bit Computer Training by Netmind

Boosting Your Digital Skills

MI1575

Taller DevOps

Serverless web App - CI/CD Docker

Barcelona
C/ Almogàvers 123
08018 Barcelona
T +34 933 041 720
info@bit.es

Madrid
Pl. Carlos Trías Bertrán 7
Edificio Sollube 1ª Planta
28020 Madrid
T +34 914 427 703

www.bit.es

Taller DevOps

Overview

This article covers exactly how to import, build, deploy, and test a Java application written with Spring Boot Reactive as the runtime and use docker to deploy in a CI/CD pipeline.

The Source Code

Fork the projects from GitHub repository:

https://github.com/azure-samples/springboot-hello-azure

Multi-Stage Docker Images

This is a neat feature in recent versions of Docker and has helped me circumvent some limitations and deliver a developer experience that is equally awesome on your local computer as well on the CI/CD platform.

The basic springboot-hello-azure/Dockerfile contains the following instructions, and it is useful for building/testing the Docker image locally.

Dockerfile:

```
FROM openjdk:8-jdk-alpine
COPY target/*.jar /usr/src/app/myapp.jar
EXPOSE 8080
CMD [ "/usr/bin/java","-jar","/usr/src/app/myapp.jar" ]
```

Dockerfile.cicd:

```
FROM maven:3.5-jdk-8-slim as mavenBuild
COPY src /usr/src/myapp/src
COPY pom.xml /usr/src/myapp
RUN mvn -f /usr/src/myapp/pom.xml clean package

FROM payara/micro:5.182 as runtime
COPY --from=mavenBuild /usr/src/myapp/target/*.war $DEPLOY_DIR/ROOT.warEXPOSE
8080
```

The second one above as stated earlier on, has the multi-state build surpport and configuration. This is the version we will use later in this article to import/create Docker image on Azure DevOps project wizard.

Create a DevOps Project on Azure

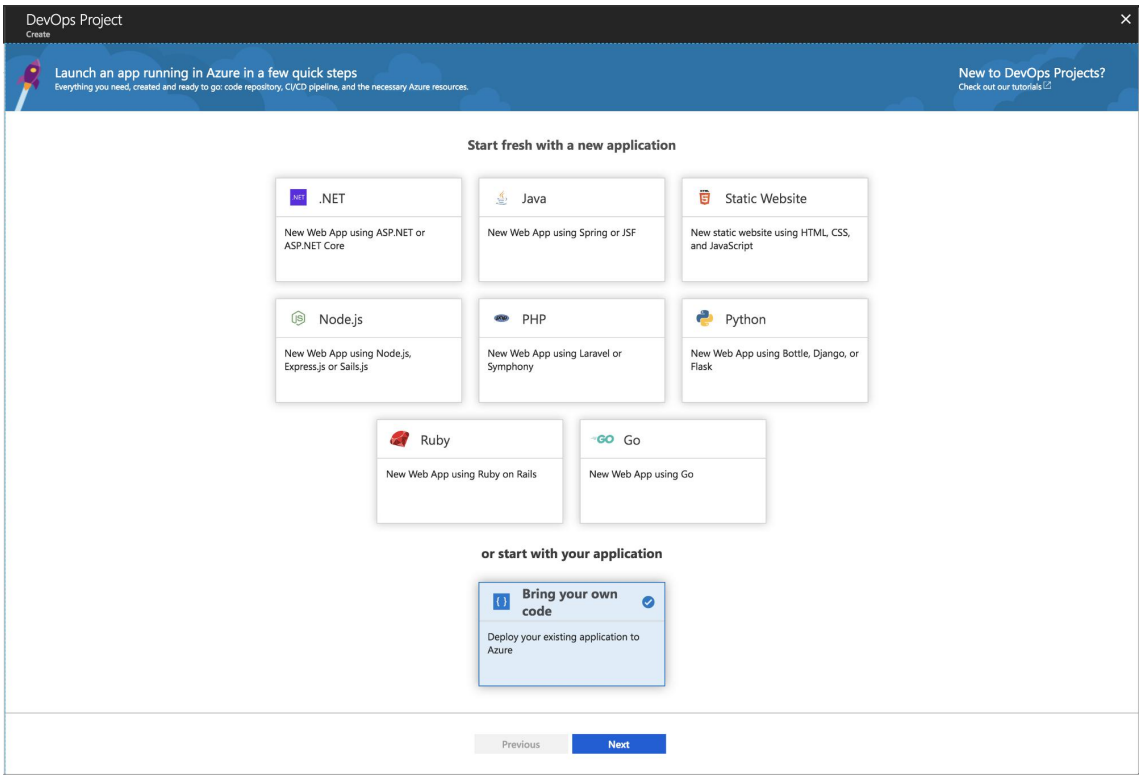
Go right now to the Azure Portal, and hit Create DevOps Project. If you don't see this tile on your dashboard, search for DevOps on the left menu inside "All Services". You must find "DevOps Projects". Once you open it, you should be able to Add a new project.

The steps below will walk you through the project creation wizard, so stay tuned as we check up to 8 steps.

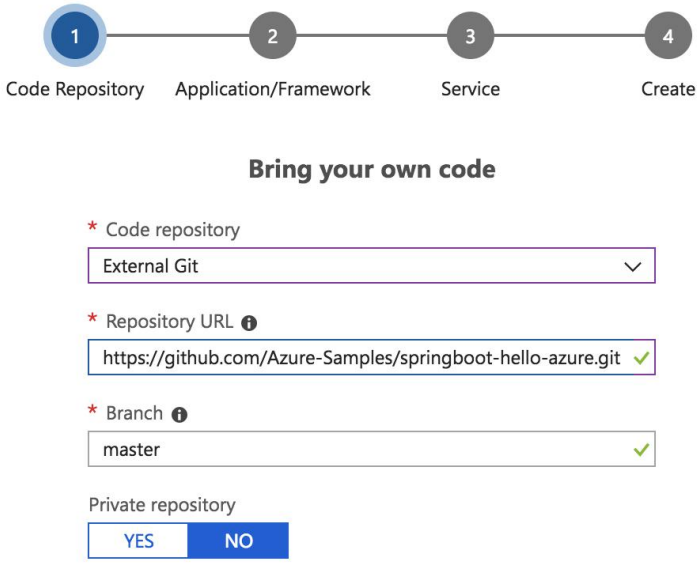
1. Choose bring your own code
2. Select which git repository to use (Spring Boot or MicroProfile)
3. Indicate it is a Dockerized app
4. Select which Azure service to deploy to
5. Indicate which exact Dockerfile to use (hint: Dockerfile.cicd)
6. Configure Visual Studio Teams Service, and also set the name of your app (must be globally unique).

Step 1: New Sample or Existing Code? Bring your own!

Make sure you select Bring your own code

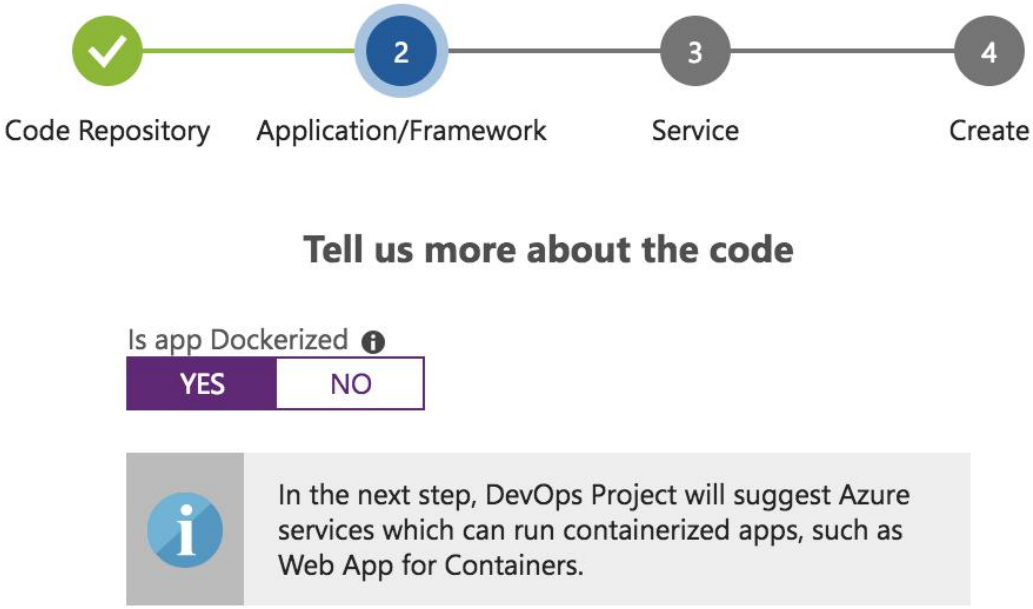


Step 2: Select External Git Repository



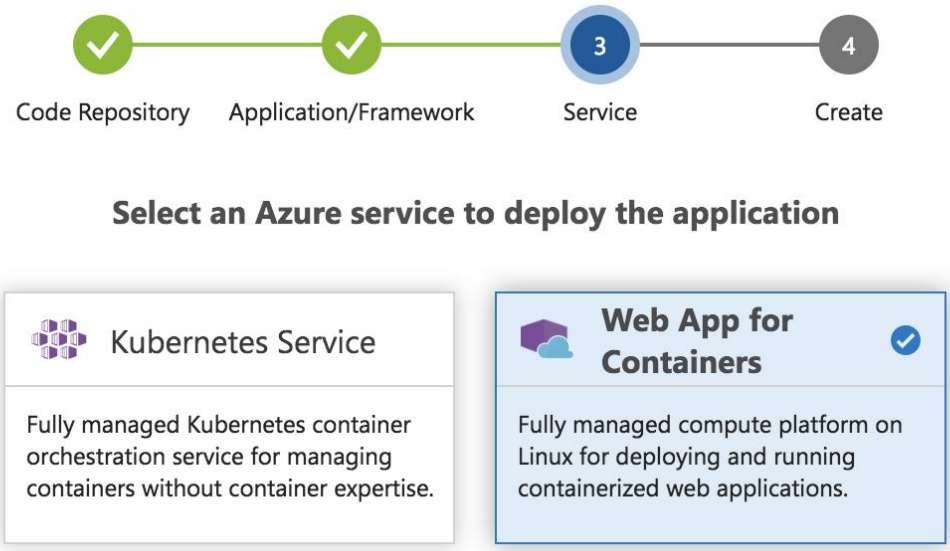
Step 3: Indicate this is a Dockerized application

We will be building and deploying using Docker, so doesn't matter the framework or platform, we will build a Docker image. Therefore, here you select Dockerized application.



Step 4: Select Web App for Containers

Make sure you mark the option that says Web App for Containers.



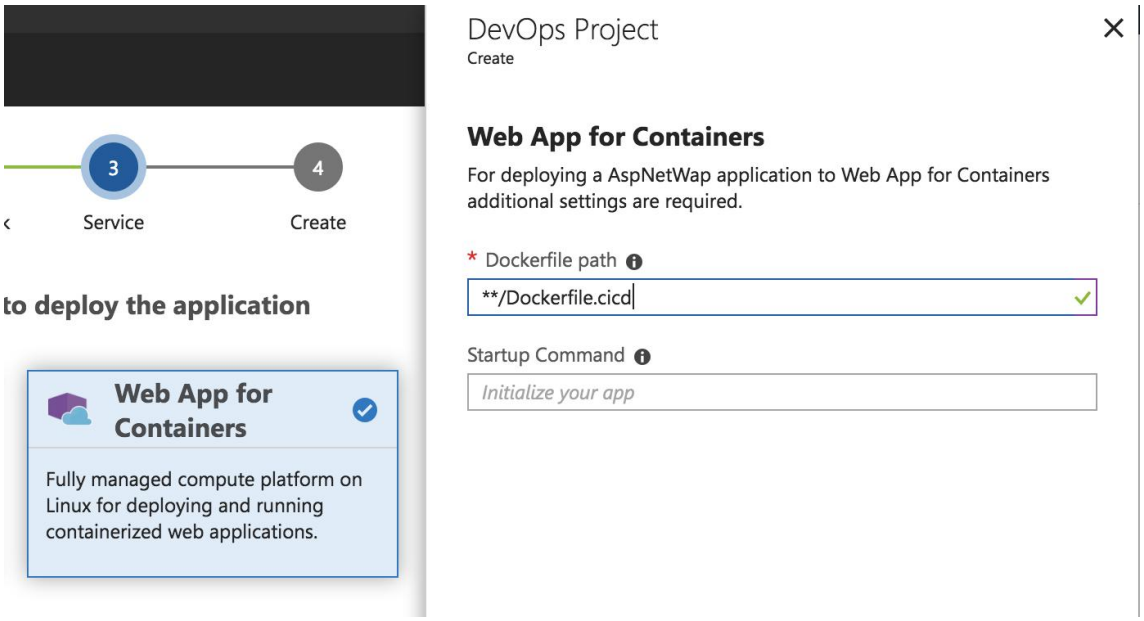
This is a simple service that can spin up dockerized web applications with a load balancer in front and other perks for scaling. It costs less than Kubernetes but of course won't provide the entire feature set of Kubernetes.

I did not test this with AKS (Azure Kubernetes Service), but please feel free to try it out and post in the comments!

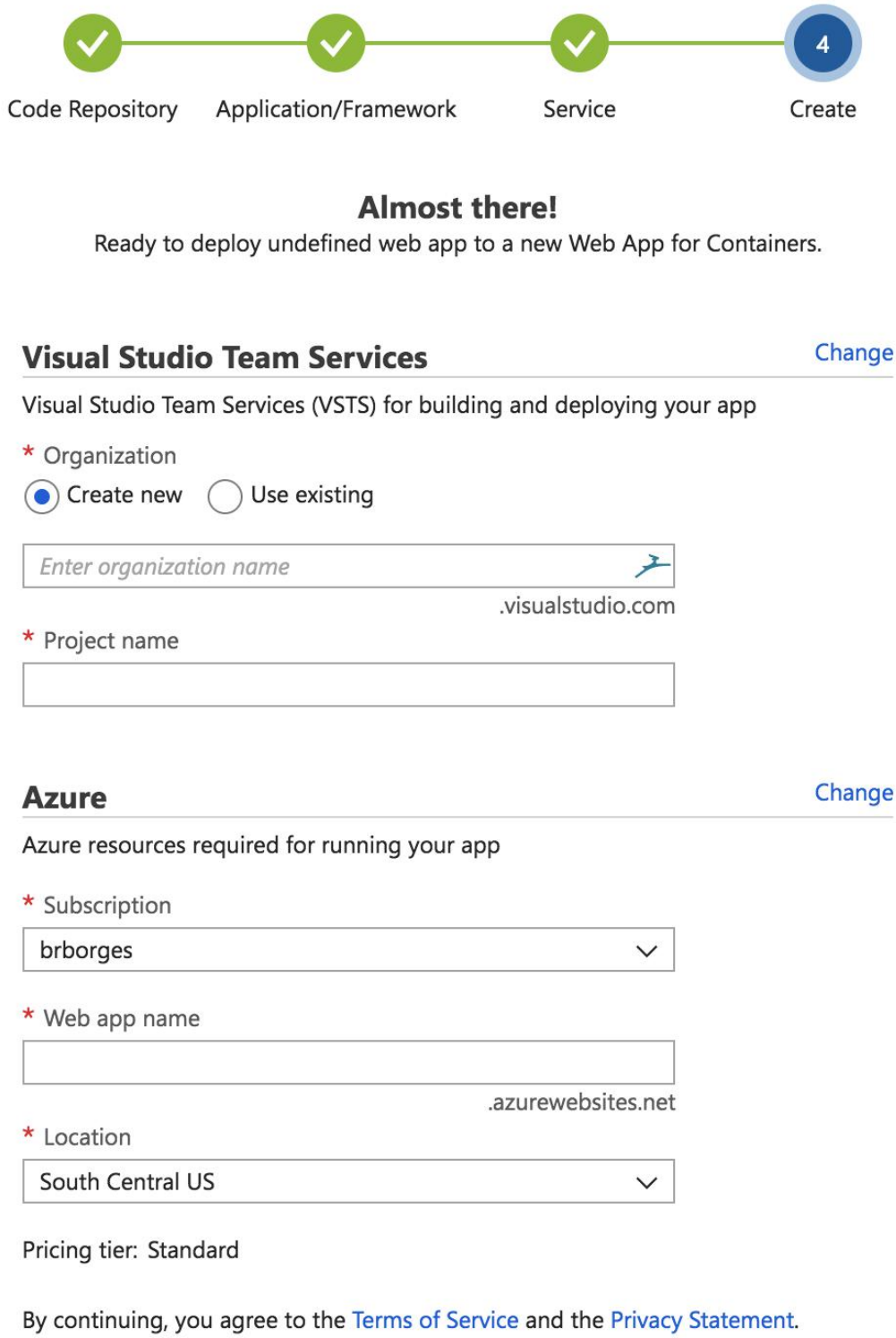
Step 5: Indicate which exact Dockerfile to use (Hint: Dockerfile.cicd)

Since this is a Dockerized application, it simplifies things if we also use Docker to run the entire build cycle of this project. For that, we have Dockerfile.cicdfor both projects which you can use to compile/package the Maven project, and then right away build a Docker image with the artifact produced by Maven.

This is exactly what we want.

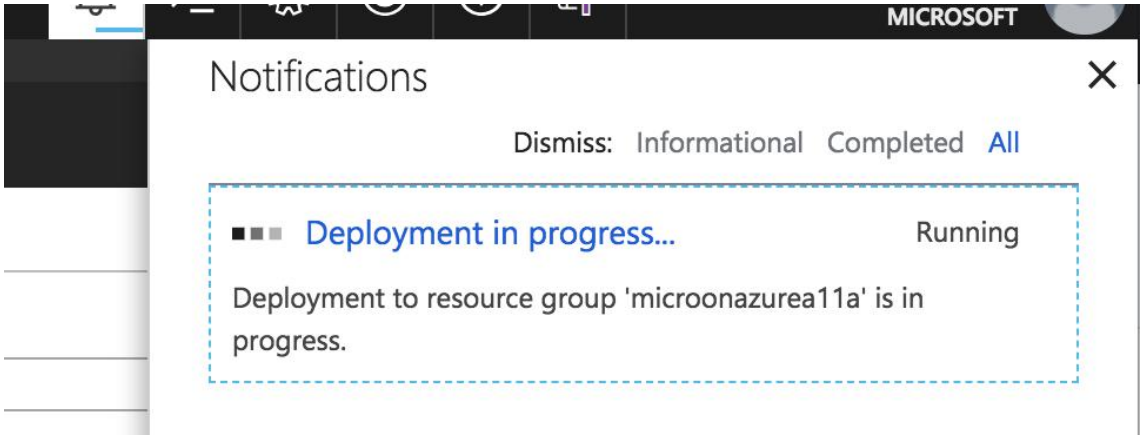


Step 6: Fill the last form to set up your VSTS environment and Azure resources (web app name).

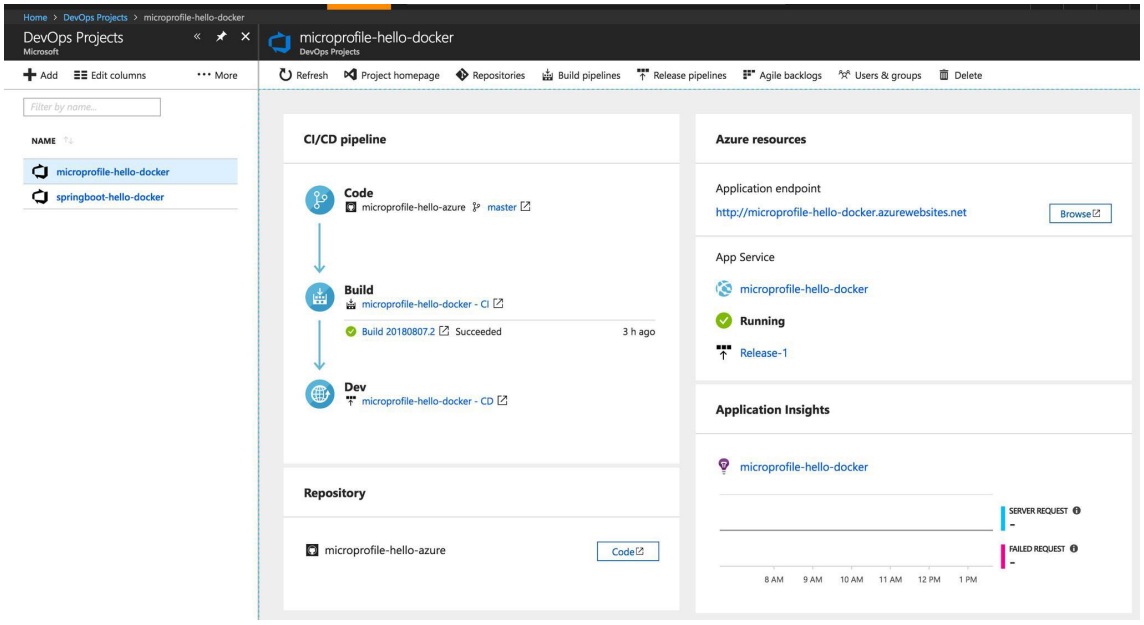


Wait and see…

The deployment will be in progress, so keep an eye and wait for everything to be complete.



Once the deployment is ready, here's the amazing dashboard you will have for your DevOps project, and all of its resources created for you automatically:



DevOps Project dashboard

Your application is now running in the Cloud—Microsoft Azure to be more precise!
Visit http://<your-app>.azurewebsites.net/api/hello once deployment is done.