

Звіт
З лабораторної роботи №3
Студента групи МІТ-31
Добровольського Арсенія Михайловича
Варіант №9

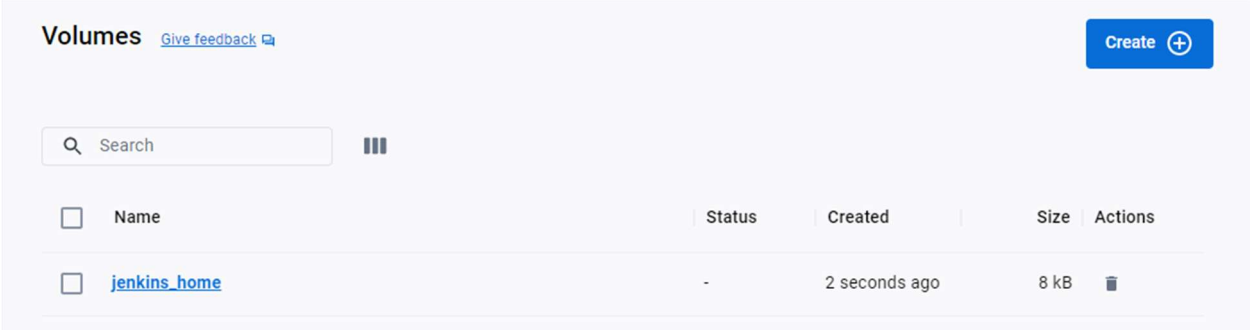
Тема роботи: Конвеєр безперервної інтеграції з використанням Jenkins.

Мета роботи: навчитися створювати елементарний конвеєр автоматизованого випуску програмного забезпечення (application delivery pipeline).

Хід виконання роботи

1. Створення volume у Docker:

```
C:\Users\Fijitsu E546>docker volume create jenkins_home  
jenkins_home
```



Volumes Give feedback					Create +
<input type="text" value="Search"/>					
<input type="checkbox"/>	Name	Status	Created	Size	Actions
<input type="checkbox"/>	jenkins_home	-	2 seconds ago	8 kB	

2. Створення мережі для налаштування взаємодії контейнерів Jenkins та Docker:

```
C:\Users\Fijitsu E546>docker network create jenkins  
9ea4d19119bd65456a63cdd4d93cfa3fc7aa08cd7e4dba800ba8de18e8a70951
```

3. Запуск контейнера docker in docker:

```
C:\Users\Fijitsu E546>docker run --name jenkins-docker --detach --privileged --network jenkins
--network-alias docker --env DOCKER_TLS_CERTDIR=/certs --volume jenkins-docker-certs:/certs/cli
ent --mount src=jenkins_home,target=/var/jenkins_home --restart unless-stopped docker:dind
Unable to find image 'docker:dind' locally
dind: Pulling from library/docker
96526aa774ef: Pull complete
308a300cc30f: Pull complete
4f4fb700ef54: Pull complete
5fcb7611fd77: Pull complete
be9955e691a8: Pull complete
1b5ce73eb848: Pull complete
fe45098d8d9e: Pull complete
843bf259cc12: Pull complete
e56cd900ceac: Pull complete
605d1390b383: Pull complete
8f7854084fed: Pull complete
0a217d94afa5: Pull complete
50e7dd98501d: Pull complete
68811508944a: Pull complete
Digest: sha256:1dfc375736e448806602211e09a9b1390eb110548dcb839eef374da357ca5f5d
Status: Downloaded newer image for docker:dind
2d05f2ed325911841ab7276f50f75e28aa38c51edd15d2b4cc0ef416010d26eb
```

Volumes Give feedback						Create
<input type="text" value="Search"/>						
<input type="checkbox"/>	Name	Status	Created	Size	Actions	
<input type="checkbox"/>	27f8f62da9636db3b88d8a547825143611c31562f68e1f7ddeb9df9b1ec9	in use	3 minutes ago	336 kB		
<input type="checkbox"/>	jenkins-docker-certs	in use	3 minutes ago	28 kB		
<input type="checkbox"/>	jenkins_home	in use	11 minutes ago	8 kB		

Containers [Give feedback](#)




Container CPU usage ⓘ
0.00% / 400% (4 cores allocated)

Container memory usage ⓘ
47.86MB / 5.98GB

Show charts ▾

☰

☒ Only show running containers

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	<div><div> jenkins-docker</div><div>2d05f2ed3259 </div></div>	docker:dind	Running	0%		3 minutes ago	<div><div>■</div><div>:</div><div></div></div>

4. Створення Dockerfile:

```
Dockerfile U X
Lab3 > Dockerfile > ...
1 FROM jenkins/jenkins:lts
2 USER root
3 RUN apt-get update && apt-get install -y apt-transport-https \
4     ca-certificates curl gnupg2 \
5     software-properties-common
6 RUN curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
7 RUN apt-key fingerprint 0EBFCD88
8 RUN add-apt-repository \
9     "deb [arch=amd64] https://download.docker.com/linux/debian \
10    $(lsb_release -cs) stable"
11 RUN apt-get update && apt-get install -y docker-ce-cli
12 RUN jenkins
13 RUN jenkins-plugin-cli
```

5. Створення docker-образу:

```
D:\KNU_FIT\3rd_grade\Programming technologies\pt2023\Lab3>docker build -t lab3-jenkins:1.1 .
[+] Building 415.5s (12/12) FINISHED
=> [internal] load .dockerignore                                docker:default 3.0s
=> => transferring context: 2B                                  0.1s
=> [internal] load build definition from Dockerfile            3.7s
=> => transferring dockerfile: 548B                             0.7s
=> [internal] load metadata for docker.io/jenkins/jenkins:lts 7.9s
=> [auth] jenkins/jenkins:pull token for registry-1.docker.io 0.0s
=> [1/7] FROM docker.io/jenkins/jenkins:lts@sha256:80587de2dac2bb701cd0b14d35988e591d6 53.3s
=> => resolve docker.io/jenkins/jenkins:lts@sha256:80587de2dac2bb701cd0b14d35988e591d62 0.8s
=> => sha256:80587de2dac2bb701cd0b14d35988e591d62589fd337a4b584f4c52101 2.36kB / 2.36kB 0.0s
=> => sha256:683d24e196c4cc90cabdefc74942a5af842dca9f836399027ecd1a9144 2.58kB / 2.58kB 0.0s
=> => sha256:39353289457d5c5a6e537e4e39745624427aa61b4770a952b9d4ecb6 12.38kB / 12.38kB 0.0s
=> => sha256:de38613af57f6912dac28e588c69c3343ce500b443bcc54d5836ca12a3 1.24kB / 1.24kB 0.8s
=> => sha256:e7dfbe47a42421e539313f92f9a55420c5d83cef426f347360f0d00 61.35MB / 61.35MB 17.3s
=> => sha256:63a7b790559df16d8c274a190e98c4178933bc07756c95c50a2ffec1 8.68MB / 8.68MB 4.0s
=> => sha256:5e2a9b06c446c792c609d74f7487d0ca9e7fe4d9a6f7f6edb91c0c538be87a 183B / 183B 2.1s
=> => sha256:aaaca84f7bc70e402a11fb752ddbcbcf64b4d6f17fda86d3ee37085 89.34MB / 89.34MB 29.9s
=> => sha256:b7d3a74d9ec16713fd23c837f694fe8ab8e71f6bfad6ce49828117ab5947b6 190B / 190B 4.6s
=> => sha256:c1655dbb8674653fecf213128cd716d110d0010daac14ab6d42900eaa 6.09MB / 6.09MB 7.4s
=> => sha256:8517c106c3457c35aa523eab75a6392c102e6254040d3160dca92d9 74.02MB / 74.02MB 32.0s
=> => sha256:6c94ede8b80860505bfea9f26ffde7aed397f55beee6ed47c3fd1d2fc 1.92kB / 1.92kB 18.5s
=> => extracting sha256:e7dfbe47a42421e539313f92f9a55420c5d83cef426f347360f0d00ff19e92 16.7s
=> => sha256:f94c7e55f7718a793dcb8e6d26ce15a91f9aab709df8e46be67f80550 1.16kB / 1.16kB 19.4s
=> => sha256:8f558801c0511e9fb522708fb94f0a5353f08910c2b3287850a4b1c3aeafa 391B / 391B 20.0s
=> => extracting sha256:63a7b790559df16d8c274a190e98c4178933bc07756c95c50a2ffec1e87db 0.3s
=> => extracting sha256:de38613af57f6912dac28e588c69c3343ce500b443bcc54d5836ca12a3b04ea 0.0s
=> => extracting sha256:5e2a9b06c446c792c609d74f7487d0ca9e7fe4d9a6f7f6edb91c0c538be87a1 0.0s
=> => extracting sha256:aaaca84f7bc70e402a11fb752ddbcbcf64b4d6f17fda86d3ee37085315aad1b 1.2s
=> => extracting sha256:b7d3a74d9ec16713fd23c837f694fe8ab8e71f6bfad6ce49828117ab5947b6d 0.0s
=> => extracting sha256:c1655dbb8674653fecf213128cd716d110d0010daac14ab6d42900eaa7a80f 0.1s
=> => extracting sha256:8517c106c3457c35aa523eab75a6392c102e6254040d3160dca92d99b994f60 1.3s
=> => extracting sha256:6c94ede8b80860505bfea9f26ffde7aed397f55beee6ed47c3fd1d2fc309380 0.0s
=> => extracting sha256:f94c7e55f7718a793dcb8e6d26ce15a91f9aab709df8e46be67f80550438548 0.0s
=> => extracting sha256:8f558801c0511e9fb522708fb94f0a5353f08910c2b3287850a4b1c3aeafaa9 0.0s
=> [2/7] RUN apt-get update && apt-get install -y apt-transport-https ca-certi 281.2s
=> [3/7] RUN curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - 5.2s
=> [4/7] RUN apt-key fingerprint 0EBFCD88 2.9s
=> [5/7] RUN add-apt-repository "deb [arch=amd64] https://download.docker.com/li 9.4s
=> [6/7] RUN apt-get update && apt-get install -y docker-ce-cli 33.3s
=> [7/7] RUN jenkins-plugin-cli 5.6s
=> exporting to image 6.5s
=> => exporting layers 6.3s
```


Images

[Give feedback](#)

Local

Hub

Artifactory

EARLY ACCESS

1.18 GB / 1.48 GB in use

7 images

Last refresh: 7 hours ago

Q Search

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	lab3-jenkins 77ffae78bf8e	1.1	In use	6 hours ago	852.91 MB	▶ ⋮ 🗑
<input type="checkbox"/>	docker 5b912308540a	dind	In use	2 days ago	336.75 MB	▶ ⋮ 🗑
<input type="checkbox"/>	pawnshop_app f2fe80852ee8	v1.0	Unused	7 days ago	1.01 GB	▶ ⋮ 🗑
<input type="checkbox"/>	busybox a416a98b71e2	latest	Unused	3 months ago	4.26 MB	▶ ⋮ 🗑
<input type="checkbox"/>	docker/volumes-backup-extension 6872a696b721	1.1.4	Unused	5 months ago	118.52 MB	▶ ⋮ 🗑
<input type="checkbox"/>	felipecruz/alpine-tar-zstd 31988344315d	latest	Unused	1 year ago	6.98 MB	▶ ⋮ 🗑

Showing 7 items

6. Запуск контейнера з Jenkins:

```
D:\KNU_FIT\3rd_grade\Programming technologies\pt2023\Lab3>docker run --name jenkins-blueocean
--detach --network jenkins --env DOCKER_HOST=tcp://docker:2376 --env DOCKER_CERT_PATH=/certs/c
lient --env DOCKER_TLS_VERIFY=1 --volume jenkins_home:/var/jenkins_home --volume jenkins-docke
r-certs:/certs/client:ro --publish 8085:8080 --publish 50000:50000 --restart unless-stopped la
b3-jenkins:1.1
f4c305cbf51aae27f968fdb25334ee461654f60b98a97328cd3a35147852559a
```

Containers

[Give feedback](#)

Container CPU usage ⓘ

0.17% / 400% (4 cores allocated)

Container memory usage ⓘ

993.58MB / 5.98GB

[Show charts](#)

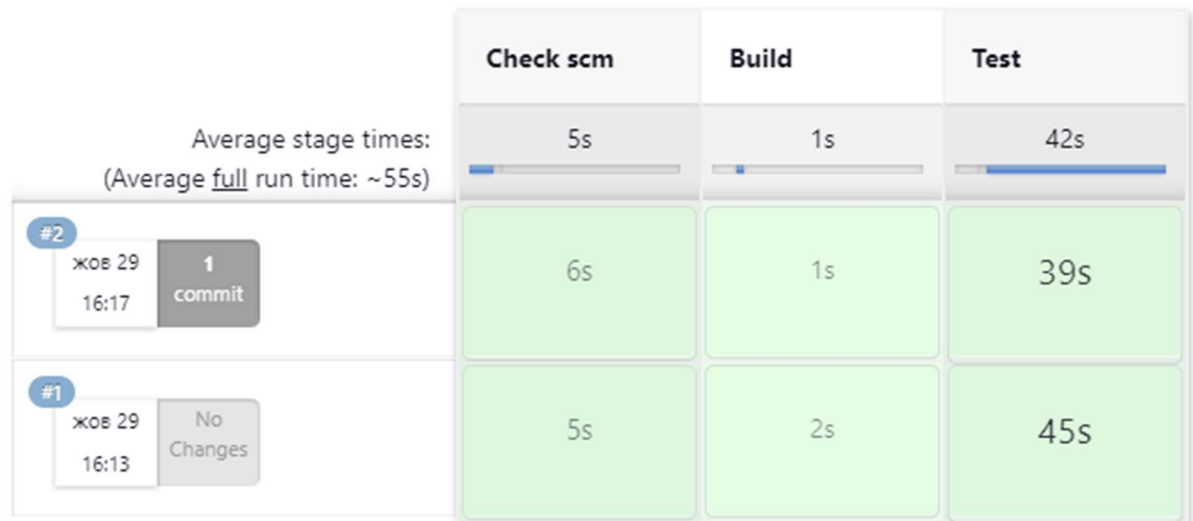
Q Search

Only show running containers

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	jenkins-blueocean f4c305cbf51a	lab3-jenkins:1	Running	0.16%	50000:50000 Show all ports (2)	7 hours ago	■ ⋮ 🗑
<input type="checkbox"/>	jenkins-docker 2d05f2ed3259	docker:dind	Running	0.01%		7 hours ago	■ ⋮ 🗑

7. Результат запуску конвеєра у Jenkins після проведення усіх необхідних налаштувань:

Stage View



```
15:43:20 + python3 Lab3/pawnshop_tests.py
```

```
15:43:20
```

```
15:43:20 Running tests...
```

```
15:43:20 -----
```

```
15:43:20 .x....
```

```
15:43:20 -----
```

```
15:43:20 Ran 6 tests in 0.001s
```

```
15:43:20
```

```
15:43:20 OK (expected failures=1)
```

```
15:43:20
```

```
15:43:20 Generating XML reports...
```

Всі етапи виконання конвеєра відображаються зеленим кольором, що свідчить про те, що конвеєр було успішно запущено і всі тести виконалися.

8. Доповнимо Jenkinsfile, додавши декілька фрагментів для автоматичного формування docker-образу і вивантаження його на Dockerhub:

```
environment {
|   DOCKERHUB_CREDENTIALS = credentials('arsgoo-dockerhub')
| }

stage('Build') {
|   steps {
|       echo "Building ...${BUILD_NUMBER}"
|       sh 'cp Lab3/Dockerfile .'
|       sh 'docker build -t arsgoo/pawnshop_tests:latest .'
|       echo "Build completed"
|   }
| }
}
```

```
}
stage('Login') {
|   steps {
|       sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
|   }
| }
stage('Push') {
|   steps {
|       sh 'docker push arsgoo/pawnshop_tests:latest'
|   }
| }
post {
|   always {
|       sh 'docker logout'
|   }
| }
}
```

Ці фрагменти коду забезпечують автоматичне формування docker-образу arsgoo/pawnshop_tests, вхід в акаунт Dockerhub, вивантаження туди даного образу і наостанок вихід з акаунту.

9. Результат запуску оновленого конвеєру:

		Declarative: Checkout SCM	Check scm	Build	Test	Login	Push	Declarative: Post Actions
Average stage times: (Average full run time: ~2min 9s)		3s	10s	13s	54s	3s	27s	1s
#9	жов 30 15:42 No Changes	2s	10s	12s	54s	3s	5s	2s

```

15:43:20 + python3 Lab3/pawnshop_tests.py
15:43:20
15:43:20 Running tests...
15:43:20 -----
15:43:20 .x...
15:43:20 -----
15:43:20 Ran 6 tests in 0.001s
15:43:20
15:43:20 OK (expected failures=1)
15:43:20
15:43:20 Generating XML reports...

```

10. Дослідження тригерів, доступних у Jenkins:

- Build periodically

Автоматично перезапускає конвеєр через рівні проміжки часу.

☒ Build periodically ?

Schedule ?

H/15 * * * *

Would last have run at Monday, October 30, 2023 at 7:42:52 PM Coordinated Universal Time; would next run at Monday, October 30, 2023 at 7:57:52 PM Coordinated Universal Time.

			Declarative: Checkout SCM	Check scm	Build	Test	Login	Push	Declarative: Post Actions
Average stage times: (Average <u>full</u> run time: ~1min 55s)			3s	9s	12s	48s	2s	21s	1s
#17	Job 30	No Changes	2s	6s	11s	42s	2s	1min 0s	1s
#16	Job 30	No Changes	1s	5s	6s	32s	2s	5s	1s
#15	Job 30	No Changes	1s	5s	6s	41s	2s	5s	1s
#14	Job 30	No Changes	1s	5s	6s	32s	2s	6s	2s

У наведеному вище випадку, конвеєр запускається кожні 15 хвилин (20:42, 20:57, 21:12, 21:27, наступний запуск відбудеться о 21:42 відповідно).

- Poll SCM

Цей тригер через рівні інтервали перевіряє наявність змін у джерелі і перезапускає конвеєр ЛИШЕ у разі виявлення таких змін. Тобто, задавши cron-рядок із попереднього прикладу, Jenkins кожні 15 хвилин перевірятиме джерело на нові зміни, проте зовсім необов'язково, що конвеєр буде перезапускатися після кожної такої перевірки.

- GitHub hook

GitHub hook – це HTTP-запит, який GitHub надсилає Jenkins, коли відбувається певна подія в репозиторії (наприклад, push). Цей запит Jenkins якраз і використовує для запуску конвеєра.

Для того, щоб GitHub зміг отримати доступ до Jenkins, слід перетворити локальну адресу Jenkins на публічну URL-адресу. Для цього використаємо програму ngrok, в якій задамо команду **ngrok http 8085** (8085 – порт, на якому власне працює Jenkins). В результаті отримаємо таке вікно, в якому знаходимо сформовану адресу (підкреслена червоним):

```
ngrok
Introducing Always-On Global Server Load Balancer: https://ngrok.com/r/gslb
Session Status      online
Session Expires     1 hour, 45 minutes
Terms of Service     https://ngrok.com/tos
Version             3.3.5
Region              Europe (eu)
Latency              36ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://bcf0-91-123-151-115.ngrok.io -> http://localhost:8085
Connections
  ttl    opn    rt1    rt5    p50    p90
    0     0     0.00  0.00  0.00  0.00
```

Далі копіюємо цю адресу, переходимо в потрібний GitHub репозиторій, обираємо Settings -> Webhooks -> Add webhook, і у поле Payload URL вставляємо посилання, додавши до нього /github-webhook/ (закриваючий слеш повинен бути обов'язково!):

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, etc). More information can be found in [our developer documentation](#).

Payload URL *

`https://bcf0-91-123-151-115.ngrok.io/github-webhook/`

Content type

`application/x-www-form-urlencoded`

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

☒ **Enable SSL verification** ☐ **Disable (not recommended)**

Which events would you like to trigger this webhook?

☒ Just the push event.

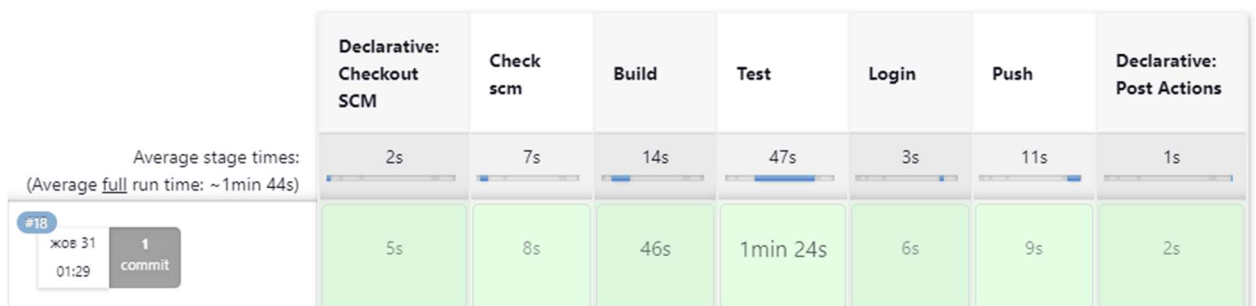
☐ Send me everything.

Webhooks Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ `https://410c-91-123-151-115.ngrok.... (push)` Edit Delete

Робимо невеликі зміни в коді проєкту і «пушимо» їх в репозиторій для того, щоб перевірити, чи відреагує на них Jenkins.



Так, дійсно, після виконання push Jenkins моментально виявив нові зміни у репозиторії і автоматично перезапустив конвеєр. Результат виконання, як і раніше, успішний.

- Trigger builds remotely (e.g, from scripts)

Цей тригер дозволяє віддалено ініціювати запуск конвеєра (наприклад, з іншого браузера або клієнта). Розглянемо два випадки запуску збірки: через нову вкладку того самого сеансу браузера і через інший браузер.

1) Нова вкладка

Переходимо у налаштування конвеєра і натискаємо на checkbox навпроти **Trigger builds remotely (e.g., from scripts)**. Далі у поле **Authentication Token** вписуємо довільний токен (наводжу свій на скріншоті нижче):

☒ Trigger builds remotely (e.g., from scripts) ?

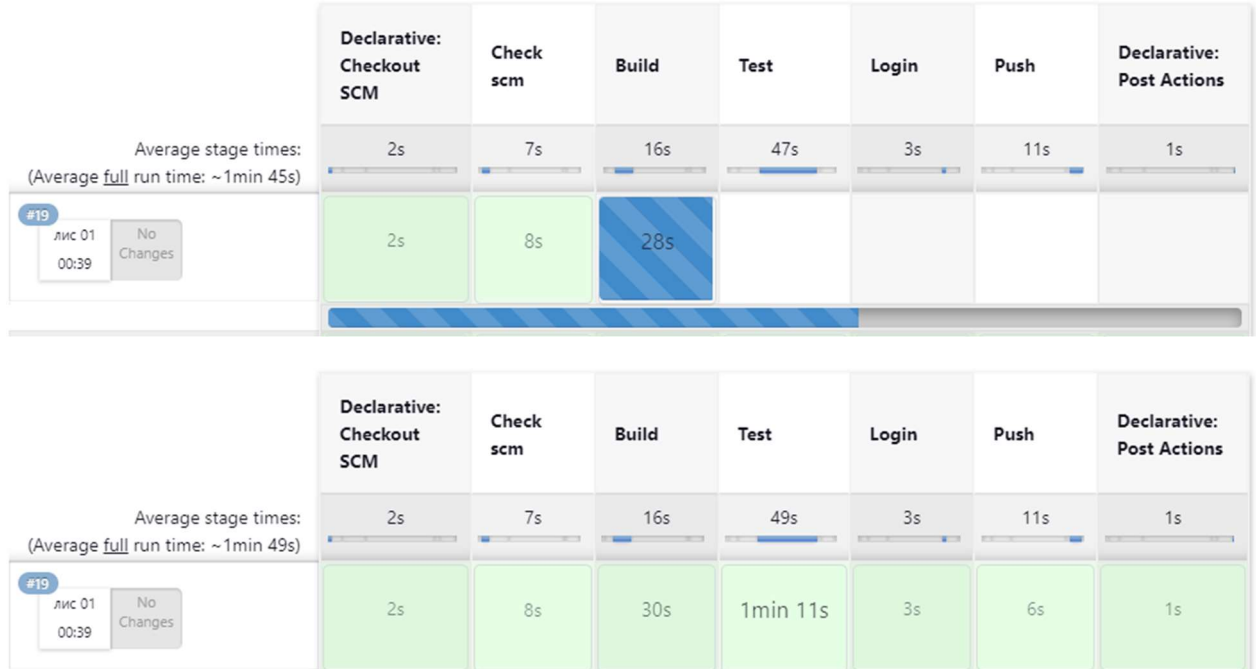
Authentication Token

ars13mit

Use the following URL to trigger build remotely: JENKINS_URL/job/Lab3/build?token=TOKEN_NAME or /buildWithParameters?token=TOKEN_NAME

Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

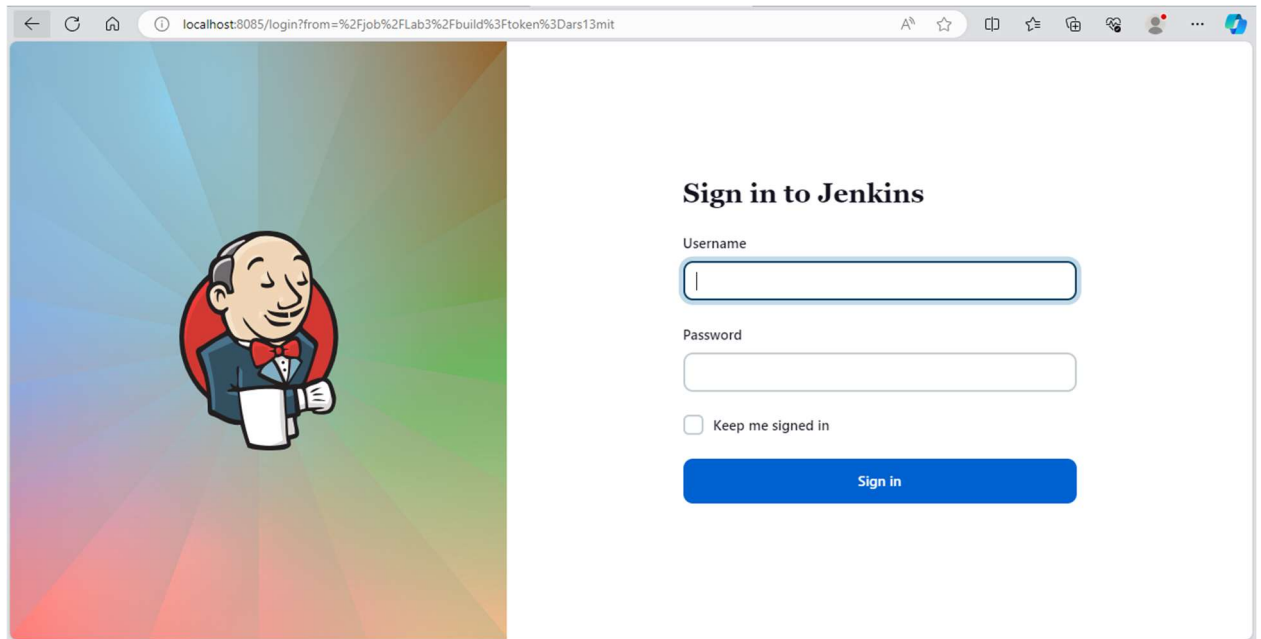
Зберігаємо зміни і переходимо на нову вкладку, в якій вставляємо посилання <http://localhost:8085/job/Lab3/build?token=ars13mit>.



Jenkins миттєво отримав вказівку та ініціював запуск пайплайну. Результат знову успішний.

2) Інший браузер

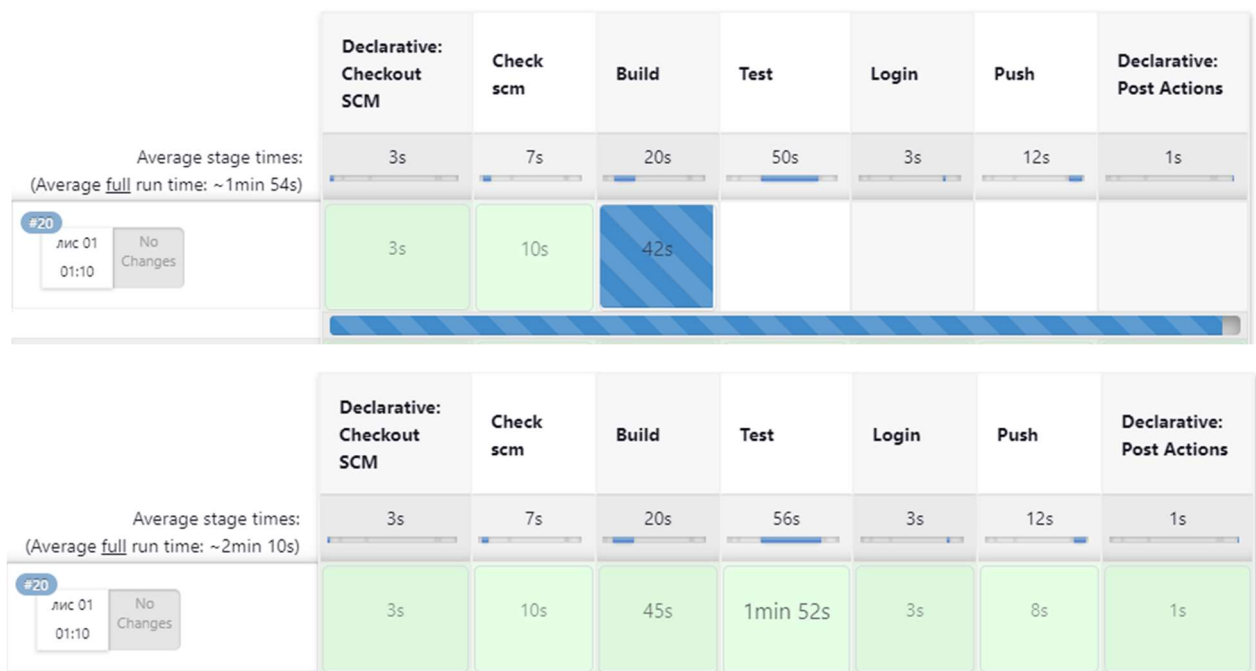
Спробуємо вставити теж саме посилання у вкладці іншого браузера:



Бачимо вікно входу у Jenkins. Воно виникає через те, що надсилати такого роду запити може лише авторизована особа, яка має відповідні права. Проте може виникнути ситуація, коли в людини наприклад немає акаунту в Jenkins, а запустити конвеєр необхідно. В такому разі слід додатково встановити плагін **Build Authorization Token Root**, який дозволяє усім користувачам відправляти запити на запуск конвеєра без необхідності авторизації.

Встановивши плагін, необхідно знову ж таки вписати **Authentication Token** у налаштуваннях пайплайну (для зручності залишу попередній).

Далі необхідно вставити у вкладку браузера посилання у форматі <https://jenkins.example.com/buildByToken/build?token=Secure-Token&job=TestJob>, де **https://jenkins.example.com** – локальна адреса, за якою функціонує Jenkins, **Secure-Token** – сформований нами токен, а **TestJob** – назва конвеєру (в моєму випадку це `http://localhost:8085, ars13mit` і `Lab3` відповідно). Таким чином, моє посилання виглядатиме так: <http://localhost:8085/buildByToken/build?token=ars13mit&job=Lab3>. Вставимо його у браузер:



Як і передбачалося, конвеєр вкотре запустився і без помилок пройшов усі необхідні етапи.

Варто зауважити, що кроки, описані в цьому пункті, також будуть актуальними і для випадку запуску конвеєра з вікна «Інкогніто» того самого браузера.

Висновок: в ході виконання лабораторної роботи ми детально познайомилися із середовищем автоматизації Jenkins, створили власний елементарний конвеєр автоматизованого випуску програмного забезпечення, а також на практиці дослідили використання різноманітних тригерів для автоматизації запуску пайплайну.

Врешті-решт, було забезпечено автоматичне формування docker-образу проєкту і вивантаження його на Dockerhub.