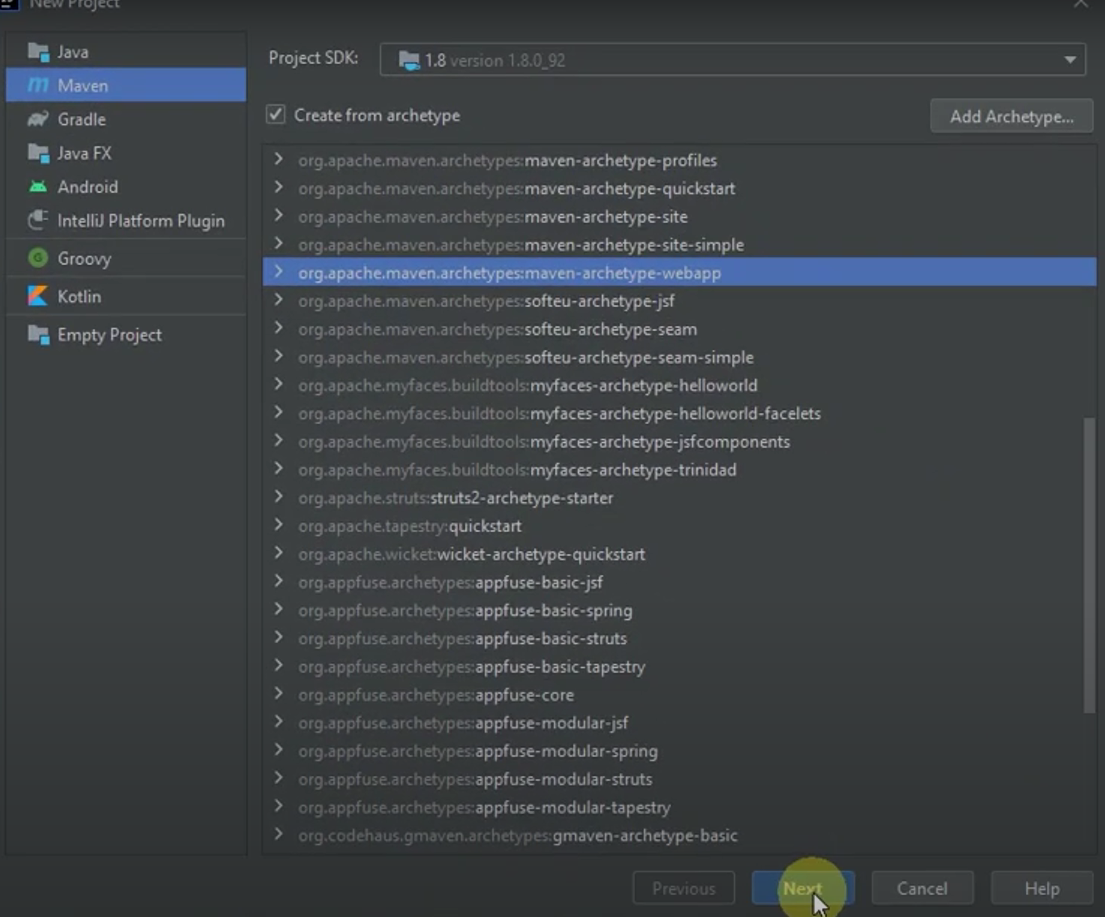
**Create & deploy a sample web app using DevOps methodology**

**1. creating sample web app**

* Download and install IntelliJ IDEA <https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows>
* Open the IntelliJ IDEA and go to New Project and select maven
* After that select SDK version and archetype as maven- archetype- webapp



* After that select the project name and location

Graphical user interface, text, application

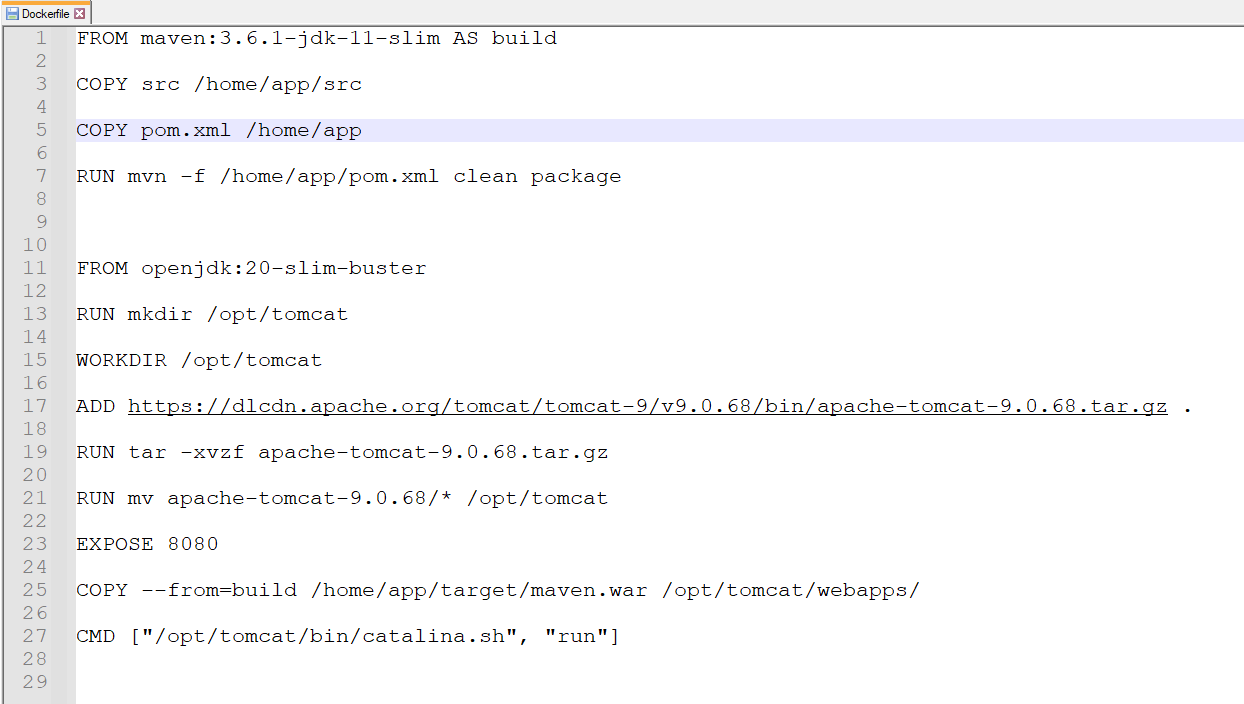
Description automatically generated

Graphical user interface, text, application

Description automatically generated

**2. Create a Dockerfile**

* Dockerfile is use for creating docker image from where container is created. It contain configuration of the application.
* To create Dockerfile, create a file name with “Dockerfile”.
* Add the configuration setting.
* To run the web app in container , required build tool e.g. maven and to host e.g. tomcat for web container.



* Use maven image from dockerhub & copy all the project file to main directory
* Run the maven command to build “mvn clean package”.
* Host the app in web server, required tomcat.
* Taking base image as java from dockerhub and download tomcat from https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.68/bin/apache-tomcat-9.0.68.tar.gz
* Copy build from maven with the help of multi staging to tomcat directory.
* Run the “catlina.sh” command to see the result in 8080 port.

**3. Git**

* Git is version control for tracking files.
* In order to update all the file in Azure directory, using automated git command with the help of shell script.

Graphical user interface, text, application

Description automatically generated

* Adding remote repo url and update all the changes in staged area.
* Commit the changes and push to remote directory.

**4. Manifests**

* Manifests file use for K8 configuration
* Create a deployment.yml file which is used to create no of containers from images.
* Create a service.yml file which is used to establish connections with service with the help of load balancer.

Graphical user interface, application

Description automatically generated with medium confidence

Text

Description automatically generated

**5. CI/CD**

* In order to create a CI/CD pipeline, first we need to create resource.
* Log in to portal.azure.com and create a resource group name as maven and create a container registry in order to store docker image. Then create Kubernetes cluster in order to manage containerized applications.
* Create one more resource group name as sonar. Create container registry and app service.
* Pull a sonar image from docker hub and push into sonar container registry.
* After that go to the dev.azure.com and create a new pipeline and choose azure repo and configure to K8 service.
* Then select cluster and container registry which you have created earlier in maven resource group.
* After that review the azure pipeline and edit in yaml file.
* Yaml file contain the configuration of CI/CD pipeline.
* Create 3 stages Build, waitingapproval and Deploy.
* In Build stage create a sonar task for code review, checks and debug. Create a task for build and push an image to container registry.
* In Build stage configure the sonar and create copyfile and publishpipelineartifact task for manifests file to copy to the pipeline workspace, which will be used in K8.
* In waitingapproval stage create a task for validation for earlier stage in order to proceed to the next stage.
* In last Deploy stage, create a task to pull image from container registry. Create an another task to deploy in k8 cluster with the help of manifests file.
* After reviewing, run the pipeline.

**6. Execution**

* Before running the pipeline, we need to configuration sonar.
* Go to sonar resource group and start app service in order to run sonar.
* Then log in with admin and create a new password.
* Set the configuration name, devops url & access token in order to fetch code from azure.
* Create access token from azure user setting. After that, select azure directory which you want to code review.
* Install SonarQube extension in azure devops and set the server url and token.
* Also update the project key in azure pipeline in build stage.

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

* After that go to the maven resource group and start the k8 service.
* Run the pipeline. After completion of build stage review the approval stage.
* After completion of deploy stage go to the k8 service and start the connection in cloud shell of azure.
* With the help of kubectl command we can get no of nodes, pods, description etc.
* kubectl get nodes- shows no of nodes kubectl get pods- shows no of pods & its status kubectl get describe pod pod\_name- shows the description of specific pod kubectl get svc- show connection type, ports etc
* Copy the external ip in order to see the output.

Graphical user interface, application

Description automatically generated