



Jenkins

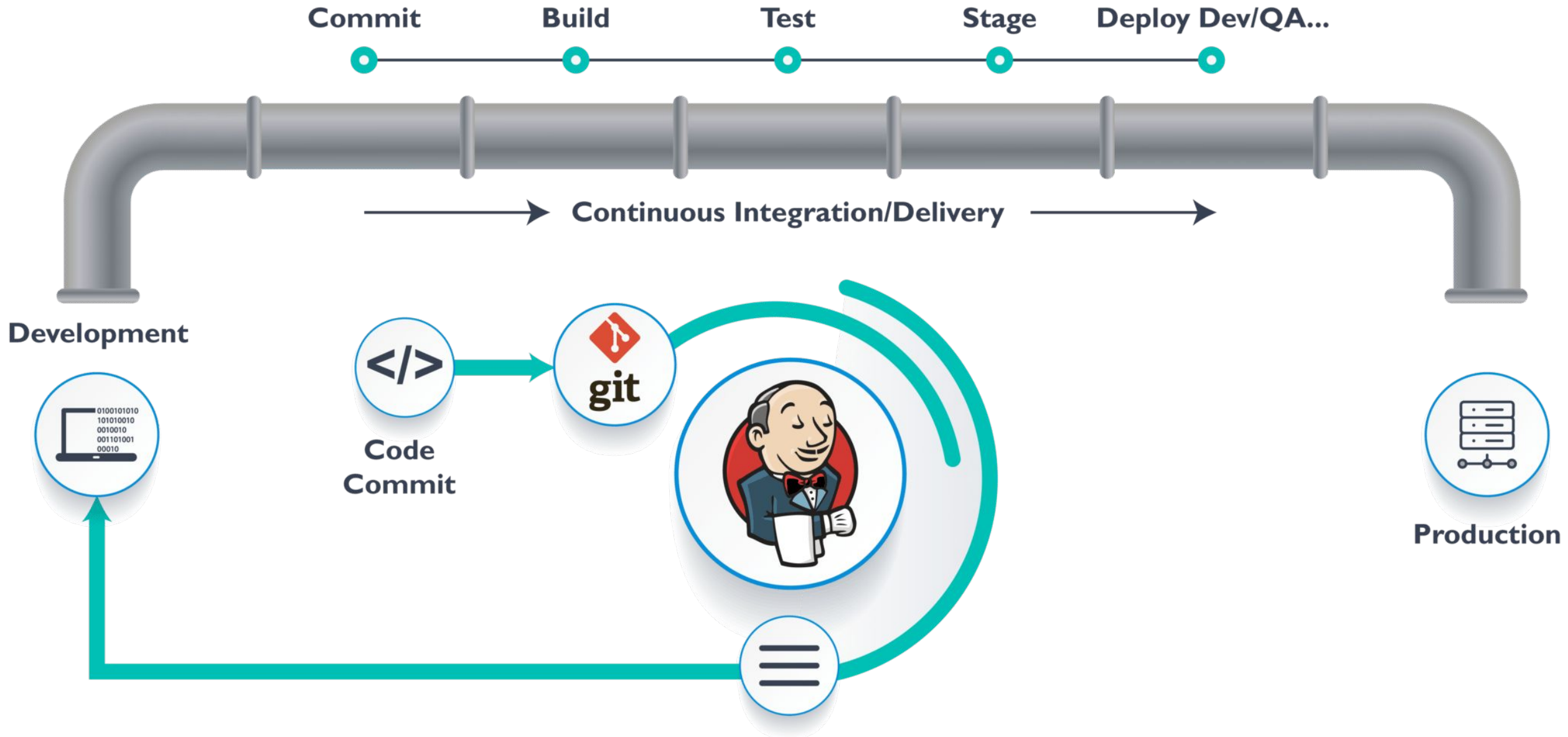
Dariusz Grabowski



Czym jest Jenkins?

- narzędzie oparte o technologię JAVA do automatyzacji procesów związanych z wytwarzaniem oprogramowania
- Za pomocą strumieni CI/CD umożliwia:
 - pobieranie źródeł z repozytoriów
 - budowanie aplikacji
 - uruchamianie i przetwarzanie testów funkcjonalnych
 - uruchamianie statycznej analizy kodu
 - tworzenie paczek i umieszczanie ich w repozytoriach
 - wdrażanie aplikacji do środowiska produkcyjnego

Jenkins



Which Continuous Integration systems do you regularly use? (%)

Jenkins / Hudson



Travis CI



Gitlab CI



TeamCity



CircleCI



Microsoft TFS / VSTS



Bamboo



Microsoft Team
Foundation Build



GoCD



AppVeyor



Codeship



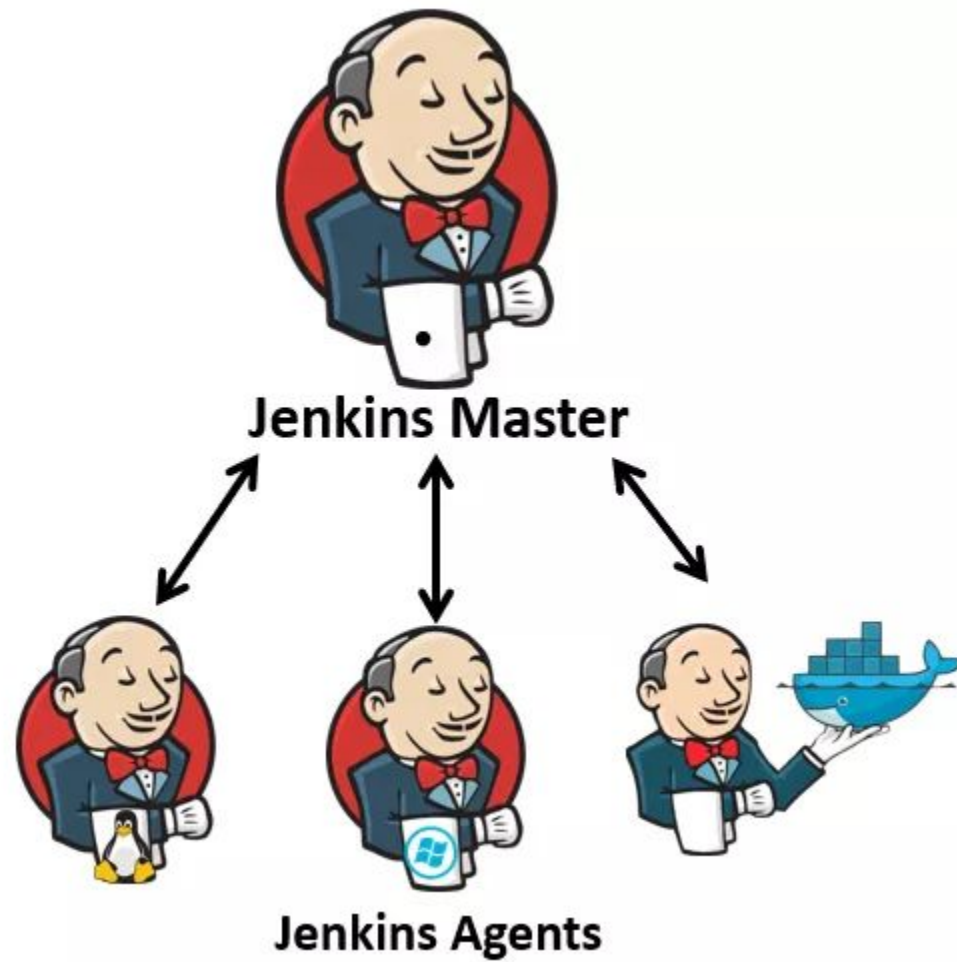
Drone



Other



Jenkins



Jenkins

The screenshot shows the Jenkins dashboard in a web browser. The browser's address bar displays 'localhost:8080'. The Jenkins header includes the logo, a red notification badge with the number '3', a search bar, and links for 'admin' and 'log out'. A sidebar on the left contains navigation links: 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'My Views', 'Open Blue Ocean', 'Lockable Resources', 'Credentials', and 'New View'. The main content area features a table of build components under the 'All' tab. The table has columns for status (S), weather icon (W), name, last success/failure times, duration, favorite status (Fav), and the number of issues. The components listed are bin1, bin2, depMgr, libA, libB, and libC. At the bottom, there are links for 'Icon: S M L' and three RSS feeds: 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

S	W	Name	Last Success	Last Failure	Last Duration	Fav	# Issues
		bin1	4 mo 24 days - log	N/A	0.18 sec		-
		bin2	4 mo 24 days - log	N/A	0.26 sec		-
		depMgr	4 mo 24 days - #36	4 mo 24 days - #34	2 min 25 sec		-
		libA	4 mo 24 days - log	N/A	0.21 sec		-
		libB	4 mo 24 days - log	N/A	0.1 sec		-
		libC	4 mo 24 days - log	N/A	0.14 sec		-

Icon: [S](#) [M](#) [L](#)


[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

- Job - obiekt w jenkinsie odpowiadający za budowanie komponentu
- Build - jedno uruchomienie/wykonanie obiektu typu job
- Stage - jeden z kroków w trakcie wykonywania joba
- Node - instancja jenkins slave





Tworzenie joba


Jenkins - tworzenie joba


 **Jenkins** 3 [?](#) [admin](#) | [log out](#)


Jenkins [ENABLE AUTO REFRESH](#)


 **New Item** [add description](#)


 People


 Build History


 Project Relationship


 Check File Fingerprint


 Manage Jenkins

 My Views




















 Open Blue Ocean

 Lockable Resources




 Credentials

 New View

[All](#) [AllComponents](#) [+](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav	# Issues
		bin1	4 mo 24 days - log	N/A	0.18 sec		-
		bin2	4 mo 24 days - log	N/A	0.26 sec		-
		depMgr	4 mo 24 days - #36	4 mo 24 days - #34	2 min 25 sec	 	-
		libA	4 mo 24 days - log	N/A	0.21 sec		-
		libB	4 mo 24 days - log	N/A	0.1 sec		-
		libC	4 mo 24 days - log	N/A	0.14 sec		-

Icon: [S](#) [M](#) [L](#)


[Legend](#)  [RSS for all](#)  [RSS for failures](#)  [RSS for just latest builds](#)

Jenkins - tworzenie joba


Jenkins ▸ All ▸

Enter an item name


» Required field




Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.




Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

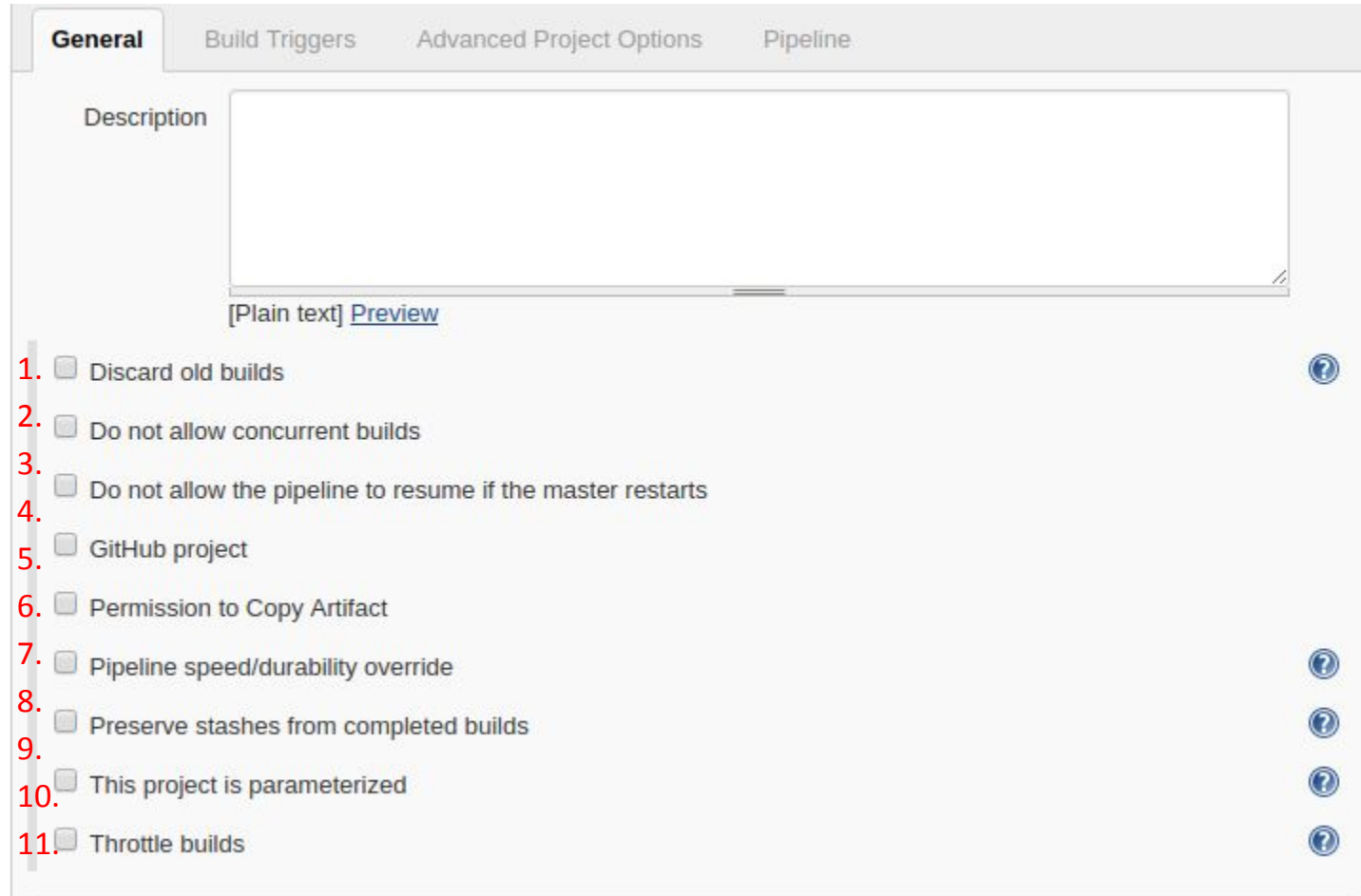


External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.



Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple machines.

Jenkins - konfiguracja



The screenshot shows the Jenkins configuration interface for a project, specifically the 'General' tab. The interface includes a 'Description' text area, a '[Plain text] Preview' link, and a list of 11 configuration options, each with an unchecked checkbox and a help icon. The options are numbered 1 through 11 on the left side of the image.

General Build Triggers Advanced Project Options Pipeline

Description

[Plain text] [Preview](#)

- ☐ Discard old builds
- ☐ Do not allow concurrent builds
- ☐ Do not allow the pipeline to resume if the master restarts
- ☐ GitHub project
- ☐ Permission to Copy Artifact
- ☐ Pipeline speed/durability override
- ☐ Preserve stashes from completed builds
- ☐ This project is parameterized
- ☐ Throttle builds

Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Enable Artifactory trigger
- ☐ GitHub hook trigger for GITScm polling
- ☐ Poll SCM
- ☐ Disable this project
- ☐ Quiet period
- ☐ Trigger builds remotely (e.g., from scripts)



Jenkins - konfiguracja

Jenkins > myFirstJob >

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition


Pipeline script
Pipeline script
Pipeline script from SCM

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

Save Apply


Jenkins - konfiguracja


**Jenkins**


3


[?](#) [admin](#) | [log out](#)


Jenkins > myFirstJob > [ENABLE AUTO REFRESH](#)


 [Back to Dashboard](#)


 [Status](#)


 [Changes](#)


 [Build Now](#)


 [Delete Pipeline](#)


 [Configure](#)



 [Full Stage View](#)

 [Open Blue Ocean](#)


 [Rename](#)

 [Pipeline Syntax](#)


 **Build History** [trend](#)

 [RSS for all](#)  [RSS for failures](#)

Pipeline myFirstJob

 [add description](#)

[Disable Project](#)

 [Recent Changes](#)

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Page generated: Aug 27, 2019 11:36:22 AM UTC [REST API](#) [Jenkins ver. 2.176.2](#)

Jenkinsfile

```
node('MyBuilderNode') { // uruchomienie joba na nodzie o takiej nazwie
    def mvnHome
    stage('Preparation') { // wyświetlana nazwa stage'a
        // pobranie kodu z repozytorium
        git 'https://github.com/jglick/simple-maven-project-with-tests.git'
        // Użycie skonfigurowanego wcześniej narzędzia mvn
        mvnHome = tool 'M3'
    }
    stage('Build') {
        // Uruchomienie budowania
        // "sh" - wykonanie polecenia w shellu
        if (isUnix()) {
            sh "'${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore clean package"
        } else {
            bat("/"${mvnHome}\bin\mvn" -Dmaven.test.failure.ignore clean package/ )
        }
    }
    stage('Results') {
        // zebranie wyników testów w celu ich przetworzenia
        junit '**/target/surefire-reports/TEST-*.xml'
        // archiwizacja artefaktów
        archive 'target/*.jar'
    }
}
```


Jenkins pipeline

The screenshot displays the Jenkins web interface for a job named 'myFirstJob 1'. The browser address bar shows 'localhost:8080/blue/organizations/jenkins/myFirstJob/detail/myFirstJob/1/...'. The job's status is 'Success', indicated by a green checkmark. The pipeline consists of five stages: 'Start', 'Preparation', 'Build', 'Results', and 'End'. All stages are marked with green checkmarks, signifying a successful execution. The 'Results' stage is currently selected and highlighted with a blue circle. Below the pipeline diagram, the 'Results' section shows a log entry: 'Results - <1s' with a green checkmark icon, followed by a collapsed log entry '> Results - Print Message' with a duration of '<1s'. The interface includes a top navigation bar with tabs for 'Pipeline', 'Changes', 'Tests', and 'Artifacts', along with a 'Logout' button.

jenkins / myFirstJob / #1

localhost:8080/blue/organizations/jenkins/myFirstJob/detail/myFirstJob/1/...

✓ myFirstJob 1

Pipeline Changes Tests Artifacts

Branch: — 4s No changes

Commit: — a minute ago Started by user admin

Start Preparation Build Results End

Results - <1s

> Results — Print Message <1s

Multibranch pipeline

The screenshot shows the Jenkins web interface for a Multibranch Pipeline named 'bin1'. The left sidebar contains navigation links: Up, Status, Configure, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, People, Build History, Project Relationship, Check File Fingerprint, Open Blue Ocean, Rename, Config Files, Pipeline Syntax, and Credentials. The main content area displays the 'bin1' pipeline details, including a table of branches and a build queue at the bottom.

Jenkins 3 search admin | log out

Jenkins > bin1 > [ENABLE AUTO REFRESH](#)

bin1

Branches (2)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav	# Issues
		feature-01	4 mo 24 days - #5	4 mo 24 days - #1	30 sec		-
		trunk	4 mo 24 days - #10	4 mo 24 days - #8	1 min 48 sec		-

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Build Queue [-](#)



Jenkinsfile syntax

```
node {  
  dir('myDir') { // jeśli nie istnieje - zostanie utworzony  
    echo "wykonaj operacje w katalogu myDir"  
    if(fileExists('file.txt')) {  
      echo "file.txt istnieje w katalogu myDir"  
      sleep(5) // czekaj 5 sek  
    }  
  }  
  timeout(time: 20, unit: 'SECONDS') {  
    echo "Ta operacja może trwać maksymalnie 20 sec"  
  }  
  waitUntil {  
    echo "Ta operacja będzie się powtarzać tak długo aż zwróci kod zerowy"  
    echo "Można wykorzystać do omijania błędów związanych z dostępem do  
sprzętu"  
  }  
  withEnv(['LD_LIBRARY_PATH=/some/directory']) {  
    echo "Wykonaj polecenie ze zmienną $LD_LIBRARY_PATH"  
  }  
}
```

```
node {  
  sh './set-up.sh'  
  try {  
    sh 'might fail'  
    echo 'Succeeded!'  
  } catch (err) {  
    echo "Failed: ${err}"  
  } finally {  
    sh './tear-down.sh'  
  }  
  echo 'Printed whether above succeeded or failed.'  
}
```

```
node {  
  stage('Build app') {  
    node('builderNode') {  
      sh 'g++ main.cpp'  
      stash name: 'myapp', includes: 'a.out'  
    }  
  }  
  stage('Run app') {  
    node('runEnvironment') {  
      unstash 'myapp'  
      sh './a.out'  
    }  
  }  
}
```

```
node {  
  parallel(  
    gcc: {  
      node('gccCompiler') {  
        sh 'g++ main.cpp'  
      }  
    },  
    clang: {  
      node('clangCompiler') {  
        sh 'clang++ main.cpp'  
      }  
    }  
  )  
}
```

Lockable resources

```
node {  
  lock('my-embedded-device') {  
    echo "Używam urządzenia na wyłączność"  
    echo "Inne joby czekają na jego zwolnienie"  
  }  
  build 'inny-job'  
}
```

Lockable Resources

Resource	Status	Labels	Action
clearcase-scm access to the scm	LOCKED by P.test.1 #261	scm	<button>Unlock</button>
cantatapp license	FREE	licenses	<button>Reserve</button>
rhapsody rhapsody generation	FREE	licenses	<button>Reserve</button>
J-Link SN=238003806	FREE	J-Link	<button>Reserve</button>

Labels

Label	Free resources
licenses	2
J-Link	1
scm	0


```
node {  
  docker.image('ubuntu:19.10').inside {  
    echo "Uruchamiam polecenie w kontenerze"  
  }  
  
  docker.image('ubuntu:19.10').inside("-p 80:80") {  
    echo "Kontener uruchomiony z wystawionym portem 80"  
  }  
  
  def mojObraz = docker.build("moj-obraz", "katalog/z/dockerfile")  
  mojObraz.inside {  
    echo "Polecenie w kontenerze ze zbudowanego obrazu"  
  }  
}
```

Artifactory

```
node {
  stage('build') {
    def server = Artifactory.server 'MY_ARTIFACTORY'
    def client = Artifactory.newConanClient userHome: "${env.WORKSPACE}".toString() +
"/moj/conan/home"
    def serverName = client.remote.add server:server, repo: 'nazwa-repozytorium-conana'
    dir('build') {
      client.run(command: "install ..")
      client.run(command: "build ..")
    }
    //Tworzenie paczki
    client.run(command: "create . ci/training")
  }

  stage('Publish') {
    // upload paczki do artifactory
    String command = "upload * --all -r ${serverName} --confirm"
    def buildInfo = client.run(command: command)
    buildInfo.env.collect()
    // publikowanie ustawień środowiska do artifactory
    server.publishBuildInfo buildInfo
  }
}
```

Shared library

```
(root)
+- src                                # Struktura biblioteki pipeline
|   +- org
|       +- foo
|           +- Bar.groovy # klasa org.foo.Bar
+- vars
|   +- foo.groovy         # globalne zmienne
|   +- foo.txt            # i ich pliki pomocnicze
+- resources              # Pliki konfiguracyjne json/yml/txt
|   +- org
|       +- foo
|           +- bar.json    # static helper data for org.foo.Bar
```

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use @Grab.

Library

Name

my-shared-library

?

Default version

master

?

Load implicitly

☐

?

Allow default version to be overridden

☒

?

Retrieval method

☒ Modern SCM

?

Shared library

```
// Przykład pliku biblioteki pipeline
// src/org/foo/Zot.groovy
package org.foo

def checkOutFrom(repo) {
    git url: "git@github.com:jenkinsci/${repo}"
}

return this
```

```
// Importowanie
@Library('somelib')
import org.foo.Zoo()

// użycie w skrypcie pipeline
def z = new org.foo.Zot()
z.checkOutFrom(repo)
```



Statyczna analiza kodu

Record issues plugin

Steps

Sample Step recordIssues: Record compiler warnings and static analysis results ▼

Static Analysis Tools

Tool CPD ⓘ

Report File Pattern

[Fileset 'includes'](#) setting that specifies the report files to scan for issues, such as 'myproject/target/checkstyle-results.xml'. If you leave this field empty, then the default file pattern '**/cpd.xml' will be used.

Skip Symbolic Links ☐

Toggle whether the file scanner will skip symbolic links. This is useful when the scanned directory contains links that create a recursive structure (may not work on Windows).

Report Encoding ⓘ

Encoding of your report files.

High severity threshold

Minimum number of duplicated lines for high severity warnings.

Normal severity threshold

Minimum number of duplicated lines for normal severity warnings.

Custom ID

Optional custom ID (URL) of this tool, overwrites the built-in ID 'cpd'.

Custom Name

Optional custom display name of the tool, overwrites the built-in name 'CPD'.

```
# cpplint - sprawdza zgodność z Google c++ style guide
```

```
$ pip install cpplint
```

```
$ cpplint --output=vs7 \
```

```
`find -name "*.cpp" -o -name "*.h"` > cpplint_results.xml
```

```
# -----
```

```
# cppcheck - sprawdza nie tylko styl ale też błędy funkcjonalne
```

```
$ sudo apt-get install cppcheck
```

```
$ cppcheck --enable=all --language=c++ --xml sourcesDir 2>
```

```
cppcheck_results.xml
```

PMD - duplikacja kodu

```
$ wget  
https://github.com/pmd/pmd/releases/download/pmd_releases%2F6.17.0/pmd-bin-  
6.17.0.zip  
$ unzip pmd-bin-6.17.0.zip  
$ alias cpd="`pwd`pmd-bin-6.17.0/bin/run.sh cpd"  
  
# użycie  
$ cpd --minimum-tokens 100 --files /path/to/cpp/source \  
    --language cpp \  
    --format xml
```


PMD - duplikacja kodu

Overview

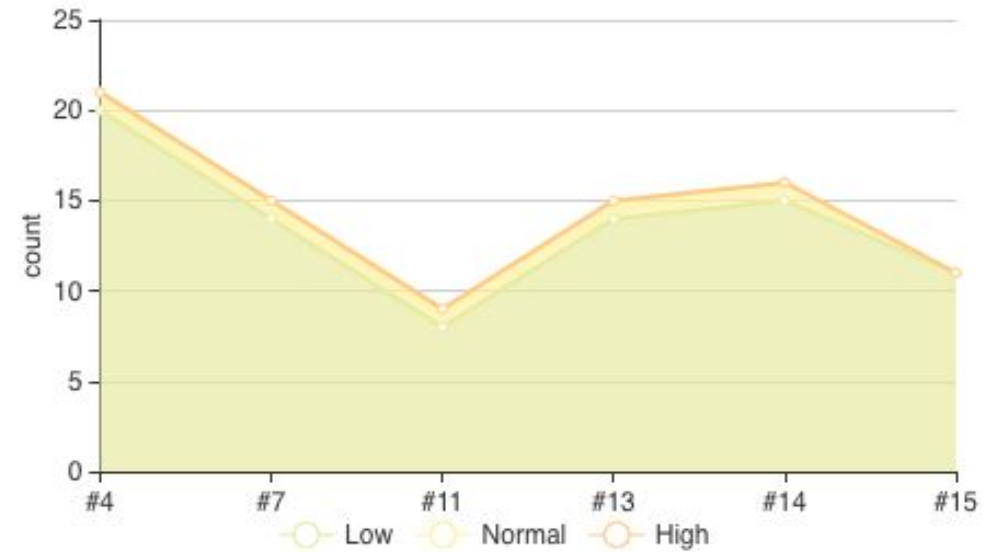


Reference Comparison



New Outstanding Fixed

History



valgrind

```
#CMakeLists.txt
find_program(MEMORYCHECK_COMMAND $ENV{VALGRIND_BINARY_PATH})
set(MEMORYCHECK_COMMAND_OPTIONS "--trace-children=yes \
    --leak-check=full --child-silent-after-fork=yes \
    --xml=yes --xml-file=memcheck_test_%p.memcheck" )
```

```
# Conan profile:
(...)
[env]
VALGRIND_BINARY_PATH=/
usr/bin/valgrind
```

```
node { // Jenkinsfile
    stage('Test') {
        sh 'ctest -D ExperimentalMemCheck'
        sh 'grep --files-with-matches -r -Pzo
"status\\>\\n\\n\\<errorcounts" `find -name memcheck*` >
toDelete.txt'
        sh 'rm `cat toDelete.txt` toDelete.txt'
    }
    stage('Publish report') {
        publishValgrind (pattern: '**/*.memcheck')
    }
}
```



Pytania?

Zadanie

1. Wrzucić kod z poprzedniego zadania na prywatne repozytorium
2. Uruchomić środowisko jenkins + artifactory (katalog Pobrane/ci_env)
 - a. `docker-compose up`
3. Dodać usera: admin/passwd
4. Stworzyć joba pipeline w którym należy:
 - a. pobrać kod z repozytorium
 - b. zbudować aplikację z użyciem conana
 - c. wysłać paczkę do artifactory
 - d. * zrobić to samo ale w kontenerze dockerowym