

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



Java Techie

Follow

6 min read · Dec 10, 2021



47



1





## Prerequisite

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

**M** inikube 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 .

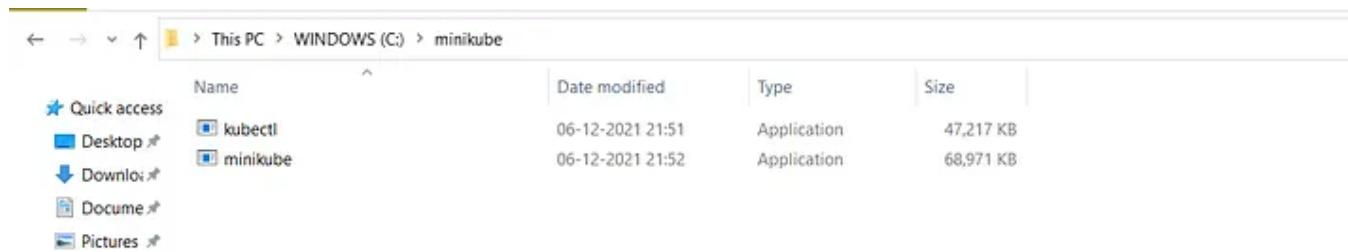
Step 2 :

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

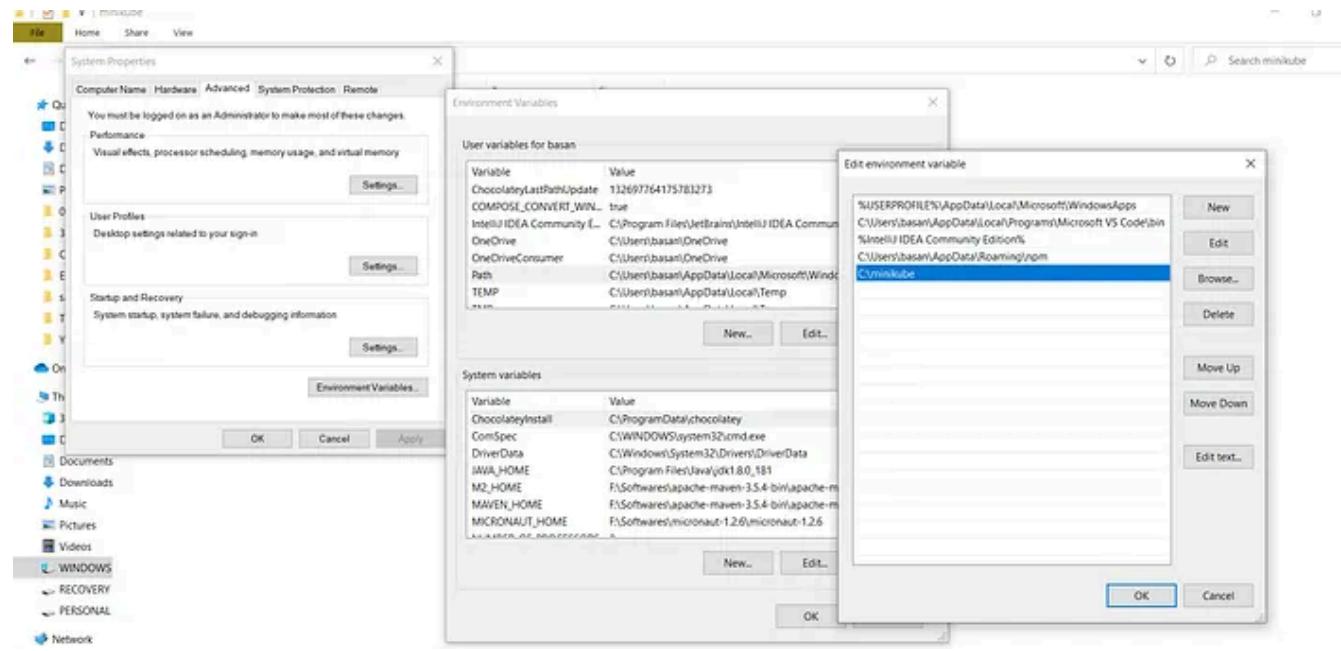
Step 3:

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

```
C:\Users\basan>minikube start --driver=docker
* minikube v1.21.0 on Microsoft Windows 10 Home Single Language 10.0.18363 Build 18363
* minikube 1.24.0 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.24.0
* To disable this notice, run: 'minikube config set WantUpdateNotification false'

* Using the docker driver based on user configuration
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.20.7 on Docker 20.10.7 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Step 7 :

After successfully started minikube , you can verify minikube status

```
minikube status
```

```
C:\Users\basan>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubecfg: Configured
```

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

## D eploy 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/etcld         3.4.13-0  0369cf4303ff  15 months ago  253MB
k8s.gcr.io/coredns       1.7.0    bfe3a36ebd25  17 months ago  45.2MB
kubernetesui/metrics-scrapers  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>

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>dir
Volume in drive C is WINDOWS
Volume Serial Number is E08D-DAD9

Directory of C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo

07-12-2021  23:41    <DIR>          .
07-12-2021  23:41    <DIR>          ..
07-12-2021  17:55            395 .gitignore
07-12-2021  23:38    <DIR>          .idea
07-12-2021  17:55    <DIR>          .mvn
07-12-2021  23:40            142 Dockerfile
07-12-2021  17:55            891 HELP.md
07-12-2021  17:55            10,070 mvnw
07-12-2021  17:55            6,608 mvnw.cmd
07-12-2021  23:38            1,305 pom.xml
07-12-2021  23:38            7,693 springboot-k8s-demo.iml
07-12-2021  17:55    <DIR>          src
07-12-2021  23:41    <DIR>          target
                           7 File(s)        27,104 bytes
                           6 Dir(s)   367,437,398,016 bytes free

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

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
a84cf6d68b5ce: Pull complete
e8b3bf2315954: Pull complete
e598fa43a7e7: 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
    ---> d4387afb6db
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

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	9a07b5b4bfa	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-scrapers	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 :

---

Get Java Techie's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

Subscribe

---

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
```

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>kubectl create deployment spring-boot-k8s --image=springboot-k8s-app:1.0 --port=8080
deployment.apps/spring-boot-k8s created
```

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
```

```
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>
```

## 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:  8/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
```

```
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>kubectl logs spring-boot-k8s-58dc4b544f-rpxjq
.
.
.
:: Spring Boot ::      (v2.6.1)

2021-12-09 16:00:08.661  INFO 1 --- [           main] c.j.k8s.SpringbootK8sDemoApplication : Starting SpringbootK8sDemoApplication v0.0.1-SNAPSHOT using Java 1.8.0_312 on spring-boot-k8s-58dc4b544f-rpxjq with PID 1 (/springboot-k8s-demo.jar started by root in /)
2021-12-09 16:00:08.663  INFO 1 --- [           main] c.j.k8s.SpringbootK8sDemoApplication : No active profile set, falling back to default profiles: default
2021-12-09 16:00:09.668  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-12-09 16:00:09.671  INFO 1 --- [           main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-12-09 16:00:09.671  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.55]
2021-12-09 16:00:09.717  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2021-12-09 16:00:09.717  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 983 ms
2021-12-09 16:00:10.291  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-12-09 16:00:10.328  INFO 1 --- [           main] c.j.k8s.SpringbootK8sDemoApplication : Started SpringbootK8sDemoApplication in 2.495 seconds (JVM running for 3.177)

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

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 object

```
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
```

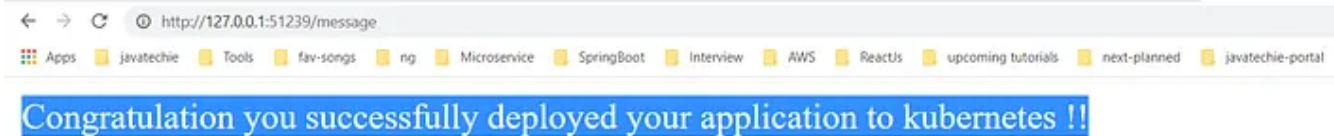
```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>minikube service spring-boot-k8s --url
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.131352s
* Restarting the docker service may improve performance.
* Starting tunnel for service spring-boot-k8s.
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | spring-boot-k8s | | http://127.0.0.1:51239 |
|-----|-----|-----|-----|
http://127.0.0.1:51239
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

NAMESPACE	NAME	TARGET PORT	URL
default	spring-boot-k8s		http://127.0.0.1:51239

http://127.0.0.1:51239

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.

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>minikube dashboard
* Enabling dashboard ...
  - Using image kubernetesui/dashboard:v2.1.0
  - Using image kubernetesui/metrics-scraper:v1.0.4
* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:51271/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default 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/http:kubernetes-dashboard/proxy/#/workloads?namespace=default

Incognito

javatechie Tools fav-songs ng Microservice SpringBoot Interview AWS Reactjs upcoming tutorials next-planned javatechie-portal confluence Other bookmarks Reading list

kubernetes default Search

Workloads

Workloads (8)

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Service (8)

- Ingresses
- Services
- Config and Storage
- Config Maps (8)
- Persistent Volume Claims (8)
- Secrets (8)
- Storage Classes

Workload Status

Deployments

Pods

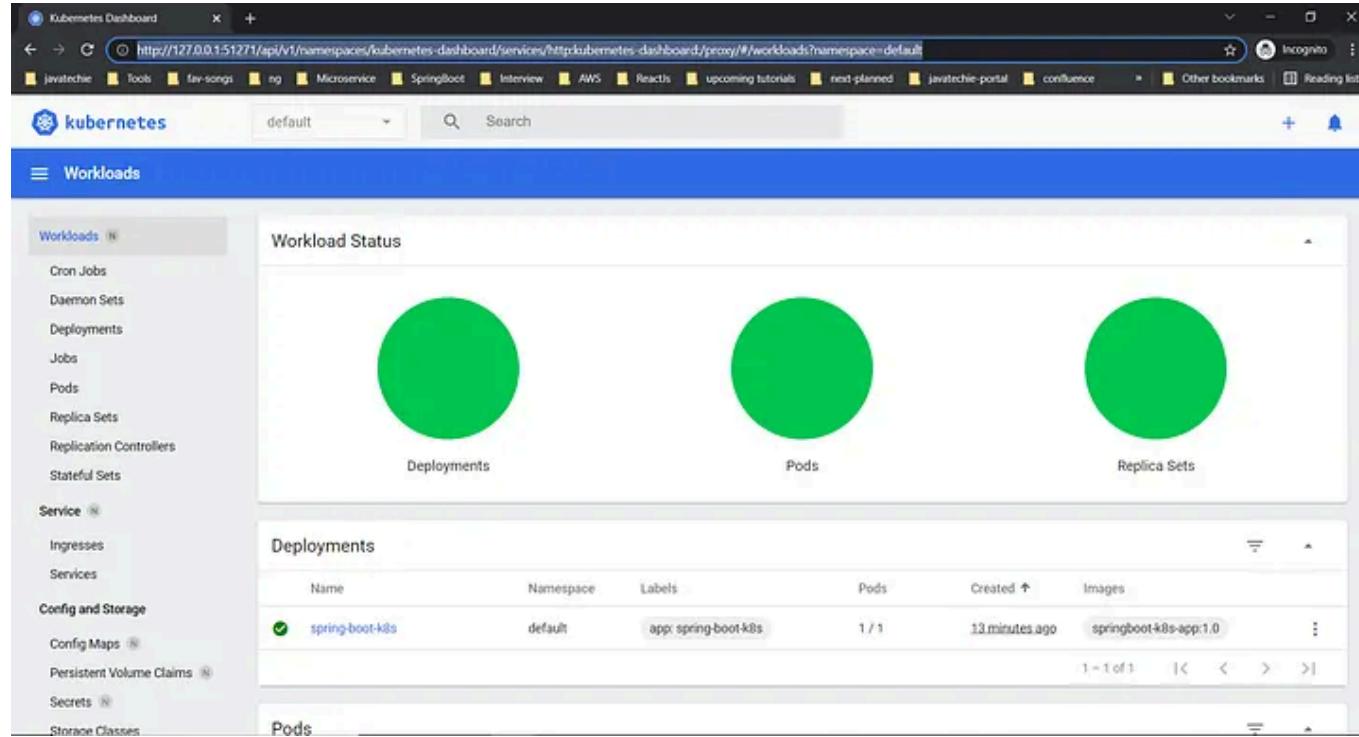
Replica Sets

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



The screenshot shows the Kubernetes Dashboard interface. The left sidebar has a navigation menu with the following items:

- 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

The main content area is titled "Workloads" and shows three tabs: "Workloads", "Pods", and "Replica Sets".

- Workloads:** Shows one entry: "spring-boot-k8s" in the "default" namespace with label "app: spring-boot-k8s". Status: 1/1, Created: 14 minutes ago, Image: "springboot-k8s-app:1.0".
- Pods:** Shows one entry: "spring-boot-k8s-50dcb544f-rpqjq" in the "default" namespace with labels "app: spring-boot-k8s" and "pod-template-hash: 50dcb544f". Node: minikube, Status: Running, Restarts: 0, CPU Usage: -, Memory Usage: -, Created: 14 minutes ago.
- Replica Sets:** Shows one entry: "spring-boot-k8s-50dcb544f" in the "default" namespace with labels "app: spring-boot-k8s" and "pod-template-hash: 50dcb544f". Status: 1/1, Created: 14 minutes ago, Image: "springboot-k8s-app:1.0".

**D**eployment 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.

```
C:\Users\basan\Downloads\javatechie-codebase\springboot-k8s-demo>minikube stop
* Stopping node "minikube" ...
* Powering off "minikube" via SSH ...
* 1 nodes stopped.

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

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

Follow

Go One Step ahead

## 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.

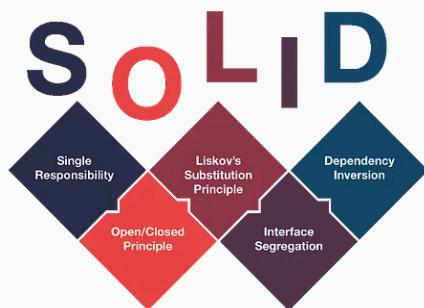
[! Because you are using a Docker driver on windows, the terminal... more](#)



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



Oct 4 37



 Java Techie

## The Evolution Of Switch Statement From Java 7 to Java 17

Hi Folks ,

Dec 24, 2021 624 11



Mar 2 30 1



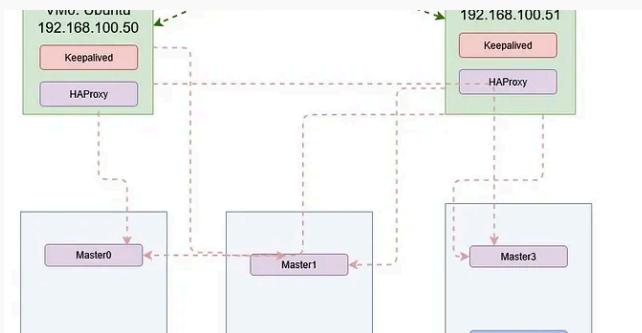
 Java Techie

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

Why Shedlock?

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



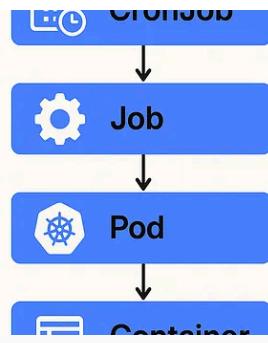
Freddie A

### 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



 DevOpsDynamo

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

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

 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