

Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
Maven
Testing (Foundation)
Java Intermediate
HTML
CSS
Javascript
Spring Boot
Selenium
Sonarqube
Advanced Testing (Theory)
Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps

Jenkins Agents


Contents

- [Overview](#)
 - [What is an agent](#)
 - [Why use Agents](#)
- [Tutorial](#)
 - [Prerequisites](#)
 - [Step 1: SSH Keys](#)
 - [Step 2: Give Jenkins the keys](#)
 - [Step 4: Create Your Docker Agent](#)
 - [Starting your agent](#)
 - [Connecting to your agent](#)
 - [Updating your agent](#)
 - [Step 5: Register The Agent With Jenkins](#)
 - [Step 6: Set up a Pipeline Project](#)
 - [Step 8: CI/CD](#)
- [Conclusion](#)
- [Exercises](#)
 - [Automation](#)

Overview

In this module, we will look at Jenkins Agents: what they are, what they do, and how you can configure this to your advantage.

Out of the box, you will see that Jenkins has a **master** Build Executor which can run two jobs concurrently and are **Idle**.

<u>Build Executor Status</u>		^
 master		
1	Idle	
2	Idle	

What is an agent

An agent, or Node, is a distinct unit, such as a *Virtual Machine*, that is running the *Jenkins Agent* and capable of receiving *jobs* from Jenkins, whether those be stages of a Pipeline, Freestyle Project, or the entire build process. Agents, however, are only capable of working on a single job at a time until the job completes with success or failure.

Why use Agents

Agents are a ubiquitous pattern in computing which help to separate the "Job Issuer" (Jenkins) from the "Job Runner" (Agents). In doing so, we are granted asymmetry - we can have as many Agents as we want per Jenkins instance. This means that we can dedicate certain agents to certain jobs, for instance, a **build** agent and a **deploy** agent.

Typically, deployment is a much quicker process than building artefacts, so it would make sense to have multiple build agents - this would mean we could, for example, build separate images for **integration**, **staging** and **production** at the

Jenkins Introduction
Jenkins Pipeline
<div><div></div>Jenkins Agents</div> <div><div></div>Pipeline</div> <div><div></div>Pipeline Snippet Generator</div> <div><div></div>Credentials</div>
Markdown
IDE Cheatsheet

same time, using three different build agents, while leaving the deploy agent ready to deploy the artefacts as soon as they are ready.

Tutorial

In this tutorial, we are going to set up an agent that we can selectively send jobs to using a label.

By doing so we will have two agents:

- the default agent that we are going to task as our **Deploy** agent, and
- a new containerised agent that we are going to task as our **Build** agent

Prerequisites

For this tutorial, you will need a virtual machine (ubuntu 20.04) with Jenkins installed, an admin user configured, and the default plugins installed (at least **SSH Build Agents** plugin).

You will also need to have **Docker** installed on this VM - follow the directions here: <https://docs.docker.com/engine/install/ubuntu/>

Jenkins very helpfully provide a Docker Image for their agent:

<https://hub.docker.com/r/jenkins/ssh-agent>

N.B. It's ALWAYS worth checking for official docker images

Finally, we are going to build a Hello World Java program in order to illustrate how agents might be useful. You do not need to have a background in Java.

Step 1: SSH Keys

In order to authenticate between Jenkins and the Agent we will need to provide some keys. We're going to be doing this via **ssh**, so simply generate yourself a key pair with:

```
ssh-keygen -f ~/.ssh/jenkins_agent_key
```

There is no need to add a passphrase, and the defaults are perfectly good for our needs.

Step 2: Give Jenkins the keys

Now we need Jenkins to be aware of the keys we just generated, so go to your Jenkins Dashboard and click the **Manage Jenkins** option in the left-hand menu and then the **Manage Credentials** button



Dashboard

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

Manage Jenkins

System Configuration

Configure System

Global Tool Configuration

Manage Plugins

Manage Nodes and Clouds

Security

Configure Global Security

Manage Credentials

Configure Credential Providers

Manage Users

Select the **Add Credentials** option from the dropdown on the **global** item.

Dashboard

Credentials

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

Credentials

T	P	Store	Domain	ID	Name
		Jenkins	(global)	jenkins	jenkins (jenkins agent)

Icon: S M L

Add credentials

Stores scoped to Jenkins

P	Store	Domains
	Jenkins	(global)

Fill in the Form:

Kind: SSH Username with private key;

id: jenkins

description: The jenkins agent ssh key

username: jenkins

Private Key: select Enter directly and press the Add button to insert your private key from ~/.ssh/jenkins_agent_key

Passphrase: <empty>

Dashboard

Credentials

System

Global credentials (unrestricted)

Back to credential domains

Add Credentials

Kind

SSH Username with private key

Scope

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

ID

Description

Username

Private Key

Enter directly

OK

To get your private key simply copy the output from:

```
cat ~/.ssh/jenkins_agent_key
```

Step 4: Create Your Docker Agent

Starting your agent

You will need your corresponding ssh public key to the credentials you just provided to Jenkins.

To get your public key simply copy the output from:

```
cat ~/.ssh/jenkins_agent_key.pub
```

```
docker run -d -p 2222:22 jenkins/ssh-agent:latest "<YOUR PUBLIC KEY>"
```

This will download the docker image provided by the Jenkins devs, push your ssh key into it, and forward the ports from 22 on the container to 2222 on your host machine. This last step is very important as ports must be unique and so we cant have a container listening on port 22 when the host is also listening on that port.

The container will also start in a detached state (-d).

To confirm that the container is in fact running you can list the docker processes with `docker ps`.

Connecting to your agent

At this point it is worth testing that everything is working as expected, which we can do via ssh:

```
ssh jenkins@localhost -p 2222 -i .ssh/jenkins_agent_key
```

This should connect you to the container, or throw an error such as `permission denied`. This is most likely due to the permissions on the key you are passing: keys being rather important to security, it is forbidden to use a key that anyone else might be able to edit. This is a common issue, especially after using `ssh-keygen`, but it has a simple solution:

```
chmod 0600 .ssh/jenkins_agent_key
```

This will make sure that only your user has access to the key.

Once we have validated that ssh can connect, we can simply `exit` out of the container.

Updating your agent

There is an issue still open for the container that requires a manual fix (Jenkins devs are working on it though so this will be patched at some point in the future):

Connect to your instance and become root via docker exec:

```
docker exec -it <NAME||ID> /bin/bash
```

This will interactively (-i) execute (`exec`) the `/bin/bash` command (the bash shell) within the container provided (your agent container), as either the name or id (which can be just the first few characters) as shown by a `docker ps`.

Once that executes you will be bumped to the bash process within the jenkins agent container, where you should paste the following and press enter:

```
VAR1="HOME=|USER=|MAIL=|LC_ALL=|LS_COLORS=|LANG="
VAR2="HOSTNAME=|PWD=|TERM=|SHLVL=|LANGUAGE=|_="
VAR="${VAR1}|${VAR2}"
env | egrep -v "^(${VAR})" >> /etc/environment
```

Once that completes you can `exit` out of the container back to your host.

Step 5: Register The Agent With Jenkins

Now that we have a running agent on our host, we need to register this with Jenkins so that it can push jobs to it.

Dashboard

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

Build Queue

No builds in the queue.

Build Executor Status

master

1 Idle

Manage Jenkins

System Configuration

Configure System

Configure global settings and paths.

Global Tool Configuration

Configure tools, their locations and automatic installers.

Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

Configure Global Security

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

Manage Credentials

Configure credentials

Configure Credential Providers

Configure the credential providers and types

Manage Users

Create/delete/modify users that can log in to this Jenkins

Once again, go to the Jenkins Dashboard and select the **Manage Jenkins** option in the left-hand menu before going to the **Manage Nodes and Clouds** section.

Dashboard

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

New View

Manage Jenkins

System Configuration

Configure System

Configure global settings and paths.

Global Tool Configuration

Configure tools, their locations and automatic installers.

Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

Go to the **New Node** option in the left-hand menu and fill in the Node/Agent Name **agent1** and select the type **Permanent Agent**.

Dashboard

Nodes

Back to Dashboard

Manage Jenkins

New Node


Configure Clouds

Node Monitoring

Fill in the Form:

Remote root directory:	/home/jenkins
Label:	agent1
Usage:	only build jobs with label expression
Launch method:	Launch agents by SSH
Host:	localhost
Credentials:	jenkins (e.g. the credentials we registered in Step 2)
Verification Strategy:	Manually trusted key verification

Click on the **agent1**, select **Launch agent** and wait a moment for Jenkins to connect to the container and make the agent available. You should receive a confirmation message: **Agent successfully connected and online**



Agent agent1 (agent1)

Mark this node temporarily offline

Launch agent

Projects tied to agent1

S	W	Name ↓	Last Success	Last Failure	Last Duration
		Build	58 min - #26	4 hr 51 min - #22	1.7 sec

Icon: S M L

Legend

Atom feed for all

Atom feed for failures

Atom feed for just latest builds

DashboardNodesagent1

UID=1000
USER=jenkins
_=""'
[01/06/21 15:58:43] [SSH] Checking java version of /home/jenkins/jdk/bin/java
Couldn't figure out the Java version of /home/jenkins/jdk/bin/java
bash: /home/jenkins/jdk/bin/java: No such file or directory

[01/06/21 15:58:43] [SSH] Checking java version of java
[01/06/21 15:58:43] [SSH] java -version returned 1.8.0_275.
[01/06/21 15:58:43] [SSH] Starting sftp client.
[01/06/21 15:58:43] [SSH] Copying latest remoting.jar...
Source agent hash is 79A835B446D70E09D00436F87F21CD7C. Installed agent hash is 79A835B446D70E09D00436F87F21CD7C
Verified agent jar. No update is necessary.
Expanded the channel window size to 4MB
[01/06/21 15:58:43] [SSH] Starting agent process: cd "/home/jenkins" && java -jar remoting.jar -workDir /home/jenkins
/home/jenkins/remoting/jarCache
Jan 06, 2021 3:58:44 PM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /home/jenkins/remoting as a remoting work directory
Jan 06, 2021 3:58:44 PM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /home/jenkins/remoting
<---[JENKINS REMOTING CAPACITY]--->channel started
Remoting version: 4.6
This is a Unix agent
Evacuated stdout
Agent successfully connected and online

Now we can push jobs to our agent!.

Step 6: Set up a Pipeline Project

We are going to build a Pipeline project. The project will be very simple, using only two stages:

- Build** : during this stage we are going compile and build an artefact
- Deploy** : during this stage we will take the artefact and deploy/run it

The example we are going to use is a Hello World Java class:

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello world!");  
    }  
}
```

Set up your own GitHub repo along the lines of <https://github.com/qasbirchall/java-hello-world>


HINT: GitHub has a fork button.


Set up a new Pipeline Project called **java-app** with your repository as the **GitHub project**


Enter an item name


java-app


» 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.

 **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.

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

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

OK

GeneralBuild TriggersAdvanced Project OptionsPipeline

Description

Simple Pipeline to Build and Deploy a Hello World.java application

[Plain text] Preview

☐ Discard old builds

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the master restarts

☒ GitHub project

Project url

https://github.com/qasbirchall/java-hello-world/

Advanced...

Now we want to configure a pipeline with:

- a **Build** stage: this will compile our Java class, build a Jar artefact and archive it for permanent use.
- a **Deploy** stage: that will execute the successfully built artefact. In reality this stage would most likely push the Jar out to the various Tomcat servers running in our environments, but for the purposes of this tutorial we will keep things simple.

The Pipeline can be very neatly configured using:

```
pipeline {
    agent any
    options {
        skipStagesAfterUnstable()
    }
    stages {
        stage('Build') {
            agent { label 'agent1' }
            steps {
                echo '----- BUILD -----'
                checkout([$class: 'GitSCM', branches: [[name: '*/main']],
doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],
userRemoteConfigs: [[url: 'https://github.com/qasbirchall/java-hello-world']]])
                sh 'javac HelloWorld.java'
                sh 'jar cvfe HelloWorld.jar HelloWorld *.class'
                archiveArtifacts artifacts: 'HelloWorld.jar', followSymlinks:
false, onlyIfSuccessful: true
                echo '----- BUILD -----'
            }
        }
        stage('Deploy') {
            agent { label '!agent1' }
            steps {
                echo '----- DEPLOY -----'
                sh 'java -jar
$JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/archive/HelloWorld.jar'
                echo '----- DEPLOY -----'
            }
        }
    }
}
```

Pipeline

Definition

Pipeline script

Script

1 pipeline {
2 agent any
3 options {
4 skipStagesAfterUnstable()
5 }
6 stages {
7 stage('Build') {
8 agent { label 'agent1' }
9 steps {
10 echo '----- BUILD -----'
11 checkout([\$class: 'GitSCM', branches: [[name: '*/main']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[url: 'https://github.com/qasbi
12 sh 'javac HelloWorld.java'
13 sh 'jar cvfe HelloWorld.jar HelloWorld *.class'
14 archiveArtifacts artifacts: 'HelloWorld.jar', followSymlinks: false, onlyIfSuccessful: true
15 echo '----- BUILD -----'
16 }
17 }18 }19 }20 }21 }22 }23 }24 }25 }26 }27 }28 }29 }30 }31 }32 }33 }34 }35 }36 }37 }38 }39 }40 }41 }42 }43 }44 }45 }46 }47 }48 }49 }50 }51 }52 }53 }54 }55 }56 }57 }58 }59 }60 }61 }62 }63 }64 }65 }66 }67 }68 }69 }70 }71 }72 }73 }74 }75 }76 }77 }78 }79 }80 }81 }82 }83 }84 }85 }86 }87 }88 }89 }90 }91 }92 }93 }94 }95 }96 }97 }98 }99 }100 }101 }102 }103 }104 }105 }106 }107 }108 }109 }110 }111 }112 }113 }114 }115 }116 }117 }118 }119 }120 }121 }122 }123 }124 }125 }126 }127 }128 }129 }130 }131 }132 }133 }134 }135 }136 }137 }138 }139 }140 }141 }142 }143 }144 }145 }146 }147 }148 }149 }150 }151 }152 }153 }154 }155 }156 }157 }158 }159 }160 }161 }162 }163 }164 }165 }166 }167 }168 }169 }170 }171 }172 }173 }174 }175 }176 }177 }178 }179 }180 }181 }182 }183 }184 }185 }186 }187 }188 }189 }190 }191 }192 }193 }194 }195 }196 }197 }198 }199 }200 }201 }202 }203 }204 }205 }206 }207 }208 }209 }210 }211 }212 }213 }214 }215 }216 }217 }218 }219 }220 }221 }222 }223 }224 }225 }226 }227 }228 }229 }230 }231 }232 }233 }234 }235 }236 }237 }238 }239 }240 }241 }242 }243 }244 }245 }246 }247 }248 }249 }250 }251 }252 }253 }254 }255 }256 }257 }258 }259 }260 }261 }262 }263 }264 }265 }266 }267 }268 }269 }270 }271 }272 }273 }274 }275 }276 }277 }278 }279 }280 }281 }282 }283 }284 }285 }286 }287 }288 }289 }290 }291 }292 }293 }294 }295 }296 }297 }298 }299 }300 }301 }302 }303 }304 }305 }306 }307 }308 }309 }310 }311 }312 }313 }314 }315 }316 }317 }318 }319 }320 }321 }322 }323 }324 }325 }326 }327 }328 }329 }330 }331 }332 }333 }334 }335 }336 }337 }338 }339 }340 }341 }342 }343 }344 }345 }346 }347 }348 }349 }350 }351 }352 }353 }354 }355 }356 }357 }358 }359 }360 }361 }362 }363 }364 }365 }366 }367 }368 }369 }370 }371 }372 }373 }374 }375 }376 }377 }378 }379 }380 }381 }382 }383 }384 }385 }386 }387 }388 }389 }390 }391 }392 }393 }394 }395 }396 }397 }398 }399 }400 }401 }402 }403 }404 }405 }406 }407 }408 }409 }410 }411 }412 }413 }414 }415 }416 }417 }418 }419 }420 }421 }422 }423 }424 }425 }426 }427 }428 }429 }430 }431 }432 }433 }434 }435 }436 }437 }438 }439 }440 }441 }442 }443 }444 }445 }446 }447 }448 }449 }450 }451 }452 }453 }454 }455 }456 }457 }458 }459 }460 }461 }462 }463 }464 }465 }466 }467 }468 }469 }470 }471 }472 }473 }474 }475 }476 }477 }478 }479 }480 }481 }482 }483 }484 }485 }486 }487 }488 }489 }490 }491 }492 }493 }494 }495 }496 }497 }498 }499 }500 }501 }502 }503 }504 }505 }506 }507 }508 }509 }510 }511 }512 }513 }514 }515 }516 }517 }518 }519 }520 }521 }522 }523 }524 }525 }526 }527 }528 }529 }530 }531 }532 }533 }534 }535 }536 }537 }538 }539 }540 }541 }542 }543 }544 }545 }546 }547 }548 }549 }550 }551 }552 }553 }554 }555 }556 }557 }558 }559 }560 }561 }562 }563 }564 }565 }566 }567 }568 }569 }570 }571 }572 }573 }574 }575 }576 }577 }578 }579 }580 }581 }582 }583 }584 }585 }586 }587 }588 }589 }590 }591 }592 }593 }594 }595 }596 }597 }598 }599 }600 }601 }602 }603 }604 }605 }606 }607 }608 }609 }610 }611 }612 }613 }614 }615 }616 }617 }618 }619 }620 }621 }622 }623 }624 }625 }626 }627 }628 }629 }630 }631 }632 }633 }634 }635 }636 }637 }638 }639 }640 }641 }642 }643 }644 }645 }646 }647 }648 }649 }650 }651 }652 }653 }654 }655 }656 }657 }658 }659 }660 }661 }662 }663 }664 }665 }666 }667 }668 }669 }670 }671 }672 }673 }674 }675 }676 }677 }678 }679 }680 }681 }682 }683 }684 }685 }686 }687 }688 }689 }690 }691 }692 }693 }694 }695 }696 }697 }698 }699 }700 }701 }702 }703 }704 }705 }706 }707 }708 }709 }710 }711 }712 }713 }714 }715 }716 }717 }718 }719 }720 }721 }722 }723 }724 }725 }726 }727 }728 }729 }730 }731 }732 }733 }734 }735 }736 }737 }738 }739 }740 }741 }742 }743 }744 }745 }746 }747 }748 }749 }750 }751 }752 }753 }754 }755 }756 }757 }758 }759 }760 }761 }762 }763 }764 }765 }766 }767 }768 }769 }770 }771 }772 }773 }774 }775 }776 }777 }778 }779 }780 }781 }782 }783 }784 }785 }786 }787 }788 }789 }790 }791 }792 }793 }794 }795 }796 }797 }798 }799 }800 }801 }802 }803 }804 }805 }806 }807 }808 }809 }810 }811 }812 }813 }814 }815 }816 }817 }818 }819 }820 }821 }822 }823 }824 }825 }826 }827 }828 }829 }830 }831 }832 }833 }834 }835 }836 }837 }838 }839 }840 }841 }842 }843 }844 }845 }846 }847 }848 }849 }850 }851 }852 }853 }854 }855 }856 }857 }858 }859 }860 }861 }862 }863 }864 }865 }866 }867 }868 }869 }870 }871 }872 }873 }874 }875 }876 }877 }878 }879 }880 }881 }882 }883 }884 }885 }886 }887 }888 }889 }890 }891 }892 }893 }894 }895 }896 }897 }898 }899 }900 }901 }902 }903 }904 }905 }906 }907 }908 }909 }910 }911 }912 }913 }914 }915 }916 }917 }918 }919 }920 }921 }922 }923 }924 }925 }926 }927 }928 }929 }930 }931 }932 }933 }934 }935 }936 }937 }938 }939 }940 }941 }942 }943 }944 }945 }946 }947 }948 }949 }950 }951 }952 }953 }954 }955 }956 }957 }958 }959 }960 }961 }962 }963 }964 }965 }966 }967 }968 }969 }970 }971 }972 }973 }974 }975 }976 }977 }978 }979 }980 }981 }982 }983 }984 }985 }986 }987 }988 }989 }990 }991 }992 }993 }994 }995 }996 }997 }998 }999 }1000 }1001 }1002 }1003 }1004 }1005 }1006 }1007 }1008 }1009 }1010 }1011 }1012 }1013 }1014 }1015 }1016 }1017 }1018 }1019 }1020 }1021 }1022 }1023 }1024 }1025 }1026 }1027 }1028 }1029 }1030 }1031 }1032 }1033 }1034 }1035 }1036 }1037 }1038 }1039 }1040 }1041 }1042 }1043 }1044 }1045 }1046 }1047 }1048 }1049 }1050 }1051 }1052 }1053 }1054 }1055 }1056 }1057 }1058 }1059 }1060 }1061 }1062 }1063 }1064 }1065 }1066 }1067 }1068 }1069 }1070 }1071 }1072 }1073 }1074 }1075 }1076 }1077 }1078 }1079 }1080 }1081 }1082 }1083 }1084 }1085 }1086 }1087 }1088 }1089 }1090 }1091 }1092 }1093 }1094 }1095 }1096 }1097 }1098 }1099 }1100 }1101 }1102 }1103 }1104 }1105 }1106 }1107 }1108 }1109 }1110 }1111 }1112 }1113 }1114 }1115 }1116 }1117 }1118 }1119 }1120 }1121 }1122 }1123 }1124 }1125 }1126 }1127 }1128 }1129 }1130 }1131 }1132 }1133 }1134 }1135 }1136 }1137 }1138 }1139 }1140 }1141 }1142 }1143 }1144 }1145 }1146 }1147 }1148 }1149 }1150 }1151 }1152 }1153 }1154 }1155 }1156 }1157 }1158 }1159 }1160 }1161 }1162 }1163 }1164 }1165 }1166 }1167 }1168 }1169 }1170 }1171 }1172 }1173 }1174 }1175 }1176 }1177 }1178 }1179 }1180 }1181 }1182 }1183 }1184 }1185 }1186 }1187 }1188 }1189 }1190 }1191 }1192 }1193 }1194 }1195 }1196 }1197 }1198 }1199 }1200 }1201 }1202 }1203 }1204 }1205 }1206 }1207 }1208 }1209 }1210 }1211 }1212 }1213 }1214 }1215 }1216 }1217 }1218 }1219 }1220 }1221 }1222 }1223 }1224 }1225 }1226 }1227 }1228 }1229 }1230 }1231 }1232 }1233 }1234 }1235 }1236 }1237 }1238 }1239 }1240 }1241 }1242 }1243 }1244 }1245 }1246 }1247 }1248 }1249 }1250 }1251 }1252 }1253 }1254 }1255 }1256 }1257 }1258 }1259 }1260 }1261 }1262 }1263 }1264 }1265 }1266 }1267 }1268 }1269 }1270 }1271 }1272 }1273 }1274 }1275 }1276 }1277 }1278 }1279 }1280 }1281 }1282 }1283 }1284 }1285 }1286 }1287 }1288 }1289 }1290 }1291 }1292 }1293 }1294 }1295 }1296 }1297 }1298 }1299 }1300 }1301 }1302 }1303 }1304 }1305 }1306 }1307 }1308 }1309 }1310 }1311 }1312 }1313 }1314 }1315 }1316 }1317 }1318 }1319 }1320 }1321 }1322 }1323 }1324 }1325 }1326 }1327 }1328 }1329 }1330 }1331 }1332 }1333 }1334 }1335 }1336 }1337 }1338 }1339 }1340 }1341 }1342 }1343 }1344 }1345 }1346 }1347 }1348 }1349 }1350 }1351 }1352 }1353 }1354 }1355 }1356 }1357 }1358 }1359 }1360 }1361 }1362 }1363 }1364 }1365 }1366 }1367 }1368 }1369 }1370 }1371 }1372 }1373 }1374 }1375 }1376 }1377 }1378 }1379 }1380 }1381 }1382 }1383 }1384 }1385 }1386 }1387 }1388 }1389 }1390 }1391 }1392 }1393 }1394 }1395 }1396 }1397 }1398 }1399 }1400 }1401 }1402 }1403 }1404 }1405 }1406 }1407 }1408 }1409 }1410 }1411 }1412 }1413 }1414 }1415 }1416 }1417 }1418 }1419 }1420 }1421 }1422 }1423 }1424 }1425 }1426 }1427 }1428 }1429 }1430 }1431 }1432 }1433 }1434 }1435 }1436 }1437 }1438 }1439 }1440 }1441 }1442 }1443 }1444 }1445 }1446 }1447 }1448 }1449 }1450 }1451 }1452 }1453 }1454 }1455 }1456 }1457 }1458 }1459 }1460 }1461 }1462 }1463 }1464 }1465 }1466 }1467 }1468 }1469 }1470 }1471 }1472 }1473 }1474 }1475 }1476 }1477 }1478 }1479 }1480 }1481 }1482 }1483 }1484 }1485 }1486 }1487 }1488 }1489 }1490 }1491 }1492 }1493 }1494 }1495 }1496 }1497 }1498 }1499 }1500 }1501 }1502 }1503 }1504 }1505 }1506 }1507 }1508 }1509 }1510 }1511 }1512 }1513 }1514 }1515 }1516 }1517 }1518 }1519 }1520 }1521 }1522 }1523 }1524 }1525 }1526 }1527 }1528 }1529 }1530 }1531 }1532 }1533 }1534 }1535 }1536 }1537 }1538 }1539 }1540 }1541 }1542 }1543 }1544 }1545 }1546 }1547 }1548 }1549 }1550 }1551 }1552 }1553 }1554 }1555 }1556 }1557 }1558 }1559 }1560 }1561 }1562 }1563 }1564 }1565 }1566 }1567 }1568 }1569 }1570 }1571 }1572 }1573 }1574 }1575 }1576 }1577 }1578 }1579 }1580 }1581 }1582 }1583 }1584 }1585 }1586 }1587 }1588 }1589 }1590 }1591 }1592 }1593 }1594 }1595 }1596 }1597 }1598 }1599 }1600 }1601 }1602 }1603 }1604 }1605 }1606 }1607 }1608 }1609 }1610 }1611 }1612 }1613 }1614 }1615 }1616 }1617 }1618 }1619 }1620 }1621 }1622 }1623 }1624 }1625 }1626 }1627 }1628 }1629 }1630 }1631 }1632 }1633 }1634 }1635 }1636 }1637 }1638 }1639 }1640 }1641 }1642 }1643 }1644 }1645 }1646 }1647 }1648 }1649 }1650 }1651 }1652 }1653 }1654 }1655 }1656 }1657 }1658 }1659 }1660 }1661 }1662 }1663 }1664 }1665 }1666 }1667 }1668 }1669 }1670 }1671 }1672 }1673 }1674 }1675 }1676 }1677 }1678 }1679 }1680 }1681 }1682 }1683 }1684 }1685 }1686 }1687 }1688 }1689 }1690 }1691 }1692 }1693 }1694 }1695 }1696 }1697 }1698 }1699 }1700 }1701 }1702 }1703 }1704 }1705 }1706 }1707 }1708 }1709 }1710 }1711 }1712 }1713 }1714 }1715 }1716 }1717 }1718 }1719 }1720 }1721 }1722 }1723 }1724 }1725 }1726 }1727 }1728 }1729 }1730 }1731 }1732 }1733 }1734 }1735 }1736 }1737 }1738 }1739 }1740 }1741 }1742 }1743 }1744 }1745 }1746 }1747 }1748 }1749 }1750 }1751 }1752 }1753 }1754 }1755 }1756 }1757 }1758 }1759 }1760 }1761 }1762 }1763 }1764 }1765 }1766 }1767 }1768 }1769 }1770 }1771 }1772 }1773 }1774 }1775 }1776 }1777 }1778 }1779 }1780 }1781 }1782 }1783 }1784 }1785 }1786 }1787 }1788 }1789 }1790 }1791 }1792 }1793 }1794 }1795 }1796 }1797 }1798 }1799 }1800 }1801 }1802 }1803 }1804 }1805 }18


```
pipeline {  
  agent any
```

This specifies a Pipeline config with a default agent of **any**, meaning any agent can pick up this pipeline (we override this at the stage level).

```
options {  
  skipStagesAfterUnstable()  
}
```

This specifies that we would like to fail fast - there is no point deploying something that failed to build or, more likely, failed the tests.

```
stages {  
  stage('Build') {  
    agent { label 'agent1' }  
    ...  
  }  
  stage('Deploy') {  
    agent { label '!agent1' }  
    ...  
  }  
}
```

Declare our stages and what agent can run them - overriding the earlier setting for **any**.

```
checkout([$class: 'GitSCM', branches: [[name: '*/main']],  
doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],  
userRemoteConfigs: [[url: 'https://github.com/qasbirchall/java-hello-world']]])
```

This tells Jenkins to checkout our repository from the **main** branch: you should provide your own repository as set up earlier.

```
sh 'javac HelloWorld.java'  
sh 'jar cvfe HelloWorld.jar HelloWorld *.class'
```

This runs the two shell commands we need to compile and build our code into a jar file.

```
archiveArtifacts artifacts: 'HelloWorld.jar', followSymlinks: false,  
onlyIfSuccessful: true
```

Once we have our jar file successfully built (**onlyIfSuccessful: true**) we archive it for later use.

```
sh 'java -jar  
$JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/archive/HelloWorld.jar'
```

Finally, we want to take the archive and execute it as our **Deploy** stage.

Notice how Jenkins archives per Job per Build - safely keeping each version of our application/artefact in its own directory.

This can be very useful for revision control, roll-backs and auditing, but can lead to some heavy disk usage if the artefacts are generally large and/or undergo a high volume of changes.

Step 8: CI/CD

We should now be able to trigger our project (manually at this point).

First notice how this Stage (**Build**) is first picked up by the default master agent before being delegated to our specified agent: **agent1**.

Build Executor Status			^
master			
1	Idle		
2	java-app	#7 (Build)	
agent1			
1	java-app	#7 (Build)	

Upon success, the Pipeline will move to the **Deploy** stage and run the Jar artefact we generated in the **Build** stage.
Notice how this Stage (**Deploy**) is carried out by our original/default agent.

Build Executor Status			^
master			
1	java-app	#7 (Deploy)	
2	java-app	#7 (Deploy)	
agent1			
1	Idle		

Check the console log for the project to see that we have the expected output, it should show something like:

```

> git checkout -f 2ba02f7714fd2fb1357624d9fc7040b50f34e73e # timeout=10
> git rev-list --no-walk 2ba02f7714fd2fb1357624d9fc7040b50f34e73e # timeout=10
[Pipeline] sh
+ javac HelloWorld.java
[Pipeline] sh
+ jar cvfe HelloWorld.jar HelloWorld HelloWorld.class
added manifest
adding: HelloWorld.class(in = 426) (out= 289)(deflated 32%)
[Pipeline] archiveArtifacts
Archiving artifacts
[Pipeline] echo
----- BUILD -----
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] node
Running on Jenkins in /home/jenkins/.jenkins/workspace/java-app@2
[Pipeline] {
[Pipeline] echo
----- DEPLOY -----
[Pipeline] sh
+ java -jar /home/jenkins/.jenkins/jobs/java-app/builds/6/archive/HelloWorld.jar
Hello world!
[Pipeline] echo
----- DEPLOY -----
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Conclusion

As we have seen, it is relatively simple to register build agents with Jenkins and selectively push jobs to them based on labels. This begs the question of why would you want to do this? As mentioned already, concurrency is enough of a bonus to make this pattern pretty much ubiquitous in computing.

This tutorial should point the way to other advantages specific to DevOps and CI/CD, such as the ability to provide distinct build agents for each version of Java, for instance, or maybe we would like to have distinct Deploy agents for each of our environments to limit exposure should something go wrong.

Exercises

Automation

Automate the process so that we build and deploy a new artefact in response to changes to the repository.

HINT: you will probably need to set up a GitHub web hook.