Звіт

З практичної роботи №2

Студента групи МІТ-31

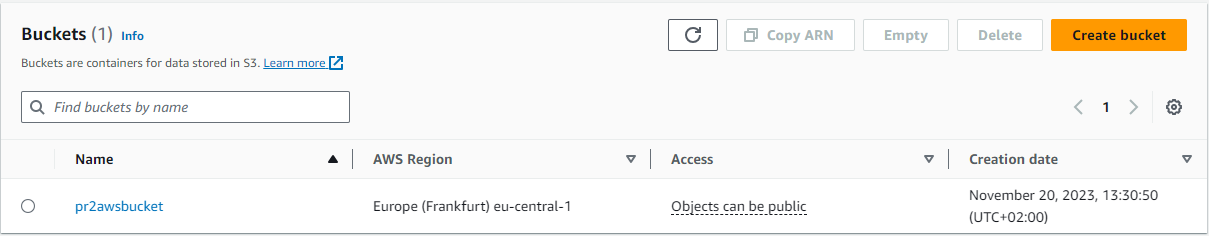
Добровольського Арсенія Михайловича

**Тема роботи:** Розгортання додатку на хмарній платформі AWS

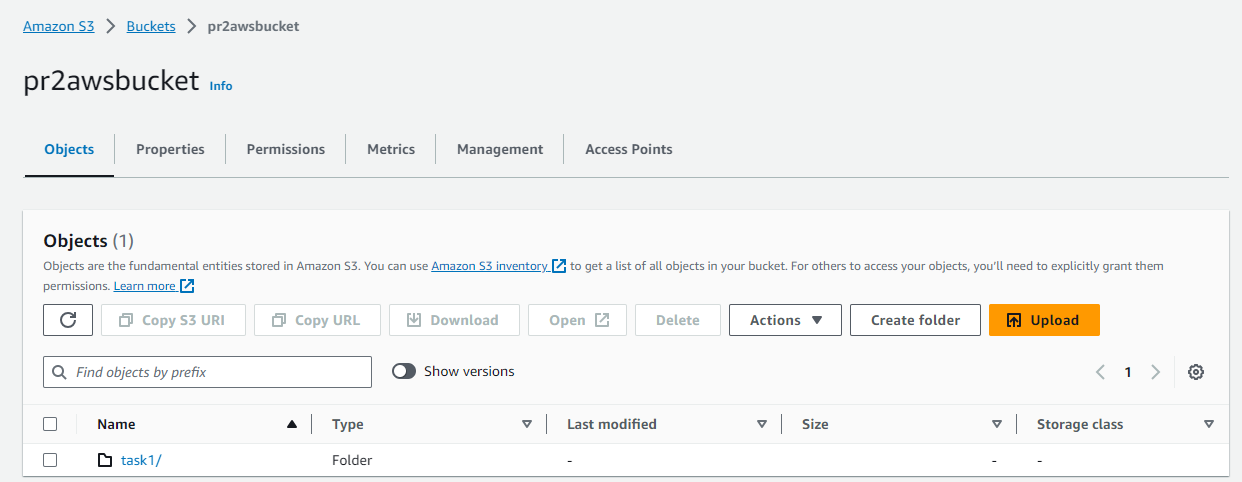
**Мета роботи:** ознайомитися із базовими сервісами AWS; навчитися розгортати додаток на хмарній платформі AWS.

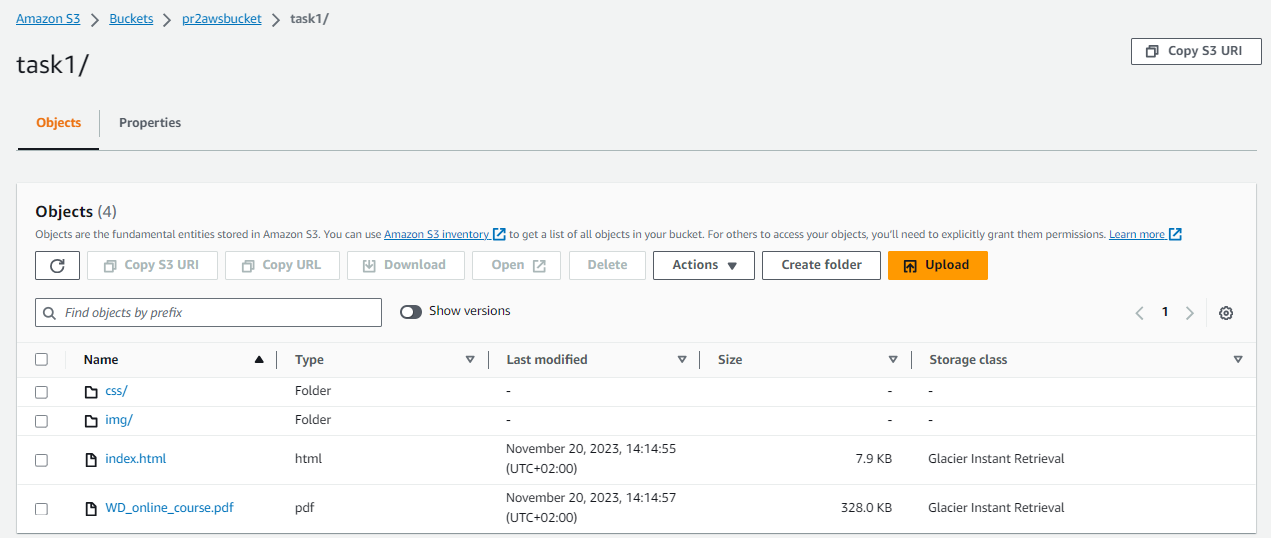
**Завдання 1 (AWS S3)**

Створення і перегляд кошика:

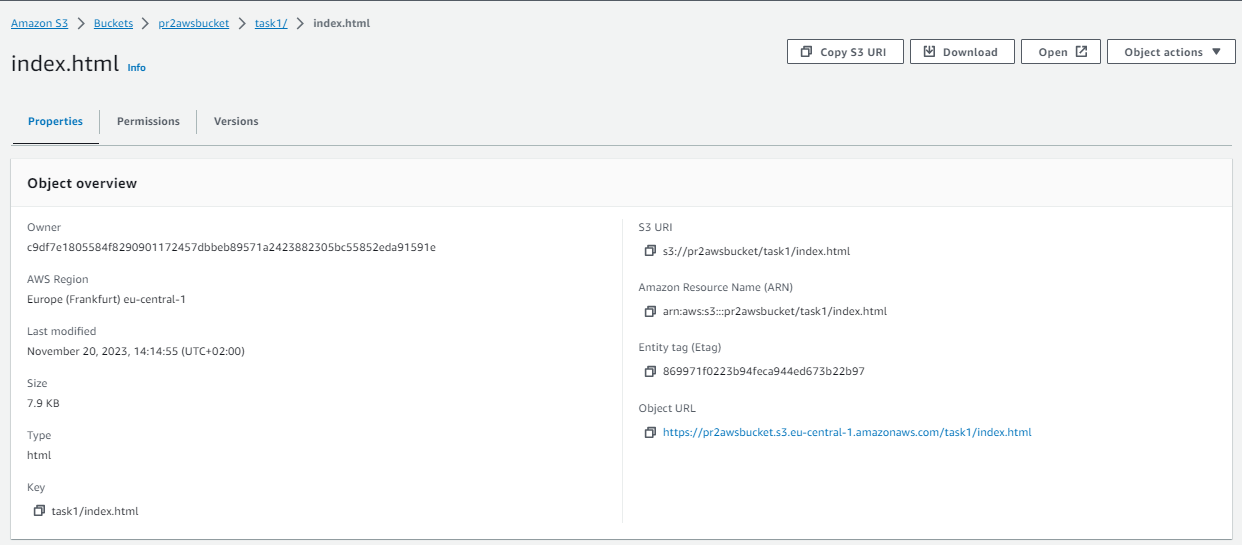


Створимо у цьому кошику папку task1, в яку завантажимо декілька файлів для статичного вебсайту (файли взято з напрацювань минулорічної дисципліни):

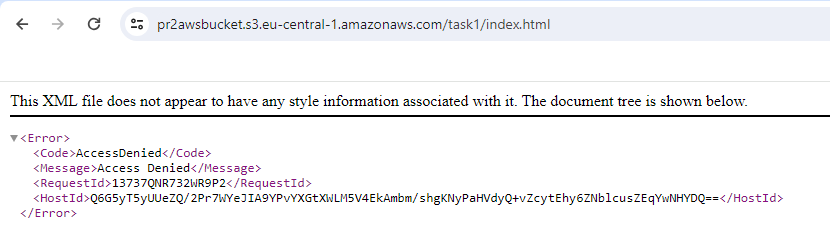




Переглянемо властивості одного з файлів (наприклад, index.html):



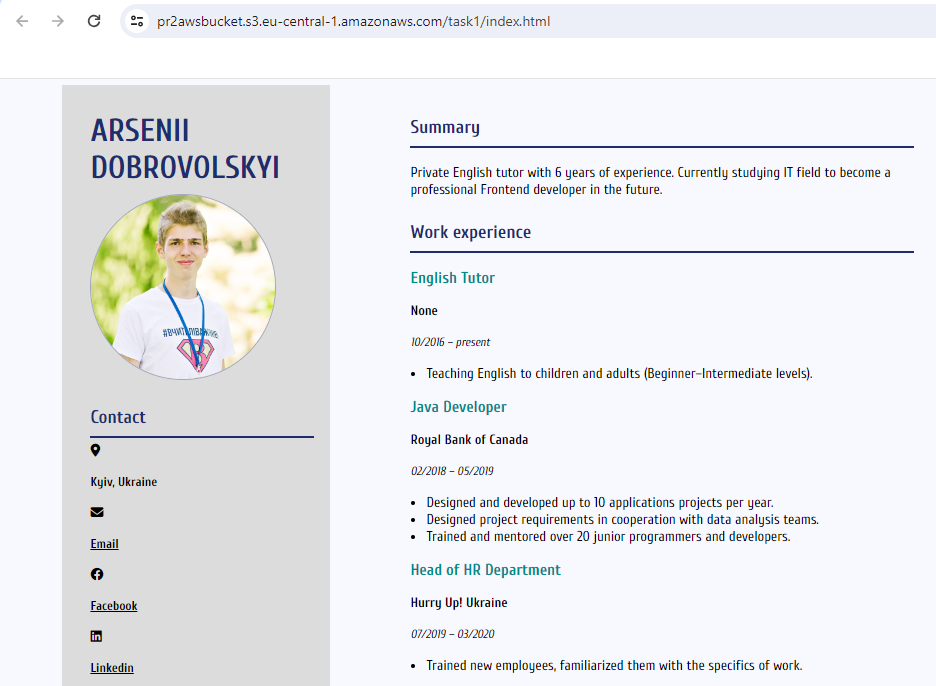
При спробі відкрити цей файл у браузері отримуємо повідомлення:



Для виправлення цієї помилки потрібно виконати 2 кроки:

1. У вікні кошика на вкладці **permissions** у секції **object ownership** дозволити використання ACL (ACLs enabled)
2. У вікні файлу в правому верхньому куті обрати **Object actions -> Make public using ACL** (якщо файлів декілька, це можна також зробити у вікні кошика, вибравши всі файли, потім **Actions -> Make public using ACL**)

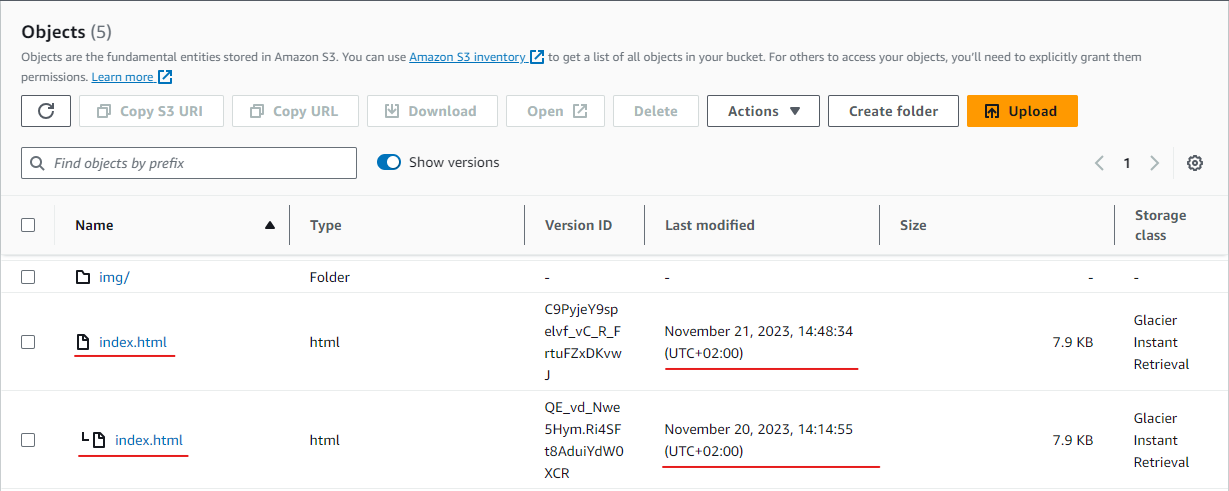
Виконавши ці кроки, вебсторінка справно завантажується за [посиланням](https://pr2awsbucket.s3.eu-central-1.amazonaws.com/task1/index.html):



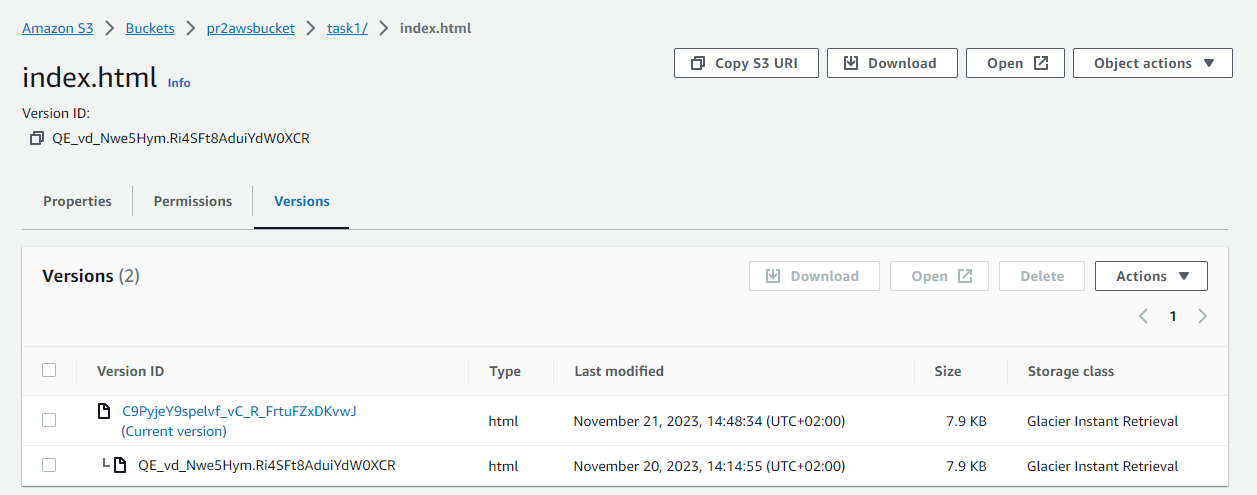
Дослідимо версіювання та міжрегіональну реплікацію.

***Версіювання*** дозволяє зберігати різні версії одного і того ж об'єкта в тому самому бакеті. Якщо версіювання увімкнено для бакету, кожен раз, коли відбуваються зміни в об’єкті, система зберігає нову версію цього об'єкта. Це неабияк стає в пригоді, коли виникає потреба відновити попередню версію файлу або відновитися після помилок чи неправильних змін.

Внесемо невеликі зміни у файл index.html і повторно вивантажимо його у наш бакет. Далі ввімкнемо повзунок **Show versions** для того, щоб побачити версії нашого файлу (підкреслені червоним):



Кожна версія об'єкта отримує унікальний ідентифікатор, який можна використовувати для відновлення конкретної версії за необхідності:



Для того, щоб «відкотитися» до попередньої версії файлу, AWS пропонує два підходи:

* Скопіювати попередню версію об’єкта в той самий бакет

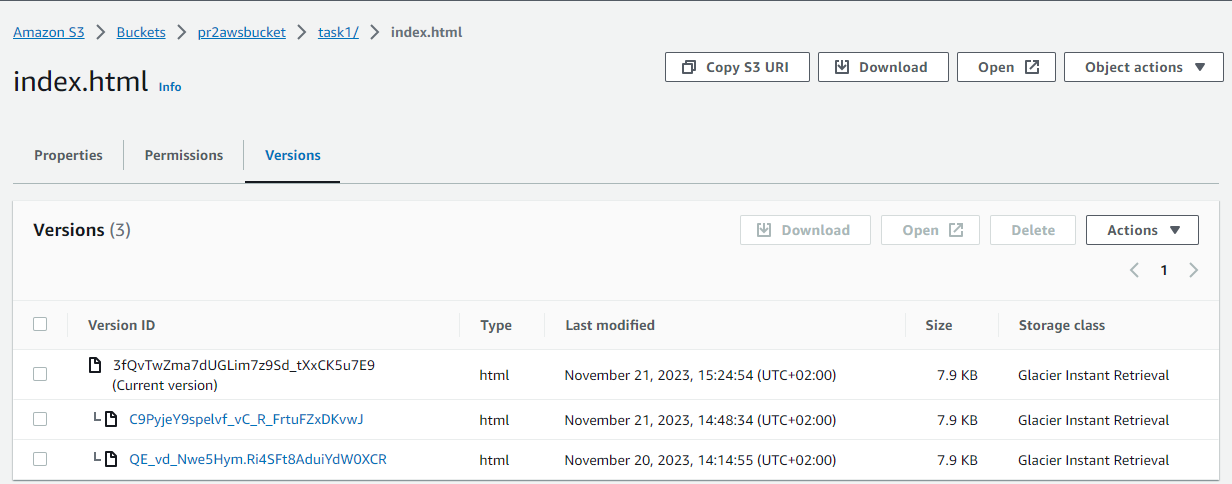
Скопійований об’єкт стає поточною версією цього об’єкта, при цьому всі версії об’єкта зберігаються.

* Назавжди видалити поточну версію об'єкта

При видаленні поточної версії об’єкта, попередня версія цього об’єкта фактично стає поточною.

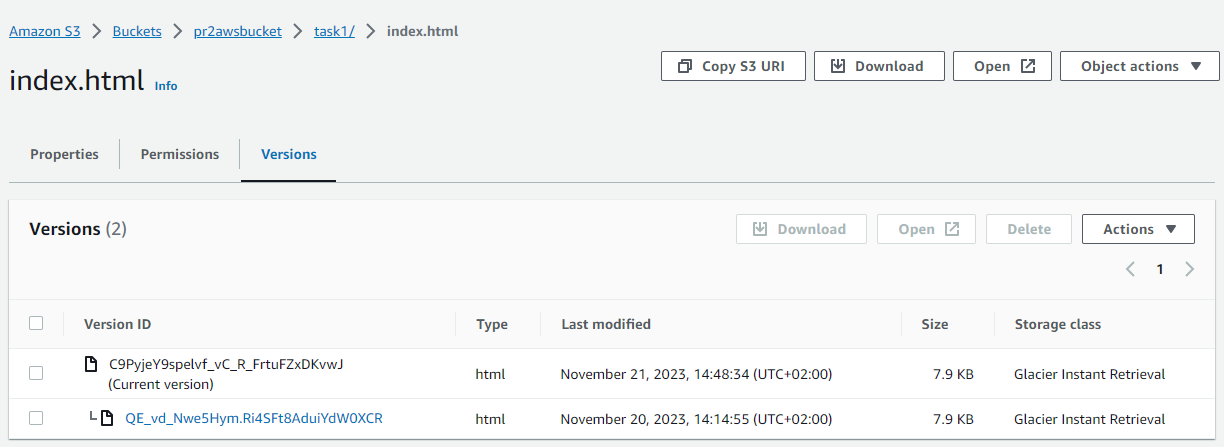
Розглянемо обидва способи.

Для реалізації першого завантажимо попередню версію файлу і знову вивантажимо її у бакет:



AWS присвоїв новий ідентифікатор вивантаженій версії, і вона стала поточною.

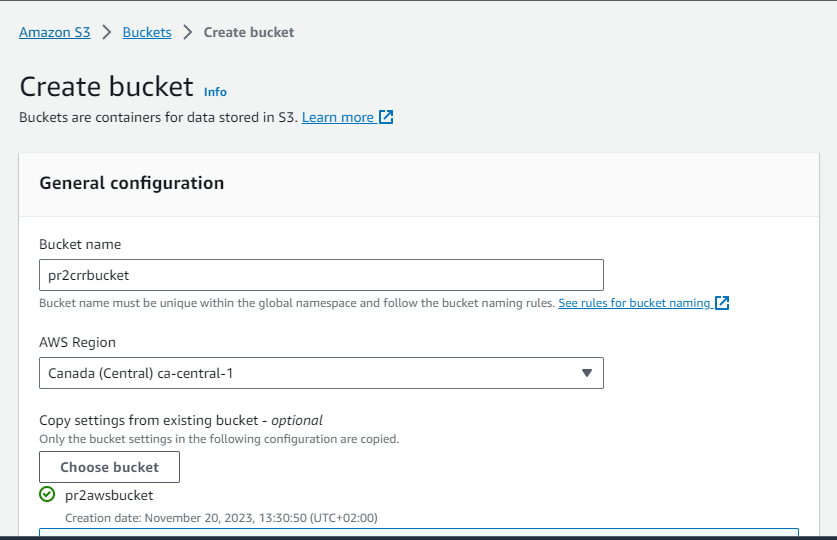
Видалити версію можливо, вибравши checkbox навпроти потрібної версії і натиснувши кнопку **Delete** праворуч вгорі.



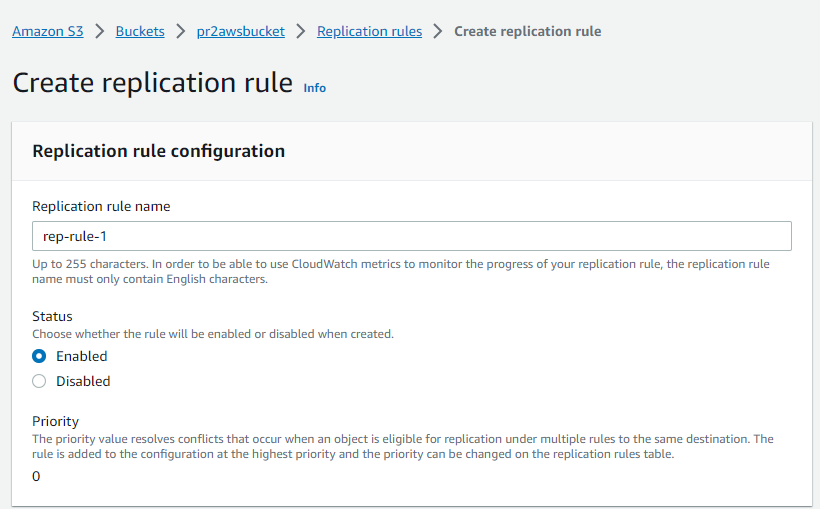
Видаливши останню версію, попередня знову стала поточною.

***Міжрегіональна реплікація*** дозволяє автоматично копіювати об'єкти з одного бакету в одному регіоні в інший бакет в іншому регіоні. Ця функція створена для забезпечення резервного копіювання у випадку втрати даних або регіону, відновлення даних та забезпечення вищого рівня доступності та відмовостійкості.

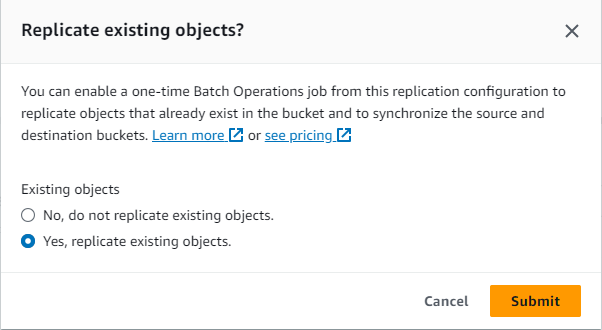
Створимо новий бакет у регіоні, відмінному від регіону першого бакета:



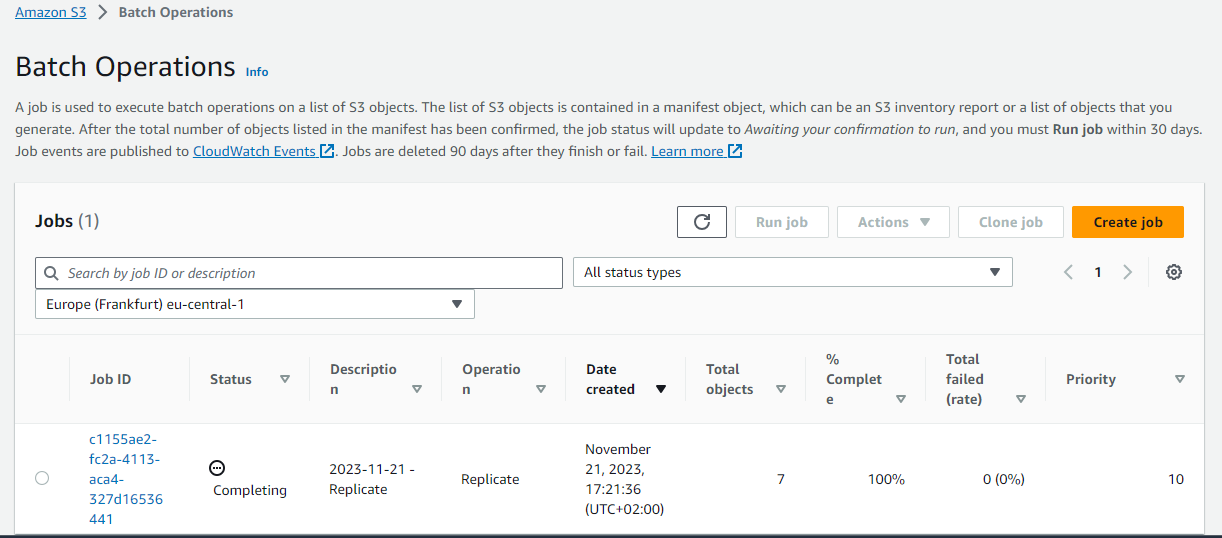
Для того, щоб файли могли копіюватися з першого бакета в другий, потрібно у першому створити ***правило реплікації***. Для цього переходимо у перший бакет, обираємо вкладку **Management**, в ній розділ **Replication rules -> Create replication rule**:

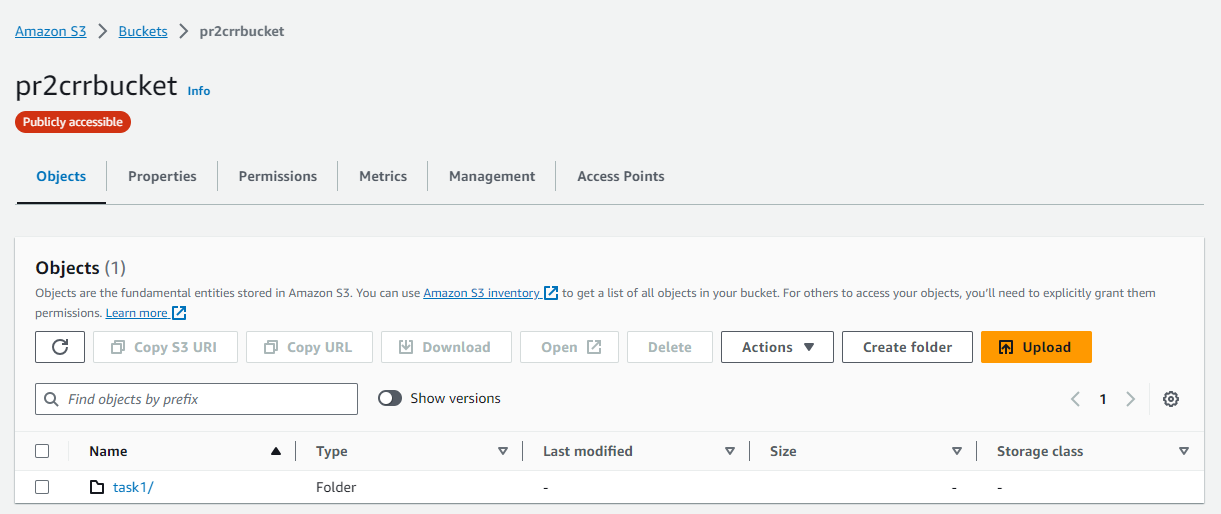


При створенні правила з’явиться вікно із запитанням, чи потрібно скопіювати вже наявні об’єкти у першому бакеті в другий. Обираємо варіант «Так»:

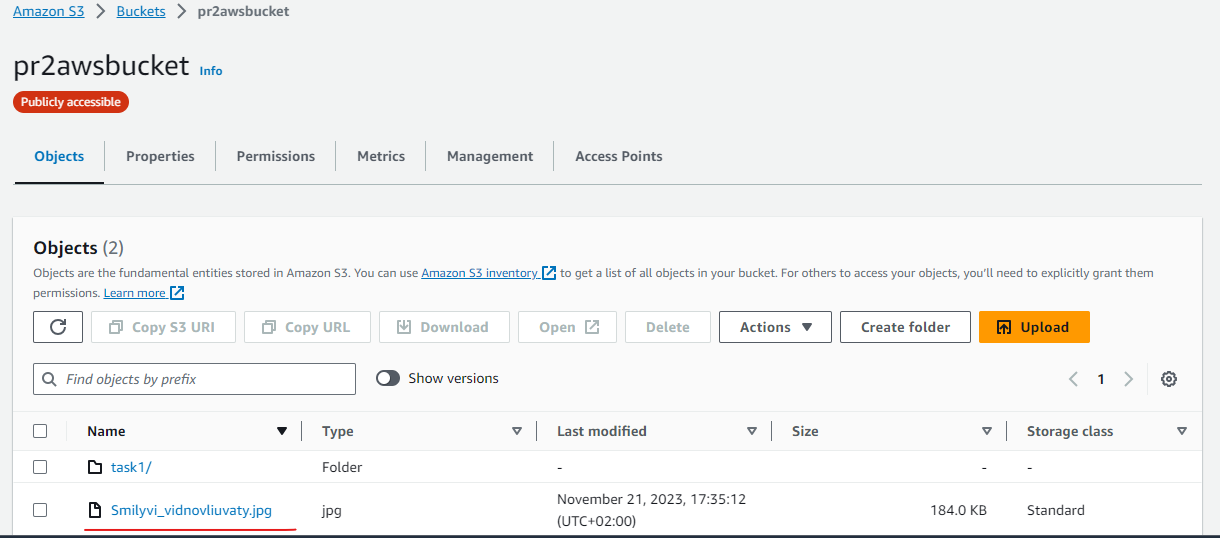


Копіювання файлів займає деякий час, це нормально:

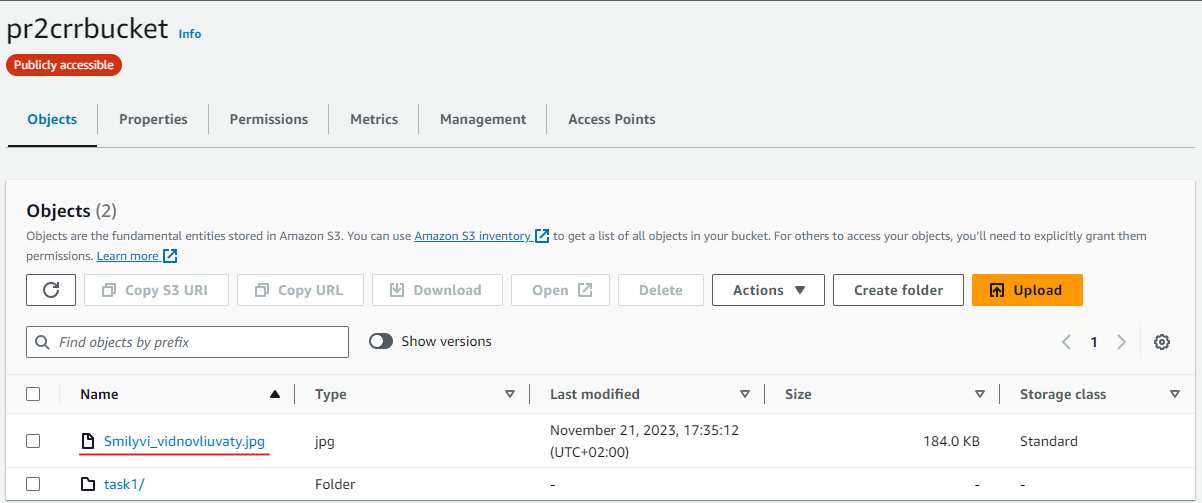




Додамо новий файл у перший бакет, з якого відбувається реплікація об’єктів (підкреслений червоним):

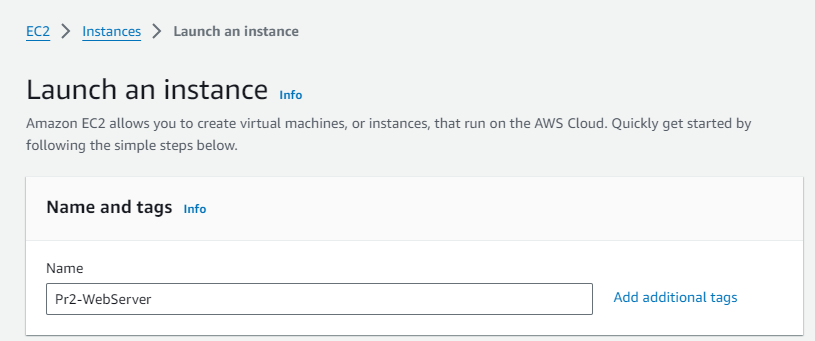


Переходимо у новостворений бакет і бачимо, що там файл також з'явився:

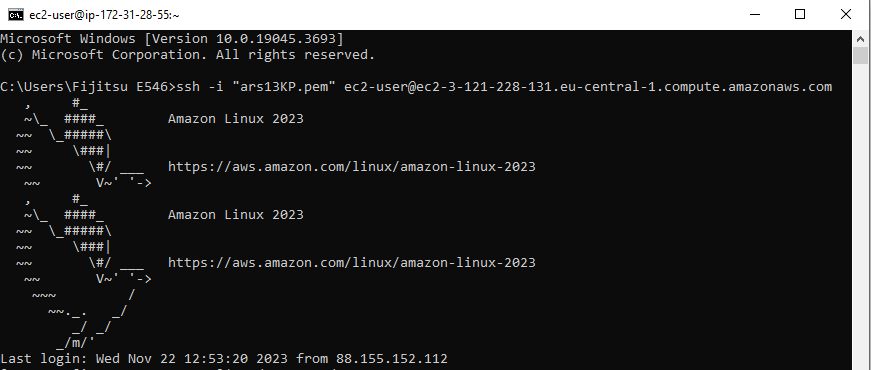


**Завдання 2 (EC2)**

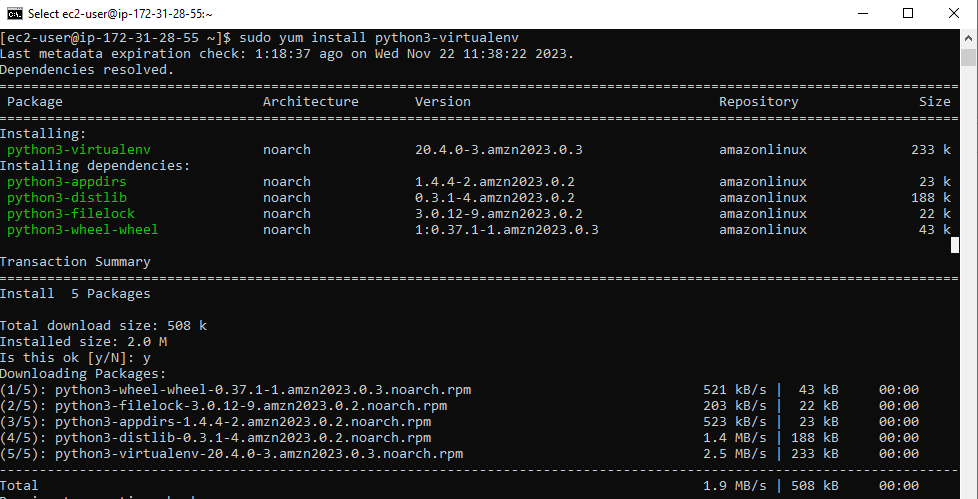
Для початку створимо EC2 Instance:



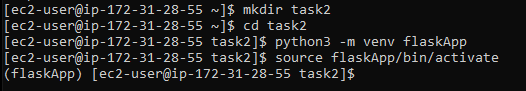
Під'єднаємося до створеного EC2 Instance через SHH. Для цього відкриваємо командний рядок і вводимо там таку команду:



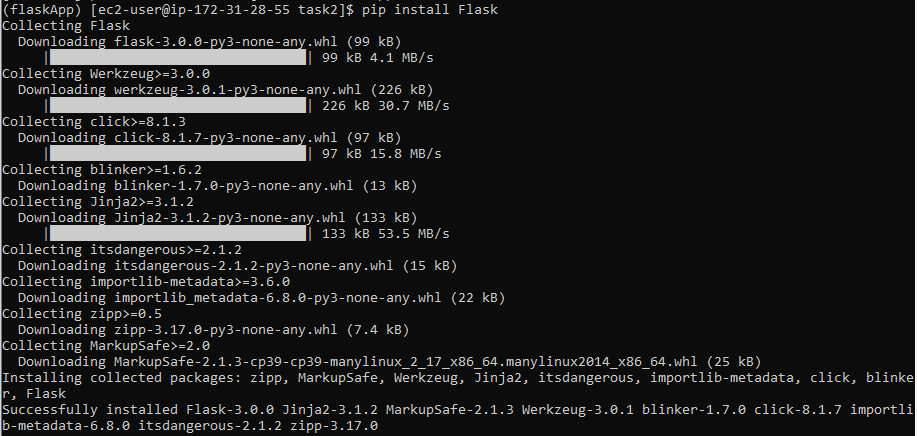
Далі встановлюємо віртуальне середовище для Python:



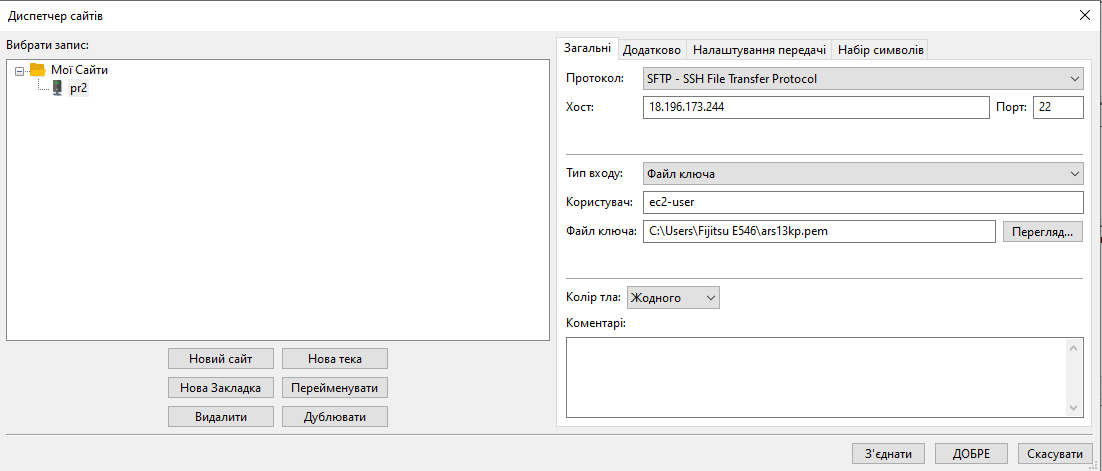
Створюємо папку **task2**, в якій розгортаємо віртуальне середовище **flaskApp** і активуємо його:

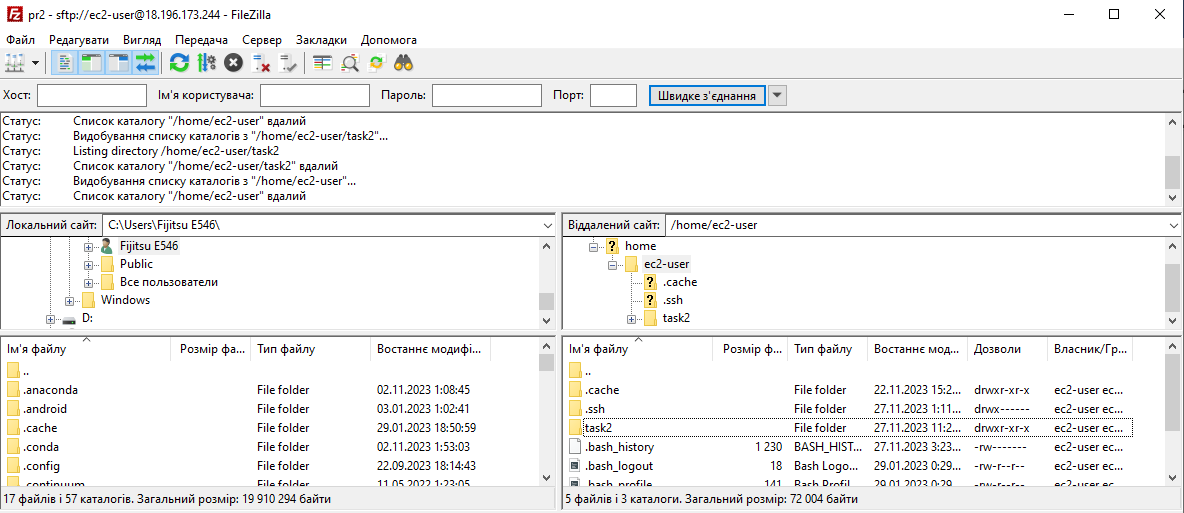


Встановлюємо Flask:

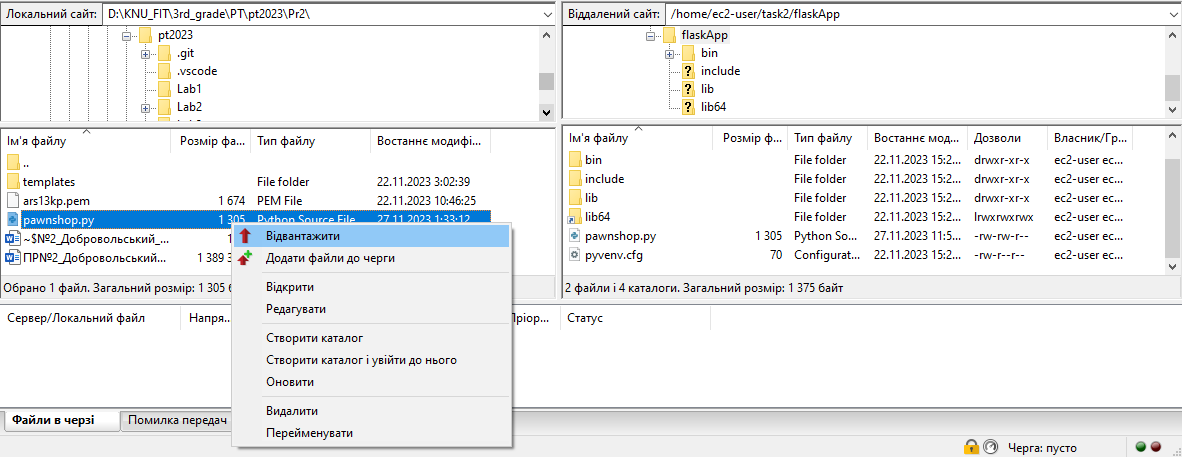


Тепер необхідно скопіювати файл з Flask-застосунком до EC2 Instance. Для цього скористаємося програмою FileZilla. У верхньому лівому куті обираємо **Файл -> Диспетчер сайтів** і створюємо новий сайт, вказавши необхідні параметри (наводяться на скриншоті):

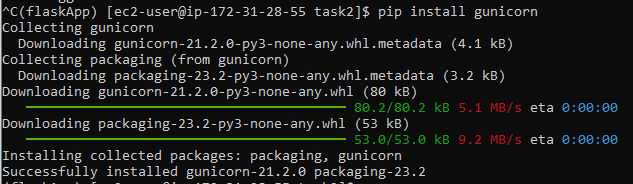




Вдало під’єднавшись до EC2 Instance (праворуч), скопіюємо наш файл до його сховища. Обираємо потрібний файл і натискаємо кнопку «Відвантажити»:

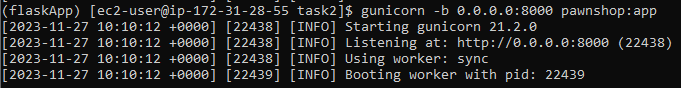


Встановлюємо Gunicorn:



Gunicorn – це популярний HTTP-сервер WSGI для програм Python. Він слугуватиме мостом між додатком Flask та Інтернетом.

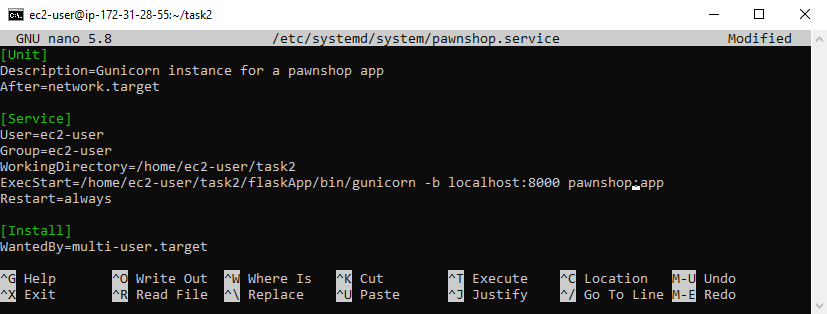
Перевірка роботи програми локально з Gunicorn:



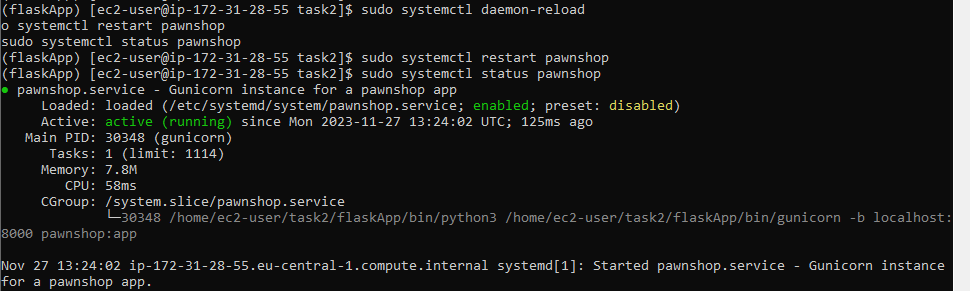
Для синхронної роботи Gunicorn та EC2 Instance скористаємося менеджером ініціалізації та управління процесами Systemd. У папці **/etc/systemd/system** необхідно створити файл з розширенням **.service**, в якому вказуємо, як має поводитися gunicorn у разі перезавантаження системи. Цей файл міститиме 3 частини:

* Unit – для опису проєкту та основних залежностей служби
* Service – для визначення користувача/групи, від імені якого/якої відбуватиметься запуск служби. Сюди також входять команда для запуску, змінні середовища, права доступу тощо
* Install – за допомогою спеціальних інструкцій повідомляє systemd, у який момент під час процесу завантаження служба повинна запускатися

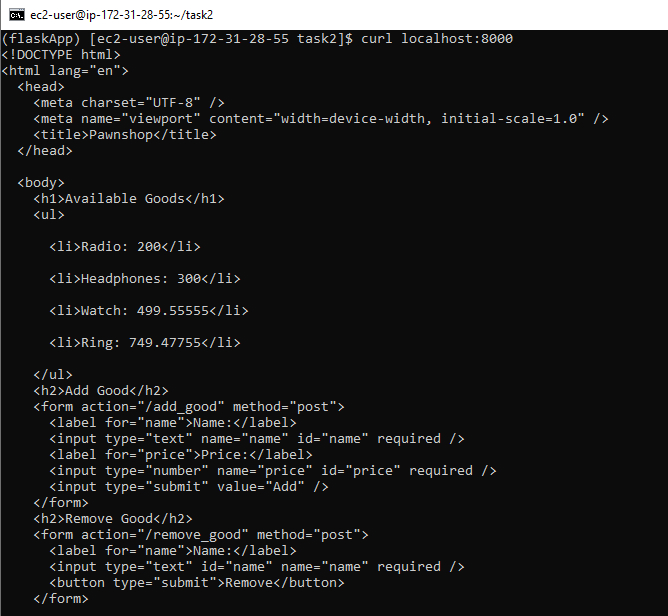
Текст файлу такий:



Запускаємо службу та перевіряємо її статус:

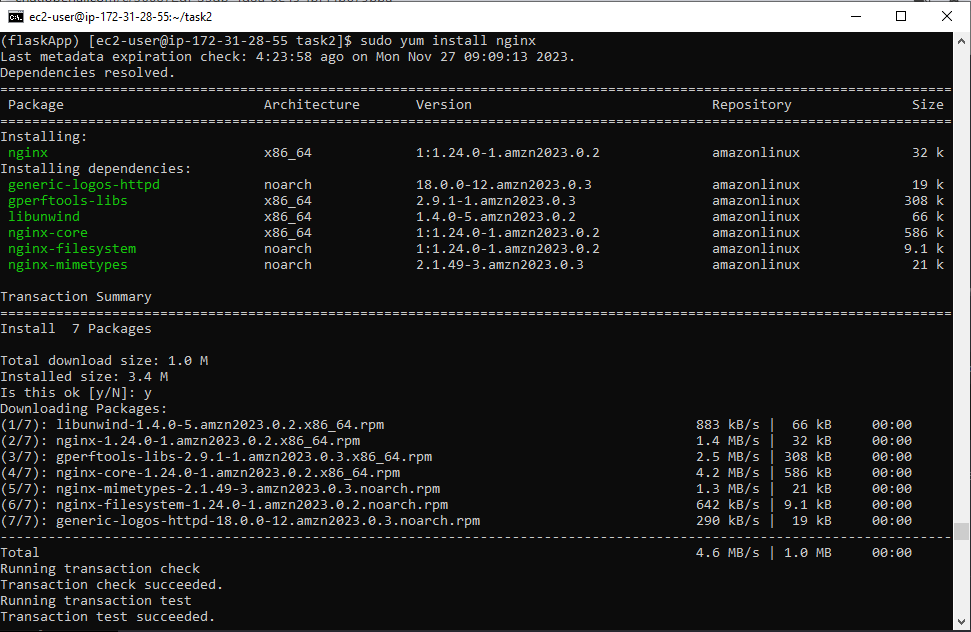


Перевірка працездатності застосунку:



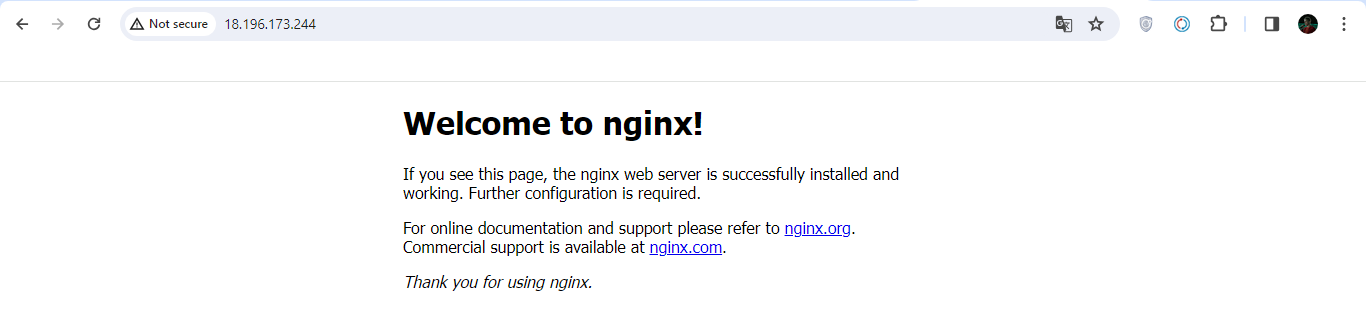
Зокрема, необхідно використати Nginx Webserver, який прийматиме запити від користувача та направлятиме їх до Gunicorn.

Спершу встановлюємо Nginx, який діятиме як зворотний проксі для Flask-застосунку:

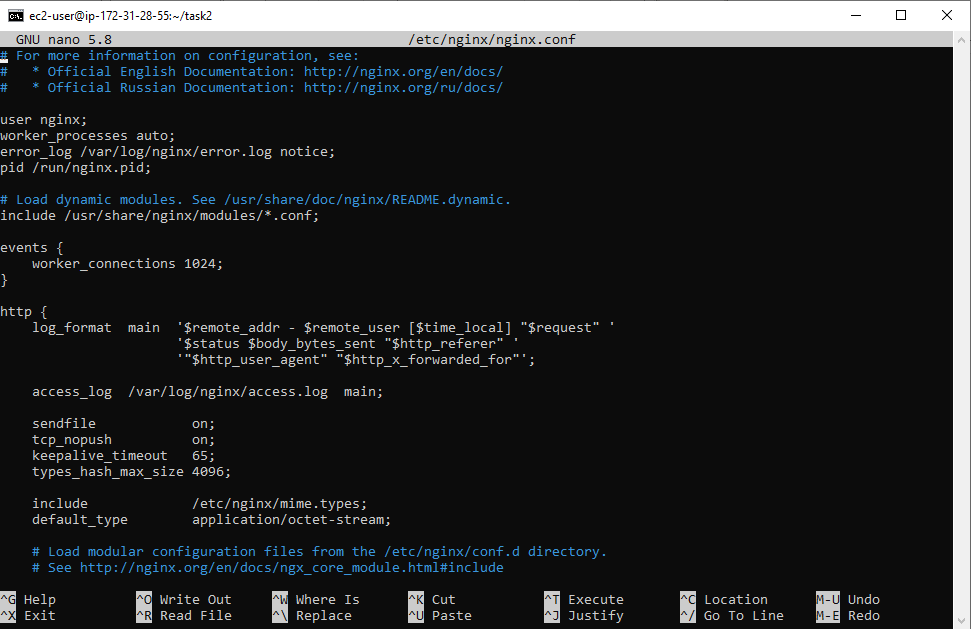


Запускаємо службу Nginx і у браузері переходимо за публічною IP-адресою нашого EC2 Instance для того, щоб побачити сторінку Nginx за замовчуванням:

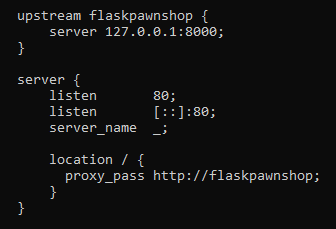




Створимо файл конфігурації Nginx. Для цього вводимо команду **sudo nano /etc/nginx/nginx.conf**, після чого потрапляємо у таке вікно:



Вносимо необхідні зміни і зберігаємо файл:



Наостанок у вікні браузера переходимо за [публічною IP-адресою](http://18.196.173.244/) EC2 Instance і бачимо успішно розгорнутий застосунок:



**Висновок:** в ході виконання практичної роботи було ретельно опрацьовано два базових сервіси AWS, а саме S3 та EC2, і розглянуто такі поняття, як версіювання, міжрегіональна реплікація, EC2 Instance та ін. Насамкінець, було реалізовано розгортання Flask-застосунку на хмарній платформі AWS. Для виконання практичної роботи було залучено додаткове програмне забезпечення FileZilla, за допомогою якого значно спростився процес обміну файлами між локальним (комп’ютер) та віддаленим (EC2 Instance) сховищами даних.