

[Back to workspace](#)



Continuous Integration and Continuous Delivery Fundamentals

[Home](#) [Course](#) [Continuous Integration and Continuous Delivery Fundamentals](#) [Labs and Practical Task](#) [Lab3: Continuous Integration and Delivery using Jenkins](#)

[COURSEWARE](#)

[PROGRESS](#)

[DISCUSSION](#)

Navigation



PREVIOUS



NEXT

[Continuous Integration and Continuous Delivery Fundamentals](#)

[Prescreening Quiz](#)



[Theory](#)



[Labs and Practical Task](#)
[practical tasks](#)

Lab1: Continuous Deployment Using GitHub Actions

Lab2: Continuous Integration Using GitLab

Lab3: Continuous Integration and Delivery using Jenkins

Lab4: Maven

Lab5: Gradle

[Complete Course quiz](#)

Lab3: Continuous Integration and Delivery using Jenkins



Task:

Setting up a Multibranch pipeline and regular Jenkins pipelines with manual or auto triggers for deploying an application.

Objective:

The objective of this task is to train systems engineers to set up a Multibranch pipeline and a Manual pipeline for deploying an application with different ports depending on envs and changing logo.svg files depending on envs (branch name) as well. The engineers will learn how to create two branches - main and dev, in GIT these branches will be in envs role as well, configure stages for checkout, build, test, build docker image, and deploy, and make changes to the picture and ports for each branch.

Prerequisites:

To complete this task, the engineers should have the following prerequisites:

- Knowledge of Git, Docker, Jenkins
- Understanding of application deployment process
- Knowledge of basic scripting
- Installed Jenkins on Linux VM and list of plugins: Docker Pipeline, Docker plugin, Git plugin, Groovy, NodeJs plugin, Pipeline.

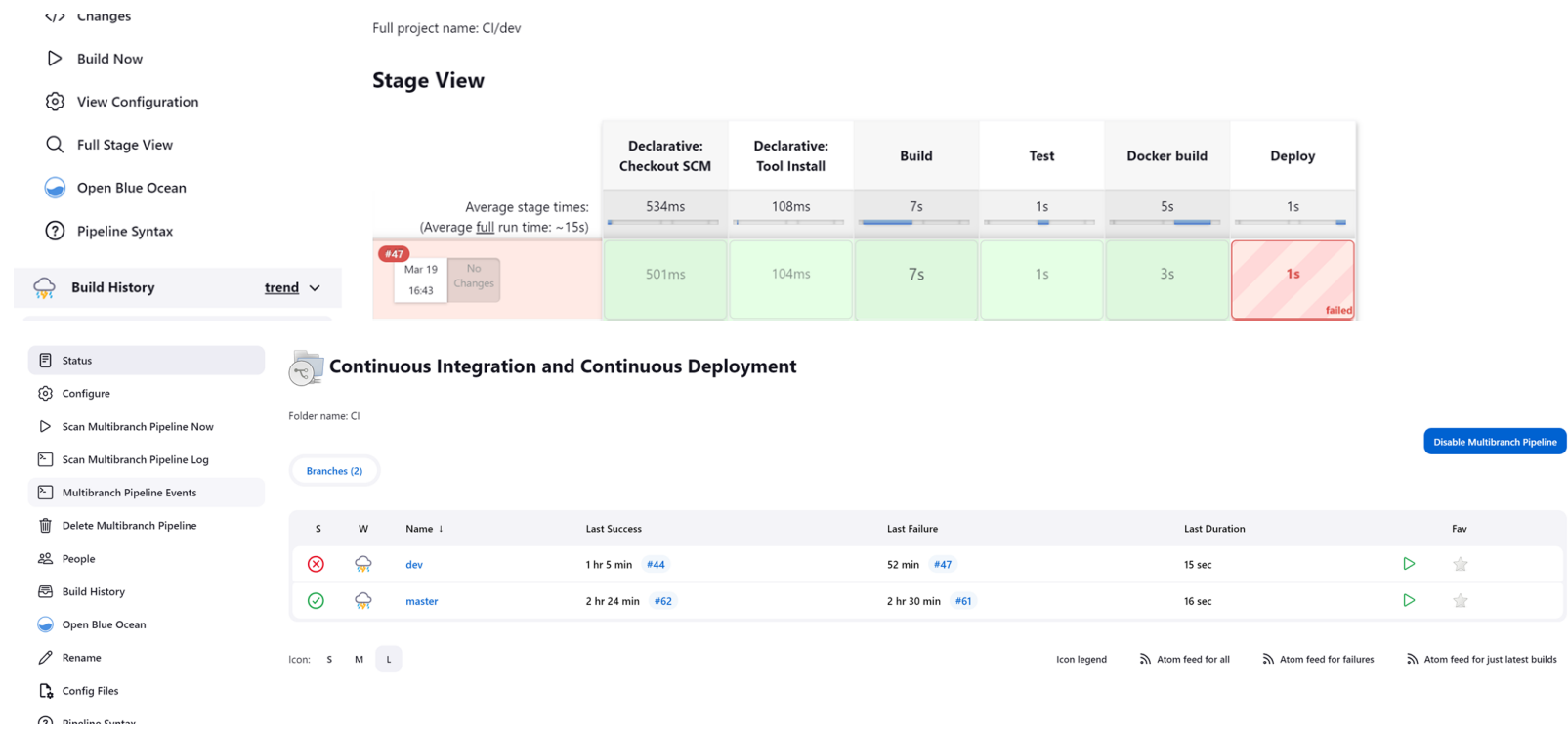
Details:

After triggering main or dev branch you have such stages as checkout – build – test – build docker image – deploy. You must change the picture "logo.svg" for main and dev branches. You can put any pictures you want there but in the .svg format. You should see the difference after application deployment. Also, you should change ports, for the main branch it is 3000 and 3001 for dev as well. It looks like http://localhost:3000 or http://localhost:3001. You perform change of logo.svg in corresponding branches dev and main and you should add conditional logic to your pipeline to set port number depending on branch.

Global goal is creation of two Jenkins pipelines. The first is a multibranch pipeline called "CICD", the second one is a regular pipeline called

"CD_deploy_manual" and it should be manually triggered to execute deployment. You should pull this repo - <https://github.com/epam-msdp/cicd-pipeline> to your local machine, then create your own repo on GitHub and push files contained inside this repo, after this you must create a new branch called dev and, in the end, you will see two branches "main" and "dev"

Predictive result:



Steps:

Click the arrows to see more information.

1. Configure global tools configuration Node 7.8.0.

The screenshot shows the "Global Tool Configuration" page for "NodeJS" in Jenkins.

NodeJS installations (Edited)

List of NodeJS installations on this system

Add NodeJS

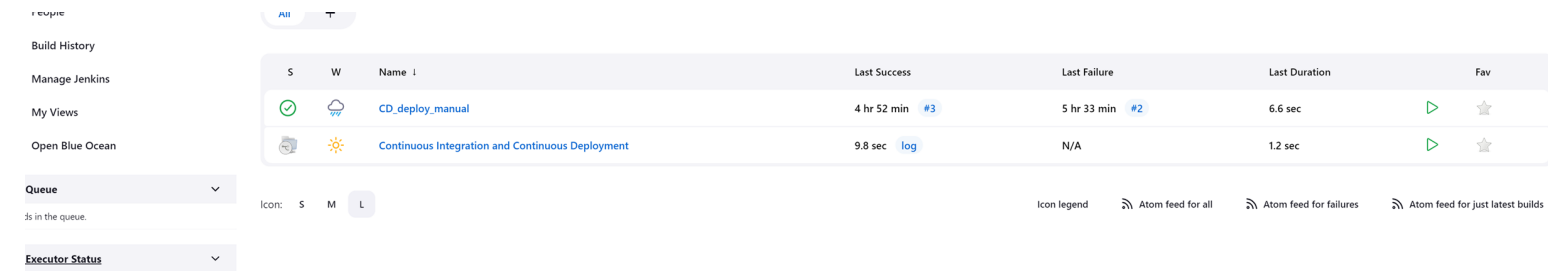
NodeJS Name: node

☒ Install automatically ?

Install from nodejs.org

Version: NodeJS 7.8.0

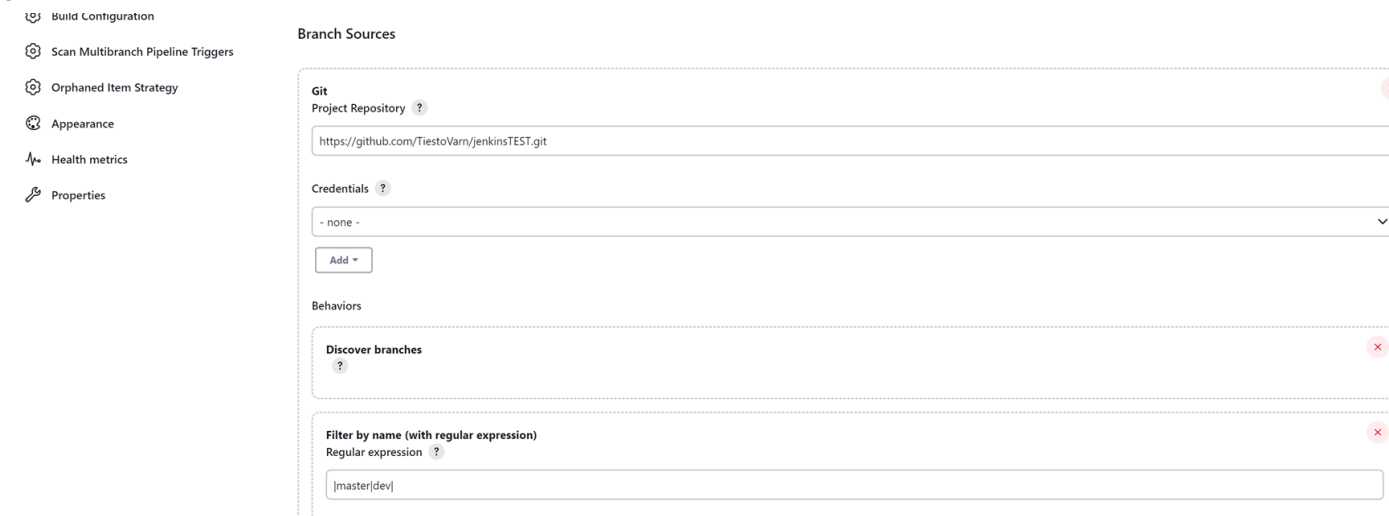
2. Create multibranch and one regular pipeline.



The screenshot shows the Jenkins dashboard with a table of builds. The table has columns for status (S, W), name, last success, last failure, last duration, and favorite status. Two builds are listed: 'CD_deploy_manual' and 'Continuous Integration and Continuous Deployment'.

S	W	Name	Last Success	Last Failure	Last Duration	Fav
✓	☁	CD_deploy_manual	4 hr 52 min #3	5 hr 33 min #2	6.6 sec	▶ ☆
☁	☀	Continuous Integration and Continuous Deployment	9.8 sec log	N/A	1.2 sec	▶ ☆

3. Configure GIT.



The screenshot shows the 'Branch Sources' configuration page in Jenkins. It includes a sidebar with options like 'Build Configuration', 'Scan Multibranch Pipeline Triggers', 'Orphaned Item Strategy', 'Appearance', 'Health metrics', and 'Properties'. The main content area is titled 'Branch Sources' and contains a 'Git' section with fields for 'Project Repository' (https://github.com/TiestoVarn/jenkinsTEST.git), 'Credentials' (none), and 'Behaviors' (Discover branches, Filter by name (with regular expression) [master|dev]).

4. Set up scan multibranch pipeline every 1 min. (Ignore it if you have a public IP and you can set up webhooks in GitHub - it is much better way).

5. Configure role base access. Create three groups (dev, qa, devops and grant different access rights) for example devops can do anything with Jenkins. Dev team can just run jobs and QA can just read logs in the pipelines but can't run.

6. Configure pipeline and stages of CICD pipeline.

- The Jenkinsfile should be the same for main and dev branches.
- Setup your NodeJS platform version in “global tools configuration”.
- Set up GitHub ssh creds for Jenkins.
- Command for build of NodeJS application - `npm install`.
- Command for testing of NodeJS application - `npm test`.
- Build docker images:
for main
`docker build -t nodemain:v1.0.`
for dev
`docker build -t nodedev:v1.0.`
- Before you run your container you should stop and delete all previously running containers. Try to make the lowest downtime.
- Run your application in docker container:
for main branch
`docker run -d --expose 3000 -p 3000:3000 nodemain:v1.0`
for dev branch
`docker run -d --expose 3001 -p 3001:3000 nodedev:v1.0.`

7. "Create a new pipeline called CD_deploy_manual, which should be used for manually deploying to either the 'main' or 'dev' environment. Since we do not have separate environments, 'main' and 'dev' branches are used to imitate real environments with different application versions. This pipeline should include two parameters:

- 'main/dev', to specify the target environment
- 'Image tag', to specify the tag of the Docker image to be deployed."

Advanced tasks:

Click the arrows to see more information.

In result you will have two different docker image IDs locally (nodemain:v1.0 and nodedev:v1.0). Right now, during deployment our pipeline deletes all containers, but since we have different envs, it is better to keep containers not related to selected env untouched. Try to adjust pipeline to delete containers only for deployed env.

Create your own repository in docker hub. Setup docker credentials in Jenkins and push your newly created images into your repository. In this case you should create two additional pipelines called "Deploy_to_main" and "Deploy_to_dev" and trigger them automatically at the end of Multibranch pipeline depending on branch. These two pipelines will pull your images from repository and deploy them into matching env (Docker pull nodemain:v1.0 and docker pull nodedev:v1.0 for dev branch and next docker run -d -expose -p 3000:3000 your image). Please pay attention to the fact that you should fix your existing Multibranch pipeline with push stage.

Create SharedLib for this project (**Documentation**).

```
@Library([ 'JenkinsTesLib', 'jenkinslib@master' ]) _
DeployToMaster(anyparam: "anyvalue")
```

Add vulnerability scanning for docker images using Trivy.

Add additional check for docker file using **Hadollint** before build stage.

Use docker agents for build.

Appendix:

Click each heading to see more information.

- GitHub API token scopes for Jenkins

Jenkins' **scope requirements** depend on the task/s you would like to perform:

- `admin:repo_hook` - for managing hooks at GitHub Repositories level including for **Multibranch Pipeline**
- `admin:org_hook` - for managing hooks at GitHub Organizations level for **GitHub Organization Folders**
- `repo` - to see **private repos**. Please note that this is a parent scope, allowing full control of private repositories that includes:
- `repo:status` - to manipulate commit statuses
- `repo:repo_deployment` - to manipulate deployment statuses
- `repo:public_repo` - to access to public repositories
- `read:org` and `user:email` - recommended minimum for **GitHub OAuth Plugin**- scopes.

Search GitHub Pull requests Issues Gist

Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys
- Security
- Blocked users
- Repositories
- Organizations
- Saved replies
- Authorized applications
- Installed integrations

Developer settings

- OAuth applications
- Integrations
- Personal access tokens**

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Token description

support-token

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> <code>repo:status</code>	Access commit status
<input checked="" type="checkbox"/> <code>repo_deployment</code>	Access deployment status
<input checked="" type="checkbox"/> <code>public_repo</code>	Access public repositories
<input type="checkbox"/> admin:org	Full control of orgs and teams
<input type="checkbox"/> <code>write:org</code>	Read and write org and team membership
<input type="checkbox"/> <code>read:org</code>	Read org and team membership
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> <code>write:public_key</code>	Write user public keys
<input type="checkbox"/> <code>read:public_key</code>	Read user public keys
<input type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> <code>write:repo_hook</code>	Write repository hooks
<input type="checkbox"/> <code>read:repo_hook</code>	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks

How to add Trivy to Jenkins pipeline

```
stage('Scan Docker Image for Vulnerabilities') {
  steps {
    script {
      def vulnerabilities = sh(script: "trivy image --exit-code 0 --severity HIGH,MEDIUM,LOW --no-progress ${registry}:${env.BUILD_ID}", returnStdout: true).trim()
      echo "Vulnerability Report:\n${vulnerabilities}"
    }
  }
}
```

As a result, you should provide us with one .pdf file that includes:

1. Two screenshots of Jenkins UI with your pipelines
2. Two Jenkinsfile created by Jenkins
3. Two screenshots of browser with deployed application from main and dev branches
4. Any additional tasks related code if tasks were done

Upload the .pdf file to the platform using the "Upload your assignment" button below and click "Submit".

Please pay attention, that you have only one attempt to submit your file!



Unfortunately, checking this task is not yet automated. Therefore, your solution may be left unchecked if you take this course without the support of a mentor. Otherwise, provide your mentor with access to your private GitLab repository. This will allow them to review your work and provide feedback and guidance as needed.

Please name your .pdf in the following way: Cloud_DevOps_CICD_[your_first_name]_[your_last_name].pdf

Upload your assignment