CLOUDBEES UNIVERSITY

CLOUDBEES JENKINS PLATFORM

Pipeline with Docker (Day 4)

TOC

THE PROJECT - PART 2 (DAY 4)

TOC

CLOUDBEES UNIVERSITY

REVIEW OF DAY 3

- Defined the project steps
- Created a new Pipeline job
- Defined steps that clone the code
- Defined steps that run pre-deployment tests
- Defined steps that build and push the container
- Defined steps that deploy the container
- Defined steps that run post-deployment tests

TOC

CLOUDBEES UNIVERSITY

IN THIS UNIT: YOU WILL LEARN

- How to use more advanced features of the CloudBees Pipeline plugin and Docker

IN THIS UNIT: YOU WILL BE ABLE TO

- Create advanced deployment lifecycle with Jenkins Pipeline and Docker

THE PROJECT (CONT.)

- Request manual confirmation before deployment
- Run steps in parallel
- Control the steps execution
- Add checkpoints on critical or after long running processes
- Visualize the deployment pipeline
- Convert the job into a template
- Leverage multi branch pipeline and Jenkinsfile features

KEY PIPELINE DSL – INPUT: TASK

- Add an input with the message Please confirm deployment to production and the button with the text Submit.
- The input should be requested prior to deployment to production.
- Add the parameter for additional notes and echo the value.
- Make sure that only the user manager is allowed to use this input.
- Use user/pass manager/manager to login before approving the deployment.

KEY PIPELINE DSL – INPUT: SOLUTION

```
def response = input message: 'Please confirm deployment to production',
    ok: 'Submit',
    parameters: [[
        $class: 'StringParameterDefinition',
        defaultValue: '',
        description: 'Additional comments',
        name: ''
    ]],
    submitter: 'manager'
echo response
```

KEY PIPELINE DSL – PARALLEL: TASK

Modify the script so that service and db containers are pulled in parallel.

CLOUDBEES UNIVERSITY

KEY PIPELINE DSL – PARALLEL: SOLUTION

```
//    docker.image("localhost:5000/training-books-ms").pull()
//    docker.image("mongo").pull()
    def pull = [:]
    pull["service"] = {
        docker.image("localhost:5000/training-books-ms").pull()
    }
    pull["db"] = {
        docker.image("mongo").pull()
    }
    parallel pull
```

TOC

CLOUDBEES UNIVERSITY

REALITY CHECK

- Questions on the prep and the Project?
- Clear on the input and parallel DSL?
- Ready for the execution control?

TOC

KEY PIPELINE DSL – EXECUTION CONTROL

```
retry(10) {
    // some block
}

sleep time: 10, unit: 'MINUTES'

waitUntil {
    // some block
}

timeout(time: 100, unit: 'SECONDS') {
    // some block
}

while(something) {
    // do something
    if (something_else) {
        // do something else
    }
```

KEY PIPELINE DSL – EXECUTION CONTROL: TASK

Change the script so that the execution of post-deployment tests is retried in case of a failure.

KEY PIPELINE DSL – EXECUTION CONTROL: SOLUTION

```
retry(2) {
    sh "./run_tests.sh"
}
```

KEY PIPELINE DSL – EXECUTION CONTROL: TASK

Change the script so that there is a pause of 2 seconds after the service is deployed.

KEY PIPELINE DSL – EXECUTION CONTROL: SOLUTION

```
sleep 2
```

MID-BREAK

(10) minutes for learner re-integration.

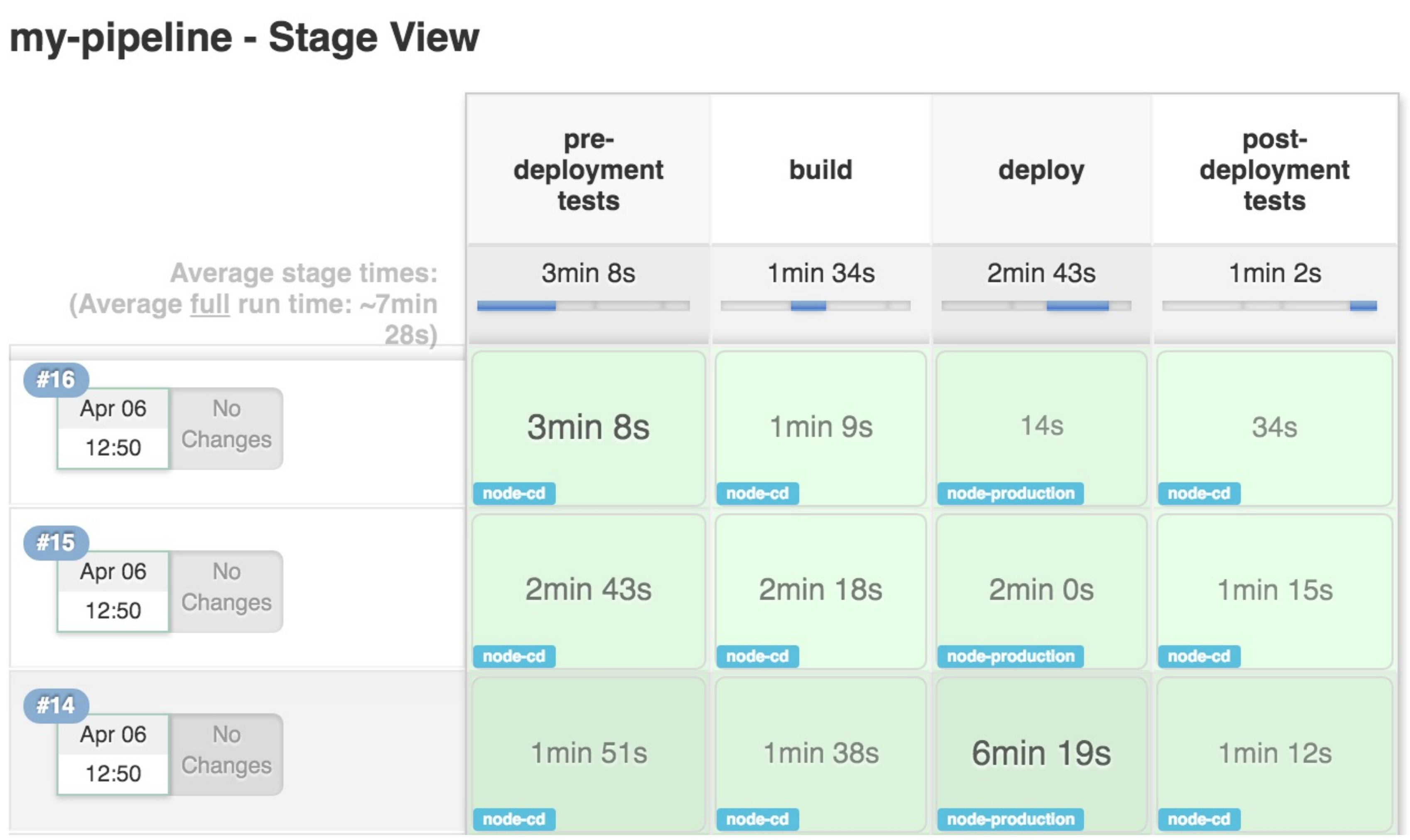
KEY PIPELINE DSL – CHECKPOINT: TASK

Add checkpoint deploy between the nodes cd and production.

KEY PIPELINE DSL – CHECKPOINT: SOLUTION

```
checkpoint "deploy"
```

PIPELINE STAGE VIEW



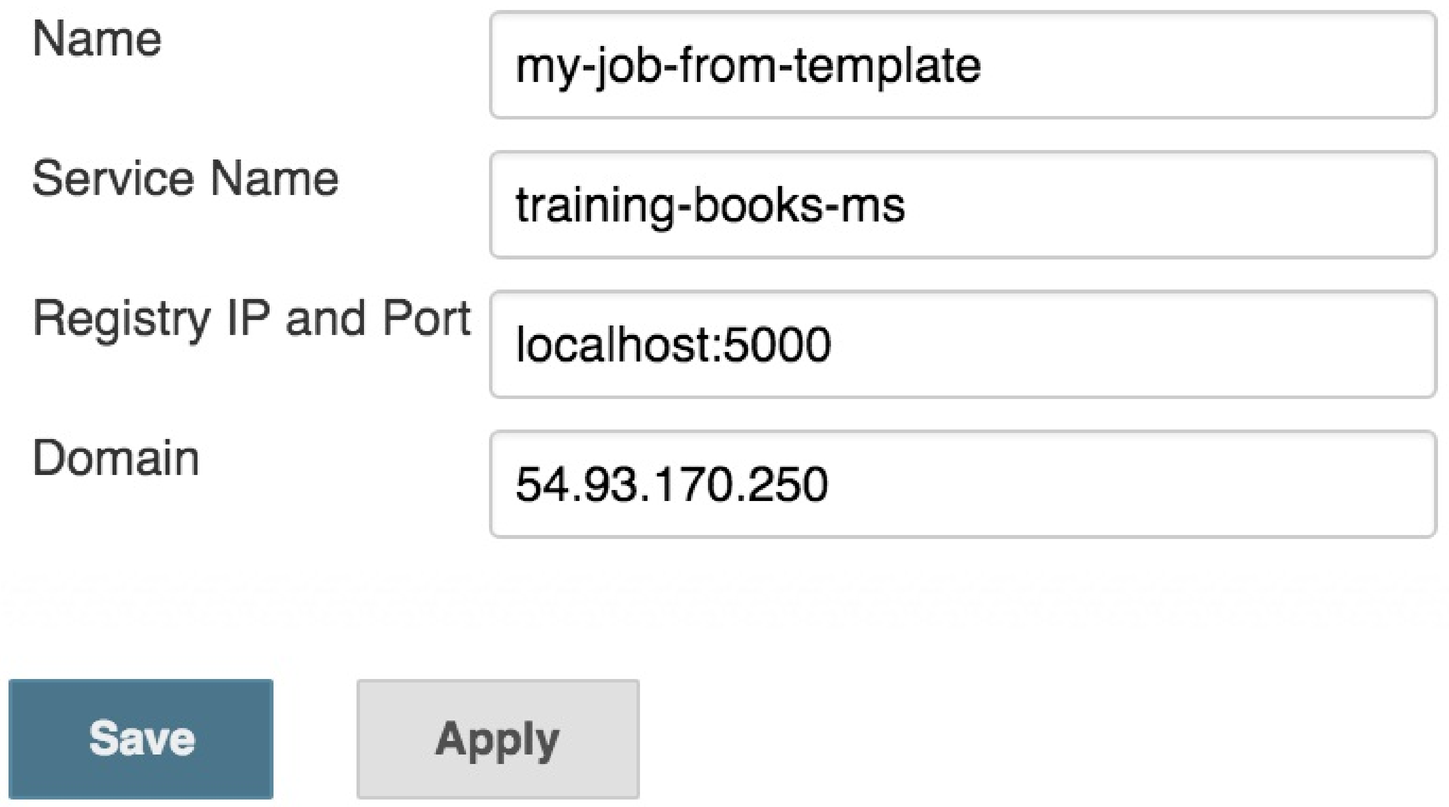
my-pipeline - Stage View

	pre-deployment tests	build	deploy	post-deployment tests
Average stage times: (Average full run time: ~7min 28s)	3min 8s	1min 34s	2min 43s	1min 2s
#16 Apr 06 12:50 No Changes	3min 8s node-cd	1min 9s node-cd	14s node-production	34s node-cd
#15 Apr 06 12:50 No Changes	2min 43s node-cd	2min 18s node-cd	2min 0s node-production	1min 15s node-cd
#14 Apr 06 12:50 No Changes	1min 51s node-cd	1min 38s node-cd	6min 19s node-production	1min 12s node-cd

PIPELINE AND JOB TEMPLATE: TASK

- Convert the deployment pipeline script into a template named my-template.
- Replace the service name (training-books-ms), the registry IP and port (localhost:5000) and domain (http://<IP>:8081) with variables.
- Create a new job called my-pipeline-from-template. The job type should be my-template.

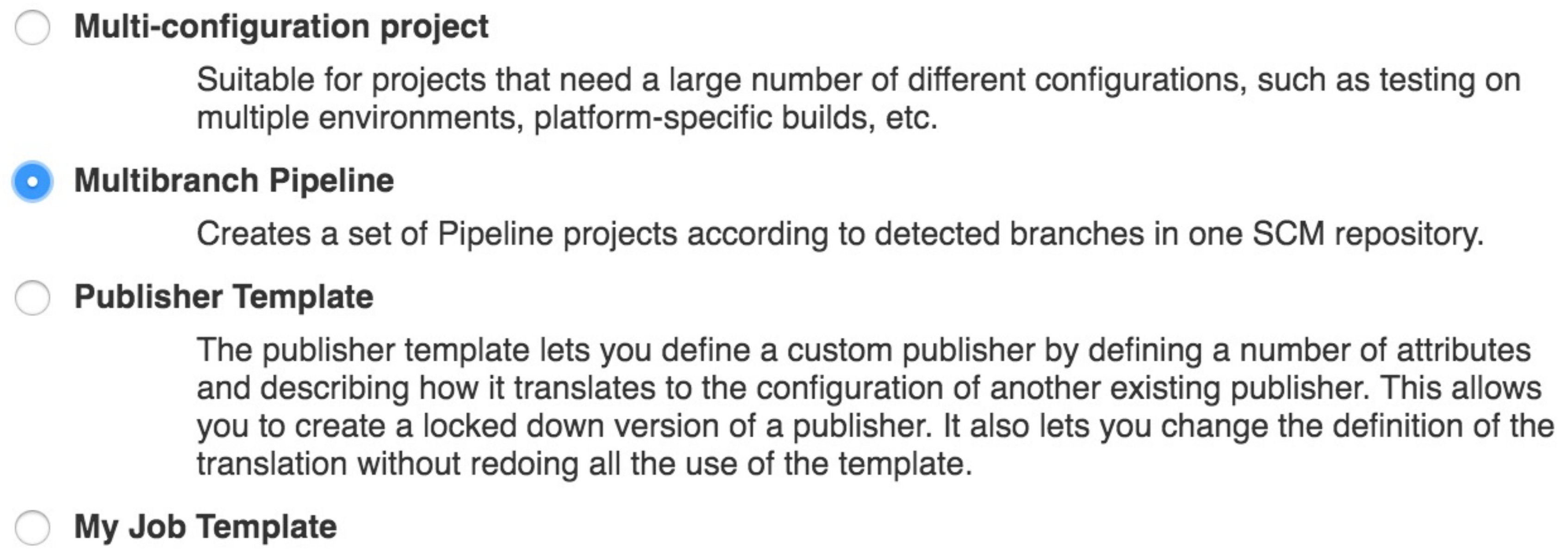
PIPELINE AND JOB TEMPLATE: SOLUTION



Name	my-job-from-template
Service Name	training-books-ms
Registry IP and Port	localhost:5000
Domain	54.93.170.250

CLOUDBEES UNIVERSITY

MULTI-BRANCH PIPELINE AND JENKINSFILE



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

[✓ **Multibranch Pipeline**]

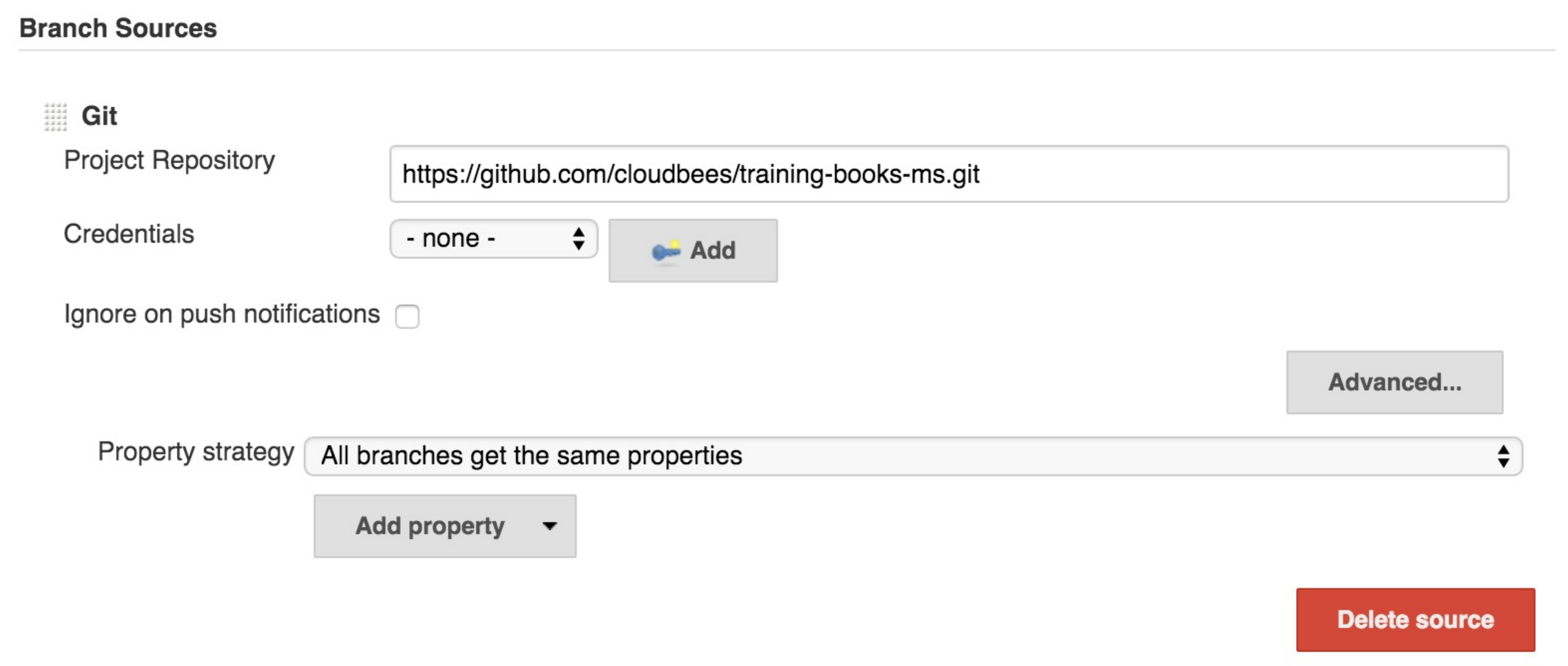
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Publisher Template

The publisher template lets you define a custom publisher by defining a number of attributes and describing how it translates to the configuration of another existing publisher. This allows you to create a locked down version of a publisher. It also lets you change the definition of the translation without redoing all the use of the template.

My Job Template

MULTI-BRANCH PIPELINE AND JENKINSFILE



Branch Sources

Git

Project Repository: https://github.com/cloudbees/training-books-ms.git

Credentials: - none -

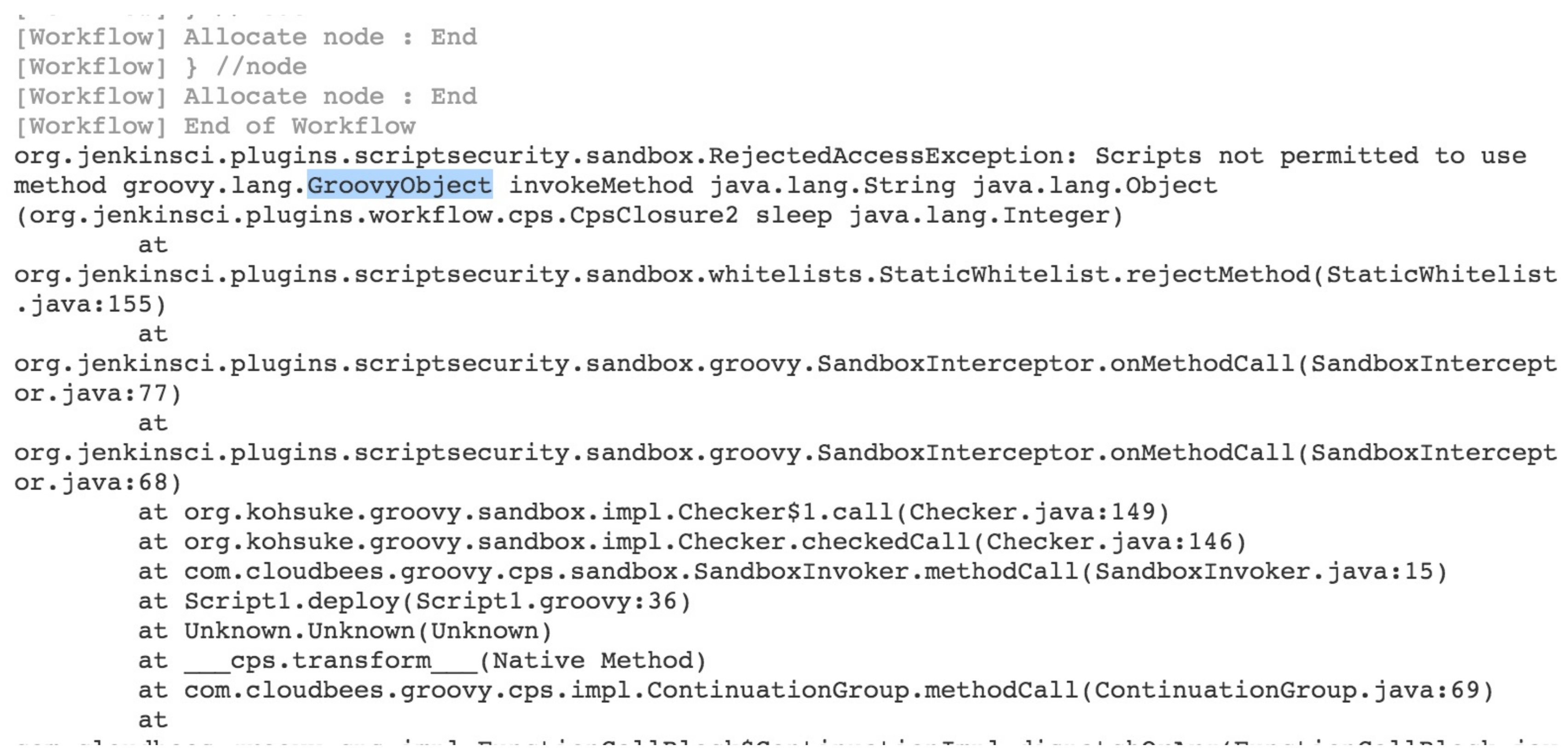
Ignore on push notifications: ☐

Property strategy: All branches get the same properties

MULTI-BRANCH PIPELINE AND JENKINSFILE

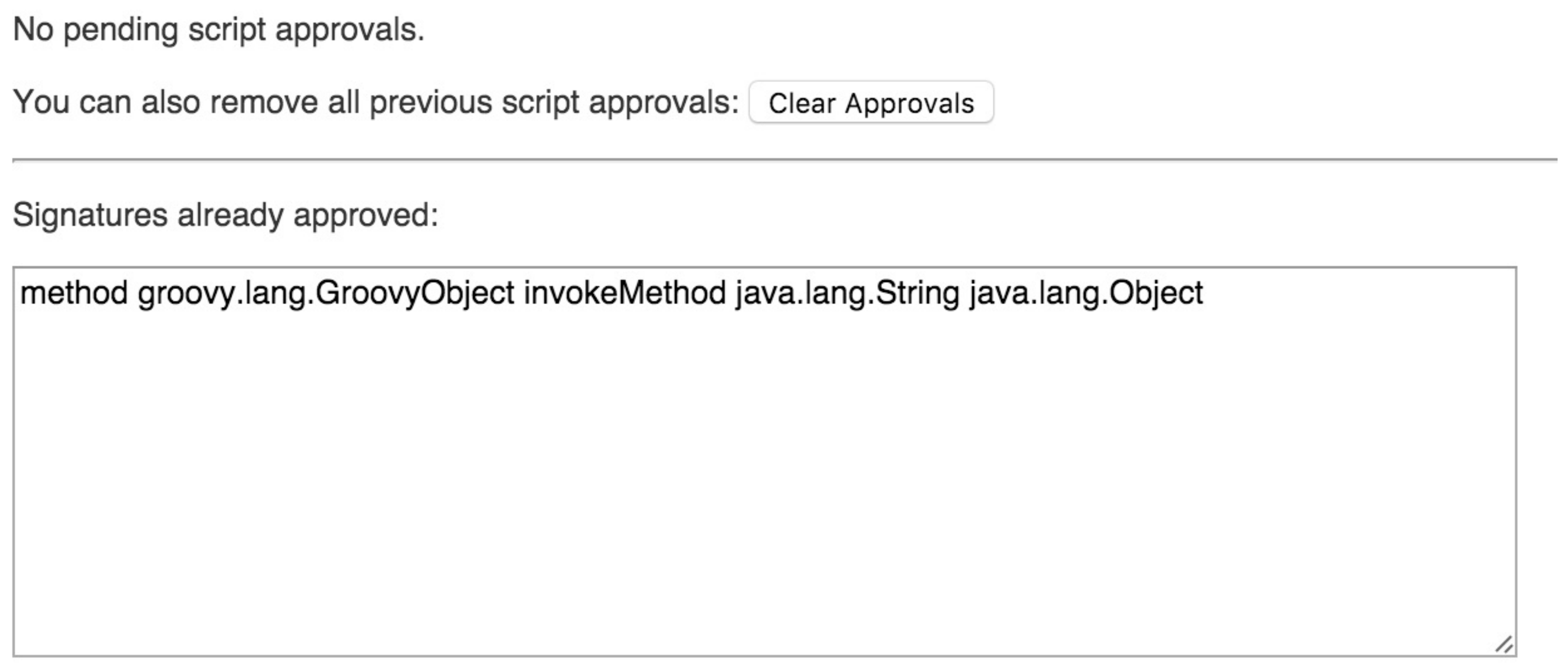
```
def serviceName = "training-books-ms"
def registry = "localhost:5000"
def flow

node("cd") {
    git "https://github.com/cloudbees/${serviceName}.git"
    flow = load "/mnt/scripts/pipeline-common.groovy"
    flow.runPreDeploymentTests(serviceName, registry)
    flow.build(serviceName, registry)
}
checkpoint "deploy"
node("cd") {
    flow.deploy(serviceName, registry)
    flow.runPostDeploymentTests(serviceName, registry, "http://<IP>:8081")
}
```

MULTI-BRANCH PIPELINE AND JENKINSFILE



```
[Workflow] Allocate node : End
[Workflow] } //node
[Workflow] Allocate node : End
[Workflow] End of Workflow
org.jenkinsci.plugins.scriptsecurity.sandbox.RejectedAccessException: Scripts not permitted to use
method groovy.lang.GroovyObject invokeMethod java.lang.String java.lang.Object
(org.jenkinsci.plugins.workflow.cps.CpsClosure2 sleep java.lang.Integer)
        at
org.jenkinsci.plugins.scriptsecurity.sandbox.whitelists.StaticWhitelist.rejectMethod(StaticWhitelist
.java:155)
        at
org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SandboxInterceptor.onMethodCall(SandboxIntercept
or.java:77)
        at
org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.SandboxInterceptor.onMethodCall(SandboxIntercept
or.java:68)
        at org.kohsuke.groovy.sandbox.impl.Checker$1.call(Checker.java:149)
        at org.kohsuke.groovy.sandbox.impl.Checker.checkedCall(Checker.java:146)
        at com.cloudbees.groovy.cps.sandbox.SandboxInvoker.methodCall(SandboxInvoker.java:15)
        at Script1.deploy(Script1.groovy:36)
        at Unknown.Unknown(Unknown)
        at ___cps.transform___(Native Method)
        at com.cloudbees.groovy.cps.impl.ContinuationGroup.methodCall(ContinuationGroup.java:69)
        at
```

MULTI-BRANCH PIPELINE AND JENKINSFILE



No pending script approvals.

You can also remove all previous script approvals:

Signatures already approved:

```
method groovy.lang.GroovyObject invokeMethod java.lang.String java.lang.Object
```

PIPELINE GLOBAL LIBRARY

- Ability to share common parts of Pipeline scripts across multiple jobs
- Keep scripts DRY
- Shared library script Git repository
- Pipeline step load vs global library

Please visit Pipeline Global Library (https://github.com/jenkinsci/workflow-cps-global-lib-plugin/blob/master/README.md) for more info.

THE PROJECT - PART 2: REVIEW

THE PROJECT - PART 2: REVIEW

- Pipeline Job
- Job Template
- Multibranch Pipeline And Jenkinsfile

COURSE REVIEW

COURSE REVIEW

- The Need For The Pipeline
- What Is Cloudbees Pipeline?
- The Syntax And The Snippet Generator
- Docker Containers
- Docker Tools
- The Project

THE PROJECT - PART 2: EXERCISE

The Project - Part 2: Exercise

SYLLABUS AND REFERENCES

- CloudBees Pipeline Plugin Suite (http://documentation.cloudbees.com/docs/cje-user-guide/workflow.html)
- CloudBees Docker Pipeline Plugin (http://documentation.cloudbees.com/docs/cje-user-guide/docker-workflow.html)
- Docker Docs (https://docs.docker.com/)