This is a Jenkins pipeline script written in Groovy, which is a programming language used for scripting in Jenkins. The pipeline consists of several stages that define the steps to be executed in the pipeline.

Let's go through the pipeline step by step:

1. The **agent any** line specifies that the pipeline can run on any available agent (slave node) in the Jenkins environment.
2. The **options** block is used to define additional pipeline options. In this case, the **timestamps** option is enabled, which adds timestamps to the console output for each step.
3. The **tools** block is used to define the tools that the pipeline needs. In this case, the **maven** tool with version 'Maven-3.8.4' is configured. This ensures that the correct version of Maven is available for the pipeline.
4. The **stages** block defines the different stages of the pipeline. Each stage represents a logical step in the building process.

a. The first stage is named **"Git Checkout".** Within this stage, the pipeline performs a git checkout operation to fetch the source code from a GitHub repository. The **credentialsId** parameter specifies the ID of the Jenkins credential containing the necessary credentials for accessing the repository, and the **url** parameter specifies the URL of the repository.

b. The second stage is named **"Maven Clean Build".** Within this stage, the pipeline executes a shell command (**sh**) to run Maven with the **clean install** goals. The **-f** option is used to specify the location of the Maven project file (pom.xml).

c. The third stage is named **"SonarQube Analysis".** Within this stage, the pipeline sets up the environment for SonarQube analysis using the **withSonarQubeEnv** block. The SonarQube scanner is executed with the **clean install** goals and the additional **sonar:sonar** goal to perform the analysis. The SonarQube environment is configured with the name 'josh'.

d. The next stages involve building and working with Docker images. The "Build Docker Image" stage checks the Docker version (**docker version**) and then builds a Docker image with the tag 'king-httpd' (**docker build -t king-httpd .**).

e. The **"Docker Image list"** stage lists the Docker images available on the system (**docker image list**).

f. The **"Docker Image Tag"** stage tags the previously built image as 'josh1991/king-httpd:king-httpd' (**docker tag king-httpd josh1991/king-httpd:king-httpd**).

g. The **"Docker Login to Hub Docker"** stage performs a Docker login operation using the Docker Hub credentials stored in the Jenkins credential with ID 'DOCKER_HUB_PASSWORD'. The credentials are retrieved and passed to the **docker login** command.

h. The **"Docker Image Push"** stage pushes the Docker image 'josh1991/king-httpd:king-httpd' to the Docker Hub repository (**docker push josh1991/king-httpd:king-httpd**).