Звіт

З лабораторної роботи №3

Студента групи МІТ-31

Добровольського Арсенія Михайловича

Варіант №9

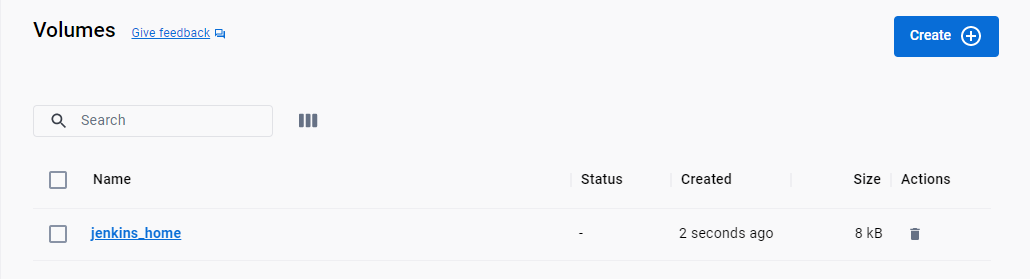
**Тема роботи:** Конвеєр безперервної інтеграції з використанням Jenkins.

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

Хід виконання роботи

1. Створення volume у Docker:

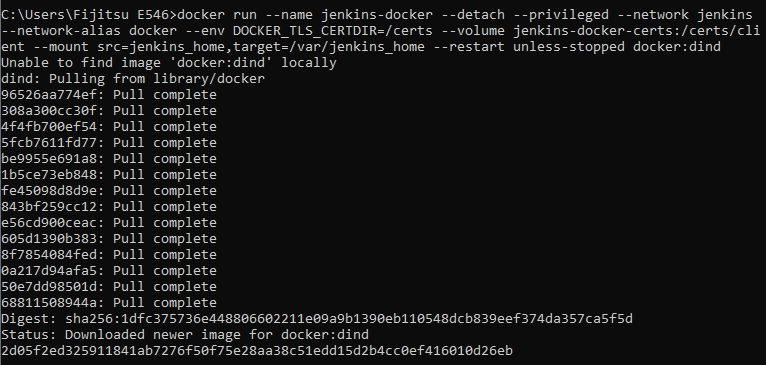


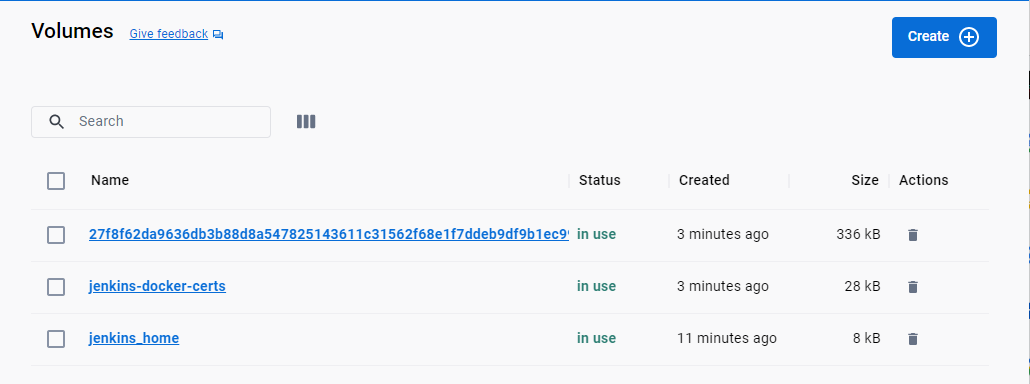


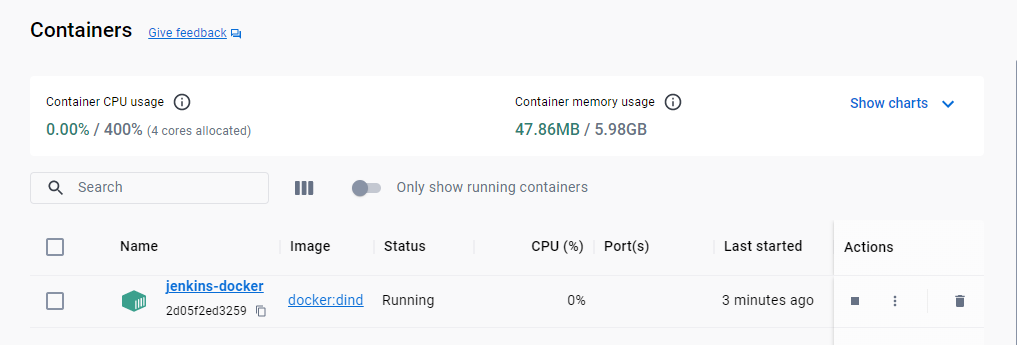
1. Створення мережі для налаштування взаємодії контейнерів Jenkins та Docker:



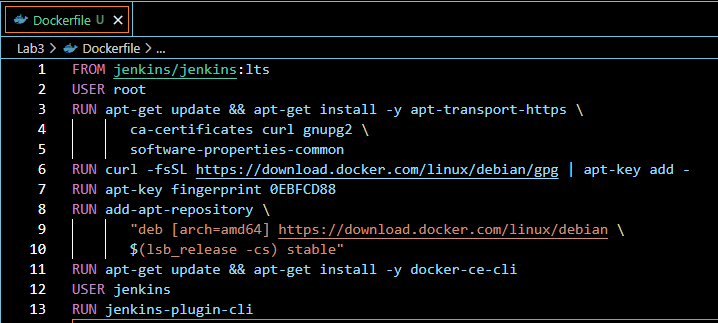
1. Запуск контейнера docker in docker:



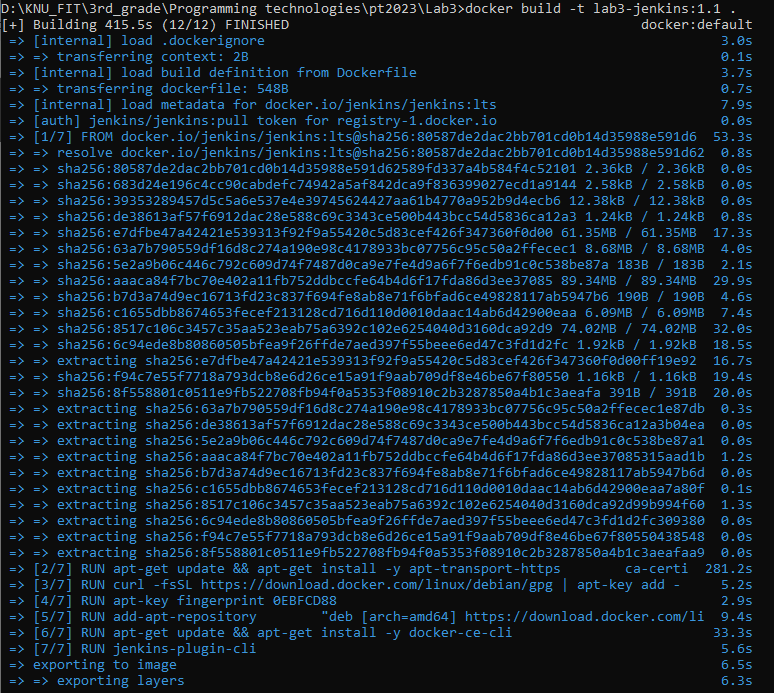


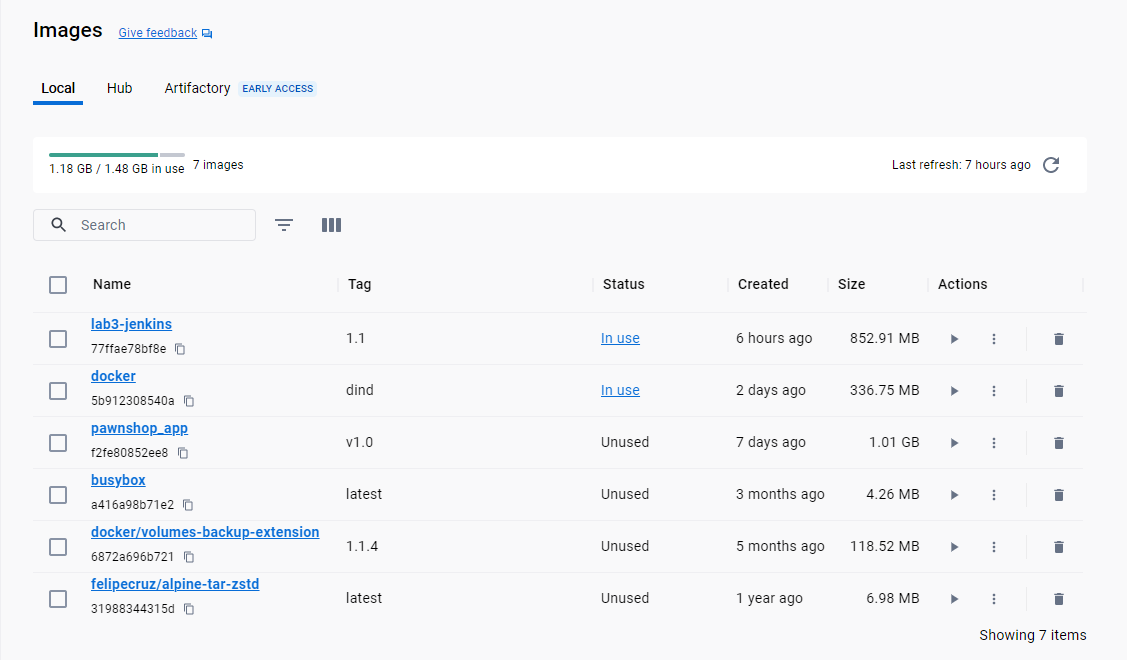


1. Створення Dockerfile:

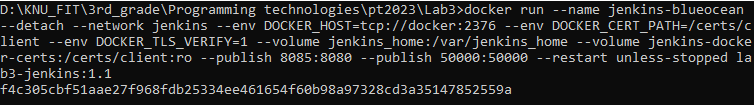


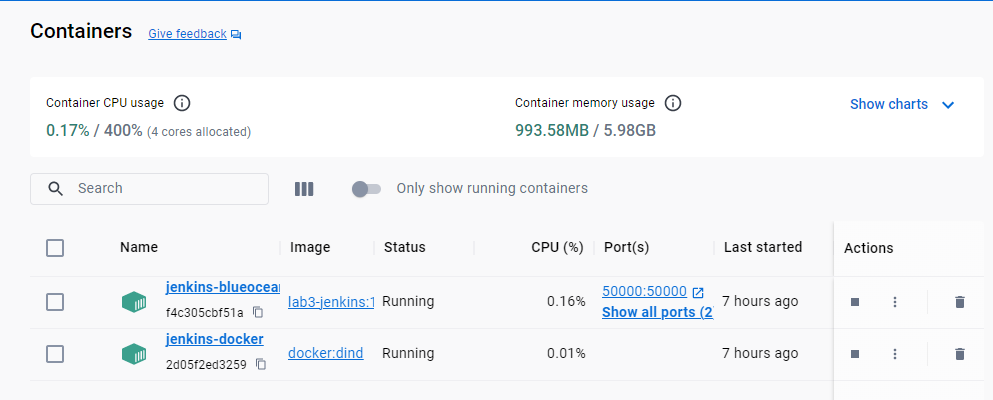
1. Створення docker-образу:



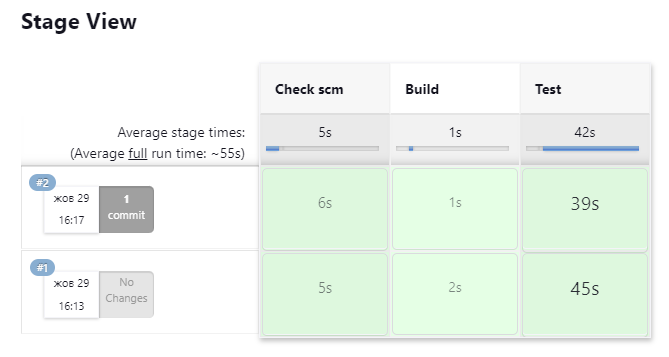


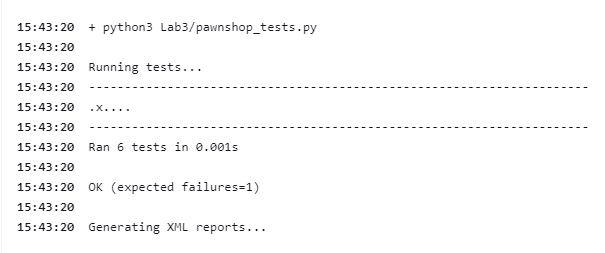
1. Запуск контейнера з Jenkins:





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

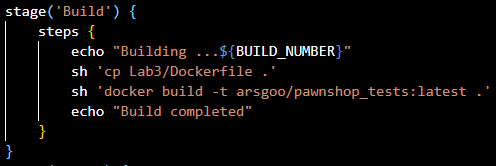


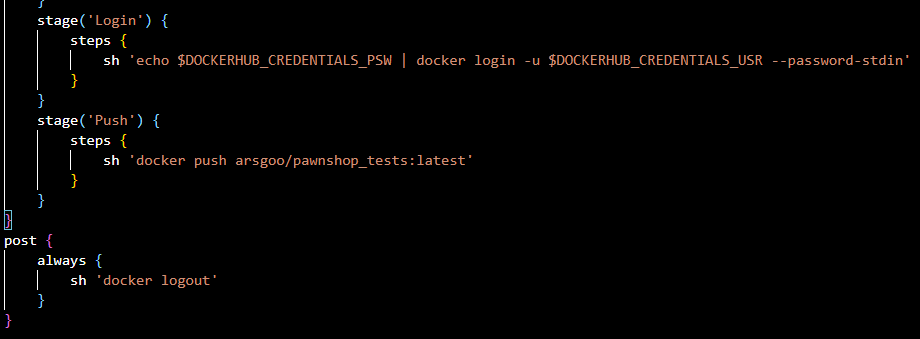


Всі етапи виконання конвеєра відображаються зеленим кольором, що свідчить про те, що конвеєр було успішно запущено і всі тести виконалися.

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

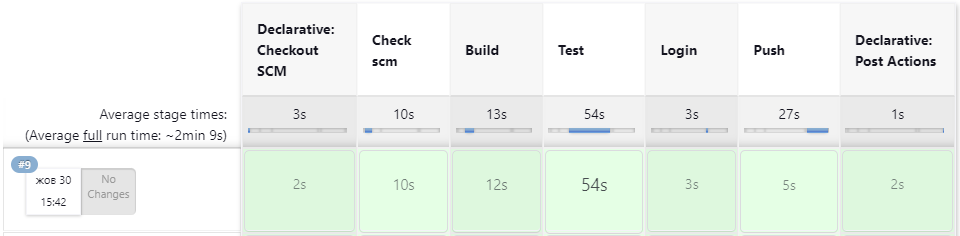


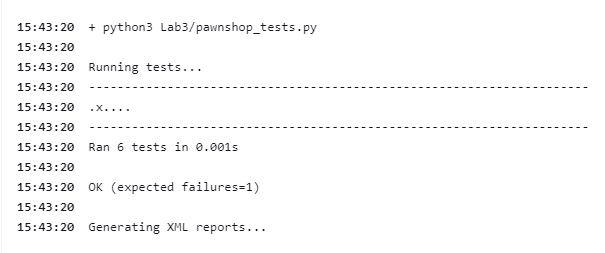




Ці фрагменти коду забезпечують автоматичне формування docker-образу arsgoo/pawnshop\_tests, вхід в акаунт Dockerhub, вивантаження туди даного образу і наостанок вихід з акаунту.

1. Результат запуску оновленого конвеєру:

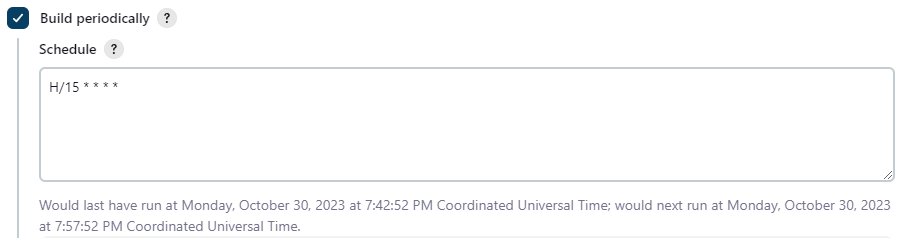


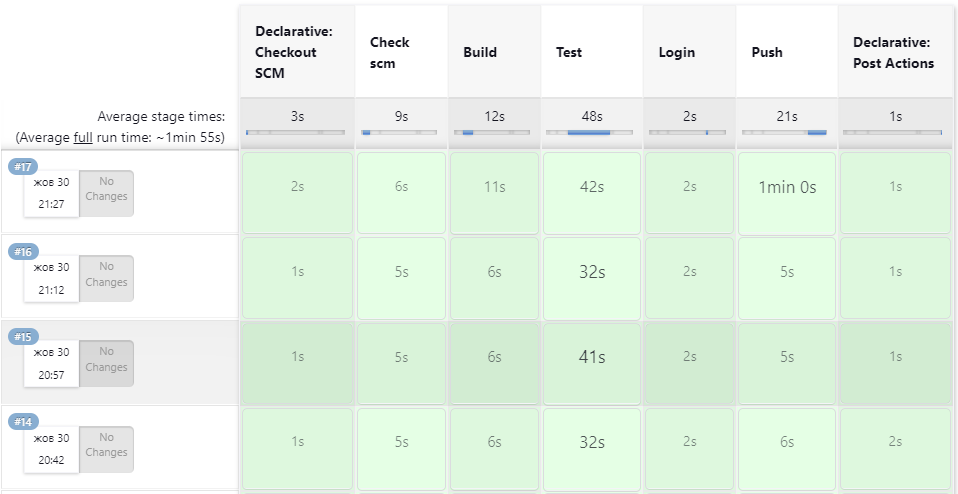


1. Дослідження тригерів, доступних у Jenkins:

* Build periodically

Автоматично перезапускає конвеєр через рівні проміжки часу.





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

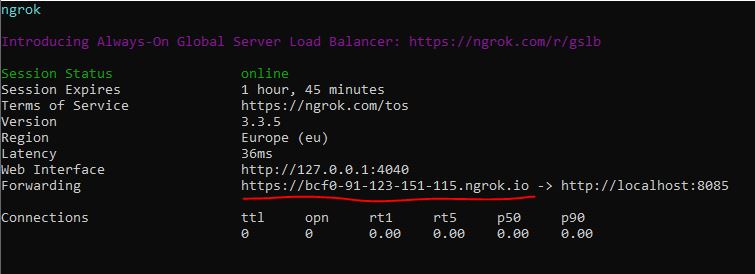
* Poll SCM

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.

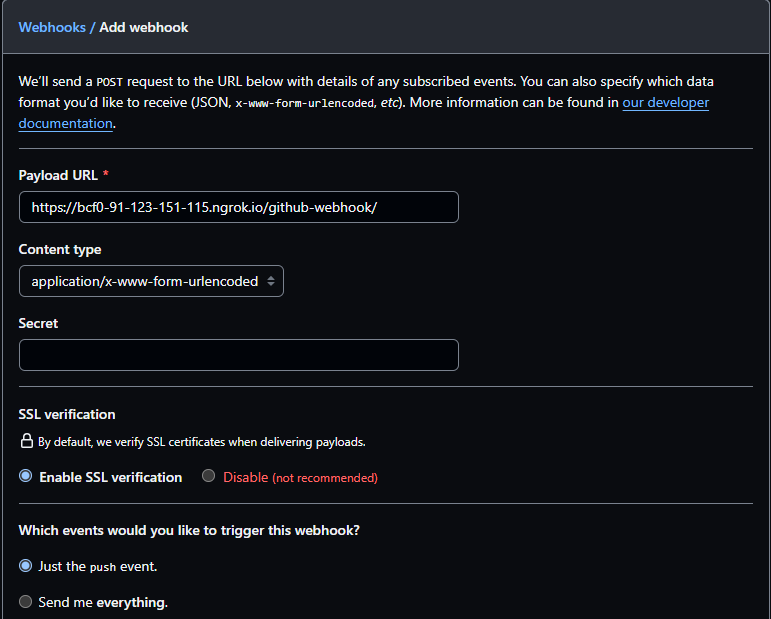
* GitHub hook

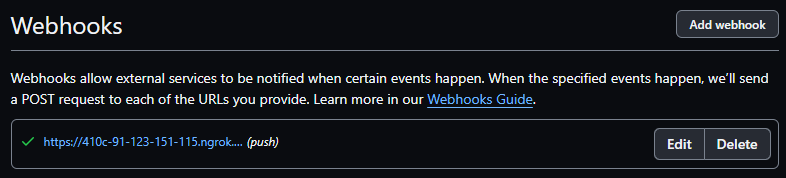
GitHub hook – це HTTP-запит, який GitHub надсилає Jenkins, коли відбувається певна подія в репозиторії (наприклад, push). Цей запит Jenkins якраз і використовує для запуску конвеєра.

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):

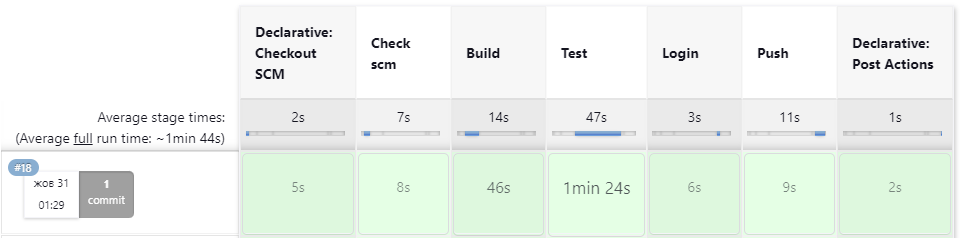


Далі копіюємо цю адресу, переходимо в потрібний GitHub репозиторій, обираємо Settings -> Webhooks -> Add webhook, і у поле Payload URL вставляємо посилання, додавши до нього /github-webhook/ (закриваючий слеш повинен бути обов’язково!):





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



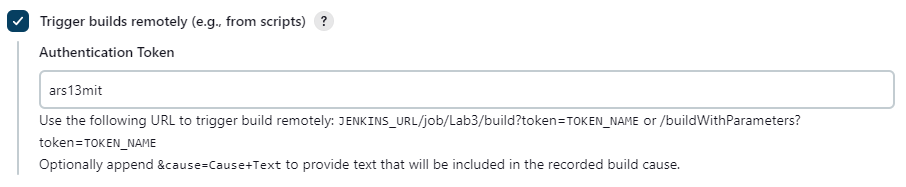
Так, дійсно, після виконання push Jenkins моментально виявив нові зміни у репозиторії і автоматично перезапустив конвеєр. Результат виконання, як і раніше, успішний.

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

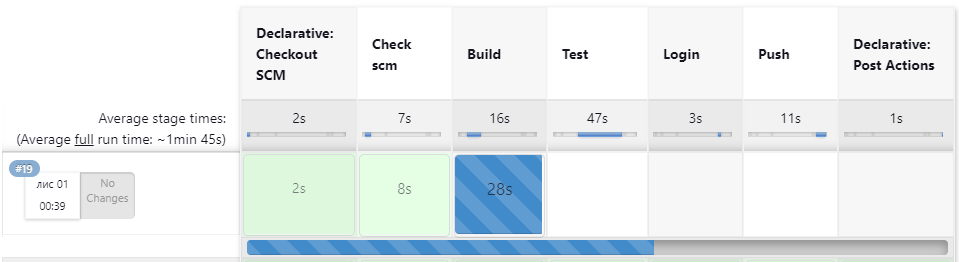
Цей тригер дозволяє віддалено ініціювати запуск конвеєра (наприклад, з іншого браузера або клієнта). Розглянемо два випадки запуску збірки: через нову вкладку того самого сеансу браузера і через інший браузер.

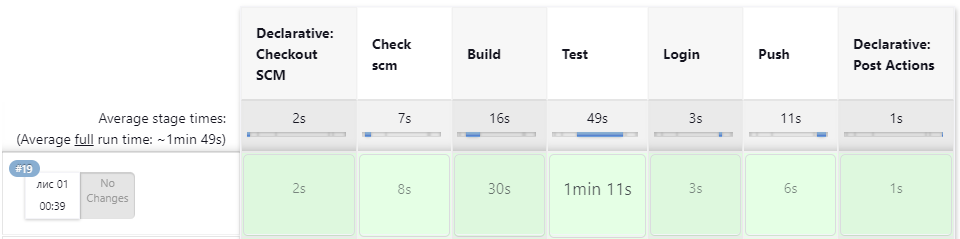
1. Нова вкладка

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):



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

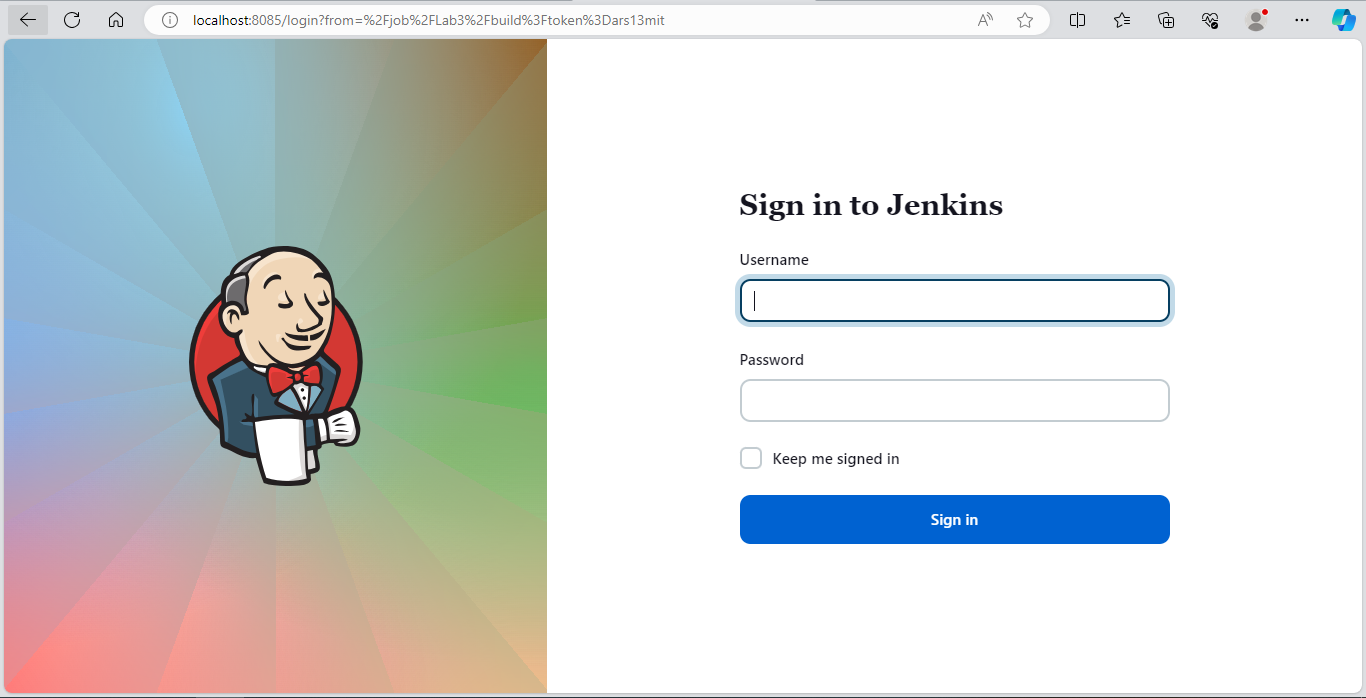




Jenkins миттєво отримав вказівку та ініціював запуск пайплайну. Результат знову успішний.

1. Інший браузер

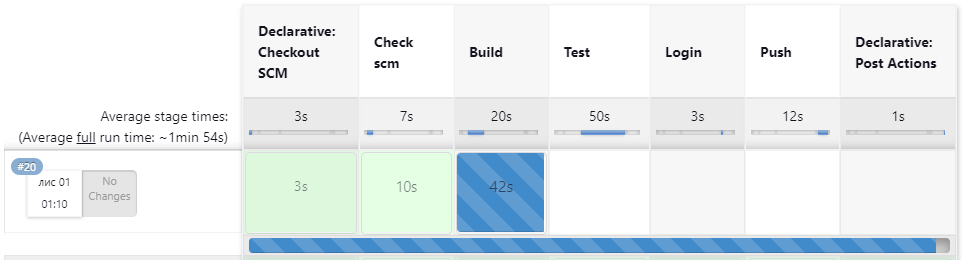
Спробуємо вставити теж саме посилання у вкладці іншого браузера:

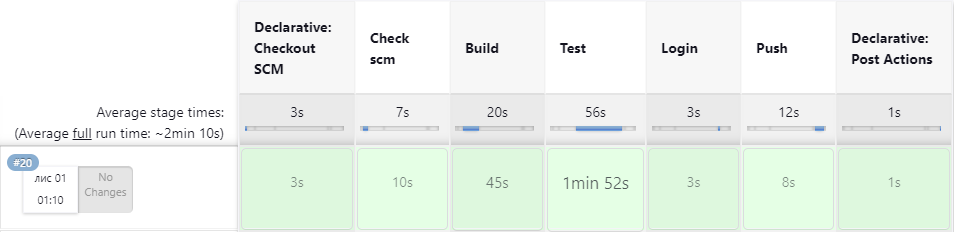


Бачимо вікно входу у Jenkins. Воно виникає через те, що надсилати такого роду запити може лише авторизована особа, яка має відповідні права. Проте може виникнути ситуація, коли в людини наприклад немає акаунту в Jenkins, а запустити конвеєр необхідно. В такому разі слід додатково встановити плагін **Build 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**](https://jenkins.example.com) – локальна адреса, за якою фукціонує Jenkins, **Secure-Token** – сформований нами токен, а **TestJob** – назва конвеєру (в моєму випадку це [http://localhost:8085](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.