#k8s #node #pod #deployment #secret #service #namespace #configmap

- **Name**: Iskander Nafikov
- **E-mail**: i.nafikov@innopolis.university
- **GitHub**: https://github.com/iskanred/iu-devsecops-course/tree/main
- **DockerHub**: https://hub.docker.com/repository/docker/iskanred
- **Username**: `i.nafikov`
- **Hostname**: `macbook-KN70WX2HPH`

Task 1 - Preparation

1.1

> **Task description**
>
> Choose an application that has such characteristics as (points with "star" are not required but recommended):
>
> - any non-confidential data like in JSON, XML or TOML format
> - application configuration values (or environment variables)
> - any secrets (like authorization credentials)
> - availability from outside (on Internet) *
> - has any health checks or/and status endpoints *
>
> It could be any yours application or open-source project that meets the requirements. You are even able to work with different applications in different lab tasks but try to avoid it. However, it's **much better** to work with the single project but it's acceptable to involve only the required application functionality regarding the particular task requirements. Put the git link to your chosen project into report and provide a brief project description for what it is and how it works.

- For this lab I updated my application `current-time-server` from the previous lab. I added new `/secret` endpoint to satisfy the requirement to have any secrets.
- Below is the brief project description of this application:

> The service provides information about current time in Europe/Moscow timezone and also shows the number of getting this information (number of visits). Moreover, it provides a hashed value of a secret string that is configured as an environment variable.

- You can check more detailed description in the `README.md` of the application.

- Here are the links:
  - **GitHub**: https://github.com/iskanred/iu-devsecops-course/tree/main/lab-04/current-time-server/
  - **DockerHub**: https://hub.docker.com/repository/docker/iskanred/current-time-server/general

1.2

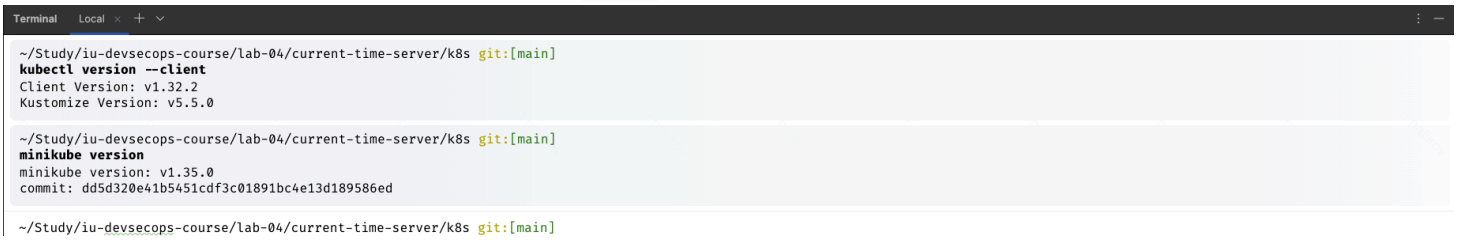
> **Task description**
>
> Get familiar with Kubernetes (k8s) and concepts.

✅

1.3

> **Task description**
>
> Install and set up the necessary tools:
>
> - kubectl
> - minikube or its alternative

- I installed these tools successfully using `brew`



- What is more, I copied kube config values for `minikube` into a single `kubeconfig.yaml` file in a working directory

Local × kubeconfig.yaml

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat kubeconfig.yaml
apiVersion: v1

kind: Config
current-context: minikube

clusters:
  - name: minikube
    cluster:
      server: https://127.0.0.1:32769
      certificate-authority: /Users/i.nafikov/.minikube/ca.crt
      extensions:
        - name: cluster_info
          extension:
            provider: minikube.sigs.k8s.io
            version: v1.35.0
            last-update: Tue, 18 Feb 2025 16:47:47 MSK

contexts:
  - name: minikube
    context:
      cluster: minikube
      user: minikube
      namespace: default
      extensions:
        - name: context_info
          extension:
            provider: minikube.sigs.k8s.io
            version: v1.35.0
            last-update: Tue, 18 Feb 2025 16:47:47 MSK

users:
  - name: minikube
    user:
      client-certificate: /Users/i.nafikov/.minikube/profiles/minikube/client.crt
      client-key: /Users/i.nafikov/.minikube/profiles/minikube/client.key
```

- Below is the content of this file

```yaml
apiVersion: v1

kind: Config
current-context: minikube

clusters:
  - name: minikube
    cluster:
      server: https://127.0.0.1:32769
      certificate-authority: /Users/i.nafikov/.minikube/ca.crt
      extensions:
        - name: cluster_info
          extension:
            provider: minikube.sigs.k8s.io
            version: v1.35.0
            last-update: Tue, 18 Feb 2025 16:47:47 MSK

contexts:
  - name: minikube
    context:
      cluster: minikube
      user: minikube
      namespace: default
      extensions:
        - name: context_info
          extension:
            provider: minikube.sigs.k8s.io
            version: v1.35.0
```

```
            last-update: Tue, 18 Feb 2025 16:47:47 MSK

users:
  - name: minikube
    user:
      client-certificate: /Users/i.nafikov/.minikube/profiles/minikube/client.crt
      client-key: /Users/i.nafikov/.minikube/profiles/minikube/client.key
```

- I can check if I copied properties correctly

```
kubectl --kubeconfig kubeconfig.yaml config get-contexts
```

- Seem so!

1.4

> **Task description**
>
> Get access to Kubernetes Dashboard.

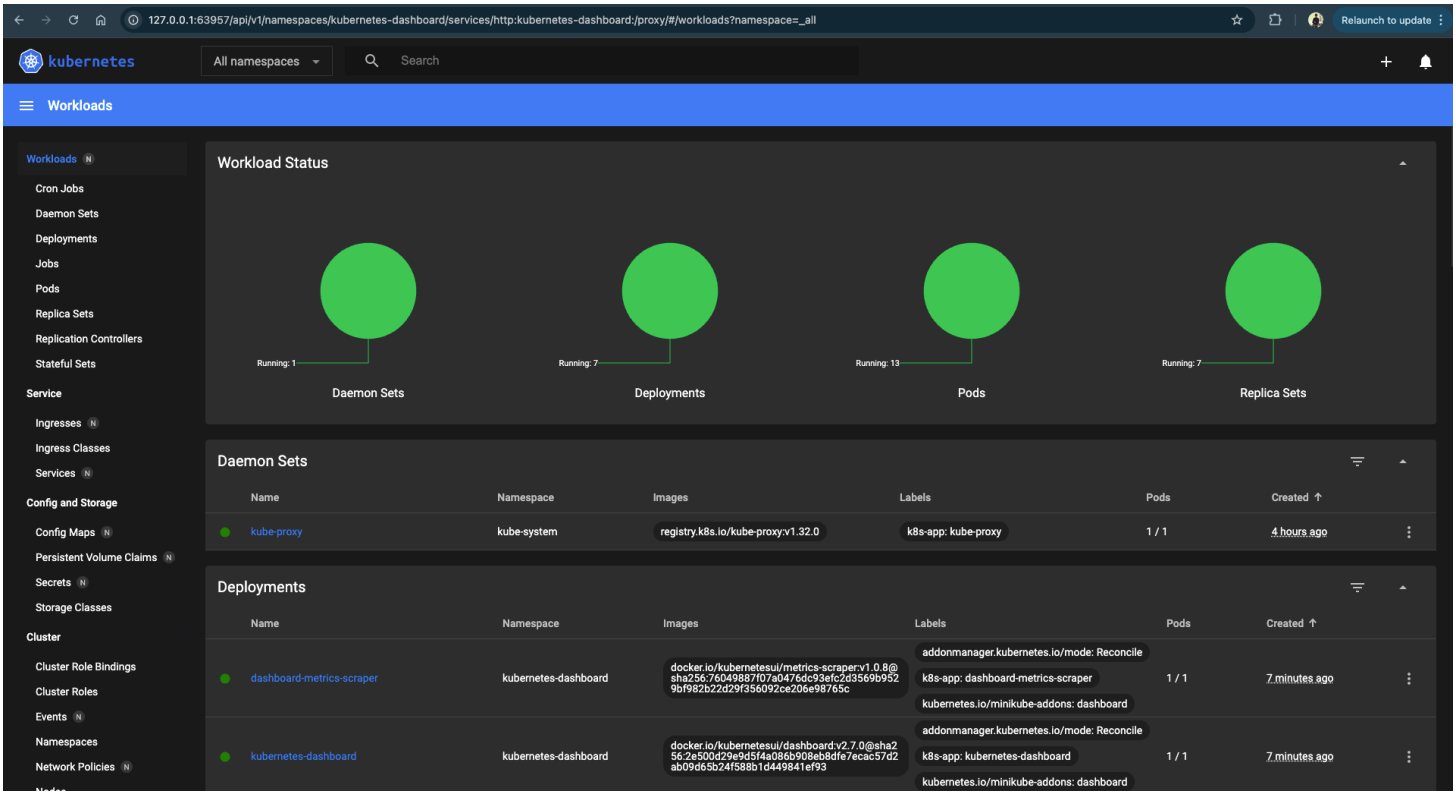
- I accessed Kubernetes Dashboard using `minkube`

```
minikube dashboard
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
minikube dashboard
🔌  Enabling dashboard ...
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:63957/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
```



- Also, it's better to mention in the task description how to access it because at first I tried to deploy it by my own using **helm** by this instruction but got into a trouble 😅

Task 2 - k8s Nodes

> We are going to start to learn Kubernetes from kubectl command line that is dedicated to work with k8s cluster

1

🖊 Task description

After installing all required lab tools and starting `minikube` cluster, use `kubectl` commands to `get` and `describe` your cluster nodes.

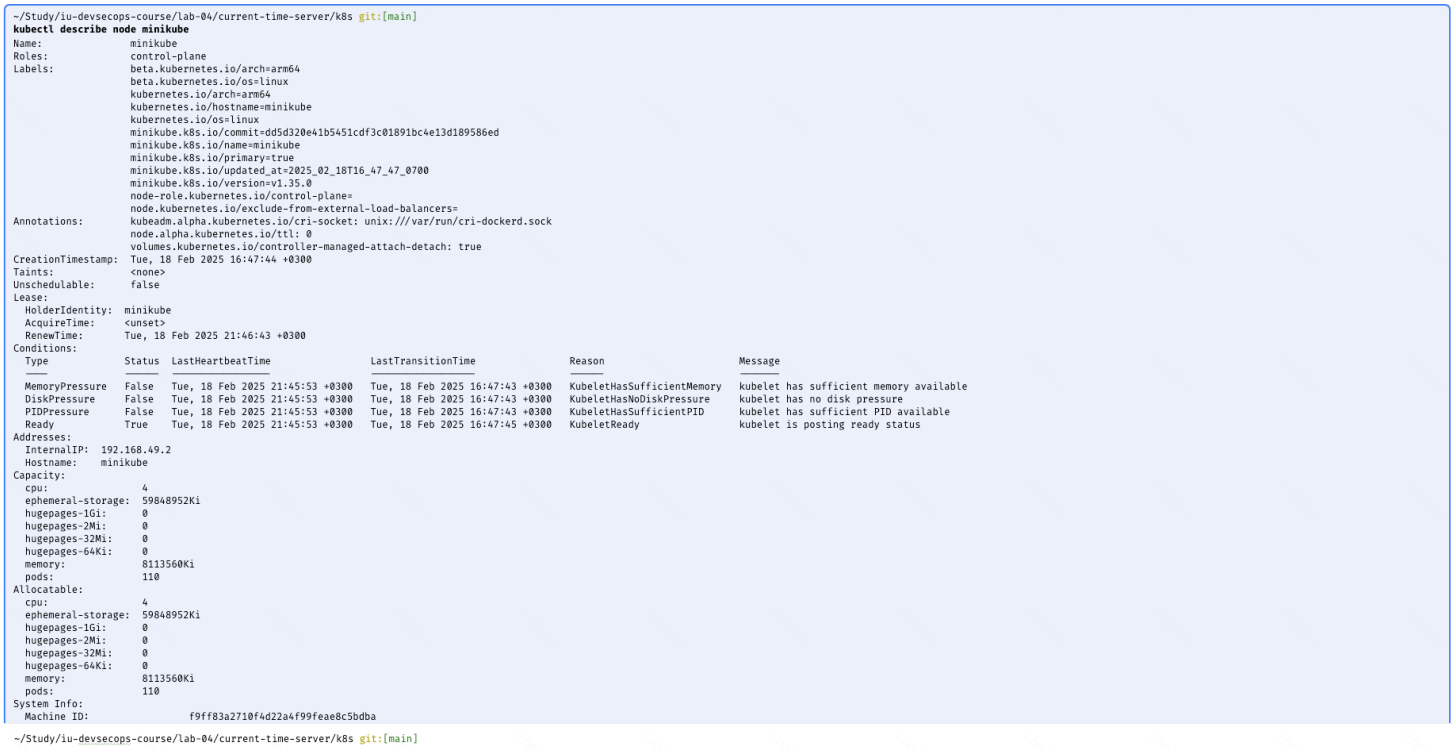
- Using `get` we can get list our nodes

```
kubectl get node
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get node
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   4h56m   v1.32.0
```

- We see that there is only one node in our cluster: `minikube` with a `control-plane` role
- Using `describe` we can get detailed information about our nodes

```
kubectl describe node minikube
```



```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe node minikube
Name:               minikube
Roles:              control-plane
Labels:             beta.kubernetes.io/arch=arm64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=arm64
                    kubernetes.io/hostname=minikube
                    kubernetes.io/os=linux
                    minikube.k8s.io/commit=dd5d320e41b5451cdf3c01891bc4e13d189586ed
                    minikube.k8s.io/name=minikube
                    minikube.k8s.io/primary=true
                    minikube.k8s.io/updated_at=2025_02_18T16_47_47_0700
                    minikube.k8s.io/version=v1.35.0
                    node-role.kubernetes.io/control-plane=
                    node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/cri-dockerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 18 Feb 2025 16:47:44 +0300
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  minikube
  AcquireTime:     <unset>
  RenewTime:       Tue, 18 Feb 2025 21:46:43 +0300
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  MemoryPressure   False   Tue, 18 Feb 2025 21:45:53 +0300   Tue, 18 Feb 2025 16:47:43 +0300   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Tue, 18 Feb 2025 21:45:53 +0300   Tue, 18 Feb 2025 16:47:43 +0300   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Tue, 18 Feb 2025 21:45:53 +0300   Tue, 18 Feb 2025 16:47:43 +0300   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Tue, 18 Feb 2025 21:45:53 +0300   Tue, 18 Feb 2025 16:47:45 +0300   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  192.168.49.2
  Hostname:    minikube
Capacity:
  cpu:                4
  ephemeral-storage:  59848952Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  hugepages-32Mi:     0
  hugepages-64Ki:     0
  memory:             8113560Ki
  pods:               110
Allocatable:
  cpu:                4
  ephemeral-storage:  59848952Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  hugepages-32Mi:     0
  hugepages-64Ki:     0
  memory:             8113560Ki
  pods:               110
System Info:
  Machine ID:                 f9ff83a2710f4d22a4f99feae8c5bdba
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

- We see much information there: address of a node inside the Docker cotnainer, allocated resources, system info, k8s events, and etc.

2

Task description

Get the more detailed information about the particular node (it's fine if you have one node in the cluster).

- Basically the most detailed information can be obtained using `describe`
- However, we can also specify `wide` output for `get` command

```
kubectl get –o wide
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get node minikube -o wide
NAME       STATUS   ROLES           AGE     VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION     CONTAINER-RUNTIME
minikube   Ready    control-plane   5h20m   v1.32.0   192.168.49.2   <none>        Ubuntu 22.04.5 LTS   6.8.0-31-generic   docker://27.4.1
```

3

> **Task description**
>
> Get the OS and CPU information.

- We can use `--output` or `-o` flag with an option `custom-columns` to specify what pieces of information we want to output

```
kubectl get node minikube -o=custom-
columns='OS:.status.nodeInfo.osImage,CPU:.status.nodeInfo.architecture'
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get node minikube -o=custom-columns='OS:.status.nodeInfo.osImage,CPU:.status.nodeInfo.architecture'
OS                  CPU
Ubuntu 22.04.5 LTS  arm64
```

- Also, we can use `jsonpath` option

```
kubectl get node -o jsonpath="{range .items[*]}{'Name: '}{.metadata.name}{'\t'}
{'OS: '}{.status.nodeInfo.osImage}{'\t'}{'CPU: '}{.status.nodeInfo.architecture}
{'\n'}{end}"
```

```
Local  Local 2
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get nodes -o jsonpath="{range .items[*]}{'Name: '}{.metadata.name}{'\t'}{'OS: '}{.status.nodeInfo.osImage}{'\t'}{'CPU: '}{.status.nodeInfo.architecture}{'\n'}{end}"
Name: minikube  OS: Ubuntu 22.04.5 LTS  CPU: arm64
```

4

> **Task description**
>
> Put the results into report.

✅

Task 3 - k8s Pod

> `Pod` is one of the simplest and basic k8s unit. It's like an abstraction over Docker containers. Inside pods containers run your application.

1

> **Task description**

Figure out the necessary `Pod` spec fields

- **`apiVersion`** : This field specifies the API version of Kubernetes used.
- **`kind`** : This indicates the type of resource for creating. For pods it is `Pod`
- **`metadata`** : This section contains metadata about the `Pod` , such as its name, namespace, labels, and annotations.
- **`spec`** : This section contains the specification of the `Pod` , including the containers that should be created and ports that should be exposed to k8s virtual network.

2

Task description

Write a `Pod` spec for your chosen application, deploy the application and run a pod

- I created a file `pod.yaml` with `Pod` spec in my working directory

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
ls | grep pod
pod.yaml
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: current-time-server-pod
  labels:
    app: current-time-server
spec:
  containers:
    - name: current-time-server
      image: iskanred/current-time-server:1.0.0
      ports:
        - containerPort: 8080
```

- Here is the `pod.yaml`

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: current-time-server-pod
  labels:
    app: current-time-server
spec:
  containers:
    - name: current-time-server
      image: iskanred/current-time-server:1.0.0
      ports:
        - containerPort: 8080
```

3

Task description

With `kubectl` , get the pods, pod logs, describe pod, go into pod shell.

4

Task description

Make sure that your app is working correctly inside `Pod`

- Using `exec` command I executed a command inside a Pod's container

```
kubectl exec current-time-server-pod -- curl localhost:8080/time -w '\n'
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod -- curl localhost:8080/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0    572      0 --:--:-- --:--:-- --:--:--   615
22:57:36
```

- We see that the applications works

5

Task description

Put the results into report

✅

Task 4 - k8s Service

> We use k8s `Services` to make an application accessible from outside the Kubernetes virtual network, provide an compatible IP address and link to DNS name to pod, to route internal/external traffic within pods.

1

Task description

Figure out the necessary `Service` spec fields

- **`apiVersion`**: This field specifies the API version of Kubernetes used.
- **`kind`**: This indicates the type of resource for creating. For services it is `Service`
- **`metadata`**: This section contains metadata about the `Service`, such as its name, namespace, labels, and annotations
- **`spec`**: This section contains the specification of the `Service`, including the service type, selector for pods
- **`spec.ports`**: This section defines the ports mapping configuration for the Service

2

Task description

Write a `Service` spec for your pod(s) and deploy the `Service`.

- First I change my `Pod` spec and created another one alongside:

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
ls | grep pod
pod-1.yaml
pod-2.yaml
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat pod-1.yaml
apiVersion: v1
kind: Pod
metadata:
  name: current-time-server-pod-1
  labels:
    app: current-time-server
spec:
  containers:
    - name: current-time-server
      image: iskanred/current-time-server:1.0.0
      ports:
        - containerPort: 8080
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat pod-2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: current-time-server-pod-2
  labels:
    app: current-time-server
spec:
  containers:
    - name: current-time-server
      image: iskanred/current-time-server:1.0.0
      ports:
        - containerPort: 8080
```

- Then I created a `Service` using the following spec:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: current-time-server-service
spec:
  type: ClusterIP
  selector:
    app: current-time-server
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
ls | grep service
service.yaml
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: current-time-server-service
spec:
  type: ClusterIP
  selector:
    app: current-time-server
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

- Finally, I deployed them all:

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f pod-1.yaml
pod/current-time-server-pod-1 created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f pod-2.yaml
pod/current-time-server-pod-2 created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f service.yaml
service/current-time-server-service created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po
NAME                     READY   STATUS    RESTARTS   AGE
current-time-server-pod-1   1/1     Running   0          15s
current-time-server-pod-2   1/1     Running   0          12s
```

3

Task description

With `kubectl`, get the `Services` and `describe` them.

- `get`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get svc
NAME                         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
current-time-server-service   ClusterIP   10.105.84.72   <none>        80/TCP    105s
kubernetes                   ClusterIP   10.96.0.1      <none>        443/TCP   6h50m
```

- `describe`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe svc current-time-server-service
Name:                     current-time-server-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=current-time-server
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.105.84.72
IPs:                      10.105.84.72
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
Endpoints:                10.244.0.23:8080,10.244.0.24:8080
Session Affinity:         None
Internal Traffic Policy:  Cluster
Events:                   <none>
```

4

Task description

Make sure that pods can communicate between each other using DNS names, check pods addresses.

- We could see that the service has been actually deployed and explored my pods' endpoints

Properties

Created	2m 52s ago 2025-02-18T23:36:16+03:00
Name	current-time-server-service
Namespace	default
Annotations	endpoints.kubernetes.io/last-change-trigger-time=2025-02-18T20:36:16Z

Subsets

Addresses

IP	Hostname	Target
10.244.0.23		current-time-server-pod-1
10.244.0.24		current-time-server-pod-2

Ports

Port	Name	Protocol
8080		TCP

Events

No events found

IP

- Here are the pods IP addresses inside the cluster

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po current-time-server-pod-1 -o=custom-columns='Pod IP:.status.podIP'
Pod IP
10.244.0.23
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po current-time-server-pod-2 -o=custom-columns='Pod IP:.status.podIP'
Pod IP
10.244.0.24
```

- We can make intercommunication without a service by these IP

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod-1 -- curl 10.244.0.24:8080/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2466      0 --:--:-- --:--:-- --:--:--  2666
23:43:22
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod-2 -- curl 10.244.0.23:8080/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2064      0 --:--:-- --:--:-- --:--:--  2666
23:43:11
```

- However, now it is also possible to communicate **by the same IP through the service**: `10.105.84.72` and `80` port

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod-2 -- curl 10.105.84.72/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2418      0 --:--:-- --:--:-- --:--:--  2666
23:47:19
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod-1 -- curl 10.105.84.72/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0    908      0 --:--:-- --:--:-- --:--:--   888
23:48:01
```

DNS

- Apart from that we can now refer to the service **by DNS name**

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod-1 -- curl http://current-time-server-service/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   1842      0 --:--:-- --:--:-- --:--:--  2000
00:55:58
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-pod-2 -- curl http://current-time-server-service/time -w '\n'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   1812      0 --:--:-- --:--:-- --:--:--  2000
00:59:09
```

- This is possible because of the DNS server `kybe-system/coredns` and `kybe-system/kube-dns` service

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl -n kube-system get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
coredns   1/1     1            1           23m
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl -n kube-system get svc
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   23m
```

- We see that our pods have predefined domains that are resolved by the `kube-dns` service

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec -it current-time-server-pod-1 -- cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local tcsbank.ru
options ndots:5
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec -it current-time-server-pod-2 -- cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local tcsbank.ru
options ndots:5
```

5

Task description

Delete any `Pod`, recreate it and check addresses again. Make sure that traffic is routed to the new `Pod` correctly

- I deleted the pod `current-time-server-pod-2` and create it again

```
kubectl delete -f pod-2.yaml
kubectl apply -f pod-2.yaml
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl delete -f pod-2.yaml
pod "current-time-server-pod-2" deleted
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n default
NAME                       READY   STATUS    RESTARTS      AGE
current-time-server-pod-1   1/1     Running   1 (13m ago)   20h
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f pod-2.yaml
pod/current-time-server-pod-2 created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n default
NAME                       READY   STATUS    RESTARTS      AGE
current-time-server-pod-1   1/1     Running   1 (15m ago)   20h
current-time-server-pod-2   1/1     Running   0             27s
```

- We can notice that this pod changed its IP address

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po current-time-server-pod-2 --output=custom-columns='IP:.status.podIP'
IP
10.244.0.13
```

- Then I made an HTTP request to my service 10 times

```
for i in {1..10}; do kubectl exec current-time-server-pod-1 -- curl
http://current-time-server-service/time; done
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
for i in {1..10}; do kubectl exec current-time-server-pod-1 -- curl http://current-time-server-service/time; done
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0    871      0 --:--:-- --:--:-- --:--:--   888
21:31:05  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2347      0 --:--:-- --:--:-- --:--:--  2666
21:31:05  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0    990      0 --:--:-- --:--:-- --:--:--  1000
21:31:05  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   1863      0 --:--:-- --:--:-- --:--:--  200021:31:05
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2643      0 --:--:-- --:--:-- --:--:--  400021:31:06
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2255      0 --:--:-- --:--:-- --:--:--  266621:31:06
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2391      0 --:--:-- --:--:-- --:--:--  266621:31:06
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2889      0 --:--:-- --:--:-- --:--:--  4000
21:31:06  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2642      0 --:--:-- --:--:-- --:--:--  400021:31:06
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     8  100     8    0     0   2955      0 --:--:-- --:--:-- --:--:--  400021:31:06
```

- We can see the this pods has corresponding logs

```
kubectl logs current-time-server-pod-2
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl logs current-time-server-pod-2

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.3)

2025-02-19T18:26:31.956Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt
 v1.0.0 using Java 17.0.11 with PID 1 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T18:26:31.957Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1
 default profile: "default"
2025-02-19T18:26:32.947Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080
 (http)
2025-02-19T18:26:32.951Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T18:26:32.951Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache
 Tomcat/10.1.13]
2025-02-19T18:26:32.982Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
 WebApplicationContext
2025-02-19T18:26:32.982Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization
  completed in 947 ms
2025-02-19T18:26:33.442Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path
 '/actuator'
2025-02-19T18:26:33.471Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
 with context path ''
2025-02-19T18:26:33.481Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in
 1.818 seconds (process running for 2.102)
2025-02-19T18:29:53.746Z  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet
 'dispatcherServlet'
2025-02-19T18:29:53.746Z  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-02-19T18:29:53.749Z  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 3 ms
2025-02-19T18:29:53.781Z  INFO 1 --- [nio-8080-exec-1] i.d.c.service.VisitsService              : Visits counter has been incremented to 1
2025-02-19T18:29:53.783Z  INFO 1 --- [nio-8080-exec-1] i.d.c.controller.TimeController          : GET /time
2025-02-19T18:29:55.217Z  INFO 1 --- [nio-8080-exec-2] i.d.c.service.VisitsService              : Visits counter has been incremented to 2
2025-02-19T18:29:55.217Z  INFO 1 --- [nio-8080-exec-2] i.d.c.controller.TimeController          : GET /time
2025-02-19T18:31:05.662Z  INFO 1 --- [nio-8080-exec-3] i.d.c.service.VisitsService              : Visits counter has been incremented to 3
2025-02-19T18:31:05.662Z  INFO 1 --- [nio-8080-exec-3] i.d.c.controller.TimeController          : GET /time
2025-02-19T18:31:05.762Z  INFO 1 --- [nio-8080-exec-4] i.d.c.service.VisitsService              : Visits counter has been incremented to 4
2025-02-19T18:31:05.762Z  INFO 1 --- [nio-8080-exec-4] i.d.c.controller.TimeController          : GET /time
2025-02-19T18:31:06.091Z  INFO 1 --- [nio-8080-exec-5] i.d.c.service.VisitsService              : Visits counter has been incremented to 5
2025-02-19T18:31:06.091Z  INFO 1 --- [nio-8080-exec-5] i.d.c.controller.TimeController          : GET /time
2025-02-19T18:31:06.288Z  INFO 1 --- [nio-8080-exec-6] i.d.c.service.VisitsService              : Visits counter has been incremented to 6

~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

Task description

Learn about `Loadbalancer` and `NodePort` .

ClusterIP

- **Description**: This is the default Service type. Such a Service exposes the service on a cluster-internal IP. It makes the Service accessible only from within the cluster.
- **Use Case**: Necessary for internal communication between Pods within the cluster. For example, microservices that need to communicate with each other can use a `ClusterIP` Service.

NodePort

- **Description**: A `NodePort` service exposes the `Service` on each Node's IP at a static port. A request to the Node's IP at the NodePort is forwarded to the Service.
- **Use Case**: Useful for development and testing, or for when you want to expose the Service externally without a load balancer. It's a straightforward method to expose a service available via any of the cluster nodes.

LoadBalancer

- **Description**: This type of Service provisions an external load balancer (if the cloud provider supports this feature), which directs traffic to the `ClusterIP` of the Service. The load balancer is assigned a public IP address.
- **Use Case**: Suitable for production applications where you want to expose a Service to the internet or external clients. This is commonly used with cloud providers like AWS, GCP, or Azure.

ExternalName

- **Description**: An `ExternalName` Service maps the Service to a DNS name. It does not proxy any traffic but serves as a way to reference an external service based on a name.
- **Use Case**: Useful for integrating external services into the Kubernetes cluster. Instead of using an IP or a fixed address, you can reference the external service by its DNS name.

7

Task description

Deploy an external `Service` to access your application from outside, e.g., from your local host

- I decided to use `LoadBalancer` service type
- Here is the updated `service.yaml` file content

```yaml
apiVersion: v1
kind: Service
metadata:
  name: current-time-server-service
spec:
  type: LoadBalancer
  selector:
    app: current-time-server
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  externalIPs:
    - 192.168.49.2
```

- Now let's recreate our service

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f service.yaml
service/current-time-server-service configured
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po current-time-server-service -o=wide
Error from server (NotFound): pods "current-time-server-service" not found                                    Exit code 1
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get svc current-time-server-service -o=wide
NAME                          TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE   SELECTOR
current-time-server-service   LoadBalancer   10.111.137.238   192.168.49.2   80:30467/TCP     21h   app=current-time-server
```

- First, let's try to connect to the minikube node that is running inside a Docker container

```
docker exec -it minikube curl http://localhost:30467/time -w '\n'
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
docker exec -it minikube curl http://localhost:30467/time -w '\n'
22:18:34
```

- After, let's I tunnel node's connection to my host machine

```
minikube service current-time-server-service
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
minikube service current-time-server-service
|-----------|-------------------------------|-------------|---------------------------|
| NAMESPACE |             NAME              | TARGET PORT |            URL            |
|-----------|-------------------------------|-------------|---------------------------|
| default   | current-time-server-service   |          80 | http://192.168.49.2:30467 |
|-----------|-------------------------------|-------------|---------------------------|
🏃  Starting tunnel for service current-time-server-service.
|-----------|-------------------------------|-------------|---------------------------|
| NAMESPACE |             NAME              | TARGET PORT |            URL            |
|-----------|-------------------------------|-------------|---------------------------|
| default   | current-time-server-service   |             | http://127.0.0.1:52413    |
|-----------|-------------------------------|-------------|---------------------------|
🎉  Opening service default/current-time-server-service in default browser...
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

- Now we can access it locally

- This is funny to understand that there are so many ports in such a chain:
`Pod's 8080 --> Node's Internal 80 --> Node's External 30467 --> Host's 52413`

8

> **Task description**
>
> Put the results into report

✅

Task 5 - k8s Deployment

> `Deployment` is Kubernetes manifest to manage Pods automatically by Controller. It helps to release, scale and upgrade Pods.

1

> **Task description**
>
> Figure out the necessary `Deployment` spec fields.

- **`apiVersion`** : This field specifies the API version of Kubernetes used.
- **`kind`** : This indicates the type of resource for creating. For deployments it is `Deployment`
- **`metadata`** : This section contains metadata about the Deployment, such as its name, namespace, labels, and annotations. The name of the Deployment is particularly important for managing and identifying it.
- **`spec`** : The specification for the Deployment, which includes the following necessary fields:
  - **`replicas`** : Specifies the desired number of Pod replicas. This is how many instances of your application you want to run.
  - **`selector`** : A label selector that identifies the Pods that belong to this Deployment. It must match the labels defined in the Pod template.
- **`template`** : A Pod template used to create the Pods. It contains the specification of the Pods, including metadata and spec for the containers.

2

Task description

Make sure that you wiped previous `Pod` manifests. Write a `Deployment` spec for your pod(s) and deploy the application.

- Firstly, I deleted pods

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl delete -f pod-1.yaml
pod "current-time-server-pod-1" deleted
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl delete -f pod-2.yaml
pod "current-time-server-pod-2" deleted
```

- I created `deployment.yaml` file with 2 pod replicas remaining pods' configuration the same

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
ls | grep deployment
deployment.yaml
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
          ports:
            - containerPort: 8080
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

- Below is the content of this file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
```

```
        ports:
          - containerPort: 8080
```

- Here is the result of deploying the deployment

```
kubectl apply -f deployment.yaml
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml
deployment.apps/current-time-server-deployment created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get deploy -o=wide
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS           IMAGES                                SELECTOR
current-time-server-deployment   2/2     2            2           66s   current-time-server   iskanred/current-time-server:1.0.0
 app=current-time-server
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -o=wide
NAME                                           READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS
 GATES
current-time-server-deployment-555c44f85d-k5btb   1/1     Running   0          91s   10.244.0.19   minikube   <none>           <none>
current-time-server-deployment-555c44f85d-p57g4   1/1     Running   0          91s   10.244.0.20   minikube   <none>           <none>
```

3

Task description

With `kubectl`, get the `Deployments` and describe them.

- I called `get` and `describe` commands for the deployment

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get deploy -o=wide
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS           IMAGES                                SELECTOR
current-time-server-deployment   2/2     2            2           3m9s   current-time-server   iskanred/current-time-server:1.0.0
 app=current-time-server
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe deploy
Name:                   current-time-server-deployment
Namespace:              default
CreationTimestamp:      Wed, 19 Feb 2025 22:52:49 +0300
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=current-time-server
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=current-time-server
  Containers:
   current-time-server:
    Image:         iskanred/current-time-server:1.0.0
    Port:          8080/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:   current-time-server-deployment-555c44f85d (2/2 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m19s  deployment-controller  Scaled up replica set current-time-server-deployment-555c44f85d from 0 to 2
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

- We here
  - `StrategyType` for upgrading the deployment with its configuration
  - `NewReplicaSet` for controlling pods' replicas creation and management
  - and other useful information

4

Task description

Update your `Deployment` manifest to scale your application to three replicas.

- The most convenient way to scale the number of replicas of a running deployment is to use `kubectl scale` command.
- It may be useful for temporary increasing the number of pods in conditions of high traffic load.
- Here is the scaling to 3 replicas:

```
kubectl scale deployment current-time-server-deployment --replicas=3
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl scale deployment current-time-server-deployment --replicas=3
deployment.apps/current-time-server-deployment scaled
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get deploy
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
current-time-server-deployment   3/3     3            3           11m
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubecvtl get po -o=wide
zsh: command not found: kubecvtl                                                Exit code 127
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -o=wide
NAME                                          READY   STATUS    RESTARTS   AGE   IP            NODE       NOMINATED NODE   READINESS
 GATES
current-time-server-deployment-555c44f85d-8dhv8   1/1     Running   0          25s   10.244.0.21   minikube   <none>           <none>
current-time-server-deployment-555c44f85d-k5btb   1/1     Running   0          11m   10.244.0.19   minikube   <none>           <none>
current-time-server-deployment-555c44f85d-p57g4   1/1     Running   0          11m   10.244.0.20   minikube   <none>           <none>
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

5

Task description

Access pod shell and logs using `Deployment` labels.

- We access all deployment's pods logs

```
kubectl logs -l app=current-time-server
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl logs -l app=current-time-server
2025-02-19T20:04:18.324Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt v1.0.0 using Java 17.0.11 with PID 1
 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T20:04:18.325Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1 default profile: "default"
2025-02-19T20:04:19.338Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2025-02-19T20:04:19.343Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T20:04:19.343Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.13]
2025-02-19T20:04:19.417Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-02-19T20:04:19.417Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1010 ms
2025-02-19T20:04:19.894Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path '/actuator'
2025-02-19T20:04:19.931Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2025-02-19T20:04:19.942Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in 1.928 seconds (process running for 2.202)
2025-02-19T19:52:51.425Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt v1.0.0 using Java 17.0.11 with PID 1
 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T19:52:51.426Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1 default profile: "default"
2025-02-19T19:52:53.694Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2025-02-19T19:52:53.702Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T19:52:53.703Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.13]
2025-02-19T19:52:53.791Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-02-19T19:52:53.792Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2166 ms
2025-02-19T19:52:54.714Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path '/actuator'
2025-02-19T19:52:54.833Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2025-02-19T19:52:54.901Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in 3.989 seconds (process running for 4.758)
2025-02-19T19:52:51.307Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt v1.0.0 using Java 17.0.11 with PID 1
 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T19:52:51.309Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1 default profile: "default"
2025-02-19T19:52:53.611Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2025-02-19T19:52:53.617Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T19:52:53.617Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.13]
2025-02-19T19:52:53.711Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-02-19T19:52:53.712Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2297 ms
2025-02-19T19:52:54.900Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path '/actuator'
2025-02-19T19:52:55.000Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2025-02-19T19:52:55.013Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in 4.378 seconds (process running for 4.869)
```

- Besides, for accessing logs of the specific we should list all the pods of the deployment using the label

```
kubectl get pods -l app=current-time-server
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
    kubectl get pods -l app=current-time-server
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-555c44f85d-8dhv8   1/1     Running   0          14m
current-time-server-deployment-555c44f85d-k5btb   1/1     Running   0          25m
current-time-server-deployment-555c44f85d-p57g4   1/1     Running   0          25m
```

- And then get logs of a desired pod

```
kubectl logs current-time-server-deployment-555c44f85d-8dhv8
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl logs current-time-server-deployment-555c44f85d-8dhv8

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.3)

2025-02-19T20:04:18.324Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt
 v1.0.0 using Java 17.0.11 with PID 1 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T20:04:18.325Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1
 default profile: "default"
2025-02-19T20:04:19.338Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080
 (http)
2025-02-19T20:04:19.343Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T20:04:19.343Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache
 Tomcat/10.1.13]
2025-02-19T20:04:19.417Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
 WebApplicationContext
2025-02-19T20:04:19.417Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization
  completed in 1010 ms
2025-02-19T20:04:19.894Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path
 '/actuator'
2025-02-19T20:04:19.931Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
 with context path ''
2025-02-19T20:04:19.942Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in
 1.928 seconds (process running for 2.202)

~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

- In the same way we can access pod's shell

```
kubectl exec -it current-time-server-deployment-555c44f85d-8dhv8 -- bash
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get pods -l app=current-time-server
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-555c44f85d-8dhv8   1/1     Running   0          20m
current-time-server-deployment-555c44f85d-k5btb   1/1     Running   0          31m
current-time-server-deployment-555c44f85d-p57g4   1/1     Running   0          31m
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec -it current-time-server-deployment-555c44f85d-8dhv8 -- bash
user@current-time-server-deployment-555c44f85d-8dhv8:/current-time-server$ exit
exit
```

6

Task description

Make any application configuration change in your `Deployment` yaml and try to update the application. Monitor what are happened with pods (`--watch`).

- For application update I added a `command` for pods' containers

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
          command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
          ports:
            - containerPort: 8080
```

- I applied the new deployment configuration and started watching for pods

```
kubectl apply -f deployment.yaml && \
kubectl get pods -l app=current-time-server --watch
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep command
          command: ['sh', '-c', 'echo "Hello, Kubernetes!" && sleep 3600']
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml && kubectl get pods -l app=current-time-server --watch
deployment.apps/current-time-server-deployment configured
NAME                                          READY   STATUS              RESTARTS   AGE
current-time-server-deployment-555c44f85d-b9rzx   1/1     Running             0          20s
current-time-server-deployment-555c44f85d-mrfmk   1/1     Running             0          21s
current-time-server-deployment-8595ff49b6-4n5pf   1/1     Terminating         0          85s
current-time-server-deployment-8595ff49b6-jmszj   1/1     Terminating         0          84s
current-time-server-deployment-8595ff49b6-sncm2   0/1     ContainerCreating   0          0s
current-time-server-deployment-8595ff49b6-sncm2   1/1     Running             0          2s
current-time-server-deployment-555c44f85d-b9rzx   1/1     Terminating         0          22s
current-time-server-deployment-8595ff49b6-7zs7d   0/1     Pending             0          0s
current-time-server-deployment-8595ff49b6-7zs7d   0/1     Pending             0          0s
current-time-server-deployment-8595ff49b6-7zs7d   0/1     ContainerCreating   0          0s
current-time-server-deployment-555c44f85d-b9rzx   0/1     Error               0          22s
current-time-server-deployment-555c44f85d-b9rzx   0/1     Error               0          23s
current-time-server-deployment-555c44f85d-b9rzx   0/1     Error               0          23s
current-time-server-deployment-8595ff49b6-7zs7d   1/1     Running             0          1s
current-time-server-deployment-555c44f85d-mrfmk   1/1     Terminating         0          24s
current-time-server-deployment-555c44f85d-mrfmk   0/1     Error               0          24s
current-time-server-deployment-555c44f85d-mrfmk   0/1     Error               0          25s
current-time-server-deployment-555c44f85d-mrfmk   0/1     Error               0          25s
current-time-server-deployment-8595ff49b6-jmszj   0/1     Error               0          95s
current-time-server-deployment-8595ff49b6-jmszj   0/1     Error               0          95s
current-time-server-deployment-8595ff49b6-jmszj   0/1     Error               0          95s
current-time-server-deployment-8595ff49b6-4n5pf   0/1     Error               0          97s
current-time-server-deployment-8595ff49b6-4n5pf   0/1     Error               0          97s
current-time-server-deployment-8595ff49b6-4n5pf   0/1     Error               0          97s
```

- We can check if the application was updated by accessing its logs

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl logs -l app=current-time-server
Hello, Kubernetes!
Hello, Kubernetes!
```

7

Task description

Rollback to previous application version using `Deployment` .

- At first, let's check the history of the deployment

```
kubectl rollout history deployment/current-time-server-deployment
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl rollout history deployment/current-time-server-deployment
deployment.apps/current-time-server-deployment
REVISION  CHANGE-CAUSE
5         <none>
6         <none>
```

- We can notice that the `REVISION` is so hight because I debugged the deployment upgrading process in the previous task
- Now let's rollout the deployment to the previous revision

```
kubectl rollout undo deployment/current-time-server-deployment && \
kubectl get pods -l app=current-time-server --watch
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl rollout undo deployment/current-time-server-deployment && kubectl get pods -l app=current-time-server --watch
deployment.apps/current-time-server-deployment rolled back
NAME                                           READY   STATUS              RESTARTS   AGE
current-time-server-deployment-555c44f85d-ltfkf   0/1     ContainerCreating   0          0s
current-time-server-deployment-8595ff49b6-7zs7d   1/1     Running             0          11m
current-time-server-deployment-8595ff49b6-sncm2   1/1     Running             0          11m
current-time-server-deployment-555c44f85d-ltfkf   1/1     Running             0          1s
current-time-server-deployment-8595ff49b6-7zs7d   1/1     Terminating         0          11m
current-time-server-deployment-555c44f85d-z9c7v   0/1     Pending             0          0s
current-time-server-deployment-555c44f85d-z9c7v   0/1     Pending             0          0s
current-time-server-deployment-555c44f85d-z9c7v   0/1     ContainerCreating   0          0s
current-time-server-deployment-555c44f85d-z9c7v   1/1     Running             0          1s
current-time-server-deployment-8595ff49b6-sncm2   1/1     Terminating         0          11m
^C
```

Exit code 1

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl rollout status deployment/current-time-server-deployment
deployment "current-time-server-deployment" successfully rolled out
```

- We see the status of our rollout is successful
- Let's check pods' logs

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl logs -l app=current-time-server
2025-02-19T21:01:28.016Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt
 v1.0.0 using Java 17.0.11 with PID 1 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T21:01:28.017Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1
 default profile: "default"
2025-02-19T21:01:30.324Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080
 (http)
2025-02-19T21:01:30.329Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T21:01:30.329Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache
 Tomcat/10.1.13]
2025-02-19T21:01:30.432Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
 WebApplicationContext
2025-02-19T21:01:30.434Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization
  completed in 2321 ms
2025-02-19T21:01:31.521Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path
 '/actuator'
2025-02-19T21:01:31.627Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
 with context path ''
2025-02-19T21:01:31.691Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in
 4.183 seconds (process running for 4.51)
2025-02-19T21:01:29.205Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Starting CurrentTimeServerApplicationKt
 v1.0.0 using Java 17.0.11 with PID 1 (/current-time-server/current-time-server-1.0.0.jar started by user in /current-time-server)
2025-02-19T21:01:29.210Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : No active profile set, falling back to 1
 default profile: "default"
2025-02-19T21:01:31.592Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080
 (http)
2025-02-19T21:01:31.599Z  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-02-19T21:01:31.599Z  INFO 1 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache
 Tomcat/10.1.13]
2025-02-19T21:01:31.692Z  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
 WebApplicationContext
2025-02-19T21:01:31.693Z  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization
  completed in 2362 ms
2025-02-19T21:01:32.299Z  INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 4 endpoint(s) beneath base path
 '/actuator'
2025-02-19T21:01:32.345Z  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
 with context path ''
2025-02-19T21:01:32.398Z  INFO 1 --- [           main] i.d.c.CurrentTimeServerApplicationKt     : Started CurrentTimeServerApplicationKt in
 3.898 seconds (process running for 4.625)
```

- We see that the rollout was successful indeed!

- Finally, we can notice that our `ReplicaSet`s were updated by deployment controller

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe deploy
Name:                   current-time-server-deployment
Namespace:              default
CreationTimestamp:      Wed, 19 Feb 2025 22:52:49 +0300
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 7
Selector:               app=current-time-server
Replicas:               2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=current-time-server
  Containers:
   current-time-server:
    Image:         iskanred/current-time-server:1.0.0
    Port:          8080/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:  current-time-server-deployment-8595ff49b6 (0/0 replicas created)
NewReplicaSet:   current-time-server-deployment-555c44f85d (2/2 replicas created)
Events:
  Type    Reason             Age                From                   Message
  ----    ------             ----               ----                   -------
  Normal  ScalingReplicaSet  34m (x2 over 39m)  deployment-controller  Scaled down replica set current-time-server-deployment-555c44f85d from 3 to 2
  Normal  ScalingReplicaSet  33m (x2 over 35m)  deployment-controller  (combined from similar events): Scaled up replica set
   current-time-server-deployment-555c44f85d from 0 to 1
  Normal  ScalingReplicaSet  33m (x3 over 39m)  deployment-controller  Scaled up replica set current-time-server-deployment-8595ff49b6 from 0 to 1
  Normal  ScalingReplicaSet  33m (x3 over 39m)  deployment-controller  Scaled down replica set current-time-server-deployment-555c44f85d from 2 to 1
  Normal  ScalingReplicaSet  33m (x3 over 39m)  deployment-controller  Scaled up replica set current-time-server-deployment-8595ff49b6 from 1 to 2
  Normal  ScalingReplicaSet  33m (x3 over 39m)  deployment-controller  Scaled down replica set current-time-server-deployment-555c44f85d from 1 to 0
  Normal  ScalingReplicaSet  21m (x2 over 36m)  deployment-controller  Scaled up replica set current-time-server-deployment-555c44f85d from 0 to 1
  Normal  ScalingReplicaSet  21m (x3 over 36m)  deployment-controller  Scaled down replica set current-time-server-deployment-8595ff49b6 from 2 to 1
  Normal  ScalingReplicaSet  21m (x3 over 36m)  deployment-controller  Scaled up replica set current-time-server-deployment-555c44f85d from 1 to 2
  Normal  ScalingReplicaSet  21m (x3 over 36m)  deployment-controller  Scaled down replica set current-time-server-deployment-8595ff49b6 from 1 to 0
```

8

Task description

Set up `requests` and `limits` for CPU and Memory for your application pods. Provide a PoC that it works properly. Explain and show what is happened when app reaches the CPU usage limit? And memory limit?

Exploring consumption

- Below is the updated `deployment.yaml` file. Notice that I removed `container.command` from the manifest

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
```

```yaml
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
          ports:
            - containerPort: 8080
          resources:
            requests:
              memory: 256Mi
              cpu: 0.5
            limits:
              memory: 512Mi
              cpu: 1
```

- I applied this manifest

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
          ports:
            - containerPort: 8080
          resources:
            requests:
              memory: 256Mi
              cpu: 0.5
            limits:
              memory: 512Mi
              cpu: 1
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml
deployment.apps/current-time-server-deployment configured
```

- We can see that these limits/requests were applied to our pods

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -o=custom-columns='Resources:.spec.containers[0].resources'
Resources
map[limits:map[cpu:1 memory:512Mi] requests:map[cpu:500m memory:256Mi]]
map[limits:map[cpu:1 memory:512Mi] requests:map[cpu:500m memory:256Mi]]
```

- What is more we can notice that their QoS is indeed `Burstable` because they do have memory and CPU limits/requests but they do not equal

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -o=custom-columns='Name:.metadata.name,QoS:.status.qosClass'
Name                                          QoS
current-time-server-deployment-bd9664c9c-kdhfr   Burstable
current-time-server-deployment-bd9664c9c-xtxlk   Burstable
```

- Now let's check pods' metrics using minikube's `metrics-server` addon

```
minikube addons enable metrics-server
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
minikube addons list | grep metrics
| metrics-server             | minikube | enabled ✅   | Kubernetes                  |
```

- Now we can monitor our pods' performance Uusing `kubectl top` command

```
kubectl top pods -l app=current-time-server
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl top pods -l app=current-time-server
NAME                                              CPU(cores)   MEMORY(bytes)
current-time-server-deployment-bd9664c9c-kdhfr   2m           125Mi
current-time-server-deployment-bd9664c9c-xtxlk   2m           122Mi
```

- We see that my pods consume only `2m` which is 0.002 of CPU core cycle time and about `125Mi` of memory

Proof of concept

- We can infer that current thresholds for limits for both memory and CPU are too big since the application does no use so much
- Let's determine some other boundaries and test if they work actually!

CPU limits

> My hypothesis is that **with smaller limit of CPU** time my **pods start slower**.

- This might happen because pods have lack of performance at the startup while application is configured: Spring beans should be created, JIT-compilation should "warm up" code regions, and etc.

Preparation

- To test startup duration I added `startupProbe` which access application's healthcheck endpoint

```yaml
apiVersion: apps/v1
kind: Deployment
<...>
      containers:
        startupProbe:
          httpGet:
            path: /actuator/health
            port: 8080
          periodSeconds: 1
          failureThreshold: 10000
```

- We see that that the prober is configured to check if a pod's container is started `10000` times every `1` second
- To measure how long the pod is starting I implemented a simple `measure.sh` script

```
#!/bin/zsh

PODS=$(kubectl get po -l app=current-time-server)

# Deleting pods if exist
if [[ -n $PODS ]]; then
```

```
  echo "There are some pods with the same labels running"
  kubectl delete -f deployment.yaml

  while [[ $PODS != "" ]]
  do
    PODS=$(kubectl get po -l app=current-time-server)
    echo "Waiting for the pods to be deleted..."
    sleep 1
  done

  echo "The pods are deleted"
fi

kubectl apply -f deployment.yaml
echo "Waiting for pods to start..."

seconds=0
while [[ $(kubectl get po -l app=current-time-server -o
jsonpath='{.items[0].status.containerStatuses[0].started}') == "false" ]]
do
  (( seconds+=1 ))
  sleep 1;
done

echo "Elapsed time: $seconds seconds"
```

- It removes already existed pods with desired labels and starts a deployment. After the deployment is started the script increments the number of seconds until the status is not `false`. Finally, the script prints the number of elapsed time for pod to start.
- What is more, to make the process of testing faster I decreased our deployment to `1` replica only

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 1
<...>
```

- In addition, I configured `requests` = `limits` to simplify the process of testing `limits` making pods to have `Guaranteed` QoS.
- Finally, I gave enough memory for my pods to be sure the memory is not a bottleneck when testing CPU.

- The final `deployment.yaml` is the following:

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
          ports:
            - containerPort: 8080
          resources:
            requests:
              memory: 1Gi
              cpu: 1
            limits:
              memory: 1Gi
              cpu: 1
          startupProbe:
            httpGet:
              path: /actuator/health
              port: 8080
            periodSeconds: 1
            failureThreshold: 10000
```

Test

- Let's check if our CPU `limits` bounds actually work
- Firstly, let's start with `limits.cpu = 1`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep -B 2 cpu
            requests:
              memory: 1Gi
              cpu: 1
            limits:
              memory: 1Gi
              cpu: 1
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
./measure.sh
There are some pods with the same labels running
deployment.apps "current-time-server-deployment" deleted
Waiting for the pods to be deleted...
No resources found in default namespace.
Waiting for the pods to be deleted...
The pods are deleted
deployment.apps/current-time-server-deployment created
Waiting for pods to start...
Elapsed time: 5 seconds
```

- Now, let's change the value to `limits.cpu = 0.5`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep -B 2 cpu
            requests:
              memory: 1Gi
              cpu: 0.5
            limits:
              memory: 1Gi
              cpu: 0.5
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
./measure.sh
There are some pods with the same labels running
deployment.apps "current-time-server-deployment" deleted
Waiting for the pods to be deleted...
No resources found in default namespace.
Waiting for the pods to be deleted...
The pods are deleted
deployment.apps/current-time-server-deployment created
Waiting for pods to start...
Elapsed time: 9 seconds
```

- Let' continue with `limits.cpu = 0.25`

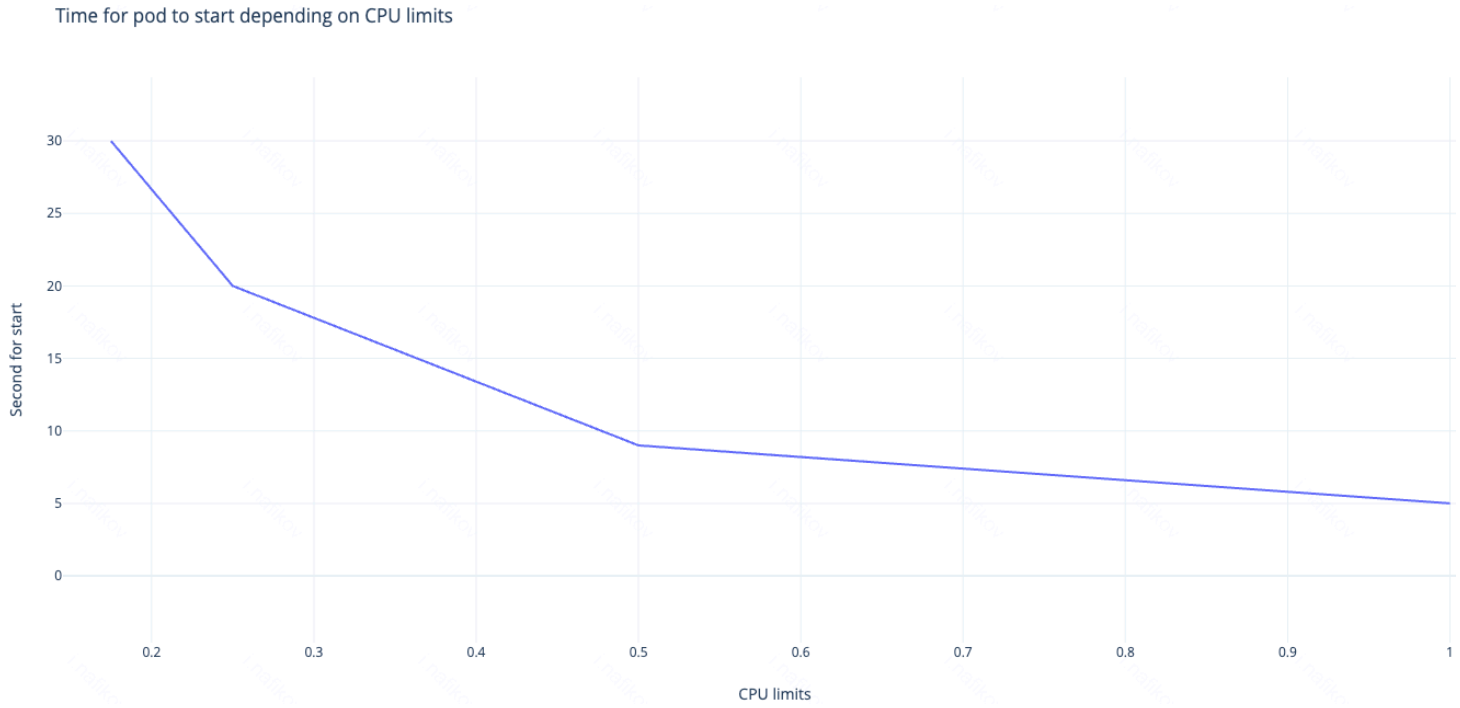
```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
./measure.sh
There are some pods with the same labels running
deployment.apps "current-time-server-deployment" deleted
Waiting for the pods to be deleted...
No resources found in default namespace.
Waiting for the pods to be deleted...
The pods are deleted
deployment.apps/current-time-server-deployment created
Waiting for pods to start...
Elapsed time: 20 seconds
```

- Finally, let' check the result with `limits.cpu = 0.175`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep -B 2 cpu
            requests:
              memory: 1Gi
              cpu: 0.175
            limits:
              memory: 1Gi
              cpu: 0.175
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
./measure.sh
There are some pods with the same labels running
deployment.apps "current-time-server-deployment" deleted
Waiting for the pods to be deleted...
No resources found in default namespace.
Waiting for the pods to be deleted...
The pods are deleted
deployment.apps/current-time-server-deployment created
Waiting for pods to start...
Elapsed time: 30 seconds
```

- Okay, let's plot our results



- We see that **`limits` mechanism actually works**!**. The smaller the `limits.cpu` the higher the amount of time for pod to start

Memory limits

> My hypothesis is that **with smaller limit of memory pods may be killed due to OOM**

- This might happen if an application has not enough memory to start / work.

Preparation

- To check if `limits.memory` actually works the strategy is similar to check the `limits.cpu` on a startup.

- However, here we have a **binary** evaluation: either the pod will start successfully or not.

- So here I can just set `limit.memory` value so small that pods will be killed almost immediately.
- To make startup faster I set `limits.cpu = 1`.
- What is more, I kept `requests = limits`.

Test

- Let's check the value `limits.memory = 100Mi`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl delete -f deployment.yaml
deployment.apps "current-time-server-deployment" deleted
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep -B 2 cpu
            requests:
              memory: 100Mi
              cpu: 1
            limits:
              memory: 100Mi
              cpu: 1
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml
deployment.apps/current-time-server-deployment created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get pod
NAME                                          READY   STATUS             RESTARTS      AGE
current-time-server-deployment-67d6658b7d-9mjbt   0/1     CrashLoopBackOff   4 (58s ago)   2m40s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -o jsonpath-as-json='{.items[0].status.containerStatuses[0].lastState}'
[
    {
        "terminated": {
            "containerID": "docker://b07e871b8f74bc1f97dcf9fcaebe60737e505ba54414450249262decf178da02",
            "exitCode": 137,
            "finishedAt": "2025-02-22T15:10:49Z",
            "reason": "OOMKilled",
            "startedAt": "2025-02-22T15:10:46Z"
        }
    }
]
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
```

- We see that our **`limit.memory` mechanism works too**! The container couldn't be started and tries to restart infinitely because `kubelet` kills it since it consumes more memory than the limit set.

9

> **Task description**
>
> Put the results into report

✅

Task 6 - k8s Secrets

> `Secret` manifest is a quite similar to `configMap`. However, we use `Secret` to work with confidential application data. Kubernetes encode secrets in Base64 format.

1

Task description

Figure out the necessary Secret spec fields.

- **`apiVersion`**: This field specifies the API version of Kubernetes used.
- **`kind`**: This indicates the type of resource for creating. For secrets it is `Secret`.
- **`metadata`**: This section contains metadata about the `Secret`, such as its name, namespace, labels, and annotations.
- **`type`**: Defines the type of Secret. Common types include:
  - `Opaque`: The default type for arbitrary data.
  - `kubernetes.io/dockerconfigjson`: For Docker registry credentials.
  - `kubernetes.io/basic-auth`: For basic authentication credentials.
  - `kubernetes.io/ssh-auth`: For SSH authentication key.
- **`data`**: The data fields where each entry is base64-encoded.
- **`stringData`**: This is similar to `data`, but it allows you to specify the values in plaintext. Kubernetes will encode them in base64 automatically when you create the Secret. This can be more convenient for users.
- It's worth to notice that secrets do not have **`spec`** field in its manifest.

2

Task description

Create and apply a new `Secret` manifest. For example, it could be login and password to login to your app or something else

- I created a secret from the following `secret.yaml` manifest:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: current-time-server-secret
  labels:
    app: current-time-server
type: Opaque
stringData:
  secret: my-super-secret
```

- And applied it

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: current-time-server-secret
  labels:
    app: current-time-server
type: Opaque
stringData:
  secret: my-super-secret
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f secret.yaml
secret/current-time-server-secret created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get secret
NAME                         TYPE     DATA   AGE
current-time-server-secret   Opaque   1      5s
```

3

Task description

With `kubectl`, get and describe your secret(s).

- `get`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get secret -o wide
NAME                         TYPE     DATA   AGE
current-time-server-secret   Opaque   1      105s
```

```
kubectl get secret current-time-server-secret -o json
{
    "apiVersion": "v1",
    "data": {
        "secret": "bXktc3VwZXItc2VjcmV0"
    },
    "kind": "Secret",
    "metadata": {
        "annotations": {
            "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"v1\",\"kind\":\"Secret\",
             \"metadata\":{\"annotations\":{},\"labels\":{\"app\":\"current-time-server\"},\"name\":\"current-time-server-secret\",
             \"namespace\":\"default\"},\"stringData\":{\"secret\":\"my-super-secret\"},\"type\":\"Opaque\"}\n"
        },
        "creationTimestamp": "2025-02-22T16:10:53Z",
        "labels": {
            "app": "current-time-server"
        },
        "name": "current-time-server-secret",
        "namespace": "default",
        "resourceVersion": "99157",
        "uid": "f32293ca-82be-4e45-b80f-f91aeb8ba804"
    },
    "type": "Opaque"
}
```

- `describe`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe secret
Name:         current-time-server-secret
Namespace:    default
Labels:       app=current-time-server
Annotations:  <none>

Type:  Opaque

Data
====
secret:  15 bytes
```

4

Task description

Decode your secret(s).

- I decoded the value of my secret using `base64`

```
kubectl get secret current-time-server-secret -o jsonpath='{.data.secret}' |
base64 -d
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get secret current-time-server-secret -o jsonpath='{.data.secret}' | base64 -d && echo '\n'
my-super-secret
```

5

Task description

Update your `Deployment` to reference to your secret as environment variable.

- I update the `deployment.yaml` as following:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: current-time-server-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: current-time-server
  template:
    metadata:
      labels:
        app: current-time-server
    spec:
      containers:
        - name: current-time-server
          image: iskanred/current-time-server:1.0.0
          ports:
            - containerPort: 8080
          resources:
            requests:
              memory: 512Mi
              cpu: 1
            limits:
```

```
            memory: 512Mi
            cpu: 1
          startupProbe:
            httpGet:
              path: /actuator/health
              port: 8080
            periodSeconds: 1
            failureThreshold: 10000
          env:
            - name: SECRET
              valueFrom:
                secretKeyRef:
                  name: current-time-server-secret
                  key: secret
```

- Then I applied it

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep -A 5 env
          env:
            - name: SECRET
              valueFrom:
                secretKeyRef:
                  name: current-time-server-secret
                  key: secret
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml
deployment.apps/current-time-server-deployment created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -l app=current-time-server
NAME                                              READY   STATUS    RESTARTS   AGE
current-time-server-deployment-5b469bc57-7pmvm    1/1     Running   0          18s
current-time-server-deployment-5b469bc57-hs9k9    1/1     Running   0          18s
```

6

🖊 Task description

Make sure that you are able to see your secret inside pod.

- To check if I can see my secret I tunneled the service to my local host using `minikube`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get svc current-time-server-service
NAME                          TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
current-time-server-service   LoadBalancer   10.98.173.1   192.168.49.2   80:30954/TCP   46h
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
minikube service current-time-server-service
|-----------|-----------------------------|-------------|---------------------------|
| NAMESPACE |            NAME             | TARGET PORT |            URL            |
|-----------|-----------------------------|-------------|---------------------------|
| default   | current-time-server-service |          80 | http://192.168.49.2:30954 |
|-----------|-----------------------------|-------------|---------------------------|
🏃  Starting tunnel for service current-time-server-service.
|-----------|-----------------------------|-------------|------------------------|
| NAMESPACE |            NAME             | TARGET PORT |          URL           |
|-----------|-----------------------------|-------------|------------------------|
| default   | current-time-server-service |             | http://127.0.0.1:53950 |
|-----------|-----------------------------|-------------|------------------------|
🎉  Opening service default/current-time-server-service in default browser...
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

- Finally, I could easily get the **MD5**-hashed value of my secret by accessing the `/secret` endpoint of my application

```
~/Study/iu-devsecops-course/lab-04/current-time-server git:[main]
curl http://127.0.0.1:53950/secret -w '\n'
MD5 hash of secret is: ad44f1be904b9375641b666bd4bbc531
```

- The response is

```
MD5 hash of secret is: ad44f1be904b9375641b666bd4bbc531
```

- Let's compare this value with the hash value of my secret:

```
~/Study/iu-devsecops-course/lab-04/current-time-server git:[main]
curl http://127.0.0.1:53950/secret | awk '{print $6}'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    55  100    55    0     0   8587      0 --:--:-- --:--:-- --:--:--  9166
ad44f1be904b9375641b666bd4bbc531
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server git:[main]
echo -n "my-super-secret" | md5
ad44f1be904b9375641b666bd4bbc531
```

- We see that they are actually equal

7

> **Task description**
>
> Put the results into report

✅

Task 7 - k8s configMap

> `ConfigMap` is Kubernetes manifest to store application configuration setting in two ways: key-value pairs as environment variables and text (usually JSON) data as dedicated file into container filesystem. We usually use configMap to store non sensitive data as plain text

1

> **Task description**
>
> Figure out the necessary `configMap` spec fields.

- **`apiVersion`** : This field specifies the API version of Kubernetes used.
- **`kind`** : This indicates the type of resource for creating. For config maps it is `ConfigMap` .
- **`metadata`** : This section contains metadata about the `ConfigMap` , such as its name, namespace, labels, and annotations.

- `data`: A map of key-value pairs, where the keys are strings and the values can be strings or binary data encoded in base64. This field holds the configuration information.
- `binaryData`: (optional) Similar to `data`, but here values must be base64 encoded. It's used for binary data (e.g., a configuration file stored in binary format).

2

Task description

Modify your `Deployment` manifest to set up some app configuration via environment variables

- I decided to use the same value for obtaining a secret but now from a config map, not a secret
- I reused the same `deployment.yaml` but edited the secret source to config map

```
<...>
    env:
      - name: SECRET
        valueFrom:
          configMapKeyRef:
            name: current-time-server-config
            key: secret
```

3

Task description

Create a new `configMap` manifest. In data spec, put some app configuration as key-value pair (it could be the same as in previous exercise). In the `Deployment.Pod` spec add the connection to key-value pair from `configMap` yaml file

- Below is my `configmap.yaml`

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: current-time-server-config
  labels:
    app: current-time-server
data:
  secret: my-config-secret
```

- I applied it

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f configmap.yaml
configmap/current-time-server-config created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get configmap -l app=current-time-server
NAME                         DATA   AGE
current-time-server-config   1      21s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get configmap -l app=current-time-server -o json
{
    "apiVersion": "v1",
    "items": [
        {
            "apiVersion": "v1",
            "data": {
                "secret": "my-config-secret"
            },
            "kind": "ConfigMap",
            "metadata": {
                "annotations": {
                    "kubectl.kubernetes.io/last-applied-configuration": "{\"apiVersion\":\"v1\",
                     \"data\":{\"secret\":\"my-config-secret\"},\"kind\":\"ConfigMap\",\"metadata\":{\"annotations\":{},
                     \"labels\":{\"app\":\"current-time-server\"},\"name\":\"current-time-server-config\",\"namespace\":\"default\"}}\n"
                },
                "creationTimestamp": "2025-02-22T17:14:57Z",
                "labels": {
                    "app": "current-time-server"
                },
                "name": "current-time-server-config",
                "namespace": "default",
                "resourceVersion": "102338",
                "uid": "2a9c1d04-06af-4562-b263-1a1b59110b08"
            }
        }
    ],
    "kind": "List",
    "metadata": {
        "resourceVersion": ""
    }
}
```

- Then, I applied my `deployment.yaml`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl delete -f secret.yaml
secret "current-time-server-secret" deleted
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml
deployment.apps/current-time-server-deployment configured
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -l app=current-time-server
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-bbd4db585-hdd6w   1/1     Running   0          30s
current-time-server-deployment-bbd4db585-pmlpz   1/1     Running   0          25s
```

- Let's check again if secret exist inside my pods

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get svc current-time-server-service
NAME                         TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
current-time-server-service   LoadBalancer   10.98.173.1    192.168.49.2   80:30954/TCP   47h
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
minikube service current-time-server-service
|-----------|-----------------------------|-------------|-------------------------|
| NAMESPACE |            NAME             | TARGET PORT |           URL           |
|-----------|-----------------------------|-------------|-------------------------|
| default   | current-time-server-service |          80 | http://192.168.49.2:30954 |
|-----------|-----------------------------|-------------|-------------------------|
🏃  Starting tunnel for service current-time-server-service.
|-----------|-----------------------------|-------------|-------------------------|
| NAMESPACE |            NAME             | TARGET PORT |           URL           |
|-----------|-----------------------------|-------------|-------------------------|
| default   | current-time-server-service |             | http://127.0.0.1:56853 |
|-----------|-----------------------------|-------------|-------------------------|
🎉  Opening service default/current-time-server-service in default browser...
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server git:[main]
curl http://127.0.0.1:56853/secret | awk '{print $6}'f
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    55  100    55    0     0   5126      0 --:--:-- --:--:-- --:--:--  5500
e8d4a4f2a01da1d337b45d94b67c7687
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server git:[main]
echo -n "my-config-secret" | md5
e8d4a4f2a01da1d337b45d94b67c7687
```

- We see that it worked!

4

Task description

Create a new file like `config.json` file and put some json data into

- Here is `config.json`:

```
{
  "application": "current-time-server",
  "object": {
    "name": "value"
  }
}
```

5

Task description

Create a new one `configMap` manifest. Connect `configMap` yaml file with `config.json` file to read the data from it.

- I created the following `current-time-server-config-json`:

```
kubectl create configmap current-time-server-config-json --from-file config.json
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl create configmap current-time-server-config-json --from-file=config.json
configmap/current-time-server-config-json created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl label configmap current-time-server-config-json app=current-time-server --overwrite=true
configmap/current-time-server-config-json labeled
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get configmap current-time-server-config-json -o=yaml
apiVersion: v1
data:
  config.json: |
    {
      "application": "current-time-server",
      "object": {
        "name": "value"
      }
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2025-02-22T17:40:22Z"
  labels:
    app: current-time-server
  name: current-time-server-config-json
  namespace: default
  resourceVersion: "103656"
  uid: 8acd2e11-d01b-40e8-b505-755bf20a4da7
```

6

Task description

Update your `Deployment` to add `Volumes` and `VolumeMounts` .

- I updated my `deployment.yaml`

```yaml
<...>
          volumeMounts:
            - name: current-time-server-config-json-volume
              mountPath: /current-time-server
      volumes:
        - name: current-time-server-config-json-volume
          configMap:
            name: current-time-server-config-json
```

- And applied it

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
cat deployment.yaml | grep -B 10 -A 10 volume
              path: /actuator/health
              port: 8080
            periodSeconds: 1
            failureThreshold: 10000
          env:
            - name: SECRET
              valueFrom:
                configMapKeyRef:
                  name: current-time-server-config
                  key: secret
          volumeMounts:
            - name: current-time-server-config-json-volume
              mountPath: /etc/config
      volumes:
        - name: current-time-server-config-json-volume
          configMap:
            name: current-time-server-config-json
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml
deployment.apps/current-time-server-deployment created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -l app=current-time-server
NAME                                              READY   STATUS    RESTARTS   AGE
current-time-server-deployment-685779988b-lktkv   1/1     Running   0          8s
current-time-server-deployment-685779988b-nbx5p   1/1     Running   0          8s
```

7

Task description

With `kubectl`, check the `configMap` details. Make sure that you see the data as plain text

- We can check it see that it displays as a JSON text:

```
kubectl get configmap current-time-server-config-json -o
jsonpath='{.data.config\.json}'
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get configmap current-time-server-config-json -o jsonpath='{.data.config\.json}'
{
  "application": "current-time-server",
  "object": {
    "name": "value"
  }
}
```

- However it also displays as a plain text when we do not specify config field

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get configmap current-time-server-config-json -o jsonpath='{.data}'
{"config.json":"{\n  \"application\": \"current-time-server\",\n  \"object\": {\n    \"name\": \"value\"\n  }\n}\n"}
```

8

Task description

Check the filesystem inside app container to show the loaded file data on the specified path.

- Let's check pods' filesystem

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-685779988b-lktkv   1/1     Running   0          7m28s
current-time-server-deployment-685779988b-nbx5p   1/1     Running   0          7m28s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-deployment-685779988b-lktkv -- ls /etc/config
config.json
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-deployment-685779988b-lktkv -- cat /etc/config/config.json
{
  "application": "current-time-server",
  "object": {
    "name": "value"
  }
}
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-deployment-685779988b-nbx5p -- ls /etc/config
config.json
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec current-time-server-deployment-685779988b-nbx5p -- cat /etc/config/config.json
{
  "application": "current-time-server",
  "object": {
    "name": "value"
  }
}
```

- And we see that the volume `current-time-server-config-json-volume` were actually mounted to both my pods!

9

Task description

Put the results into report

✅

Task 8 - k8s Namespace

> `Namespace` Kubernetes Manifest is designed for different projects and deployment environments isolation. With `Namespaces` we can separate App 1 deployment from App 2 deployment, manage (and isolate) cluster resources for them, define users list to have access either to App 1 or to App 2 deployment. Using `Namespaces` , we also can define a different environments like DEV, TEST, STAGE. In that way, `Namespaces` is a required feature for a real Kubernetes production clusters.

1

Task description

Figure out the necessary `Namespace` spec fields.

- **`apiVersion`** : This field specifies the API version of Kubernetes used.
- **`kind`** : This indicates the type of resource for creating. For namespace it is `Namespace` .
- **`metadata`** : This section contains metadata about the `ConfigMap` , such as its name, namespace, labels, and annotations.
  - **`name`** : The name of the Namespace. This must be a unique identifier **within the cluster** and typically follows DNS label formatting requirements (lowercase alphanumeric characters or '-' and must start and end with an alphanumeric character).

2

Task description

Create a two different `Namespaces` in your k8s cluster

- I created two manifests: `nginx-namespace.yaml` and `current-time-server.yaml`
- `nginx-namespace.yaml`

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: nginx-namespace
  labels:
    app: nginx
```

- `current-time-server.yaml`

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: current-time-server-namespace
  labels:
    app: current-time-server
```

- And applied them

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f nginx-namespace.yaml
namespace/nginx-namespace created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f current-timer-server-namespace.yaml
namespace/current-time-server-namespace created
```

3

Task description

Using `kubectl`, get and describe your `Namespaces`.

- `get`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get namespace -l app -o=wide
NAME                          STATUS   AGE
current-time-server-namespace   Active   74s
nginx-namespace                 Active   82s
```

- `describe`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe namespace -l app
Name:         current-time-server-namespace
Labels:       app=current-time-server
              kubernetes.io/metadata.name=current-time-server-namespace
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.


Name:         nginx-namespace
Labels:       app=nginx
              kubernetes.io/metadata.name=nginx-namespace
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.
```

4

Task description

Deploy two different applications in two different `Namespaces` with `kubectl`. By the way, it's acceptable even just to deploy the same objects (same previous app) in the different `Namespaces` but with different resources names.

- It may be already clear that I am going to deploy my `current-time-server` application with all its resources to `current-time-server-namespace` and a simple **nginx** application to `nginx-namespace`.

`current-time-server`

- I created all the necessary resources

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl create configmap current-time-server-config-json --from-file config.json -n current-time-server-namespace
configmap/current-time-server-config-json created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f secret.yaml -n current-time-server-namespace
secret/current-time-server-secret created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f service.yaml -n current-time-server-namespace
service/current-time-server-service created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f deployment.yaml -n current-time-server-namespace
deployment.apps/current-time-server-deployment created
```

- And it worked

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n current-time-server-namespace
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-698d7d6ff8-qb9z2   1/1     Running   0          3m21s
current-time-server-deployment-698d7d6ff8-tlcgq   1/1     Running   0          3m26s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get svc -n current-time-server-namespace
NAME                         TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
current-time-server-service   LoadBalancer   10.101.210.172   192.168.49.2   80:32254/TCP   5m36s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
docker exec minikube curl -s http://localhost:32254/time
21:34:36Z
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
docker exec minikube curl -s http://localhost:32254/secret
MD5 hash of secret is: ad44f1be904b9375641b666bd4bbc531
```

nginx

- I created a simple Deployment with an **nginx** container. The manifest for it is `nginx-deployment.yaml`:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
```

```
        ports:
          - containerPort: 80
```

- Also, I created a simple Service for make this Deployment accessible from outside. The manifest for it is `nginx-service.yaml`:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  externalIPs:
    - 192.168.49.2
```

- I applied these resources inside the `nginx-namespace` namespace

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f nginx-deployment.yaml -n nginx-namespace
deployment.apps/nginx-deployment created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl apply -f nginx-service.yaml -n nginx-namespace
service/nginx-service created
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n nginx-namespace
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-96b9d695-mfl7c   1/1     Running   0          22s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get svc -n nginx-namespace
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
nginx-service   LoadBalancer   10.102.218.176   192.168.49.2   80:32212/TCP   13s
```

- Finally, I could access the **nginx** application

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
docker exec minikube curl -s http://localhost:32212/ -w '\n'
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

5

Task description

With `kubectl`, get and describe pods from different Namespaces witn `-n` flag.

- `get`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n current-time-server-namespace
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-698d7d6ff8-qb9z2   1/1     Running   0          14m
current-time-server-deployment-698d7d6ff8-tlcgq   1/1     Running   0          14m
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n nginx-namespace
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-96b9d695-mfl7c   1/1     Running   0          5m55s
```

- `describe`

```
kubectl describe po -n current-time-server-namespace
Name:             current-time-server-deployment-698d7d6ff8-qb9z2
Namespace:        current-time-server-namespace
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sat, 22 Feb 2025 21:31:08 +0300
Labels:           app=current-time-server
                  pod-template-hash=698d7d6ff8
Annotations:      <none>
Status:           Running
IP:               10.244.0.120
IPs:
  IP:           10.244.0.120
Controlled By:  ReplicaSet/current-time-server-deployment-698d7d6ff8
Containers:
  current-time-server:
    Container ID:   docker://fee4b3f8161429c6c68cb48daef146539040d1413a810794e3b50917afcf366e
    Image:          iskanred/current-time-server:1.0.0
    Image ID:       docker-pullable://iskanred/current-time-server@sha256:b78f5bec42d299048c0bde0c6e6a853db3ac555cc888b491dc90b367b9da5a3d
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 22 Feb 2025 21:31:08 +0300
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     1
      memory:  512Mi
    Requests:
      cpu:     1
      memory:  512Mi
    Startup:   http-get http://:8080/actuator/health delay=0s timeout=1s period=1s #success=1 #failure=10000
    Environment:
      SECRET:  <set to the key 'secret' in secret 'current-time-server-secret'>  Optional: false
    Mounts:
      /etc/config from current-time-server-config-json-volume (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-48gj9 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl describe po -n nginx-namespace
Name:             nginx-deployment-96b9d695-mfl7c
Namespace:        nginx-namespace
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sat, 22 Feb 2025 21:39:49 +0300
Labels:           app=nginx
                  pod-template-hash=96b9d695
Annotations:      <none>
Status:           Running
IP:               10.244.0.121
IPs:
  IP:           10.244.0.121
Controlled By:  ReplicaSet/nginx-deployment-96b9d695
Containers:
  nginx:
    Container ID:   docker://42ac554acbba9364aa9b9c5cabb4c4a8361de8ee61eba7acbd0c936f893a1fcf
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e34
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 22 Feb 2025 21:40:06 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2pkjz (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-2pkjz:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
```

6

Task description

Can you see and can you connect to the resources from different Namespaces?

- Yes. For example in `nginx-namespace`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n nginx-namespace
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-96b9d695-mfl7c   1/1     Running   0          8m41s
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec -n nginx-namespace nginx-deployment-96b9d695-mfl7c  -- curl -s http://localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

- Or in `current-time-server-namespace`

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl get po -n current-time-server-namespace
NAME                                          READY   STATUS    RESTARTS   AGE
current-time-server-deployment-698d7d6ff8-qb9z2   1/1     Running   0          18m
current-time-server-deployment-698d7d6ff8-tlcgq   1/1     Running   0          18m
```

```
~/Study/iu-devsecops-course/lab-04/current-time-server/k8s git:[main]
kubectl exec -n current-time-server-namespace current-time-server-deployment-698d7d6ff8-qb9z2  -- curl -s http://localhost:8080/time -w '\n'
21:49:34
```

7

Task description

Put the results into report

✅

Post scriptum

- You can find all the materials of this lab in my GitHub repo of the course https://github.com/iskanred/iu-devsecops-course/tree/main/lab-04
- Here is my file structure with files related to k8s for this lab