# Build and publish app containers

## Overview

1. Compile the apps locally
2. Add Dockerfile to apps
3. Build docker images
4. Publish docker images to **Azure Container Registy**

## Create Azure Container Registry

1. Navigate to Azure portal
2. Create Azure Container Registry in your "afsdemo" resource group
3. Name it "afsdemo$user"
4. Save registry url, which should be similar to this - afsdemogurba.azurecr.io
5. Save username and password from access keys blade
6. Execute login command on your workstation

```
docker login afsdemo$user.azurecr.io -u afsdemo$user -p
SECRET
```

## Create git branch for your changes

Execute "git checkout -b $user", replacing $user with your user name

## Dockerize ID service

Add file named 'Dockerfile' to id/ directory:

```
FROM microsoft/dotnet:2.1-sdk-alpine AS build
WORKDIR /app
COPY . ./build/
WORKDIR /app/build
RUN dotnet restore
RUN dotnet publish -c Release -o out

FROM microsoft/dotnet:2.1-runtime-alpine AS runtime
WORKDIR /app
COPY --from=build /app/build/out ./
ENTRYPOINT ["dotnet", "id_service.dll"]
```

Make sure that Dockerfile is included in published files. Add following section to id_service.csproj file:

```
<ItemGroup>
    <None Update="Dockerfile">

<CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>

<CopyToPublishDirectory>PreserveNewest</CopyToPublishDirectory
>
    </None>
  </ItemGroup>
```

Try starting the container manually

```
docker run -p 8088:8088 --name afs-demo-id -d
afsdemogurba.azurecr.io/afs-demo-id:latest
// note! use your custom registry url
```

Open http://localhost:8088/swagger in a browser to see it works. Note that mongo DB connection is not available yet, thus token requests will fail with a timeout.

Remove this manually created container:

```
docker rm -fv afs-demo-id
```

## Dockerize API

Add file named 'Dockerfile' to api/ directory:

```
FROM node:alpine

WORKDIR /usr/src/app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 8080
CMD [ "npm", "start" ]
```

Change the path to token keys in controller.js

```
// Use relative path to the key file instead of the absolute
path

const fs = require('fs');
//var path = require('path').basename(__dirname);
//var cert = fs.readFileSync(path +
'/app_data/jwt_1530633205_public.pem');
var cert =
fs.readFileSync('app_data/jwt_1530633205_public.pem');
```

Try starting the container manually

```
docker run -p 8080:8080 --name afs-demo-api -d
afsdemogurba.azurecr.io/afs-demo-api:latest
// note! use your custom registry url
```

Open http://localhost:8080/user/admin in a browser to see it works. Note that mongo DB connection is not available yet, thus requests will return nothing.

Remove this manually created container:

```
docker rm -fv afs-demo-api
```

## Dockerize App

Add file named 'Dockerfile' to app/ directory:

```
FROM nginx:stable-alpine
COPY . /usr/share/nginx/html
```

Try starting the container manually

```
docker run -p 8089:80 --name afs-demo-app -d
afsdemogurba.azurecr.io/afs-demo-app:latest
// note! use your custom registry url
```

Open http://localhost:8080/user/admin in a browser to see it works. Note that mongo DB connection is not available yet, thus requests will return nothing.

Remove this manually created container:

```
docker rm -fv afs-demo-app
```

## Build and push docker images

Add file named 'build.cmd' to the of your project:

```
SET afs-user=YOUR_USER
SET afs-version=latest
SET afs-registry=afsdemo%afs-user%.azurecr.io
SET afs-registry-user=afsdemo%afs-user%
SET afs-proxy=http://10.128.61.8:9090
SET build-proxy-args=--build-arg HTTP_PROXY=%afs-proxy%
--build-arg HTTPS_PROXY=%afs-proxy%

echo ON

REM Login to container registry
docker login %afs-registry% -u %afs-registry-user%

REM Build ID service
docker build -t %afs-registry%/afs-demo-id:%afs-version%
%build-proxy-args% id
docker push %afs-registry%/afs-demo-id:%afs-version%

REM Build API
docker build -t %afs-registry%/afs-demo-api:%afs-version%
%build-proxy-args% api
docker push %afs-registry%/afs-demo-api:%afs-version%

REM Build App
docker build -t %afs-registry%/afs-demo-app:%afs-version%
%build-proxy-args% app
docker push %afs-registry%/afs-demo-app:%afs-version%
```

Run the command and ensure that all images are pushed to the Azure container registry.

## Optional. Application CI/CD pipeline