Working with Containers

Chris B. Behrens
Software Architect

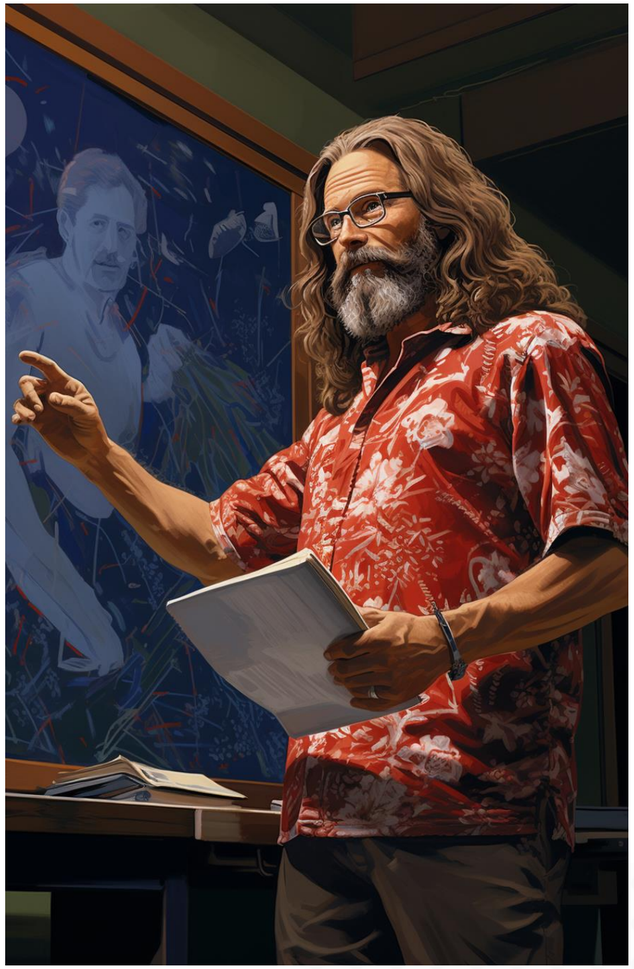
@chrisbbehrens

Where Do We Keep This?

Other places offer private registries...

But it's nice to have everything in the same place

What We're Going to Cover



I'm going to assume you know a bit about containers

If not, check out my course *Running Jenkins in Docker*

Visual Studio has some pretty good Docker tools

A pretty generic Dockerfile for applications

Multi-stage container build

Multiple FROM statements

Use what you need to build

And then use a slim runtime

Demo

Launch our project in Docker

Look at the process in Visual Studio

Examine the Dockerfile

Talk about what it all means

The Multi-stage Dockerfile

The Dockerfile

```
#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["Globomantics.Web/Globomantics.Web.csproj", "Globomantics.Web/"]
RUN dotnet restore "Globomantics.Web/Globomantics.Web.csproj"
COPY . .
WORKDIR "/src/Globomantics.Web"
RUN dotnet build "Globomantics.Web.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "Globomantics.Web.csproj" -c Release -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Globomantics.Web.dll"]
```

Demo

Create an entirely new build

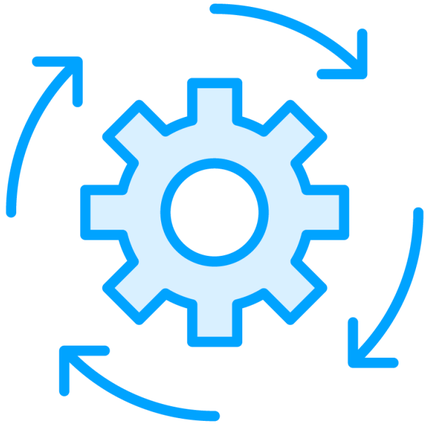
Builds the code

Using our multi-stage Dockerfile from Visual Studio

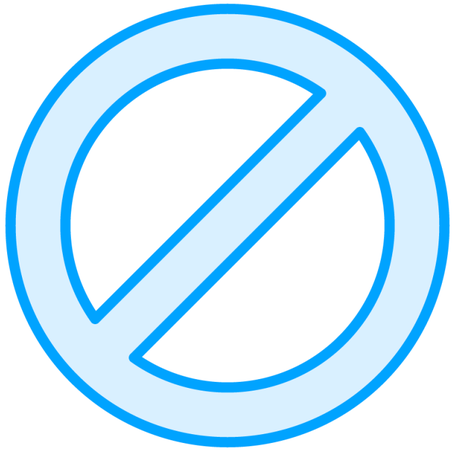
Pushes it to our new Docker registry in Artifactory

The Problem

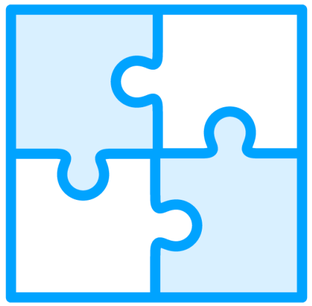
Build and Drift



We're building after testing



Ideally, we wouldn't



The two builds are very close to each other

So, there's little opportunity for drift

The Dockerfile

```
COPY ["nuget.config", "."]

COPY ["Globomantics.Web/Globomantics.Web.csproj",
"Globomantics.Web/"]

COPY
["Globomantics.Communication/Globomantics.Communication.csproj",
"Globomantics.Communication/"]

RUN dotnet restore "Globomantics.Web/Globomantics.Web.csproj"
```

The Wrong Way to Fix This

Trying to cram all this into the Dockerfile

We could probably make this work, but no

The Right Way to Fix This

1. Globmantics.Web build performs whatever testing and validation are necessary
2. Globmantics.Web build publishes the same way that the Dockerfile does, and pushes the result to a new Artifact feed
3. The final step of the Globomantics.Web build is to pass the version of the package to the Docker build
4. The Docker build pulls the published content from the new artifact feed and packages it into an image
5. The Docker build pushes the corresponding docker image to the Docker feed in Artifactory

Demo

- Look at how I've modified the builds
- Examine the new feed I've created in Artifactory
- Execute a build
- Watch it do its work as it publishes the content
- Triggers the image build

We've Broken the Steps Up

Wrap-up

We could just use an artifact in Jenkins

Two reasons

If something goes wrong, this way is easy to resume in the middle

Having those files separated ends up being useful

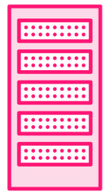
Artifactory and Deployment

When Is Actual Deployment?

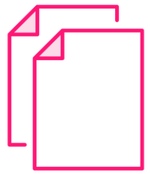
A single, bare-metal server

One we're going to replace

Deployment Steps



Provision new hardware



Copy the new version of the software to the new hardware

Transition the network names, ports, and resources from the old hardware to the new hardware

Once that’s complete, turn off the old server

Deployment

The delivery of software product in order to deliver value.

Two Edge Cases

We take a server down and fiddle with it

That's not deployment

We provision new code for the new server

Yes, we deployed twice

Pushing the image to Artifactory is the penultimate stage of deployment to Production*

Pushing the image to Artifactory is the penultimate stage of deployment to Production*

Pushing the image to Artifactory is the penultimate stage of deployment to the Production artifact store*


nuget-
dev
nuget-
staging
nuget-
prod

The Two Mistakes

Trying to execute tests or other validation in your Dockerfile

Building your binary too early and needing to rebuild it later

Summary

- Artifactory and Docker images
- The Dockerfile template in Visual Studio
- Creating a Docker registry in Artifactory
- Setting up our local tools to connect to it
- A Jenkins build to push to it