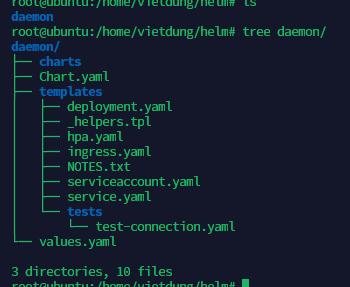


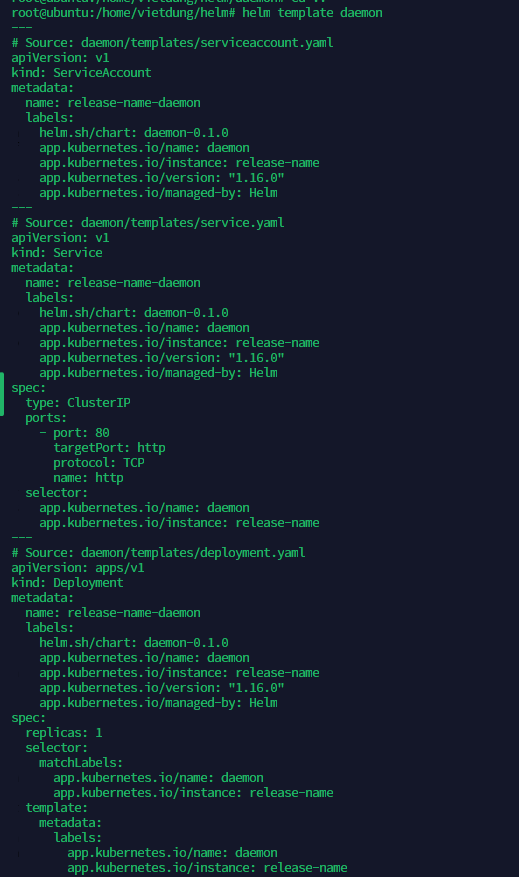
* Tạo 1 chart

helm create daemon



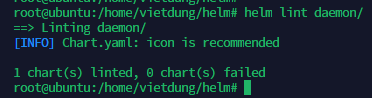
* Check template .yml mà helm chart đó tạo ra

helm template daemon



* Check lỗi của helm chart

helm lint daemon



0 có lỗi

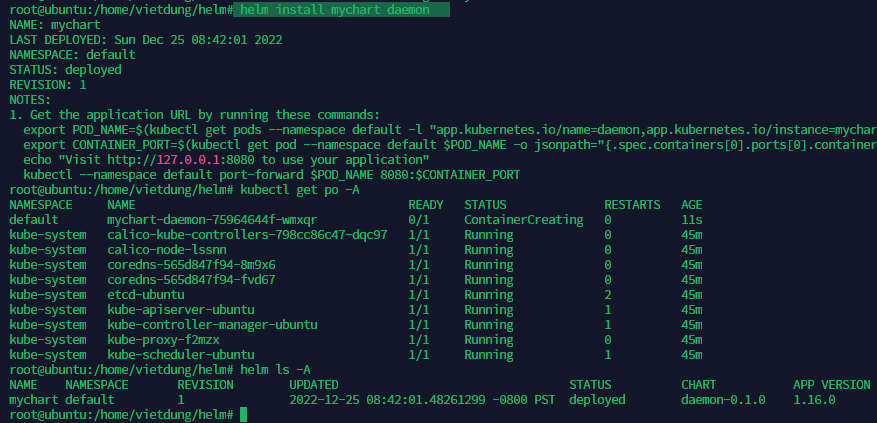
helm install daemon --debug --dry-run daemon

**deploy**

helm install mychart daemon

daemon : chart vừa tạo

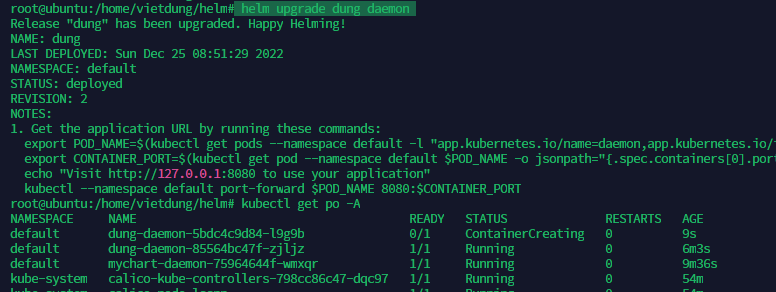
mychart : đặt tên, phiên bản cho chart này



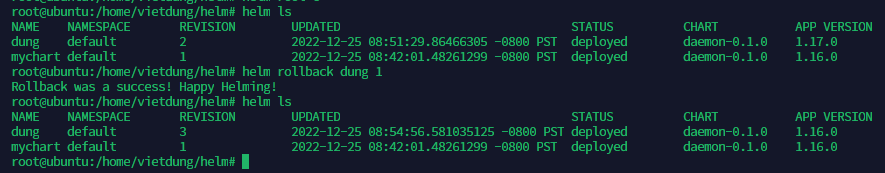
Oke đã có 1 helm **mychart** trong list chart

**Upgrade**

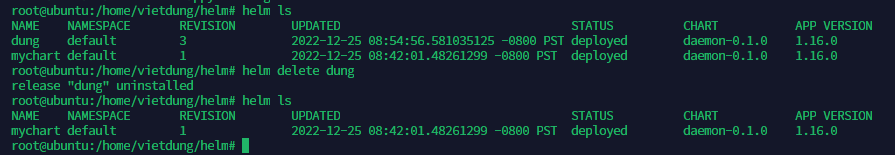
Tăng cái version trong file chart lên r upgrade



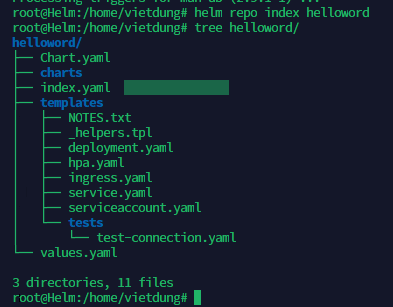
**Hạ cấp - Rollback**



Xóa



# Add repo helm

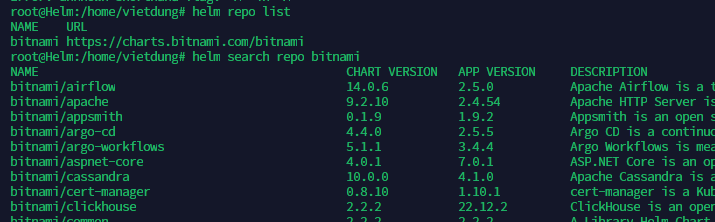


* Show repo

helm repo list

* Show charts trong repo

helm search repo bitnami



Xóa repo

helm repo remove bitnami

helm repo add bitnami https://charts.bitnami.com/bitnami  
 helm search repo bitnami  
 helm install my-release bitnami/<chart>

helm create <tên chart>

helm create vietdung

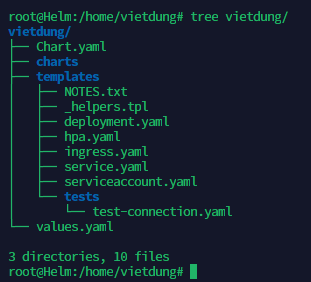
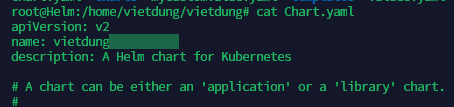


Chart.yaml : định danh tên chart, version của chart

Template: tất cả resource để deploy 1 app lên k8s

Version của chart là bắt buộc theo kiểu



Nếu đổi tên chart ở đây ta sẽ có chart mới

# File template



.values.image : gọi đến giá trị của image trong values file

apiVersion: apps/v1

kind: Deployment

metadata:

  name: {{ .Release.Name }}

  labels: {{ include "common.labels" . | indent 6 }}  # gtri key-value của labels lùi 6 dòng

spec:

  replicas: 3

  selector:

    matchLabels: {{ include "common.labels" . | indent 8 }}

  template:

    metadata:

      labels: {{ include "common.labels" . | indent 8 }}

    spec:

      containers:

      - name: {{ .Release.Name }}

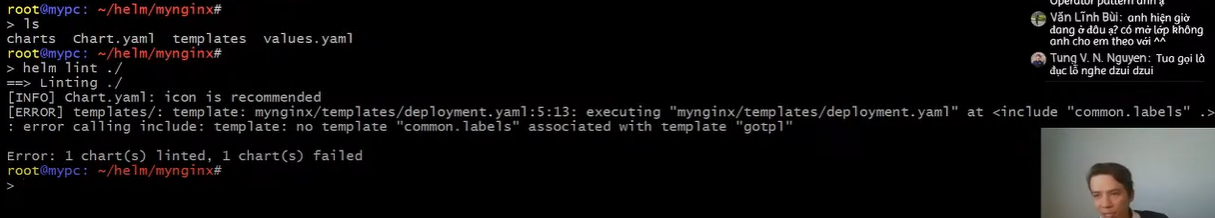
        image: {{ .Values.image }}

        ports:

        - containerPort: {{ .Values.port }}

include "common.labels : variable global scope

* check



Lỗi file deployment.yaml ở **at <include "common.labels" .** không có cái nào để gọi

* ta cần định nghĩa 1 file .**tpl** ngang hàng vs file deployment.yaml

\_all.tpl

# varible global

{{/\* common labels \*/}}        # đây là command

{{- define "common.labels" }}  # đây là define cho common.labels, kthuc define là end

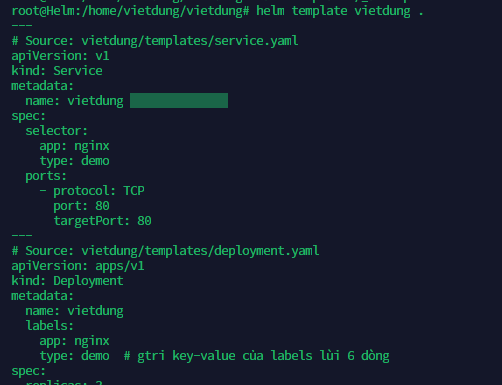
app: nginx

type: demo

identity\_key: {{ .Values.identity\_key }}

{{- end }}

Release.Name : tên biến truyền vào khi chạy helm



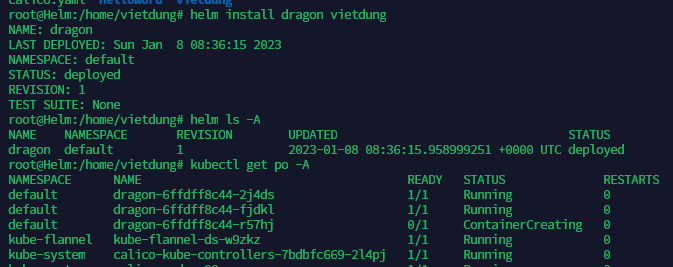
Ví dụ:

helm template Release.Name <chartname>

Chạy: helm template vietdung <chartname>

* Release.Name = vietdung

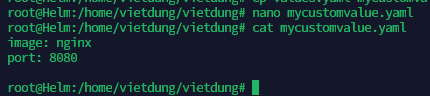
Oke install , đặt tên service là dragon



Dragon = chart = Release.Name = deployment, service name

Nếu ta muốn đổi port của service dùng file values khác

* Tạo 1 file value mới

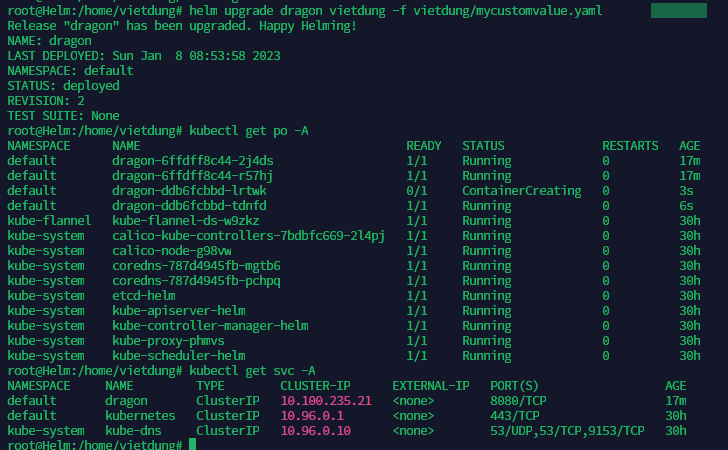


Rồi upgrade lại helm

helm upgrade dragon vietdung -f vietdung/mycustomvalue.yaml

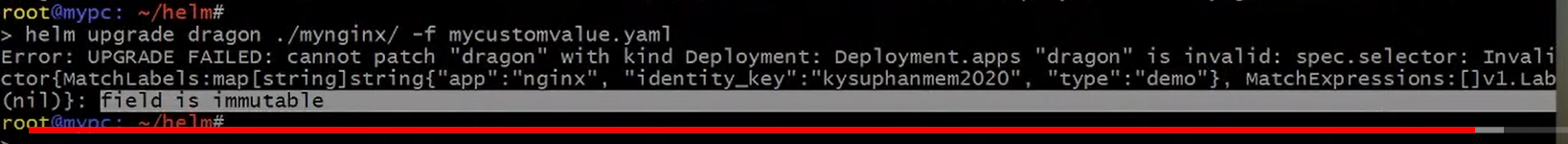
helm upgrade <chart\_name> <chart\_file> -f <values\_file>

mặc định values file là values.yaml



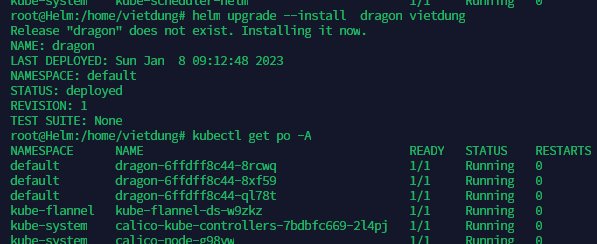
Lỗi khi upgrade

Lỗi này do khi upgade bị conflic với label của bản cũ



helm upgrade --install dragon vietdung

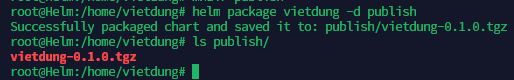
2 lựa chọn là install và upgrade



## Helm package: đóng gói helm

helm package <chart\_file> -d <folder\_save\_chart>

helm package vietdung -d publish

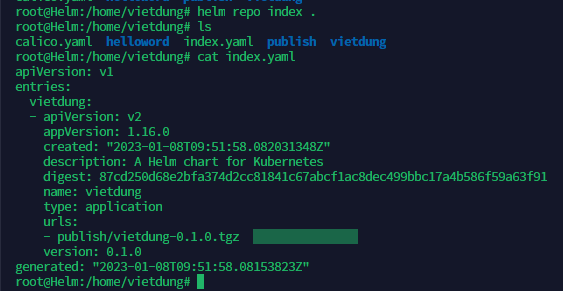


Nhưng muốn publish ra internet ta sẽ cần có 1 file index (index.html)

helm repo index .

helm n sẽ scan thư mục hiện tại hoặc chỉ định, nó sẽ tìm file có đuôi là .tgz

file .tgz là file đã được đóng gói trong lệnh <helm package …>

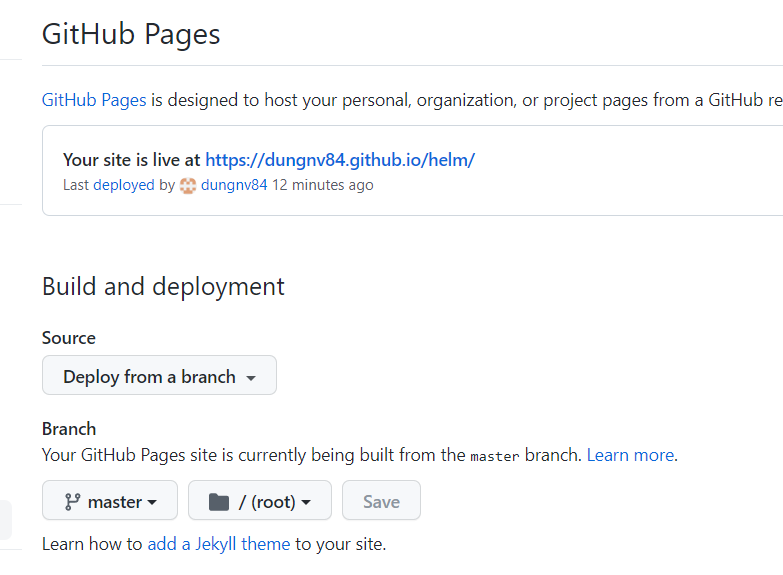


## Pushlish chart into github

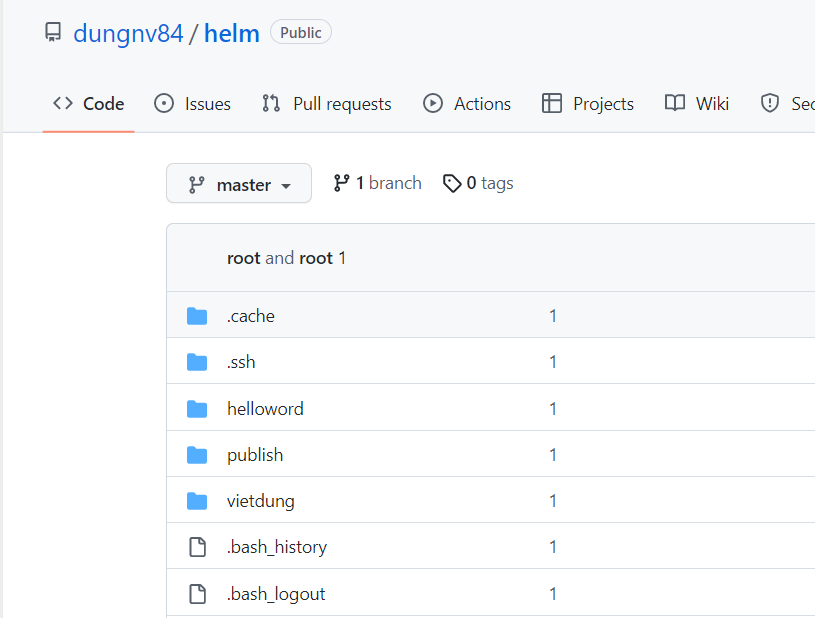
Helm repo index <folder publish chart>

Tạo 1 repo public trên github

Vào repo -> setting -> github page



Push package

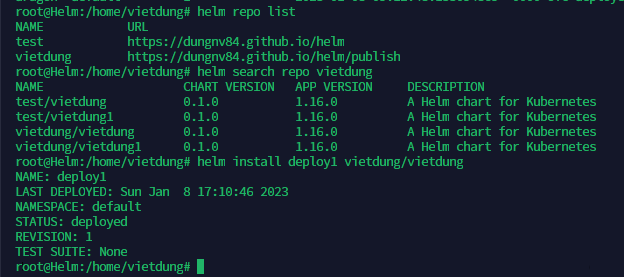
 file index.yaml cùng cấp

* Pull repo helm về

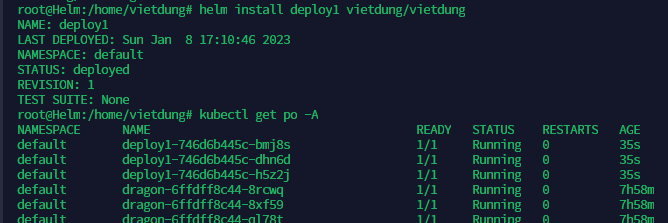
helm repo add test https://dungnv84.github.io/helm

helm repo add vietdung <https://dungnv84.github.io/helm/publish>

miễn là đường dẫn phải chứa file index.yaml



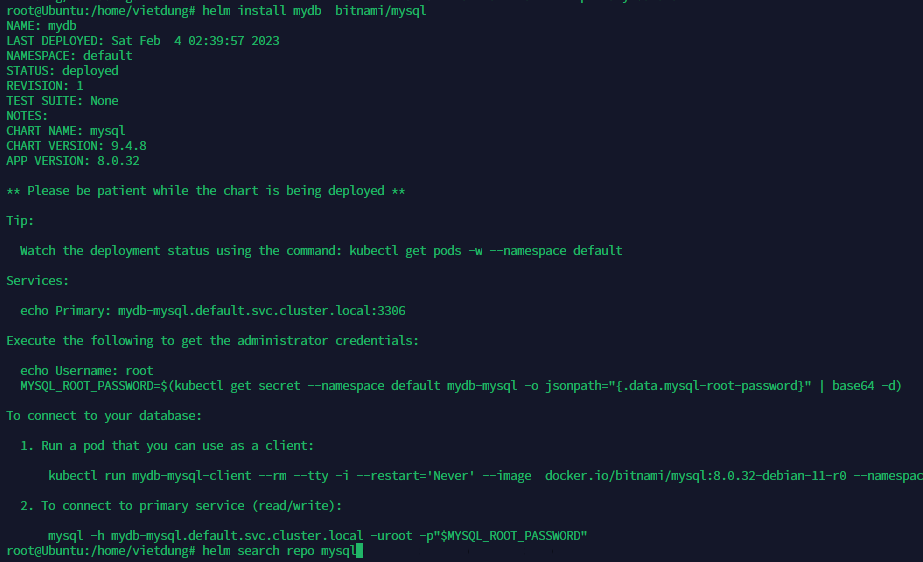
Deploy1 là tên app



# Helm install mysql

helm search repo mysql

helm install mydb bitnami/mysql



Các command gợi ý dòng 1 2

echo Username: root

MYSQL\_ROOT\_PASSWORD=$(kubectl get secret --namespace default mydb-mysql -o jsonpath="{.data.mysql-root-password}" | base64 -d)

1. Run a pod that you can use as a client:

kubectl run mydb-mysql-client --rm --tty -i --restart='Never' --image docker.io/bitnami/mysql:8.0.32-debian-11-r0 --namespace default --env MYSQL\_ROOT\_PASSWORD=$MYSQL\_ROOT\_PASSWORD --command -- bash

2. To connect to primary service (read/write):

mysql -h mydb-mysql.default.svc.cluster.local –u root –p "$MYSQL\_ROOT\_PASSWORD" DB\_NAME