

Deployment: Spring Boot application with Postgresql database

1 Configuring Java + Maven + Spring project for deployment

Add PostgreSQL driver as dependency in your project's **pom.xml**. Copy and paste the below snippet, place it as a child element of **<dependencies>** element.

```
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <scope>runtime</scope>
</dependency>
```

Create a file named **Dockerfile** (no file extension) in your coding project's root. Content of the **Dockerfile** for a **Java + Maven** project is (use Copy + Paste):

```
FROM eclipse-temurin:17-jdk-focal as builder
WORKDIR /opt/app
COPY .mvn/ .mvn
COPY mvnw pom.xml ./
RUN chmod +x ./mvnw
RUN ./mvnw dependency:go-offline
COPY ./src ./src
RUN ./mvnw clean install -DskipTests
RUN find ./target -type f -name '*.jar' -exec cp {} /opt/app/app.jar \; -quit
FROM eclipse-temurin:17-jre-alpine
COPY --from=builder /opt/app/*.jar /opt/app/
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/opt/app/app.jar" ]
```

The above content is a generalized **Dockerfile** configuration and can be used for other Spring Boot projects.

Create a new deployment profile for your application. You need to create a new file in the **<your app>/src/main/resources/** directory.

Name the new file **application-rahti.properties**.

Content of the **application-rahti.properties** is (use Copy + Paste):

```
spring.datasource.url=jdbc:postgresql://${POSTGRESQL_SERVICE_HOST}:${POSTGRESQL_SERVICE_PORT}/${DB_NAME}
spring.datasource.username=${DB_USER}
spring.datasource.password=${DB_PASSWORD}
spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
```

Commit the above changes and push them to your GitHub repository.

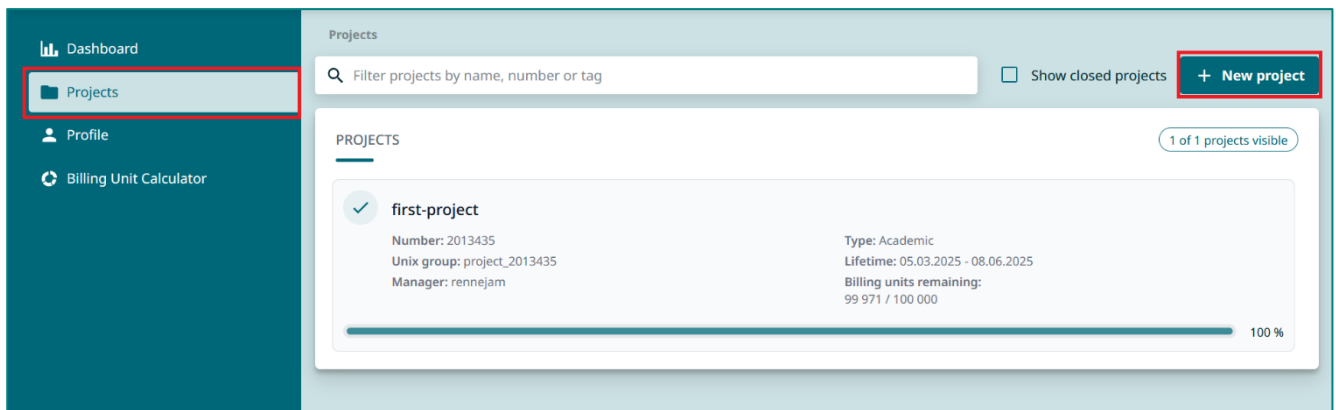
2 Creating a project in CSC

NOTE! If you already have created a project in CSC, you are unable to create another one. In that case proceed to chapter 3 of these instructions.

CREATE A NEW PROJECT

Project creation not allowed
Students may have only one project at a time but they can be a member of several CSC projects.
[Read more here](#)

Login to CSC at <https://my.csc.fi/login> using you Haka user account (Haaga-Helia credentials) or your CSC user account (username sent to your email). Navigate to Projects view and start creating a new project.

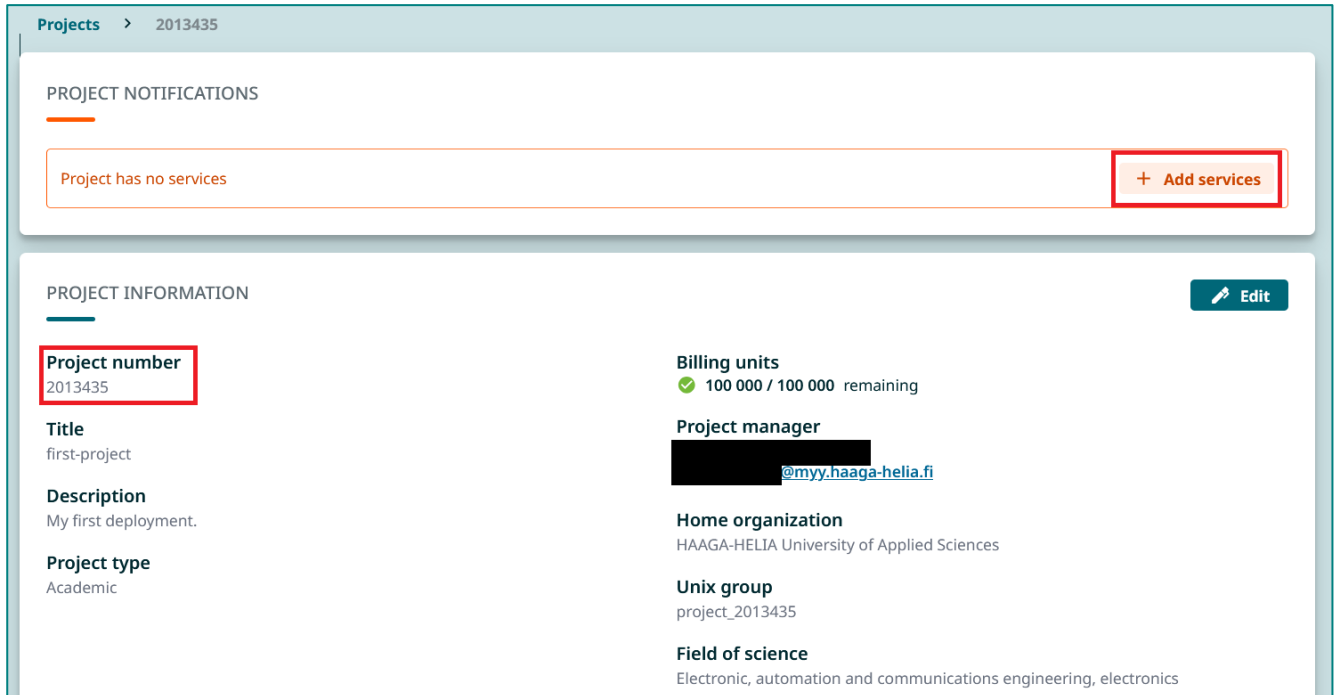


The screenshot shows the CSC Projects dashboard. On the left is a dark teal sidebar with navigation links: Dashboard, Projects (highlighted with a red box), Profile, and Billing Unit Calculator. The main content area is light blue and titled 'Projects'. It features a search bar 'Filter projects by name, number or tag', a checkbox 'Show closed projects', and a red-outlined button '+ New project'. Below this, a project card for 'first-project' is displayed with details: Number: 2013435, Unix group: project_2013435, Manager: rennejam, Type: Academic, Lifetime: 05.03.2025 - 08.06.2025, and Billing units remaining: 99 971 / 100 000. A progress bar at the bottom of the card is at 100%.

Fill in your project's information as instructed in the form. Read and accept all terms of use and privacy notice. Choose **Create project** to continue.

You have now created your CSC project. Within this project you can start adding CSC's services. For your deployment you only need **Rahti – Container Cloud**. Click **Add services** and select **Rahti**.

NOTE: check your project number. You will need it later!



The screenshot shows the CSC project management interface for project 2013435. The interface is divided into two main sections: PROJECT NOTIFICATIONS and PROJECT INFORMATION.

PROJECT NOTIFICATIONS

- A message states: "Project has no services".
- A red button labeled "+ Add services" is visible.

PROJECT INFORMATION

Project number: 2013435 (highlighted with a red box)

Title: first-project

Description: My first deployment.

Project type: Academic

Billing units: 100 000 / 100 000 remaining

Project manager: [redacted]@myy.haaga-helia.fi

Home organization: HAAGA-HELIA University of Applied Sciences

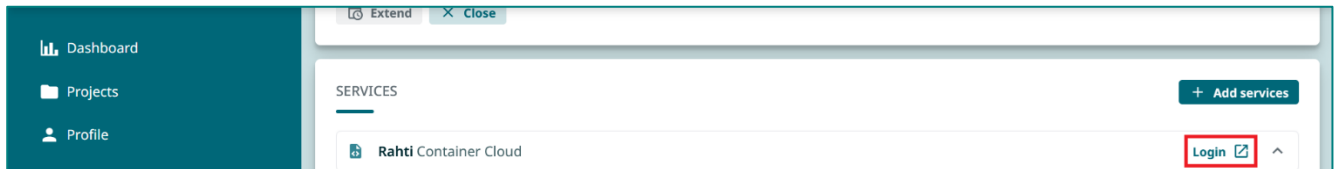
Unix group: project_2013435

Field of science: Electronic, automation and communications engineering, electronics

Once you have added the **Rahti** service you might have to wait up to 60 mins for you to gain access after activation. So be patient!

3 Creating a project in Rahti

After you have gained access to **Rahti** service, you can login to **Rahti** at your CSC project's **Projects** view. Choose your (only) project and scroll down to **Services**. Clicking **Login** will take you to **Rahti** landing page.

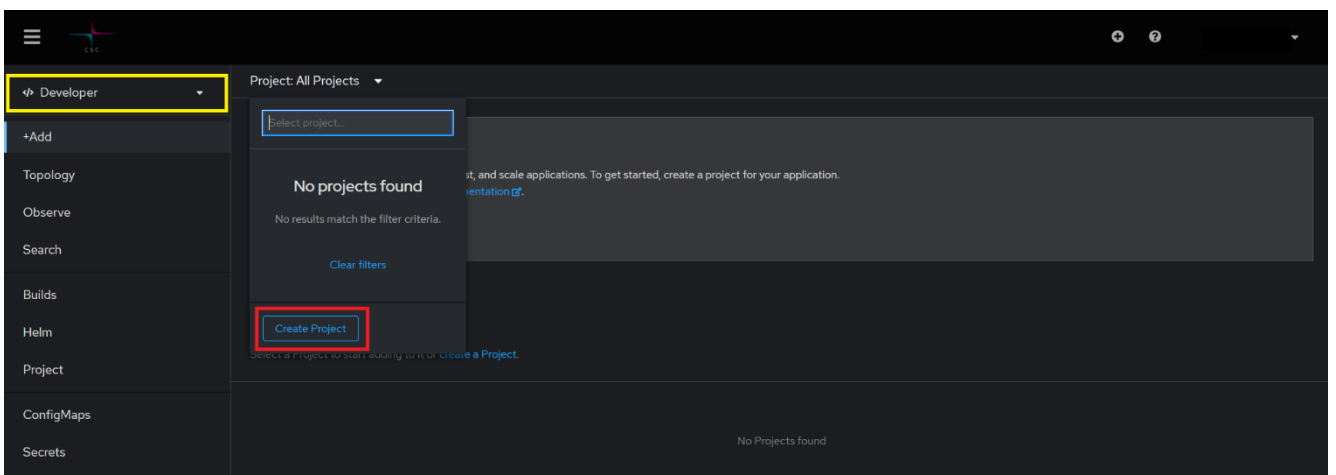


Proceed the login process, you need to click **Login** buttons a few times in different pages. Use either Haka or CSC as your authentication method when prompted.

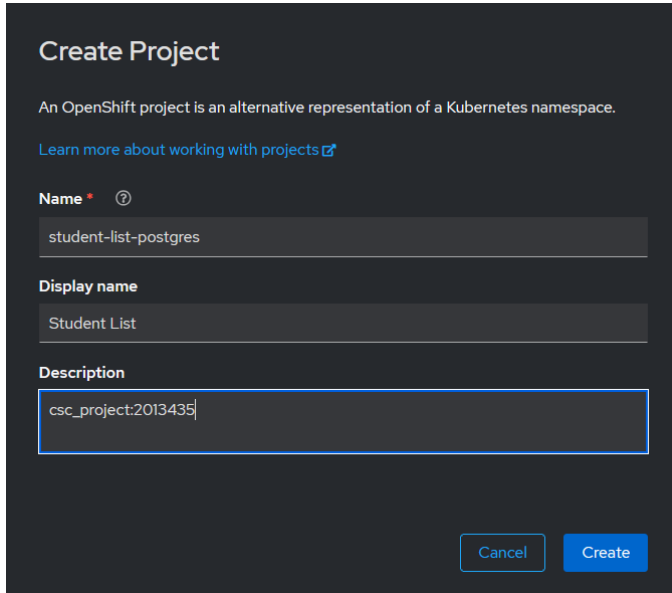
Successfully logging in directs you to **Rahti** console. When starting the console for the first time take the site's tour showing important navigation options. After the tour you are ready to create your **Rahti project**.

Click the **Project: All projects** dropdown menu and choose **Create Project** or click the text **Create a Project**. If you don't have these options visible in your page, make sure you are in **Developer** view (marked with yellow).

NOTE! You can have more than one **Rahti projects** in one **CSC Project**.



Give your project a name and a display name. In the description write **csc_project:<your CSC project number>**. You will find your **CSC project number** in your CSC Project's **Project information** view. Proceed by clicking **Create**.



Create Project

An OpenShift project is an alternative representation of a Kubernetes namespace.

[Learn more about working with projects](#)

Name * ⓘ

student-list-postgres

Display name

Student List

Description

csc_project:2013435

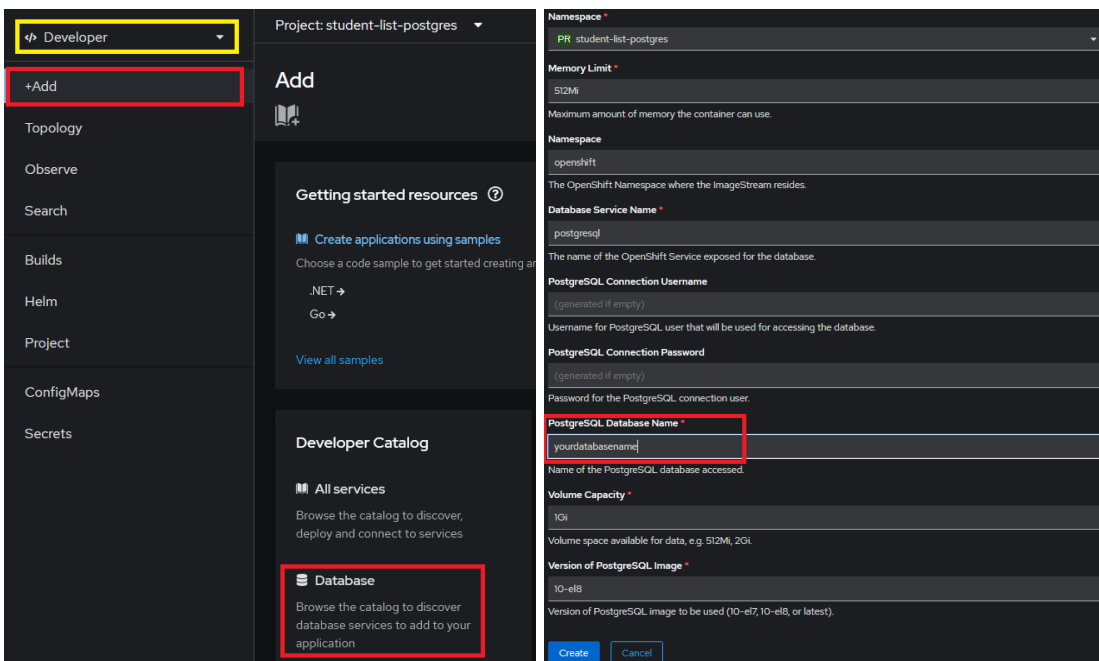
4 Creating a PostgreSQL database instance

After creating a project in Rahti it's time to create a new database instance for your application.

In **Developer** view, go to **+Add** page and start creating a database for your project. Choose **PostgreSQL**. Make sure you **don't select the Ephemeral** version! Click **Instantiate Template** to continue the process.

You can leave all the default values in the form where you create your database. Leaving **PostgreSQL Connection Username** and **Password** empty will make the system generate random credentials for you. This is fine within the scope of Haaga-Helia UAS course work.

If you want to specify **PostgreSQL Connection Username** and **Password**, **don't use weak a password! Always use strong passwords even when practicing!**

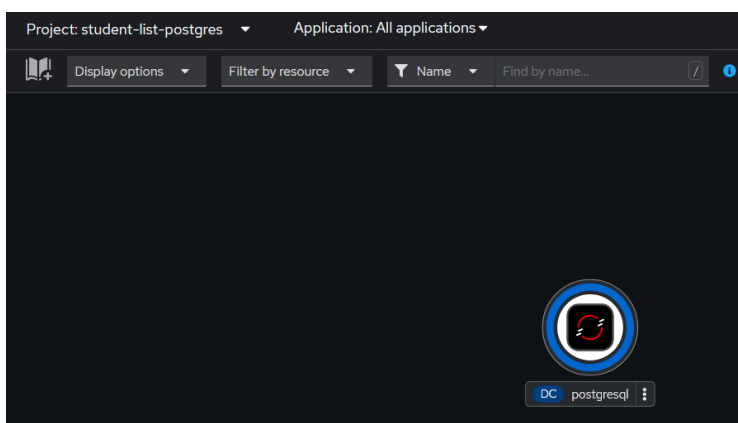


The screenshot shows the Rahti Developer Catalog interface. On the left, the 'Developer' view is selected, and the '+Add' button is highlighted. The main area shows the 'Add' page for the 'student-list-postgres' project. The 'Getting started resources' section is visible. The 'Developer Catalog' section shows 'All services' and 'Database' services. The 'Database' service is selected, and the 'PostgreSQL Database Name' field is highlighted. The form on the right contains the following fields:

- Namespace**: PR student-list-postgres
- Memory Limit**: 512Mi
- Maximum amount of memory the container can use**: 512Mi
- Namespace**: openshift
- Database Service Name**: postgresql
- PostgreSQL Connection Username**: (generated if empty)
- PostgreSQL Connection Password**: (generated if empty)
- PostgreSQL Database Name**: yourdatabasename
- Volume Capacity**: 1Gi
- Version of PostgreSQL Image**: 10-el8

The 'Create' button is visible at the bottom right of the form.

Successful creation of a database will show as a new **DeploymentConfig** object in your **Topology** view.

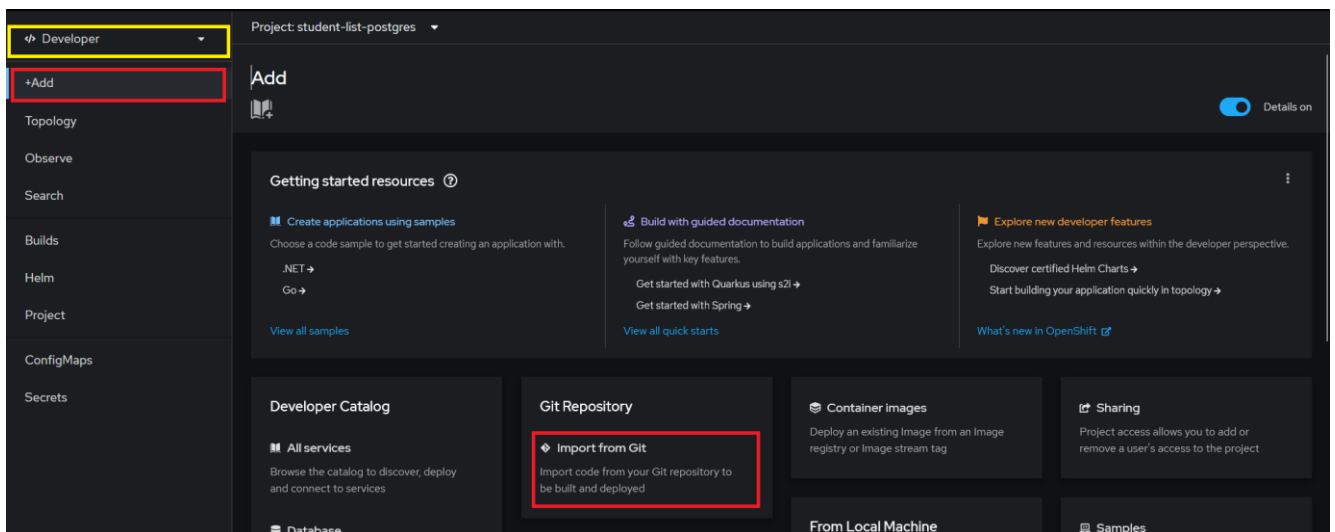


The next step is to include your SpringBoot application into your Rahti project.

5 Including SpringBoot application into Rahti

You will need to have your SpringBoot application in GitHub before continuing these instructions. The following instructions assume that you have set your GitHub repository **Public**.

In **Rahti** dashboard make sure you are in **Developer** view: go to **+Add** page and choose **Import from Git**.



Copy and paste your GitHub repository's URL into the corresponding form field and choose **Dockerfile** as your **Import Strategy** if it is not the suggested one. You can leave rest of the form fields to their default values. Move on to **Create**.


Git

Git Repo URL *

Validated

> Show advanced Git options

Multiple import strategies detected
The Dockerfile at Dockerfile is recommended.

 Dockerfile [Edit Import Strategy](#)

General

Application

Select an Application to group this component.

Name *

A unique name given to the component that will be used to name associated resources.

[Create](#) [Cancel](#)

NOTE! Rahti might not detect the **Dockerfile** if it is not in your GitHub project's root. In this case open **Show advanced Git options** under **Git Repo URL** form field. In **Context dir** form field update the value to the name of the subdirectory that contains **pom.xml** and **Dockerfile**.


Context dir

Optional subdirectory for the source code, used as a context directory for build.

Successful creation will take you to your project's **Topology** view. Click the graphical representation of your deployment to open your deployment controls.

Project: student-list-postgres Application: All applications [View shortcuts](#)

Display options Filter by resource Name Find by name...



rahti2-test-postgres Actions

Health checks
Container rahti2-test-postgres does not have health checks to ensure your application is running correctly. [Add health checks](#)

Details Resources Observe

Pods

Waiting for the build
Waiting for the first build to run successfully. You may temporarily see "ImagePullBackOff" and "ErrImagePull" errors while waiting.
[Show waiting pods with errors](#)

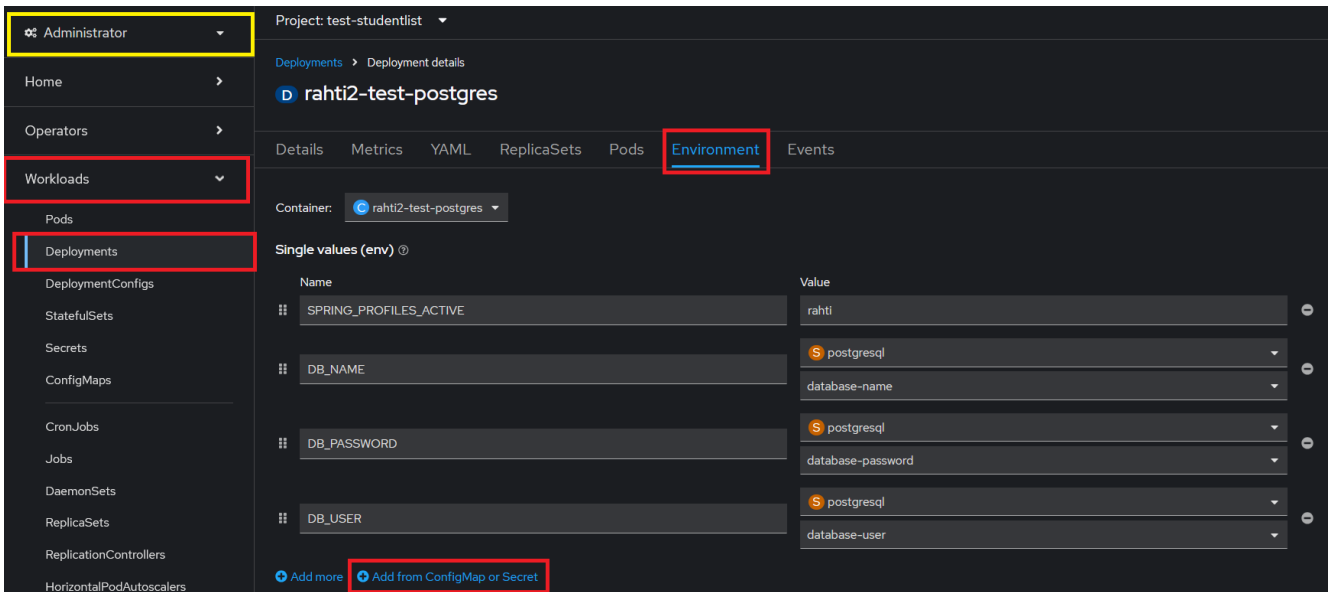
No Pods found for this resource.

You have now deployed your application into **Rahti**. But it is not in working condition yet. You still need to configure the environment variables for the JDBC connection.

6 Configuring environment variables

In **Administrator** view: **workloads** → **Deployments** → <your deployment> → **Environment**.

Set your environment variables as shown below. Click the circled **Add from ConfigMap or Secret** to get more form fields. Click **Save** at the bottom of the page to activate your changes.



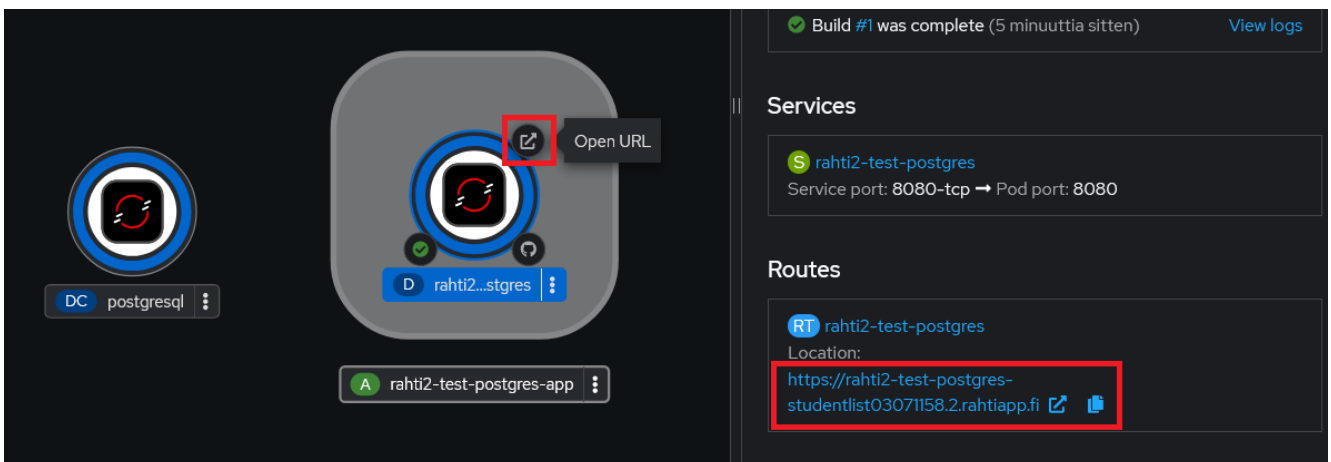
The screenshot shows the Kubernetes Administrator interface. On the left sidebar, the **Workloads** menu is expanded, and **Deployments** is selected. The main panel shows the **Environment** tab for the deployment **rahti2-test-postgres**. Under **Single values (env)**, there are four environment variables defined:

Name	Value
SPRING_PROFILES_ACTIVE	rahti
DB_NAME	postgresql
DB_PASSWORD	database-name
DB_USER	database-password

At the bottom, there is a button labeled **Add from ConfigMap or Secret** which is circled in red.

Congratulations! You have now deployed your Spring application with a proper database to CSC/Rahti!

To get the URL for your app go back to **Developer** view → **Topology**. Click your application's **Open URL** shortcut or navigate to deployment's quick controls (navigation pane on the right) and scroll down to **Routes**. It might take few minutes before you are able to access your deployment's URL in browser.



The screenshot shows the Kubernetes Developer interface. On the left, the **Topology** view displays a diagram of the application architecture. A red box highlights the **Open URL** button next to the **rahti2-test-postgres-app** component. On the right, the **Routes** section shows the URL for the deployment:

RT rahti2-test-postgres
Location:
<https://rahti2-test-postgres-studentlist03071158.2.rahtiapp.fi>

The URL is circled in red.