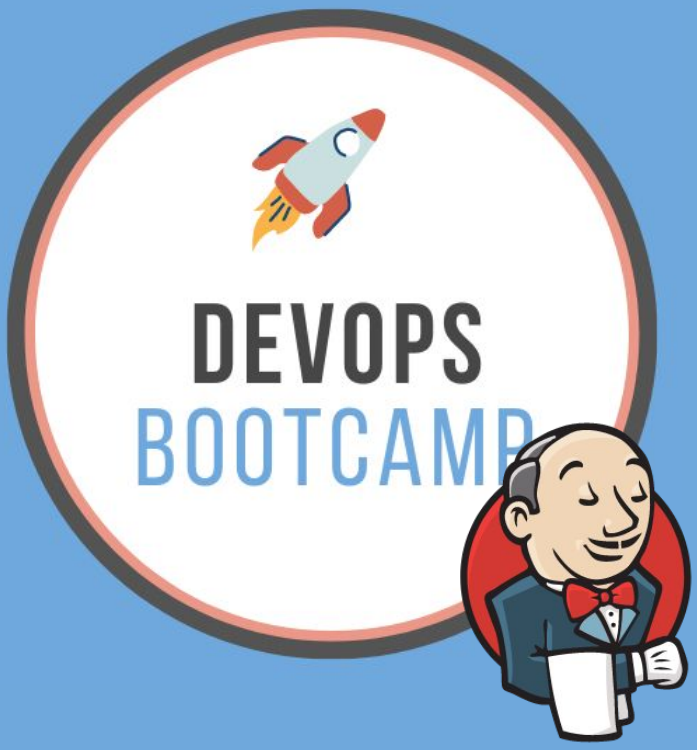
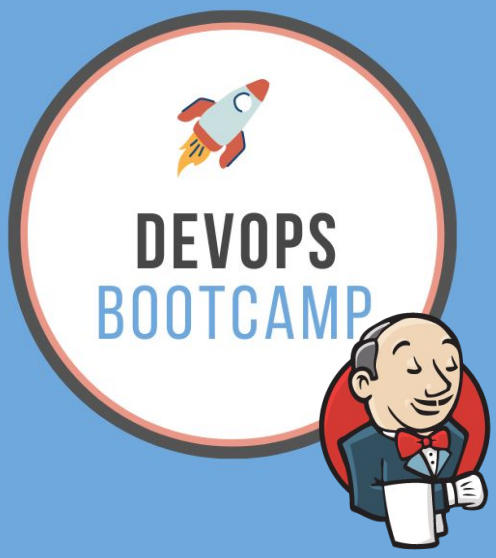



## Module Checklist

# Build Automation & CI/CD with Jenkins

By Techworld with Nana




# Video Overview

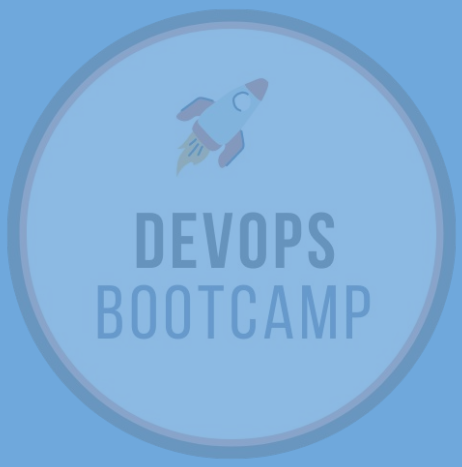
- ★ Introduction to Build Automation
- ★ Install Jenkins
- ★ Jenkins UI Tour
- ★ Install Build Tools
- ★ Jenkins Basics Demo
- ★ Docker in Jenkins
- ★ Freestyle to Pipeline Job
- ★ Introduction to Pipeline Job
- ★ Jenkinsfile Syntax
- ★ Create full Pipeline
- ★ Introduction to Multibranch Job
- ★ Wrap Up Jenkins Job
- ★ Credentials in Jenkins
- ★ Jenkins Shared Library
- ★ Trigger Jenkins Job - Webhook
- ★ Versioning your application - Part 1
- ★ Versioning your application - Part 2

| Demo Projects | |
|---|---|
| Java Maven Project | https://gitlab.com/nanuchi/java-maven-app |
| Jenkins Shared Library | https://gitlab.com/nanuchi/jenkins-shared-library |

# Check your progress... 1/7

## Introduction to Build Automation

- ❏ Watched video




## Install Jenkins

- ❏ Watched video
- ❏ **Demo executed**
  - ❏ Created a Server (Droplet) on DigitalOcean
  - ❏ Configured Firewall Rules to open port 22 and port 8080 for our new server
  - ❏ Installed Docker on DigitalOcean Droplet
  - ❏ Started Jenkins Docker container with named volume
  - ❏ Initialized Jenkins

**Useful Links:**

- Jenkins Docker Image: https://hub.docker.com/r/jenkins/jenkins

**Useful Commands:**

```
docker run -p 8080:8080 -p 50000:50000 -d -v
jenkins_home:/var/jenkins_home jenkins/jenkins:lts
docker volume inspect jenkins_home

# print the initialisation password
cat
/var/snap/docker/common/var-lib-docker/volumes/jenkins_home/
_data/secrets/initialAdminPassword
```

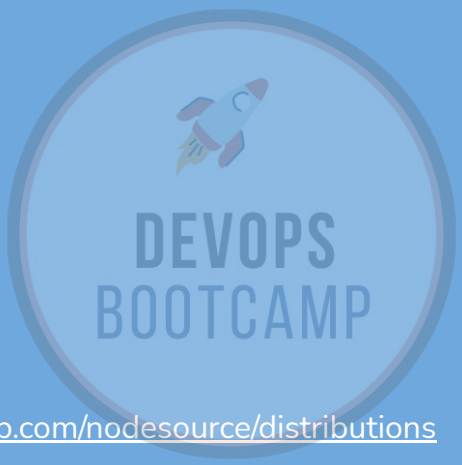
## Jenkins UI Tour

- ❏ Watched video

# Check your progress... 2/7

## Install Build Tools

- ❑ Watched video
- ❑ **Demo executed**
  - ❑ Configured Plugin for Maven
  - ❑ Installed npm and node in Jenkins container




**Useful Links:**

- Node Installation for Linux distributions: https://github.com/nodesource/distributions
- Jenkins Plugins: https://plugins.jenkins.io/

**Useful Commands:**

```
# enter container as root
docker exec -u 0 -it a3f305b0b63f bash

# check with Linux distro container is running
cat /etc/issue

apt update
apt install curl

curl -sL https://deb.nodesource.com/setup_10.x -o nodesource_setup.sh
bash nodesource_setup.sh
apt install nodejs

nodejs -v
npm -v
```

## Jenkins Basics Demo

- ❑ Watched video
- ❑ **Demo executed - create a simple Freestyle Job**
  - ❑ Configured Git Repository to checkout the code from
  - ❑ Configured Job to run tests and build Java Application

**Useful Links:**

- Java-maven-app (master): https://gitlab.com/nanuchi/java-maven-app
- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs

# Check your progress... 3/7

## Docker in Jenkins

- ❏ Watched video
- ❏ **Demo executed - push Image to DockerHub Repository**
  - ❏ Made Docker available in Jenkins container (mount docker runtime inside container as a volume)
  - ❏ Fixed permissions on docker.sock
  - ❏ Configured Job to build Docker Image
  - ❏ Configured Job to push Image to DockerHub
    - ❏ Prerequisite: Account on DockerHub
    - ❏ Created a repository on DockerHub
    - ❏ Created Credentials for DockerHub in Jenkins UI
    - ❏ Tag Docker Image with your DockerHub repository, login and push to repository
- ❏ **Demo executed - push Image to Nexus Repository**
  - ❏ Configured “insecure-registries” on Droplet server (daemon.json file)
  - ❏ Fixed permission for docker.sock again after restart of Jenkins container
  - ❏ Created Credentials for Nexus in Jenkins UI
  - ❏ Tag Docker Image with your Nexus host and repository, login and push to repository

**Useful Links:**

- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs

# Check your progress... 4/7

## Docker in Jenkins

**Useful Commands:**

```
# create jenkins container with mounted docker
docker run -p 8080:8080 -p 50000:50000 -d -v jenkins_home:/var/jenkins_home -v
/var/run/docker.sock:/var/run/docker.sock -v $(which docker):/usr/bin/docker jenkins/jenkins:lts

# enter as root and modify docker.sock permission
docker exec -u 0 -it 0c73a1692b75 bash
chmod 666 /var/run/docker.run

# edit or create /etc/docker/daemon.json ON THE HOST
vim /etc/docker/daemon.json
{
  "insecure-registries" : ["167.99.248.163:8083"]
}

# restart docker
systemctl restart docker

# start container again
docker start 0c73a1692b75

# enter as root and modify docker.sock permission
docker exec -u 0 -it 0c73a1692b75 bash
chmod 666 /var/run/docker.run
```

## Freestyle to Pipeline Job

- ❑ Watched video

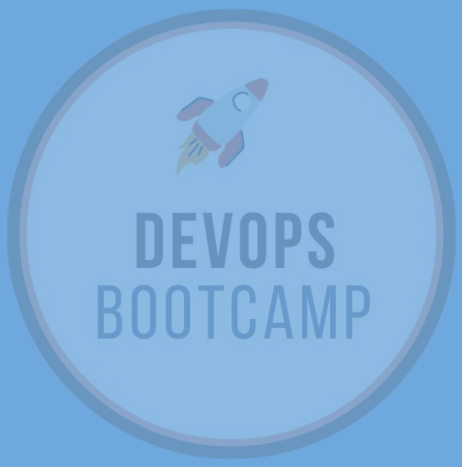
## Introduction to Pipeline Job

- ❑ Watched video
- ❑ **Demo executed - create a basic Pipeline Job**
  - ❑ Configured Git Repository
  - ❑ Created a valid Jenkinsfile with required fields

**Useful Links:**

- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs

# Check your progress... 5/7




## Jenkinsfile Syntax

- ❑ Watched video
- ❑ **Demo executed**
  - ❑ Used Post attribute
  - ❑ Defined a Condition
  - ❑ Used an environment variable
  - ❑ Used Tools Attribute
  - ❑ Used a Parameter
  - ❑ Used an external Groovy Script
  - ❑ Used an Input Parameter

**Useful Links:**

- List available environment variables: *<your-ip-of-jenkins>:8080*//env-vars.html
- Pipeline Syntax: https://www.jenkins.io/doc/book/pipeline/syntax/
- Jenkinsfile & Groovy example: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs/Jenkinsfile-syntax

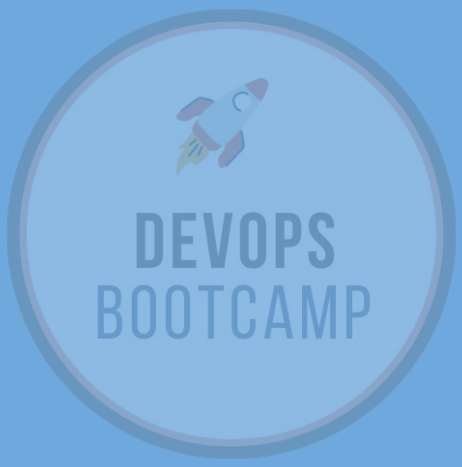
## Create full Pipeline

- ❑ Watched video
- ❑ **Demo executed - create a full Pipeline Job**
  - ❑ Build Jar
  - ❑ Build Docker Image
  - ❑ Push to private Repository DockerHub

**Useful Links:**

- Jenkinsfile & Groovy example: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs/Jenkinsfile-simple-pipeline

# Check your progress... 6/7




## Introduction to Multibranch Job

- ❑ Watched video
- ❑ **Demo executed**
  - ❑ Added branch based logic in Jenkinsfile

## Wrap Up Jenkins Jobs

- ❑ Watched video

## Credentials in Jenkins

- ❑ Watched video

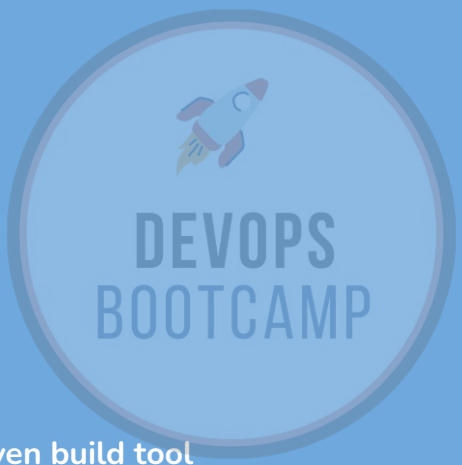
## Jenkins Shared Library

- ❑ Watched video
- ❑ **Demo executed**
  - ❑ Created Shared Library Project/Repository
  - ❑ Made Shared Library globally available in Jenkins
  - ❑ Used Shared Library in Jenkinsfile
  - ❑ Used Parameters in Shared Library
  - ❑ Extracted logic into Groovy Classes
  - ❑ Define Shared Library in Jenkinsfile directly (project scoped)

**Useful Links:**

- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-shared-lib
- Jenkins-shared-library: https://gitlab.com/nanuchi/jenkins-shared-library

# Check your progress... 7/7




## Trigger Jenkins Jobs - Webhooks

- ❏ Watched video
- ❏ **Demo executed**

## Versioning your application - Part I

- ❏ Watched video
- ❏ **Demo executed: Increment version locally with maven build tool**
- ❏ **Demo executed: Increment version in Jenkins Pipeline**
  - ❏ Configured Jenkinsfile to increment version
  - ❏ Adjusted Dockerfile file
  - ❏ Executed Jenkins Pipeline

**Useful Links:**

- Maven Build-Helper Plugin: https://www.mojohaus.org/build-helper-maven-plugin/parse-version-mojo.html
- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs/Jenkinsfile-version-increment

## Versioning your application - Part II

- ❏ Watched video
- ❏ **Demo executed: Commit version upgrade from Jenkins to Git**
- ❏ **Demo executed: Ignore Jenkins Commit from Triggering Pipeline**

**Useful Links:**

- Java-maven-app: https://gitlab.com/nanuchi/java-maven-app/-/tree/jenkins-jobs/Jenkinsfile-version-increment

# More Resources...

## Best practices

**Pipeline Script/Jenkinsfile related best practices:**

- Use Pipeline as Code: Store your Jenkinsfile in Git Repository (instead of inline script in Jenkins). Good for history, working in a team etc.
- Call your Pipeline Script the default name: Jenkinsfile
- Non-Setup work should occur within a stage block. Makes your builds easier to visualize, debug etc.
- Make sure to put #!/usr/bin/env groovy at the top of the file so that IDEs, GitHub diffs, etc properly detect the language and do syntax highlighting for you.
- Input Parameter: input shouldn't be done within a node block. It's recommended to use timeout for the input step in order to avoid waiting for an infinite amount of time, and also control structures (try/catch/finally).

**Other best practices:**

- Use Automatic versioning
- Store common pipeline code in a Shared Library, so that it can be used by other projects/teams

- Best practices and how to put it to practice: https://www.lambdatest.com/blog/jenkins-best-practices/