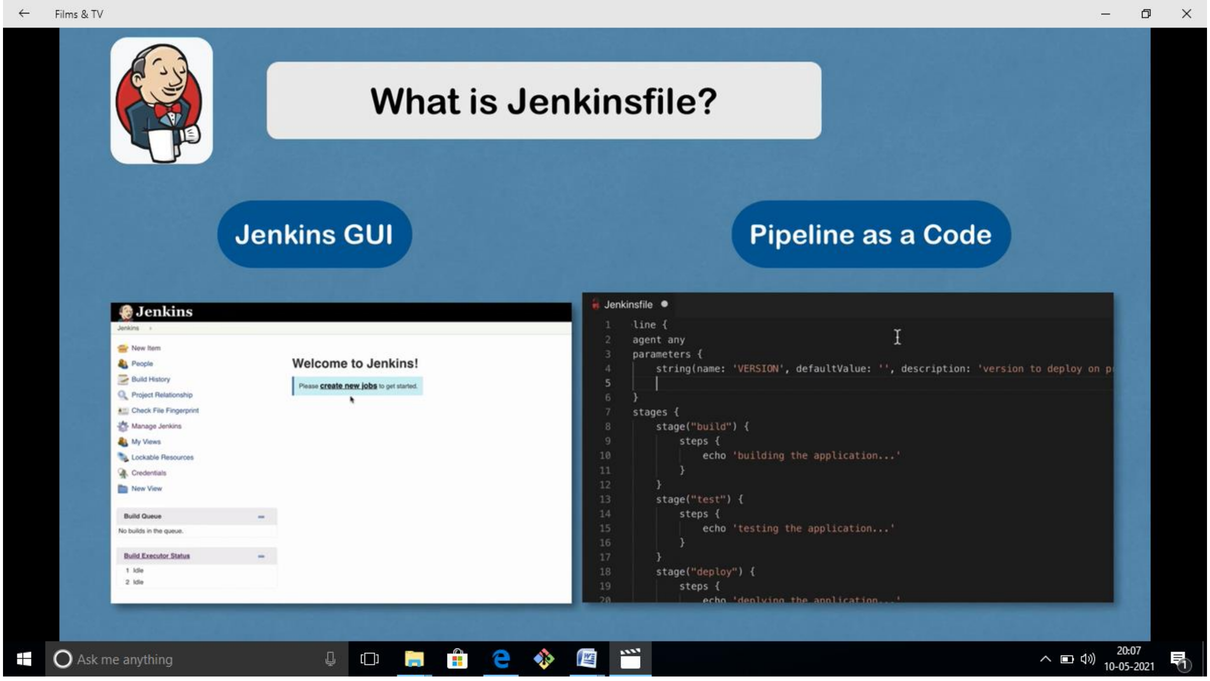
JENKINS IS PIPELINE AS CODE



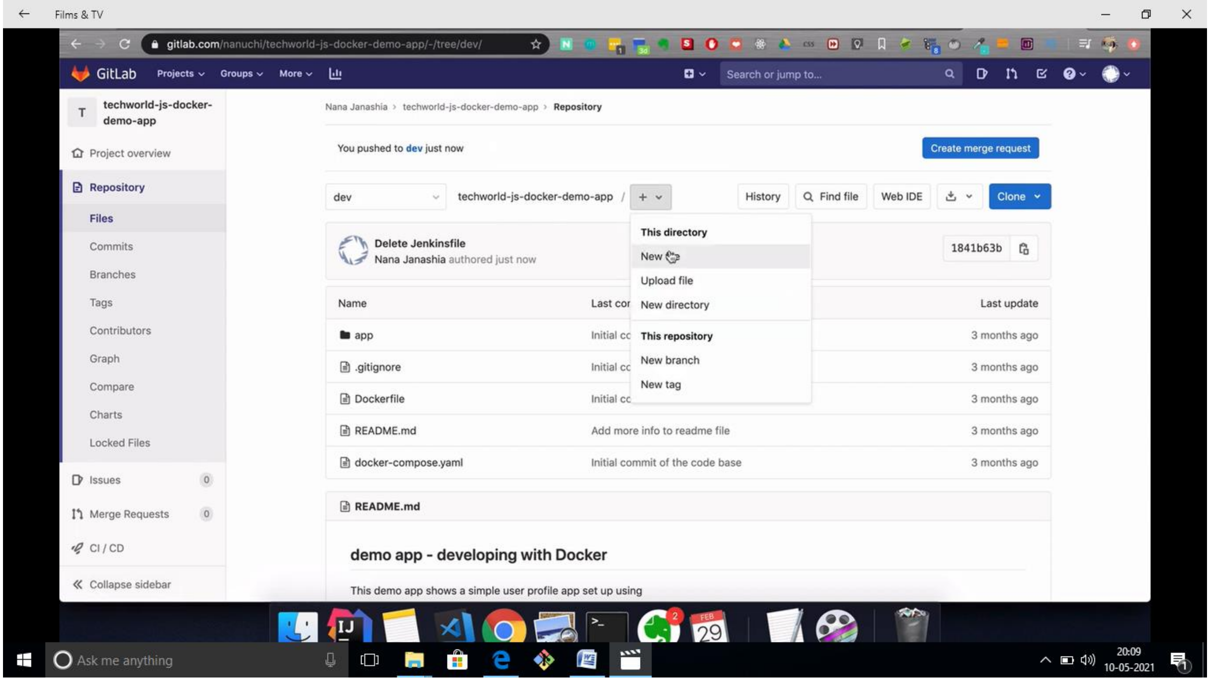
What is Jenkinsfile?

Jenkins GUI

Pipeline as a Code

```
line {
agent any
parameters {
    string(name: 'VERSION', defaultValue: '', description: 'version to deploy on p

}
stages {
    stage("build") {
        steps {
            echo 'building the application...'
        }
    }
    stage("test") {
        steps {
            echo 'testing the application...'
        }
    }
    stage("deploy") {
        steps {
            echo 'deploying the application...'
```

CREATE A NEW FILE IN GITHUB



You pushed to dev just now

Delete Jenkinsfile
Nana Janashia authored just now

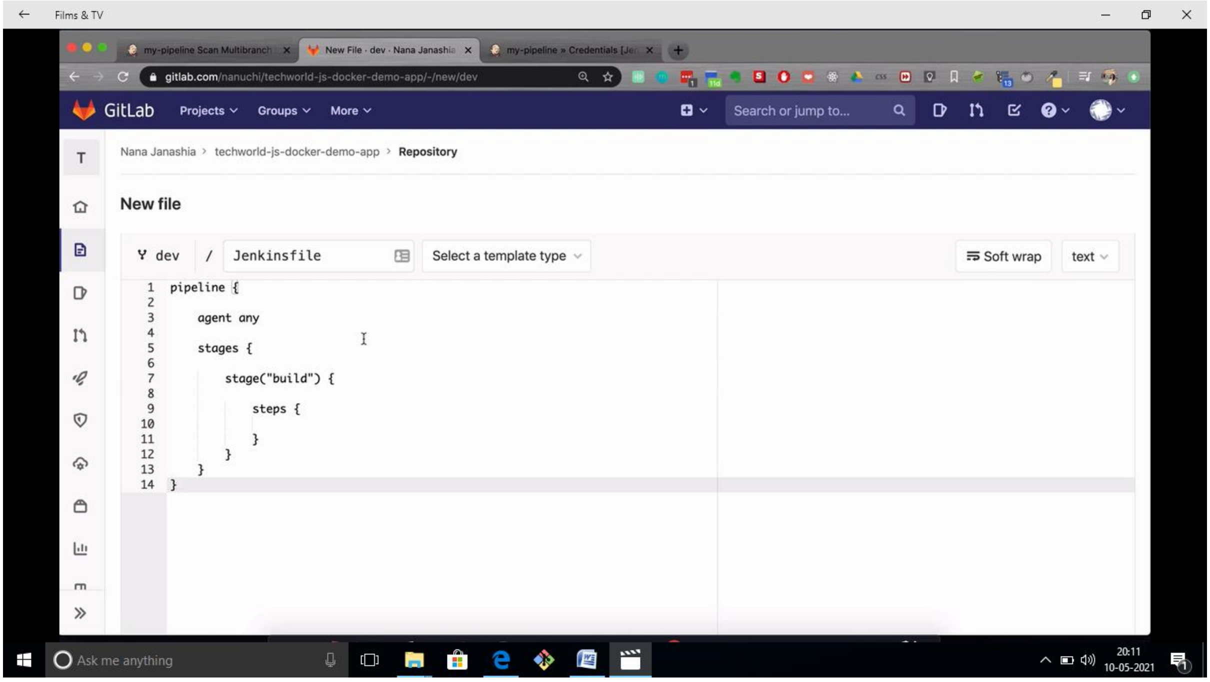
Name	Last cor	Last update
app	Initial cc	3 months ago
.gitignore	Initial cc	3 months ago
Dockerfile	Initial cc	3 months ago
README.md	Add more info to readme file	3 months ago
docker-compose.yaml	Initial commit of the code base	3 months ago

README.md

demo app - developing with Docker

This demo app shows a simple user profile app set up using

BASIC SYANTX



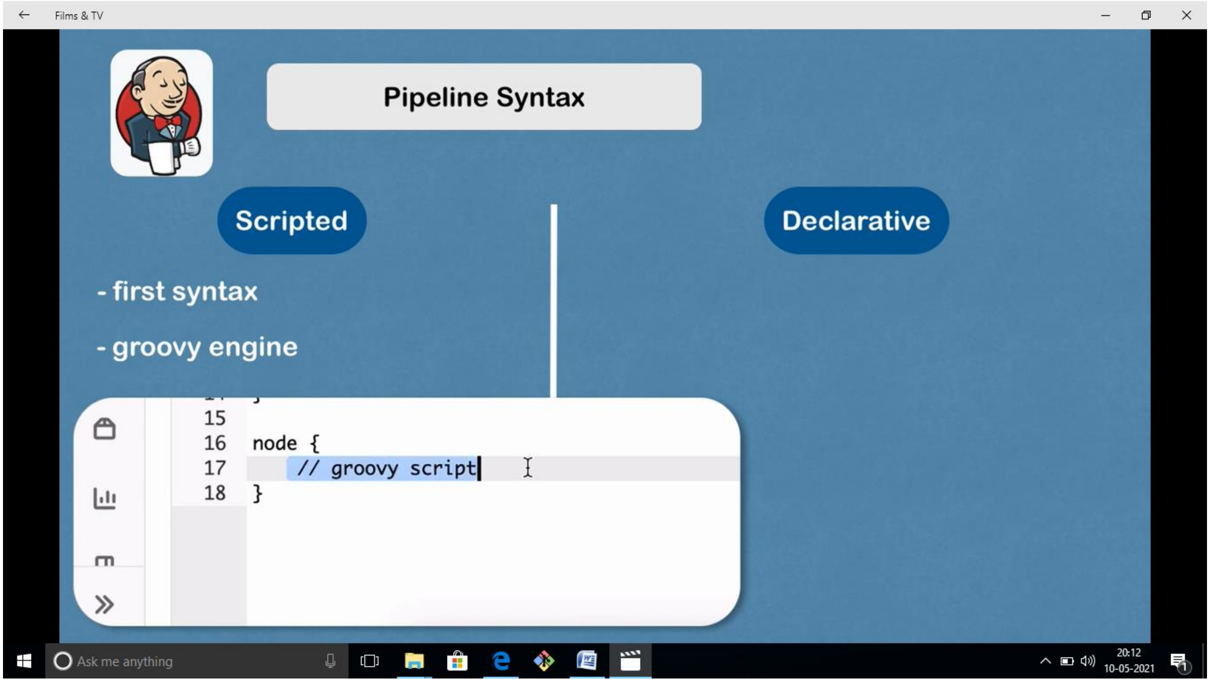
Films & TV
my-pipeline Scan Multibranch
New File · dev · Nana Janashia
my-pipeline » Credentials [Jenkins]
gitlab.com/nanuchi/techworld-js-docker-demo-app/-/new/dev
GitLab
Projects
Groups
More
Search or jump to...
Select a template type
Soft wrap
text
Ask me anything
20:11
10-05-2021

Nana Janashia > techworld-js-docker-demo-app > Repository

New file

dev / Jenkinsfile

```
pipeline {

    agent any

    stages {

        stage("build") {

            steps {

            }
        }
    }
}
```



Pipeline Syntax

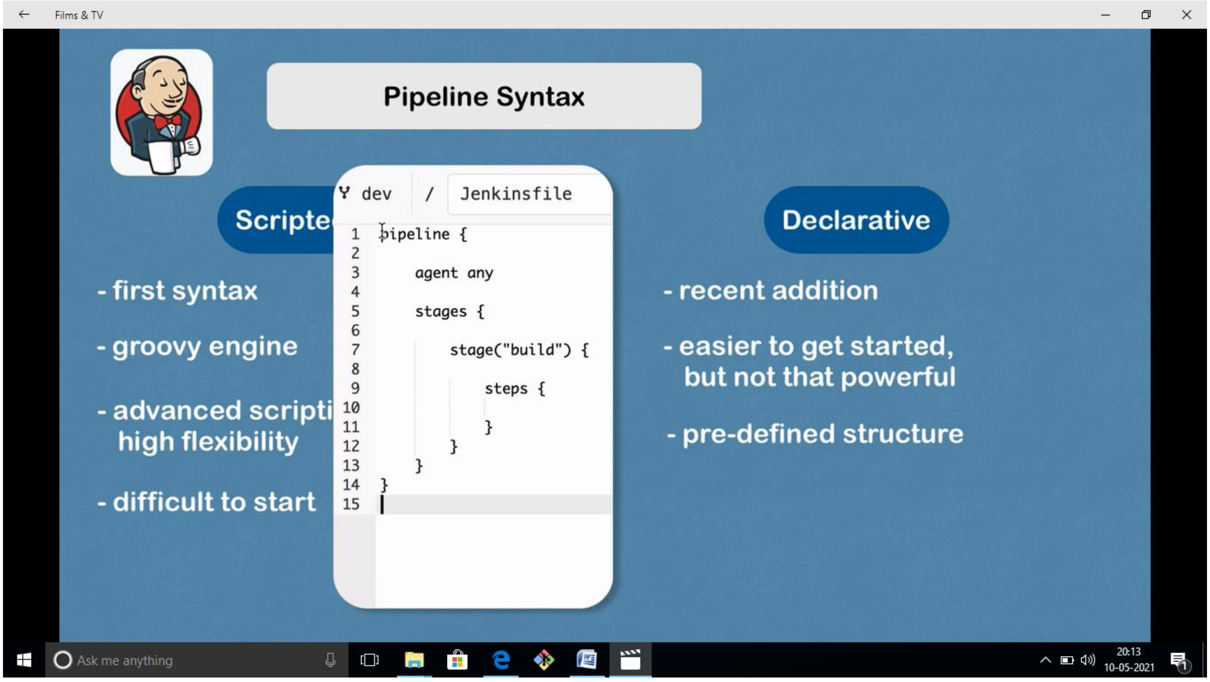
Scripted

Declarative

- first syntax

- groovy engine

```
node {
    // groovy script
}
```



Pipeline Syntax

Scripte

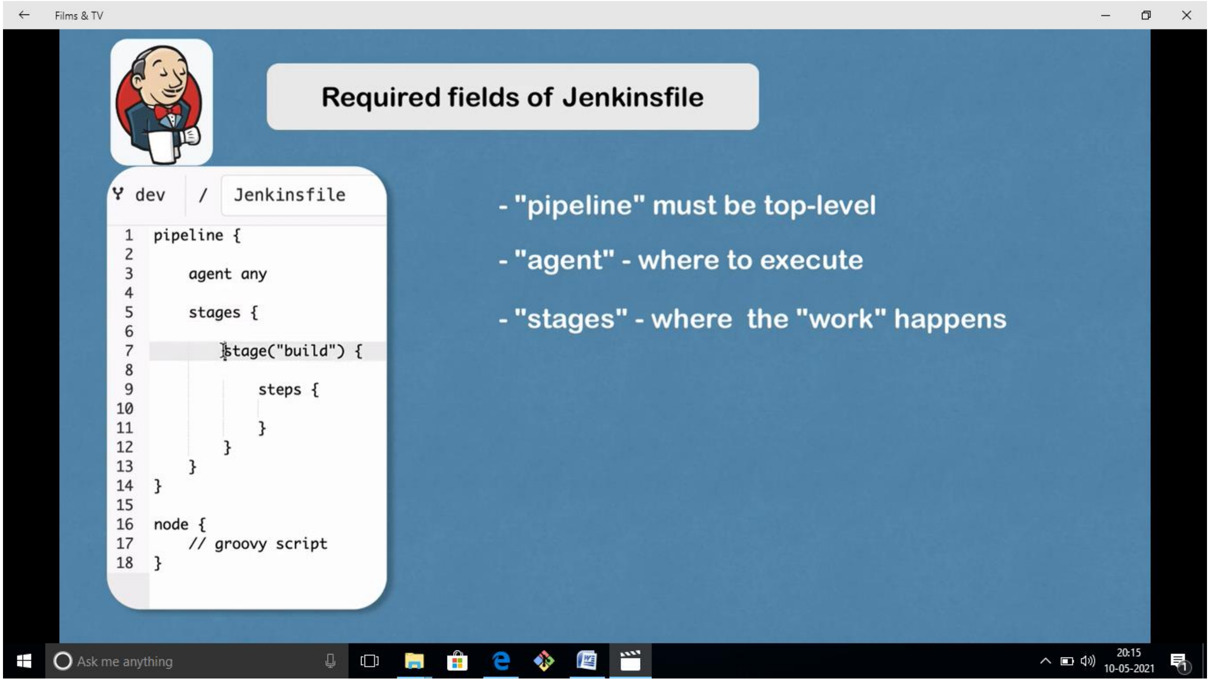
- first syntax
- groovy engine
- advanced scripti high flexibility
- difficult to start

dev / Jenkinsfile

```
pipeline {

    agent any

    stages {

        stage("build") {

            steps {

            }
        }
    }
}

```

Declarative

- recent addition
- easier to get started, but not that powerful
- pre-defined structure



Required fields of Jenkinsfile

dev / Jenkinsfile

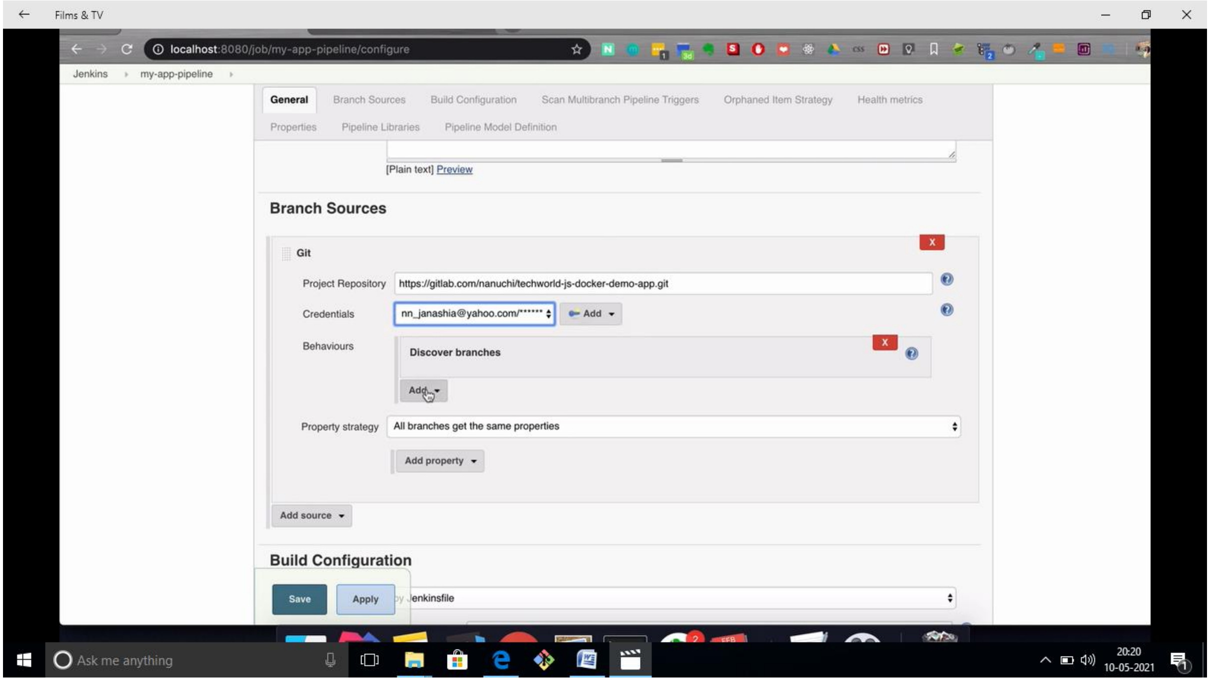
```
pipeline {

    agent any

    stages {

        stage("build") {

            steps {

            }
        }
    }
}

node {
    // groovy script
}
```

- "pipeline" must be top-level
- "agent" - where to execute
- "stages" - where the "work" happens

THIS IS JENKINS FILE

THIS IS IN GITHUB

GOTO SECOND STEP TO START JENKINS



Branch Sources

Git

Project Repository: https://gitlab.com/nanuchi/techworld-js-docker-demo-app.git

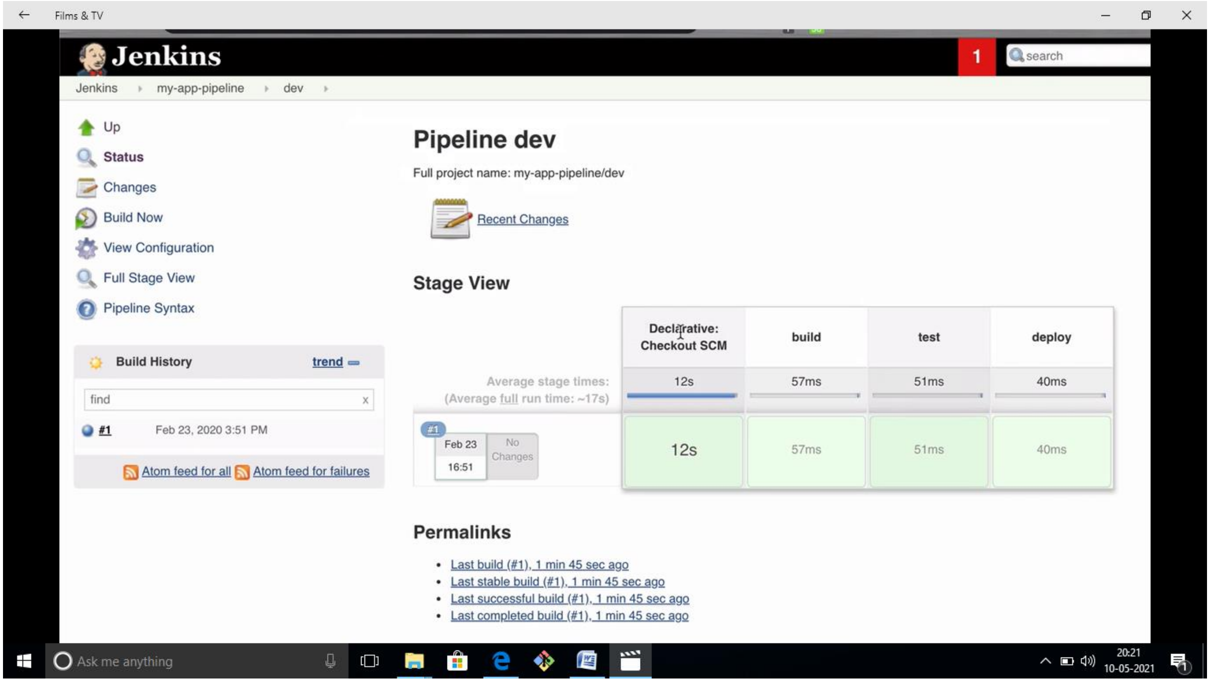
Credentials: nn_janashia@yahoo.com/******

Behaviours: Discover branches

Property strategy: All branches get the same properties

Build Configuration

by Jenkinsfile



#1 Feb 23, 2020 3:51 PM

Pipeline dev

Full project name: my-app-pipeline/dev

Recent Changes

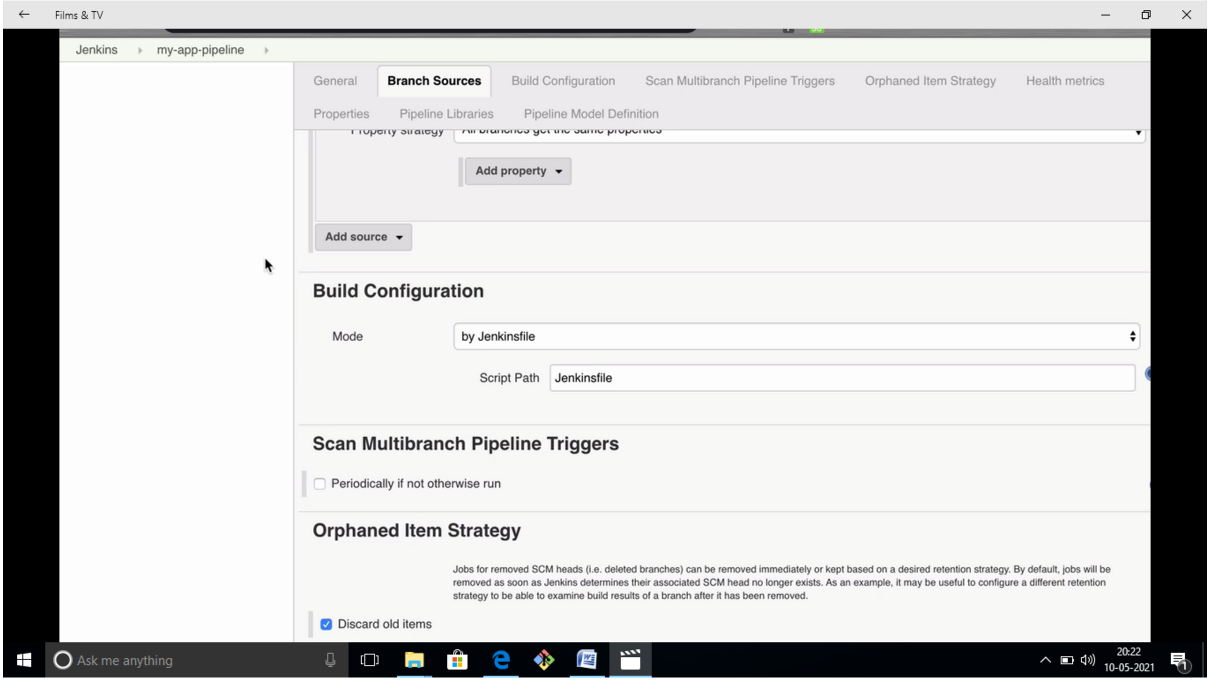
Stage View

	Declarative: Checkout SCM	build	test	deploy
Average stage times: (Average full run time: ~17s)	12s	57ms	51ms	40ms
#1 Feb 23 16:51 No Changes	12s	57ms	51ms	40ms

Permalinks

- Last build (#1), 1 min 45 sec ago
- Last stable build (#1), 1 min 45 sec ago
- Last successful build (#1), 1 min 45 sec ago
- Last completed build (#1), 1 min 45 sec ago

BUILD JINKINS



Build Configuration

Mode: by Jenkinsfile

Script Path: Jenkinsfile

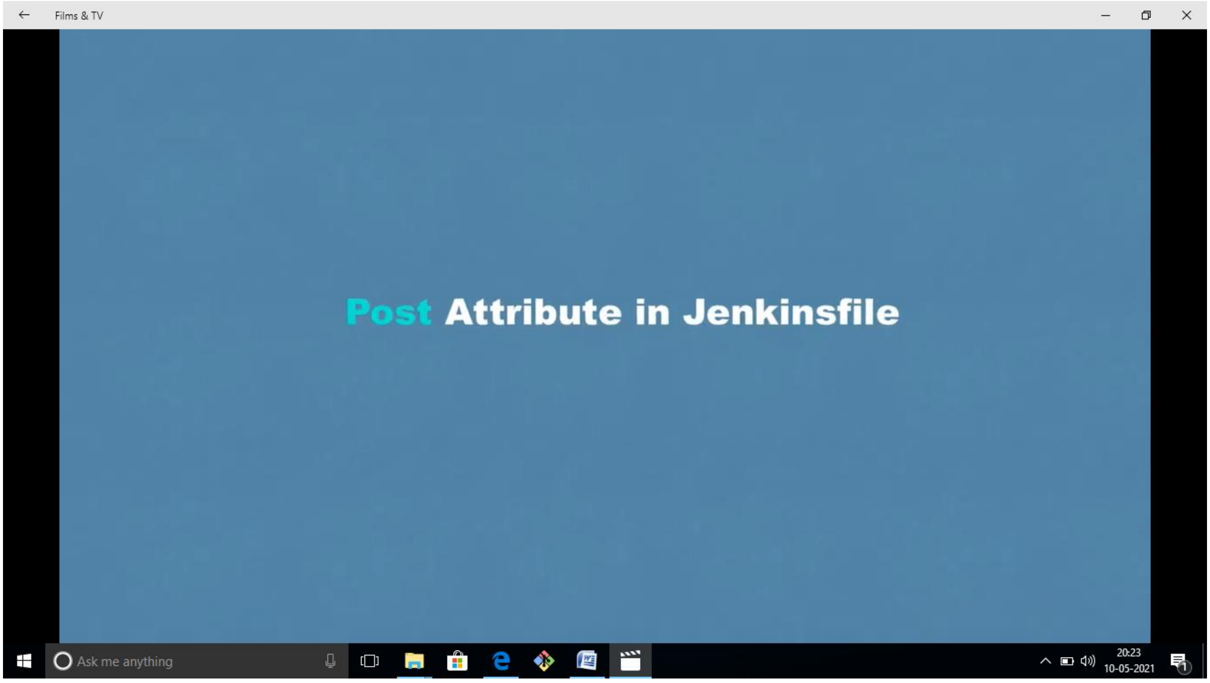
Scan Multibranch Pipeline Triggers

□ Periodically if not otherwise run

Orphaned Item Strategy

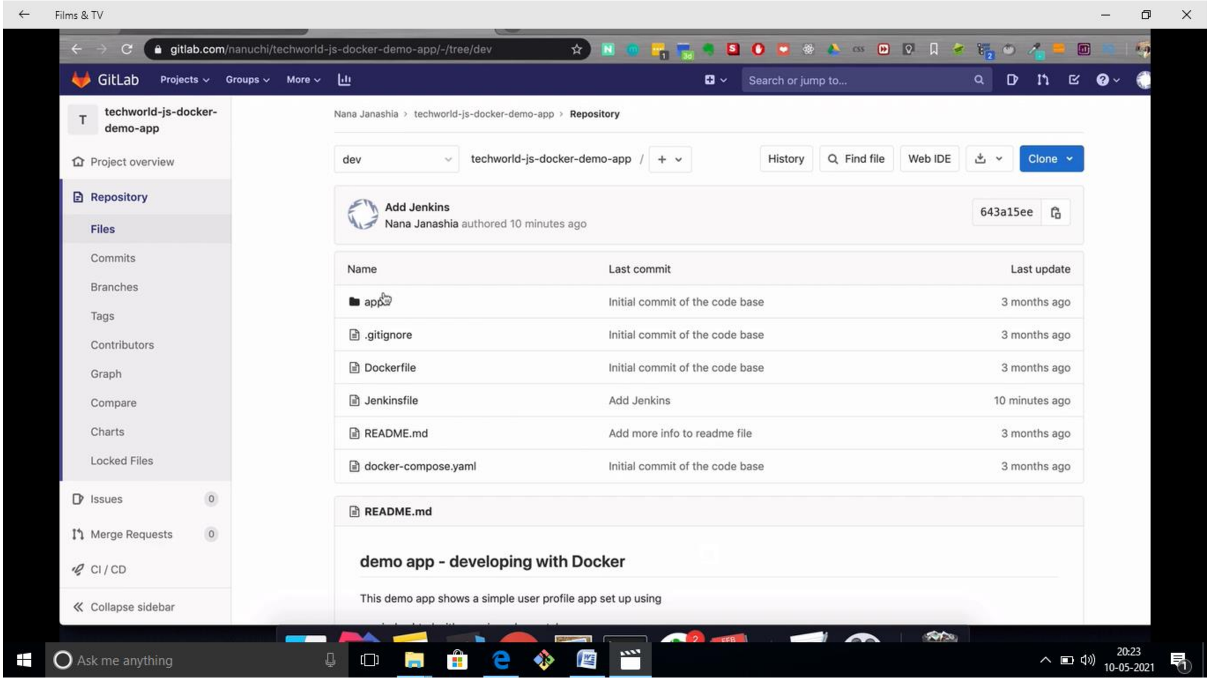
Jobs for removed SCM heads (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs will be removed as soon as Jenkins determines their associated SCM head no longer exists. As an example, it may be useful to configure a different retention strategy to be able to examine build results of a branch after it has been removed.

☑ Discard old items



Post Attribute in Jenkinsfile

GOTO GITHUB AND OPEN JENKINS FILE



gitlab.com/nanuchi/techworld-js-docker-demo-app/-/tree/dev

Nana Janashia › techworld-js-docker-demo-app › Repository

dev / techworld-js-docker-demo-app /

Add Jenkins
Nana Janashia authored 10 minutes ago — 643a15ee

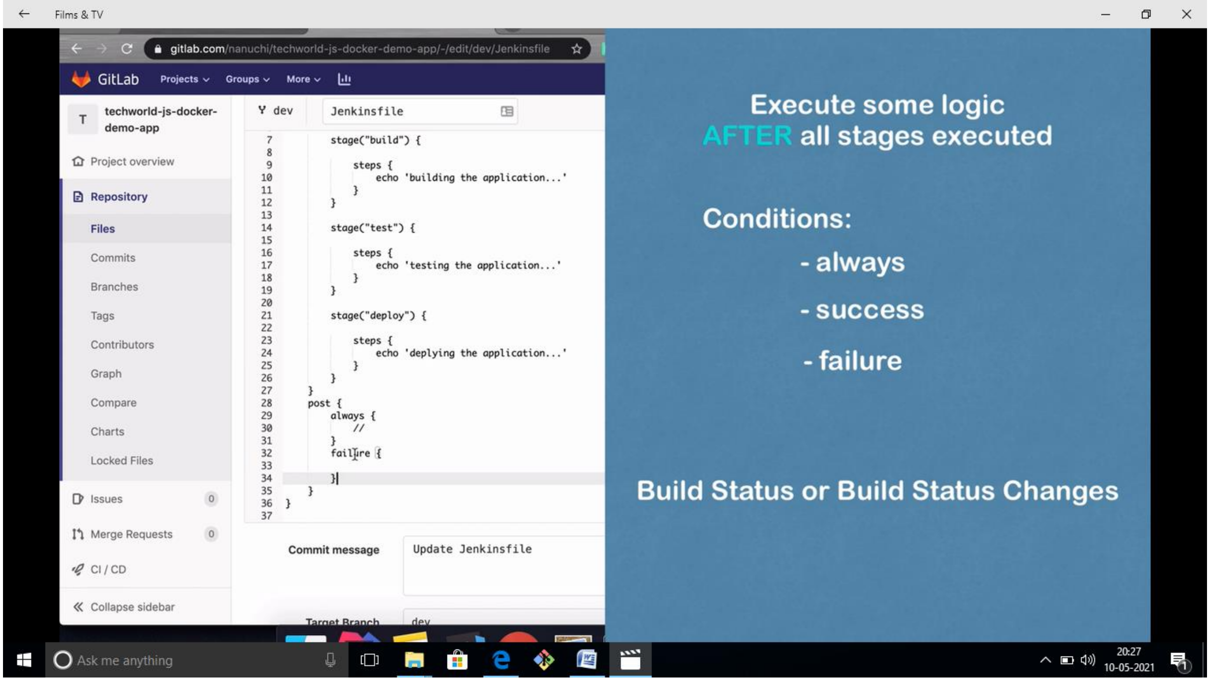
Name	Last commit	Last update
app	Initial commit of the code base	3 months ago
.gitignore	Initial commit of the code base	3 months ago
Dockerfile	Initial commit of the code base	3 months ago
Jenkinsfile	Add Jenkins	10 minutes ago
README.md	Add more info to readme file	3 months ago
docker-compose.yaml	Initial commit of the code base	3 months ago

README.md

demo app - developing with Docker

This demo app shows a simple user profile app set up using

POST BUILD ACTIONS



gitlab.com/nanuchi/techworld-js-docker-demo-app/-/edit/dev/Jenkinsfile

dev — Jenkinsfile

```
        stage("build") {

            steps {
                echo 'building the application...'
            }
        }

        stage("test") {

            steps {
                echo 'testing the application...'
            }
        }

        stage("deploy") {

            steps {
                echo 'deplying the application...'
            }
        }
    }
    post {
        always {
            //
        }
        failure {

        }
    }
}
```

Commit message: Update Jenkinsfile

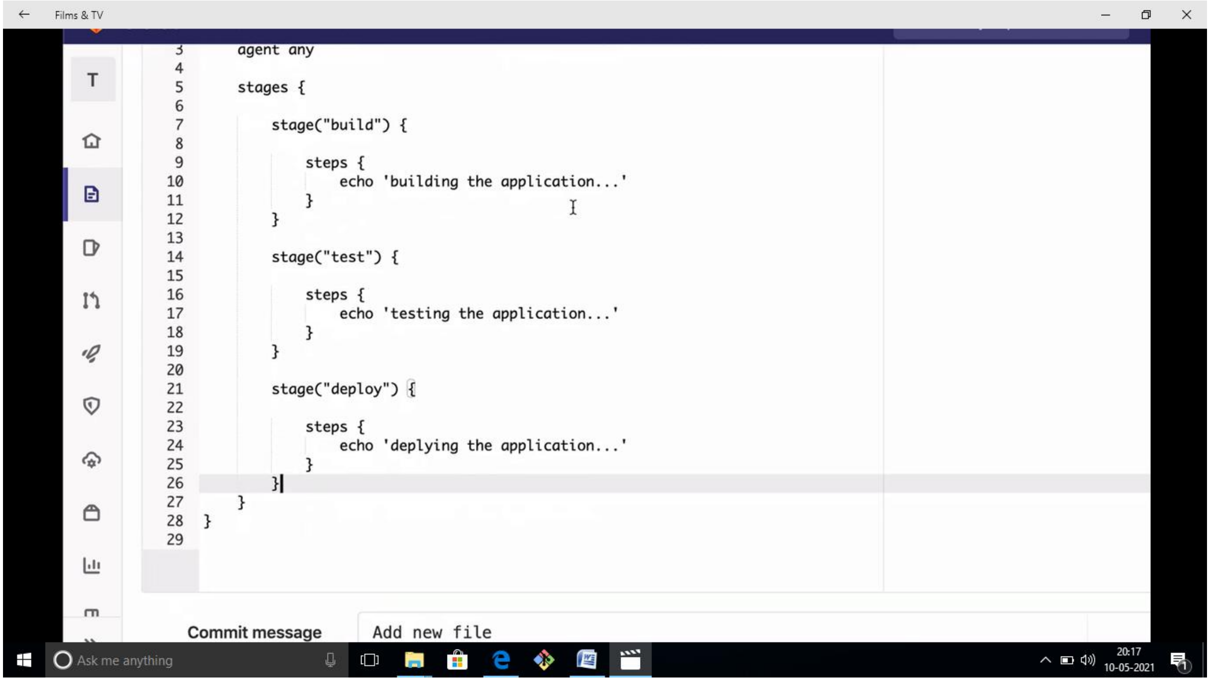
Target Branch: dev

Execute some logic
AFTER all stages executed

Conditions:

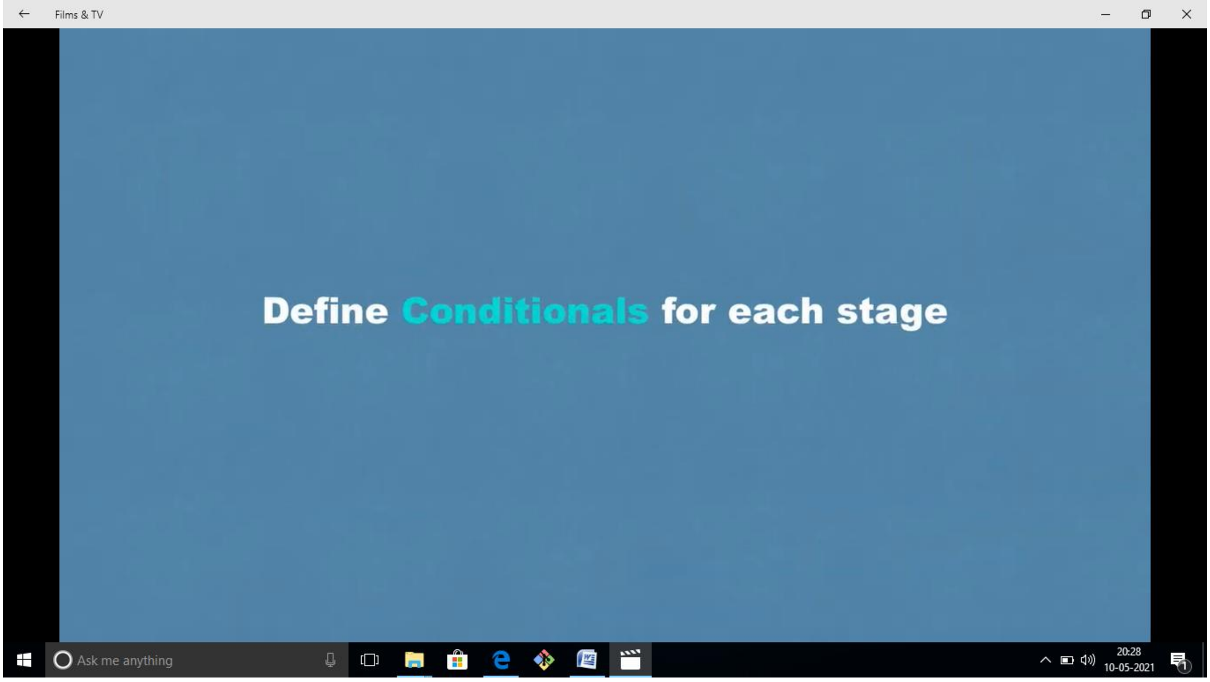
- always
- success
- failure

Build Status or Build Status Changes



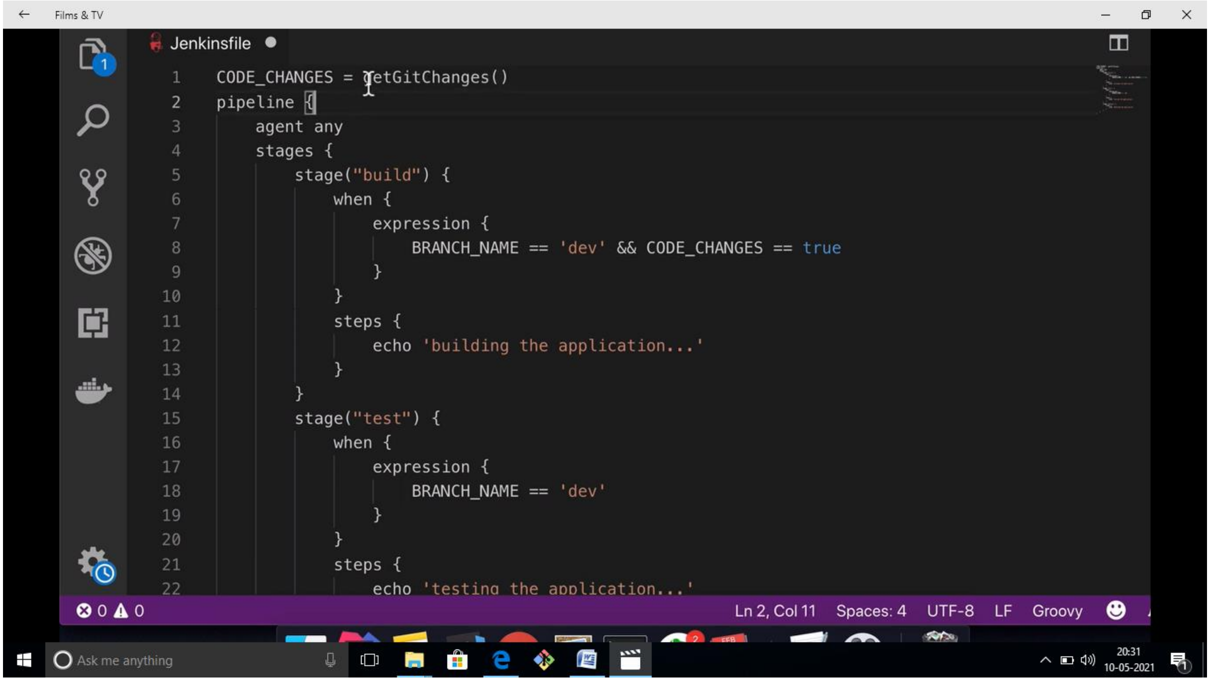
```
    agent any

    stages {

        stage("build") {

            steps {
                echo 'building the application...'
            }
        }

        stage("test") {

            steps {
                echo 'testing the application...'
            }
        }

        stage("deploy") {

            steps {
                echo 'deplying the application...'
            }
        }
    }
}
```

Commit message: Add new file

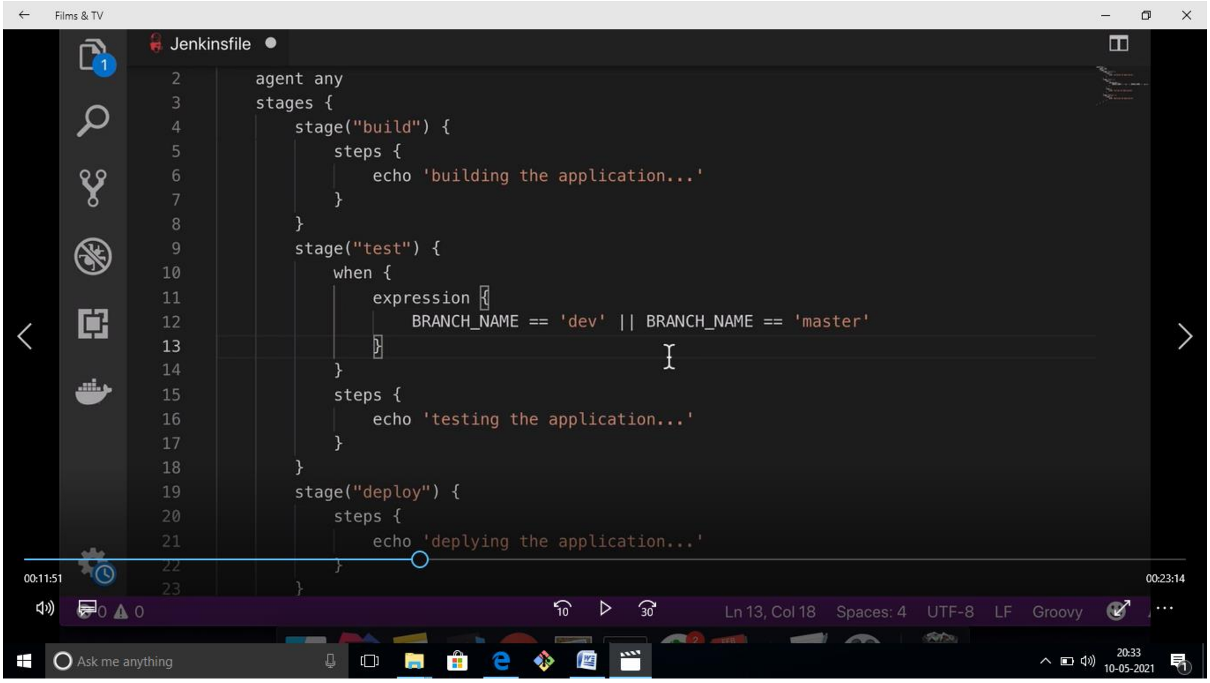


Define Conditionals for each stage

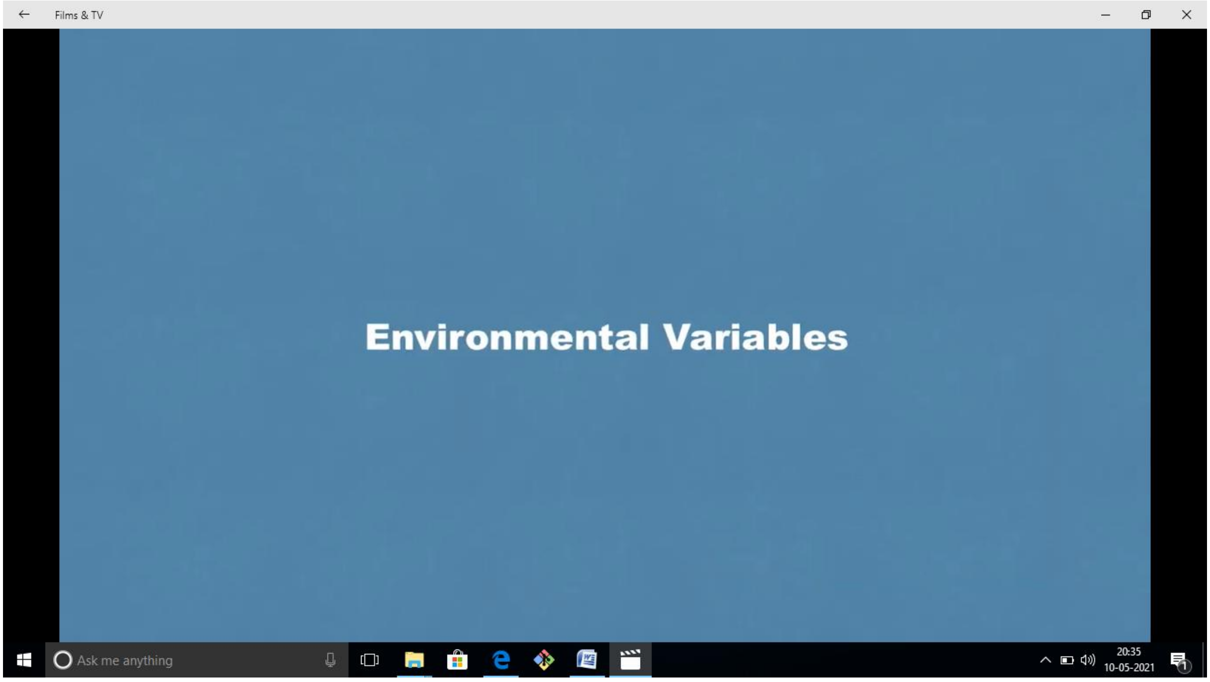
IN STAGES WE US CONDITIONAL EXPRESSINS



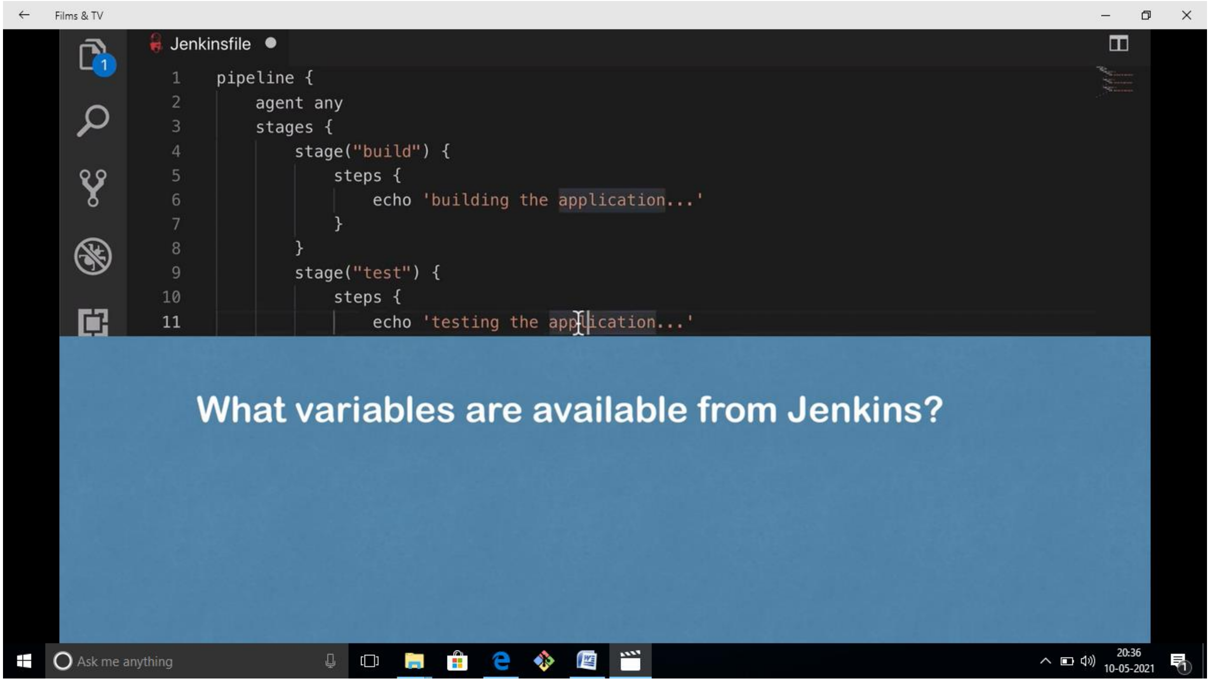
```
CODE_CHANGES = getGitChanges()
pipeline {
    agent any
    stages {
        stage("build") {
            when {
                expression {
                    BRANCH_NAME == 'dev' && CODE_CHANGES == true
                }
            }
            steps {
                echo 'building the application...'
            }
        }
        stage("test") {
            when {
                expression {
                    BRANCH_NAME == 'dev'
                }
            }
            steps {
                echo 'testing the application...'
```



```
    agent any
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
            }
        }
        stage("test") {
            when {
                expression {
                    BRANCH_NAME == 'dev' || BRANCH_NAME == 'master'
                }
            }
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
            }
        }
```



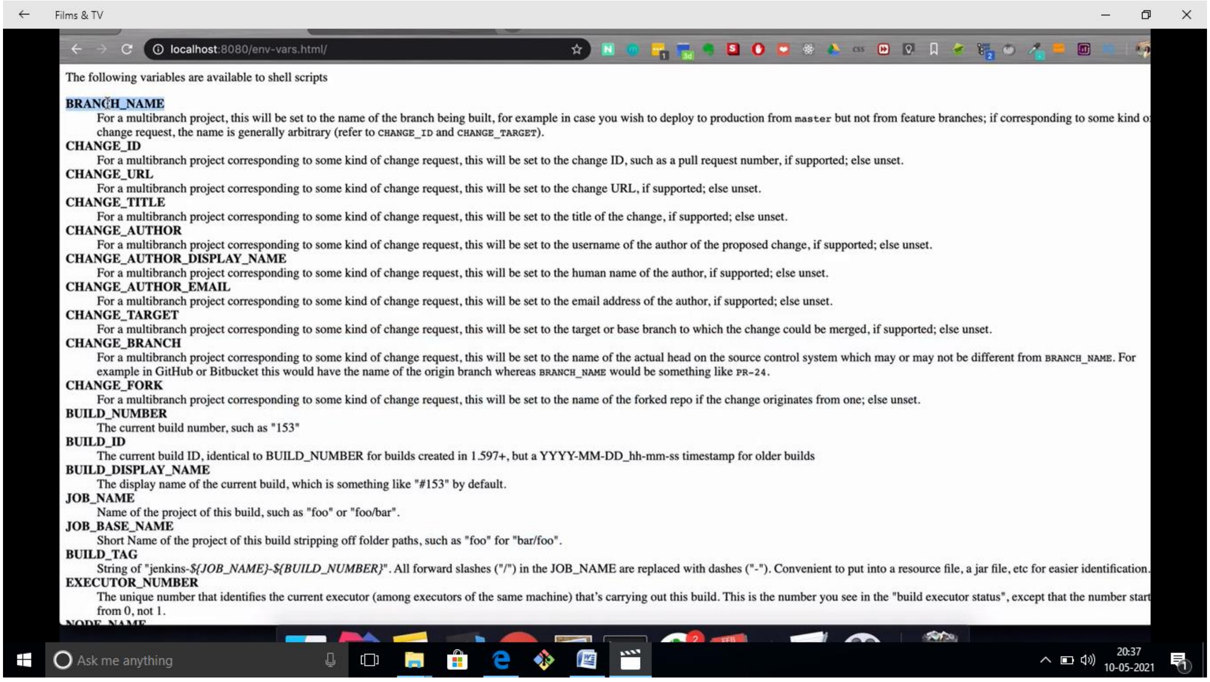
Environmental Variables



```
pipeline {
    agent any
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
```

What variables are available from Jenkins?

THESE ARE ENVIRONMENT VARIABLES



The following variables are available to shell scripts

BRANCH_NAME
For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from `master` but not from feature branches; if corresponding to some kind o change request, the name is generally arbitrary (refer to `CHANGE_ID` and `CHANGE_TARGET`).

CHANGE_ID
For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.

CHANGE_URL
For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.

CHANGE_TITLE
For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.

CHANGE_AUTHOR
For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.

CHANGE_AUTHOR_DISPLAY_NAME
For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.

CHANGE_AUTHOR_EMAIL
For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.

CHANGE_TARGET
For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.

CHANGE_BRANCH
For a multibranch project corresponding to some kind of change request, this will be set to the name of the actual head on the source control system which may or may not be different from `BRANCH_NAME`. For example in GitHub or Bitbucket this would have the name of the origin branch whereas `BRANCH_NAME` would be something like `PR-24`.

CHANGE_FORK
For a multibranch project corresponding to some kind of change request, this will be set to the name of the forked repo if the change originates from one; else unset.

BUILD_NUMBER
The current build number, such as "153"

BUILD_ID
The current build ID, identical to BUILD_NUMBER for builds created in 1.597+, but a YYYY-MM-DD_hh-mm-ss timestamp for older builds

BUILD_DISPLAY_NAME
The display name of the current build, which is something like "#153" by default.

JOB_NAME
Name of the project of this build, such as "foo" or "foo/bar".

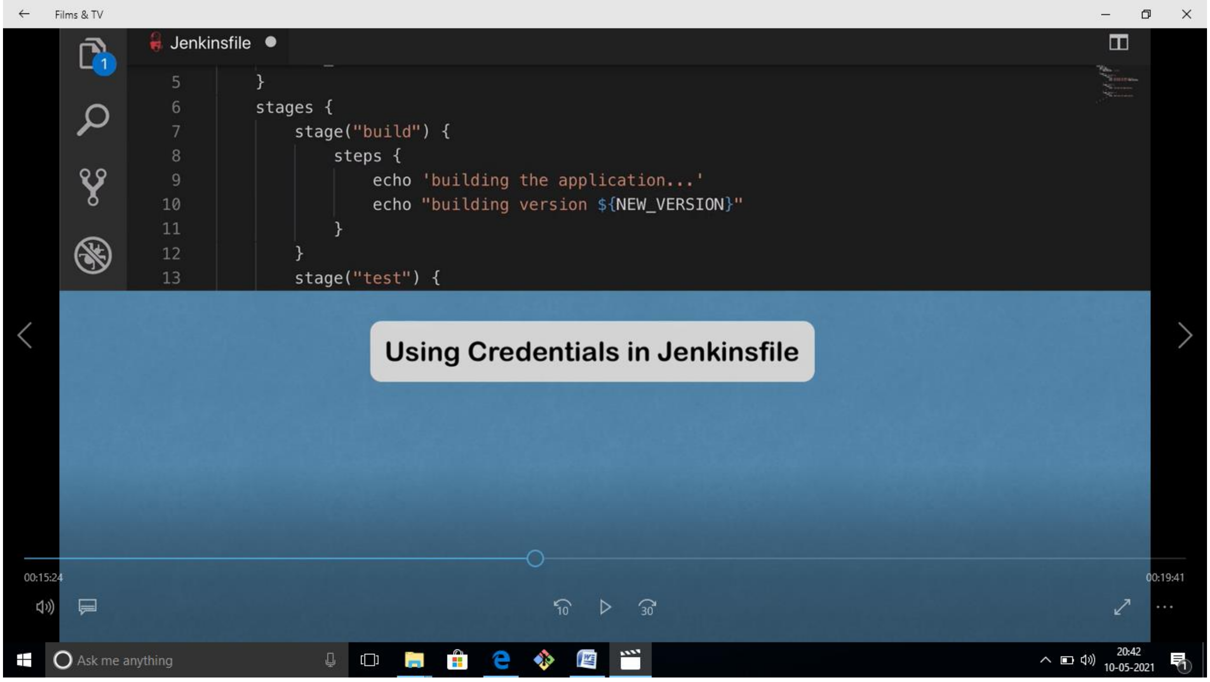
JOB_BASE_NAME
Short Name of the project of this build stripping off folder paths, such as "foo" for "bar/foo".

BUILD_TAG
String of "jenkins-*${JOB_NAME}-${BUILD_NUMBER}*". All forward slashes ("/") in the JOB_NAME are replaced with dashes ("-"). Convenient to put into a resource file, a jar file, etc for easier identification.

EXECUTOR_NUMBER
The unique number that identifies the current executor (among executors of the same machine) that's carrying out this build. This is the number you see in the "build executor status", except that the number start from 0, not 1.

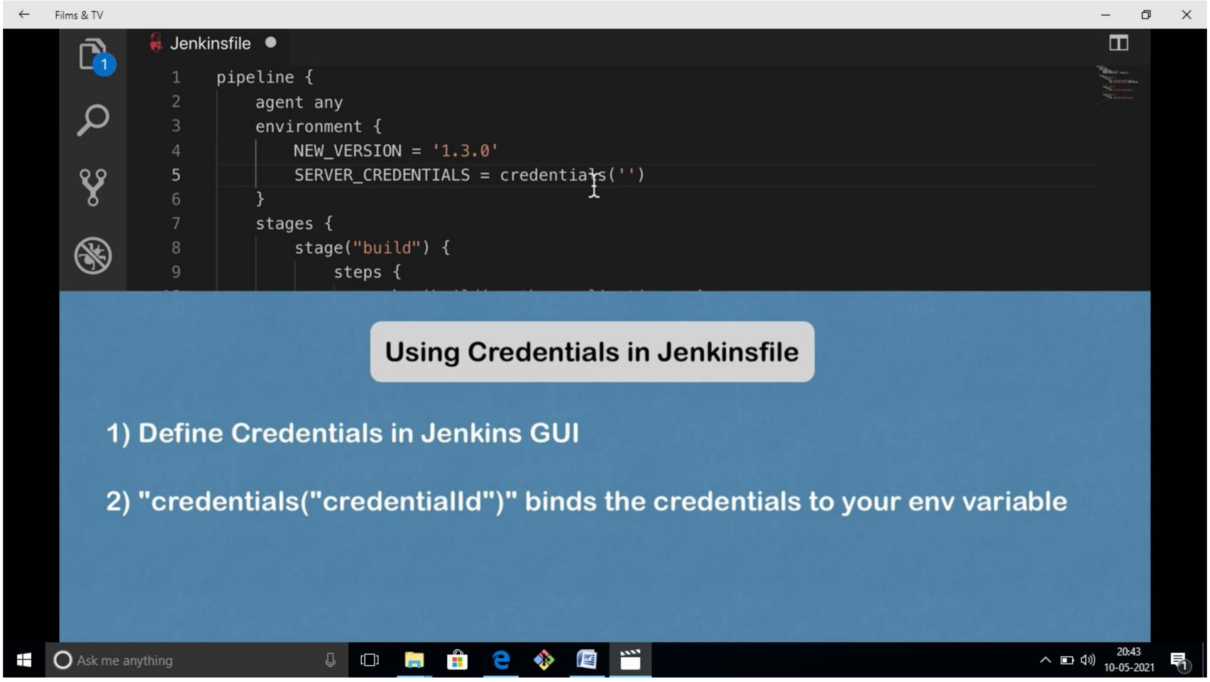
WHATEVER VARIABLE YOU DEFINE THATS APPLY FOR ALL STAGES

ALSO USE CREDINTIALS IN ENVIRONMENT VARIABLES



```
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
```

Using Credentials in Jenkinsfile

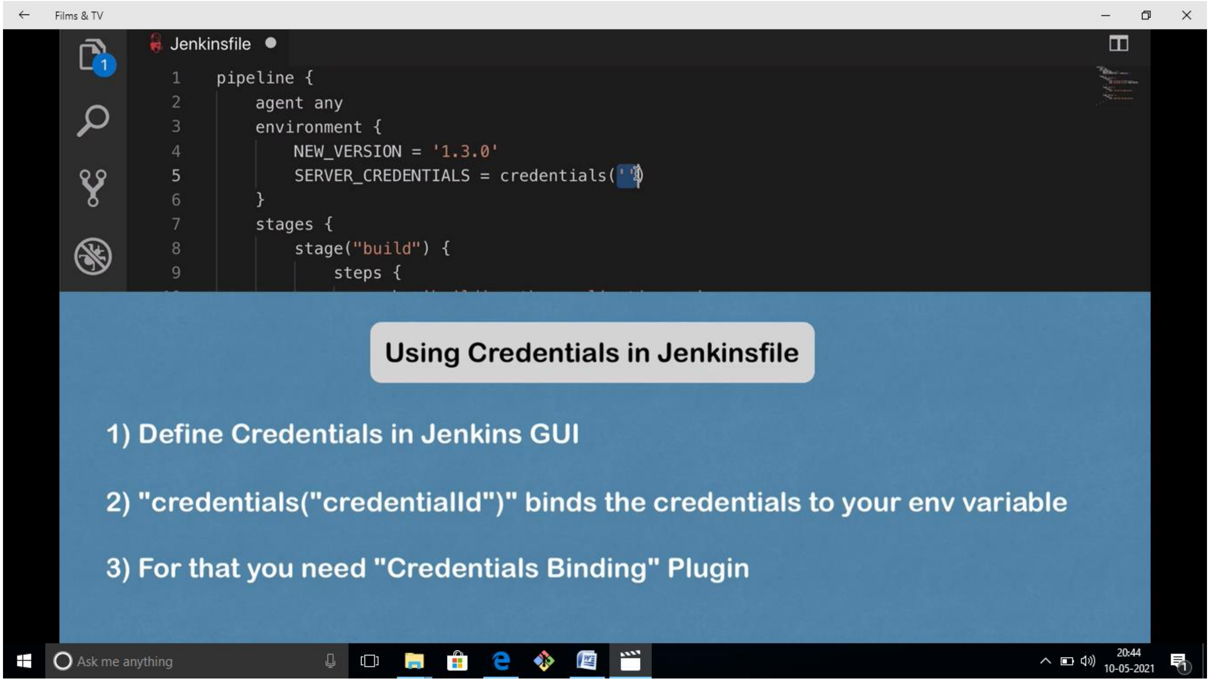


```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.3.0'
        SERVER_CREDENTIALS = credentials('')
    }
    stages {
        stage("build") {
            steps {
```

Using Credentials in Jenkinsfile

1) Define Credentials in Jenkins GUI

2) "credentials("credentialId")" binds the credentials to your env variable



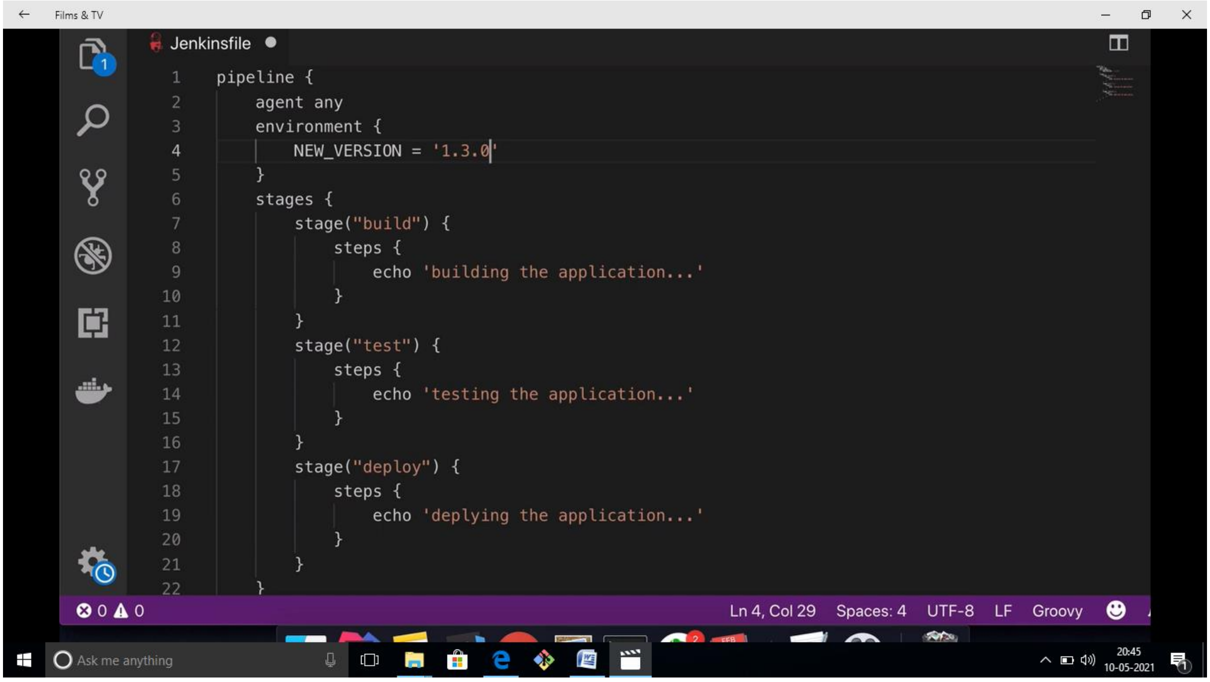
```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.3.0'
        SERVER_CREDENTIALS = credentials('')
    }
    stages {
        stage("build") {
            steps {
```

Using Credentials in Jenkinsfile

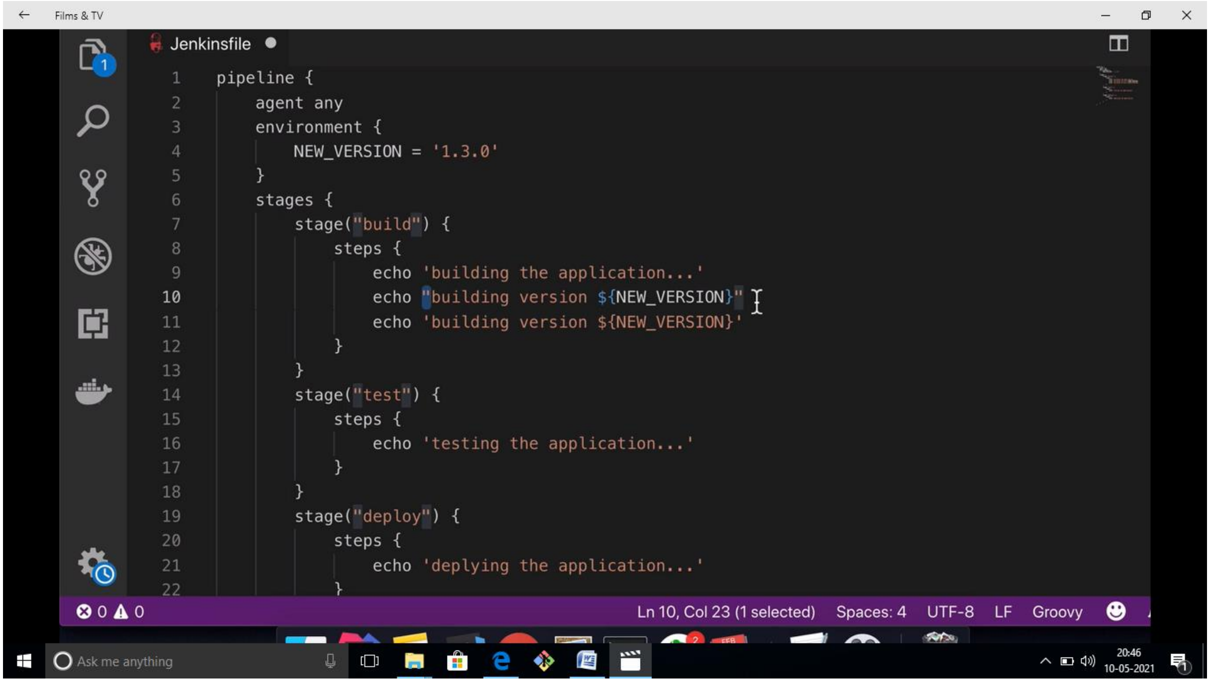
1) Define Credentials in Jenkins GUI

2) "credentials("credentialId")" binds the credentials to your env variable

3) For that you need "Credentials Binding" Plugin



```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.3.0'
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
            }
        }
    }
```

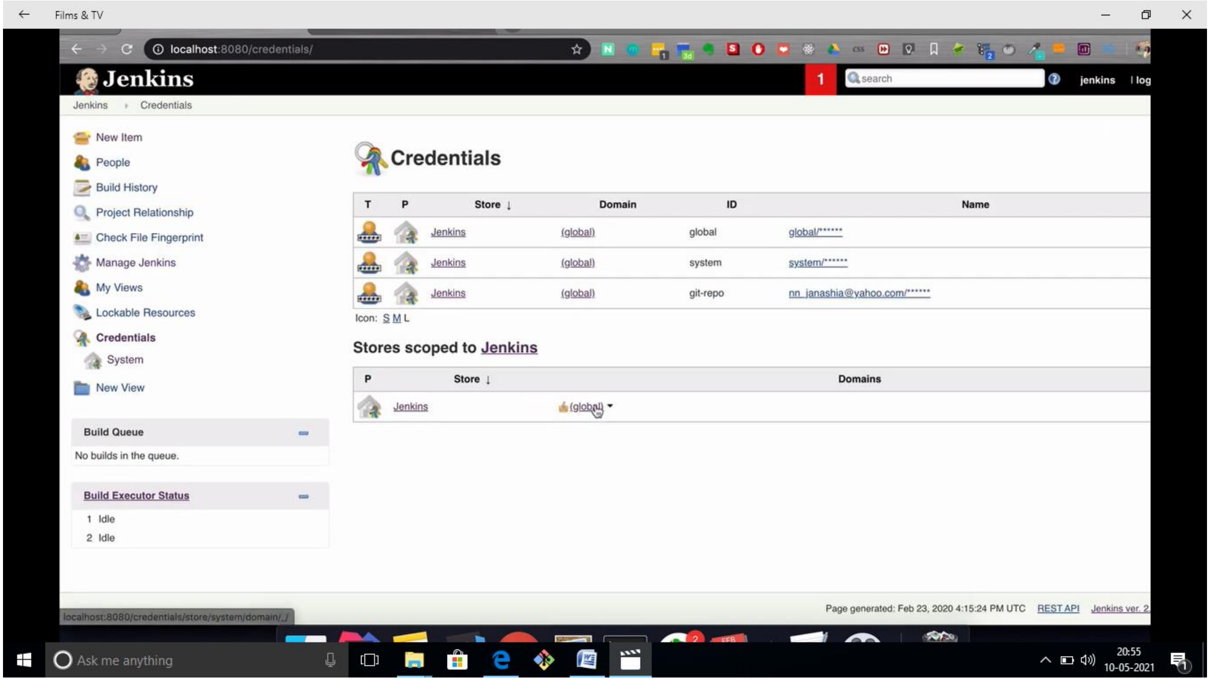


```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.3.0'
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
                echo "building version ${NEW_VERSION}"
                echo 'building version ${NEW_VERSION}'
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
            }
```

TO SET CREDINTIALS GOTO JENKINS

AND CREDINTIALS

AND SELCT GLOBAL OPTION.



Credentials

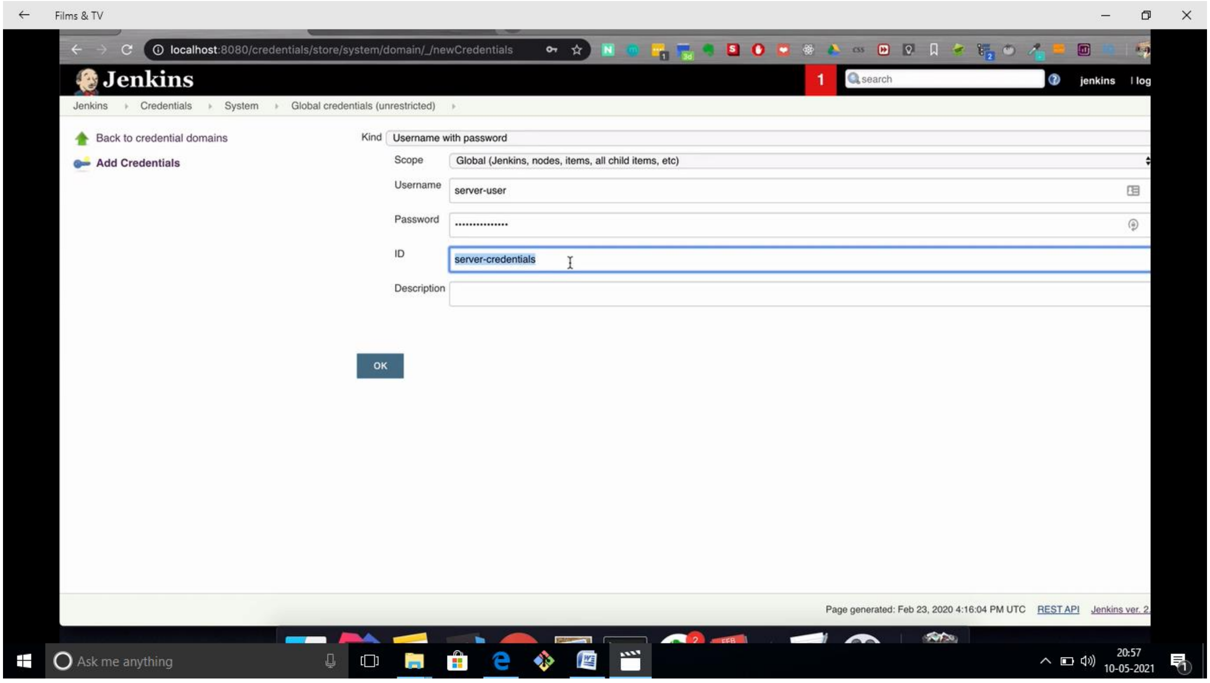
T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	global	global/******
		Jenkins	(global)	system	system/******
		Jenkins	(global)	git-repo	nn_janashia@yahoo.com/******

Icon: S M L

Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	(global)

Page generated: Feb 23, 2020 4:15:24 PM UTC REST API Jenkins ver. 2



Films & TV
localhost:8080/credentials/store/system/domain/_/newCredentials
Jenkins
search
jenkins | log
Jenkins › Credentials › System › Global credentials (unrestricted) ›
Back to credential domains
Add Credentials
OK
Ask me anything

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

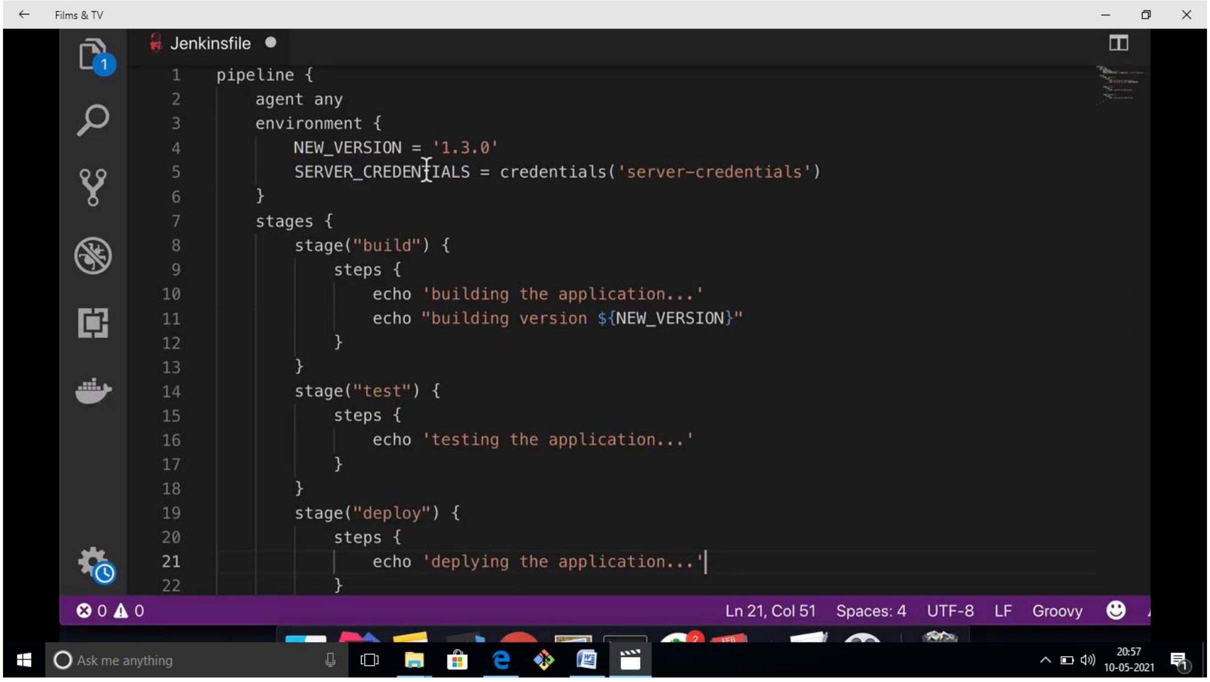
Username: server-user

Password: ···············

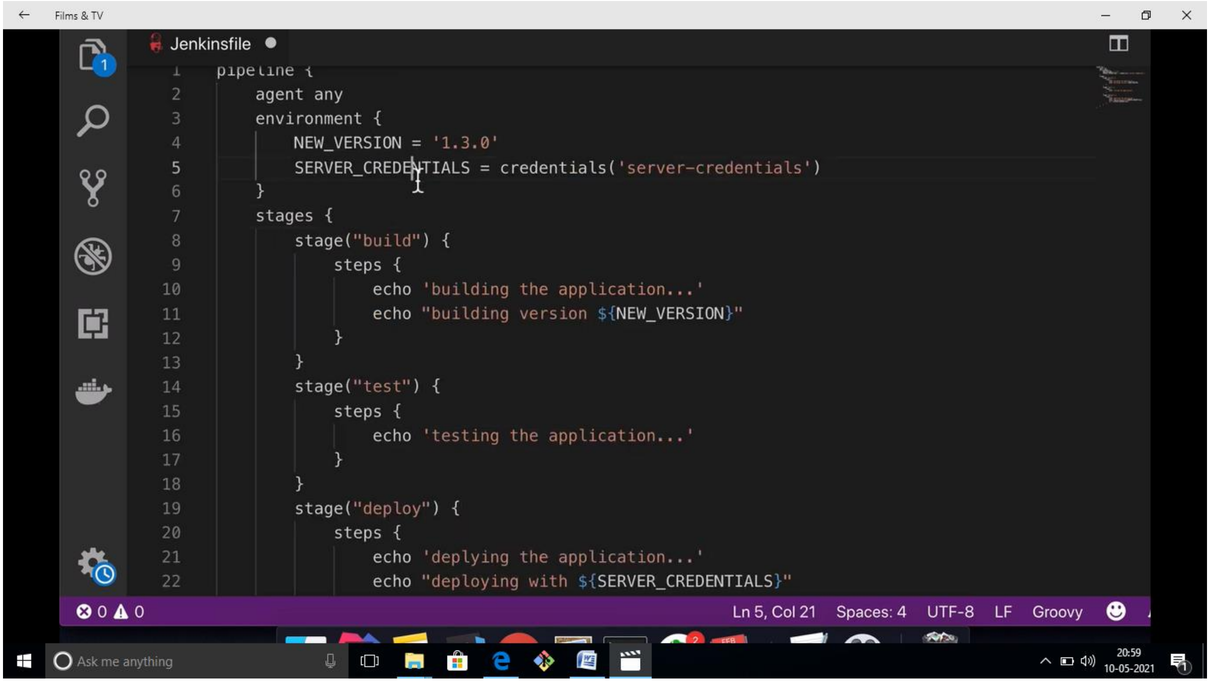
ID: server-credentials

Description:

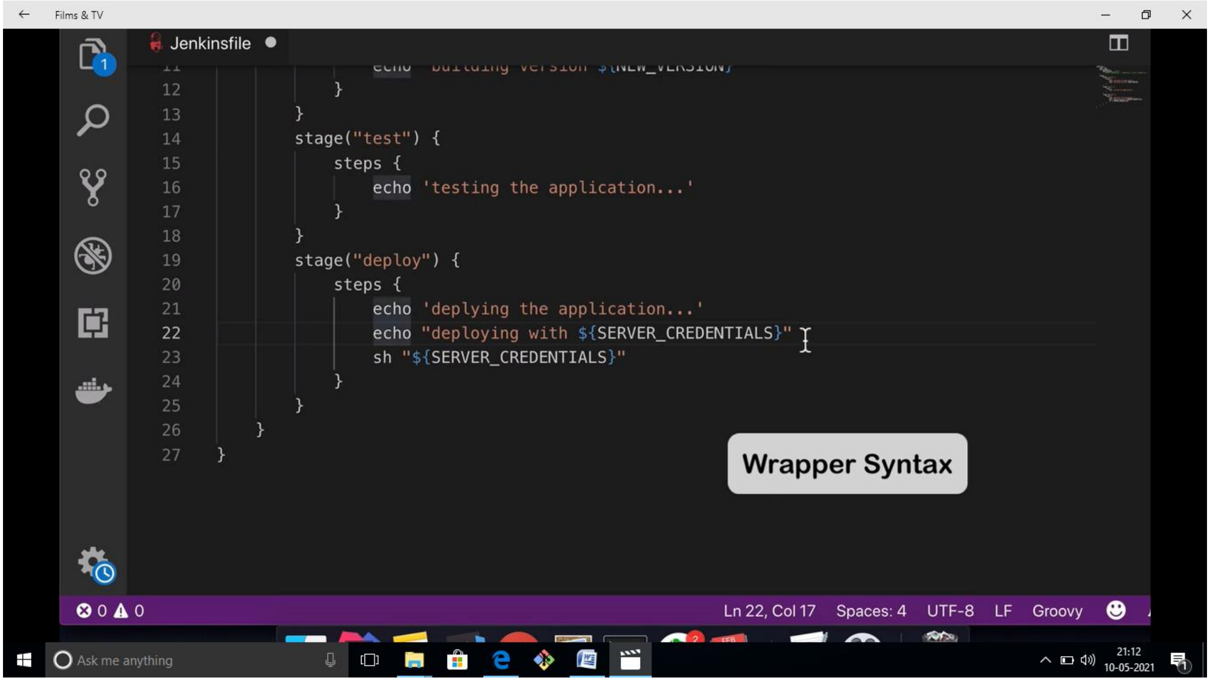
Page generated: Feb 23, 2020 4:16:04 PM UTC REST API Jenkins ver. 2



```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.3.0'
        SERVER_CREDENTIALS = credentials('server-credentials')
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
            }
```

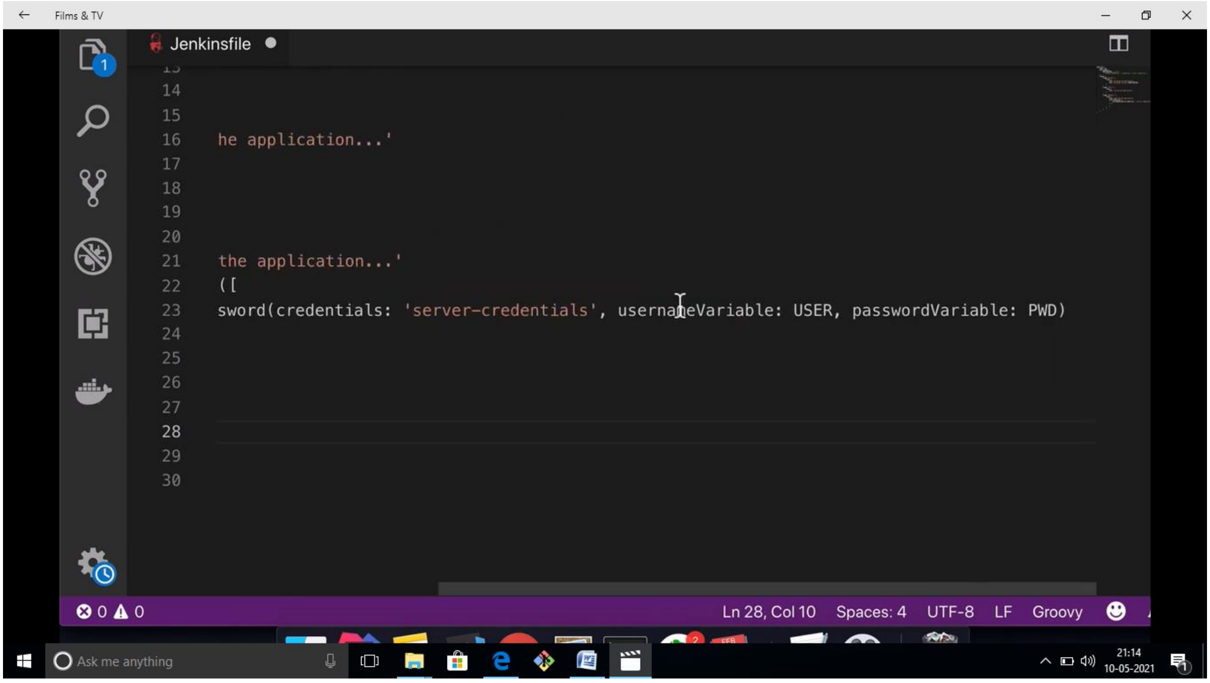


```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.3.0'
        SERVER_CREDENTIALS = credentials('server-credentials')
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
                echo "deploying with ${SERVER_CREDENTIALS}"
```

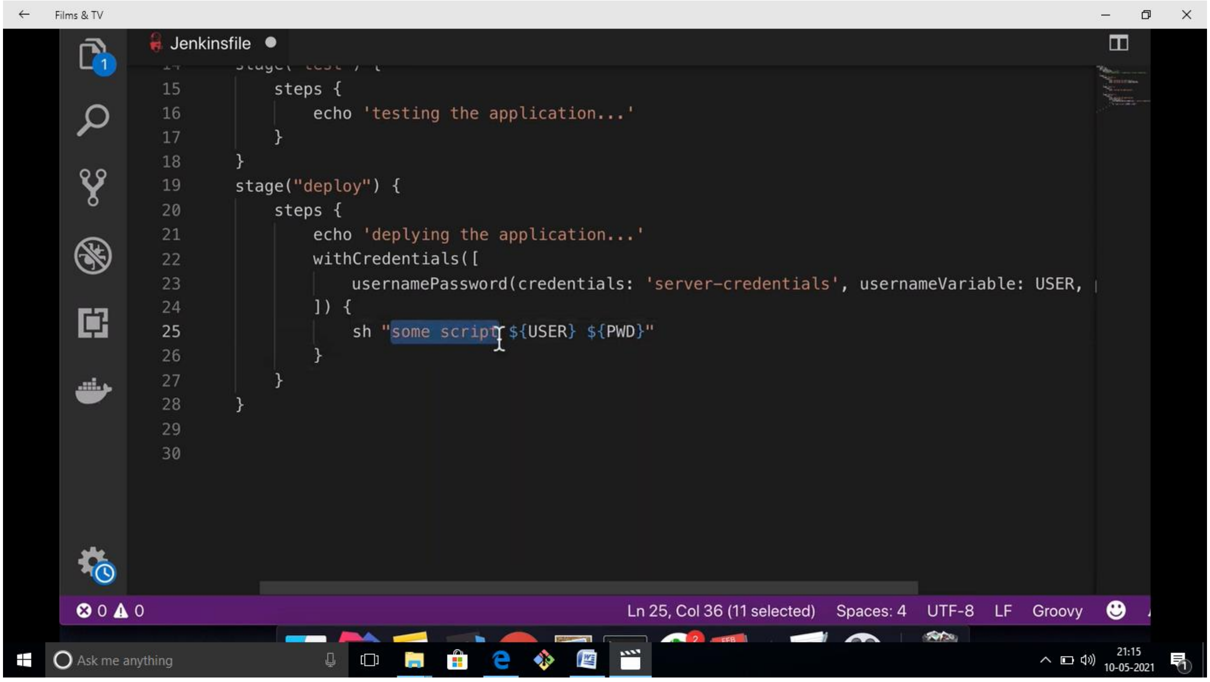


```
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
                echo "deploying with ${SERVER_CREDENTIALS}"
                sh "${SERVER_CREDENTIALS}"
            }
        }
    }
}
```

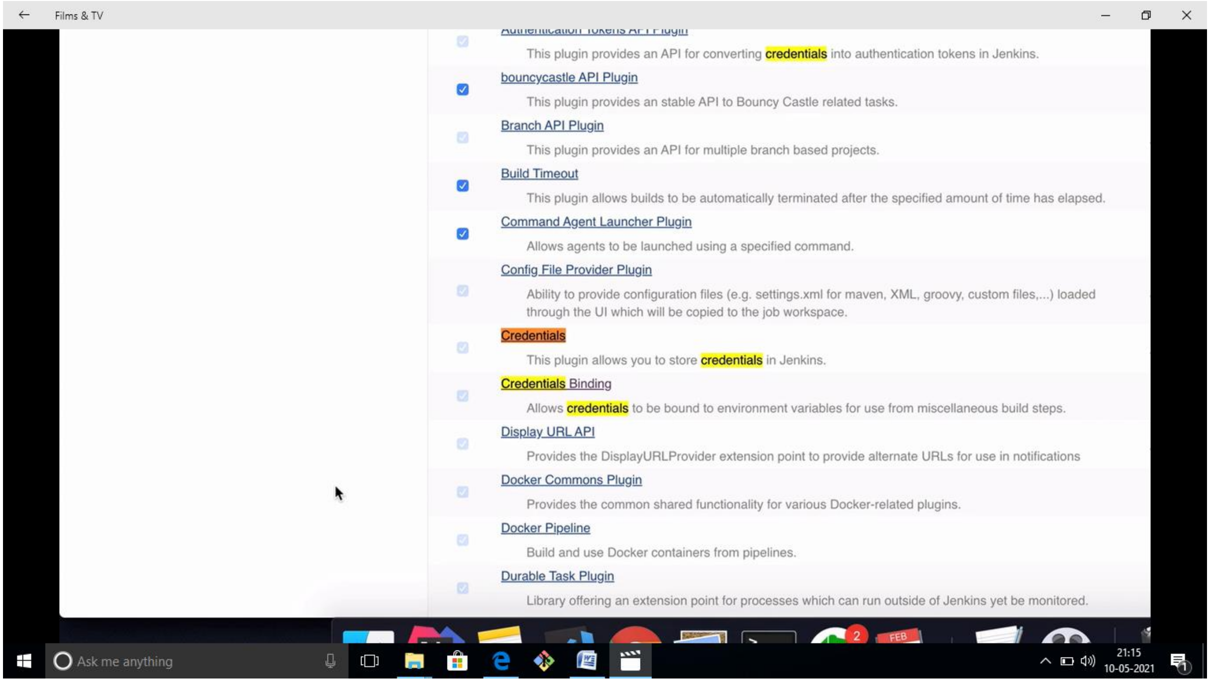
Wrapper Syntax



```
he application...'




the application...'
([
sword(credentials: 'server-credentials', usernameVariable: USER, passwordVariable: PWD)
```



```
        steps {
            echo 'testing the application...'
        }
    }
    stage("deploy") {
        steps {
            echo 'deplying the application...'
            withCredentials([
                usernamePassword(credentials: 'server-credentials', usernameVariable: USER,
            ]) {
                sh "some script ${USER} ${PWD}"
            }
        }
    }
```



☑ Authentication Tokens API Plugin
This plugin provides an API for converting credentials into authentication tokens in Jenkins.

☑ bouncycastle API Plugin
This plugin provides an stable API to Bouncy Castle related tasks.

☑ Branch API Plugin
This plugin provides an API for multiple branch based projects.

☑ Build Timeout
This plugin allows builds to be automatically terminated after the specified amount of time has elapsed.

☑ Command Agent Launcher Plugin
Allows agents to be launched using a specified command.

☑ Config File Provider Plugin
Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace.

☑ Credentials
This plugin allows you to store credentials in Jenkins.

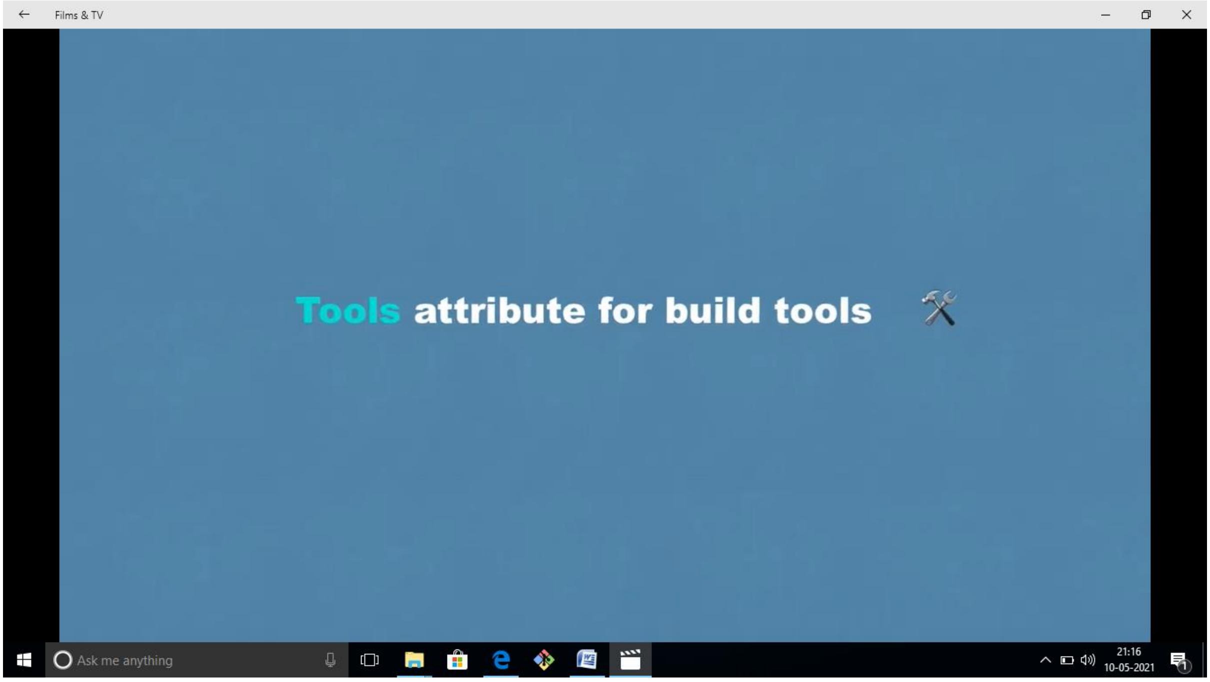
☑ Credentials Binding
Allows credentials to be bound to environment variables for use from miscellaneous build steps.

☑ Display URL API
Provides the DisplayURLProvider extension point to provide alternate URLs for use in notifications

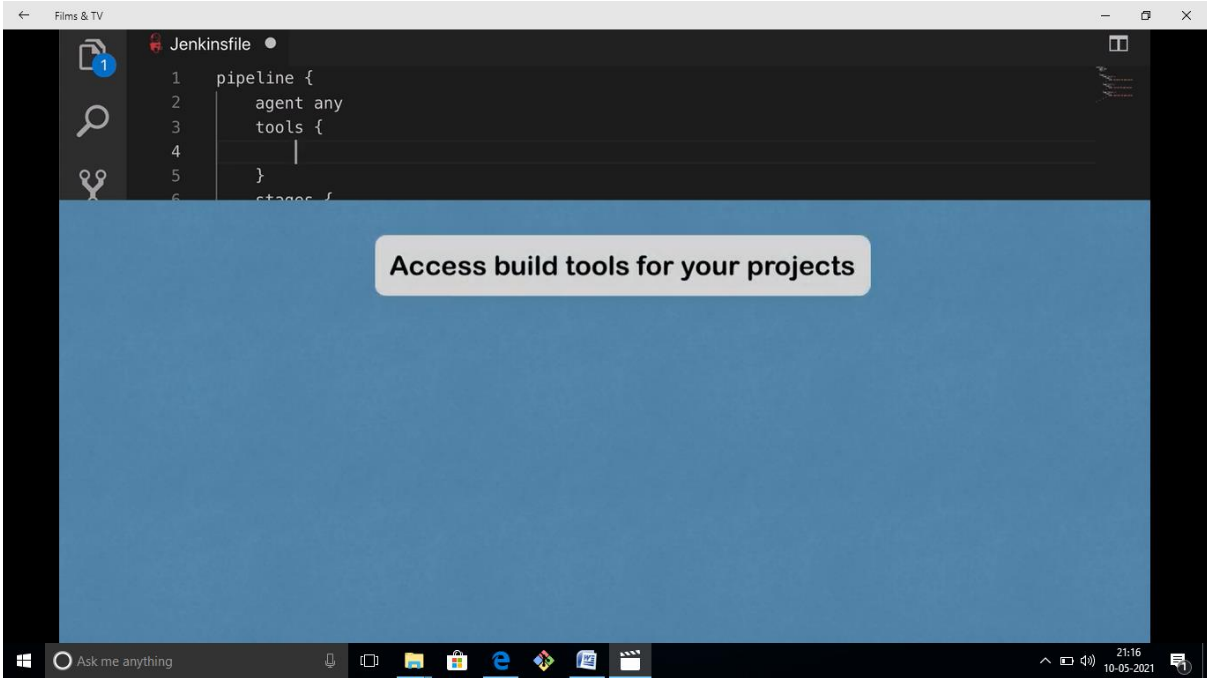
☑ Docker Commons Plugin
Provides the common shared functionality for various Docker-related plugins.

☑ Docker Pipeline
Build and use Docker containers from pipelines.

☑ Durable Task Plugin
Library offering an extension point for processes which can run outside of Jenkins yet be monitored.

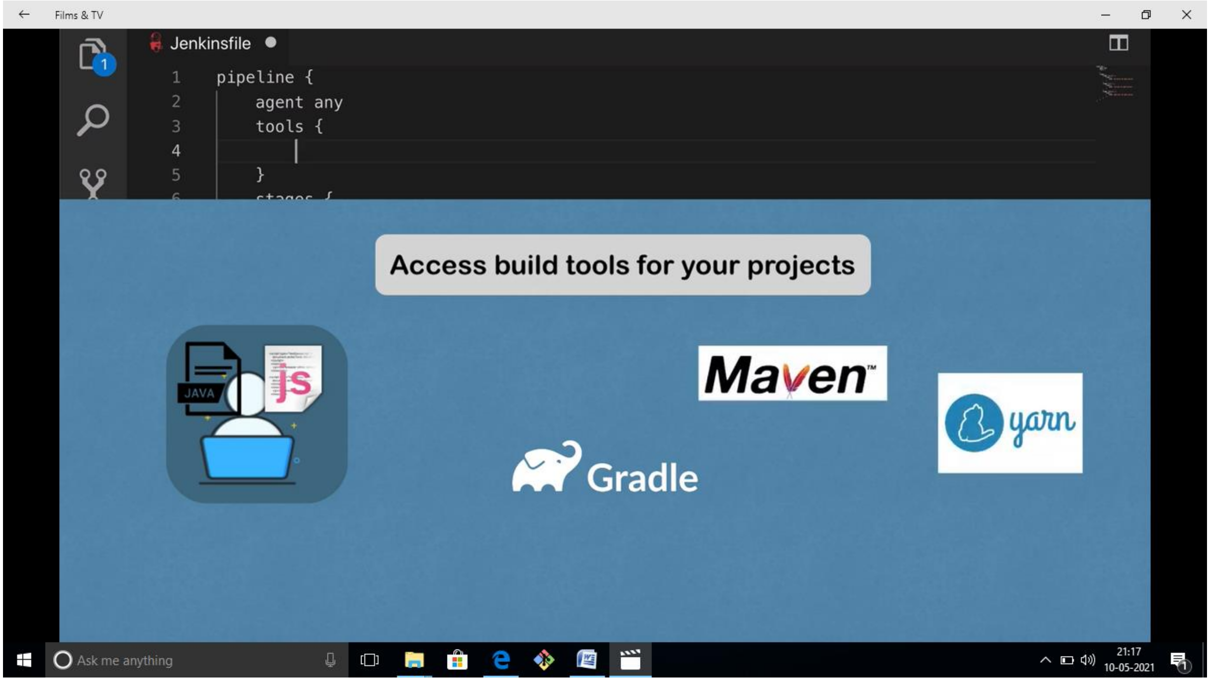


Tools attribute for build tools 🛠



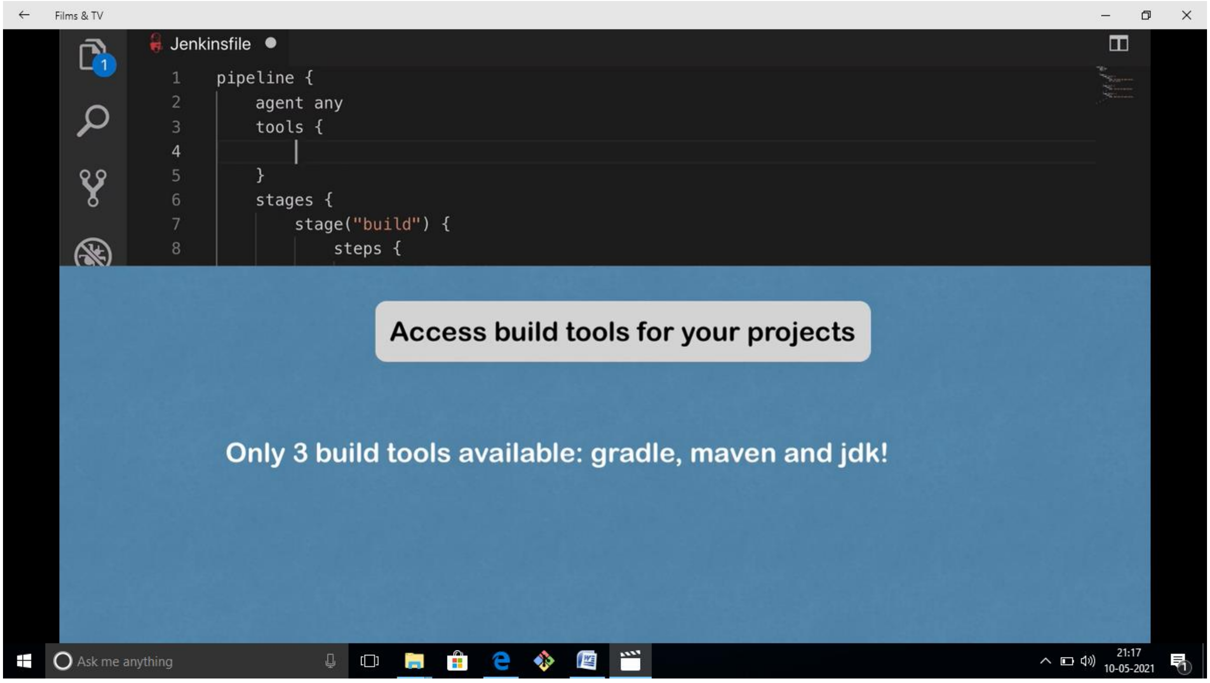
```
pipeline {
    agent any
    tools {
        |
    }
    stages {
```

Access build tools for your projects



```
pipeline {
    agent any
    tools {
        |
    }
    stages {
```

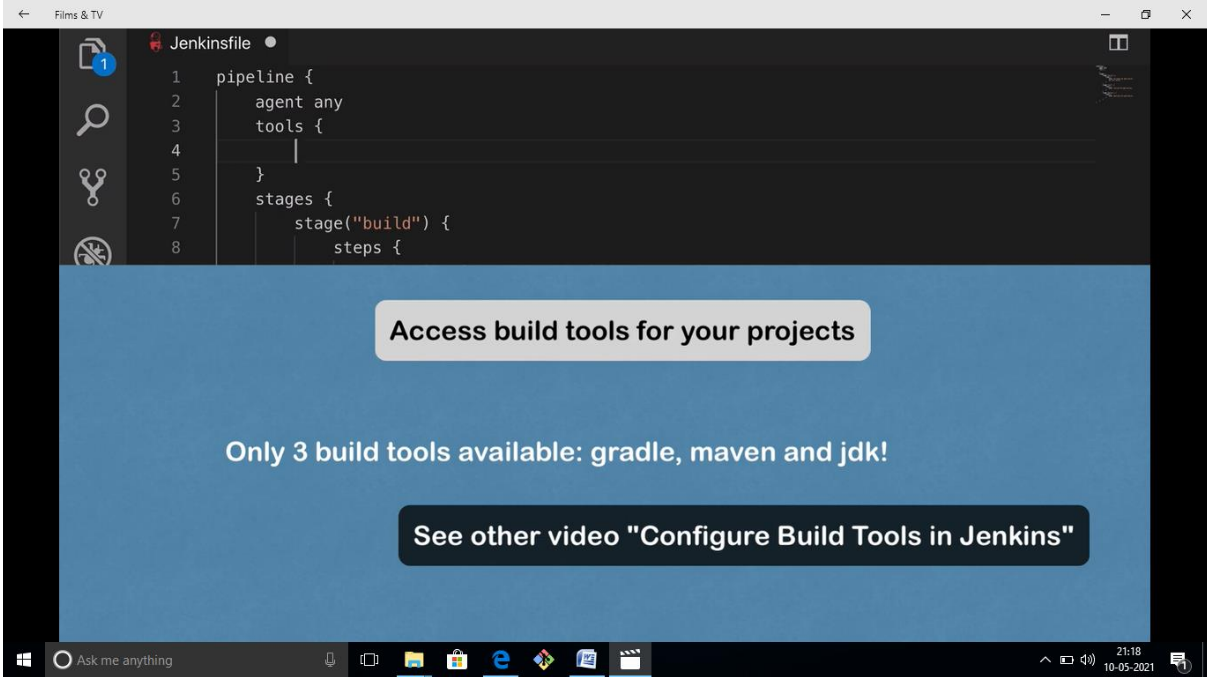
Access build tools for your projects



```
pipeline {
    agent any
    tools {
        |
    }
    stages {
        stage("build") {
            steps {
```

Access build tools for your projects

Only 3 build tools available: gradle, maven and jdk!

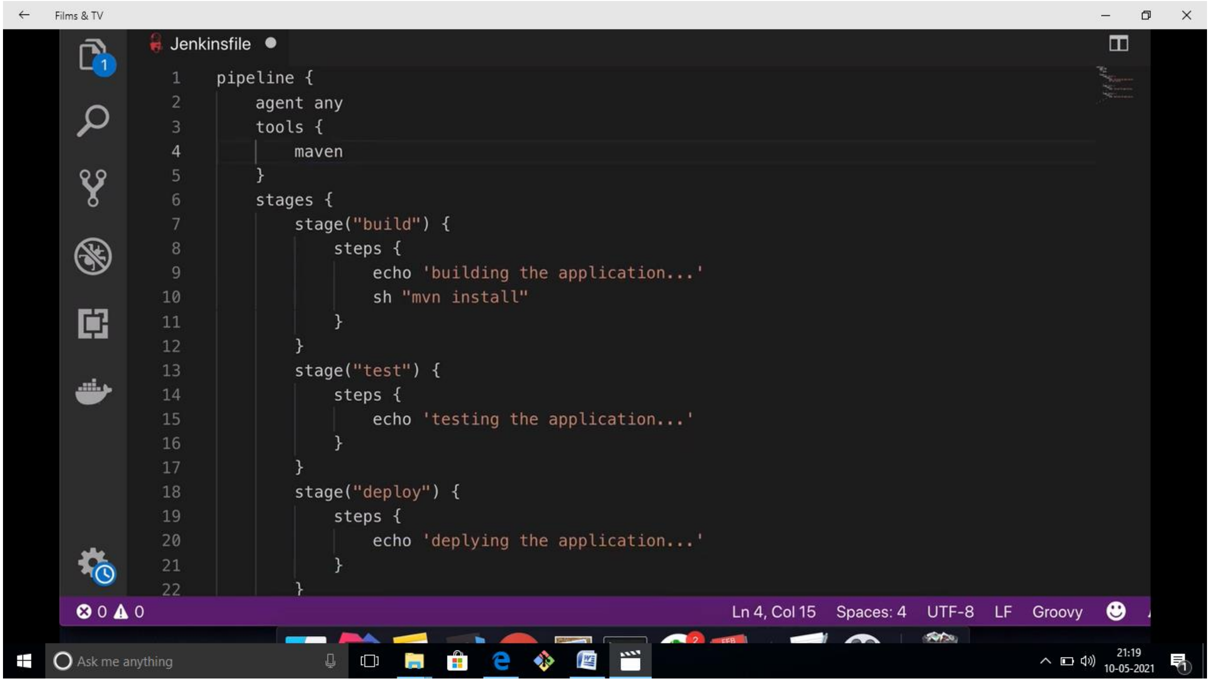


```
pipeline {
    agent any
    tools {
        |
    }
    stages {
        stage("build") {
            steps {
```

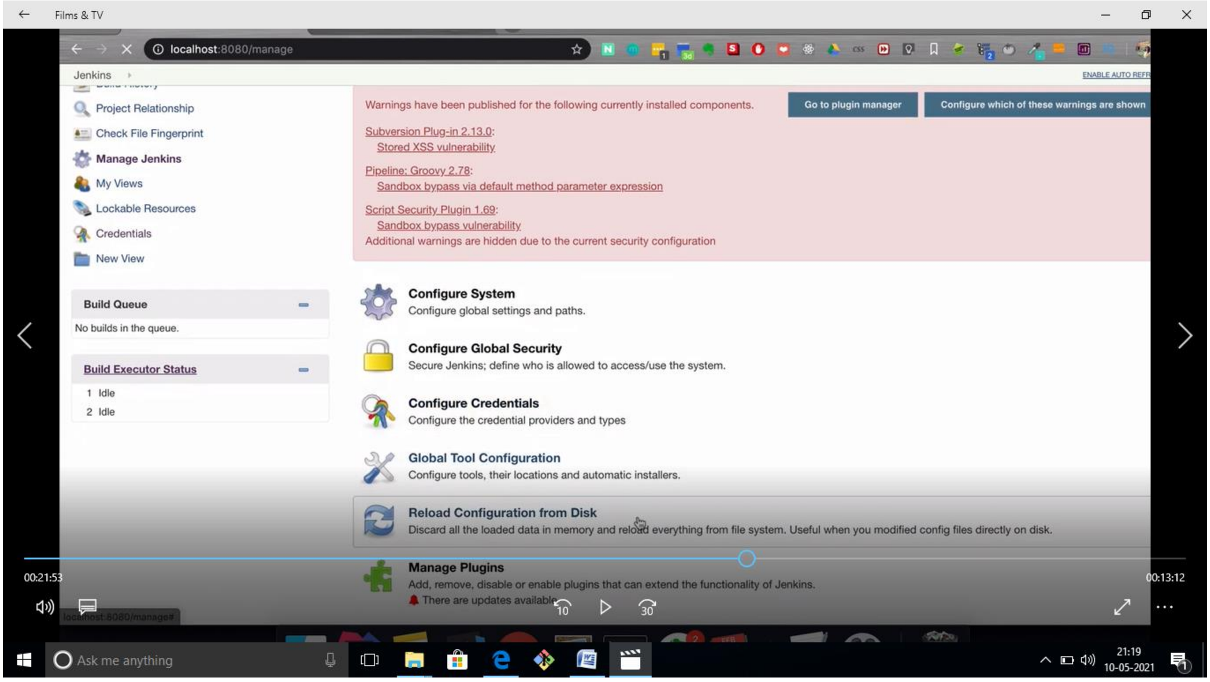
Access build tools for your projects

Only 3 build tools available: gradle, maven and jdk!

See other video "Configure Build Tools in Jenkins"



```
pipeline {
    agent any
    tools {
        maven
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
                sh "mvn install"
            }
        }
        stage("test") {
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deplying the application...'
            }
        }
```



No builds in the queue.

1 Idle
2 Idle

Warnings have been published for the following currently installed components.

Subversion Plug-in 2.13.0:
Stored XSS vulnerability

Pipeline: Groovy 2.78:
Sandbox bypass via default method parameter expression

Script Security Plugin 1.69:
Sandbox bypass vulnerability

Additional warnings are hidden due to the current security configuration

Configure System
Configure global settings and paths.

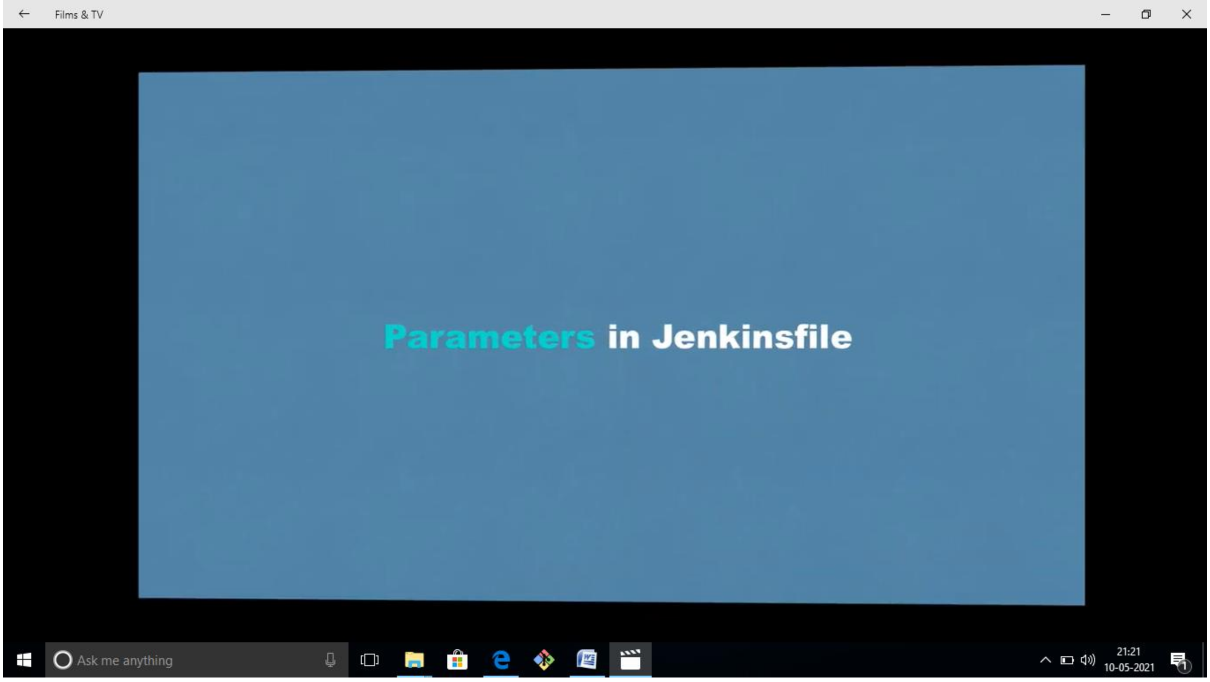
Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

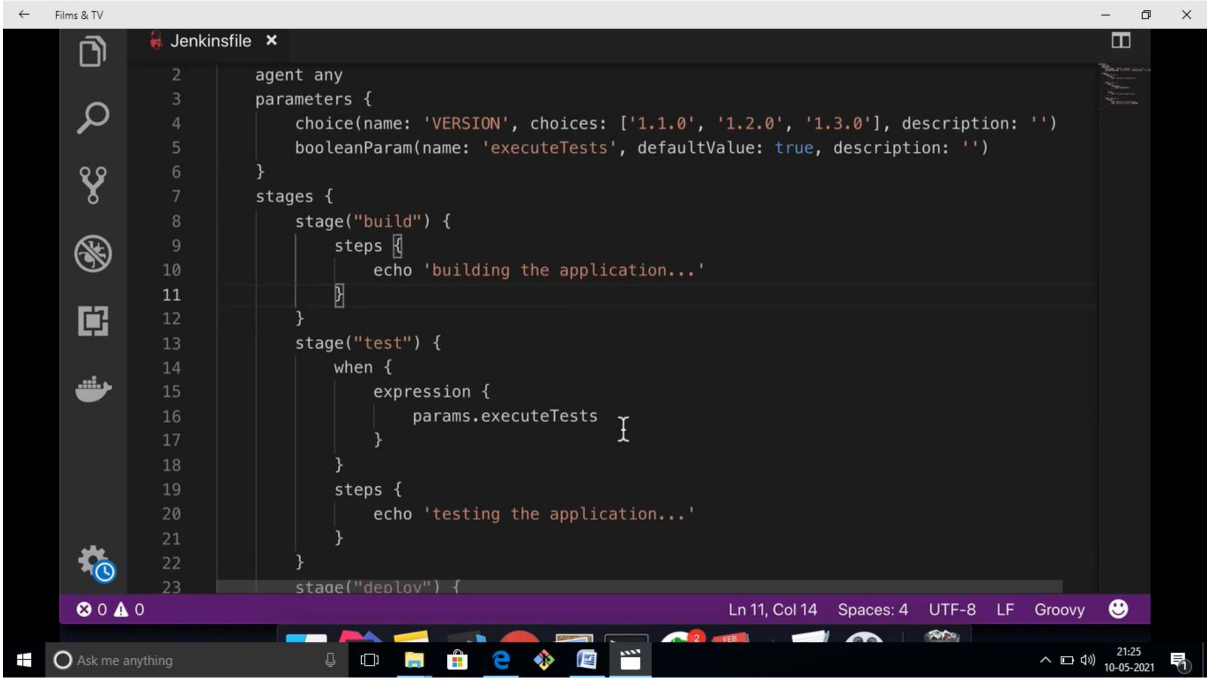
Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
There are updates available

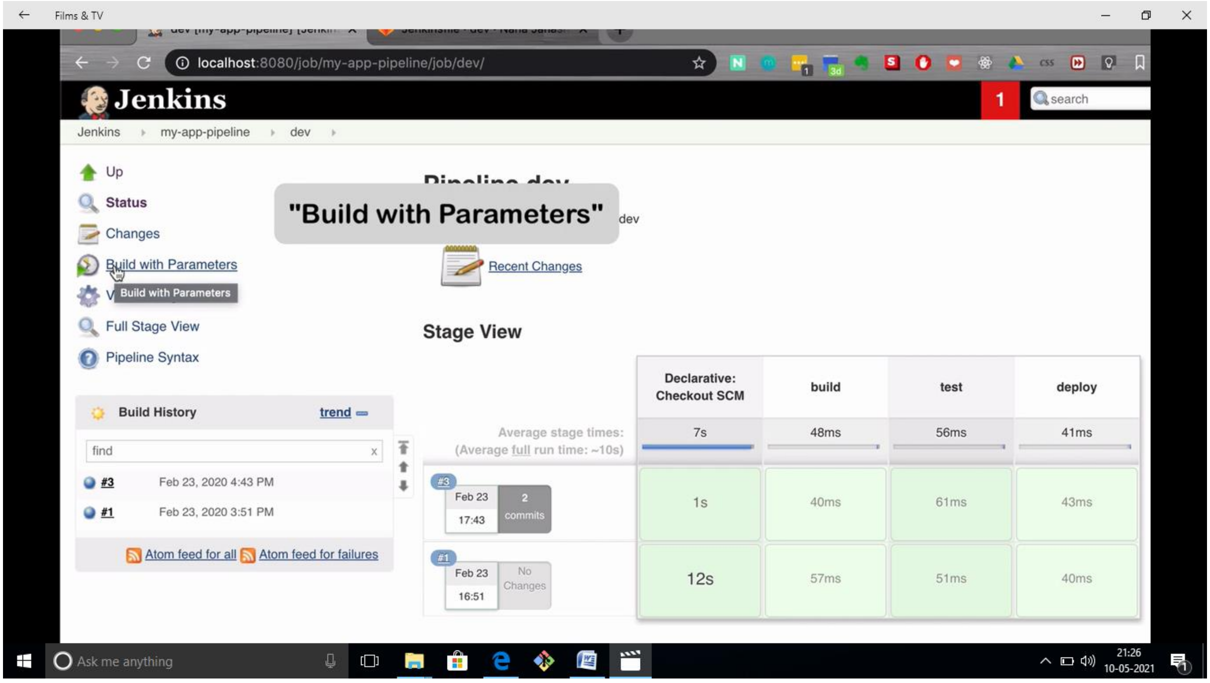


Parameters in Jenkinsfile



```
    agent any
    parameters {
        choice(name: 'VERSION', choices: ['1.1.0', '1.2.0', '1.3.0'], description: '')
        booleanParam(name: 'executeTests', defaultValue: true, description: '')
    }
    stages {
        stage("build") {
            steps {
                echo 'building the application...'
            }
        }
        stage("test") {
            when {
                expression {
                    params.executeTests
                }
            }
            steps {
                echo 'testing the application...'
            }
        }
        stage("deploy") {
```

⊗ 0 ⚠ 0 Ln 11, Col 14 Spaces: 4 UTF-8 LF Groovy

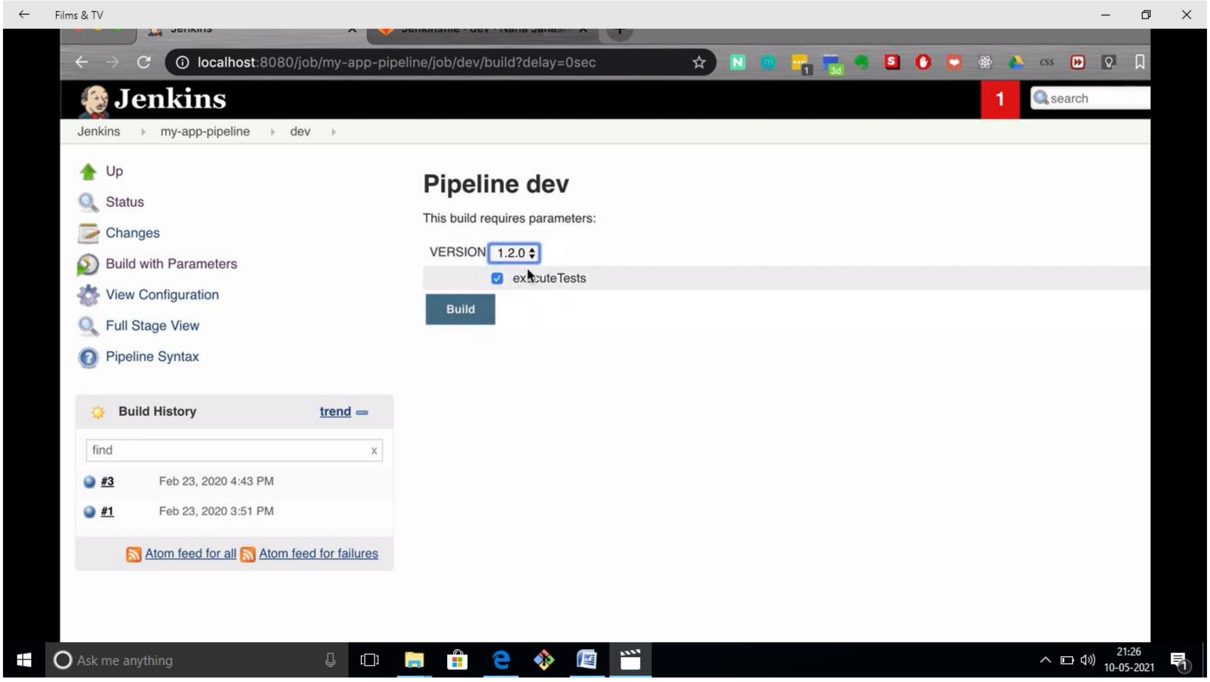


"Build with Parameters"

Build	Date
#3	Feb 23, 2020 4:43 PM
#1	Feb 23, 2020 3:51 PM

Stage View

	Declarative: Checkout SCM	build	test	deploy
Average stage times: (Average full run time: ~10s)	7s	48ms	56ms	41ms
#3 Feb 23 17:43 2 commits	1s	40ms	61ms	43ms
#1 Feb 23 16:51 No Changes	12s	57ms	51ms	40ms



localhost:8080/job/my-app-pipeline/job/dev/build?delay=0sec

Jenkins › my-app-pipeline › dev ›

Pipeline dev

This build requires parameters:

VERSION 1.2.0

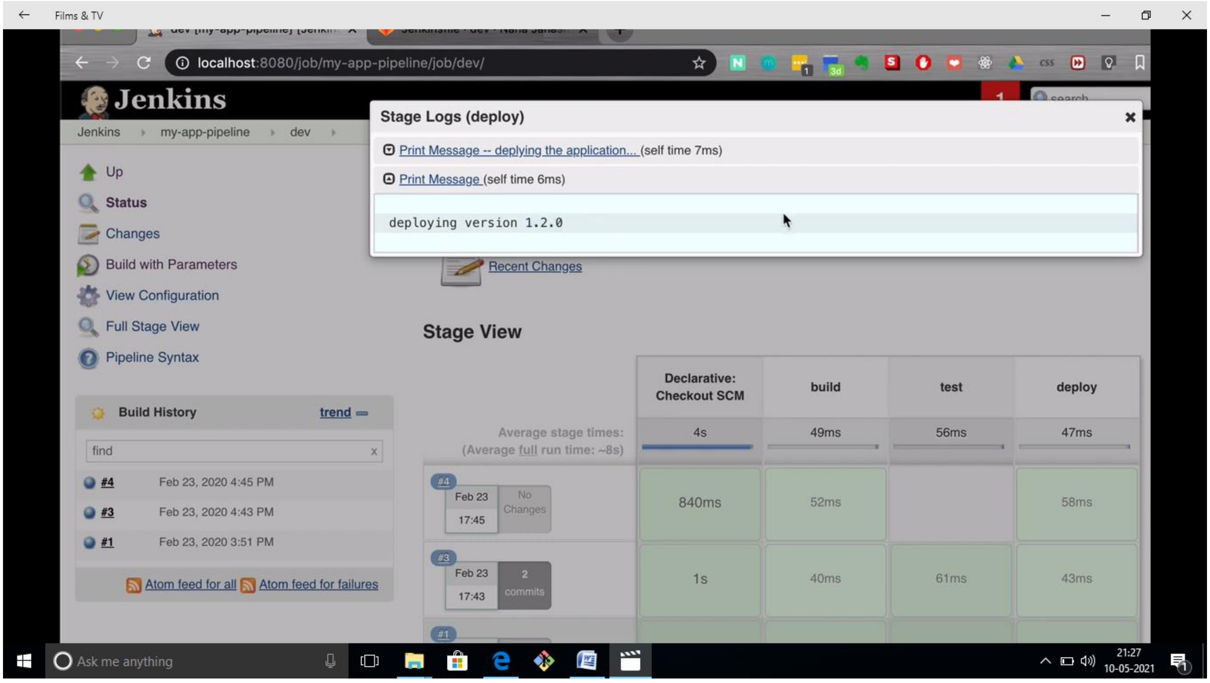
☑ executeTests

Build

Build History trend

#3	Feb 23, 2020 4:43 PM
#1	Feb 23, 2020 3:51 PM

Atom feed for all Atom feed for failures



localhost:8080/job/my-app-pipeline/job/dev/

Jenkins › my-app-pipeline › dev ›

Stage Logs (deploy)

Print Message -- deplying the application... (self time 7ms)

Print Message (self time 6ms)

```
deploying version 1.2.0
```

Stage View

	Declarative: Checkout SCM	build	test	deploy
Average stage times: (Average full run time: ~8s)	4s	49ms	56ms	47ms
#4 Feb 23 17:45 No Changes	840ms	52ms		58ms
#3 Feb 23 17:43 2 commits	1s	40ms	61ms	43ms

Build History trend

#4	Feb 23, 2020 4:45 PM
#3	Feb 23, 2020 4:43 PM
#1	Feb 23, 2020 3:51 PM

Atom feed for all Atom feed for failures