Deploy Application Using Jenkins, Tomcat Server, and PollSCM

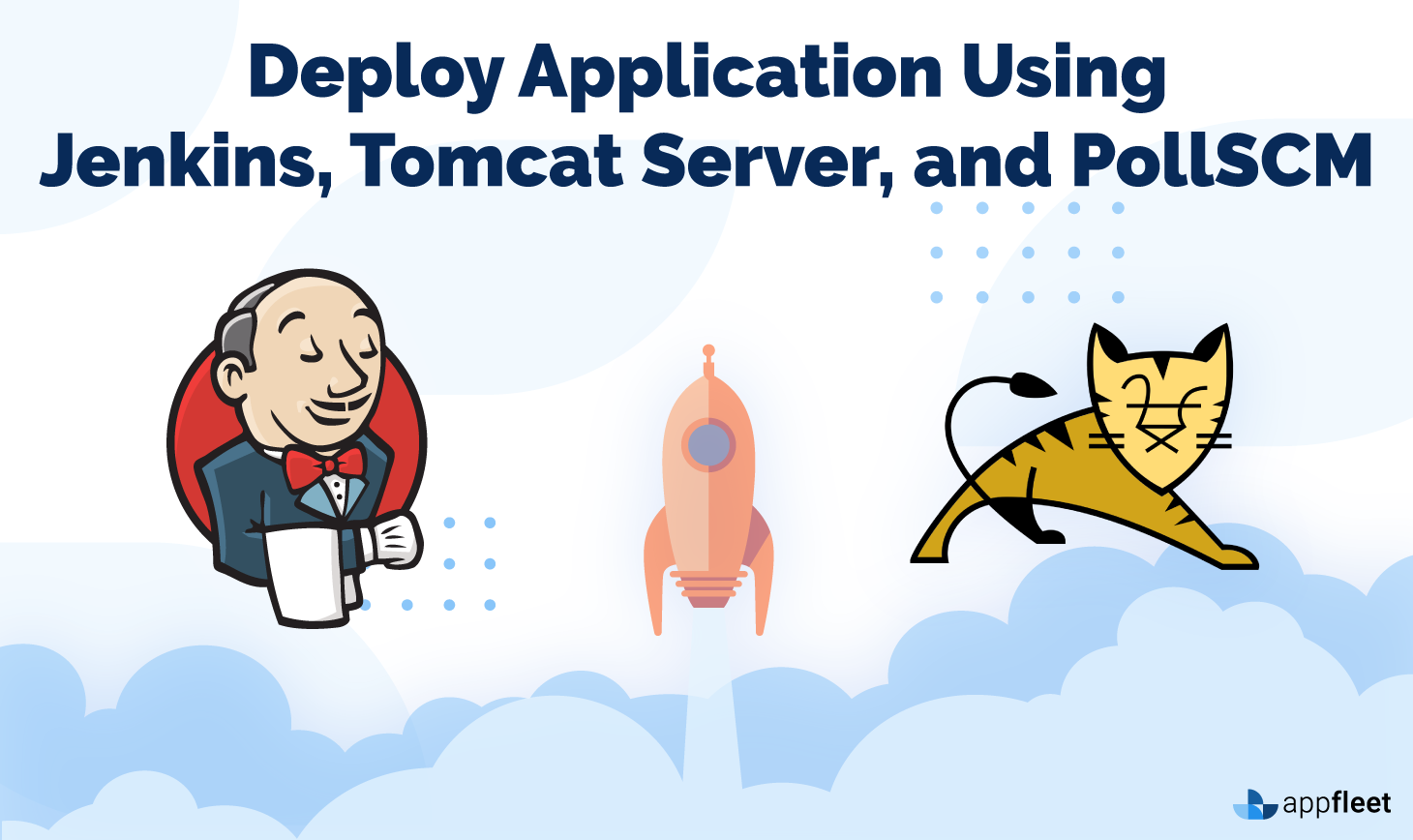
[](https://medium.com/@appfleet?source=post_page-----4d358fa0d6ee----------------------)

[appfleet team](https://medium.com/@appfleet?source=post_page-----4d358fa0d6ee----------------------)

Follow

[Apr 23](https://medium.com/appfleet/deploy-application-using-jenkins-tomcat-server-and-pollscm-4d358fa0d6ee?source=post_page-----4d358fa0d6ee----------------------) · 7 min read





In this guide, we are going to use Docker to configure Jenkins and Tomcat to achieve Continuous Integration/Continuous Deployment.

We will install and set up Tomcat and Jenkins using Docker, and thereafter we are going to create a Jenkins Job to deploy your Java application over the Tomcat server.

This guide is assuming that you already have basic knowledge of Docker and Docker is up and running on your host machine. So let’s begin with the guide.

**Jenkins Set up**

You can spin up a Docker container using the [Jenkins](https://hub.docker.com/r/jenkins/jenkins/) image.

$ docker container run -dit -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock -v /data/jenkins/jenkins\_home:/var/jenkins\_home --name jenkins\_container jenkins/jenkins

We are running Jenkins on port 8080. You can see the running container here:

$ docker ps  
   
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
ce46107c9f83 jenkins/jenkins:lts "/sbin/tini” Up 1 minutes Up 1 minutes 0.0.0.0:8080->8080/tcp jenkins\_container

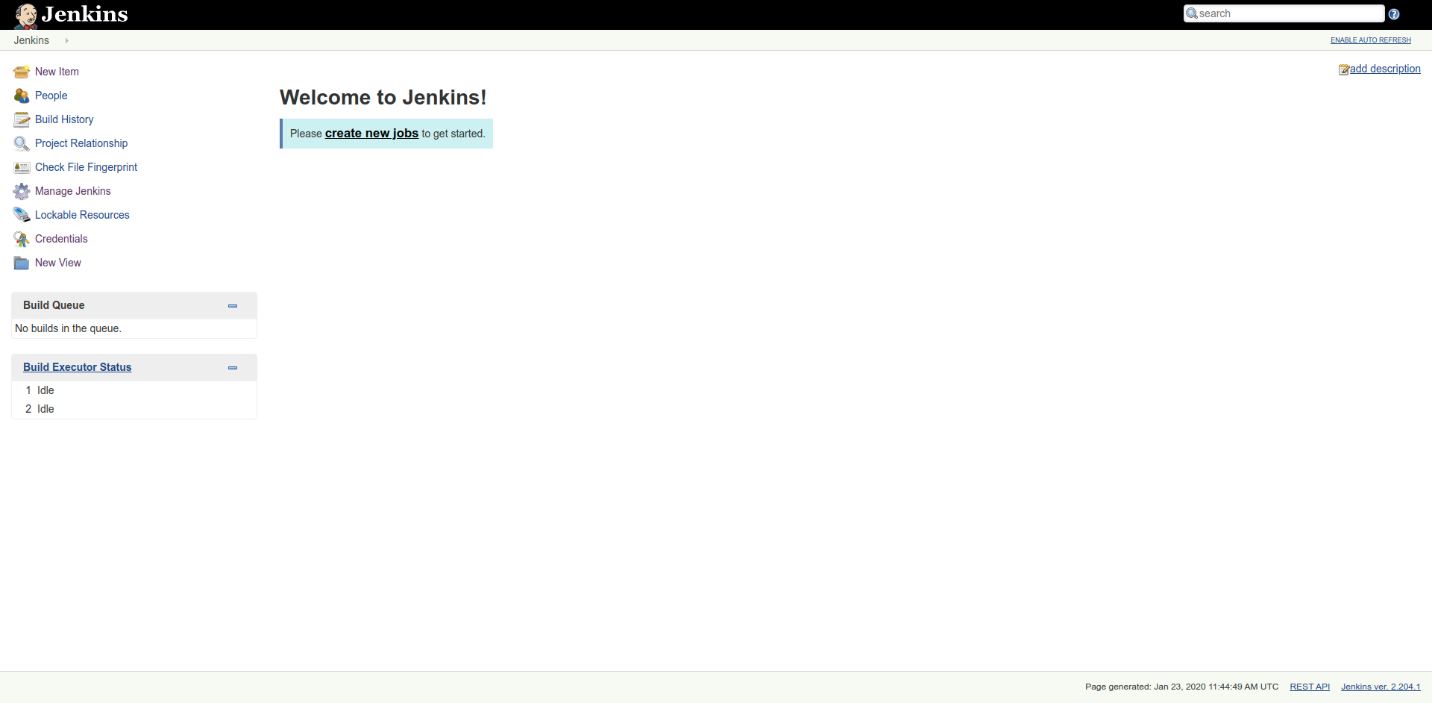
To extract the Jenkins password from Jenkins’s container, you can run below-mentioned command:

$ docker exec -it jenkins\_container cat /var/jenkins\_home/secrets/initialAdminPassword  
   
gSzHTGfbtXSpcBXMyJ6gzmNfNH7BjhK23

Copy the admin password from your terminal and open http://your-ip-addr:8080 in your preferred browser. Next, run the initial setup wizard and select *Recommended Plugins*. Jenkins will install all the necessary plugins.

If everything goes well, you will land on the Jenkins’ dashboard area, which looks something like this:





**Tomcat Set up**

Now, it’s time to spin the Tomcat container. For this, we are going to use CentOS, which will enable us to set up a customized Tomcat server. You can also use tomcat image as per your choice, however for this guide we are going to stick to the plan. So let’s start by pulling the latest Centos image from Docker Hub:

$ docker pull centos  
   
Using default tag: latest  
latest: Pulling from library/centos  
8a29a15cefae: Pull complete   
Digest: sha256:fe8d824220415eed5477b63addf40fb06c3b049404242b31982106ac204f6700  
Status: Downloaded newer image for centos:latest  
docker.io/library/centos:latest

To run the container from the pulled image in detached mode, use the command below:

$ docker run -dit --name tomcat\_container -p 8888:8080 centos

As Jenkins has already occupied port 8080 on the machine, here we have opened port 8888 for Tomcat in spite the default port being 8080.

To install Tomcat in your Centos container, follow the steps:

$ docker exec -it eca291a36127 /bin/bash

Go to /opt directory, in your tomcat\_container:

[root@eca291a36127 /]# cd opt/

Open <https://tomcat.apache.org/download-80.cgi> on your browser. Find “*Core*” and download the Tomcat binary. At the time of writing this article, the current version is [8.5.50](http://apachemirror.wuchna.com/tomcat/tomcat-8/v8.5.50/bin/apache-tomcat-8.5.50.tar.gz).

[root@eca291a36127 /]# yum install wget  
[root@eca291a36127 /]# wget <http://apachemirror.wuchna.com/tomcat/tomcat-8/v8.5.50/bin/apache-tomcat-8.5.50.tar.gz>

Extract the tar file:

[root@eca291a36127 /]# tar -xvzf apache-tomcat-8.5.50.tar.gz  
[root@eca291a36127 /]# cd apache-tomcat-8.5.50/bin/

Here you are going to find two shell script files named startup.sh and shutdown.sh, assign “execute” permission to both. Both these files allow us to start and stop the Tomcat server, within our container:

[root@eca291a36127 bin]# chmod +x shutdown.sh  
[root@eca291a36127 bin]# chmod +x startup.sh

To run the Tomcat server, Java is required. In order to do so, run following command to satisfy the requirement. Here we are going to install OpenJDK 7 using **Yum**:

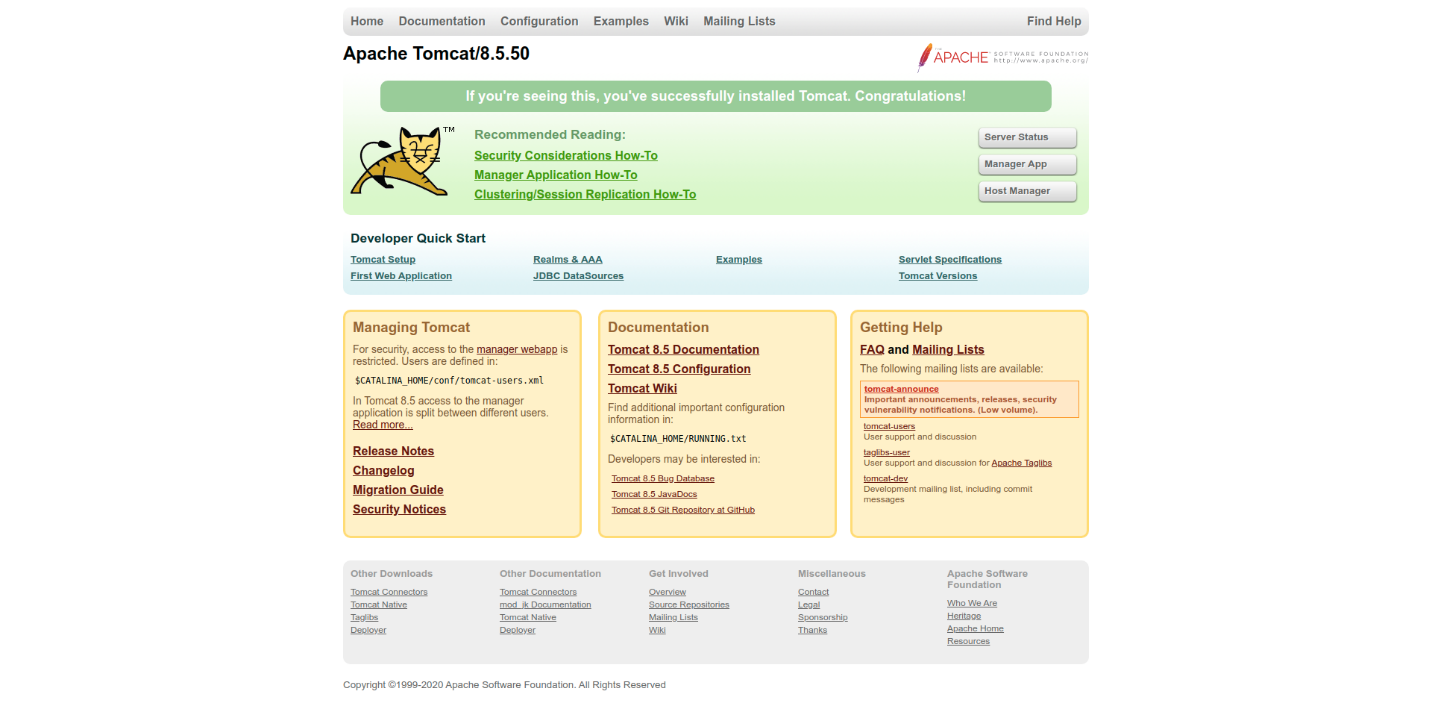
[root@eca291a36127 /]# yum install java-1.7.0-openjdk-devel

It’s time to run our Tomcat server; use startup.sh file to do so:

[root@eca291a36127 /]# ./startup.sh   
Using CATALINA\_BASE: /opt/apache-tomcat-8.5.50  
Using CATALINA\_HOME: /opt/apache-tomcat-8.5.50  
Using CATALINA\_TMPDIR: /opt/apache-tomcat-8.5.50/temp  
Using JRE\_HOME: /usr  
Using CLASSPATH: /opt/apache-tomcat-8.5.50/bin/bootstrap.jar:/opt/apache-tomcat-8.5.50/bin/tomcat-juli.jar  
Tomcat started.

In your terminal, you can see the message Tomcat started. Now, go to your browser; Tomcat should be accessible at [http://your-ip-addr:8888.](http://your-ip-addr:8888./)





In case you want to access the Manage App, you will have to do some additional settings, as by default, the Manager App section can only be accessed from the local system (i.e., 127.0.0.1).  
To overcome this, you will need to make some changes to context.xml file. To do so follow the below-mentioned steps:

$ docker exec -it tomcat\_container /bin/bash

Now, find the context.xml file:

$ find / -name context.xml  
   
/opt/apache-tomcat-8.5.50/conf/context.xml  
/opt/apache-tomcat-8.5.50/webapps/host-manager/META-INF/context.xml  
/opt/apache-tomcat-8.5.50/webapps/manager/META-INF/context.xml

Edit the file that resides in webapps directory i.e. /opt/apache-tomcat-8.5.50/webapps/host-manager/META-INF/context.xml, find the Value class name org.apache.catalina.valves.RemoteAddrValve and comment it:

...  
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"  
 allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->  
...

Let’s do the same thing on the other file as well, i.e., /opt/apache-tomcat-8.5.50/webapps/manager/META-INF/context.xml.

To access the default username and password through the Manager App section, go to conf/tomcat-users.xml and make the following changes:

$ /opt/apache-tomcat-8.5.50/conf && nano tomcat-users.xml

Now, as part of your pipeline, you will have to add a user and define roles on your Tomcat server. To copy the file from the remote machine (in our case, the Jenkins server), you will need to define the role *Manager Script*, copy the lines and add it to the end of the file just before the closing tag </tomcat-users>:

...   
<role rolename="manager-gui"/>  
 <role rolename="manager-script"/>  
 <role rolename="manager-jmx"/>  
 <role rolename="manager-status"/>  
 <user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>  
 <user username="deployer" password="deployer" roles="manager-script"/>  
 <user username="tomcat" password="s3cret" roles="manager-gui"/>  
</tomcat-users>  
...

Since you have made configuration changes in the Tomcat server, you will need to restart the server for the changes to take effect.

$ cd /opt/apache-tomcat-8.5.50/bin/  
$ ./shutdown.sh  
$ ./startup.sh   
   
Using CATALINA\_BASE: /opt/apache-tomcat-8.5.50  
Using CATALINA\_HOME: /opt/apache-tomcat-8.5.50  
Using CATALINA\_TMPDIR: /opt/apache-tomcat-8.5.50/temp  
Using JRE\_HOME: /usr  
Using CLASSPATH: /opt/apache-tomcat-8.5.50/bin/bootstrap.jar:/opt/apache-tomcat-8.5.50/bin/tomcat-juli.jar  
Tomcat started.

You can now access the Tomcat Web Application Manager dashboard; username and password are tomcat and s3cret, respectively.

Let us move to the next part of the tutorial.

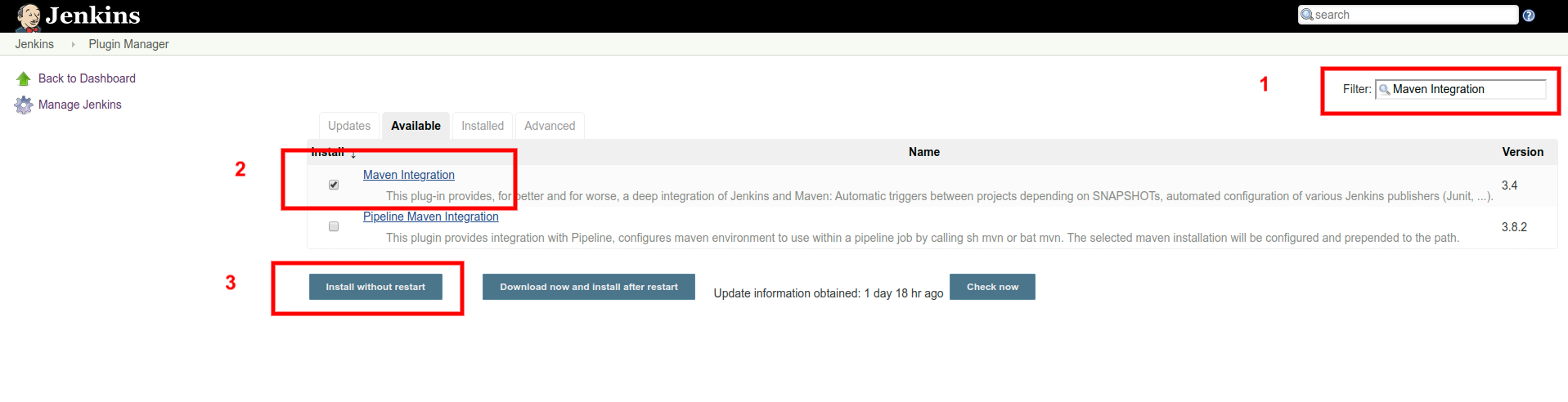
**Configure Jenkins**

It’s time to set up Maven and an additional plugin so that Jenkins can connect to Tomcat container over SSH. For this, we are going to install Maven Integration and Publish over SSH plugin in Jenkins.

Login to your Jenkins environment and go to *Dashboard > Manage Jenkins > Manage Plugins > Available*, as shown below:

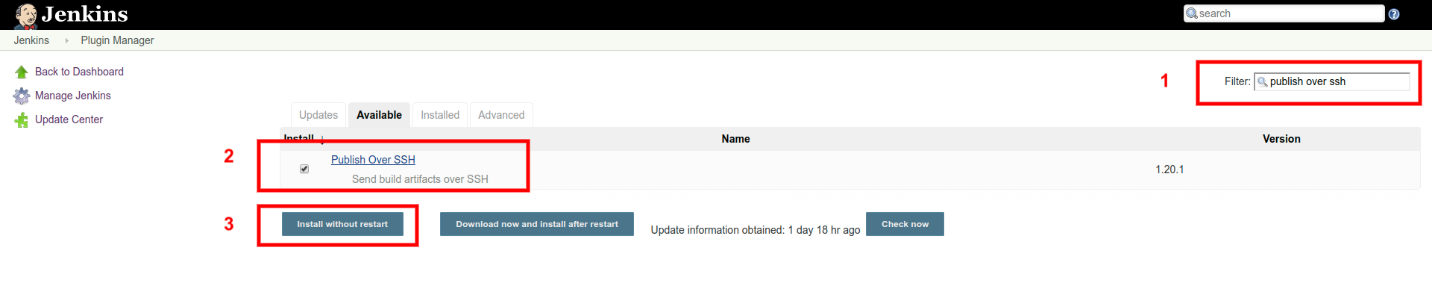
For Maven Integration:





For Publish over SSH:





Post-installation, you need to configure SSH server details so that Jenkins can connect the Tomcat container. In Jenkins, go to *Dashboard > Manage Jenkins > Configure System* and find *SSH Servers*. Enter details as shown below:





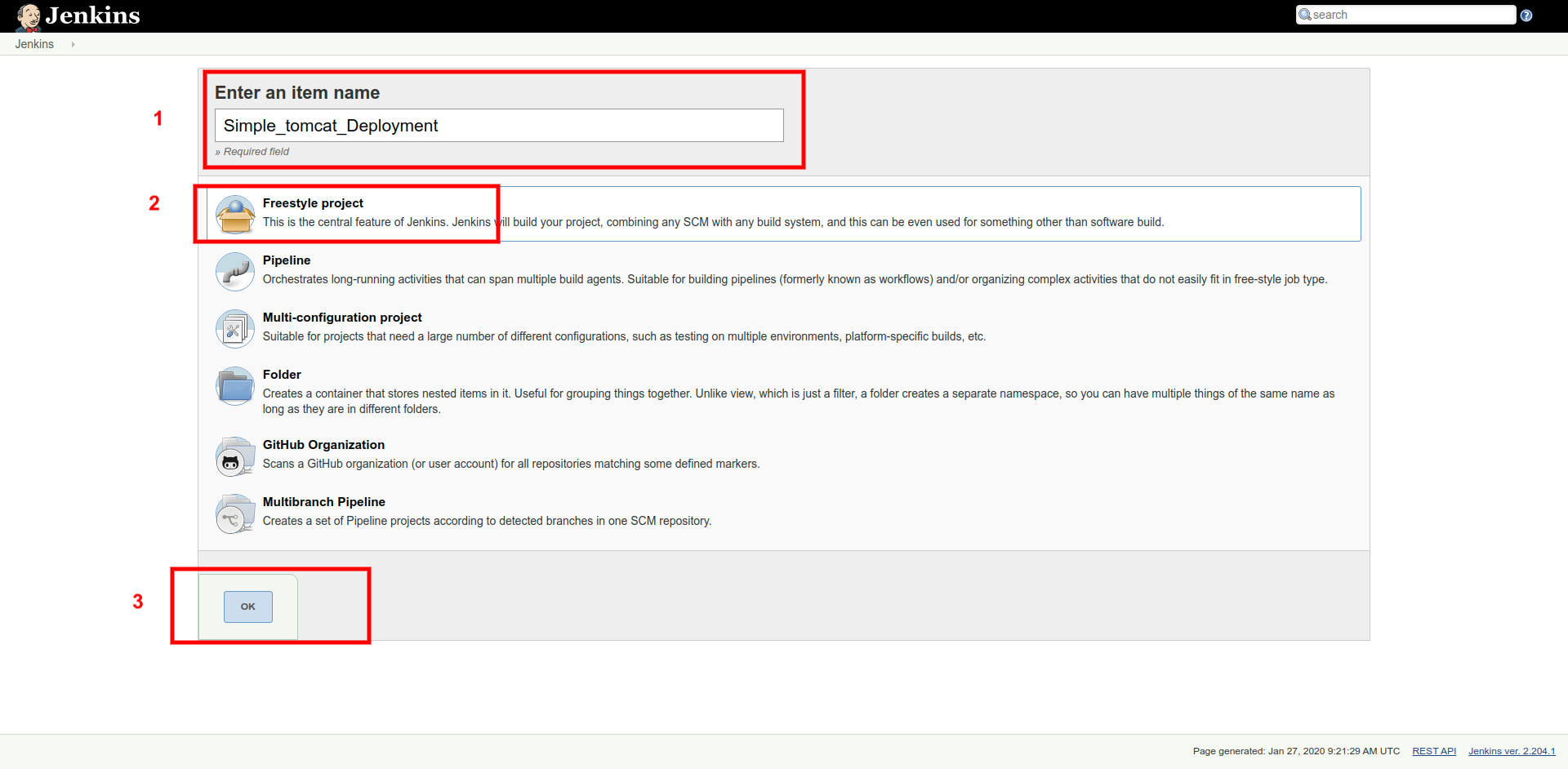
Test your connection before saving the details. For the sake of this article, we have used the password authentication method, however, it is not recommended in a production environment. We recommend you use a SSH key instead.

**Deploying WAR file on Tomcat Server using Jenkins**

As you are through with the configuration part, now you can proceed to create a Jenkins Job.

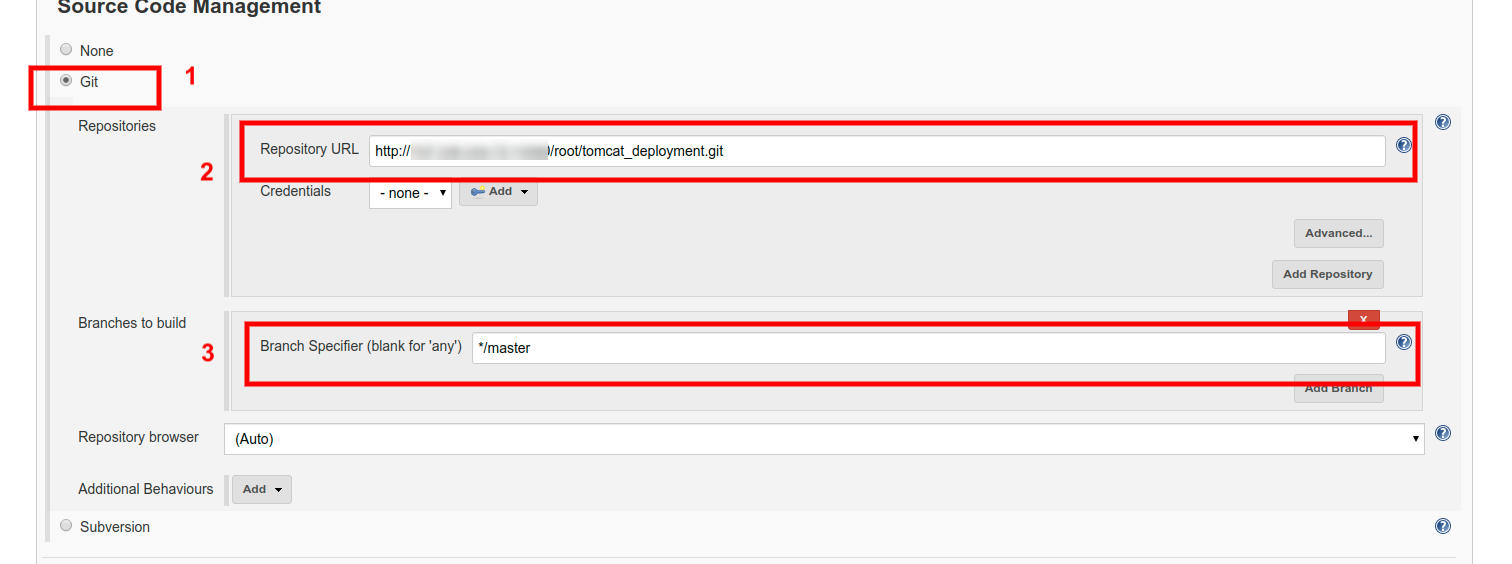
For this simple job, you would need to use *Freestyle Project*. Create a new item, as shown below:





**SCM**  
Go to the Source Code Management (also known as SCM), and select *Git*. Add your Github or GitLab URL in *Repository URL* field:

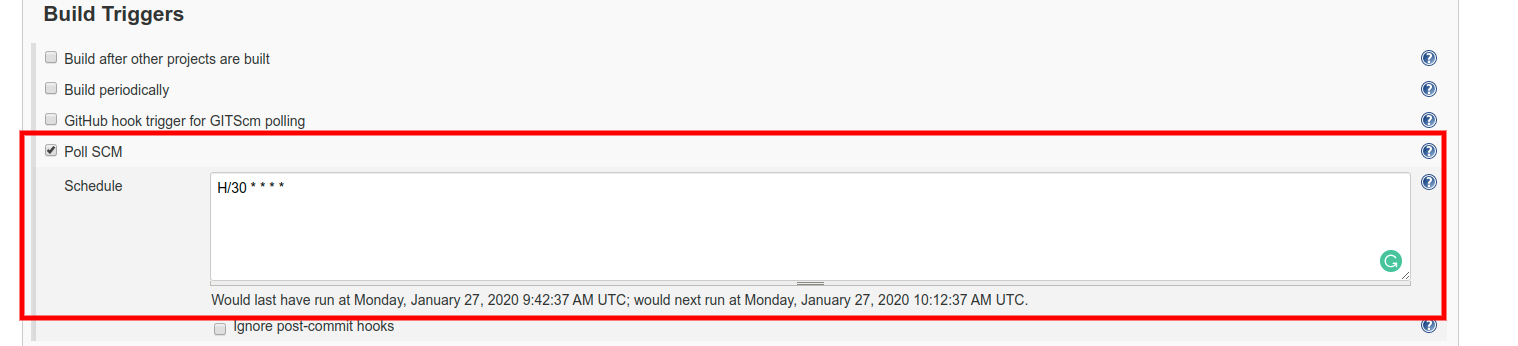




**Poll SCM**  
In the *Build Triggers* section, select *Poll SCM*. It periodically polls the VCS to check if any changes were made. During the poll, if any changes were detected in repo since the last build, a new build is initiated.

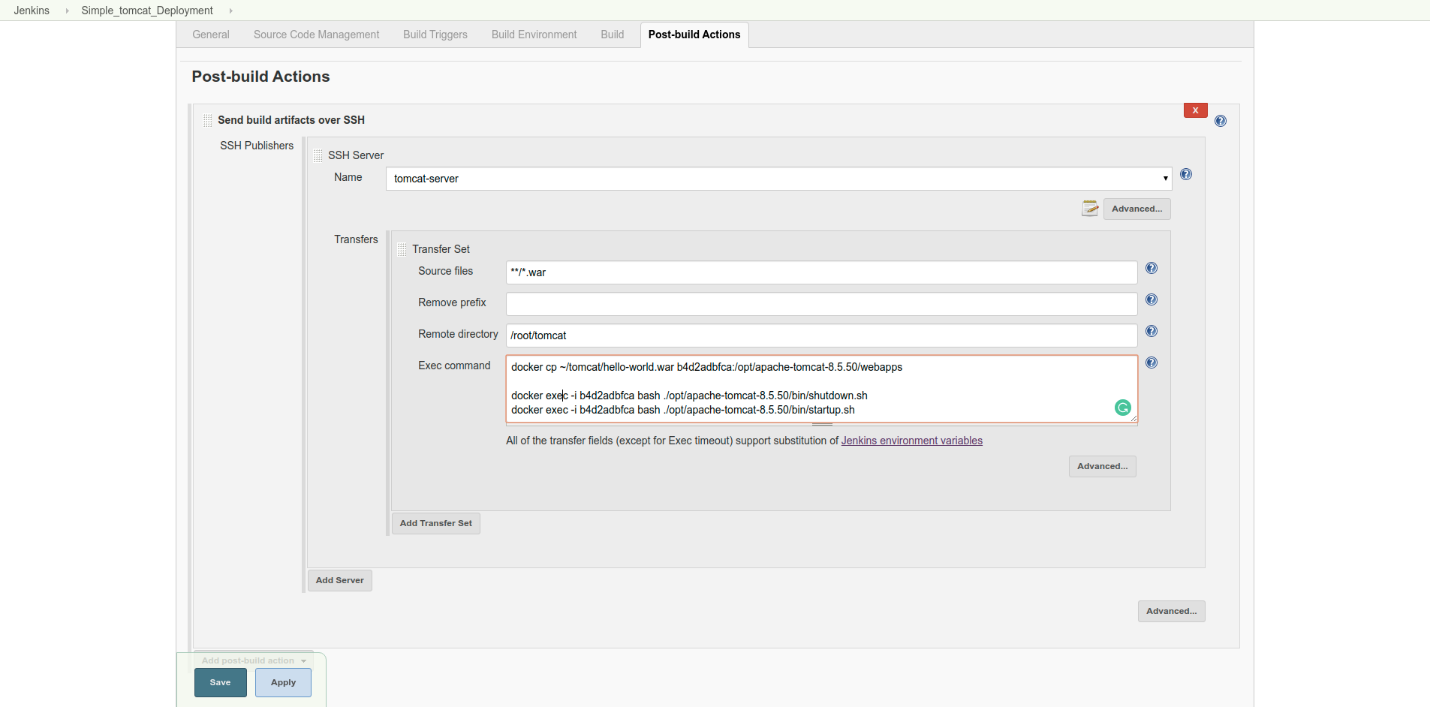
Let’s say you want to poll SCM every 30 minutes, then in the text field, add H/30 \* \* \* \* (i.e., every 30 minutes, every hour, every day, every month, and every day of the week).





**Post-Build Step**  
Go to *Post-build Actions* section and from the drop-down menu, select *Send build artifacts over SSH*, as shown below:





In the *Source files* field, mention the file name that is going to be uploaded to the server. In case you are not sure about the file name, you can use the pattern as mentioned above \*\*/\*.war.

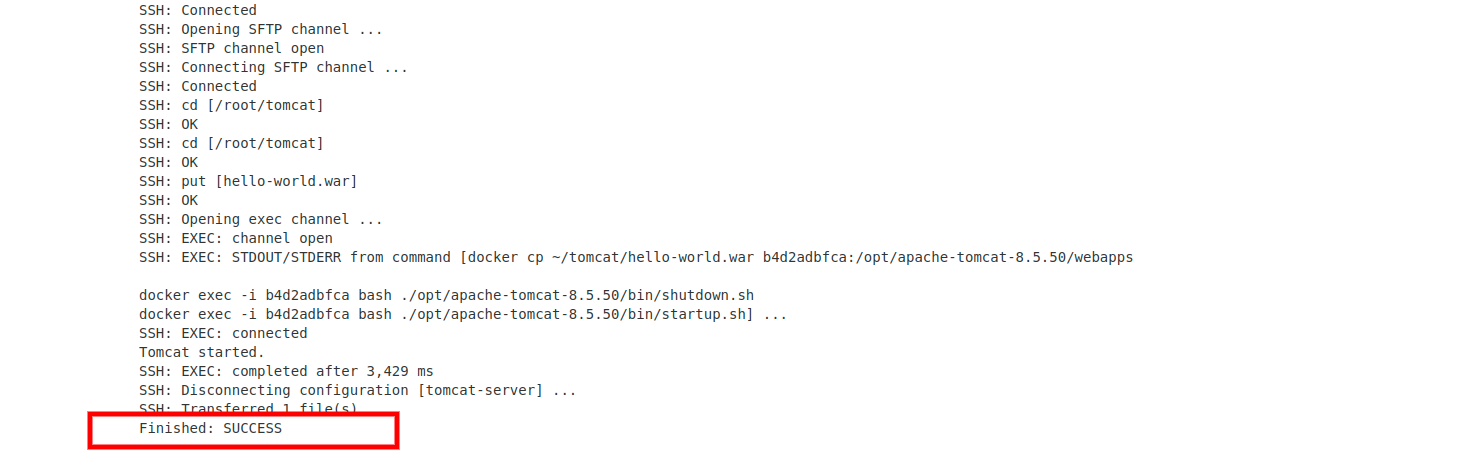
Other than this, make sure that you mention the *Remote Directory*; in our case, it is /root/tomcat. Whereas, for *Exec command*, as you are using Docker for your Tomcat server, use the below-mentioned commands:

docker cp ~/tomcat/hello-world.war tomcat\_container:/opt/apache-tomcat-8.5.50/webapps  
  
docker exec -i tomcat\_container bash ./opt/apache-tomcat-8.5.50/bin/shutdown.sh  
  
docker exec -i tomcat\_container bash ./opt/apache-tomcat-8.5.50/bin/startup.sh

The above will copy hello-world.war file to your dockerized tomcat container in /opt/apache-tomcat-8.5.50/webapps. Plus, you are making sure that you restart Tomcat to see the changes.

**Build Now**  
Initiate the build process from the Project’s Jenkins Page, and go to the console log output to see if the build was a success or if there is any error/exception.





Your build finished with a SUCCESS message. Now go to your web browser and open http://your-ip-addr:8888/hello-world. You will see your application index page content. In our case, output is:



