Integrate containers with one another

Overview

1. Expose container settings
2. Test containers integration in a docker-compose solution

Expose API settings

Add API settings file called api/config.js:

```
var config = {};
config.mongo_conn_string = 'mongodb://localhost:27017/users';
module.exports = config;
```

Reference config in api/settings.js:

```
// reference config in server.js
var express = require('express'),
  app = express(),
  config = require('./config'),

// update mongo client initialization
mongoose.connect(config.mongo_conn_string);
```

Note! this config file will be mapped as a volume

Create the app_data directory in your solution root and copy api/config.js to app_data/config.js.

Change connection string in app_data/config.js:

```
config.mongo_conn_string = 'mongodb://mongo:27017/users';
// Note! 'mongo' is an alias that we will define later in the
docker compose YML file. The service will be located by this
alias rather than IP address or host name.
```

Expose ID settings

Copy id/appsettings.json to app_data/appsettings.json.

Change connection string in app_data/appsettings.json:

```
"MongoConnectionString" : "mongodb://mongo:27017/users"
```

Expose App settings

Create app/app_config.js file:

```
var config = {};
config.id_url = 'http://localhost:8088';
config.api_url = 'http://localhost:8080';
```

Include app_config.js file in your index.html

```
<head>
  <script src="app_config.js"></script>
</head>
```

Update app/index.js file to use settings from external script:

```
var app = new Vue({
  el: '#app',
  data: {
    id_url: config.id_url,
    api_url: config.api_url,
    ...
```

Copy app/app_config.js to app_data/app_config.js.

Change connection string in app_data/app_config.js:

```
var config = {};
config.id_url = 'http://id:8088';
config.api_url = 'http://api:8080';
// Note! We use aliases that we will later define in the
docker compose YML file.
```

Move mongo-seed directory to solution root

Copy /api/mongo-seed to ./mongo-seed.

This solution structure will allow us to start and build all docker images using docker-compose.

Test your containers with docker-compose

Add "docker-compose.yml" file to the project root:

```
version: '3.5'
services:
  mongo:
    image: mongo
    container_name: mongo
    restart: always
    ports:
      - '27017:27017'

  mongo-seed:
    container_name: mongo-seed
    build: ./mongo-seed
    links:
      - mongo

  id:
    container_name: id
    image: afsdemogurba.azurecr.io/afs-demo-id:latest
    volumes:
      - ./app_data/appsettings.json:/app/appsettings.json
    ports:
      - "8088:8088"
    links:
      - mongo

  api:
    container_name: api
    image: afsdemogurba.azurecr.io/afs-demo-api:latest
    volumes:
      - ./app_data/config.js:/usr/src/app/config.js
    ports:
      - "8080:8080"
    links:
      - mongo

  app:
    container_name: app
    image: afsdemogurba.azurecr.io/afs-demo-app:latest
    volumes:
      -
./app_data/app_config.js:/usr/share/nginx/html/app_config.js
    ports:
      - "8089:80"
    links:
      - id
      - api
```

Rebuild and push updated docker containers:

```
./build
```

Run docker-compose solution:

```
docker-compose up -d
```

Observe started containers using "docker ps" command. The output should look similar to this:

```
c:\dev\cloudws\afs-demo-apps>docker ps
CONTAINER ID        IMAGE
COMMAND                  CREATED             STATUS
PORTS                      NAMES
0fc94d91100c
afsdemogurba.azurecr.io/afs-demo-app:latest   "nginx -g
'daemon of…"   11 seconds ago      Up 9 seconds
0.0.0.0:8089->80/tcp       app
053c69f6835b        afsdemogurba.azurecr.io/afs-demo-id:latest
"dotnet id_service.d…"   12 seconds ago      Up 10 seconds
0.0.0.0:8088->8088/tcp     id
5e470d7e8ab5
afsdemogurba.azurecr.io/afs-demo-api:latest   "npm start"
12 seconds ago      Up 10 seconds       0.0.0.0:8080->8080/tcp
api
ae0c1688a729        mongo
"docker-entrypoint.s…"   13 seconds ago      Up 12 seconds
0.0.0.0:27017->27017/tcp   mongo
4e3773228f56        portainer/portainer:latest
"/portainer"             5 weeks ago         Up 4 days
0.0.0.0:9000->9000/tcp     4e3773228f56_portainer// Note that
'portainer' container is not part of this solution.
```

Open following urls in a browser to see it all containers are started:

- http://localhost:8089/
- http://localhost:8088/swagger
- http://localhost:8080/user/admin