

Containerizing the Java RESTful Engine

Overview

This article will discuss how to containerize the Windward Java RESTful Engine using Docker. With this method, it is simple to create a container running an instance of the Windward Java RESTful engine. Note that while this guide assumes that you are using a Windows operating system, this can be done on a Linux-based OS as well.

Requirements

- [Docker Desktop](#) installed, and configured to use Linux Containers
- A valid license key for the Java RESTful engine

Licensing

Windward licensing exists for many different needs. There are Pros and Cons for each type of [License](#). The FLEX license is meant to be used in automated management environments such as Container management systems, Azure services and auto scaling virtual environments. If you are planning on hosting your application with the Java RESTful Engine with any of these types of environments, we highly recommend the purchase of a FLEX (per page) license instead of a PRO (per machine/instance) license.

i Using a **PRO** license limits the number of containers that can be run the RESTful Engine concurrently to the number of engines purchased. Both **PRO** and **FLEX** license keys will be released automatically when the container shuts down.

Verify Docker Installation

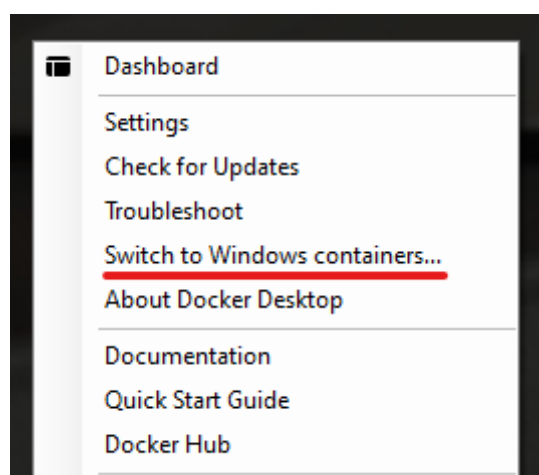
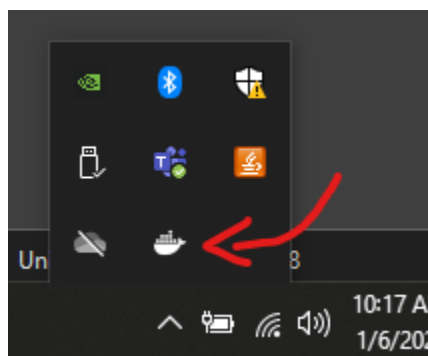
This guide assumes that you have Docker Desktop installed and configured on your machine. If it is not, follow the Docker Desktop installation documentation [here](#). To verify that your installation was successful, open a command prompt or Powershell window and run the command:

```
docker
```

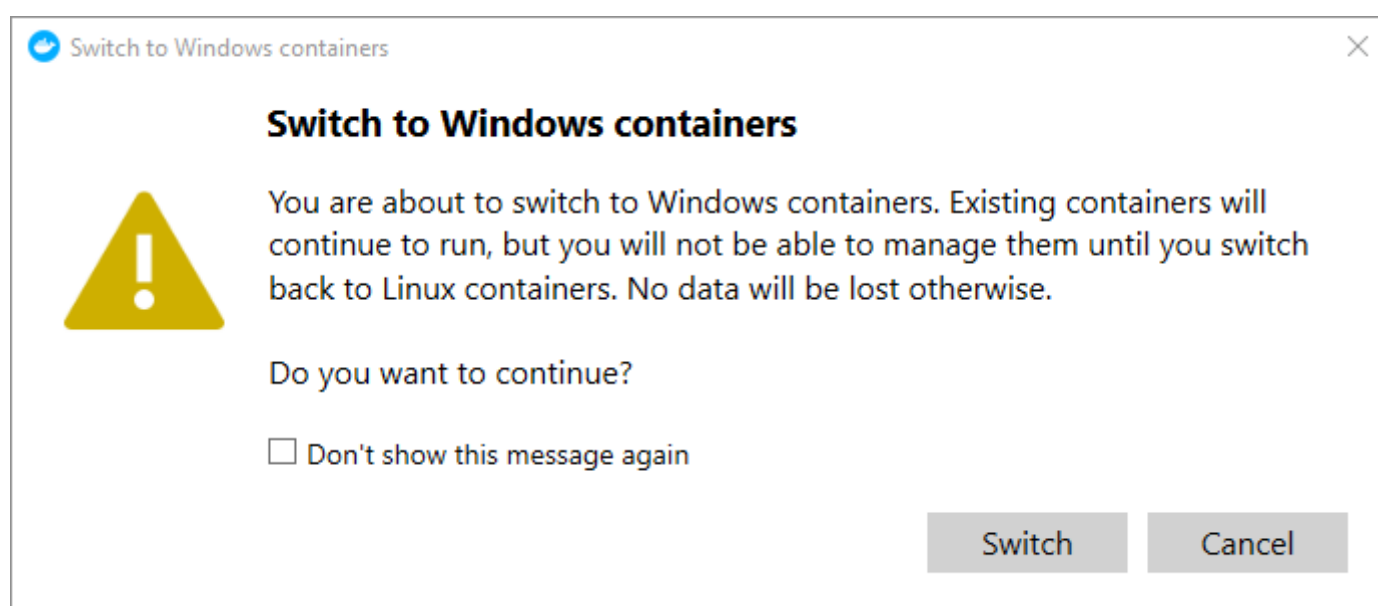
 Click to copy

If it returns a list of arguments, then Docker is installed on your machine. Additionally, Docker must be configured to run Linux Containers to successfully containerize the Windward

JavaREStful engine. To check, right-click on the docker icon located in the hidden icons section of the Windows taskbar.

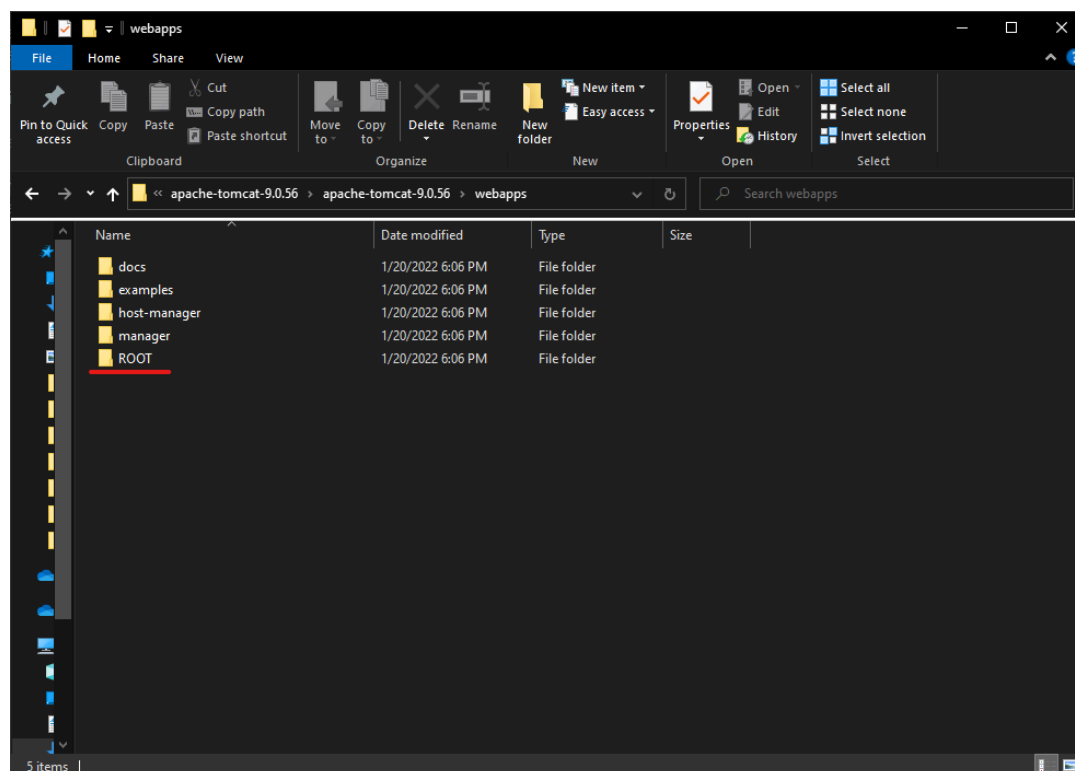


If the menu that appears says "Switch to Linux containers..." then Docker is configured to run Windows containers. If the menu says "Switch to Windows containers..." then Docker is configured to run Linux containers. To switch, simply click that option in the menu and select "Switch" in the resulting pop-up.

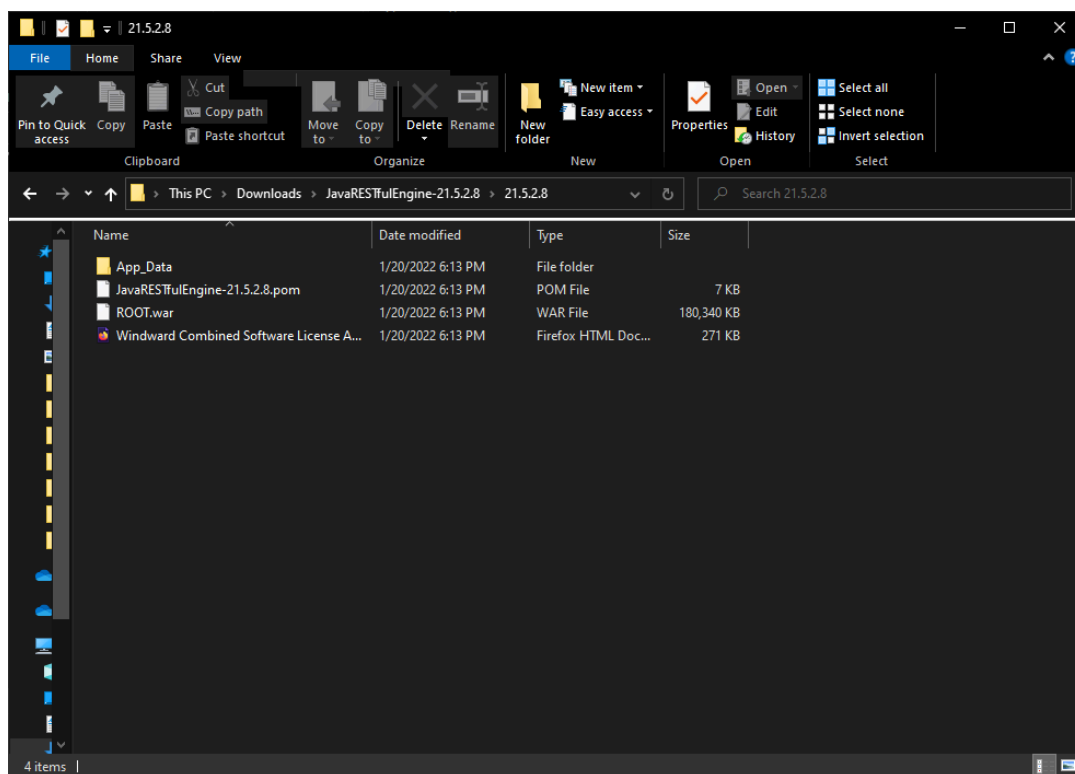


Getting Started

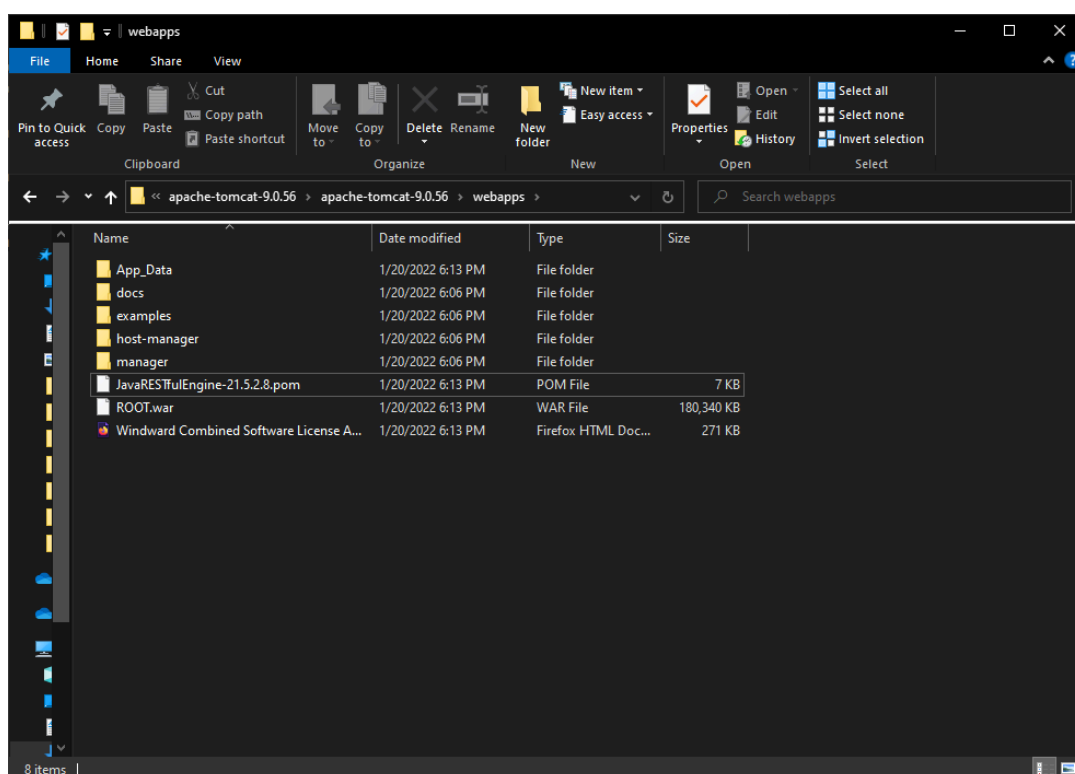
To begin, visit the Windward Samples github [here](#) and download the samples. Inside, you will find the JavaRESTful folder in Container/Java RESTful Engine, which contains the Dockerfile necessary for this guide. Next, download the Java RESTful engine .zip file from [here](#). You will also need Apache Tomcat 9 Core .zip file, which you can download [here](#). Once downloaded, extract the Tomcat 9 and Java RESTful engine to a location of your choice. We will now install the Java RESTful engine inside the Tomcat 9 download. Navigate to the webapps folder inside of the Tomcat 9 directory, and delete the ROOT folder.



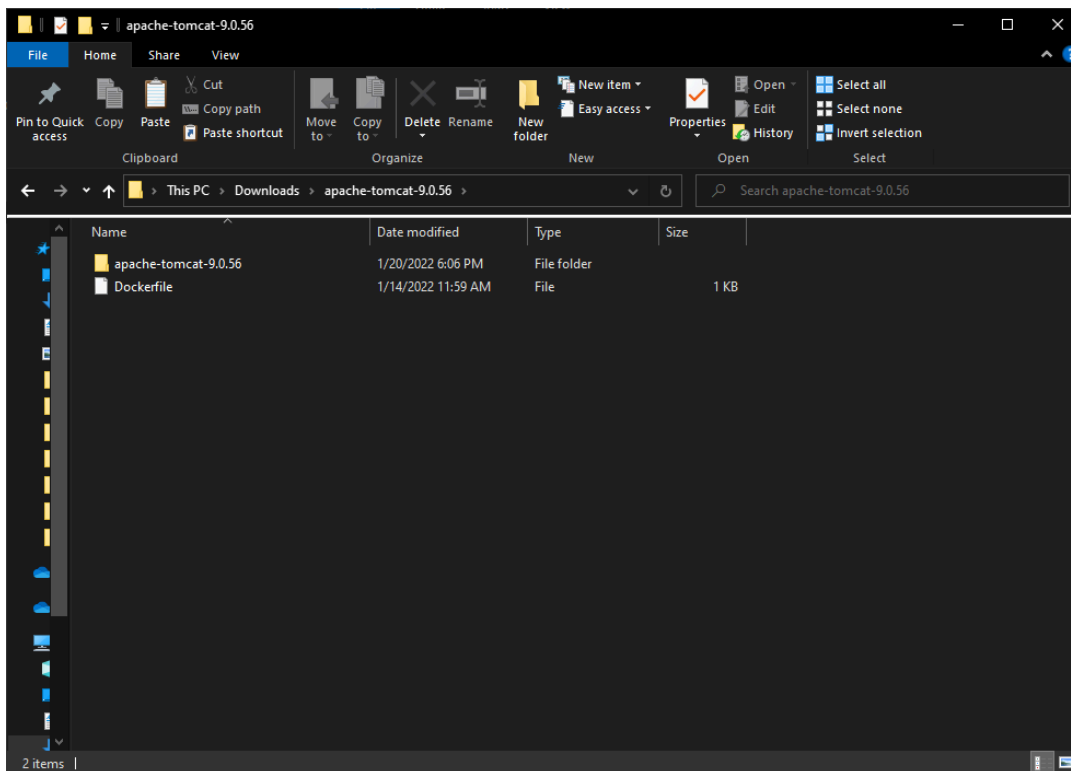
Now we can copy over the necessary files. Inside of the Java RESTful engine directory, you will find the following:



Copy all of the above files, and paste them inside of the webapps folder in the Tomcat directory. Once complete, the webapps folder should contain:



It is almost time to create the image for the container. The final step is to move the Dockerfile located in the Java RESTful Engine sample folder to the root Tomcat directory.



And that's all that we need to build the image!

Dockerfile

```

1 # Grab Alpine Linux base image
2 FROM alpine:latest as base
3
4 # Add necessary tools
5 RUN apk update
6 RUN apk upgrade
7 RUN apk add --no-cache bash
8 RUN apk add --no-cache openjdk8
9
10 # Copy the tomcat installation over to the /opt directory, and set as working directory
11 COPY ./ /opt
12
13 # Replace VERSION_NUMBER below to match the version of tomcat that you are using (Ex:apache-tomcat-9.0.56)
14 WORKDIR /opt/apache-tomcat-VERSION_NUMBER
15
16 # Set the container to start the Tomcat Server when launched, and sleep so that the container doesn't exit afterwards
17 ENTRYPOINT ./bin/catalina.sh start && sleep infinity

```


The Dockerfile manages how Docker will build the image for the container. For this guide, Docker is going to start with the latest alpine Linux image, and then install a bash shell and openjdk8 for the Tomcat server. From there, Docker will copy over the Tomcat installation we just finished putting together into the /opt directory and set the entry point for the image to the start script provided with Tomcat. This will result in the server launching when you start the container created from this image. The sleep infinity command is so that once the start script terminates, Docker won't stop running the container. Normally, once the command/program that is set as the entry point terminates, Docker will stop running the container. For more information on the Dockerfile, see Docker's documentation [here](#). For this Dockerfile to work,

you need to change the VERSION_NUMBER on line 14 so that it matches the name of the apache-tomcat folder. And that is the Dockerfile! Its now time to build the image for the container.

Building the Image

The next step is to build the image that we want to use to create a container. With command prompt or Powershell, navigate to the root folder of the Tomcat directory, where the Dockerfile is located. To build the image, run the command:

```
docker build -t java-restful-engine .
```

 Click to copy

This command will build an image using the Dockerfile in the current directory, and name the image "java-restful-engine."

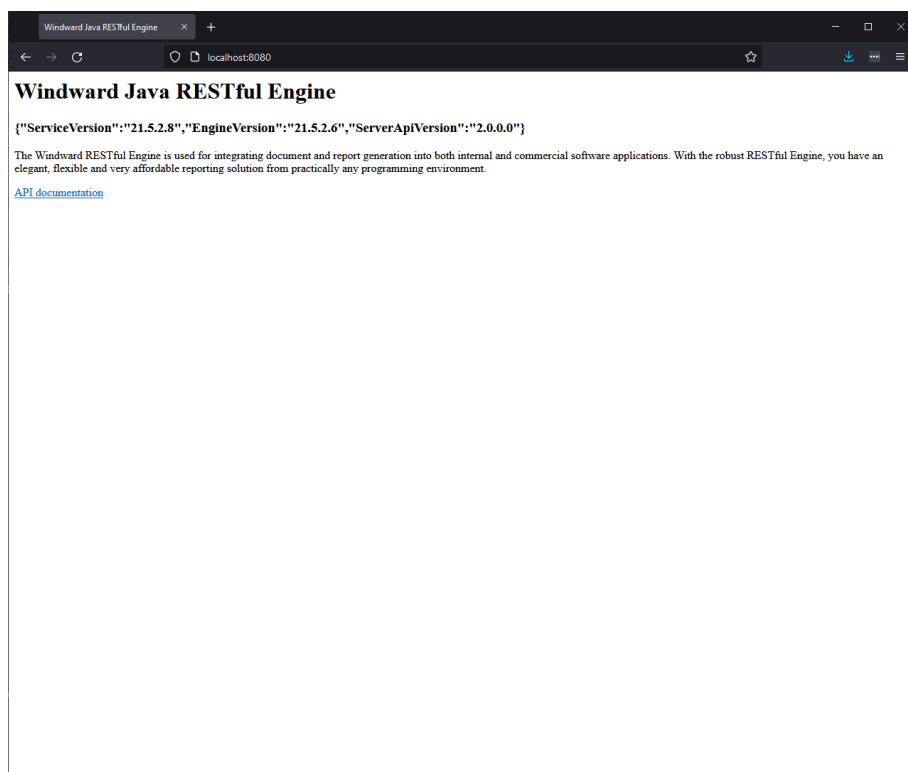
Creating the Container

To create and run a container from the image, simply enter the following command in the same command prompt or Powershell window that you used to create the image:

```
docker run -it --name javarestful -p 8080:8080 java-restful-engine
```

 Click to copy

This command creates a container from the "java-restful-engine" image named javarestful, and maps port 8080 of the container to the local machine's port 8080. If everything was successful, you should be able to visit localhost:8080 and see the following in a browser.



Configuring the Java RESTful Engine

Now that the container is running, we can configure the Windward Java RESTful engine instance in the container. First, connect to the running container using the following command:

```
docker exec -it javarestful bash
```

 Click to copy

This command will connect you to the container's bash shell, allowing you full access to the container. In order to use this instance of the Java RESTful engine, we need to set our license key in the WindwardReports.properties file. To do so, navigate into the ./webapps/ROOT/WEB-INF/classes folder and open the WindwardReports.properties file in a text editor of your choice. Note that the image doesn't have a text editor installed, and you can use the following command to install vim:

```
apk add vim
```

 Click to copy

Once you have opened the WindwardReports.properties file in a text editor, simply replace "[[LICENSE]]" with your license key, and save your changes. Once you've entered your license key, simply restart the Tomcat server using the following commands in the /opt/apache-tomcat-VERSION/bin directory:

```
./catalina.sh stop
```

Click to copy

```
./catalina.sh start
```

Click to copy

And that's it! You have successfully containerized the Java RESTful engine, and you can communicate with it by sending requests to localhost:8080. For more information on using the Java RESTful engine, see our API [here](#).