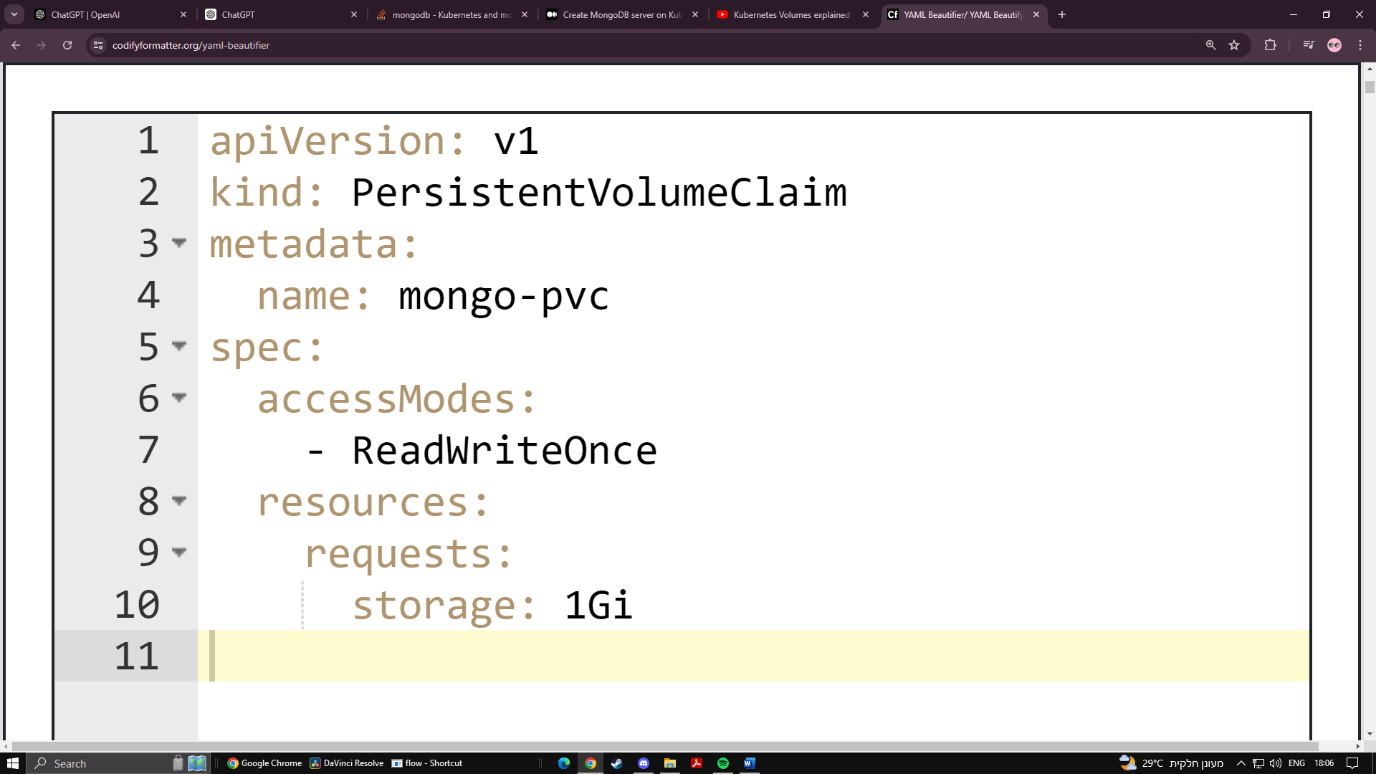


The volume part of a Kubernetes deployment allows you to attach persistent storage to your pods. This is essential for applications like databases that need to retain data even if the pods are restarted or rescheduled. Here's how the volume setup works in the context of the provided mongo-deployment.yaml:

**PersistentVolumeClaim (PVC)**

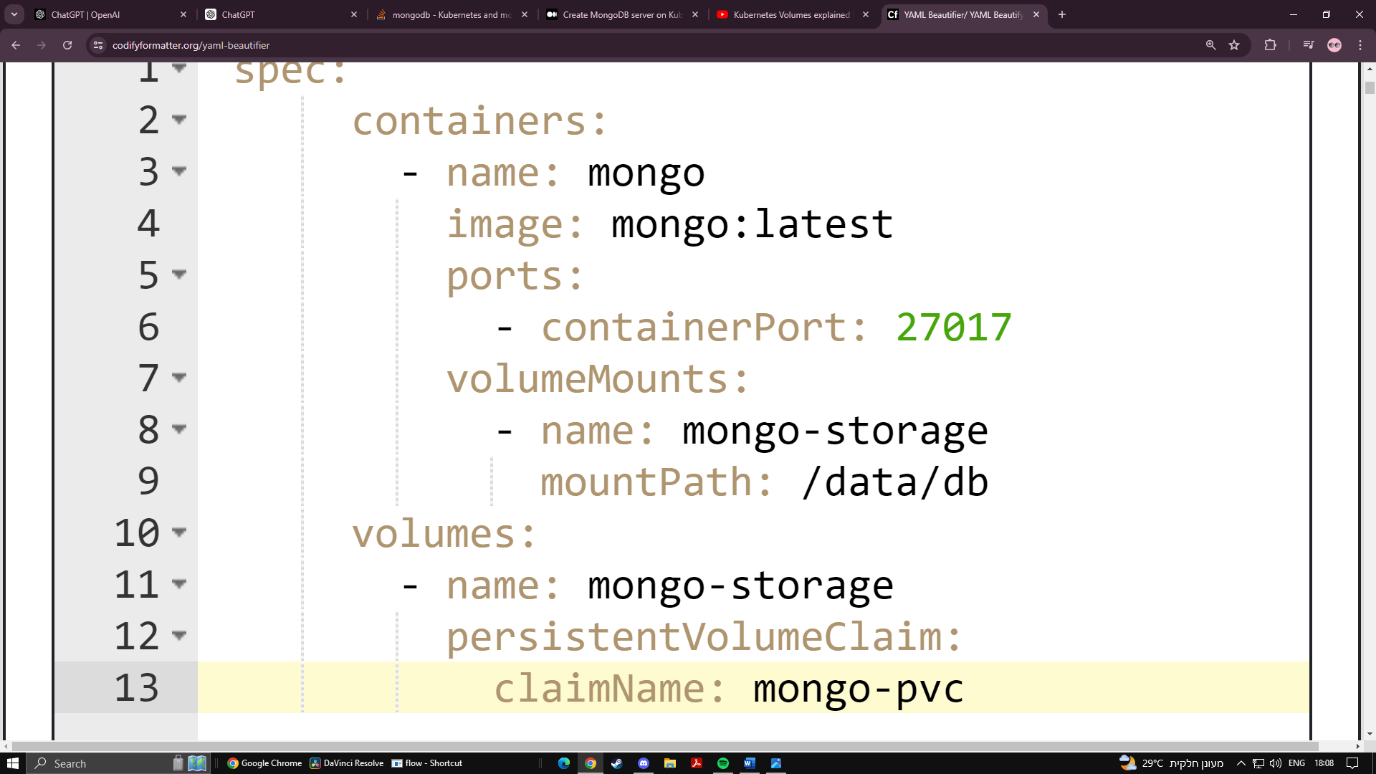
The PersistentVolumeClaim is a request for storage by a user. It is an abstraction over the underlying storage provider. You define the amount of storage you need and the access mode, and Kubernetes will find a suitable PersistentVolume to bind to this claim. 

* **accessModes:** Specifies the type of access required, in this case, ReadWriteOnce means the volume can be mounted as read-write by a single node.
* **resources.requests.storage:** Specifies the amount of storage requested, in this case, 1Gi (1 Gigabyte).

**Deployment**

In the mongo-deployment.yaml file, the volume and volumeMounts sections are used to specify how this PVC should be used by the MongoDB container.

Here’s the relevant part from the mongo-deployment.yaml file:



* **containers.volumeMounts:** This section specifies where the volume should be mounted inside the container.
  + **name:** The name of the volume mount.
  + **mountPath:** The path inside the container where the volume will be mounted. For MongoDB, this is typically /data/db.
* **volumes:** This section defines the volume that will be available to the pod.
  + **name:** The name of the volume, which should match the volumeMounts.name.
  + **persistentVolumeClaim:** Specifies that this volume should be bound to the PersistentVolumeClaim defined earlier.
    - **claimName:** The name of the PersistentVolumeClaim (in this case, mongo-pvc).

**How It Works**

1. **PVC Creation:** When you apply the mongo-pvc.yaml, Kubernetes will try to find a matching PersistentVolume in the cluster that satisfies the request (1Gi storage, ReadWriteOnce access).
2. **PVC Binding:** Once a matching PersistentVolume is found, the PersistentVolumeClaim will be bound to it.
3. **Volume Assignment:** In the mongo-deployment.yaml, the volume defined by mongo-pvc is attached to the pod and mounted at /data/db in the MongoDB container.
4. **Persistent Storage:** Data written to /data/db inside the MongoDB container is stored on the persistent volume, ensuring data persists across pod restarts or rescheduling.

**Applying the Configuration**

Apply the configuration files in the following order to ensure that the PVC is available when the deployment is created:

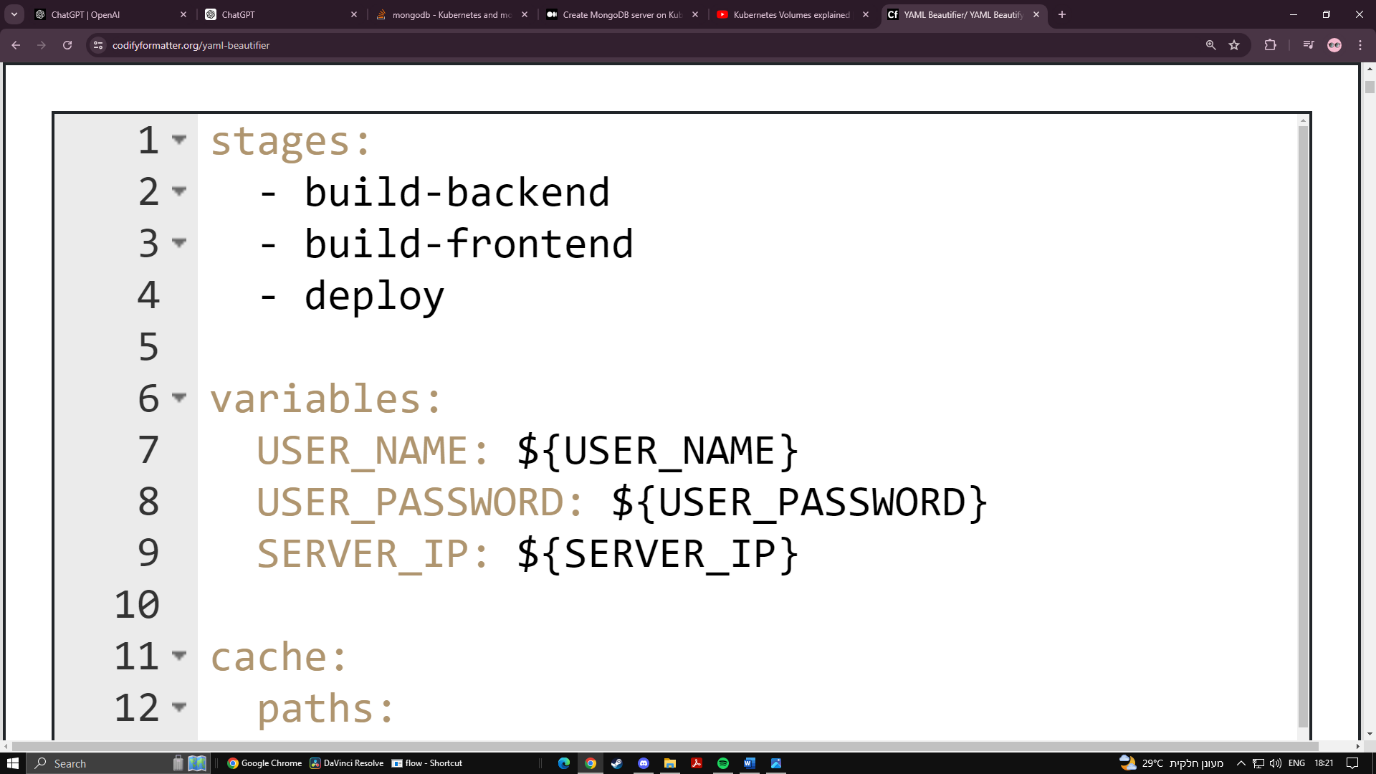
sh

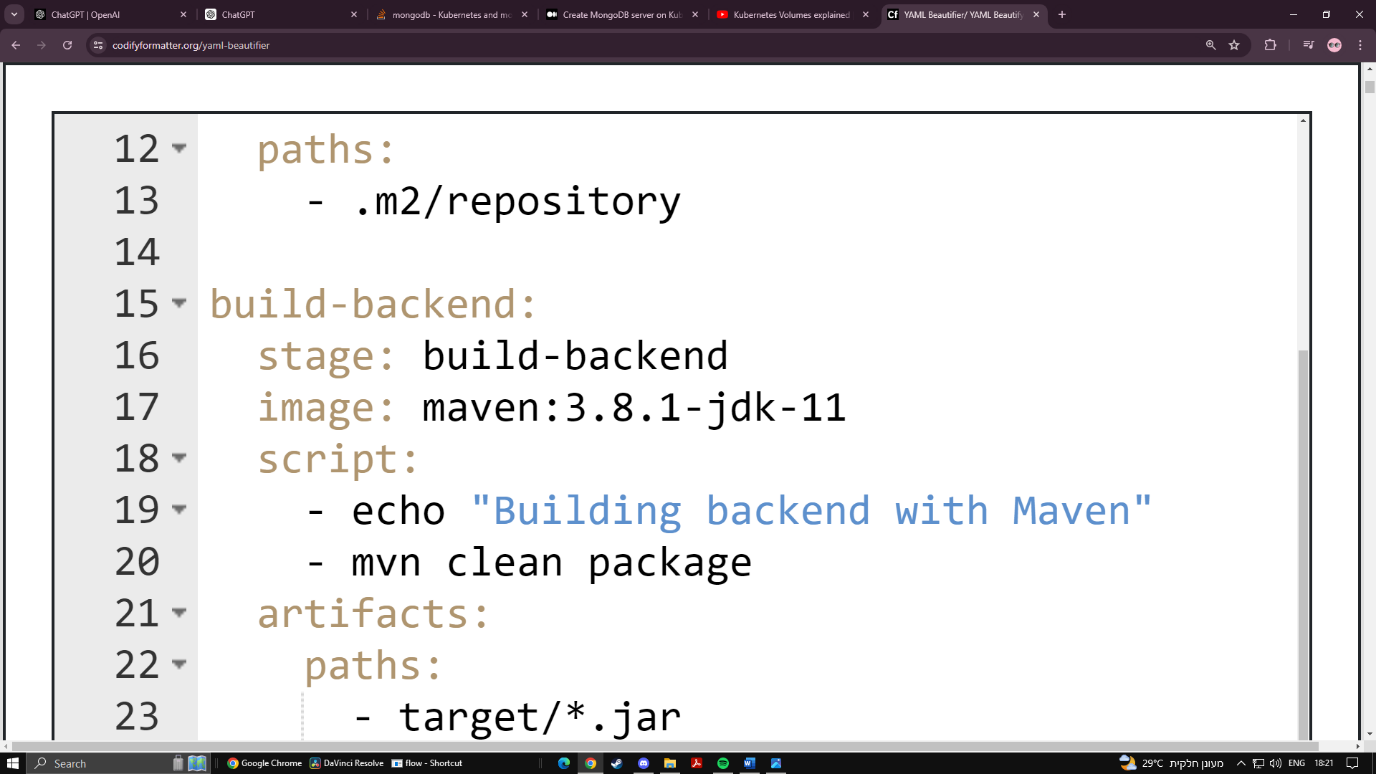
Copy code

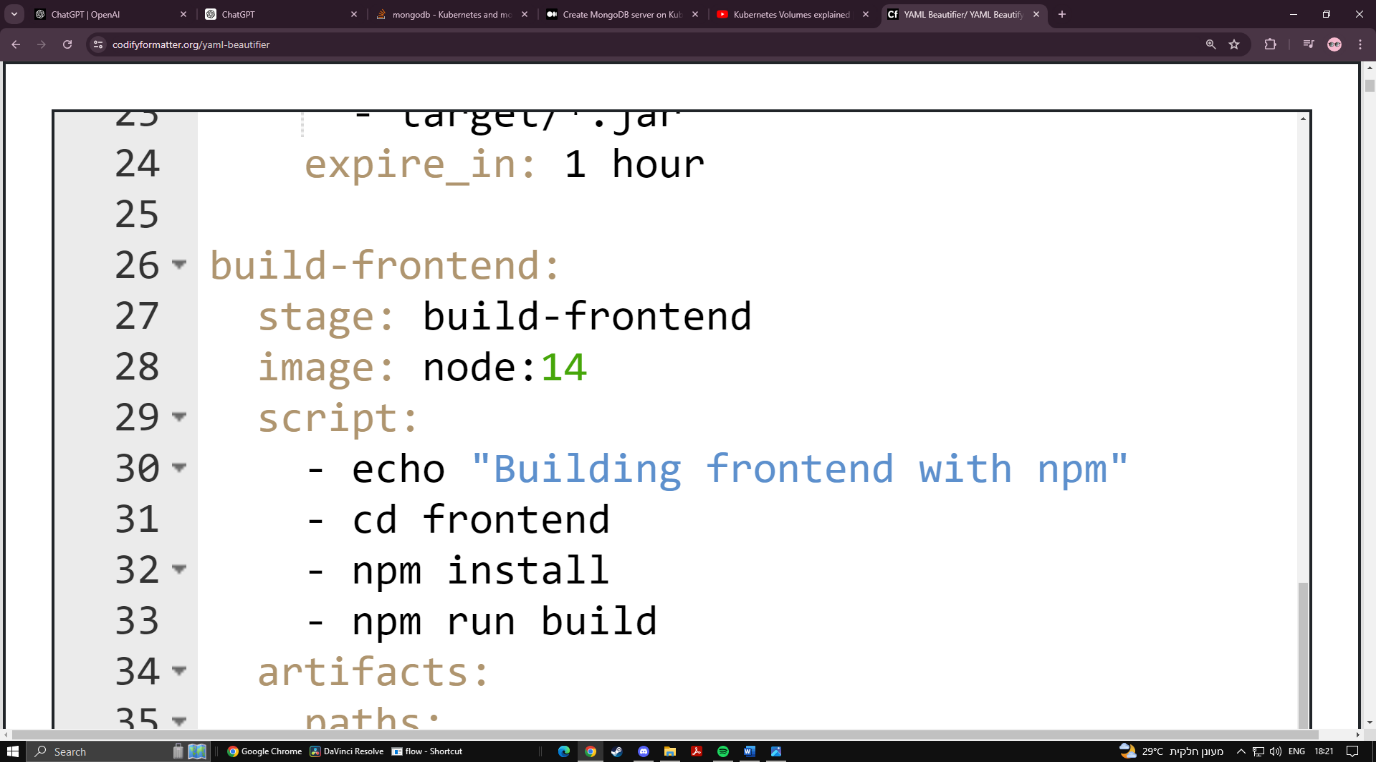
kubectl apply -f mongo-pvc.yaml

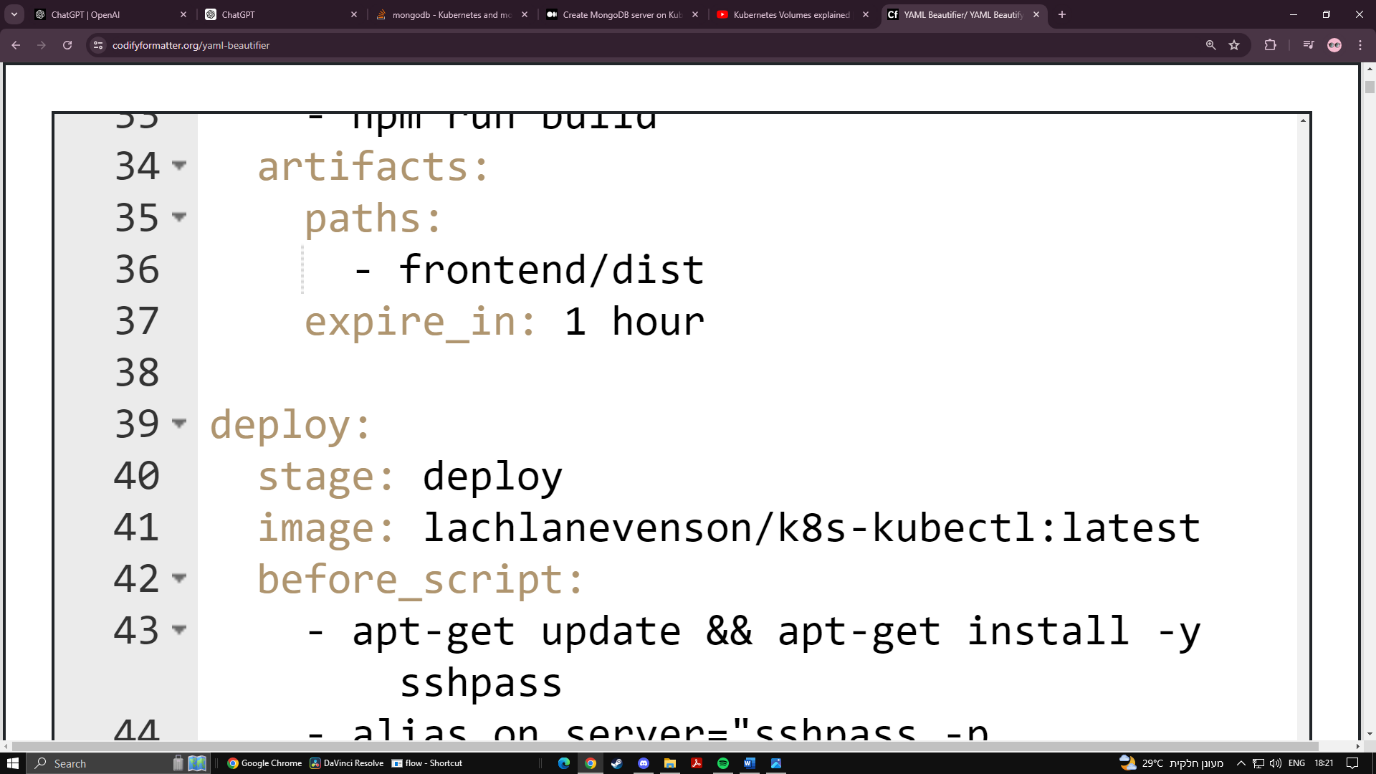
kubectl apply -f mongo-deployment.yaml

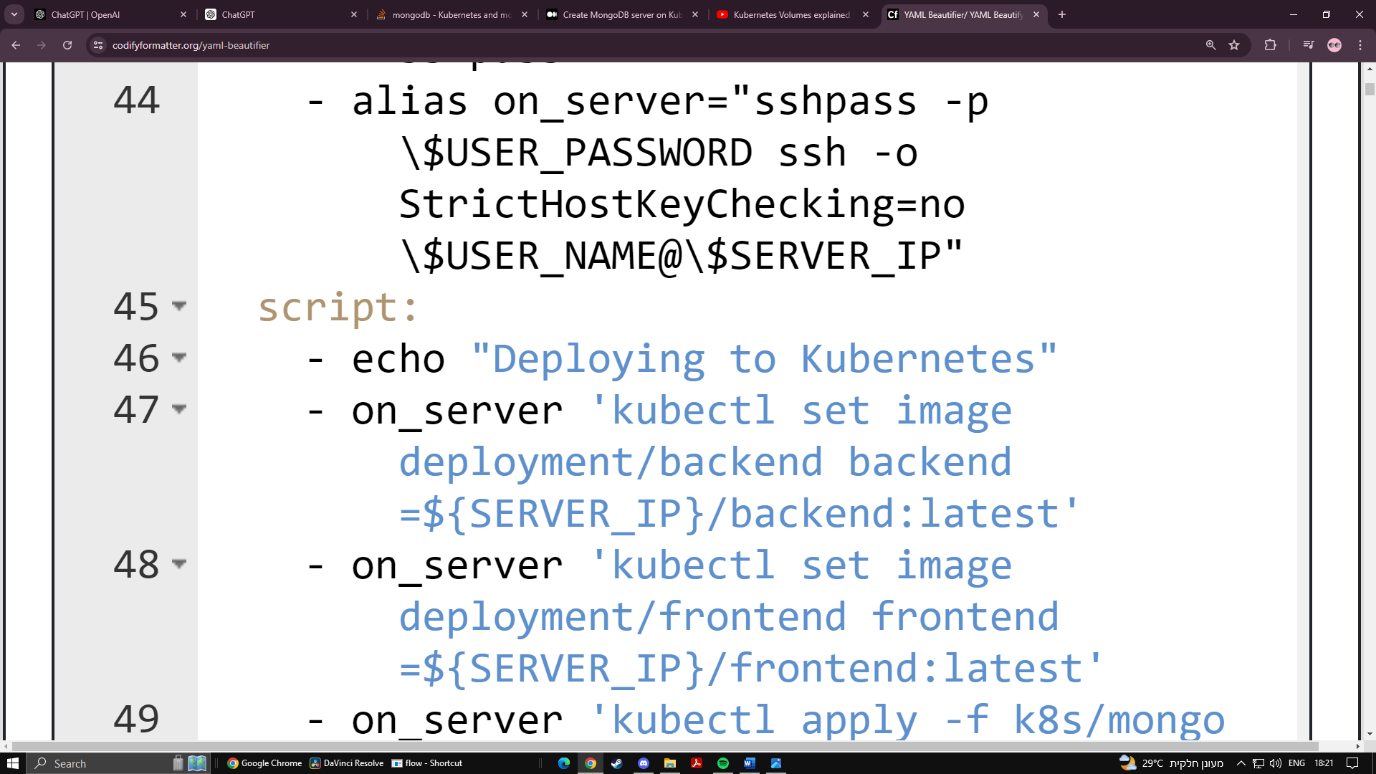
CICD

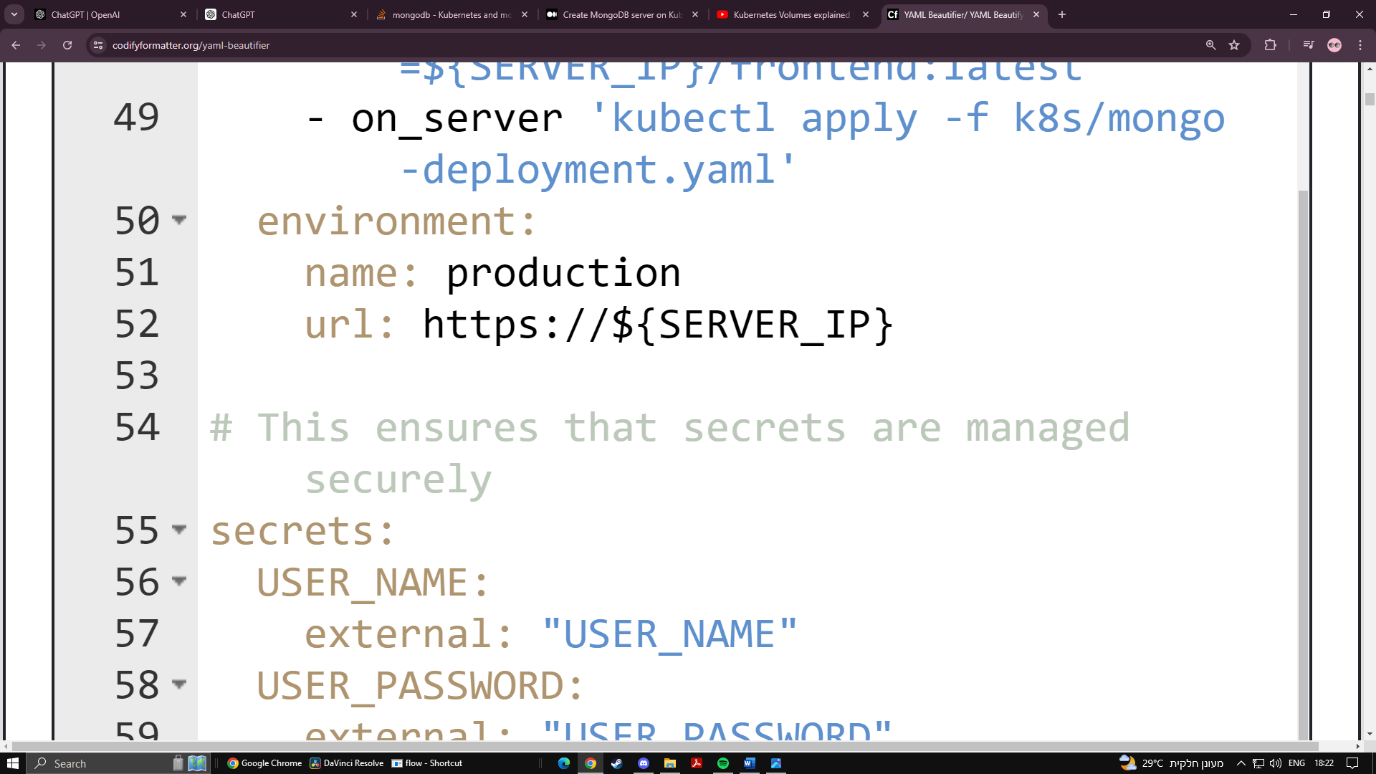












If the prod is a windows machine:

build-backend:

stage: build-backend

image: maven:3.8.1-jdk-11

script:

- echo "Building backend with Maven"

- mvn clean package

artifacts:

paths:

- target/\*.jar

expire\_in: 1 hour

build-frontend:

stage: build-frontend

image: node:14

script:

- echo "Building frontend with npm"

- cd frontend

- npm install

- npm run build

artifacts:

paths:

- frontend/dist

expire\_in: 1 hour

deploy:

stage: deploy

image: mcr.microsoft.com/windows/servercore:ltsc2022

before\_script:

- apt-get update && apt-get install -y sshpass

- echo "alias on\_server=\"sshpass -p \$USER\_PASSWORD ssh -o StrictHostKeyChecking=no \$USER\_NAME@\$SERVER\_IP\"" >> ~/.bashrc

- source ~/.bashrc

script:

- echo "Deploying to Kubernetes"

- on\_server 'kubectl set image deployment/backend backend=${SERVER\_IP}/backend:latest'

- on\_server 'kubectl set image deployment/frontend frontend=${SERVER\_IP}/frontend:latest'

- on\_server 'kubectl apply -f k8s/mongo-deployment.yaml'

environment:

name: production

url: https://${SERVER\_IP}