



SOP on CI/CD of TL-Meet & TL-Meet-Post-build

Table of Contents

Contents

SOP on CI/CD of TL-Meet & TL-Meet-Post-build	1
Table of Contents.....	2
TL-Meet.....	3
Aim:	3
Steps to create pipeline for TL-Meet	3
Step 1: Create a Job.....	3
Step 2: Choose Freestyle Project.....	3
Step 3: Configure the Project	4
Step 4: Add Project URL	4
Step 5: Add Source Code Management.....	4
Step 6: Add Branch	4
Step 7: Build Triggers	4
Step 9: Build Environment.....	4
Step 10: Build Steps	5
PURPOSE OF ABOVE COMMANDS.....	5
Step 10: Post-Build Actions	8
TL-Meet-Post-Build	9
Aim:	9
Steps to create pipeline for TL-Meet-Post-Build	10
Step 1: Create a Job.....	10
Step 2: Choose Freestyle Project.....	10
Step 3: Configure the Project	10
Step 4: Add Project URL	10
Step 6: Add Source Code Management.....	10
Step 7: Add Branch	10
Step 8: Build Triggers	10
Step 9: Build Environment.....	10
Send files or execute commands over SSH after the build runs	10
Purpose of above commands.....	11
Provide Node & npm bin/ folder to Path.....	11
Step 10:	12

Document Control

Version History

Version Detail	Date	Prepared By	Reviewed By
V1	13/10/2023	Priya Apotikar	Anand Jain & Ganesh Jadhao
-	-	-	-

TL-Meet

Aim:

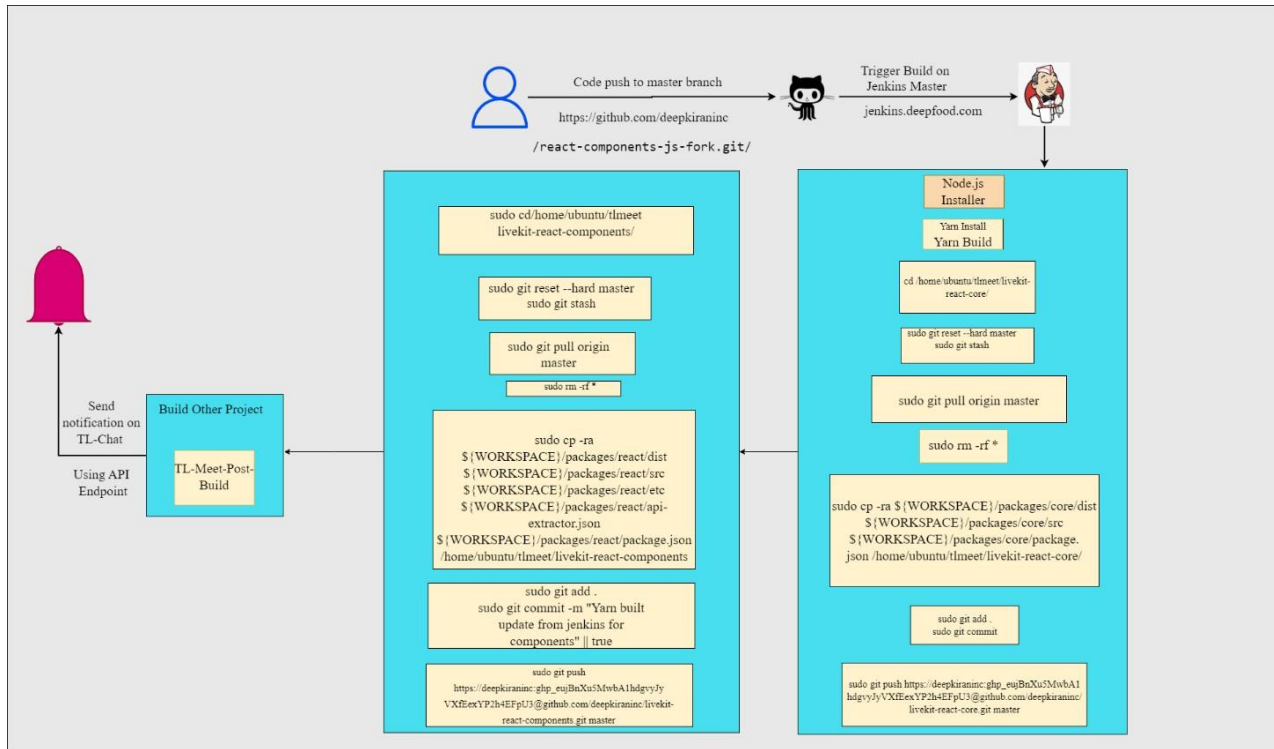


Figure 1: Architecture of TL-Meet

Prerequisite:

- NodeJs 18.16.0
- Git URL

Navigate to Jenkins

jenkins.deepfoods.com

Steps to create pipeline for TL-Meet

Step 1: Create a Job

On the Jenkins dashboard, click on 'New Item' at the left top side of the dashboard.

Step 2: Choose Freestyle Project

Enter a name under 'Enter an item name'. Then select 'Freestyle project' and click on 'OK'.

Step 3: Configure the Project

We can set the project name, description, and other settings.

Project: **Teamlocus**

Pipeline Author's: Priya Apotikar & Anand Jain

Responsible Developer's: **Apurv Bhavsar**

Step 4: Add Project URL:

[https://github.com/deepkiraninc/react-components-js-fork.git/](https://github.com/deepkiraninc/react-components-js-fork.git)

Step 5: Add Source Code Management

Git: Add Repository URL and Credentials

[https://github.com/deepkiraninc/react-components-js-fork.git/](https://github.com/deepkiraninc/react-components-js-fork.git)

Step 6: Add Branch:

***/deep**


Step 7: Build Triggers:

Github hook trigger for GIT Scm

Step 9: Build Environment

Provide Node & npm bin/ folder to Path

- Add Nodejs from Manage Jenkins > Tool>
- Provide required information to add NodeJS
- Provide NodeJS 18.16.0



The screenshot shows the 'Install from nodes.org' configuration form in Jenkins. It includes a checkbox for 'Install automatically', a 'Version' dropdown set to 'NodeJS 18.16.0', a checkbox for 'Force 32bit architecture', a text field for 'Global npm packages to install' containing 'npm install -g yarn turbo firebase-tools @angular/cli', and a text field for 'Global npm packages refresh hours' set to '72'.

Figure 2: Add NodeJS in Manage Jenkins

Step 10: Build Steps:

Add Execute shell Command

This job builds on Jenkins master



```
Command
See the list of available environment variables

yarn install
yarn build

cd /home/ubuntu/tlmeet/livekit-react-core/
sudo git reset --hard origin/master
sudo git stash
sudo git pull origin master
sudo rm -rf *
sudo cp -ra ${WORKSPACE}/packages/core/dist ${WORKSPACE}/packages/core/src ${WORKSPACE}/packages/core/package.json /home/ubuntu/tlmeet/livekit-react-core/
sudo git add .
sudo git commit -m "Yarn built update from jenkins for core" || true
sudo git push https://deepkiraninc:XXXXXXXXXXXXXXXXXXXX@github.com/deepkiraninc/livekit-react-core.git master

cd /home/ubuntu/tlmeet/livekit-react-components/
sudo git reset --hard origin/master
sudo git stash
sudo git pull origin master
sudo rm -rf *
sudo cp -ra ${WORKSPACE}/packages/react/dist ${WORKSPACE}/packages/react/src ${WORKSPACE}/packages/react/etc ${WORKSPACE}/packages/react/api-extractor.
json ${WORKSPACE}/packages/react/package.json /home/ubuntu/tlmeet/livekit-react-components
sudo git add .
sudo git commit -m "Yarn built update from jenkins for components" || true
sudo git push https://deepkiraninc:XXXXXXXXXXXXXXXXXXXX@github.com/deepkiraninc/livekit-react-components.git master
```

Figure 3: Execute shell Command

PURPOSE OF ABOVE COMMANDS

#yarn install
#yarn build

yarn install and yarn build are two common commands used in JavaScript and Node.js development, particularly when managing and building projects using the Yarn package manager. These commands serve different purposes.

□ yarn install:

After running yarn install, the dependencies listed in package.json will be downloaded and installed, making project ready to run

□ yarn build:

yarn build is used for building and preparing project for production deployment. It is a custom script defined in the scripts section of your package.json file.

#cd /home/ubuntu/tlmeet/livekit-react-core/:

This command changes the current working directory to the specified path, "/home/ubuntu/tlmeet/livekit-react-core/," where the project is located.

#sudo git reset --hard master:

This command resets the current branch ("master") to match the remote branch's state. It discards all local changes and switches to the state of the "master" branch as it exists in the remote repository.

#sudo git stash:

This command stashes any uncommitted changes in the working directory, allowing you to save your changes temporarily without committing them.

#sudo git pull origin master:

This command pulls the latest changes from the remote Git repository "https://github.com/deepkiraninc/livekit-react-core.git" into the current branch "master". It updates your local branch with the changes from the remote repository.

*#sudo rm -rf *:*

This command recursively removes all files and directories in the current directory

*# sudo cp -ra \${WORKSPACE}/packages/core/dist \${WORKSPACE}/packages/core/src
\${WORKSPACE}/packages/core/package.json /home/ubuntu/deepkiraninc/livekit-react-core/:*

This command copies the specified files and directories from \${WORKSPACE} to the destination directory "/home/ubuntu/deepkiraninc/livekit-react-core/". It involves copying files related to the "core" package of your project.

#sudo git add .:

This command stages all changes in the current directory for the next commit. It prepares the changes made by the previous cp command for committing.

#sudo git commit -m "Yarn built update from Jenkins for core" || true:

This command commits the staged changes with the provided commit message. The || true is used to ensure that the command returns a success status code even if there are no changes to commit.

#sudo git push origin master:

This command pushes the committed changes to the remote Git repository, specifically to the "master" branch.

#cd /home/ubuntu/tlmeet/livekit-react-components/:

This command changes the current working directory to the specified path, "/home/ubuntu/deepkiraninc/livekit-react-components/", where the project is located.

#sudo git reset --hard master:

This command resets the current branch "master" to match the remote branch's state. It discards all local changes and switches to the state of the "master" branch as it exists in the remote repository.

#sudo git stash:

This command stashes any uncommitted changes in the working directory, allowing you to save your changes temporarily without committing them.

#sudo git pull origin master:

This command pulls the latest changes from the remote Git repository "https://github.com/deepkiraninc/livekit-react-components.git" into the current branch "master". It updates your local branch with the changes from the remote repository.

*#sudo rm -rf *:*

This command recursively removes all files and directories in the current directory.

*#sudo cp -ra \${WORKSPACE}/packages/core/dist \${WORKSPACE}/packages/core/src
\${WORKSPACE}/packages/core/package.json /home/ubuntu/tlmeet/livekit-react-core/*

This command copies the specified files and directories from \${WORKSPACE} to the destination directory "/home/ubuntu/deepkiraninc/livekit-react-components." It appears to involve copying files related to the "react" package of your project.

#sudo git add . :

This command stages all changes in the current directory for the next commit.

#sudo git commit -m "Yarn built update from Jenkins for components" || true:

This command commits the staged changes with the provided commit message. The || true is used to ensure that the command returns a success status code even if there are no changes to commit.

#sudo git push origin master:

This command pushes the committed changes to the remote Git repository, specifically to the "master" branch

Step 10: Post-Build Actions:

"Post-Build Actions" are a set of actions that are executed after the main build process has completed, regardless of whether the build was successful or failed. These actions provide a way to perform additional tasks, such as archiving artifacts, sending notifications, or triggering other jobs, once the build is finished.

The purpose of post-build actions is to automate and streamline various tasks that should occur as part of the build and deployment process.



Figure4: Post-Build Action

Here we Trigger the TLMeet-Post-Build Pipeline.

Step 11:

Once you've added all required details, scroll to the bottom of the page and click Apply, then Save. Here **TLmeet-Post-Build** job will trigger and build.

TL-Meet-Post-Build

This Pipeline trigger after TL-Meet will be successful.

Aim:

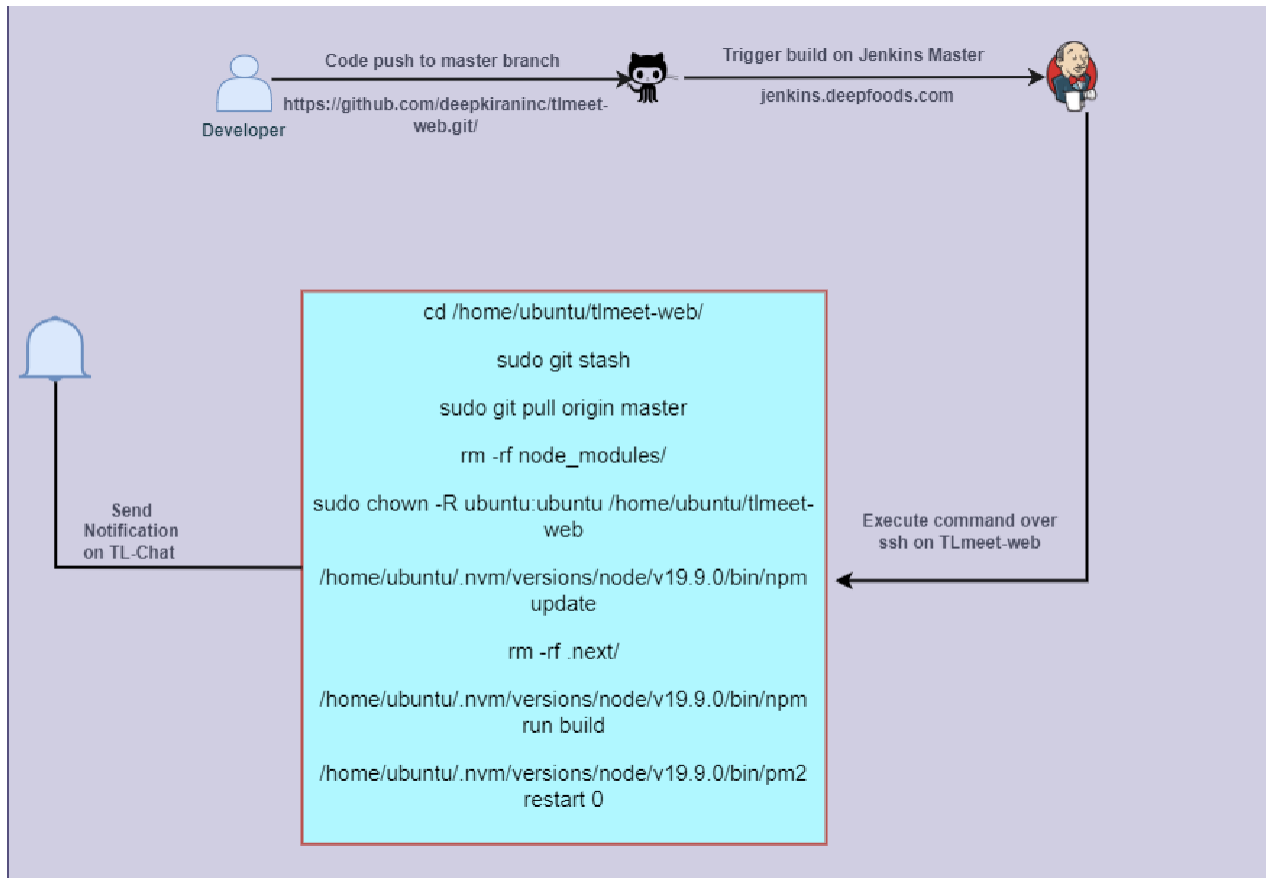


Figure 5: TL-Meet-Post-Build

Prerequisite:

- Git URL
- SSH Sever
- NodeJs 18.16.0

Navigate to Jenkins

jenkins.deepfoods.com

Steps to create pipeline for TL-Meet-Post-Build

Step 1: Create a Job

On the Jenkins dashboard, click on 'New Item' at the left top side of the dashboard.

Step 2: Choose Freestyle Project

Enter a name under 'Enter an item name'. Then select 'Freestyle project' and click on 'OK'.

Step 3: Configure the Project

We can set the project name, description, and other settings.

Project: **Teamlocus**

Pipeline Author's: Priya Apotikar & Anand Jain

Responsible Developer's: **Apurv Bhavsar**

Step 4: Add Project URL:

<https://github.com/deepkiraninc/tlmeet-web.git/>

Step 6: Add Source Code Management

Git: Add Repository URL and Credentials

<https://github.com/deepkiraninc/tlmeet-web.git/>

Step 7: Add Branch:

*/master

Step 8: Build Triggers:

Github hook trigger for GIT SCM

Step 9: Build Environment

Send files or execute commands over SSH after the build runs

Add SSH Publishers: TL-Meet-Web

Add Exec command:

Following commands added in executive shell

```
Exec command ?
source ~/.bashrc
cd /home/ubuntu/tlmeet-web/
sudo git stash
sudo git pull origin master
rm -rf node_modules/
sudo chown -R ubuntu:ubuntu /home/ubuntu/tlmeet-web
/home/ubuntu/.nvm/versions/node/v19.9.0/bin/npm update
rm -rf .next/
/home/ubuntu/.nvm/versions/node/v19.9.0/bin/npm run build
/home/ubuntu/.nvm/versions/node/v19.9.0/bin/pm2 restart 0
```

Figure 6: Executive Shell Command

Purpose of above commands

#git stash:

This command stashes any uncommitted changes in the working directory

#git pull origin master:

This command pulls the latest changes from the remote Git repository

#/home/ubuntu/.nvm/versions/node/v19.9.0/bin/npm update:

This command runs the npm update command using a specific version of Node.js (v19.9.0). It's used to update the packages (dependencies) in your Node.js project to their latest compatible versions as specified in your package.json file.

#rm -rf .next/:

This command removes the .next directory. In the context of a Next.js application, the .next directory typically contains the output of the Next.js build process, including the compiled JavaScript files, assets, and other build artifacts. Removing it is often done before rebuilding the application to ensure a clean and fresh build.

#/home/ubuntu/.nvm/versions/node/v19.9.0/bin/npm run build:

This command runs the npm run build script using the specified version of Node.js.

#/home/ubuntu/.nvm/versions/node/v19.9.0/bin/pm2 restart 3:

This command restarts a process managed by PM2, a process manager for Node.js applications. It's instructing PM2 to restart a process with the ID "3." The process managed by PM2 could be a Node.js application, and restarting it can be useful after deploying a new version or making changes to the application.

Provide Node & npm bin/ folder to Path

- Add Nodejs from Manage Jenkins > Tool>
- Provide required information to add NodeJS
- Provide NodeJS 19.9.0 (default)

SOP on TL Meet & TL Meet Post Build

NodeJS
Name
default

☒ Install automatically ?

Install from nodejs.org

Version
NodeJS 19.9.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install
Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours
Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Add installer

Figure 7: Add NodeJS in Manage Jenkins

☒ Provide Node & npm bin/ folder to PATH

NodeJS Installation
Specify needed nodejs installation where npm installed packages will be provided to the PATH

default

npmrc file
- use system default -

Cache location
Default (~/.npm or %APP_DATA%\npm-cache)

☐ SSH Agent

☐ Terminate a build if it's stuck

☐ With Ant ?

Figure 8: Add NodeJS and execute shell command

Step 10: Once you have added, scroll to the bottom of the page and click Apply, then Save.