**Template Docker – To start**

Install docker en Jenkins + voorbeeld van app die door docker lokaal op PC wordt gepushed

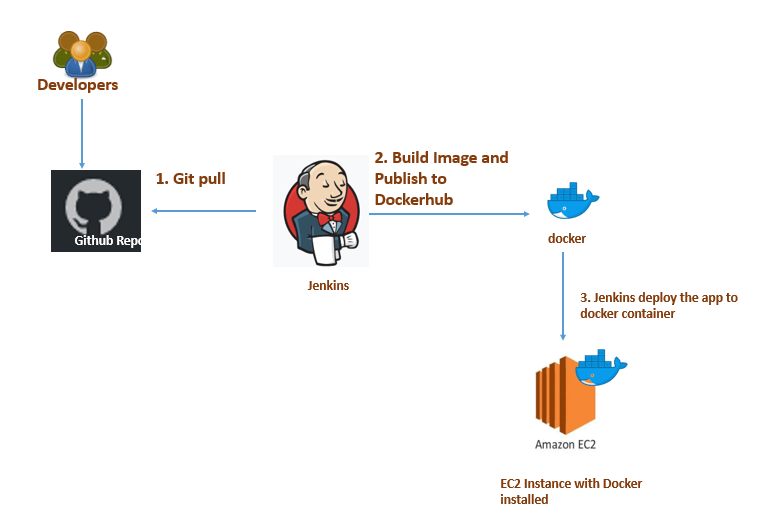
<https://www.youtube.com/watch?v=Y160f9-xZY4>

Manuel – Eerste aanzet jenkins en docker laten samenwerken. Einddoel python3 laten installeren door de klanten via docker omgeving op de hosts Pc’s.

<https://www.youtube.com/watch?v=13FpCxCClLY>

<https://github.com/devops4solutions/CI-CD-using-Docker>

# CI/CD using Jenkins and Docker



In this blog, we will explore the CI/CD process using Jenkins and Docker. We will deploy the sample java application using Docker container. We will be deploying war file in a tomcat container. At last,this process will help us in achieving continuous integration and continuous deployment for your application inside a container

### Prerequisite:

1. Two AWS EC2 instance ( one is Jenkins Server and another is tomcat server)
2. Follow [Jenkins](https://devops4solutions.com/jenkins-installation-on-linux/)Installation article if not already installed
3. Integration between [Jenkins and Docker](https://devops4solutions.com/integrate-jenkins-with-docker/)
4. Make sure ssh connection is already setup between for jenkins user between two AWS EC2 Instances.
5. Docker Pipeline Plugin

### Agenda:

1. Create a sample application in java
2. Compile the project using maven
3. Package the project in war file
4. Create a Dockerfile which contains tomcat server
5. Push the code in the [github](http://git@github.com:devops4solutions/CI-CD-using-Docker.git)
6. Create a pipeline job in Jenkins and trigger the build

### Sample Application in Java

Firstly, create a sample registration and login page in [jsp](https://github.com/devops4solutions/CI-CD-using-Docker/tree/master/src/main/webapp).

### Compile and package the project using Maven

Created a pom.xml file to compile and package the project in war file. Clone the git [repo](https://github.com/devops4solutions/CI-CD-using-Docker.git)and run the mvn command, this will generate a war file under target folder

git clone <https://github.com/devops4solutions/CI-CD-using-Docker.git>  
mvn package

Now we have generated the war file and we can deploy it on our tomcat server. In general, we used to install tomcat and java on the server and then copy the war file under webapps folder and restart the service either manually or using Ansible

Now, we will see how we can deploy the tomcat application inside a docker container.

We will use the image of the tomcat software which has inbuilt java and will create the tomcat container using our custom Dockerfile

### Dockerfile

We will create a Dockerfile using below instructions

1. `FROM tomcat:latest` — This is the base image that we will be using
2. You can add Labels in your docker images and provide who is the Maintainer of the Dockerfile
3. Now we will add our generated war inside a tomcat/webapps folder which will be present inside a Docker container
4. Expose the port — This means that your tomcat server inside a docker container can be access on this port
5. At last, we are starting the tomcat service using catalina.sh file

FROM tomcat:latest

LABEL maintainer="Nidhi Gupta"

ADD ./target/LoginWebApp-1.war /usr/local/tomcat/webapps/

EXPOSE 8080

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

### Jenkins Pipeline

Now we will write [Jenkinsfile](https://github.com/devops4solutions/CI-usingAnsible/blob/master/Jenkinsfile)to build an image using the Dockerfile which we have created above

pipeline {  
 agent any  
   
 tools  
 {  
 maven "Maven"  
 }  
 stages {  
 stage('checkout') {  
 steps {  
   
 git branch: 'master', url: '[https://github.com/devops4solutions/CI-CD-using-Docker.git'](https://github.com/devops4solutions/CI-CD-using-Docker.git%27)  
   
 }  
 }  
 stage('Execute Maven') {  
 steps {  
   
 sh 'mvn package'   
 }  
 }

stage('Docker Build and Tag') {  
 steps {  
   
 sh 'docker build -t samplewebapp:latest .'   
 sh 'docker tag samplewebapp nikhilnidhi/samplewebapp:latest'  
 //sh 'docker tag samplewebapp nikhilnidhi/samplewebapp:$BUILD\_NUMBER'  
   
 }  
 }  
   
 stage('Publish image to Docker Hub') {  
   
 steps {  
 withDockerRegistry([ credentialsId: "dockerHub", url: "" ]) {  
 sh 'docker push nikhilnidhi/samplewebapp:latest'  
 // sh 'docker push nikhilnidhi/samplewebapp:$BUILD\_NUMBER'   
 }  
   
 }  
 }  
   
 stage('Run Docker container on Jenkins Agent') {  
   
 steps   
 {  
 sh "docker run -d -p 8003:8080 nikhilnidhi/samplewebapp"  
   
 }  
 }  
 stage('Run Docker container on remote hosts') {  
   
 steps {  
 sh "docker -H ssh://jenkins@172.31.28.25 run -d -p 8003:8080 nikhilnidhi/samplewebapp"  
   
 }  
 }  
 }  
 }

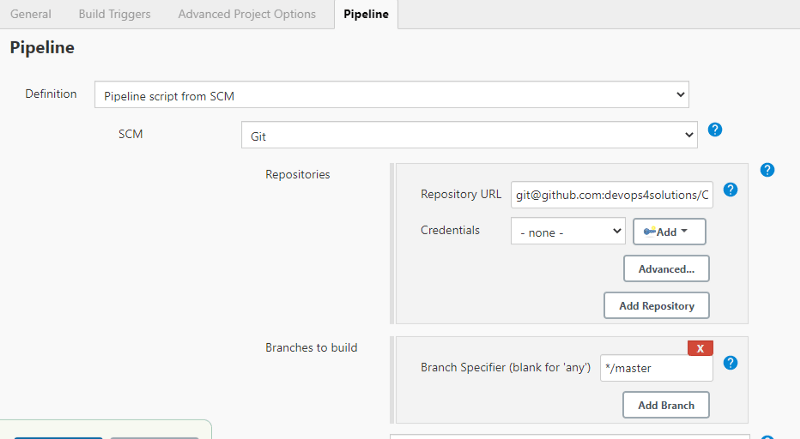
### Configure a Jenkins Pipeline project

* Create a pipeline job in Jenkins

Afbeelding met tekst

Automatisch gegenereerde beschrijving

* Configure gitrepo. I have used ssh url because my jenkins is already [integrated with github](https://devops4solutions.com/setup-ssh-between-jenkins-and-github/)

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Click on Save and trigger the pipeline.

Afbeelding met tafel

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

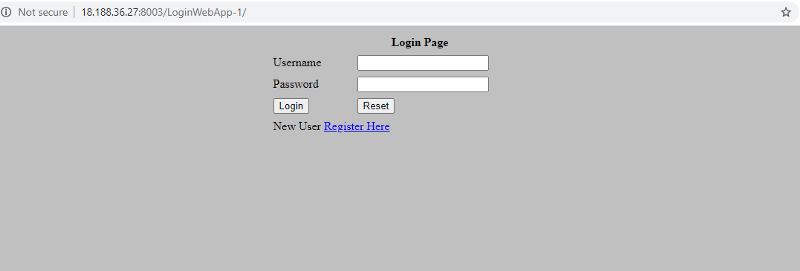
Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijving

Now, you can browse your application by using the url [http://yourip:8080/LoginWebApp-1/](http://18.222.127.218:8080/LoginWebApp-1/) and you will see the login page as shown below:



Congratulations,you have successfully setup the CI/CD process using Jenkins and Docker. You can also integrate your job with [sonarqube](https://devops4solutions.com/jenkins-sonarqube-integration/)to check the code quality and code coverage. You have successfully deployed your first tomcat application inside a Docker container