Kubernetes’in paket managerı helm dir.

Helm is the package 
manager for Kubernetes 
HELM 

Helm de paket dediğimiz bizim manifest dosyalarımız, yaml file larımız.

Biz oluşturduğumuz dosyaları paketleyip her yerde kullanabiliyoruz.

Yaml file ların paketlenmiş haline chart, bulunduğu yere reposityory deniyor.

Release ise chartın instance yani çalışan hali. cluster a indirilip çalıştırılan halidir.

Helm yükleme ve vesion görme komutları:

Helm in dokümantasyonunda var nereye nasıl yükleneceği.

helm - - help ile inceleyebiliriz.

bütün chartların ortak bulunduğu yere ArtifactHUB deniyor. helm hub ile burada arama yapabiliyoruz.

<https://artifacthub.io/packages/search>

helm search repo

komutu hub daki bütün chartları listeliyor:

https : //artifacthub.io/packages/helm/geek- cookb... 
Lilly configurable Zwave to IOTT Gateway and Co... 
https : //artifacthub io/packages/helm/t ruecharts... 
ully configurable Zwave to mTT gateway and Co... 
https : //artifacthub. io/packages/helm/k8sonlab/z... 
elmchart for zwave-js-ui 
• helm search hub I wc -l 
1085e 
1 

10850 satır varmış.

helm search hub wordpress

bu şekilde de sadece wordpress chartlarını görebiliriz.

bir de helm search repo var bu da lokaldeki repoları gösterir.

@ helm repo Is 
Error: no repositories to show 

helm repo ls dediğimizde şu an lokalde bir repomuz yok.

önce bir chart reposunu lokale eklememiz gerekiyor.

helm repo add bitnami <https://charts.bitnami.com/bitnami>

komutu ile (bitnami yerine başka isim de verebilriz.) bitnami reposunu lokale ekliyoruz.

tekrar helm repo ls dediğimde:

bitnami reposunu görüyoruz, URL ile bağlıyor. chartlar inmiş olmuyor, chartların listesi geliyor.

helm search repo bitnami

komutuyla da lokalde bitnami ismini verdiğimiz repoyu araştır demiş oluyoruz.

saedece helm search repo girersek birden fazla repo olduğunda hepsini inceler.

helm pull bitnami/mysql

komutuyla da docker da imagei lokale aldığımız gibi kuberneteste chartı lokale indiroyoruz. yani bu kez istediğimiz chartı indirmiş oluyoruz.

neyi çekiyoruz? lokale eklediğimiz bitnami reposundan mysql chartını indir. ziplenmiş halde gelir.

tar -xvf mysql-\*.tgz

komutuyla ziplenmiş dosyayı dışarı çıkartabiliyoruz.

dosya yapılarını bir sonraki derste konuşacaz. içinde bir çok manifest dosyaları var. biz kendimiz de bir uygulama koymak için bu şekilde chart haline getirecektik.

helm install mysql-release bitnami/mysql

komutu da bitnami reposundaki mysql chartını install eder. şu ana kadar install etmemiştik, pull etmiştik. install ettiğimizde bu bir release olacak, burada release e mysql-release ismini verdik.

install ettik:

çalışmadı çünkü her chartın çalışması için gerekli configürasyonlar olur. örneğin burada provider dan volume gerekiyor, storage class gerekiyor.

bir repoyu çalıştırmadan önce helm repo update komutuyla lokaldeki repoları güncelleriz.

oluşan release e bakmak için helm ls:

release silmek için de helm uninstall:

şimdi wordpress çalıştıralım:

farklı değişkenleri var wordpressin veya diğer chartların. hangi değişkenleri olduğuna böyle parametrelerden de bakabiliriz, terminalden helm show values bitnami/mysql

komutuyla bakabiliriz.

bu değişkenleri kendimize göre ayarlayabilir ya da default çalıştırabiliriz.

helm show chart bitnami/mysql

komutu da chartın genel bilgilerni verir.

chartın içinde ne varsa görmek istiyorsak:

helm show all bitnami/mysql

bu komutun çıktısı yukarıdaki parametreleri terminalde uzun uzun gösterir.

Peki bir sürü değişken var dedik biz bunları nasıl değiştirecez? - - set flagiyle:

h 
elm install my-release \ 
- -set wordpressUsername=admin \ 
-set wordpressPassword=password \ 
-set mariadb. auth . rootPassword=secretpassword \ 
bitnami/wordpress 

burada username password değiştiriliyor.

Bizim chartlarda önceliğimiz kendi chartımızı repomuzu nasıl ouşturup kullanırız.

Helm de bir uygulamanını bütün kubernetes objelerini içine paketleyebiliyoruz. Release oluşturunca belki 50 tane k8s dosyası oluşuyor. Zipleyip repolarda saklıyoruz.

Helmdeki repoyu önce lokalimize ekliyoruz sonra ihtiyacımız olan chartı install ediyoruz. Bu şekilde release çalışmış oluyor.

Helm sayesinde uygulamamızın çalışması için gerekli yaml dosyalarını paketleyerek kullanmasını istediğimiz kişilere ulaştırmak çok kolay.

<https://helm.sh/docs/intro/install/>

Kendi chartımızı oluştururken bize örnek bir dizin getiriyor biz de kendimize göre düzenliyoruz.

helm create clarus-chart

Komutuyla chartımızı oluşturuyoruz.

v clarus-chart 
> charts 
> templates 
.helmignore 
! Chart.yaml 
! values.yaml 

Böyle bir dizin oluşturuyor.

.helmignore repoda bulunmasını istemediğimiz dosyaları belirtmek için, bu dosyaları pakete gömmez.

Values örnek bir dosya. Bize dinamik bir yapı sağlıyor. Uygulamayı pkaetleyip repoya koyduk, bir başkası da bunu çalıştırırken kendi değerleri ile çalıştırabilir. Terraform tfvars gibi.

Chart.yaml örnek bir dosya

# pipeline. Library charts do not de 
type: application 
# This is the chart version. This ve 
# to the chart and its templates, ir 
# Versions are expected to follow SE 
e. 1.0 
version : 
# This is the version number of the 
# incremented each time you make chi 
# follow Semantic Versioning. They 
# It is recommended to use it with 
appVersion : 
"1.16. e" 

Chartı versiyonlayabiliyoruz.

Templates içinde de yaml dosyaları var:

> tests 
helpers.tpl 
deployment.yaml 
h pa.yaml 
ingress.yaml 
NOTES.txt 
service.yaml 
serviceaccount.yaml 

Bunlar da örnek dosyalar.

Biz bu chart dizinine kendi dosyalarımızı koyacağız.

rm -rf clarus-chart/templates/\*

Komutuyla templates içini silelim. Templates içinde önce configmap dosyamızı oluşturalım. Örnek yaparken configmap kullanacağız çünkü içinde data var ve bu sayede yapılan değişiklikleri hemen görebilecez.

O README.md 
configmap.yaml X 
clarus-chart > templates > ! configmap.yaml 
1 
2 
3 
5 
6 
apiVersion: VI 
kind: Configmap 
metadata : 
name: clarus-chart-config 
data: 
myvalue : 
"Hello World" 

Chartımız hazır. Bunu install ederek configmap imizin oluşması gerekir:

helm install helm-demo clarus-chart

• helm install helm-demo clarus-chart 
NAME: helm-demo 
LAST DEPLOYED: Thu Apr 13 2023 
WIESPACE: default 
STATUS: deployed 
REVISION: 1 
TEST SUITE: None 

configmap oluştu

• ubuntu@kube-master : 
clarus-chart-config 
kube-root -ca . crt 
k get 
DATA 
1 
1 
cm 
AGE 
16s 
24h 

bu cm yi biz apply ile oluşturuyorduk normalde. şimdi chart ile oluşturduk.

helm ls dediğimizde:

helm-dem 
o ubu 
UAMESPACE 
default 
-master : "lesson$ 
helrn Is 
REVISICN 
11 
UPDATED 
2023-øa-13 
UTC 
STATUS 
deployed 
CHART 
clarus -chart-ø 
APP VERSI(R 
1.16.ø 

buradaki version chart.yaml dan geliyor.

• kubectl describe cm clarus-chart-config 
clarus-chart-config 
Name : 
default 
Namespace: 
Labels : 
app. kubernetes . io/managed -by—Helm 
Annotations : 
meta . helm. sh/release-name: helm-demo 
meta . helm. sh/release-namespace: default 
Data 
myvalue: 
Hello World 
BinaryData 
Events : 
<none> 

cm yi describe deidğimizde data yı da burada görebiliyoruz. şimdi chartı uninstall dediğimizde bütün objeler silinir.

• helm uninstall helm-demo 
release "helm-demo" 
uninstalled 
• k get cm 
DATA AGE 
kube-root-ca . crt 
1 
24h 

uninstall yerine delete komutu da çalışır.

values.yaml içerisine gidelim ve içini silelim.

course: DevOps

key-value sini yapıştıralım. configmap te bunu data olarak alalım:

apiVersion: v1

kind: ConfigMap

metadata:

name: clarus-chart-config

data:

myvalue: "clarus-chart configmap example"

course: {{ .Values.course }}

burada key-value yi elle yazmadık {{ .Values.course }} ile bu değeri çekebildik.

şimdi helm install --debug --dry-run mydryrun clarus-chart

komutuyla --dry-run dediğimizde objeyi oluşturmadan sanki oluşmuş gibi bize bilgileri getirir.

MANIFEST : 
# Source: clarus-chart/templates/configmap.yaml 
apiVersion: VI 
kind: ConfigMap 
metadata : 
name: clarus-chart-config 
data: 
myvalue: "clarus-chart configmap example" 
course: DevOps 

burada course keyinin value sini DevOps olarak gördük.

• kubectl create deploy mydeploy 
deployment. apps/mydeploy created (dry run) 
o ubunt*ube-masterw/lesson$ 
- -Image=nganx - 
-dry-run:client 

bu komutla bir objenin yaml dosyasını docs a gitmeden alabiliyoruz. dry run create ettikten sonra -o yaml ile yaml dosyaısın gördük.

deployment. apps/mydeploy created (dry run) 
• kubectl 
apiVersion: apps/vl 
kind; Deployment 
metadata ; 
creationTimestamp; 
labels . 
app: mydepIOY 
spec 
replicas: I 
selector: 
match Labels : 
app: mydeploy 
strategy: { } 
template: 
metadata : 
null 
creati onTimestanp ; 
labels ; 
app; oy 
spec: 
Inage: ngxnx 
name: ngxnx 
resources: {l 
status: {} 
o ubuntu€kube-master 
null 
kubectl 
create deploy mydeploy 
create deploy mydeploy 
- -Image-nglnx - 
--lmage-nglnx - 
-dry-run-client 
-dry-run=client 
-o yaml 
11 
-o yaml > mydeploy.ym 

yukarıdaki komut ile de bu yaml dosyası içeriğini my.deploy.yml a yazdırıyoruz.

status; {} 
• kubectl Create deploy mydeploy - -imaee=neinx - —dry-run—client -O yan1 > mydeploy_ynl 
@ kubeetl expose deploy mydepIOY - 
fr•an deploytnents_apps "mydeploy" not found 
• kubectl apply -f 
deployment. apps/mydeploy created 
• kubectl expose deploy mydeploy --port=8ø 
- dry - 
- -dry-run—client 
-o yanl myservice.yrnl 

mydeploy u apply ettikten sonra expose ile service objesini de oluşturabildik:

app; myaeploy 
name. mydeploy 
spec . 
replicas: I 
selector: 
match Labels: 
app: mydeploy 
strategy: 
template: 
metadata 
creationTimestamp: 
V" iabels•. 
• • app: , nydG 
containers; 
Image; nglnx 
name; nglnx 
resources: 
null 

dry-run ile oluşturmadan gördük ama şimdi de helm install ile release i oluşturalım:

helm install myvalue clarus-chart

clarus-chart tan myvalue isminde bir release oluştur.

helm ls diyelim:

• helm install myvalue clarus-chart 
NAME: myvalue 
LAST DEPLOYED: Thu 
WIESPACE: default 
STATUS: deployed 
REVISION: 1 
TEST SUITE: None 
WIESPACE 
myvalue default 
Apr 13 2023 
helm Is 
REVISION 
1 
UPDATED 
2023-04 
+øøøø 
UTC 
STATUS 
deployed 
CHART 
APP VERSION 
-13 
clarus -chart -0.1.01.16.0 

helm get manifest myvalue

komutu da myvalue release i içindeki manifest dosyalarını döndürür. bizim şu an sadece bir dosyamız var ancak ne kadar olursa o kadarını getirir:

• helm get manifest myvalue 
Source: clarus -chart/templates/configmap. yaml 
apiVersion: VI 
kind: ConfigMap 
etadata: 
m 
name: clarus-chart-config 
data: 
myvalue: "clarus-chart configmap example" 
course: DevOps 

helm delete myvalue ile release i silelim:

• helm delete myvalue 
release "myvalue" uninstalled 
o ubuntußkube-master:æ$ 

şimdi values içindeki değeri komut satırında da değiştrebiliyoruz onu dry-run ile görelim:

helm install --debug --dry-run setflag clarus-chart --set course=AWS

debug da burda debugging yapıyor, komutta hata olması durumunda.

setflag burada release in ismi. --set flagiyle course un valuesini AWS olarak değiştirdik.

terraformda da --var flagiyle variable, --varfile ile de variable dosyası editleyebiliyorduk.

apiVersion: v1

kind: ConfigMap

metadata:

name: {{ .Release.Name }}-config

data:

myvalue: "clarus-chart configmap example"

course: {{ .Values.course }}

configmap i bu şekilde güncelleyelim. .Release.Name ile yazdığımız kısımda komut satırında yazacağımız release ismini buraya yazacak:

helm install --debug --dry-run builtin-object clarus-chart

HOOKS : 
MANIFEST : 
# Source: clarus-chart/templates/configmap.yaml 
apiVersion: VI 
kind: ConfigMap 
metadata : 
name: builtin-object-config 
data: 
myvalue: "clarus-chart configmap example" 
course: DevOps 

bu sayede configmap te isim vermiyoruz da kullanan kişi release ne isim verdiyse configmap ve diğer objelerinde ismi bu şekilde değişmiş oluyor.

values dosyasını güncelleyelim:

course: DevOps

lesson:

topic: helm

helmste 60 ın üzerinde fonksiyon var. bunların bazılarını görecez:

<https://helm.sh/docs/chart_template_guide/builtin_objects/>

<https://helm.sh/docs/chart_template_guide/function_list/>

<https://pkg.go.dev/text/template>

<https://masterminds.github.io/sprig/>

configmap dosyamızı güncelliyoruz:

apiVersion: v1

kind: ConfigMap

metadata:

name: {{ .Release.Name }}-config

data:

myvalue: "clarus-chart configmap example"

course: {{ quote .Values.course }}

topic: {{ upper .Values.lesson.topic }}

time: {{ now | date "2006.01.02" | quote }}

quote, upper, now fonksiyonlarını kullanıyoruz. quote tırnak içinde getirir. date ile de tarihi bu formda getir diyoruz. yani now un değerini buraya gir.

helm install --debug --dry-run morevalues clarus-chart

komutuyla morevalues release ini görelim:

HOOKS : 
MANIFEST : 
# Source: clarus-chart/templates/configmap.yaml 
apiVersion: VI 
kind: ConfigMap 
metadata : 
name: morevalues-config 
data : 
myvalue: "clarus-chart configmap example" 
course: "DevOps " 
topic: HELM 
time: "2023.04.13" 

install edince bir de şöyle bir kısım geliyor ilk kısımda

NAME: morevalues 
LAST DEPLOYED: Thu Apr 13 2023 
WIESPACE: default 
STATUS: pending-install 
REVISION: 1 
TEST SUITE: None 
USER-SUPPLIED VALUES: 

bilgi olarak. bunlar template altında NOTES.txt içerisinde yazılan bilgilerle customize edilebilir. buraya başkaları release imizi çalıştırdığında gereken bilgileri vermek için yazılır. README dosyası gibi.

v templates 
! configmap.yaml 
NOTES.txt 

bu yazım ile yazılacak dosya ismi. bu dosya içinde built-in object ler yani dinamik değerler kullanılabilir.

Thank you for installing {{ .Chart.Name }}.

Your release is named {{ .Release.Name }}.

To learn more about the release, try:

$ helm status {{ .Release.Name }}

$ helm get all {{ .Release.Name }}

bunları NOTES.txt içine yazalım.

helm install notes-demo clarus-chart

release i notes-demo ismiyle çalıştırıyoruz:

• helm install notes-demo clarus-chart 
NAME: notes-demo 
LAST DEPLOYED: Thu Apr 13 2023 
WIESPACE: default 
STATUS: deployed 
REVISION: 1 
TEST SUITE: None 
NOTES: 
Thank you for installing clarus-chart. 
Your release is named notes-demo. 
To learn more about the release, try: 
$ helm status notes-demo 
$ helm get all notes-demo 

yazdığımız notlar release oluşturulurken bu şekilde oluşturan kişiye yazdırılacak. bu chartın nasıl kullanılacağını gösterir kullanıcıyla.

release i silelim.

chartlar uzak repolarda tutuluyor. repoda dosyalarımız paket halinde tutuluyor.

tree komutuna ihtiyacımız olabilir, yükleyelim.

Chartmuseum reposunu kullanacağız, go dilinde yazılan ve cloud storage ların da desteklendiği bir repo.

curl <https://raw.githubusercontent.com/helm/chartmuseum/main/scripts/get-chartmuseum> | bash

komutuyla chartmuseum u yüklüyoruz. bu komut da chartmuseum docs te var.

şimdi kendimize göre configure ediyoruz. bu komutlar da docs tan alınıyor:

chartmuseum --debug --port=8080 \

--storage="local" \

--storage-local-rootdir="./chartstorage"

burada şu porttan yayın yapsın diyoruz.

chartmuseum - -debug 
-storage="local" \ 
-storage-local-rootdir=" . /chartstorage" 
-port=8Ø8Ø \ 
DEBUG 
DEBUG 
INFO 
DEBUG 
Fetching chart list from storage 
{"repo": 
No change detected between cache and storage 
{"repo": 
Starting ChartMuseum 
{"host" : 
••Ø.Ø.Ø.Ø••, "port": 8080} 
Starting internal event listener 

yeni bir terminalde çalıştıralım çünkü termianli işgal edecek.

biz bunu hangi makinede kurduysak o makinede görülebilir. biz masterda çalıştırdık. görebilmek için önce instance ın 8080 portunu açalım.

Güvenli deéil 54.147.21.126:8080 
Course: AWS 13-TU... Google Takvim 13... O Your Repositories DeW3ps - Microsoft... (3) "devops" Searc... 
Welcome to ChartMuseum! 
HacklnScienc 
If you see this page, the ChartMuseum web server is successfully installed and 
working. 
For online documentation and support please refer to the GitHub project. 
Thank you for using ChartMuseum. 

daha sonra public ip ile 8080 portunda görüyoruz. gerçek hayatta çok kullanılmıyor, container ile de çalışabiliyor charmuseum. dışarıda daha çok github veya cloud providerlar kullanılıyor.

bu repodaki chartları kullanabilmemiz için bu repoyu locale çekebilmemiz lazım. url si olmadığı için public ip sini giriyoruz.

helm repo add mylocalrepo [http://<public-ip>:8080](about:blank)

komutuyla repoyu lokale çekiyoruz.

• helm repo Is 
WIE 
URL 
bitnami 
https : / / charts . bitnami.com/bitnami 
mylocalrepo 
http://54.147.21.126:8080/ 
o ubuntu@kube 

az önce chartmuseum da repo oluşturduk şimdi de lokale çektik. ancak bu repoda henüz bir chart yok.

helm search repo mylocalrepo

komutuyla gördük chart olmadığını.

• helm search repo mylocalrepo 
No results found 
o ubuntußkube-master:æ$ 

dersin bu anına kadar rpeo kullanmadık. lokalden çalıştık. şimdi chartımızı paketleyip repoya gönderelim:

helm package clarus-chart

• helm package clarus-chart 
Successfully packaged chart and saved it to: /hune/ubuntu/clarus-chart-ø.l.ø.tgz 

versionlayarak paketlendi. version Chart.yaml dan geldi.

şimdi paketimizi bir klasöre atalım (zorunlu değil)

mkdir my-charts

mv clarus-chart-0.1.0.tgz my-charts

repomuzda olmazsa olmaz bir dosya var index dosyası. bu chartları tutuyor. repomuzda index imizi oluşturalım:

helm repo index my-charts --url [http://<public-ip>:8080](about:blank)

helm repo index le oluşturuyoruz. my-charts paketimizin bulunduğu klasör ismi. gerisi de repomuzun url si.

v my-charts 
clarus-chart-0.1.0.tgz 
! index.yaml 

my-charts klasörü altında index.yaml oluştu.

/-charts > ! index.yaml 
1 
2 
3 
5 
6 
7 
8 
9 
12 
13 
15 
apiVersion: VI 
entries : 
clarus-chart : 
apiVersion: v2 
appVersion : 
1.16.ø 
created 
description: A Helm chart for Kubernetes 
digest: 67e1d3e1f986e6a83e695eeeae3c9Ø972639793eeeb3afØe91Øba69ddc98b39Ø 
name: clarus-chart 
type: application 
urls: 
http : / /54.147.21.126 : 8Ø80/c1arus-chart-Ø.1.0. tgz 
version: 0.1.0 
generated • 
. "2Ø23-øa-13T19: 54: 51.99435735Z" 

halen repoya göndermedik.

my-charts içine girelim.

curl --data-binary "@clarus-chart-0.1.0.tgz" [http://<public](about:blank) ip>:8080/api/charts

curl --data-binary ile dosya gönderebiliyoruz. curl kapsamlı bir komut.

• curl --data-binary "@c1arus-chart-Ø.1.Ø.tgz" htt 
{"saved":true}ubuntu@kube-master:æ/my-charts$ 

şu anda paketi repoya gönderidk.

C A Güvenli deéil 54.147.21.126:8080/api/charts 
AWS Management... Course: AWS 13-TU... Google Takvim 13... O Your Repositor 
["clarus -chart" : [{"name" : "clarus-chart" , "version" : "O. I.e" , "description" : "A Hell 
-hart-e.l.ø. tgz"], "created . 
" • "2023-e4-13T19 : 58 : 21.551784589Z 
" , "digest" : "67aId3e 

/api/charts a gittiğimizde gördük. ancak lokalde göremiyoruz:

{"saved":true}ubuntu@kube-master:æ/my-charts$ helm search repo mylocalrepo 
No results found 

bunun için repoyu update etmemiz gerkeiyor.

• helm repo update 
Hang tight while we grab the latest from your chart repositories... 
.. Successfully got an update from the "mylocalrepo" chart repository 
.. Successfully got an update from the "bitnami" chart repository 
Update Complete. •Happy Helming! 
• N/my-charts$ 
• ubuntu@kube-master . 
mylocalrepo/clarus-chart 
Kubernetes 
o ubuntu@kube-master . 
N/my-charts 
helm search repo mylocalrepo 
CHART VERSION APP VERSION 
ø.l.ø 
1.16.ø 
DESCRIPTION 
A Helm chart for 

artık repomuzda chartımızı lokalden de görebiliyoruz.

başkası bu uygulamamızı deploy etmek isterse repomuza url den ulaşabilir ve bu repo içerisindeki chbu chartı çekip install eder.

diyelim ki uygulamamızda değşiklik yaptık ve version ekleyecez. bunun için yeni bir pket oluşturuyoruz.

Chart.yaml da verisonu güncelledik ve paket oluşturma komutunu girdik:

• helm package clarus-chart 
Successfully packaged chart and saved it to: /hune/ubuntu/clarus-chart-ø.l.l.tgz 

helm repo index komutu bizim için yeni verisonu ekleyerek index i de güncelleyecek:

helm repo index my-charts --url [http://http://54.147.21.126/:8080](http://http//54.147.21.126/:8080)

digest • 
name: clarus-chart 
type: application 
urls: 
http : / /http:/54.147.21.126/ : 808Ø/c1arus -chart-e .1.1. tgz 
version: 0.1.1 
apiVersion: v2 
appVersion : 
1.16.ø 
created 
description: A Helm chart for Kubernetes 
digest: 67e1d3e1f986e6a83e695eeeae3c9Ø972639793eeeb3afØe91Øba69ddc98b39Ø 
name: clarus-chart 
type: application 
urls: 
http : / /http:/54.147.21.126/ : 808Ø/c1arus -chart-e .1. e . tgz 
version: 0.1.0 
generated 

komtuyla uzak repoya da yeni versionu gönderelim:

curl --data-binary "@clarus-chart-0.1.1.tgz" <http://54.147.21.126/:8080/api/charts>

lokalde görmek için de lokali update ediyoruz.

artifecthub dan inddireceğimiz bir plugin ile bu manuel işlemler kolaylaşıyor ve helm push komutuyla pkaeti gönderebiliriz.

büyük projede de s3 için böyle bir plugin kullanacaz.

<https://github.com/chartmuseum/helm-push>

helm plugin install <https://github.com/chartmuseum/helm-push>

komutuyla plugin i indirdik.

Chart.yaml da da versionu 0.2.0 yapalım.

helm cm-push clarus-chart mylocalrepo

komutuyla direk chart ı push edebiliriz:

• helm cm-push clarus-chart mylocalrepo 
Pushing clarus-chart-Ø.2.Ø.tgz to mylocalrepo... 
Done. 
O ubuntußkube-master 

biz manuel olarak paketliyorduk, indexi güncelliyorduk vs. cm push plugin bizim için bunları yaptı.

C A Güvenli deéil 54.147.21.126:8080/api/charts 
AWS Management... Course: AWS 13-TU... Google Takvim 13... 
{ "clarus -chart" : [{"name" : "clarus-chart" , "version" : "O. 2.e" , "descript 
chart-e.2.ø. tgz"], "created . 
" • "2023-e4-13T2e : 22.259414734z" , "diges 
chart", "version" : "a.I.O", "description": "A Helm chart for Kubernetes 
" , " digest 
" : "67a1d3e1f986e6a83e695eeaae3c9a 
e4-13T19 : 58 : 21.551784589Z 

browserda version 0.2.0 ı görüyoruz.

versionu değiştirmeseydik aynı versiondan chart var uyarısı alırdık. ancak bunu da --force ile override yapabiliriz.

loaklimizi güncellediğimizde yeni versionu görebiliyoruz:

• helm 
mylocalrepo/clarus-chart 
search repo mylocalrepo 
CHART VERSION 
ø.2.ø 
APP VERSION 
1.16.ø 
DESCRIPTION 
A Helm chart for Kubernetes 

cm-push pluginiyle komut satırında da version güncelleme yapabiliyoruz:

helm cm-push clarus-chart mylocalrepo --version="1.2.3"

• helm 
mylocalrepo/clarus-chart 
search repo mylocalrepo 
CHART VERSION 
1.2.3 
APP VERSION 
1.16.ø 
DESCRIPTION 
A Helm chart for Kubernetes 

şimdi configmap e count=first key ini ekleyelim:

apiVersion: v1

kind: ConfigMap

metadata:

name: {{ .Release.Name }}-config

data:

myvalue: "clarus-chart configmap example"

course: {{ .Values.course }}

topic: {{ .Values.lesson.topic }}

time: {{ now | date "2006.01.02" | quote }}

count: "first"

helm cm-push clarus-chart mylocalrepo --version="1.2.4"

yeni versionla cm-push edelim.

bu chartı çalıştırmak için lokal repomuzu güncellememiz gerkiyor.

helm repo update 
helm search repo mylocalrepo 
helm install from-local-repo •mylocalrepo/clarus-chart 

mylocalrepo/ prefixini ekleyerek de lokaldeki chartı çalıştırmasını istiyoruz. update ettiğimiz için pull etmemize gerek yok.

• kubectl 
DATA 
frun-local-repo-config 5 
kube-root -ca . crt 
1 
• kubectl 
DATA 
from-local-repo-config 5 
get cm 
AGE 
2m13s 
26h 
get cm frun-local-repo-config 
AGE 
2m28s 
-o wide 

cm yi gördük.

configmap teki first yerine second yazalım.

1.2.5 versionu ile push edelim chartı:

helm cm-push clarus-chart mylocalrepo --version="1.2.5"

helm repo update diyoruz.

şimdi configmap içeriğindeki data first mü gelir second mu? repoyu update etmek chartı update etmek demek değildir. uygulama halen eski versionda çalışıyor.

helm upgrade komutuyla chartı güncellersek yeni versiona geçer:

helm upgrade from-local-repo mylocalrepo/clarus-chart

daha önce release e ne isim verdiysek upgrade ederken o ismi kullanmamız gerekiyor.

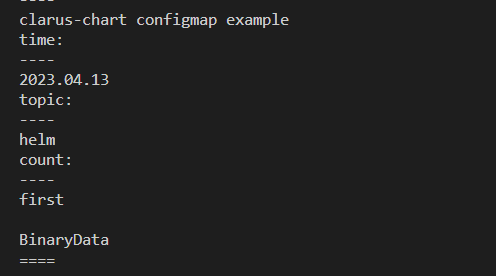
kubectl ger cm describe ile baktığımızda data yı second olarak görüyoruz:

Namespace: 
Labels : 
Annotations : 
Data 
count : 
second 
course: 
default 
app. kubernetes . io/managed -by—Helm 
meta . helm. sh/release-name: frun-local-repo 
meta . helm. sh/release-namespace: default 

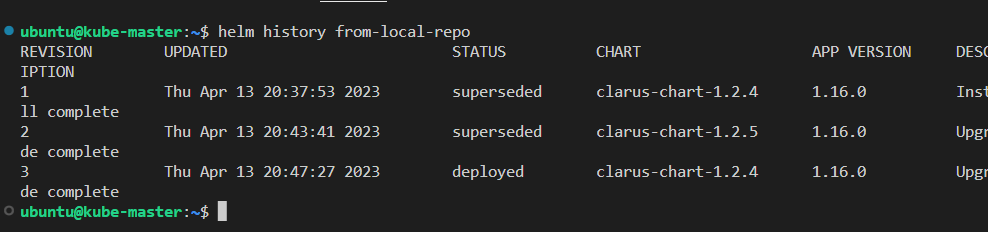
helm upgrade --install komutunda da o release yoksa release i oluşturuyor ve sonraki adımlarda da varsa upgrade anlamına gelir.

upgrade komutuyla eski bir versiona da geçebiliyoruz.

versiona geçtik ve describe komutu ile cm ye bakalım:



count first oldu.



helm history ile de versionları görüyoruz.

helm rollback komutuyla da versionun sıra numarasıyla o versiona gidebiliriz:

helm rollback from-local-repo 1



releaseleri silebiliriz

helm uninstall from-local-repo

helm repo remove mylocalrepo

repoyu da kaldırıyoruz bu komutla

Kubernetes’in paket managerı helm dir.

Helm is the package 
manager for Kubernetes 
HELM 

Helm de paket dediğimiz bizim manifest dosyalarımız, yaml file larımız.

Biz oluşturduğumuz dosyaları paketleyip her yerde kullanabiliyoruz.

Yaml file ların paketlenmiş haline chart, bulunduğu yere reposityory deniyor.

Release ise chartın instance yani çalışan hali. cluster a indirilip çalıştırılan halidir.

Helm yükleme ve vesion görme komutları:

Helm in dokümantasyonunda var nereye nasıl yükleneceği.

helm - - help ile inceleyebiliriz.

bütün chartların ortak bulunduğu yere ArtifactHUB deniyor. helm hub ile burada arama yapabiliyoruz.

<https://artifacthub.io/packages/search>

helm search repo

komutu hub daki bütün chartları listeliyor:

https : //artifacthub.io/packages/helm/geek- cookb... 
Lilly configurable Zwave to IOTT Gateway and Co... 
https : //artifacthub io/packages/helm/t ruecharts... 
ully configurable Zwave to mTT gateway and Co... 
https : //artifacthub. io/packages/helm/k8sonlab/z... 
elmchart for zwave-js-ui 
• helm search hub I wc -l 
1085e 
1 

10850 satır varmış.

helm search hub wordpress

bu şekilde de sadece wordpress chartlarını görebiliriz.

bir de helm search repo var bu da lokaldeki repoları gösterir.

@ helm repo Is 
Error: no repositories to show 

helm repo ls dediğimizde şu an lokalde bir repomuz yok.

önce bir chart reposunu lokale eklememiz gerekiyor.

helm repo add bitnami <https://charts.bitnami.com/bitnami>

komutu ile (bitnami yerine başka isim de verebilriz.) bitnami reposunu lokale ekliyoruz.

tekrar helm repo ls dediğimde:

bitnami reposunu görüyoruz, URL ile bağlıyor. chartlar inmiş olmuyor, chartların listesi geliyor.

helm search repo bitnami

komutuyla da lokalde bitnami ismini verdiğimiz repoyu araştır demiş oluyoruz.

saedece helm search repo girersek birden fazla repo olduğunda hepsini inceler.

helm pull bitnami/mysql

komutuyla da docker da imagei lokale aldığımız gibi kuberneteste chartı lokale indiroyoruz. yani bu kez istediğimiz chartı indirmiş oluyoruz.

neyi çekiyoruz? lokale eklediğimiz bitnami reposundan mysql chartını indir. ziplenmiş halde gelir.

tar -xvf mysql-\*.tgz

komutuyla ziplenmiş dosyayı dışarı çıkartabiliyoruz.

dosya yapılarını bir sonraki derste konuşacaz. içinde bir çok manifest dosyaları var. biz kendimiz de bir uygulama koymak için bu şekilde chart haline getirecektik.

helm install mysql-release bitnami/mysql

komutu da bitnami reposundaki mysql chartını install eder. şu ana kadar install etmemiştik, pull etmiştik. install ettiğimizde bu bir release olacak, burada release e mysql-release ismini verdik.

install ettik:

çalışmadı çünkü her chartın çalışması için gerekli configürasyonlar olur. örneğin burada provider dan volume gerekiyor, storage class gerekiyor.

bir repoyu çalıştırmadan önce helm repo update komutuyla lokaldeki repoları güncelleriz.

oluşan release e bakmak için helm ls:

release silmek için de helm uninstall:

şimdi wordpress çalıştıralım:

farklı değişkenleri var wordpressin veya diğer chartların. hangi değişkenleri olduğuna böyle parametrelerden de bakabiliriz, terminalden helm show values bitnami/mysql

komutuyla bakabiliriz.

bu değişkenleri kendimize göre ayarlayabilir ya da default çalıştırabiliriz.

helm show chart bitnami/mysql

komutu da chartın genel bilgilerni verir.

chartın içinde ne varsa görmek istiyorsak:

helm show all bitnami/mysql

bu komutun çıktısı yukarıdaki parametreleri terminalde uzun uzun gösterir.

Peki bir sürü değişken var dedik biz bunları nasıl değiştirecez? - - set flagiyle:

h 
elm install my-release \ 
- -set wordpressUsername=admin \ 
-set wordpressPassword=password \ 
-set mariadb. auth . rootPassword=secretpassword \ 
bitnami/wordpress 

burada username password değiştiriliyor.

Bizim chartlarda önceliğimiz kendi chartımızı repomuzu nasıl ouşturup kullanırız.

Helm de bir uygulamanını bütün kubernetes objelerini içine paketleyebiliyoruz. Release oluşturunca belki 50 tane k8s dosyası oluşuyor. Zipleyip repolarda saklıyoruz.

Helmdeki repoyu önce lokalimize ekliyoruz sonra ihtiyacımız olan chartı install ediyoruz. Bu şekilde release çalışmış oluyor.

Helm sayesinde uygulamamızın çalışması için gerekli yaml dosyalarını paketleyerek kullanmasını istediğimiz kişilere ulaştırmak çok kolay.

<https://helm.sh/docs/intro/install/>

Kendi chartımızı oluştururken bize örnek bir dizin getiriyor biz de kendimize göre düzenliyoruz.

helm create clarus-chart

Komutuyla chartımızı oluşturuyoruz.

v clarus-chart 
> charts 
> templates 
.helmignore 
! Chart.yaml 
! values.yaml 

Böyle bir dizin oluşturuyor.

.helmignore repoda bulunmasını istemediğimiz dosyaları belirtmek için, bu dosyaları pakete gömmez.

Values örnek bir dosya. Bize dinamik bir yapı sağlıyor. Uygulamayı pkaetleyip repoya koyduk, bir başkası da bunu çalıştırırken kendi değerleri ile çalıştırabilir. Terraform tfvars gibi.

Chart.yaml örnek bir dosya

# pipeline. Library charts do not de 
type: application 
# This is the chart version. This ve 
# to the chart and its templates, ir 
# Versions are expected to follow SE 
e. 1.0 
version : 
# This is the version number of the 
# incremented each time you make chi 
# follow Semantic Versioning. They 
# It is recommended to use it with 
appVersion : 
"1.16. e" 

Chartı versiyonlayabiliyoruz.

Templates içinde de yaml dosyaları var:

> tests 
helpers.tpl 
deployment.yaml 
h pa.yaml 
ingress.yaml 
NOTES.txt 
service.yaml 
serviceaccount.yaml 

Bunlar da örnek dosyalar.

Biz bu chart dizinine kendi dosyalarımızı koyacağız.

rm -rf clarus-chart/templates/\*

Komutuyla templates içini silelim. Templates içinde önce configmap dosyamızı oluşturalım. Örnek yaparken configmap kullanacağız çünkü içinde data var ve bu sayede yapılan değişiklikleri hemen görebilecez.

O README.md 
configmap.yaml X 
clarus-chart > templates > ! configmap.yaml 
1 
2 
3 
5 
6 
apiVersion: VI 
kind: Configmap 
metadata : 
name: clarus-chart-config 
data: 
myvalue : 
"Hello World" 

Chartımız hazır. Bunu install ederek configmap imizin oluşması gerekir:

helm install helm-demo clarus-chart

• helm install helm-demo clarus-chart 
NAME: helm-demo 
LAST DEPLOYED: Thu Apr 13 2023 
WIESPACE: default 
STATUS: deployed 
REVISION: 1 
TEST SUITE: None 

configmap oluştu

• ubuntu@kube-master : 
clarus-chart-config 
kube-root -ca . crt 
k get 
DATA 
1 
1 
cm 
AGE 
16s 
24h 

bu cm yi biz apply ile oluşturuyorduk normalde. şimdi chart ile oluşturduk.

helm ls dediğimizde:

helm-dem 
o ubu 
UAMESPACE 
default 
-master : "lesson$ 
helrn Is 
REVISICN 
11 
UPDATED 
2023-øa-13 
UTC 
STATUS 
deployed 
CHART 
clarus -chart-ø 
APP VERSI(R 
1.16.ø 

buradaki version chart.yaml dan geliyor.

• kubectl describe cm clarus-chart-config 
clarus-chart-config 
Name : 
default 
Namespace: 
Labels : 
app. kubernetes . io/managed -by—Helm 
Annotations : 
meta . helm. sh/release-name: helm-demo 
meta . helm. sh/release-namespace: default 
Data 
myvalue: 
Hello World 
BinaryData 
Events : 
<none> 

cm yi describe deidğimizde data yı da burada görebiliyoruz. şimdi chartı uninstall dediğimizde bütün objeler silinir.

• helm uninstall helm-demo 
release "helm-demo" 
uninstalled 
• k get cm 
DATA AGE 
kube-root-ca . crt 
1 
24h 

uninstall yerine delete komutu da çalışır.

values.yaml içerisine gidelim ve içini silelim.

course: DevOps

key-value sini yapıştıralım. configmap te bunu data olarak alalım:

apiVersion: v1

kind: ConfigMap

metadata:

name: clarus-chart-config

data:

myvalue: "clarus-chart configmap example"

course: {{ .Values.course }}

burada key-value yi elle yazmadık {{ .Values.course }} ile bu değeri çekebildik.

şimdi helm install --debug --dry-run mydryrun clarus-chart

komutuyla --dry-run dediğimizde objeyi oluşturmadan sanki oluşmuş gibi bize bilgileri getirir.

MANIFEST : 
# Source: clarus-chart/templates/configmap.yaml 
apiVersion: VI 
kind: ConfigMap 
metadata : 
name: clarus-chart-config 
data: 
myvalue: "clarus-chart configmap example" 
course: DevOps 

burada course keyinin value sini DevOps olarak gördük.

• kubectl create deploy mydeploy 
deployment. apps/mydeploy created (dry run) 
o ubunt*ube-masterw/lesson$ 
- -Image=nganx - 
-dry-run:client 

bu komutla bir objenin yaml dosyasını docs a gitmeden alabiliyoruz. dry run create ettikten sonra -o yaml ile yaml dosyaısın gördük.

deployment. apps/mydeploy created (dry run) 
• kubectl 
apiVersion: apps/vl 
kind; Deployment 
metadata ; 
creationTimestamp; 
labels . 
app: mydepIOY 
spec 
replicas: I 
selector: 
match Labels : 
app: mydeploy 
strategy: { } 
template: 
metadata : 
null 
creati onTimestanp ; 
labels ; 
app; oy 
spec: 
Inage: ngxnx 
name: ngxnx 
resources: {l 
status: {} 
o ubuntu€kube-master 
null 
kubectl 
create deploy mydeploy 
create deploy mydeploy 
- -Image-nglnx - 
--lmage-nglnx - 
-dry-run-client 
-dry-run=client 
-o yaml 
11 
-o yaml > mydeploy.ym 

yukarıdaki komut ile de bu yaml dosyası içeriğini my.deploy.yml a yazdırıyoruz.

status; {} 
• kubectl Create deploy mydeploy - -imaee=neinx - —dry-run—client -O yan1 > mydeploy_ynl 
@ kubeetl expose deploy mydepIOY - 
fr•an deploytnents_apps "mydeploy" not found 
• kubectl apply -f 
deployment. apps/mydeploy created 
• kubectl expose deploy mydeploy --port=8ø 
- dry - 
- -dry-run—client 
-o yanl myservice.yrnl 

mydeploy u apply ettikten sonra expose ile service objesini de oluşturabildik:

app; myaeploy 
name. mydeploy 
spec . 
replicas: I 
selector: 
match Labels: 
app: mydeploy 
strategy: 
template: 
metadata 
creationTimestamp: 
V" iabels•. 
• • app: , nydG 
containers; 
Image; nglnx 
name; nglnx 
resources: 
null 

dry-run ile oluşturmadan gördük ama şimdi de helm install ile release i oluşturalım:

helm install myvalue clarus-chart

clarus-chart tan myvalue isminde bir release oluştur.

helm ls diyelim:

• helm install myvalue clarus-chart 
NAME: myvalue 
LAST DEPLOYED: Thu 
WIESPACE: default 
STATUS: deployed 
REVISION: 1 
TEST SUITE: None 
WIESPACE 
myvalue default 
Apr 13 2023 
helm Is 
REVISION 
1 
UPDATED 
2023-04 
+øøøø 
UTC 
STATUS 
deployed 
CHART 
APP VERSION 
-13 
clarus -chart -0.1.01.16.0 

helm get manifest myvalue

komutu da myvalue release i içindeki manifest dosyalarını döndürür. bizim şu an sadece bir dosyamız var ancak ne kadar olursa o kadarını getirir:

• helm get manifest myvalue 
Source: clarus -chart/templates/configmap. yaml 
apiVersion: VI 
kind: ConfigMap 
etadata: 
m 
name: clarus-chart-config 
data: 
myvalue: "clarus-chart configmap example" 
course: DevOps 

helm delete myvalue ile release i silelim:

• helm delete myvalue 
release "myvalue" uninstalled 
o ubuntußkube-master:æ$ 

şimdi values içindeki değeri komut satırında da değiştrebiliyoruz onu dry-run ile görelim:

helm install --debug --dry-run setflag clarus-chart --set course=AWS

debug da burda debugging yapıyor, komutta hata olması durumunda.

setflag burada release in ismi. --set flagiyle course un valuesini AWS olarak değiştirdik.

terraformda da --var flagiyle variable, --varfile ile de variable dosyası editleyebiliyorduk.

apiVersion: v1

kind: ConfigMap

metadata:

name: {{ .Release.Name }}-config

data:

myvalue: "clarus-chart configmap example"

course: {{ .Values.course }}

configmap i bu şekilde güncelleyelim. .Release.Name ile yazdığımız kısımda komut satırında yazacağımız release ismini buraya yazacak:

helm install --debug --dry-run builtin-object clarus-chart

HOOKS : 
MANIFEST : 
# Source: clarus-chart/templates/configmap.yaml 
apiVersion: VI 
kind: ConfigMap 
metadata : 
name: builtin-object-config 
data: 
myvalue: "clarus-chart configmap example" 
course: DevOps 

bu sayede configmap te isim vermiyoruz da kullanan kişi release ne isim verdiyse configmap ve diğer objelerinde ismi bu şekilde değişmiş oluyor.

values dosyasını güncelleyelim:

course: DevOps

lesson:

topic: helm

helmste 60 ın üzerinde fonksiyon var. bunların bazılarını görecez:

<https://helm.sh/docs/chart_template_guide/builtin_objects/>

<https://helm.sh/docs/chart_template_guide/function_list/>

<https://pkg.go.dev/text/template>

<https://masterminds.github.io/sprig/>

configmap dosyamızı güncelliyoruz:

apiVersion: v1

kind: ConfigMap

metadata:

name: {{ .Release.Name }}-config

data:

myvalue: "clarus-chart configmap example"

course: {{ quote .Values.course }}

topic: {{ upper .Values.lesson.topic }}

time: {{ now | date "2006.01.02" | quote }}

quote, upper, now fonksiyonlarını kullanıyoruz. quote tırnak içinde getirir. date ile de tarihi bu formda getir diyoruz. yani now un değerini buraya gir.

helm install --debug --dry-run morevalues clarus-chart

komutuyla morevalues release ini görelim:

HOOKS : 
MANIFEST : 
# Source: clarus-chart/templates/configmap.yaml 
apiVersion: VI 
kind: ConfigMap 
metadata : 
name: morevalues-config 
data : 
myvalue: "clarus-chart configmap example" 
course: "DevOps " 
topic: HELM 
time: "2023.04.13" 

install edince bir de şöyle bir kısım geliyor ilk kısımda

NAME: morevalues 
LAST DEPLOYED: Thu Apr 13 2023 
WIESPACE: default 
STATUS: pending-install 
REVISION: 1 
TEST SUITE: None 
USER-SUPPLIED VALUES: 

bilgi olarak. bunlar template altında NOTES.txt içerisinde yazılan bilgilerle customize edilebilir. buraya başkaları release imizi çalıştırdığında gereken bilgileri vermek için yazılır. README dosyası gibi.

v templates 
! configmap.yaml 
NOTES.txt 

bu yazım ile yazılacak dosya ismi. bu dosya içinde built-in object ler yani dinamik değerler kullanılabilir.

Thank you for installing {{ .Chart.Name }}.

Your release is named {{ .Release.Name }}.

To learn more about the release, try:

$ helm status {{ .Release.Name }}

$ helm get all {{ .Release.Name }}

bunları NOTES.txt içine yazalım.

helm install notes-demo clarus-chart

release i notes-demo ismiyle çalıştırıyoruz:

• helm install notes-demo clarus-chart 
NAME: notes-demo 
LAST DEPLOYED: Thu Apr 13 2023 
WIESPACE: default 
STATUS: deployed 
REVISION: 1 
TEST SUITE: None 
NOTES: 
Thank you for installing clarus-chart. 
Your release is named notes-demo. 
To learn more about the release, try: 
$ helm status notes-demo 
$ helm get all notes-demo 

yazdığımız notlar release oluşturulurken bu şekilde oluşturan kişiye yazdırılacak. bu chartın nasıl kullanılacağını gösterir kullanıcıyla.

release i silelim.

chartlar uzak repolarda tutuluyor. repoda dosyalarımız paket halinde tutuluyor.

tree komutuna ihtiyacımız olabilir, yükleyelim.

Chartmuseum reposunu kullanacağız, go dilinde yazılan ve cloud storage ların da desteklendiği bir repo.

curl <https://raw.githubusercontent.com/helm/chartmuseum/main/scripts/get-chartmuseum> | bash

komutuyla chartmuseum u yüklüyoruz. bu komut da chartmuseum docs te var.

şimdi kendimize göre configure ediyoruz. bu komutlar da docs tan alınıyor:

chartmuseum --debug --port=8080 \

--storage="local" \

--storage-local-rootdir="./chartstorage"

burada şu porttan yayın yapsın diyoruz.

chartmuseum - -debug 
-storage="local" \ 
-storage-local-rootdir=" . /chartstorage" 
-port=8Ø8Ø \ 
DEBUG 
DEBUG 
INFO 
DEBUG 
Fetching chart list from storage 
{"repo": 
No change detected between cache and storage 
{"repo": 
Starting ChartMuseum 
{"host" : 
••Ø.Ø.Ø.Ø••, "port": 8080} 
Starting internal event listener 

yeni bir terminalde çalıştıralım çünkü termianli işgal edecek.

biz bunu hangi makinede kurduysak o makinede görülebilir. biz masterda çalıştırdık. görebilmek için önce instance ın 8080 portunu açalım.

Güvenli deéil 54.147.21.126:8080 
Course: AWS 13-TU... Google Takvim 13... O Your Repositories DeW3ps - Microsoft... (3) "devops" Searc... 
Welcome to ChartMuseum! 
HacklnScienc 
If you see this page, the ChartMuseum web server is successfully installed and 
working. 
For online documentation and support please refer to the GitHub project. 
Thank you for using ChartMuseum. 

daha sonra public ip ile 8080 portunda görüyoruz. gerçek hayatta çok kullanılmıyor, container ile de çalışabiliyor charmuseum. dışarıda daha çok github veya cloud providerlar kullanılıyor.

bu repodaki chartları kullanabilmemiz için bu repoyu locale çekebilmemiz lazım. url si olmadığı için public ip sini giriyoruz.

helm repo add mylocalrepo [http://<public-ip>:8080](about:blank)

komutuyla repoyu lokale çekiyoruz.

• helm repo Is 
WIE 
URL 
bitnami 
https : / / charts . bitnami.com/bitnami 
mylocalrepo 
http://54.147.21.126:8080/ 
o ubuntu@kube 

az önce chartmuseum da repo oluşturduk şimdi de lokale çektik. ancak bu repoda henüz bir chart yok.

helm search repo mylocalrepo

komutuyla gördük chart olmadığını.

• helm search repo mylocalrepo 
No results found 
o ubuntußkube-master:æ$ 

dersin bu anına kadar rpeo kullanmadık. lokalden çalıştık. şimdi chartımızı paketleyip repoya gönderelim:

helm package clarus-chart

• helm package clarus-chart 
Successfully packaged chart and saved it to: /hune/ubuntu/clarus-chart-ø.l.ø.tgz 

versionlayarak paketlendi. version Chart.yaml dan geldi.

şimdi paketimizi bir klasöre atalım (zorunlu değil)

mkdir my-charts

mv clarus-chart-0.1.0.tgz my-charts

repomuzda olmazsa olmaz bir dosya var index dosyası. bu chartları tutuyor. repomuzda index imizi oluşturalım:

helm repo index my-charts --url [http://<public-ip>:8080](about:blank)

helm repo index le oluşturuyoruz. my-charts paketimizin bulunduğu klasör ismi. gerisi de repomuzun url si.

v my-charts 
clarus-chart-0.1.0.tgz 
! index.yaml 

my-charts klasörü altında index.yaml oluştu.

/-charts > ! index.yaml 
1 
2 
3 
5 
6 
7 
8 
9 
12 
13 
15 
apiVersion: VI 
entries : 
clarus-chart : 
apiVersion: v2 
appVersion : 
1.16.ø 
created 
description: A Helm chart for Kubernetes 
digest: 67e1d3e1f986e6a83e695eeeae3c9Ø972639793eeeb3afØe91Øba69ddc98b39Ø 
name: clarus-chart 
type: application 
urls: 
http : / /54.147.21.126 : 8Ø80/c1arus-chart-Ø.1.0. tgz 
version: 0.1.0 
generated • 
. "2Ø23-øa-13T19: 54: 51.99435735Z" 

halen repoya göndermedik.

my-charts içine girelim.

curl --data-binary "@clarus-chart-0.1.0.tgz" [http://<public](about:blank) ip>:8080/api/charts

curl --data-binary ile dosya gönderebiliyoruz. curl kapsamlı bir komut.

• curl --data-binary "@c1arus-chart-Ø.1.Ø.tgz" htt 
{"saved":true}ubuntu@kube-master:æ/my-charts$ 

şu anda paketi repoya gönderidk.

C A Güvenli deéil 54.147.21.126:8080/api/charts 
AWS Management... Course: AWS 13-TU... Google Takvim 13... O Your Repositor 
["clarus -chart" : [{"name" : "clarus-chart" , "version" : "O. I.e" , "description" : "A Hell 
-hart-e.l.ø. tgz"], "created . 
" • "2023-e4-13T19 : 58 : 21.551784589Z 
" , "digest" : "67aId3e 

/api/charts a gittiğimizde gördük. ancak lokalde göremiyoruz:

{"saved":true}ubuntu@kube-master:æ/my-charts$ helm search repo mylocalrepo 
No results found 

bunun için repoyu update etmemiz gerkeiyor.

• helm repo update 
Hang tight while we grab the latest from your chart repositories... 
.. Successfully got an update from the "mylocalrepo" chart repository 
.. Successfully got an update from the "bitnami" chart repository 
Update Complete. •Happy Helming! 
• N/my-charts$ 
• ubuntu@kube-master . 
mylocalrepo/clarus-chart 
Kubernetes 
o ubuntu@kube-master . 
N/my-charts 
helm search repo mylocalrepo 
CHART VERSION APP VERSION 
ø.l.ø 
1.16.ø 
DESCRIPTION 
A Helm chart for 

artık repomuzda chartımızı lokalden de görebiliyoruz.

başkası bu uygulamamızı deploy etmek isterse repomuza url den ulaşabilir ve bu repo içerisindeki chbu chartı çekip install eder.

diyelim ki uygulamamızda değşiklik yaptık ve version ekleyecez. bunun için yeni bir pket oluşturuyoruz.

Chart.yaml da verisonu güncelledik ve paket oluşturma komutunu girdik:

• helm package clarus-chart 
Successfully packaged chart and saved it to: /hune/ubuntu/clarus-chart-ø.l.l.tgz 

helm repo index komutu bizim için yeni verisonu ekleyerek index i de güncelleyecek:

helm repo index my-charts --url [http://http://54.147.21.126/:8080](http://http//54.147.21.126/:8080)

digest • 
name: clarus-chart 
type: application 
urls: 
http : / /http:/54.147.21.126/ : 808Ø/c1arus -chart-e .1.1. tgz 
version: 0.1.1 
apiVersion: v2 
appVersion : 
1.16.ø 
created 
description: A Helm chart for Kubernetes 
digest: 67e1d3e1f986e6a83e695eeeae3c9Ø972639793eeeb3afØe91Øba69ddc98b39Ø 
name: clarus-chart 
type: application 
urls: 
http : / /http:/54.147.21.126/ : 808Ø/c1arus -chart-e .1. e . tgz 
version: 0.1.0 
generated 

komtuyla uzak repoya da yeni versionu gönderelim:

curl --data-binary "@clarus-chart-0.1.1.tgz" <http://54.147.21.126/:8080/api/charts>

lokalde görmek için de lokali update ediyoruz.

artifecthub dan inddireceğimiz bir plugin ile bu manuel işlemler kolaylaşıyor ve helm push komutuyla pkaeti gönderebiliriz.

büyük projede de s3 için böyle bir plugin kullanacaz.

<https://github.com/chartmuseum/helm-push>

helm plugin install <https://github.com/chartmuseum/helm-push>

komutuyla plugin i indirdik.

Chart.yaml da da versionu 0.2.0 yapalım.

helm cm-push clarus-chart mylocalrepo

komutuyla direk chart ı push edebiliriz:

• helm cm-push clarus-chart mylocalrepo 
Pushing clarus-chart-Ø.2.Ø.tgz to mylocalrepo... 
Done. 
O ubuntußkube-master 

biz manuel olarak paketliyorduk, indexi güncelliyorduk vs. cm push plugin bizim için bunları yaptı.

C A Güvenli deéil 54.147.21.126:8080/api/charts 
AWS Management... Course: AWS 13-TU... Google Takvim 13... 
{ "clarus -chart" : [{"name" : "clarus-chart" , "version" : "O. 2.e" , "descript 
chart-e.2.ø. tgz"], "created . 
" • "2023-e4-13T2e : 22.259414734z" , "diges 
chart", "version" : "a.I.O", "description": "A Helm chart for Kubernetes 
" , " digest 
" : "67a1d3e1f986e6a83e695eeaae3c9a 
e4-13T19 : 58 : 21.551784589Z 

browserda version 0.2.0 ı görüyoruz.

versionu değiştirmeseydik aynı versiondan chart var uyarısı alırdık. ancak bunu da --force ile override yapabiliriz.

loaklimizi güncellediğimizde yeni versionu görebiliyoruz:

• helm 
mylocalrepo/clarus-chart 
search repo mylocalrepo 
CHART VERSION 
ø.2.ø 
APP VERSION 
1.16.ø 
DESCRIPTION 
A Helm chart for Kubernetes 

cm-push pluginiyle komut satırında da version güncelleme yapabiliyoruz:

helm cm-push clarus-chart mylocalrepo --version="1.2.3"

• helm 
mylocalrepo/clarus-chart 
search repo mylocalrepo 
CHART VERSION 
1.2.3 
APP VERSION 
1.16.ø 
DESCRIPTION 
A Helm chart for Kubernetes 

şimdi configmap e count=first key ini ekleyelim:

apiVersion: v1

kind: ConfigMap

metadata:

name: {{ .Release.Name }}-config

data:

myvalue: "clarus-chart configmap example"

course: {{ .Values.course }}

topic: {{ .Values.lesson.topic }}

time: {{ now | date "2006.01.02" | quote }}

count: "first"

helm cm-push clarus-chart mylocalrepo --version="1.2.4"

yeni versionla cm-push edelim.

bu chartı çalıştırmak için lokal repomuzu güncellememiz gerkiyor.

helm repo update 
helm search repo mylocalrepo 
helm install from-local-repo •mylocalrepo/clarus-chart 

mylocalrepo/ prefixini ekleyerek de lokaldeki chartı çalıştırmasını istiyoruz. update ettiğimiz için pull etmemize gerek yok.

• kubectl 
DATA 
frun-local-repo-config 5 
kube-root -ca . crt 
1 
• kubectl 
DATA 
from-local-repo-config 5 
get cm 
AGE 
2m13s 
26h 
get cm frun-local-repo-config 
AGE 
2m28s 
-o wide 

cm yi gördük.

configmap teki first yerine second yazalım.

1.2.5 versionu ile push edelim chartı:

helm cm-push clarus-chart mylocalrepo --version="1.2.5"

helm repo update diyoruz.

şimdi configmap içeriğindeki data first mü gelir second mu? repoyu update etmek chartı update etmek demek değildir. uygulama halen eski versionda çalışıyor.

helm upgrade komutuyla chartı güncellersek yeni versiona geçer:

helm upgrade from-local-repo mylocalrepo/clarus-chart

daha önce release e ne isim verdiysek upgrade ederken o ismi kullanmamız gerekiyor.

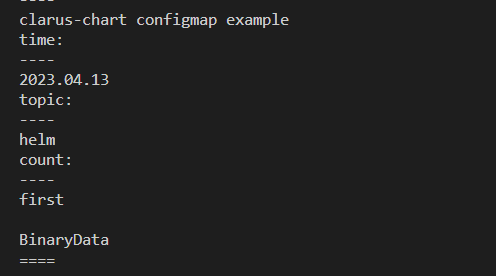
kubectl ger cm describe ile baktığımızda data yı second olarak görüyoruz:

Namespace: 
Labels : 
Annotations : 
Data 
count : 
second 
course: 
default 
app. kubernetes . io/managed -by—Helm 
meta . helm. sh/release-name: frun-local-repo 
meta . helm. sh/release-namespace: default 

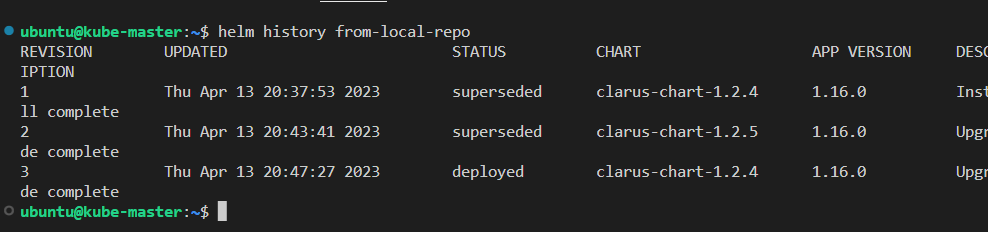
helm upgrade --install komutunda da o release yoksa release i oluşturuyor ve sonraki adımlarda da varsa upgrade anlamına gelir.

upgrade komutuyla eski bir versiona da geçebiliyoruz.

versiona geçtik ve describe komutu ile cm ye bakalım:



count first oldu.



helm history ile de versionları görüyoruz.

helm rollback komutuyla da versionun sıra numarasıyla o versiona gidebiliriz:

helm rollback from-local-repo 1



releaseleri silebiliriz

helm uninstall from-local-repo

helm repo remove mylocalrepo

repoyu da kaldırıyoruz bu komutla

bu işlemleri github da yapalım..

mygithubrepo isminde bir repo oluşturalım.

mkdir mygithubrepo

cd mygithubrepo

echo "# mygithubrepo" >> README.md

git init

git add README.md

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

git commit -m "first commit"

git branch -M main

git remote add origin [https://github.com/<your](https://github.com/%3Cyour) github name>/mygithubrepo.git

git push -u origin main

komutlarıyla da git reposunun lokaldeki reposunu oluşturalım.

helm create demogitrepo

komutuyla demogitrepo isimli chartımızı oluşturalım.

mygitrepo klasörüne gidelim ve helm package ../demogitrepo

komutuyla demogitrepoyu paket hale getirelim.

bu klsörde index dosyamızı da oluşturalım:

helm repo index .

• helm create demogitrepo 
Creating demogitrepo 
• cd mygithubrepo/ 
• helm package .. /demogitrepo 
Successfully packaged chart and saved it to: /hune/ubuntu/mygithubrepo/demogitrepo-ø.l.ø.tgz 
• helm repo index 

yaptığımız değişiklikleri commitleyip push edelim.

yemreaydeniz/ mygithubrepo Public 
< > Code @ 
Issues 
Pull requests @ Actions 
Projects 
Wiki 
Pin 
Security Insights Settings 
Go to file 
Add file • 
@ Unwa 
< > Code 
O tags 
main 
branch 
yemreaydeniz demogitrepo chart is added 
README.md 
demogitrepo-O. 1.0.tgz 
index.yaml 
first commit 
demogitrepo chart is added 
demogitrepo chart is added 
2 commits 
22039ae now 
3 minutes ago 
now 
now 

bu artık uzak bir chart reposu.

bu repoyu lokale eklerken;

helm repo add --username <github-user-name> --password <personel-access-token> my-github-repo '[https://raw.githubusercontent.com/<github-user-name>/mygithubrepo/main](https://raw.githubusercontent.com/%3Cgithub-user-name%3E/mygithubrepo/main)'

komutunu kullanıyoruz.

ekledikten sonra lokalimiz:

• helm repo list 
bitnami 
mylocalrepo 
URL 
https : / / charts . bitnami.com/bitnami 
http://54.147.21.126:8080/ 
my-github-repo https : / /raw. githubusercontent. cun/yemreaydeniz/mygithubrepo/main 

şimdi repoya yeni bir chart ekleyelim:

cd ..

helm create second-chart

cd mygithubrepo

helm package ../second-chart

helm repo index .

git add .

git commit -m "second chart is added"

git push

bu komutlarla ikinci chartı ekleyip repoya push ettik.

lokali update ettik:

• helm repo update 
Hang tight while we grab the latest from your chart repositories... 
.. Successfully got an update from the "mylocalrepo" chart repository 
.. Successfully got an update from the "my-github-repo" chart repository 
.. Successfully got an update from the "bitnami" chart repository 
Update Complete. •Happy Helming! 
• helm search repo my-github-repo 
CHART VERSION APP VERSION 
my-github-repo/demogitrepo 
ø.l.ø 
1.16.ø 
DESCRIPTION 
A Helm chart for Kubernetes 
o ubuntußkube-master:æ/mvzithubreøo 

ancak github repodan lokale güncelleme çekerken biraz gecikme olabiliyor. bu nedenle halen ikinci chart görünmüyor.

helm install github-repo-release my-github-repo/second-chart

komutuyla ikinci charttan bir release oluşturalım.

objeleri kontrol edelim:

kubectl get deployment

kubectl get svc

<https://raw.githubusercontent.com/ofidan/myapp/main>

burada da phonebook projenin chartı var.