1. **Descarga los archivos del repositorio elegido.**

Repositorio Elegido - https://github.com/docker/awesome-compose/tree/master/aspnet-mssql

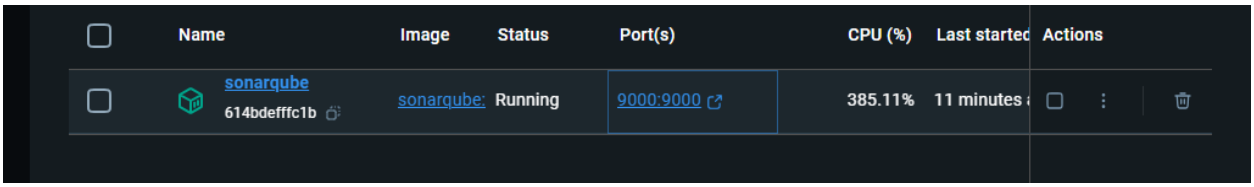
Enlace Git repositorio - https://github.com/rduverge/aspnet-mssql.git

Clonamos el repositorio en local.

2. **Analisis de Sonarqube**

Instalar instancia local de SonarQube desde imagen de Docker

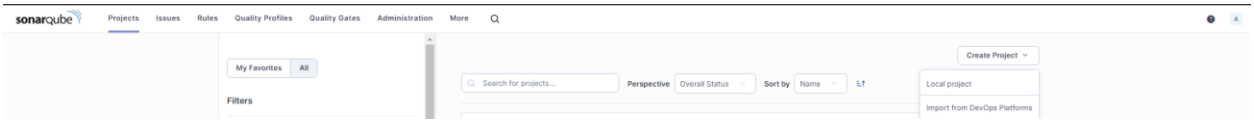
$ docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

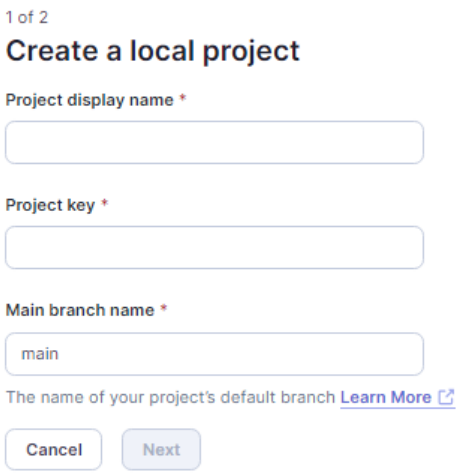


Lo tengo en un contenedor de Docker en el puerto 9000.

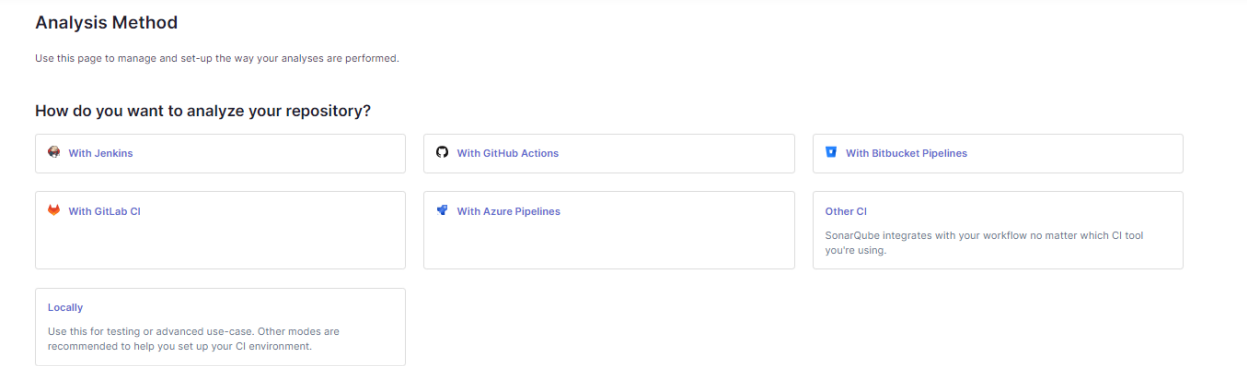
- login: admin
- password: admin

Creo un nuevo proyecto

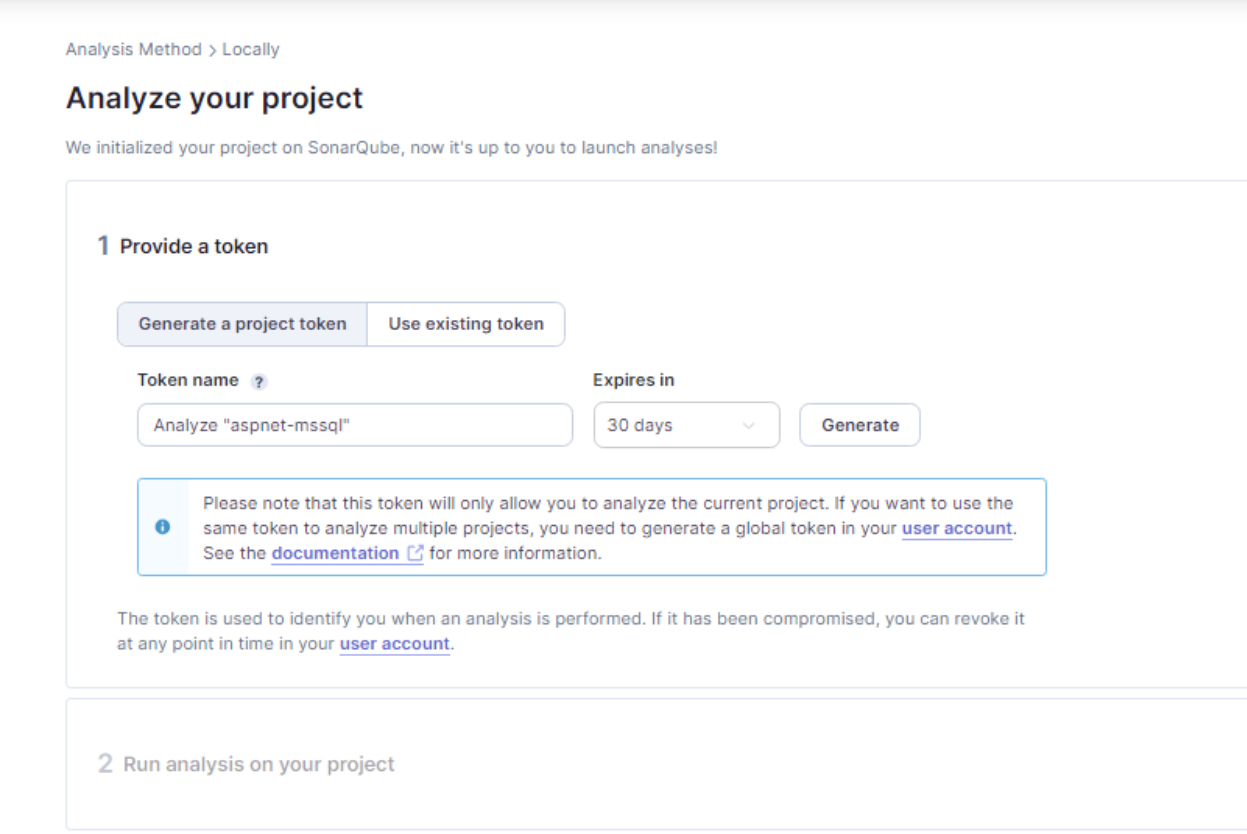




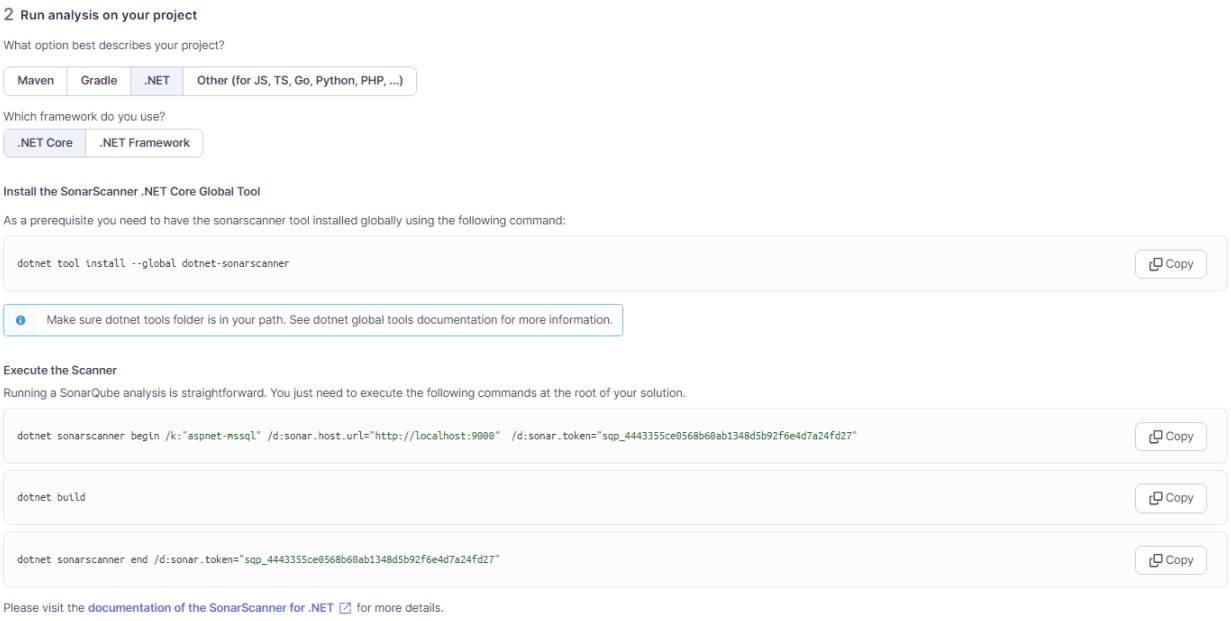
Selecciono un repositorio local



Genero un token para analizar el proyecto



Se selecciona el tipo de proyecto, este caso .NET



Install the SonarScanner .NET Core Global Tool

dotnet tool install --global dotnet-sonarscanner

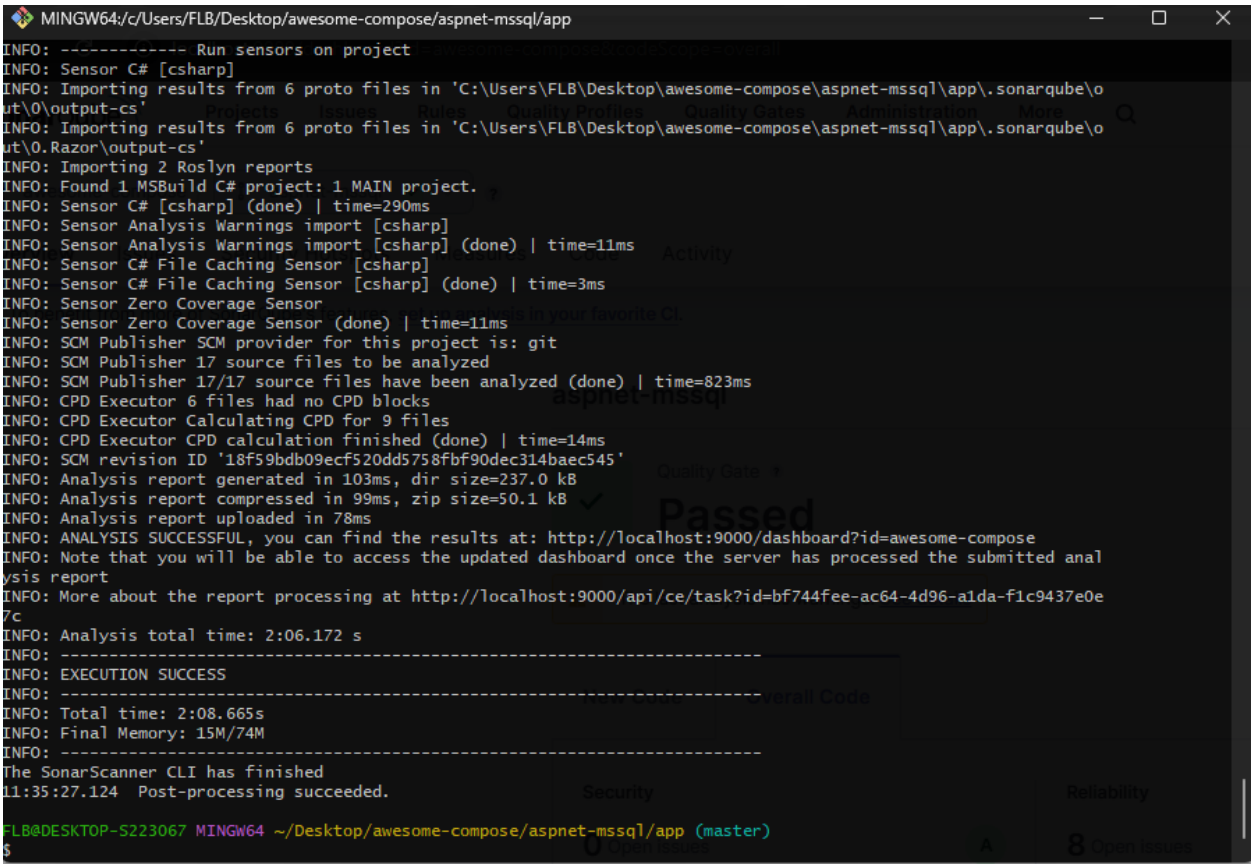
Execute the Scanner

dotnet sonarscanner begin -k:"aspnet-mssql" -d:sonar.host.url="http://localhost:9000" -d:sonar.token="{token}"

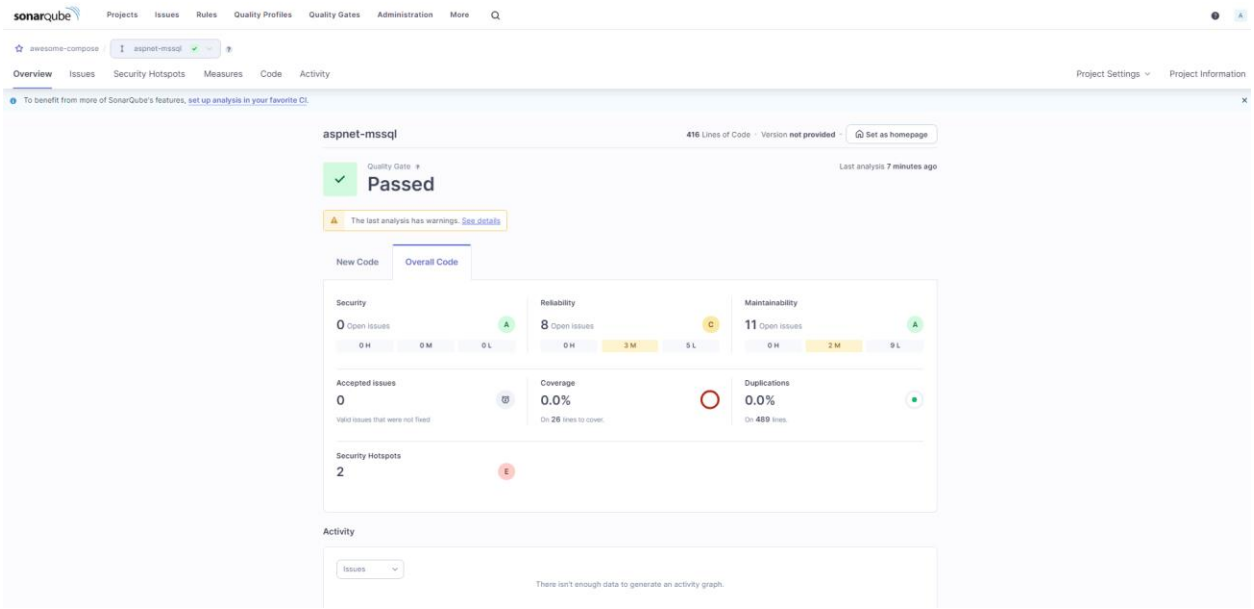
dotnet build

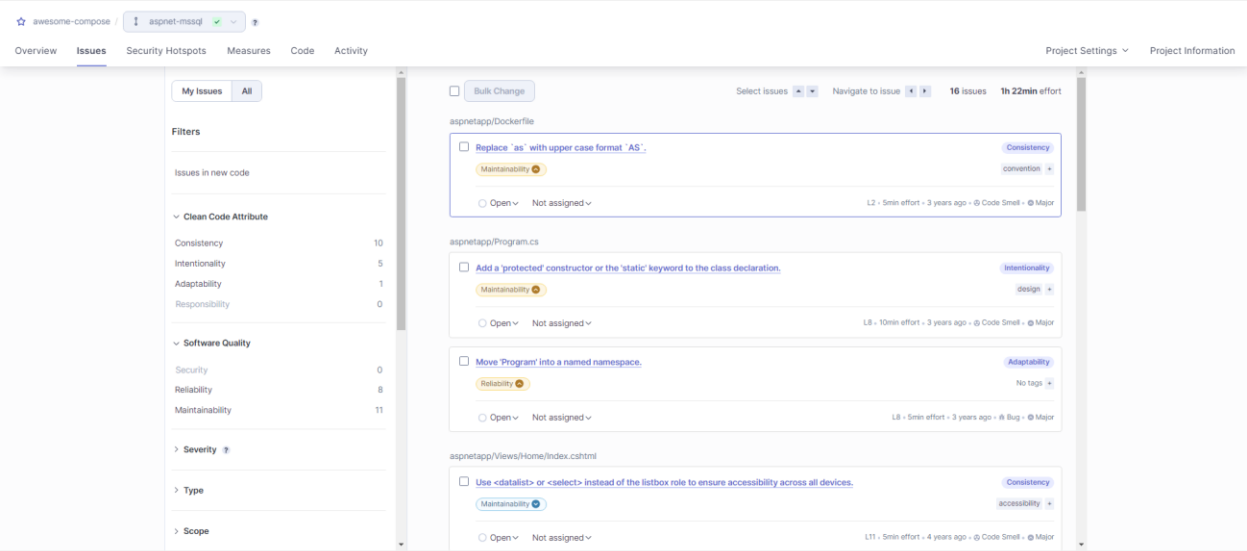
dotnet sonarscanner end /d:sonar.token="{token}"

Resultado de los comandos en consola

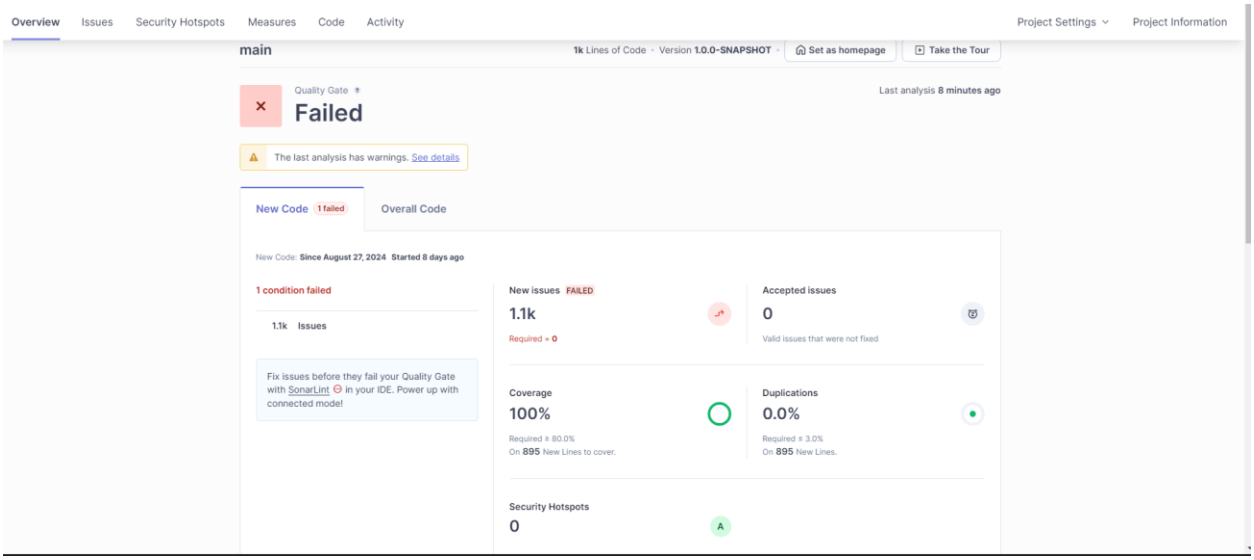


- **Resultado de Analisis exitoso**



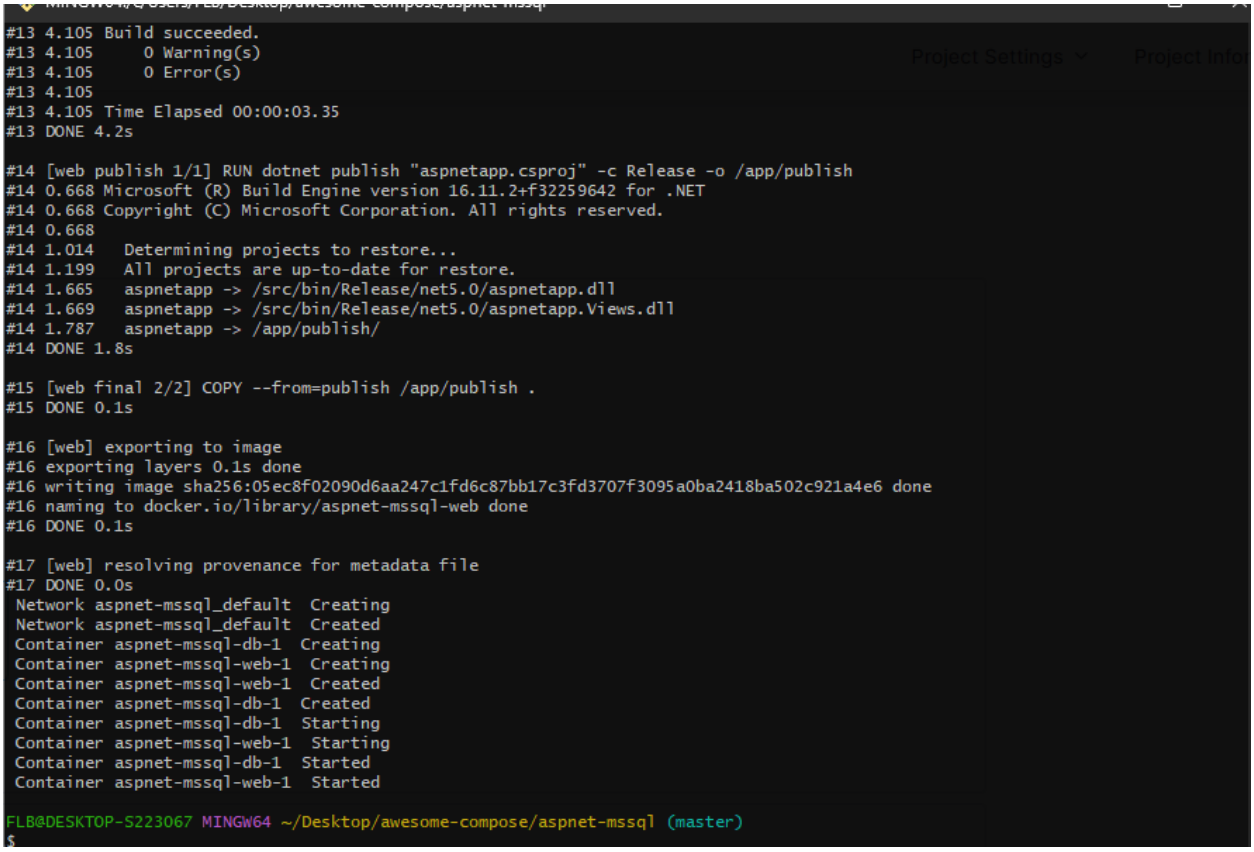


- **Resultado de Analisis Fallido**

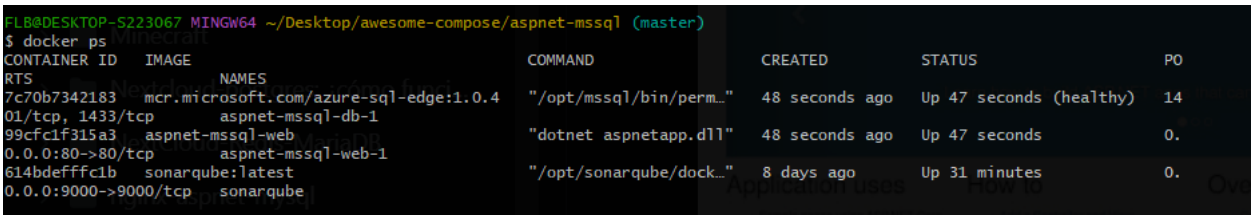


Compilar la aplicación

docker compose up -d



Docker ps



La aplicación está en el puerto http://localhost:80.

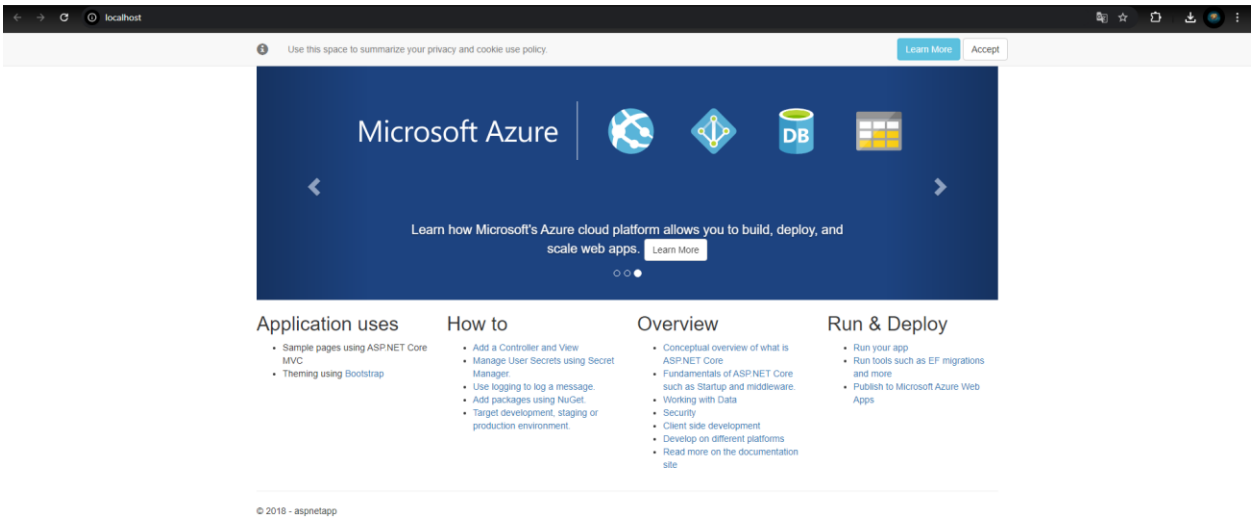
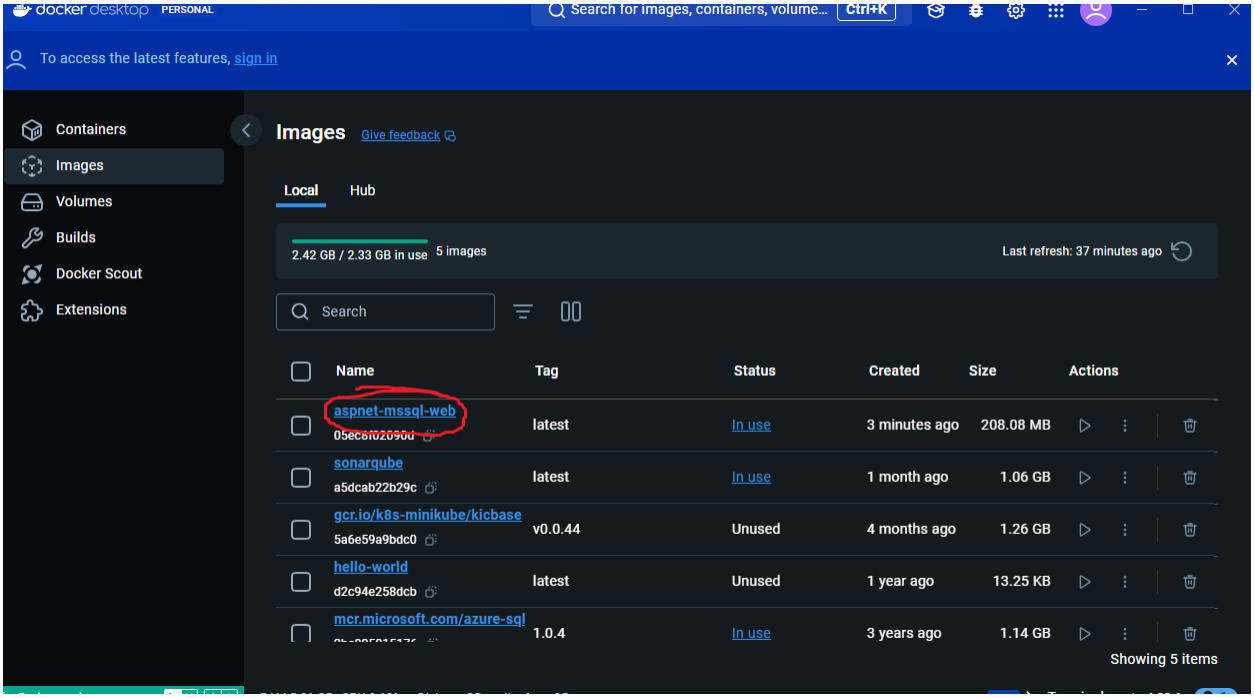


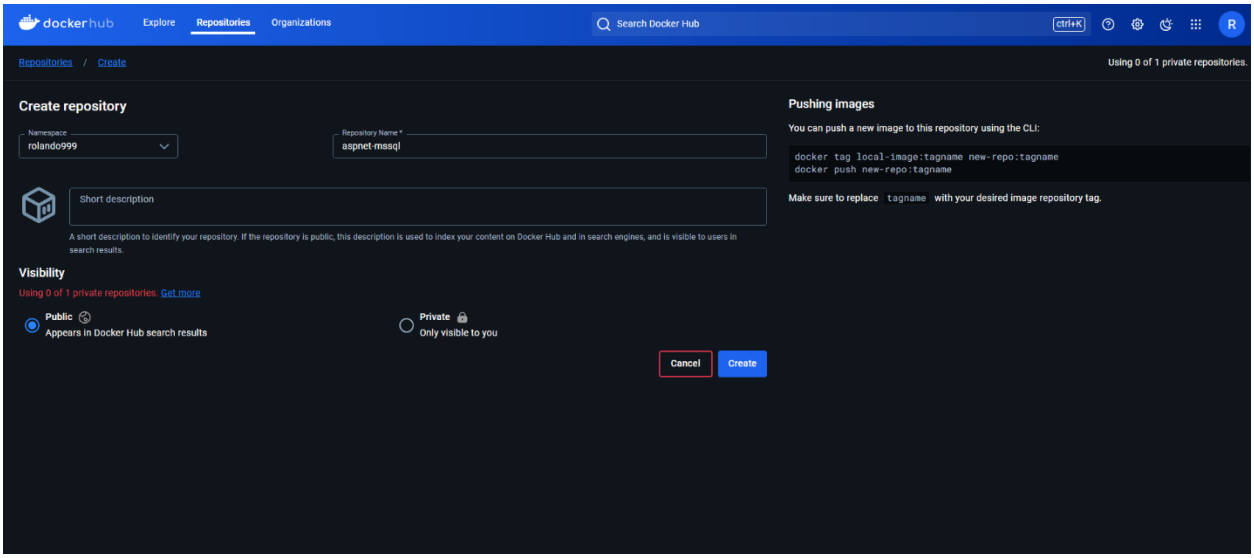
Imagen Docker



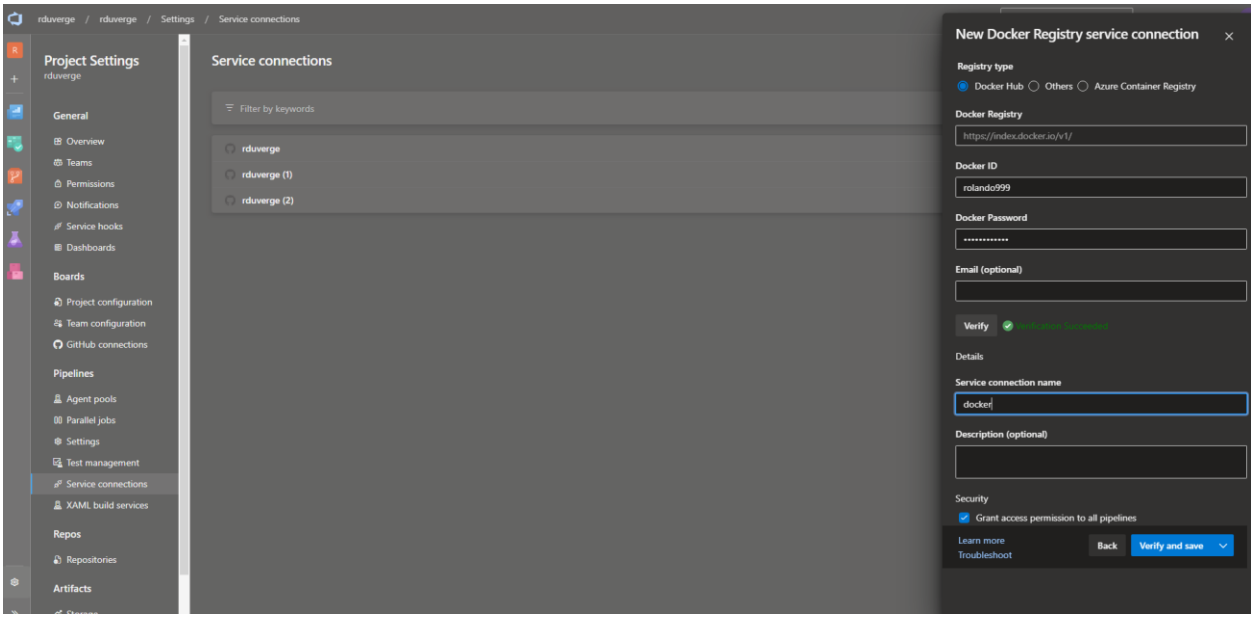
Subir la imagen a dockerhub/ACR/ECR desde el pipeline yaml.

Creo una cuenta en Docker hub

Creo el repositorio



Registro la conexión con Docker hub desde azure



1. **Pipeline que genera la imagen de Docker y la sube a Docker Hub. También dentro del Dentro del pipeline se ejecuta lo siguiente en bash .**

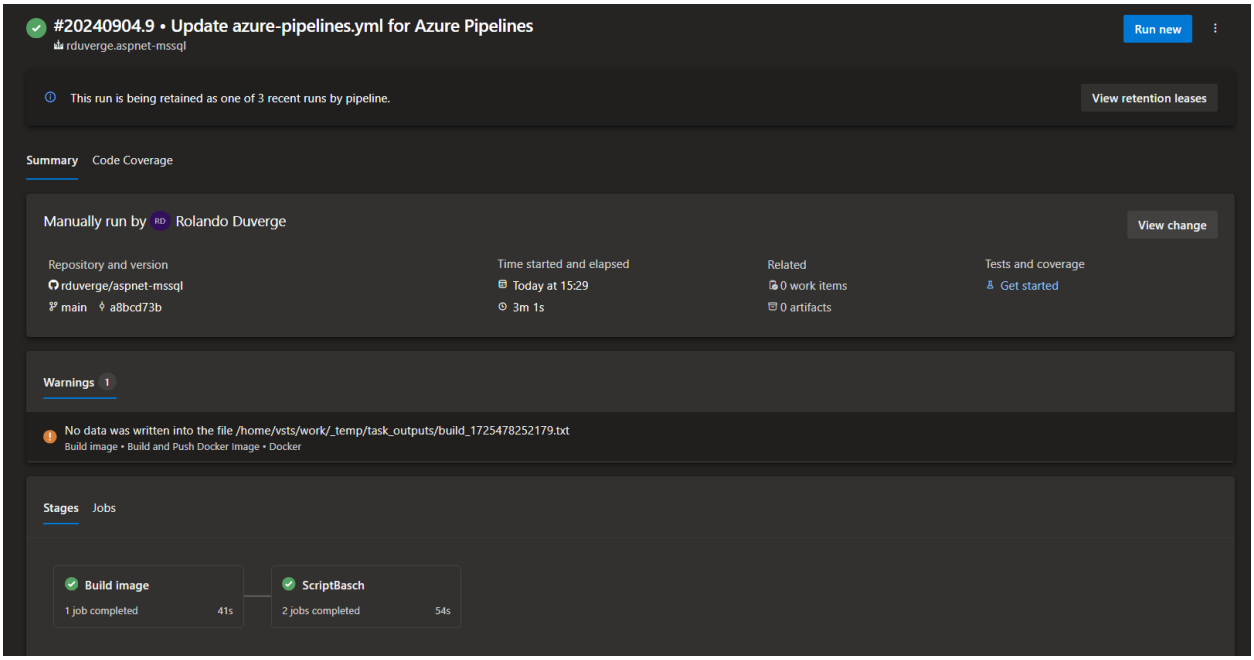
    a. **Imprime Hola Mundo 10 veces en pantalla con un job paralelo.**

    b. **Script que cree 10 archivos con la fecha y luego lo imprima en consola**

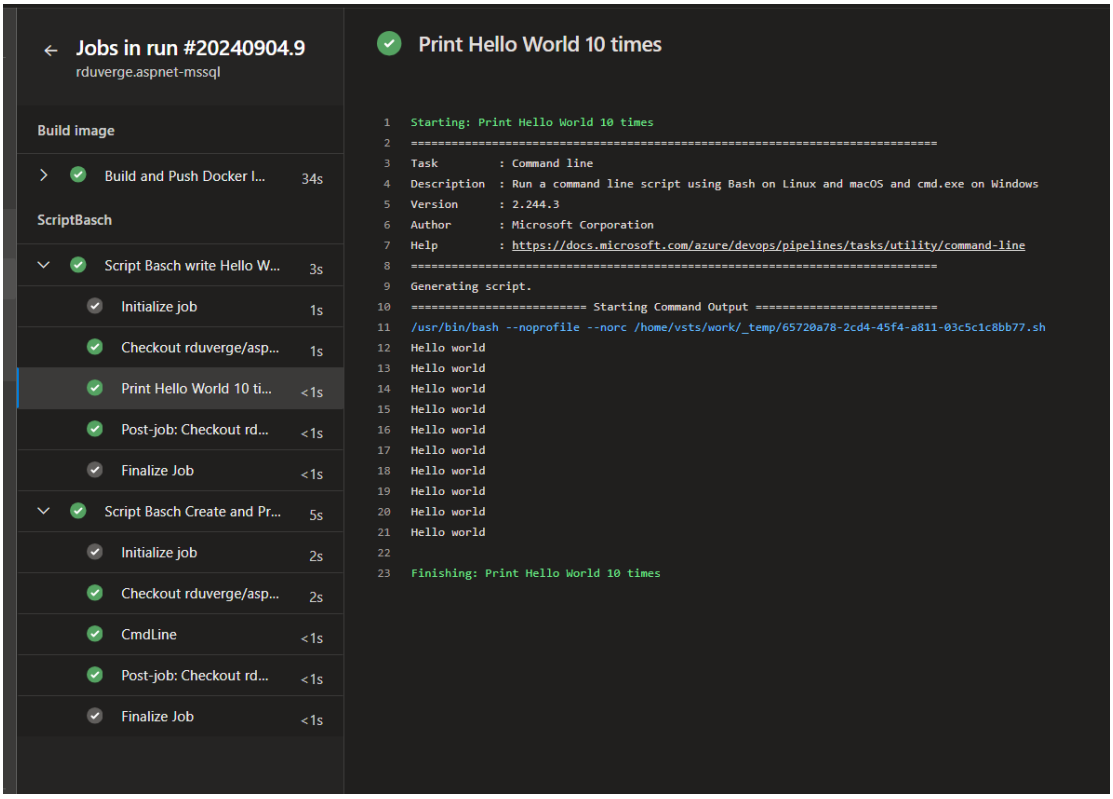
main rduverge/aspnet-mssql / azure-pipelines.yml *

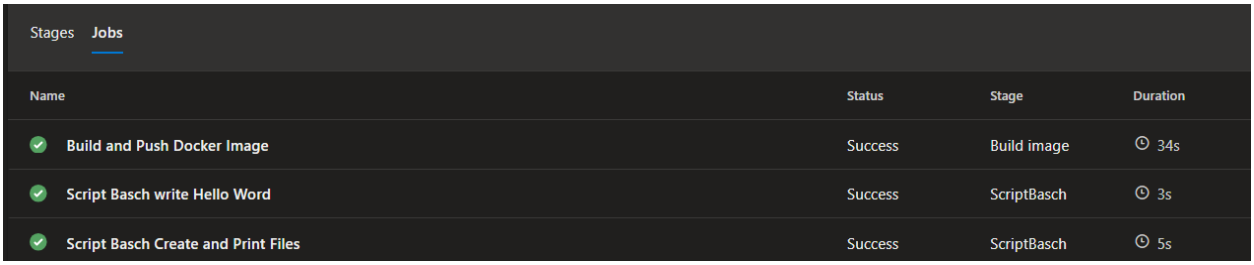
```yaml
# Docker
# Build a Docker image
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
- main

resources:
- repo: self

variables:
  tag: '$(Build.BuildId)'

stages:
- stage: BuildAndPush
  displayName: Build image
  jobs:
  - job: BuildAndPushDockerImage
    displayName: 'Build and Push Docker Image'
    pool:
      vmImage: ubuntu-latest
    steps:
    Settings
    - task: Docker@2
      inputs:
        containerRegistry: 'docker'
        repository: 'rolando999/aspnet-mssql'
        command: 'buildAndPush'
        Dockerfile: '**/Dockerfile'

- stage: ScriptBaschPrint
  displayName: ScriptBasch
  jobs:
  - job: ScriptBasch
    displayName: 'Script Basch write Hello Word'
    steps:
      - script: |
          for i in {1..10}
          do
            echo 'Hello world'
          done
        displayName: 'Print Hello World 10 times'

  - job: ScriptBaschFiles
    displayName: 'Script Basch Create and Print Files'
    steps:
      - script: |
          for i in {1..10}
          do
            filename="file_$i.txt"
            date > $filename
          done
          cat $filename
```

Resultado ejecución del Pipeline (se adjuntan los logs en el repositorio (logs_15.zip))



Script Hola Mundo





Script que crea 10 archivos con la fecha y la imprime en consola

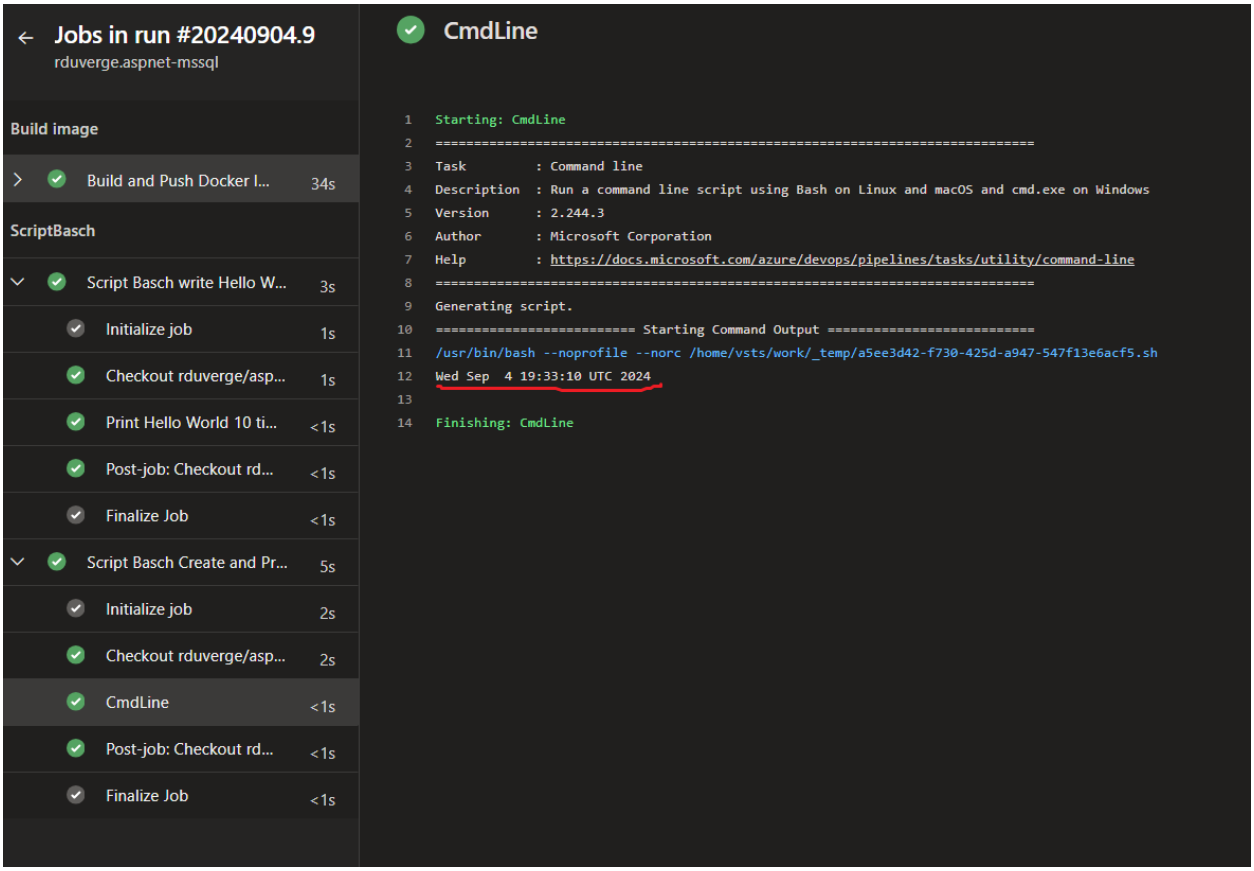
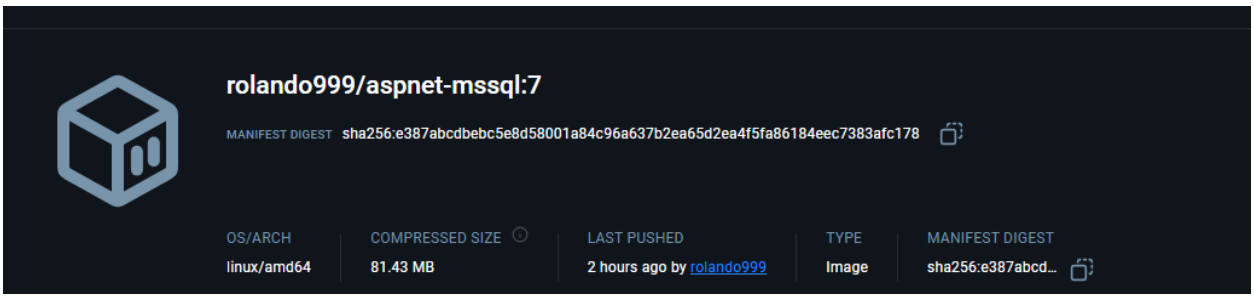
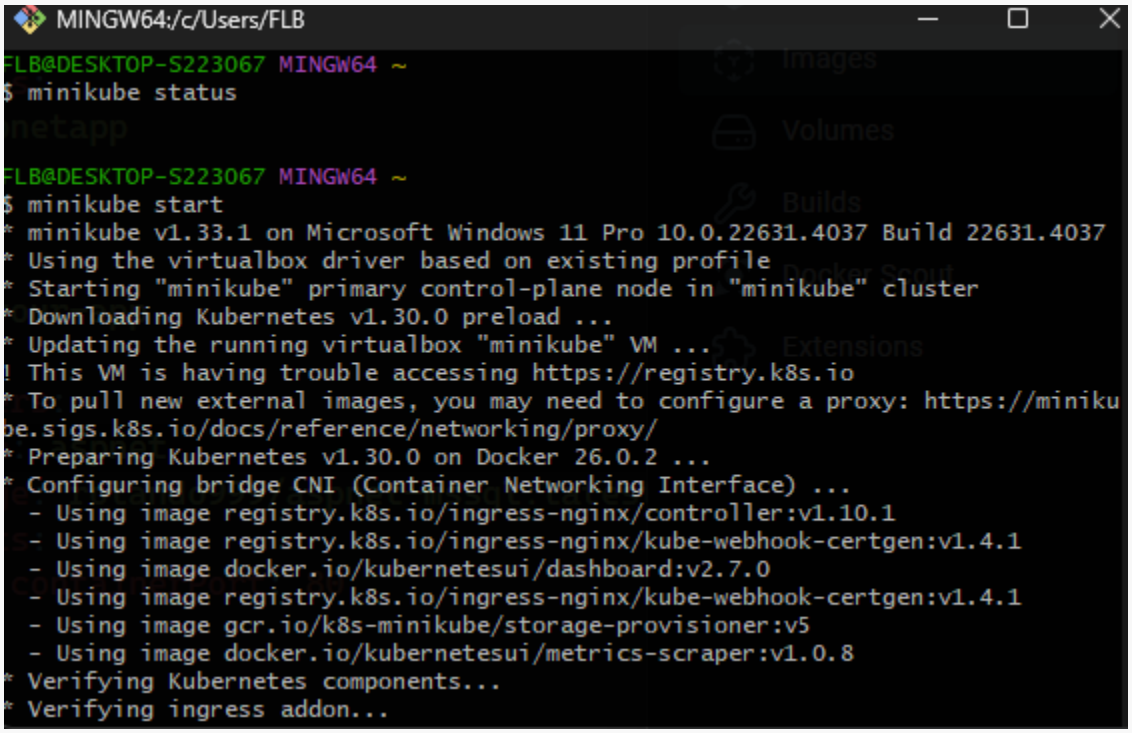


Imagen en docker hub

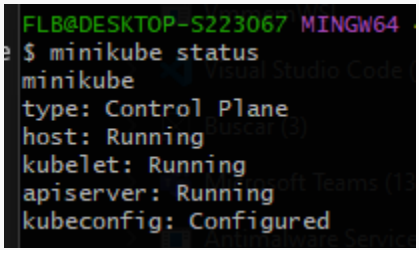


1. **Despliega la app a un clúster de kubernetes en este caso minikube con el driver de VirtualBox**

```
minikube start
```




```
minikube status
```



Creo el yaml del Deployment y ejecuto.

service-config-file.yaml U deployment-config-file.yaml U

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: aspnetapp-pod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: aspnetapp
  template:
    metadata:
      labels:
        app: aspnetapp
    spec:
      containers:
        - name: aspnet
          image: rolando999/aspnet-mssql:latest
          ports:
            - containerPort: 80

```

kubectl apply -f deployment-config-file.yaml

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl apply -f deployment-config-file.yaml
deployment.apps/aspnetapp-pod created
```

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl get pods
NAME                              READY   STATUS              RESTARTS   AGE
aspnetapp-pod-7c996c6cd6-f99cw    0/1     ContainerCreating   0          10s
```

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
aspnetapp-pod-7c996c6cd6-jthdg    1/1     Running   0          34s
```

Creo el Yaml del Service y ejecuto.

service-config-file.yaml U

```yaml
apiVersion: apps/v1
kind: Service
metadata:
  name: aspnetapp-service-pod
spec:
  type: NodePort
  selector:
    app: aspnetapp
  ports:
    - port: 80
      targetPort: 80
      NodePort: 30000
```

$ kubectl apply -f service-config-file.yaml

```
kubectl apply -f service-config-file.yaml
service/aspnetapp-service-pod created
```

Kubectl describe deployment aspnetapp-pod

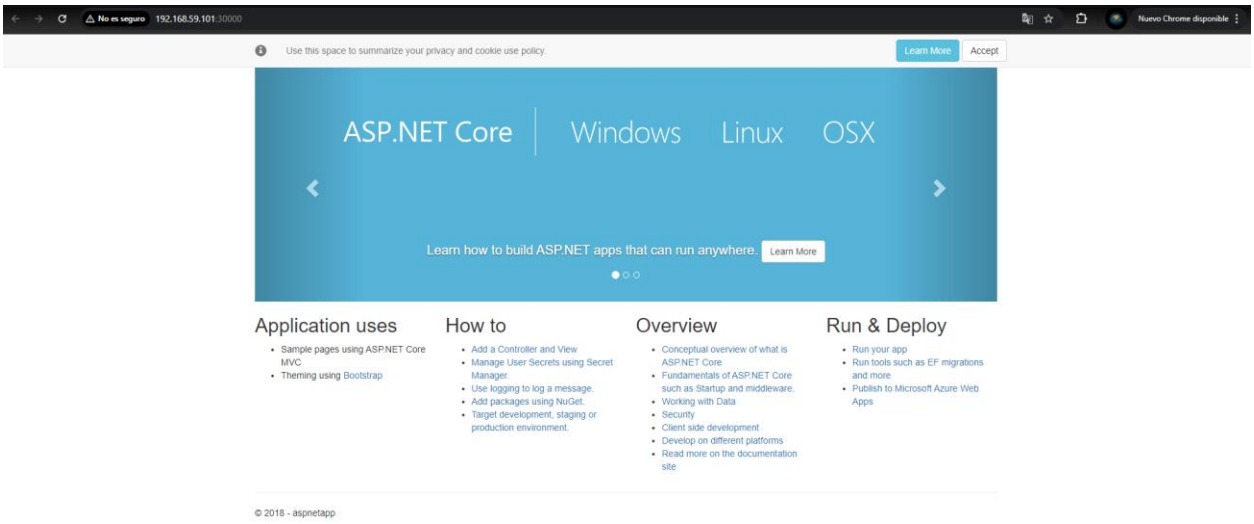
Kubectl get services

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl describe deployment aspnetapp-pod
Name:                   aspnetapp-pod
Namespace:              default
CreationTimestamp:      Wed, 04 Sep 2024 20:40:29 -0400
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=aspnetapp
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=aspnetapp
  Containers:
   aspnet:
    Image:         rolando999/aspnet-mssql:18
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   aspnetapp-pod-7c996c6cd6 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  75s   deployment-controller  Scaled up replica set aspnetapp-pod-7c996c6cd6 to

FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl get services
NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
aspnetapp-service-pod   NodePort    10.104.107.122   <none>        80:30000/TCP   17m
kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP        4h17m
```

Para ver la URL del servicio

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ minikube service aspnetapp-service-pod
|-----------|----------------------|-------------|-----------------------------|
| NAMESPACE |         NAME         | TARGET PORT |             URL             |
|-----------|----------------------|-------------|-----------------------------|
| default   | aspnetapp-service-pod |          80 | http://192.168.59.101:30000 |
|-----------|----------------------|-------------|-----------------------------|
* Opening service default/aspnetapp-service-pod in default browser...
```

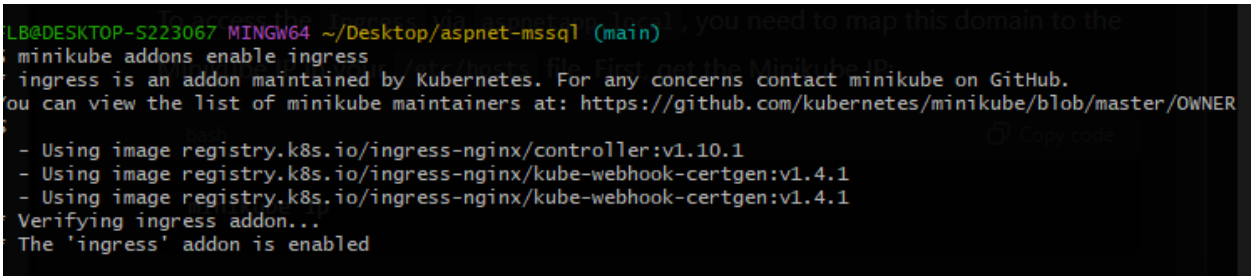
App desplegada en Kubernetes



Crea un endpoint externo accesible (ingress) para la aplicación

Habilito el ingress con

minikube addons enable ingress



Creo el yaml con la configuracion del ingress y lo aplico

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: aspnetapp-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: aspnetapp.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: aspnetapp-service-pod
            port:
              number: 80
```

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl apply -f ingress-config-file.yaml
ingress.networking.k8s.io/aspnetapp-ingress created
```

```
FLB@DESKTOP-S223067 MINGW64 ~/Desktop/aspnet-mssql (main)
$ kubectl get ingress
NAME                CLASS   HOSTS             ADDRESS          PORTS   AGE
aspnetapp-ingress   nginx   aspnetapp.local   192.168.59.101   80      44s
```

Enlace Git repositorio

https://github.com/rduverge/aspnet-mssql.git