D387 – Advanced Java

By Nate Edge

Student ID: 011178911

Deploying a Spring Application to a simple Cloud Platform

So, you just finished building your first Spring Boot application. Now you want to show off your development skills to all of your non-technical friends. How do you do this? Easy, just send your GitHub repository to your friends. Then have them jump on the command line, possibly setup an SSH key, pull your repository, download an IDE, setup a JDK, and run the back-end application. If this is a full stack app, you'll also have to walk them through how to install whichever front end framework your application is built on and run a server for the front end. At this point, most of your friends are going to be lost and have no interest in testing out your application. Isn't there an easier way to share your creations with the world?

Enter, the cloud! The cloud is essentially just a collection of servers that host software online. For people to be able to use an application deployed to the cloud, it is as easy as clicking a link. For most non-technical people, this is likely the extent of the work they will want to do anyway.

So how do you deploy something to the cloud? If this is a small-time project, like your first full stack Spring Boot application, I'd recommend using something like GitHub Pages. If you've been committing your code to a GitHub repository, deploying your application to the cloud will be pretty straightforward.

First, go into your GitHub repository. Next, click the settings button in the navbar. On the lefthand side, under 'Code and Automation', click the Pages button. Under 'Build and Deployment', make sure that 'Source' is set to *Deploy from a branch.* Under 'Branch', select *main* and select the */root* directory. Now click Save. Note: If you own or want to purchase a domain, you can set a custom domain name. For basic projects, it's probably best that you use the default GitHub domain.

After clicking save, you will have to wait 5-10 minutes for your application to go live. Once it's live, you can access the URL to your web application in two places. The first one is on the main page of your repo, on the right side under 'Deployments'. Click this button, then click the link and viola! Your application is live, in the cloud! The link should look something like your-user-name.github.io/your-repo-name/. You can also access this URL back in Settings > Pages > GitHub Pages.

Deploying a Spring Application to a Public Cloud Platform (Microsoft Azure)

Okay, so deploying applications to GitHub Pages is pretty cool. But what if you if you have an application that is going to have a lot more users than just your friends? If you are at the point where scalability is an issue, you probably want to consider hosting your application with a Public Cloud provider. Most companies with enterprise applications host their apps on Public Cloud providers such as Amazon Web Services, Microsoft Azure, or Google Cloud Platform. If you work for a company in Asia, you might be using Alibaba Cloud.

The second most popular cloud provider in the world, Microsoft Azure, is a great option if you are new to the cloud. In a few steps, you can have your Spring Boot application up and running on Azure. First, you will need to create a Microsoft Azure account. You will also need to setup a container registry, which is essentially a repository for your container image. If you packaged your application inside of a Docker container, I would recommend using Docker Hub. You'll want to upload your image to Docker Hub and edit the visibility settings to public. This is an important step, as your image will need to be public in order for Azure to find it.

Next, sign into your Azure account and create a new repository and select *Container App.* Create a resource group and give it a name, such as *application-name-repository*. Complete the rest of the *Basics* form, click next, and fill out the form under the *App Settings* tab. Give your container a similar name, such as *application-name-container*. Select *Docker Hub* as your image source, select public, and enter your image tag (found on Docker Hub).

Enable Ingres, a controller that exposes your app to network requests, allowing you to control the traffic to your containerized app. Select HTTP as the transfer protocol, enable the container to accept traffic to anywhere, and select port 8080 as the target port to route traffic to. Click the Review button, review your completed form to ensure accuracy, and click the Create button. You should see a message that says 'Deployment is in Progress'. After waiting for a few minutes, if the deployment is successful, you will now see a message that says 'Your Deployment is complete.' Click the 'Go to Resource' button, then click the Application URL. You can now see your live application. As always, it is best practice to test the application to ensure that everything is running smoothly.

Hosting applications to the cloud offers many benefits, such as security, user accessibility, reliability, and scalability. Deploying applications to the cloud is an important part of Software Development. Whether you are creating small applications for fun, or enterprise applications for millions of users, this is an important an important skill for any Software Developer to have.