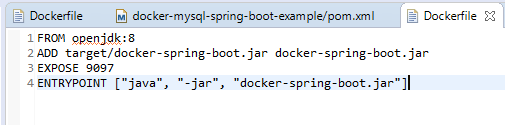
**https://www.youtube.com/watch?v=FlSup\_eelYE**

**https://www.youtube.com/watch?v=fvEWoy1xOvo**

**Docker--Spring Boot**

1. create a Spring boot project from Spring initializer

Application.properties file

🡪Server.port=8085

1. create a jar of that project
2. Add Docker file(inside this we need to mention what type of application we need and what we file we need to add)
3. Cd to that project where dockerfile is present
4. Run Docker in computer

Check docker run or no

🡪docker -v or docker version

🡪docker build -f Dockerfile -t jarfilename(docker-spring-boot) . (. Means current directory)

f🡪filename t🡪tagname

e.g docker build -f Dockerfile -t docker-spring-boot .

It will add the jar in docker container.

1. Check images created

🡪docker images

1. Push application in docker(this is the way we run docker image)🡪container is created

🡪docker run -p 8085:8085 imagename(docker-spring-boot)

e.g docker run -p 8085:8085 docker-spring-boot

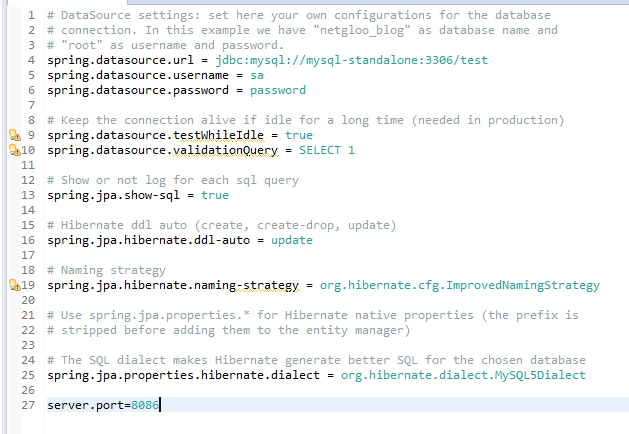
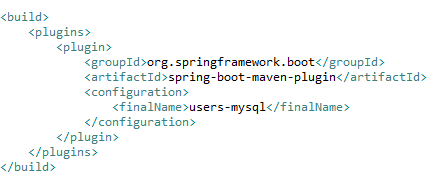
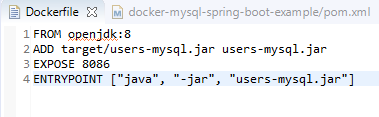
1. Goto browser

🡪localhost:8085/rest/docker/hello

1. **Press cntrl+C in terminal to shut down the image.**
2. **🡪clear**
3. **🡪history**

**Docker--Spring Boot+mySql**

Application.properties



Giving name for jar file in pom file

**Steps**

1. For any thing, goto <https://hub.docker.com/> and type in search there we can see the command.
2. For MySql, in command terminal

🡪docker pull mysql

1. Check docker images if mysql must be there

🡪docker images

1. Start MySQL instance this creates a docker container

Yo chahi docker hub ko website ma diyako(mero ma kam gar6 yo)

🡪$ docker run --name some-mysql -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:tag

Yo chahi tutorial ma bhanayko (mero ma yok am gardaena)

🡪$ docker run --name some-mysql -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -e MYSQL\_DATABASE=databasename -e MYSQL\_USER=sa -e MYSQL\_ PASSWORD= password -d mysql:tag

e.g $ docker run --name mysql-standalone -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=test -e MYSQL\_USER=sa -e MYSQL\_ PASSWORD=password -d mysql:latest

1. No check the container

🡪docker container ls

Here container name is mysql-standalone is created

1. Cd to that project where dockerfile is present
2. Build docker file

Docker build . -t users-mysql( mathiko different6)

1. Check docker images users-mysql must be there.

🡪docker images

Now do this to run docker container

🡪 docker run -p 8086:8086 –-name nameofcontainer(users-mysql) –-link containeralreadycreated(mysql-standalone:mysql) -d imagename(users-mysql)

🡪e.g docker run -p 8086:8086 –-name users-mysql –-link mysql-standalone:mysql -d users-mysql

1. No check the container

🡪docker container ls

1. To check the logs

🡪docker logs mysql-standalone

🡪docker logs users-mysql

1. Goto browser

🡪localhost:8086/all/

1. To stop docker container

🡪docker stop users-mysql

1. Remove particular image

🡪docker image rm users-mysql

1. Remove particular container

🡪docker container rm container\_ID

**Docker Basic Commands**

Check the version of Docker: docker version or docker -v

Check the detailed information on the running/stopped containers: docker info

Docker images can be downloaded from **Docker hub using docker commands**.

Lets **pull an image from docker hub** using pull command.

**Download a image from docker hub**

docker pull <<image name>> e.g. docker pull nginx

##### Docker Commands - Images

Verify the downloaded docker images:docker images ↵

View all the commands that were run with an image via a container: docker history <<Image Name>> ↵

e.g. docker history nginx

Remove Docker Images: docker rmi <<Image Name>> ↵ e.g. docker rmi nginx

**Download and run an image in docker container using run command**

docker run <<Image Name>> ↵ e.g. docker run --name nginxservice -d nginx

*--name --> to specify a name for the running service. In this example, it is nginxservice*

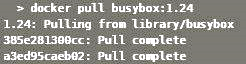
*-d --> to run the service in the background*

##### Docker Version Tag

You must have noticed that on pulling the image from the Registry, the tagged version of the image is displayed:

* Docker pull hello-world
* Using default tag:latest

For example, docker pull busybox:1.24 will download the corresponding version:



##### Docker Search

We can also search for the images in the Docker Hub registry by using docker search subcommand. Let us search for ubuntu images in the Docker Hub, and limit the search result only to 20 because we have more than 2000 images on Ubuntu:

docker search ubuntu | head -20

##### Docker Commands - Container

Now lets run list of commands on the service/ container

**List running containers**

**docker ps**

Know the IP address of the running container:docker inspect <Container Name>

e.g. docker inspect nginxservice

Print the stats for a running Container:docker stats <<Container Name>> ↵ e.g. docker stats nginxservice

Pause the processes in a running container:docker pause <<Container Name >> ↵

e.g. docker pause nginxservice

Unpause the processes in a running container:docker unpause <<Container Name >> ↵

e.g. docker unpause nginxservice

##### Docker Commands - Container (Contd 1...)

Kill the processes in a running container:docker kill <<Container Name >> ↵e.g. docker kill nginxservice

Start the same container:docker start <<Container Name>> ↵ e.g. docker start nginxservice

**Stop the running container**

docker stop <<Container Name >> ↵ e.g. docker stop nginxservice

List all containers (This includes containers in a all states):

We will be able to see the container we just stopped listed here. docker ps -a

Delete a container:docker rm <<Container Name >> ↵ e.g. docker rm nginxservice

**To remove all stopped containers: docker container prune**

Note: Instead of using the Container Name, all the above commands can be executed with the container id as well.

##### Docker Commands - Container (Contd 2...)

Export a container:docker export <<Container Name>> <<file\_Name>>.tar ↵

e.g. Lets run a service using docker run command.

docker run --name newnginxservice -d nginx

docker export newnginxservice > test.tar

Import a container: docker import <<Remote URL/Image Name.tar>> ↵ e.g. docker import test.tar

##### Docker daemon Commands

Stop Docker daemon process:service docker stop

Start Docker daemon process:service docker start

You may not be able to try these 2 commands since you would not have access to root on Katacoda playground.

##### Diagnose Run Issues

In case you are having a problem with downloading the images and running them, please follow these steps to check whether the docker service is running on your system or not:

* Check the running status of docker:service docker status
* Restart Docker service in your system:service docker restart

**Docker Installation**

Prerequisite for installation of Docker on Linux are:

* 64-bit architecture Linux
* Linux kernel must be 3.10 or later

**Here are the steps for installing the community edition in Ubuntu 16.04:**

* Add the GPG key for the official Docker repository to the system:

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

* Now add the Docker repository to APT sources:

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

1. Now update the package database with the Docker packages: $ sudo apt-get update
2. Check for the policy: $ apt-cache policy docker-ce
3. Install Docker: $ sudo apt-get install -y docker-ce
4. Check the installed version of Docker:$ sudo docker --version

**Congratulations!** You have successfully installed Docker version 17.03.0 community edition.

##### Installation Using Automated Script

**Are you feeling this procedure lengthy? There is a shortcut to this process.**

Just run the below command to install Docker

* curl command $ sudo curl -sSL https://get.docker.io/ | sh

##### Uninstall Docker CE

This command is used to uninstall Docker CE package in Ubuntu machine. $ sudo apt-get purge docker-ce

**Step 1 - Create a Dockerfile**

1. Create a new directory DockerExample:mkdir DockerExample.
2. Move the newly created directory:cd DockerExample.
3. Add a file named Dockerfile:vi Dockerfile.

**Dockerfile**

FROM tomcat:jre8-alpine

# For wget to work

RUN apk update \

&& apk add ca-certificates wget \

&& update-ca-certificates

# Copy tomcat server.xml

WORKDIR /usr/local/tomcat

# Start tomcat

CMD ["catalina.sh", "run"]

##### Step 2 - Build a Docker Image

Build a docker image using below command.

All the four steps in the docker file will be executed one by one. You will see a message Successfully Built "Image Id".

docker build -t tomcatimage .

-t --> Tags the name 'tomcatimage' to the newly created image.

**Note**: Do not forget to add a '.' dot at the end of the build command.

You can run the below command to view the complete details of the image. docker inspect tomcatimage

##### Step 3 - Verify Image

Verify the image retrieved by running command. This should list the newly created image. *docker images*

##### Step 4 - Push to Registry

Let us push the created image to Docker hub.

Set up a Docker Hub account. Fill your profile details on the main Docker Hub website and click Sign Up. Activate your account from the email sent.

Go back to Katacoda page and continue with the commands to push the image to docker hub.

1. Login to docker hub using the below command: docker login --username <username>

key in the password once prompted.

1. Command to tag the image with the repository image name:

docker tag tomcat01 <username>/tomcatimage

1. Now let us push the image to the hub: docker push <username>/tomcatimage

Now the image 'tomcat01' is available in the docker hub.

##### Step 5 - Pull from Registry

Now lets learn to pull an image from the docker hub.

Open a new Katacoda session and type the following command.

1. Login to docker hub using the same docker login command
2. Type the below command to pull the added image: docker pull <username>/tomcatimage
3. Verify if the image is available by running docker command: docker images

##### Step 6 - Run Image

To run the tomcat image, type the below command

*docker run --name tomcatRunner -p 8080:80 -d tomcatimage*

tomcatRunner - container name

8080 - port of host machine

80 - port of container

-d - Run the daemon process in the background

##### Step 7 - Verify Container

Now let us verify if the new container created is running. Type in command *docker ps*

This will list the newly created container.