*Local:  
username: Vishnu*

*password: Biscuit@000*

***CI (Continuous Integration):*** *Multiple developers works on different modules, the will push/commit code to single repository , then code will be built, and Run the tests (unit, integration tests)*

*Identify early bugs, faster feedback, save time, cost*

*A cartoon character and a cloud

Description automatically generated with medium confidence*

*CD(Continuous Delivery):*

*Commit code -> Github -> Build code -> Run tests -> Deploy to Test Environment*

*CD(Continuous Deployment):*

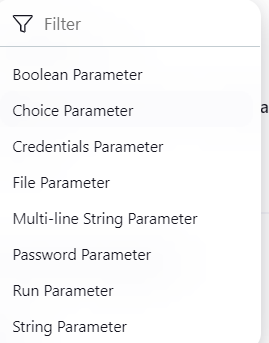
*Commit Code -> Github -> Build code -> Run tests -> Deploy to production directly*

*freestyle:*

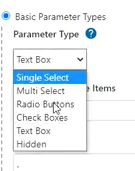
*A screenshot of a computer

Description automatically generated*

*This project is parameterized:default*

**

*Use extended choice parameter plugin : checkbox, radio button (multi-select,*

**

*Manage Jenkins -> Credentials provider*

*Manage Jenkins -> user*

*Triggers:*

*A screenshot of a computer

Description automatically generated*

*MINUTE HOUR DOM MONTH DOW*

|  |  |
| --- | --- |
| *MINUTE* | *Minutes within the hour (0–59)* |
| *HOUR* | *The hour of the day (0–23)* |
| *DOM* | *The day of the month (1–31)* |
| *MONTH* | *The month (1–12)* |
| *DOW* | *The day of the week (0–7) where 0 and 7 are Sunday.* |

*the "Poll SCM" trigger is used to periodically check the source code management (SCM) system for changes. If changes are detected, Jenkins automatically triggers a new build. This feature is particularly useful in environments where you want to automate builds but do not have the ability to configure webhooks or post-commit hooks in the SCM system.*

*\*\*\*\*\* - every minute*

*H\*\*\*\* - every hour*

*H/15 \*\*\*\* - every 15 minutes*

*Plugins:*

1. *.NET SDK support plugin*
2. *MsBuild plugin*
3. *Nunit plugin*
4. *SonarQube plugin*
5. ***SpecFlow+ LivingDoc Plugin***
6. *Email extension plugin*
7. *Git, github plugins*
8. *Test Results Analyzer plugin*
9. *Role strategy plugin*

* *By default Jenkins run tests sequentially*
* *We can configure to run tests parallely*
* *Parallel Test Executor Plugin, Jenkins file(YAML) parallel({})*

*stage('Test') {*

*parallel {*

*stage('Unit Tests') {*

*steps {*

*sh 'sleep 5s'*

*sh 'echo "Running unit tests"'*

*// Add commands to run unit tests*

*}*

*}*

*stage('Integration Tests') {*

*steps {*

*sh 'echo "Running integration tests"'*

*// Add commands to run integration tests*

*}*

*}*

*}*

*Pipeline:*

*A screenshot of a computer

Description automatically generated*

*Advanced project Options*

*Pipeline: script*

*General : Description, Build Triggers, some Options*

*A screenshot of a chat

Description automatically generated*

A screenshot of a computer

Description automatically generated

***Run Jenkins from Docker:***

1. *docker pull jenkins/jenkins:lts*
2. *docker run -d -p 8081:8080 -p 50001:50000 -v jenkins\_home:/var/jenkins\_home --name jenkins jenkins/jenkins:lts*
3. *docker run -d -p 8081:8080 -p 50001:50000 -v C:/Users/Vishnu\_Kolluri/Documents/Docker/JenkinswithDocker:/var/jenkins\_home --name jenkins jenkins/jenkins:lts*
   1. **-p 8081:8080**: Map port 8080 of the container to port 8080 on the host, used for the Jenkins web UI.
   2. **-p 50001:50000**: Map port 50000 of the container to port 50000 on the host, which is used for agent connections. (master, slave)
4. *docker logs Jenkins*
   1. *provides password, copy password and open Jenkins*

*6217c4f138b04354883993a779fb4ef5*

*A screenshot of a computer

Description automatically generated*

*Install suggested plugins*

*Username:VishnuKolluri*

*Password: Biscuit@000*

*Epam mail*

[*http://localhost:8081/*](http://localhost:8081/)

*name: Vishnu Kolluri*

*A screenshot of a computer

Description automatically generated*

*docker-compose up -d*

Azure DevOps pipeline:

Connect:

A screenshot of a computer

Description automatically generated

Select: select the repository

Configure using YAML file

A screenshot of a computer

Description automatically generated

pipeline {

agent any

stages {

stage('Hello') {

steps {

echo 'Hello World'

}

}

}

}

pipeline {

agent any // This specifies that the pipeline can run on any available agent

environment {

// Define any environment variables if required

DOTNET\_CLI\_HOME = "$WORKSPACE/.dotnet"

}

stages {

stage('Clean') {

steps {

// Cleaning up the workspace

echo 'Cleaning the workspace...'

bat "dotnet clean" // Use 'bat' for Windows or 'sh' for Unix/Linux

}

}

stage('Restore') {

steps {

// Restoring the project dependencies

echo 'Restoring dependencies...'

bat "dotnet restore"

}

}

stage('Build') {

steps {

// Building the project

echo 'Building the project...'

bat "dotnet build --configuration Release"

}

}

stage('Test') {

steps {

// Running tests on the project

echo 'Running tests...'

bat "dotnet test --logger \"trx;LogFileName=test\_results.xml\""

}

}

}

post {

always {

// This block can be used to perform actions regardless of the pipeline outcome

echo 'Cleaning up...'

cleanWs() // Cleans up the workspace

}

success {

// Actions to perform if the pipeline succeeds

echo 'Build and test succeeded.'

}

failure {

// Actions to perform if the pipeline fails

echo 'Build or test failed.'

}

}

}