By Mikael Sukoinen

PART OF TUTORIAL SERIES
Deploying Vaadin applications to the cloud

Deploying a Java web app to the Heroku cloud

This tutorial shows how to deploy a Java web application to the Heroku cloud. Heroku provides free cloud deployment for up to 5 applications. However, due to their automatic dyno restarts, this platform is best suited to quick demo uploads, rather than full production deployments of Vaadin applications. You can find more information about this on their website.

We use the latest Spring Boot starter app in our example. You can see the example app running on the Heroku cloud here.

You can deploy the application directly from your command line or from a GitHub repository.

PREREQUISITES:

- Create a Heroku account at https://www.heroku.com/home.
- Verify that you have Java installed by running the `java --version` command in your terminal. If not, download and install the latest version on your computer.
- Verify that you have Git installed by running the `git --version` command in your terminal. If not, download and install the latest version on your computer.

Deploying from the command line

Generating a JAR

A JAR (Java Archive) is a package file that merges Java class files and associated metadata and resources, such as text and images, into one distributable file. This is the default file format for your Vaadin app if you created it on https://vaadin.com/start.

To generate a `.jar` file from the downloaded project:

1. Download and open the starter project from http://vaadin.com/start/latest. Select **Spring Boot** as the technology stack and fill the **Maven Group ID** and **Project Name** as you see fit (or leave them at the defaults).

Choose Vaadin version

Vaadin 15 (Latest)

Download Online workspace Eclipse

Technology stack

- (•) Spring Boot
- ( ) CDI and Java EE
- ( ) Plain Java Servlet

Maven Group ID

com.packagename.myapp

Project Name

My Starter Project

DOWNLOAD CUSTOMIZE APP

2. Change the `server.port` in your Spring project `application.properties` file residing in `src/main/resources` to: `server.port=${port:8080}`. You can find more details about this in the Spring documentation.
3. Generate a `.jar` from your application using the `mvn package -Pproduction` maven goal.
4. Navigate to the folder containing the generated `.jar` (usually `/target`).

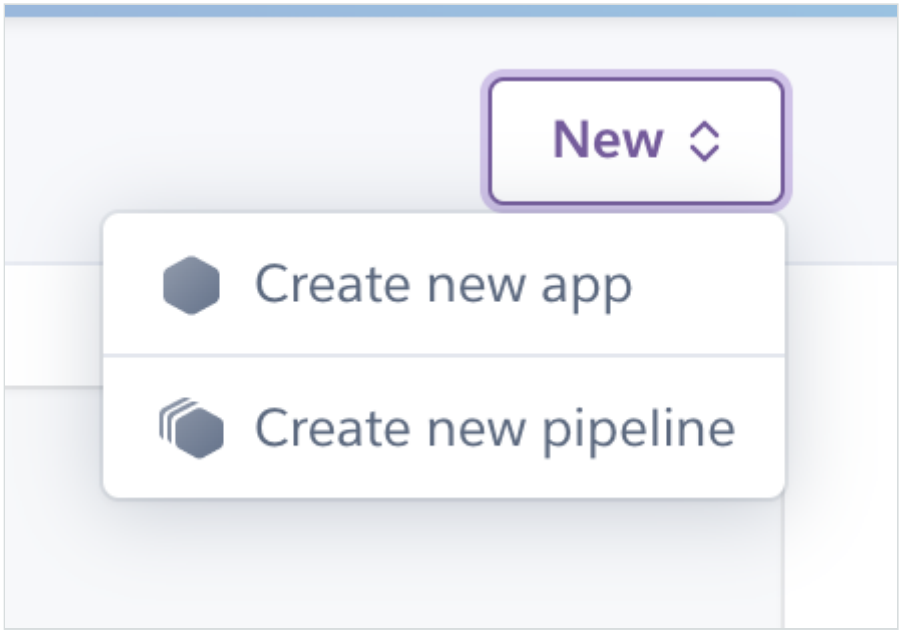
Creating and deploying a Heroku application

You can create and deploy a Heroku application in your Heroku dashboard or using the Java CLI plugin.

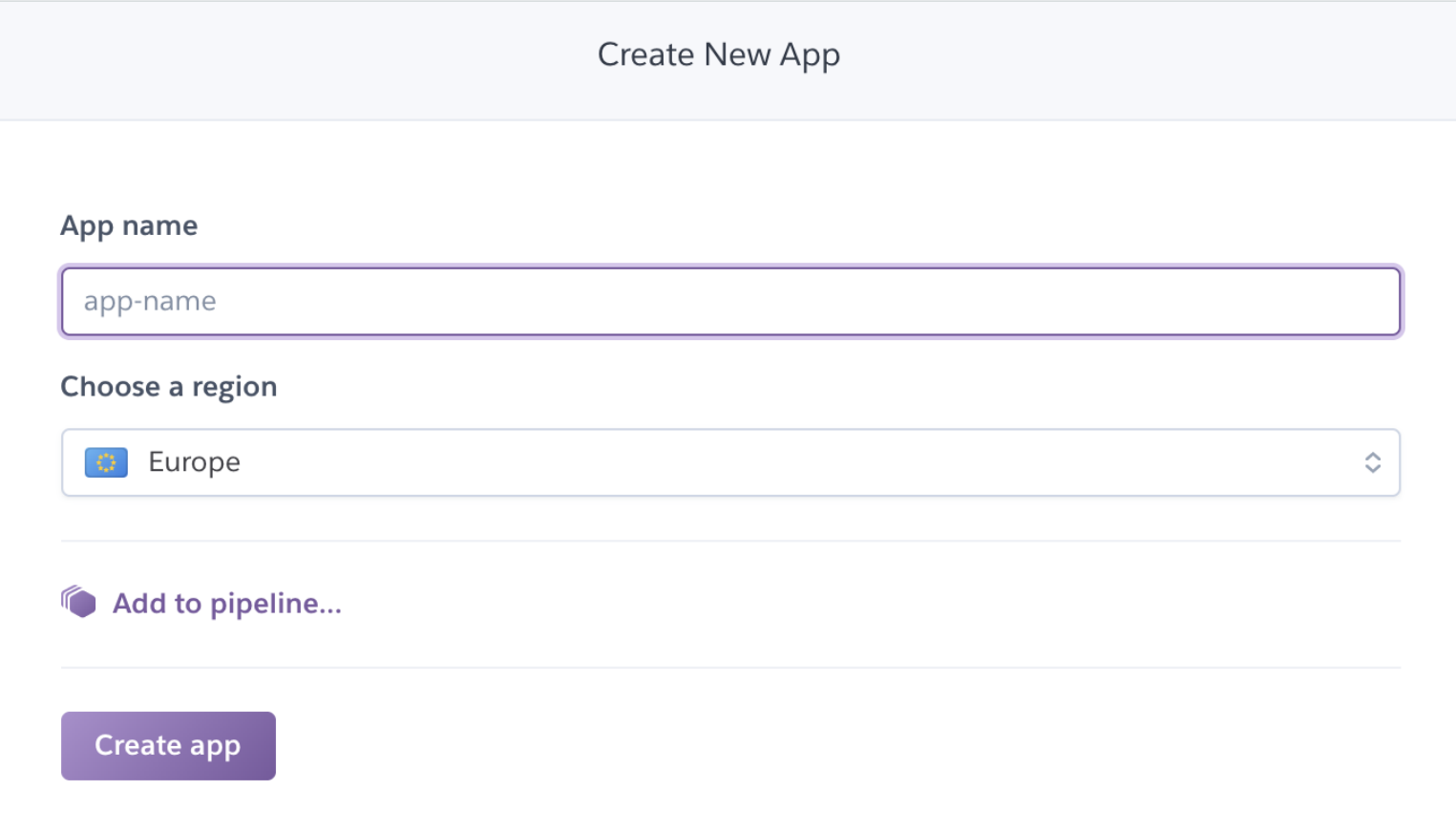
> **NOTE**
>
> Replace all instances of `APP_NAME` with the real name of your application.

To create and deploy an application in your Heroku dashboard:

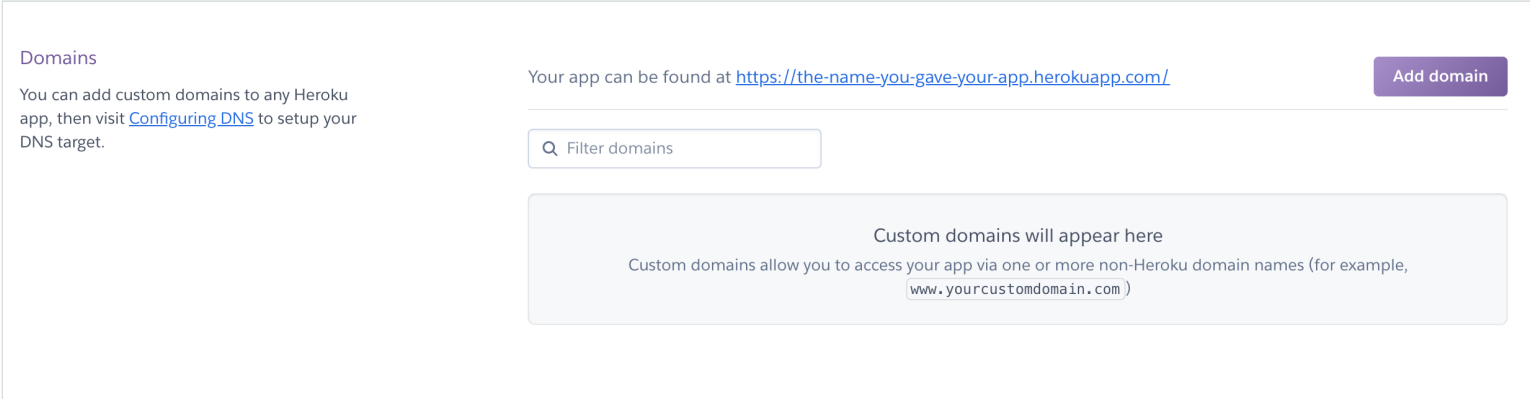
1. Go to your Heroku dashboard (https://dashboard.heroku.com/apps) and create a new app by selecting **New** > **Create new app**.



2. Name your Heroku application and choose a region. Then click on **Create app**.



3. Install the Heroku CLI and login to your Heroku dashboard `heroku login`.
4. Install the Java CLI plugin by running the `heroku plugins:install java` command in your terminal.
5. Deploy the JAR file using the `heroku deploy:jar my-app.jar --app APP_NAME` command. Replace `my-app.jar` with your actual filename.
6. Check the URL of the deployed app under **Domains**: https://dashboard.heroku.com/apps/APP_NAME/settings. The application should be running there.



To create and deploy an application using the Java CLI plugin:

1. Install the Heroku CLI and login to your Heroku dashboard `heroku login`.
2. Install the Java CLI plugin by running the `heroku plugins:install java` command in your terminal.
3. Run the 'heroku create APP_NAME` command.
4. Deploy the JAR file using the `heroku deploy:jar my-app.jar --app APP_NAME` command. Replace `my-app.jar` with your actual filename.
5. Open the app using the `heroku open --app APP_NAME` command.

> **TIP**
>
> You can open the application log by running the `heroku logs --tail --app APP_NAME` command in your terminal to troubleshoot any possible errors.

> **TIP**
>
> If your application is packaged as a WAR, deploy your app using the `heroku war:deploy myapp.war --app APP_NAME` command instead. You can find detailed instructions about this in the Heroku documentation.

Setting up a GitHub CI pipeline

You can deploy an application directly from a GitHub repository instead of uploading it as a JAR. However, this approach does require some tweaking of your project.

Configuring the project for GitHub

1. Start by creating a `heroku-settings.xml` file and add the following content. This is used to instruct Maven which profiles to enable by default.

heroku-settings.xml

COPY

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xs

    <!-- activate by setting the MAVEN_SETTINGS_PATH config var to heroku-settings.xml in Herok
    See https://devcenter.heroku.com/articles/using-a-custom-maven-settings-xml for more detail
     -->

    <activeProfiles>
        <activeProfile>production</activeProfile>
        <activeProfile>npm</activeProfile>
    </activeProfiles>
</settings>
```

2. Add the following section to the `<profiles>` section of your`pom.xml`. There are two relevant profiles: `npm` and `production`. `production` is included automatically in the project by default, but `npm` must be configured manually:

pom.xml COPY

```
<profile>
  <id>npm</id>
  <build>
      <plugins>
       <plugin>
          <groupId>com.github.eirslett</groupId>
          <artifactId>frontend-maven-plugin</artifactId>
          <!-- Use the latest released version:
          https://repo1.maven.org/maven2/com/github/eirslett/frontend-maven-plugin/ -->
          <version>1.9.1</version>
          <executions>
              <execution>
                  <id>install node and npm</id>
                  <goals>
                      <goal>install-node-and-npm</goal>
                  </goals>
                  <!-- optional: default phase is "generate-resources" -->
                  <phase>generate-resources</phase>
              </execution>
          </executions>
          <configuration>
              <nodeVersion>v12.13.0</nodeVersion>
          </configuration>
          </plugin>
      </plugins>
  </build>
</profile>
```

3. Create a new file `Procfile` (without a file extension) in the root directory of your application and add the following content. This file tells Heroku what to run on startup.

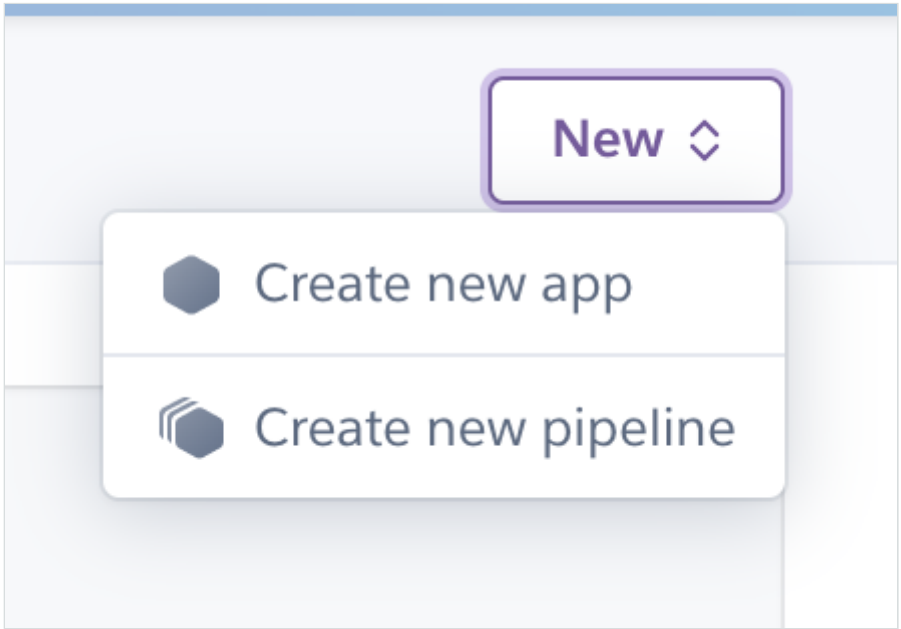
```
web: java -jar target/PROJECT_NAME-PROJECT_VERSION.jar $PORT
```

- Substitute your project name and details for the JAR name. In our case it is `starter_app-2.0-SNAPSHOT.jar`.
- This file must reside in the same folder as your `pom.xml`.

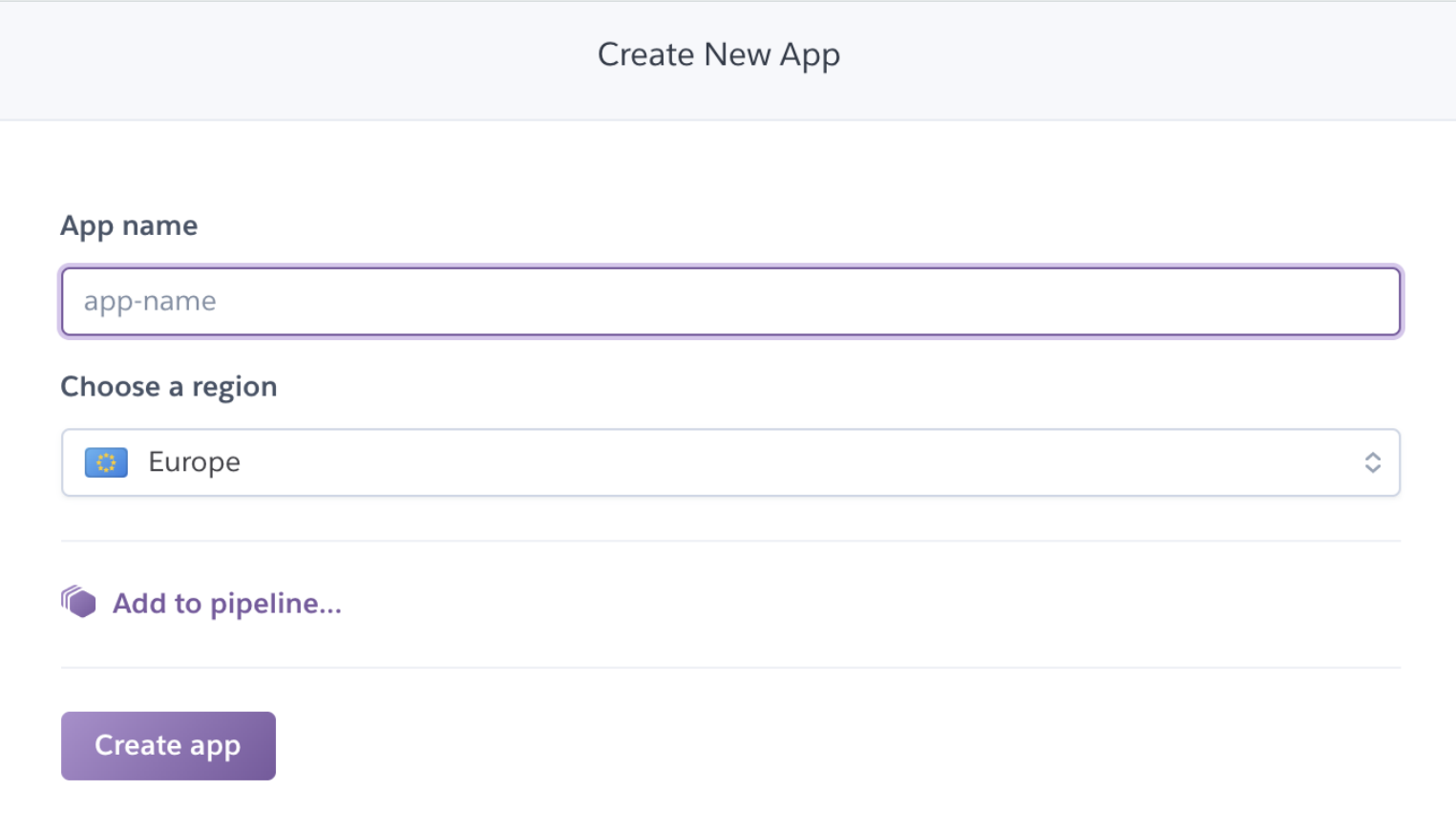
4. Push the code to your Github repository.

Deploying from GitHub

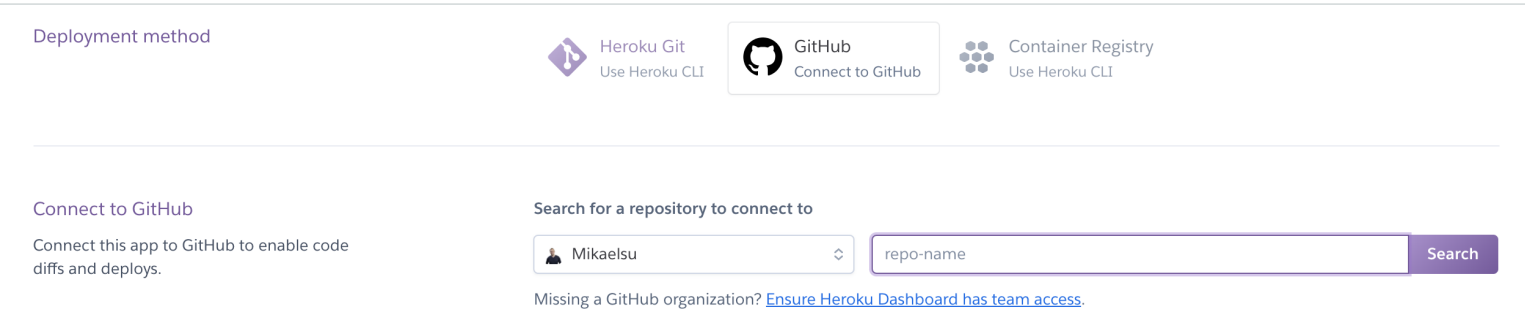
1. In your Heroku dashboard (https://dashboard.heroku.com/apps), create a new app by selecting **New** > **Create new app**.



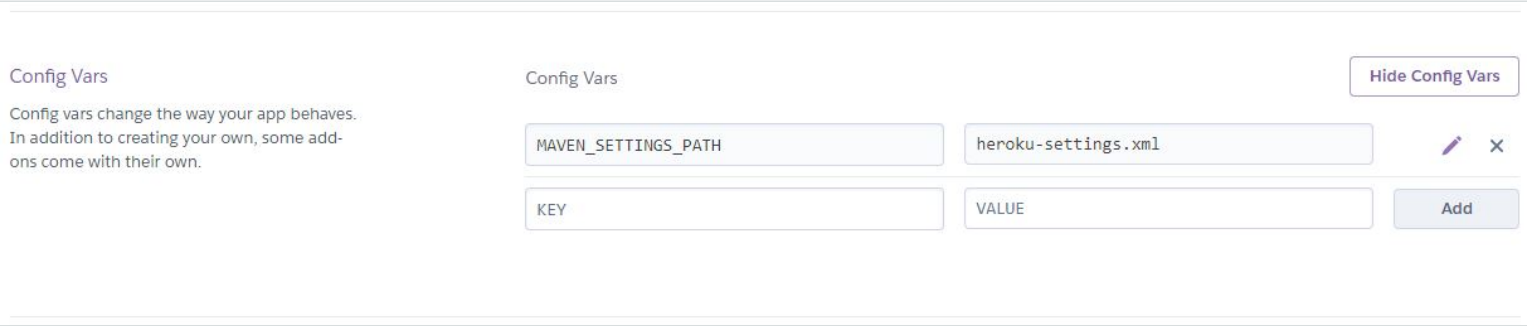
2. Name your Heroku application and choose a region. Then click on **Create app**.



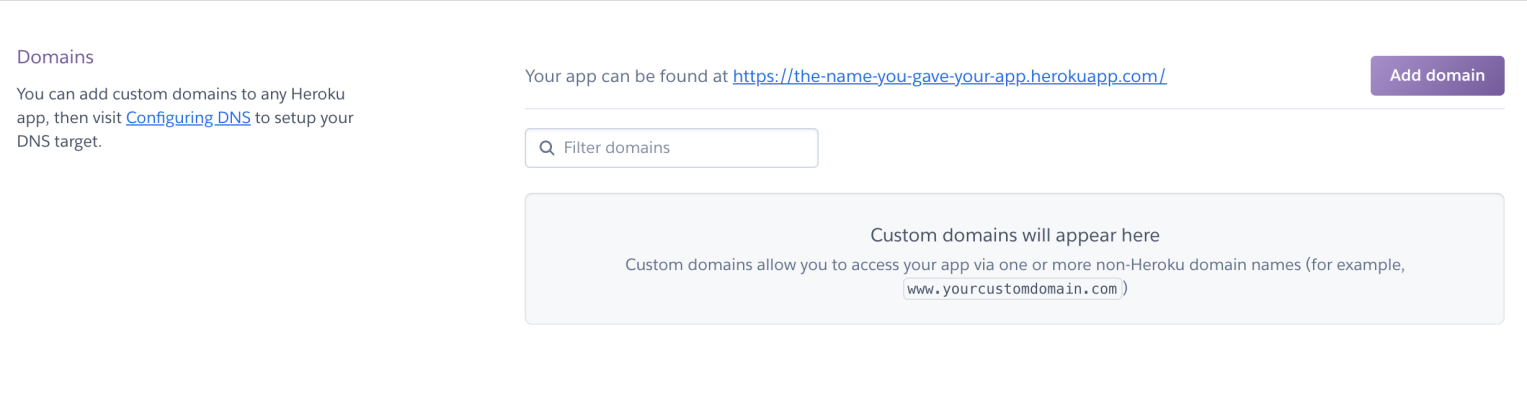
3. Connect to the GitHub repository to which you uploaded your application.



4. Set the `MAVEN_SETTINGS_PATH` configuration variable to `heroku-settings.xml` in the Heroku project settings tab.



5. Check the URL of the deployed app under **Domains**: https://dashboard.heroku.com/apps/APP_NAME/settings. The application should be running there.



You can find the source code on GitHub.

Next steps

Our Learning Center contains tutorials and videos on how to build your next Java web application with the Vaadin framework. Try our Quick start tutorial to learn more!

Any questions? Please let us know by commenting below.

Co-authored by Anastasia Smirnova and Mikael Sukoinen

Comments (3)

Frank Zillus 3 months ago

Please document the Steps, which are different to Java EE and plain Java Servlet. Tnx!

Shourya Bansal 5 months ago

When trying to deploy through github, I keep getting an error "unable to access jarfile target/my-file-name-file-version.jar

Anastasia Smirnova 5 months ago

Hi,

have you used our default Spring Starter project? What is the Vaadin versiou you have? Also, what is the content of your `Procfile` and `heroku-settings.xml` ?

USE CASES

Swing migration

UX consulting

COMPANY

About us

Get in touch

Team

Careers

Trademark