

Jenkins practical task

1. Install Jenkins according to documentation (Java based or docker based).
2. Make a base setup of Jenkins (User configuration, plugin installation).
3. Add a Dockerfile to your repository with spring-petclinic.
4. Create 2 docker repositories on your own Nexus Repository ([Instruction](#)) or <https://hub.docker.com/> called "main" and "mr".
5. Add Jenkinsfile and describe the following behavior there:
 - a. The pipeline for a merge request should include the following jobs:
 - i. Checkstyle: Use [Maven](#) or [Gradle](#) checkstyle plugin to generate a code style report. It should be available as a job artifact.
 - ii. Test (with Maven or Gradle)
 - iii. Build: Build without tests (with Maven or Gradle).
 - iv. Create a docker image: Using your Dockerfile in the spring-petclinic repo, create a docker image with spring-petclinic application, tag it using GIT_COMMIT (short) and push it to the "mr" repository.

Note: Pipelines should be executed in Jenkins agents
 - b. The pipeline for the main branch should include the following job:
 - i. Create a docker image: Build a docker image and push it to the "main" repository.

Note: Pipelines should be executed in Jenkins agents

1. Install Jenkins according to documentation (Java based or docker based).

I have decided to use Docker based Jenkins distribution, so first we need to run and configure Docker image for Jenkins master and agent node.

Run container with Jenkins master:

```
05:14:15 adrwal@olek-desktop-pc jenkins → docker run \
--name jenkins-blueocean-task \
--restart=on-failure \
--detach \
--network jenkins \
--publish 8080:8080 \
--publish 50000:50000 \
--volume jenkins-data:/var/jenkins_home \
jenkins/jenkins
Unable to find image 'jenkins/jenkins:latest' locally
latest: Pulling from jenkins/jenkins
Digest: sha256:4bc56852fb6e174a634c1e0615ffbb60441bea74d91f2684f2f1668ffdd107dd
Status: Downloaded newer image for jenkins/jenkins:latest
a08828fb26862924c3f884bb5f4da87d4bb4de1526866acd7aa3924f5271933b
05:14:55 adrwal@olek-desktop-pc jenkins →
```

Run container with Docker daemon, because our agent will need to use Docker to build spring-petclinic app:

```
05:20:16 adrwal@olek-desktop-pc jenkins → docker run \
  --name jenkins-docker \
  --rm \
  --detach \
  --privileged \
  --network jenkins \
  --network-alias docker \
  --env DOCKER_TLS_CERTDIR=/certs \
  --volume jenkins-docker-certs:/certs/client \
  --volume jenkins-data:/var/jenkins_home \
  --publish 2376:2376 \
  docker:dind \
  --storage-driver overlay2
d235a1686b68ce2c83e14f35141fecafbba76aacc3f24190864fdc26885427a6
05:21:08 adrwal@olek-desktop-pc jenkins →
```

Run agent:

```
05:19:12 adrwal@olek-desktop-pc jenkins → docker run -it --rm --name agent \
  --network jenkins \
  --env DOCKER_HOST=tcp://docker:2376 \
  --env DOCKER_CERT_PATH=/certs/client \
  --env DOCKER_TLS_VERIFY=1 \
  --volume jenkins-docker-certs:/certs/client:ro \
  jenkins/jnlp-agent-docker
WARNING: Are you running agent from an interactive console?
If so, you are probably using it incorrectly.
See https://wiki.jenkins.io/display/JENKINS/Launching+agent+from+console
<===[JENKINS REMOTING CAPACITY]===>r00ABXNyABpodWRzb24ucmVtb3RpbmcuQ2FwYWJpbGl0eQAAAAAAAAABAgABSgAEbWFza3hwAAAAAAAAAAf4=
```

At the end we should have 3 running containers.

```
05:21:08 adrwal@olek-desktop-pc jenkins → docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d235a1686b68	docker:dind	"dockerd-entrypoint..."	18 seconds ago	Up 17 seconds	2375/tcp, 0.0.0.0:2376->2376/tcp	jenkins-docker
9c82abd52b77	jenkins/jnlp-agent-docker	"/usr/local/bin/jenk..."	About a minute ago	Up About a minute		agent
a08828fb2686	jenkins/jenkins	"/usr/bin/tini -- /u..."	6 minutes ago	Up 6 minutes	0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp	jenkins-blueocean-task

```
05:21:26 adrwal@olek-desktop-pc jenkins →
```

2. Make a base setup of Jenkins (User configuration, plugin installation).

We can now access jenkins on localhost:8080

← → ↻ localhost:8080/login?from=%2F

Robota ACloudGuru In... ACloudGuru In... ACloudGuru In... Docker

Zaczynamy

Odblokuj Jenkinsa

Aby zapewnić, że Jenkins jest bezpiecznie uruchomiony przez administratora, hasło zostało zapisane do pliku logów (nie masz pewności, gdzie go znaleźć?) oraz w pliku na serwerze:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Skopiuj hasło z jednej z powyższych lokalizacji i wklej poniżej.

Hasło administratorskie:

```
05:27:36 adrwal@olek-desktop-pc ~ → docker exec -it jenkins-blueocean-task cat /var/jenkins_home/secrets/initialAdminPassword
cd392107cd794b0b8cf1274a64a853ea
05:29:19 adrwal@olek-desktop-pc ~ →
```

← → ↻ localhost:8080

Robota ACloudGuru In... ACloudGuru In... ACloudGuru In... Docker

Instalacja wtyczek

Instalacja wtyczek

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding	** Icons API
<input type="checkbox"/> Timestampers	<input type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input type="checkbox"/> Gradle	Folders
<input type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Pipeline Graph View	OWASP Markup Formatter
<input type="checkbox"/> Git	<input type="checkbox"/> SSH Build Agents	<input type="checkbox"/> Matrix Authorization Strategy	<input type="checkbox"/> PAM Authentication	** ASM API
<input type="checkbox"/> LDAP	<input type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer	<input type="checkbox"/> Dark Theme	

** - zależności wymagane

Jenkins 2.487

localhost:8080

ACloudGuru In... ACloudGuru In... ACloudGuru In... Docker

Dodawanie pierwszego użytkownika

Stwórz pierwszego administratora

Login

admin

Hasło

.....

Powtórz hasło

.....

Pełna nazwa

Aleksander Drwal

Adres e-mail

adrwal@griddynamics.com

Jenkins 2.487

Pomiń i kontynuuj jako admin

Zapisz i zakończ

localhost:8080

ACloudGuru In... ACloudGuru In... ACloudGuru In... Docker

Dodawanie pierwszego użytkownika

Konfiguracja instancji

URL Jenkinsa:

http://localhost:8080/

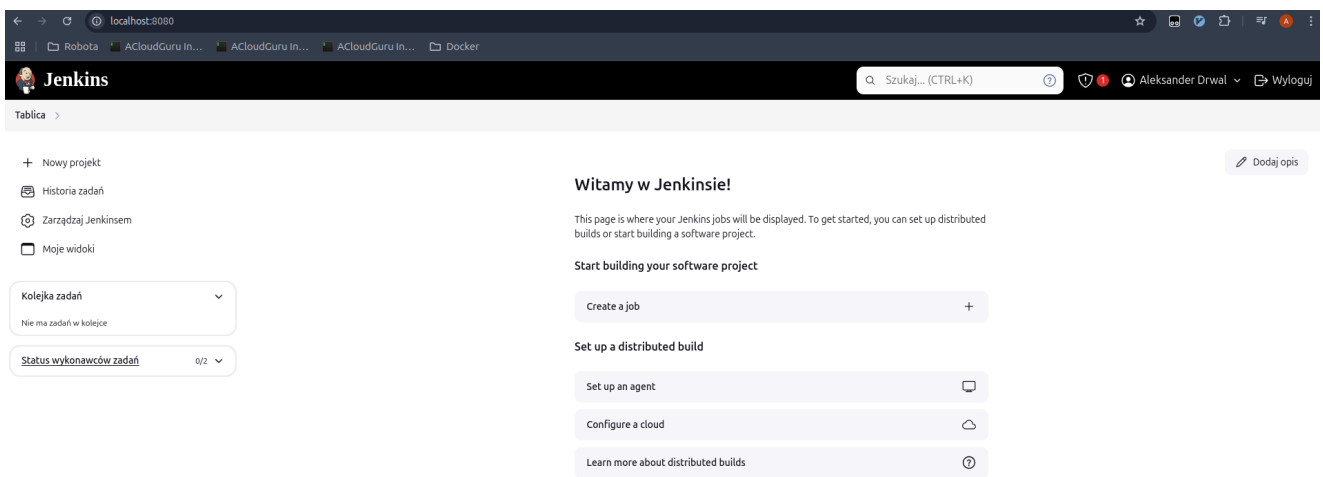
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.487

Nie teraz

Zapisz i zakończ



After the initial setup is complete, we need to add our Docker based agent. **Manage Jenkins** > **Nodes** > **+ New Node**

New node

Node name

docker-agent-1

Type



Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

Number of executors ?

1

Remote root directory ?

/home/jenkins

Labels ?

docker

Usage ?

Use this node as much as possible

Launch method ?

Launch agent by connecting it to the controller

Availability ?

Keep this agent online as much as possible

Node Properties

Save

After we click Save, our agent still isn't available. We need to launch `agent.jar` on it to make it available to Jenkins master.

Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space
	Built-In Node	Linux (amd64)	In sync	
	docker-agent-1		N/A	
Data obtained		8 sec	8 sec	

docker-agent-1

Agent docker-agent-1

Add descriptionMark this node temporarily offline

Run from agent command line: (Unix)

```
curl -s0 http://localhost:8080/jnlpJars/agent.jar
java -jar agent.jar -url http://localhost:8080/ -secret a339961324bce981397d06533d06520ee6887b7cfd9a93f8b130b2a54a2c58f -name "docker-agent-1" -webSocket -workDir "/home/jenkins"
```





Run from agent command line: (Windows)

I changed localhost from above snippet to host's IP to access Jenkins master from inside of the agent's container.

```
05:39:49 adrwai@olek-desktop-pc ~$ docker exec -it agent bash
9c82abd52b77:~$ cd agent/
9c82abd52b77:~$ curl --output agent.jar http://192.168.1.167:8080/jnlpJars/agent.jar
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 1362k 100 1362k 0 0 40.0M 0 --:--:-- --:--:-- --:--:-- 40.3M
9c82abd52b77:~$ agent$ ls
agent.jar
9c82abd52b77:~$ agent$ java -jar agent.jar -url http://192.168.1.167:8080/ -secret a339961324bce981397d06533d06520ee6887b7cfd9a93f8b130b2a54a2c58f -name "docker-agent-1" -webSocket -workDir "/home/jenkins"
Nov 27, 2024 4:41:05 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/jenkins/remoting as a remoting work directory
Nov 27, 2024 4:41:05 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/jenkins/remoting
Nov 27, 2024 4:41:05 PM hudson.remoting.Launcher createEngine
INFO: Setting up agent: docker-agent-1
Nov 27, 2024 4:41:05 PM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3283.v92c105e0f819
Nov 27, 2024 4:41:05 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/jenkins/remoting as a remoting work directory
Nov 27, 2024 4:41:06 PM hudson.remoting.Launcher$CuiListener status
INFO: WebSocket connection open
Nov 27, 2024 4:41:06 PM hudson.remoting.Launcher$CuiListener status
INFO: Connected
[]
```

Nodes

+ New NodeConfigure MonitorsRefresh

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	47.32 GiB	1024.00 MiB	47.32 GiB	0ms 
	docker-agent-1	Linux (amd64)	In sync	47.32 GiB	1024.00 MiB	47.32 GiB	109ms 
Data obtained		15 sec	15 sec	15 sec	15 sec	15 sec	15 sec

Icon: S M L

Legend

We can validate that this agent works as expected by running simple job on it which runs docker command `Create a job > Freestyle project`.

We leave everything default and just set 2 below fields:

☒ Restrict where this project can be run ?

Label Expression ?

docker

Label `docker` matches 1 node. Permissions or other restri

Build Steps

Automate your build process with ordered tasks like code comp

≡ Execute shell ?

Command


See [the list of available environment variables](#)

```
docker run --rm ubuntu
```

After running the build, we can see that everything is working correctly.

 Status

 Changes

 Console Output

 Edit Build Information

 Timings

Console Output

```
Started by user Aleksander Drwal
Running as SYSTEM
Building remotely on docker-agent-1 (docker) in workspace /home/jenkins/workspace/docker-test
[docker-test] $ /bin/sh -xe /tmp/jenkins10696336494306289559.sh
+ docker run --rm ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
afad30e59d72: Pulling fs layer
afad30e59d72: Verifying Checksum
afad30e59d72: Download complete
afad30e59d72: Pull complete
Digest: sha256:278628f08d4979fb9af9ead44277dbc9c92c2465922310916ad0c46ec9999295
Status: Downloaded newer image for ubuntu:latest
Finished: SUCCESS
```

We can also install Docker pipeline plugin for easier use of docker in pipelines.

[Report an issue with this plugin](#)

Docker Pipeline 580.vc0c340686b_54

Build and use Docker containers from pipelines.

[Report an issue with this plugin](#)



3. Add a Dockerfile to your repository with spring-petclinic.

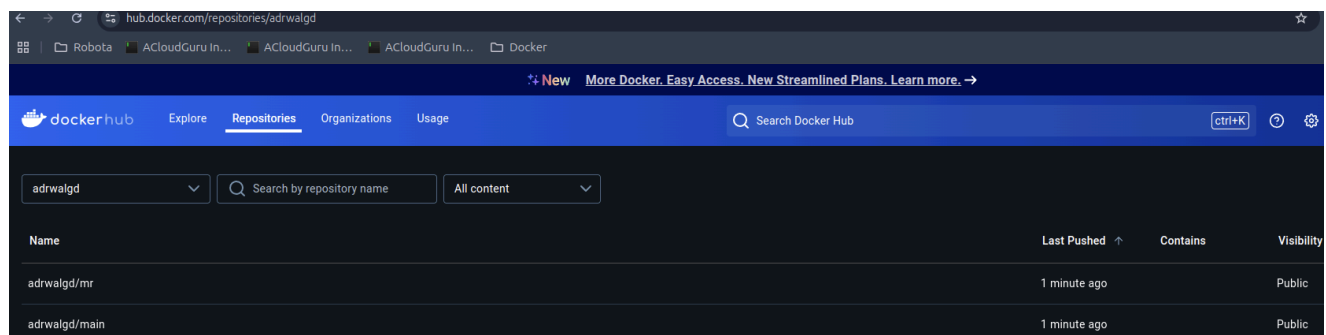
Dockerfile contents:

```
FROM maven:3.8.7-openjdk-18-slim AS build

RUN mkdir /app
COPY . /app
WORKDIR /app
RUN mvn package

# Minimal runtime image - only JRE
FROM gcr.io/distroless/java21-debian12 AS runtime
COPY --from=build /app/target/*.jar /app.jar
ENTRYPOINT [ "java" ]
CMD [ "-jar", "/app.jar" ]
```

4. Create 2 docker repositories on your own Nexus Repository ([Instruction](#)) or <https://hub.docker.com/> called “main” and “mr”.



5. Pipelines

a. merge request pipeline

First we need to add credentials for Github and Docker hub.

Create Github access token for Jenkins to setup webhooks/access private repository and add it to jenkins (Dashboard > Manage Jenkins > Credentials > System > Global credentials > + Add Credentials).

Note

jenkins-lab

What's this token for?

Expiration

This token expires *on Tue, Dec 3 2024*. To set a new expiration date, you must [regenerate the token](#).

Select scopes

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

<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks

Jenkins Credentials Provider: Jenkins

Kind

Username with password

Scope ?

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

Username ?

adrwalGD

☐ Treat username as secret ?

Password ?

.....

ID ?

github-token

Description ?

Github Token

Cancel Add

Add credentials for docker hub Dashboard > Manage Jenkins > Credentials > System > Global credentials > + Add Credentials . In password place access token generated on docker hub.

New credentials

Kind

Username with password

Scope ?

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

Username ?

adrwalgd

☐ Treat username as secret ?

Password ?

.....

ID ?

dockerhub

Description ?

dockerhub

Create

Now we create new project (Dashboard > All > New item > Multibranch Pipeline) and configure source as Github repository and set it to discover only PRs.

Plain text [Preview](#)

Branch Sources

GitHub ✕
Credentials ?
adrwalGD/***** (Github Token) ▼
+ Add

☒ Repository HTTPS URL
Repository HTTPS URL ?

Credentials ok. Connected to https://github.com/adrwalGD/spring-petclinic. Validate

☐ Repository Scan - Deprecated Visualization

Behaviours
Discover branches ✕
? Strategy ?
 ▼

Because we will be running maven commands, we need to add maven to Jenkins

Dashboard > Manage Jenkins > Tools > Add Maven :

Maven installations

Add Maven

≡

Maven

Name

M3

☒ Install automatically ?

≡

Install from Apache

Version

3.9.9

Add Installer ▾

Creating actual pipeline logic

Below is pipeline to handle merge requests:

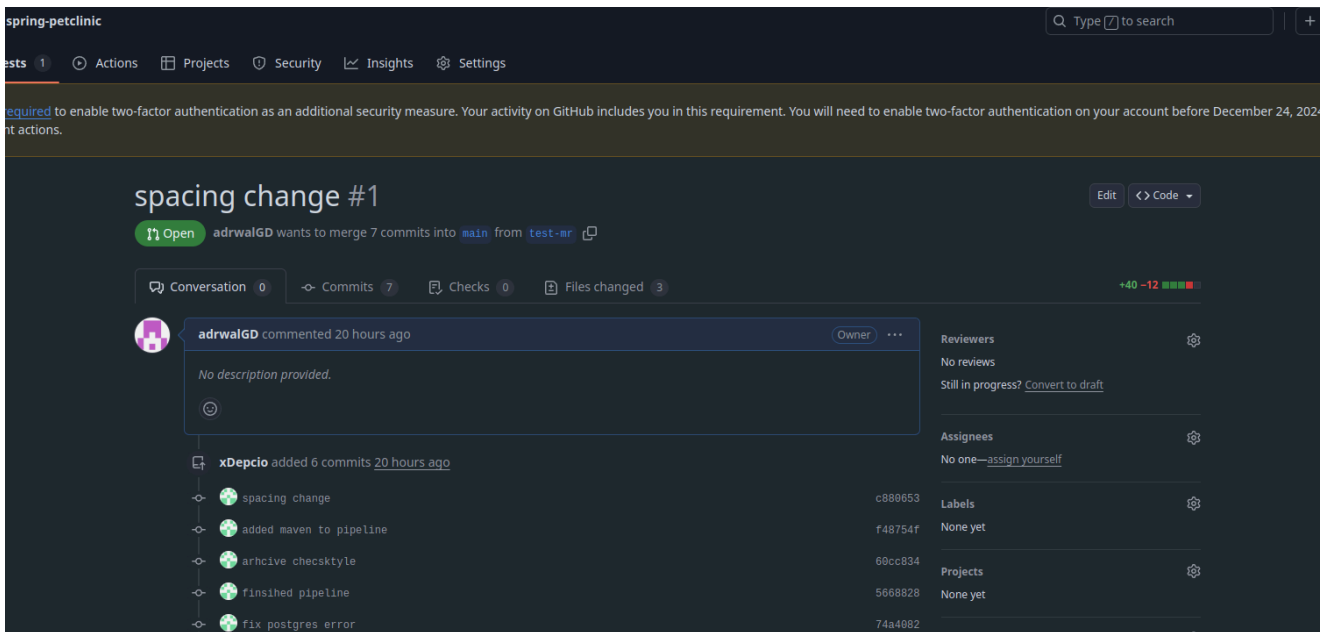
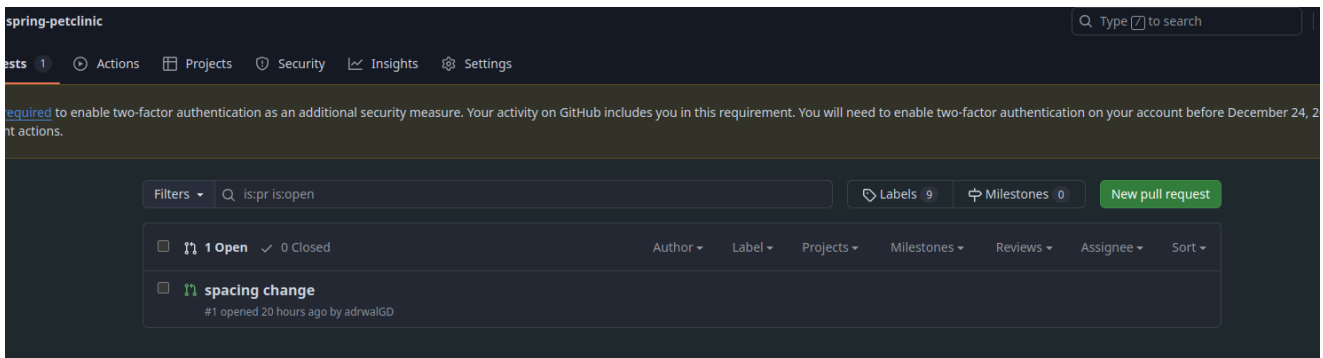
```

pipeline {
  agent {
    label 'docker'
  }
  tools {
    maven 'M3'
  }

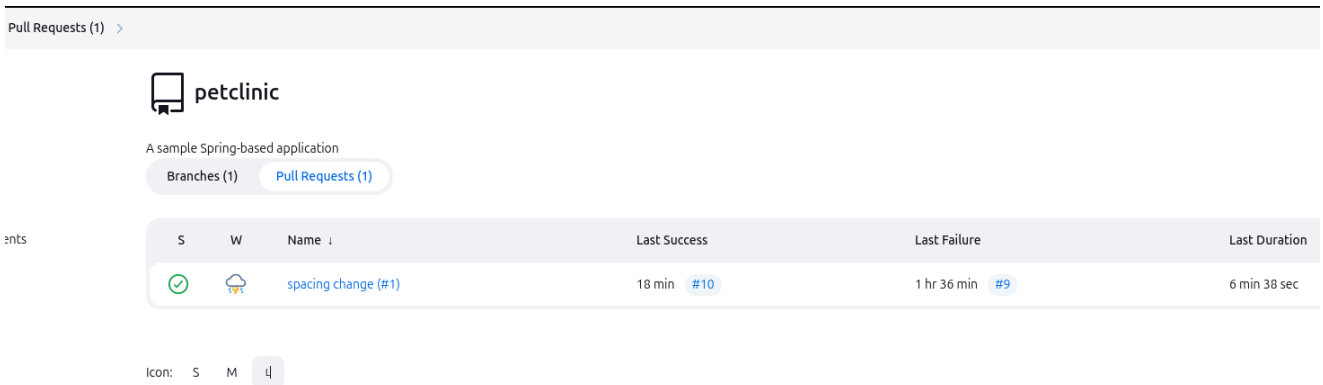
  stages {
    stage('Checkstyle') {
      steps {
        sh 'mvn checkstyle:checkstyle'
        sh 'tar -czf checkstyle-result.tar.gz target/reports'
        archiveArtifacts artifacts: 'checkstyle-result.tar.gz', onlyIfSuccessful: true
      }
    }
    stage('Tests') {
      steps {
        sh 'mvn test'
      }
    }
    stage('Build without tests') {
      steps {
        sh 'mvn package -DskipTests'
      }
    }
    stage('Build docker image') {
      steps {
        script {
          docker.build("adrwalgd/mr:$GIT_COMMIT")
          docker.withRegistry('https://registry.hub.docker.com', 'dockerhub') {
            docker.image("adrwalgd/mr:$GIT_COMMIT").push()
          }
        }
      }
    }
  }
}

```

To test that everything is working correctly I created new pull request on Github with this pipeline already on the main branch (as well as on PR branch).



Back on Jenkins after scanning the repository we can see that branch containing the pull request has been detected and ran.



Checkstyle artifact has been archived:

spacing change (#1)

Full project name: petclinic/PR-1



Last Successful Artifacts


 [checkstyle-result.tar.gz](#) 330.67 KiB  [view](#)


Permalinks


- [Last build \(#10\), 20 min ago](#)
- [Last stable build \(#10\), 20 min ago](#)
- [Last successful build \(#10\), 20 min ago](#)
- [Last failed build \(#9\), 1 hr 37 min ago](#)
- [Last unsuccessful build \(#9\), 1 hr 37 min ago](#)
- [Last completed build \(#10\), 20 min ago](#)

And job has finished with success and built image has been pushed to docker hub:

 Status


 Changes

 Console Output

 Edit Build Information

 Delete build '#10'

 Timings

 Git Build Data


 Pipeline Overview


 Pipeline Console

 Restart from Stage

 Replay

 Pipeline Steps

 Workspaces

 Previous Build

Console Output

Skipping 1,741 KB.. [Full Log](#)

```
#11 235.9 Progress (1): 262/334 kB
Progress (1): 266/334 kB
Progress (1): 270/334 kB
Progress (1): 274/334 kB
Progress (1): 278/334 kB
Progress (1): 282/334 kB
Progress (1): 286/334 kB
Progress (1): 290/334 kB
Progress (1): 294/334 kB
Progress (1): 298/334 kB
Progress (1): 303/334 kB
Progress (1): 307/334 kB
Progress (1): 311/334 kB
Progress (1): 315/334 kB
Progress (1): 319/334 kB
Progress (1): 323/334 kB
Progress (1): 327/334 kB
Progress (1): 331/334 kB
Progress (1): 334 kB
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/com/
#11 236.1 [INFO] Recompiling the module because of changed depende
#11 236.1 [INFO] Compiling 24 source files with javac [debug param
#11 236.8 [INFO]
#11 236.8 [INFO] --- maven-resources-plugin:3.3.1:testResources (d
```

```

[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline

```

Could not update commit status, please check if your scan selected

GitHub has been notified of this commit's build result

Finished: SUCCESS

The screenshot shows the Docker Hub interface for the repository 'adrwalgd'. The 'Tags' tab is selected, displaying a list of tags. The first tag is '7807c903c7124c6520701a75f7ea783338e164d5', which was last pushed 15 minutes ago by 'adrwalgd'. Below the tag name, a table lists the details for this tag:

Digest	OS/ARCH	Last pull	Compressed size
2588b3c122b6	linux/amd64	15 minutes ago	113.38 MB

At the top right of the tag list, there is a button that says 'docker pull adrwalgd/mr:7807c903c7124c6520701a75f7ea783338e164d5' with a 'Copy' icon next to it.

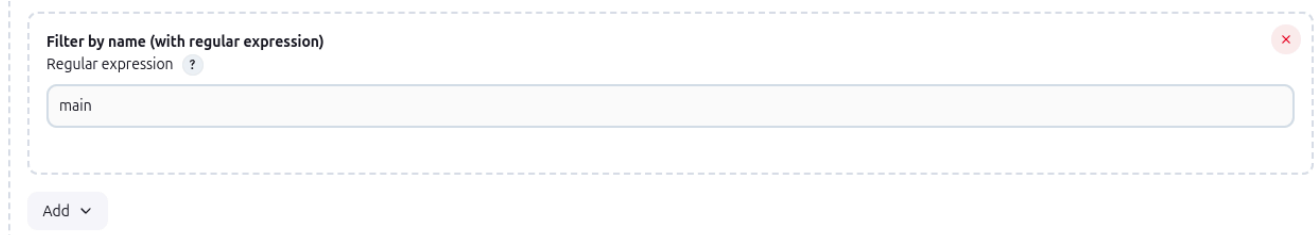
b. pipeline for main branch

To handle updates for the main branch correctly we need to:

1. Update project configuration in Jenkins to also detect main branch.
2. Update pipeline to run only Docker build and push stage on main branch.

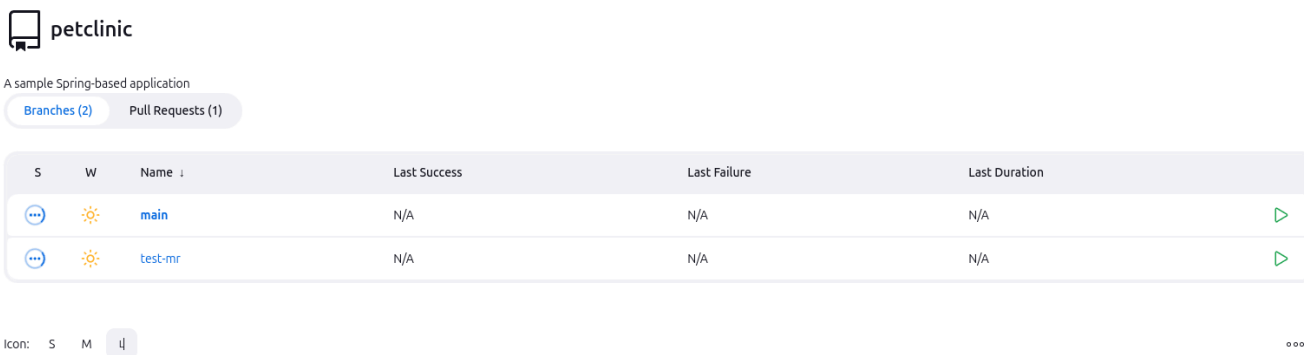
1. Update project configuration in Jenkins

In `Dashboard > [project name] > Configuration` we add new source with same credentials and repo URL, but different branch discovery strategy (only branch named main).



Filter by name (with regular expression)
Regular expression ?
main
Add ▾

After saving we can see that now also branch main has been detected:



petclinic
A sample Spring-based application
Branches (2) Pull Requests (1)

S	W	Name	Last Success	Last Failure	Last Duration
...	☀	main	N/A	N/A	N/A
...	☀	test-mr	N/A	N/A	N/A

Icon: S M ij

2. Update pipeline

I updated Jenkinsfile on main branch and rebased branch with PR on top of main so that now both of them have the same Jenkinsfile, but we can still see the differences between build steps on PR branch and main.

In updated Jenkinsfile we run one of two stages based on the branch name:

```
pipeline {  
    agent {  
        label 'docker'  
    }  
    tools {  
        maven 'M3'  
    }  
  
    stages {  
        stage('Merge Request') {  
            when {  
                expression {  
                    env.BRANCH_NAME != 'main'  
                }  
            }  
        }  
    }  
}
```

```

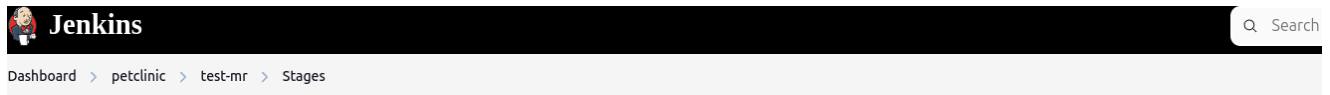
    }
  }
  stages {
    stage('Checkstyle') {
      steps {
        sh 'mvn checkstyle:checkstyle'
        sh 'tar -czf checkstyle-result.tar.gz
target/reports'
        archiveArtifacts artifacts: 'checkstyle-
result.tar.gz', onlyIfSuccessful: true
      }
    }
    stage('Tests') {
      steps {
        sh 'mvn test'
      }
    }
    stage('Build without tests') {
      steps {
        sh 'mvn package -DskipTests'
      }
    }
    stage('Build docker image') {
      steps {
        script {
          docker.build("adrwalgd/mr:$GIT_COMMIT")
        }
      }
    }
  }
  docker.withRegistry('https://registry.hub.docker.com', 'dockerhub') {
    docker.image("adrwalgd/mr:$GIT_COMMIT").push()
  }
}

stage('main change') {
  when {
    branch 'main'
  }
  stages {
    stage('Build and push Docker Image') {
      steps {
        script {
          docker.build("adrwalgd/main:$GIT_COMMIT")
        }
      }
    }
  }
  docker.withRegistry('https://registry.hub.docker.com', 'dockerhub') {
    docker.image("adrwalgd/main:$GIT_COMMIT").push()
  }
}

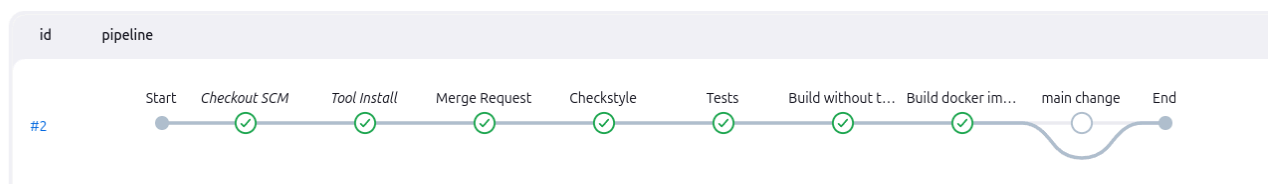
```



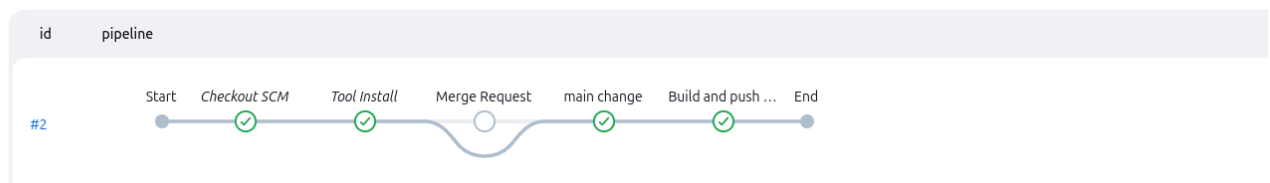
After running the build in Jenkins we can validate that correct stages were run for specific branches:



Build test-mr



Build main



And that Docker images has been pushed to mr and main repository:

adrwalgd / [Repositories](#) / [mr](#) / [Tags](#)

General

Tags

Builds

Collaborators

Webhooks

Settings

☐

Sort by

Newest

Delete

☐

TAG

e0063074c8a2b0449179ae17262eec08cd6411f5

Last pushed 19 hours ago by [adrwalgd](#)

Digest	OS/ARCH	Last pull
c5cc72e6858d	linux/amd64	---

☐

TAG

7807c903c7124c6520701a75f7ea783338e164d5

Last pushed 19 hours ago by [adrwalgd](#)

Digest	OS/ARCH	Last pull
2588b3c122b6	linux/amd64	20 hours ago

adrwalgd / [Repositories](#) / [main](#) / [Tags](#)

General

Tags

Builds

Collaborators

Webhooks

Settings

☐

Sort by

Newest

Delete

☐

TAG

ffc3df21d58d78b892515062c9d207facee357ae

Last pushed 19 hours ago by [adrwalgd](#)

Digest	OS/ARCH	Last pull
f103ea66e5b3	linux/amd64	19 hours ago