Step 1: Create and Host a GoLang Program

A. Write the GoLang Program

Create a simple GoLang program that serves the current date and time

```
package main

import (
	"fmt"
	"net/http"
	"time"
)

func handler(w http.ResponseWriter, r *http.Request) {
	currentTime := time.Now().Format(time.RFC3339)
	fmt.Fprintf(w, "Current date and time: %s", currentTime)
}

func main() {
	http.HandleFunc("/", handler)
	fmt.Println("Server is starting...")
	http.ListenAndServe(":8080", nil)
}
```

B. Create a Dockerfile

Create a Dockerfile to build the GoLang application into a Docker image.

Dockerfile:

```
# Use the official Go image to create a build artifact.
# This image has the Go compiler and standard library.
FROM golang:1.20 AS builder

# Set the Current Working Directory inside the container
WORKDIR /app

# Copy the Go source code into the container
COPY . .

# Build the Go app
RUN go build -o main .

# Use a minimal image to run the Go app
FROM debian:bullseye-slim

# Copy the binary from the builder stage
COPY --from=builder /app/main /main
```

```
# Expose port 8080 to the outside world
EXPOSE 8080

# Command to run the executable
CMD ["/main"]
```

C. Build and Push Docker Image

Build the Docker image:

```
docker build -t yourdockerhubusername/date-time-app .
```

Step 2: Deploy the Container to Kubernetes

A. Create Kubernetes Deployment and Service Configuration

deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: date-time-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: date-time-app
  template:
    metadata:
      labels:
        app: date-time-app
    spec:
      containers:
      - name: date-time-app
        image: yourdockerhubusername/date-time-app
        ports:
        - containerPort: 8080
```

service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: date-time-app-service
spec:
  selector:
    app: date-time-app
```

```
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: LoadBalancer
```

B. Deploy to Kubernetes
Apply the Deployment and Service YAML files:

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

Step 3: Expose the Application to the Internet
Verify the Service

Check the status of your service to get the external IP:

```
kubectl get services
```

The EXTERNAL-IP field will show the public IP address where your application is accessible.

Access the Application

Open a web browser and navigate to the external IP address to see the current date and time served by your GoLang application.