*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Feature Document

User Story #547

**Team Member:**

Andres Villa

**Product Owner(s)**:

Masoud Sadjadi

**Mentor(s)**:

Frank Hernandez

**Instructor**: Masoud Sadjadi

# **User Story**

## **Use Case**

**Use Case ID:(MJ-547) Implement Docker CI**

Details:

Actor: Developer

Pre-conditions:

1. Developer has pushed code to github repo to a respective branch

Description:

1. Use case begins when github recognizes a push has occurred to the repository
2. Docker hub will recognize that a push has a occurred, and begin a build
3. Docker will pull the repository using the latest commit
4. Docker will begin building an image using the Dockerfile from the commit

Post-conditions:

1. The number of triggered builds in Docker goes up by 1
2. The build will complete, and report a failure or a success

Alternative Courses of Action:

1. After step D.4, the branch may be automatically merged if the build was a success

Exceptions:

N/A

Related Uses Case:

N/A

Decision Support:

Frequency: Low. Estimated average of one addition per application execution.

Criticality: Low. Allows the Member to add data into the social aspect integration features.

Risk: Medium. Implementing this use case employs standard web-based technology.

Constraints:

1. Usability:
   1. No training time  required.
   2. User guide should provide more in-depth information on Docker if necessary.
2. Reliability:
   1. Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
   2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
3. Performance:
   1. The build should be triggered within 20 seconds
4. Supportability:
   1. The docker build should support any branch on the repository

Modification History:

Owner: Andres Villa.

Initiation date: 01/28/2016

Date last modified: 01/28/2016

## 

## **Sequence Diagram**

547-DockerSequenceDiagram.png

## 

## 

## **Class Diagram**

…(Not used for Docker)

## 

**Unit & Integration Tests**

**Test Case 1 (Sunny Day)**

**ID: MJ-SD-547-01**

**Purpose**

* Ensure that the Docker image contains the appropriate services

**Precondition**

* Docker build was completed successfully

**Input**

* docker run -P -d andresvilla/mobilejudge8:dockerBranch

**Expected Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80> loads mobile judge website

**Actual Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80> loads mobile judge website

**Test Case 2 (Sunny Day)**

**ID: MJ-SD-547-02**

**Purpose**

* Ensure that the Docker does not hold any true database information

**Precondition**

* Docker build was completed successfully

**Input**

* docker run -P -d andresvilla/mobilejudge8:dockerBranch

**Expected Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80>/phpmyadmin/ and logging in, shows that there is no database in phpmyadmin

**Actual Result**

* “docker ps” - will show the running container along with a mapping of its ports
* connecting to localhost:<mapped Port to :80>/phpmyadmin/ and logging in, shows that there is no database in phpmyadmin

**Test Case 3 (Rainy Day)**

**ID: MJ-RD-547-01**

**Purpose**

* Ensure that Docker image does not update if a bad build is triggered

**Precondition**

* Code was successfully pushed to GitHub

**Input**

* Non functioning code has been pushed to GitHub repository

**Expected Result**

* Docker container will not update the image with the latest broken build

**Actual Result**

* Docker container will not update the image with the latest broken build

**Test Case 4 (Rainy Day)**

**ID: MJ-RD-547-02**

**Purpose**

* Ensure that Docker Image is independent of Docker container

**Precondition**

* Code was successfully pushed to GitHub

**Input**

* Docker container is started up
* Entire docker container is deleted (sudo rm -rf \*) from root directory

**Expected Result**

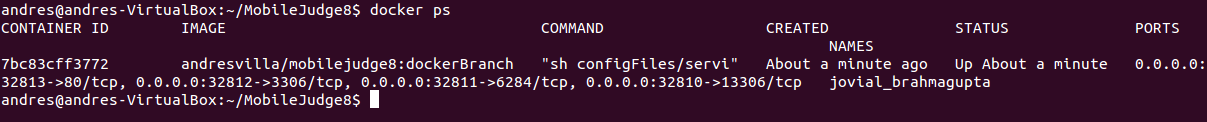
* Docker image will remain functional and identical to image prior to test

**Actual Result**

* Docker image will remain functional and identical to image prior to test

## **User Guide**

(ports forwarding for container will be dispayed as shown below, localhost:32813 will be the mapping for nginx)



walkthrough for Docker and Docker CI

<https://docs.google.com/presentation/d/1fiULnU9GVffffVlnkDzTjy4CwJ3Z0nQAfr0dWcrs57A/edit#slide=id.gef5bcc529_0_170>

Ensure that docker is installed:

“sudo apt-get install docker”

To run a container with an image of your current build In the folder containing the Dockerfile:

“docker build -t <any name> .”

Once this completes successfully, run:

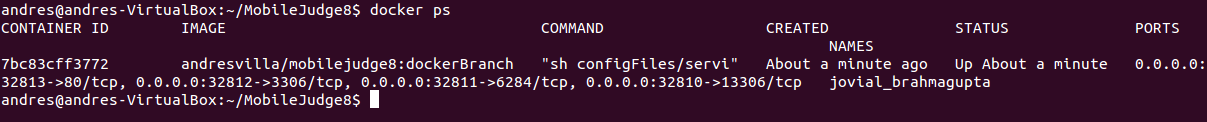
“docker run -d -P <prevName>

-d = runs the container in the background

-P = forwards any ports specified in the dockerfile from the container to the host machine

If you would like to get access to the container run

“docker ps” //this will show you the container name (jovial\_brahmagupta in the image below)



“docker exec -it <Container\_id or Names> bash”// will give u access to a bash shell in the container

If you would like to ensure that the required services are running, then run

“cd /”

“sh configFiles/services.sh”