kubernetes

- شرح ال **Kubernetes** بالكامل لكورس الباشمهندس **Ahmed Mohey** الموجود علي قناه **Codographia**

- رابط الكورس علي ال **YouTube**

https://www.youtube.com/watch?v=jTggu1HiKyY&list=PLX1bW_GeBRhDCHijCrMO5F-oHg52rRBpl

- رابط الكورس علي **Udemy** لعدم وجود بعض الفيديوهات

https://www.udemy.com/course/kubernetes-step-by-step-in-arabic/?couponCode=ST3MT72524

- ال Slides المستخدمه في شرح الكورس

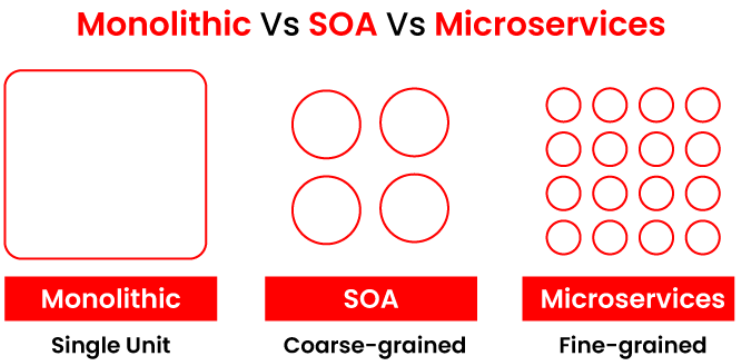
https://drive.google.com/file/d/1-nuxV8gTwJX_xDexJgi113JeEpB3PEbr/view?usp=sharing

- موجود بجانب كل عنوان الفيديو المستخدم في الشرح

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/

Monolithic vs SOA vs Microservices



• Monolithic Architecture

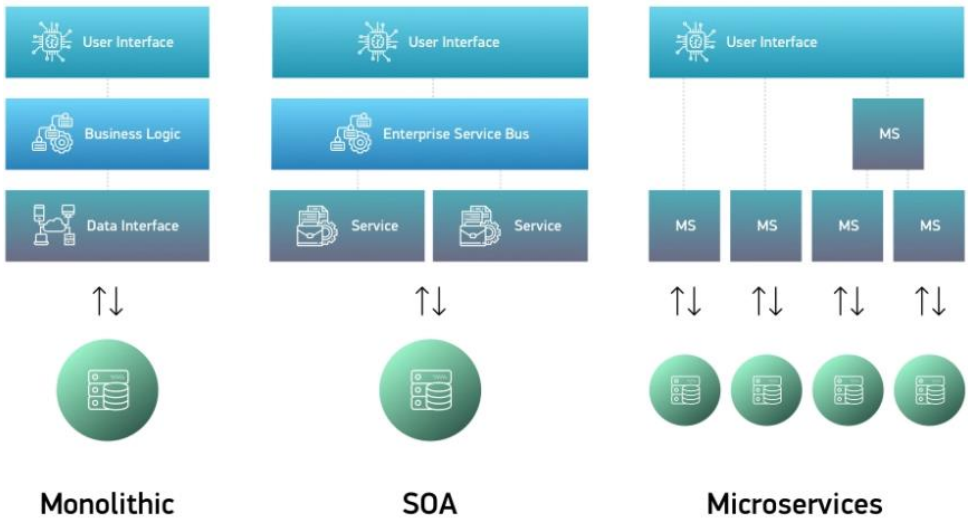
- هو عباره عن ال Traditional Model علشان ت design اي Software لفتره طويله وهو ببساطه عباره عن Single Piece of Software وكل حاجه موجوده في one Piece ودا مأدي ان كل حاجه معتمده علي حاجه تانيه
- طالما ال Monolithic Architecture شغال علي Single Process ف ال Scaling بتاعت ال Features لو انا عايز اعمل Scaling ل features فيه بيكون مستحيل ف لازم ت deploy new instance علي Server تاني وغالبا بتحطهم خلف Load Balancer
- 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 Downtime بتاعه كبير وال Service بتقف عند ال Clients لفتره من الوقت

• SOA Architecture

- ال SOA اختصار ل Service Oriented Architecture وده تحسين كبير لل Monolithic 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 small development team
- وكل Services منها ممكن تعملها deploy بشكل مستقل هي Service قائمه بذاتها وال Team المسئول عن ال Microservices يقدر ي Update ال Service من غير مايعمل rebuilding او redeploying للابلكيشن كله
- وال Services بي Communicate مع بعض باستخدام ال APIs وكل Service بتكون isolated عن ال Service التانيه



- هنلاقي في المثال اللي في الشكل ان في ال Monolithic فيه Data Interface وفيه Business Logic وفيه User Interface . في ال SOA هنلاقيه مقسملي ال Services ل اتنين اوتلاته او اكتر وفي ال SOA فيه حاجه اسمها Enterprise Service Bus بيوصل لل User Interface . اما ال Microservices هتفصل كل ده ب Microservices قائمه بذاتها جواها ال Data Base بتاعتها وال Code Base الخاص بيها وليها ال Team 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 ان هي ممكن تشتغل علي Linux او Mac او Windows بدون اي مشكله . وال Containers هي افضل حاجه علشان ت deliver ال Microservices علشان هي بت Providing Portable و Isolated Virtual environments يعني بيئات عمل متنقله ومعزوله عن بعض علشان الابلكيشن ي Run من غير اي Interference من اي Running Application تاني
- 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 محتاج حاجه اسمها **Containers Orchestrators**

• Containers Orchestrators

- ال Containers Orchestrators عباره عن tools بت group ال System بتاعك علشان ت form clusters وبيعملها automations ومن اشهر ال Tools هي Marathon و Nomad و Azure Service Fabric وال Amazon Elastic Container Service (ECS) وال Docker Swarm وال Kubernetes

• Kubernetes as a Service

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

What is Kubernetes

- ال **Kubernetes** عباره عن

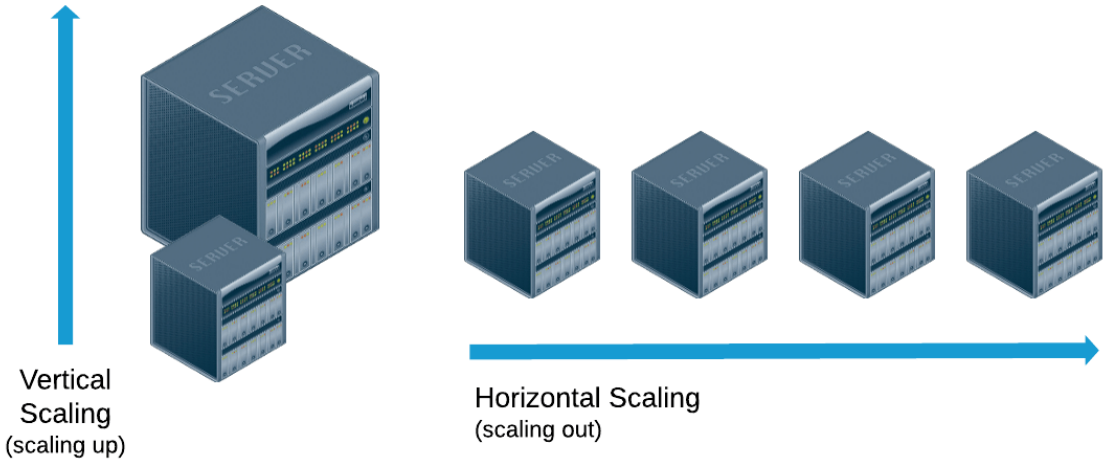
  Open Source System for automating deployment ,scaling ,and management of containerized applications

- 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 علي 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 يوفرلك ان انت تعمل Horizontal و Vertical Scaling لل Pods بناءا علي ال CPU وال Kubernetes بي automatically start new Pods لما ال threshold يكون reached
- فيه عندي نوعين من ال Scaling في عندي ال Vertical Scaling وال Horizontal Scaling
- ال **Vertical Scaling** او ال **scaling up** ان انا بعدل ال resources Attributes زي ال RAM وال CPU وال Hard Disks لكل Node في ال Clusters يعني انا مثلا عايز ازود ال Performance بتاع Application معين بدل 2 GB من الرام ل 4GB ف ده اسمه Vertical Scaling ان انا بزود ال Attributes الموجوده لل Resource بتاعت ال Container او ال VM
- ال **Horizontal Scaling** او ال **scaling out** ان بدل اما انت بتزود ال Resource بتاعتك هو بيضفلك New Nodes علشان تحسنلك ال Performance



2- ال Self-Healing

- ال Kubernetes بيقدر ي replace و reschedules ال Containers من ال failed node 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 unresponsive containers

3- ال Automated rollouts and rollbacks

- ال Kubernetes بيقدر يعمل rolls out و rolls back لل Applications updates وال Configuration changes وبي monitoring ال Application's health to prevent any downtime يعني لو انت مثلا عندك MySQL اصدار معين وعندك كذا replicaset او اكتر من Node او اكتر من Pods او اكتر من Containers شغالين ب Version قديمه وانت عايز تحط version تاني ف هو بيقدر انه يعمل rollouts لكل ال Pods من غير مايحصل اي مشكله وبيعمل rollbacks في حاله لو حصل اي مشكله

4- ال Secret and configuration management

- 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

5- ال Portability

- ان ال Clusters اللي انت بتعمله بال Kubernetes يقدر يشتغل علي اي Linux Distributions او اي Processor Architectures زي ال Virtual Machines او اي Cloud Providers زي ال AWS وال Azure وال Google Cloud Platform وبيقدر كمان يشغل Container runtimes غير ال Dockers

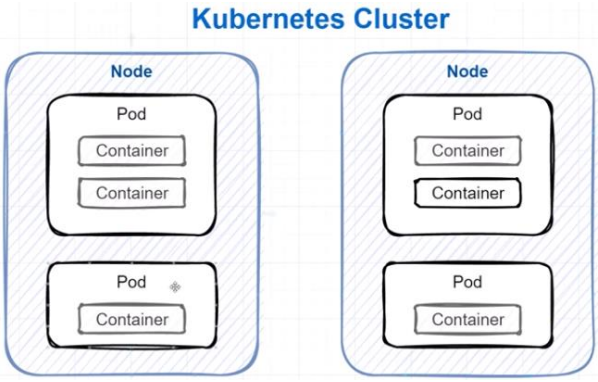
Kubernetes Architecture

- ال Kubernetes بيتقسم لحاجتين ل Worker Nodes و Master Nodes

Worker Nodes

➢ **ال Worker Nodes** بيوفر بيئه عمل لل Applications بتاعتك من خلال Containerized Microservices من خلال ال Containers اللي بت run ال Microservices وبتكون مغلفه في حاجه اسمها Pods وال Pod ده عباره عن Container او اكتر من Container جواه ال Containerized Application 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 resources
- دي كده صوره مبسطه لل Kubernetes Cluster هتلاقيه جايبلك اكتر من Node وجوه كل Node بيكون فيه اكتر من Pod وجوه كل Pod ممكن يكون فيه Container او اكتر



Worker Node Components

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

1- Container Runtime

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

2- Node Agent – kubelet

- ال kubelet عباره عن Agent بيكون running في كل Node بيقدر ي Communicate مع ال Control Plane وهو ال Component الرئيسي في ال Master Node وال Master Node هو ده العقل المدبر لل Kubernetes اللي بي manage كل حاجه ف بيستقبل ال Pod Definitions من ال API Server وهو اللي

بيتعامل مع ال Container runtime وبيقدر كمان ي Monitors ال health وال resources لل Pods اللي مشغله ال Containers

- وال kubelet بي Connect علي ال Container runtime من خلال Plugin based interface اسمه CRI(Container Runtime Interface)

3- Proxy-Kube-Proxy

- عباره عن Network Agent بيكون شغال علي كل Node ومسئول عن التعديلات اللي هتحصل وصيانه كل ال Network Rules علي ال Node يعني اي حاجه بخصوص ال Network في ال Node ال Kube-Proxy هو اللي بيعملها Manage
- يعني نقدر نقول ان ال Kube-Proxy مسئول عن ال TCP وال UDP وال SCTP

4- Addons

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

Master Node

- ال **Master Node** هو ده العقل المدبر في ال Kubernetes وال Master Node بيوفر بيئه عمل ل ال Control Plane اللي مسئوله عن التحكم في ال Kubernetes cluster
- علشان تقدر تتعامل مع ال Kubernetes Clusters ال User بيبعت request لل Control Plane الاول من خلال ال CLI او من خلال ال Web UI او من خلال 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 Control Plane وجواهم عدت Components

1- API Server(kube-apiserver)

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

2- Scheduler (kube-scheduler)

- ال Scheduler بي assigns Pods لل Nodes 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

3- Controller Manager (kube-controller-manager)

- 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 ل ال Desired State من الحاله الحاليه ل الحاله اللي مرغوب فيها ان انا اوصلها في ال Kubernetes يعني انا مثلا لو عندي pod موجود فيه Container وال Container ده حصل ليه اي مشكله بقي failed ف ال Controller manager هو اللي مسئول في الحاله دي انه يتصرف

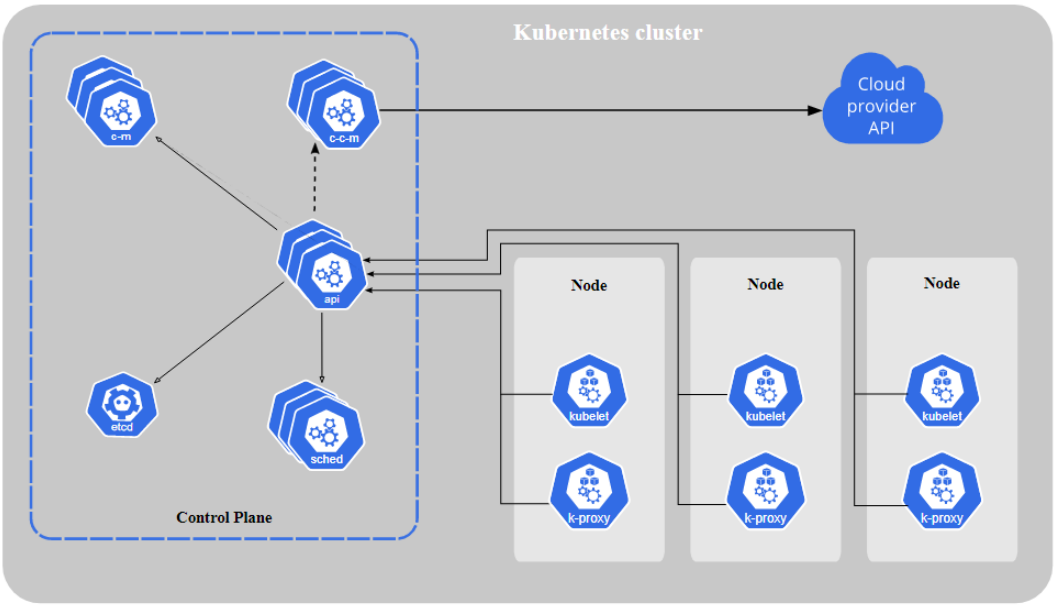
4- Data Store(etcd)

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

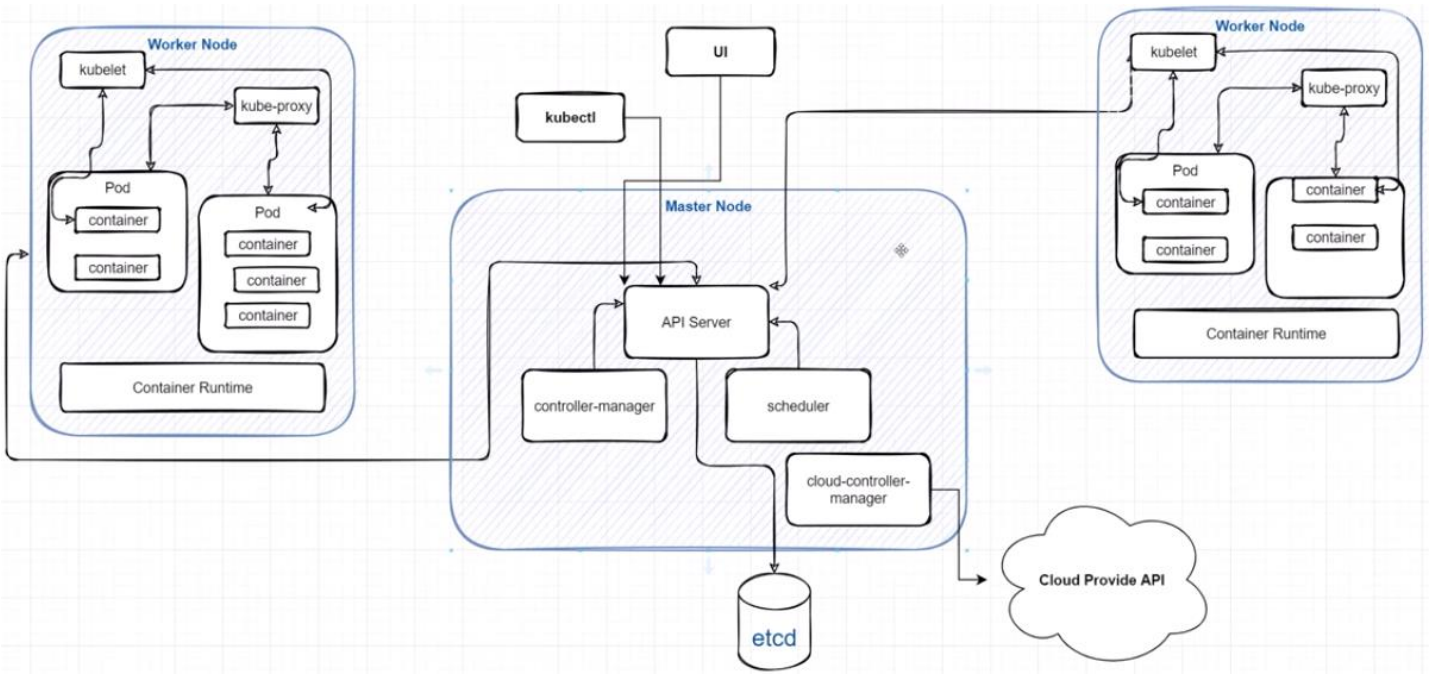
5- Kubectl

- ال **Kubectl** هو ال Command Line Interface بتاع ال Kubernetes

➢ في الصوره دي هتلاقيه بيقولك انفيه Control plane موجود جوه ال Master Node وعندنا ال Controller Manager اللي هو ال c-m وهنلاقي كمان موجود ال Cloud Controller Manager اللي هو c-c-m وهنلاقي كمان API وال etcd اللي بيتعامل معاها ال API وده كله بيتعامل مع ال Nodes المختلفه اللي جواها ال kubelet اللي هو ال Agent ال Running المسئول عن ان هو ي Run ال Pods جوه ال Nodes



- لو بصينا في الصوره دي هنلاحظ ان انا بيكون عندي User Interface(UI) او من خلال ال kubectl ببدء اكتب Commands against ال API Server او ال Kubernetes بيروح لل API Server فيبتدي يشوف ال commands دي صح ولا غلط او لو صح ايه اللي المفروض يعمله بالظبط ومن هنا يبتدي يشغل Node او Pod اين كان ال Process اللي هنعملها كل ده موجود جوه ال Master Node او ال Control Plane واللي جواه كمان ال Controller-manager وال scheduler وال Cloud Controller manager وهنلاقي سهم نازل من ال API Server رايح لل etcd لان احنا قولنا ان ال API server الوحيد اللي بيتعامل مع ال etcd database ومن ال API Server بنقدر نكلم كل Worker Node من خلال ال Agent اللي شغال عندنا اللي اسمه ال Kubelet وال Kubelet هو اللي هيبتدي يتعامل مع ال Containers وال Pods الموجوده عنده في ال Node يعني عندنا Pod جواها اتنين containers وفيه Podتانيه جواها اتنين Containers تانيين ف كل ده بي Run بناءا علي Container Runtime موجود جوه ال Node وزي ماقولنا ان ال kube-proxy هو اللي بيقدر ي manage ال Network بداخل كل Worker Node



Kubernetes Concepts Explained

1- Pod

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

2- Node

- ال Node عباره عن Worker Machine جوه ال Kubernetes ممكن تبقي Virtual او Physical 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 Kubernetes Control Plane هو اللي بينظم عملية كل Node او كل Pod انه scheduling علي انهي Node

3- Cluster

- ال Cluster في ال Kubernetes عباره عن مجموعه من ال Nodes اللي بت Run Containerized Applications اناحنا بيكون عندنا Application موجود جوه ال Container ومتوزع علي عدة Pods او Nodes ف ال Clusters بتقدر تRun ال Application ده
- ال Kubernetes Clusters بيقدر يخلي ال Containers تشتغل من خلال اكتر من Machines مش شرط تكون machine واحده بس لا هي تقدر تشتغل من خلال اكتر من Machines او اكتر من Environments زي ال Virtual وال Physical وال cloud-based وال on-premises

4- Namespaces

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

5- Service

- ال Service هي ال Logical Abstraction من ال deployed group of pods in a cluster يعني ال Pods اصلا مش متصممه ان اي حد يقدر يعمل access عليها من بره يعني لو انا عملت Application وحطيته علي Pod هو مش هيعرف ي access ال Pod بشكل Direct يعني احنا بنعمل Service وال Service عباره عن مجموعه من ال Pods وال User بي Access ال Service دي مش ال Pods نفسه
- ال Pod دايما بتكون ephemeral هي حاجه مش ثابته هي حاجه متغيره بشكل كبيرعلي عكس ال Service ال Service بتكون ثابته
- ال Service بتخليك تشتغل مع group of pods وبتمدك ب Specific Functions وبيديك IP Address مخصوص يقدر اليوزر ي access الابلكيشن من خلال ال Service في ال Kubernetes مش من خلال ال Pod بشكل مباشر

6- Deployment

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

7- Workloads

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

8- Volume

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

9- ReplicaSet

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

10- Ingress

- ال Ingress هو اللي بي manage لو حد عايز ي access ال Services من ال Kubernetes Clusters من بره فهو اللي بيقدر يmanage من خلال ال HTTP وال HTTPS وكمان بيقدر ي Provide ال load balancing و SSL Termination وال name-based virtual hosting

11- Orchestration

- ال Orchestration ان انت بتخلي ال effort اللي انت بتعمله بشكل manual علشان ت run containerized workload ف بي Manage ال lifecycle بتاعت ال containers وبيعمل حاجه اسمها Provisioning و deployment وبيعمل كمان Scaling سواء كان Up او Down وبيعمل كمان كل الامور المتعلقه بال Networking وال Load balancing

Installing Minikube on Ubuntu

خطوات تنزيل ال Minikube علي ال Ubuntu

```
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

- لو انا عايز اعمل start ل ال minikube هتستخدم **minikube start**

```
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:
😄  minikube v1.33.1 on Ubuntu 22.04
✨  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 **minikube status**

```
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
```

- علشان اعمل deployment هقوله **kubectl create deployment** وبعدين اسم ال image

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

- لو انا عايز اشوف ايه اللي اتعمله Create هقوله **kubectl get deployments**

```
mohamed@MohamedAtef:~$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
nginx-app   1/1     1            1           28m
```

- لو انا عايز اعرف ال Pods اللي عندي هقوله **kubectl get pods**

```
mohamed@MohamedAtef:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
nginx-app-69999bf9b8-2zxnv   1/1     Running   0          30m
```

- لو عايز اعرف ال Services اللي عندي هقوله **kubectl get services**

```
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       41m
```

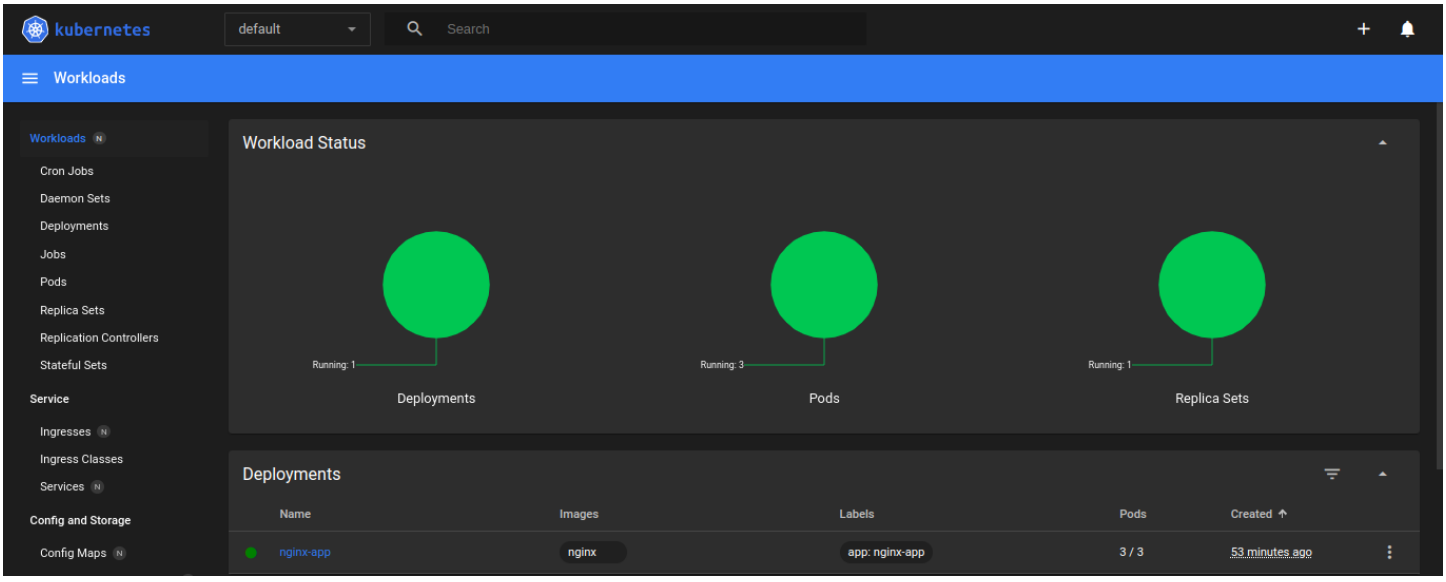
- لو انا عايز ازود ال scale بتاعي ل 3 هستخدم kubectl scale deployment وبعدين اسم ال image وبعدين استخدم اوبشن =replicas-- وبعدين احدد انا عايز ازود ال scale ل كام

```
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 ال dashboard minikube بيقدمها من خلال اني استخدم ال command ده **minikube dashboard** بعد اما انفذ ال commad ده هيفتحلي URL اقدر أ manage منه او اشوف كل 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": 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 Create ل Pod
- علشان نكتب ال Ymal File هنحتاج اننا نفتح ال File باي editor موجود عندنا وليكن هنستخدم ال Visual Studio Code ممكن افتح ال Ymal File باستخدام ال CLI عن طريق اني اقوله code وبعدين اسم ال Ymal File واهم حاجه ان امتداد ال file يكون .yml. هيفتحلي ال File باستخدام ال Visual Studio Code

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

- اول حاجه هنكتبها في ال ymal File هنكتب kind

```
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 **kubectl apply -f** 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 pod 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 command ده عباره عن انه بيجبلي وصف عن كل حاجه تخص ال pod دي من حيث ال Container وال IP وال Image وهكذا

```
mohamed@MohamedAtef:~$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sun, 21 Jul 2024 22:43:47 +0300
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.0.23
IPs:
  IP:  10.244.0.23
Containers:
  nginx:
    Container ID:   docker://118011f741fce5b1f478a9ae1c59f9410126a8c3fd0b945165559d37461a1a27
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 21 Jul 2024 22:43:50 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-8pb28 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-8pb28:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  24m   default-scheduler  Successfully assigned default/nginx to minikube
  Normal  Pulling    24m   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     24m   kubelet            Successfully pulled image "nginx:latest" in 1.66s (1.66s including waiting). Image size: 187599276 bytes.
  Normal  Created    24m   kubelet            Created container nginx
  Normal  Started    24m   kubelet            Started container nginx
```

هنلاقيه هنا مثلا قايلك اسم ال pod ايه

واتعملها start امتي

وجايبلك كمان ال status بتاعت ال pod هل هي Running ولا

وجايبلك كمان معلومات عن ال Container من حيث ال Container ID وايه هي ال Image وكمان جايبلي ال Port اللي هو شغال عليه واشتغلت امتي وكل التفاصيل الخاصه بال Pod دي

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 وبعدين Created وبعدين Started

- لو عايز اعرض نفس ال description بس في هيئه json format هستخدم **kubectl get pods** وبعدين اوبشن **-o** وبعدين **json**

```
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":
"{\"apiVersion\":\"v1\",\"kind\":\"Pod\",\"metadata\":{\"annotations\":{},\"name\":\"nginx\",\"namespace\":\"default\"},\"spec\":{\"containers\":[{\
"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 ده command بستخدمه في حاله لو انا معنديش Browser وعايز اعرض محتوي ال website في ال CLI ف لما انفذ ال command هيعرضلي ال document الخاصه بال nginx 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 ل 'Hello World from Pod in Kubernetes' ف هستخدم command اسمه echo وبعدين اضيف المحتوي اللي اللي هو 'Hello World from Pod in Kubernetes' وبعدين علامه اكبر من علشان اعمل redirect وبعدين اضيف المسار اللي فيه ال index.html الخاص بال nginx وده بيكون تحت /usr/share/nginx/html/

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

- لو عملنا curl تاني هنلاقيه عرضلي ال message اللي انا كتبتها في ال index.html

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

- لو انا عايز اخرج بره ال nginx هقوله **exit**

```
root@nginx:/# exit
exit
mohamed@MohamedAtef:~$
```

- لو انا عايز اعرض ال logs اللي بتاعت ال nginx هستخدم **kubectl logs nginx**

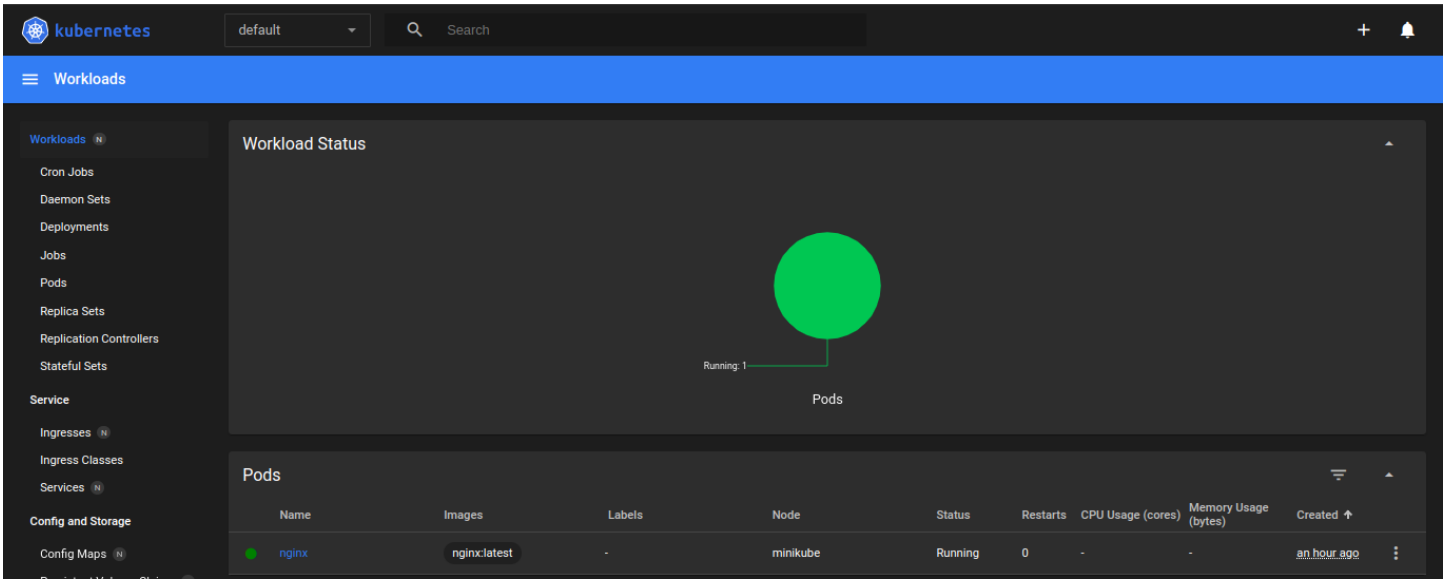
```
mohamed@MohamedAtef:~$ kubectl logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
/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 انا هستخدم **minikube dashboard** وهو هيفتحلي ال dashboard علي ال Browser علي طول او ممكن استخدم **minikube dashboard url** وهو هيديني ال url اللي هستخدمه علشان افتح ال dashboard عن طريق ان انا هاخد ال url ده واحطه في ال Browser او اني اضغط علي CTRL واضغط علي 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 ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
http://127.0.0.1:34133/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
```

ده ال URL اللي هتستخدمه علشان افتح ال dashboard

- بعد اما ال dashboard فتح معانا هنلاحظ ان ان احنا معندناش غير pod واحده بس اسمها nginx وال image بتاعتها اسمها nginx:latest ومعمول ليها Running



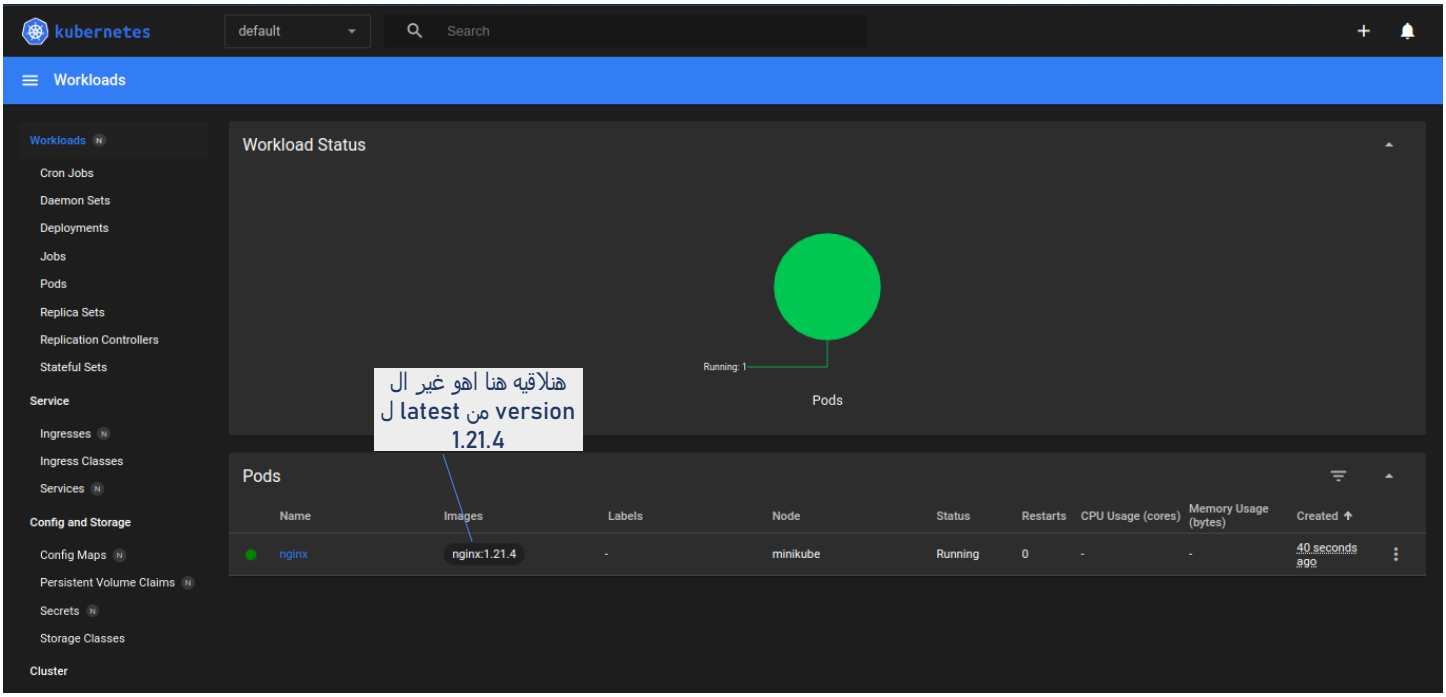
- طب افرض لو انا عدلت في ال version بتاعت ال nginx اللي عندي بدل اما هي latest نخلياه مثلا 1.21.4

```yaml
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 configured طب ليه 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
```

- تعال نشوف التغيير اللي عملناه علي ال dashboard

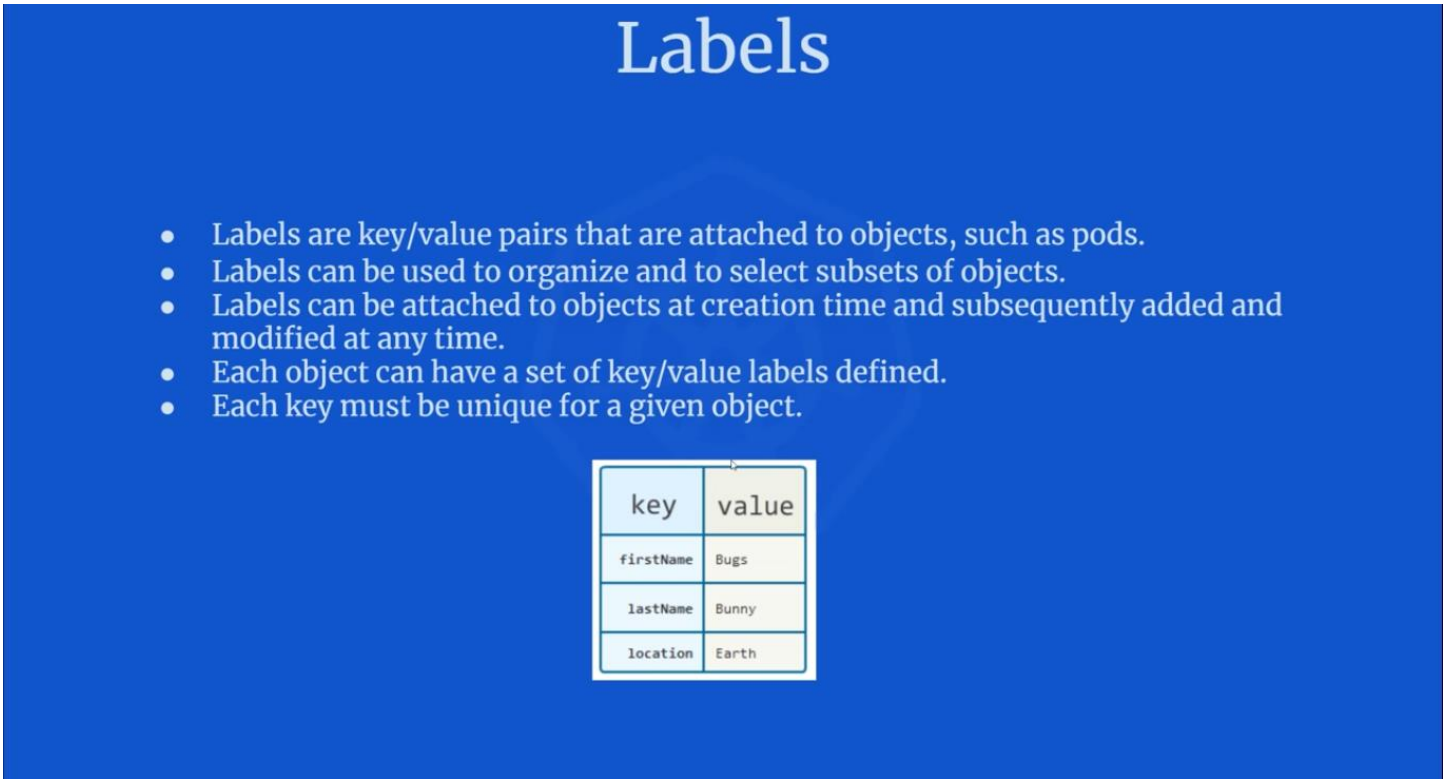


- لو انا عايز اعمل delete لل pod اللي اسمه nginx هستخدم

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



- ال Labels عباره عن key value pairs بتكون attached لأي object زي ال Pods
- ال key value pairs زي داتا بيز صغيره هي مش داتا بيز حقيقيه بس تقدر تسجل فيها Key و value ال key بيبقي عباره عن اسم زي مثلا FirstName و Last Name و ال Location وهكذا و اقدؤ احط كمان لل key احطله 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 configuration file بتاعنا وممكن بعد كده نضيفه في ال runtime 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 علي حسب انت عايز تميزهم ب ايه
- لازم ال Key يكون unique

- هنشوف مثال عن ازاي نضيف labels ف انا عملت create ل yml file باسم nginx.yml وضفت فيه

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

- بعد كده هعمل apply لل nginx.yml

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

- لو عملنا get pods هنلاقي فعلا اتعمل create لل pod اللي اسمها nginx

```
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>
```

- هنضيف Label عن طريق ان احنا هنستخدم ال command ده **kubectl label pod** وبعدين اضيف ال pod اللي عايز اضيف ال label فيها وليكن ف المثال بتاعنا هو ال **nginx** وبعد كده هضيف ال label عن طريق ان احنا هنضيف key و value وليكن هقوله ان ال key هو ال **owner** هوال value هو **mohamedatef**

```
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 هنلاقيه 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 mohamedatef 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 -l وهيطلعلي نفس ال OutPut

```
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 هنلاقي عندي two lables واحد اسمه owner=mohamedatef والتاني اسمه release=dev

```
mohamed@MohamedAtef:~$ kubectl get pods --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 two pods

```
mohamed@MohamedAtef:~$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          87m
nginx2   1/1     Running   0          91s
```

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

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

- لو جينا شوفنا ال Labels تاني هنلاقي 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 select ل 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 owner 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 طريقه اسمها **Equality-based requirement**

- والطريقه دي اللي احنا استخدمناها دلوقتي ان انا استخدم ال equal اقوله مثلا owner = mohamedatef او استخدم ال non equal زي مثلا owner != mohamedatef

```
environment = production
tier != frontend
```

2- تاني طريقه اسمها **Set-based requirement**

- زي مثلا ان انا هقوله ان ال key هو ال environment وبعدين in 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 مابين single quotes هضيف ال labels عن طريق اني هقوله مثلا ال key هو release وبعدين in وبعدين بين قوسين هضيف ال key زي (dev) او انا عندي اكتر من label بنفس ال key بس ال value مختلفه هقوله مثلا (dev, prod) وهكذا

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

- وممكن كمان استخم condition

```
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 اانا 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
```

- هنعمل apply لل webserver

```
mohamed@MohamedAtef:~$ kubectl apply -f webserver.yml
pod/webserver created
```

- لو جينا شوفنا ال Labels هنلاقيه ضافلي كل ال Lables الخاصه بال webserver

```
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 lables

```
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

- ال Deployments بتمثل مجموعه من ال identical Pods او ال Pods اللي متطابقه بشكل 100% وال Deployment من خلاله بيقدر يوظف replica علشان يقدر بشكل اوتوماتيك ي replace اي instance حصلها 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 اوتوماتيك
- 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 وصلنلها ولا لا

➢ علشان اعمل Create لل Deployment هتسخدم **kubectl create deployment** وبعدين بحددله اسم ال deployment وليكن هسميه nginx وبعد كده بحددله ال Image اللي هجيب منها ال Pod او ال container وليكن مثلا ال image اللي هنستخدمها هي nginx عن طريق اني هقوله --image=nginx

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

➢ هنشوف ال pods اللي عندي بعد اما عملنا create لل Deployment

```
mohamed@MohamedAtef:~$ kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/nginx-bf5d5cf98-6zdzz    1/1     Running   0          84s
NAME                         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes           ClusterIP   10.96.0.1    <none>        443/TCP   8d
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx        1/1     1            1           84s
NAME                         DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-bf5d5cf98   1         1         1       84s
```

- هنلاحظ انه عندنا **deployment.apps/nginx** عمل create ل deployment واسمه nginx وهنلاقي كمان فيه عندنا **replicaset.apps/nginx-bf5d5cf98** عمل create ل replicaset اسمها nginx وحط جنبها رقم اللي هو ده bf5d5cf98 هنلاقي كمان فيه **pod/nginx-bf5d5cf98-6zdzz** عمل create ل pod باسم nginx وهنلاقي جنبه الرقم بتاع ال replicaset اللي هو ده bf5d5cf98 وحاطط جنب الرقم ده رقم تاني zdzz6 لما نيجي نعدل بعد كده ونقوله ان احنا عايزين اكتر من pod في ال replicaset هنلاقي ان الرقم ده bf5d5cf98 متشابه والرقم ده zdzz6 بيتغير
- ف ال command اللي احنا استخدمناه عمل create ل 3 حاجات عمل create لل pod و ال replicaset وال deployment
- 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=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:               app=nginx
  Containers:
```

هنلاقي هنا بعض المعلومات زي الاسم وال namespace والوقت وال Labels وال Selector وال Replicas وهكذا

- هشنوف ال pods اللي عندنا

```
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 **kubectl scale deployment** وبعدين احددله اسم ال deployment وبعدين استخدم اوبشن **--replicas=** وبعدين اضيف عدد ال replica

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

- لو عملنا تاني kubectl get pods هنلاقيه عمل create ل 7 pods

```
mohamed@MohamedAtef:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-bf5d5cf98-2j6cp    1/1     Running   0          81s
nginx-bf5d5cf98-6zdzz    1/1     Running   0          172m
nginx-bf5d5cf98-c7rnf    1/1     Running   0          81s
nginx-bf5d5cf98-kxtbp    1/1     Running   0          81s
nginx-bf5d5cf98-qnctj    1/1     Running   0          81s
nginx-bf5d5cf98-rpgdd    1/1     Running   0          81s
nginx-bf5d5cf98-w8qkz    1/1     Running   0          81s
```

هنلاقيه هنا عمل create ل **7 pods** وهنلاحظ ان اسم ال pods زي بعضه ماعدا الجزء الاخير من الارقام زي ماقولنا قبل كده

ده الجزء الخاص بال deployment

ده الجزء الخاص بال replicaset

وده الجزء اللي بيتغير كل اما اعمل replicaset

➢ لو انا عايز اعرف ال IP بتاعت ال pods دي

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

➢ انا عندي 7 pods لو انا عايز اقلل ال pods دي ان انا اعمل scaled up

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

➢ لو عملنا تاني kubectl get pods هنلاقيه قلل 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

```
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
```

هنلاقيه في ال Replicas بيقولك ان فيه 5 desired

هنلاقيه هنا في ال Events بيقولك ان ال حصل scaled down من 7 ل 5

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 وهكتب فيه الاتي
- اول حاجه هحددله ال apiVersion بتاعت ال deployment وهي apps/v1

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webserver
spec:
  replicas: 7
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web-
    spec:
      containers:
        - name: web
          image: nginx
          ports:
            - containerPort: 80
```

اول حاجه هحددله ال apiVersion بتاعت ال deployment وهي apps/v1

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

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

تحت ال metadata بحددله الاسم بتاع ال deployment وهو webserver

بعد كده بحددله ال spec تفاصيل ال deployment او ايه هي ال state اللي انت عايزها لل object

تحت spec بحددله ال replicas بحددله عدد ال replica اللي انا عايزها ف انا عايز 7 replica مثلا

في ال Selector بحددله ال Pod اللي هبتدي اعمله عدد سبعه replicas ف ف حددنهاله web

بعد كده في ال template بتاعت ال pod عملناله assign ل labels اللي اسمه app:web

بعد كده حددناله ال container وهيكون اسمه web وال image هتكون nginx وال port هيكون 80

- هنعمل apply لل deployment-file.yml

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

- هنعمل list لل pods اللي عملنا ليها create عن طريق ال ReplicaSet

```
mohamed@MohamedAtef:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
webserver-587f6fb7c9-b2m72   1/1     Running   0          19m
webserver-587f6fb7c9-d4r6s   1/1     Running   0          19m
webserver-587f6fb7c9-hmwbt   1/1     Running   0          19m
webserver-587f6fb7c9-nf7kv   1/1     Running   0          19m
webserver-587f6fb7c9-tqf2j   1/1     Running   0          19m
webserver-587f6fb7c9-xnrkl   1/1     Running   0          19m
webserver-587f6fb7c9-zkvq6   1/1     Running   0          19m
```

- لو مثلا 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 ازاي
- هنعمل delete لاي Pod

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

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

```
mohamed@MohamedAtef:~$ kubectl get pods
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 هنلاقيه عمل create واقدر اني استخدم ال pod دي

```
mohamed@MohamedAtef:~$ kubectl get pods
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 علي مصطلحين مهمين جدا وهما ال **Imperative** وال **Declarative**

- ال **Imperative** بكل بساطه هي عباره عن ال commands اللي بتكتبها في CLI زي مثلا kubectl وتديله مثلا command زي run او create او apply لل Kubernetes API زي ماحنا شوفنا قبل كده ال commands اللي احنا كتبناها واحنا بن create ال deployment كل ده اسمه Imperative ان انا بدي command واضح وصريح لل API علشان يعملي الحاجه اللي انا عايزها وليكن في المثال ده **kubectl create deployment nginx-dep --image=nginx** ان احنا بنعمل create ل deployment او انا بعمل Scale

- اما ال **Declarative** بكل بساطه هي عباره عن ال yml file ان احنا بنكتب configuration file ومن خلاله بحددله مثلا ال deployment وال replicas وال name وهكذا
- افضل طريقه اني استخدمها هي ال **Declarative** علشان ال Security

Services in Kubernetes - Kubernetes

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

- **ايه الفرق مابين 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 access ال application الموجود علي ال container جوه ال pod

انواع ال Services في ال Kubernetes

- اول نوع منهم واللي شفناه قبل كده وده ال default value هو ال **ClusterIP** ودي معناها ان ال Services 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 انت شغال عليه يعني لو انت عندك مجموعه من ال Pods عليهم nginx او اي حاجه حتي لو ابلكيش انت اللي عامله وعايز ت connect عليه وهو نوعه CluserIP هيبقي by default مش هيشتغل لان زي ماقولنا انك متقدرش تعمل requests لل Pods من بره ال cluster
- تاني نوع وهو ال **NodePord** النوع ده بيخلي ال service accessible تقدر تستخدمها on a static port لازم تحددله port معين لازم تشتغل عليه .معني كده ان ال Requests اللي جايه من خارج ال Cluster بتقدر تت handle ويطلعلك output من جوه ال Pods من خلال ال Services اللي هيكون نوعها NodePord
- تالت نوع وهو ال **LoadBalancer** و دي بتخلي ال service accessible بشكل خارجي من خلال اي Cloud Provider عنده **LoadBalancer** زي ال GCP وال AWS وال Azure وال Cloud هي ال هتعمل Create لل **LoadBalancer** واللي بشكل automatically هتعمل routes لل Request اللي بتجيله لل Kubernetes Services
- رابع نوع وهو ال **ExternalName** وده انت مبتحددلهوش اي Port ولا endpoints هو بيستخدم Alias ل External services علشان يوجهها وده 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 علي 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

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    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
```

- بعد كده هعمل apply لل apache-deployment

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

- لو عملنا list لل pods

```
mohamed@MohamedAtef:~$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/apache-deployment-776cb74cf6-c7hjt  1/1     Running   0          6m15s
pod/apache-deployment-776cb74cf6-pgf8x  1/1     Running   0          6m15s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   9d

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache-deployment   2/2     2            2           6m15s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-deployment-776cb74cf6  2         2         2       6m15s
```

هنلاقيه عمل create ل two pods علشان احنا في ال yml file احنا محددين ليه انه يعمل 2 من ال replicas

دي ال service بتاعت ال Kubernetes ونوعها ClusterIP زي مشوفنا في الشرح وال service دي تبقي موجوده by default

هنلاقيه كمان عمل create ل ال deployment باسم apache-deployment

هنلاقيه كمان عمل create ل ال replicaset

- هنعمل اللي احنا شوفناه ان احنا هن expose او نخلي ال Deployment دي ليها Service علشان نقدر ن access ال Server ال apache ده
- ف علشان نعمل كده هنستخدم command طويل شويه ان انا هستخدم **kubectl expose deployment** وبعد كده بحددله اسم ال deployment دي وهي ال apache-deployment وبعد كده بحددله ال name عن طريق اني استخدم =name-- وبعد كده بحددله نوع ال Service عن طريق =type-- وبعد كده بحددله ال Port عن طريق =port-- ,بعد كده بحددله ال Target Port

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

- لو عملنا List تاني هنلاقي ان عمل create ل service اسمها apache-service

```
mohamed@MohamedAtef:~$ kubectl get all
NAME                                        READY   STATUS    RESTARTS   AGE
pod/apache-deployment-776cb74cf6-c7hjt      1/1     Running   0          36m
pod/apache-deployment-776cb74cf6-pgf8x      1/1     Running   0          36m

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP    9d
service/ apache-service       ClusterIP   10.100.103.48   <none>        8080/TCP   4m10s

NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache-deployment      2/2     2            2           36m

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-deployment-776cb74cf6      2         2         2       36m
```

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

```
mohamed@MohamedAtef:~$ kubectl port-forward service/apache-service 8090:8080
Forwarding from 127.0.0.1:8090 -> 8080
Forwarding from [::1]:8090 -> 8080
```

ده ال IP اللي احنا هنستخدمه

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

It works!

- ممكن كمان اعمل access علي ال service عن طريق اننا نستخدم ال **minikube** ب command 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

```
mohamed@MohamedAtef:~$ minikube service apache-service
W0723 17:16:15.090362   32843 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
|-----------|----------------|-------------|--------------|
| NAMESPACE |      NAME      | TARGET PORT |     URL      |
|-----------|----------------|-------------|--------------|
| default   | apache-service |             | No node port |
|-----------|----------------|-------------|--------------|
😿  service default/apache-service has no node port
❗  Services [default/apache-service] have type "ClusterIP" not meant to be exposed, however for local development minikube allows you to access this !
🏃  Starting tunnel for service apache-service.
|-----------|----------------|-------------|------------------------|
| NAMESPACE |      NAME      | TARGET PORT |          URL           |
|-----------|----------------|-------------|------------------------|
| default   | apache-service |             | http://127.0.0.1:46463 |
|-----------|----------------|-------------|------------------------|
🎉  Opening service default/apache-service in default browser...
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
Xlib:  extension "NV-GLX" missing on display ":0".
libva error: /usr/lib/x86_64-linux-gnu/dri/iHD_drv_video.so init failed
[33064:33064:0723/171618.213468:ERROR:object_proxy.cc(576)] Failed to call method: org.freedesktop.ScreenSaver.GetActive: object_path= /org/freedesktop/ScreenSaver: org.freedesktop.DBus.Error.NotSupported: This method is not implemented
vdhcoapp is running successfully. This is not intended to be used directly from the command line. You should press Ctrl+C to exit. If your browser is unable to detect the coapp, run: "vdhcoapp install".
Created TensorFlow Lite XNNPACK delegate for CPU.
```

ده كده شكل ال **output** بعد اما نفذنا ال **command** باستخدام ال **minikube**

It works!

- **تاني مثال هنشتغل علي ال nginx**
- ده كده ال yml file اللي هنشتغل عليه في حاله ال nginx هونفس ال code اللي استخدمناه في حاله ال apache بس مع تغيير ال name في حاله ال metadata وال container وكمان غيرنا ال image

```yaml
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
```

- هنعمل apply لل nginx

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

- لو عملنا list علي ال deployment هنلاقيه بيقولنا ان فيه عندي ال nginx وال apache

```
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 هنعمل expose وبعدين نحددله ال deployment

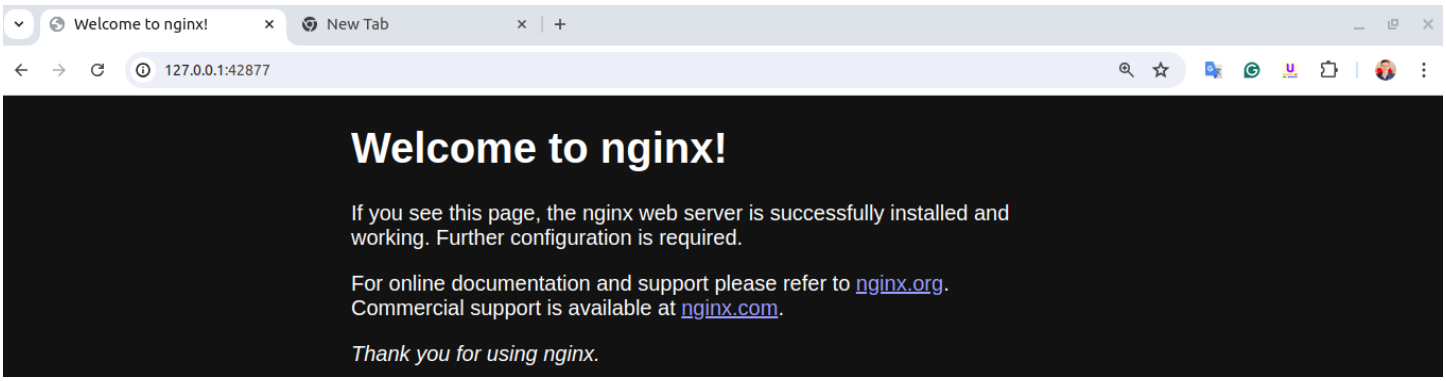
```
mohamed@MohamedAtef:~$ kubectl expose deployment nginx-deployment
service/nginx-deployment exposed
```

- لو علمنا list لل services هنلاحظ ان انا بقي عندي 3 services ومن 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

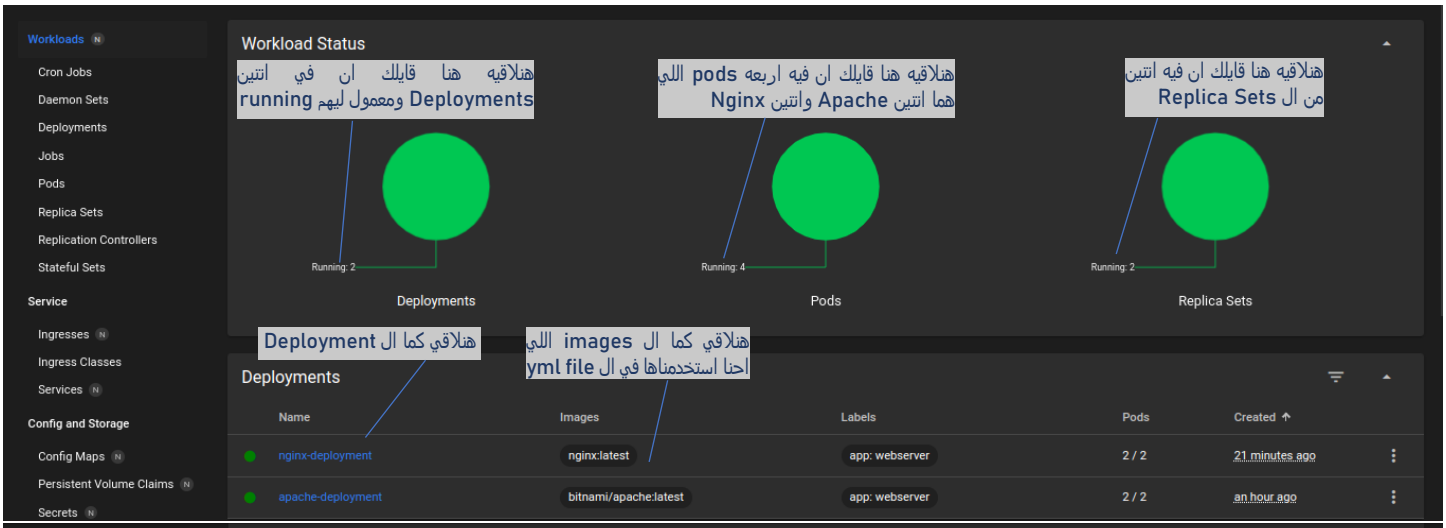
```
mohamed@MohamedAtef:~$ kubectl get services
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
apache-service      ClusterIP   10.106.139.92   <none>        8080/TCP   56m
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP    9d
nginx-deployment    ClusterIP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

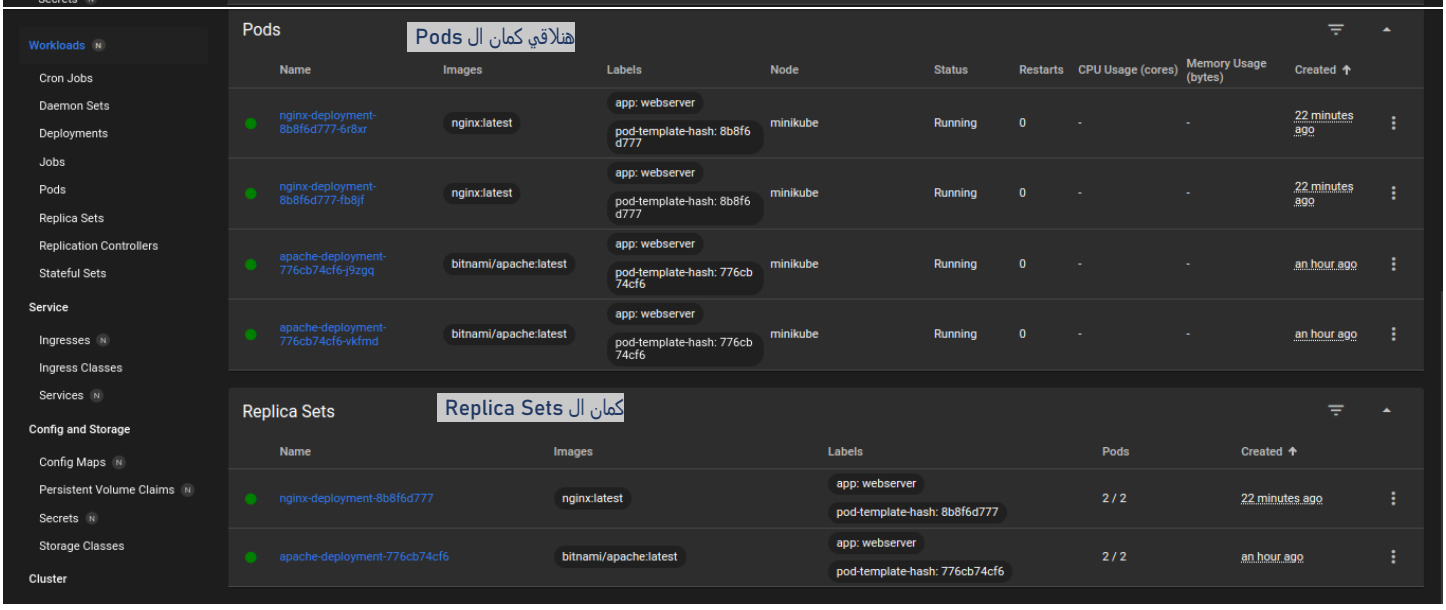
- ف هنشغل ال service دي عن طريق ال minikube وهيفتح علي طول علي ال browser ال default page بتاعت ال nginx

```
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

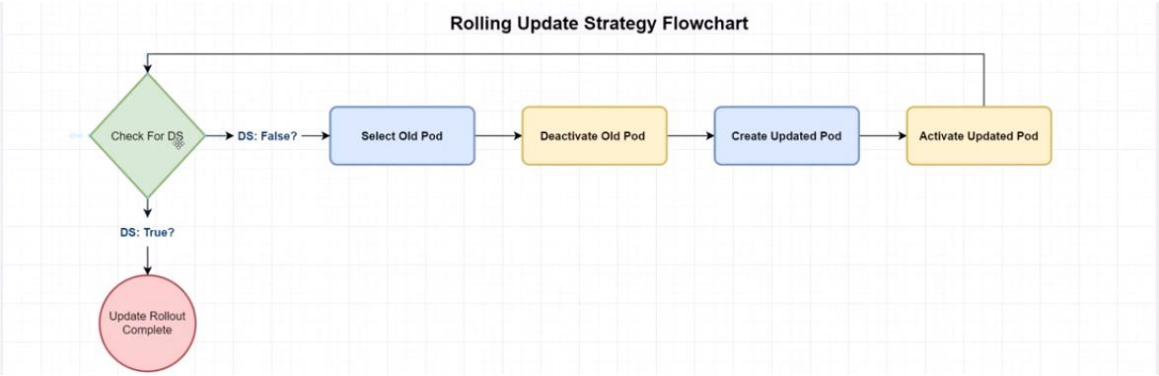
- هنتكلم عن احد اهم المفاهيم المهمه جدا في ال Kubernetes وهي ال **Deployment Strategies** اللي هو ازاي تعمل Update ل ابلكيشن شغال عندك من غير مايكون عندك Down Time كبير او ان يحصل انقطاع للخدمه بشكل ملحوظ للمستخدمين
- والهدف كمان من ال **Deployment Strategies** ان احنا نعمل automate لعمليه ال Updating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

1- اول نوع وهو ال **Rolling Update Deployment**

- ودي بتكون ال Default Deployment Strategies لان احنا زي ماكنا بنشوف واحنا بنعمل describe في الامثله اللي اخدناها كنا بنلاقيها موجوده جوه ال Deployment وان هي Rolling Update. وال mechanize اللي شغاله بيها ال Strategies دي ان هي بتعمل Replace لل pod علي حدا يعني بتاخد pod قديم بتعمله update للجديد وتروح علي 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

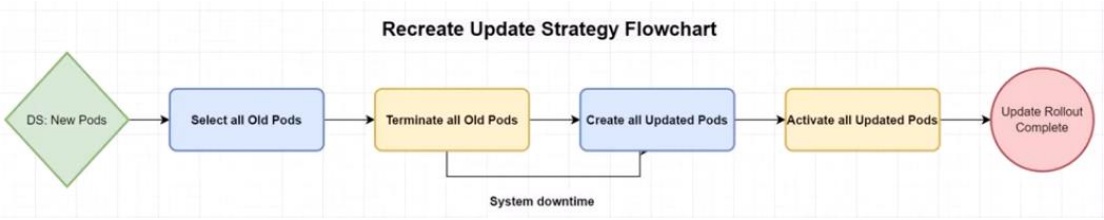
- فيه عندي مصطلحين مهمين جدا في ال Rolling Update Deployment وهما ال **maxSure** وال **maxUnavailable**

- ال **maxSure** ببساطه هي اللي بتحدد ال Maximum Number او Percentage من ال Pods فوق ال Specified number of replicas انت مثلا محدد في ال Yaml File او في ال Deployment بتاعتك ان انت عندك 4 replicas ف انت بقي ممكن تقوله في ال maxSure لما يجي يعمل Rolling update خليلي ال maxSure دي 25% ف يزودلك 25% من ال Pods او تقوله مثلا زودلي pod واحده او اتنين او تلاته وهكذا
- ال maxUnavailable ده ببساط بي declares ال Maximum Number او Percentage من ال Pods اللي بتكون unavailable غير متاحه في خلال عمليه ال Update



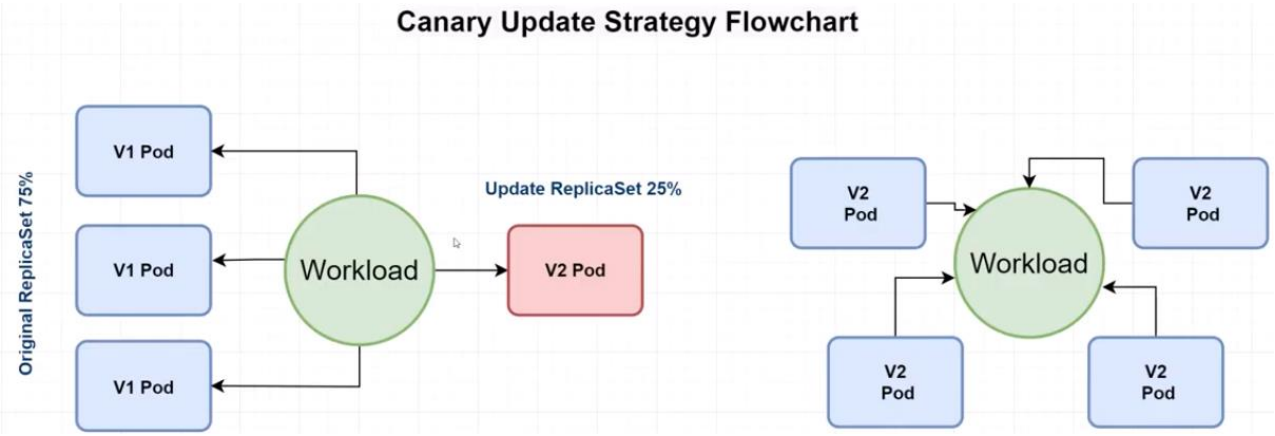
2- تاني نوع وهو ال **Recreate Update Deployment**

- النوع ده بيعمل Shut down لكل ال pods القديمه وليكن عندك 10 pods بتعملهم shut down وبعدين تعمل replace ل new pods
- ف ال Deployment بتعمل Select لكل ال outdated Pods ال 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 وهو ال **Canary Update Strategy**

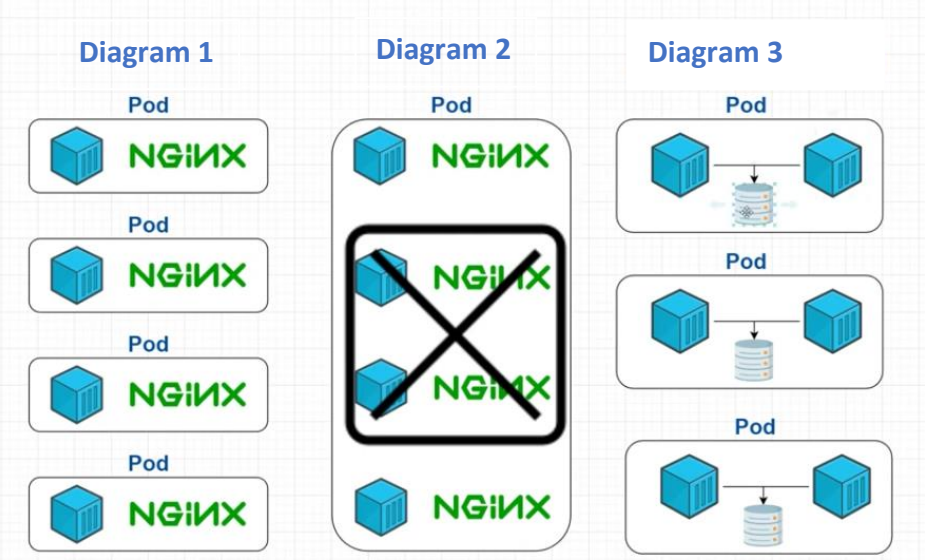
- وده عباره عن Partial update process او عمليه تطويربتسمحلك ان انت ت Test ال Version الجديد من البرنامج بتاعك من خلال ال Users نفسهم من غير الالتزام ان انت ت replace كل حاجه عند ال Users بمعني ان ال Deployment Strategy بت Create بعض ال Pods الجديده وبتخلي اغلب ال Cluster عندك لسه بال Pods القديمه وده غاليا بيكون الربع يعني بياخدوا الربع من ال Pods دي بياخدوا 25% من الحاجات الجديده ويبتدو يجربوها واغلب ال Users مش بيحسو باي اختلاف بيفضلو شغالين علي السيستم القديم وجزء بسيط من ال Users بتوع ال system من غير مايعرفوا بيكونوا بيجربوا الابلكيشن الجديد
- 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



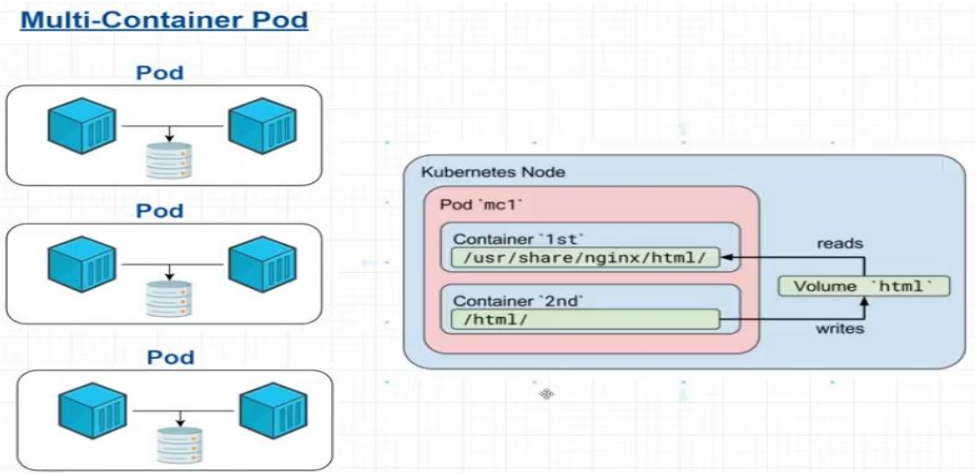
- ده كده ال Flowchart الخاص بال **Canary Update Strategy** هنلاقي عندنا workload انها شغاله علي سيستم وفيه عندي 3 pods شغالين علي V1 من ال Pod وبنسبه 75% من ال ReplicaSet وعلي الناحيه التانيه بعض من ال Workload بيمثل 25% هو اللي شغال علي V2 من ال pod او ال Update ReplicaSet زي ماقولنا لو مفيش مشاكل بتبتدي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

Multi-Container Pods and Containers Communication in Kubernetes

- هنشوف ازاي هنعمل create ل اكتر من container جوه ال Pod الواحد او بما يسمي بال **Multi-Container Pods** في ال Kubernetes



- **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 container فقط
- **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**
  - ده المفهوم الصح لل **Multi-Container Pod** بمعني ان يكون عندك pod فيها containers مختلفه images مختلفه وبت share نفس ال Resources يعني بت share نفس ال Storage ونفس ال IP Address وهكذا.
  - طب ايه اللي يخلينا نستخدم technic زي ده لان ممكن يكون فيه بعض التعقيدات لان ساعات بيكون عندك بعض ال Containers او ال Systems او ال Process اللي انت عايز تعملها محتاجه **Helper Prosses** او **Helper Container** علشان تقدر تشتغل زي مثلا يكون عندك اي Web Servers ومحتاج تعمله Log Server علي Container منفصل بحيث الاتنين يشتغلوا مع بعض او علي سبيل المثال WordPress محتاج يكون الابلكيشن علي Container و ال MySQL موجود علي Container تاني كل ده داخل نفس ال Pod ف الافضل دايما ان انت تخلي process واحده جوه ال Container علشان تتجنب ال Troubleshooting

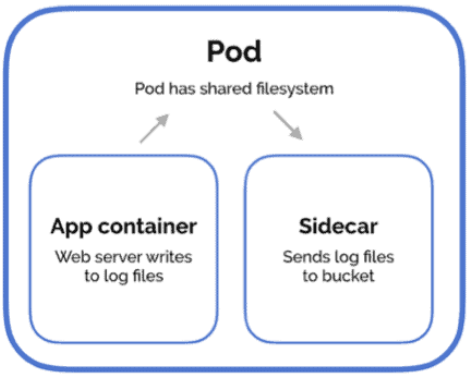


- هنلاقي فيه Area معمول ليها Shared مابين ال Two Container ف هنلاقي container رقم 1 وcontainer رقم 2 فيه Area بينهم وسيطه ف هنلاقي container رقم 2 بي write علي file اسمه html وال Volume اسمه html وال Container رقم 1 بيبتدي يقرا من ال Volume اللي فيها html ف انا اقدر اكتب في ال file اللي اسمه html في 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

1- Sidecar Design Pattern

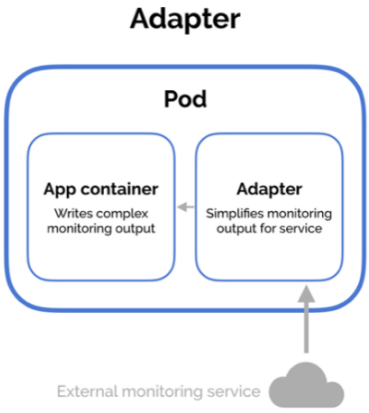
- ال **Sidecar Design Pattern** بيكون موجود فيه ال Main Application زي ال Web Application وبيكون موجود معاه ال **Helper Container** او ال **Helper Process** بيكون ليها مسئوليه مهمه بس مش اساسيه علشان تشغل الابلكيشن نفسه يعني ممكن الابلكيشن يشتغل عادي مفيش اي مشكله بدون ال Container التاني او بدون ال 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 ان انت ممكن يكون عندك ال Application Container بي writes مثلا log files ف بيروح لل shared Area وبعدين يروح علي ال Sidecar ال Container المساعد ويبدأ ي Send ال log files في ال bucket

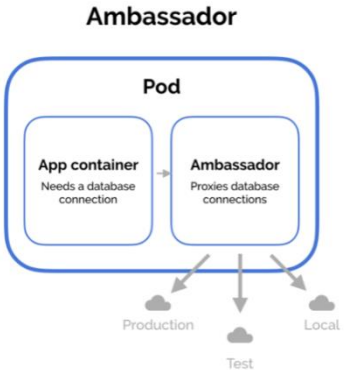
Adapter Design Pattern -2

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



Ambassador Design Pattern -3

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

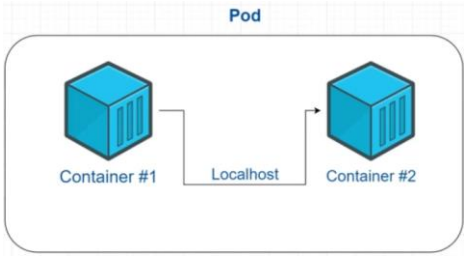


- لو شوفنا ال **Diagram** ده هنلاقيه بيقولك ان ال APP Container ده لو عايز database connection هيروح لل Ambassador او ال Helper Container علشان يعمل Proxy لل database connection علشان يقدر يطلعها علي Test او Local او Production

Communication Inside a Multi Consider Pod

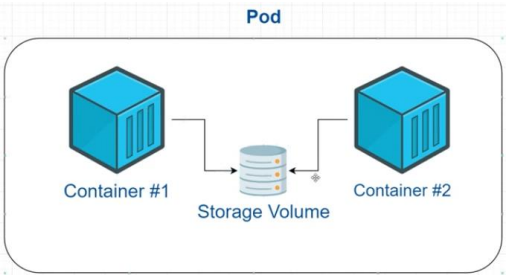
- عايزين نفهم ازاي ال Communication بيحصل بين ال Containers جوه ال Pods ف فيه عندنا 3 طرق رئيسيه بيقدر يحصل عمليه الاتصال بين ال Containers جوه ال Pod وهما ال **Shared Network Namespace** و **Shared Storage Volumes** و **Shared Process Namespace**

1- Shared Network Namespace



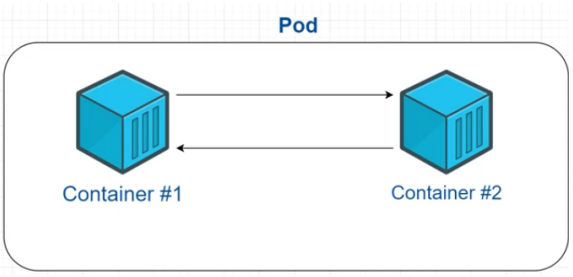
- ال **Shared Network Namespace** ده عباره عن ان كل ال Containers الموجوده في ال Pod بتتشارك في نفس ال Network Namespace لذالك كل ال Container تقدر تتواصل مع بعض علي ال Localhost يعني مثلا لو عندنا ال Container رقم واحد listing علي Port 8080 وال Container رقم 2 بي listing علي Port 8081 ف يقدر container 1 يكلم container 2 ب ان انت تكتبله Localhost:8080 او localhost:8081 وهكذا

2- Shared Storage Volumes



- 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

3- Shared Process Namespace



- ال Shared Process Namespace ده عباره ان احنا بيكون عندنا Flag او Attribute اسمه Shared Process Namespace موجود جوه ال Pod specification في ال Yml File بنقدر بس ان احنا نعمله enable ف بيبتدي Container 1 ي communicate مع Container 2 وهكذا والعكس

Deploy Multi-Container Pod in Kubernetes

- **هنشوف ازاي هنعمل Deploy ل Multi-container Pod في ال Kubernetes**
- اللي احنا هنعمله هنعمل Pod جواه اتنين container ال container الاولاني هيكون NGINX Web Server بيطلع ال Default Page بس هنغير ال Default Page من خلال ال Container التاني اللي هيكون عباره عن 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

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

اول حاجه هحددله ال apiVersion بتاعت ال Pod وهي v1

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

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 two-containers-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 اي سبب من الاسباب بيحصله exit ال Kubernetes بيعمل create ل واحد تاني 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 ال Pod حصله crash او انقفل 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 mountPath وال shared-area هيكون رايح علي ال Path ده /usr/share/nginx/html اللي هيكون جواه ال html page ال Default بتاعت ال nginx اللي اسمها .html

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

بالنسبه لل command انا عايز اديله command جوه ال Terminal هنا ف هيبتدي يفتحلي ال Terminal وابتدي انقل ال command ده echo Hello from Codographia container علي ال index.html في ال shared area اللي مابين ال Two Container 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 بتستخدمها لو انت عايز تبعت Argument او فيه Parameters معاك رايحه تتنفذ علي command جوه ال Container . فممكن تشبه ال args انها زي ال CMD Filed في ال Docker

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

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

- لو عملنا List لل Pods هنلاقيه ظاهرنا ان فيه two-containers-pod

```
mohamed@MohamedAtef:~$ kubectl get pod
NAME                READY   STATUS    RESTARTS   AGE
two-containers-pod  1/2     NotReady  0          117s
```

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

```
mohamed@MohamedAtef:~$ kubectl exec -it two-containers-pod -c nginx-container -- /bin/bash
root@two-containers-pod:/#
```

هنلاقيه هنا اهو بعد اما نفذت ال command دخل علي ال terminal بتاعت ال nginx ف اقدر اني انفذ اي command

- ف لو جينا نفذنا ال command ده curl 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
```

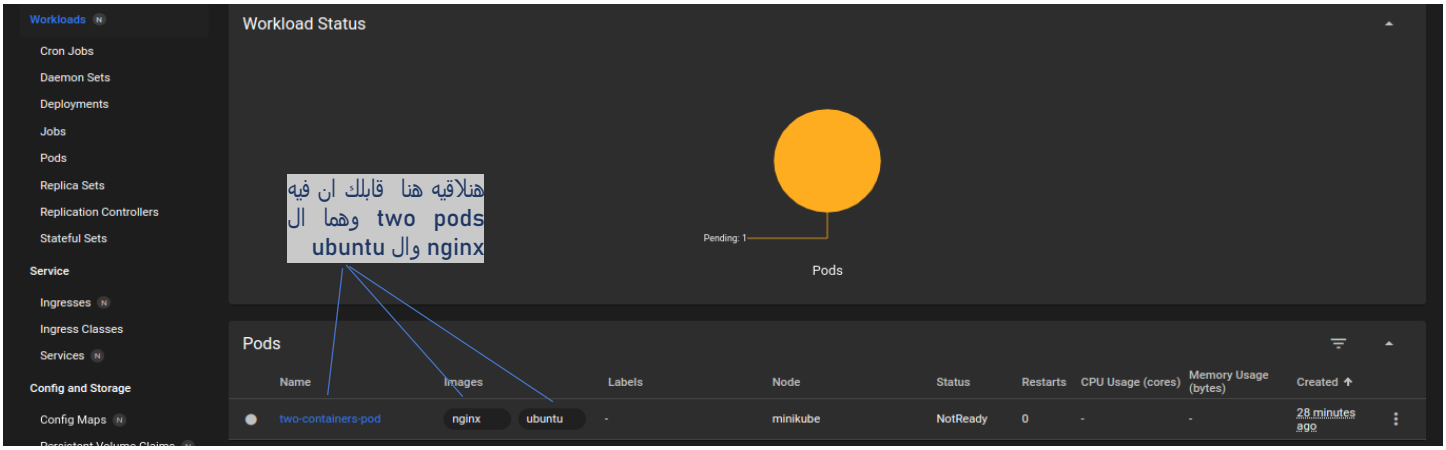
لما نفذنا ال command جوه ال nginx ظهرلنا ال message اللي كنا ضايفنها في ال Container التاني في ال args

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

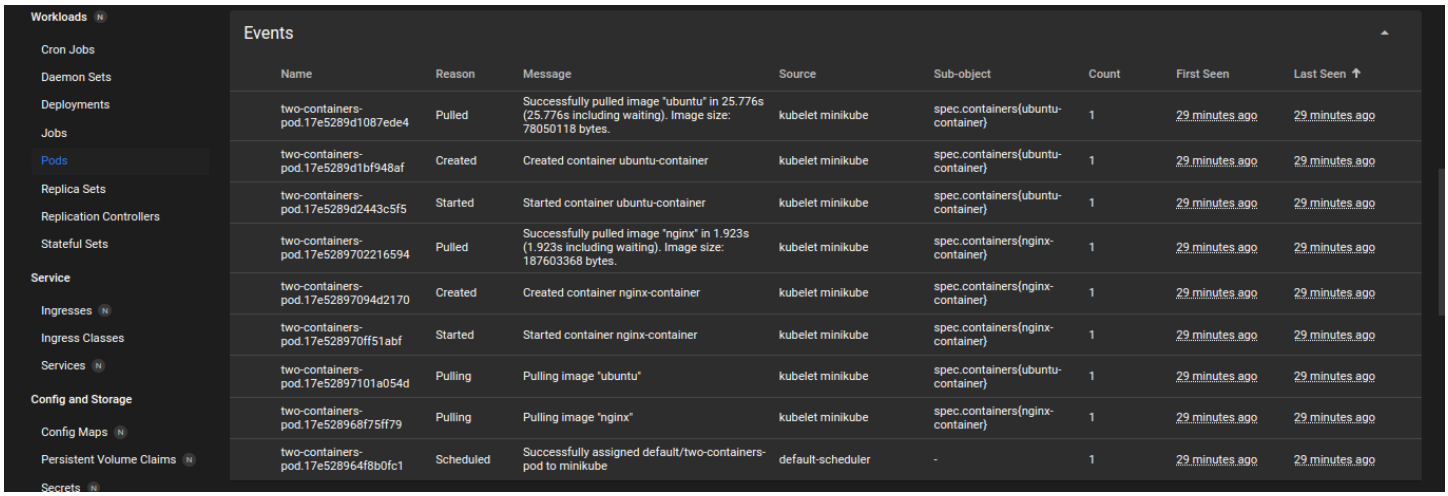
```
mohamed@MohamedAtef:~$ kubectl get pods two-containers-pod -o jsonpath='{.spec.containers[*].name}'
nginx-container  ubuntu-container
```

هتلاقيه قايلك ان فيه two container واحد اسمه nginx-container والتاني اسمه ubuntu-container

- لو فتحنا ال Dashboard وشوفنا الكلام ده هنلاقي

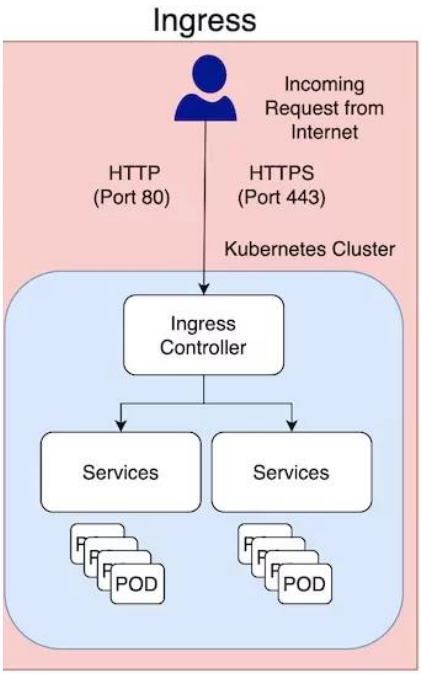


- ولو دخلنا جوه ال two-containers-pod في ال Dashboard زي مظاهر قدامنا في ال صوره وروحنا علي ال Events هنلاحظ المراحل اللي عدا بيها ال pod من اول اما عمل Pulling لحد اما اعمل Starte



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 بيسمحلك للدخول علي ال Kubernetes Services من خارج ال Clusters وبتقدر ت configure ال access ده عن طريق ان انت بت Create مجموعه من ال Rules هي ال بت defined وبتعرف ال connection ازاي توصل لل Services جوه ال Kubernetes 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** وال **Ingress**
- تعال نفتكر ايه هو ال **Services** ال Services عباره عن frontend للابلكيشن بتاعك اللي موجود جوه ال Cluster ال Services بتقوم بشكل automatically ان هي تعمل reroutes لل Traffic لل Pods المتاحه وبتعمل Distribution علي حسب ال Traffic زي ماشوفنا قبل كده ان احنا نقدر ن create اكتر من replicaset فيها اكتر من pod وتبدء توزع ال Traffic عليهم



- 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 80 او HTTPS علي port 443 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 Containers اللي عليها ال Application
  - يعني 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 Implementation نفسها بتاعت عمليه الدخول لل Services
  - لو افترضنا ال Ingress ده computer ف تقدر تقول ان ال Ingress Controller هو ال Programmer اللي بيعرف ال computer ياخد أ action ازاي ونقدر كمان نقول ان ال Ingress rules عباره عن المدير اللي بيقول لل Programmer يعمل الحاجه دي ازاي
  - ونقدر 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 Load Balancer
  - 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 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
```

➢ بعد كده هنعمل Scale

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

➢ لو عملنا List لل Pods هنلاقيه عمل اربعه container

```
mohamed@MohamedAtef:~$ kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginxservice-7999ff5dc4-lj9x6   1/1     Running   0          3m
pod/nginxservice-7999ff5dc4-llr8w   1/1     Running   0          8m43s
pod/nginxservice-7999ff5dc4-qxg2d   1/1     Running   0          5m38s
pod/nginxservice-7999ff5dc4-vkqrn   1/1     Running   0          5m38s

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP   10d

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginxservice   4/4     4            4           8m43s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/nginxservice-7999ff5dc4   4         4         4       8m43s
```

➢ هنعمل Create علشان ن expose Services

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

➢ لو عملنا get لل services هنلاقي فيه Services اسمها nginx ونوعها NodePort

```
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 ازاي ن Create ال Ingress ده ونتعامل معاه ون create rule بسيطه

➢ هنقوله kubectl create ingress وبعد كده بكتبله اسم ال Ingress واحنا هنسميه nginxservice-ingress وبعدين احددله rule عن طريق اني اقوله --rule= وبعد كده بقوله ايه هي ال rule دي ف هنقوله مابين double quotes ”/=nginxservice:80“

```
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
```

- بعد كده هنعمل SSH علي ال minikube

```
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 هنلاقيه انه زود nginxservice.io

```
docker@minikube:~$  cat /etc/hosts
127.0.0.1localhost
::1         localhost ip6-localhost ip6-loopback
fe00::0   ip6-localnet
ff00::0   ip6-mcastprefix
ff02::1   ip6-allnodes
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
```

- ف لو عملت curl nginxservice.io هيجبلي ال nginx page

```
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 Host وال Host ده Option ممكن تختاره او لا لو مبتحددش ال HOST ده ال Rule بت apply لكل ال HTTP Inbound Traffic اللي جايه علي ال IP Address اللي في الحاله بتاعتنا لو هنجرب بيكون ال minikube اما لو ال Host ده Provided ان احنا محددنهوله في ال Yaml file
- فيه list من المسارات زي مثلا ال /testpath او علي حسب انت هتسمي ال Path ده ايه كل واحده منهم لازم تكون بتسمع او مرتبطه ب backend متعرفه ب Service.name و service.port.name و service.port.number .وال Path اللي انت بتحدده لازم يتطابق مع المحتوي بتاع ال request اللي جيالله قبل ما ال Load Balancer يعمل redirect علي ال Traffic علي ال Services المناسبه 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 type path مطابق او موازي ليه وفيه 3 انواع من ال Paths دي وهما ال Implementation Specific وال Exact وال Prefix
  1- ال Implementation Specific بيكون ال Path بيتعمله تطابق بناءا علي ال Ingress Class
  2- ال Exact لازم ال URL Path اللي انت هتدهوله لازم ي matches ويكون case sensitivity
  - 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 matches وهيرجعلي ال Services ولكن لو هو مثلا ال Path ب /foo وال Request ب /bar مش هيعمل matches ومش هيرجعلي ال Services . حتي لو انا قولتله /foo وفي ال Request قولتله /foo/ مش 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

3- ال Prefix بي matches based علي ال URL Path Prefix ومنفصل ب / وال matching بيكون case sensitive وبيتم علي مستوي ال element يعني element by element

- يعني لو جينا نبص علي الامثله اللي قدامناا لما انت في ال Prefix قولتله /foo ف قالك ان /foo هتشتغل و /foo/ هتشتغل هي 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

- **ال Ingress Class**

- ال ingress Class بيحصلها implementation من different controllers وكل واحد من ال controllers بيجي ب different configuration وكل Ingress منهم بيقدر يحدد ال class اللي بتحتوي علي configuration زياده بخصوص ال Controller
- احنا عندنا اكتر من Ingress Controller اللي موجود by default لما نزلنا ال minikube هو ال Ingress Controller

- **ال IngressClass scope**

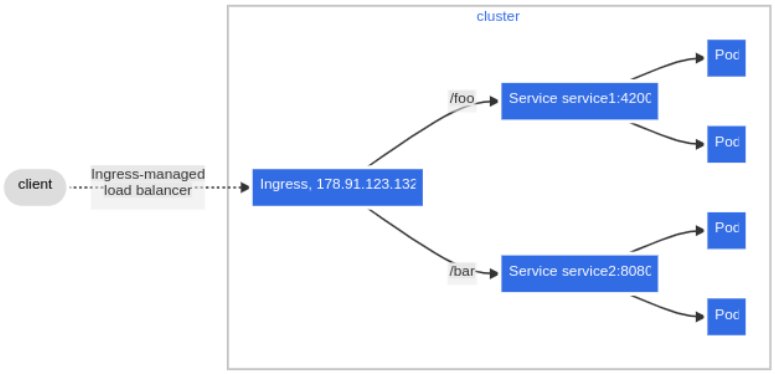
- انت ممكن تخلي ال Parameters بتاعتك متشافه علي مستوي ال cluster-wide او علي مستوي ال NameSpace

• Types of Ingress

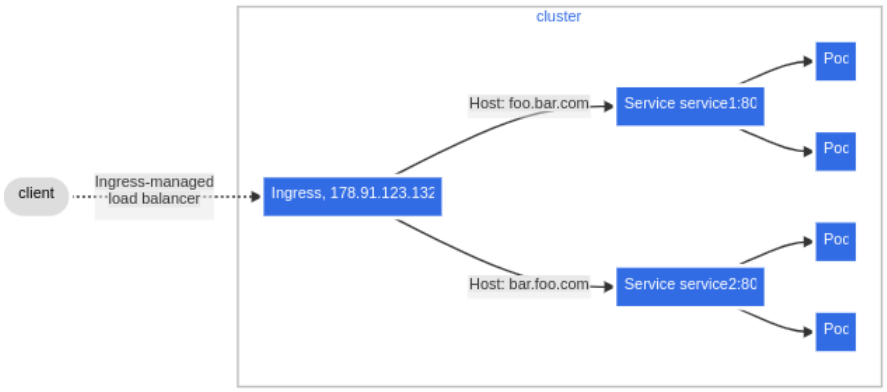
- في عندي انواع من ال **Ingress**

1- اول نوع اسمه **Single Service** ان انت عندك Ingress بيروح يشاور علي Service واحده اين كان ال service دي بتعمل ايه تديله مثلا اسم ال Service وال Port وبعدين تروح تشغلها سوا عملتلها curl من جوه ال SSH بتاعت ال minikube او روحت عملتلها turn on من ال minikube

2- تاني نوع اسمه **Simple fanout** وده بي Routes ال Traffic من Single IP Address to more than one service بناءا علي ال HTTP URL اللي احنا بندهوله يعني ال Client يقدر يدخل علي نفس ال Server او علي نفس ال IP Address يديله ال IP Address ومن خلال ال IP Address الواحد ممكن يكون عندنا اكتر من Service ب applications مختلفه تماما زي المثال اللي قدامنا ان ممكن اروح علي path اسمه /foo وده بيوديني علي service1 علي port 4200 وليها ال Pods بتاعتها او ممكن اروح علي path تاني اسمه /bar ب Service تانيه علي service2 مثلا علي port 8080 وليها ال Pods بتاعتها وبيكون جواها containers مختلفه عن ال service التانيه



3- تالت نوع وهو ال Name based virtual hosting النوع ده بيقدر ي support hosts مختلفه بمعني ان ال Ingress بيوزع ال Traffic بناءا علي ال host اللي متحدد علي ال Request من غير مايكون عندك اكتر من Server زي المثال اللي قدامنا ده ان انت ممكن تحط كذا host علي نفس ال IP وكل host يروح علي application ف هنلاقي ال host اللي اسمه foo.bar.com راح علي service1 علي port 80 علي application تاني وال host اللي اسمه bar.foo.com راح علي service2 علي port 80 علي application تاني



Kubernetes Storage Architecture

- ال **Kubernetes Storage** مبنيه علي ال Abstraction of volumes المقصود ب ال Abstraction of volumes ان ال Volumes تكون مجرده مش مرتبطه ب Pods او Nodes معينه وال volumes بتعرف بال basic entities بتقدر ال Containers جوه ال Pods انها ت Access ال Storage دي وتقدر انك تعملها mount بأختلاف ال Storage Infrastructure Types ممكن تبقي Local Storage او NFS وممكن تبقي Cloud Storage Service زي ال AWS وال Azure
- ال **Volumes** في ال Kubernetes تقدر تتحكم فيهم بطريقتين اول طريقه وهي ال Ephemeral او ال non-persistent والطريقه التانيه هي ال Persistent Storage

1- Non-Persistent Storage

- 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 بتبقي remove

2- Persistent Storage

- ال Kubernetes بي Support انواع كتيره جدا من ال Persistent Storage Models زي ال Files وال block storage وال object storage وال cloud services
- وال Storage تقدر تعملها referenced directly من ال Pod ولكن ده بي violates وبيخترق مبدء ال Portability بتاعت ال Pod ولكن ال Pod لازم تستخدم ال Persistent Volumes و Persistent Volumes Claims او ز ي ماهتشوفيه بعد كده بال PV و ال PVC بيستخدموا الحاجتين دول علشان يعرفوا ال Storage requirements of their application ان هما يرفوا احتياجات الابلكيشن بتاعهم من ال Storage المختلفه
- ال **Persistent Volumes(PV)** عباره عن قطعه من ال Storage جوه ال Clusters ال Administrator هو اللي بيوفرها وهي resource في ال Clusters زيها زي ال Node. وال PV عباره عن volume بتتوصل وعندها lifecycle مستقله بشكل تام عن ال Pod اللي بيستخدم ال Persistent Volumes دي .يعني احنا بنعمل create ل object وبنوصل ال Pod بال PV 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 لل Storage جاي من ال User وزي ما ال Pod بيستخدم ال resources بتاعت ال Node ال PVC بي consume ال PV resources. يعني زي ما ال Pod بي request specific levels من ال resources زي مثلا نقدر نحدد ال Pod ده هياخد قد ايه من ال CPU وال Memory . ال Claims نفس الكلام تقدر ت request specific size حجم معين و access modes زي مثلا ReadWriteOnce و ReadOnlyMany 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 Assign Volume ل Pod علشان يشتغل ك Storage لل Pod
- هنعمل create ل yaml file هن create منه ل Pod موجود فيها two containers ل Image اسمها ubuntu

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: vols-demo
spec:
  containers:
    - name: ubuntu1
      image: ubuntu
      command:
        - sleep
        - "3600"
      volumeMounts:
        - name: vol
          mountPath: /ubuntu1
    - name: ubuntu2
      image: ubuntu
      command:
        - sleep
        - "3600"
      volumeMounts:
        - name: vol
          mountPath: /ubuntu2
  volumes:
    - name: vol
      emptyDir: {}
```

اول حاجه حددنا ال apiVersion وهو v1 بعد كده حددنا ال kind اللي احنا هنعمله create وهو ال Po بعد كده في ال metadata حددنا اسم ال pod اللي هنعلمها create وهنسميها مثلا vols-demo

بعد كده في ال spec هنكتب ال containers ف هنكت اول حاجه ال container الاولاني وهيكون اسمه ubuntu1 وال image اللي هيستخدمها هتكون ubuntu وهيعمل run ل command انه يعمل sleep لمده 3600 ثانيه وال volumeMounts بتاعه هيكون اسمه vol وال mountPath اللي هيكون جوه ال container هيكون في المسار د /etc/secret-volume

بعد كده هحددله ال container التاني وهيكون نفس ال container الاولاني بس باختلافي اسم ال container هيكون ubuntu2 وال mountPath هيكون /ubuntu2 وال two container بيشاورو علي نفس ال volume

- هنعمل apply لل Pod

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

- لو علمنا get لل Pod هنلاقيه عمل create ل pod اسمها

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

➢ لو عملنا describe لل pod

```
mohamed@MohamedAtef:~$ kubectl describe pod vols-demo
Name:             vols-demo
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Mon, 29 Jul 2024 21:46:45 +0300
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.0.115
IPs:
  IP:  10.244.0.115
Containers:
  ubuntu1:
    Container ID:  docker://53030eef7f732f90b3dbbfd47e13a76f1a19919e2393b47871e0bc460e960d3a
    Image:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      3600
    State:          Running
      Started:      Mon, 29 Jul 2024 21:46:48 +0300
    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:         ubuntu
    Image ID:      docker-pullable://ubuntu@sha256:2e863c44b718727c860746568e1d54afd13b2fa71b160f5cd9058fc436217b30
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      3600
    State:          Running
      Started:      Mon, 29 Jul 2024 21:46:50 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /ubuntu2 from vol (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-cvnhg (ro)
```

هنلاقي هنا اهو اسم ال pod وهو vols-demo واتعمله creation امته وال status بتاعه وال ip اللي شغال عليه

هنلاقي هنا تفاصيل عن ال containers ف هنا تفاصيل عن اول container وهو ubuntu1

هنا ال mounts بتاعت ubuntu1 موجوده في /ubuntu1

هنا تفاصيل عن ال container التاني اللي اسمه ubuntu2

ومعمول ليها mount تحت /ubuntu2

➢

- هنعمل create ل folder مثلا جوه ال container الاولاني اللي اسمه ubuntu1 وهنلاقيه ظهر في ال container التانيه بالظبط ف معني ذلك ان هما sharing مع بعض

- ف هنعمل create لملف اسمه NewFile جوه ال container اللي اسمه ubuntu1 ف استخدمنا ال command ده ان انا قولتله kubectl exec -it وبعدين حددتله اسم ال Pod وهي vols-demo وبعدين اسنخدمت اوبشن -c علشان اكتب command وبعين حددتله اسم ال container وهو ubuntu1 وبعدين قولتله – وبعدين اكتب ال command اللي انا عايز انفذذه ف علشان احنا هنعمل ملف استخدمنا command اسمه touch وبعدين حددنا ليه ال Path بتاع ال volume واسم الملف اللي هنعمله createاللي هو /ubunu1/NewFile

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

- ممكن استخدم نفس ال command ف ان انا اعمل List واشوف الملف الي احنا عملناه

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

- هنعمل create ل file تاني باسم MohamedAtef

```
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 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 Administrator بتاع السيستم هو اللي بيوفرلك مكان ال Storage ده وبيكون مستقل عن ال Pod

Access Modes

- عندي 4 انواع من ال Access Modes

1- ReadWriteOnce

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

2- ReadWriteMany

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

3- ReadOnlyMany

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

4- ReadWriteOncePod

- 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 وهنسميه مثلا pv.yaml وبعدين هنكتب فيه الاتي

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv-storage
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/tmp/data"
```

انا هنا حددت ال Kind هيكون PersistentVolume

وبعدين 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 ان انا 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 ال Path 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 الي عندي هستخدم

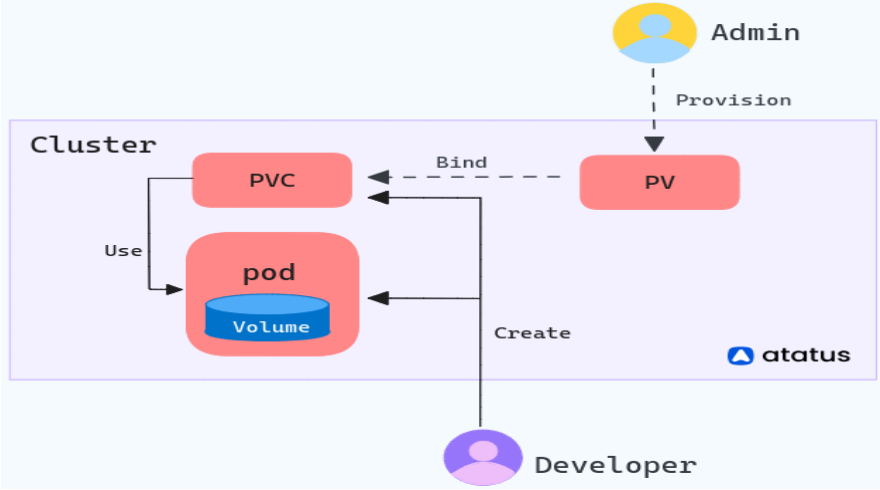
```
mohamed@MohamedAtef:~$ kubectl get pv
NAME         CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pv-storage   2Gi        RWO            Retain           Available           <unset>                                 3m25s
```

- لو شوفنا describe عن ال pv اللي عملنا ليه create

```
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:    RWO
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 **Persistent Volumes Claims (PVC)** هو عباره عن request لمساحه ال User عايز يخدها وبيتم تخصصها لل Pod وال Pods بيقدر ي request specific levels من ال resources تحديدا من ال CPU وال Memory وممن كمان يحدد specific size و access mode
- 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 ال Volume جوه ال Pod اللي ال developer عامله Create



- هنشوف ال yaml file الخاص بال PVC وازاي ن Create ال PVC
- هنعمل yaml وهنسميه مثلا pvc.yaml وبعدين هنكتب فيه الاتي

```yaml
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 يكون 1Gi

- هنعمل create لل PVC

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

- هعمل List للل PVC هنلاقيه عمل create لPVC وال name بتاعه pvc-storage وال status بتاعته اتعملها **Bound** ان هو اتربط ب pv وال Volume اللي اتربط بيه هو ده **pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3** هو عمل كده بشكل اتوماتيك انه ربط ال PV بال Volume وال capacity اللي احنا حددنهاله هي **1Gi** وال access modes اللي احنا حددناه وهو **RWO** وال storage class هيكون **standard** وده بيكون ال default 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-ae27-7f51c30977f3   1Gi        RWO            standard       <unset>                 112s
```

- لو احنا عملنا list للل Persistent Volume(PV) هنلاحظ ان هو عمل create ل **pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3** وهنلاحظ كمان ان ال RECLAM POLICY معمول Delete ومعني ده انonce ان ال PVC مبقاش يستخدم ال volume ده **pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f** ف ال Kubernetes هتروح تعمل delete للل PVC يعني ال PVC بيستخدم دلوقتي ال Volume اللي اسمه **pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3** اول لما مستخدمش ال PVC ده هيعمله Delete علي طول

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

- لو عملنا describe للل 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 create by
ده provisioner هو اللي بيوفر ال volumes ده او ال Path
علي الجهاز لو انت شغال علي الكلاود او منزل ال Kubernetes

- لو عملنا Delete ل PVC اللي اسمه pvc-storage

```
mohamed@MohamedAtef:~$ kubectl delete pvc pvc-storage
persistentvolumeclaim "pvc-storage" deleted
```

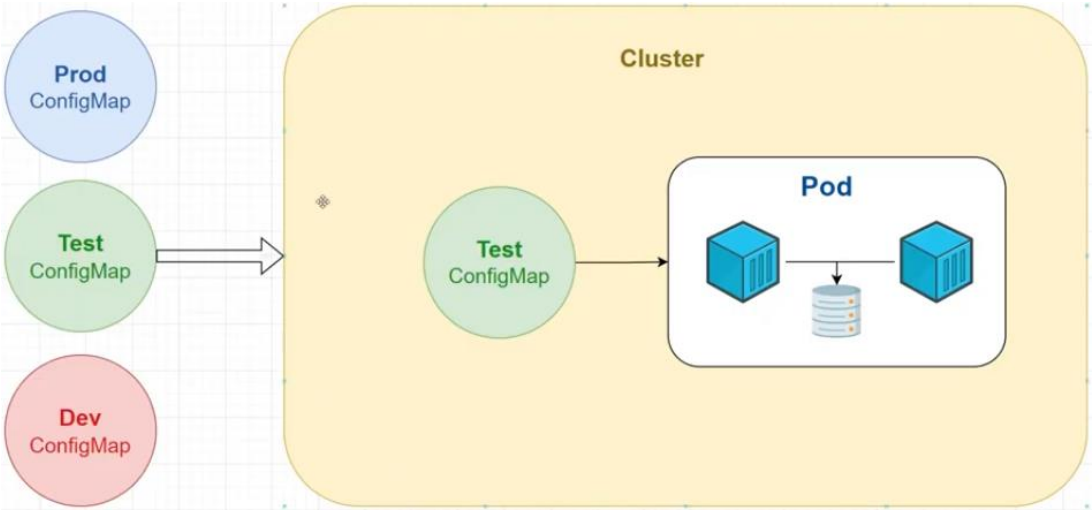
- وبعدين لو عملنا list للل pv هنلاقي ال **pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f** اللي اتعمل create by default اتعمله delete او اما عملت delete ل pvc-storage

```
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 running pods
- وال Config Maps تقريبا بتكون مناسبه لكل المواقف اللي انت محتاج تمد الابلكيشن بتاعك ب Configuration values داخل ال Pods زي مثلا ال IP Address وال Application database server او URL بتاع ال Proxy Services
- لو بصينا علي ال diagram ده هنلاقي ال Cluster الكبير اللي بيكون فيه كل حاجه او اغلب ال object في ال Kubernetes وعندنا Pod واحد جواه container بتقدر تمد الابلكيشن اللي جوه ال Pod ببعض ال Configuration Parameters حاجات متغيره بس هي مش related لل Application. اقدر يكون عندي اكتر من Configuration Map او Parameters او Files وهي مش مرتبطه ارتباط وثيق بال Pod فالي انت بتعمله بتعمل inject انك تقوله انا عايز اشغل ال Pod علي ال Test او اشغله علي ال Prod اوال Dev وجوه ال Production وال Test وال Dev بيكون فيه عدت Parameters واحد او اكتر من واحد فيهم مثلا ايبهات مختلفه او اسامي سيرفرات مختلفه او اين كان نوع ال Parameters



- هنعمل Create ل new file ال file ده هيكون ال Config Map بتاعي هنسميه مثلا db.properties مش لازم يكون ليه امتداد عادي لو سميته بس db انا مسميه زي ما موجود في ال Documentation بتاعت ال Kubernetes .بعد اما عملنا ال file هنبدء ان احنا نكتب ال Parameters او ال Configuration اللي احنا عايزينها جوه ال file وال configuration دي عباره عن key-value pairs ف هبدأ ان انا اقوله ان ال environment بتاعتك هتكون Production ف ال environment دي هي ال Key وال Production هي ال Value وممكن اقوله كمان ان ال databaseserver هتكون hrmsdb ودي اختصار ل Human Resource Management System Database . وممكن احطله كمان IP ف انا هديله ال Localhost اللي هو 172.0.0.1

```
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
- قبل اما نعمل create لل file فيه default config map موجوده اصلا لما بت install ال Kubernetes من خلال ال minikube ف لو قولتله kubectl get configmap هنلاقي config map اسمه kube-root-ca.crt ودي بيكون موجود فيها ال Certification اللي بتنزل by default لما بتيجي تنزل ال Kubernetes

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

- هنبدء ن create ال Config map ف هقولك kubectl create configmap وبعدين اديله اي اسم لل config map اللي هعمله create وليكن انا هسميه config-file وبعدين اقوله --from-file= هحددله هنا مكان ال file اللي انا كاتب فيه ال Parameters انا عندي ال file موجود تحت /hom/mohmes/

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

- لو عملنا list لل configmap هنلاقيه عمل create ل config-file 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
```

- او ممكن اختصر كلمه configmap ب cm زي

```
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 config 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>
```

- لو انا عايز اعمل delete ل config map

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

- هنعمل create ل config map من folder ف هيعملها create من مجموعه files ف هنعمل file اسمه وليكن mydatabasecred.conf وهنكتب فيها ال Parameters دي

```
MYSQL_USER=mohamed
MYSQL_ROOT_PASSWORD=password
```

- اللي احنا هنعمله بدل اما عملنا create جوه ال Kubernetes لل config map من file واحد ف احنا نقدر ن create من two files مختلفين جواهم environment variables مختلفين واسماء مختلفه ف انا عندي ال file اللي استخدمناه في المثال الاولاني اللي اسمه db.properties وال file التاني اللي لسه عاملينه اللي اسمه mydatabasecred.conf الملفين دول جوه folder اسمه ConfigMapDemo ف انا بدل اما احددله file واحد بس لا هحددله ال folder وهو هيعمل create لل config map لكل ال configuration file الموجوده في ال Folder

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

- لو عملنا list لل config maps هنلاقيه قايلك ان ال Name الجديد اسمه keys وعندي في ال DATA حاجتين

```
mohamed@MohamedAtef:~$ kubectl get configmap
NAME                 DATA   AGE
keys                 2      63s
kube-root-ca.crt     1      13d
```

- لو شوفنا التفاصيل بتاعت ال config map

```
mohamed@MohamedAtef:~$ kubectl get configmap keys -o yaml
apiVersion: v1
data:
  db.properties: |
    evn=prod
    databaseserver=hrmsdb
    ip=172.0.0.1
  mydatabasecred.conf: "MYSQL_USER=mohamed\nMYSQL_ROOT_PASSWORD=password    \n\n"
kind: ConfigMap
metadata:
  creationTimestamp: "2024-07-27T18:31:24Z"
  name: keys
  namespace: default
  resourceVersion: "129724"
  uid: 968d0d2d-ccfb-428c-8ac2-4e4bbbe8f428
```

هنلاقي هنا عندي حاجتين في ال data هنلاقي عندي الجزء الخاص بال db.properties والجزء الخاص ب mydatabasecred.conf

- احنا شوفنا كده اننا نقدر ن create ال Config Map من خلال file واحد او من خلال folder ممكن كمان ان احنا نعمل create لل Config Map 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:~$ 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 Password وهكذا
- ال Kubernetes Secrets عباره عن معلومات حساسه زي مثلا ال Login username وال Password وال Tokens 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 expose البيانات الحساسه لما تيجي تعمل deploy لل Application في ال Kubernetes
- من المعلومات المهمه ان ال secrets بيكون عندها size limit اللي هو 1MB وده مقصود ومتعمد ان ال size يكون مش اكبر من كده علشان متحتفظش فيه ببيانات اكبر من كده .ولما تيجي ت implement ال secrets عندك ف تقدر انك ت mount ال secrets دي علي شكل volumes او Environment Variables داخل كل Pod

Types of Secrets

- فيه عندنا انواع عديده من ال secrets في ال Kubernetes

1- Opaque

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

2- Service Account Token

- ده بيحتفظ بال token اللي بت identifying ال Service accounts بعد اما ال Pod يتعملها Create الKubernetes بشكل اتوماتيك بيروح ي create ال secrets ده ويعمله associates بيخليه يتعاون مع ال Pod علشان يقدر يوفر Secure Access لل API . والطريقه دي تقدر تعملها Disable

3- Basic Authentication

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

4- SSH authentication

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

5- TLS

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

6- Docker Config

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

7- BootStrap token

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

Applying Secrets in Kubernetes

- هنشوف ازاي نطبق مفهوم ال Secrets في ال Kubernetes
- علشان اعمل create ل Secrets هستخدم **kubectl create Secret** وبعدين هحدد نوع ال secret ده ف هستخدم ال generic اللي هو ال Opaque وبعد كده بديله اسم ال Secret وليكن هسميه test- Secret وبعد كده هستخدم ال literal في اني اديله ال Parameters بدل ال file

```
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 Secrets

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

- هنعمل list لل Secrets

```
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 username اللي اسمه admin

- 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 وهيظهرلك الباسورد او الاسم

Managing Secrets using Configuration File

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

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
stringData:
  server: database-server
```

حددتله ال apiVersions هتكون v1 وال Kind اللي هنستخدمه نوعه Secret وفي ال metadata حددتله اسم ال Secret اللي هنعمله وهو db-secret ونوع ال Secret هيكون Opaque

وهنا في ال data بتكتب ال secret نفسه اللي انت عايز تخبيه انا هنا مديله ال username وال password وبتدهمله في حاله encoded base64

لو انت عايز تديله ال username وال password ك string data هتستخدم ال type اللي اسمه stringData ف انا قولتله ان ال server هيكون database-server ف هو هيعمله encoded

- هنعمل apply لل db-secret.yaml

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

- لو عملنا list لل secret اللي عندي هنلاقي اتعمل لل create db-secret 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:
  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
```

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

هنلاقيه جايبلك 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

- ف هنعمل yaml file تاني وهنسميه db-secret-un.yaml وهيكون عباره عن

```yaml
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 اسمه config.yaml علشان اديله ال items اللي انا عايزها ف احنا هنا مدينله اسم قناه اليوتيوب ومديله كمان username و Password

- هنعمل apply لل db-secret-un.yaml

```
mohamed@MohamedAtef:~$ kubectl apply -f db-secret-un.yaml
secret/db-secret-un created
```

- لو عملنا Describe لل db-secret-un

```
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

- لو جينا شوفنا تفاصيل اكتر عن ال db-secret-un

```
mohamed@MohamedAtef:~$ kubectl get secret db-secret-un -o yaml
apiVersion: v1
data:
  config.yaml:
eXRjaGFubmVsOiAiaHR0cHM6Ly93d3cueW91dHViZS5jb20vYy9jb2RvZ3JhcGhpYSIKdXNlcm5hbWU6IGFkbWluCnBhc3N3b3JkOiBwYXNzd29yZAo=
kind: Secret
metadata:
  annotations:
    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
```

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

هنشوف ازاي هنعمل create ل Pod وال Pod ده هنديله ال Secret من خلال ال Configuration file

- هنعمل secret من ال Terminal من خلال ال Kubectl بدل اما نعمل yaml file

```
mohamed@MohamedAtef:~$ kubectl create secret generic server-user --from-literal=server-username='ca_admin'
secret/server-user created
```

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

```yaml
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 واسمه key وال server-username وده نفس ال server-username اللي موجود في ال secret اللي عملناه

- هنعمل apply لل Pod

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

- لو عملنا list لل Pod هنلاقيه عمل create لل pod بنفس الاسم env-server-user-secret

```
mohamed@MohamedAtef:~$ kubectl get pod
NAME                  READY   STATUS    RESTARTS   AGE
env-server-user-secret  1/1     Running   0          60s
```

- لو عملنا Describe لل Pod اللي عملناه اللي اسمه env-server-user-secret وروحنا في ال describe علي الجزء الخاص ب ال Environment هنلاقي ال SECRET_SERVERUSER اللي كنا كاتبينه في ال file الخاص بال Pod ف معني كده ان ال Pod ده اتا assign ليه ال secret بشكل سليم

```
mohamed@MohamedAtef:~$ kubectl describe pod env-server-user-secret
Environment:
    SECRET_SERVERUSER:  <set to the key 'server-username' in secret 'server-user'> 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 create ل secret بس بنوع مختلف عن ال Opaque

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

```yaml
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 Kubernetes.io/basic-auth وال basic-auth دا نوع من انواع ال secret وحددنا في ال stringData ال username وال Password لان ال Kubernetes بيكون طالب الاسم والباسورد

- هنعمل apply لل basic-auth.yaml

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

- لو عملنا get لل secret هنلاقيه عمل create ل secret جديد باسم secret-basic-auth ونوعه Kubernetes.io/basic-auth

```
mohamed@MohamedAtef:~$ kubectl get secret
NAME                TYPE                       DATA   AGE
db-secret           Opaque                     3      3h48m
db-secret-un        Opaque                     1      174m
secret-basic-auth   Kubernetes.io/basic-auth   2      94s
server-user         Opaque                     1      77m
```

هنشوف ازاي نعمل secret ون assign ال secret من خلال ال Volume

- اللي احنا هنعمله هنعمل create ل two files واحد لل secret وواحد لل volume
- اول حاجه هنعمل create ل secret file وليكن هسميه my-secret.yaml وهكتب فيه الاتي

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: vol-secret
data:
  username: YWRtaW4=
  password: bW9oYW1lZA==
```

- بعد كده هنعمل apply لل secret

```
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 ل create ل yaml file لل pod باسم مثلا pod-volume-secret.yaml وهنكتب فيه الاتي

```yaml
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 secret-vol-pod

وفي ال containers ادناله اسم secret-container وال Image اللي هنستخدمها هي nginx

وال Volume اللي احنا هنعمله mount هيكون اسمه secret-volume وال path اللي جوه ال container هيكون /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
```

- لو عملنا get هنلاقيه عمل create ل Pod جديد اسمه secret-vol-pod

```
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 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 الجزء الخاص بال Volumes هنلاقيه بيقولك ان فيه secret-volume وال type بتاعه secret وبيقولك ان ال volume معموله population بواسطه ال Secret وال SecretName اللي واصل بيه هو vol-secret وال optional 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 command ونشوف التفاصيل دي
- علشان اعملconnect 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 ls علي الpath اللي كنا محددينه في ال Volume Mount اللي هو /etc/secret-volume لو عملنا list للpath ده بستخدام ls command هنلاحظ ان موجود ال username وال 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 هنلاقيه قايلك ان ال username هو admin

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

- ولو في حاله ال Password هنلاقيه هنا قايلك ان ال password هي Mohamed

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

• ملاحظه

- لو انا عايز احول من encode ل decode ممكن استخدم ال command ده وليكن انا عايز كلمه Mohamed تتحول ل decode ف هو حول Mohamed ل bW9oYW1lZA==

```
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 نهايه ال command اوبشن -d

```
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 from the scratch من البدايه زي اول مره انت عملتها عامل زي ال Webserver ال Apache وال Nginx وعامل زي ال CDN(Content Delivery Network) او ال Printing Services
- ال **Stateful** عباره عن Applications بيحتفظ بالبيانات زي ال user profile وال preferences وال user actions وال Permissions تقدر تتخيل ان ال stateful application لازم يكون بي involve فيها نوع من انواع قواعد البيانات زي SQL Server او MySQL او MongoDB لان بيكون فيه read و write في عمليه ال stateful applications

What Is StatefulSets

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

Why Not PVC?

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

Components of a Kubernetes StatefulSet Configuration Manifest

- فيه 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 Clients تقدر ت Connect علي ال Pod بتاعنا باستخدام ال DNS entry

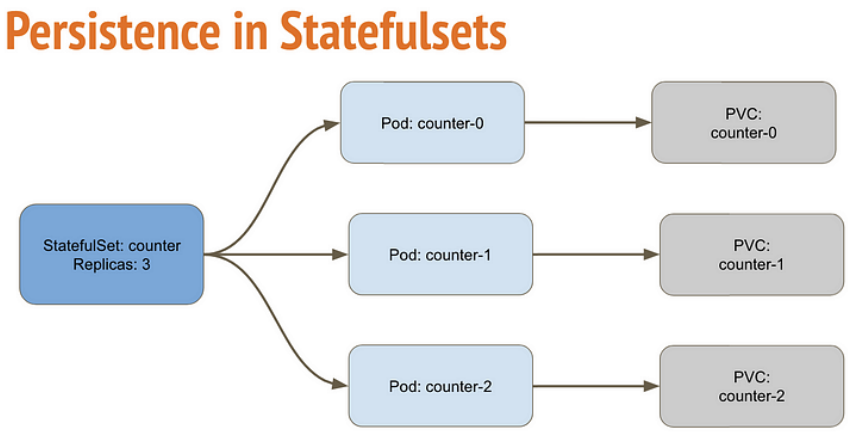
3- تالت حاجه ودي الاهم وهي ال **Volume Claim Template** ودي بتسمح لل Administrator ان هو يعمل Provision لل stateful storage باستخدام ال Persistent Volumes

StatefulSets VS Deployments in Kubernetes

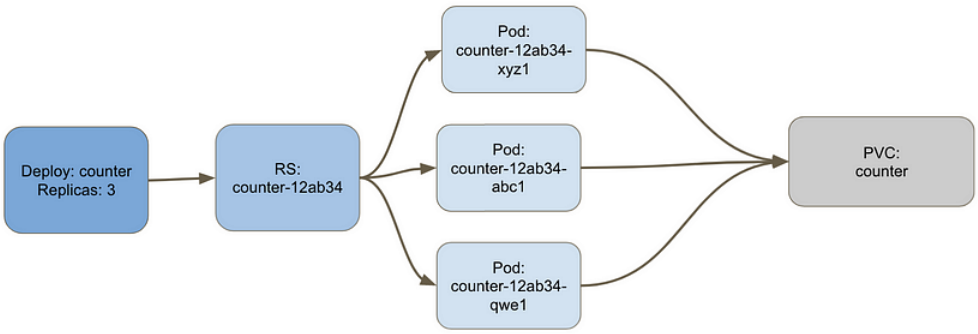
- هنلاقي ان ال StatefulSet شبه ال Deployments بس فيه اختلافات جوهريه مابينهم

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

- 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 و counter-2 وكل pod من ال pods دي عنده ال PVC الخاص بيه بالرقم بتاعه وده علي عكس ال Deployments



- في ال Deployment هنلاقي اسمه counter وال replicas بتكون 3 وال deployments بتعمل ل create ل replicaset (RS) وال replicaset هي اللي بتتحكم في ال Pods وهنلاقي الpod واخده جزء من اسم ال deployment وجزء randomly وكل ال Pods بي access او بي share نفس ال PVC



- **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 Service 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 ال StatefulSets مش شرط ان ال Pod ت terminate بال order اللي انت عملتله create بيها ودس بتمثل بعض ال risk
- واكيد اخدت بالك ان انت لازم manually ت create ال headless لل service علشان تقدر تستخدم ال StatefulSets في السيستم عندك

Applying StatefulSets in Kubernetes

- هنشوف حاجه عمليه عن ال StatefulSets وازاي نقدر نطبقها في ال Kubernetes
- ده كده configuration file خاص ب StatefulSets وزي مواضح هنعمل create ل Two objects وهما ال Service وال StatefulSets

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  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
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      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 nginx

- 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 serviceNameهتكون nginx
- وال replicas هتكون 3 واحنا قولنا ان ال replicas مش هتعمل ل create ل replicaset لا هي هتعمل ل 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 الجديده اللي متسخدمناش 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
```

- هنعمل apply لل yaml file اللي فيه ال Configuration بتاعتي اول اما ضغطنا enter قالك انه عمل create ل service اسمها nginx وعمل create ل staefulset.apps اسمه web

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

- لو عملنا kubectl get all هنلاقي اتعمل create لتلاته Pod واخد اسم ال staefulset اللي هو ال web وجنبها zero و 1 و 2 بالترتيب وعمل create ل service اسمها nginx

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

- وعمل create ل PV

```
mohamed@MohamedAtef:~$ kubectl get pvc
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS     CLAIM                 STORAGECLASS   VOLUMEATTRIBUTESCLASS
REASON   AGE
pv-storage                                 2Gi        RWO            Retain           Available                        <unset>                 31h
pvc-362e5aa5-1843-409e-af9e-899bb658eb1b   1Gi        RWO            Delete           Bound      default/www-web-0     standard       <unset>                 43m
pvc-9e5d3909-7939-4ddf-ae27-7f51c30977f3   1Gi        RWO            Delete           Released   default/pvc-storage   standard       <unset>                 28h
```

- 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 Kubernetes بيقدر بشكل Automatically يعمل Create ل Pod او اكتر .وهيفضل يعمل execution لل Pods دي لحد اما ي Specified Number او يوصلوا لعدد معين انهم يكونوا successfully terminate
- لما تعمل delete لل job هي لوحدها هتلغي ال Pods اللي هي عملتلها Create
- ولما تعمل suspending لل Job انت توقفها مش تلغيها ف هي هتعمل delete لل active pods الموجوده لحد اما ترجع تكمل عمل ال Job دي
- تقدر تستخدم ال Job ان انت تشغل اكتر من Pod علي التوازي

Parallel Execution for Jobs

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

1- Non-Parallel Jobs

- النوع ده مبيشتغلش علي التوازي بيشتغل علي التوالي و pod واحد بس اللي بيشتغل ولو ال Pod ده متعملهوش Fails ال Job هتعتبر انها Complete او اما ال Pod دي تكون Successfully بيعتبر ان ال Pod دي Terminate لان ال job دي بقت Complete

2- Parallel Jobs with a fixed completion count

- يعني Job متوازيه بتشتغل مع بعض ب fixed completion count

3- Parallel Jobs with a work queue

- ده ب Involve running multiple Jobs بشكل concurrently علي التوازي او مع بعض . وفي حالات كتيره مش افضل حاجه ان انت تسمح ل Job ان هي تنتهي قبل اما ت starting another one

CronJobs

- ال cron job هي هي ال job بس بتعمل running automation ل Task 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 generate timely emails او لو انت عايز ت schedule jobs بخصوص ال User-activity

kubernetes

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/