Static 
Dynamic 

Daha önce static provisioningde önceden belirlenen ebatlarda PV ler oluyordu ve PVC ile userlar volume talep ettiğinde talebine en yakın PV ile eşleşiyordu.

Dynamic provisioning ise önceden rezerve edilmiş PV ler yok, storage classlar oluşturuluyor ve PVC ler ile talep oluşturuluyor ve storage class objesiyle eşleşiyor ve talebi kadar bir volume u alabiliyor. Storage class oluştururken boyut bilgisi girmiyoruz, PVC ne kadar boyutta talep ediyorsa o kadar volume alıyor.

CD 
C) Provisioning 
Administrator 
Request a Claim 
SCSI 
NFS 
GCE 
Persistent Volumes Pool 
contair.er 
cytairø 
Claim request honoured 3 
Mount the PVC to a Pod 
User 
Volume 
pod 

A StorageCIass provides a way for administrators to describe the 
"classes" of storage they offer. 
storageclass 
(aws-ebs 
type:gp2) 
storageclass 
(aws-ebs 
type:iol) 

Admin birden çok storage class objesi oluşturabilir. Storage miktarı ile ilgili değil, farklı tiplerde objeler oluşturulur bu durumda.

PV Pool 
PV-2 
PV-3 
User 
Administrator 
StorageClass 
PVC 
POD 

Yöntem statik ile aynı ancak burada dynamic bir olay var.

PVC ile talep oluşturduğumuzda ise iki türlü bağlanma yöntemi oluyor;

Immediate ve WaitForFirstConsumer.

Immediate PVC oluşur oluşmaz gider storage class ile eşleşerek volume e bağlanır. WaitForFirstConsumer ise hemen bağlanmaz, o PVC belirtilerek pod oluşunca volume bağlanıyor.

kind: StorageC1ass 
apiVersion: storage . 
metadata : 
name: aws—standard 
annotations : 
k8s. 
io/vl 
storageclass . kubernetes . i o/ is—default 
true 
provisioner: kubernetes . io/aws—ebs 
parameters : 
type: gp2 
fsType: ext4 
—class : 
Provisioner: Each StorageClass 
has a provisioner that determines 
what volume plugin is used for 
provisioning PVs. 
Parameters: Storage Classes 
have parameters that describe 
volumes belonging to the storage 
class. Different parameters may be 
accepted depending on 
the 
provisioner 

Storage classlarda bu şekilde bir annotation girildiğinde onun default olduğunu belirtir.

provisionlar kubernetes docs ta mevcut:

Volume Plugin 
AWSEIastic810ckStore 
AzureFiIe 
Azu reDisk 
CephFS 
Cinder 
FlexVoIume 
GCEPersistentDisk 
VsphereVoIume 
PortworxVoIume 
Local 
Internal Provisioner 
Config Example 
AWS Eas 
Azure File 
Azure Disk 
OpenStack Cinder 
GCE PO 
N FS 
ceph RBD 
VS p here 
Portworx Volume 
Local 

hangisini kullanacaksak gidip ona tıklar ve hangi parametreleri girebileceğimizi görebiliriz. EBS için bakalım:

AWS EBS 
apiVersion: storage. k8s . io/vl 
kind: StorageC1ass 
metadata : 
name: slow 
provisioner: kubernetes . io/aws-ebs 
parameters : 
type: iol 
iopsPerGB: "18" 
fsType: ext4 
• type : iol , gp2 , scl , sti . See AWS docs for details. Default: gp2 
• zone (Deprecated): AWS zone. If neither zone nor zones is specified, volumes 
are generally round-robin-ed across all active zones where Kubernetes cluster has 
a node. zone and zones parameters must not be used at the same time. 
• zones (Deprecated): A comma separated list of AWS zone(s). If neither zone nor 
zones is specified, volumes are generally round-robin-ed across all active zones 
where Kubernetes cluster has a node. zone and zones parameters must not be 
used at the same time. 

storageclass ta kapasite belirlemiyoruz, provisionerı belirliyoruz, PVC de talep edilen kadarını provisionerdan kullanır.

t2.micro 
- kubectl 
- eksctl 
- aws configure / Role 
$ eksctl create cluster 
NODE-I 
t2.micro 
NODE-2 
t2,micro 
sc 
CONTROL PLANE/EKS 
PVC 
POD 

Handsonda bize bir cluster ihtiyacımız olacak, bunun için bir instance kullanacaz ve oraya kubectl ile ekstctl kuracağız, aws credentials larımızı girecek ya da role atayacaz. EKS ise bize ng oluşturup iki tane node oluşturacak.

Daha sonra ingress konusunu göreceğiz. Ingress i EKS de oluşturmak daha kolay:

StoreFrontUl 
Accounting 
Inventory 
Shipping 

Ingress uygulamamıza dış dünyadan ulaşılmasını sağlayan ve bir domain adı altında bunu yapan bir API objesidir.

NodePort: 30011 
StoreFront 
Service 
StoreFrontlJl 
Deployment 
NodePort: 30022 
Accounting 
Service 
Accounting 
Deployment 
NodePort: 30033 
Inventory 
Service 
Inventory 
Deployment 
NodePort: 30044 
Shipping 
Shipping 
Deployment 

biz internet sitemizin dış dünyadan ulaşılabilmesi için service objelerini kullanıyorduk, ve sitenin farklı kısımları için farklı deploymentlar ve faklı service ler oluyor. Internet sitesine bu şekilde port girerek bağlanmak gerçekçi değil.

clarusshop.com 
NodePort: 30011 
StoreFront 
Service 
StoreFrontUl 
Deployment 
accounting.clarusshop.com 
NodePort: 30022 
Accounting 
Service 
Accounting 
Deployment 
inventory.clarusshop.com 
NodePort: 30033 
Inventory 
Service 
Inventory 
Deployment 
4 
shipping.clarusshop.com 
N0dePort: 30044 
Shipping 
Service 
Shipping 
Deployment 

Ancak biz service objesinin tipini LoadBalancer tipinde yapabiliyoruz, provider kullandığımızda. Bu loadbalancer DNS ini route53 te girerek bir internet sitesi gibi domain name ile girilebilir yapabiliriz ancak her service için farklı bir loadbalancer ne kadar mantıklı? Sadece bir uygulama için 20 civarı service olabilir.

Bunun için ingress yöntemi var:

clarusshop.com 
clarusshop.com/ 
StoreFront 
Service 
clarusshop.com/accounting 
Accounting 
Service 
clarusshop.com/inventory 
Inventory 
Service 
clarusshop.com/shipping 
Shipping 
Service 

Ingress objesi clusterın içinde ancak Cluster dışında bir tane loadbalancer oluşturuyor. artık bir tane loadbalancer ım olduğu için onu route53 ile domain name e bağlıyoruz.

Ingress in rules kısmı var, orada kuralları belirliyoruz ve o kurallara göre loadbalancer ın / , accounting, inventory, shipping sayfalarına yönlendirmesini sağlıyoruz.

Ingress exposes HTTP 
and HTTPS routes from 
outside the cluster to 
services within 
the 
cluster. 
clarusshop.com/ 
StoreFront 
Service 
DAR 
clarusshop.com 
Ingress 
clarusshop.com/accounting 
Accounting 
Service 
clarusshop.com/inventory 
Inventory 
Service 
Rules: 
clarusshop.com/ 
clarusshop.com/accounting 
clarusshop.com/inventory 
clarusshop.com/shipping 
clarusshop.com/shipping 
Shipping 
Service 

Özetle, HTTP ve HTTPS trafiğini yönlendiren bir API objesidir. Bir routing objesidir diyebiliriz.

"An Ingress is a collection of rules that allow inbound connections 
to reach the cluster Services." 
client 
Ingress-managed 
load balancer 
Ingress 
cluster 
—-routing 
pod 
Service 
pod 

With Ingress, 
users do 
apiVersion: networking . k8s . io/vl 
kind: Ingress 
metadata : 
name: ingress—service 
annotations : 
kubernetes . io/ ingress . class : 
spec : 
rules : 
http: 
paths : 
path: / 
pathType: Prefix 
backend : 
service: 
name: web—service 
Port : 
number: 3000 
path: / load 
pathType: Prefix 
backend : 
servxce: 
port : 
number : 
' nginx' 
not connect directly to a 
Service. Users reach the 
Ingress endpoint, and, 
from there, the request is 
forwarded to the desired 
Service. 
ARUSWAYO 
TO REINVENT YOURSELF 
name: php—apache—service 
80 

Kubernetes ingress tipine de karışmıyor, piyasada en çok kullanılan nginx ingress tipi var.

Kubernetes cluster 
O 
O 
Cloud 
provider 
O 
O 
O 

Kubernetes yapısı; kube control manager vardı; bir değil birden fazla var. Kubernetes te çoğu objenin arkada controller ı var.

Ingress in ise burada bir controller ı yok. Ingress manifest dosyasını çalıştırınca ingress çalışmaz. Ingress controller ın kurulması lazım.

Kullanılabilecek ingress controller çok fazla ancak kubernetes in desteklediği AWS, GCE ve nginx ingress controller idir.

https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/

Kubernetes as a project supports and maintains AWS, GCE, and nginx ingress 
controllers. 

Daha önce gördüğümüz üzere kubernetes içinde de loadbalancing yapılıyor ancak

. ingress in olutşurduğu load balancer, provider ın verdiği load balancing hizmetiyle gerçekleşiyor. Yani EKS clusterında ingress kurarsak aslında AWS de bir load balancer oluşmuş oluyor.

Ingress handsonunda ise web-service ve db service oluşturacaz, bir de php-apache service oluşutracaz ve bu iki ayrı service e tek yolla ulaşabilmek için ingress objesi oluşturacaz:

cluster 
web-Service 
INGRESS 
/load 
php-apache-service 
POD 
POD 
POD 
POD 

üzerinde çalışacağımız bir t2.micro ec2 oluşturalım ve makineye bağlanalım.

bashrc ye bu komutları ekleyebiliriz ancak ekledikten sonra altına birkaç enter vuralım:

alias k='kubectl'

export PS1="\[\e[1;34m\]\u\[\e[33m\]@\h# \W:\[\e[32m\]\\$\[\e[m\] "

sistemi update ediyor ve kubectl kuruyoruz.

curl -o kubectl <https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.7/2022-06-29/bin/linux/amd64/kubectl>

chmod +x ./kubectl

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin

echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc

(bu komut ileride de işimize yarayabilir, bashrc ye bir şey yazdırmak için)

kubectl version 
ec2-use 
Client Version: VI. 23.7-eks-4721ØIØ 
-short 
-client 

kubectl versionumuzu gördük.

eksctl kuracağız, amazon EKS için resmi CLI :

<https://eksctl.io/>

Customize your cluster by using a config file. Just run 
eksctl create cluster -f cluster . yarnl 
to apply a cluster .yaml file: 
apiVersion: eksctl . io/v1a1pha5 
kind: Cluste rConfig 
metadata : 
name: basic-cluster 
region: eu-north--l 
nodeG roups : 
name: ng--l 
instanceType: m5 . large 
desiredCapacity : 
la 
name: ng-2 
instanceType: m5 . xlarge 
desiredCapacity : 
2 

burada manifest dosyası girip apply diyebiliriz ancak biz bunu tek bir komut satırında yapacaz.

AWS dokümantasyonunda da komutlar var:

<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>

bu arada Amazon 2023 ami güncellemesi geldiği için CLI version 2 geliyor artık güncelleme yapmamıza gerek yok:

• ec2- 
o ec2- 
user-U-172-31-95-222* lesson:$ aws 
user-up-172-31-95-222* lesson:$ 
- -version 
prompt/off 

eksctl i yükledik.

ec2-use 
ownload/eksctl_$(uname -s) 
ec2-use 
rap-172-31-29-123* 
ec2-use 
ø.136.ø 
ec2-use 
N: curl - -silent 
- -location "https://github.com/weaveworks/eksctl/releases/latest/d 
amd64.tar.gz" I tar xz -C /tmp 
sudo mv /tmp/eksctl /usr/local/bin 
eksctl version 

aws configure komutuyla credential tanıtalım.

ssh-keygen komutu key dosyası oluşturuyor:

`ssh-keygen -f ~/.ssh/id\_rsa`

bizim .ssh altında pem dosyamız var ve bununla instance a bağlanıyoruz. bu key in private ve public tarafı var. private tarafı anahtar. public tarafı makine içinde private tarafı da bizim lokalimizdeki pem dosyamızda duruyor.

biz cluster içindeki worker node lara bağlanma ihtiyacı duyarsak bir private key ihtiyacımız olacak. ssh-keygen komutu bize public ve private key üretiyor.

komutu girip gelen sorulara enter a basıyoruz.

v .ssh 
authorized_keys 
id rsa 
id_rsa.pub 

.ssh altında bir private ve bir pubilc key oluşturdu.

biz worker node lara bağlanabilmek için public tarafını eksctl komutuyla worker node lara gönder diyecez. bir key file ın default ismi id\_rsa dır.

biz ssh ile instance a bağlanırken -i ile isim giriyoruz çünkü ismi id\_rsa değil.

eksctl create cluster --help

komutunu girdiğimizde konsolda manuel olarak yaptığımız parametreleri görebiliyoruz.

ec2-use 
Create a cluster 
eksctl create cluster 
- -help 
Usage: eksctl create cluster [flags] 
General flags : 
n, - -name string 
- -tags stringToString 
- -region string 
-with -oidc 
- -zones strings 
- -version string 
-config-file string 
-timeout duration 
- -fargate 
to Fargate 
-dry-run 
Initial nodegroup flags : 
-nodegroup-name string 
- -without -nodegroup 
-node-type string 
-t, 
N, - -nodes int 
-nodes -min int 
EKS cluster name (generated if 
Used to tag the AWS resources . 
AWS region. Defaults to the va] 
Enable the IAM OIDC provider 
(auto-select if unspecified) 
Kubernetes version (valid optic 
load configuration from a file 
maximum waiting time for any Ic 
Create a Fargate profile schedL 
Dry-run mode that skips cluster 
name of the nodegroup 
if set, initial nodegr 
node instance type 
total number of nodes 
minimum nodes in ASG 

girmediğimiz değerler için de default değerleri giriyor.

eksctl create cluster \

--name cw-cluster \

--region us-east-1 \

--zones us-east-1a,us-east-1b,us-east-1c \

--nodegroup-name my-nodes \

--node-type t3a.medium \

--nodes 2 \

--nodes-min 2 \

--nodes-max 3 \

--ssh-access \

--ssh-public-key ~/.ssh/id\_rsa.pub \

--managed

en sondaki managed ise AWS tarafından yönteilsin demek ancak zaten AWS tarafından yöntiliyor.

Şirketler bunu yaml dosyası ile yapar.

komutlarıyla cluster ı kuralım. ya da tek satırda:

eksctl create cluster --region us-east-1 --zones us-east-1a,us-east-1b,us-east-1c --node-type t3a.medium --nodes 2 --nodes-min 2 --nodes-max 3 --name cw-cluster

EKS cluster için VPC oluşturuyor:

your vpcs (3) 
Q Filter vpcs 
Name 
cla rus-vpc-a 
eksctl-cw-cluster-cluster/VPC 
default-vpc 

Cloudformation da süreci görebiliyoruz:

CloudFormation > Stacks 
Stacks (2) 
Q Filter by stack name 
c 
Delete 
Active 
update 
Stack actions 
View nested 
Description 
Create stack 
O 
O 
Stack name 
eksctl-cw-cluster-nodegroup-my-nodes 
eksctl-cw-cluster-cluster 
Status 
@ CREATE_COMPLETE 
@ CREATE_COMPLETE 
Created time 
2023-04-08 UTC+0200 
2023-04-08 UTC+0200 
EKS Managed Nodes (SSH access: true) 
[created by eksctl] 
EKS cluster (dedicated VPC: true, dedicate 
IAM: true) [created and managed by eksc 

biz aws cli dan kurulum yaparken manuel işlemler yapmıştık, biri de config dosyasını oluşturmaktı. eksctl ise bunları yapıyor, config dosyasını da oluşturdu:

v .kube 
> cache 
config 

storage class ile EBS i kullanmamız için kubernetes in getirdiği son güncelleme ile CSI kurulumu yapmamız gerekiyor.

CRI arayüzü var Kubernetes in. Container Runtime Interface, biz burada container ın çalıştığı engine lerden farklı farklı kurabiliyorduk. Networkle alakalı da CNI Cluster Network Interface var burada da farklı farklı networkler kurabiliyoruz.

Storage için de Cluster Storage Interface var, EBS, Disk vs. istediğimizi kullanabiliyoruz. Arayüzleri API objesi gibi düşünelim.

eksctl de de şu an EBS i kurmak için kurulum yapmamız gerekiyor.

AWS dokümantasyonuna göre bunu add-on ya da manuel kurulum şeklinde yapabiliyoruz:

<https://docs.aws.amazon.com/eks/latest/userguide/ebs-csi.html>

add-on:

<https://docs.aws.amazon.com/eks/latest/userguide/managing-ebs-csi.html>

manuel:

<https://github.com/kubernetes-sigs/aws-ebs-csi-driver>

biz add-on şeklinde yapacaz.

OIDC denen bir tabir var, kubernetes te authentication ve authorization yapması için Amazon un bir kontrolü var.

oidc\_id=$(aws eks describe-cluster --name cw-cluster --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)

komutuyla değişken atıyoruz sonra grep ile çaırıyoruz:

aws iam list-open-id-connect-providers | grep $oidc\_id

ec2-use 
rap-172-31-29-123# 
" - -output text 
Issuer 
ec2-use 
rap-172-31-29-123# 
ec2-use 
rap-172-31-29 
-.:$ oidc_id=$(aws eks 
I cut -d '/' -f 5) 
-123# 
describe-cluster 
--name cw-cluster - -query 
N: aws iam list-open-id-connect-providers I grep $oidc_id 
"clust 

bizim cluster ımızda böyle bir şey olmadığı için gelmemesi lazım biz oluşturacaz:

eksctl utils associate-iam-oidc-provider --region=us-east-1 --cluster=cw-cluster --approve

eksctl utils associate-iam-oidc-provider 
- -region—us-east-I 
ec2-use 
pprove 
2023-04-08 10:43:42 C i] will create IAM Open ID Connect provider for cluster "cw-cluster" 
2023-04-08 10:43:42 created IAM Open ID Connect provider for cluster "cw-cluster" in 
-cluster=cw-clus 
in "us-east-I" 
"us-east-I" 
ec2-use 
rap-172-31-29-123* 

Amazon da ec2 ya rol atarken direk atayabiliyor muyuz? önce gidip role oluşturuyor ve profile e bağlıyoruz, makineye de profili bağlıyoruz.

Kubernetes te de buna benzer yetkilendirmeler gerekiyor, roller atanıyor. herkes her namespace e ulaşamaz. her pod a ulaşamaz.

Kuberneteste de role bağlamak için arada service account var. rolü service account a bağlıyoruz. cluster ya da podlara da rolü bu service account sayesinde yapıyoruz. bunu ileride görecez.

bash 
eksctl create iamserviceaccount \ 
-name ebs-csi-controller-sa \ 
-namespace kube-system \ 
-cluster cw-cluster \ 
-attach-policy-arn arn: aws: iam: :aws:p01icy/service-r01e/AmazonEBSCSIDriverP01icy \ 
-approve \ 
-role-only \ 
-role-name AmazonEKS EBS CSI DriverR01e 

buradaki policy ile en aşağıdaki isimle bir role oluşturuyor ve bunu cluster a bağlıyor. iamserviceaccount oluşturarak bunu yapıyor. bunu neden yapıyor? cluster ın AWS servicelerine erişebilmesini sağlamaktır. biz burada EBCCSI driver policy ile bir rol atıyoruz.

bu komutu çalıştırıyoruz.

bundan sonra bu isimle bir rol oluşacak, EKS cluster ı silince o rolü de silecek.

CloudFormation > Stacks 
Stacks (3) 
Q Filter by stack name 
c 
Delete 
Active 
update 
O 
O 
O 
Stack name 
eksctl-cw-cluster-addon- 
iamserviceaccount-kube-system-ebs-csi- 
controller-sa 
eksctl-cw-cluster-nodegroup-my-nodes 
eksctl-cw-cluster-cluster 
Status 
@ CREATE_COMPLETE 
@ CREATE_COMPLETE 
@ CREATE_COMPLETE 
Created time 
2023-04-08 UTC+0200 
2023-04-08 UTC+0200 
2023-04-08 UTC+0200 
Stack actions 
View nested 
Description 
IAM role for serviceaccoL 
system/ebs-csi-controlle 
managed by eksctl] 
EKS Managed Nodes (SSI 
[created by eksctl] 
EKS cluster (dedicated VI 
IAM: true) [created and n 

cloudformationda da görebiliyoruz.

EBS CSI için add-on ekleyecez demiştik:

Overview 
Resources 
Compute 
Networking 
Add-ons 
Authentication 
Logging 
Feedback Survey 
Tell us what ym_l think about the AWS Add-ons for EKS 
Edit 
Update history 
Remove 
Tags 
Take the survey 
more a 
Add-ons (O) Info 
Q Find add-on 
View details 
Any category 
No add-ons 
No add-ons are configured for this cluster. 
Any status 

şu an add on yok. istersek biz bunu manuel olarak konsoldan add-on ekleriz:

kube-proxy Info 
Enable service networking within your 
cluster. 
Category 
networking 
Amazon EBS CSI 
Driver Info 
Enable Amazon Elastic Block Storage 
(CBS) within your cluster 
Category 
storage 
coreDNS 
Enable service discovery within your 
cluster. 
Category 
networking 
AWS Distro for 
Open Telemetry Info 
Enable observability within your 
cluster. 
Category 
observability 
Amazon VPC CNI Info 
Enable pod netwnyrking within p•ur 
cluster. 
Category 
networking 
Amazon GuardDuty EKS 
Runtime 
Monitoring Info 
Install EKS Runtime Monitoring add-on 
within your cluster. Ensure to enable 
ERS Runtime Monitoring within 
Amazon GuardDuty. 
Category 
security 

ancak biz komutlayapacaz.

komutta AWS dok te var:

eksctl 
AWS Management Console 
AWS CLI 
To add the Amazon EBS CSI add-on using eksctl 
AmazonEKS EBS CSI DriverRote with the name of the IAM role created earlier. If your cluster is in the AWS GovCloud (US-East) 
or AWS GovCloud (US-West) AWS Regions, then replace arn:aws: with arn:aws-us-gov: . 
eksctl create addon 
-name aws-ebs-csi-driver 
-cluster my-cluster 
--service-account-role-arn arn:aws: 
If you remove the - -force option and there's a conflict with your existing settings, the command fails. You can use the resulting 
error message to troubleshoot the conflict. Before specifying this option, make sure that the Amazon EKS add-on doesn't manage 
settings that you need to self-manage. This is because those settings are overwritten with this option. For more information about 
managing Amazon EKS add-ons, see Kubernetes field management. 

<https://docs.aws.amazon.com/eks/latest/userguide/managing-ebs-csi.html>

eksctl create addon --name aws-ebs-csi-driver --cluster my-cluster --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS\_EBS\_CSI\_DriverRole --force

bu komutta accound id ve cluster ismini değiştirerek çalıştırıyoruz.

eksctl create i 
ec2-use 
-role-arn arn: aws:iam: :99571446521Ø: role/Amaz( 
2023-04-08 10: 53:-37 C i] Kubernetes version 
2023-04-08 10:53:-37 C i] using provided Servi( 
verR01e" 
2023-04-08 10: 53:-37 [i] creating addon 
ec2-use 

add-on oluştu.

EBS-volume gidip bakalım:

Volumes (3) 
Q Search 
Name 
cw-cluster-my-nodes-Node 
cw-cluster-my-nodes-Node 
volume ID 
vol-00549b38ec5dee9c5 
vol-03cdff5d3a8563dd2 
vol-oca8aa3d9b06367f8 
Type 
gp3 
gp3 
gp3 
Size 
80 Gia 
8 Gi3 
80 Gia 
lops 

node larımıza 80GB volume vermiş, bir şey belirtmediğimiz için.

şu an default bir sc miz var:

kubectl 
ec2-use 
rap-172-31-29-123# 
PROVISIONER 
gp2 (default) 
kubernetes . io/ aws -ebs 
-31-29-123* I 
ec2-use 
r@ip-172 
get sc 
RECLAIMPOLICY 
Delete 
VOLUMEBINDINGMODE 
WaitForFirstConsumer 
ALLOWVOLWEEXPANSION 
false 
AGE 
37m 

storage class ın default olduğunu belirten bu annotationdur:

ec2-use 
Name : 
IsDefau1tC1ass : 
Annotations : 
kubectl describe sc/gp2 
gp2 
Yes 
kubectl. kubernetes . io/last -applied -configt 
Class " , "metadata " : { "annotations " : { "storageclass . kubernetes . 
{"fsType" : "ext4" , "type" : "gp21'}, "provisioner" : "kubernetes . ic 
, storageclass . kubernetes . io/is -default -class—true 
Provisioner : 
Parameters : 
AllowV01umeExpansion : 
MountOptions : 
ReclaimP01icy : 
VolumeBindingMode : 
Events : 
ec2-use 
r@ip-172-31-29- 
kubernetes . io/ aws -ebs 
fsType=ext4 , type=gp2 
<unset> 
<none> 
Delete 
WaitForFirstConsumer 
<none> 
123* -.:$ 

değiştirirsek default değişir.

manifest imizi yazalım storage class için.

kind: StorageClass

apiVersion: storage.k8s.io/v1

metadata:

name: aws-standard

provisioner: kubernetes.io/aws-ebs

volumeBindingMode: WaitForFirstConsumer

parameters:

type: gp2

fsType: ext4

burada isim önemli çünkü isimle PVC ye bağlayacağız. apply ile oluşturalım.

kubectl 
ec2-use 
rap-172-31-29-123# 
PROVISIONER 
aws-standard 
kubernetes . io/ aws -ebs 
gp2 (default) 
kubernetes . io/ aws -ebs 
ec2-use 
r@ip-172 
-31-29-123* 
get storageclass 
RECLAIMPOLICY VOLUMEBINDINGMODE 
Delete 
Delete 
WaitForFirstConsumer 
WaitForFirstConsumer 
ALLOWVOLWEEXPANSION 
false 
false 
AGE 
22s 
41m 

allow volume expension ise pod volume arttırımı istediğinde arttırabilirsin mi?

şimdi henüz volume oluşmadı, PVC oluşturup talep edilecek:

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: clarus-pv-claim

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

storageClassName: aws-standard

storageclassname belirtmezsek gider defaulta bağlanır.

kubectl 
ec2-use 
persistentvolumeclaim/clarus-pv-claim 
ec2-use 
rap-172-31-29-123* 
get pv,pvc 
STATUS VOLWE 
Pending 
CAPACITY 
ACCESS MODES 
STORAGECLASS 
aws -standard 
AGE 
23s 

status u pending çünkü sc oluştururken waitforfirstconsumer dedik.

şimdi pod u oluşturacaz:

apiVersion: v1

kind: Pod

metadata:

name: test-aws

labels:

app : web-nginx

spec:

containers:

- image: nginx:latest

ports:

- containerPort: 80

name: test-aws

volumeMounts:

- mountPath: /usr/share/nginx/html

name: aws-pd

volumes:

- name: aws-pd

persistentVolumeClaim:

claimName: clarus-pv-claim

apply dediğimizde;

Every 2_es: kubectl get pv,pvc sat Apr 8 2023 
RECLAIM POLICY STATUS CLAIM 
CAPACITY ACCESS 
STORAGECLASS REASON AGE 
persistentv01urne/pvc-b72æa3c-+72d-a11a-8+67-2d28ed63adg6 
Delete 
Bound 
default/ clarus-pv-claim aws-standard 
STATUS VOLLME 
CAPACITY ACCESS STORAGECLAss 
persistentvolumeclaim/clarus-pv-cla im Bound 
d86 3Gi 
aws-standard 
AGE 
pvc-b727aa3c-f72d-411a-8f67-2d28ed63a 
2n56s 

bir PV oluştu, PVC de bound oldu. pod içine girip baktığımızda da nginx e mount edildiğini görüyoruz:

ec2-use 
root@test-aws:/# df -h 
kubectl exec -it test-aws 
- başh 
Filesystem 
overlay 
tmpfs 
tmpfs 
shm 
/ dev/nvme0n1p1 
/ dev/nvmelnl 
tmpfs 
tmpfs 
tmpfs 
Size 
806 
1.9G 
806 
2.9G 
3.36 
1.9G 
1.9G 
Used 
3.16 
o 
o 
o 
3.16 
12K 
o 
o 
Avail Use% 
776 
64M 
1.9G 
64M 
776 
2.9G 
3.36 
1.9G 
1.9G 
IX 
IX 
Mounted on 
/ dev 
/ sys/fs/cgroup 
/ dev/shm 
/etc/hosts 
/usr/share/nginx/html 
/ run/ secrets/kubernetes . İo/ serviceaccount 
/ proc/acpi 
/ sys/firmware 

konsolda da volume ü görelim:

Q Search 
Name 
cw-cluster-my-nodes-Node 
cw-cluster-my-nodes-Node 
cw-cluster-dynamic-pvc-566e.. 
volume ID 
vol-00549b38ec5dee9c5 
vol-03cdff5d3a8563dd2 
vol-oca8aa3d9b06367f8 
vol-0699b8df827c46e94 
Type 
gp3 
gp3 
gp3 
gp2 
Size 
80 Gia 
8 Gi3 
80 Gia 
3 Gi3 

şimdi pod da node da silinse ebs durduğu sürece veriler elimizde duracak. volume un silinmesi için pvc nin silinmesi lazım.

objeleri silelim.

Ingress kısmına geçiyoruz:

Ingress çok kullanılan bir obje, tipleri var tipi belirleyip o tipin de ingress controller ını kuracağız.

sudo yum install git -y

mkdir repo && cd repo

git init

git remote add origin <origin-url>

git config core.sparseCheckout true

echo "subdirectory/under/repo/" >> .git/info/sparse-checkout # do not put the repository folder name in the beginning

git pull origin <branch-name>

bu komutlarla reponun tamamı değil, tek bir dosya çekebiliyoruz.

burada to-do app için web we db deployment ve service lerimiz var.

php-apachi deployment ve service imiz var. bütün bunlar bir internet sitesini oluşturuyor. bunlara önce loadbalancer service ile ulaşacaz sonra da ingress ile hepsine tek bir dns ile ulaşacaz.

v php-apache 
! php-apache.yaml 
v to-do 
! db-deployment.yaml 
! db-pvcyaml 
! db-service.yaml 
! web-deploymentyaml 
! web-service.yaml 
ingress-service.yaml 

önce to-do yu çalıştırıyoruz. web-service imizin tipi loadbalancer, provider desteğimiz var.

r@ip-172-31-29-123# k apply -f to-do 
ec2-use 
deployment . apps/db-deployment created 
persistentvolumeclaim/database-persistent-volume-claim created 
service/db-service created 
deployment . apps/web-deployment created 
service/web-service created 
$1 
ec2-user@ip-172-31-29-123# 

db mizin bir de pvc si var.

Load balancers (1) 
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic. 
Q Filter by property or value 
Name 
ac144c0f6c80f4445acc5 
61e5050ba7d 
DNS name 
0 aci44c0f6c80f4445acc56... 
vpc ID 
04bed5ee7e0d56bf3 

web-service için load balancer oluştuğunu aws konsolda da görüyoruz.

Ivenli deéil ac144c0f6c80f444Sacc561e5050ba7d-2125251386.us-east-1.elb.amazonaws.com:3000 
Course: AWS 13-TU... Google Takvim 13... O Your Repositories DeW3ps - Microsoft... (3) "devops" Searc... 
e; HacklnScience — 
Containers Todo Example 

uygulamayı service in lb dns i ile 3000 portundan görüyoruz.

php-apache yi çalıştıralım:

ec2-use 
deployment . apps/php-apache 
service/php-apache-service 
Ingress 
created 
created 
Ingress- 
kubectl apply 
-files : 
-f php-apache 
ec2-use 
rap-172-31-29-123* 

burada da bir loadbalancer service imiz var.

Load balancers (2) 
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic. 
Q Filter by property or value 
Name 
ac144c0f6c80f4445acc5 
61e5050ba7d 
aa277b72091bd4f73a38 
f9f3d4a2a243 
DNS name 
aci44c0f6c80f4445acc56... 
aa277b72091bd4f73a38f... 
State 
vpc ID 
04bed5ee7e0d56bf3 
04bed5ee7e0d56bf3 
c 
Availe 
3 Avai 
3 Avai 

konsolda bunu da gördük.

bunu da service in lb dns i ile 80 portundan görebiliriz. php-apache biraz yavaş gelir.

C A Güvenli cleéil aa277b72091bd4f73a38f9f3d4a2a243441706770.us-east-1.elb.amazonaws.com 
AWS Management... Course: AWS 13-TU... Google Takvim 13... O Your Repositories DeW3ps - Microsoft... 
OK! 

peki 20 tane service imiz olsa 20 tane load balancer mı açacaktık?

bunun için ingress var. ingress kullanmamız için controller kurmamız gerekiyor.

<https://kubernetes.io/docs/concepts/services-networking/ingress/>

<https://kubernetes.github.io/ingress-nginx/deploy/>

bu sayfada nasıl yükleneceği yazıyor, official sayfası.

konuyla ilgili blog:

<https://banzaicloud.com/blog/k8s-ingress/>

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.0/deploy/static/provider/cloud/deploy.yaml>

komutunu girelim.

r@ip-172-31-29-123# kubectl apply -f https://raw.githubusercontent.com/ 
ec2-use 
ss -nginx/ controller-vl. 7.0/ deploy/ static/ provider/ cloud/ deploy. yaml 
namespace/ingress-nginx created 
serviceaccount/ingress-nginx created 
serviceaccount/ingress-nginx-admission created 
role. rbac . authorization . k8s . io/ingress-nginx created 
role. rbac . authorization . k8s . io/ingress-nginx-admission created 
clusterrole. rbac . authorization . k8s . io/ingress-nginx created 
clusterrole. rbac . authorization . k8s . io/ingress-nginx-admission created 
rolebinding. rbac . authorization . k8s . io/ingress-nginx created 
rolebinding. rbac . authorization . k8s . io/ingress-nginx-admission created 
clusterrolebinding. rbac . authorization . k8s . io/ingress-nginx created 
clusterrolebinding. rbac . authorization . k8s . io/ingress-nginx-admission created 
configmap/ingress-nginx-controller created 
service/ingress-nginx-controller created 
service/ingress-nginx-controller-admission created 
deployment . apps/ingress-nginx-controller created 
job. batch/ingress-nginx-admission-create created 
job. batch/ingress-nginx-admission-patch created 
ingressclass . networking. k8s . io/nginx created 
validatingwebhookconfiguration . admissionregistration . k8s . io/ingress-nginx-admission created 
ec2-use 
ineress- 
-files : 

ingress-ngnix namespace i de oluşuyor:

ec2-use 
STATUS 
default 
Active 
Active 
Ingress-nglnx 
kube-node-lease 
Active 
kube-public 
Active 
kube-system 
Active 
ec2-use 
rap-172-31-29-123# 
k get ns 
AGE 
92m 
21s 
92m 
92m 
92m 
1 
ingress -files : $ 

bu ns deki podlar:

r@ip-172-31-29-123# k 
ec2-use 
READY 
ingress-nginx-admission-create-cbjrh 
ingress-nginx-admission -patch -zq45z 
ingress -nginx-controller- 7d9674b7cf-dxvv6 1/1 
$1 
ec2-user@ip-172-31-29-123# 
get po -n ingress-nginx 
STATUS 
Cunpleted 
Cunpleted 
Running 
RESTARTS 
1 
AGE 
51s 
51s 
51s 

şimdi ingress manifest ile kurabiliriz:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-service

annotations:

kubernetes.io/ingress.class: 'nginx'

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: web-service

port:

number: 3000

- path: /load

pathType: Prefix

backend:

service:

name: php-apache-service

port:

number: 80

tipler değişiyor, hangi tipi kullanıyorsak gidip onun docs tan bakıyoruz.

<https://kubernetes.io/docs/concepts/services-networking/ingress/>

burada hangi adress hangi service e yönlendirilecek onu yazıyoruz aslında.

nginx.ingress.kubernetes.io/rewrite-target: /

burada route path i tanımlıyoruz.

spec kısmında da rules altında kuralları belirtiyoruz. bunun için de pathler belirtiyoruz:

/ web-service

/load php-service gibi.

Examples 
/aaa/bb 
Kind 
p refi 
Exact 
Exact 
Exact 
Exact 
Prefix 
Prefix 
Prefix 
Prefix 
Prefix 
Prefix 
Prefix 
Path(s) 
/ foo 
/ foo 
/ foo 
/ foo/ 
/ foo 
/ foo/ 
/aaa/bbb 
/aaa/bbb/ 
/aaa/bbb 
/aaa/bbb 
Request path(s) 
(all paths) 
/ foo 
/ bar 
/ foo/ 
/ foo 
/ foo 
/ foo 
, / foo/ 
, /foo/ 
/ aaa/bbb 
/ aaa/bbb 
/ aaa/bbb 
/ aaa/bbb/ 
/ aaa/bbb/ccc 
Matches? 
Yes 
Yes 
Yes 
Yes 
Yes 
Yes, ignores trailing slash 
Yes, matches trailing slash 
Yes, matches subpath 

docs a baktığımızda pathType olarak prefix ve exact kullanabiliyoruz. exact yazıp koyuyorsak birebir o ifade yazılmadan o sayfaya gitmez. prefix te ise esnektir.

path ten sonra da hangi service gideceğini isim ve port ile belirtiyoruz.

şimdi dosyayı çalıştırabiliriz.

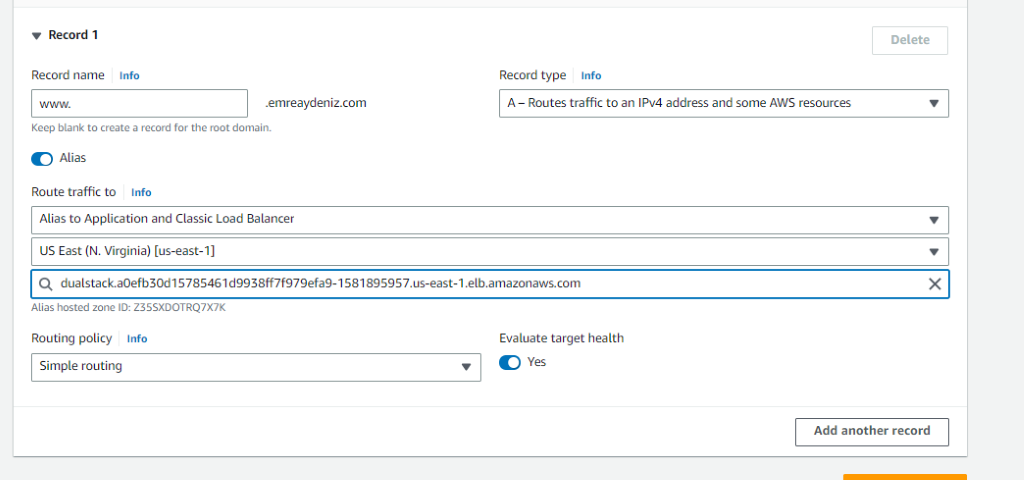
ec2-use 
AGE 
Ingress-service 
Ingress 
HOSTS 
CLASS 
<none> 
k get ing 
ADDRESS 
aøefb3Ød15785461d9938ff7f979efa9-1581895957. Us -east-I. elb. amazonaws . com 
POR 
80 

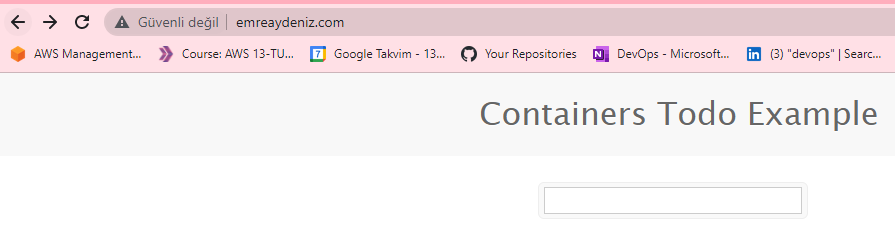
çalıştı ve lb oluştu.

konsolda da görüyoruz:

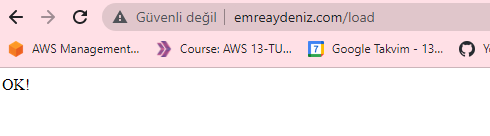
EQ > Load balancers 
Load balancers (3) 
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic. 
Q Filter by property or value 
Name 
ac144c0f6c80f4445acc5 
61e5050ba7d 
aa277b72091bd4f73a38 
f9f3d4a2a243 
aoefb30d15785461d993 
8ff7f979efa9 
DNS name 
aci44c0f6c80f4445acc56... 
aa277b72091bd4f73a38f... 
aoefb30d15785461d9938... 
State 
vpc ID 
04bed5ee7e0d56bf: 
04bed5ee7e0d56bf: 
04bed5ee7e0d56bf: 

route53 te record oluşturup domain name ile görebiliriz:





ya da lb nin dns ile de gidebilriz.

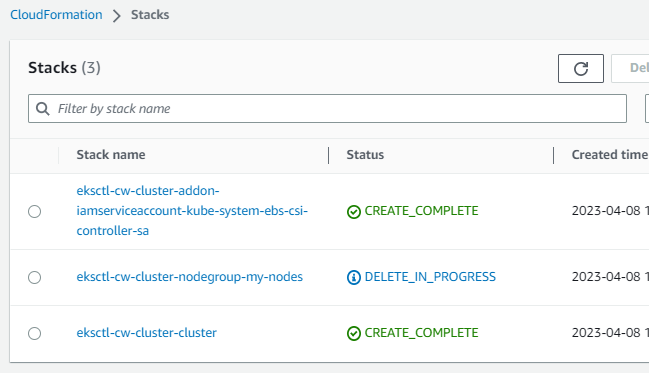


service tiplerimiz ClusterIP de olsaydı ingress sayesinde yine de service lere dışarıdan ulaşılabilirdi.

apply dediklerimizi delete dedikten sonra

eksctl delete cluster cw-cluster --region us-east-1

komutuyla cluster ı silelim



cloudformation dan süreci izleyebiliriz