Overriding ASP Net Core settings with Environment Variables in a container.

Craig Godden-Payne
May 1 · 3 min read ★

When working with an application, it is often beneficial to split out configuration from the application. The reason for this being that you can:

- Configure your application on the fly (you can override settings in an emergency, without having to deploy)
- Don't need to build environment-specific versions of your application (assuming you build a docker image with the configuration baked in)

. . .

ASP.NetCore Configuration Builder

It is very typical to see something similar in asp.netcore applications.

```
var configuration = new ConfigurationBuilder()
  .AddJsonFile("appsettings.json")
  .AddJsonFile($"appsettings.{env.EnvironmentName}.json")
  .AddEnvironmentVariables()
  .Build();
```

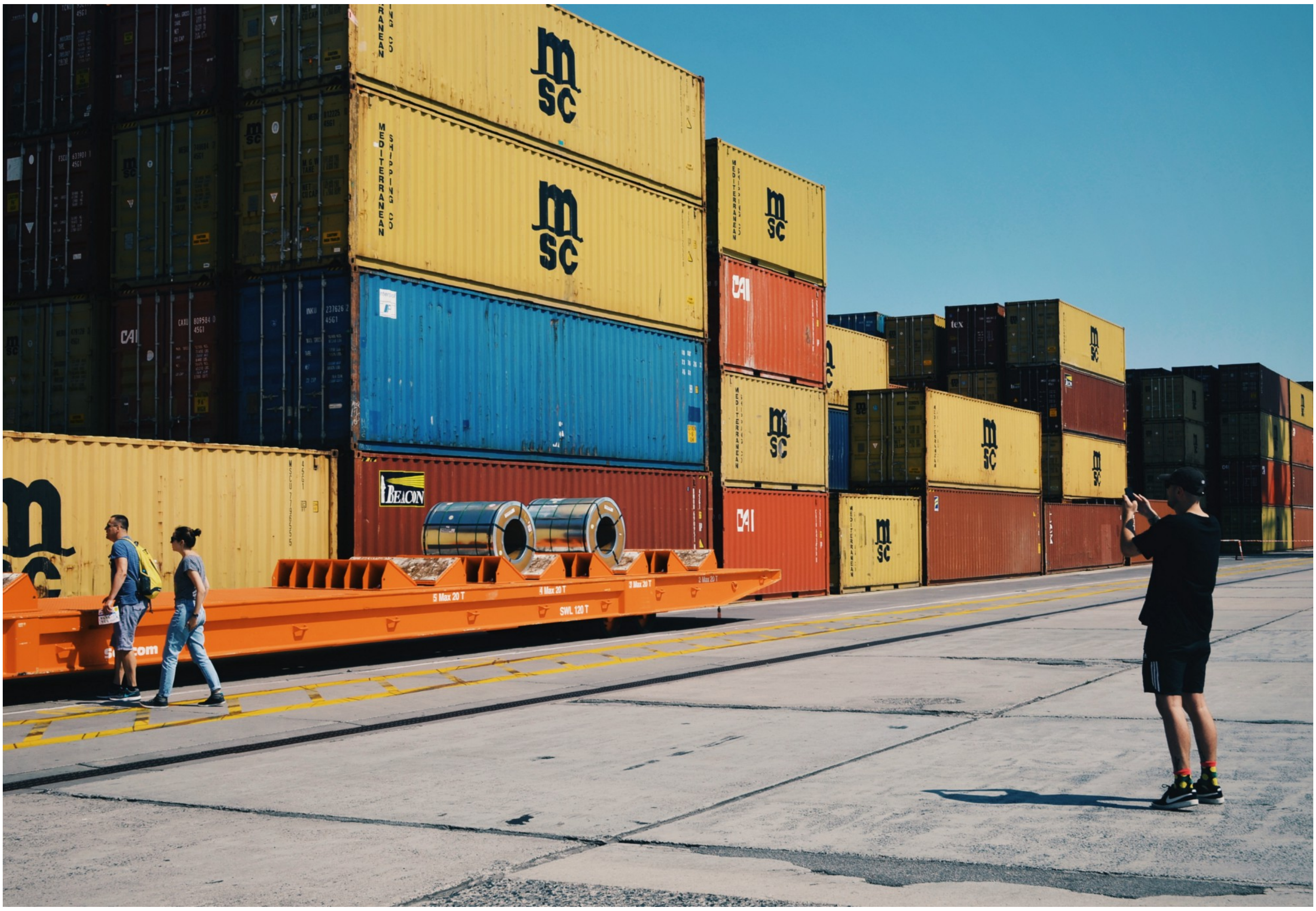
Code like this allows multiple app settings files, suffixed with the environment to be created.

This technique leads most people to create files equivalent to appsettings.qa.json, appsettings.prod.json.

. . .

It sure is handy to see all the settings in one place. Still, on the other hand, you have to be careful not to expose sensitive credentials in your settings files, and you are locking yourself into baking environment-specific docker images.

One thing you may overlook with using this templated code is that the configuration builder in dotnetcore will use multiple sources to build up our configuration, and in this case, environment variables.



. . .

Docker Environment Variables

Docker supports environment variables to be set in a variety of different ways. If you are running in something like ECS or Fargate, you can inject and replace variables via a task definition.

Docker allows environment variables to be configured in a Docker file using the `ENV` instruction. This instruction takes a key-value pair, separated by a space or an equals character. For example:

```
ENV ENVIRONMENT=Development
```

. . .

Overriding using Docker Run

Environment variables can be overridden using environment variables set in the Docker file when running the image by using the `-e` flag:

```
Docker run -e "ENVIRONMENT=QA" my-application
```

. . .

Overriding AppSettings with Environment Variables

A typical config file may look like the following, but it is still possible to override sections within it using environment variables.

```
"Logging": {
  "LogLevel": {
    "Default": "Debug",
    "System": "Information",
    "Microsoft": "Information"
  }
}
```

To override, you need to delimit the sections using a colon, and since Environment variables take precedence over JSON files, we can override at runtime!

```
Docker run -e "Logging:LogLevel:Default=Info" my-application
```