**K8s deployment with helm and *Jenkins* shared Library.**

[Problem Statement 1](#_Toc1355662794)

[Objective 1](#_Toc1544056936)

[Prerequisites 2](#_Toc1190647733)

[Value Proposition and Benefit 3](#_Toc1557365573)

[Benefits 3](#_Toc1190488125)

[Use Cases 3](#_Toc915698958)

[Execution Details 3](#_Toc1647453459)

[1. Directory Structure 3](#_Toc127842261)

[2. Creating a Helm Chart for nginx 4](#_Toc1717449974)

[3. Jenkins Shared Library Setup 5](#_Toc1519981768)

[4. Jenkins Pipeline Configuration 6](#_Toc1281958669)

[5. Deploying Nginx to Kubernetes 7](#_Toc358283928)

[Summary 8](#_Toc764268560)

# **Problem Statement**

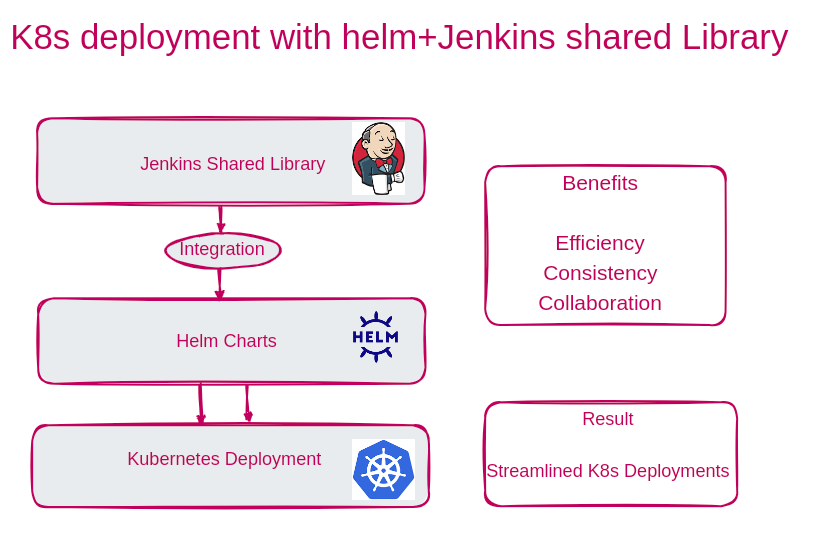
In current software deployment workflow, managing Kubernetes (K8s) applications is plagued by manual, error-prone processes that hinder our ability to scale and maintain efficiency. The absence of a standardized deployment framework creates challenges such as inconsistent configurations, time-consuming 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.

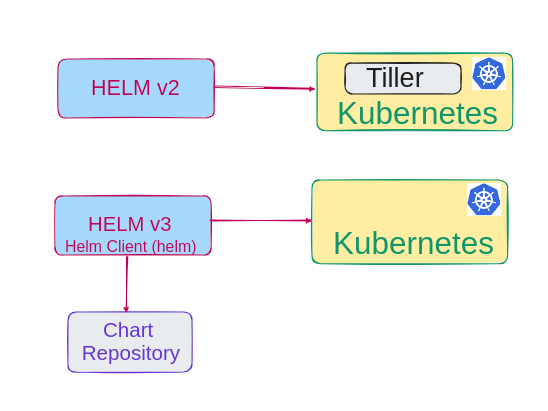
To address these challenges, there is a critical need for a technical document that outlines a solution for K8s deployment utilizing Helm and Jenkins shared libraries. This document is essential to automate and streamline our deployment processes, reducing manual errors and improving scalability. It 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**

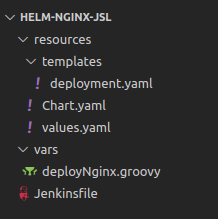
**└─ Chart.yaml**

**└─ values.yaml**

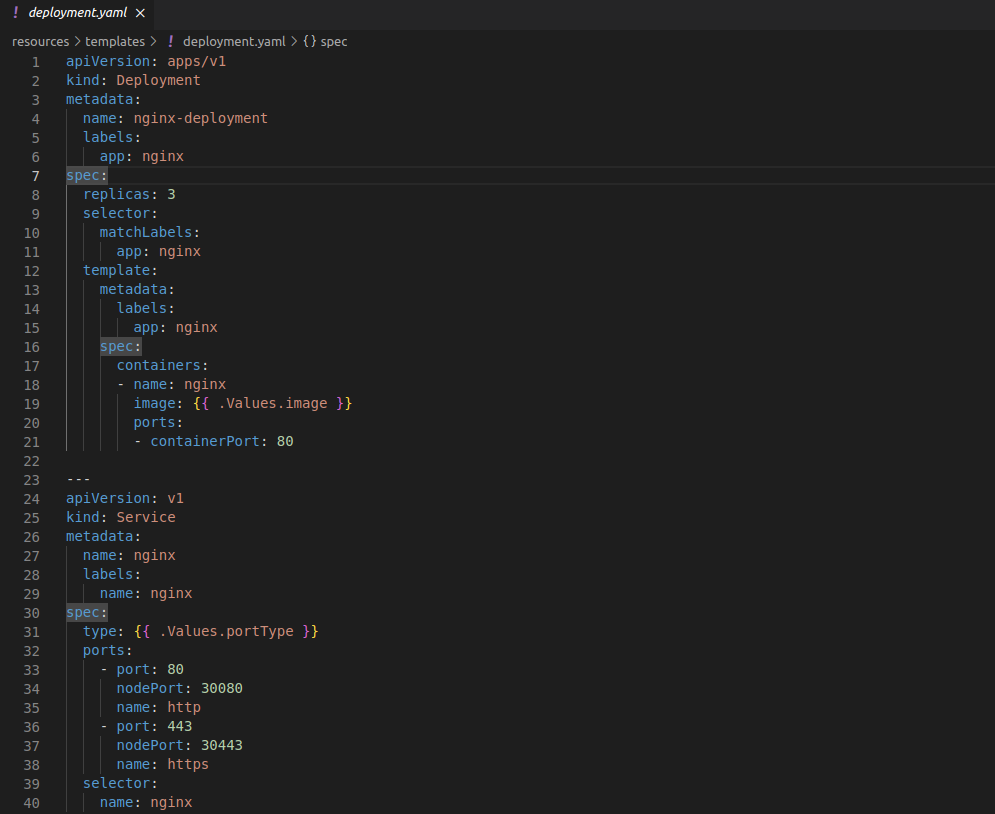
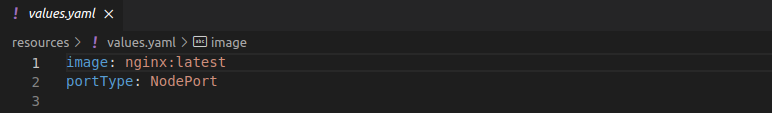
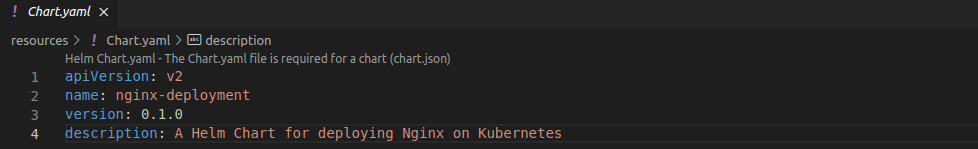
**└─ vars/**

**└─ deployNginx.groovy**

**└─ Jenkinsfile**



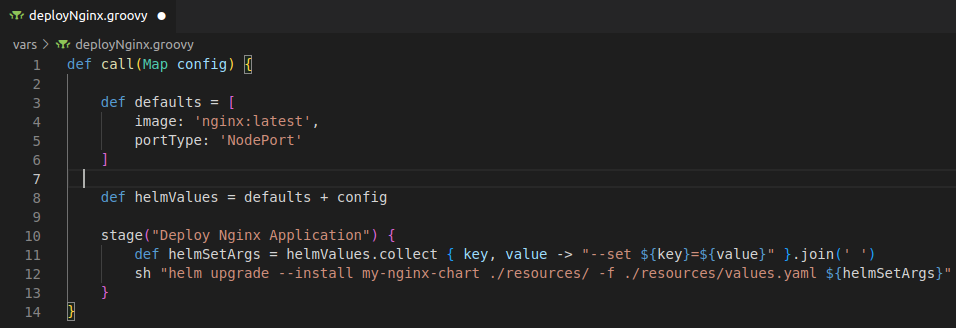
**2. Creating a Helm Chart for nginx**

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


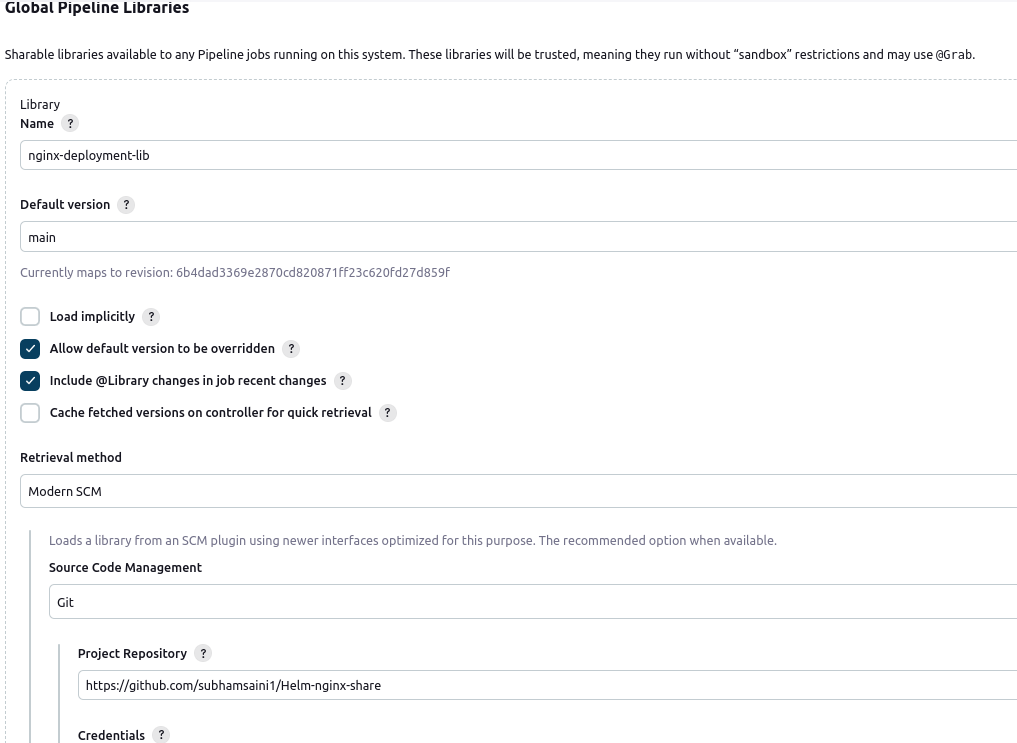
Push this code to GitHub repository form where it will be accessed by Jenkins.

**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.



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).



## **4. Jenkins Pipeline Configuration**

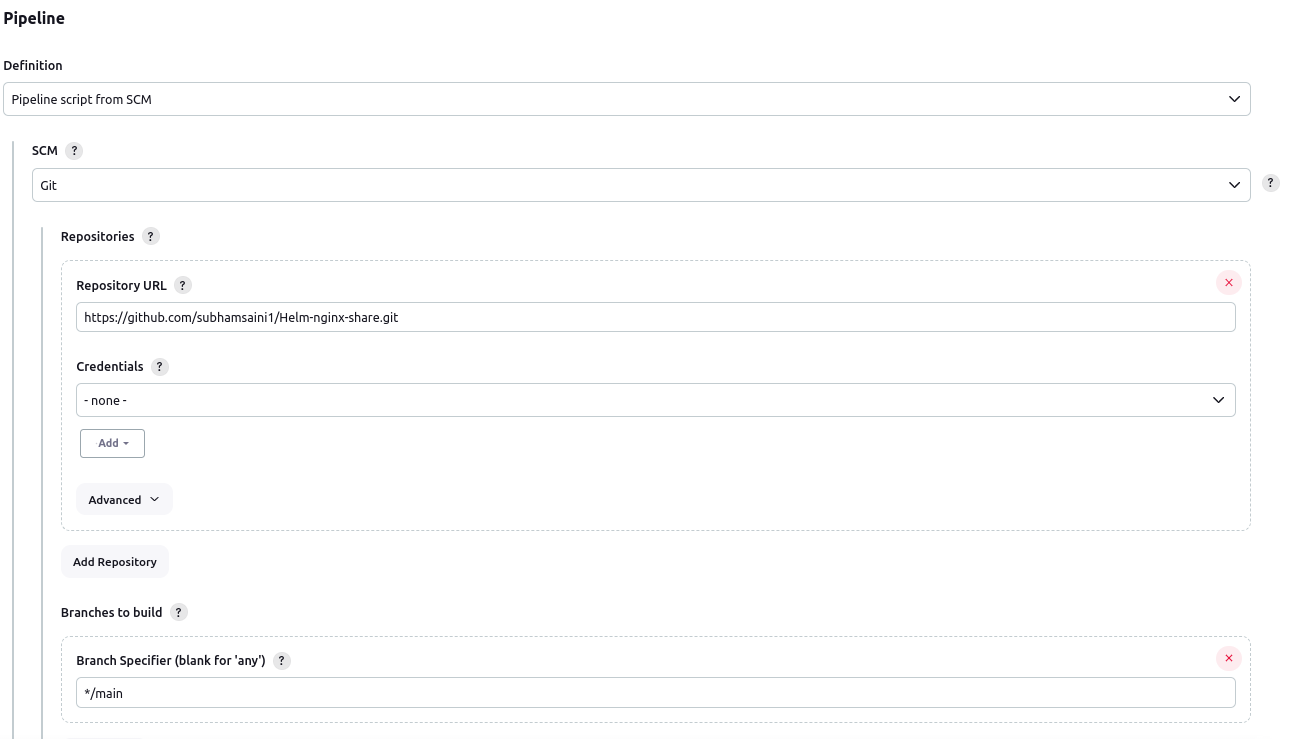
Create a Jenkins pipeline (Jenkinsfile) for Nginx deployment. Use the @Library annotation to reference shared library.  
Define deployment configuration in the deploymentConfig map within the pipeline.

Set up pipeline stages, including setting KUBECONFIG and calling the deployNginx function.

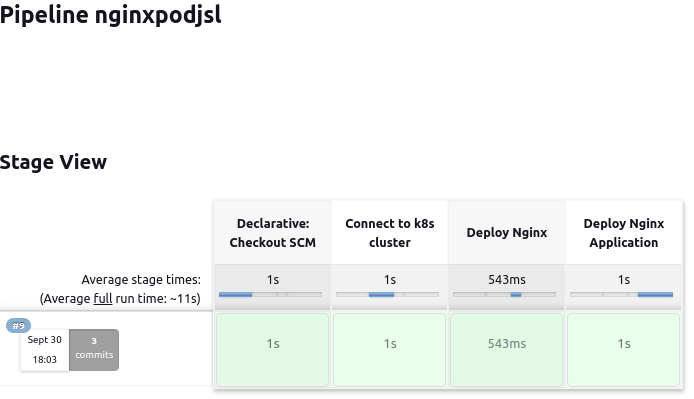
## **5. Deploying Nginx to Kubernetes**

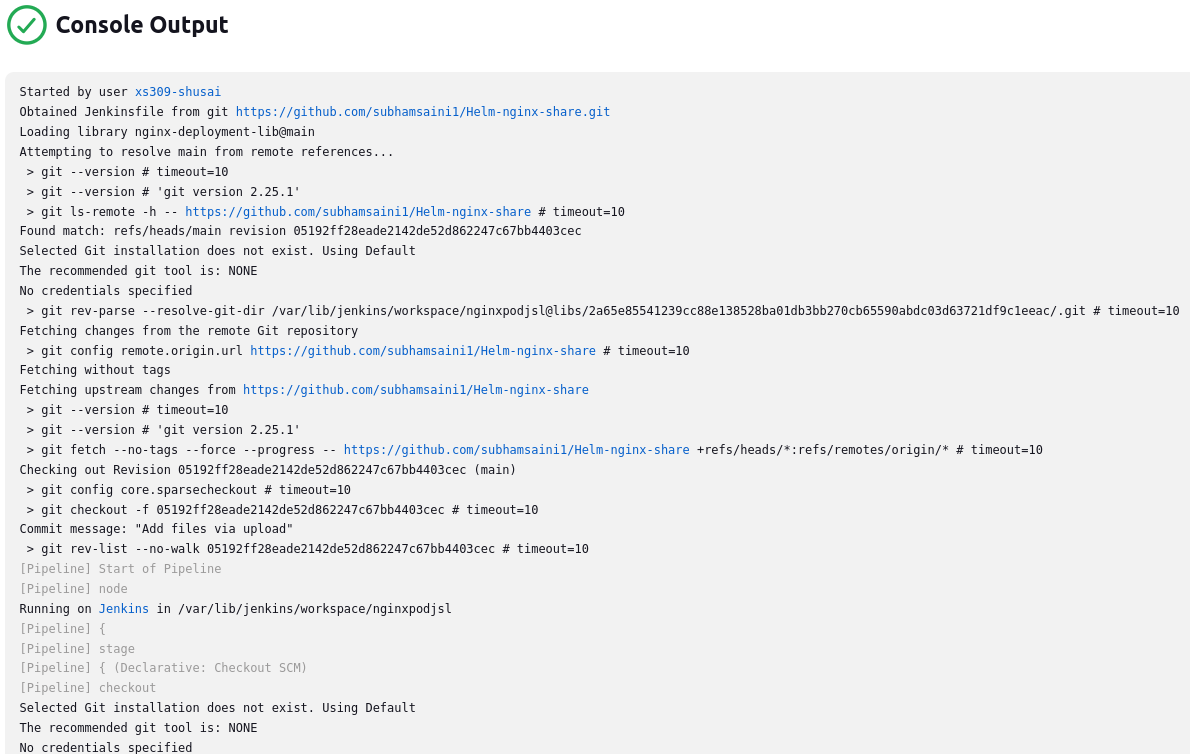
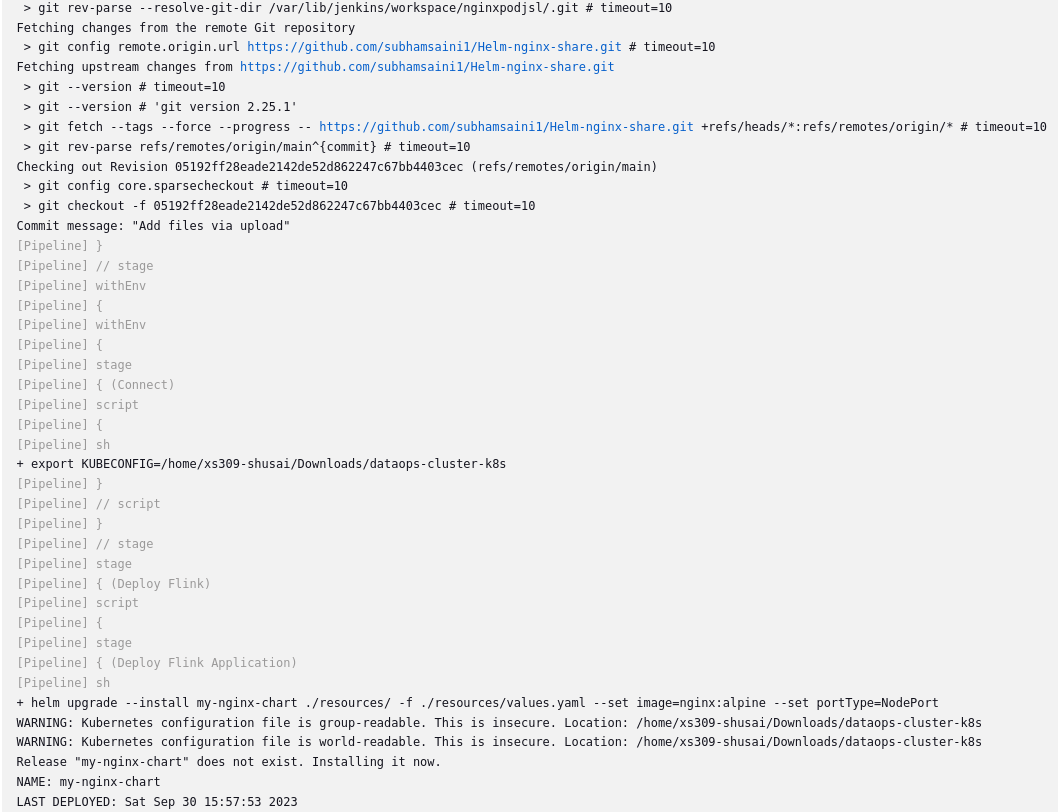
Trigger the Jenkins pipeline manually or through automation.

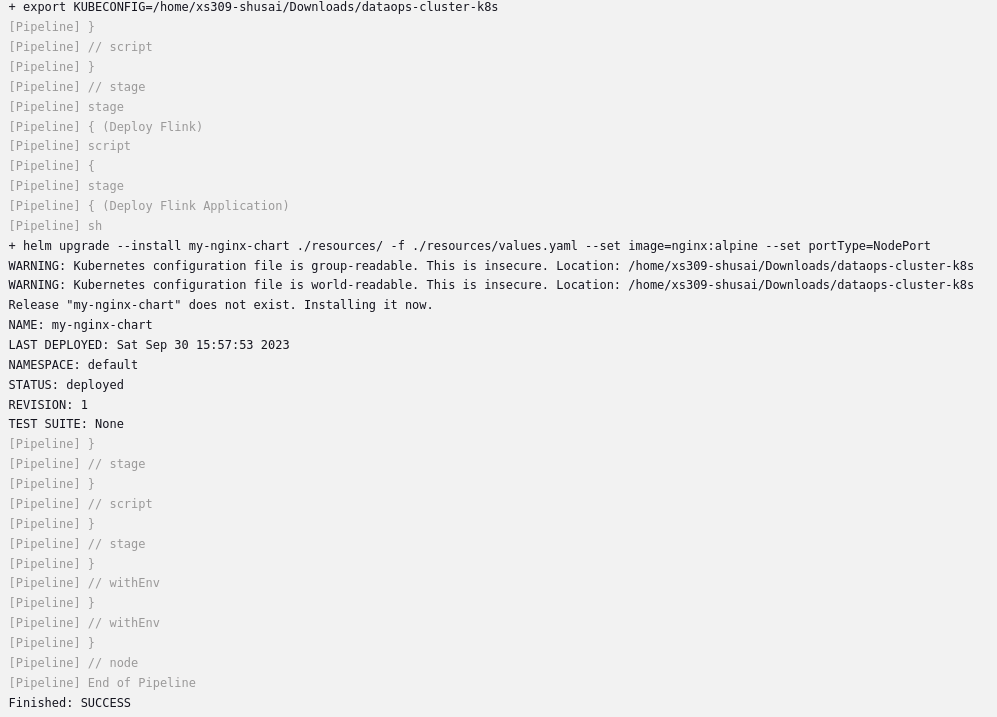
Go to [Dashboard](http://localhost:8080/) ---> [nginxpodjsl](http://localhost:8080/job/nginxpodjsl/) --> Configuration from here pipeline with SCM(Git) is created.

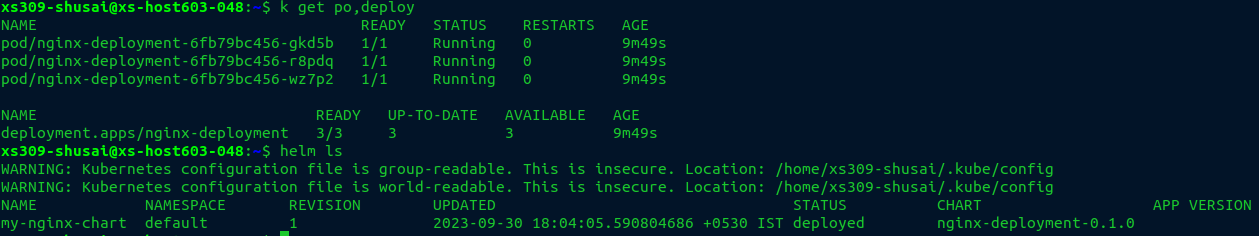


Click Save and Build. Jenkins will execute the pipeline, set the KUBECONFIG, and deploy Nginx to Kubernetes cluster based on the provided configuration.







# 

# **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.