

# Deployment, Hosting and Monitoring

SoftUni Team  
Technical Trainers



**SoftUni**



Software University

<https://softuni.bg>

# Table of Contents

1. Deployment
2. Actuator
3. Micrometer
4. Prometheus



sli.do

**#java-web**



**Deployment**

# What is Deployment?

- **Deployment** means to push changes or updates from one environment to another



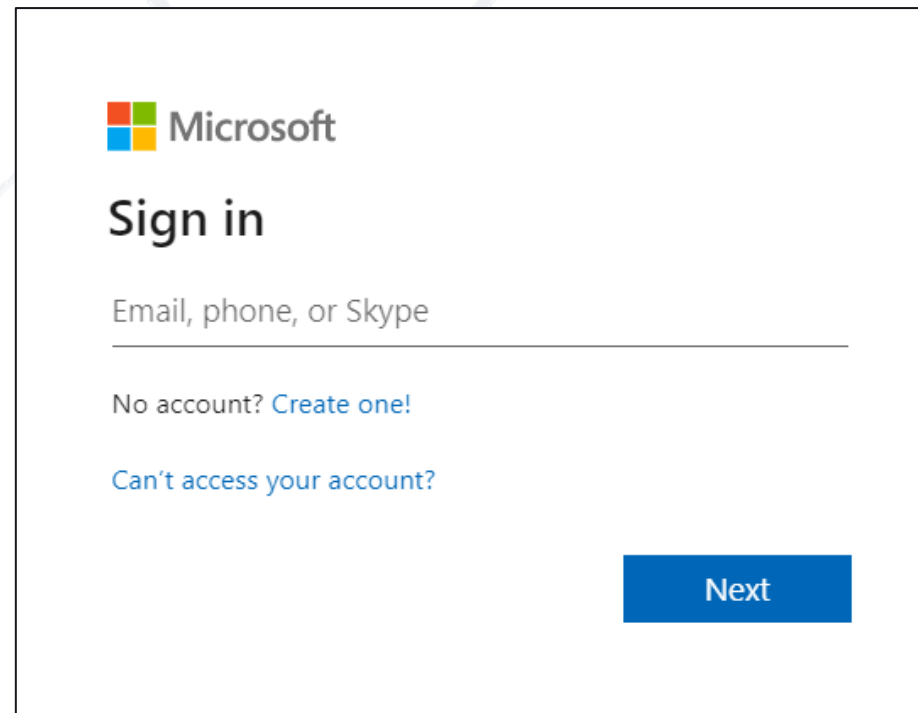
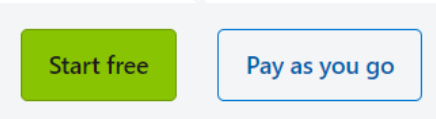
# Where to Deploy?

- We can deploy **one project** onto **multiple websites**
- Some of the deployment websites
  - Azure
  - Amazon Web Services (AWS)
  - Google Cloud Platform



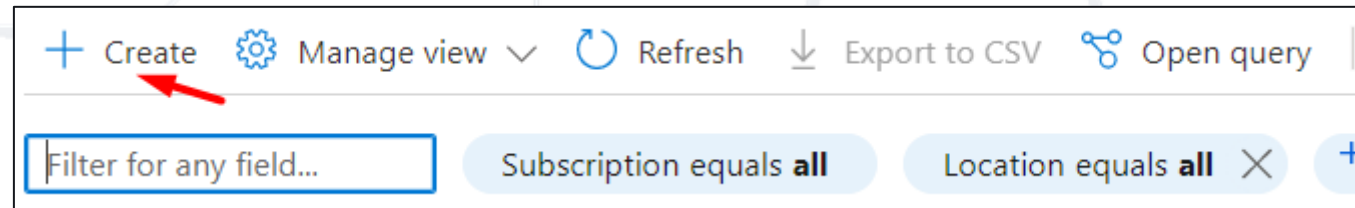
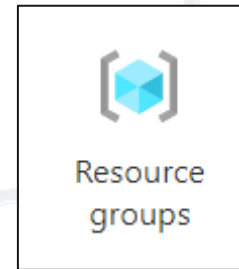
# Create an account in Azure

- Go to <https://azure.microsoft.com/en-us/free/>
- Then click on the **[Start free]** button
- Type in your **SoftUni student e-mail** and click on the **[Next]** button

A screenshot of the Microsoft sign-in page. At the top is the Microsoft logo. Below it is the text 'Sign in'. There is a text input field with the placeholder text 'Email, phone, or Skype'. Below the input field are two links: 'No account? Create one!' and 'Can't access your account?'. At the bottom right is a blue button with the text 'Next'.

# Configure Resource group

- Go to Resource groups
- Create new





- Inside the Resource group click **create** and
- Search for **Azure Spring Apps**
- Create new **Azure Spring Apps**

Subscription \* ⓘ Azure subscription 1 ▼

Resource group \* ⓘ myResourceGroup ▼  
[Create new](#)

**Service Details**

Name \* ⓘ my-service ✓

Region \* ⓘ East US ▼

Zone Redundant \* ⓘ ☐

Pricing \* ⓘ

**Basic tier**  
2 vCPUs, 4 Gi included  
Starting from 170.82 USD/month, charged per second basis  
[Change](#)



Azure Spring Apps

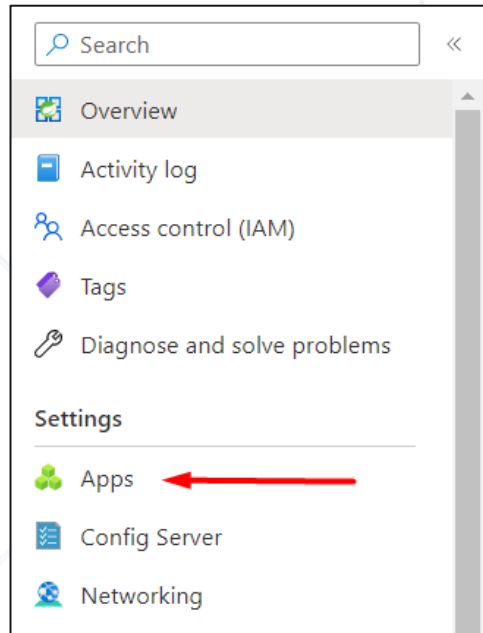
Microsoft

Azure Service

Deploy and manage your Spring apps


# Create an App

- Inside your service go to **Apps**



- Create an App

Choose the amount of vCPU and memory that you want to allocate to your new app. You can also increase app instances here.

| App name             | Deployment Type ⓘ            | Runtime platform | Image | vCPU | Memory | Instance count  |
|----------------------|------------------------------|------------------|-------|------|--------|---|
| <input type="text"/> | Artifacts (Java/.Net Core) ▼ | Java 11 ▼        | N/A   | 1 ▼  | 1 Gi ▼ | 1  |

- Inside your Resource Group click on create



- Search for **Azure Database for MySQL Flexible Server** and create one

**Server details**

Enter required settings for this server, including picking a location and configuring the compute and storage resources.

|                     |   |
|---------------------|---|
| Server name * ⓘ     | <input type="text" value="my-db"/> ✓  |
| Region * ⓘ          | <input type="text" value="East US"/> ▼  |
| MySQL version * ⓘ   | <input type="text" value="8.0"/> ▼  |
| Workload type ⓘ     | <div><input type="radio"/> For small or medium size databases</div> <div><input checked="" type="radio"/> Tier 1 Business Critical Workloads</div> <div><input type="radio"/> For development or hobby projects</div> |
| Compute + storage ⓘ | <div><b>Burstable, B1ms</b></div> <div>1 vCores, 2 GiB RAM, 20 GiB storage, 360 IOPS</div> <div><b>Geo-redundancy : Disabled</b></div> <div><a href="#">Configure server</a></div>                                    |
| Availability zone ⓘ | <input type="text" value="No preference"/> ▼  |

## ■ Networking:

Basics Networking Security Tags Review + create

### ■ Firewall rules

- Allow public access
- Add Current client Ip address

#### Firewall rules

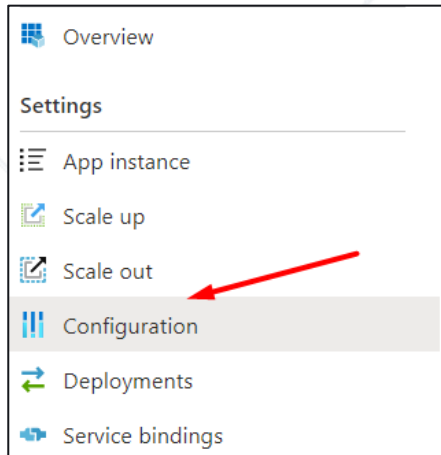
Inbound connections from the IP addresses specified below will be allowed to port 3306 on this server. [Learn more](#) ↗

☒ Allow public access from any Azure service within Azure to this server ⓘ

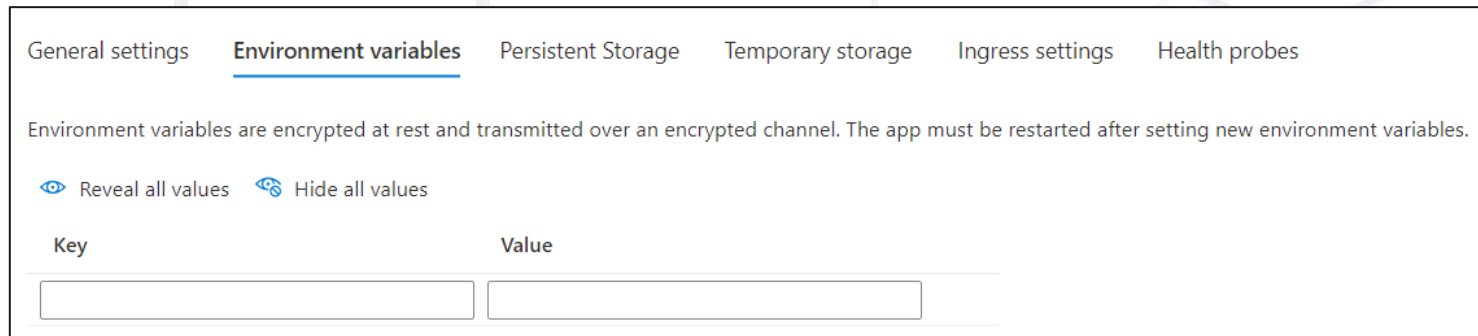
+ Add current client IP address ( [REDACTED] ) + Add 0.0.0.0 - 255.255.255.255

# Configure Environment variables

- Go inside your App (your **Group Resources** -> {service} -> Apps) and click **Configuration**



- **Environment variables**

A screenshot of the 'Environment variables' configuration page. The page has a tabbed interface with tabs for 'General settings', 'Environment variables' (which is selected), 'Persistent Storage', 'Temporary storage', 'Ingress settings', and 'Health probes'. Below the tabs, there is a text block stating: 'Environment variables are encrypted at rest and transmitted over an encrypted channel. The app must be restarted after setting new environment variables.' Below this text are two toggle buttons: 'Reveal all values' (with an eye icon) and 'Hide all values' (with a lock icon). At the bottom, there is a table with two columns: 'Key' and 'Value'. The first row of the table has two empty input fields for the key and value respectively.

| Key                  | Value                |
|----------------------|----------------------|
| <input type="text"/> | <input type="text"/> |

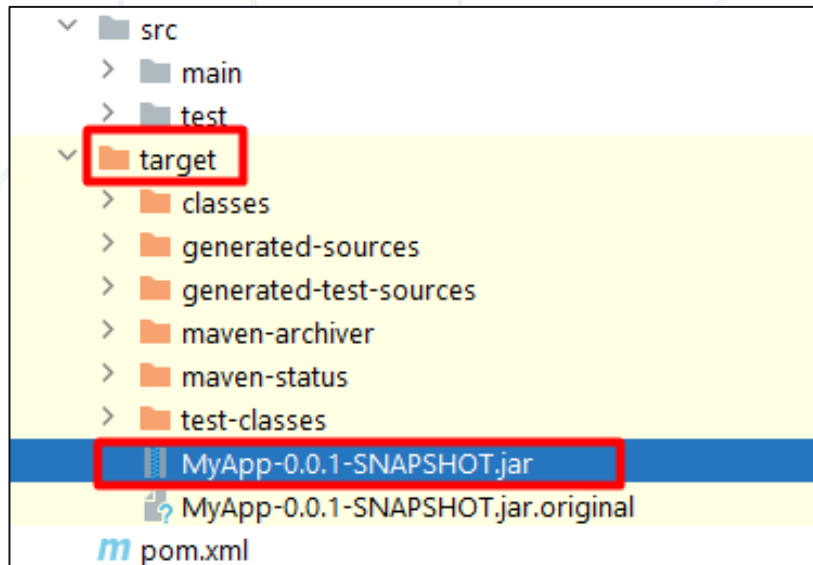
## application.yaml

```
datasource:  
  driverClassName: com.mysql.cj.jdbc.Driver  
  username: ${DB_USERNAME}  
  password: ${DB_PASSWORD}  
  url: jdbc:mysql://<db-name>.mysql.database.azure.com:3306/db?createDatabaseIfNotExist=true
```

Open terminal in the root folder of the app and run:

```
mvn clean package -DskipTests
```

Then go to target folder and copy the path to the .jar file



- Install Azure CLI (<https://learn.microsoft.com/en-us/cli/azure/install-azure-cli>)
  - Deploy the code to Azure using Azure CLI

```
az spring app deploy  
  --resource-group <name-of-resource-group>  
  --service <service-instance-name>  
  --name <app-name>  
  --artifact-path target/<app-name>-0.0.1-SNAPSHOT.jar
```





# Actuator

Monitor and manage your application

# Actuator



- Spring Boot includes a number of **additional features** to help you **monitor** and **manage** your application when you push it to production
- You can choose to manage and monitor your application by using **HTTP endpoints** or with **JMX**
- Auditing, health, and metrics gathering can also be **automatically applied** to your application

- The recommended way to enable the features is to add a dependency on the spring-boot-starter-actuator 'Starter'.

```
<dependencies>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-actuator</artifactId>  
  </dependency>  
</dependencies>
```

# Actuator Endpoints

- Endpoints let you monitor and interact with your application
- Spring Boot includes a number of **built-in endpoints** and lets you add your own
- Each individual endpoint can be enabled or disabled and exposed



- For example, by default, the health endpoint is mapped to **/actuator**

← → ↻ ⓘ localhost:8080/actuator

```
{"_links":{"self":{"href":"http://localhost:8080/actuator","templated":false},"health":{"href":"http://localhost:8080/actuator/health","templated":false},"health-path":{"href":"http://localhost:8080/actuator/health/{*path}","templated":true},"info":{"href":"http://localhost:8080/actuator/info","templated":false}}}
```



# Expose all actuator endpoints

- To expose **all** actuator **endpoints** you need to add in application.properties file:

**management.endpoints.web.exposure.include=\***

```
localhost:8080/actuator
{"_links":{"self":{"href":"http://localhost:8080/actuator","templated":false},"beans":
{"href":"http://localhost:8080/actuator/beans","templated":false},"caches-cache":
{"href":"http://localhost:8080/actuator/caches/{cache}","templated":true},"caches":
{"href":"http://localhost:8080/actuator/caches","templated":false},"health":
{"href":"http://localhost:8080/actuator/health","templated":false},"health-path":
{"href":"http://localhost:8080/actuator/health/{*path}","templated":true},"info":
{"href":"http://localhost:8080/actuator/info","templated":false},"conditions":
{"href":"http://localhost:8080/actuator/conditions","templated":false},"configprops":
{"href":"http://localhost:8080/actuator/configprops","templated":false},"env":
{"href":"http://localhost:8080/actuator/env","templated":false},"env-toMatch":
{"href":"http://localhost:8080/actuator/env/{toMatch}","templated":true},"loggers":
{"href":"http://localhost:8080/actuator/loggers","templated":false},"loggers-name":
{"href":"http://localhost:8080/actuator/loggers/{name}","templated":true},"heapdump":
{"href":"http://localhost:8080/actuator/heapdump","templated":false},"threaddump":
{"href":"http://localhost:8080/actuator/threaddump","templated":false},"metrics-requiredMetricName":
{"href":"http://localhost:8080/actuator/metrics/{requiredMetricName}","templated":true},"metrics":
{"href":"http://localhost:8080/actuator/metrics","templated":false},"scheduledtasks":
{"href":"http://localhost:8080/actuator/scheduledtasks","templated":false},"mappings":
{"href":"http://localhost:8080/actuator/mappings","templated":false}}}
```

- If you prefer all endpoints to be disabled
  - Set the **management.endpoints.enabled-by-default = false**
- Use individual endpoint enabled properties
  - On example, enable info endpoint

```
management.endpoints.enabled-by-default=false  
management.endpoint.info.enabled=true
```

- You should take care to **secure** HTTP **endpoints** in the same way that you would any other sensitive URL
- If Spring Security is present, endpoints are **secured by default**
- Example of **custom** security **configuration** for HTTP endpoints

```
@Configuration()  
public class ActuatorSecurity extends WebSecurityConfigurerAdapter {  
    @Override  
    protected void configure(HttpSecurity http) throws Exception {  
        http.requestMatcher(EndpointRequest.toAnyEndpoint()).authorizeRequests((requests) ->  
            requests.anyRequest().hasRole("ROLE_ADMIN"));  
        ...  
    }  
}
```



# Implementing Custom Endpoints (1)

- If you add a **@Bean** annotated with **@Endpoint**, any methods annotated with **@ReadOperation**, **@WriteOperation**, or **@DeleteOperation** are automatically exposed over JMX and, in a web application, over HTTP

```
@Component
@Endpoint(enableByDefault = true, id="custom")
public class CustomEndpoint {
    @ReadOperation
    public String getMyEndpoint(){
        return "My custom endpoint";
    }
}
```

# Implementing Custom Endpoints (2)

- If we want we can create Endpoints with **@RestControllerEndpoint** annotation

```
@Component
@RestControllerEndpoint( id="myRestEndpoint" )
public class MyRestEndpoint {

    @GetMapping("/test")
    @ResponseBody
    public String test(){
        return "My custom rest endpoint";
    }
}
```

# Customizing properties

- Customizing the Management Endpoint Paths
  - management.**endpoints.web.base-path**=/manage
- Customizing the Management Server Port
  - management.**server.port**=8081
- Disabling HTTP Endpoints
  - management.**server.port**=-1



- Using the Spring Boot Actuator give us a lot of **information** our application, but it's **not** very **user-friendly**
- Can be integrated with **Spring Boot Admin** for visualization, but it has it's limitations and it's less popular
- Tools like **Prometheus** and **Grafana** are more commonly used for the monitoring and visualization and are language/framework-independent
  - These tools have their own set of data formats and converting the metrics data



**Micrometer**

# Micrometer

- Solves the problem of being a **vendor-neutral data** provider
- Automatically **exposes** /actuator/metrics **data** into something your monitoring system can **understand**
- You need to include a vendor-specific micrometer dependency



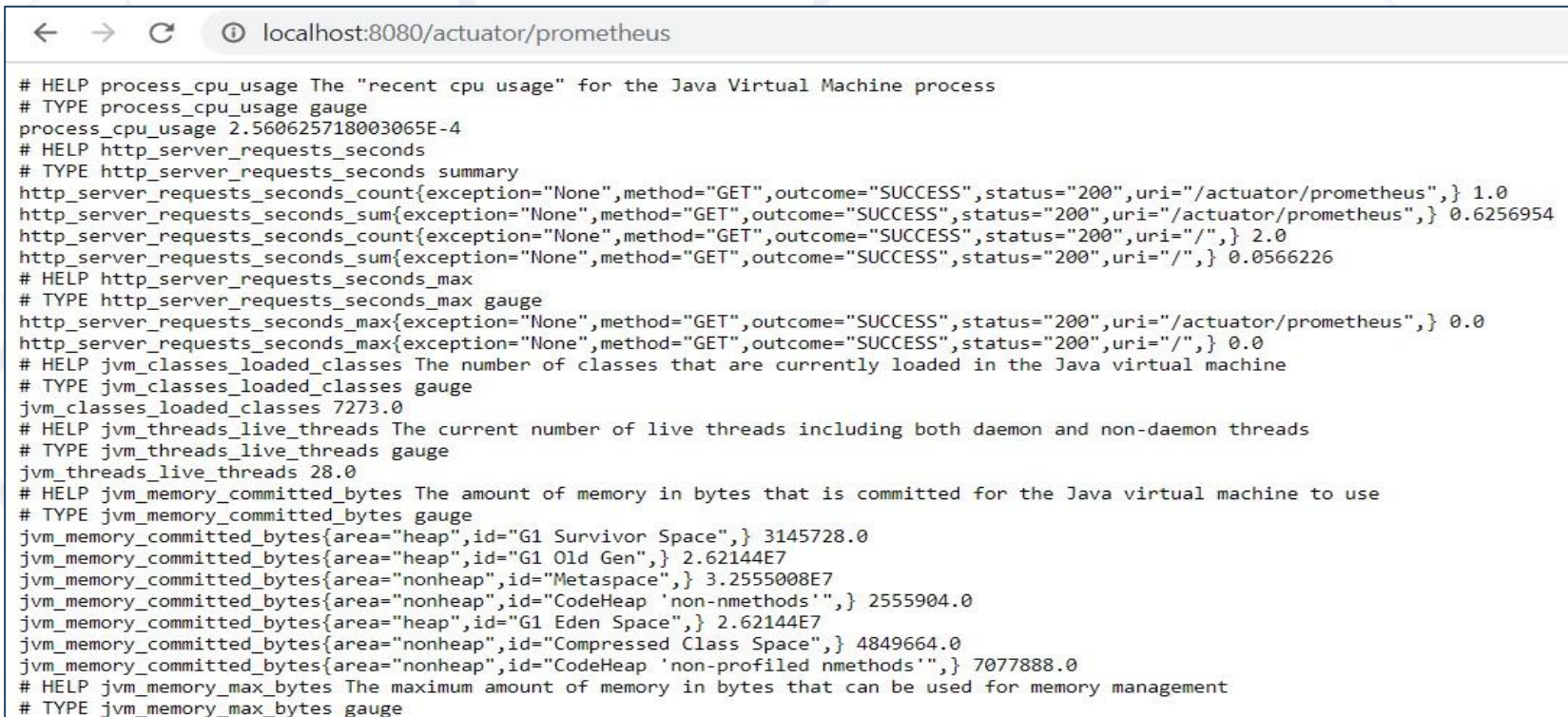
- Micrometer is a separate open-sourced project and is not in the Spring ecosystem, so we have to explicitly add it as a dependency
- If using Prometheus, add its **specific** dependency

```
<dependency>  
  <groupId>io.micrometer</groupId>  
  <artifactId>micrometer-registry-prometheus</artifactId>  
</dependency>
```



# Micrometer Example

- After adding the micrometer dependency, we have a **new endpoint** - /actuator/prometheus
- The data is formatted in **specific** for Prometheus **format**



```
localhost:8080/actuator/prometheus

# HELP process_cpu_usage The "recent cpu usage" for the Java Virtual Machine process
# TYPE process_cpu_usage gauge
process_cpu_usage 2.5606257180033065E-4
# HELP http_server_requests_seconds summary
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",} 1.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",} 0.6256954
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",} 2.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",} 0.0566226
# HELP http_server_requests_seconds_max
# TYPE http_server_requests_seconds_max gauge
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",} 0.0
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/",} 0.0
# HELP jvm_classes_loaded_classes The number of classes that are currently loaded in the Java virtual machine
# TYPE jvm_classes_loaded_classes gauge
jvm_classes_loaded_classes 7273.0
# HELP jvm_threads_live_threads The current number of live threads including both daemon and non-daemon threads
# TYPE jvm_threads_live_threads gauge
jvm_threads_live_threads 28.0
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="heap",id="G1 Survivor Space",} 3145728.0
jvm_memory_committed_bytes{area="heap",id="G1 Old Gen",} 2.62144E7
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 3.2555008E7
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'non-nmethods'",} 2555904.0
jvm_memory_committed_bytes{area="heap",id="G1 Eden Space",} 2.62144E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 4849664.0
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'non-profiled nmethods'",} 7077888.0
# HELP jvm_memory_max_bytes The maximum amount of memory in bytes that can be used for memory management
# TYPE jvm_memory_max_bytes gauge
```





# Prometheus

# Prometheus



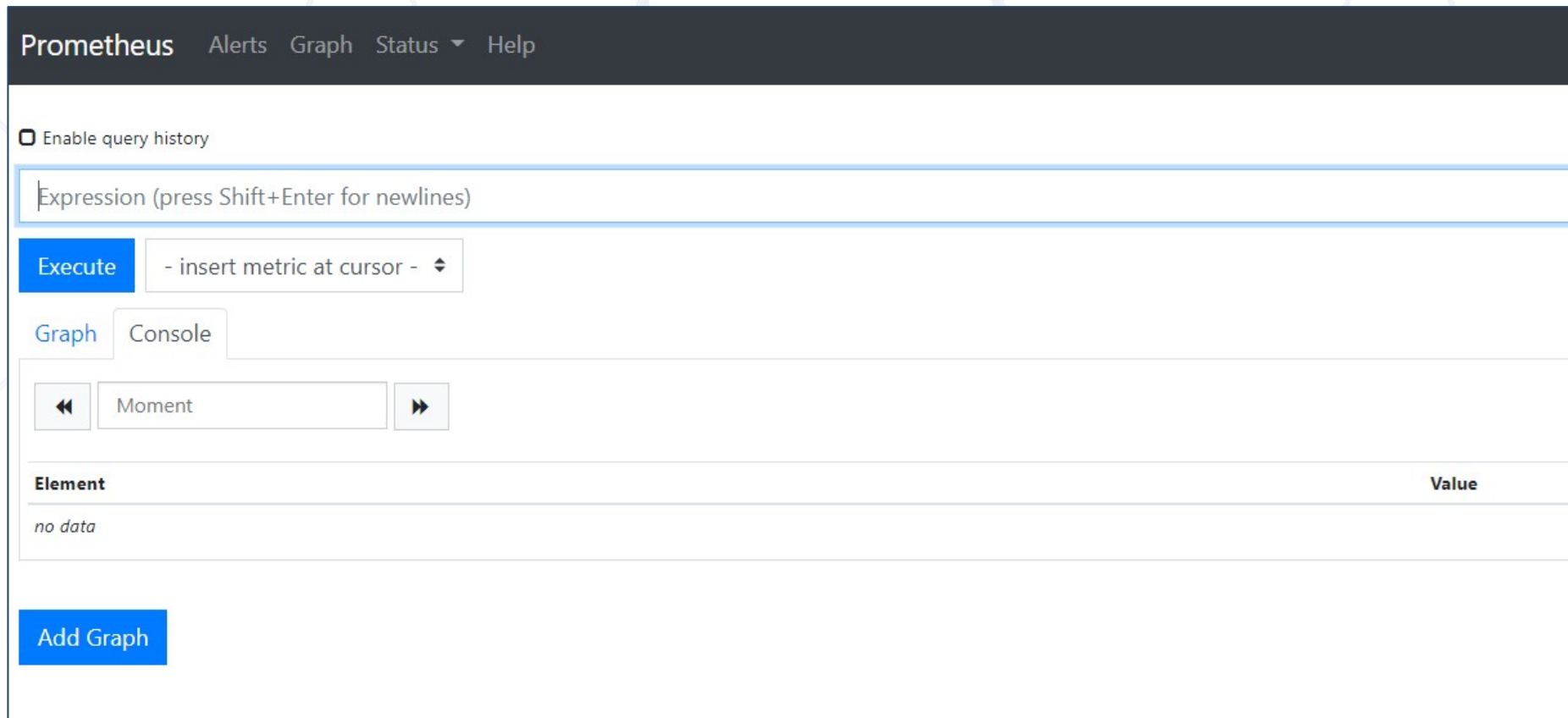
- Time-series database that **stores** the **metric** data by pulling it (using a built-in data scraper) **periodically** over HTTP
- Intervals between pulls can be configured
- Has a simple user interface where we can **visualize/query** on all of the **collected metrics**
- To configure Prometheus more precisely we using the **prometheus.yaml** file

# Download and Configure Prometheus

- You can download Prometheus from [here](#)
- Configure Prometheus with **prometheus.yaml** file

```
global:
  scrape_interval: 15s # By default, scrape targets every 15 seconds.
  # A scrape configuration containing exactly one endpoint to scrape:
  # Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from
  # this config.
  - job_name: 'prometheus'
    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
```

- After starting Prometheus, we can access it on <http://localhost:9090>



The screenshot shows the Prometheus web interface. At the top is a dark navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. Below this is a white area with a checkbox for 'Enable query history'. A large text input field is labeled 'Expression (press Shift+Enter for newlines)'. Below the input is a blue 'Execute' button and a dropdown menu showing '- insert metric at cursor -'. There are two tabs, 'Graph' and 'Console', with 'Graph' selected. Below the tabs are navigation controls: a left arrow, a text box containing 'Moment', and a right arrow. At the bottom left is a blue 'Add Graph' button. The main content area is a table with two columns: 'Element' and 'Value'. The table currently contains one row with the text 'no data' in the 'Element' column.

| Element | Value |
|---------|-------|
| no data |       |

- Prometheus provides a functional query language called **PromQL** (Prometheus Query Language)
- Let's the user **select** and **aggregate** time series data in real time
- Result of an expression can either be shown as a **graph**, viewed as **tabular data** in Prometheus' expression browser, or consumed by external systems via the **HTTP API**

- Return all time series with the metric `http_requests_total` and the **given job** and handler labels

```
http_requests_total{job="apiserver", handler="/api/comments"}
```

- Return a whole **range of time** for the same vector

```
http_requests_total{job="apiserver", handler="/api/comments"}[5m]
```

- Using **regular expressions**

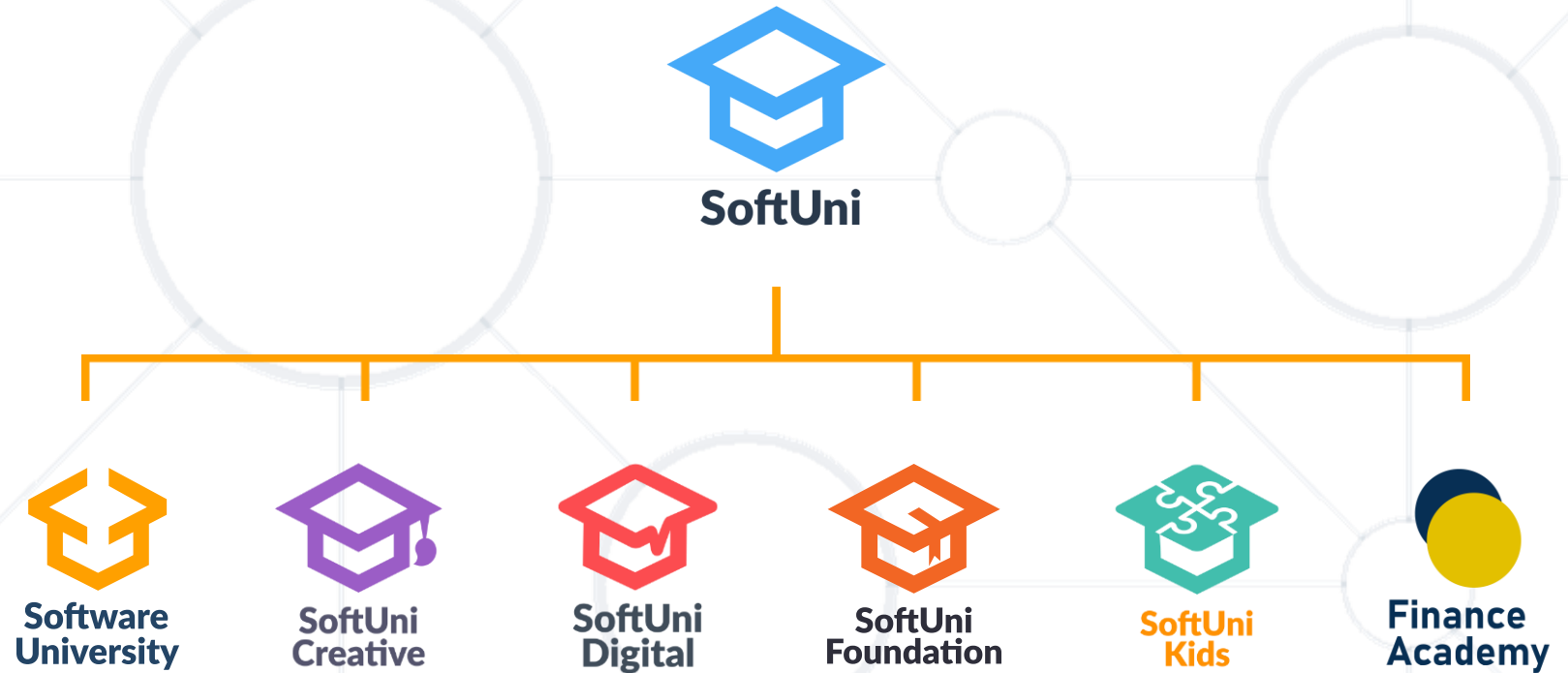
```
http_requests_total{job=~".*server"}
```

```
http_requests_total{status!~"4.."}
```

- **Deployment** means to push changes or update from one environment to another
- **Micrometer** solves the problem of being a **vendor-neutral data** provider
- **Prometheus** is a Time-series database that **stores the metric data** by pulling it (using a built-in data scraper) **periodically** over HTTP



# Questions?





# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**

 **Flutter**<sup>TM</sup>  
International

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**BOSCH**

 **Postbank**  
*Решения за твоето утре*

 **PHAR  
VISION**



**SmartIT**

**DXC**  
TECHNOLOGY

**createX**

- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Software University Forums
  - [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

