K8s deployment with helm and *Jenkins* shared Library.

Problem Statement

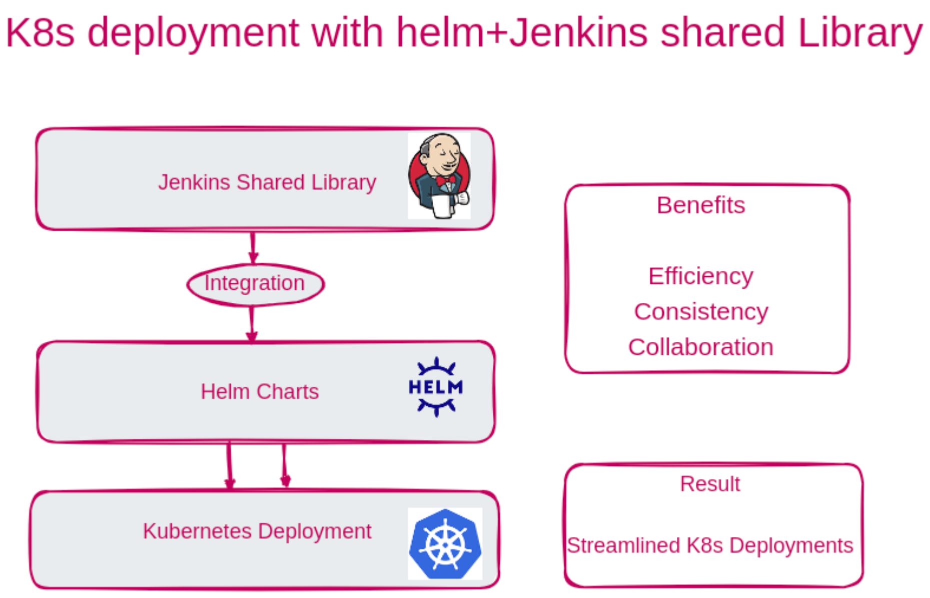
In current software deployment workflow, managing Kubernetes (K8s) applications is plagued by manual, error-prone processes that hinder our ability to scale and maintain efficiency. The absence of a standardized deployment framework creates challenges such as inconsistent configurations, time-consuming deployments, and a lack of scalability. Furthermore, the lack of a centralized and reusable approach for Jenkins pipelines complicates CI/CD management and increases the risk of errors.
To address these challenges, there is a critical need for a technical document that outlines a solution for K8s deployment utilizing Helm and Jenkins shared libraries. This document is essential to automate and streamline our deployment processes, reducing manual errors and improving scalability. It will also provide a unified approach to managing Jenkins pipelines, enhancing CI/CD efficiency. By addressing these complexities, we aim to optimize our deployment workflows, leading to increased operational efficiency and a reduction in deployment-related issues.

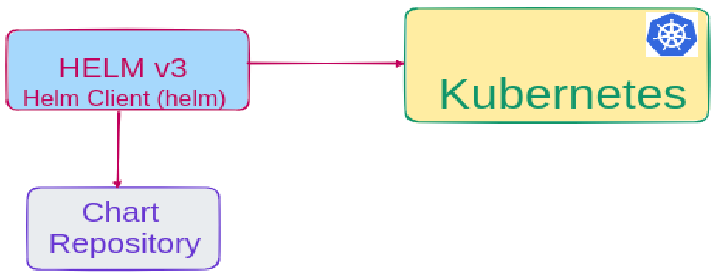
Objective

The objective of this document is to provide a step-by-step guide to solving the problem mentioned above by implementing Kubernetes deployment with Helm and Jenkins Shared Library. We will create a dynamic Helm module and utilize Jenkins Shared Library for deploying resources. We will deploy a simple Nginx application as an example.

The goal of this experiment is to set up a streamlined process for deploying simple nginx applications to a Kubernetes cluster using Helm and Jenkins Shared Library. The challenges addressed include:

- Efficiently managing Helm chart values for deployment.
- Automating the deployment process with Jenkins pipelines.
- Ensuring scalability and reusability in larger projects.





Prerequisites

Before proceeding, ensure that the following prerequisites are met:

Tools and Technologies

- A Kubernetes cluster is set up and configured with *kubectl* access.
- Helm is installed on system and Jenkins server.
- Jenkins is installed and configured with the necessary plugins.

Value Proposition and Benefit

Benefits

- Reusability: Jenkins Shared Library allows us to reuse deployment logic and scripts across multiple projects and applications.
- Automation: The deployment process is automated, reducing manual intervention and human errors.
- Customization: Helm charts and values can be easily customized for different environments (development, staging, production) without modifying the application code.
- Version Control: The entire deployment process, including Helm charts, values, and Jenkins pipelines, can be version-controlled, making it easy to track changes and rollbacks.
- Consistency: Ensures consistent deployment practices across the organization, reducing deployment-related issues.

Use Cases

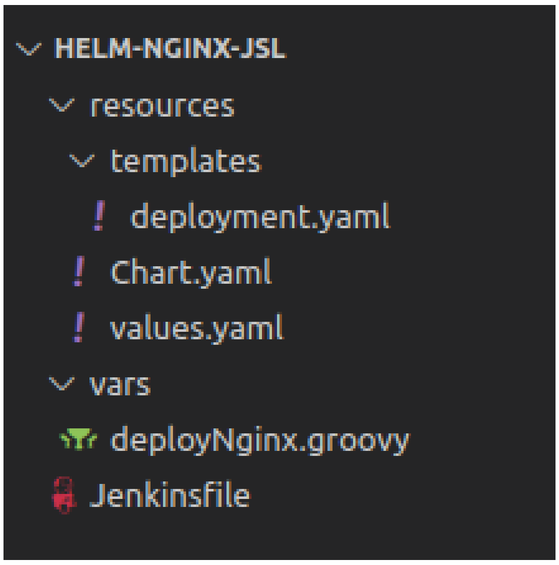
- Multi-Application Environments: When managing multiple applications within the same Kubernetes cluster, Jenkins Shared Library can help standardize and simplify the deployment process.
- Multi-Environment Deployments: For applications that need to be deployed to various environments (e.g., dev, test, prod), using dynamic Helm charts and values ensures that deployments are consistent and environment specific.
- Continuous Integration/Continuous Deployment (CI/CD): Integrating Jenkins Shared Library with CI/CD pipelines streamlines the deployment of applications as part of the automated testing and release process.
- Rollbacks and Updates: In cases where we need to roll back to a previous version or update an application, using Helm and Jenkins Shared Library allows for easy management of releases.

Execution Details

1. Directory Structure

Project directory structure is organized as follows:

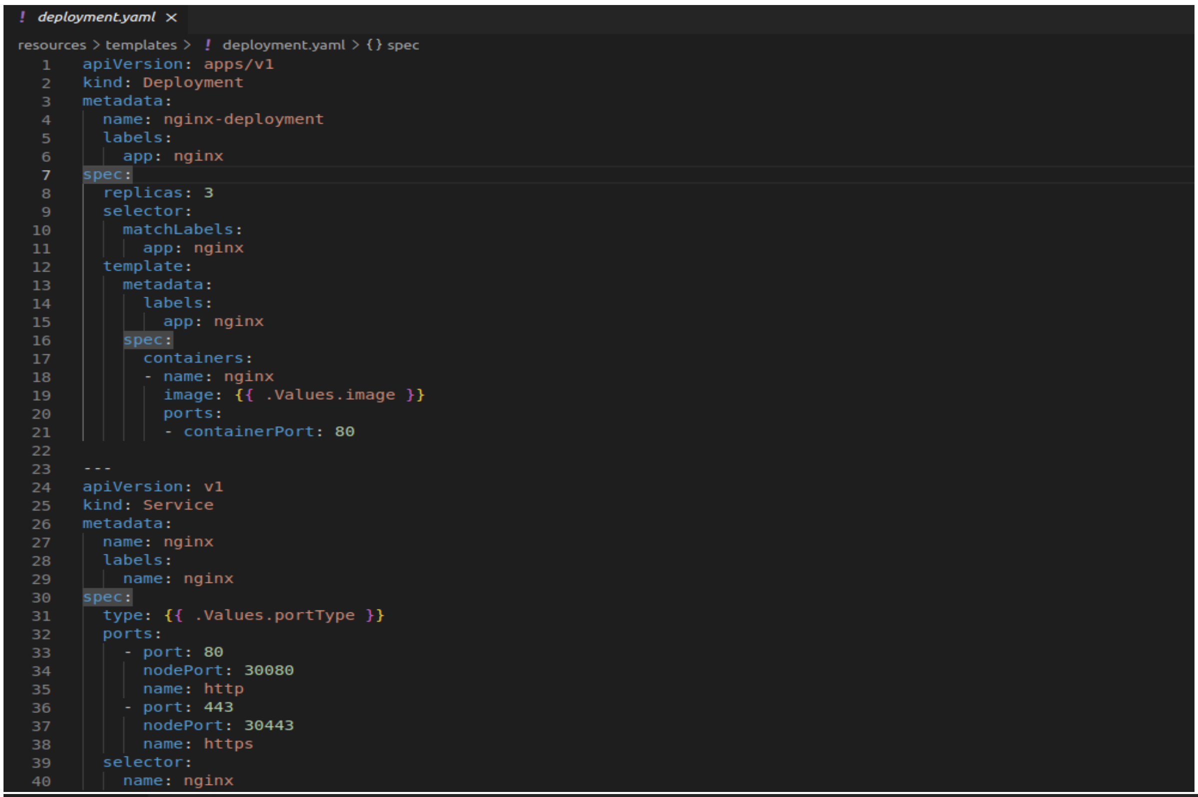
```
Helm-nginx-share/
  └── resources/
        └ template/
              └ deployment.yaml
        └ Chart.yaml
        └ values.yaml
  └ vars/
        └ deployNginx.groovy
  └ Jenkinsfile
```



```
HELM-NGINX-JSL
  resources
    templates
      deployment.yaml
    Chart.yaml
    values.yaml
  vars
    deployNginx.groovy
  Jenkinsfile
```

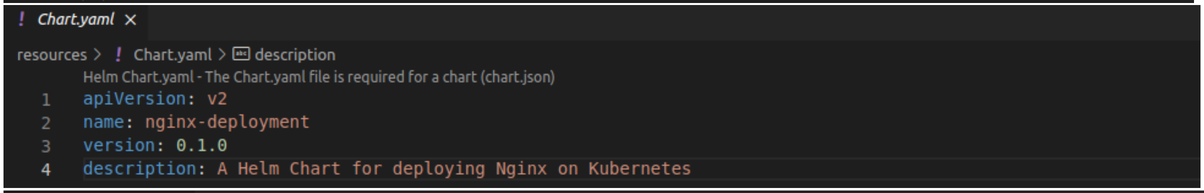
2. Creating a Helm Chart for nginx

Create a Helm chart for deploying nginx which will contain deployment.yaml, Chart.yaml and values.yaml

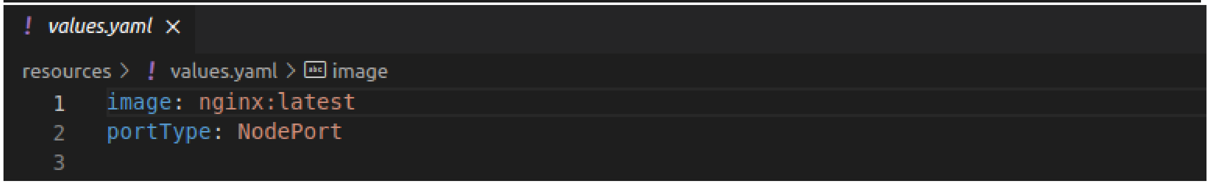


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: {{ .Values.image }}
        ports:
        - containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  type: {{ .Values.portType }}
  ports:
    - port: 80
      nodePort: 30080
      name: http
    - port: 443
      nodePort: 30443
      name: https
  selector:
    name: nginx
```


! Chart.yaml ×
resources > ! Chart.yaml > description
Helm Chart.yaml - The Chart.yaml file is required for a chart (chart.json)

```yaml
apiVersion: v2
name: nginx-deployment
version: 0.1.0
description: A Helm Chart for deploying Nginx on Kubernetes
```



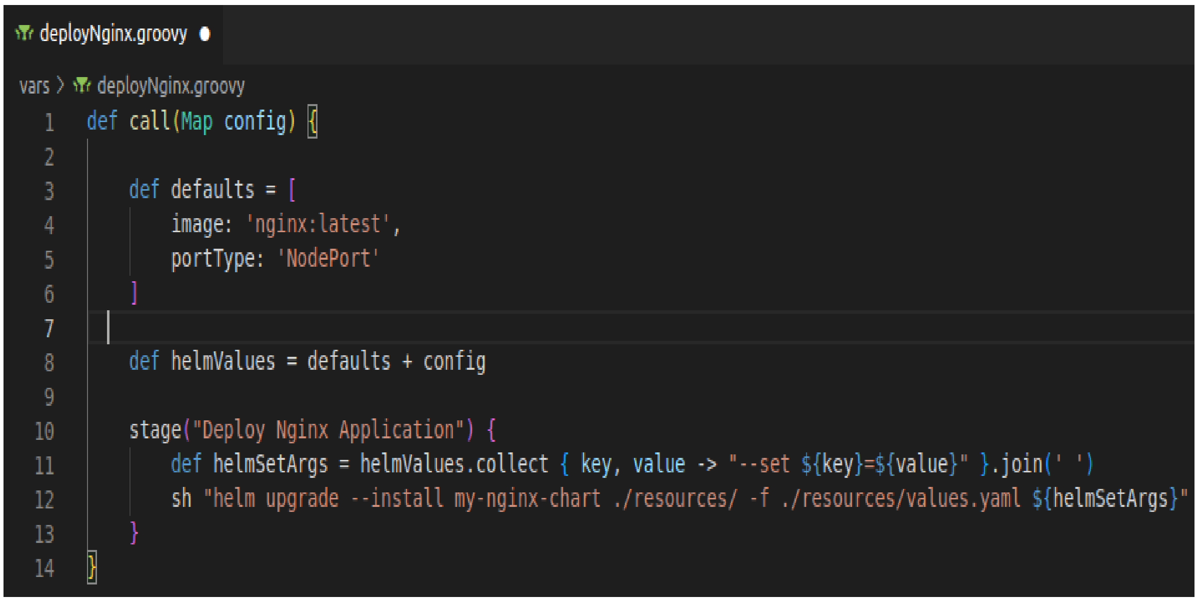
```yaml
image: nginx:latest
portType: NodePort

```

Push this code to GitHub repository form where it will be accessed by Jenkins.

3. Jenkins Shared Library Setup

Jenkins Shared Library extends Jenkins. It promotes code reuse by letting we store and share pipeline code. It enhances organization, customization, and version control for our automation workflows.
Create a Jenkins Shared Library to centralize the deployment logic. Use the deployNginx.groovy script.



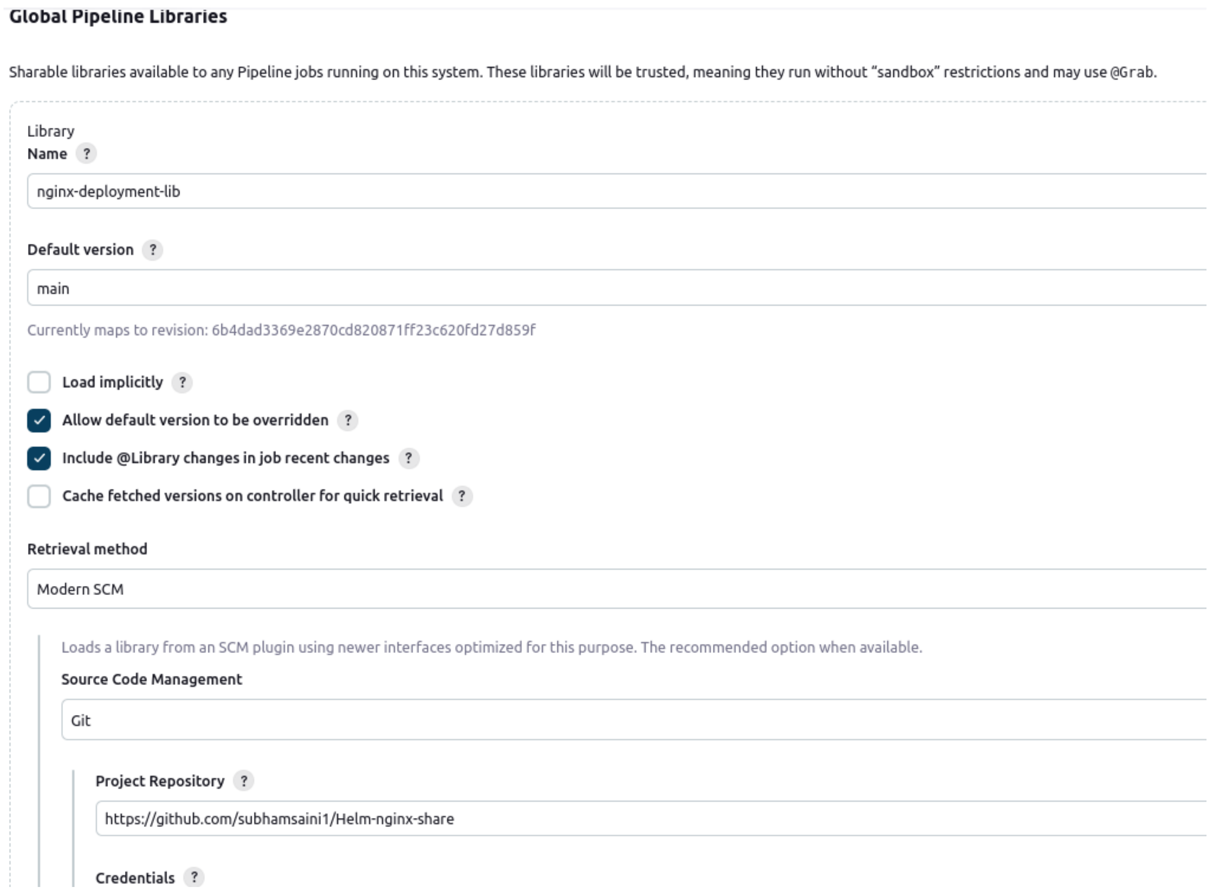
```
deployNginx.groovy
vars > deployNginx.groovy
def call(Map config) {

    def defaults = [
        image: 'nginx:latest',
        portType: 'NodePort'
    ]

    def helmValues = defaults + config

    stage("Deploy Nginx Application") {
        def helmSetArgs = helmValues.collect { key, value -> "--set ${key}=${value}" }.join(' ')
        sh "helm upgrade --install my-nginx-chart ./resources/ -f ./resources/values.yaml ${helmSetArgs}"
    }
}
```

In Jenkins, configure the library under Manage Jenkins -> Configure System -> Global Pipeline Libraries. Set a unique name (e.g., nginx-deployment-lib) and provide the library's source code location (e.g., Git repository URL).



Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.

Library
Name ?
nginx-deployment-lib

Default version ?
main
Currently maps to revision: 6b4dad3369e2870cd820871ff23c620fd27d859f

☐ Load implicitly ?
☑ Allow default version to be overridden ?
☑ Include @Library changes in job recent changes ?
☐ Cache fetched versions on controller for quick retrieval ?

Retrieval method
Modern SCM

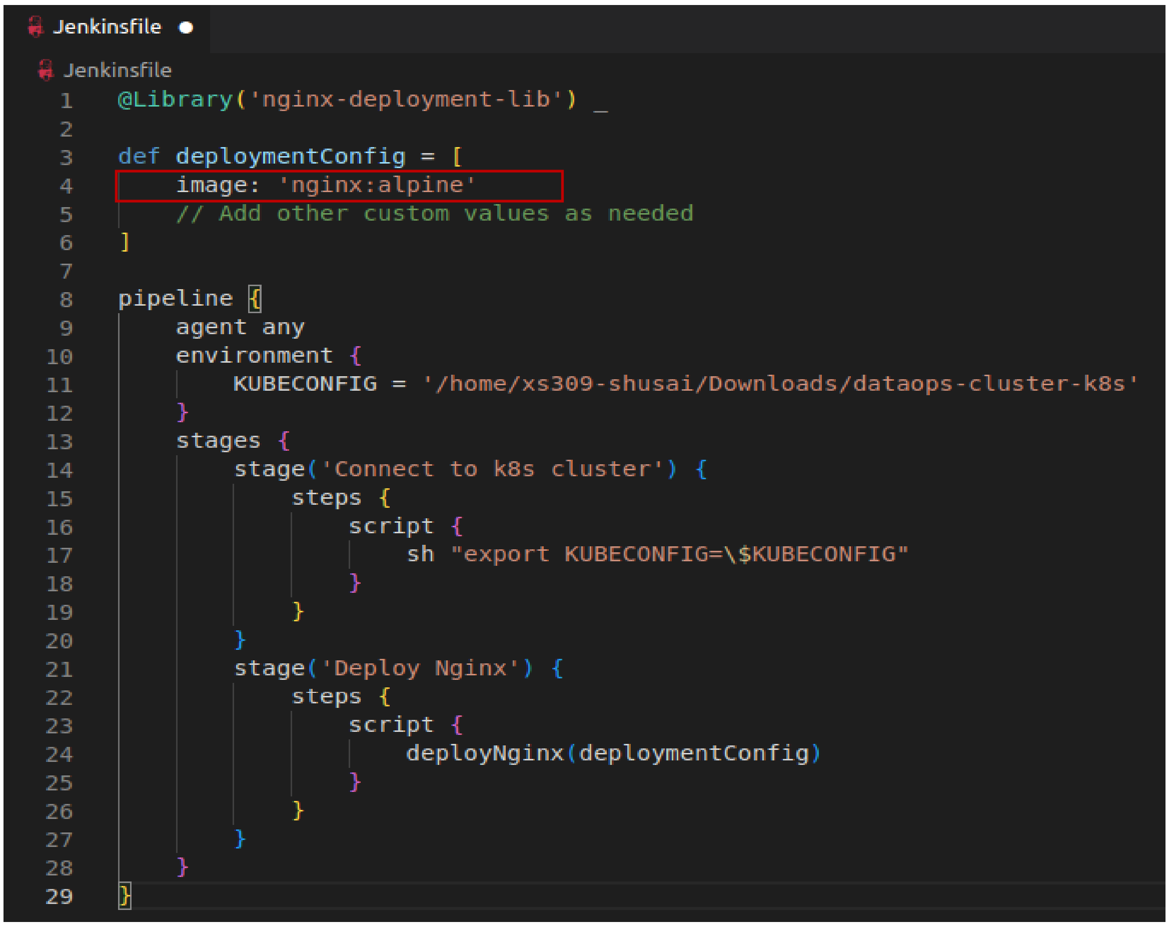
Loads a library from an SCM plugin using newer interfaces optimized for this purpose. The recommended option when available.
Source Code Management
Git

Project Repository ?
https://github.com/subhamsaini1/Helm-nginx-share

Credentials ?

4. Jenkins Pipeline Configuration

Create a Jenkins pipeline (Jenkinsfile) for Nginx deployment. Use the @Library annotation to reference shared library.
Define deployment configuration in the deploymentConfig map within the pipeline.
Set up pipeline stages, including setting KUBECONFIG and calling the deployNginx function.



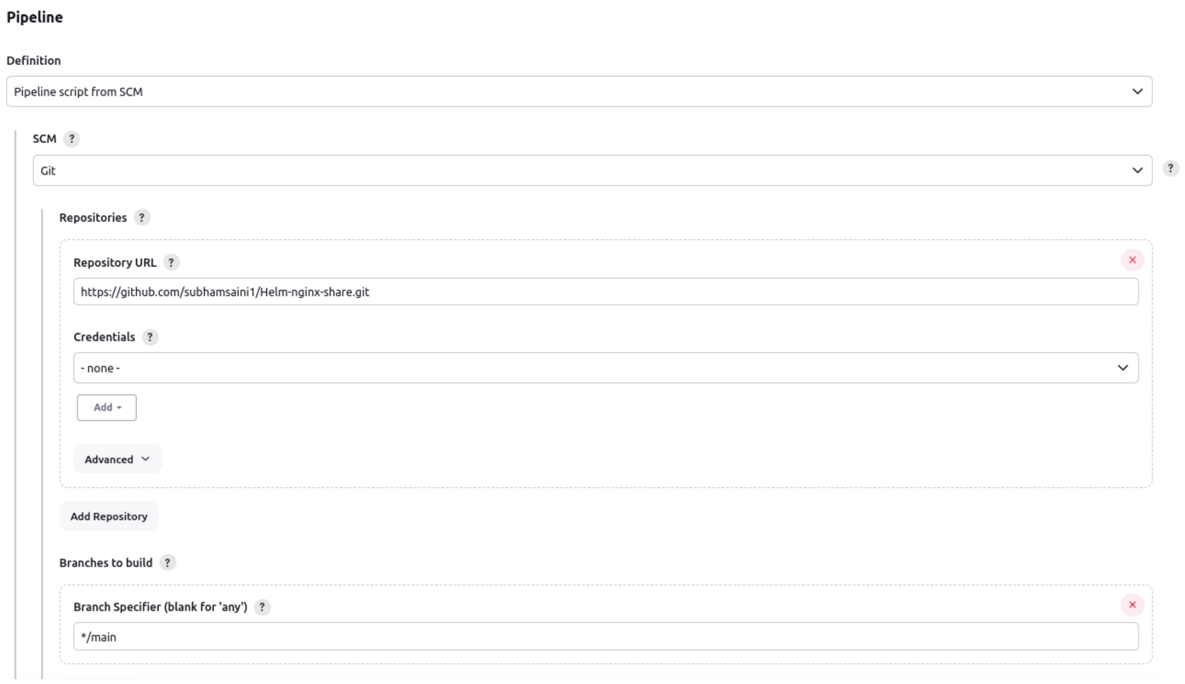
```
@Library('nginx-deployment-lib') _

def deploymentConfig = [
    image: 'nginx:alpine'
    // Add other custom values as needed
]

pipeline {
    agent any
    environment {
        KUBECONFIG = '/home/xs309-shusai/Downloads/dataops-cluster-k8s'
    }
    stages {
        stage('Connect to k8s cluster') {
            steps {
                script {
                    sh "export KUBECONFIG=\$KUBECONFIG"
                }
            }
        }
        stage('Deploy Nginx') {
            steps {
                script {
                    deployNginx(deploymentConfig)
                }
            }
        }
    }
}
```

5. Deploying Nginx to Kubernetes

Trigger the Jenkins pipeline manually or through automation.
Go to Dashboard ---> nginxpodjsl --> Configuration from here pipeline with SCM(Git) is created.


Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/subhamsaini1/Helm-nginx-share.git

Credentials

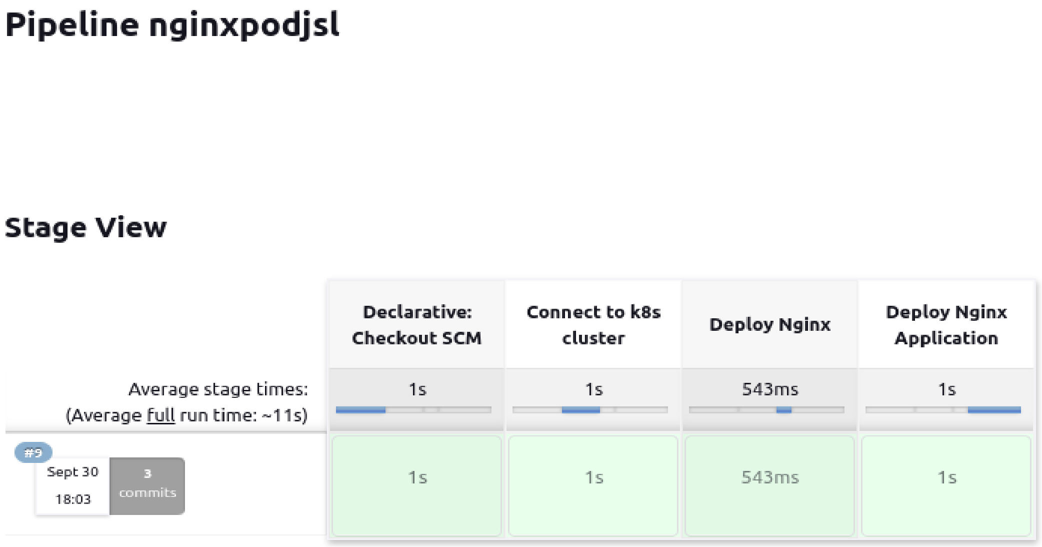
- none -

Branches to build

Branch Specifier (blank for 'any')

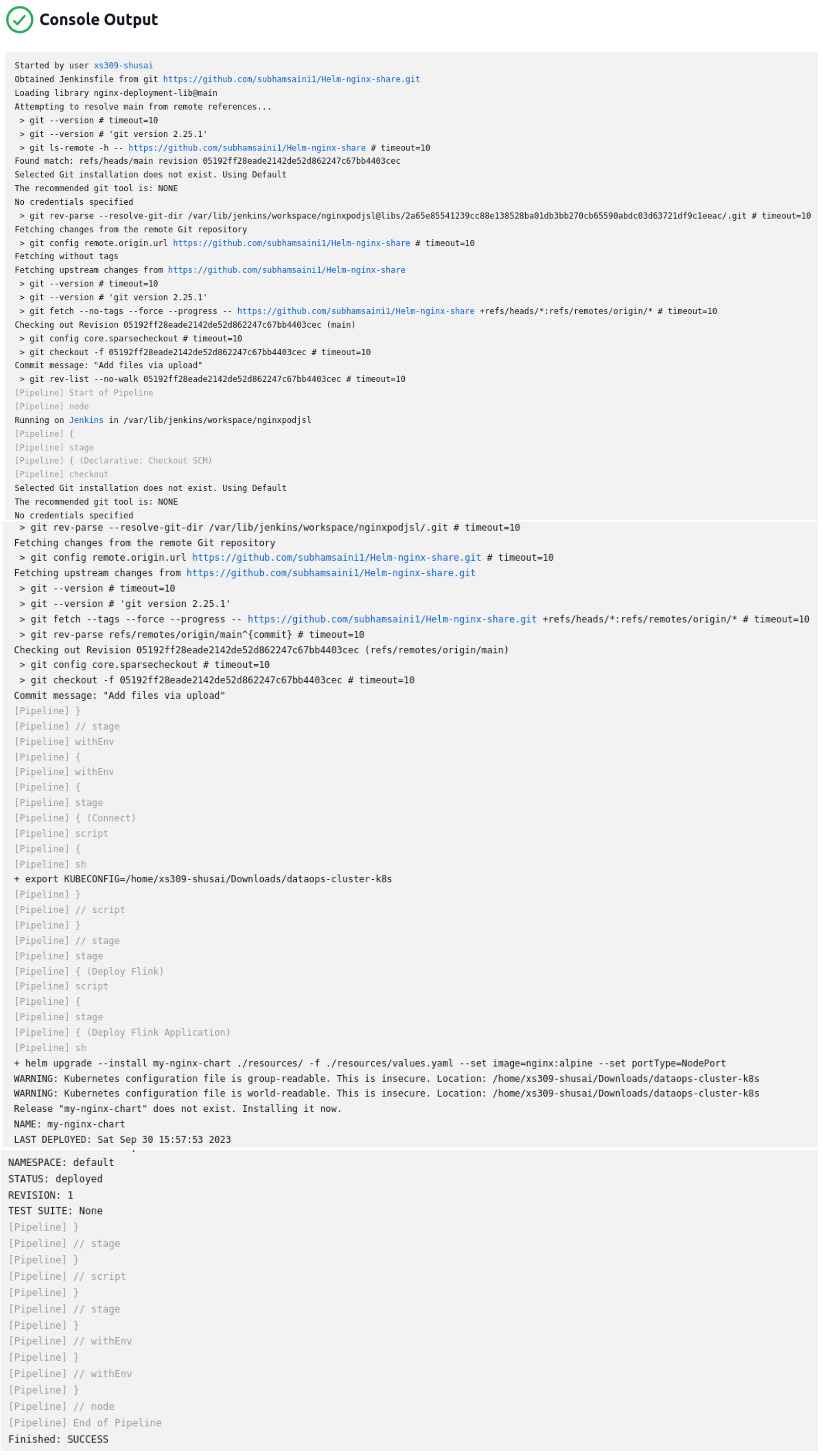
*/main

Click Save and Build. Jenkins will execute the pipeline, set the KUBECONFIG, and deploy Nginx to Kubernetes cluster based on the provided configuration.


Pipeline nginxpodjsl

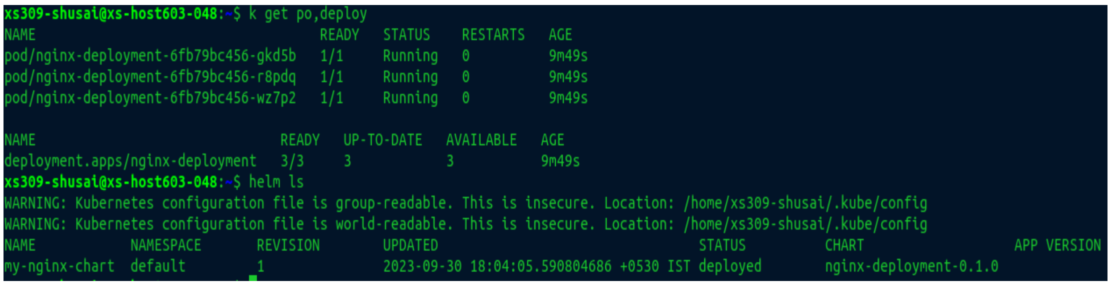
Stage View

	Declarative: Checkout SCM	Connect to k8s cluster	Deploy Nginx	Deploy Nginx Application
Average stage times: (Average full run time: ~11s)	1s	1s	543ms	1s
#9 Sept 30 18:03 3 commits	1s	1s	543ms	1s



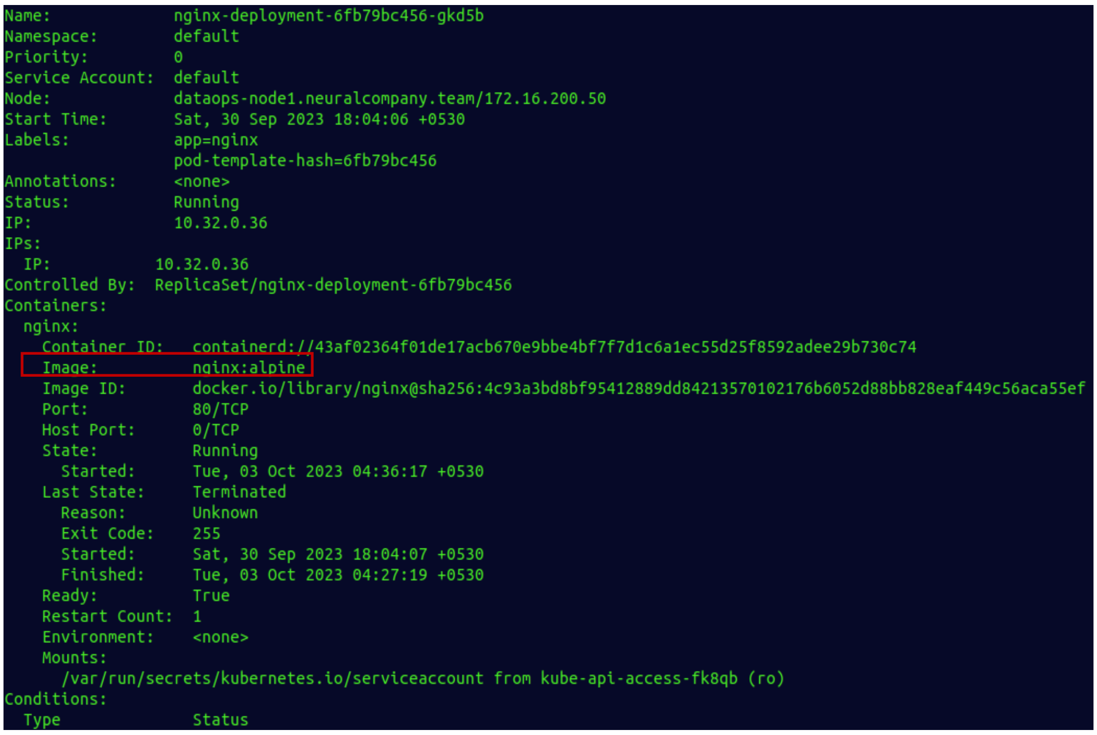
Console Output

```
Started by user xs309-shusai
Obtained Jenkinsfile from git https://github.com/subhamsaini1/Helm-nginx-share.git
Loading library nginx-deployment-lib@main
Attempting to resolve main from remote references...
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git ls-remote -h -- https://github.com/subhamsaini1/Helm-nginx-share # timeout=10
Found match: refs/heads/main revision 05192ff28eade2142de52d862247c67bb4403cec
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/nginxpodjsl@libs/2a65e85541239cc88e138528ba01db3bb270cb65590abdc03d63721df9c1eeac/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/subhamsaini1/Helm-nginx-share # timeout=10
Fetching without tags
Fetching upstream changes from https://github.com/subhamsaini1/Helm-nginx-share
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --no-tags --force --progress -- https://github.com/subhamsaini1/Helm-nginx-share +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking out Revision 05192ff28eade2142de52d862247c67bb4403cec (main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 05192ff28eade2142de52d862247c67bb4403cec # timeout=10
Commit message: "Add files via upload"
 > git rev-list --no-walk 05192ff28eade2142de52d862247c67bb4403cec # timeout=10
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/nginxpodjsl
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/nginxpodjsl/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/subhamsaini1/Helm-nginx-share.git # timeout=10
Fetching upstream changes from https://github.com/subhamsaini1/Helm-nginx-share.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/subhamsaini1/Helm-nginx-share.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 05192ff28eade2142de52d862247c67bb4403cec (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 05192ff28eade2142de52d862247c67bb4403cec # timeout=10
Commit message: "Add files via upload"
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Connect)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ export KUBECONFIG=/home/xs309-shusai/Downloads/dataops-cluster-k8s
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy Flink)
[Pipeline] script
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Deploy Flink Application)
[Pipeline] sh
+ helm upgrade --install my-nginx-chart ./resources/ -f ./resources/values.yaml --set image=nginx:alpine --set portType=NodePort
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/xs309-shusai/Downloads/dataops-cluster-k8s
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/xs309-shusai/Downloads/dataops-cluster-k8s
Release "my-nginx-chart" does not exist. Installing it now.
NAME: my-nginx-chart
LAST DEPLOYED: Sat Sep 30 15:57:53 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



```
xs309-shusai@xs-host603-048:~$ k get po,deploy
NAME                                   READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-6fb79bc456-gkd5b   1/1     Running   0          9m49s
pod/nginx-deployment-6fb79bc456-r8pdq   1/1     Running   0          9m49s
pod/nginx-deployment-6fb79bc456-wz7p2   1/1     Running   0          9m49s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           9m49s
xs309-shusai@xs-host603-048:~$ helm ls
WARNING: Kubernetes configuration file is group-readable. This is insecure. Location: /home/xs309-shusai/.kube/config
WARNING: Kubernetes configuration file is world-readable. This is insecure. Location: /home/xs309-shusai/.kube/config
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
my-nginx-chart  default         1               2023-09-30 18:04:05.590804686 +0530 IST deployed        nginx-deployment-0.1.0
```



```
Name:             nginx-deployment-6fb79bc456-gkd5b
Namespace:        default
Priority:         0
Service Account:  default
Node:             dataops-node1.neuralcompany.team/172.16.200.50
Start Time:       Sat, 30 Sep 2023 18:04:06 +0530
Labels:           app=nginx
                  pod-template-hash=6fb79bc456
Annotations:      <none>
Status:           Running
IP:               10.32.0.36
IPs:
  IP:           10.32.0.36
Controlled By:  ReplicaSet/nginx-deployment-6fb79bc456
Containers:
  nginx:
    Container ID:   containerd://43af02364f01de17acb670e9bbe4bf7f7d1c6a1ec55d25f8592adee29b730c74
    Image:          nginx:alpine
    Image ID:       docker.io/library/nginx@sha256:4c93a3bd8bf95412889dd84213570102176b6052d88bb828eaf449c56aca55ef
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 03 Oct 2023 04:36:17 +0530
    Last State:     Terminated
      Reason:       Unknown
      Exit Code:    255
      Started:      Sat, 30 Sep 2023 18:04:07 +0530
      Finished:     Tue, 03 Oct 2023 04:27:19 +0530
    Ready:          True
    Restart Count:  1
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-fk8qb (ro)
Conditions:
  Type              Status
```

Summary

Created a streamlined process for deploying nginx applications to Kubernetes using Helm and Jenkins Shared Library successfully.

Key Takeaways

- Jenkins pipelines offer automation and consistency in deployment workflow.
- The shared library allows for easy customization and scaling in larger projects.

Future Enhancements

- Explore options for improving security and secrets management.
- Consider implementing Helm chart versioning for more controlled deployments.