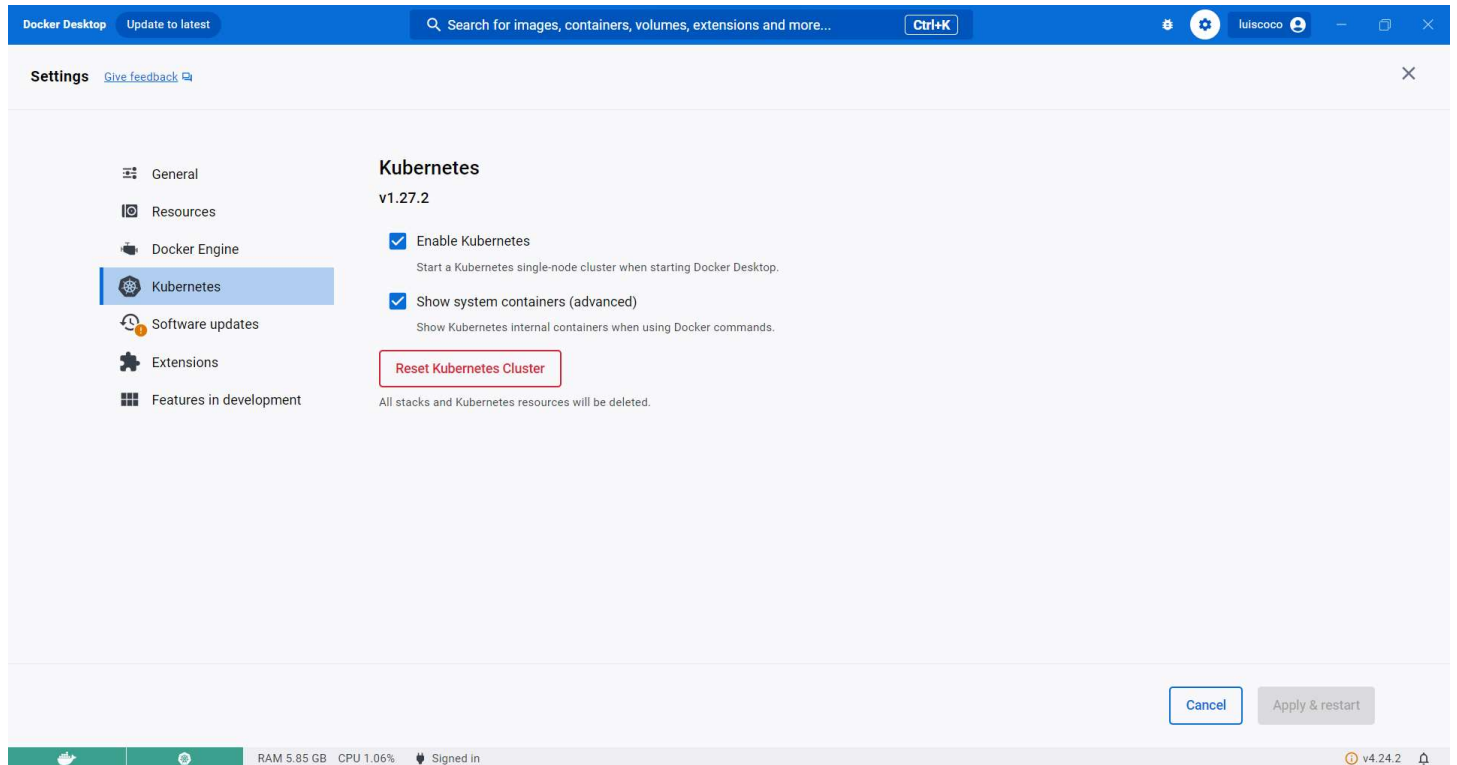


How to deploy a SpringBoot WebAPI in local Kubernetes cluster (Docker Desktop)

1. Run Docker Desktop and enable Kubernetes

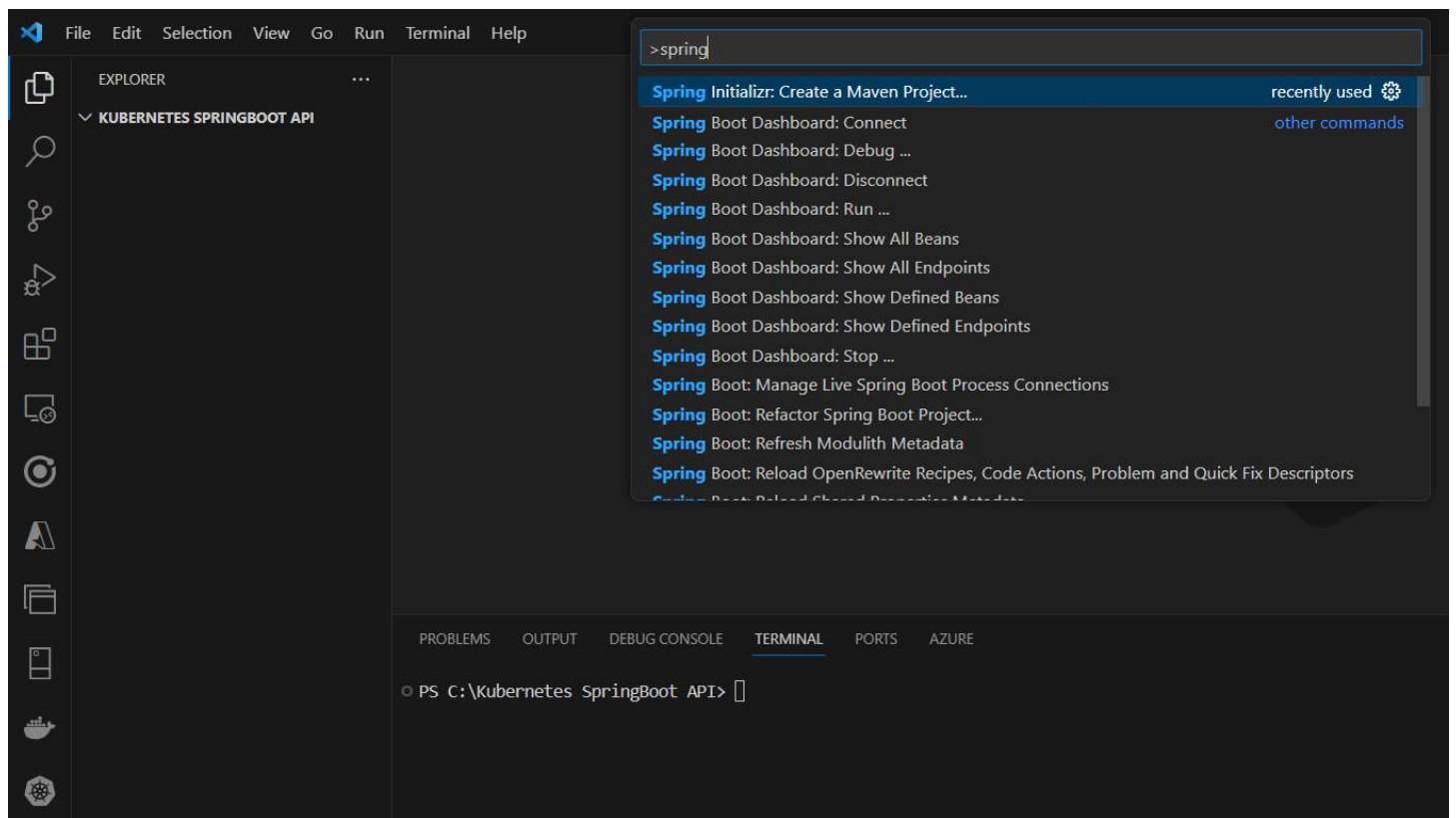


2. Create a SpringBoot WebAPI with VSCode

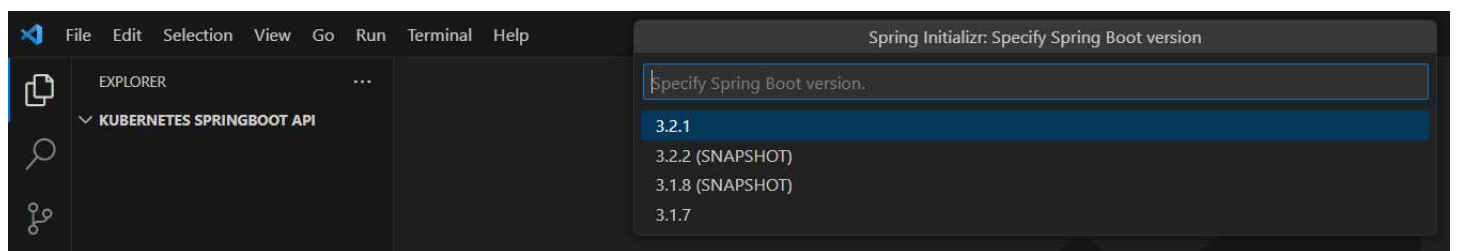
See this rep: https://github.com/luiscoco/SpringBoot_Sample1-created-with-VSCode

Press **Ctrl+Shift+P** for creating a new project in VSCode

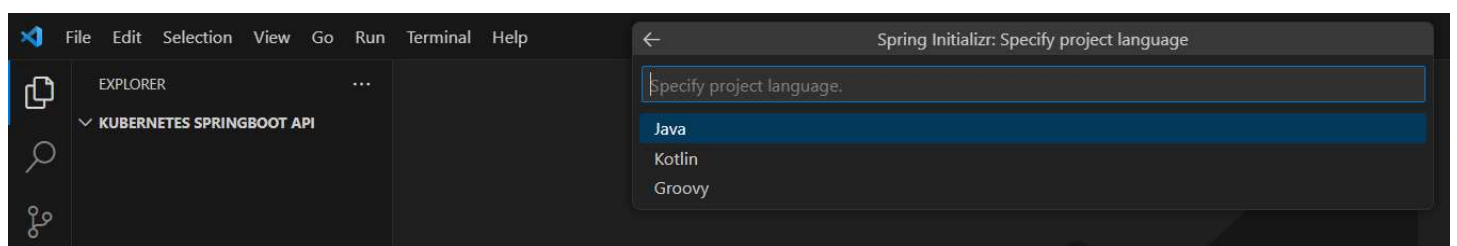
We select the first option "Spring Initializer: Create a Maven Project..."



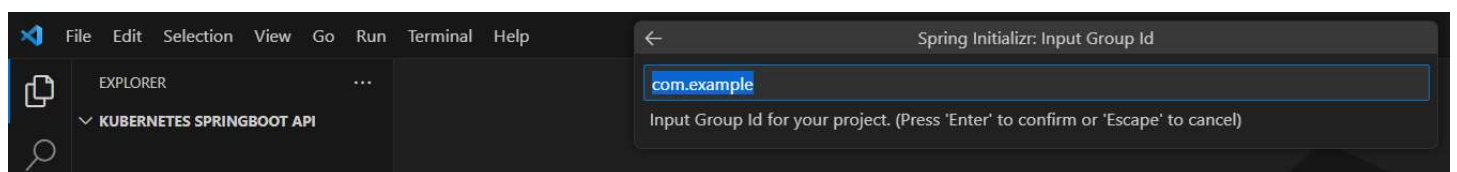
Set Spring Boot version 3.2.1



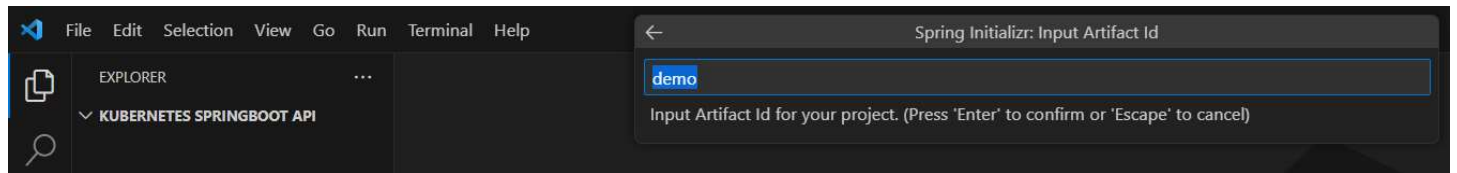
Set the language Java



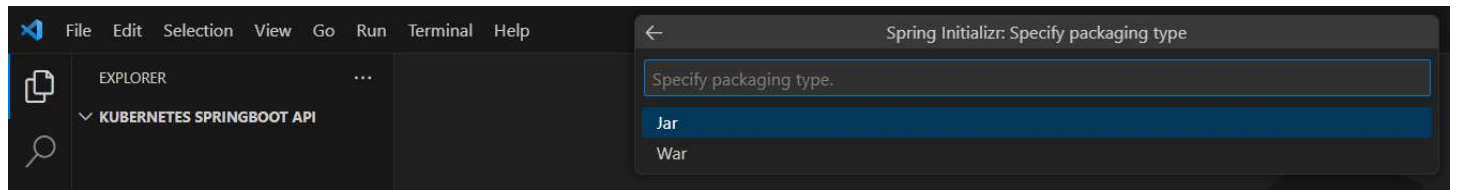
Input the Group Id project



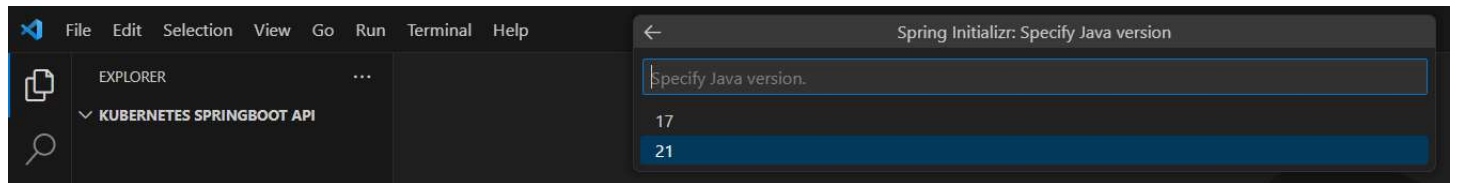
Input the Artifact Id



Specify the packing type Jar



Se the Java version 21



3. Add the Kubernetes manifest files (deployment.yml and service.yml files) to the SpringBoot WebAPI in VSCode

deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demoapi-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: demoapi
  template:
    metadata:
      labels:
        app: demoapi
    spec:
      containers:
        - name: demoapi
          image: luiscoco/demoapi:latest
          ports:
            - containerPort: 8080
```

service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: demoapi-service
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: demoapi
```

4. Install the project dependencies

Include the following libraries in the pom.xml file:

- spring-boot-starter-actuator
- spring-boot-starter-web
- spring-boot-starter-test

```
...
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
...
```

5. Add the SpringBoot WebAPI source code

DemoapiApplication.java

```
package com.example.demoapi;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
```

```
public class DemoapiApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(DemoapiApplication.class, args);  
    }  
}
```

HelloController.java

<http://localhost:8080/hello>

```
package com.example.demoapi.controller;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class HelloController {  
    @GetMapping("/hello")  
    public String sayHello() {  
        return "Hello, World!";  
    }  
}
```

6. How build the Docker image and run it

To build the Docker image execute this command:

```
docker build -t luiscoco/demoapi:latest .
```

To push the Web API docker image run this command:

```
docker build -t luiscoco/demoapi:latest .
```

To run a Docker image execute the command:

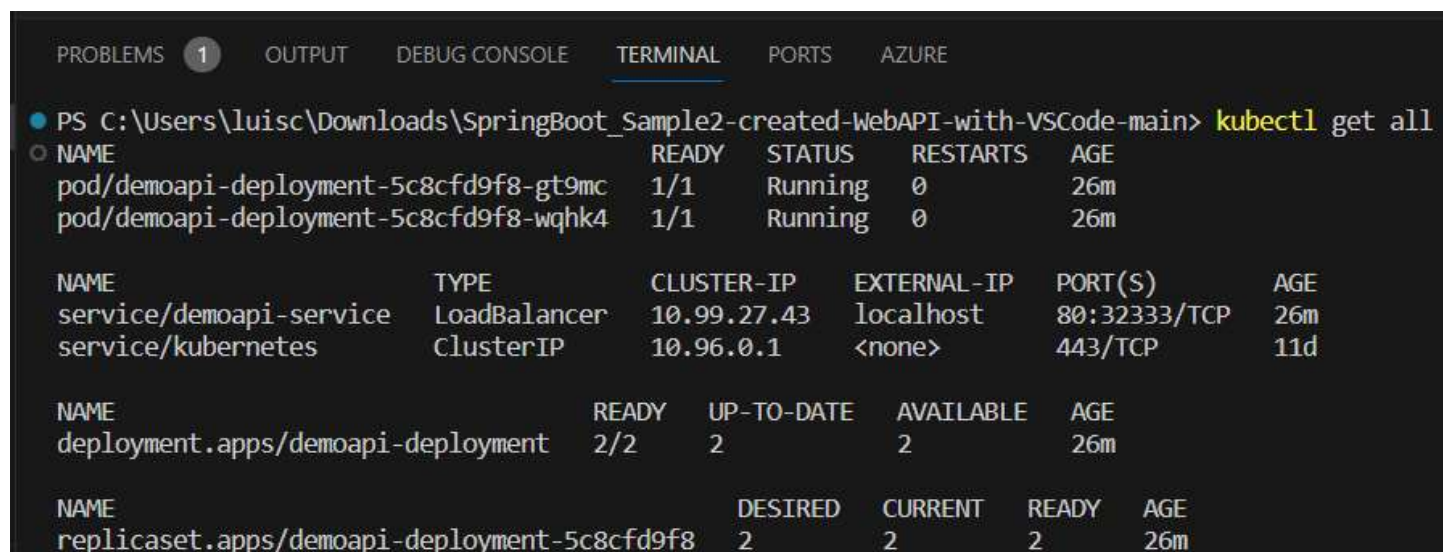
```
docker run -p 8080:8080 luiscoco/demoapi:latest
```

7. Deploy manifest Kubernetes

Execute the following commands to deploy the Kubernetes:

```
kubectl apply -f deployment.yml
```

```
kubectl apply -f service.yml
```



```
PS C:\Users\luisc\Downloads\SpringBoot_Sample2-created-WebAPI-with-VSCode-main> kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/demoapi-deployment-5c8cf9f8-gt9mc	1/1	Running	0	26m
pod/demoapi-deployment-5c8cf9f8-wqhk4	1/1	Running	0	26m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/demoapi-service	LoadBalancer	10.99.27.43	localhost	80:32333/TCP	26m
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	11d

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/demoapi-deployment	2/2	2	2	26m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/demoapi-deployment-5c8cf9f8	2	2	2	26m

Access to the WebAPI endpoint: <http://localhost/hello>

