# Syllabus for the Jenkins course

## Main goal

The course aimed to attain the intrinsic knowledge of the basic DevOps toolkit. At the end all the students would evolve the Jenkins pipeline for getting the source code from the GIT test public repository, building the artifacts using Maven, fetching them to the tomcat-based Docker container, putting (pushing) the container image to the Docker Hub and installed Sonatype Nexus repository, getting (pulling) that image from the registry and running the container with the port exposing on the local Docker.

## Initial state

Every student is supplied with the credentials for the virtual course PC and could reach it via HTTP + SSH and has an account in the DockerHub (<https://hub.docker.com/>)

## Preliminary items

1. Install the Sonatype Nexus

Reach the environment via the SSH and run the commands below:

* ***cd /opt/*** (changing directory)
* ***sudo wget*** [***https://download.sonatype.com/nexus/3/latest-unix.tar.gz***](https://download.sonatype.com/nexus/3/latest-unix.tar.gz)(downloading the latest Nexus package )
* ***sudo tar -xvzf latest-unix.tar.gz*** (unpacking the archive under the superuser)
* ***sudo ln -s /opt/nexus-3.12.0-01/bin/nexus /etc/init.d/nexus*** (adding a symbol link to the services)
* ***cd /etc/init.d*** (changing directory)
* ***sudo update-rc.d nexus defaults*** (adding the service to autostart)
* ***sudo service nexus start*** (starting the service)
* ***sudo vi /opt/sonatype-work/nexus3/etc/nexus.properties*** (open the configuration and uncomment the port variable with setting 32000:

# Jetty section

**application-port=32000**  
# application-host=0.0.0.0  
# nexus-args=${jetty.etc}/jetty.xml,${jetty.etc}/jetty-http.xml,${jetty.etc}/jetty-requestlog.xml  
# nexus-context-path=/

and save the file)

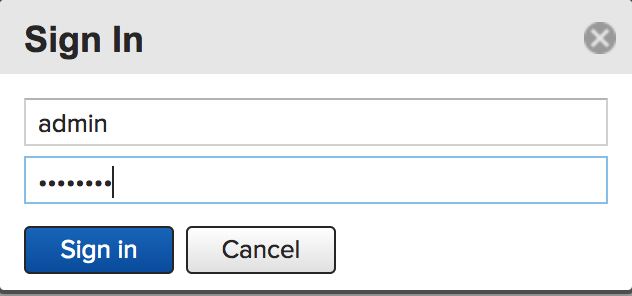
* ***sudo service nexus restart*** (restarting the service)

1. Create the docker and raw repository in the Nexus

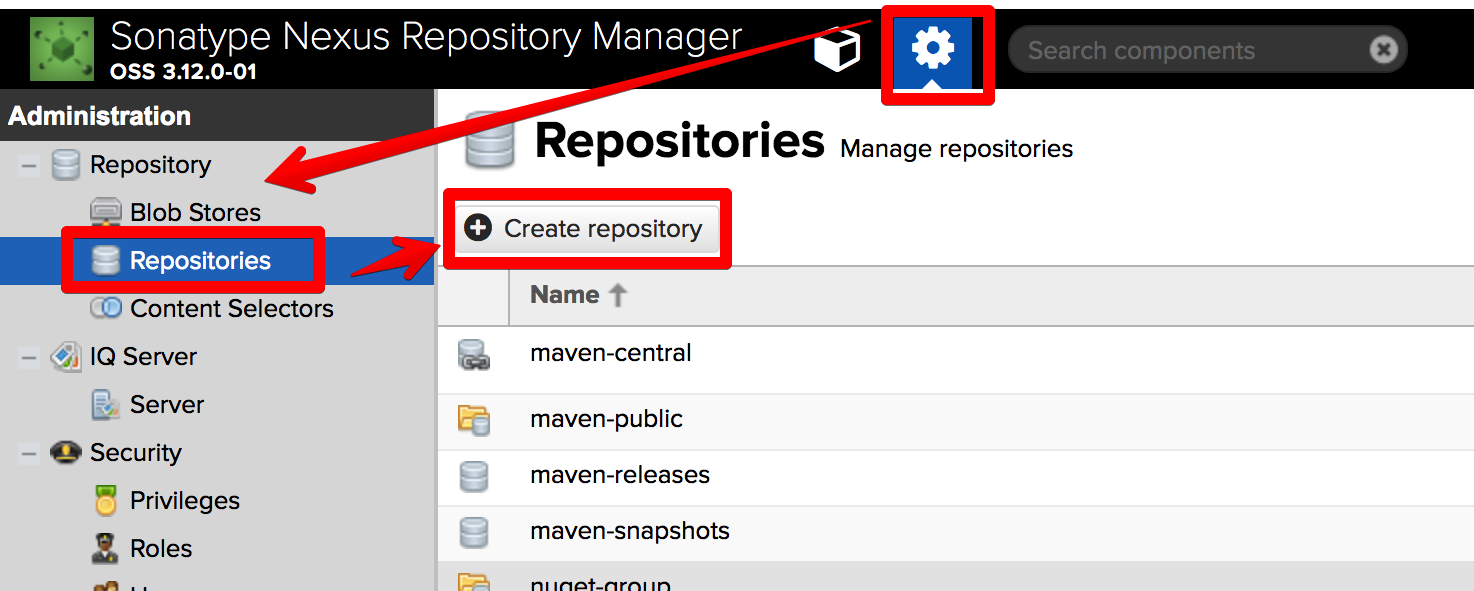
* Go to the *http://<your course PC IP>:32000/*
* Sign in under the default user and password:

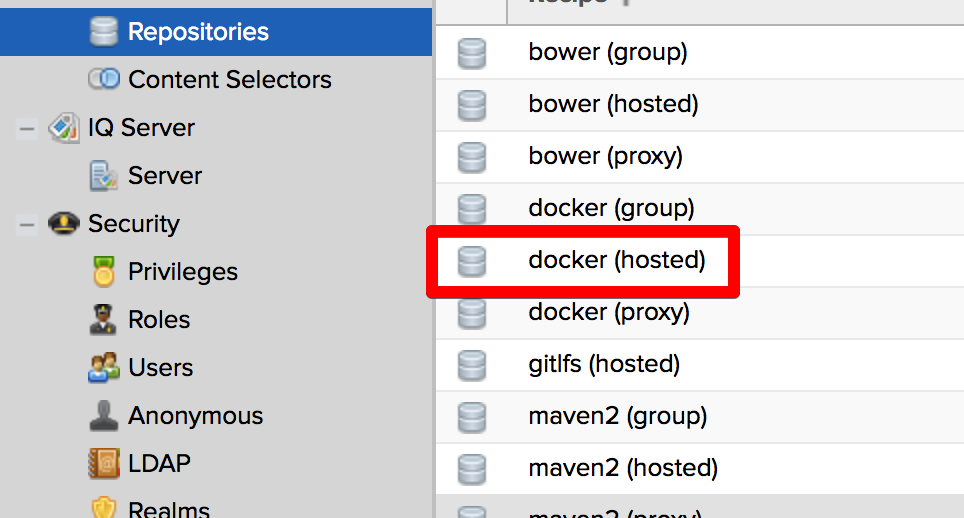


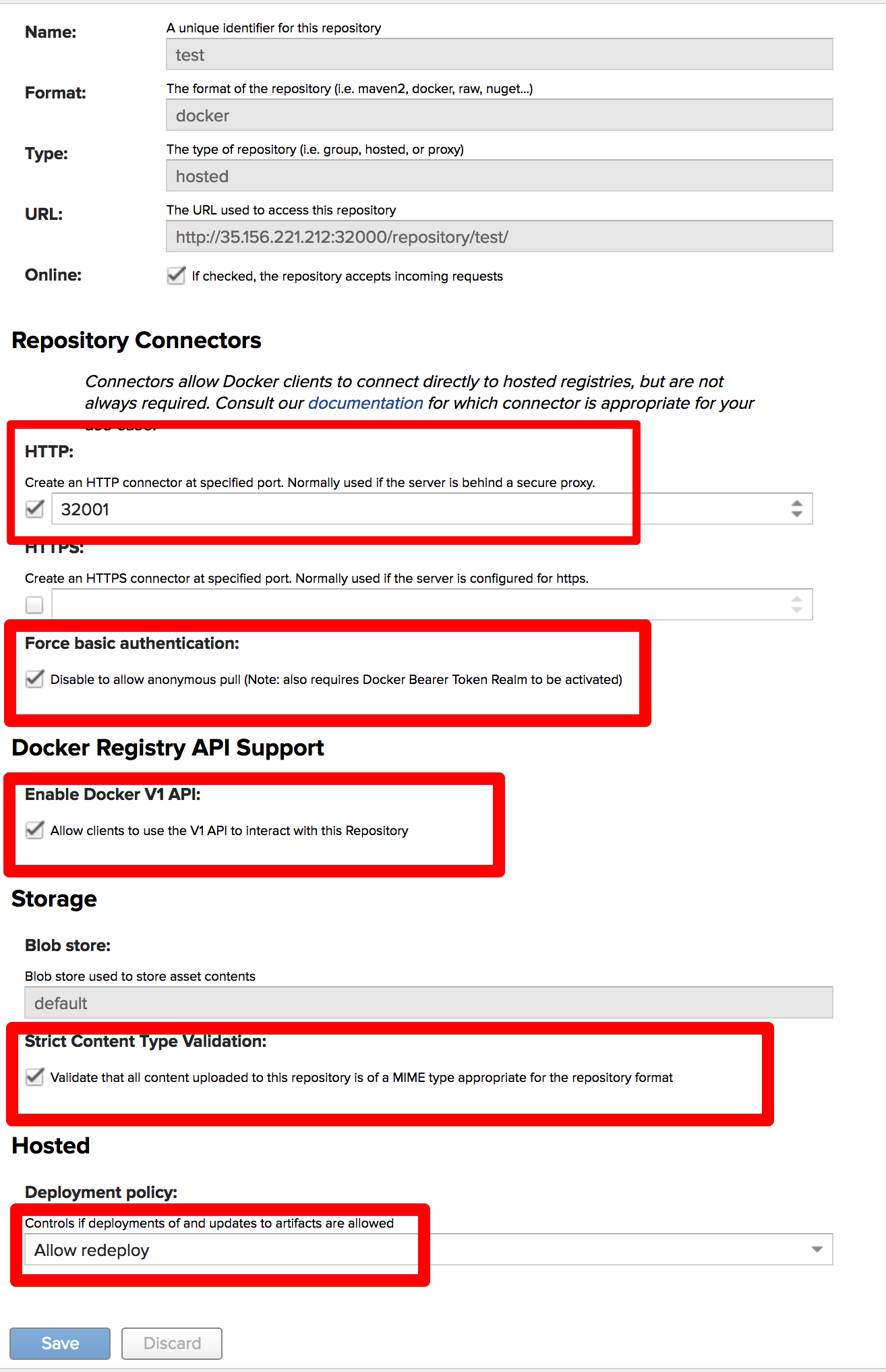
**admin/admin123**



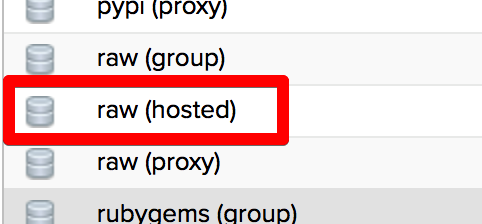
* Create the docker repository for images:

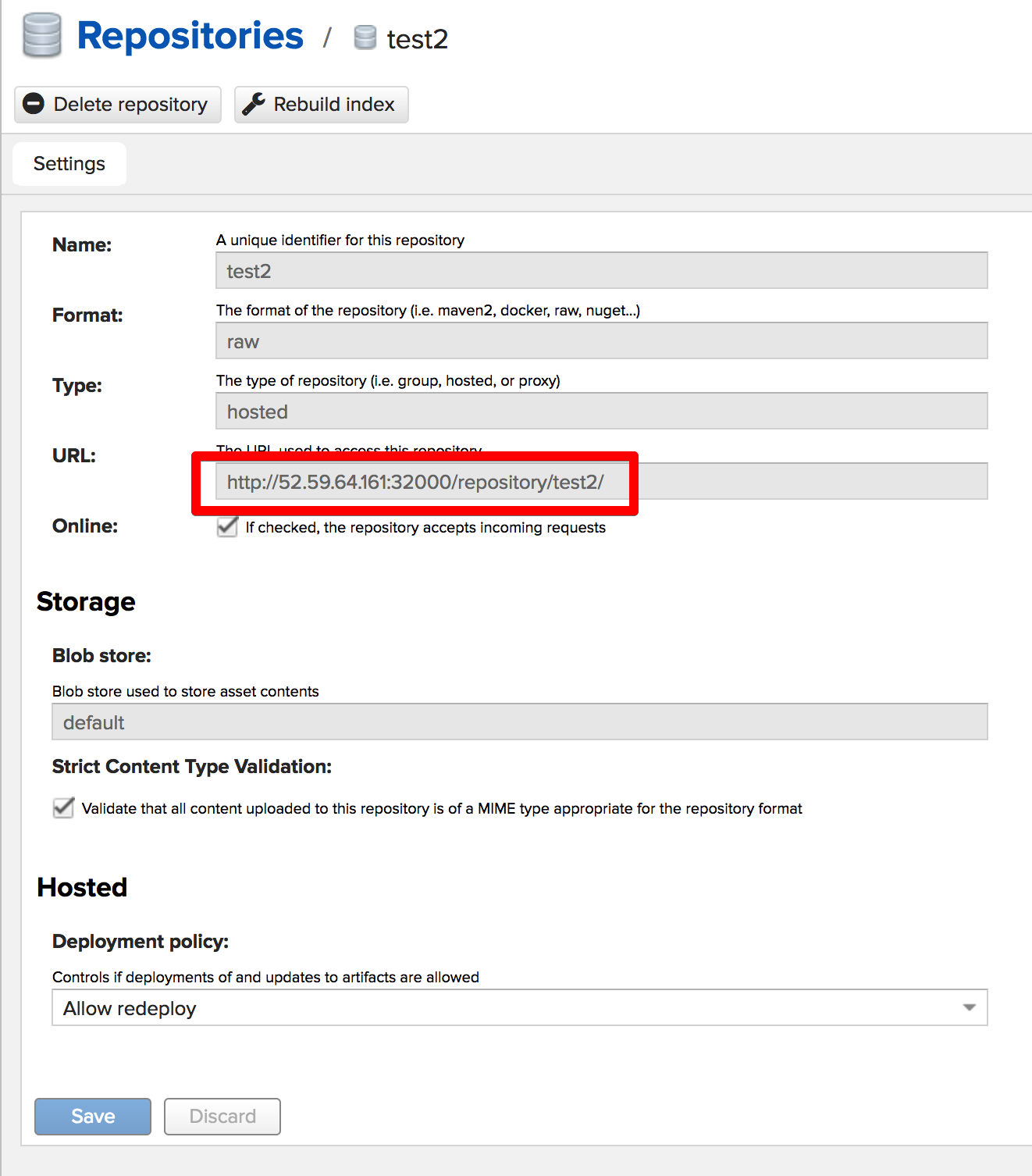






* Create the raw repository for \*.war artifacts:





1. Set Docker to allow http repository

* Create the docker daemon override config file:

***sudo vi /etc/docker/daemon.json***

and add there the configured docker repository (please copy-paste):

***{  
 "insecure-registries" : ["127.0.0.1:32001"]  
}***

* Restart the Docker service using:

***sudo service docker restart***

1. Pair the docker with the private repository:

***docker login -u 'admin' -p 'admin123' 127.0.0.1:32001***

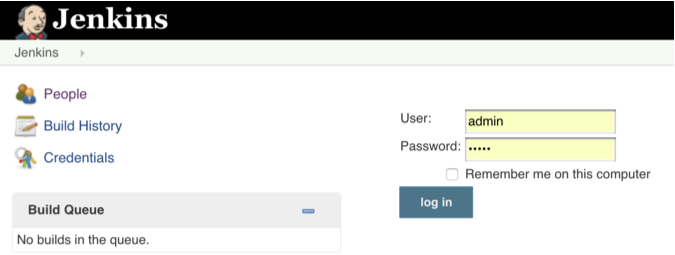
WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store  
  
Are you sure you want to proceed? [y/N] **y**  
**Login Succeeded**

1. Get to the course Jenkins environment using the following credentials:

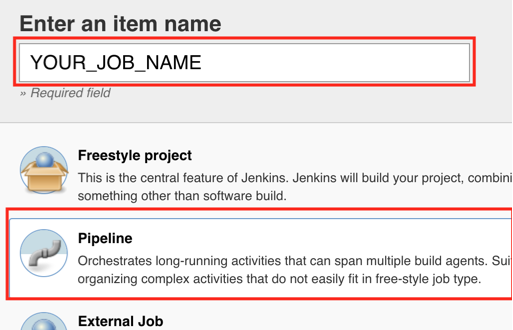
*http://<your course PC IP>:8080/*

User: *admin*

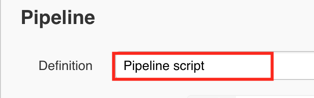
Password: *admin*



1. Create a Jenkins Pipeline job:



1. Set the pipeline definition as “Pipeline Script” and start creating the code



## Create the Groovy pipeline with the following stages:

1. Stage ‘Preparation’:

* Define Maven home variable and connect it to the M3 alias from Global Tool Configuration
* Source the course test code from the public GIT repository: https://github.com/zivkashtan/course.git

1. Stage ‘Creating Package’:

* Using the predefined in stage 1 home variable, create the package with maven
* Check the result \*.war files. The files could be found via the SSH CLI on the filesystem:

***sudo su - jenkins***

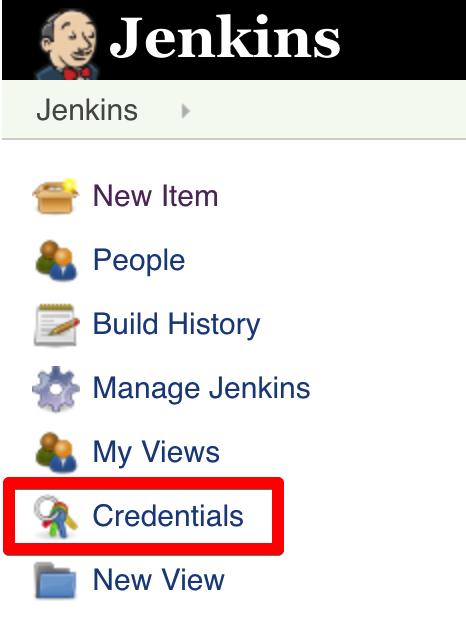
***ls -ltr ~/workspace/<YOUR\_JOB\_NAME>/web/target/***

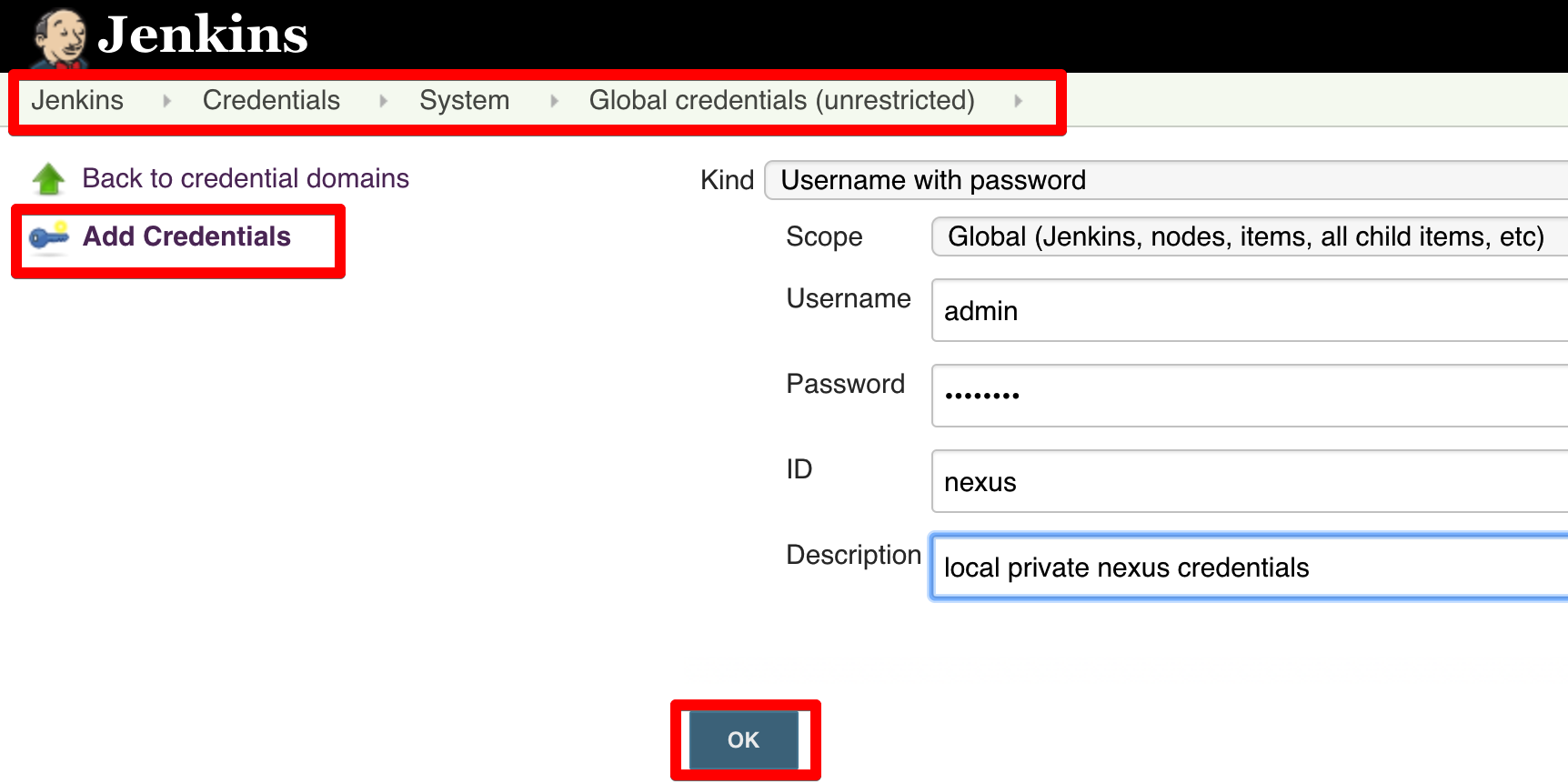
***<supervise the file>***

***exit***

1. Stage ‘Publish artifacts to Nexus registry’:

* Using the url from the **RAW** repository (test2 in this example), change the external IP to the loop one (127.0.0.1) and consider the Nexus URL value for the next step
* Add the Nexus username and login to the Jenkins Credentials(admin/admin123):





* and evoke the deploy deploy with the stored credentials:

***withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'nexus',***

***usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {***

***echo sh (returnStdout: true, script: "curl -v -u ${USERNAME}:${PASSWORD} --upload-file <PATH\_TO\_YOUR\_WARFILE> <NEXUS\_REPOSITORY\_URL>/<WARFILE\_NAME>”)***

***}***

1. Stage ‘Creating Dockerfile’:

On that stage we need to take the existed test tomcat-based docker image within the JRE8 installed and inject our \*.war artifact to the default tomcat folder for the artifacts ( tomcat:8.0.20-jre8 has that directory there: /usr/local/tomcat/webapps/). Please do bear in mind that the tomcat-based image is a part of the official docker repository and it is no need to create it from the scratch. The repository could be supervised by drilling down the following link and typing ‘tomcat’ in the search form:

<https://hub.docker.com/explore/?page=1>

The stage items are:

- Create the Dokerfile for creating the product image based on ‘tomcat:8.0.20-jre8’

- Add to the Dokerfile the \*.war file with the product: destination folder in the tomcat is

1. Stage ‘Docker build image’:

* Run the docker build command with tag (***<your\_dockerhub\_username>/time-tracker***) to create an image in the local docker

1. Stage ‘Ansible push image’:

* Create/modify the example /home/ubuntu/hosts inventory file via SSH CLI to let the ansible work with the localhost. The file looks like the following example:

***[local]***

***localhost ansible\_connection=local***

* Create/modify the example /home/ubuntu/docker\_push\_playbook.ym file via SSH CLI to trigger the ansible do a docker login to your dockehub repository and push there the built image:

***- name: Ansible Docker PUSH step***

***hosts: localhost***

***tasks:***

***- name: Log into Docker Hub and force re-authorization***

***docker\_login:***

***username: <YOUR\_DOCKERHUB\_USERNAME>***

***password: <YOUR\_DOCKERHUB\_PASSWORD>***

***email: <YOUR\_DOCKERHUB\_EMAIL>***

***reauthorize: yes***

***- name: push an image***

***docker\_image:***

***name: <YOUR\_DOCKERHUB\_USERNAME>/time-tracker***

***tag: latest***

***push: yes***

***- name: push an image to the local Nexus registry  
 docker\_image:  
 name: <YOUR\_DOCKERHUB\_USERNAME>/time-tracker  
 repository: 127.0.0.1:32001/time-tracker  
 tag: latest  
 push: yes***

* Run command via Jenkins to apply the playbook:

***ansible-playbook /home/ubuntu/docker\_push\_playbook.yml -i /home/ubuntu/hosts***

1. Stage ‘Ansible pull and run image’:

* Create/modify the example /home/ubuntu/docker\_pull\_run\_playbook.ym file via SSH CLI to trigger ansible to pull your dockehub repository image and run it with the port exposing:

***- name: Ansible Docker step***

***hosts: localhost***

***tasks:***

***- name: pull an image***

***docker\_image:***

***name: <YOUR\_DOCKERHUB\_USERNAME>/time-tracker:latest***

***- name: Start a container***

***docker\_container:***

***name: time-tracker***

***image: <YOUR\_DOCKERHUB\_USERNAME>/time-tracker:latest***

***state: started***

***ports:***

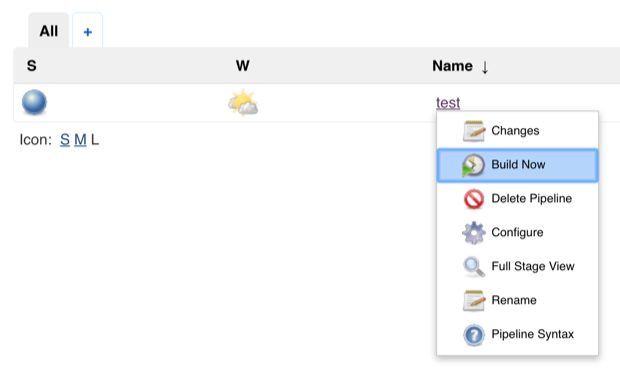
***- "80:8080"***

* Run command via Jenkins to apply the playbook:

***ansible-playbook /home/ubuntu/docker\_pull\_run\_playbook.yml -i /home/ubuntu/hosts***

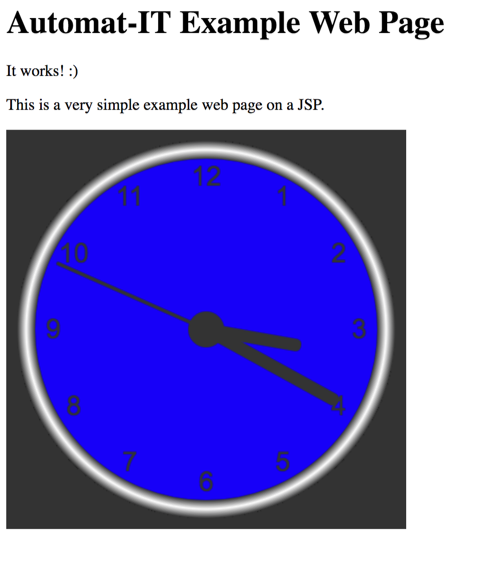
## Run the Groovy pipeline and analyze the results :

1. Run the pipeline and see the stdout:

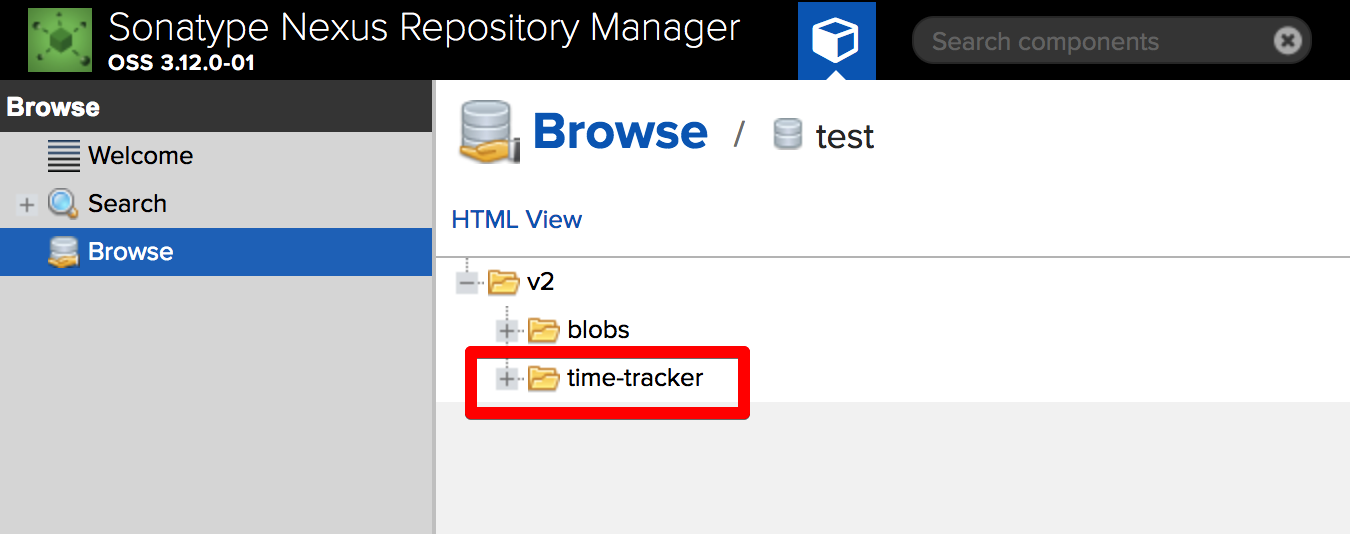


1. Verify the results using the a WEB browser:

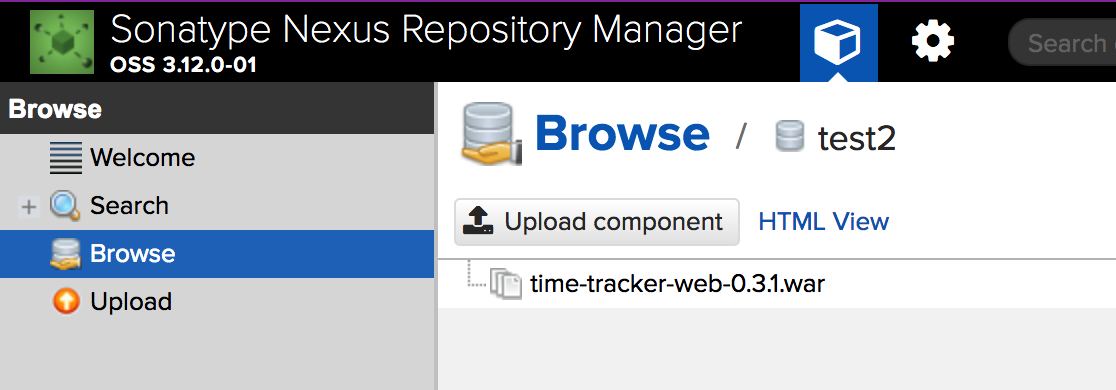
* Go to *http://<your course PC IP>:80/time-tracker-web-0.3.1/*
* You should see the tremendous Automat-IT example web page:



1. Check the Docker image pushed to the local Nexus docker hosted private repository:



1. Check the war artifact pushed to the local Nexus raw hosted private repository:



1. Switch under the jenkis user and modify the source code:

***sudo su - jenkins***

***vi ./workspace/<YOUR\_JOB\_NAME>/web/src/main/webapp/index.jsp***

Put ‘green’ instead of ‘blue’:

***...***

***function drawFace(ctx, radius) {***

***var grad;***

***ctx.beginPath();***

***ctx.arc(0, 0, radius, 0, 2\*Math.PI);***

***ctx.fillStyle = 'green';***

***ctx.fill();***

***grad = ctx.createRadialGradient(0,0,radius\*0.95, 0,0,radius\*1.05);***

***...***

1. Go to Jenkins pipeline and commit out in the Stage ‘Preparation’ the line with the git clone:

***stage('Preparation') {***

***// Get some code from a GitHub repository***

***//git 'https://github.com/zivkashtan/course.git';***

***// Get the Maven tool.***

1. Run the Pipeline and verify the results using the a WEB browser:

* Go to *http://<your course PC IP>:80/time-tracker-web-0.3.1/*
* You should see the fascinating Automat-IT example web page with green color:

