**Problem Statement**

AppleBite Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd-party vendors.

The company’s goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers.

As development progressed, they are facing multiple problems, because of various technologies involved in the project. Following are the problems:

* Building Complex builds is difficult
* Incremental builds are difficult to manage, and deploy

To solve these problems, they need to implement Continuous Integration & Continuous Deployment with DevOps using following tools:

**Git** – For version control for tracking changes in the code files

**Jenkins** – For continuous integration and continuous deployment

**Docker** – For deploying containerized applications

**Ansible** - Configuration management tools

This project will be about how to do deploy code to dev/stage/prod etc, just on a click of button.

Link for the sample PHP application: <https://github.com/edureka-devops/projCert.git>

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, a new test server should be provisioned with all the required software. Post this, the code should be containerized and deployed on the test server.

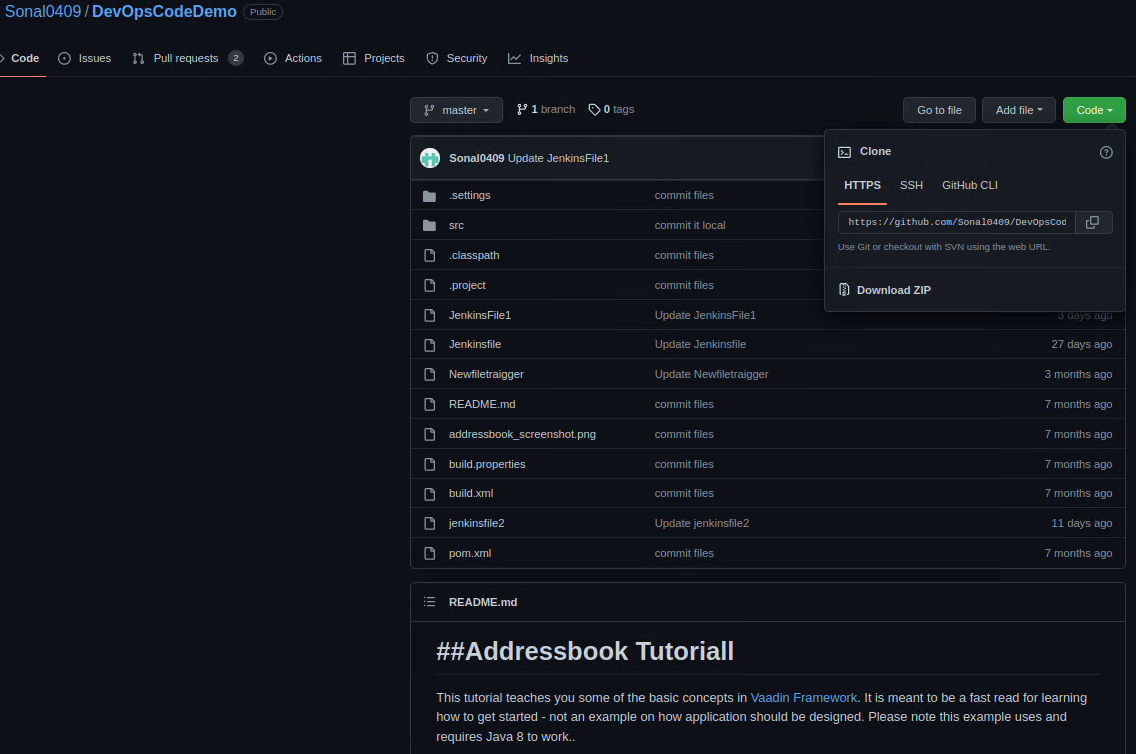
The deployment should then be built and pushed to the prod server.

All this should happen automatically and should be triggered from a push to the GitHub master

**PROJECT**

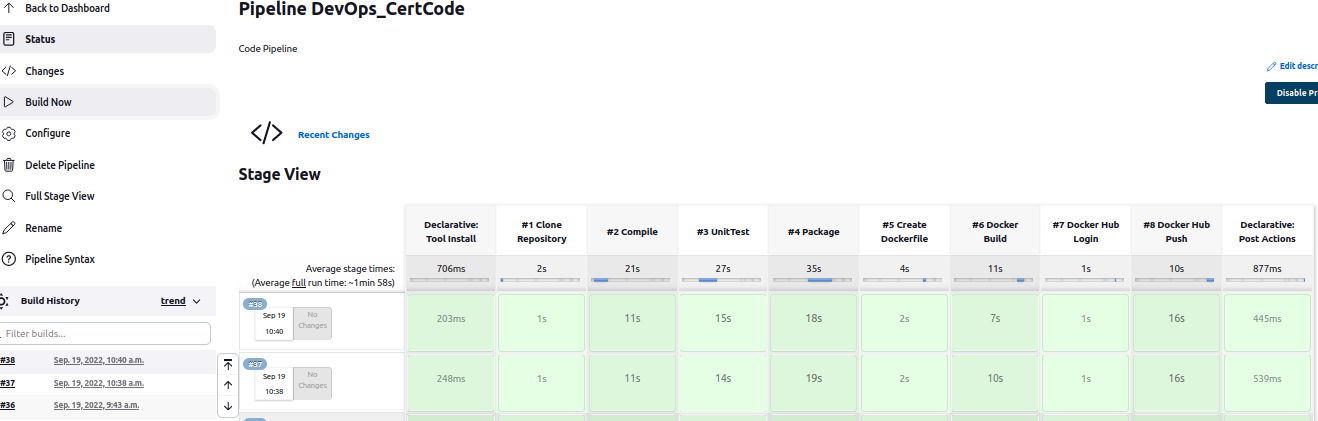
Introduction to Project

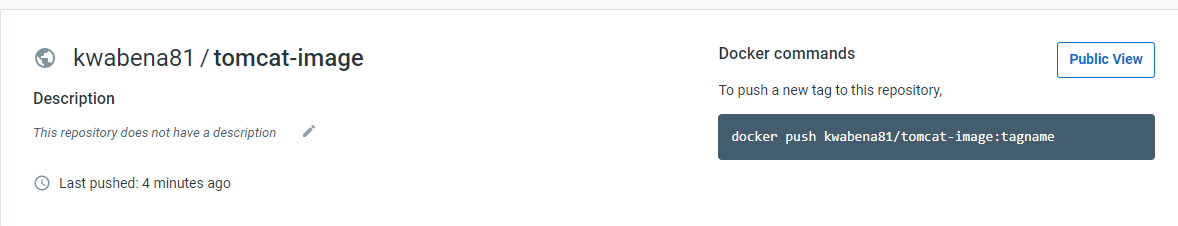
Solution:

1. **Screenshot and link of git repos**
   1. 
2. **Jenkins Pipeline & Dockerfile definition**

pipeline {     
    tools {  
        jdk 'myjava'  
        maven 'mymaven'  
    }  
     
    agent any  
     
    environment {  
DOCKERHUB\_CREDENTIALS=credentials('dockerhub-credential')  
}  
       
    stages {  
         
        stage ('#1 Clone Repository') {  
             
            steps {  
                git '<https://github.com/Sonal0409/DevOpsCodeDemo.git>'  
            }  
        }  
         
        stage ('#2 Compile ') {  
             
            steps {  
                sh 'mvn compile'  
            }  
        }  
         
         stage ('#3 UnitTest ') {  
             
            steps {  
                sh 'mvn test'  
            }  
        }  
         
        stage ('#4 Package ') {  
             
            steps {  
                sh 'mvn package'  
            }  
        }  
         
        stage ('#5 Create Dockerfile') {  
            steps {  
             
                sh 'cp /var/lib/jenkins/workspace/DevOps\_CertCode/target/addressbook.war . '  
                sh 'touch Dockerfile'  
                 
                sh 'echo "FROM tomcat:9" > Dockerfile '  
                 
                sh 'echo "LABEL maintainer: \"[kwabena81@yahoo.com](mailto:kwabena81@yahoo.com)\"" >> Dockerfile '  
                 
                sh 'echo "ADD addressbook.war /usr/local/tomcat/webapps/ " >> Dockerfile '  
                 
                sh 'echo "EXPOSE 8082 " >> Dockerfile '  
                 
                sh 'echo "CMD [\'catalina\', \'sh\'] " >> Dockerfile '  
               
            }  
             
        }  
             
        stage('#6 Docker Build') {  
            steps {  
                sh 'sudo docker build -t kwabena81/tomcat-image .'  
            }  
}  
     
stage('#7 Docker Hub Login') {  
            steps {  
   sh 'echo $DOCKERHUB\_CREDENTIALS\_PSW | sudo docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin'  
   }  
}  
     
stage('#8 Docker Hub Push') {  
       steps {  
   sh 'sudo docker push kwabena81/tomcat-image'  
   }    
   }  
    }  
     
    post {  
   always {  
       sh 'sudo docker logout '  
   }  
}  
}

1. **Jenkins Pipeline definition - OUTPUT**



1. **Docker Hub Push**
2. 
3. Create Jenkins pipeline with stages; Clone Repo 🡪 Compile🡪Unit Test 🡪 Package
4. ~~Screenshot of CI pipeline in Jenkins~~
   1. ~~Create a pipeline job with stages as compile, clone the repo, test and package~~
   2. ~~Share the pipeline code~~
5. ~~Create a Docker file 🡪 create image 🡪 push image 🡪 o Docker hub~~
   1. ~~Screenshot of #3~~
6. Ansible 🡪 git hub repo in this place inventory file with ip address of slave machine
   1. Write your playbook
      1. Install Docker
      2. Start Docker
      3. Run command to run Docker image that was previously pushed to Docker hub
         1. Share the playbook
      4. Include screenshot of container up and running
      5. Share screenshot of address application.

**Problem Statement**

AppleBite Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd-party vendors.

The company’s goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers.

As development progressed, they are facing multiple problems, because of various technologies involved in the project. Following are the problems:

* Building Complex builds is difficult
* Incremental builds are difficult to manage, and deploy

To solve these problems, they need to implement Continuous Integration & Continuous Deployment with DevOps using following tools:

**Git** – For version control for tracking changes in the code files

**Jenkins** – For continuous integration and continuous deployment

**Docker** – For deploying containerized applications

**Ansible** - Configuration management tools

This project will be about how to do deploy code to dev/stage/prod etc, just on a click of button.

Link for the sample PHP application: <https://github.com/edureka-devops/projCert.git>

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, a new test server should be provisioned with all the required software. Post this, the code should be containerized and deployed on the test server.

The deployment should then be built and pushed to the prod server.

All this should happen automatically and should be triggered from a push to the GitHub master