Title: How to automatically rebuild a Java Webapp inside Docker containers
Post Body:

I need to provide a POC as argument for the migration of workflows in my current job. Currently we do this:

1. People code on Netbeans
2. People click on build on netbeans
3. Deploy locally
4. Apply code changes
5. Netbeans rebuilds and redeploy the code.

Things to know:

- It seems tomcat detects when a new WAR is put in the directory and hot-deploys it;
- What I aim to automate is not the hot-deploy(since this is already a tomcat feature), but the build process;
- We are using Maven to build the project.
- I'm using docker-compose to get everything up in one single specification.

So far I was able to containerize the Postgres database, the PGAdmin we use and the initial build of the application using a multi-stage Dockerfile.

Tomcat app Dockerfile

```
FROM maven AS buildserver  ADD . /usr/src/mymaven/ WORKDIR /usr/src/mymaven # build the project RUN mvn -f pom.xml clean packa
```

What I am having trouble with is triggering the rebuild when there's code changes (imitating what netbeans does). I can't find in either maven's or netbeans documentation how that detection and triggering works.

I am using volumes to map the app source directory to the container in hopes that it would just work, but I was wrong.

My docker-compose.yml is as follows:

```
version: '3' services:   pgadmin:     container_name: pgadmin     image: dpage/pgadmin4     env_file:       - ../db-postgres/p
```

Any help in coming up with a solution for this is appreciated, as well as any general advice in how to make this workflow/build improve.

UPDATE

I came up with somewhat of a work around, but now I am having problem testing it.

I defined a maven container to work as a build server:

```
FROM maven  ADD . /usr/src/mymaven/ WORKDIR /usr/src/mymaven RUN apt update && apt install entr -y # build the project RUN mvn
```

and now I am defining the entrypoint on the docker-compose.yml:

```
...  buildserver:     container_name: buildserver     build:       context: .       dockerfile: maven-builder.Dockerfile     v
```

But now I am getting an error message when this container gets up:

```
find: ‘src/’: No such file or directory entr: No regular files to watch
```

Which is weird to me as I successfully build the project in the first run, but the entry-point seems to be failing.

Clarification: What I am being asked is come up with a workflow that removes the need to use the deploy from Netbeans (they want everything automatic). I looked around for a Jenkins workflow, but could not really find a way to achieve the desired results.

Accepted Answer: None
Highest Rated Answer:

According to the Netbeans docs, you can bind Maven goals to IDE actions (http://wiki.netbeans.org/MavenBestPractices section **Binding Maven goals to IDE actions**):

> *It's possible to customize the default Maven goal to IDE Action binding from the project's customizer. Right click on the project node and select 'Properties' or use the File/Project Properties main menu item to invoke the Project properties dialog. On the left hand side, select the panel named 'Actions'. The panel lists all available default project actions that can be mapped. When selecting one from the list the textfields in the bottom allow to change the values.*

It looks to me that you should bind the **Build Project** Netbeans action to a specific Maven goal. From this point, it is up to you to come up with a creative solution. You could explore the Maven Exec plugin capabilities and run custom commands during the build proccess (check I want to execute shell commands from Maven's pom.xml). For instance, it should be possible to copy the .war file from target folder to wherever you want on the filesystem, or even execute scripts inside the running container.

PS: *It looks like you are trying to do something quite odd, but I'll assume here it makes sense to you solving this somehow.*