24s
```

```
helm upgrade --install postgresql bitnami/postgresql-ha/ -f bitnami/postgresql-ha/values.yaml --create-namespace --namespace postgresql --set postgresql.replicaCount=3,persistence.enabled=true,persistence.size=1Gi,global.defaultStorageClass=standard,rwo,service.type=ClusterIP,postgresql.username=postgres,postgresql.password=Dexter123,postgresql.database=dexter --kube-context=gke_XXXX-XXXXXXX-2XXXX6_us-central1_awx-gke-cluster
```

As shown in the screenshot attached above, Pods and service for PostgreSQL in GKE Standard Cluster had been created.

Now, Created the Service for PostgreSQL in EKS Cluster as shown in the screenshot attached below. The helm chart is present in the GitHub Repo <https://github.com/singhritesh85/helm-repo-for-bitnami-postgresql-ha-onlyservice.git>.

```
helm upgrade --install postgresql bitnami/postgresql-ha/ -f bitnami/postgresql-ha/values.yaml --create-namespace --namespace postgresql --set postgresql.replicaCount=3 --set persistence.enabled=false --set persistence.enabled=false --set service.type=ClusterIP --set postgresql.username=postgres --set postgresql.password=Dexter123 --set postgresql.database=dexter --kube-context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ git clone https://github.com/singhritesh85/helm-repo-for-bitnami-postgresql-ha-onlyservice.git
Cloning into 'helm-repo-for-bitnami-postgresql-ha-onlyservice'...
remote: Enumerating objects: 56, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 56 (delta 20), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (56/56), 250.41 KiB | 2.75 MiB/s, done.
Resolving deltas: 100% (20/20), done.
[k8s-management@awx-k8s-management-node ~]$ cd helm-repo-for-bitnami-postgresql-ha-onlyservice
[k8s-management@awx-k8s-management-node helm-repo-for-bitnami-postgresql-ha-onlyservice]$ helm upgrade --install postgresql bitnami/postgresql-ha/ -f bitnami/postgresql-ha/values.yaml --create-namespace --namespace postgresql --set postgresql.replicaCount=3 --set persistence.enabled=false --set persistence.enabled=false --set service.type=ClusterIP --set postgresql.username=postgres --set postgresql.password=Dexter123 --set postgresql.database=dexter --kube-context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev
[k8s-management@awx-k8s-management-node helm-repo-for-bitnami-postgresql-ha-onlyservice]$ kubectl get svc -n postgresql --context=arn:aws:eks:us-east-2:02XXXXXXXXX6:cluster/eks-demo-cluster-dev
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
postgresql-postgresql-ha-pgpool   ClusterIP  172.20.24.254 <none>      5432/TCP  52s
postgresql-postgresql-ha-postgresql ClusterIP  172.20.98.40 <none>      5432/TCP  52s
postgresql-postgresql-ha-headless  ClusterIP  None        <none>      5432/TCP  52s
```

Then created the Kubernetes Secrets and finally the pods and service for AWX using the kubernetes manifests files present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project>

[Ansible-Awx.git](#) at the path **multicloud-multicloud/awx-postgres-secret-multicloud.yaml** and **multicloud-multicloud/ansible-awx.yaml**.

```
[k8s-management@awx-k8s-management-node ~]$ cat awx-postgres-secret-multicloud.yaml
apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "postgresql-postgresql-ha-pgpool.postgresql.svc.cluster.local"
  port: "5432"
  database: "dexter"
  username: "postgres"
  password: "Dexter123"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f awx-postgres-secret-multicloud.yaml --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
secret/awx-postgres-configuration created
[k8s-management@awx-k8s-management-node ~]$ kubectl get secrets -n awx --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev | grep "awx-postgres-configuration"
awx-postgres-configuration   Opaque    7        49s
```

Install the CRD for AWX in EKS Cluster using the commands as shown in the screenshot attached below.

```
helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
helm repo update
helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace --kube-context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
"awx-operator" has been added to your repositories
[k8s-management@awx-k8s-management-node ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "awx-operator" chart repository
...Successfully got an update from the "istio" chart repository
Update Complete. Happy Helming!
```

```
[k8s-management@awx-k8s-management-node ~]$ helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace --kube-context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
```

NAME	READY	STATUS	RESTARTS	AGE
awx-operator-controller-manager-0	2/2	Running	0	56s

```
[k8s-management@awx-k8s-management-node ~]$ cat ansible-awx.yaml
apiVersion: awx.ansible.com/v1beta1
kind: ANX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f ansible-awx.yaml --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
awx.ansible.com/ansible-awx created
```

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway --kube-context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml --context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
[k8s-management@awx-k8s-management-node ~]$ helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway --kube-context=arn:aws:eks:us-east-2:02XXXXXXX6:cluster/eks-demo-cluster-dev
```

Finally, the pods and service for PostgreSQL in GKE Standard Cluster and service for PostgreSQL in EKS Cluster and awx pods and service in EKS Cluster had been created as shown in the screenshot attached below.

```
[k8s-management@awx-k8s-management-node ~]$ kubectl get svc -n postgresql --context=gke_[REDACTED]_us-central1_awx-gke-cluster
NAME          TYPE   CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
postgresql-postgresql-ha-pgpool   ClusterIP  172.19.83.105 <none>        5432/TCP   15m
postgresql-postgresql-ha-postgresql ClusterIP  172.19.206.149 <none>        5432/TCP   15m
postgresql-postgresql-ha-postgresql-headless ClusterIP  None           <none>        5432/TCP   15m
[k8s-management@awx-k8s-management-node ~]$ kubectl get pods -n postgresql --context=gke_[REDACTED]_us-central1_awx-gke-cluster
NAME            READY   STATUS    RESTARTS   AGE
postgresql-postgresql-ha-pgpool-[REDACTED]  2/2     Running   0          15m
postgresql-postgresql-ha-postgresql-0       2/2     Running   0          15m
postgresql-postgresql-ha-postgresql-1       2/2     Running   0          15m
postgresql-postgresql-ha-postgresql-2       2/2     Running   0          15m

[k8s-management@awx-k8s-management-node ~]$ kubectl get svc -n postgresql --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
NAME          TYPE   CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
postgresql-postgresql-ha-pgpool   ClusterIP  172.20.19.201 <none>        5432/TCP   14m
postgresql-postgresql-ha-postgresql ClusterIP  172.20.43.200 <none>        5432/TCP   14m
postgresql-postgresql-ha-postgresql-headless ClusterIP  None           <none>        5432/TCP   14m

[k8s-management@awx-k8s-management-node ~]$ kubectl get pods -n awx --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev --watch
NAME            READY   STATUS    RESTARTS   AGE
ansible-awx-migration-24.6.1-v5j7q  0/2     Completed   0          9m47s
ansible-awx-task-76579cd79f-svxsb  5/5     Running   0          10m
ansible-awx-web-5c4cb4465f-cqw9x   4/4     Running   0          10m
awx-operator-controller-manager-56cbc8dd4c-n87wf  3/3     Running   0          11m

[k8s-management@awx-k8s-management-node ~]$ kubectl get svc -n awx --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev --watch
NAME          TYPE   CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE
ansible-awx-service   ClusterIP  172.20.174.97 <none>        80/TCP      11m
awx-operator-controller-manager-metrics-service ClusterIP  172.20.134.2 <none>        8443/TCP   12m
```

Then installed Gateway API to access the AWX Pods through the service present in the EKS Cluster.

```
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml --context=arn:aws:eks:us-east-2:02XXXXXXXXXX6:cluster/eks-demo-cluster-dev

helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway --kube-context=arn:aws:eks:us-east-2:02XXXXXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
customresourcedefinition.apiextensions.k8s.io/backendlspolicies.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/gatewayclasses.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/gateways.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/grpcroutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/httproutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/referencegrants.gateway.networking.k8s.io created
[k8s-management@awx-k8s-management-node ~]$ helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway --kube-context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
```

Created Kubernetes tls Secret as shown in the screenshot attached below. The SSL Certificate which had been used here is Wildcard SSL Certificate.

```
kubectl create secret tls tls-awx-ingress --cert=STAR_singhritesh85_com.crt --key=mykey.key -n awx --context=arn:aws:eks:us-east-2:02XXXXXXXXXX6:cluster/eks-demo-cluster-dev
```

```
[k8s-management@awx-k8s-management-node ~]$ kubectl create secret tls tls-awx-ingress --cert=STAR_singhritesh85_com.crt --key=mykey.key -n awx --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
secret/tls-awx-ingress created

[k8s-management@awx-k8s-management-node ~]$ kubectl get secrets -n awx --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev|grep "tls-awx-ingress"
tls-awx-ingress               kubernetes.io/tls    2      56s
```

Then **gateway-api.yaml** file is present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git>.

```
[k8s-management@awx-k8s-management-node ~]$ kubectl apply -f gateway-api.yaml --context=arn:aws:eks:us-east-2:02[REDACTED]6:cluster/eks-demo-cluster-dev
gateway.gateway.networking.k8s.io/awx-gateway created
httproute.gateway.networking.k8s.io/awx-http-to-https-redirect created
httproute.gateway.networking.k8s.io/awx-route-https created
```

[k8s-management@awx-k8s-management-node ~]\$ kubectl get svc -n awx --context=arn:aws:eks:us-east-2:021111111111:cluster/eks-demo-cluster-dev						PO
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	
ansible-awx-service	ClusterIP	172.20.174.97	<none>		27m	
awx-gateway-nginx	LoadBalancer	172.20.158.232	a[REDACTED].us-east-2.elb.amazonaws.com	80	:31635/TCP,443:31950/TCP 104s	
awx-operator-controller-manager-metrics-service	ClusterIP	172.20.134.2	<none>		28m	
43/TCP						84

Create the Record Set of CNAME Type in GCP Cloud DNS with DNS Name of the LoadBalancer Service as shown in the screenshot attached above.

Network Services / Zones / Zone: public-hosted-zone-logging-enabled / Create record set

Load balancing
← Create record set

Cloud DNS

Cloud CDN

Cloud NAT

Cloud Service Mesh (Traffic D...

Service Directory

Cloud Domains

Private Service Connect

SSL policies

Service Extensions

Marketplace

DNS name

(?)

Resource record type

(?)

TTL *

(?)

TTL unit

(?)

Canonical name

(?)

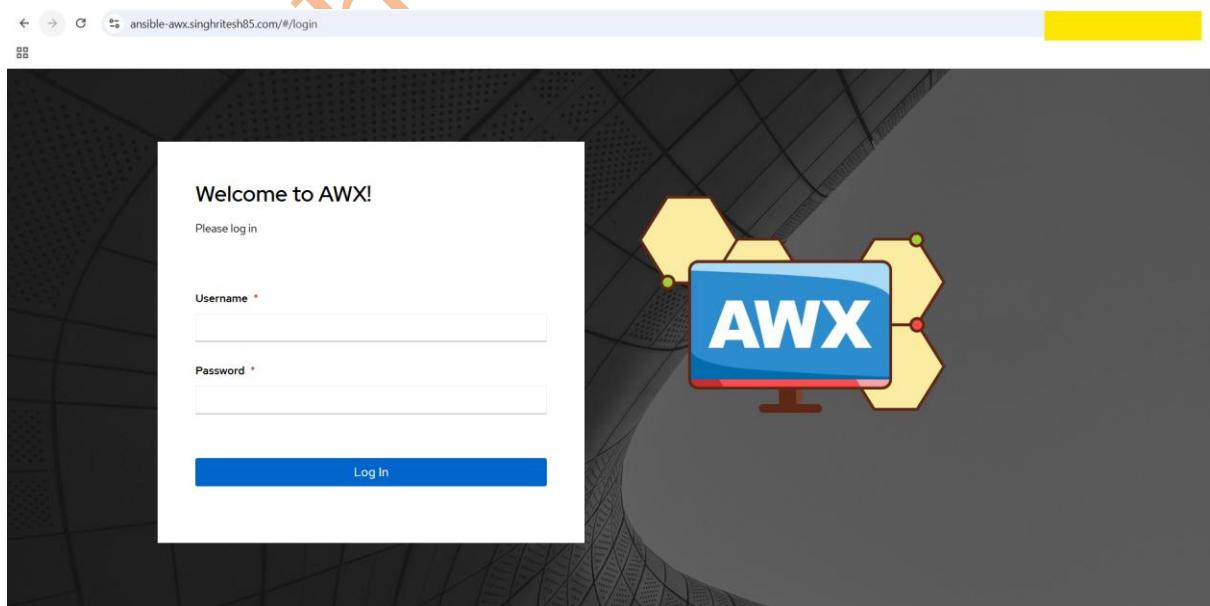
Canonical name 1 *

(?)

Example: server-1.example.com.

+ Add item
Create
Cancel

Finally, I was able to access Ansible-Awx using the URL as shown in the screenshot attached below.



Then, I had reset the Ansible-Awx **admin** user **password** using the procedure as explained below.

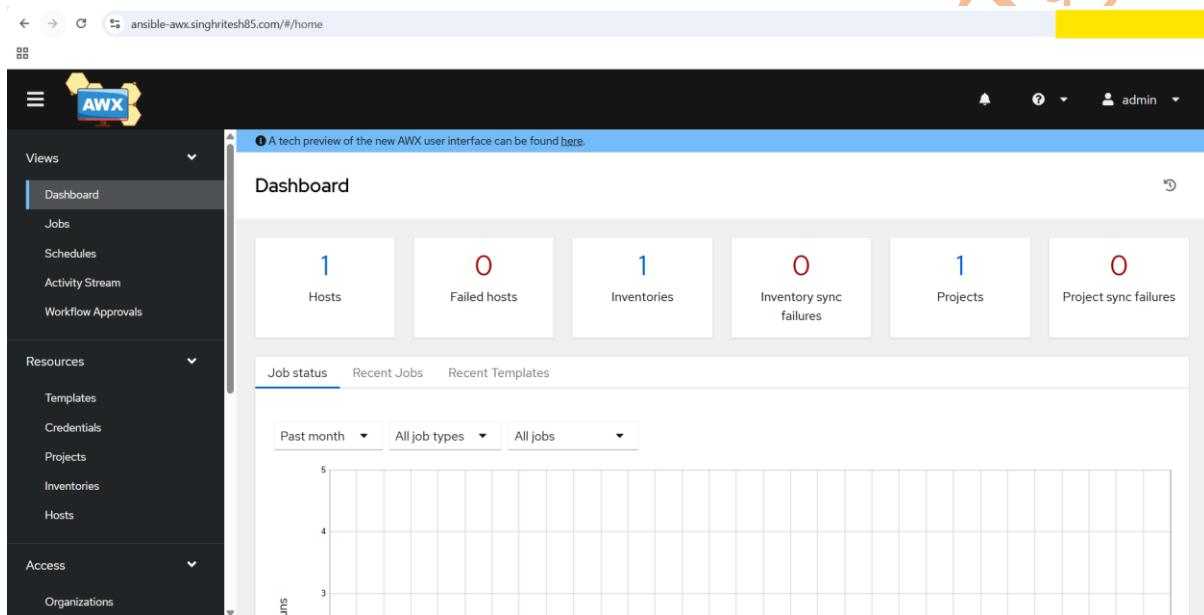
kubectl get pods -n awx

Log-into the ansible-awx-web Pod as shown below

kubectl exec -it <ansible-awx-web pod> -n awx -- bash

awx-manage update_password --username=admin --password=<provide-your-password-here>

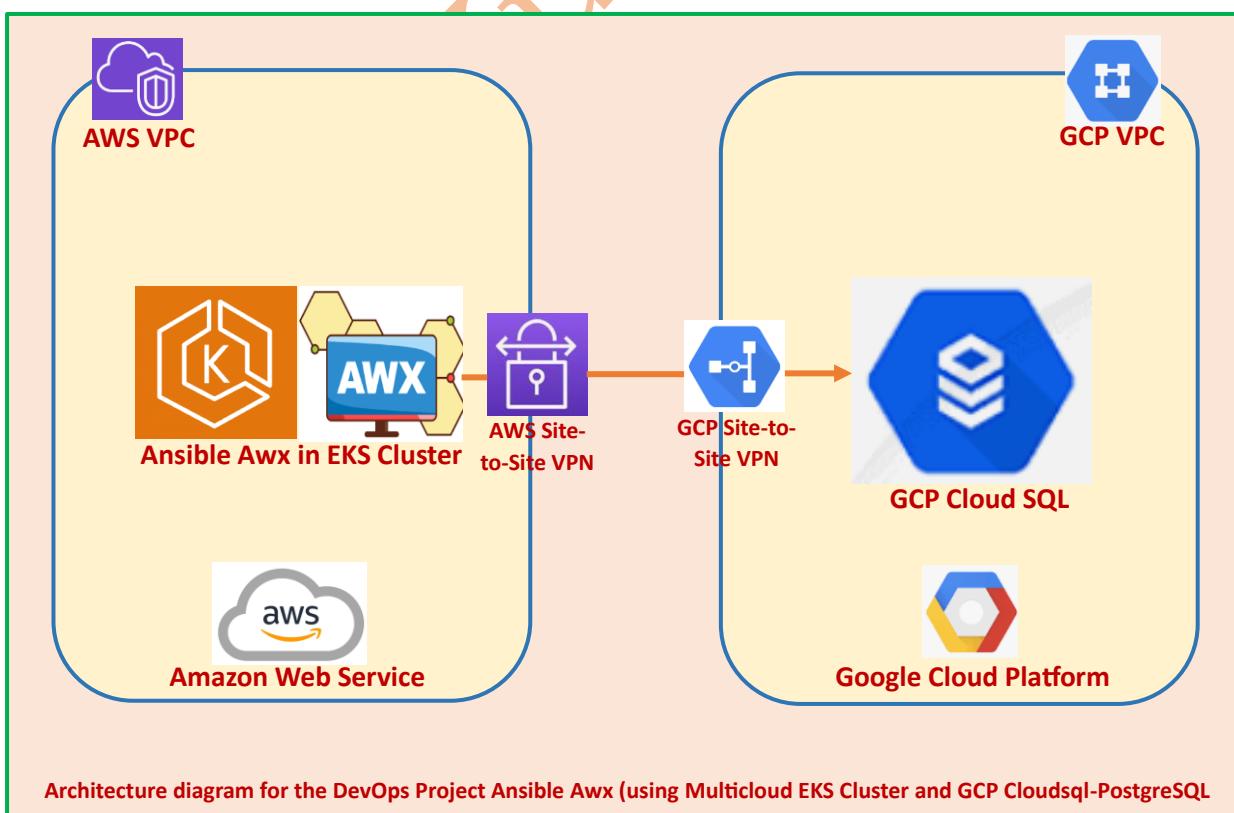
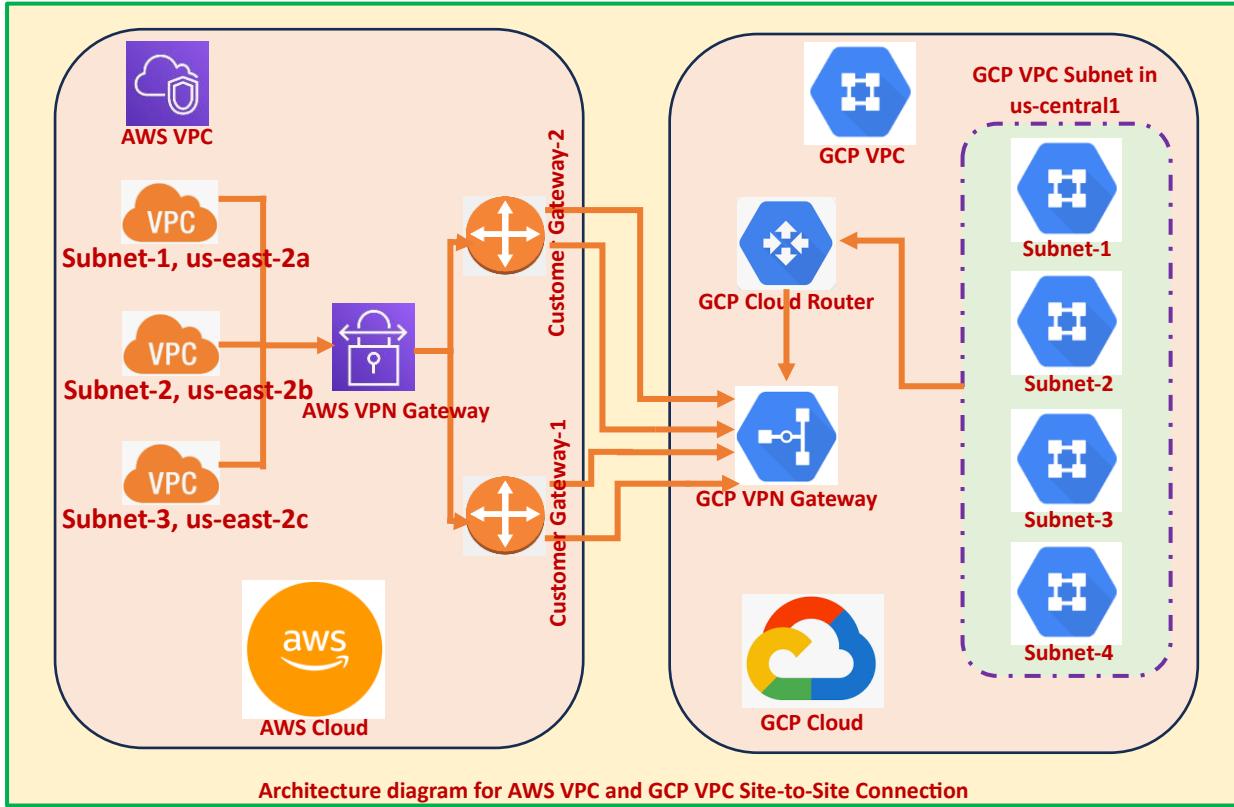
```
[k8s-management@awx-k8s-management-node main]$ kubectl get pods -n awx --context=arn:aws:eks:us-east-2:02[REDACTED] 6:cluster/eks-demo-cluster-dev
NAME          READY   STATUS    RESTARTS   AGE
ansible-awx-migration-24.6.1-[REDACTED]  0/2     Completed   0          43m
ansible-awx-task-[REDACTED]                5/5     Running    0          44m
ansible-awx-web-[REDACTED]                  4/4     Running    0          44m
awx-gateway-nginx-[REDACTED]               2/2     Running    0          18m
awx-operator-controller-manager-[REDACTED]  3/3     Running    0          45m
[k8s-management@awx-k8s-management-node main]$ kubectl exec -it ansible-awx-web-[REDACTED] -n awx --context=arn:aws:eks:us-east-2:02[REDACTED] 6:cluster/eks-demo-cluster-dev -- bash
bash-5.1$ awx-manage update_password --username=admin --password=Dexter321!
Password updated
bash-5.1$ exit
```



Now, you can proceed further with the steps as I discussed in module-1.

Module 5: - Multicloud

DevOps Project Ansible Awx (using Multicloud EKS Cluster and GCP Cloudsql-PostgreSQL)



Here I had used multicloud concept, **vendor lock-in** happens when an organisation becomes dependent on a single cloud provider for services. To get rid of such a situation organisation can opt for multicloud concept. For this project AWS Cloud and GCP Cloud had been used, in AWS Cloud Awx Pods and Service had been created and in GCP cloud CloudSQL (PostgreSQL) had been created. The Site-to-Site VPN Connection had been established between AWS Cloud VPC and GCP Cloud VPC. The password for CloudSQL (PostgreSQL) was kept in GCP Secret Manager. Awx had been accessed using the URL created with the help of Kubernetes Gateway API in AWS EKS Cluster. To create the infrastructure, Terraform had been used and it was installed in EC2 Instance and state file was kept in the S3 Bucket. The state lock in terraform had been achieved using the S3 Bucket. To authenticate and authorize GCP Account the command used was **gcloud auth application-default login** and gcloud cli had been installed using the command as shown in the screenshot attached below.

```
curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-linux-x86_64.tar.gz
tar -xf google-cloud-cli-linux-x86_64.tar.gz
./google-cloud-sdk/install.sh
```

```
[root@yellow opt]# curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-linux-x86_64.tar.gz
% Total    % Received % Xferd  Average Speed   Time     Time      Current
                                         Dload  Upload Total   Spent    Left  Speed
100 194M  100 194M    0     0  139M    0:00:01  0:00:01 --:--:--  139M
[root@yellow opt]# tar -xf google-cloud-cli-linux-x86_64.tar.gz
[root@yellow opt]# ./google-cloud-sdk/install.sh
```

Logout then login, finally run the command **gcloud auth application-default login**.

```
[root@yellow ~]# gcloud auth application-default login
```

The Terraform script to create the resources in AWS and GCP cloud and to establish the VPN Connection between AWS Cloud VPC and GCP Cloud VPC present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **multicloud/terraform-aws-eks-cluster-and-gcp-cloudsql**.

Then you can run the below commands

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created. Then you can run the command

terraform apply -auto-approve -----> Finally, Create the resources.

```

module.eks_cluster.aws_eks_addon.core_dns: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m10s elapsed]
module.eks_cluster.aws_eks_addon.core_dns: Creation complete after 14s [id=eks-demo-cluster-dev:coredns]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m20s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m30s elapsed]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.metrics_server: Still creating... [00m40s elapsed]
module.eks_cluster.aws_eks_addon.core_dns: Creation complete after 44s [id=eks-demo-cluster-dev:metrics-server]
module.eks_cluster.aws_eks_addon.ebs_csi_driver: Creation complete after 44s [id=eks-demo-cluster-dev:aws-ebs-csi-driver]
module.eks_cluster.aws_eks_addon.amazon_cloudwatch_observability: Creation complete after 45s [id=eks-demo-cluster-dev:amazon-cloudwatch-observability]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Still creating... [00m50s elapsed]
module.eks_cluster.aws_eks_addon.csi_snapshot_controller: Creation complete after 55s [id=eks-demo-cluster-dev:snapshot-controller]

Apply complete! Resources: 116 added, 0 changed, 0 destroyed.

Outputs:

aws_ec2_eks_and_karpenter_iam_role_and_gcp_cloudsql_details = {
  "cloud_sql_connection_name" = "awsx-private-dbinstance-[REDACTED]-us-central1:awx-private-dbinstance-[REDACTED]"
  "cloud_sql_db_instance_name" = "awx-private-dbinstance-d5e8dd3a"
  "cloud_sql_private_ip_address" = "10.55.122.3"
  "eks_cluster_endpoint" = "https://[REDACTED].us-east-2.eks.amazonaws.com"
  "eks_cluster_name" = "eks-demo-cluster-dev"
  "k8s_management_instance_id" = "i-000000000000000000"
  "k8s_management_private_ip" = "192.168.4.61"
  "karpenter_node_spun_iam_role_arn" = "arn:aws:iam::02-[REDACTED]6:role/karpenter-eks-noderole"
}

```

```
[k8s-management@k8s-management ~]$ kubectl get nodes
NAME                               STATUS   ROLES      AGE     VERSION
ip-192-168-1-79.us-east-2.compute.internal   Ready    <none>   5m15s   v1.33.0-eks-[REDACTED]
ip-192-168-2-212.us-east-2.compute.internal   Ready    <none>   5m14s   v1.33.0-eks-[REDACTED]
ip-192-168-3-167.us-east-2.compute.internal   Ready    <none>   5m14s   v1.33.0-eks-[REDACTED]
```

Created GCP Cloud DNS Zone as explained below. The Terraform script to create the GCP Cloud DNS Zone present in the GitHub Repo <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git> at the path **multicloud/terraform-gcp-cloud-dns**.

terraform init -----> initializes a working directory containing configuration files and installs plugins for required providers.

terraform validate -----> verify that terraform configuration file is correct or not

terraform plan -----> Check which resources are going to be created. Then you can run the command

terraform apply -auto-approve -----> Finally, Create the resources.

```

        + enable_logging = true
    }

    + dnssec_config {
        + kind      = "dns#managedZoneDnsSecConfig"
        + non_existence = (known after apply)
        + state      = "on"

        + default_key_specs (known after apply)
    }
}

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ gcp_cloud_dns_name_and_dns_name = {
    + dns_zone_nameservers = (known after apply)
    + zone_dns_name       = "singhritesh85.com."
    + zone_name            = "public-hosted-zone-logging-enabled"
}
module.gcp_cloud_dns.google_dns_managed_zone.cloud_logging_enabled_zone: Creating...
module.gcp_cloud_dns.google_dns_managed_zone.cloud_logging_enabled_zone: Creation complete after 1s [id=projects/[REDACTED]/managedZones/public-hosted-zone-logging-enabled]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

gcp_cloud_dns_name_and_dns_name = {
    "dns_zone_nameservers" = tolist([
        "",
        "",
        "",
        "",
        ""
    ])
    "zone_dns_name" = "singhritesh85.com."
    "zone_name"     = "public-hosted-zone-logging-enabled"
}

```

After running the Terraform Script update your domain provider with the generated Nameserver of the Zone of GCP Cloud DNS.

Installation of Awx Operator using Helm chart.

```

helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
helm repo update
helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace

```

```

[k8s-management@k8s-management ~]$ helm repo add awx-operator https://ansible-community.github.io/awx-operator-helm/
"awx-operator" has been added to your repositories
[k8s-management@k8s-management ~]$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "awx-operator" chart repository
Update Complete. ⚡Happy Helming!⚡
[k8s-management@k8s-management ~]$ helm install ansible-awx-operator awx-operator/awx-operator -n awx --create-namespace

```

```
[k8s-management@k8s-management ~]$ cat awx-postgres-secret-multicloud.yaml
apiVersion: v1
kind: Secret
metadata:
  name: awx-postgres-configuration
  namespace: awx
stringData:
  host: "10.55.122.3"
  port: "5432"
  database: "dexter"
  username: "dbadmin"
  password: "XXXXXXXXXX"
  sslmode: "prefer" # or 'require' if you enforce SSL on RDS
  type: "unmanaged"
type: Opaque
[k8s-management@k8s-management ~]$ kubectl apply -f awx-postgres-secret-multicloud.yaml
secret/awx-postgres-configuration created
[k8s-management@k8s-management ~]$ cat ansible-awx.yaml
apiVersion: awx.ansible.com/v1beta1
kind: AWX
metadata:
  name: ansible-awx
  namespace: awx
spec:
  postgres_configuration_secret: awx-postgres-configuration
  service_type: ClusterIP
  admin_user: admin
[k8s-management@k8s-management ~]$ kubectl apply -f ansible-awx.yaml
awx.ansible.com/ansible-awx created
```

```
[k8s-management@k8s-management ~]$ kubectl get pods -n awx
NAME                               READY   STATUS    RESTARTS   AGE
ansible-awx-migration-24.6.1-[REDACTED]   0/1     Completed  0          7m39s
ansible-awx-task-[REDACTED]                4/4     Running   0          8m52s
ansible-awx-web-[REDACTED]                 3/3     Running   0          8m54s
awx-operator-controller-manager-[REDACTED]   2/2     Running   0          10m
[k8s-management@k8s-management ~]$ kubectl get svc -n awx
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
ansible-awx-service   ClusterIP  10. [REDACTED].78  <none>    80/TCP    9m5s
awx-operator-controller-manager-metrics-service   ClusterIP  10. [REDACTED].128 <none>    8443/TCP  10m
```

Created Kubernetes Gateway API in EKS as discussed in Module-1 as shown in screenshot attached below.

```
[k8s-management@k8s-management ~]$ kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.4.0/standard-install.yaml
[REDACTED]
helm upgrade --install ngf oci://ghcr.io/nginx/charts/nginx-gateway-fabric --create-namespace -n nginx-gateway
[REDACTED]
customresourcedefinition.apiextensions.k8s.io/backendlspolicies.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/gatewayclasses.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/gateways.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/grpcroutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/httproutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/referencegrants.gateway.networking.k8s.io created
Release "ngf" does not exist. Installing it now.
Pulled: ghcr.io/nginx/charts/nginx-gateway-fabric:2.4.1
Digest: [REDACTED]
NAME: ngf
LAST DEPLOYED: Mon Feb 16 09:47:25 2020
NAMESPACE: nginx-gateway
STATUS: deployed
REVISION: 1
TEST SUITE: None
[k8s-management@k8s-management ~]$ kubectl create secret tls tls-awx-ingress --cert=STAR_singhritesh85_com.crt --key=mykey.key -n awx --context=arn:aws:eks:us-east-2:02-[REDACTED]:cluster/eks-demo-cluster-dev
secret/tls-awx-ingress created
[k8s-management@k8s-management ~]$ kubectl get svc -n awx
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
ansible-awx-service   ClusterIP  10. [REDACTED].78  <none>    80/TCP    18m
awx-operator-controller-manager-metrics-service   ClusterIP  10. [REDACTED].128 <none>    8443/TCP  20m
[k8s-management@k8s-management ~]$ kubectl apply -f gateway-api.yaml
gateway.gateway.networking.k8s.io/awx-gateway created
httproute.gateway.networking.k8s.io/awx-http-to-https-redirect created
httproute.gateway.networking.k8s.io/awx-route-https created
[k8s-management@k8s-management ~]$ kubectl get svc -n awx
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
ansible-awx-service   ClusterIP  10. [REDACTED].78  <none>    80/TCP    19m
awx-gateway-nginx   LoadBalancer 10. [REDACTED].244  a [REDACTED].us-east-2.elb.amazonaws.com  80 :31719/TCP,443:31000/TCP  6s
awx-operator-controller-manager-metrics-service   ClusterIP  10. [REDACTED].128 <none>    8443/TCP  20m
```

I did the entry for LoadBalancer Service DNS Name in GCP Cloud DNS Zone to create the Record Set of CNAME Type as shown in the screenshot attached below.

Network Services / Zones / Zone: public-hosted-zone-logging-enabled / Create record set

Load balancing

Cloud DNS

Cloud CDN

Cloud NAT

Cloud Service Mesh (Traffic D...)

Service Directory

Cloud Domains

Private Service Connect

SSL policies

Service Extensions

Marketplace

Create record set

DNS name: ansible-awx .singhritesh85.com.

Resource record type: CNAME

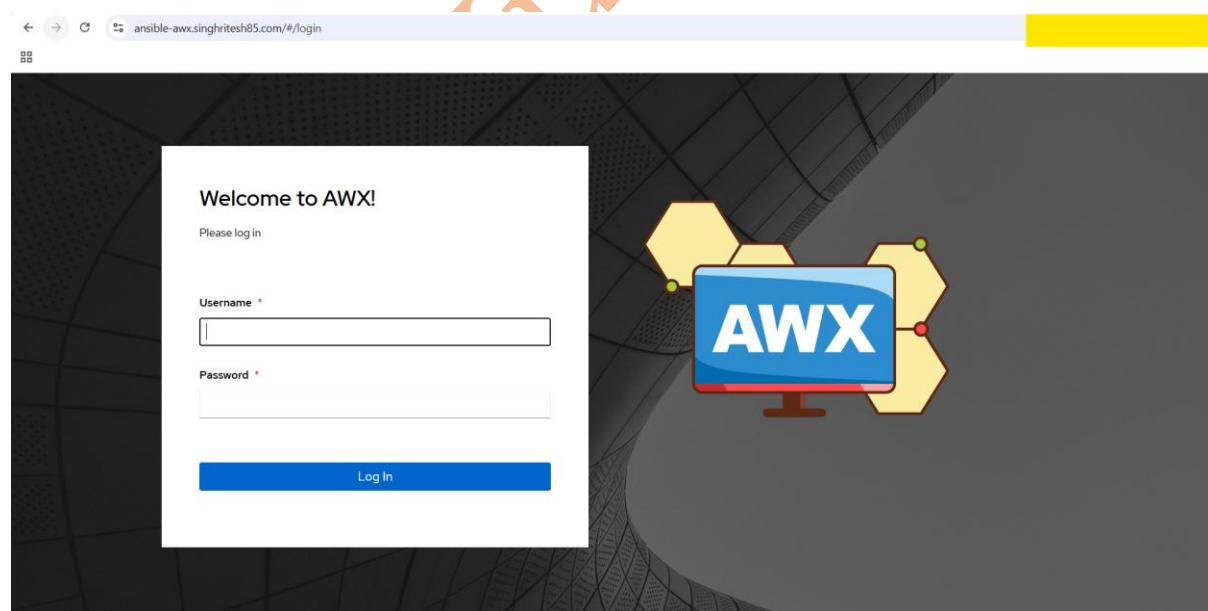
TTL: 5

TTL unit: minutes

Canonical name: Canonical name 1 * - a [REDACTED] 3.us-east-2.elb.ama

Create Cancel

Finally, I was able to access the Ansible-Awx as shown in the screenshot attached below.



Now, I will reset the Ansible-Awx **admin** user **password** using the procedure as explained below.

```
kubectl get pods -n awx
```

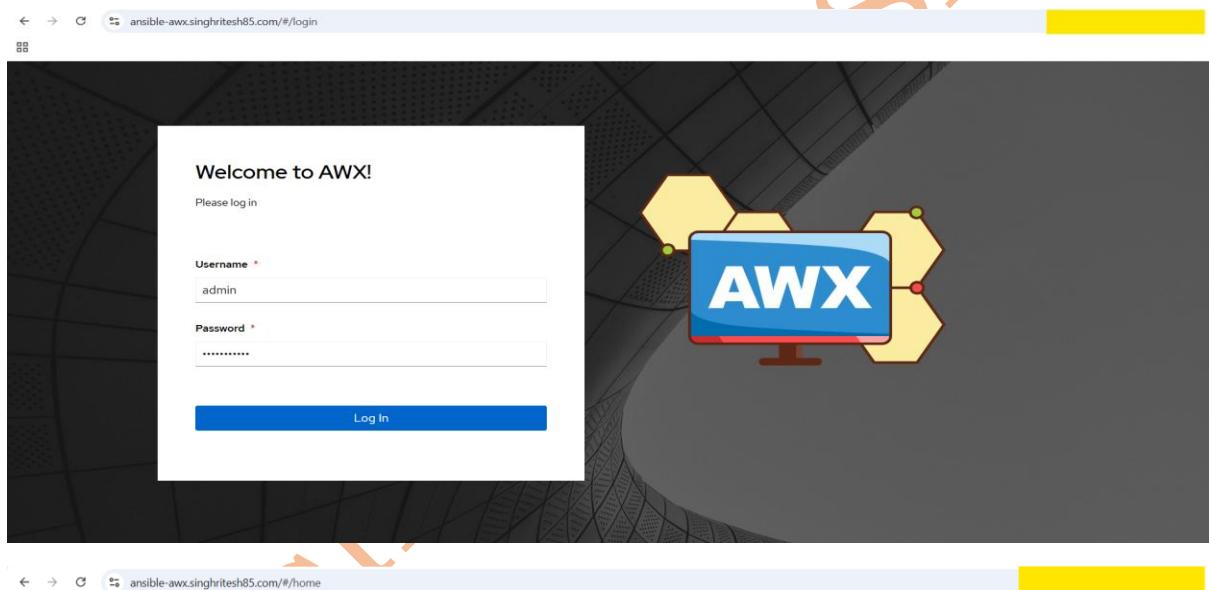
Log-into the ansible-awx-web Pod as shown below

```
kubectl exec -it <ansible-awx-web pod> -n awx -- bash
```

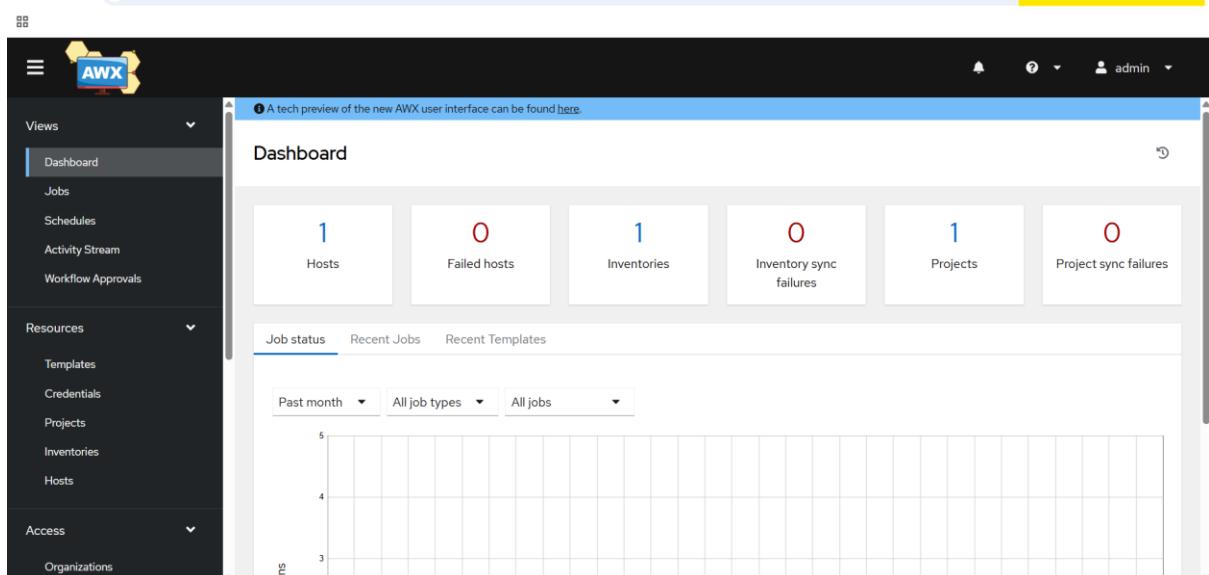
```
awx-manage update_password --username=admin --password=<provide-your-password-here>
```

```
[k8s-management@k8s-management ~]$ kubectl get pods -n awx
NAME                               READY   STATUS    RESTARTS   AGE
ansible-awx-migration-24.6.1-      0/1     Completed  0          51m
ansible-awx-task-                  4/4     Running   0          53m
ansible-awx-web-                  3/3     Running   0          53m
awx-gateway-nginx-                1/1     Running   0          33m
awx-operator-controller-manager-   2/2     Running   0          54m
[k8s-management@k8s-management ~]$ kubectl exec -it ansible-awx-web-[REDACTED] -n awx -- bash
bash-5.1$ awx-manage update_password --username=admin --password=[REDACTED]
Password updated
```

Then, I logged-into the Ansible-Awx using the updated password as shown below.



The screenshot shows the Ansible-Awx login interface. The URL in the browser is `ansible-awx.singhritesh85.com/#/login`. The form has 'Username' set to 'admin' and 'Password' set to '*****'. A blue 'Log In' button is at the bottom. To the right of the form is a decorative graphic featuring the word 'AWX' in a blue hexagonal frame, surrounded by yellow hexagons.



The screenshot shows the Ansible-Awx dashboard after log-in. The URL in the browser is `ansible-awx.singhritesh85.com/#/home`. The top navigation bar shows the user is logged in as 'admin'. The left sidebar menu includes 'Views' (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), 'Resources' (Templates, Credentials, Projects, Inventories, Hosts), and 'Access' (Organizations). The main dashboard area displays various metrics: 1 Hosts, 0 Failed hosts, 1 Inventories, 0 Inventory sync failures, 1 Projects, and 0 Project sync failures. Below these metrics is a chart showing job status over the past month, with the Y-axis ranging from 3 to 5. A message at the top of the dashboard says, 'A tech preview of the new AWX user interface can be found [here](#)'.

Now, you can proceed further with the steps as I discussed in module-1.

GitHub Repo: - <https://github.com/singhritesh85/DevOps-Project-Ansible-Awx.git>

Helm Repo: - <https://github.com/singhritesh85/helm-repo-for-bitnami-postgresql-ha.git>

<https://github.com/singhritesh85/helm-repo-for-bitnami-postgresql-ha-onlyservice.git>

Ritesh Kumar Singh