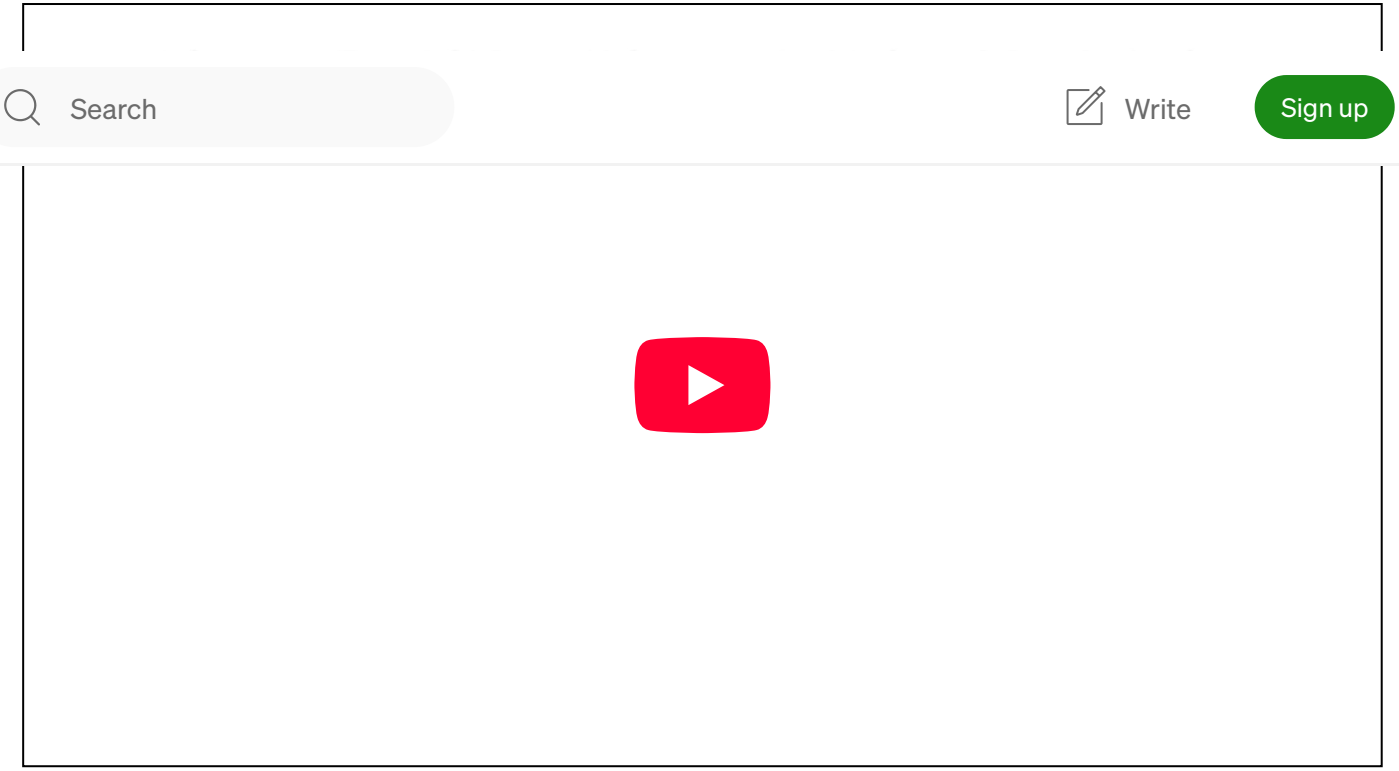
Kubernetes Tutorial | Setup Kubernetes in Windows & Run Spring boot application on k8s cluster

Java Techie Follow 6 min read · Dec 10, 2021



Prerequisite

1. Windows 8 or 10 OS
2. Min 8 GB RAM
3. Docker Desktop

Minikube setup steps in windows OS :

Minikube runs a single-node Kubernetes cluster on your machine so that you can try out Kubernetes for your daily development work.

Step 1 :

install kubectl: https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/

kubectl is a command line tool, using kubectl we can connect to k8s cluster from our computer .

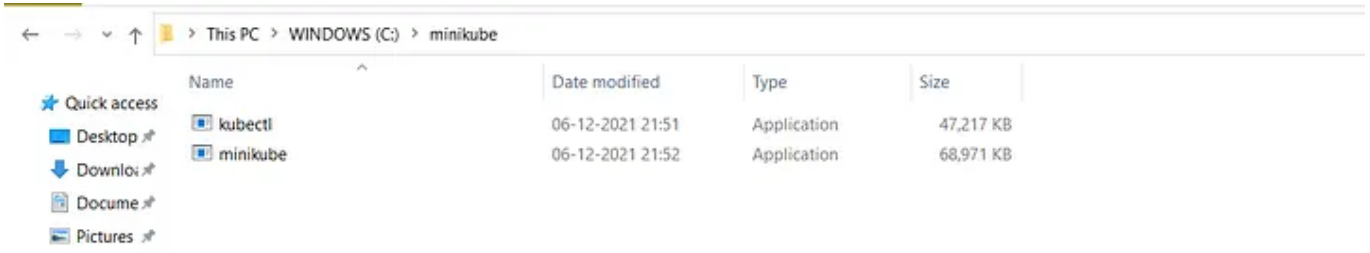
Step 2 :

install minikube : https://v1-18.docs.kubernetes.io/docs/tasks/tools/install-minikube/

after download minikube-windows-amd64 exe file rename it to minikube.exe

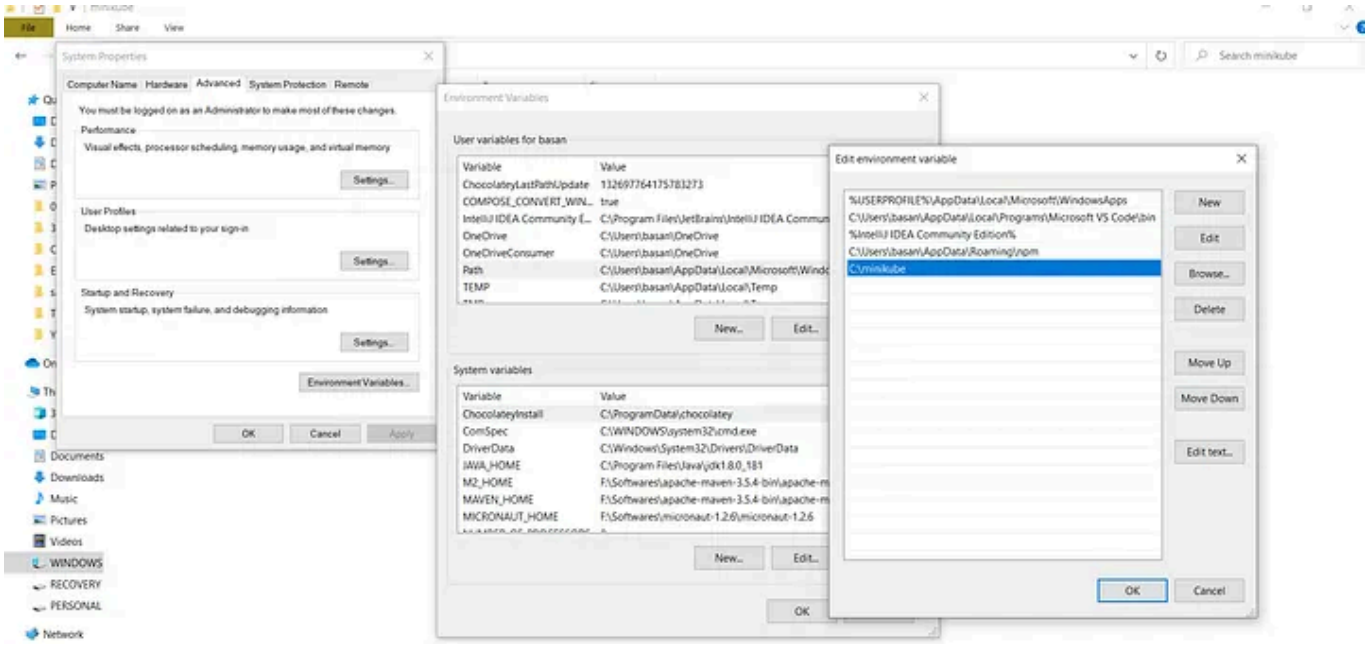
Step 3:

Once you download both exe file , just move these two file to separate directory/folder like below



Step 4:

Next set this folder path as environment variable in your windows



Step 5 :

After set environment variable , open command prompt and run command

```
minikube version
```

it should display current installed minikube version

We are good with minikube setup in windows . now you can play with k8s

Step 6 :

To create a Kubernetes cluster first we need start minikube server in our system . There is multiple driver using any of them you can start your minikube

```
minikube start --driver=<driver name>
```

1. Hyper-v

Note: Hyper-V can run on three versions of Windows 10: Windows 10 Enterprise, Windows 10 Professional, and Windows 10 Education.

2. VirtualBox

If you have enough memory in your system like 16 GB RAM and good processor then VirtualBox is right choice for you .

Download VirtualBox : https://www.virtualbox.org/wiki/Downloads

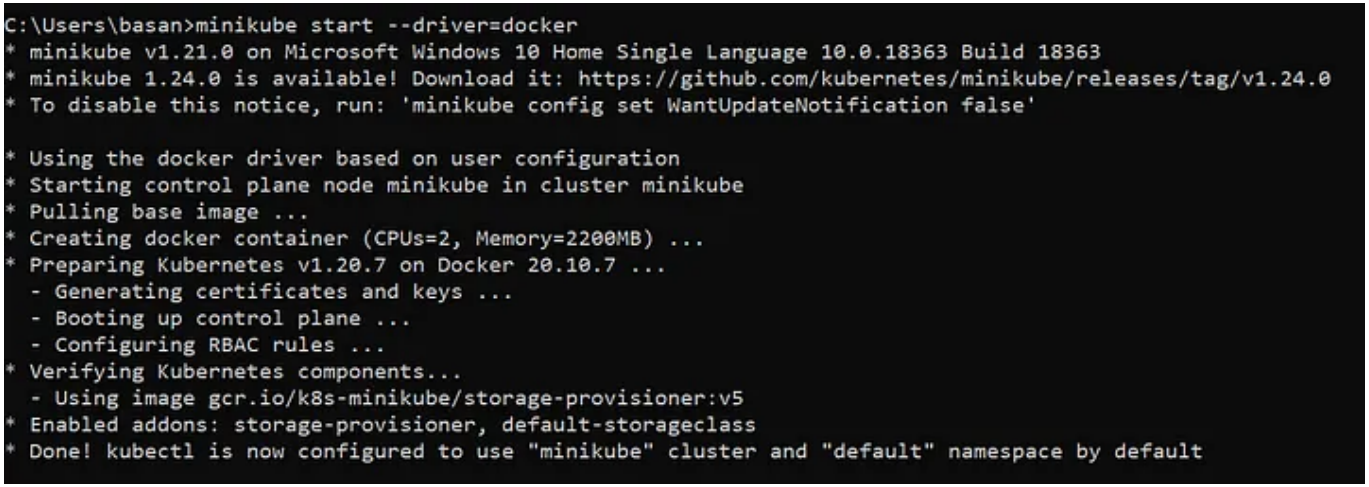
3. Docker

In this example i will use Docker as already it installed in my machine .

run below command to start minikube with docker driver

```
minikube start --driver=docker
```

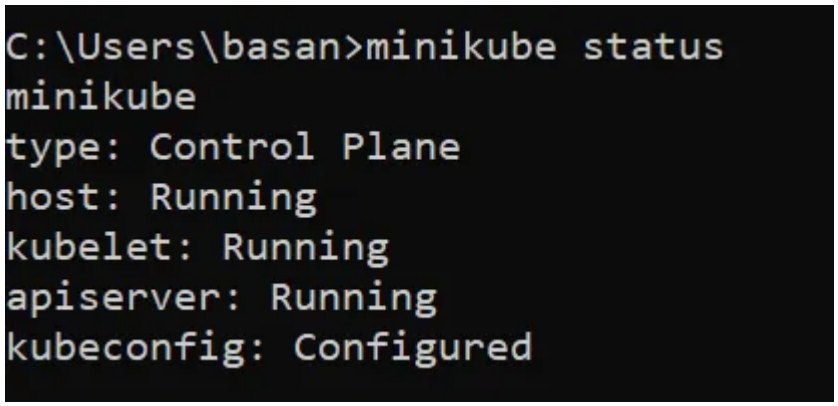
You will get below results , if your minikube starts without any error



Step 7 :

After successfully started minikube , you can verify minikube status

```
minikube status
```



Step 8:

As we know minikube will provide you single node cluster to work with k8s locally so to verify cluster info and node status we can run below commands

```
kubectl cluster-info
```

```
C:\Users\basan>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:50954
KubeDNS is running at https://127.0.0.1:50954/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

C:\Users\basan>
```

```
kubectl get node
```

```
C:\Users\basan>kubectl get nodes
NAME       STATUS   ROLES                  AGE   VERSION
minikube   Ready    control-plane,master   20m   v1.20.7

C:\Users\basan>
```

Now we have Kubernetes cluster ready with us to start our local development work

Deploy Spring boot application to Kubernetes cluster

Step 1:

To allow Kubernetes to read your docker repository you need to run below command , so that both will be in sync

```
minikube docker-env
```

```
C:\Users\basan>minikube docker-env
SET DOCKER_TLS_VERIFY=1
SET DOCKER_HOST=tcp://127.0.0.1:50956
SET DOCKER_CERT_PATH=C:\Users\basan\.minikube\certs
SET MINIKUBE_ACTIVE_DOCKERD=minikube
REM To point your shell to minikube's docker-daemon, run:
REM @FOR /f "tokens=*" %i IN ('minikube -p minikube docker-env') DO @%i

C:\Users\basan>@FOR /f "tokens=*" %i IN ('minikube -p minikube docker-env') DO @%i
```

List down all docker images

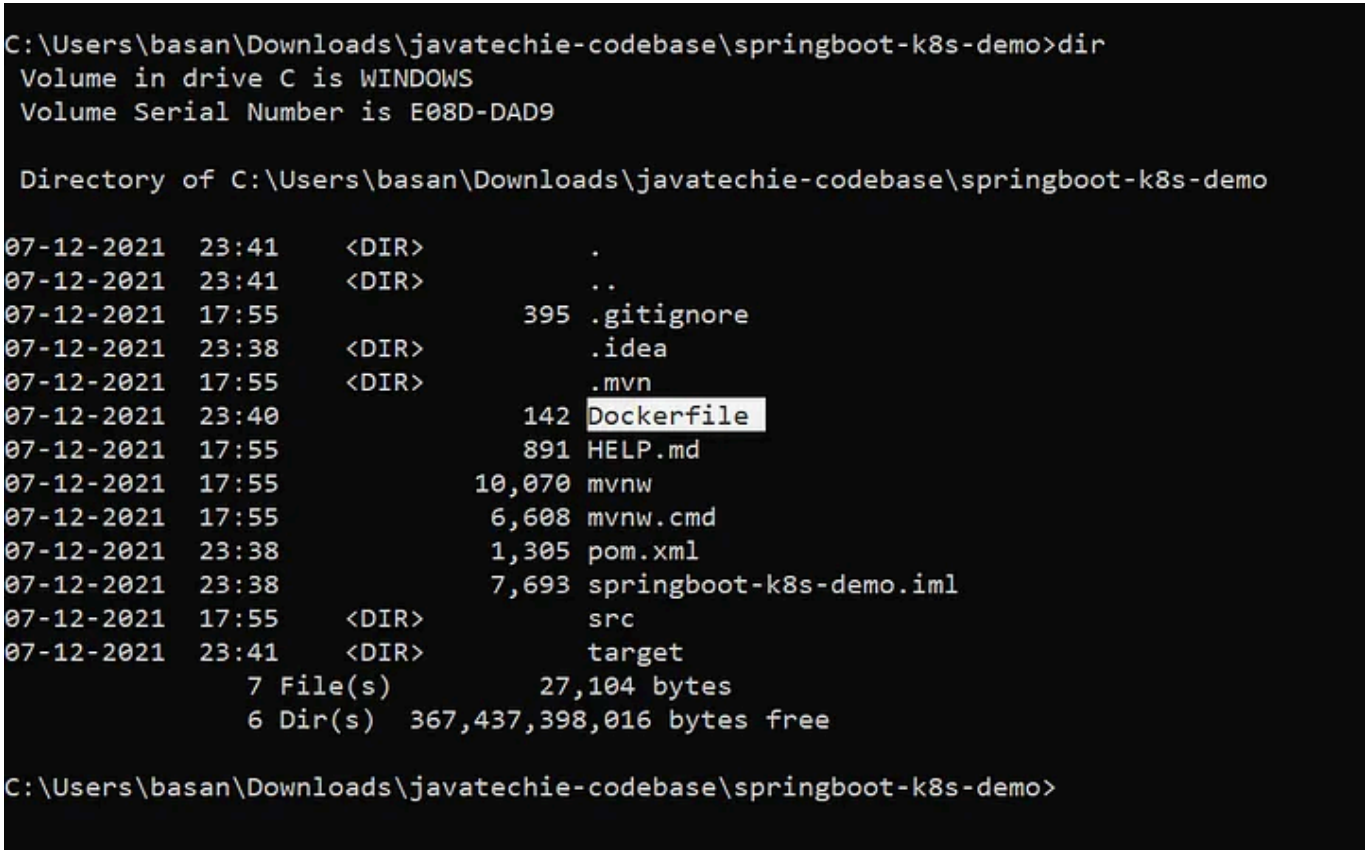
```
docker images
```

```
C:\Users\basan>docker images
REPOSITORY                                TAG        IMAGE ID       CREATED         SIZE
k8s.gcr.io/kube-proxy                     v1.20.7    ff54c88b8ecf   7 months ago    118MB
k8s.gcr.io/kube-controller-manager        v1.20.7    22d1a2072ec7   7 months ago    116MB
k8s.gcr.io/kube-apiserver                 v1.20.7    034671b24f0f   7 months ago    122MB
k8s.gcr.io/kube-scheduler                 v1.20.7    38f903b54010   7 months ago    47.3MB
gcr.io/k8s-minikube/storage-provisioner   v5         6e38f40d628d   8 months ago    31.5MB
kubernetesui/dashboard                    v2.1.0     9a07b5b4bfac   12 months ago   226MB
k8s.gcr.io/etcd                           3.4.13-0   0369cf4303ff   15 months ago   253MB
k8s.gcr.io/coredns                        1.7.0      bfe3a36ebd25   17 months ago   45.2MB
kubernetesui/metrics-scraper              v1.0.4     86262685d9ab   20 months ago   36.9MB
k8s.gcr.io/pause                          3.2        80d28bedfe5d   22 months ago   683kB

C:\Users\basan>
```

Step 2 :

Create a spring boot project then add Dockerfile and next build a docker image

GitHub source code link : https://github.com/Java-Techie-jt/springboot-k8s-example



Build docker image

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>docker build -t springboot-k8s-app:1.0 .
Sending build context to Docker daemon  17.89MB
Step 1/4 : FROM openjdk:8
8: Pulling from library/openjdk
5e0b432e8ba9: Pull complete
a84cfd68b5ce: Pull complete
e8b8f2315954: Pull complete
0598fa43a7e7: Pull complete
e0d35e3be804: Pull complete
cc526d02f40c: Pull complete
94f9f735b512: Pull complete
Digest: sha256:d847fdd469a97814a8c118bdb887402a629539002a8c95e4c288ba9389023273
Status: Downloaded newer image for openjdk:8
 ---> 5bbce51c9625
Step 2/4 : EXPOSE 8080
 ---> Running in d2d6eadc5281
Removing intermediate container d2d6eadc5281
 ---> d4387afbf6db
Step 3/4 : ADD target/springboot-k8s-demo.jar springboot-k8s-demo.jar
 ---> a88d9d4e8cd6
Step 4/4 : ENTRYPOINT ["java","-jar","/springboot-k8s-demo.jar"]
 ---> Running in 42699817fbbe
Removing intermediate container 42699817fbbe
 ---> 289317d3e22c
Successfully built 289317d3e22c
Successfully tagged springboot-k8s-app:1.0
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '
-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

view docker image in k8s

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>docker images
REPOSITORY                                TAG         IMAGE ID       CREATED          SIZE
springboot-k8s-app                        1.0         289317d3e22c   3 minutes ago    544MB
openjdk                                   8           5bbce51c9625   7 days ago       526MB
k8s.gcr.io/kube-proxy                     v1.20.7     ff54c88b8ecf   7 months ago     118MB
k8s.gcr.io/kube-apiserver                 v1.20.7     034671b24f0f   7 months ago     122MB
k8s.gcr.io/kube-controller-manager        v1.20.7     22d1a2072ec7   7 months ago     116MB
k8s.gcr.io/kube-scheduler                 v1.20.7     38f903b54010   7 months ago     47.3MB
gcr.io/k8s-minikube/storage-provisioner   v5          6e38f40d628d   8 months ago     31.5MB
kubernetesui/dashboard                    v2.1.0      9a07b5b4bfac   12 months ago    226MB
k8s.gcr.io/etcd                           3.4.13-0    0369cf4303ff   15 months ago    253MB
k8s.gcr.io/coredns                        1.7.0       bfe3a36ebd25   17 months ago    45.2MB
kubernetesui/metrics-scraper              v1.0.4      86262685d9ab   20 months ago    36.9MB
k8s.gcr.io/pause                          3.2         80d28bedfe5d   22 months ago    683kB

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

Step 3 :

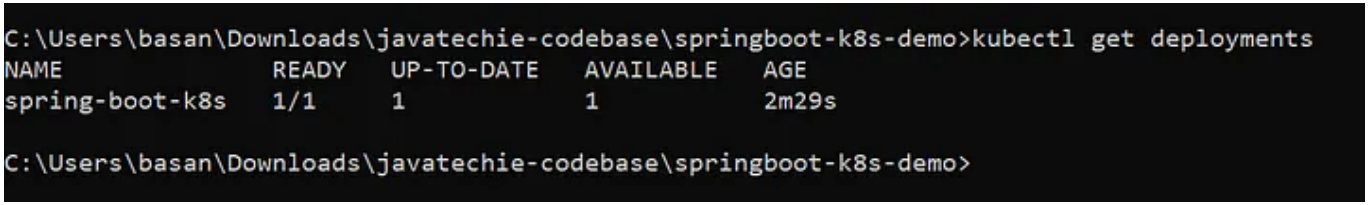
Create Deployment Object , as we know Deployments are Kubernetes objects that are used for managing pods. we can describe deployment object details using YML file but for this example let's play with command .

```
kubectl create deployment spring-boot-k8s --image=springboot-k8s-
app:1.0 --port=8080
```

With above command we are telling to k8s , create a deployment with name spring-boot-k8s and read the docker image springboot-k8s-app:1.0 then next create a pod and run my image inside containers

Verify deployment status

```
kubectl get deployments
```



Describe deployment object

```
kubectl describe deployment spring-boot-k8s
```

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl get deployments
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
spring-boot-k8s   1/1     1            1           2m29s

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl describe deployments spring-boot-k8s
Name:                   spring-boot-k8s
Namespace:              default
CreationTimestamp:      Thu, 09 Dec 2021 21:29:54 +0530
Labels:                 app=spring-boot-k8s
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=spring-boot-k8s
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=spring-boot-k8s
  Containers:
   springboot-k8s-app:
    Image:        springboot-k8s-app:1.0
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   spring-boot-k8s-58dc4b544f (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m19s  deployment-controller  Scaled up replica set spring-boot-k8s-58dc4b544f to 1

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

Now to ensure that Kubernetes successfully pull my docker image and run it inside a pods we can execute below command

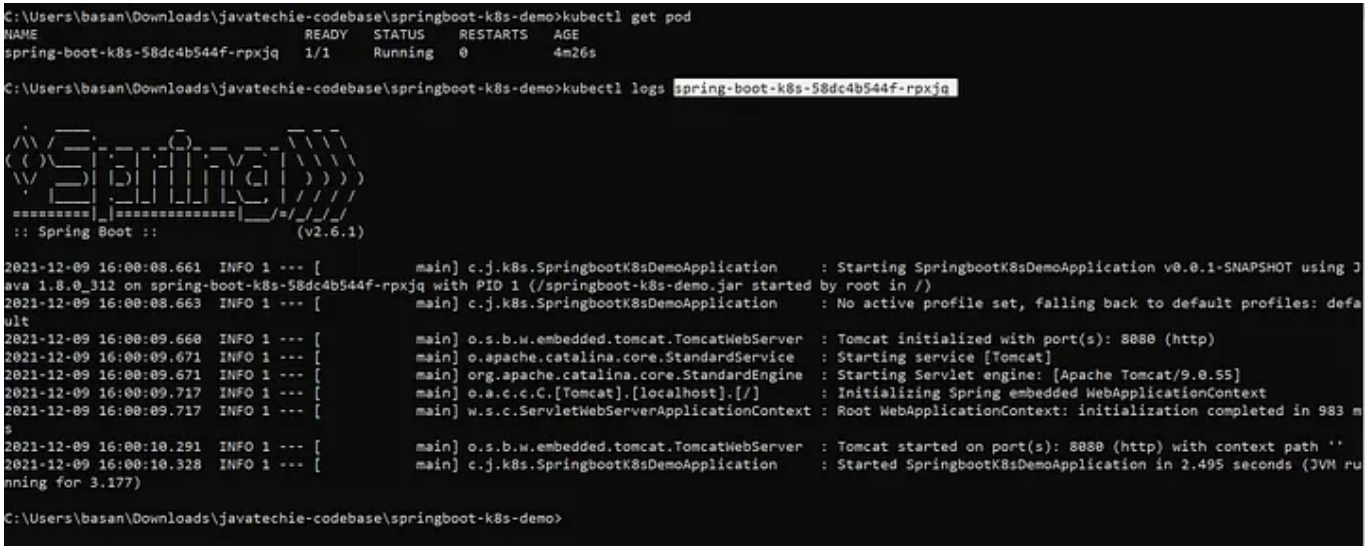
verify pod created

```
kubectl get pods
```

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
spring-boot-k8s-58dc4b544f-rpxjq    1/1     Running   0          4m26s

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

Validate docker image running inside pod

```
kubectl logs spring-boot-k8s-58dc4b544f-rpxjq
```



We are good now , our application is running inside k8s pods . now to expose this application to outside world we need to create service object .To create a Service object we need to exposes the deployment with specific service type

Step 4 :

Use below command to create service obsject

```
kubectl expose deployment spring-boot-k8s --type=NodePort
```

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl expose deployment spring-boot-k8s --type=NodePort
service/spring-boot-k8s exposed

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

Once service created you can verify that

```
kubectl get service or kubectl get svc
```

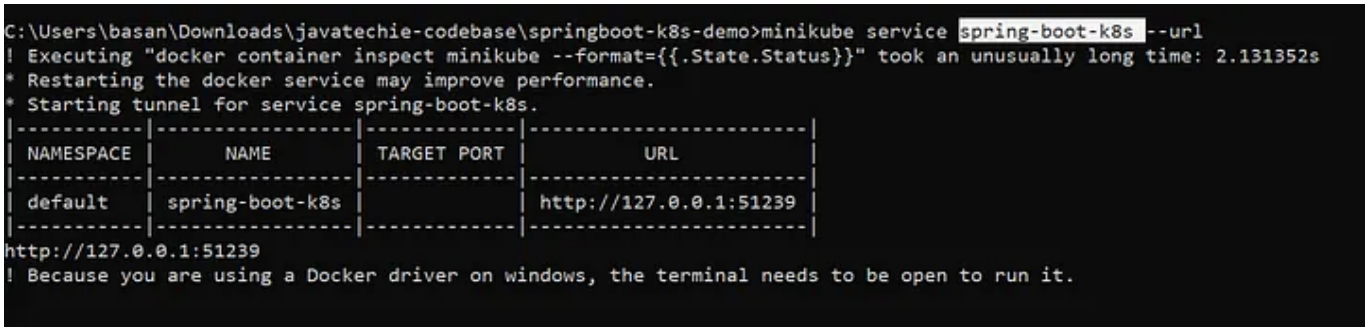
```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl get service
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP          31m
spring-boot-k8s   NodePort    10.107.65.110   <none>        8080:31388/TCP   41s

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

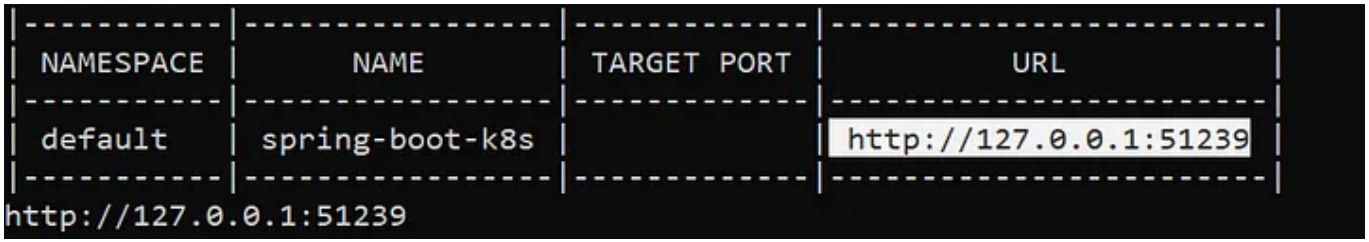
As we know all traffic will come to service and then service will redirect your request to corresponding pods based on available . since we have only one pod we can directly get the service url to access it .

Step 5 :

Start tunnel or get the proxy url of service to access it .

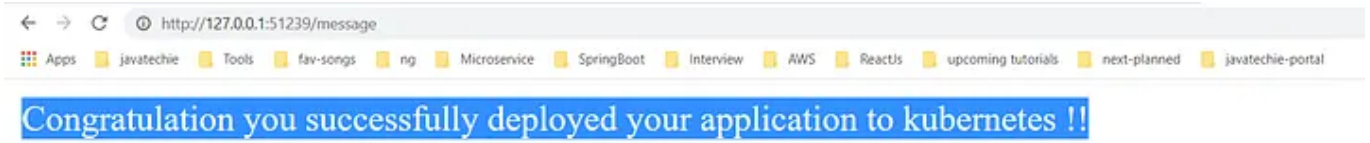
```
minikube service spring-boot-k8s --url
```





Step 6 :

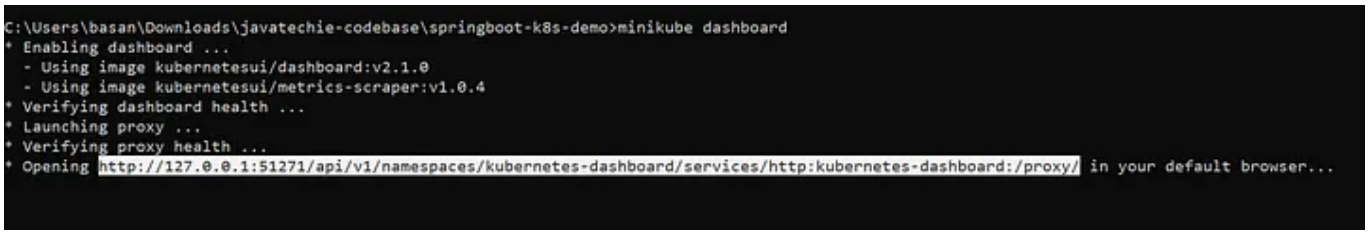
Access the url



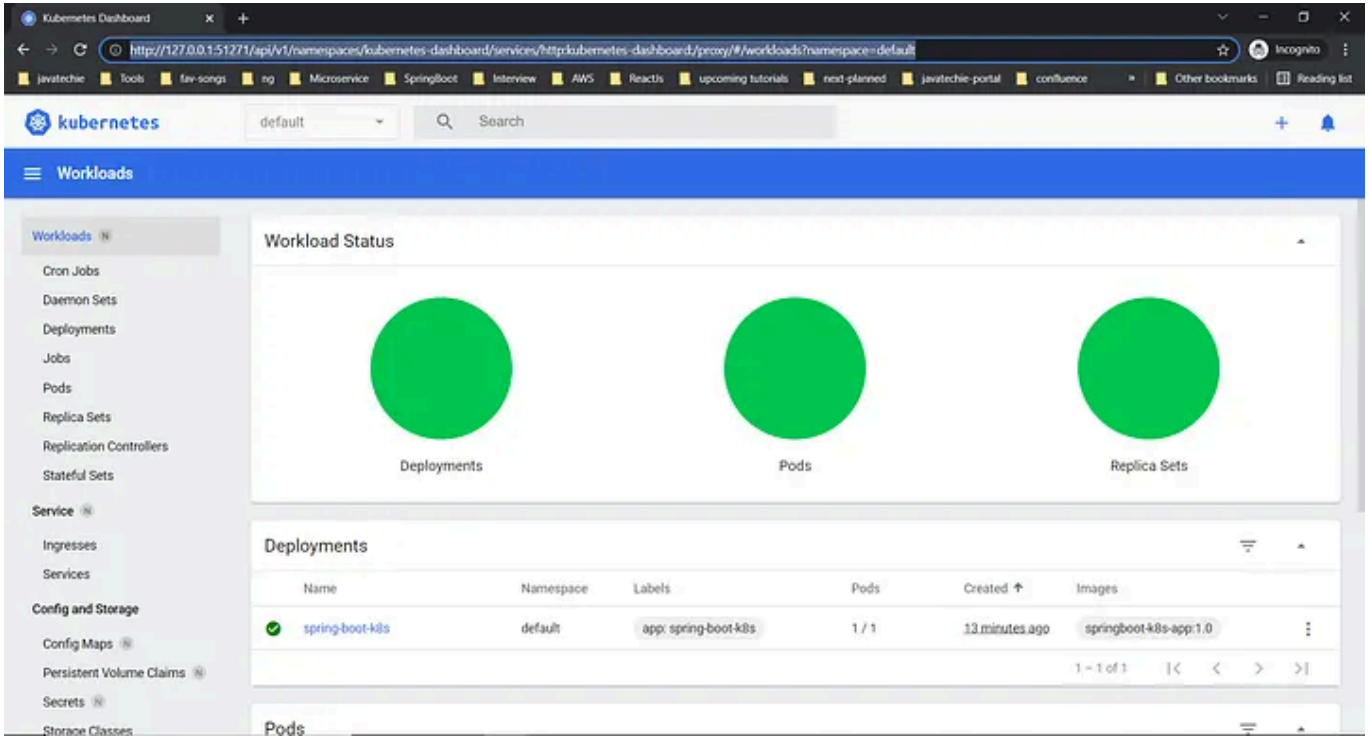
Step 7 :
You can visualize health of your pods, service and deployment using Kubernetes dashboard

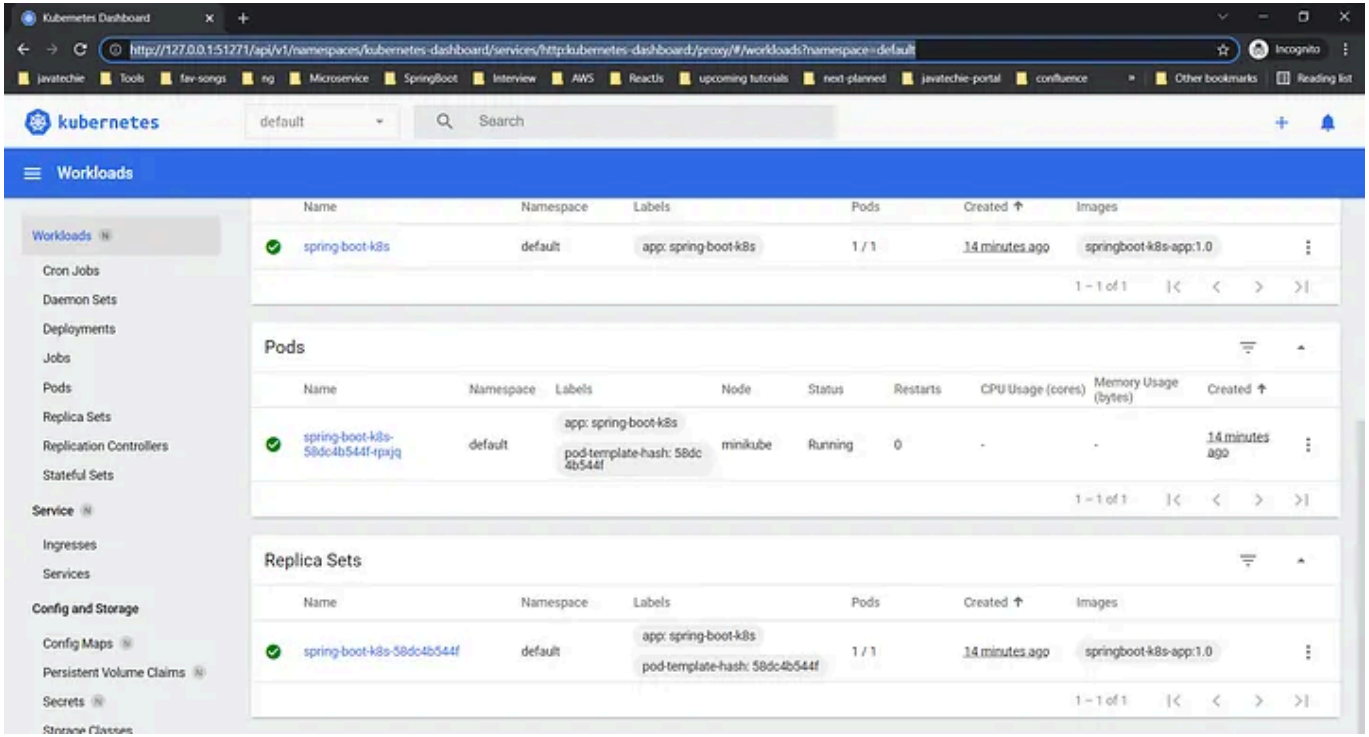
```
minikube dashboard
```

This will enable the dashboard add-on, and open the proxy in the default web browser.



You can access the above url to watch your k8s dashboard


Kubernetes Dashboard
http://127.0.0.1:51271/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/workloads?namespace=default
Incognito
kubernetes
default
Search
Workloads
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Storage Classes
Workload Status
Deployments
Pods
Replica Sets
Deployments
Name
Namespace
Labels
Pods
Created
Images
spring-boot-k8s
default
app: spring-boot-k8s
1 / 1
13 minutes ago
springboot-k8s-app:1.0
1 – 1 of 1
Pods



Deployment completed .....

Clean up local state

Step 8 :

Delete Service

```
kubectl delete service spring-boot-k8s
```

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl delete service spring-boot-k8s
service "spring-boot-k8s" deleted

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

Step 9 :

Delete Deployment

```
kubectl delete deployment spring-boot-k8s
```

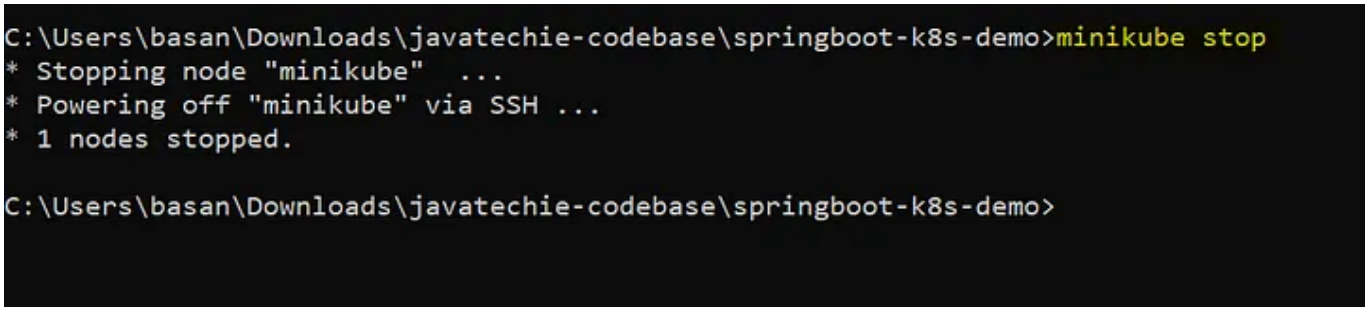
```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl delete deployment spring-boot-k8s
deployment.apps "spring-boot-k8s" deleted

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

Step 10 :

Stop minikube

```
minikube stop
```

Stops a local Kubernetes cluster. This command stops the underlying VM or container, but keeps user data intact. The cluster can be started again with the “start” command.



Step 11 :

Delete minikube

```
minikube delete
```

Deletes a local Kubernetes cluster. This command deletes the VM, and removes all associated files.

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>minikube delete
* Deleting "minikube" in docker ...
* Deleting container "minikube" ...
* Removing C:\Users\basan\.minikube\machines\minikube ...
* Removed all traces of the "minikube" cluster.

C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>
```

What we learned ?

1. How to install minikube in windows OS or How do we setup Kubernetes in windows
2. Kubernetes basic commands
3. How to deploy spring boot application to local Kubernetes cluster .
4. Kubernetes Dashboard and health check .

If you like this article then please do share with your colleagues ..

Kubernetes | Spring Boot | Javatechie | Minikube

Written by Java Techie

6.8K followers · 36 following

Go One Step ahead

Follow

Responses (1)

Write a response

What are your thoughts?

Jitendra
Apr 13, 2022

At step 5:

I am getting below proxy url, which is not working in my case.

minikube service spring-boot-k8s --url

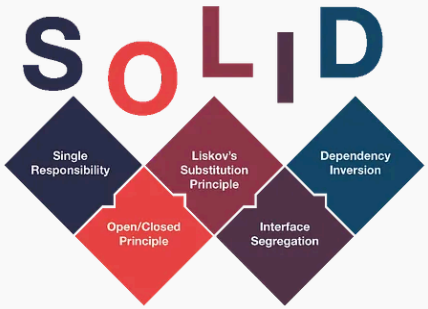
http://192.168.49.2:30873

* Starting tunnel for service spring-boot-k8s.

! Because you are using a Docker driver on windows, the terminal... more

2 replies Reply

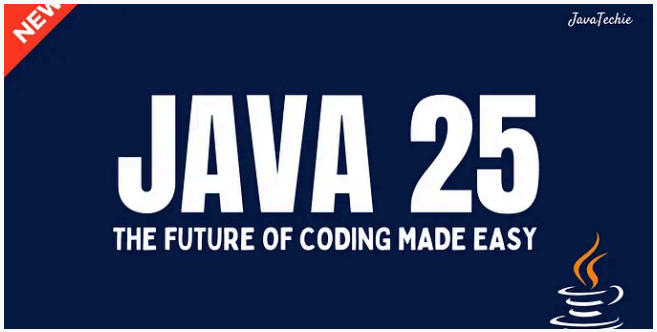
More from Java Techie



Java Techie

SOLID Design Principle Java

In this tutorial we will discuss about SOLID principle and its importance in Software...



Java Techie

Java 25: The Future of Coding Made Easy

🔥 Java 25 is almost here! The next long-term support release (LTS) drops in...

Sep 19, 2021 1K 28

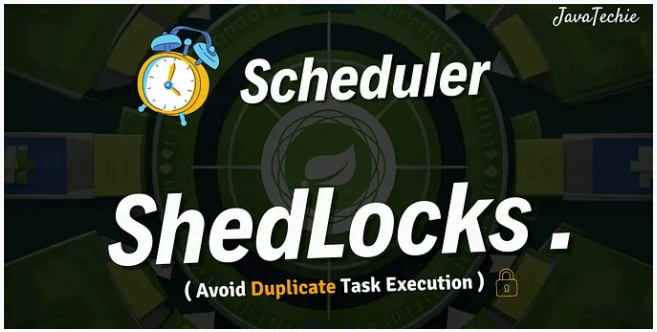
Java Techie

The Evolution Of Switch Statement From Java 7 to Java 17

Hi Folks ,

Dec 24, 2021 624 11

Oct 4 37



Java Techie

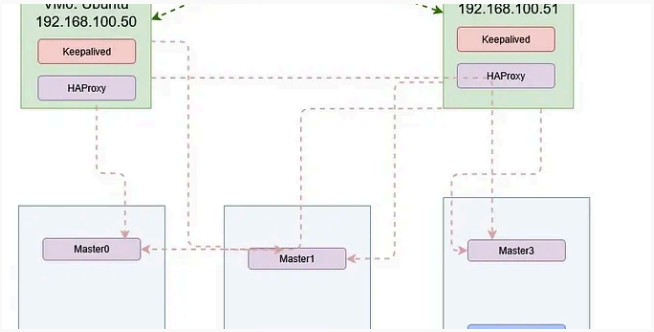
ShedLock in Spring Scheduler: Prevent Duplicate Execution in...

Why Shedlock?

Mar 2 30 1

See all from Java Techie

Recommended from Medium



In Stackademic by Jamal Mahmoudi

Creating a Kubernetes Cluster (On-Premises) Production & Stage...

Note

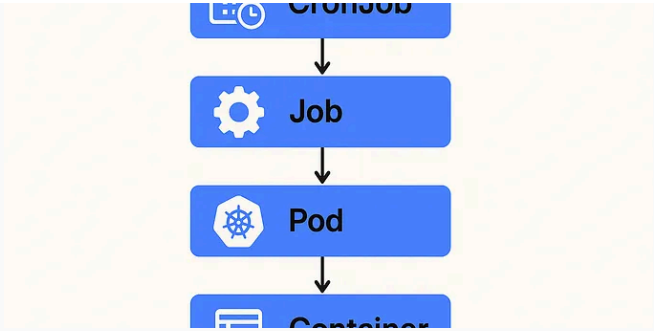
Jul 21 33 2

Freddie A

Goodbye Helm Charts—I Finally Found a Better Way to Deploy to...

Why I stopped wrestling with YAML Tetris and found a saner path to shipping code.

Sep 11 262 23



In DevOps.dev by Salwan Mohamed

Mastering Kubernetes Jobs and CronJobs: A Platform Engineer's...

A comprehensive deep-dive into finite workloads, batch processing, and schedule...

Jun 13 43

In ITNEXT by Ivan Franchin

What is the Best Embedded Web Server for Spring Boot version...

A Comparative Analysis of Tomcat, Jetty, and Undertow for Spring Boot apps using Spring...

May 6 67 4

Layer	Test Type	Tools
Domain	Unit Tests	JUnit, Mockito
UseCase	Unit/Mock Tests	JUnit
Adapters	Integration	TestContainers
Controller	Web Test	Spring MockMvc

DevOpsDynamo

Mastering Helm: Best Practices for Multi-Environment Kubernetes...

In Level Up Coding by Rahul Soni

🧱 Hexagonal Architecture in Spring Boot Microservices: A...

🆓 Not a Medium member? You can still read this full story for free — no paywall, no catch...

May 15 254

Hexagonal Architecture, also known as Ports and Adapters, is a powerful pattern for...

Jul 11 184 5

See more recommendations