

Paweł Guzek 224304
Dominik Karski 224322
Dominik Kasierski 224323
Michał Maksajda 224369
Krzysztof Szczesniak 224434
Mateusz Szewc 224438
Mateusz Zabroń 224464

Zadanie 3 – Wdrożenie i utrzymanie

Techniki utrzymania aplikacji
2021

Spis treści:

Wdrożenie bazy danych MariaDB	3
Wdrożenie aplikacji Web na serwerze Wildfly 25.0 (s2i)	7
Ręczne skalowanie	11
Odtworzenie poda	12
Health check	14
Metryki	17
Skalowanie horyzontalne	24

1. Wdrożenie bazy danych MariaDB

- a. Utworzenie pliku ConfigMap, który jest odpowiedzialny za wykonanie skryptu `create.sql` (utworzenie struktur relacyjnej bazy danych, wypełnienie danymi i nadanie uprawnień do tabel bazodanowych)
- b. Stworzenie sekretu
Kolejnym krokiem było stworzenie sekretu przechowującego zmienne środowiskowe potrzebne do stworzenia "stanowego zbioru" przez kubernetes, zawiera on informacje o nazwie bazy danych, hasła użytkownika, hasła użytkownika uprzywilejowanego oraz nazwie użytkownika.

```
kind: Secret
apiVersion: v1
metadata:
  name: mariadb-env
  namespace: tua03
data:
  MYSQL_DATABASE: dHVhMDM=
  MYSQL_PASSWORD: dHVhMDM=
  MYSQL_ROOT_PASSWORD: dHVhMDM=
  MYSQL_USER: dHVhMDM=
type: Opaque
```

- c. Utworzenie StatefulSet

Zawartość zbioru stanowego:

- wersja obrazu mariadb - wybrano `latest`,
- zmienne środowiskowe niezbędne do inicjalizacji bazy danych, takie jak `MYSQL_USER`, `MYSQL_PASSWORD`, `MYSQL_DATABASE`, `MYSQL_ROOT_PASSWORD`. Wartości zmiennych środowiskowych jest przechowywana w sekretach.
- sposób przeprowadzania mechanizmu healthcheck - wykorzystanie polecenia `mysqladmin ping`. Właściwości `initialDelaySecond` (oczekiwanie przed przeprowadzeniem pierwszej próby), `periodSeconds` (każda próba następuje co 5 sekund) oraz `timeoutSeconds` (ilość sekund, po której następuje timeout danej próby) ustawiono kolejno wartościami 30, 10 oraz 5.
- mapowanie `PersistenceVolume` `data` oraz `ConfigMap` `init-db` do odpowiednio `/var/lib/mysql` oraz `/docker-entrypoint-initdb.d`
- Utworzenie `PersistenceVolumeClaim`, czyli polecenia przyznania zasobów magazynowych dla poda, na którym uruchomiono mariadb.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mariadb
  namespace: tua03
spec:
  selector:
    matchLabels:
      app: mariadb
  serviceName: "mariadb"
  replicas: 1
  template:
    metadata:
      labels:
        app: mariadb
    spec:
      terminationGracePeriodSeconds: 10
      restartPolicy: Always
      containers:
        - name: mariadb
          image: mariadb:latest
          env:
            - name: MYSQL_USER
              valueFrom:
                secretKeyRef:
                  name: mariadb-env
                  key: MYSQL_USER
            - name: MYSQL_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mariadb-env
                  key: MYSQL_PASSWORD
            - name: MYSQL_DATABASE
              valueFrom:
                secretKeyRef:
                  name: mariadb-env
                  key: MYSQL_DATABASE
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mariadb-env
                  key: MYSQL_ROOT_PASSWORD
      livenessProbe:
        exec:
```

```

        command: ["mysqladmin", "ping", "-u",
"root", "-ptua03"]
        initialDelaySeconds: 30
        periodSeconds: 10
        timeoutSeconds: 5
    ports:
      - containerPort: 3306
        name: mariadb
    volumeMounts:
      - name: data
        mountPath: /var/lib/mysql
      - name: init-db
        mountPath: /docker-entrypoint-initdb.d
    volumes:
      - name: init-db
        configMap:
          name: init-db
    volumeClaimTemplates:
      - metadata:
          name: data
        spec:
          accessModes: [ "ReadWriteOnce" ]
          storageClassName: "thin"
          resources:
            requests:
              storage: 1Gi

```

d. Utworzenie usługi

Usługę stworzono aby umożliwić innym podom komunikację z mariadb, udostępnia ona port 3306/TCP, natomiast jej nazwa domenowa to mariadb.

```

apiVersion: v1
kind: Service
metadata:
  name: mariadb
  namespace: tua03
spec:
  selector:
    app: mariadb
  type: ClusterIP
  ports:
    - name: mariadb
      protocol: TCP

```

```
port: 3306  
targetPort: 3306
```

2. Wdrożenie aplikacji Web na serwerze Wildfly 25.0 (s2i)

Do wdrożenia aplikacji na kubernetes został wykorzystany "Helm Charts". Przygotowana konfiguracja umożliwia automatyczne wygenerowanie obrazu budowniczego, uruchomieniowego, konfiguracji usługi oraz konfiguracji routera.

```
image:
  name: tua03
  tag: latest
build:
  enabled: true
  mode: s2i
  uri: 'https://github.com/TULbaghia/tua'
  output:
    kind: ImageStreamTag
    pushSecret: builder-dockercfg-kz8pw
  triggers: {}
  s2i:
    version: '25.0'
    builderImage: quay.io/wildfly/wildfly-centos7
    runtimeImage: quay.io/wildfly/wildfly-runtime-centos7
  bootableJar:
    builderImage:
'registry.access.redhat.com/ubi8/openjdk-11:latest'
deploy:
  enabled: true
  replicas: 1
  route:
    enabled: true
    tls:
      enabled: true
      termination: edge
      insecureEdgeTerminationPolicy: Redirect
  livenessProbe:
    httpGet:
      path: /health/live
      port: admin
  readinessProbe:
    httpGet:
      path: /health/ready
      port: admin
```

Po wgraniu konfiguracji należy umieścić prawidłową nazwę obrazu w “*Deployments*”, można ją pobrać z “*ImageStreams*” > “*Image repository*”.


[ImageStreams](#) > ImageStream details


tua03-build-artifacts

[Details](#) [YAML](#) [History](#)

Image Stream details

Name
tua03-build-artifacts

Namespace
 tua03

Labels [Edit](#) 

[app.kubernetes.io/instance=wildfly](#) [app.kubernetes.io/managed-by=Helm](#) [app.kubernetes.io/name=wildfly](#)
[app.kubernetes.io/version=24.0](#) [helm.sh/chart=wildfly-1.4.0](#)



Annotations
[2 annotations](#) 

Image repository
`image-registry.openshift-image-registry.svc:5000/tua03/tua03-build-artifacts`

Image count
1

Created at
 Nov 26, 2021, 6:16 PM

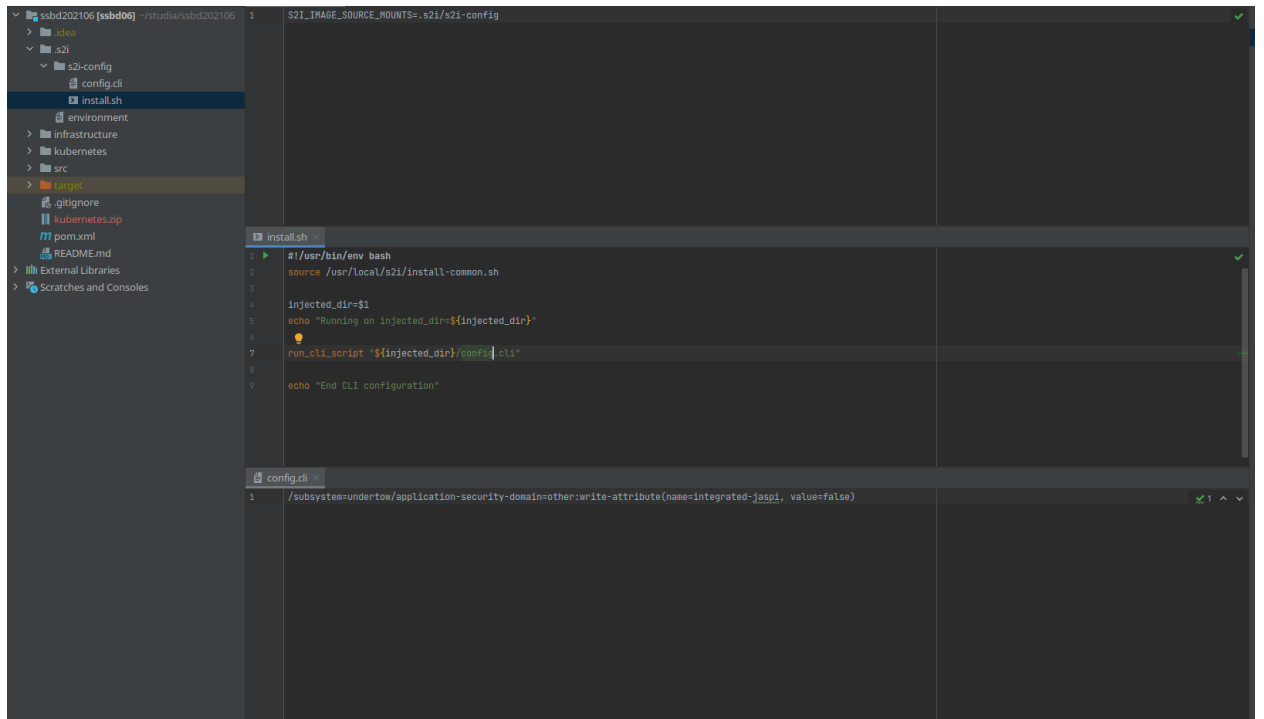
Owner
No owner

1.


```
250     - name: jolokia
251       containerPort: 8778
252       protocol: TCP
253     - name: http
254       containerPort: 8080
255       protocol: TCP
256     - name: ping
257       containerPort: 8888
258       protocol: TCP
259     - name: admin
260       containerPort: 9990
261       protocol: TCP
262     imagePullPolicy: Always
263     terminationMessagePolicy: File
264     image: "image-registry.openshift-image-registry.svc:5000/tua03/tua03-build-artifacts:latest"
265     restartPolicy: Always
266     terminationGracePeriodSeconds: 30
267     dnsPolicy: ClusterFirst
268     securityContext: {}
269     schedulerName: default-scheduler
270   strategy:
271     type: Recreate
272     revisionHistoryLimit: 10
273     progressDeadlineSeconds: 600
274   status:
275     observedGeneration: 19
276     replicas: 1
277     updatedReplicas: 1
278     readyReplicas: 1
279     availableReplicas: 1
```

2.

W celu skonfigurowania serwera Wildfly, wyłączenie zintegrowanego realma (jaspic), do repozytorium dodano katalog ".s2i" z plikiem *environment* umożliwiającym na wskazanie katalogu z konfiguracją dla budowniczego obrazu. Kolejnym krokiem było stworzenie skryptu o nazwie "*install.sh*" który służy do załadowania pliku konfiguracyjnego serwer Wildfly. Plik konfiguracyjny "*config.cli*" zawiera listę komend, które mają wykonać się podczas etapu konfiguracji serwera wykonywanego przez budowniczego s2i.



3. Ręczne skalowanie

W celu sprawdzenia podziału żądań kierowanych do aplikacji uruchomiono panel monitoringu w konsoli Openshift. W statystykach zużycia CPU można uwzględnić podział na pody, ten czynnik został wykorzystany do weryfikacji prawidłowego podziału obsługi żądań po przeskalowaniu.



Od 9:24 odnotowano znaczny wzrost zużycia CPU, ze względu na sztucznie wygenerowany ruch. Przy użyciu narzędzia ApacheBench kierowano żądania w ilości 50/s na podany endpoint. Po utworzeniu drugiego poda (oznaczony zieloną linią), można zauważyć, że część requestów zostaje przez niego obsłużona.

4. Odtworzenie poda

Za zachowanie poda po jego wyłączeniu odpowiada parametr *restartPolicy* znajdujący się w plikach konfiguracyjnych deploymentu. Odnosi się on do wszystkich kontenerów znajdujących się wewnątrz poda.

```
image: >-
  image-registry.openshift-image-registry.svc
restartPolicy: Always
terminationGracePeriodSeconds: 30
dnsPolicy: ClusterFirst
securityContext: {}
schedulerName: default-scheduler
strategy:
```

Parametr ten może przyjąć jedną z trzech wartości:

- Always - oznacza, że kontener będzie restartowany zawsze, nawet jeśli kod wyjścia będzie równy 0, jest to domyślna wartość,
- OnFailure - oznacza, że kontener będzie restartowany tylko wówczas, gdy kod wyjścia będzie różny od 0,
- Never - oznacza, że niezależnie od kodu wyjścia kontener nigdy nie będzie restartowany.

Prezentacja działania:

Usunięcie poda poprzez kliknięcie przycisku “Delete Pod”.

Project: tua03

Pods Create Pod

Filter Name Search by name...

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
grafana-operator-controller-manager-867d9dfbdc-smq2	Running	2/2	950	grafana-operator-controller-manager-867d9dfbdc	31,5 MiB	0,003 cores	26 lis 2021, 18:13
mariaadb-0	Running	1/1	0	mariaadb	117,5 MiB	0,002 cores	26 lis 2021, 20:16
prometheus-operator-5cbb499d48-6zwm5	Running	1/1	0	prometheus-operator-5cbb499d48	22,2 MiB	0,000 cores	9 lis 2021, 09:38
prometheus-wildfly-prom-0	Running	2/2	1	prometheus-wildfly-prom	32,8 MiB	0,002 cores	30 lis 2021, 13:26
prometheus-wildfly-prom-1	Running	2/2	1	prometheus-wildfly-prom	30,5 MiB	0,002 cores	1 gru 2021, 04:42
tua03-build-artifacts-7-build	Completed	0/1	0	tua03-build-artifacts-7	-	-	1 gru 2021, 02:17
tua03-build-artifacts-8-build	Completed	0/1	0	tua03-build-artifacts-8	-	-	1 gru 2
tua03-build-artifacts-9-build	Completed	0/1	0	tua03-build-artifacts-9	-	-	1 gru 2
tua03-build-artifacts-10-build	Completed	0/1	0	tua03-build-artifacts-10	-	-	1 gru 2
wildfly-6fcc8765c8-9k58n	Running	1/1	0	wildfly-6fcc8765c8	504,9 MiB	0,014 cores	9 minutes ago

Stary pod zostaje usunięty (“*Terminating*”), a w jego miejsce powstaje nowy (“*ContainerCreating*”).

Project: tua03

Pods

Create Pod

Filter Name Search by name...

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
grafana-operator-controller-manager-867d9dfbdc-smq2	Running	2/2	950	grafana-operator-controller-manager-867d9dfbdc	31,4 MiB	0,004 cores	26 lis 2021, 18:13
mariaadb-0	Running	1/1	0	mariaadb	117,5 MiB	0,002 cores	26 lis 2021, 20:16
prometheus-operator-5cbb499d48-6zwms	Running	1/1	0	prometheus-operator-5cbb499d48	22,2 MiB	0,000 cores	9 lis 2021, 09:38
prometheus-wildfly-prom-0	Running	2/2	1	prometheus-wildfly-prom	32,9 MiB	0,002 cores	30 lis 2021, 13:26
prometheus-wildfly-prom-1	Running	2/2	1	prometheus-wildfly-prom	30,4 MiB	0,002 cores	1 gru 2021, 04:42
tua03-build-artifacts-7-build	Completed	0/1	0	tua03-build-artifacts-7	-	-	1 gru 2021, 02:17
tua03-build-artifacts-8-build	Completed	0/1	0	tua03-build-artifacts-8	-	-	1 gru 2021, 18:10
tua03-build-artifacts-9-build	Completed	0/1	0	tua03-build-artifacts-9	-	-	1 gru 2021, 19:23
tua03-build-artifacts-10-build	Completed	0/1	0	tua03-build-artifacts-10	-	-	1 gru 2021, 19:49
wildfly-6fcc8765c8-9k58n	Terminating	1/1	0	wildfly-6fcc8765c8	505,1 MiB	0,009 cores	10 minutes ago
wildfly-6fcc8765c8-vdb7t	ContainerCreating	0/1	0	wildfly-6fcc8765c8	-	-	a few seconds ago

Nowy pod został utworzony i uruchomiony (“*Running*”).

Project: tua03

Pods

Create Pod

Filter Name Search by name...

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
grafana-operator-controller-manager-867d9dfbdc-smq2	Running	2/2	950	grafana-operator-controller-manager-867d9dfbdc	31,4 MiB	0,004 cores	26 lis 2021, 18:13
mariaadb-0	Running	1/1	0	mariaadb	117,4 MiB	0,002 cores	26 lis 2021, 20:16
prometheus-operator-5cbb499d48-6zwms	Running	1/1	0	prometheus-operator-5cbb499d48	22,2 MiB	0,000 cores	9 lis 2021, 09:38
prometheus-wildfly-prom-0	Running	2/2	1	prometheus-wildfly-prom	32,9 MiB	0,002 cores	30 lis 2021, 13:26
prometheus-wildfly-prom-1	Running	2/2	1	prometheus-wildfly-prom	30,5 MiB	0,001 cores	1 gru 2021, 04:42
tua03-build-artifacts-7-build	Completed	0/1	0	tua03-build-artifacts-7	-	-	1 gru 2021, 02:17
tua03-build-artifacts-8-build	Completed	0/1	0	tua03-build-artifacts-8	-	-	1 gru 2021, 18:10
tua03-build-artifacts-9-build	Completed	0/1	0	tua03-build-artifacts-9	-	-	1 gru 2021, 19:23
tua03-build-artifacts-10-build	Completed	0/1	0	tua03-build-artifacts-10	-	-	1 gru 2021, 19:49
wildfly-6fcc8765c8-vdb7t	Running	1/1	0	wildfly-6fcc8765c8	206,4 MiB	-	a minute ago

5. Health check

Aby prawidłowo funkcjonowała usługa health check niezbędne było wykonanie następujących komend wewnątrz kontenera:

```
/extension=org.wildfly.extension.microprofile.health-smallrye:add
/subsystem=microprofile-health-smallrye:add
/subsystem=microprofile-health-smallrye:write-attribute(name=security
-enabled, value=false)
```

- **Readiness probe** - określa czy kontener jest gotowy do obsługi żądań. Jeśli próba się nie powiedzie adres IP pada usuwany jest ze wszystkich jego serwisów.

Fragment pliku konfiguracyjnego, który za to odpowiada:

```
spec:
  containers:
    - resources: {}
      readinessProbe:
        httpGet:
          path: /health/ready
          port: admin
          scheme: HTTP
        timeoutSeconds: 2
        periodSeconds: 10
        successThreshold: 1
        failureThreshold: 3
```

Implementacja po stronie aplikacji:

```
@ApplicationScoped
@Readiness
public class ReadinessHealthCheck implements HealthCheck {

    private final MemoryMXBean memoryMXBean =
ManagementFactory.getMemoryMXBean();

    @Override
    public HealthCheckResponse call() {
        return HealthCheckResponse.named("heap-memory")
            .status(getUsedMemory() < 0.9 * getMaxMemory())
            .build();
    }
}
```

```

private long getUsedMemory() {
    return memoryMXBean.getHeapMemoryUsage().getUsed();
}

private long getMaxMemory() {
    return memoryMXBean.getHeapMemoryUsage().getMax();
}
}

```

- **Liveness probe** - określa czy kontener jest uruchomiony. Jeśli próba się nie powiedzie kontener zostaje wyłączony, a następne kroki zależne są od polityki restartu opisanej w poprzednim punkcie.

Fragment pliku konfiguracyjnego, który za to odpowiada:

```

spec:
  containers:
    - resources: {}
      (...)
      livenessProbe:
        httpGet:
          path: /health/live
          port: admin
          scheme: HTTP
        timeoutSeconds: 2
        periodSeconds: 10
        successThreshold: 1
        failureThreshold: 3

```

Implementacja po stronie aplikacji:

```

@ApplicationScoped
public class LivenessKeeper {

    @Getter
    @Setter
    private boolean isClicked = true;
}

@ApplicationScoped
@Liveness
public class LivenessHealthCheck implements HealthCheck {

    @Inject
    private LivenessKeeper keeper;
}

```

```

@Override
public HealthCheckResponse call() {
    return HealthCheckResponse.named("is-app-up")
        .status(keeper.isClicked())
        .build();
}

}

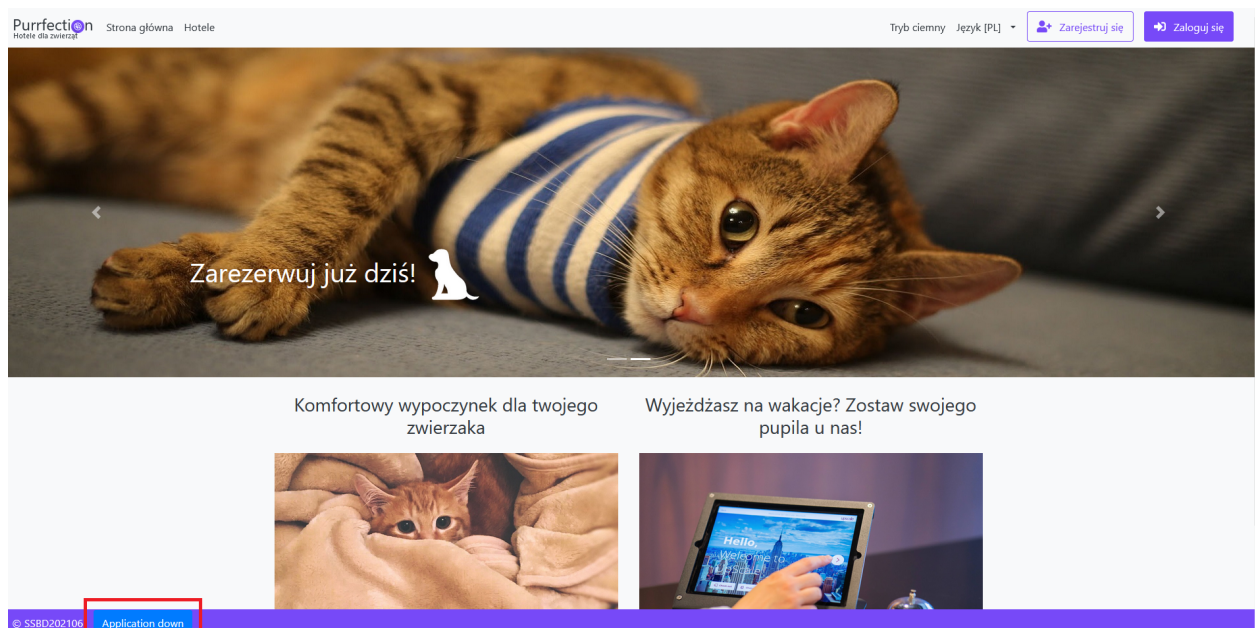
@ApplicationScoped
@Path("/health")
public class LivenessChangeController {

    @Inject
    private LivenessKeeper keeper;

    @GET
    @Path("toggle")
    public String toggle() {
        keeper.setClicked(!keeper.isClicked());
        return String.valueOf(keeper.isClicked());
    }
}

```

W warstwie widoku dodany został również przycisk odpowiedzialny za zmianę wartości pola klasy *LivenessKeeper* (*boolean isClicked*), od którego zależna jest odpowiedź zwracana przez endpoint */health/live*.



6. Metryki

Aby prawidłowo funkcjonowały aplikacyjne metryki niezbędne było wykonanie następujących komend wewnątrz kontenera:

```
/extension=org.wildfly.extension.microprofile.metrics-smallrye:add()  
/subsystem=microprofile-metrics-smallrye:add()  
/subsystem=microprofile-metrics-smallrye:write-attribute(name=security-enabled, value=false)
```

Metryki aplikacyjne zostały zaimplementowane dla endpointów:

GET /hotels

```
@GET  
@Produces(MediaType.APPLICATION_JSON)  
+ @Operation(operationId = "getAllHotelsList", summary = "getAllHotelsList")  
+ @Metered(name = "hotel.getAll.frequency", absolute = true)  
  @Timed(name = "hotel.getAll.duration", unit = MetricUnits.MILLISECONDS)  
  public List<HotelDto> getAll() throws AppBaseException {  
      return repeat(() -> hotelEndpoint.getAll(), hotelEndpoint);  
  }
```

POST /bookings

```
@POST  
@RolesAllowed("bookReservation")  
@Operation(operationId = "addBooking", summary = "addBooking")  
@Consumes({MediaType.APPLICATION_JSON})  
+ @Timed(name = "reservationTime", description = "Metrics to monitor booking  
ordering time",  
  unit = MetricUnits.SECONDS, absolute = true)  
  public void addBooking(NewBookingDto bookingDto) throws AppBaseException {  
      repeat(() -> bookingEndpoint.addBooking(bookingDto), bookingEndpoint);  
  }
```

W wildfly został udostępniony port console udostępniający metryki:

```
spec:  
  ports:  
    - name: site  
      protocol: TCP  
      port: 8080  
      targetPort: 8080  
    - name: console  
      protocol: TCP  
      port: 9990  
      targetPort: 9990
```

```
selector:
  app.kubernetes.io/instance: wildfly
  app.kubernetes.io/name: wildfly
clusterIP: 172.30.115.80
clusterIPs:
  - 172.30.115.80
```

Metryki znajdowały się pod adresem <http://wildfly-console-tua03.apps.okd.cti.p.lodz.pl/metrics>

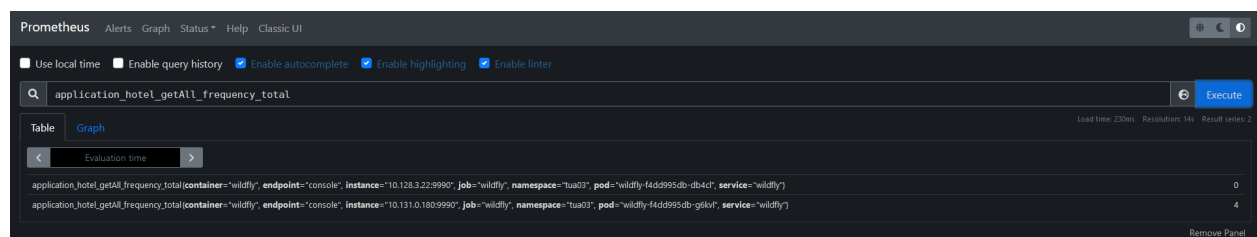
```
# TYPE application_hotel_getAll_frequency_total counter
application_hotel_getAll_frequency_total 4.0
# TYPE application_hotel_getAll_frequency_rate_per_second gauge
application_hotel_getAll_frequency_rate_per_second 4.105478715348272E-5
# TYPE application_hotel_getAll_frequency_one_min_rate_per_second gauge
application_hotel_getAll_frequency_one_min_rate_per_second 2.964393875E-314
# TYPE application_hotel_getAll_frequency_five_min_rate_per_second gauge
application_hotel_getAll_frequency_five_min_rate_per_second 1.9506298580564264E-126
# TYPE application_hotel_getAll_frequency_fifteen_min_rate_per_second gauge
application_hotel_getAll_frequency_fifteen_min_rate_per_second 1.906811900413231E-44
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_rate_per_second gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_rate_per_second 4.1054787588488476E-5
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_one_min_rate_per_second gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_one_min_rate_per_second 2.964393875E-314
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_five_min_rate_per_second gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_five_min_rate_per_second 1.9506298580564264E-126
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_fifteen_min_rate_per_second gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_fifteen_min_rate_per_second 1.906811900413231E-44
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_min_seconds gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_min_seconds 0.0
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_max_seconds gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_max_seconds 0.0
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_mean_seconds gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_mean_seconds 0.0
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_stddev_seconds gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_stddev_seconds 0.0
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds summary
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds_count 4.0
# TYPE application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds_sum gauge
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds_sum 0.586259889
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds{quantile="0.5"} 0.0
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds{quantile="0.75"} 0.0
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds{quantile="0.95"} 0.0
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds{quantile="0.98"} 0.0
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds{quantile="0.99"} 0.0
application_pl_lodz_p_it_ssb06_controllers_HotelController_hotel_getAll_duration_seconds{quantile="0.999"} 0.0
# TYPE application_reservationTime_rate_per_second gauge
application_reservationTime_rate_per_second 0.0
# TYPE application_reservationTime_one_min_rate_per_second gauge
application_reservationTime_one_min_rate_per_second 0.0
# TYPE application_reservationTime_five_min_rate_per_second gauge
application_reservationTime_five_min_rate_per_second 0.0
# TYPE application_reservationTime_fifteen_min_rate_per_second gauge
application_reservationTime_fifteen_min_rate_per_second 0.0
# TYPE application_reservationTime_min_seconds gauge
application_reservationTime_min_seconds 0.0
```

Na zainstalowanym operatorze *prometheus-operator* dodana została nowa instancja prometeusza, po czym stworzony został nowy *ServiceMonitor*, który został skonfigurowany do monitorowania serwisu udostępniającego port *console*.


ServiceMonitor został skonfigurowany zgodnie z poniższym yamlem.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  creationTimestamp: '2021-12-04T15:53:04Z'
  generation: 7
  managedFields:
    - apiVersion: monitoring.coreos.com/v1
      fieldsType: FieldsV1
      fieldsV1:
        'f:spec':
          .: {}
          'f:endpoints': {}
          'f:selector': {}
      manager: Mozilla
      operation: Update
      time: '2021-12-04T17:13:28Z'
  name: example
  namespace: tua03
  resourceVersion: '53577635'
  selfLink:
/apis/monitoring.coreos.com/v1/namespaces/tua03/servicemonitors/example
  uid: 442ccaf8-724d-40fa-8c8f-581e9b9c1b24
spec:
  endpoints:
    - interval: 10s
      port: console
  selector: {}
```

W celu przetestowania konfiguracji prometheus'a sprawdzona została metryka dotycząca liczby wywołań metody zwracającej wszystkie hotele.



Grafana została dodana komendą `oc new-app grafana/grafana`, po czym dodany został route do serwisu grafana komendą `oc expose service/grafana`. Po wejściu na url <http://grafana-tua03.apps.okd.cti.p.lodz.pl> i zalogowaniu się na konto administratora możliwe było dodanie *Data Source* prometheus'a.




Data Sources / Prometheus

Type: Prometheus

Settings

Dashboards



Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hos

Name

Prometheus

Default

☒

HTTP

URL

Access

Server (default)

Help >

Allowed cookies

Timeout

Auth

Basic auth

☐

With Credentials

☐

TLS Client Auth

☐

With CA Cert

☐

Skip TLS Verify

☐

Forward OAuth Identity

☐

Custom HTTP Headers

+ Add header

Po dodaniu *Data Source*'a zaimportowany został przygotowany wcześniej dashboard z pliku json:

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "target": {
          "limit": 100,
          "matchAny": false,
          "tags": [],
          "type": "dashboard"
        },
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "fiscalYearStartMonth": 0,
  "gnetId": null,
  "graphTooltip": 0,
  "id": 1,
  "links": [],
  "liveNow": false,
  "panels": [
    {
      "datasource": null,
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisLabel": "",
```

```
    "axisPlacement": "auto",
    "barAlignment": 0,
    "drawStyle": "line",
    "fillOpacity": 0,
    "gradientMode": "none",
    "hideFrom": {
      "legend": false,
      "tooltip": false,
      "viz": false
    },
    "lineInterpolation": "linear",
    "lineWidth": 1,
    "pointSize": 5,
    "scaleDistribution": {
      "type": "linear"
    },
    "showPoints": "auto",
    "spanNulls": false,
    "stacking": {
      "group": "A",
      "mode": "none"
    },
    "thresholdsStyle": {
      "mode": "off"
    }
  },
  "mappings": [],
  "thresholds": {
    "mode": "absolute",
    "steps": [
      {
        "color": "green",
        "value": null
      },
      {
        "color": "red",
        "value": 80
      }
    ]
  }
]
```

```
    }
  },
  "overrides": [],
},
"gridPos": {
  "h": 9,
  "w": 12,
  "x": 0,
  "y": 0
},
"id": 2,
"options": {
  "legend": {
    "calcs": [],
    "displayMode": "list",
    "placement": "bottom"
  },
  "tooltip": {
    "mode": "single"
  }
},
"targets": [
  {
    "exemplar": true,
    "expr": "application_reservationTime_mean_seconds{}",
    "interval": "",
    "legendFormat": "",
    "refId": "A"
  }
],
"title": "Mean reservation time",
"type": "timeseries"
}
],
"schemaVersion": 32,
"style": "dark",
"tags": [],
"templating": {
  "list": []
}
```

```

},
"time": {
  "from": "now-6h",
  "to": "now"
},
"timepicker": {},
"timezone": "",
"title": "Purrfect Dashboard",
"uid": "b58Wj1t7z",
"version": 1
}

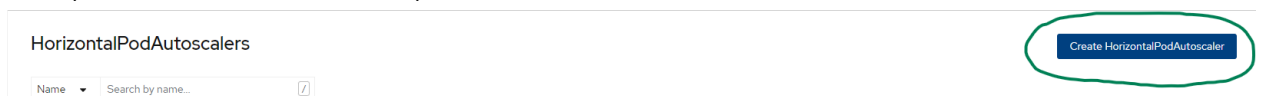
```

Dodany Dashboard widoczny był w Grafanie jako wykres średniego czasu dodawania rezerwacji:



7. Skalowanie horyzontalne

W celu wykorzystania mechanizmu skanowania horyzontalnego utworzony został nowy HPA (Horizontal Pod Autoscaler).



Jego konfiguracja zawarta w pliku YAML prezentuje się następująco:

```

apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: wildfly-hpa
  namespace: tua03
spec:
  scaleTargetRef:

```



```

    kind: Deployment
    name: wildfly
    apiVersion: apps/v1
  minReplicas: 1
  maxReplicas: 5
  metrics:
    - type: Resource
      resource:
        name: cpu
      target:
        type: AverageValue
        averageValue: 120m
  behavior:
    scaleDown:
      stabilizationWindowSeconds: 60

```

Przed zwiększeniem obciążenia aplikacji użycie jednostki przetwarzającej jest minimalne przez co liczba POD-ów jest równa 1(*currentReplicas*):

```

status:
  lastScaleTime: '2021-11-26T19:06:47Z'
  currentReplicas: 1
  desiredReplicas: 1
  currentMetrics:
    - type: Resource
      resource:
        name: cpu
      current:
        averageValue: 3m

```

Po sztucznym zwiększeniu obciążenia wykorzystanie jednostki przetwarzającej znacznie wzrasta. Po przekroczeniu limitu ustalonego w pliku konfiguracyjnym HPA, liczba pożądaných POD-ów(*desiredReplicas*) ulega zmianie, dodatkowo zmienia się znacznik czasu oznaczający czas ostatniego skalowania (*lastScaleTime*) :

```

status:
  lastScaleTime: '2021-11-26T19:53:34Z'
  currentReplicas: 2
  desiredReplicas: 4
  currentMetrics:
    - type: Resource
      resource:
        name: cpu
      current:
        averageValue: 477m

```

Następnie żądane POD-y zostają utworzone:

```
status:
  lastScaleTime: '2021-11-26T19:53:34Z'
  currentReplicas: 4
  desiredReplicas: 4
  currentMetrics:
    - type: Resource
      resource:
        name: cpu
        current:
          averageValue: 477m
```

Po spadku obciążenia następuje redukcja liczby POD-ów:

```
status:
  lastScaleTime: '2021-11-26T19:56:07Z'
  currentReplicas: 1
  desiredReplicas: 1
  currentMetrics:
    - type: Resource
      resource:
        name: cpu
        current:
          averageValue: 4m
```