

Û

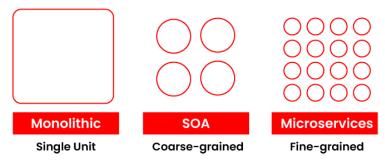
# kubernetes

- شرح ال Kubernetes بالكامل لكورس الباشمهندس Kubernetes الموجود علي قناه Codographia
  - رابط الكورس علي ال YouTube
  - https://www.youtube.com/watch?v=jTggu1HiKyY&list=PLX1b

    W GeBRhDCHijCrMO5F-oHg52rRBpl
    - رابط الكورس علي Udemy لعدم وجود بعض الفيديوهات
      <a href="https://www.udemy.com/course/kubernetes-step-by-step-in-arabic/?couponCode=ST3MT72524">https://www.udemy.com/course/kubernetes-step-by-step-in-arabic/?couponCode=ST3MT72524</a>
    - ال Slides المستخدمه في شرح الكورس https://drive.google.com/file/d/1nuxV8gTwJX xDexJgi113JeEpB3PEbr/view?usp=sharing
      - موجود بجانب كل عنوان الفيديو المستخدم في الشرح BY: Mohamed Atef Elbitawy
  - in https://www.linkedin.com/in/mohamedelbitawy/

# **Monolithic vs SOA vs Microservices**

#### **Monolithic Vs SOA Vs Microservices**



#### **Monolithic Architecture** •

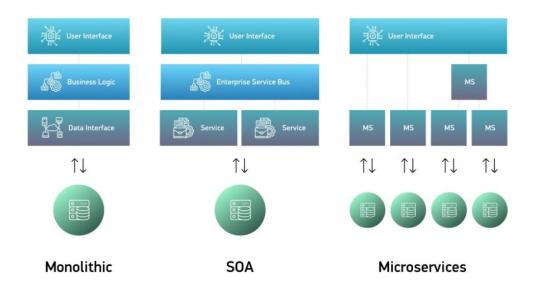
- هو عباره عن ال Traditional Model علشان ت design اي Software لفتره طويله و هو ببساطه عباره عن Single Piece و Single Piece و كل حاجه معتمده علي حاجه تانيه
- طالما ال Monolithic Architecture شغال علي Single Process ف ال Monolithic Architecture علي لو انا عايز اعمل Scaling ل features فيه بيكون مستحيل ف لازم ت Scaling علي Load Balancer علي Server
  - ال Monolithic ممكن نستخدمه لو احنا Team صغير او ال Project الخاص بينا صغير مش محتاج اي Architecture Pattern تانيه
    - خلال لما يكون فيه Patches او Update او Migrations ال Monolithic Applications بيكون ال Service بيكون ال Downtime

## **SOA Architecture** •

- ال SOA اختصار ل Service Oriented Architecture وده تحسين كبير لل Monolithic صغيره Architecture وال SOA متمحور حوالين ال Services وبيتم تقسيم ال Application ل Services صغيره بس كلها integrating وكلها بتخاطب بعض من خلال Set of API

## Microservices Architecture •

- عباره عن Services صغيره جدا بتقدملك ال Application بتاعك .وكل Services جوه النظام بتاعك هي عباره عن Services قائمه بذاتها
  - وكل Service عندها مجموعه من ال Source Code واللي بي managed بيكون Ismall بيكون development team
- وكل Services منها ممكن تعملها deploy بشكل مستقل هي Service قائمه بذاتها وال Team المسئول عن ال rebuilding ال Update من غير مايعمل Microservices او redeploying للابلكبشن كله
  - وال Services بي Communicate مع بعض باستخدام ال APIs وكل Service عن انتحاد عن التانيه الله Service عن التانيه



• هنلاقي في المثال اللي في الشكل ان في ال Monolithic فيه Data Interface وفيه Business Logic وفيه Business Logic وفيه Monolithic وفيه SOA فيه حاجه اسمها SOA . Interface لم النين او تلاته او اكتر وفي ال SOA فيه حاجه اسمها User Interface بيوصل لل Microservices . اما ال Microservices هنفصل كل ده ب Data Base بناعتها وال Code Base الخاص بيها وليها ال User Interface المسئول عنها وفي الاخر كل ده بيظهر ل User Interface

## **Containers** •

- ال Containers الحاويات هي طرق تركز على التطبيقات لتقديم تطبيقات عالية الأداء وقابلة للتطوير على أي Infrastructure و Mac على Linux بدون اي مشكله . وال المحاويات هي ممكن تشتغل علي Linux بدون اي مشكله . وال المحاويات المحاويات
- لو انت في شركه كبيره مثلا ف ممكن يكون عندك حوالي 2000 او 3000 من ال Containers ف علشان تقدر ت manage العدد الكبير ده من ال Containers Orchestrators محتاج حاجه اسمها

# **Containers Orchestrators** •

- ال Containers Orchestrators عباره عن tools بت على المحال على المحال على المحال عبياعك على المحال المحال

## **Kubernetes as a Service** •

فيه بعض الشركات العملاقه لل Cloud Providers بيقدموا حاجه اسمها Kubernetes as a Service (خي Azure Kubernetes Service (AKS) و Amazon Elastic Kubernetes Service (Amazon EKS) Google Kubernetes Engine(GKE)

in

# What is Kubernetes



# • ال Kubernetes عباره عن

Open Source System for automating deployment ,scaling ,and management of containerized applications

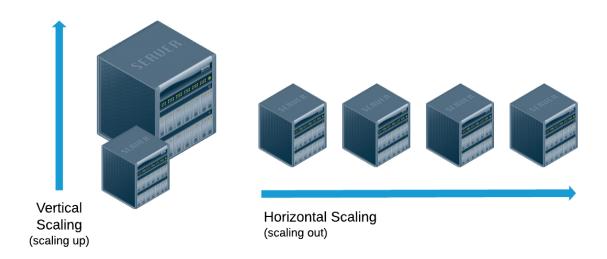
- Kubernetes هي كلمه مش انجليزي هي عباره عن كلمه يوناني ومعناها الشخص المسئول عن توجيه السفينه وتقدر تعتبر ال Kubernetes هو القائد اللي بيتحكم في سفينه فيها Containers
  - وبيطلق على ال Kubernetes دايما k8s وبتنطق Kate's علشان مابين حرف ال k وال s فيه 8 حروف
    - و Kubernetes اتكتب بلغه GO اللي طورتها Google

#### • مميزات ال Kubernetes

#### 1- اول ميزه وهي ال Scalability

- ان ال Kubernetes بيقدر يوفر لك ان انت تعمل Horizontal و Vertical Scaling بناءا علي ال reached بناءا علي ال threshold ليكون threshold يكون CPU
  - فيه عندي نو عين من ال Scaling في عندي ال Vertical Scaling وال Scaling في

  - ال Horizontal Scaling او ال scaling out ان بدل اما انت بتزود ال Horizontal Scaling بتاعتك هو بيضفلك Performance علشان تحسنلك ال New Nodes



#### Self-Healing リ -2

- ال Kubernetes بيقدر ي replace و reschedules ال Containers من ال failed node وبيعمل rules وبيعمل health check واي حاجه مش responsive هو بيتخلص منها بيعملها Kill وده based علي rules او Policy معينه علشان يمنع ال Traffic انها تروح لل Policy

#### Automated rollouts and rollbacks <sup>1</sup> -3

- ال Configuration بيقدر يعمل rolls out و rolls back و rolls out لل Application's health to prevent any ال monitoring وبي Configuration changes الله Configuration changes الله المحال المحين وعندك كذا replicaset او اكتر من MySQL او اكتر من Pods او اكتر من Pods الله والكتر من Pods الله والكتر من Pods شغالين ب Version قديمه وانت عايز تحط version تاني ف هو بيقدر انه يعمل rollbacks لكل ال Pods من غير مايحصل اي مشكله وبيعمل rollbacks في حاله لو حصل اي مشكله

#### Secret and configuration management 0 -4

ال Kubernetes بيقدر ي Manage بعض ال Sensitive Data اللي فيها name و Password بيقدر يعزلهم بعيد عن ال Container Images بحيث انها تكون محفوظه بشكل امن

#### Portability リー5

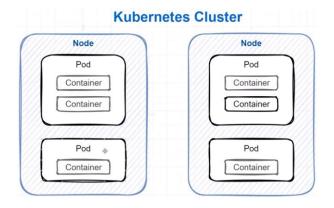
- ان ال Clusters اللي انت بتعمله بال Kubernetes يقدر يشتغل علي اي Linux Distributions او اي AWS وال كicloud Providers وال Virtual Machines وال AWS وال Aws وال Aws وال Google Cloud Platform وبيقدر كمان يشغل Azure

# **Kubernetes Architecture**

• ال Kubernetes بيتقسم لحاجتين ل Worker Nodes و Master Nodes

# **Worker Nodes**

- ◄ ال Worker Nodes بيوفر بيئه عمل لل Applications بتاعتك من خلال Worker Nodes بيوفر بيئه عمل لل Microservices وال Pod وال Pods وال Pods وبتكون مغلفه في حاجه اسمها Pods وال Pod ده عباره عن Container او اكتر من Container جواه ال Container جواه الحراد عن Pods بتاعك
  - ال Pod هي اصغر Deployable في ال Kubernetes وال Pod جواها ال Containers بتاعتك
  - يعني ال Pod ببساطه عباره عن Container او اكتر بيتشاركوا في نفس ال Storage وال Network وال Storage وال resources
- دي كده صوره مبسطه لل Kubernetes Cluster هتلاقیه جایبلك اكتر من Node وجوه كل Node بیكون فیه اكتر من Pod وجوه كل Pod بیكون فیه اكتر من Pod وجوه كل Pod ممكن یكون فیه



# **Worker Node Components**

- ال Worker Node بيتكون من ال Container Runtime وال Worker Node وال Monitoring and وال Dashboard Interface وال Proxy(kube-Proxy) وال Addons وال Loggins

#### Container Runtime -1

- علشان ال Pod يقدر ي run اكتر من Container الموجودين جواه لازم يكون متاح عنده Pod الموضوع ده Containers علشان يقدر ي run ال Containers ويعمل العمليات المختلفه علي ال Containers الموضوع ده بيتم من خلال حاجه اسمها (Container Runtime Interface) يعني Container Runtimes عاملين المعلين Kubernetes زي وسيط بيقدر يمكن Kubernetes ان هو يتعامل مع Container Runtimes ومن اشهر ال CRI-O في الد Container Runtime

## Node Agent – kubelet -2

- ال kubelet عباره عن Agent بيكون running في كل Node بيقدر ي Agent عباره عن Agent بيكون Rode بيكون Plane وهو ال Master Node وهو ال Component الرئيسي في ال Master Node وهو الله API Server الله بي Master Node كل حاجه ف بيستقبل ال Pod Definitions الله بي Master Node وهو الله

- بيتعامل مع ال Container runtime وبيقدر كمان ي Monitors ال health ال resources لل Container اللي مشغله ال Containers
- وال Connect علي ال Container runtime علي ال Container runtime سمه (Container Runtime اسمه CRI(Container Runtime Interface)

#### Proxy-Kube-Proxy -3

- عباره عن Network Agent بيكون شغال علي كل Node ومسئول عن التعديلات اللي هتحصل وصيانه كل الد Node الله Network Rules علي الله Node الله Network Rules في الله Node الله Proxy
  - يعني نقدر نقول ان ال Kube-Proxy مسئول عن ال TCP وال UDP وال

#### Addons -4

ال Addons عباره عن Cluster Features و بعض الاعمال مش Available في الاساس في ال Implemented في الاساس في ال Kubernetes في محتاجين حد خارجي سواء شركه او مجموعه developers بيقدرو ي Monitoring ال Addons ويعملولها bashboard وال Dashboard وال Logging

# **Master Node**

- ◄ ال Master Node هو ده العقل المدبر في ال Kubernetes وال Master Node بيوفر بيئه عمل ل ال Kubernetes اللي مسئوله عن التحكم في ال Control Plane
- علشان تقدر تتعامل مع ال Kubernetes Clusters ال request بيبعت request لل Control Plane الاول من خلال ال Control Plane او من خلال ال Control Plane او من خلال ال Control Plane او من خلال الحال الـ API
- علشان يحتفظ بال Cluster State لكل Cluster ال Configuration data بتتحفظ في حاجه اسمها etcd

# **Master Node Components**

• ال Master Node Components جواها ال Master Node Components وجواهم عدت

# API Server(kube-apiserver) -1

- ال API Server بي Intercepts RESTful calls ان احنا بيجلنا request او Calls من ال Users باستخدام ال Process باستخدام ال API Server انه ي API Server وي Validate في RESTfull في RESTfull من ال API Server دي ال etcd date store بيقدر يقرا تاني ال Kubernetes Clusters current state من ال
  - ف ال API Server هو اللي بيستقبل ال request اللي جايه لل Kubernetes ويبتدي يشوف انت عايز تعمل ايه ويعملهولك

## Scheduler (kube-scheduler) -2

ال Scheduler بي assigns Pods لل Assigns Pods ال Scheduler بيحدد انهي Node هو اللي مناسب لكل Pod بناءا علي بعض الشروط والقيود وال resources المتاحه يعني نقدر نقول ان ده اللي بينظم ال انهي Pod هيشتغل في انهي Node

#### Controller Manager (kube-controller-manager) -3

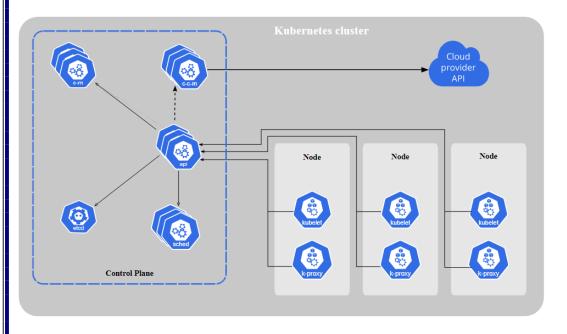
- في ال Kubernetes بيكون فيه حاجتين حاجه اسمها Current state و Desired State ف ال Controller Manager هو اللي مسئول ان هو يوصل من ال Current state من ال Desired State من ال Lurrent state من ال الحاله اللي مرغوب فيها ان انا اوصلها في ال Kubernetes يعني انا مثلا لو عندي pod موجود فيه Container وال Controller ده حصل ليه اي مشكله بقي failed ف ال manager هو اللي مسئول في الحاله دي انه يتصرف

#### Data Store(etcd) -4

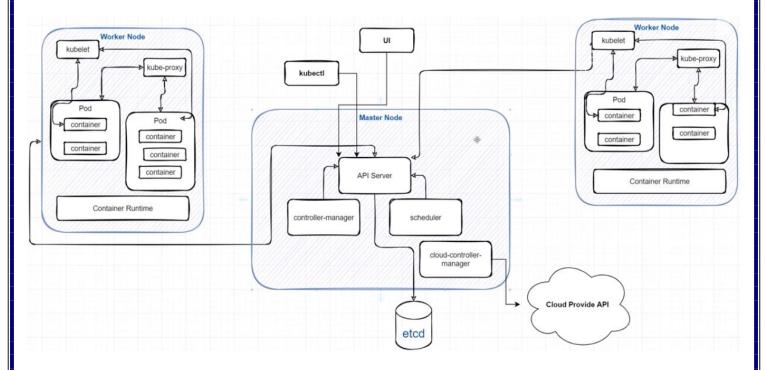
- ال database عباره عن database ال Kubernetes بيقدر يسجل حاله ال Cluster جواها وهي عباره عن distributed, reliable key-value database بيكون مناسب جدا لل distributed systems
  - ال API Server هو الوحيد اللي بيتعامل مع ال etcd data store من خلال ال API Server
  - وال etcdctl ده ال commands بتاعتك CLI Management Tool ف تقدر تعمل لل etcdctl بتاعتك commands و run المختلفه

#### **Kubectl -5**

- ال Kubernetes بتاع ال Command Line Interface باع ال Kubernetes







in

# **Kubernetes Concepts Explained**

#### Pod -1

- ال Pod عباره عن containers او اكتر من Containers جواهم ال Applications او ال pod عباره عن بتاعتي

#### Node -2

- ال Node عباره عن Worker Machine جوه ال Kubernetes ممكن تبقي Virtual او Control Plane اللي Machines علي حسب ال Cluster نفسه فين وكل Node بيتعمله manage بال Control Plane اللي موجود في ال Master Node وكل Node ممكن يكون جواها اكتر من Pod وال Node و Node و Node و Node

#### Cluster -3

- ال Cluster في ال Kubernetes عباره عن مجموعه من ال Nodes اللي بت Kubernetes عباره عن مجموعه من ال Pods اللي بت Pods او Application ومتوزع علي عدة Pods او Nodes ف ال Container بتقدر تRun ال Run
- ال Kubernetes Clusters بيقدر يخلي ال Containers تشتغل من خلال اكتر من Machines مش شرط تكون machines واحده بس لا هي تقدر تشتغل من خلال اكتر من Machines او اكتر من Physical وال Virtual وال Virtual وال

## Namespaces -4

- ال Namespaces عباره عن طريقه بنقدر نقسم ال Clusters بتاعتنا ل Virtual Sub-Clusters بتبقي مفيده لما يكون عندنا اكتر من Team شغالين على اكتر من مشروع بيشيرو ال Team

#### Service -5

- ال Service هي ال Logical Abstraction من ال Logical Abstraction يعني ال Application يعني ال Pods اصلا مش متصممه ان اي حد يقدر يعمل access عليها من بره يعني لو انا عملت Pods وال Service هو مش هيعرف ي Rods ال Pods بشكل Direct يعني احنا بنعمل Pods وال Pods نفسه Service عباره عن مجموعه من ال Pods وال Pods ال
- ال Pod دايما بتكون ephemeral هي حاجه مش ثابته هي حاجه متغيره بشكل كبير علي عكس ال Service ال Service بتكون ثابته
- ال Service بتخليك تشتغل مع group of pods وبتمدك ب Specific Functions وبيديك Pod الله ال Specific Functions مش من خلال ال مخصوص يقدر اليوزر ي access الابلكيشن من خلال ال Pod بشكل مباشر

#### **Deployment -6**

ال deployment ببساطه بنقول لل Kubernetes ازاي هيعمل Create او هيعمل Modify لل Pods اللي Replica اللي بداخلها ال scale من ال deployment من ال Number بتاعتي . ال deployment ممكن يعملو scale ل اي pods

#### Workloads -7

- ال Workload هي عباره عن ال Application اللي بي Run علي ال Workload سواء كان ال Single Component ه عباره عن ال Several او اكتر من Application شغالين مع بعض

#### Volume -8

- ال Volume هو شبه ال container volume ف ال Docker ف ال Docker علي كل ال Container علي كل ال الم Pods بغض النظر جواها Container او اكتر . وال Volume هيتعمله Pod لما ال Pod يحصله volume ممكن يكون عنده اكتر من volume موجودين معاه

#### ReplicaSet -9

ال ReplicaSet هو مصطلح في ال Kubernetes المقصود بيه ان هو يحافظ على عدد معين من النسخ المماثله لل Pods في حياه ال Pod وبيضمن اتاحه عدد معين من ال Identical Pods

#### Ingress -10

ال Ingress هو اللي بي manage لوحد عايز ي access ال Services من ال manage ال Ingress من ال Ingress ال load load ال Provide من بره فهو اللي بيقدر ي manage من خلال ال HTTP وال HTTPS وكمان بيقدر ي SSL Termination و balancing و balancing

#### **Orchestration -11**

ال Orchestration ان انت بتخلي ال effort اللي انت بتعمله بشكل manual علشان ت run ال Orchestration وبيعمل حاجه اسمها containers ف بي Manage ال lifecycle بتاعت ال containers و بيعمل حاجه اسمها Provisioning وبيعمل كمان كل الامور Down او Down وبيعمل كمان كل الامور Load balancing وال

**•** 

# Installing Minikube on Ubuntu

# خطوات تنزيل ال Minikube علي ال

sudo apt install -y curl wget apt-transport-https

wget

https://storage.googleapis.com/minikube/releases/latest/minikube-linu x-amd64

sudo cp minikube-linux-amd64 /usr/local/bin/minikube

sudo chmod +x /usr/local/bin/minikube

curl -LO

https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

minikube start --driver=docker

# **Basic Kubectl Commands**

## minikube start ل ال start هتسخدم start لو انا عايز اعمل

#### mohamed@MohamedAtef:~\$ minikube start

W0721 18:06:36.165587 24433 main.go:291] Unable to resolve the current Docker CLI context "default": context "default": context not found: open

/home/mohamed/.docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f/meta.json:

- e minikube v1.33.1 on Ubuntu 22.04
- \$\rightarrow\$ Using the docker driver based on existing profile
- 👍 Starting "minikube" primary control-plane node in "minikube" cluster
- Pulling base image v0.0.44 ...
- 🏃 Updating the running docker "minikube" container ...
- 📸 Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
- 🔎 Verifying Kubernetes components...
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
- Some dashboard features require the metrics-server addon. To enable all features please run minikube addons enable metrics-server
- 🗱 Enabled addons: storage-provisioner, default-storageclass, dashboard
- 🤣 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

#### → لو عايز اعرف ال status الخاصه ب ال minikube هستخدم status ك لو عايز اعرف ال

#### mohamed@MohamedAtef:~\$ minikube status

W0721 18:09:09.938569 28439 main.go:291] Unable to resolve the current Docker CLI context "default": context "default": context not found: open

/home/mohamed/.docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f/

minikube

type: Control Plane host: Running kubelet: Running apiserver: Running

✓ لو انا عايز اعرف ال Versions

mohamed@MohamedAtef:~\$ kubectl version

Client Version: v1.30.2

Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3

Server Version: v1.30.0

image هقوله kubectl create deployment وبعدين اسم ال kubectl create deployment

mohamed@MohamedAtef:~\$ kubectl create deployment --image=nginx nginx-app deployment.apps/nginx-app created

★ kubectl get deployments هقوله Create هقوله اللي اتعمله اللي اتعمله اللي اتعمله اللي العملة العملة العملة العملة العملة اللي العملة اللي العملة الع

nginx-app 1/1 1 1 28m

له لو انا عايز اعرف ال Pods اللي عندي هقوله kubectl get pods ك

mohamed@MohamedAtef:~\$ kubectl get pods

NAME READY STATUS RESTARTS AGE nginx-app-69999bf9b8-2zxnv 1/1 Running 0 30m

## لا عايز اعرف ال Services اللي عندي هقوله Services اللي عندي العرف ال

mohamed@MohamedAtef:~\$ kubectl get services

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP

◄ لو عايز اعرف ال replicaset الموجوده عندي

mohamed@MohamedAtef:~\$ kubectl get replicaset

NAME DESIRED CURRENT READY AGE nginx-app-69999bf9b8 1 1 1 1 41m

◄ لو انا عایز ازود ال scale بتاعي ل 3 هستخدم kubectl scale deployment وبعدین اسم ال image وبعدین استخدم اوبشن =replicas - وبعدین احدد انا عایز ازود ال scale ل کام

mohamed@MohamedAtef:~\$ kubectl scale deployment nginx-app --replicas=3 deployment.apps/nginx-app scaled

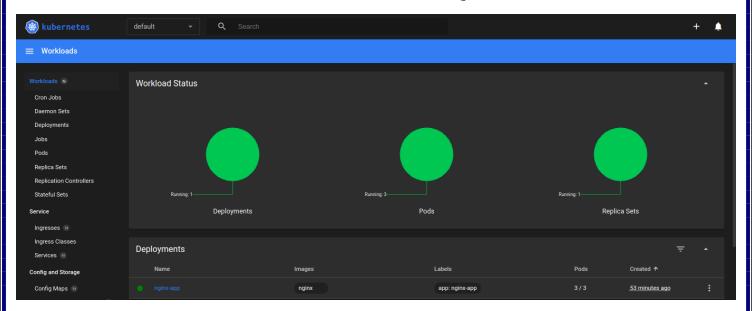
→ طب لو انا عايز اشوف كل ده في interface بدل اما بشوفه في ال CLI فيه عندي dashboard ال ommak الم يقدمها من خلال اني استخدم ال commad ده عيدمها من خلال اني استخدم ال manage ده الكلام اللي اتكلمنا عليه

#### mohamed@MohamedAtef:~\$ minikube dashboard

W0721 21:24:58.964294 71195 main.go:291] Unable to resolve the current Docker CLI context "default": /home/mohamed/.docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f/meta.json: no such file or directory

- Verifying dashboard health ...
- Launching proxy ...
- Verifying proxy health ...
- Opening http://127.0.0.1:45025/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...

◄ وده شكل ال Dashboard بعد اما يفتح



◄ لو عايز اعمل Delete لكل ال Deployment الموجوده عندي

mohamed@MohamedAtef:~\$ kubectl delete deployment --all deployment.apps "nginx-app" deleted

# **Writing Pods in Kubernetes**

- هنعمل Ymal File هيعمل •
- علشان نكتب ال Ymal File هنحتاج اننا نفتح ال File باي editor موجود عندنا وليكن هنستخدم ال Ymal File مركن الله Ymal File ممكن افتح ال Ymal File باستخدام ال CLI عن طريق اني اقوله Studio Code وبعدين اسم ال Ymal Studio Code واهم حاجه ان امتداد ال file يكون yml. هيفتحلي ال File باستخدام ال

mohamed@MohamedAtef:~\$ code helloworld-po.yml

لا kind هنكتبها في ال ymal File هنكتبها في ال

apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:latest
ports:
- containerPort: 80

◄ بعد اما كتبنا ال yml file هعمل apply لل yml file علشان اعمل apply هستخدم yml file وبعدين apply وبعدين اسم ال yml file ولازم اكون واقف في نفس ال Path اللي فيه ال yml file علشان اعمل apply

mohamed@MohamedAtef:~\$ kubectl apply -f helloworld-po.yml pod/nginx created

هنلاقيه هنا قالك انه عمل create لل pod اللي اسمها nginx

♦ هستخدم kubectl get pod علشان اتاكد هل فعلا اتعمل create لل ولا لا

mohamed@MohamedAtef:-\$ kubectl get pod

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 15m

هنلاقيه هنا قالك ان فعلا عندي Pod اسمها nginx وبقاله 15 ثانيه معمول ليها Create

◄ وممكن كمان باستخدم نفس ال command اني احددله pod معين اني اقدر اعمل select واقدر اعمل عليه operation

mohamed@MohamedAtef:~\$ kubectl get pod nginx

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 15m

➤ من اهم ال command اللي لازم تكون عارفها هي kubectl describe pod وبعدين اديله اسم ال pod ال من اهم ال container وال الله عن انه بيجبلي وصف عن كل حاجه تخص ال pod دي من حيث ال Image وال وال الله وال الله والـ الـ الله والـ الله والـ الله والـ الله والـ الله والـ الله والـ ال

```
mohamed@MohamedAtef:~$ kubectl describe pod nginx
                                                                                                          هنلاقیه هنا مثلا قایلك اسم ال pod ایه
Name:
Namespace:
                default
            n
Priority:
Service Account: default
            minikube/192.168.49.2
Node:
                                                                                                                       واتعملها start امتى
              Sun, 21 Jul 2024 22:43:47 +0300
Start Time:
Labels:
Annotations:
               <none>
                                                                                  وجايبلك كمان ال status بتاعت ال pod هل هي Running ولالا
Status:
             Running
IP:
           10.244.0.23
IPs:
 IP: 10.244.0.23
Containers:
 nginx:
  Container ID: docker://118011f741fce5b1f478a9ae1c59f9410126a8c3fd0b945165559d37461a1a27
  Image ID:
               docker-pullable://nginx@sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
             80/TCP
  Port:
                                                                                            وجايبلك كمان معلومات عن ال Container من حيث
  Host Port:
               0/TCP
                                                                                            ال Container ID وايه هي ال Image وكمان
  State:
             Running
                                                                                            جايبلي ال Port اللي هو شغال عليه واشتغلت امتي
   Started:
              Sun, 21 Jul 2024 22:43:50 +0300
                                                                                                         وكل التفاصيل الخاصه بال Pod دي
  Ready:
  Restart Count: 0
  Environment: <none>
   /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-8pb28 (ro)
Conditions:
                   Status
 Type
 PodReadyToStartContainers True
 Initialized
                    True
 Ready
                    True
 ContainersReady
                        True
 PodScheduled
                       True
Volumes:
 kube-api-access-8pb28:
                  Projected (a volume that contains injected data from multiple sources)
  Type:
  TokenExpirationSeconds: 3607
  ConfigMapName:
                        kube-root-ca.crt
  ConfigMapOptional:
                        <nil>
  DownwardAPI:
                       true
QoS Class:
                     BestEffort
Node-Selectors:
Tolerations:
                     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
                                                             وهنا لو خت بالك هنلاقي ال Events وده بيكون عباره عن ان ال pod بيكون ليه life cycle
                                                             من اول ماشتغل من حيث من اول ماتعمله Scheduled وبعدين Pulling وبعدين Pulled
 Type Reason Age From
                                     Message
                                                                                                          وبعدين Created وبعدين
 Normal Scheduled 24m default-scheduler Successfully assigned default/nginx to minikube
                                      Pulling image "nginx:latest"
 Normal Pulling 24m kubelet
                                      Successfully pulled image "nginx:latest" in 1.66s (1.66s including waiting). Image size: 187599276 bytes.
 Normal Pulled
                  24m kubelet
 Normal Created 24m kubelet
                                       Created container nginx
 Normal Started 24m kubelet
                                      Started container nginx
```

```
mohamed@MohamedAtef:~$ kubectl get pods -o json
mohamed@MohamedAtef:~$ kubectl get pods -o json
      "apiVersion": "v1",
     "items":
                 "apiVersion": "v1",
                 "kind": "Pod",
                 "metadata": {
                       "annotations": {
                            "kubectl.kubernetes.io/last-applied-configuration":
"{\annotations}":{\nginx}",\namespace}":\default\"},\namespace":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}":{\nginx}",\namespace}":{\nginx}",\namespace}":{\nginx}":{\nginx}",\namespace}":{\nginx}":{\nginx}",\namespace}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}":{\nginx}
"image\":\"nginx:latest\",\"name\":\"nginx\",\"ports\":[{\"containerPort\":80}]}}}\n"
                      "creationTimestamp": "2024-07-21T19:43:47Z",
                      "name": "nginx",
                      "namespace": "default",
                      "resourceVersion": "30984",
                      "uid": "02eb3e81-d626-4e17-9428-76ba5b89b093"
                },
                 "spec": {
                       "containers": [
                                  "image": "nginx:latest",
                                 "imagePullPolicy": "Always",
                                 "name": "nginx",
                                  "ports": [
                                              "containerPort": 80,
                                              "protocol": "TCP"
                                  "resources": {},
                                  "terminationMessagePath": "/dev/termination-log",
                                 "terminationMessagePolicy": "File",
                                  "volumeMounts": [
                                             "mountPath": "/var/run/secrets/kubernetes.io/serviceaccount",
                                             "name": "kube-api-access-8pb28",
                                             "readOnly": true
                      "dnsPolicy": "ClusterFirst",
                      "enableServiceLinks": true,
                      "nodeName": "minikube",
                       "preemptionPolicy": "PreemptLowerPriority",
                       "priority": 0,
                      "restartPolicy": "Always",
                      "schedulerName": "default-scheduler",
                      "securityContext": {},
                       "serviceAccount": "default",
                       "serviceAccountName": "default",
                      "terminationGracePeriodSeconds": 30,
                      "tolerations":
```

◄ لو انا عايز اتعامل مع ال nginx ده واعمل فيه شويه commands هستخدم

mohamed@MohamedAtef:~\$ kubectl exec -it nginx -- /bin/bash root@nginx:/#

كده احنا عملنا connect علي ال nginx هنلاقيه كاتب #/.root@nginx واقدر اكتب commands عاديه

◄ بعد اما دخلنا علي ال nginx ممكن مثلا انفذ command علشان اجيب ال Linux Distribution اللي شغاله علي ال container ده ف اقوله cat /etc/os-release

root@nginx:/# cat /etc/os-release
PRETTY\_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION\_ID="12"
VERSION="12 (bookworm)"
VERSION\_CODENAME=bookworm
ID=debian
HOME\_URL="https://www.debian.org/"
SUPPORT\_URL="https://www.debian.org/support"
BUG\_REPORT\_URL="https://bugs.debian.org/"
root@nginx:/#

◄ ممكن كمان اقوله نفذلي curl http://localhost ال curl http://localhost بستخدمه في حاله لو انا معنديش
 document وعايز اعرض محتوي ال website في ال CLI ف لما انفذ ال command هيعرضلي ال nginx الخاصه بال html

root@nginx:/# curl http://localhost
PRETTY\_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION\_ID="12"
VERSION="12 (bookworm)"
VERSION\_CODENAME=bookworm
ID=debian
HOME\_URL="https://www.debian.org/"
SUPPORT\_URL="https://www.debian.org/support"
BUG\_REPORT\_URL="https://bugs.debian.org/"
root@nginx:/# curl http://localhost

◄ لو انا عايز اغير محتوي ال localhost واتاكد انها شغاله هقوله مثلا ان انا عايز اغير محتوي ال localhost له المحتوي الله المحتوي الله المحتوي الله echo السمه command الله المحتوي الله المحتوي الله المحتوي الله الله الله المحتوي الله الله الله الله الله الله المحتوي الله الله الله المحتوي المح

root@nginx:/# echo 'Hello World from Pod in Kubernetes' > /usr/share/nginx/html/index.html

index.html تاني هنالقيه عرضلي ال message اللي انا كتبتها في ال curl كالمنا الله الله عرضلي الله عر

root@nginx:/# curl http://localhost Hello World from Pod in Kubernetes

exit هقوله nginx لو انا عايز اخرج بره ال

root@nginx:/# exit

mohamed@MohamedAtef:~\$

# لو انا عايز اعرض ال logs اللي بتاعت ال nginx هستخدم logs اللي بتاعت ال

mohamed@MohamedAtef:~\$ kubectl logs nginx /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/ /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh /docker-entrypoint.sh: Configuration complete; ready for start up 2024/07/21 19:43:50 [notice] 1#1: using the "epoll" event method 2024/07/21 19:43:50 [notice] 1#1: nginx/1.27.0 2024/07/21 19:43:50 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14) 2024/07/21 19:43:50 [notice] 1#1: OS: Linux 6.5.0-35-generic 2024/07/21 19:43:50 [notice] 1#1: getrlimit(RLIMIT\_NOFILE): 1048576:1048576 2024/07/21 19:43:50 [notice] 1#1: start worker process 30 2024/07/21 19:43:50 [notice] 1#1: start worker process 31 2024/07/21 19:43:50 [notice] 1#1: start worker process 32 127.0.0.1 - - [21/Jul/2024:21:05:47 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.88.1" "-" 127.0.0.1 - - [21/Jul/2024:21:16:23 +0000] "GET / HTTP/1.1" 200 35 "-" "curl/7.88.1" "-"

◄ ممكن كمان ان انا اشوف ال Pods بتاعتي عن طريق ال dashboard زي ماشوف قبل كده عن طريق ان أنا هستخدم minikube dashboard وهو هيفتحلي ال dashboard علي الو او ممكن استخدم minikube dashboard وهو هيديني ال url اللي هستخدمه علشان افتح ال dashboard عن طريق ان انا هاخد ال url ده واحطه في ال Browser او انبي اضغط على CTRL واضغط على ال url

#### mohamed@MohamedAtef:~\$ minikube dashboard --url

W0722 00:24:21.576189 166554 main.go:291] Unable to resolve the current Docker CLI context "default": context "default": context not found: open

/home/mohamed/.docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f/meta.json: no such file or directory

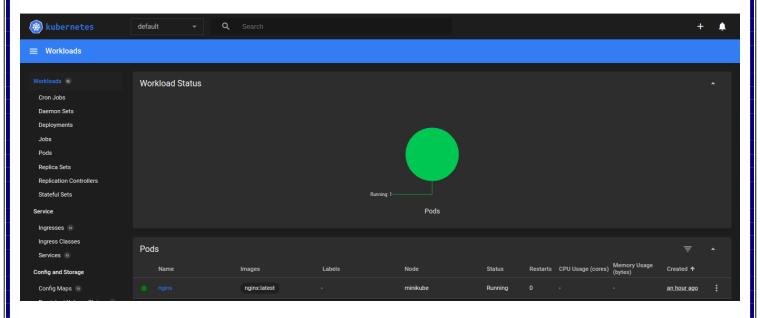
Verifying dashboard health ...

ده ال URL اللي هتسخدمه علشان افتح ال dashboard

- 📝 Launching proxy ...
- Verifying proxy health ...

http://127.0.0.1:34133/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/

image وال nginx واحده بس اسمها dashboard فتح معانا هنلاحظ ان ان احنا معندناش غير pod واحده بس اسمها nginx وال Running والديها بتاعتها اسمها nginx:اatest ومعمول ليها واللها المعانية المعانية



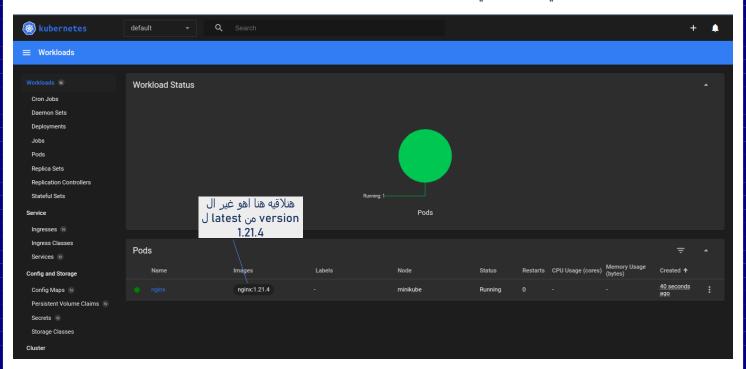
✓ طب افرض لو انا عدلت في ال version بتاعت ال nginx اللي عندي بدل اما هي latest نخلياه مثلا 1.21.4

apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- name: nginx
image: nginx:1.21.4
ports:
- containerPort: 80

◄ وجينا عملنا تاني build لل yml file هنلاحظ ان ان هو هيقولك ان ال pod اتعملها build طب ليه معمول create مظهرش انه اتعمله create زي مظهر اول مره واحنا بنعمل build علشان ال nginx موجوده ف هي معمول ليها create ف اي تغير بعد كده هيعمله configrured

mohamed@MohamedAtef:~\$ kubectl apply -f helloworld-po.yml pod/nginx configured

dashboard التغيير اللي عملناه على ال



✓ لو انا عايز اعمل delete اللي اسمه nginx هستخدم

mohamed@MohamedAtef:~\$ kubectl delete pod nginx pod "nginx" deleted

# Labels and Selectors in Kubernetes

• هنتكلم عن ال Labels وال Selectors في ال Kubernetes احد اهم المفاهيم في ال Labels هي ال Labels وعمليه Kubernetes وال Selectors لان هتلاقي ان هي بتسهل عليك عمليه تمييز كل object في ال Selectors وعمليه ال Selection ان انت تقدر تختار لما يكون عندك Pods و Nodes كتير

# Labels

- Labels are key/value pairs that are attached to objects, such as pods.
- Labels can be used to organize and to select subsets of objects.
- Labels can be attached to objects at creation time and subsequently added and modified at any time.
- Each object can have a set of key/value labels defined.
- Each key must be unique for a given object.

	-
key	value
firstName	Bugs
lastName	Bunny
location	Earth

- ال Labels عباره عن bject رى ال key value pairs بتكون bject لأى bject زى ال
- ال key value pairs زي داتا بيز صغيره هي مش داتا بيز حقيقيه بس تقدر تسجل فيها Key و value ال key ال بيبقي عباره عن اسم زي مثلا FirstName و Location و الله value و الدول الحطله value
  - بنقدر بال Labels ان احنا ننظم ال Object جوه ال Kubernetes لمجموعات فرعيه بنائا علي ال key وال value
  - ممكن ان احنا نحط ال Labels واحنا بن Create مثلا Pod في ال yml file او في ال Labels وحمكن ان احنا نحط ال Labels عادي بعد اما نعمل create لل pod
- ممكن كل object يكون ليه اكتر من Key/value pairs يعني مش شرط كل pod واحده يكون ليها label واحد بس ممكن تحط اكتر من واحد عادي على حسب انت عايز تميز هم ب ايه

in

• لازم ال Key یکون unique

🗸 هنشوف مثال عن ازاى نضيف labels ف انا عملت yml file ل create وضفت فيه

apiVersion: v1 kind: Pod metadata: name: nginx spec:

containers:

- name: nginx

image: nginx:1.21.4

ports:

- containerPort: 80

nginx.yml لل apply بعد كده هعمل حمياً

mohamed@MohamedAtef:~\$ kubectl apply -f nginx.yml pod/nginx created

nginx اللي اسمها pod لل create هنلاقي فعلا اتعمل get pod اللي اسمها

mohamed@MohamedAtef:~\$ kubectl get pods
NAME READY STATUS RESTARTS AGE
nginx 1/1 Running 0 51s

➤ احنا لسه معملناش Labels بس لو انا عايز اشوف ال Labels ال assign لل pod ده هنلاقيه ان انضاف عندي خانه جديده اسمها LABELS ومكتوب none ان بيقولك ان مفيش اي Labels معمول ليه assign لل Pod ده

mohamed@MohamedAtef: \$ kubectl get pods --show-labels

NAME READY STATUS RESTARTS AGE LABELS

nginx 1/1 Running 0 4m20s <none>

pod عن طريق ان احنا هنستخدم ال command ده Label عن طريق ان احنا هنستخدم ال Label عن طريق ان اللي عايز اضيف ال label عن طريق ان اللي عايز اضيف ال label عن طريق ان اللي عايز اضيف ال value و يكن هقوله ان ال key هو ال value هو towner هو العلام و ال

mohamed@MohamedAtef: \$ kubectl label pod nginx owner=mohamedatef pod/nginx labeled

هنلاقيه هنا قايلك ان ال pod اللي اسمها nginx اتحطلها labeled

◄ لو جينا شوفنا ال label تاني بعد اما ضفنا هنالقيه ضافلي ال Lables

mohamed@MohamedAtef:~\$ kubectl get pods --show-labels
NAME READY STATUS RESTARTS AGE LABELS

nginx 1/1 Running 0 51m owner=mohamedatef

◄ وليكن مثلا انا عندي Labels كتيره وعايز اعمل Select لل Pods بس اللي ال owner بتاعها
 ◄ emohamedatef عن طريق ان هستخدم اوبشن selector--

mohamed@MohamedAtef:~\$ kubectl get pods --selector owner=mohamedatef

NAME READY STATUS RESTARTS AGE nginx 1/1 Running 0 63m

✓ وممكن كمان بدل اما استخدم اوبشن -selector استخدم اوبشن ا- و هيطلعلي نفس ال OutPut

in

mohamed@MohamedAtef:~\$ kubectl get pods -l owner=mohamedatef

NAME READY STATUS RESTARTS AGE nginx 1/1 Running 0 63m

◄ زي ماقولنا قبل كده ان ممكن احط اكتر من label ل ال pod الواحد ف لو جينا ضفنا ل نفس ال pod اللي عندي
 اللي اسمها nginx ضفنا ليها label تاني اسمه release=dev

mohamed@MohamedAtef:-\$ kubectl label pod nginx release=dev pod/nginx labeled

◄ لو جينا شوفنا ال LABELS تاني هنالقي عندي two lables واحد اسمه owner=mohamedatef والتاني اسمه release=dev

 ${\bf mohamed@MohamedAtef:} \ {\bf \$} \ {\bf kubectl} \ {\bf get} \ {\bf pods} \ {\bf --show-labels}$ 

NAME READY STATUS RESTARTS AGE LABELS

nginx 1/1 Running 0 73m owner=mohamedatef,release=dev

- 🔻 انا ضفت pod كمان اسمها nginx2 علشان الموضوع يكون مفهوم اكتر
  - لو عملنا get pods هنلاقي ان فيه عملنا عملنا

mohamed@MohamedAtef:~\$ kubectl get pods

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 87m

nginx2 1/1 Running 0 91s

owner=ahmedatef التانيه اللي اسمه nginx2 هنضيف ليها label اسمه pod التانيه اللي اسمه عنصيف لل

mohamed@MohamedAtef: \$ kubectl label pod nginx2 owner=ahmedatef pod/nginx2 labeled

◄ لو جينا شوفنا ال Labels تاني هنالقي ان انا عندي two pods وكل pod ليها ال LABELS الخاص بيها

mohamed@MohamedAtef: \$ kubectl get pods --show-labels
NAME READY STATUS RESTARTS AGE LABELS

nginx 1/1 Running 0 91m owner=mohamedatef,relrase=dev

nginx2 1/1 Running 0 5m21s owner=ahmedatef

◄ ممكن بقي زي معملنا قبل كده ان انا احددله ال label اللي انا عايزه باستخدام ال Selector وليكن انا عايز اعمل
 صمكن بقي زي معملنا قبل كده ان انا احددله ال label اللي اسمه owner=ahmedatef

mohamed@MohamedAtef:~\$ kubectl get pods --selector owner=ahmedatef

NAME READY STATUS RESTARTS AGE nginx2 1/1 Running 0 8m39s

→ ممكن كمان استخدم ال equal وال non equal بمعني ممكن اقوله مثلا عايزك تعمل select لل label ال ahmedatef بتاعهم مسمهموش ahmedatef عن طريق اني هستخدم =!

mohamed@MohamedAtef:~\$ kubectl get pods --selector owner!=ahmedatef

NAME READY STATUS RESTARTS AGE nginx 1/1 Running 0 98m

- للتوضيح فيه عندي طريقتين لل Selector
- 1- اول طریقه اسمها Equality-based requirement
- والطريقه دي اللي احنا استخدمناها دلوقتي ان انا استخدم ال equal اقوله مثلا owner = mohamedatef او استخدم ال non equal

environment = production tier != frontend

#### 2- تانی طریقه اسمها Set-based requirement

- زي مثلاً ان انا هقوله ان ال key هو ال environment وبعدين بين قوسين هضيف ال value الطريقه دي شبه ال in في الداتا بيز

environment in (production, qa) tier notin (frontend, backend) partition !partition

◄ هنعمل selector باستخدام الطريقه التانيه عن طريق اني هستخدم kubectl get pods --selector وبعدين المايين single quotes هضيف ال labels عن طريق اني هقوله مثلا الله key هو release وبعدين الله in وبعدين الله key الله غذي الله key الله (dev) إو انا عندي اكتر من label بنفس الله key بس الله value مختلفه هقوله مثلا (dev, prod) و هكذا

```
mohamed@MohamedAtef:-$ kubectl get pods --selector release in (dev)'
NAME READY STATUS RESTARTS AGE
nginx 1/1 Running 0 117m
```

```
mohamed@MohamedAtef: $ kubectl get pods --selector 'release in (dev), owner in (mohamedatef)'
NAME READY STATUS RESTARTS AGE
nginx 1/1 Running 0 121m
```

◄ وممكن كمان استخدم notin ان اعرضلي اي حاجه ماعدا الحاجه اللي هحددها زي مثلا

```
mohamed@MohamedAtef:~$ kubectl get pods --selector 'relrase notin (dev)'
NAME READY STATUS RESTARTS AGE
nginx2 1/1 Running 0 37m
```

﴾ طب لو انا عايز احط ال Labels دي من خلال ال Configuration file اللي هو ال yml file وليكن مثل ااانا عندي yml file اسمه webserver وعايز احط فيه labels ف علاشن احط ال Labels لازم اضيف الlabels تحت ال metadata باسم: labels وبعد كده هضيف كل ال Labels اللي انا عايز احطها

```
apiVersion: v1
kind: Pod
metadata:
name: webserver
labels:
owner: mohamedatef
webserver: nginx
country: EG
spec:
containers:
- name: nginx
image: nginx:1.21.4
ports:
- containerPort: 80
```

# webserver الل apply حنعمل 4

mohamed@MohamedAtef:~\$ kubectl apply -f webserver.yml

pod/webserver created

# ✓ لو جينا شوفنا ال Labels هنالقيه ضافلي كل ال Lables الخاصه بال Labels

mohamed@MohamedAtef:~\$ kubectl get pods --show-labels

NAME READY STATUS RESTARTS AGE LABELS

nginx 1/1 Running 0 136m owner=mohamedatef,relrase=dev

nginx2 1/1 Running 0 49m owner=ahmedatef

webserver 1/1 Running 0 63s country=EG,owner=mohamedatef,webserver=nginx

# 👃 ممكن كمان لو عملنا describe لل pod اللي اسمها webserver نقدؤ نشوف ال

mohamed@MohamedAtef:~\$ kubectl describe pod webserver

Name: webserver
Namespace: default
Priority: 0
Service Account: default

Node: minikube/192.168.49.2

Start Time: Mon, 22 Jul 2024 15:08:19 +0300

Labels: country=EG

owner=mohamedatef

webserver=nginx -

Annotations: <none>
Status: Running
IP: 10.244.0.31

IPs:

IP: 10.244.0.31 Containers: nginx:

Container ID: docker://8924f058052e210fb4ef9ff5c1ad7552209f35d2b8c692ba90f5ccfa095c617c

Image: nginx:1.21.4

هنلاقي هنا ال Lables

# **Deployments in Kubernetes**

# Deployments

Deployments represent a set of multiple, identical Pods with no unique identities.

A Deployment runs multiple replicas of your application and automatically replaces any instances that fail or become unresponsive.

**Deployments** help ensure that one or more instances of your application are **available** to serve user requests.

In a deployment, you can describe the desired state for your application and Kubernetes will constantly check if this state is matched.

- ال Deployments بتمثل مجموعه من ال identical Pods اللي متطابقه بشكل 100% وال instance بتمثل مجموعه من ال replace علشان يقدر بشكل اوتوماتيك ي Pods اي replace علشان يقدر بشكل اوتوماتيك و fail او unresponsive
- تقدر تقول ان ال Deployments بتتاكد ان المجموعه من ال instance او ال Pods اللي انت مشغلهم هما دايما available انهم يستقبلو ال requests من ال user بشكل او توماتيك
- ال Deployments بيقدر يوصف مايسمي في ال Kubernetes بال desired state ايه الحاله اللي انا عايز اوصلها للابلكيشن ف بيفضل بشكل دايما يعمل check هل ال state دي احنا وصلناها و لا لا
- ≥ علشان اعمل Create لل Deployment هتسخدم Deployment هتسخدم لل Create وبعدين بحددله اسم ال container وليكن هسميه nginx وبعد كده بحددله ال Image اللي هجيب منها ال Pod او ال Pod او ال image=nginx وليكن مثلا ال image=nginx عن طريق اني هقوله image=nginx

mohamed@MohamedAtef:~\$ kubectl create deployment nginx --image=nginx deployment.apps/nginx created

Deployment لل create اللي عندي بعد اما عملنا pods لل pods

```
mohamed@MohamedAtef:~$ kubectl get all
NAME
                         READY STATUS RESTARTS AGE
pod/nginx-bf5d5cf98-6zdzz
                         1/1
                              Running
                                          0
                         TYPE
                                 CLUSTER-IP EXTERNAL-IP
                                                           PORT(S) AGE
service/kubernetes
                         ClusterIP
                                    10.96.0.1
                                                            443/TCP 8d
                         READY UP-TO-DATE AVAILABLE AGE
NAME
deployment.apps/nginx
NAME
                         DESIRED CURRENT READY AGE
replicaset.apps/nginx-bf5d5cf98 1 1
```

- ◄ هنلاحظ انه عندنا deployment المجاه عمل deployment وهنلاقي وهنلاقي واسمه nginx وهنلاقي المجاه واسمه nginx وحط كمان فيه عندنا replicaset المجاه وحط المجاه وحط المجاه وحط وحط وحط وحمل المجاه وحط وحمل المجاه والمجاه والمجاه وحمل المجاه والمحاه والمحاء
- ◄ ف ال command اللي احنا استخدمناه عمل create ل 3 حاجات عمل create لل pod و ال create و ال deployment و ال
  - ✓ ممكن استخدم ال describe لو انا عايز More details عن ال deployment اللي احنا لسه عاملين ليه create

#### mohamed@MohamedAtef:~\$ kubectl describe deployment nginx

Name: nginx Namespace: default

CreationTimestamp: Mon, 22 Jul 2024 16:53:17 +0300

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 1

Selector: app=ngin>

Replicas: 1 desired | 1 updated | 1 total | 1 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds:

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

◄ هشنوف ال pods اللي عندنا

هنلاقي هنا بعض المعلومات زي الاسم وال namespace

والوقت وال Labeles وال Selector وال Replicas وهكذا

#### mohamed@MohamedAtef:~\$ kubectl get pods

NAME READY STATUS RESTARTS AGE nginx-bf5d5cf98-6zdzz 1/1 Running 0 158m

◄ لو انا عايز ازود ال replicaset هستخدم replicaset هستخدم kubectl scale deployment وبعدين احددله اسم ال replica وبعدين استخدم اوبشن =replicas- وبعدين اضيف عدد ال

mohamed@MohamedAtef:~\$ kubectl scale deployment nginx --replicas=7 deployment.apps/nginx scaled

7 pods J create هنلاقیه عمل kubectl get pods لو عملنا تاني

mohamed@MohamedAtef:	\$ kubec	tl get pods			
NAME	READY	STATUS	RESTART:	S AGE	هنلاقیه هنا عمل create ل 7 pods وهنلاحظ ان اسم ال
nginx-bf5d5cf98- <mark>2j6cp</mark>	1/1	Running	0	81s	pods زي بعضه ماعدا الجزء الاخير من الارقام زي ماتولنا
nginx-bf5d5cf98- <mark>6zdzz</mark>	1/1	Running	0	172m	قبل کده
nginx-bf5d5cf98- <mark>c7rnf</mark>	1/1	Running	0	81s	
nginx-bf5d5cf98- <mark>kxtbp</mark>	1/1	Running	0	81s	
nginx-bf5d5cf98- <mark>qnctj</mark>	1/1	Running	0	81s	
nginx-bf5d5cf98- <mark>rpgdd</mark>	1/1	Running	0	81s	
nginx-bf5d5cf98- <mark>w8qkz</mark>	1/1	Running	0	81s	
ده الجزء الخاص بال deployment replicaset	بيتغير اعمل	ودہ الجزء اللم کل اما replicaset			

in

## ◄ لو انا عايز اعرف ال ١٦ بتاعت ال pods دي

```
mohamed@MohamedAtef:~$ kubectl get pods -o wide
                        READY STATUS
                                        RESTARTS AGE
                                                         ΙP
                                                                  NODE
                                                                           NOMINATED NODE READINESS GATES
nginx-bf5d5cf98-2j6cp
                         1/1
                              Running
                                                  9m58s 10.244.0.36 minikube <none>
                                                                                           <none>
                                           0
                                                          10.244.0.32 minikube <none>
nginx-bf5d5cf98-6zdzz
                         1/1
                              Running
                                                  3h1m
                                                                                           <none>
                                                  9m58s 10.244.0.37 minikube <none>
nginx-bf5d5cf98-c7rnf
                         1/1
                              Running
                                                                                           <none>
                                                  9m58s 10.244.0.34 minikube <none>
nginx-bf5d5cf98-kxtbp
                         1/1
                              Running
                                           0
                                                                                          <none>
                         1/1
                              Running
                                           0
                                                  9m58s 10.244.0.35 minikube <none>
nginx-bf5d5cf98-qnctj
                                                                                          <none>
nginx-bf5d5cf98-rpgdd
                         1/1
                              Running
                                           0
                                                  9m58s 10.244.0.38 minikube <none>
                                                                                           <none>
                                                  9m58s 10.244.0.33 minikube <none>
nginx-bf5d5cf98-w8qkz
                              Running
                                                                                           <none>
```

mohamed@MohamedAtef:~\$ kubectl scale deployment nginx --replicas=5 deployment.apps/nginx scaled

◄ لو عملنا تاني kubectl get pods هنالقيه قال ال pods ل 5

mohamed@MohamedAtef:~\$ kubectl get pods NAME READY STATUS RESTARTS AGE nginx-bf5d5cf98-6zdzz 1/1 Running 0 3h53m nginx-bf5d5cf98-c7rnf 1/1 Running 0 62m nginx-bf5d5cf98-kxtbp 1/1 Running 0 62m nginx-bf5d5cf98-qnctj 1/1 Running 0 62m nginx-bf5d5cf98-w8qkz 1/1 Running 0 62m

♦ لو عملنا describe لل deployment

هنلاقیه فی ال Replicas بیقولك ان فیه 5 desired

mohamed@MohamedAtef:-\$ kubectl describe deployment nginx

Name: nginx Namespace: default

CreationTimestamp: Mon, 22 Jul 2024 16:53:17 +0300

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 1

Selector: app=nginx

Replicas: 5 desired | 5 updated | 5 total | 5 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template: Labels: app=nginx Containers: nginx:

Image: nginx
Port: <none>

Conditions:

Type Status Reason

Progressing True NewReplicaSetAvailable Available True MinimumReplicasAvailable

OldReplicaSets: <none>

NewReplicaSet: nginx-bf5d5cf98 (5/5 replicas created)

Events:

Type Reason Age From Message

Normal ScalingReplicaSet 2m58s deployment-controller Scaled down replica set nginx-bf5d5cf98 to 5 from 7

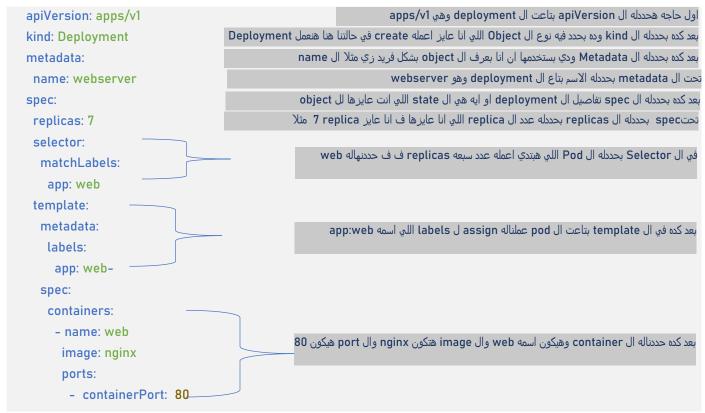
in

down من 7 ل 5

هنلاقيه هنا في ال Events بيقولك ان ال حصل scaled

# **Deployment Manifest File in YAML**

- ✓ هنشوف ازاي هنعمل ال configuration file الخاص بال deployment في ال YAML
- ◄ اول حاجه هعمل create ل ymal file باي اسم وليكن هسميه deployment-file ويكون امتداده yml. بعد كده هفتح ال file باي editor و هكتب فيه الاتي
  - ≥ اول حاجه هحددله ال apiVersion بتاعت ال deployment وهي apiVersion



deployment-file.yml لل apply هنعمل ∢

mohamed@MohamedAtef: \$ kubectl apply -f deployment-file.yml deployment.apps/webserver created

ReplicaSet عن طريق ال pods اللي عملنا ليها pods عن طريق ال

```
mohamed@MohamedAtef:~$ kubectl get pods
NAME
                                 READY STATUS RESTARTS AGE
                                                       0
webserver-587f6fb7c9-b2m72
                                    1/1
                                         Running
                                                             19m
webserver-587f6fb7c9-d4r6s
                                         Running
                                                       0
                                    1/1
                                                             19m
webserver-587f6fb7c9-hmwbt
                                    1/1
                                        Running
                                                       0
                                                             19m
webserver-587f6fb7c9-nf7kv
                                                      0
                                                             19m
                                    1/1
                                         Running
webserver-587f6fb7c9-tqf2i
                                    1/1
                                         Running
                                                             19m
webserver-587f6fb7c9-xnrkl
                                    1/1
                                         Running
                                                       0
                                                             19m
                                                      0
webserver-587f6fb7c9-zkvq6
                                    1/1
                                         Running
                                                             19m
```

◄ لو مثلاً في ال Kubernetes وقع منه pod او حصل اي مشكله في container فالي بيحصل انه بيقوملك واحد
 تاني ف احنا هنعمل ايه هنعمل delete لأحد ال pod ونشوف هيتعامل از اي

Pod لاي delete كا هنعمل ←

mohamed@MohamedAtef:-\$ kubectl delete pod webserver-587f6fb7c9-zkvq6 pod "webserver-587f6fb7c9-zkvq6" deleted

← لو عملنا list تاني لل pods هنلاقيه بيعمل create ل container و Pod مكان اللي عملنا ليه delete

NAME	READY	STATUS	RESTARTS	AGE
webserver-587f6fb7c9-b2m72	1/1	Running	0	40m
webserver-587f6fb7c9-d4r6s	1/1	Running	0	40m
webserver-587f6fb7c9-gqz5m	1/1	Running	0	103s
webserver-587f6fb7c9-hmwbt	1/1	Running	0	40m
webserver-587f6fb7c9-nf7kv	1/1	Running	0	40m
webserver-587f6fb7c9-qxcw2	0/1	ContainerCreating	0	4s
webserver-587f6fb7c9-tqf2j	1/1	Running	0	40m

◄ لو عملنا List تاني هنالقيه عمل create واقدر اني استخدم ال pod دي

NAME	READY	STATUS	RESTARTS	AGE
webserver-587f6fb7c9-b2m72	1/1	Running	0	40m
webserver-587f6fb7c9-d4r6s	1/1	Running	0	40m
webserver-587f6fb7c9-gqz5m	1/1	Running	0	103s
webserver-587f6fb7c9-hmwbt	1/1	Running	0	40m
webserver-587f6fb7c9-nf7kv	1/1	Running	0	40m
webserver-587f6fb7c9-qxcw2	0/1	Running	0	4s
webserver-587f6fb7c9-tqf2j	1/1	Running	0	40m

◄ لو انا عايز اعمل delete لكل ال Pods اللي عندي مره واحده

mohamed@MohamedAtef:~\$ kubectl delete deployment webserver deployment.apps "webserver" deleted

# Imperative and Declarative Configuration in Kubernetes



- هنتعرف على مصطلحين مهمين جدا وهما ال Imperative وال
- ال Imperative بكل بساطه هي عباره عن ال commands اللي بتكتبها في CLI زي مثلا kubectl وتديله مثلا commands وتديله مثلا commands زي ماحنا شوفنا قبل كده ال commands زي ماحنا شوفنا قبل كده ال run و command اللي احنا كتبناها واحنا بن create الله واحنا بن deployment كل ده اسمه Imperative ان انا بدي kubectl create deployment واضح وصريح لل API علشان يعملي الحاجه اللي انا عايزها وليكن في المثال ده deployment ان احنا بنعمل create الله واحنا بنعمل deployment لو انا بعمل عمل المقال ده المؤلفة والمؤلفة والمؤلفة وتدا المؤلفة وتدا ال
  - اما ال Declarative بكل بساطه هي عباره عن ال yml file ان احنا بنكتب Declarative ومن خلاله بحديله مثلا ال deployment وال name وهكذا
    - افضل طريقه اني استخدمها هي ال Declarative علشان ال

# Services in Kubernetes - Kubernetes

• ال Services عباره عن طريقه بنقدر ان احنا نعرض بيها ال Application للعالم الخارجي

# • ایه الفرق مابین ال Services وال Deployments

- ال Deployments هي طريقه بن launch بيها او بنشغل بيها او بنتاكد بيها ان ال Pods بتكون Running
- اما ال Service هي اللي Interface بين ال Pods وبين اي حد عايز ي Service ال application الموجود على ال على ال container على ال

# ✓ انواع ال Services في ال Services

- اول نوع منهم واللي شفناه قبل كده وده ال default value هو ال ClusterIP ودي معناها ان ال Pods بتكون متشافه بين ال Pods فقط زي ماحنا جربنا اننا ندخل علي ال Pods ونشوف ايه الابلكيشن او البروسيس اللي عليه وبياكد ان انت متقدرش تعمل requests لل pods من بره ال cluster اللي عليه يعني لو انت عندك مجموعه من ال Pods عليهم nginx او اي حاجه حتي لو ابلكيش انت اللي عامله و عايز ت لو انت عندك مجموعه من ال CluserIP هيبقي by default مش هيشتغل لان زي ماقولنا انك متقدرش تعمل requests
- تاني نوع وهو ال NodePord النوع ده بيخلي ال service accessible النوع ده بيخلي ال NodePord تقدر تستخدمها Cluster لازم تحددله port لازم تحددله port معين لازم تشتغل عليه معني كده ان ال Requests اللي جايه من خارج ال output بتقدر تت handle ويطلعلك output من جوه ال Pods من خلال ال NodePord
- تالت نوع وهو ال LoadBalancer و دي بتخلي ال LoadBalancer بشكل خارجي من خلال اي LoadBalancer عنده LoadBalancer زي ال GCP وال Azure وال Azure هي ال هتعمل Cloud Provider لل Request اللي بتجيله لل LoadBalancer اللي بتجيله لل Kubernetes Services
- و رابع نوع و هو ال **ExternalName** و ده انت مبتحددلهوش اي Port و endpoints هو بيستخدم Alias ل External services هو التحديد علشان يوجهها و ده شبه ال DNS
- فيه حاجه كمان اسمها Ingress دي مش service بس هي بتكون اقرب ل entry point لل Cluster بتاعك وممكن من خلال ال Ingress تقدر انت ت expose multiple Services على نفس ال IP

# **Apache and Nginx Services in Kubernetes**

- هنشوف ازاي هنعمل Apache و Nginx Services في ال Kubernetes
  - اول مثال هنشتغل علي ال Apache
- او حاجه هنعمل create ل Yaml file لل apache باي اسم انا هسميه apache-deployment واكتب جواه الاتي

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: apache-deployment
  app: webserver
spec:
 selector:
  matchLabels:
   app: webserver
 replicas: 2
 template:
  metadata:
   labels:
    app: webserver
  spec:
   containers:
   - name: apache
    image: bitnami/apache:latest
    ports:
    - containerPort: 80
```

apache-deployment الل apply بعد كده هعمل ♦

mohamed@MohamedAtef: \$ kubectl apply -f apache-deployment.yml deployment.apps/apache-deployment created

pods لل list لل pods

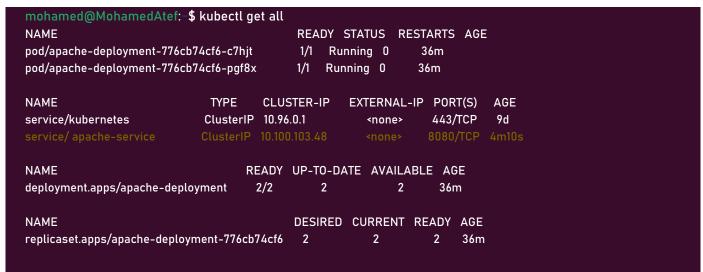
√

```
mohamed@MohamedAtef:~$ kubectl get all
                                         READY STATUS RESTARTS AGE
                                                                                   هنلاقیه عمل create ل two pods علشان احنا فی ال
pod/apache-deployment-776cb74cf6-c7hjt
                                                                   6m15s
                                         1/1
                                                Running
                                                                                   yml file احنا محددين ليه انه يعمل 2 من ال replicas
                                                                   6m15s
pod/apache-deployment-776cb74cf6-pgf8x
                                        1/1
                                                Running
                       TYPE
                               CLUSTER-IP EXTERNAL-IP PORT(S) AGE
                                                                          دي ال service بتاعت ال Kubernetes ونوعها ClusterIP
service/kubernetes
                     ClusterIP
                                 10.96.0.1
                                              <none>
                                                         443/TCP
                                                                          ري مشوفنا في الشرح وال service دي تببقي موجوده by default
                                        هنلاقیه کمان عمل create ل ال deployment باسم deployment
                                        2/2
deployment.apps/apache-deployment
                                                                                                      apache-deployment
NAME
                                          DESIRED CURRENT READY AGE
                                                                                       هنلاقیه کمان عمل create ل ال replicaset
replicaset.apps/apache-deployment-776cb74cf6
                                                                   6m15s
```

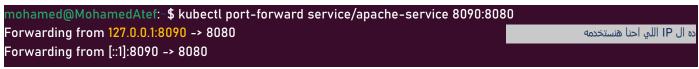
- ◄ هنعمل اللي احنا شوفناه ان احنا هن expose او نخلي ال Deployment دي ليها Service علشان نقدر ن apache ال Server ال Server ال علي المحافظة علي المحافظة علي المحافظة المحا
- ◄ ف علشان نعمل كده هنستخدم command طويل شويه ان انا هستخدم kubectl expose deployment وبعد كده بحددله الله name عن محددله اسم ال deployment دي و هي ال apache-deployment وبعد كده بحددله الله Port طريق اني استخدم =name-. وبعد كده بحددله ال Service عن طريق =type-. وبعد كده بحددله ال Target Port

mohamed@MohamedAtef: \$ kubectl expose deployment apache-deployment --name=apache-service --type=ClusterIP --port=8080 --target-port=8080

ع اسمها service اسمها service ل create اسمها List ك لو عملنا List الله عملاقي ان عمل عملنا عمل عملنا كانس



← فعلشان نعمل ا access لل server هنعمل Port Forward علشان نبتدى ن Test ال Service دي علي ال Localhost بتاعنا



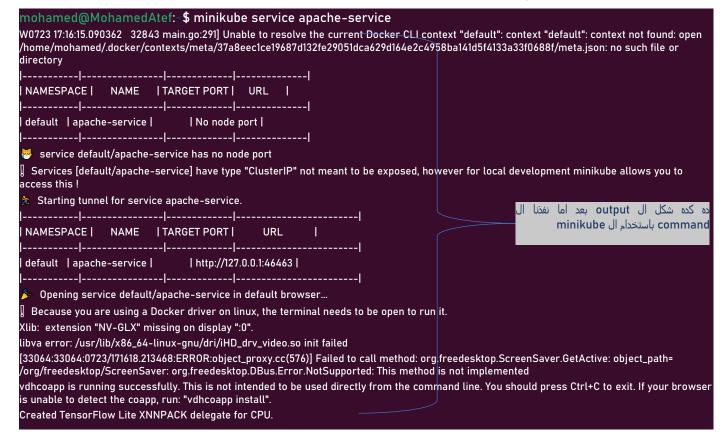
◄ كده المفروض انها اشتغلت اقدر استخدم ال IP ده 127.0.0.1 علي ال port ده 8090 علشان نشوف ال Service بتاعت ال Service اللي عملناها ف احنا هناخد ال IP ونفتح اي Browser ونحط ال IP ده هنالقيه اشتغل معانا عادى وقايلنا !It works



in

# It works!

◄ ممكن كمان اعمل access علي ال service عن طريق اننا نستخدم ال minikube ب اسمه service اسمه service وبعد كده بحددله ال service اللي انا عايز اعمل access عليها بعد اما ننفذ ال command هيطلعلي output بالشكل ده وهيفتح ل واحده ال Browser ويدخل علي ال service ده





# It works!

- تانى مثال هنشتغل على ال nginx
- ده كده ال yml file اللي هنشتغل عليه في حاله ال nginx هونفس ال code اللي استخدمناه في حاله ال metadata السي مع تغيير ال name في حاله ال metadata وال container وكمان غيرنا ال

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
labels:
  app: webserver
spec:
 selector:
  matchLabels:
   app: webserver
 replicas: 2
 template:
  metadata:
   labels:
    app: webserver
  spec:
   containers:
   - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

nginx الل apply خنعمل عالم

mohamed@MohamedAtef:-\$ kubectl apply -f nginx-deployment.yml deployment.apps/nginx-deployment created

◄ لو عملنا list على ال deployment هنالقيه بيقولنا ان فيه عندي ال nginx وال

```
mohamed@MohamedAtef:-$ kubectl get deployment

NAME READY UP-TO-DATE AVAILABLE AGE

apache-deployment 2/2 2 2 49m

nginx-deployment 2/2 2 2 66s
```

◄ بعد كده هنعمل لل nginx دي service بس مش هنحطلها اسم زي ماعملنا في ال apache كل اللي احنا هعمله ان احنا هعمله العنادة ويعدين نحددله ال deployment

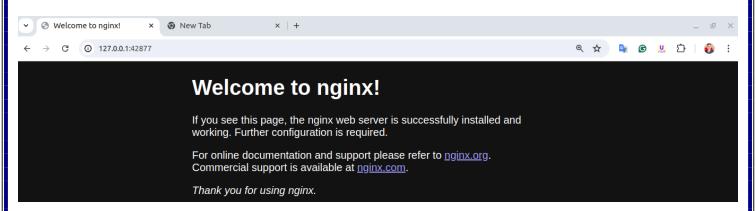
mohamed@MohamedAtef:~\$ kubectl expose deployment nginx-deployment service/nginx-deployment exposed

﴾ لو علمنا list لل services هنلاحظ ان أنا بقي عندي services و من ضمنهم ال nginx ولو خت بالك هتلاقي ان اسم ال service اسمها nginx-deployment طب ليه حط الاسم ده وانا اصلا وانا بعمل ال expose محددتلهوش اي اسم لانه by default لو انت محددتلهوش اسم هو هياخد نفس اسم ال Deployment

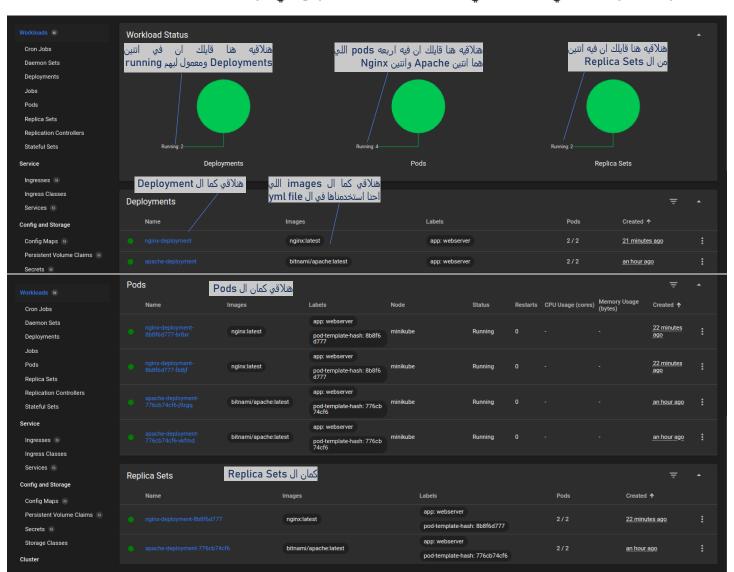
```
mohamed@MohamedAtef:~$ kubectl get services
                                 CLUSTER-IP
                                               EXTERNAL-IP PORT(S)
apache-service
                   ClusterIP
                               10.106.139.92
                                              <none>
                                                          8080/TCP
                                                                      56m
kubernetes
                   ClusterIP
                               10.96.0.1
                                             <none>
                                                          443/TCP
                                                                      9d
                               10.104.51.77
                                                                     2m30s
 iginx-deployment ClusterIP
                                                          80/TCP
                                             <none>
```

♦ ف هنشغل ال service دي عن طريق ال minikube وهيفتح علي طول علي ال service ال nginx بتاعت ال nginx

mohamed@MohamedAtef:~\$ minikube service nginx-deployment



→ لو جينا شوفنا كل اللي عملنا ده علي ال dashboard عن طريق اني اقوله minikube dashboard

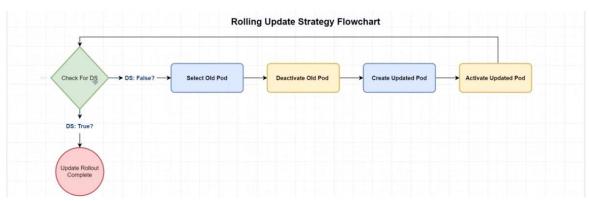


# **Deployment Strategies in Kubernetes**

- هنتكلم عن احد اهم المفاهيم المهمه جداا في ال Kubernetes وهي ال Deployment Strategies اللي هو ازاي تعمل Update ل بلكيشن شغال عندك من غير مايكون عندك Down Time كبير او ان يحصل انقطاع للخدمه بشكل ملحوظ للمستخدمين
- والهدف كمان من ال Deployment Strategies ان احنا نعمل automate لعمليه ال Deployment Strategies في ال ف بناءا علي ال Strategies اللي انت هتختارها ال Deployment هيقدر يعدل البرنامج بتاعك في ال action ان هو بيعمل action ان هو بيعمل new pods ل create اللي resources اللي محطوطه لكل pod
  - انواع ال Deployment Strategies

#### 1- اول نوع و هو ال Rolling Update Deployment

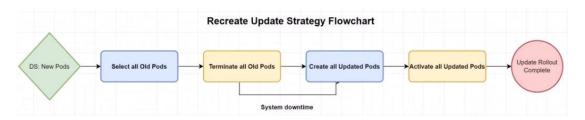
- ودي بتكون ال Default Deployment Strategies لان احنا زي ماكنا بنشوف واحنا بنعمل describe في mechanize وال Rolling Update وان هي Rolling Update وال الامثله اللي اخدناها كنا بنلاقيها موجوده جوه ال Replace وان هي Strategies اللي شغاله بيها ال Strategies دي ان هي بتعمل Pod علي حدا يعني بتاخد pod قديم بتعمله update للجديد وتروح على اللي بعد القديم تخليه جديد وهكذا يعني one by one
- الميزه ان احنا نعمل ال Rolling Update ان هو بيعمل مستوي pod by pod بيمسك pod pod علي حدا يعمل موضوع ال Update ليهم وباقي السيستم بيفضل active حتي لو شغال علي version قديمه
- ممكن يحصل minor performance reduction ان ال Performance يقل شويه لان انت خلال عمليه ال update بتخسر بعض ال pod لان فيه بعض ال pods بتقع علي ماتعمل Update
- فيه عندي مصطلحين مهمين جدا في ال Rolling Update Deployment وهما ال maxSure وال maxSure وال maxUnavailable
- ال maxSure ببساطه هي اللي بتحدد ال Maximum Number او Percentage من ال Pods فوق ال الله Percentage بتاعتك ان انت Specified number of replicas انت مثلاً محدد في ال Yaml File او في ال Specified number of replicas خليلي ال عندك 4 replicas في الله ممكن تقوله في ال maxSure لما يجي يعمل Rolling update خليلي ال عندك 25% ف يزودلك 25% من ال Pods او تقوله مثلا زودلي pod واحده او اتنين او تلاته و هكذا
- ال maxUnavailable ده ببساط بي declares ال Maximum Number او Percentage من ال Pods اللي بتكون unavailable غير متاحه في خلال عمليه ال Update



لا تنسونا من صالح الدعاء

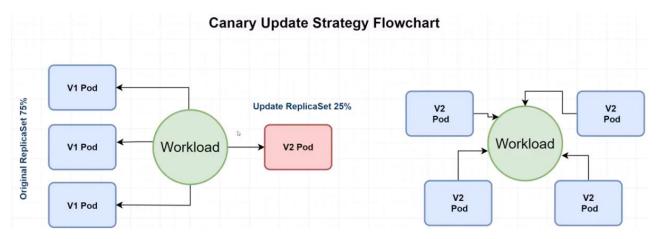
#### 2- تاني نوع و هو ال Recreate Update Deployment

- النوع ده بيعمل Shut down لكل ال pods القديمه وليكن عندك shut down وبعدين تعمل new pods لكل ال replace
- · ف ال Deployment بتعمل Select لكل ال outdated Pods اللي مش واخده ال Update الجديد وتخليها كلهم مره واحده deactivate وتروح ت create كل ال new pods مره واحده
  - النوع ده مناسب لل systems اللي مينفعش تشتغل بشكل جزئي زي البنوك



#### 3- تالت نوع و هو ال Canary Update Strategy

- وده عباره عن Partial update process الجديد من Version المنافعة ال
- ولو مفيش في الابلكيشن الجديد اي bugs يبقي كده كل حاجه تمام وبن scale up وبنطلع بال full rollout ولو فيه bugs هنرجع لل 25% او اين كانت النسبه كام نرجع لل Pods القديمه لحد اما ال Pods دي تكون Fixed

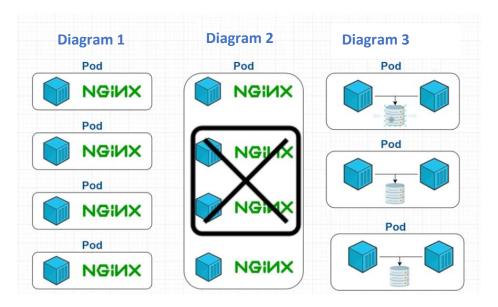


ده كده ال Flowchart الخاص بال Pod الخاص بال ReplicaSet هنلاقي عندنا ReplicaSet انها شغاله علي سيستم وفيه عندي 3 pods شغالين علي V1 من ال Pod وبنسبه 75% من ال Workload وعلي الناحيه التانيه بعض من ال Workload بيمثل %25 هو اللي شغال علي V2 من ال pod او ال Workload زي ماقولنا لو مفيش مشاكل بتبتدي كل ال Pods تروح علي ال V2 وتبقي كل ال workload شغاله V2 ولو حصل مشكله او حاجه نقدر ناخد النسبه اللي محددنها ونرجعها على ال Version القديمه لحد اما نحل المشكله

Mohamed Atef Elbitawy أنسونا من صالح الدعاء

# Multi-Container Pods and Containers Communication in Kubernetes

• هنشوف ازاي هنعمل create ل اكتر من container جوه ال Pod الواحد او بما يسمي بال Kubernetes • هنشوف ازاي هنعمل Pods



#### • اول Diagram

- ده بيمثل اللي احنا كنا بنعمله قبل كده باختلاف ال Images مش شرط تكون NGINX ان احنا بيكون عندنا Pod جواه container جواه NGINX مثلا او اكتر من Pod جواهم عدد واحد

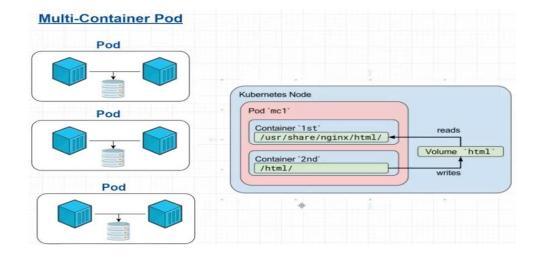
#### • تاني Diagram

- ده مفهومه غلط بمعني ان مش مقصود ان انت تعمل Pod واحد وجواه تعمل اكتر من container المفهموم ده غلط او تصور غلط زي مواضح في ال Diagram التاني

#### • تالت Diagram

Mohamed Atef Elbitawy

- ده المفهوم الصح لل Multi-Container Pod بمعني ان يكون عندك pod فيها containers مختلفه IP بمعني ان يكون عندك Storage ونفس ال Resources ونفس ال Address وهكذا.

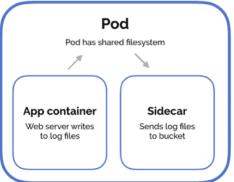


- هنلاقي فيه Area معمول ليها Shared مابين ال Two Container ف هنلاقي container رقم 1 وcontainer رقم 2 معمول ليها html وال html وال write وال write وال html اللي فيها html ف انا اقدر اكتب في ال html اللي اسمه html في ال container التاني وال container الاولاني يقرا من المكان اللي فيه ال html
  - ف هنشوف ازاي بي share نفس ال Storage بين ال Two Container جوه ال Pod الواحده

## **Design-Patterns of Multi Container Pod**

#### Sidecar Design Pattern -1

- ال Sidecar Design Pattern بيكون موجود فيه ال Main Application بيكون ليها مسئوليه مهمه بس مش وبيكون موجود معاه ال Helper Container بيكون ليها مسئوليه مهمه بس مش اساسيه علشان تشغل الابلكيشن نفسه يعني ممكن الابلكيشن يشتغل عادي مفيش اي مشكله بدون ال Sidecar Container بس هو مهم علشان بيعمل Function تانيه
- . يعني هو عباره عن Container زياده موجود في ال Pod بتقدر تحسن او تزود ال Functionality بتاعت ال Main Container

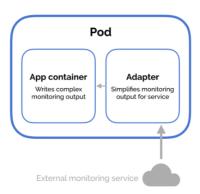


- ف ال Diagram ده بيقولك ال مكن يكون عندك ال log files مثلاً writes وبعدين يروح علي ال shared Area وبعدين يروح علي ال Sidecar المساعد ويبدأ ي Send ال Send في ال Sidecar

#### **Adapter Design Pattern -2**

- ال Adapter وظيفته ان هو ي connect ال Main Container ال Main Container الاساسي اللي عليه الابلكيشن بالعالم الخارجي . او علي سبيل المثال ال Helper Container يقدر يعمل re-routes اللي جايه من ال Main Container للعالم الخارجي لل External World يعني مثلا فيه main container باعت من ال request الخارجي و في ال Helper Container عشان تطلع للعالم الخارجي و في ده بيخلي ال request عشان تطلع للعالم الخارجي و في ده بيخلي ال connect external databases من غير اي main container ال
- ف نقدر نلخص ال adapter design pattern ان هو Container بي Transform ال output of the المخرجات اللي من ال container الاساسي لخارج ال Pod للعالم الخارجي

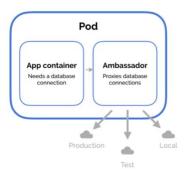
#### **Adapter**



#### **Ambassador Design Pattern -3**

- ال Ambassador بيستخدم علشان ي connect العالم الخارجي ف ال Container العالم الخارجي ف ال Ambassador بينت بيقدر بيعت Network Requests on behalf of the main application بينت الابلكيشن نفسه مش هو اللي بيبعت بيكون فيه Helper التابع ليه هو اللي بيبعت ال Requests نقدر نقول عليه ببساطه عباره عن Proxy بيخلي ال Container التانيه ت Connect علي ال Port علي ال Main Container
- · نقدر نلخص ال Ambassador ان هو container بيقدر ي Proxy ال Network Connection ل ال Main ل ال Network Connection Container

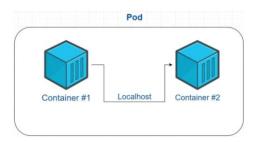
#### **Ambassador**



- لو شوفنا ال Diagram ده هنلاقیه بیقولك ان ال APP Container ده لو عایز Diagram هیروح لله طtabase connection علشان یقدر لله Helper Container علشان یعمل Proxy لله Ambassador علشان یقدر لله Local او Local او Production

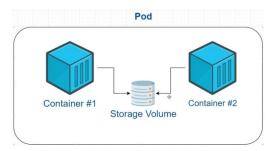
#### **Communication Inside a Multi Consider Pod**

• عايزين نفهم ازاي ال Communication بيحصل بين ال Containers جوه ال Pods ف فيه عندنا 3 طرق رئيسيه بيقدر يحصل عمليه الاتصال بين ال Containers جوه ال Pod وهما ال Shared Network و Shared Process Namespace و Shared Storage Volumes و Shared Network Namespace



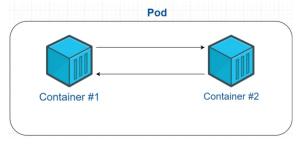
ال Shared Network Namespace ده عباره عن ان كل ال Containers الموجوده في ال Pod بتتشارك في نفس ال Network Namespace لذالك كل ال Container تقدر تتواصل مع بعض علي ال Network Namespace يعني مثلاً لو عندنا ال Container رقم واحد listing علي Port 8080 وال Container رقم 2 بي واحد container علي 8081 ال Localhost:8080 و يقدر 1 container 2 يكلم container 2 بان انت تكتبله Localhost:8080 او هكذا

#### **Shared Storage Volumes -2**



- ف ال **Shared Storage Volumes** بيكون كل ال Containers الموجوده جوه ال Pod بيقدرو يتشاركو في نفس ال Volume سواء ان هما يعملوا read او modify

#### **Shared Process Namespace -3**



- ال Shared Process Namespace ده عباره ان احنا بيكون عندنا Flag او Attribute اسمه Shared Process Namespace ده عباره ان احنا بيكون عندنا Flag او Shared Process Namespace في ال Pod specification في ال Pod specification في ال Container 1 عي communicate وهكذا والعكس

in

# Deploy Multi-Container Pod in Kubernetes

- هنشوف ازاي هنعمل Deploy ل Multi-container Pod في ال
- اللي احنا هنعمله هنعمل Pod جواه اتنين container ال container الاولاني هيكون Pod جواه اتنين Pod اللي اللي هيكون عباره عن بيطلع ال Default Page بس هنغير ال Default Page من خلال ال Linux Ubuntu هتغير ال Default Page الموجوده
  - ف علشان نعمل كده هنعمل create ل Yml file وليكن هنسمي ال file ده MultiContainerPod.yaml وهنكتب جوه ال file الاتي

apiVersion: v1

kind: Pod metadata:

name: two-containers-pod

spec:

restartPolicy: Never

volumes:

- name: shared-area

emptyDir: {}

containers:

- name: nginx-container

image: nginx volumeMounts:

- name: shared-area

mountPath: /usr/share/nginx/html

- name: ubuntu-container

image: ubuntu volumeMounts:

- name: shared-area
mountPath: /pod-area

command: ["/bin/sh"]

تاني container هيكون اسمه ubuntu-container وال image اسمها ubuntu وال volumeMounts بتاعته هتكون اسمها shared-area وال mountPath بيشاور على ال Path اللى اسمه pod-area/

بعد كده بحددله ال kind وده بحدد فيه نوع ال Object اللي انا عايز اعمله create في حالتنا هنا هنعمل Pod

هنيجي في ال spec هنلاقي فيه حاجه اسمها restartPolicy ومتحدده ب Never زي مانت شايف اي container في ال

Kubernetes بيحصله crash او بيوقف ل أي ُسبب من الاسباب بيحصله exit ال Kubernetes بيعمل create ل واحد تّاني من اول وجديد ف ده احنا مش عايزينه هنا ف عملنا ال restartPolicy ب Never

بعد كده ال Volumes ال Volumes اللي احنا استخدمناه في المثال ده اسمه emptyDir وده بيحصله Create اول اما ال Pod بيحصلها Create وطالما ال Pod بتكون Running ف ال emptyDir هتكون Running ولو

الُ Pod حصله crash او اتقفل لاي سبب ف ال content اللي هيبقي موجود جوه ال emptyDir مش هيتاثر لحد

اما انت تعمل delete لل Pod ف بالتالي ال emptyDir هيحصله delete معاه . ومحددله الاسم shared-area

بعد كده بحددله ال Container ان اول container عندي هو ال NGINX هيكون اسمه nginx-container وال

image هتكون nginx ومحتاج جوه ال container احددله ال volumeMounts بتاعته ف هقوله ال name الخاص بال volumeMount هو ال shared-area وال mountPath هيكون رايح علي ال Path ده

usr/share/nginx/html/ اللي هيكون جواه ال html page اللي اسمها

بعد كده بحددله ال Metadata ودي بستخدمها ان انا بعرف ال object بشكل فريد زي مثلا ال name

بعد كده كتبت اسم ال Pod اللي هعملها Create وانا سميتها Pod وانا سميتها

اول حاجه هحددله ال apiVersion بتاعت ال Pod وهي v1

بالنسبه لل command انا عايز اديله command جوه ال Terminal هنا ف هيبتدي يفتحلي ال Terminal وابتدي انقل ال Tommand ده index.html علي echo Hello from Codographia container في ال shared area اللي مابين ال Two Container اللي مابين ال احنا بستخدم ال command لما نكون عايزين ن run command علي ال container نفسه عايز مثلا افتح باش وهكذا . فممكن تشبه ال command انه زي ال Entry Point field في ال Docker

ال args بتستخدمُها أو آنت عايز تبعت Ārgumenْt أو فيه Parameters معاك رايحه تتنفذ علي command جوه ال Container . فممكن نشبه ال args انها زي ال CMD Filed في ال Docker

args: ["-c", "echo Hello from Codographia container > /pod-area/index.html"]

◄ كده احنا جهزنا ال yml file ووضحنا كل حاجه فيه بالتفصيل الخطوه اللي بعد كده ان احنا نعمله apply لما نعمل two containers هيظهرلنا ان عمل apply

mohamed@MohamedAtef:~\$ kubectl apply -f MultiContainerPod.yml pod/two-containers-pod created

two-containers-pod هنلاقیه ظاهرنا ان فیه Pods لل List لو عملنا

mohamed@MohamedAtef:~\$ kubectl get pod

NAME READY STATUS RESTARTS AGE two-containers-pod 1/2 NotReady 0 117s

Mohamed Atef Elbitawy لا تنسونا من صالح الدعاء

◄ عايز اشوف ايه ال output دلوقتي اللي جوه ال Nginx Server المفروض اننا غيرناه ل ال output دي nginx مش عاملين
 طي الله احنا مش عاملين Hello from Codographia container ف مش هينفع نفتح ال Nginx علي ال Browser بتاعت ال Browser ف اللي احنا هنعمله انه هنخليه يفتحلي ال Terminal بتاعت ال

🗡 ف لو جينا نفذنا ال command ده courl localhost علشان يعرضلي ال message

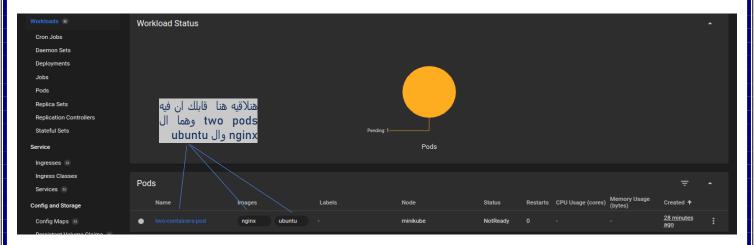
mohamed@MohamedAtef:~\$ kubectl exec -it two-containers-pod -c nginx-container -- /bin/bash root@two-containers-pod:/# curl localhost Hello from Codographia container التاني في ال Container

◄ لو انا عايز اعرف ال container الموجوده في ال Pod هتسخدم ال command ده

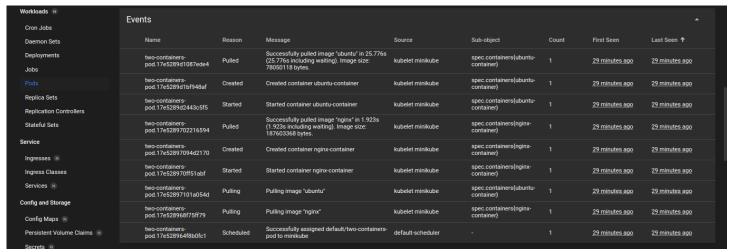
mohamed@MohamedAtef:~\$ kubectl get pods two-containers-pod -o jsonpath='{.spec.containers[\*].name}'

nginx-container واحد اسمه two container واحد اسمه two container واحد اسمه

◄ لو فتحنا ال Dashboard وشوفنا الكلام ده هنالقي



﴾ ولو دخلنا جوه ال two-containers-pod في ال Dashboard زي مظاهر قدامنا في ال صوره وروحنا علي ال Starte المراحل اللي عدا بيها ال pod من اول اما عمل Pulling لحد اما اعمل Starte



# What is Ingress in Kubernetes

- ذكرنا في الاول ان فيه 3 طرق ان احنا نقدر نوصل Services او نشغل Application جوه ال Pod وقولنا ان فيه طريقه رابعه مختلفه شويه وهي ال Ingress
- ال Ingress ده عباره عن API Object بي allow access بيسمحك للدخول علي ال API Object ده عباره عن Ingress مجموعه من ال من خارج ال Create وبتقدر ت configure ال access ده عن طريق ان انت بت Clusters جوه ال Kubernetes وبتعرف ال connection ازاي توصل لل Services جوه ال Clusters اللي عندي
- فيه عندي اكتر من طريقه ان احنا ن expose ال Applications اللي جوه ال Pods من خلال ال Services زي ال ClusterIP وال NodePort وال LoadBalancer
- تعال نفتكر ايه هو ال Services الله Services عباره عن frontend للابلكيشن بتاعك اللي موجود جوه ال Pods المتاحه Services الله عنه عمل Cluster الله Services المتاحه وبتعمل Distribution علي حسب ال Traffic زي ماشوفنا قبل كده ان احنا نقدر ن Distribution علي حسب الله Traffic عليهم اكتر من pod وتبدء توزع الله Traffic عليهم

Ingress

Incoming Request from Internet

HTTPS (Port 443)

Kubernetes Cluster

Ingress Controller

Services

Services

- ال Diagram ده بيوضحلك سريعا الIngress شغال ازاي
- انت عندك request جاي من خارج ال Cluster بتاعك من العالم الخارجي سواء كان بيشتغل علي HTTP علي port 443 او HTTPS علي HTTP علي port 443 علي مايسمي بال بيبتدء يدخل ال Kubernetes Clusters اللي عندنا علي مايسمي بال Ingress Controller وال Ingress Controller هو اللي يبدء يوزع علي حسب ال Rules اللي اتكلمنا عليها هيروح عند انهي Services اللي في الاخر وارها شويه Pods اللي عليها الـ Containers اللي عليها الـ Pods
  - بعني نقدر نقول ان ال Ingress مصنوع من حاجتين من ال API object و Ingress Controller
    - ال API Object هو عباره عن طريقه علشان نقدر ن expose بيها ال Services خارج ال Kubernetes Clusters
      - اما ال Ingress Controller هو المسئول عن عمليه ال Services نفسها بتاعت عمليه الدخول لل Services
- لو افترضنا ال Ingress ده computer ف تقدر تقول ان ال Ingress Controller هو ال ngress Controller هو ال المدير اللي بيعرف ال computer عباره عن المدير اللي بيعرف ال Programmer يعمل الحاجه دي ازاي اللي بيقول ال
- و و نقدر نعرف ال Ingress Controller هو Load balancer هو في الحقيقه هو مش Load Balancer بس هو بيقدر يقوم بالعمليه بتاعت ال Load Balancer هو بيعمل حاجات تانيه من ضمنها ال
  - ال Ingress Rules هي عباره عن

Set of rules for processing inbound HTTP traffic. An Ingress with no rules sends all traffic to a single default backend service

# **Applying Ingress in Kubernetes**

• اول حاجه هنعملها هنعمل create ل Deployment وهنعمله Scale وهن expose ال Service ونبتدي ازاي نبدء ن Create ال Ingress ونبتدي ازاي نبدء ن Create ال Ingress ونتعامل معاه

← ف هنعمل create ل Deployment اسمها nginxservice وال Image هتكون Nginx وهتكون Latest

mohamed@MohamedAtef:-\$ kubectl create deploy nginxservice --image=nginx:latest deployment.apps/nginxservice created

Scale بعد كده هنعمل ◄

mohamed@MohamedAtef:~\$ kubectl scale deploy nginxservice --replicas=4 deployment.apps/nginxservice scaled

```
mohamed@MohamedAtef:~$ kubectl get all
NAME
                                   READY STATUS RESTARTS AGE
pod/nginxservice-7999ff5dc4-lj9x6
                                          Running
                                                       0
pod/nginxservice-7999ff5dc4-llr8w
                                  1/1
                                         Running
                                                       0
                                                             8m43s
pod/nginxservice-7999ff5dc4-qxg2d
                                  1/1
                                                             5m38s
                                         Running
                                                       n
pod/nginxservice-7999ff5dc4-vkqrn
                                  1/1
                                                             5m38s
                                         Running
NAME
                            CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/kubernetes ClusterIP
                              10.96.0.1
                                                      443/TCP 10d
                                          <none>
                            READY UP-TO-DATE AVAILABLE AGE
deployment.apps/nginxservice
                                                          8m43s
                                   DESIRED CURRENT READY AGE
replicaset.apps/nginxservice-7999ff5dc4
```

expose Services علشان ن Create خ

mohamed@MohamedAtef:-\$ kubectl expose deploy nginxservice --port=80 --type=NodePort service/nginxservice exposed

◄ NodePort ونوعها nginx اسمها Services ونوعها services فيه services ونوعها الله عملنا عملنا

```
mohamed@MohamedAtef:~$ kubectl get services

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 10d
nginxservice NodePort 10.111.42.242 <none> 80:30541/TCP 2m14s
```

- 🗲 هنشوف از اي ن Create ال Ingress ده ونتعامل معاه ون create rule بسيطه
- > هنقوله kubectl create ingress وبعد كده بكتبله اسم ال Ingress واحنا هنسميه kubectl create ingress وبعدين احددله rule عن طريق اني اقوله =rule وبعد كده بقوله ايه هي ال rule دي ف هنقوله مابين احددله aginxservice:80" double quotes =/"

mohamed@MohamedAtef:-\$ kubectl create ingress nginxservice-ingess --rule="/=nginxservice:80" ingress.networking.k8s.io/nginxservice-ingess created

◄ لو عايز اشوف ال Ingress اللي عندي هستخدم

mohamed@MohamedAtef:-\$ kubectl get ingress

NAME CLASS HOSTS ADDRESS PORTS AGE nginxservice-ingess <none> \* 80 91s

◄ علشان ن access الابلكيشن اللي هو ال nginx من خلال ال Ingress اللي انا عملتله Create ف اول حاجه لازم نجبها هي ال IP بتاع ال Minikube

mohamed@MohamedAtef:~\$ minikube ip

192.168.49.2

minikube على ال SSH بعد كده هنعمل

mohamed@MohamedAtef:~\$ minikube ssh

docker@minikube:~\$

◄ هنحتاج اننا نحط في ال Hosts file جوه ال minikube ال IP بتاع ال minikube واحط جواه الاسم اللي هنادي عليه بعد كده

docker@minikube:~\$ sudo /bin/sh -c 'echo "192.168.49.2 nginxservice.io" >> /etc/hosts'

◄ لو شفنا محتوي ال file اللي اسمه hosts باستخدام cat command هنلاقیه آنه زود

docker@minikube:~\$ cat /etc/hosts

127.0.0.1 localhost

::1 localhost ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

102...i ipo-attiloues

ff02::2 ip6-allrouters

192.168.49.2 minikube

192.168.49.1 host.minikube.internal

192.168.49.2 control-plane.minikube.internal

192.168.49.2 nginxservice.io

nginx page هيجبلي ال curl nginxservice.io هيجبلي ال

docker@minikube:~\$ curl nxservice.io

◄ لو انا عايز اشوف ال Ingress هستخدم

mohamed@MohamedAtef:-\$ kubectl describe ingress nginx-service-ingress

➤ لو انا عايز اشوف ال Configuration file او ال Yaml file بتاع ال Ingress حتي لو احنا مكتبنهوش بستخدم

mohamed@MohamedAtef:~\$ kubectl edit ingress nginxservice-ingress

← لو انا عايز اعمل reset لل minikube cluster من غير منمسح ال minikube cluster نفسه

mohamed@MohamedAtef:~\$ kubectl delete all --all -n default

# Ingress Types, Ingress Rules and Ingress Path Types

#### Ingress Rules •

- ال Ingress Rules عباره عن HTTP Rule وال HTTP Rule بتحتوي علي المعلومات الاتيه بتحتوي علي ال Ingress Rules وال HTTP Inbound ممكن تختاره او لا لو مبتحددش ال HOST ده ال apply بت Rule لكل ال Option ممكن تختاره او لا لو مبتحددش ال Traffic اللي جايه علي ال Address اللي في الحاله بتاعتنا لو هنجرب بيكون ال minikube اما لو ال Yaml file ده Provided ان احنا محددنهوله في ال
- فيه list من المسارات زي مثلا ال testpath/ او علي حسب انت هتسمي ال Path ده ايه كل واحده منهم لازم تكون service.port.name و service.port.name و service.port.name متعرفه ب backend متعرفه ب Load Balancer اللي جياله قبل ما ال request اللي جياله قبل ما ال Services المناسبه لكل request علي ال Services علي ال Services المناسبه لكل request علي ال Services المناسبه لكل request علي ال Services علي ال Services المناسبة لكل request
  - ال backend بيكون عباره عن خليط او مزيج بين ال Services وال Port names

#### Path Types •

- كل Path انت هتحدده في ال Ingress لازم يكون ليه corresponding type Path لازم يكون ليه Ingress وال Exact وال الته وال الته وفيه 3 انواع من ال Path دي وهما ال Path وال Path وال Prefix
  - 1- ال Implementation Specific بيكون ال Path بيتعمله تطابق بناءا على ال Ingress Class
  - 2- ال Exact لازم ال URL Path اللي انت هتدهوله لازم ي matches ويكون URL Path
- يعني في الامثله اللي قدامنا دي في ال Exact لما اجي اقوله ان ال path بتاعي اللي انا معرفهوله في ال Ingress هو foo لما اديله في ال Request ال foo وانا ب request ال services دي بال URL ها هو matches وهيرجعلي ال Services ولكن لو هو مثلا ال Path ب foo وال Request مش هيعمل matches ومش هيرجعلي ال Services . حتي لو انا قولتله foo/ وفي ال Request قولتله foo/ وفي ال Request الله ومثل هيعمل matches لان في ال Request ال foo/ فيها / زياده عن اللي في ال Path
  - ف لازم ال Path وال Request يكونوا زي بعض

#### Examples

Kind	Path(s)	Request path(s)	Matches?
Prefix	/	(all paths)	Yes
Exact	/foo	/foo	Yes
Exact	/foo	/bar	No
Exact	/foo	/foo/	No
Exact	/foo/	/foo	No

in

1- ال Prefix بي matches based علي ال matches based يعني Prefix بيكون Prefix ومنفصل ب / وال element by element يعني element وبيتم علي مستوي ال case sensitive يعني لو جينا نبص علي الامثله اللي قدامناا لما انت في ال Prefix قولتله ألى في الله الله الله على كمان

Prefix	/foo	/foo, /foo/	Yes
Prefix	/foo/	/foo, /foo/	Yes
Prefix	/aaa/bb	/aaa/bbb	No
Prefix	/aaa/bbb	/aaa/bbb	Yes
Prefix	/aaa/bbb/	/aaa/bbb	Yes, ignores trailing slash
Prefix	/aaa/bbb	/aaa/bbb/	Yes, matches trailing slash
Prefix	/aaa/bbb	/aaa/bbb/ccc	Yes, matches subpath
Prefix	/aaa/bbb	/aaa/bbbxyz	No, does not match string prefix
Prefix	/ , /aaa	/aaa/ccc	Yes, matches /aaa prefix
Prefix	/,/aaa,/aaa/bbb	/aaa/bbb	Yes, matches /aaa/bbb prefix

## Ingress Class り・

- ال ingress Class بيحصلها ingress class من implementation وكل واحد من ال class اللي بتحتوي controllers وكل class اللي بتحتوي علي configuration زياده بخصوص ال Controller
- اللي موجود Ingress Controller اللي موجود by default هو ال الم انزلنا ال Ingress Controller هو ال

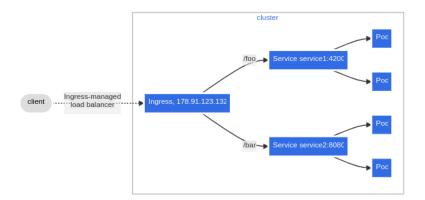
### IngressClass scope リ •

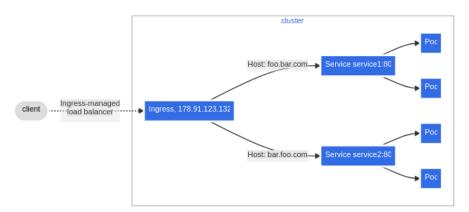
- انت ممكن تخلي ال Parameters بتاعتك متشافه علي مستوي ال cluster-wide او علي مستوي ال NameSpace

in

#### Types of Ingress •

- في عندي انواع من ال Ingress
- 1- اول نوع اسمه Single Service ان انت عندك Ingress بيروح يشاور علي Service واحده اين كان ال Service والم Service دي بتعمل ايه تديله مثلا اسم ال Service وال Port وبعدين تروح تشغلها سوا عملتلها Service ال minikube ال وروحت عملتلها on ال Service ال SSH بتاعت ال Service ال





# **Kubernetes Storage Architecture**

- ال Kubernetes Storage مبنيه علي ال Abstraction of volumes المقصود ب ال Kubernetes Storage المقصود ب ال Volumes بتعرف بال volumes تكون مجرده مش مرتبطه ب Pods او Nodes معينه وال Volumes بتعرف بال basic entities بتقدر ال Containers جوه ال Pods انها ت Storage الله Dods دي وتقدر انك تعملها basic entities بأختلاف ال NFS وممكن تبقي Local Storage ممكن تبقي AWS وممكن تبقي Cloud Storage وممكن تبقي AWS وكل كالله كالله
- ال Volumes في ال Kubernetes تقدر تتحكم فيهم بطريقتين اول طريقه وهي ال Ephemeral او ال -non persistent و الطريقه التانيه هي ال Persistent Storage

#### Non-Persistent Storage -1

بيكون by default ال Kubernetes Storage بتكون Temporary او اللي هي non-persistent .واي Storage بيتم تحديدها ك جزء من ال Container جوه ال Pod بتشتغل حيز من مساحه ال Host نفسه ودايما بيكون متاح طالما ال Pod موجود واول ما ال Pod يكون غير متاح ال Storage بتبقى Pod موجود واول ما ال Pod

#### Persistent Storage -2

- ال Support بي Support انواع كتيره جدا من ال Persistent Storage Models زي ال Files وال cloud services
- وال Storage تقدر تعملها Pod ولكن ال Pod ولكن ده بي violates وبيخترق مبدء ال Persistent و Persistent و Persistent و Pod ولكن ال Pod لازم تستخدم ال Pod بتاعت ال Pod ولكن ال Pod لازم تستخدم ال Pod بيستخدموا الحاجتين دول علشان يعرفوا ال Volumes Claims او زي ماهتشوفيه بعد كده بال PV و ال Storage requirements of their application المختلفه Storage
- ال Persistent Volumes(PV) عباره عن قطعه من ال Storage جوه ال Persistent Volumes عباره عن Polume و Volume بيوفرها وهي resource في ال Clusters زيها زي ال Node. وال عباره عن resource دي يعني بتتوصل وعندها lifecycle مستقله بشكل تام عن ال Pod اللي بيستخدم ال Persistent Volumes دي يعني احنا بنعمل object ل create وبنوصل ال Pod بال Pod بال Pod ده ولو ال Pod ده اتقفل او اتلغي مثلا ده مبيأثرش علي PV او ال Object بتاعنا
- ال Persistent Volumes Claims(PVC) عباره عن request عباره عن User لل Pv resources جاي من ال Pv وزي ما ال Pv بي Pv resources بتاعت ال Node ال Pv بي resources ال Pv بيعني زي ما ال request specific levels من ال request specific levels زي مثلا نقدر نحدد ال Pod ده هياخد قد ايه من ال Pod عدد وال ReadWriteMany وال ReadWriteOnce و ReadWriteOnce و ReadWriteMany وال ReadWriteMany و ReadwriteMany و ReadwriteMany
  - ◄ لو انا عايز شرح لل Volume ممكن استخدم ال command ده

mohamed@MohamedAtef:~\$ kubectl explain pod.spec.volumes

# Configure Pod to Use Volume for Storage

- هنشوف نموذج عملي عن ال Volume في ال Kubernetes وهنبدأ ن Pod كا Assign Volume علشان يشتغل كا Pod علشان يشتغل كا Storage لل Pod كا
  - هنعمل yaml file ل yaml file منه ل Pod منه ل create سمها ubuntu



in

Pod لل apply هنعمل ←

mohamed@MohamedAtef:~\$ kubectl apply -f vols-demo.yaml pod/vols-demo created

→ لو علمنا get لل Pod هنالقيه عمل get اسمها

mohamed@MohamedAtef:~\$ kubectl get pods
NAME READY STATUS RESTARTS AGE
vols-demo 2/2 Running 0 59s

#### pod ال describe لل pod ال describe

mohamed@MohamedAtef:~\$ kubectl describe pod vols-demo vols-demo Name: default Namespace: Priority: 0 Service Account: default هنلاقي هنا اهو اسم ال pod وهو -vols demo واتعمله creation امته وال minikube/192.168.49.2 Node: status بتاعه وال ip اللي شغال عليه Start Time: Mon, 29 Jul 2024 21:46:45 +0300 Labels: <none> Annotations: <none> Status: Running IP: 10.244.0.115 هنلاقي هنا تفاصيل عن ال containers ف هنا تفاصيل IPs: عن اول container وهو ubuntu1 IP: 10.244.0.115 Containers: ubuntu1: Container ID: docker://53030eef7f732f90b3dbbfd47e13a76f1a19919e2393b47871e0bc460e960d3a Image: Image ID: docker-pullable://ubuntu@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30 Port: <none> Host Port: <none> Command: sleep 3600 State: Started: Mon, 29 Jul 2024 21:46:48 +0300 هنا ال mounts بتاعت ubuntu1 موجوده في ubuntu1/ Ready: True Restart Count: 0 Environment: <none> Mounts: /ubuntu1 from vol (rw) /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-cvnhg (ro) ubuntu2: Container ID: docker://e558419cfa0499199d7458c6bbb872493ebe119839d294f3b18789e9ea9e8221 Image: Image ID: docker-pullable://ubuntu@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30 Port: Host Port: <none> هنا تفاصيل عن ال container التاني اللي اسمه ubuntu2 Command: sleep 3600 State: Running Started: Mon, 29 Jul 2024 21:46:50 +0300 Ready: True Restart Count: 0 ومعمول ليها mount تحت ubuntu2/ Environment: <none> Mounts: /ubuntu2 from vol (rw) /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-cvnhg (ro)

- ◄ هنعمل create ل folder مثلا جوه ال container الاولاني اللي اسمه buntu1 وهنالقيه ظهر في ال sharing التانيه بالظبط ف معنى ذلك ان هما sharing مع بعض
- → ف هنعمل create لملف اسمه NewFile جوه ال container اللي اسمه ubuntu1 ف استخدمنا ال create جوه ال container اللي اسمه vols-demo و بعدين استخدمت اوبشن c ده ان انا قولتله it استخدمت اوبشن عددتله اسم ال container وهو ubuntu1 وبعدين قولتله وبعدين اكتب ال علشان اكتب مددتله اسم ال container وهو command اللي انا عايز انفذذه ف علشان احنا هنعمل ملف استخدمنا command اسمه volume وبعدين حددنا ليه ال Path بتاع ال volume واسم الملف اللي هنعمله فنعمله والته اللي هنعمل ملف استخدمنا Path بناع الحديث الملف اللي الملف اللي هنعمله الملف اللي هنعمله و اللي هنعمله و اللي هنعمله و اللي الملف اللي هنعمله و اللي هنعمل ملف اللي هنعمله و اللي هندم و اللي و اللي هندم و اللي و اللي

mohamed@MohamedAtef:~\$ kubectl exec -it vols-demo -c ubuntu1 -- touch /ubuntu1/NewFile

◄ ممكن استخدم نفس ال command ف ان انا اعمل List واشوف الملف الى احنا عملناه

mohamed@MohamedAtef:~\$ kubectl exec -it vols-demo -c ubuntu1 -- ls /ubuntu1 NewFile

MohamedAtef ل file تاني باسم create حنعمل ل d create ل الني باسم

mohamed@MohamedAtef:-\$ kubectl exec -it vols-demo -c ubuntu1 -- touch /ubuntu1/MohamedAtef

✓ لو عملنا List تاني هنلاقي ال two files اللي احنا عملناهم

mohamed@MohamedAtef: \$ kubectl exec -it vols-demo -c ubuntu1 -- ls /ubuntu1 NewFile MohamedAtef

◄ لو احنا عملنا List علي ال Container التاني اللي اسمه ubuntu2 هنلاقي نفس ال two files اللي احنا عملنا Location ليهم create موجودين لانهم بيتشاروا في نفس ال Location

mohamed@MohamedAtef:~\$ kubectl exec -it vols-demo -c ubuntu2 -- ls /ubuntu2 NewFile MohamedAtef

# **Configuring Persistent Volumes in Kubernetes**

• احنا قولنا ان ال Persistent Volume(PV) عباره عن Storage بيكون موجود جوه ال Clusters ال Pod بيكون مستقل عن ال Pod بتاع السيستم هو اللي بيوفرلك مكان ال Storage ده وبيكون مستقل عن ال

# **Access Modes**

• عندي 4 انواع من ال Access Modes

#### ReadWriteOnce -1

- بيسمح ل Node واحده فقط هي اللي بت Access ال Volume اللي اتحدد انه read-write mode وكل ال Volume وكل ال Node في ال Node ده تقدر ت read وت write من ال Volume ده بس مش من خارج ال Pods

#### ReadWriteMany -2

ممكن Multiple Nodes تقدر ت Read وت Write لل Volume يعني لو عندك Node1 و Node2 فيقدروا يعملوا Access على ال Volume ده

#### ReadOnlyMany -3

- هنا ال Volume هيكون read-only بس Accessible من Volume

#### ReadWriteOncePod -4

- انت هنا بتدي ال Access ل Pod واحد فقط علي ال Volume ده

◄ هنشوف ال yaml file الخاص بال PV وازاي ن Create ال PV

◄ هنعمل yaml وهنسمیه مثلا pv.yaml وبعدین هنکتب فیه الاتي

kind: PersistentVolume

apiVersion: v1 metadata:

name: pv-storage

spec:
capacity:
storage: 2Gi
accessModes:
- ReadWriteOnce
hostPath:

path: "/tmp/data"

انا هنا حددت ال Kind هیکون PersistentVolume وبعدین حددنا ال version وهو v1

بعد كده تحت ال metadata حددت اسم ال Volume ف هسميه pv-storage

بعد كده في ال spec ضفنا حاجه جديده وهي ال capacity وبتيجي ان اناا ببدء احددله ال capacity وتحت ال capacity وبتيجي في ال storage انك تقوله انه ياخد one او two وال Gi مش معناها جيجا بايت لا معناها gibibytes

بعد كده بتحددله حاجه مهمه جدا وهي ال AccessModes واحنا محددينه هيكون ReadWriteOnce

بعد كده بحددله ال hostPath مكان ال Storage علي ال Host وده الPath بتاعه tmp/data/ وممكن تحددله اي Path انت عايزه

◄ هنعمل create ونشوف ال output بتاعه

mohamed@MohamedAtef: \$ kubectl create -f pv.yaml persistentvolume/pv-storage created

#### ◄ علشان اعرف ال PV الى عندي هستخدم

mohamed@MohamedAtef:~\$ kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE pv-storage 2Gi RWO Retain Available <unset> 3m25s

#### حن ال pv اللي عملنا ليه describe عن ال pv للي عملنا ليه

mohamed@MohamedAtef:~\$ kubectl describe pv pv-storage

Name: pv-storage
Labels: <none>
Annotations: <none>

Finalizers: [kubernetes.io/pv-protection]

StorageClass:

Status: Available

Claim:

Reclaim Policy: Retain
Access Modes: RW0
VolumeMode: Filesystem
Capacity: 2Gi
Node Affinity: <none>

Message: Source:

Type: HostPath (bare host directory volume)

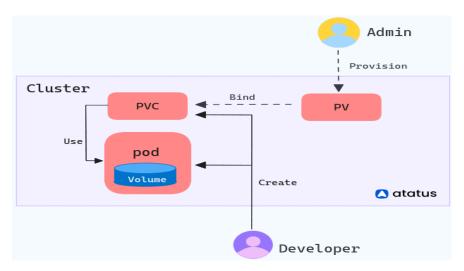
Path: /tmp/data

HostPathType:

Events: <none>

## **Configuring Persistent Volumes Claims in Kubernetes**

- احنا قولنا ال Pods المساحه ال Persistent Volumes Claims (PVC) هو عباره عن request المساحه ال User عايز يخدها وبيتم تخصصها لل Pods وال Pods بيقدر ي resources من ال resources تحديدا من ال CPU وال Memory وممن كمان يحدد specific size و Pods
- لو بصينا علي ال diagram ده هنلاحظ اللي بيحصل ان ال Administrator هو اللي بيوفرلك ال PV او المساحه من ال Storage علي ال Server وبيتعملها بطريقه ما Bind بال PVC وبيتم استخدام ال PVC بواسطه المساحه من ال Pod اللي ال Pod اللي ال developer عامله Create عامله



- PVC ال Create وازاي ن yaml file الخاص بال عالم yaml file وازاي المحاص بال عالم المحاص الحاص بال عالم المحاص ا
  - ♦ هنعمل yaml وهنسميه مثلا pvc.yaml وبعدين هنكتب فيه الاتي

kind: PersistentVolumeClaim

apiVersion: v1 metadata:

name: pvc-storage

spec:

accessModes:

- ReadWriteOnce

resources: requests: storage: 1Gi

هنا حددنا ال resources ان انا محتاج ال requests لل storage یکون 1Gi

PVC ال create كل PVC

mohamed@MohamedAtef: \$ kubectl create -f pvc.yaml persistentvolumeclaim/pvc-storage created

◄ هعمل List لل PVC هنلاقیه عمل create وال PVC وال pvc-storage وال status بتاعته اتعملها pvc-storage وال PVC وال PVC-9e5d3909-7939-4ddf-ae27-7f51c3097f3 اللي اتربط بیه هو ده Pvc-9e5d3909-7939-4ddf-ae27-7f51c3097f3 اللي احنا حددنهاله هي 1Gi وال PV بال Volume وال Volume اللي احنا حددنهاله هي 1Gi وال access modes اللي احنا حددناه و هو RWO وال storage class هيكون ال storage class لو انت محددتلهوش storage class في ال yaml file هو هيختاره اتوماتيك

mohamed@MohamedAtef:~\$ kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE pvc-storage Bound pvc-9e5d3909-7939-4ddf-ae27-7f5lc30977f3 1Gi RWO standard <unset> 112s

◄ لو احنا عملنا list لل Persistent Volume(PV) هنلاحظ ان هو عمل list لل الفروعوبي والمحنوب الفروعوبي الفروع الفروع

mohamed@MohamedAtef:~\$ kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE pv-storage 2Gi RWO Retain Available 
 v-storage

pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3 1Gi RWO Delete Bound default/pvc-storage standard <unset> 6h13m

pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f لل pv اللي اسمه pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f

mohamed@MohamedAtef:~\$ kubectl describe pv pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3

Name: pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3

Labels: <none

Annotations: hostPathProvisionerIdentity: a5f809eb-d142-4d2d-b096-a5c4119cdd20

pv.kubernetes.io/provisioned-by: k8s.io/minikube-hostpath

Finalizers: [kubernetes.io/pv-protection]

StorageClass: standard Status: Bound

Claim: default/pvc-storage

Reclaim Policy: Delete
Access Modes: RWO
VolumeMode: Filesystem
Capacity: 1Gi
Node Affinity: <none>

Message:

Source:
Type: HostPath (bare host directory volume)

Path: /tmp/hostpath-provisioner/default/pvc-storage

HostPathType: Events: <none> هنلاحظ ان في الPath هنلاقي مسار /tmp/hostpath-provisioner/default/pvc-storage هو جايب ال Path ده منين اللي حصل ان ال minikube عامل vath و داو ال Path على volumes ل provisioner هو اللي بيوفر ال volumes ده او ال Path ده على الجهاز لو انت شغال علي الكلاود او منزل ال Kubernetes

mohamed@MohamedAtef:-\$ kubectl delete pvc pvc-storage

persistentvolumeclaim "pvc-storage" deleted

◄ وبعدين لو عملنا list لل pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f اللي اتعمل pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f
 ال انعمل pvc-storage لل delete او اما عملت delete ل

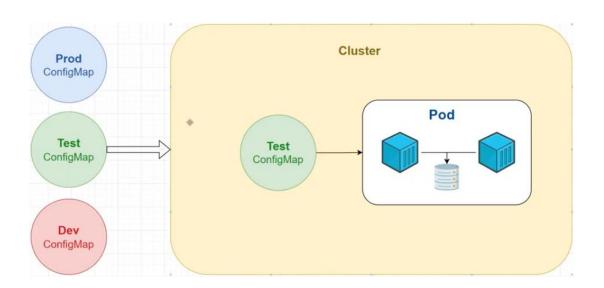
mohamed@MohamedAtef:~\$ kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE
pv-storage 2Gi RWO Retain Available <unset> 8h

# **Config Maps in Kubernetes**

#### **Config Maps** •

- ال Config Maps عباره عن object موجود في ال Kubernetes بيقدر يخليك تخزن Configuration يتم استخدامها بواسطه object تانيه بتكون عباره عن key-value pairs . وال config Maps مصممين ان هما يحتفظوا بال Configuration Parameters وبيتم حقنهم في ال configuration Parameters
- وال Config Maps تقريبا بتكون مناسبه لكل المواقف اللي انت محتاج تمد الابلكيشن بتاعك ب Configuration و URL بتاع ال values داخل ال Pods زي مثلا ال IP Address وال Proxy Services



```
evn=prod
databaseserver=hrmsdb
ip=172.0.0.1
```

- ◄ احنا كده معملناش غير file فيه ال Parameters اللي حددنهاله ف هنروح علي ال Kubernetes ونقوله اعملي من الملف ده Config Map
- ك قبل اما نعمل create لل file فيه default config map موجوده اصلا لما بت اinstall ال kube-root من kube-root فيه kubectl get configmap هنلاقي config map خلال ال minikube في في الله في Certification الله بتنزل ال by default ودي بيكون موجود فيها ال

```
mohamed@MohamedAtef:~$ kubectl get configmap
NAME DATA AGE
kube-root-ca.crt 1 13d
```

mohamed@MohamedAtef: \$ kubectl create configmap config-file --from-file=/home/mohamed/db.properties configmap/config-file created

◄ لو عملنا list لل configmap هنالقيه عمل create ل create من حوالي 5 ثواني بنفس الاسم اللي احنا محددينه

```
mohamed@MohamedAtef:~$ kubectl get configmap

NAME DATA AGE

config-file 1 15s

kube-root-ca.crt 1 13d
```

ري cm ب configmap ب cm زي  $\sqrt{\phantom{a}}$ 

```
mohamed@MohamedAtef:~$ kubectl get cm
```

NAME DATA AGE config-file 1 15s kube-root-ca.crt 1 13d

➤ اقدر اشوف التفاصيل بتاعت ال Config map واشوف هو اخد انهي Values وعملها save جوه ال Save جوه ال map في ال Kubernetes بشكل عملي ازاي

```
mohamed@MohamedAtef:~$ kubectl get configmap config-file -o yaml
apiVersion: v1
data:
db.properties: |
evn=prod
databaseserver=hrmsdb
ip=172.0.0.1
kind: ConfigMap
metadata:
creationTimestamp: "2024-07-27T16:36:17Z"
name: config-file
namespace: default
resourceVersion: "124208"
uid: 44a95b45-8675-4297-a1f3-d929c6c6cbf1
```

◄ ممكن كمان استخدم ال describe علشان اشوف معلومات تانيه

```
mohamed@MohamedAtef:-$ kubectl describe configmap config-file

Name: config-file

Namespace: default

Labels: <none>

Annotations: <none>

Data

====

db.properties:
----
evn=prod
databaseserver=hrmsdb
ip=172.0.0.1
BinaryData
====

Events: <none>
```

config map ∪ delete ل عايز اعمل

mohamed@MohamedAtef:-\$ kubectl delete configmap config-file configmap "config-file" deleted

◄ هنعمل config map ل create من config map ف هيعملها file ف هنعمل file اسمه وليكن create اسمه وليكن mydatabasecred.conf
 ◄ هنعمل parameters ل وهنكتب فيها ال Parameters دي

MYSQL\_USER=mohamed
MYSQL\_ROOT\_PASSWORD=password

mohamed@MohamedAtef: \$ kubectl create configmap keys --from-file=/home/mohamed/ConfigMapDemo/configmap/keys created

🔾 لو عملنا list لل config maps هنالقيه قايلك ان ال Name الجديد اسمه keys وعندي في ال

mohamed@MohamedAtef:~\$ kubectl get configmap

NAME DATA AGE

keys 2 63s

kube-root-ca.crt 1 13d

config map التفاصيل بتاعت ال التفاصيل بتاعت

mohamed@MohamedAtef:~\$ kubectl get configmap keys -o yaml apiVersion: v1 data: db.properties: | هنلاقي هنا عندي حاجتين في ال data هنلاقي عندي الجزء الخاص بال evn=prod db.properties والجزء الخاص ب db.properties databaseserver=hrmsdb ip=172.0.0.1 mydatabasecred.conf: "MYSQL\_USER=mohamed\nMYSQL\_R00T\_PASSW0RD=password \n\n" kind: ConfigMap metadata: creationTimestamp: "2024-07-27T18:31:24Z" name: keys namespace: default resourceVersion: "129724" uid: 968d0d2d-ccfb-428c-8ac2-4e4bbbe8f428

◄ احنا شوفنا كده اننا نقدر ن create ال config Map من خلال file واحد او من خلال folder ممكن كمان أن ال CMD احنا نعمل create لل CMD عن طريقه حاجه اسمها literal ان انا هدهاله بشكل مباشر من ال CMD

mohamed@MohamedAtef:~\$ kubectl create configmap newconfig --from-literal=env=test --from-literal=ipaddress=172.0.0.5

➤ لو انت عايز تشوف باقي ال commands بتاعت ال Config Map هتستخدم

mohamed@MohamedAtef: \$\times\$ kubectl create configmap --help

## Secrets in Kubernetes

- ◄ ال Secrets وال Config Maps تقريبا هما نفس الحاجه بس الفكره ان ال Secrets بنحتفظ فيها بالبيانات اللي بتكون حساسه زي مثلا User name و هكذا
- ◄ ال Kubernetes Secrets عباره عن معلومات حساسه زي مثلا ال Login username وال Password وال Kubernetes وال Keys وباقي الحاجات اللي بتحتفظ ببعض البيانات الحساسيه اللي بيتم استخدامها في البيئه بتاعت ال Kubernetes
- ◄ الهدف الرئيسي من استخدام ال Secrets في ال Kubernetes ان انت تقلل الخطوره او ال risk من ان انت ت
   Application لل deploy في ال expose
- ≥ من المعلومات المهمه ان ال secrets بيكون عندها size limit اللي هو 1MB وده مقصود ومتعمد ان ال secrets عندك يكون مش اكبر من كده علشان متحتفظش فيه ببيانات اكبر من كده ولما تيجي ت implement ال secrets عندك و volumes داخل كل Environment Variables داخل كل Pod داخل كل العنون على شكل secrets الله على شكل على شكل على العنون من الله على المناسكة عندك العنون من المناسكة على المناسكة على

## **Types of Secrets**

★ فيه عندنا انواع عديده من ال secrets في ال Kubernetes

#### Opaque -1

- ده بيكون ال Default type of secrets يعني لو انت جيت في ال Configuration file محددتلهوش نوع معين من ال secrets ف هيكون

#### Service Account Token -2

#### **Basic Authentication -3**

- وده عباره عن basic authentication credentials بس لازم تمدله username و Password

#### SSH authentication -4

- وده بيحتفظ بالداتا الضروريه علشان تقدر ت establish ال SSH Connection والdata field بتاعت الداتا الضروريه علي type الله SSH-Privatekey key-value pair ده لازم تحتوي علي type

#### TLS -5

- النوع ده بيحتفظ ب ال certificates وال associated keys اللي بتستخدم علشان ال TLS ولازم تتاكد ان يكون موجود معاك ال tls.crt وال tls.crt علشان النوع ده يشتغل بشكل سليم

#### **Docker Config -6**

- النوع ده بيحتفظ بال credentials الخاصه بال Docker registry ل اي credentials

#### BootStrap token -7

- دي عباره عن Tokens بتستخدم خلال node bootstrap process زي مثلاً انها بتستخدم ان انت تعمل sign ConfigMaps

# **Applying Secrets in Kubernetes**

mohamed@MohamedAtef: \$ kubectl create secret generic test-secret --from-literal='username=admin' --from-literal='password=987654we'

➤ لو انا عايز اعرف ال commands اللي هستخدمها مع ال

mohamed@MohamedAtef:~\$ kubectl create secret generic -h

Secrets الل list حنعمل ◄

mohamed@MohamedAtef:~\$ kubectl get secret

NAME TYPE DATA AGE test-secret Opaque 2 9h

◄ ممكن كمان نستخدم ال describe علشان نشوف تفاصيل عن ال Secrets اللي عملناه

mohamed@MohamedAtef:~\$ kubectl describe secret test-secret

Name: test-secret
Namespace: default
Labels: <none>
Annotations: <none>

Type: Opaque

Data

password: 8 bytes username: 5 bytes

هنلاقيه هنا قايلك اسم ال secret وهو test-secret وال Namespace بيكون ال default ومفيش Labels ولا Annotations ونوعه Opaque وجايبلك الباسورد وال Username بس هنا هو مش جايبلي الباسورد اللي انا كتبتها اللي هي we987654 ولا ال admin اللي اسمه username

◄ ممكن نعرف تفاصيل اكتر عن طريق ال command ده

mohamed@MohamedAtef:~\$ kubectl get secret test-secret -o yaml

apiVersion: v1 data:

password: OTg3NjU0d2U= username: YWRtaW4=

kind: Secret metadata:

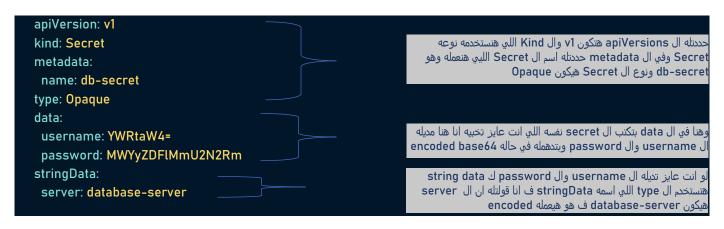
creationTimestamp: "2024-07-27T20:06:22Z"

name: test-secret namespace: default

هنلاقيه هنا جايبلك شكل ال yaml file ف لو لاحظنا هنلاقي في ال Data ال Password وال username جايبهملك encode مش جايبلك ال Password ولا ال Username اللي انت كاتبهم ف ممكن تاخد الباسورد اللي ظاهر عندك وتروح علي اي موقع بيحول ال encoded base64 ل decode

# Managing Secrets using Configuration File

- ➤ هنشوق ازاي هنقدر ن Manage ال Secrets باستخدام ال Configuration File وهنشوف انواع تانيه غير ال Opaque
  - 🗸 هنعمل Secrets ل Secrets ف هنعمل yaml file وليكن باسم db-secret.yaml وهنكتب جواه الاتي



db-secret.yaml الل apply الل apply الله حد

mohamed@MohamedAtef: \$ kubectl apply -f db-secret.yaml
secret/db-secret created

♦ لو عملنا list لل secret اللي عندي هنالقي اتعمل create من 67 ثانيه

mohamed@MohamedAtef: \$ kubectl get secret

NAME TYPE DATA AGE db-secret Opaque 3 67s

◄ لو جينا شوفنا تفاصيل اكتر عن ال db-secret اللي احنا عملناه

mohamed@MohamedAtef:~\$ kubectl get secret db-secret -o yaml apiVersion: v1 data:
 password: MWYyZDFlMmU2N2Rm server: ZGF0YWJhc2Utc2VydmVy username: YWRtaW4= kind: Secret metadata:

هنلاقي هنا جايبلك ال username وال password اللي احنا كتبناهم في ال yaml file وجايبهملي encoded حتي ال server حول ال string اللي احنا كنا كاتبينه اللي هو database-server من Encode ل Decode

annotations:
kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","data":{"password":"MWYyZDFlMmU2N2Rm","username":"YWRtaW4="},"kind":"Secret","metadata":{"annotations":{},"name":"db-secret","namespace":"default"},"stringData":{"server":"database-server"},"type":"Opaque"}

creationTimestamp: "2024-07-29T10:45:43Z"

name: db-secret namespace: default resourceVersion: "146606"

uid: 540b85d9-7611-4a85-9d41-186871cb3778

type: Opaque

هنلاقیه جایبلك كمان اتعمله creation امتي وال name بتاعه وال namespace ونوع ال secret

## فيه طريقه تانيه نقدر نديله في ال configuration file البيانات اللي احنا عايزنها

apiVersion: v1 kind: Secret metadata:

name: db-secret-un

type: Opaque stringData: config.yaml:

ytchannel: "https://www.youtube.com/c/codographia"

username: admin password: password

نفس ال file اللي استخدمناه بس مسحنا الجزء الخاص بال data وحطينا ال stringData ان هو اللي هيعمل للبيانات Encode وبزود تحت ال stringData بزود type جديد اسمهtig.yaml علشان اديله ال items اللي انا عايزها ف احنا هنا مدينله اسم قناه اليوتيوب ومديله كمان username و Password

db-secret-un.yaml الله apply هنعمل على الم

 ${\color{blue} mohamed@MohamedAtef:} {\color{blue} *} \textbf{ kubectl apply -f db-secret-un.yaml}$ 

secret/db-secret-un created

♦ لو عملنا Describe لل db-secret

mohamed@MohamedAtef: \$ kubectl describe secret db-secret-un

Name: db-secret-un
Namespace: default
Labels: <none>
Annotations: <none>

Type: Opaque

Data

config.yaml: 86 bytes

هتلاقيه هنا جايبلك ال config.yaml بيكون 86 bytes ومجابش اي تفاصيل تانيه

db-secret-un اكتر عن ال موفنا تفاصيل اكتر عن ال

mohamed@MohamedAtef:~\$ kubectl get secret db-secret-un -o yaml

apiVersion: v1

data:

config.yaml:

eXRjaGFubmVs0iAiaHR0cHM6Ly93d3cueW91dHViZS5jb20vYy9jb2RvZ3JhcGhpYSIKdXNlcm5hbWU6IGFkbWluCnBhc3N3b3Jk0iBwYXNzd29vZAo=

kind: Secret metadata:

annotations:

هنلاقي هنا التفاصيل اللي كنا مدينهاله في ال yaml file من حيث ال username وال Password وال

kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"Secret","metadata":{"annotations":{},"name":"db-secret-

un","namespace":"default"},"stringData":{"config.yaml":"ytchannel: \"https://www.youtube.com/c/codographia\"\nusername: admin\npassword: password\n"},"type":"Opaque"}

creationTimestamp: "2024-07-29T11:39:35Z"

name: db-secret-un namespace: default resourceVersion: "149188"

uid: 7608169b-4500-4970-a466-f07f44b79531

type: Opaque

# هنشوف ازاي هنعمل create ل Pod وال Pod ده هندیله ال Secret هنشوف ازاي هنعمل Configuration file

yaml file من ال Terminal من خلال ال Kubectl من ال secret هنعمل عالم secret من ال العمل عالم على العمل

mohamed@MohamedAtef: \$ kubectl create secret generic server-user --from-literal=server-username='ca\_admin' secret/server-user created

vaml file ل Pod ل create هنعمل secret من خلال ال Pod كده احنا عملنا

apiVersion: v1 kind: Pod metadata:

name: env-server-user-secret

spec:

containers:

- name: envvar-container

image: nginx

env:

- name: SECRET\_SERVERUSER

valueFrom:
secretKeyRef:
name: server-user
key: server-username

في ال environment بديله اسم ال environment اللي احنا عايزينه وال value هتيجي من ال server-user وال key اسمه server-username وده نفس ال server-username اللي موجود في ال secret اللي عملناه

Pod ال apply لن Pod ♦

mohamed@MohamedAtef: \$ kubectl apply -f server-user.yml
pod/env-server-user-secret created

← لو عملنا list لل Pod هنالقيه عمل create لل pod بنفس الاسم Pod فنالقيه عمل

mohamed@MohamedAtef:~\$ kubectl get pod

NAME READY STATUS RESTARTS AGE env-server-user-secret 1/1 Running 0 60s

ك لو عملنا Describe لل Pod اللي عملناه اللي اسمه Describe وروحنا في ال Describe والمحافق الله عملناه اللي الله عملناه الله على الجزء الخاص ب ال Environment هنالقي ال SECRET\_SERVERUSER الله كنا كاتبينه في ال الخاص بال Pod ف معنى كده ان ال Pod ده ات assign ليه ال secret بشكل سليم

mohamed@MohamedAtef:~\$ kubectl describe pod env-server-user-secret

**Environment:** 

 ${\tt SECRET\_SERVERUSER:} \ \ {\tt <set\ to\ the\ key\ 'server-username'\ in\ secret\ 'server-user'} \ \ {\tt Optional:\ false}$ 

Mounts:

/var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-f5xz6 (ro)

Conditions:

◄ ممكن نستخدم ال command ده لو احنا عايزين نشوف تفاصيل بشكل مباشر عن ال secret

mohamed@MohamedAtef: \$ kubectl exec env-server-user-secret -- env

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

HOSTNAME=env-server-user-secret

SECRET\_SERVERUSER=ca\_admin

KUBERNETES\_SERVICE\_PORT\_HTTPS=443

KUBERNETES\_PORT=tcp://10.96.0.1:443

KUBERNETES\_PORT\_443\_TCP=tcp://10.96.0.1:443

هنلاقي هنا ال SECRET\_SERVERUSER اللي احنا كنا كاتبينه واحنا بنعمل ال secret من خلال ال Terminal اللي هو ca\_admin

## هنعمل secret ل secret بس بنوع مختلف عن ال create

◄ هنعمل create ل yaml file باسم مثلا basic-auth.yaml وهنكتب في ال file ده الاتي

apiVersion: v1 kind: Secret metadata:

name: secret-basic-auth type: Kubernetes.io/basic-auth

stringData:

username: admin password: pa\$\$word

في ال file هنا ادناله ال apiVersion وادناله ال kind وهو Secret وفي ال metadata كتبنا الاسم وليكن secret-basic-auth انت ممكن تسميه اي اسم عادي

جينا هنا في ال type حددناله نوع ال secret هيكون type هيكون type ال وال basic-auth دا نوع من انواع ال secret وحددنا في ال stringData ال username وال Password لان ال Kubernetes بيكون طالب الاسم والباسور د

basic-auth.yaml لل apply هنعمل ◄

mohamed@MohamedAtef:~\$ kubectl apply -f basic-auth.yaml secret/secret-basic-auth created

◄ لو عملنا get لل secret هنلاقیه عمل create ل secret-basic-auth ونوعه secret-basic-auth ونوعه Kubernetes.io/basic-auth

mohamed@MohamedAtef:~\$ kubectl get secret

NAMETYPEDATAAGEdb-secretOpaque33h48mdb-secret-unOpaque1174msecret-basic-authKubernetes.io/basic-auth294sserver-userOpaque177m

## هنشوف ازاي نعمل secret ون assign ون secret ال عامن خلال ال

◄ اللي احنا هنعمله هنعمل create ل two files واحد لل secret وواحد لل

◄ اول حاجه هنعمل create ل secret file وليكن هسميه my-secret.yaml وهكتب فيه الاتي

apiVersion: v1 kind: Secret metadata:

name: vol-secret

data:

username: YWRtaW4= password: bW9oYW1lZA==

secret لل apply بعد كده هنعمل ج

mohamed@MohamedAtef:~\$ kubectl apply -f my-secret.yaml

secret/vol-secret createdd

→ بعد كده هنعمل create لل Pod اللي هيكون فيه ال Volume ف هعمل create لل pod باسم مثلا pod-volume-secret.yaml وهنكتب فيه الاتي

apiVersion: v1

kind: Pod metadata:

name: secret-vol-pod

spec:

containers:

- name: secret-container

image: nginx volumeMounts:

- name: secret-volume

mountPath: /etc/secret-volume

readOnly: true

volumes:

- name: secret-volume

secret:

secretName: vol-secret

جينا هنا حددناله kind ان احنا هنعمل pod وادناله اسم kind جينا هنا

وفي ال container ادناله اسم secret-container وال Image اللي هنستخدمها هو، ngipy

وال Volume اللي احنا هنعمله mount هيكون اسمه secret-volume وال volume وال mount اللي جوه ال etc/secret-volume

وال Volume اللي هنا لازم ي match ال volumeMount ف واحنا بن create ال volume هيكون اسمه secret-volume

ولازم يحصل link مابين ال configuration file اللي بي create ال Pod اللي احنا بنعمله ده ومابين ال configuration file الخاص بال secret اللي اسمه my-secret.yaml ف انا هاجي في ال secretName هكتب اسم ال secret اللي عملنا ليه cerate وهو vol-secret

♦ هنعمل create لل Pod

mohamed@MohamedAtef:~\$ kubectl apply -f pod-volume-secret.yml pod/secret-vol-pod created

>> Loreate هنلاقیه عمل get جدید اسمه get با عملنا get و عملنا عمل

mohamed@MohamedAtef:~\$ kubectl get pod

NAME READY STATUS RESTARTS AGE env-server-user-secret 1/1 Running 1 (46m ago) 178m secret-vol-pod 1/1 Running 0 60s

⇒ هنعمل describe ونشوف ال description بتاعت ال Pod لما ال Secret بي description بي assign as a volme بيكون عامل ازاي

#### mohamed@MohamedAtef:~\$ kubectl describe pod secret-vol-pod

Volumes:

secret-volume:

Type: Secret (a volume populated by a Secret)

SecretName: vol-secret

Optional: false kube-api-access-r8fpd:

لو روحنا في ال description علي الجزءء الخاص بال Volumes هنلاقيه بيقولك ان فيه secret-volume وال type بتاعه secret وبيقولك ان ال volume معموله secret براسطه ال Secret وال votenal اللي واصل بيه هو vol-secret وال poptional معمول ليها false

- ◄ لو احنا عايزين نشوف اللي احنا عملناه اللي هو ال Username وال Password جوه ال container نفسه داخل ال Pod عن طريق احنا هنعمل connect علي ال Pod وندخل علي ال pod ونستخدم كذا pod ونشوف التفاصيل دى
  - → علشان اعمل connect او ادخل جوه ال pod هستخدم

mohamed@MohamedAtef:~\$ kubectl exec -i -t secret-vol-pod -- /bin/bash root@secret-vol-pod:/#

➤ احنا كده دخلنا جوه ال container ف لو عملنا Is علي الpath اللي كنا محددينه في ال container الله الاحداد الله و etc/secret-volume لله و etc/secret-volume لله و password وال password والـ password

root@secret-vol-pod:/# ls /etc/secret-volume/
password username

◄ لو عايزين نشوف القيم ذات نفسها هنستخدم ال command ده لو في حاله ال username هنالقيه قايلك ان ال admin هو username

root@secret-vol-pod:~# echo "\$(cat /etc/secret-volume/username)"
admin

→ ولو في حاله ال Password هنا قايلك ان ال Password هنا قايلك ان ال

root@secret-vol-pod:~# echo "\$(cat /etc/secret-volume/password)" mohamed

#### ملاحظه

◄ لو انا عایز احول من encode ل encode ممکن استخدم ال command ده ولیکن انا عایز کلمه decode تتحول ل bw9oyw1lZA== ل Mohamed

mohamed@MohamedAtef:~\$ echo -n 'mohamed' | base64 bW9oYW1lZA==

◄ لو انا عايز اعمل العكس اني احول من decode ل encode هتسخدم نفس ال command ده بس هحط في
 نهایه ال command او بشن b-

mohamed@MohamedAtef:~\$ echo -n 'bW9oYW1lZA==' | base64 -d mohamed

## Stateful Sets in Kubernetes

## Stateful VS Stateless in Kubernetes

- > ال Stateless عباره عن Applications لا تحتفظ بالمعلومات في العمليات السابقه اللي انت قومت بيها .كل مره انت بتنفذ ال Operation ده بتنفذها من البدايه زي اول مره انت عملتها عامل زي ال Printing ال Apache وال Nginx وعامل زي ال (CDN(Content Delivery Network) او ال Services
- ال Applications عباره عن Applications بيحتفظ بالبيانات زي ال user profile وال Applications وال Stateful عباره عن Permissions تقدر تتخيل ان ال stateful application لازم يكون بي Permissions وال SQL Server أو Write و write في عمليه ال SQL Server أنواع قواعد البيانات زي SQL Server او Stateful applications

## What Is StatefulSets

- ◄ ال StatefulSet في ال Kubernetes عباره عن Object بي Manage او يتحكم في مجموعه من ال Pods او rescheduled
   هويات مميزه من خلال ان هو بيعمل assign ل persistent ID حتى لو ال Pod اتعمله Pod
- ◄ وال StatefulSet بتساعد ان انت تحافظ علي ال uniqueness او التفرد والترتيب بتاع ال Pods . من خلال cluster لل attach بيقدروا بشكل efficiently ان هما يعملوا attach لل Administrator ل pod across failure ل volumes
  - ← ال StatefulSet controller بي Deploys ال Pods ال Pods باستخدام similar specifications بس ال Pod مش interchangeable
    - ← ال StatefulSet مش بتنشئ ReplicaSet زي ال StatefulSet
    - ← وكمان متقدرش ترجع بال replicas بتاعت ال pod في ال StatefulSet لل Previous versions
- ◄ ال StatefulSet بيتم استخدامها بالاخص في الابلكيشن اللي بينطلب منها Stateful بيتم استخدامها بالاخص في الابلكيشن اللي بينطلب منها workloads and ordered ,automated rolling updates

# Why Not PVC?

- Pod ومش كل Object Shared across the pods هو عباره عن Persistent Volume Claim(PVC) ومش كل State الخاصه بتاعته
- ◄ اما ال StatefulSet بيكون كل Pod عنده ال State بتاعته او بمعني عملي اكتر ان كل Pod هيكون عنده ال Volume

## **Components of a Kubernetes StatefulSet Configuration Manifest**

- Components رئيسيه بتكون موجوده في ال Components رئيسيه بتكون موجوده في ال Configuration Manifest file
- 1- اول حاجه بتكون موجوده وهي ال StatefulSet نفسها ودي عباره عن template بت define ان انت عايز كام replica من ال container او ال Pod
- 2- بيكون موجود حاجه اسمها Headless service وده عباره عن Network Domain Controller اللي بيخلي ال Pod على ال Connect على ال
- 3- تالت حاجه ودي الاهم وهي ال Volume Claim Template ودي بتسمح لل Administrator ان هو يعمل Persistent Volumes ال stateful storage لل Provision

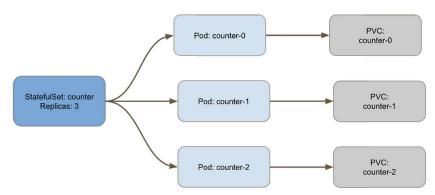
## **StatefulSets VS Deployments in Kubernetes**

🗸 هنلاقي ان ال StatefulSet شبه ال Deployments بس فيه اختلافات جو هريه مابينهم

Deployments	StatefulSet
ال Deployment بتمثل ال Stateless	ال StatefulSet بتمثل ال Stateful
ال Pods في ال Deployment بت assigned ليها ID بيتكون من اسم ال Deployment و random hash علشان ي temporarily unique identity	ال Pod في ال StatefulSet بيحصل علي Identity ثابته وبتتكون من ال StatefulSet name و sequence number
ال Deployments ال Pods فيها Identical كلهم زي بعض وبيكونوا interchanged	ال pods في ال StatefulSet مبيكونوش Identical ولا interchangeable
تقدر تعمل replica لل pods ب new replica في اي وقت	لما تيجي تشغل ال StatefulSet علي node تانيه هي بتحافظ علي ال identity بتاعتها
في الDeployments كل ال share بي share ال PVC و ال Volume	کل Pod بیحصل علی Unique Volume and PVC
علشان نقدر نتعامل في ال Deployments مع ال Pods داخل ال Deployments بنعمل Service بنعمل	هنا بن create حاجه اسمها Headless service هي اللي بتقدر ت handle ال Pod من خلال ال Pod وال Clients
ال Pods بيتعملها create او Delete بشكل randomly	ال Pods بيتعمله creations بترتيب معين ومبيتعملهاش delete بشكل عشوائي

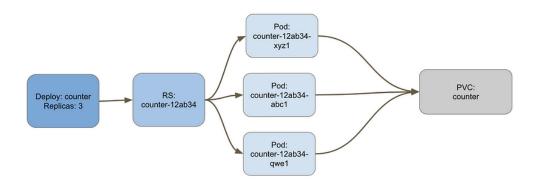
﴾ لو شوفنا ال Diagram ده هنلاقي ازاي في ال StatefulSets بيتم عمليه التعامل بين ال Pods وال PVC ف هنلاقي ان في عندي StatefulSets اسمه Counter وال Replicas بتكون 3 ف بيكون عندنا pod اسمه counter-0 و counter-1 وكل pod من ال pods دي عنده ال PVC الخاص بيه بالرقم بتاعه وده على عكس ال Deployments

#### **Persistence in Statefulsets**



⇒ في ال Deployment هنلاقي اسمه counter وال replicas بتكون 3 وال Deployments بتعمل create بتعمل deployments ل وال replicaset (RS) وهنلاقي ال podd واخده جزء من اسم ال replicaset (RS) وجزء randomly وجزء Pods وكل ال Pods بي access او بي share نفس ال PVC

## **Persistence in Deployments**



#### بعض عيوب ال StatefulSets

- بيقولك There is no built-in way to resize volumes يعني مفيش طريقه built-in او بشكل مباشر تقدر منها تعمل resize لل Volumes بعد ال Initial creation الموضوع ده ممكن يعملك مشكله كبيره لو ال Scale up بتاعتك بدئت ت scale up او تكبر او الداتا تكون عليها بشكل كبير
- وال Volumes are not deleted by default ان كل مره بنخلص بنروح نعمل delete لل Volumes بشكل منفصل
- وانت بت delete اللي انت عملتله StatefulSets مش شرط ان ال Pod ت terminate بال order اللي انت عملتله create بيها ودس بتمثل بعض ال
- واكيد اخدت بالك ان انت لازم manually ت create ال service لل service علشان تقدر تستخدم ال StatefulSets

# **Applying StatefulSets in Kubernetes**

- ✓ هنشوف حاجه عمليه عن ال StatefulSets وازاي نقدر نطبقها في ال Kubernetes
- ◄ ده کده configuration file خاص ب StatefulSets وزي مواضح هنعمل create وهما ال StatefulSets وهما ال StatefulSets

apiVersion: v1 kind: Service metadata: name: nginx labels: app: nginx ports: - port: 80 name: web clusterIP: None selector: app: nginx apiVersion: apps/v1 kind: StatefulSet metadata: name: web spec: selector: matchLabels: app: nginx serviceName: "nginx" replicas: 3 minReadySeconds: 10 template: metadata: labels: app: nginx terminationGracePeriodSeconds: 10 containers: image: registry.k8s.io/nginx-slim:0.8 - containerPort: 80 name: web volumeMounts: - name: www mountPath: /usr/share/nginx/html volumeClaimTemplates: - metadata: name: www accessModes: [ "ReadWriteOnce" ] storageClassName: "standard" resources: requests:

storage: 1Gi

دي ال configuration بتاعت ال Service وفي ال Service الحاجه الجديده اللي مستخدمناش قبل كده وهي ان في ال ClusterIP موجود none علشان زي ماقولنا في الشرح ان احنا محتاجين نخلي ال Service دي حاجه اسمها Headless service ان انا مش عايز اعمل Load Balancer او Cluster IP allocated ل اي serves وهنلاقي في ال Configuration بتاعت ال StatefulSets هنلاقيه مشوار علي ال Service اللي اسمها Siginx

- · دي ال configuration بتاعت ال StatefulSet
- ف انا هنا عرفت ال kind ان انا عايز اعمل StatefulSet واديتله اسم web
  - وال labels بتاعنا nginx
  - وال serviceNameهتکون nginx
- واًل replicas هتكون 3 واُحنا قولنًا ان ال replicas مش هتعمل create لا replicaset لا هي هتعمل create لل Pod علي طول وهيكون اسمه زي ال web وبعدين serial number
- من الحاجات اللي هتشوفها وهي ال minReadySeconds وده بيكون Options ممكن تحطه او متحطهوش واللي بيعمله انه بي define عدد الثواني ان لازم يكون خلالها ال Pod معمول ليه running من غير ما ال container يحصله crash او حاجه
- هنلاقي كمان فيه حاجه جديده وهي ال terminationGracePeriodSeconds وده الوقت اللي انت بتديه لل Kubernetes اللي بتقوله من خلاله اعمل termination لل Pod ده في خلال الوقت ده اللي هو مثلا 10 ثواني
- هنلاقي كمان الحاجه الجديدة اللّي متسخدمنهاش قبل كده وهي ال volumeClaimTemplates

#### ◄ قبل اما نعمل apply لل yaml file لو انا عايز اعرف ال Storage Class الموجود عندي هتسخدم

mohamed@MohamedAtef:~\$ kubectl get storageclass

NAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE standard (default) k8s.io/minikube-hostpath Delete Immediate false 13d

mohamed@MohamedAtef:~\$ kubectl apply -f simple-sts.yaml service/nginx created statefulset.apps/web created

◄ لو عملنا kubectl get all هنالقي اتعمل create لتلاته Pod واخد اسم ال staefulset اللي هو ال web وجنبها
 مانا service و د و عمل create ل service اسمها service

# mohamed@MohamedAtef:~\$ kubectl get all NAME READY STATUS RESTARTS AGE pod/web-0 1/1 Running 0 4m8s pod/web-1 1/1 Running 0 4m8s pod/web-2 1/1 Running 0 4m8s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d16h service/nginx ClusterIP None <none> 80/TCP 4m8s

NAME READY AGE statefulset.apps/web 3/3 4m8s

- ◄ لو اخدت بالك هتلاقي ان مفيش replicaset اتعملها create لانه عمل pod من خلال ال statefulset لان دي من خصائص ال statefulset
  - ◄ وكمان بيعمل create لل PVC انه بيعمل create لكل واحده PVC خاص بيها

#### mohamed@MohamedAtef:~\$ kubectl get pvc

STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE www-web-0 Bound pvc-362e5aa5-1843-409e-af9e-899bb658eb1b 1Gi 25s standard <unset> www-web-1 Bound pvc-952e5aa5-1843-152-af9e-899bb658eb1b RWO standard <unset> 40s www-web-2 Bound pvc-451254a5-1843-652-af9e-899bb658eb1b **RWO** <unset> 55s standard

#### PV ا create وعمل

#### mohamed@MohamedAtef:~\$ kubectl get pvc

CAPACITY ACCESS MODES RECLAIM POLICY STATUS STORAGECLASS VOLUMEATTRIBUTESCLASS NAME CI AIM REASON AGE pv-storage 2Gi RWO Retain Available <unset> default/www-web-0 standard pvc-362e5aa5-1843-409e-af9e-899bb658eb1b 1Gi Delete RWO Bound <unset> 43m pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3 1Gi Delete Released default/pvc-storage standard RWO

#### ◄ لو انا عايز اعرف ال StatefulSet اللي عندي

#### mohamed@MohamedAtef:~\$ kubectl get statefulset

NAME READY AGE web 0/3 18m

# **Jobs and Cronjobs In Kubernetes**

# **Jobs**

- ◄ ال Job في ال Kubernetes بتبقي مسئوله انها تنشئ Pod او اكتر يعني هو Object موجود في ال execution بيقدر بشكل Automatically يعمل Pod ل Create ل و اكتر وهيفضل يعمل Kubernetes لل Specified Number وي لحد اما ي Pods او يوصلوا لعدد معين انهم يكونوا
- ◄ ولما تعمل suspending لل Job انت توقفها مش تلغيها ف هي هتعمل delete لل suspending الموجوده لحد اما ترجع تكمل عمل ال Job دي
  - ◄ تقدر تستخدم ال Job ان انت تشغل اكتر من Pod على التوازي

### **Parallel Execution for Jobs**

- ∠ هنتكلم عن التنفيذ المتوازي لعمليات ال jobs في ال Kubernetes 
  ﴿
  - ◄ فيه عندي 3 انواع رئيسيه

#### Non-Parallel Jobs -1

- النوع ده مبيشتغلش علي التوازي بيشتغل علي التوالي و pod واحد بس اللي بيشتغل ولو ال Pod ده متعملهوش Job ال Fails ال Job هتعتبر انها Complete او اما ال Pod دي تكون Successfully بيعتبر ان ال Pod دي Pod دي Pod لان ال Job دي بقت Pod
  - Parallel Jobs with a fixed completion count -2
  - يعنى Job متوازيه بتشتغل مع بعض ب Job متوازيه بتشتغل
    - Parallel Jobs with a work queue -3
  - ده ب Involve running multiple Jobs بشكل concurrently علي التوازي او مع بعض . وفي starting another one علي المتات كتيره مش افضل حاجه ان انت تسمح ل Job ان هي تنتهي قبل اما ت

# **CronJobs**

- ◄ ال cron job هي هي ال job بس بتعمل running automation ل rask معين بناءا علي schedule وليكن
   عايز تاخد back مره في اليوم او مرتين في الشهر وهكذا ف انت اللي بتحدد ال task دي هتتنفذ امتي
- ◄ وتقدر ان انت ت define ال Time اللي بيعدي بين ال Jobs او ال frequency اللي هترن ال job دي كام مره
- ◄ ف ال CronJobs تقدر تستخدمها لو انت عايز تعمل Backups او ان انت تعمل cronJobs او generate timely emails او لو انت عايز ت schedule jobs بخصوص ال User-activity



Û

# kubernetes

- شرح ال Kubernetes بالكامل لكورس الباشمهندس Kubernetes الموجود علي قناه Codographia
  - رابط الكورس علي ال YouTube
  - https://www.youtube.com/watch?v=jTggu1HiKyY&list=PLX1b W GeBRhDCHijCrMO5F-oHg52rRBpl
  - رابط الكورس علي Udemy لعدم وجود بعض الفيديوهات
    <a href="https://www.udemy.com/course/kubernetes-step-by-step-in-arabic/?couponCode=ST3MT72524">https://www.udemy.com/course/kubernetes-step-by-step-in-arabic/?couponCode=ST3MT72524</a>
  - ال Slides المستخدمه في شرح الكورس https://drive.google.com/file/d/1nuxV8gTwJX xDexJgi113JeEpB3PEbr/view?usp=sharing
    - موجود بجانب كل عنوان الفيديو المستخدم في الشرح
      - **BY: Mohamed Atef Elbitawy**
  - in <a href="https://www.linkedin.com/in/mohamedelbitawy/">https://www.linkedin.com/in/mohamedelbitawy/</a>