Experiment 06 - Jenkins Pipeline

| Roll No. | 37 |
| --- | --- |
| Name | Mikil Lalwani |
| Class | D15-B |
| Subject | DevOps Lab |
| LO Mapped | LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements  LO3: To understand the importance of Jenkins to Build and deploy Software Applications on server environment |
|  |  |

**Aim**: To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to build and deploy an application over the tomcat server.

**Introduction**:

*(Introduction to pipeline and its advantages)*

In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.

It is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.

In other words, a Jenkins Pipeline is a collection of jobs or events that brings the software from version control into the hands of the end users by using automation tools. It is used to incorporate continuous delivery in our software development workflow.

A pipeline has an extensible automation server for creating simple or even complex delivery pipelines "as code", via DSL (Domain-specific language).

What is a Continuous Delivery Pipeline?

In a Jenkins Pipeline, every job has some sort of dependency on at least one or more jobs or events.

Jenkins Pipeline

The above diagram represents a continuous delivery pipeline in Jenkins. It contains a collection of states such as build, deploy, test and release. These jobs or events are interlinked with each other. Every state has its jobs, which work in a sequence called a continuous delivery pipeline.

A continuous delivery pipeline is an automated expression to show your process for getting software for version control. Thus, every change made in your software goes through a number of complex processes on its way to being released. It also involves developing the software in a repeatable and reliable manner, and progression of the built software through multiple stages of testing and deployment.

**Build Tool**:

*(Explain the build tool used)*

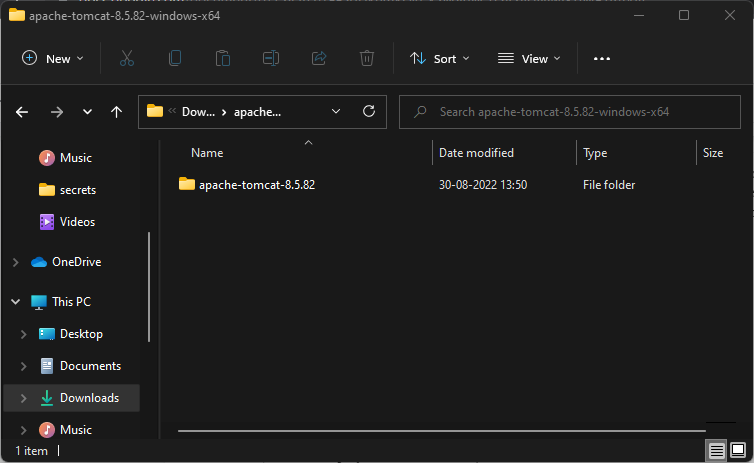
Git

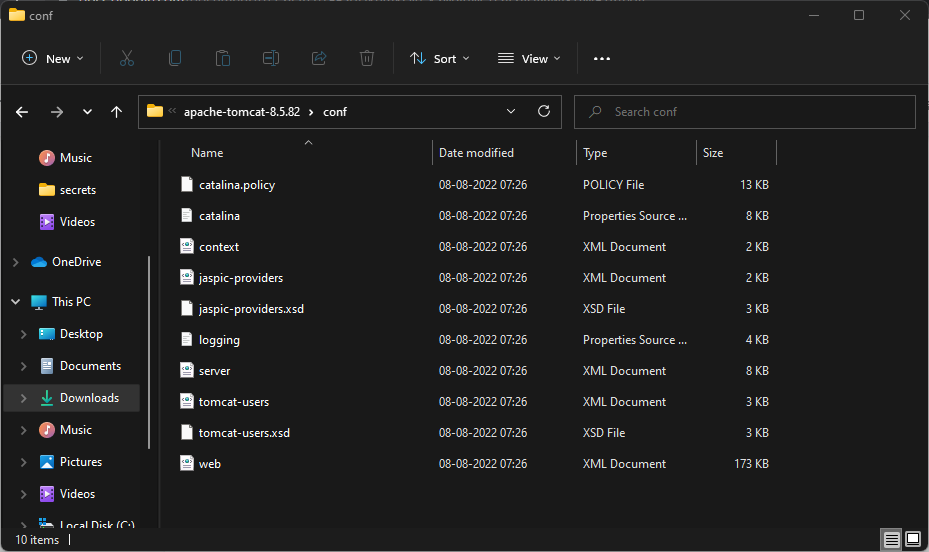
Tomcat

Jenkins

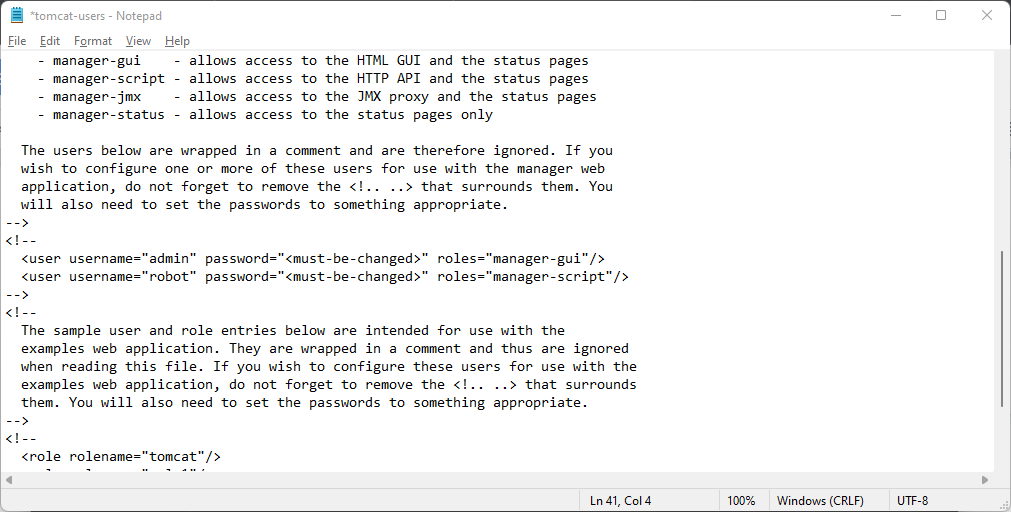
**Pipeline Jobs**:

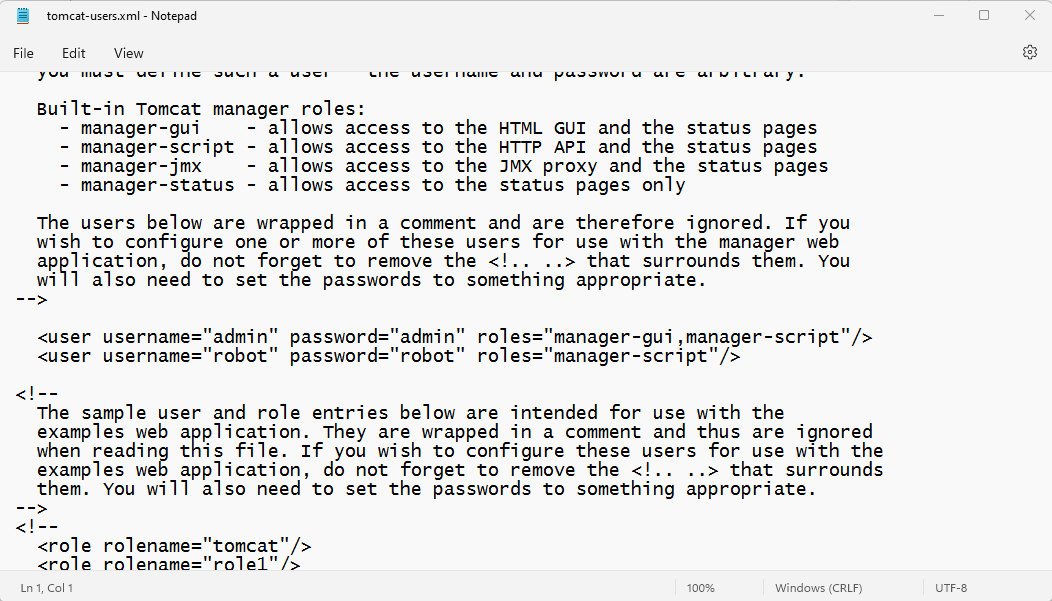
1. Install Apache Tomcat.



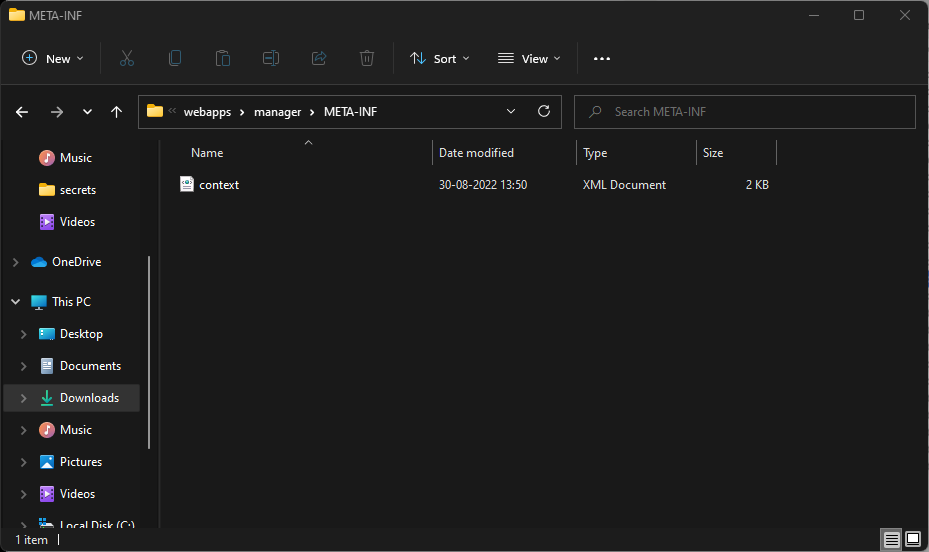


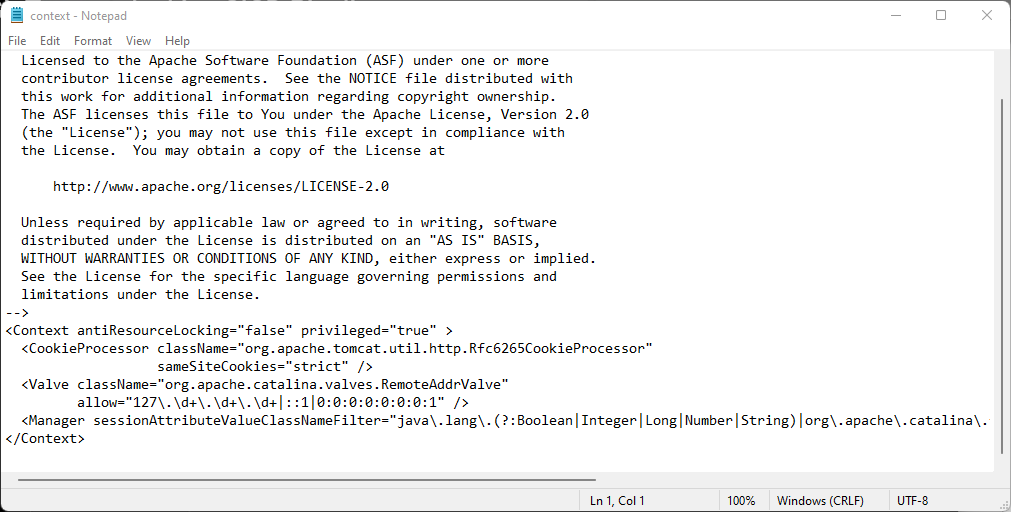
1. Go to apache-tomcat-10/conf/tomacat-users.xml and uncomment username and set password as your choice.

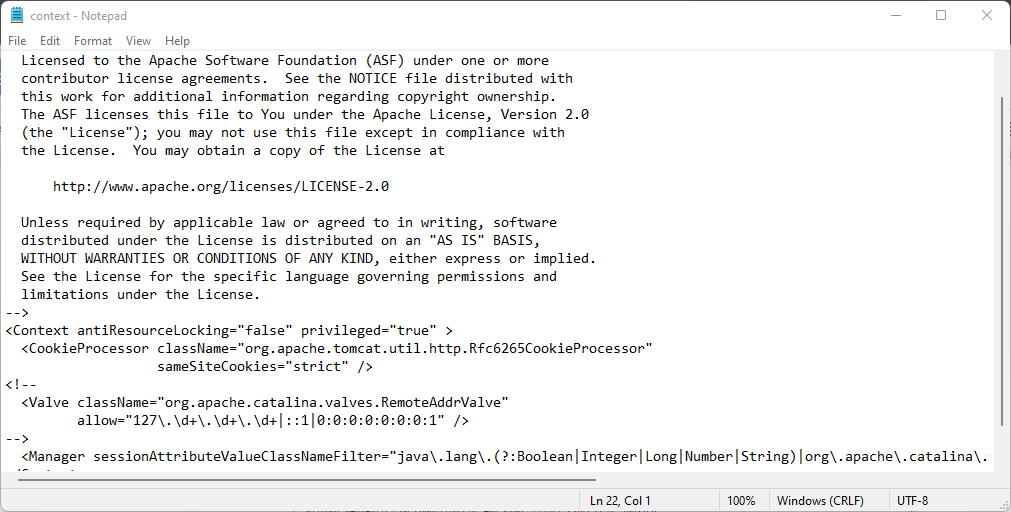




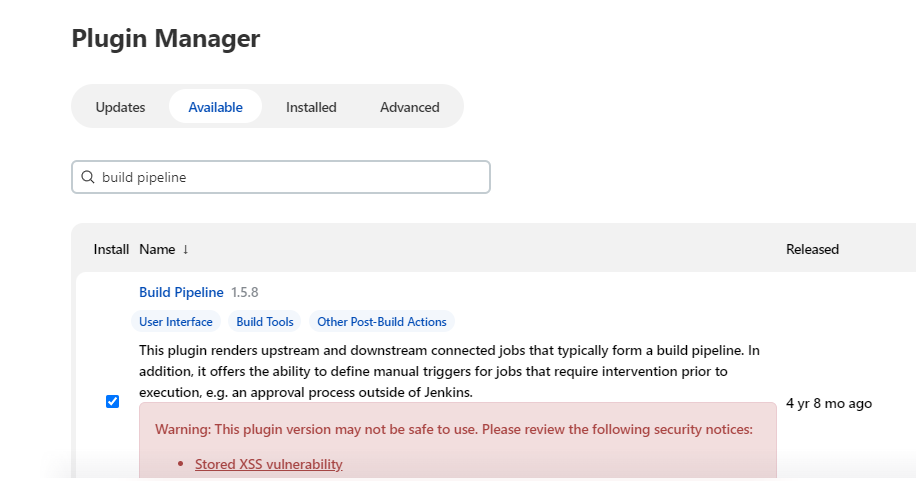
1. Goto apache-tomcat-10/webapps/manager/META-INF/context and comment Valve classname as shown below.

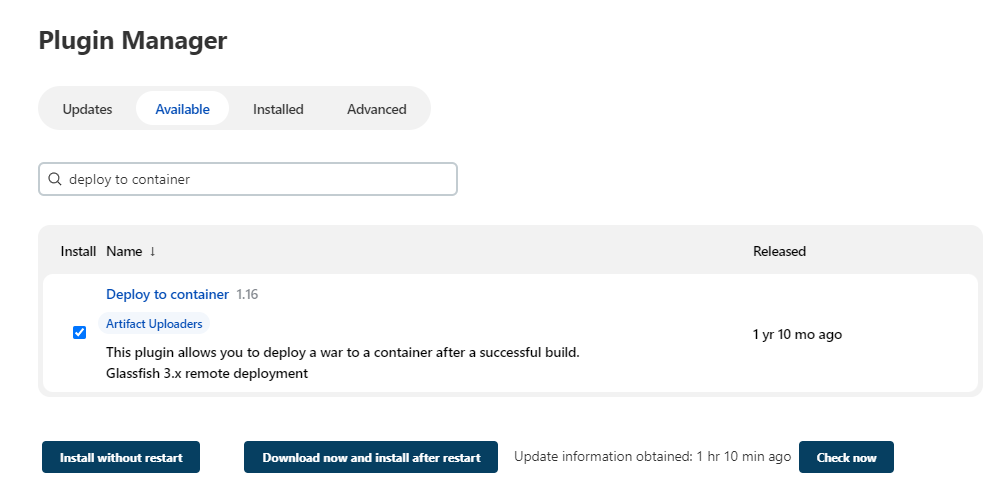




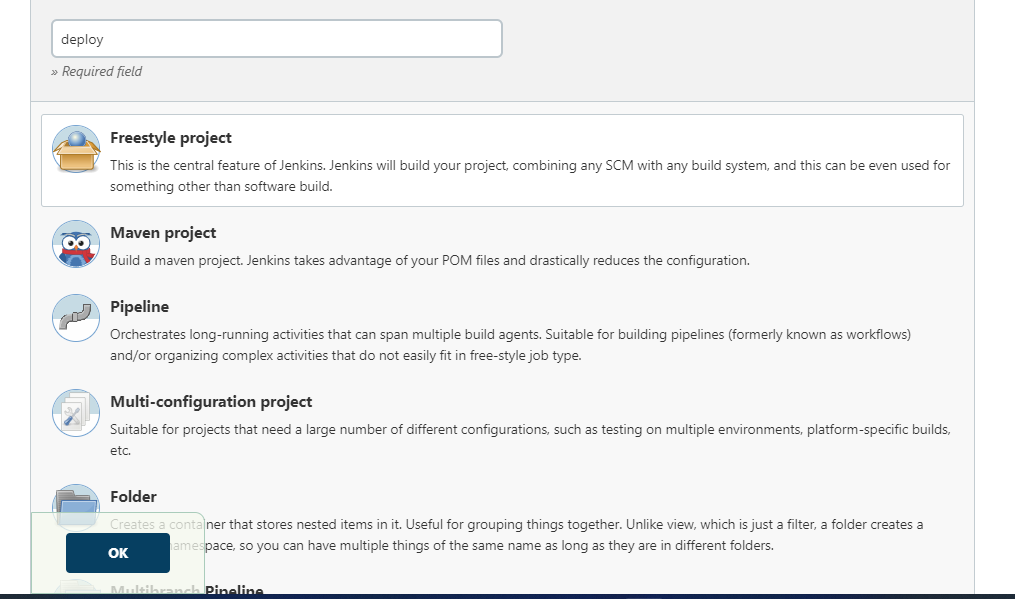
**

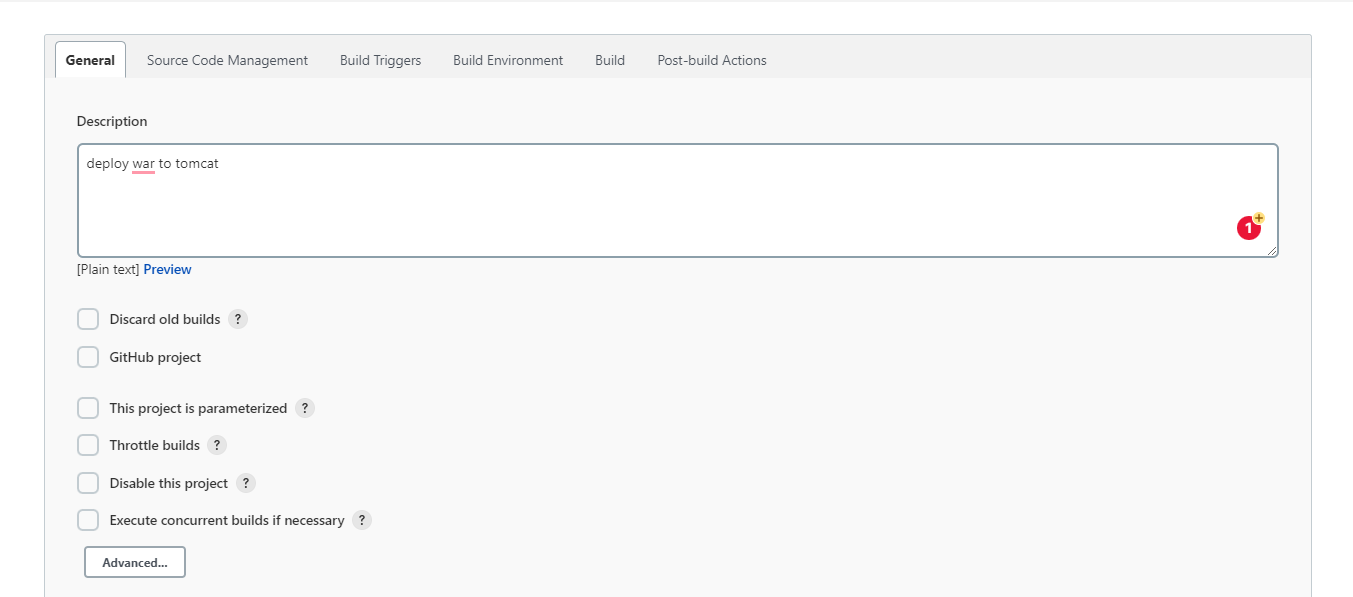
1. Install the build pipeline and deploy to container plugins.

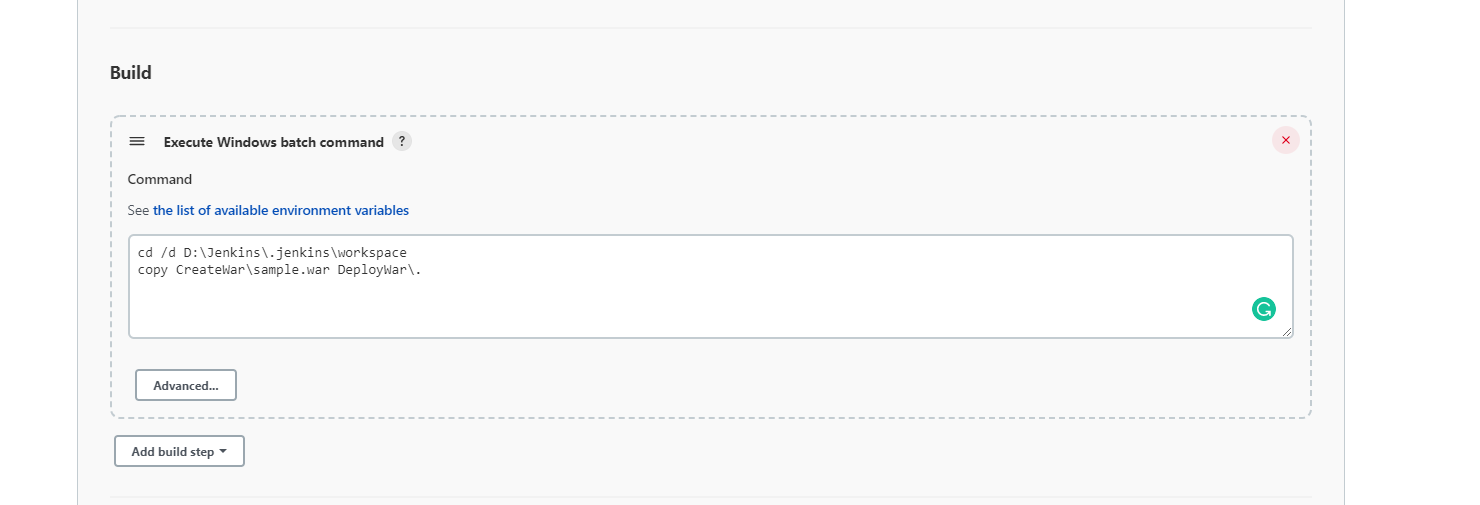
**

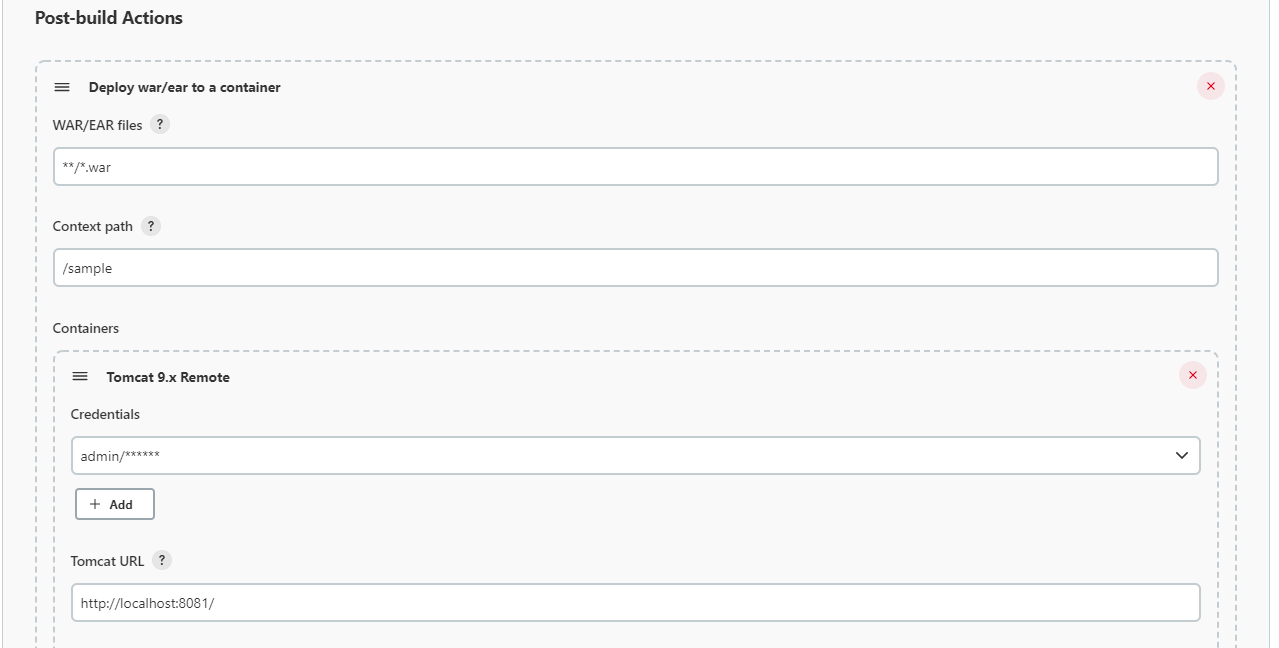
**

1. Create a new freestyle project and name DeployWar and enter details as shown below.

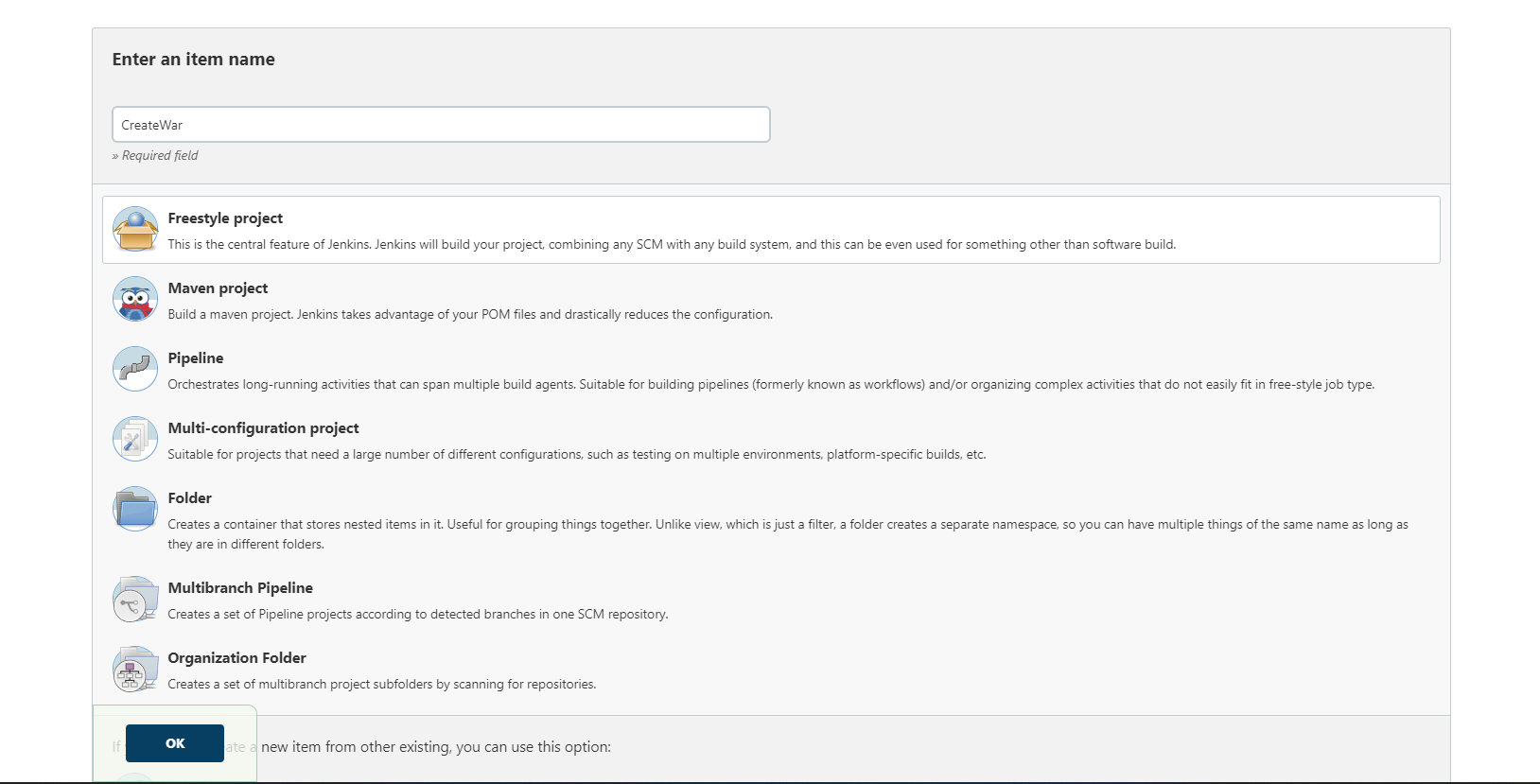
**

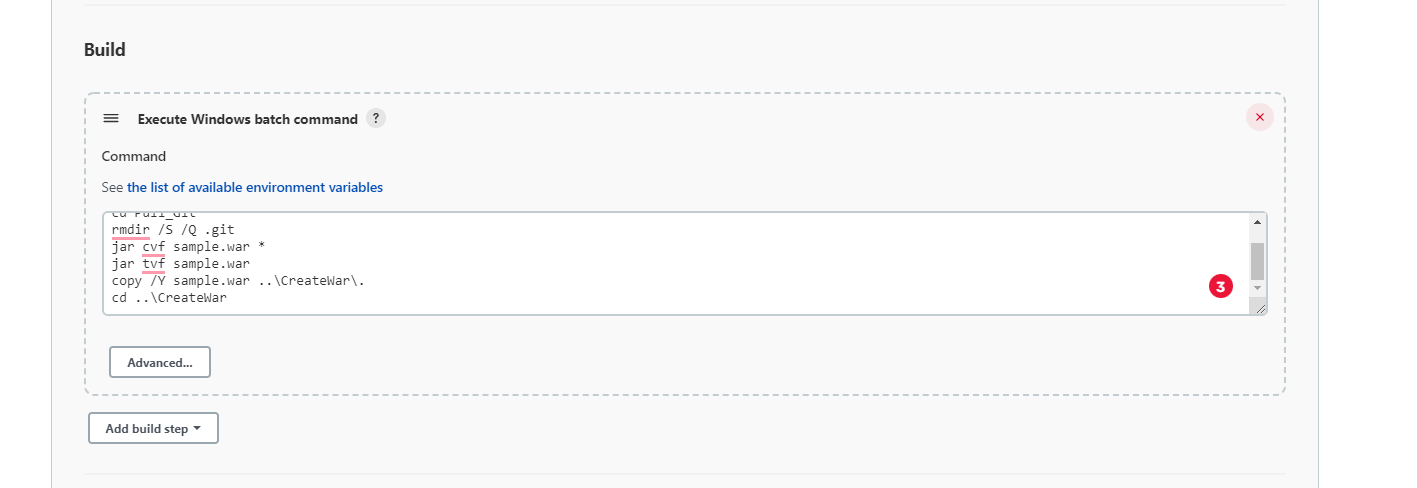
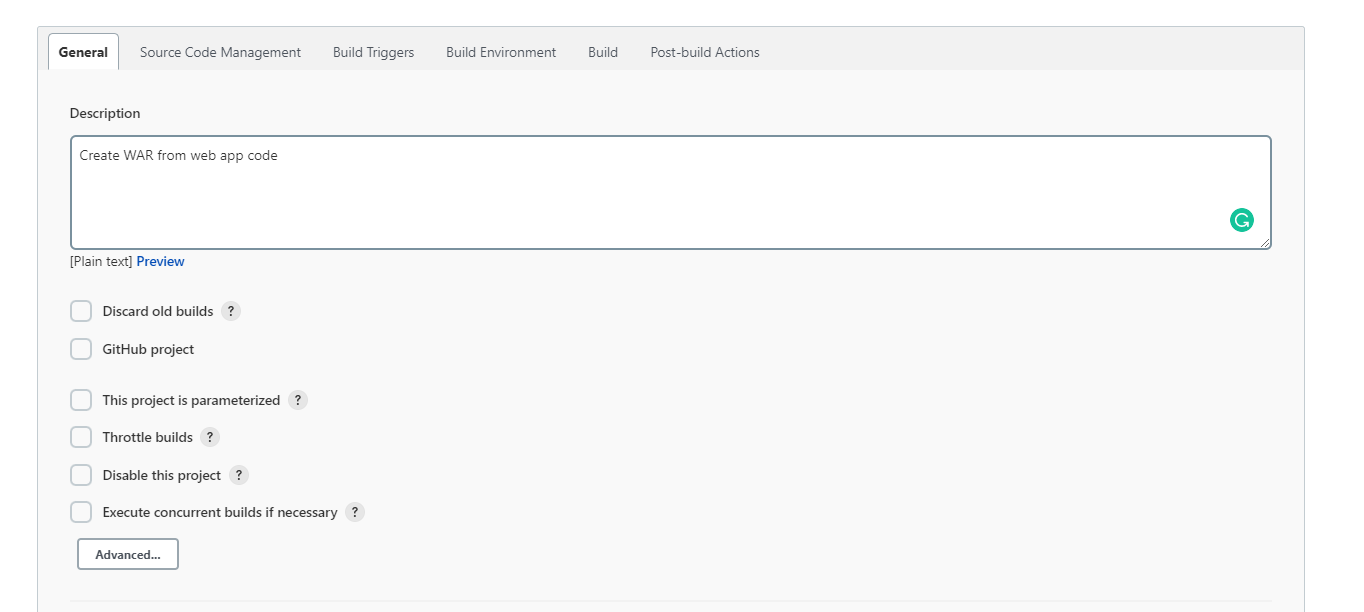
**

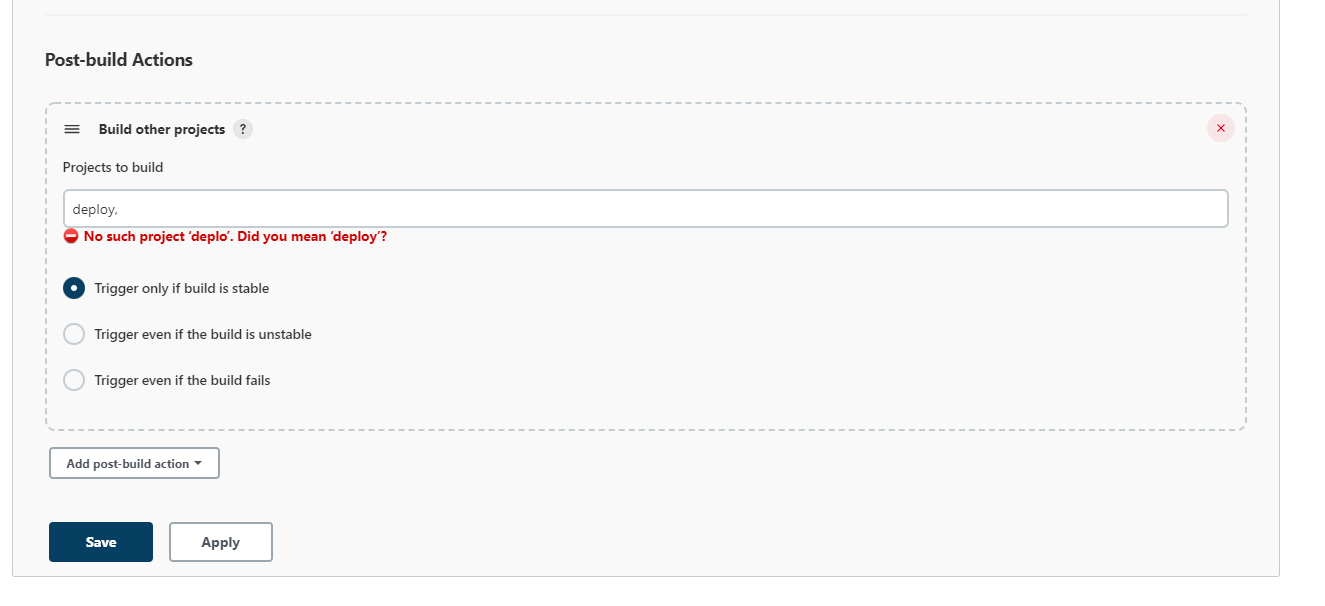
**

**

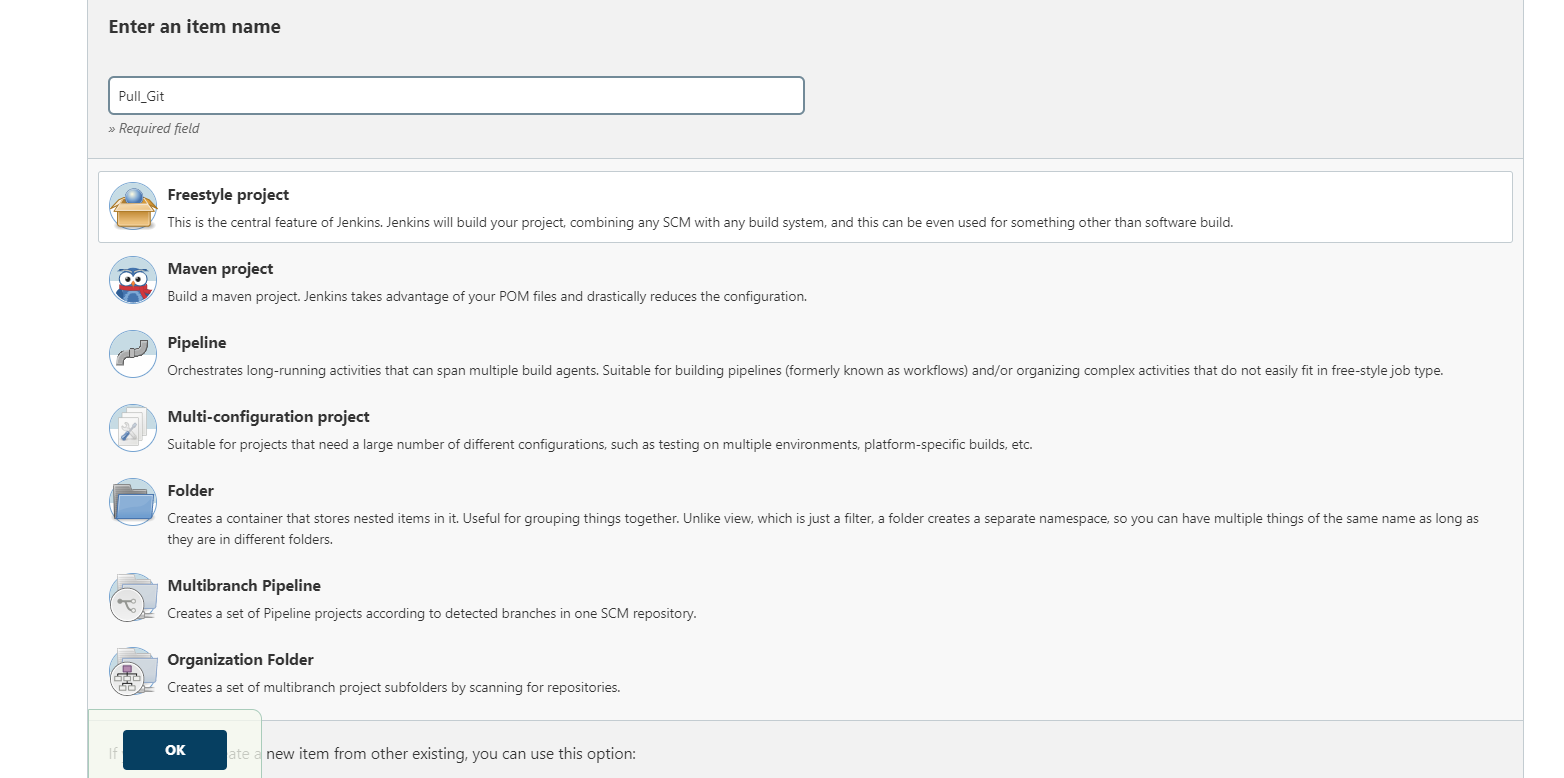
1. Create a new freestyle project and name CreateWar and enter details as shown below.

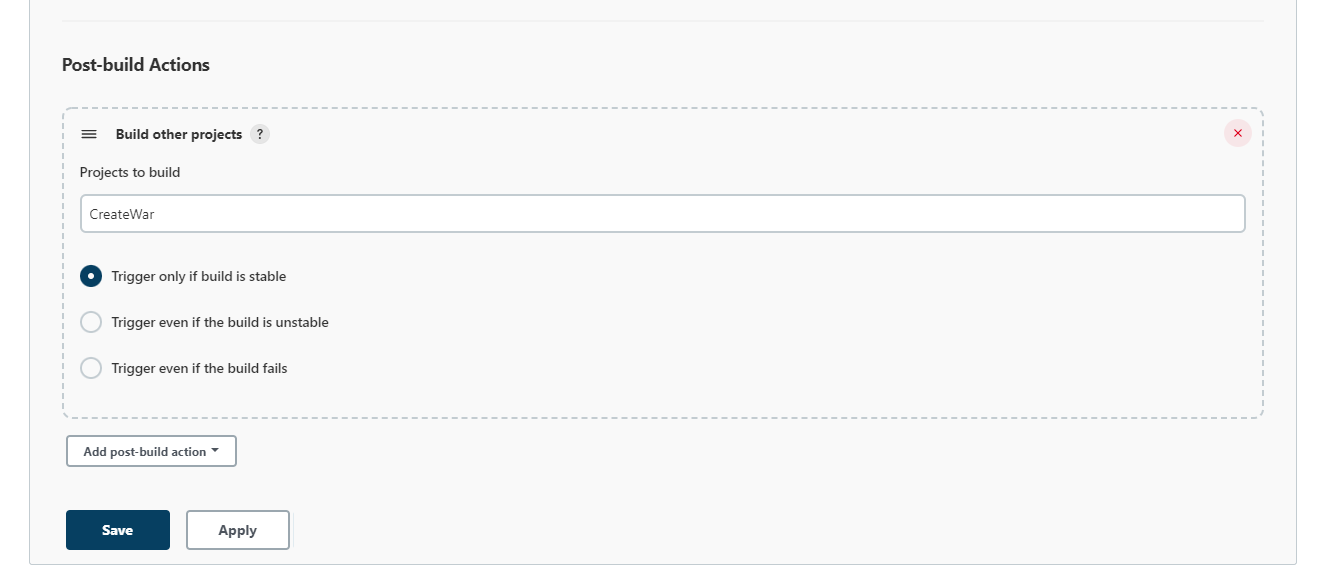
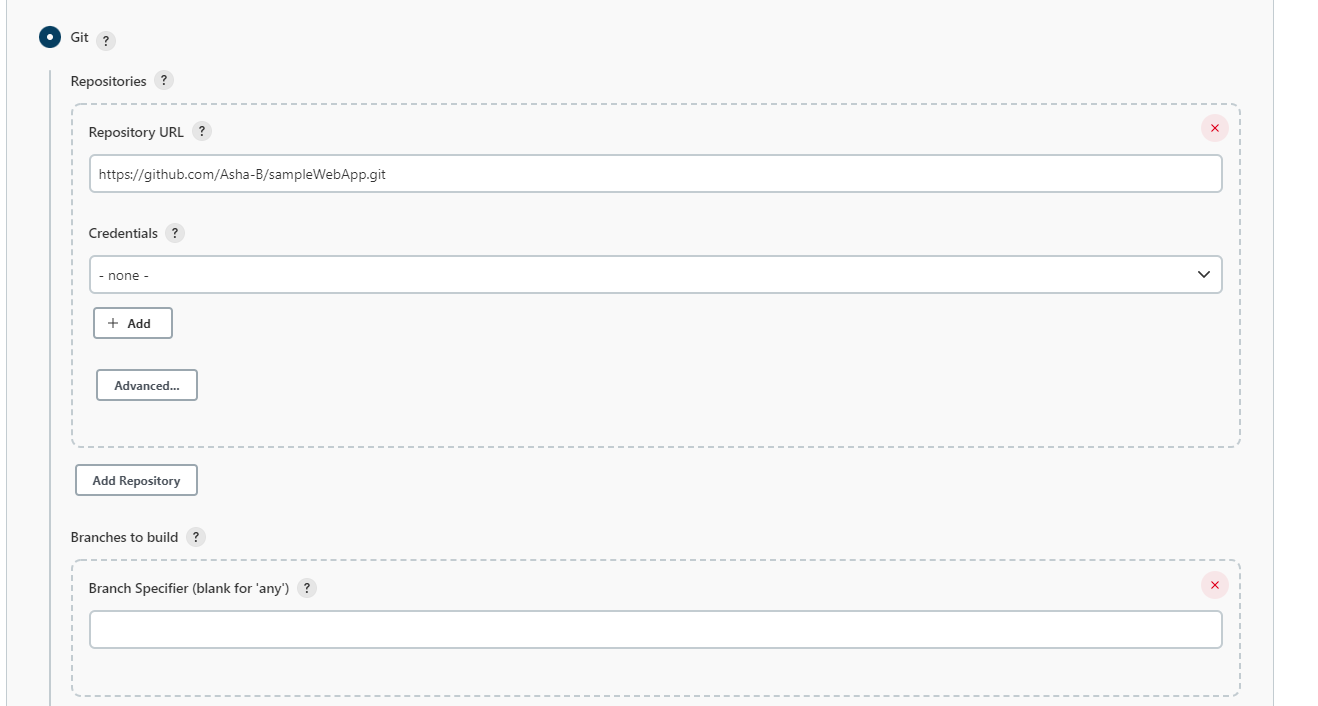
**

**

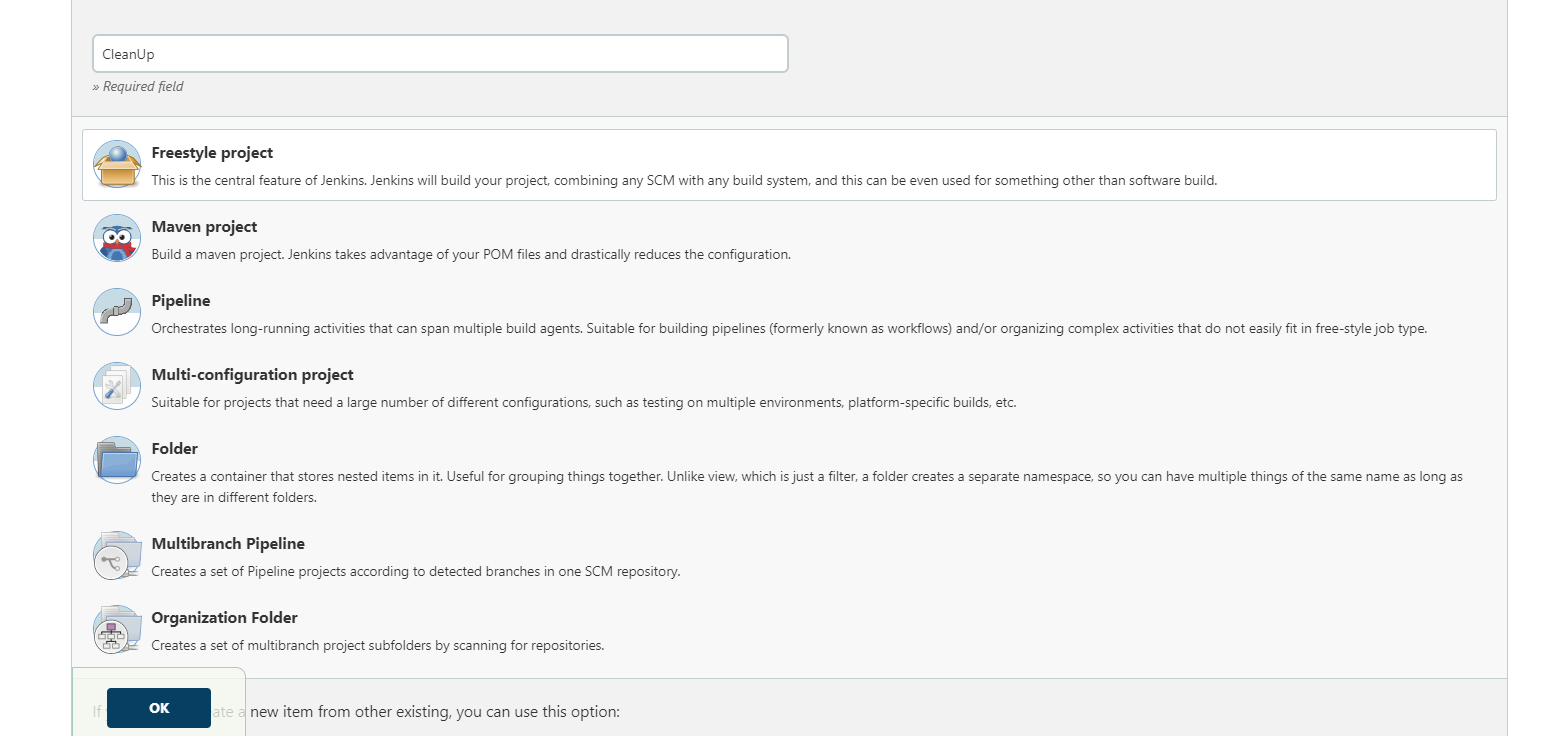
**

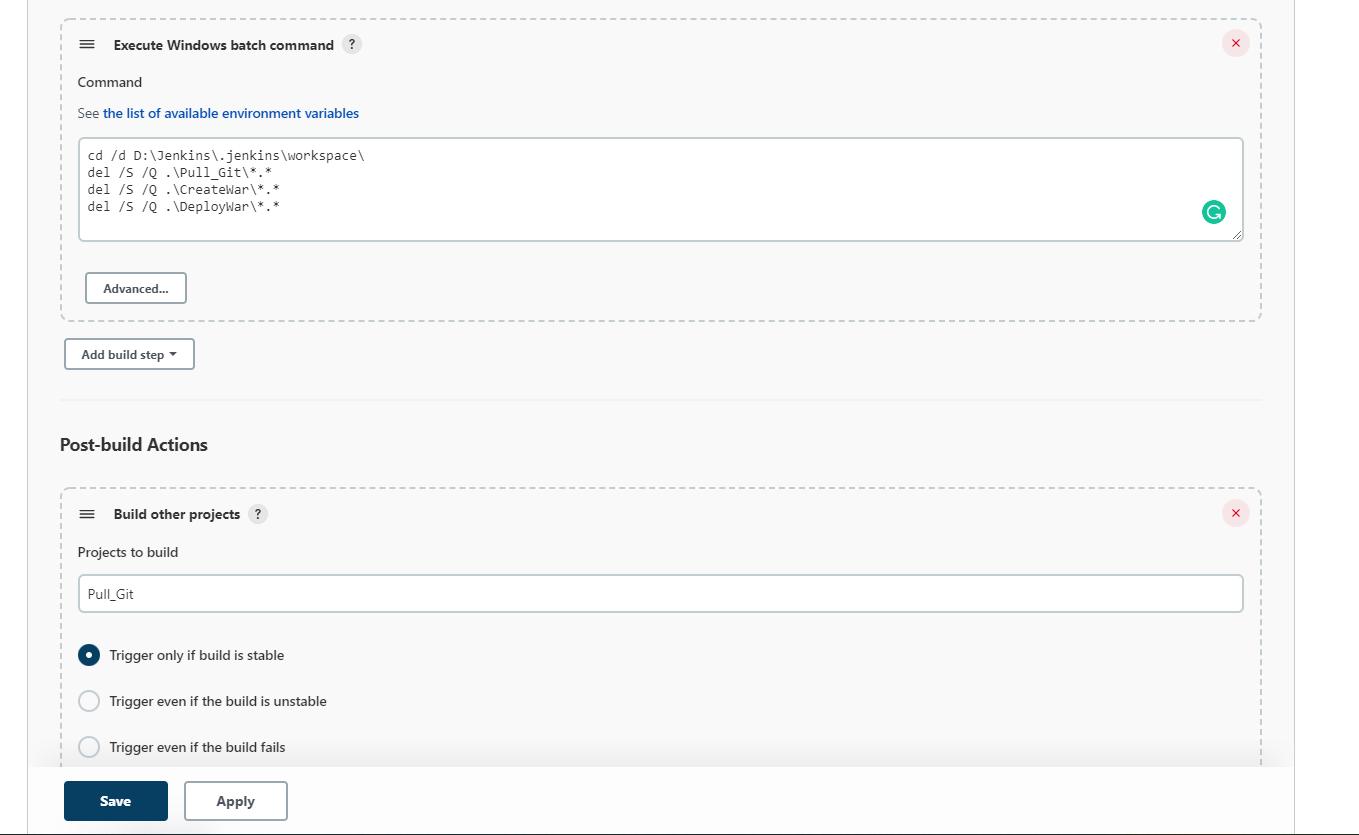
1. Create a new freestyle project and name Pull\_Git and enter details as shown below.

**

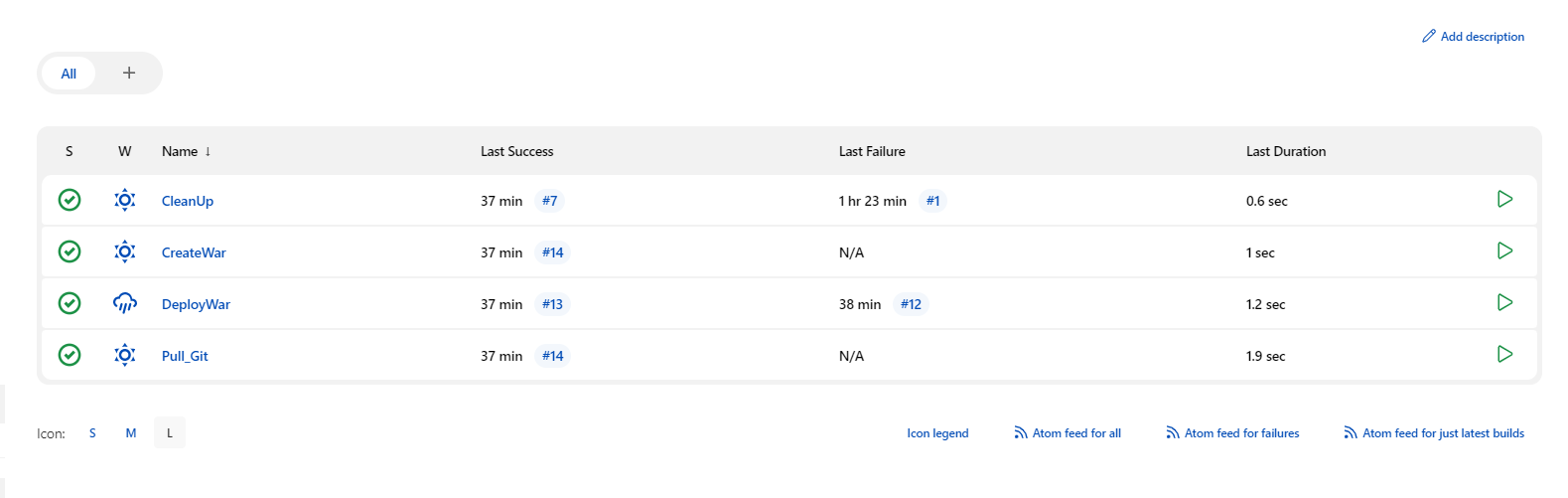
**

1. Create a new freestyle project and name CleanUp and enter details as shown below.

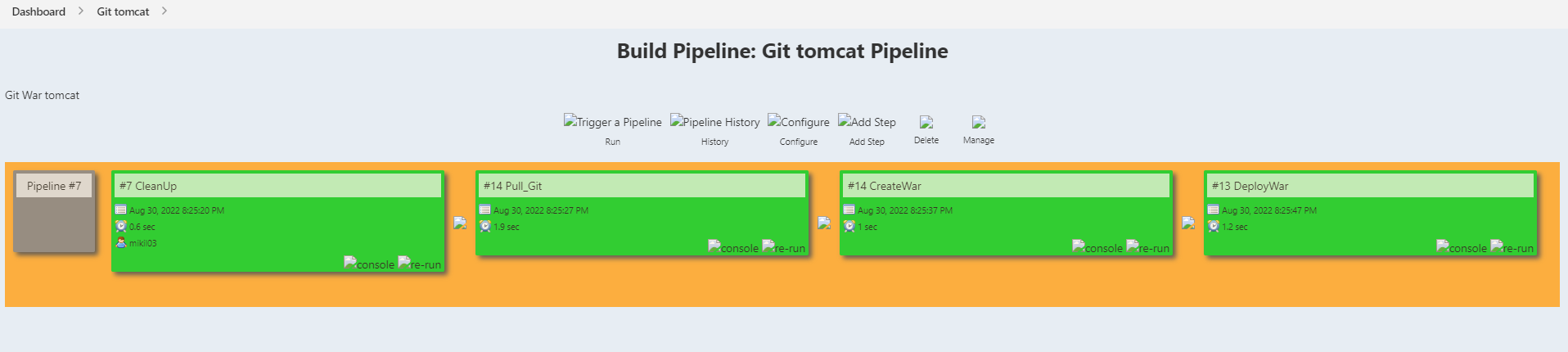
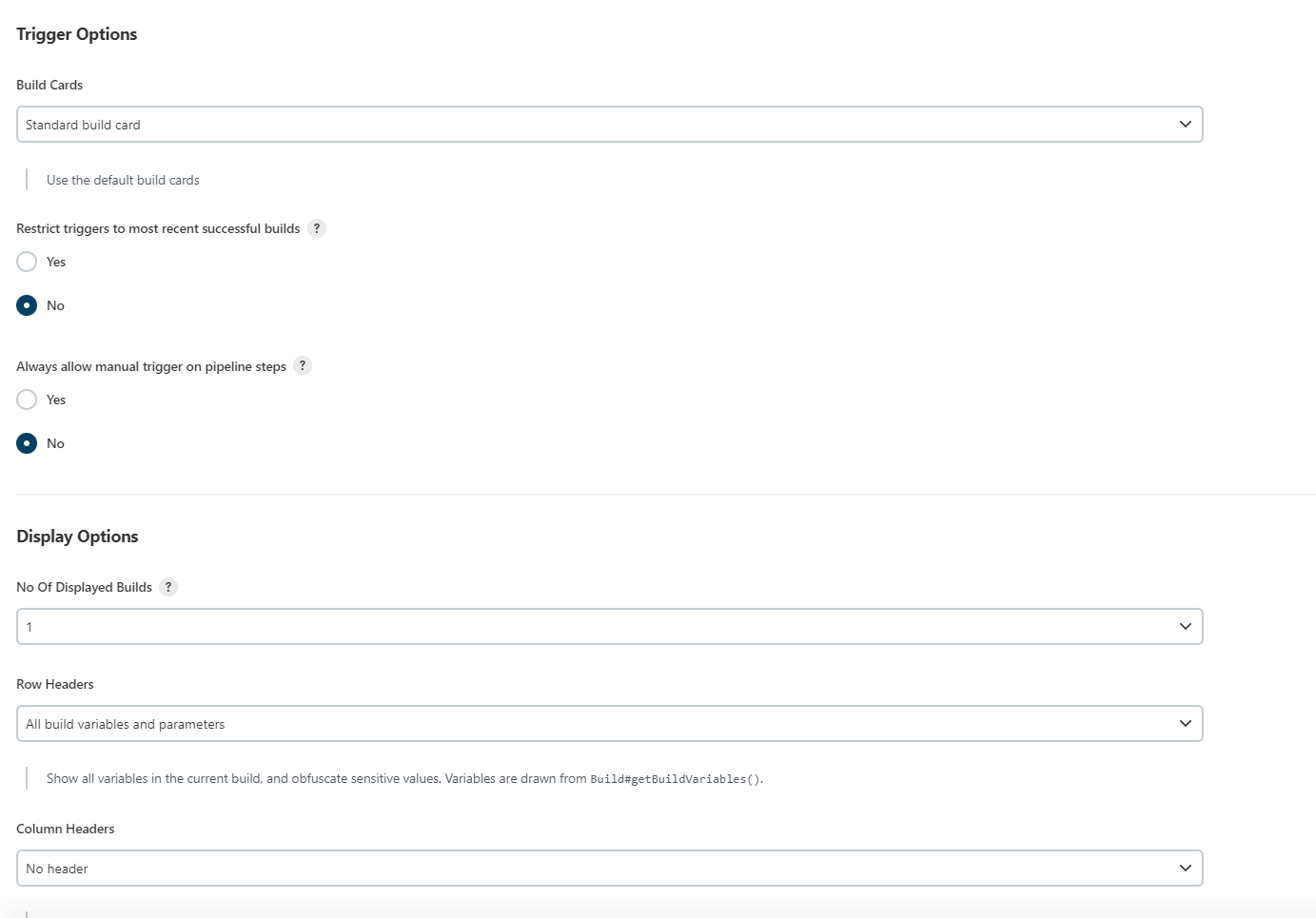
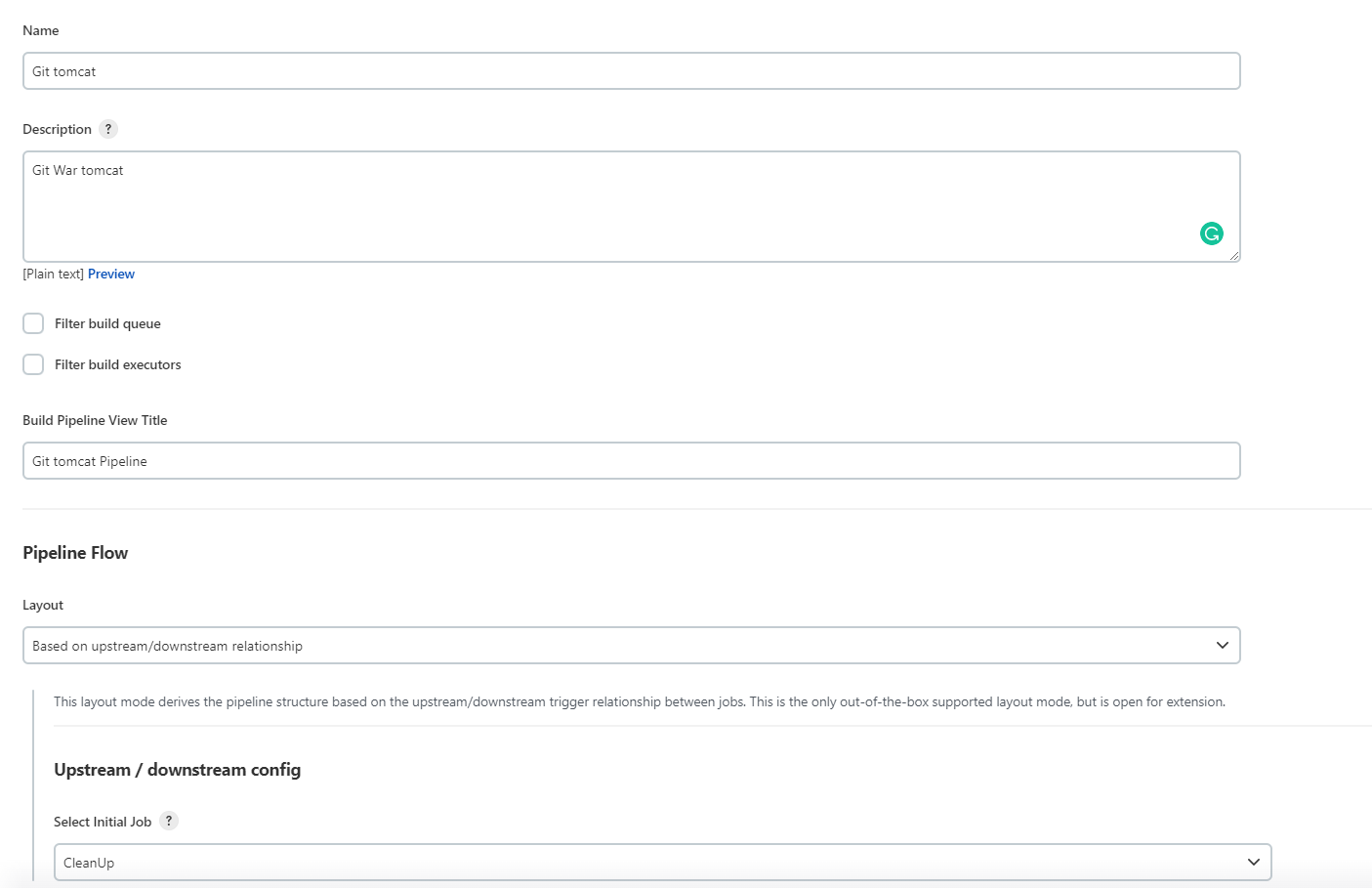
**

**

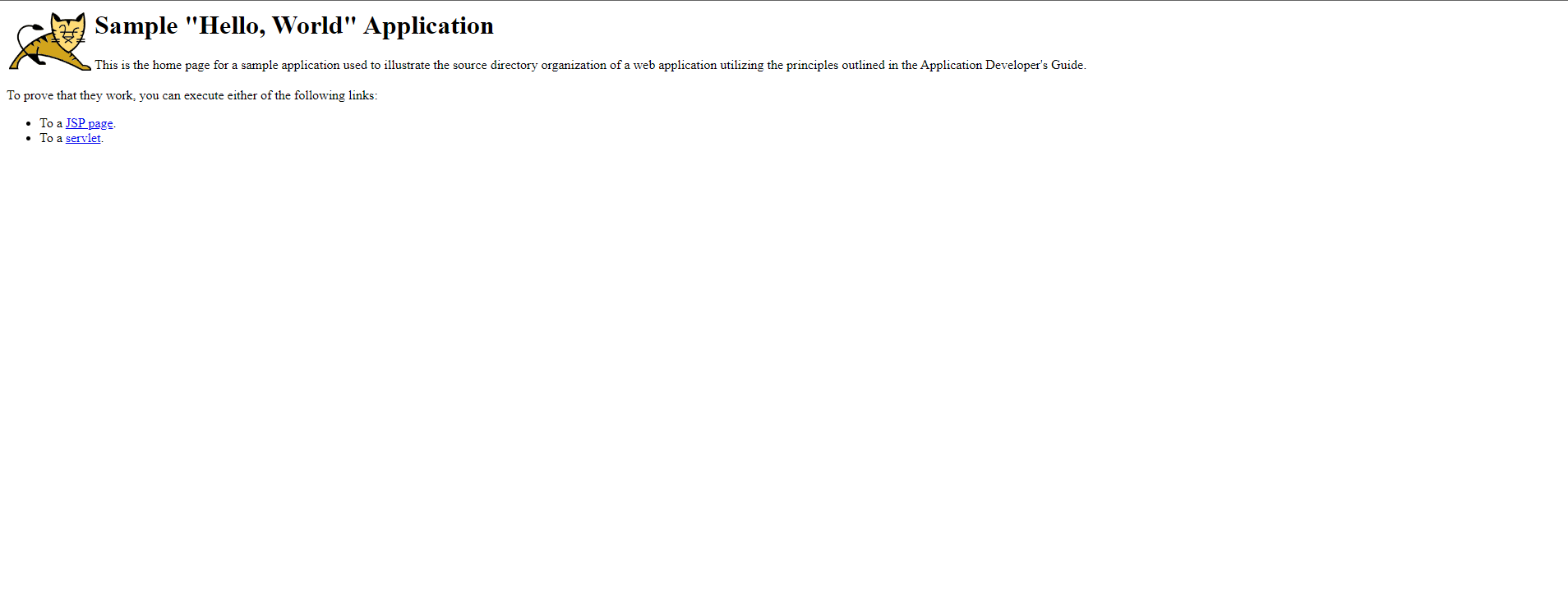
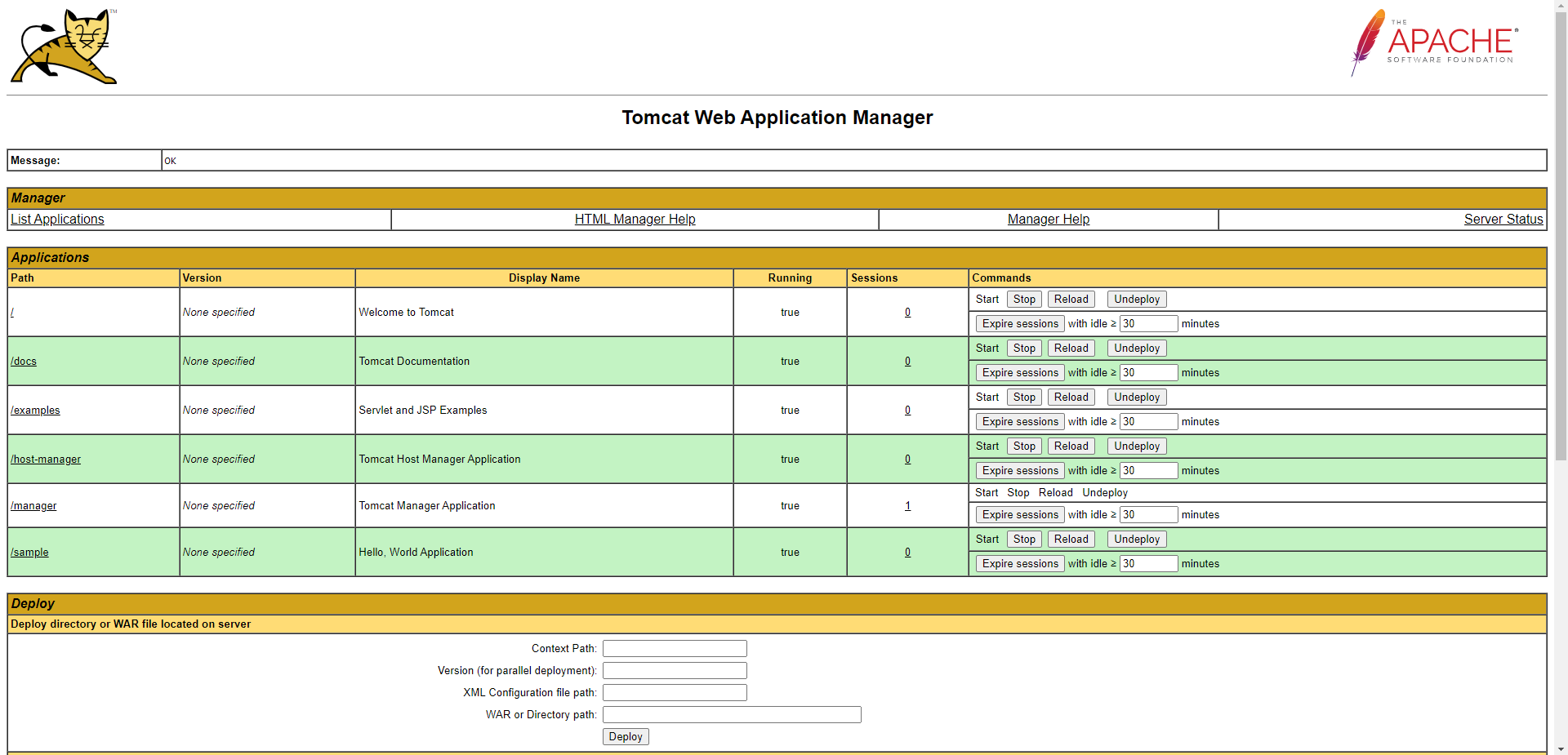
1. Now build the Pull\_Git project.

**

1. Create a new Pipeline.**

**

1. Now when the pipeline execution is complete we can see a sample folder in tomcat server at localhost 8081. In this folder we can see our webapp.

******

**Conclusion**

Thus we successfully built a Jenkins Pipeline.