To deploy a Go application to production using Git, Jenkins, and Docker

you need to follow a CI/CD pipeline that automates the process of building, testing, and deploying your application. Here’s a comprehensive step-by-step guide:

# Prerequisites

Install Git: Ensure Git is installed on your development and Jenkins servers.

Install Jenkins: Install Jenkins on a server and ensure it has necessary plugins (Git, Docker, and possibly others like Go and Pipeline plugins).

Install Docker: Install Docker on your Jenkins server and on the server where the application will be deployed.

Set Up a Git Repository: Have a Git repository for your Go application. You can use GitHub, GitLab, or any other Git service.

Create a Jenkins Pipeline: Use Jenkins Pipeline to define your CI/CD process.

Docker Hub Account: (Optional) To push Docker images to Docker Hub.

Step-by-Step Process

Step 1: Write Your Go Application

Create your Go application and commit it to your Git repository.

main.go:

go

Copy code

package main

import (

"fmt"

"net/http"

)

func helloWorld(w http.ResponseWriter, r \*http.Request) {

fmt.Fprintf(w, "Hello, World!")

}

func main() {

http.HandleFunc("/", helloWorld)

http.ListenAndServe(":8080", nil)

}

go.mod:

go

Copy code

module my-go-app

go 1.18

Step 2: Create a Dockerfile

Create a Dockerfile in the root of your repository.

Dockerfile:

dockerfile

Copy code

# Use an official Golang runtime as a parent image

FROM golang:1.18-alpine

# Set the Current Working Directory inside the container

WORKDIR /app

# Copy go mod and sum files

COPY go.mod go.sum ./

# Download all dependencies. Dependencies will be cached if the go.mod and go.sum files are not changed

RUN go mod download

# Copy the source code into the container

COPY . .

# Build the Go app

RUN go build -o main .

# Expose port 8080 to the outside world

EXPOSE 8080

# Command to run the executable

CMD ["./main"]

Step 3: Set Up Jenkins

Create a New Pipeline Job:

Go to Jenkins Dashboard -> New Item -> Pipeline.

Enter the job name and select "Pipeline".

Configure the Pipeline:

In the pipeline configuration, set up your SCM to point to your Git repository.

Define your pipeline script. You can use either a Jenkinsfile in your repository or write the script directly in Jenkins.

Jenkinsfile (to be placed in your Git repository):

groovy

Copy code

pipeline {

agent any

environment {

DOCKERHUB\_CREDENTIALS = credentials('dockerhub-credentials-id')

}

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-username/your-repo.git'

}

}

stage('Build') {

steps {

script {

docker.build('my-go-app')

}

}

}

stage('Test') {

steps {

// Add Go test commands here if you have any

echo 'Running tests...'

}

}

stage('Push to Docker Hub') {

steps {

script {

docker.withRegistry('https://index.docker.io/v1/', DOCKERHUB\_CREDENTIALS) {

docker.image('my-go-app').push('latest')

}

}

}

}

stage('Deploy') {

steps {

script {

// Assuming you have ssh access to your production server and Docker installed there

sh '''

ssh user@production-server 'docker pull your-username/my-go-app:latest && docker stop my-go-app || true && docker rm my-go-app || true && docker run -d -p 8080:8080 --name my-go-app your-username/my-go-app:latest'

'''

}

}

}

}

post {

always {

cleanWs()

}

}

}

Step 4: Run the Pipeline

Trigger the Pipeline:

You can trigger the pipeline manually or set up a webhook in your Git repository to trigger it automatically on each commit.

Monitor the Pipeline:

Monitor the pipeline's progress in Jenkins. Ensure each stage (Checkout, Build, Test, Push, Deploy) completes successfully.

Step 5: Access Your Deployed Application

Once the deployment is complete, access your Go application at http://your-production-server:8080.

Additional Tips

Testing: Incorporate Go tests in your pipeline to ensure code quality.

Docker Hub Credentials: Store your Docker Hub credentials securely in Jenkins credentials.

SSH Keys: Ensure your Jenkins server has SSH access to your production server for deployment.

Error Handling: Implement error handling and notifications in your Jenkinsfile to alert you in case of failures.

By following these steps, you can automate the deployment of your Go application to production using Git, Jenkins, and Docker.