Звіт

З лабораторної роботи №4 та практичної роботи №3

Студента групи МІТ-31

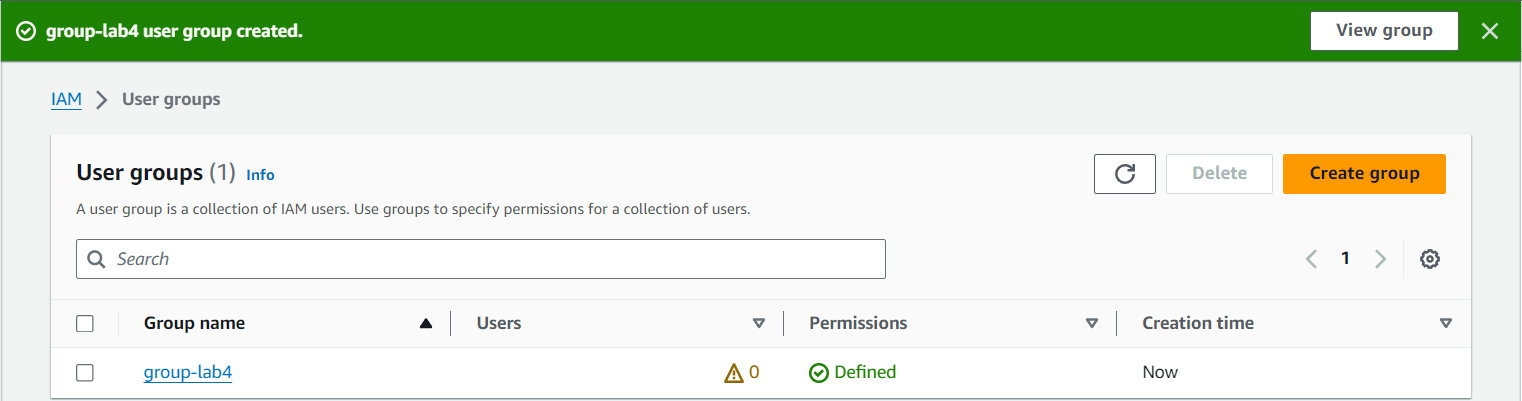
Добровольського Арсенія Михайловича

**Тема роботи:** Інфраструктура як код (IaC)

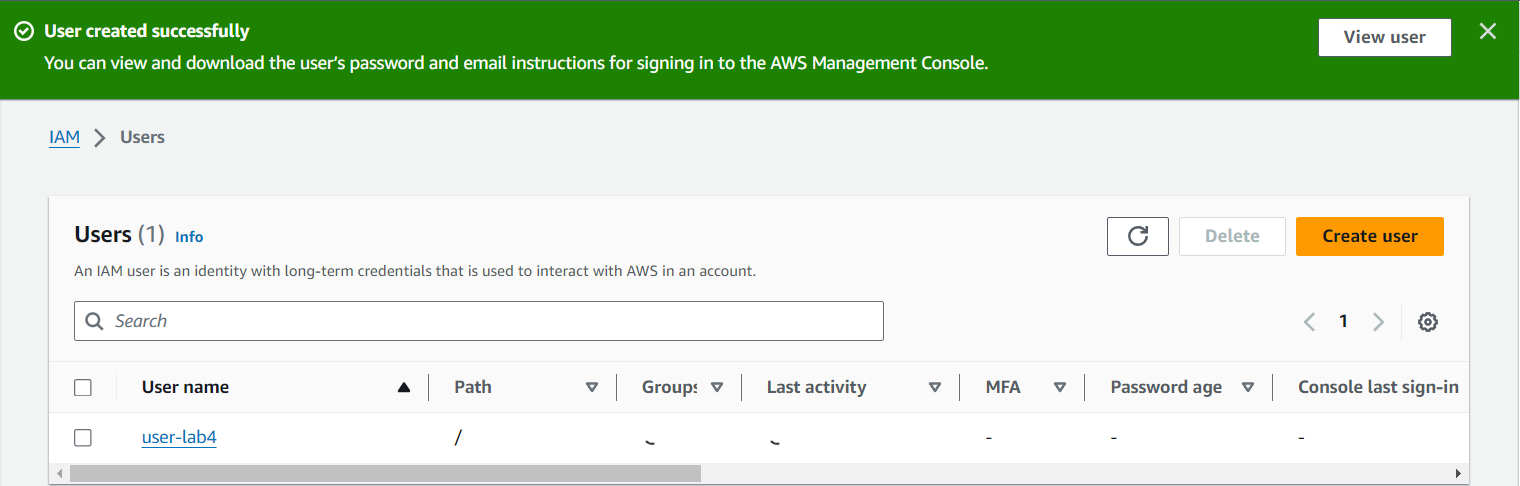
**Мета роботи:** розглянути поняття та принципи організації інфраструктури як коду; навчитися використовувати Terraform для розгортання інфраструктури.

**Завдання 1 (створення базового робочого процесу із використанням підходу Infrastructure as Code)**

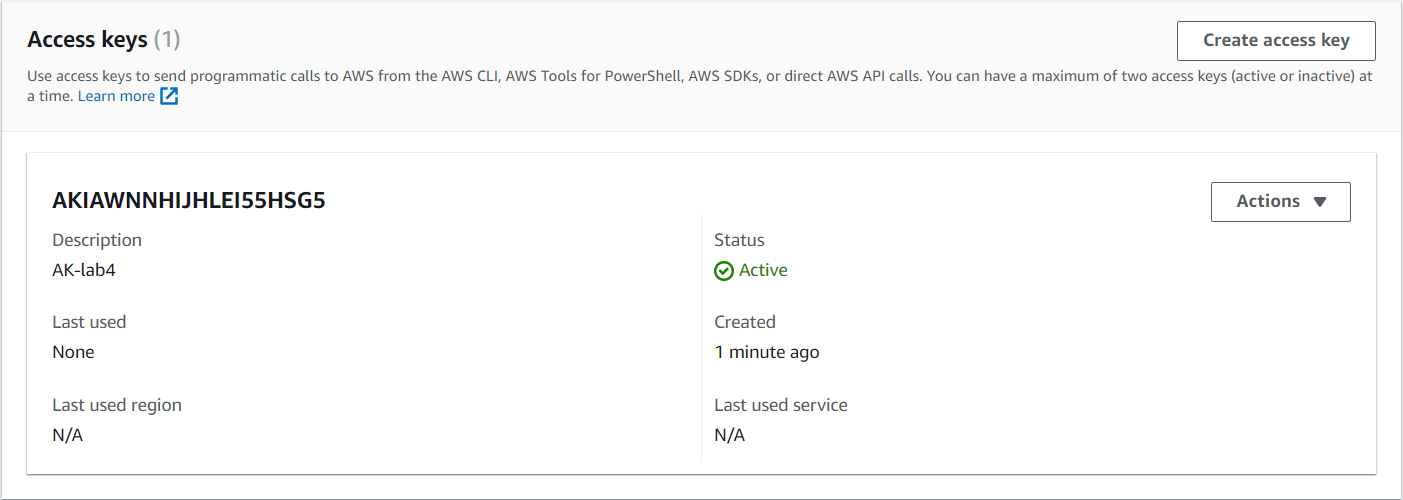
Створюю новий user group:



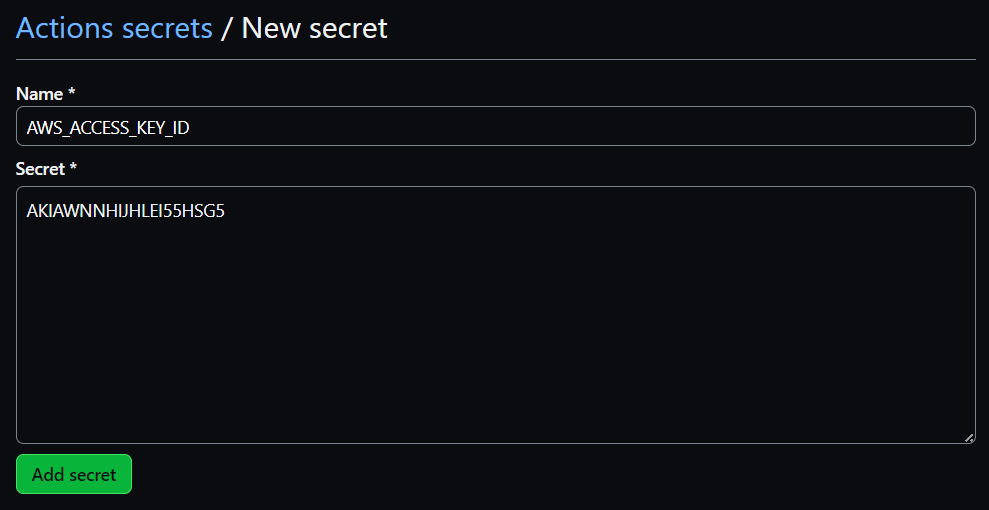
Створюю користувача, якого додаю до новоствореної групи користувачів:

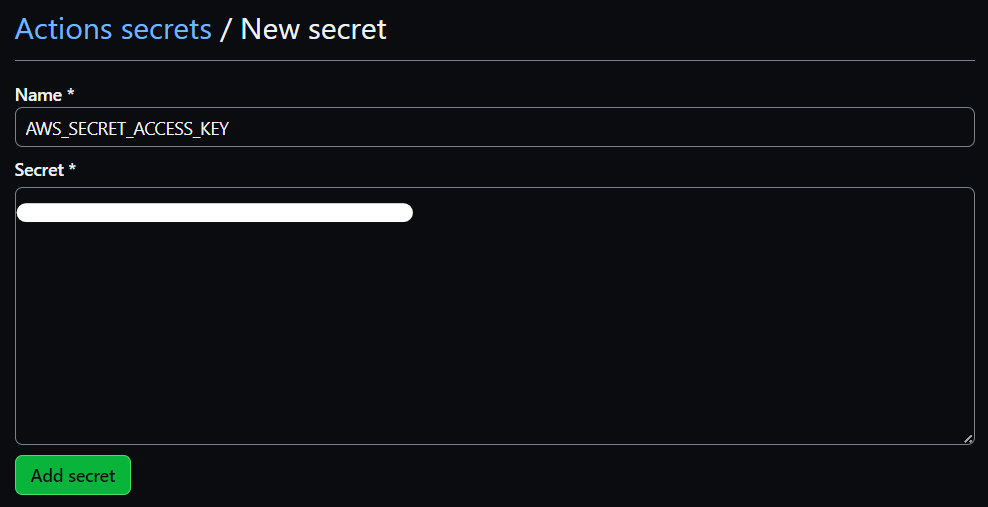


Для того, щоб можна було підключатися під цим користувачем з GitHub, створюю ключ доступу для нього:

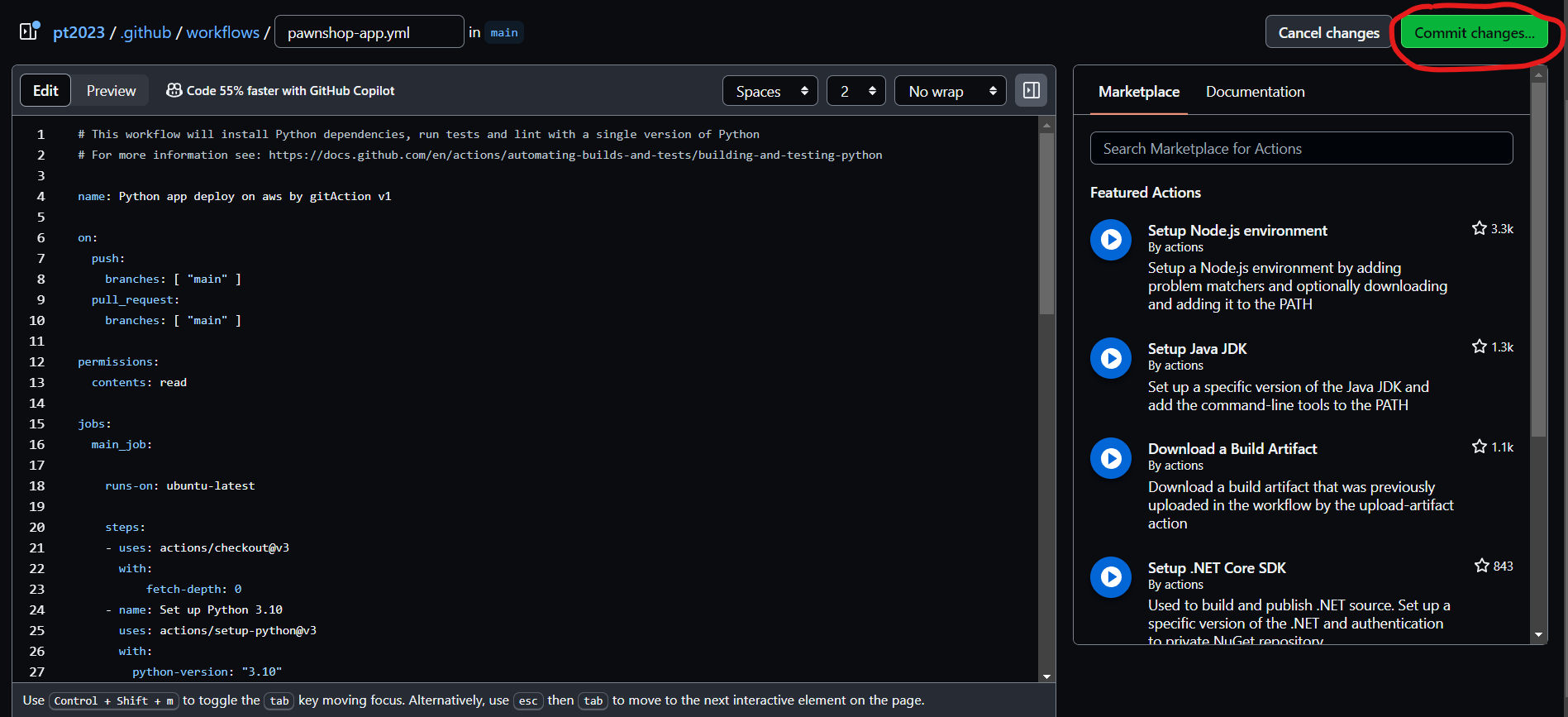


Переходжу на GitHub, в обраному репозиторії натискаю **Settings -> Secrets and variables -> Actions** **-> New repository secret**. У розгорнутому вікні по черзі записую обидві частини свого ключа доступу:

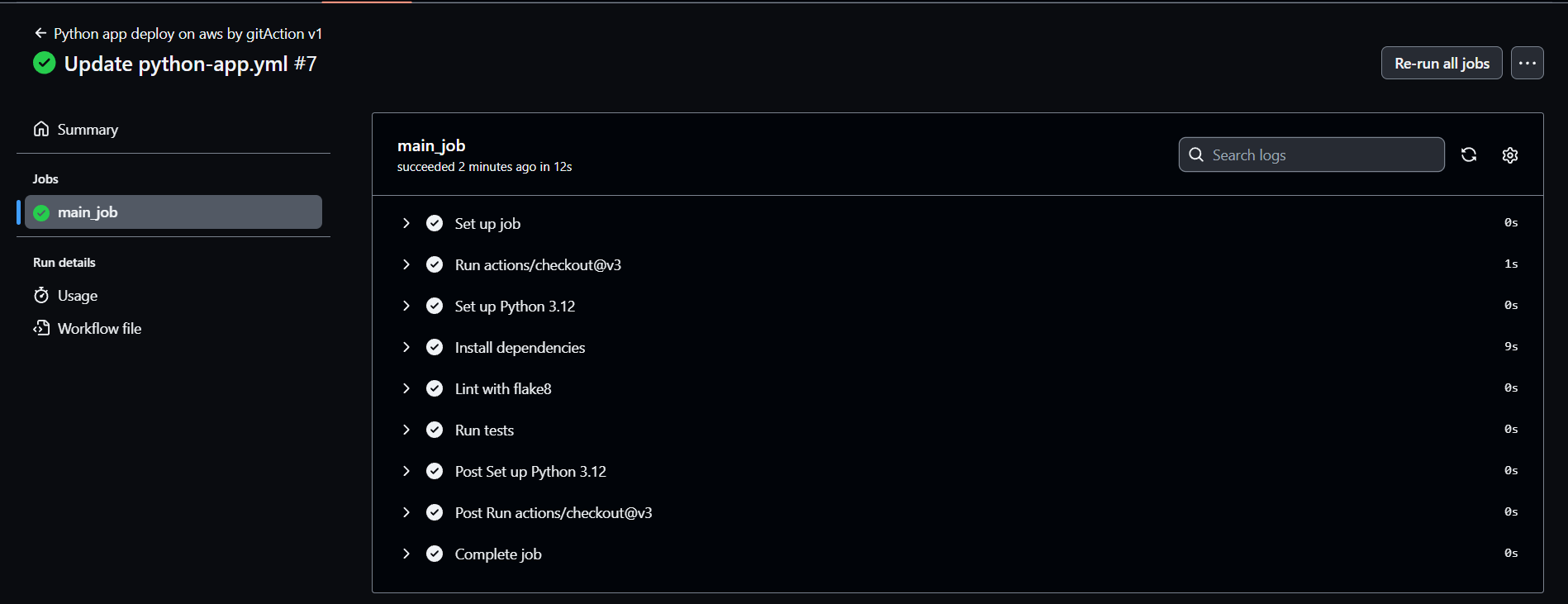




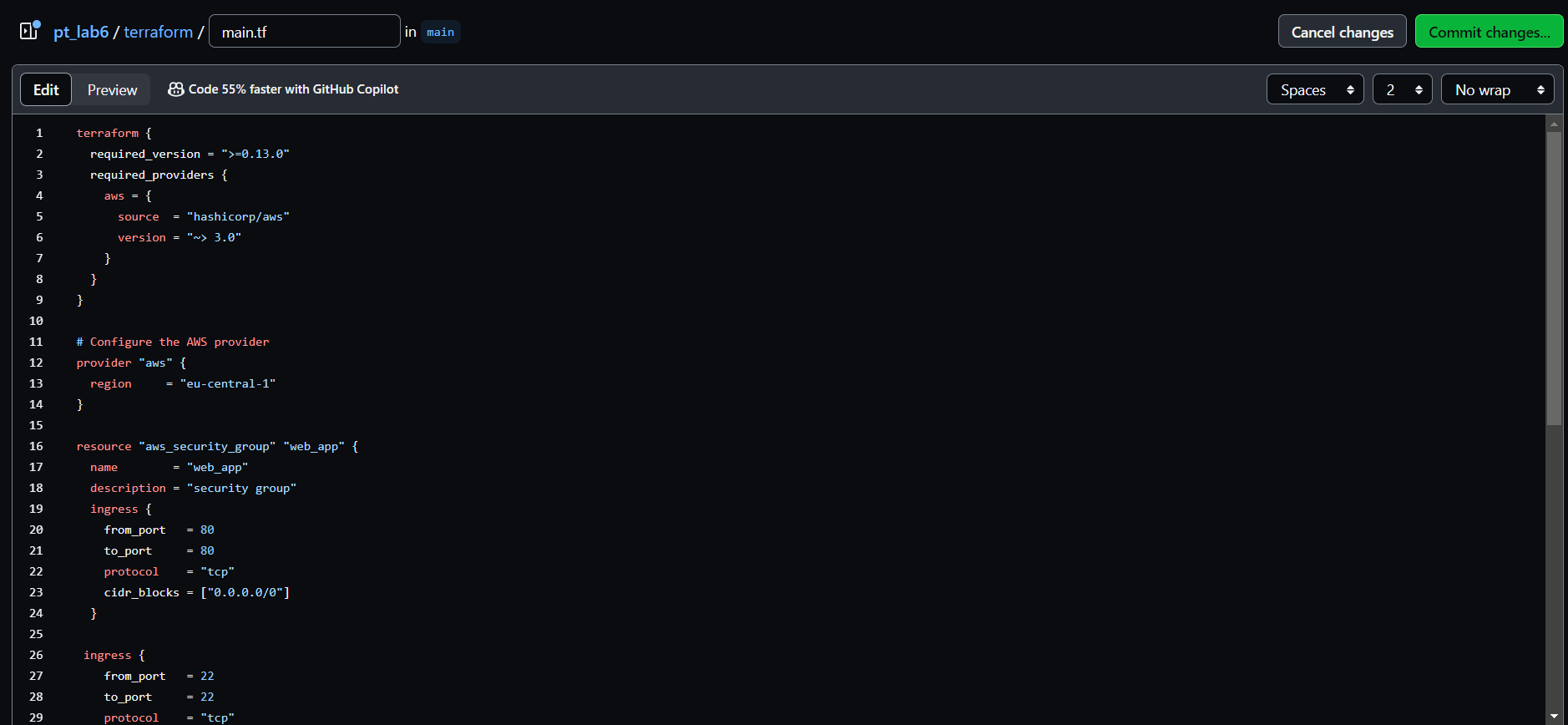
Задля створення робочого процесу на GitHub переходжу до вкладки **Actions** і обираю варіант **Python application**. Далі у файлі з розширенням \*.yml вношу необхідні зміни і натискаю **Commit changes**:



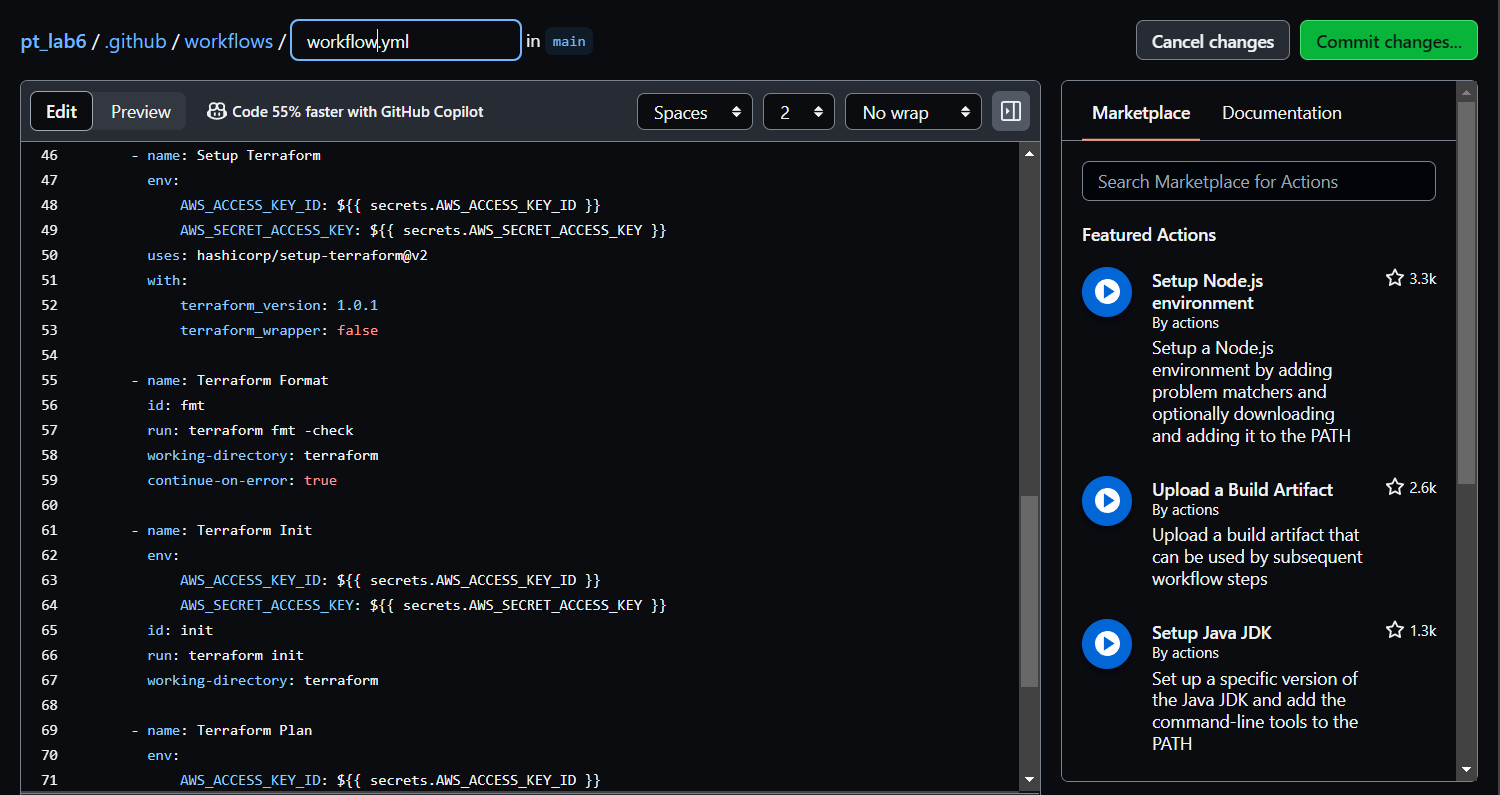
Одразу після створення комміту активується конвеєр, який успішно відпрацьовує усі етапи:



Створюю каталог **terraform** у репозиторії, в який завантажую файл **main.tf** (опис інфраструктури)з таким вмістом:



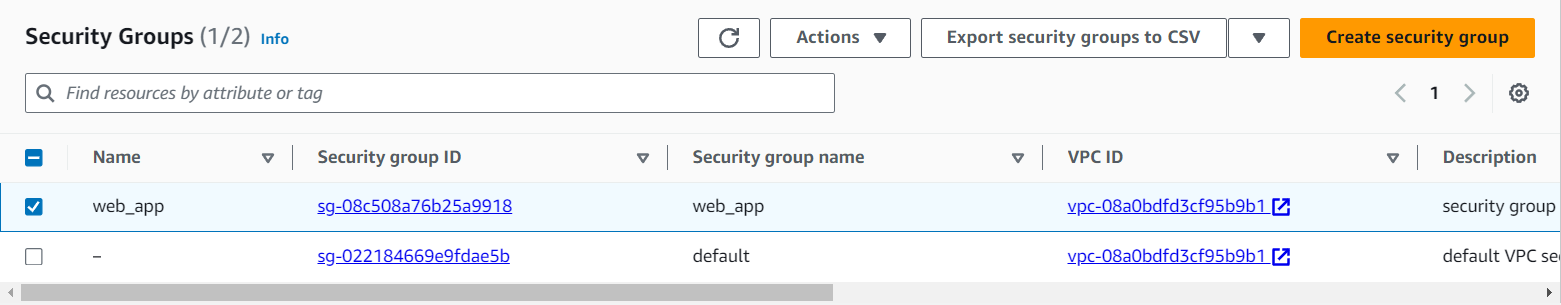
Доповнюю workflow.yml файл кодом для запуску відповідних виконавців команд terraform і роблю комміт:



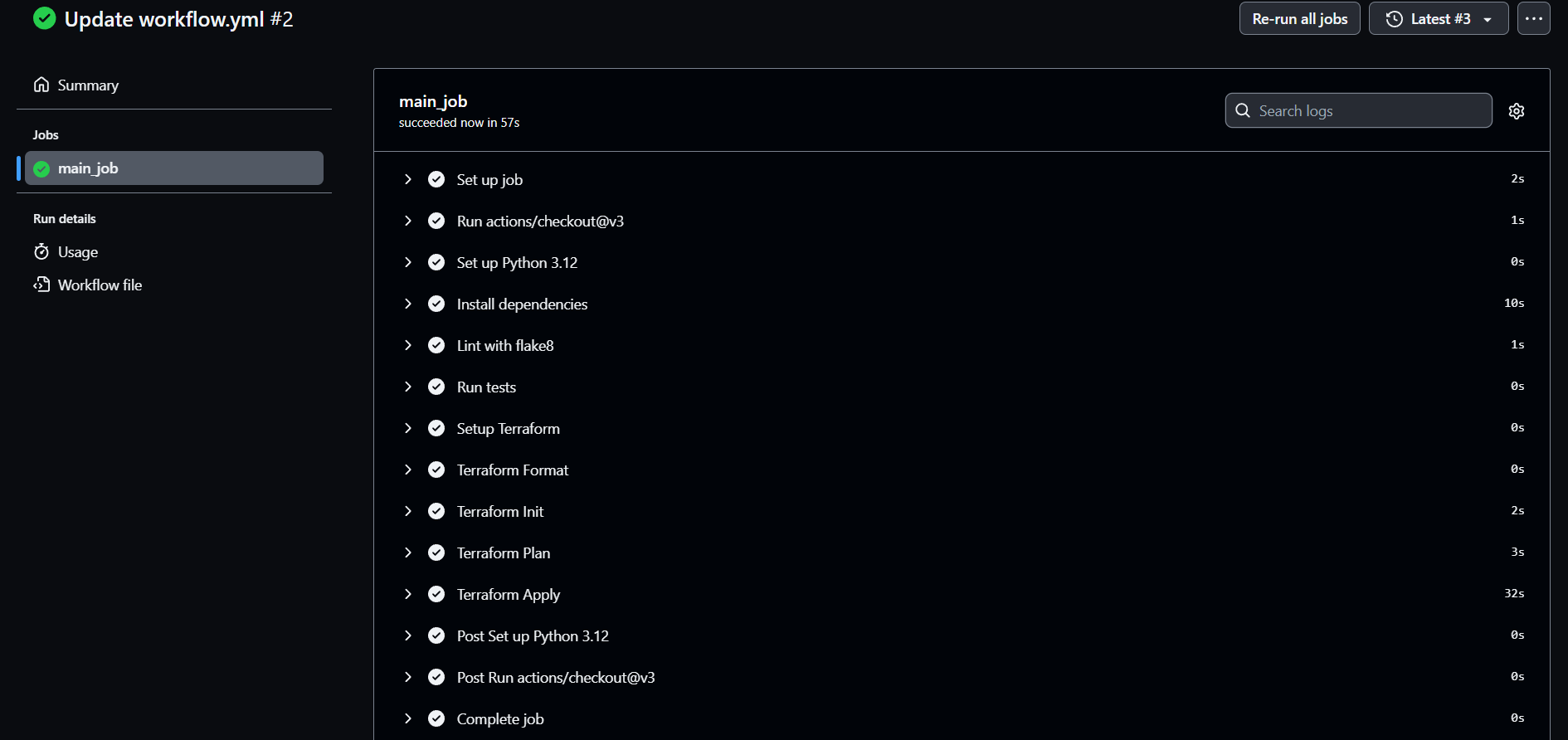
Конвеєр знову запускається, проте виникає помилка:



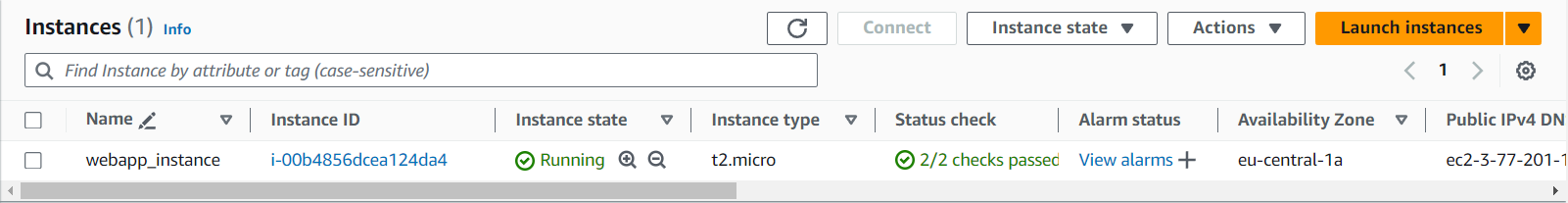
Для її усунення слід в AWS перейти у сервіс **EC2 -> Security Groups** та видалити групу web\_app (виділена пташкою):

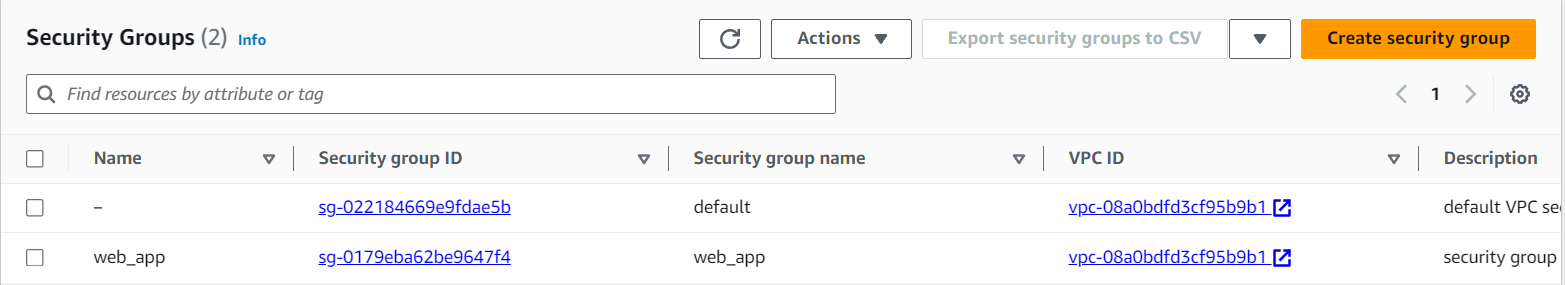


Тоді конвеєр успішно спрацьовує:



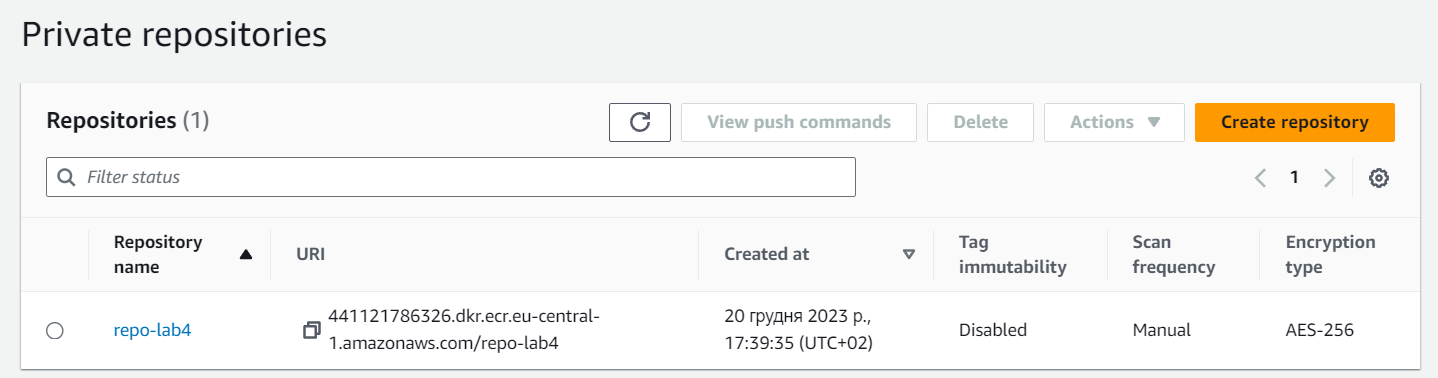
Відповідна інфраструктура також розгортається у хмарі AWS:



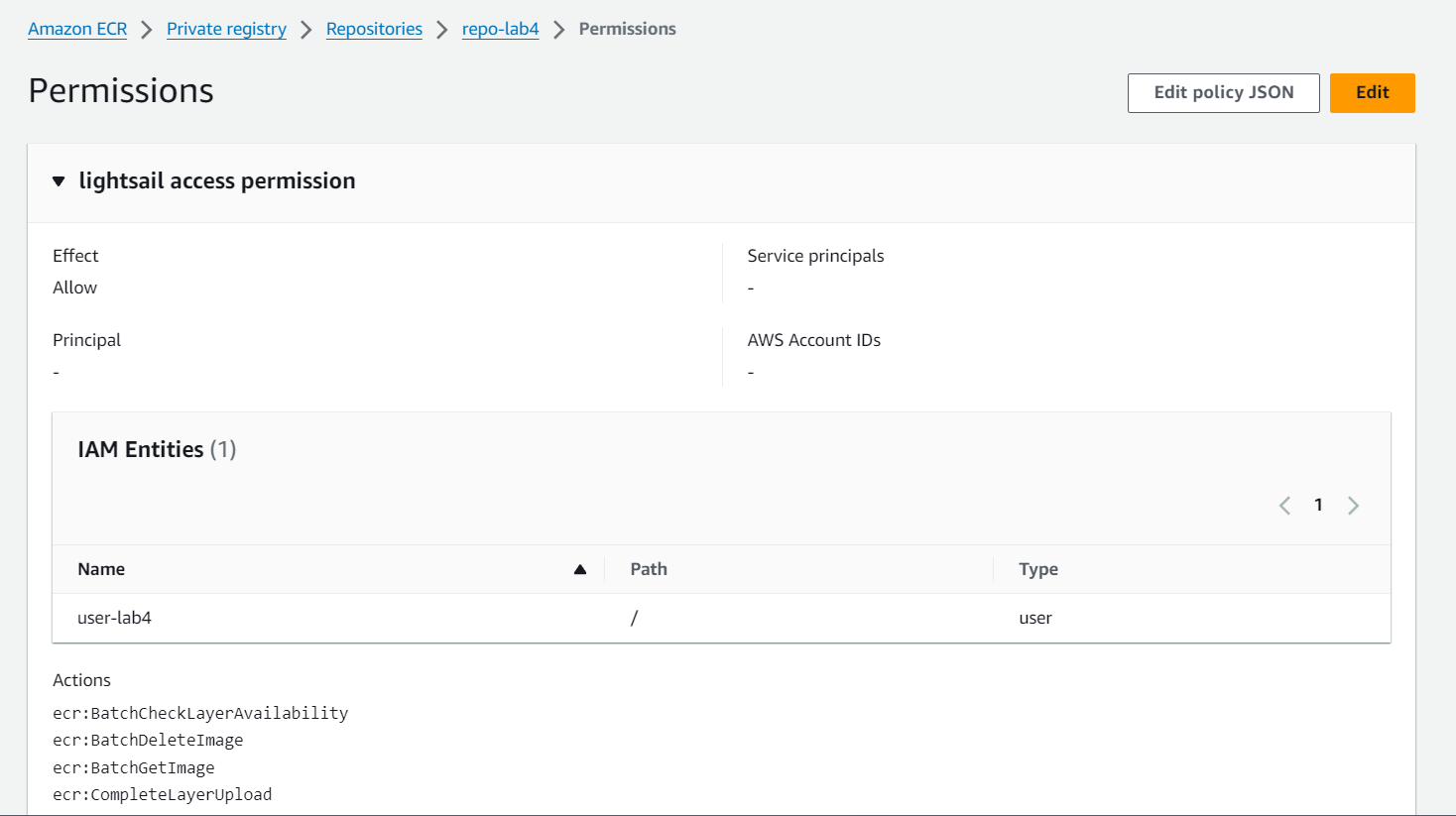


Для розміщення застосунку у хмарі скористаємося сервісом Amazon Lightsail.

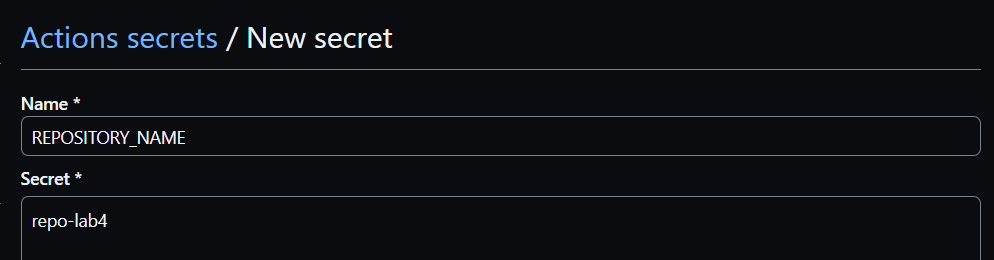
Спершу створюю новий приватний репозиторій за допомогою сервісу Amazon Elastic Container Registry:



Для того, щоб Lightsail отримав доступ до репозиторію, надам йому відповідні дозволи. **Обираю репозиторій -> Actions -> Permissions.** Результат має вигляд:

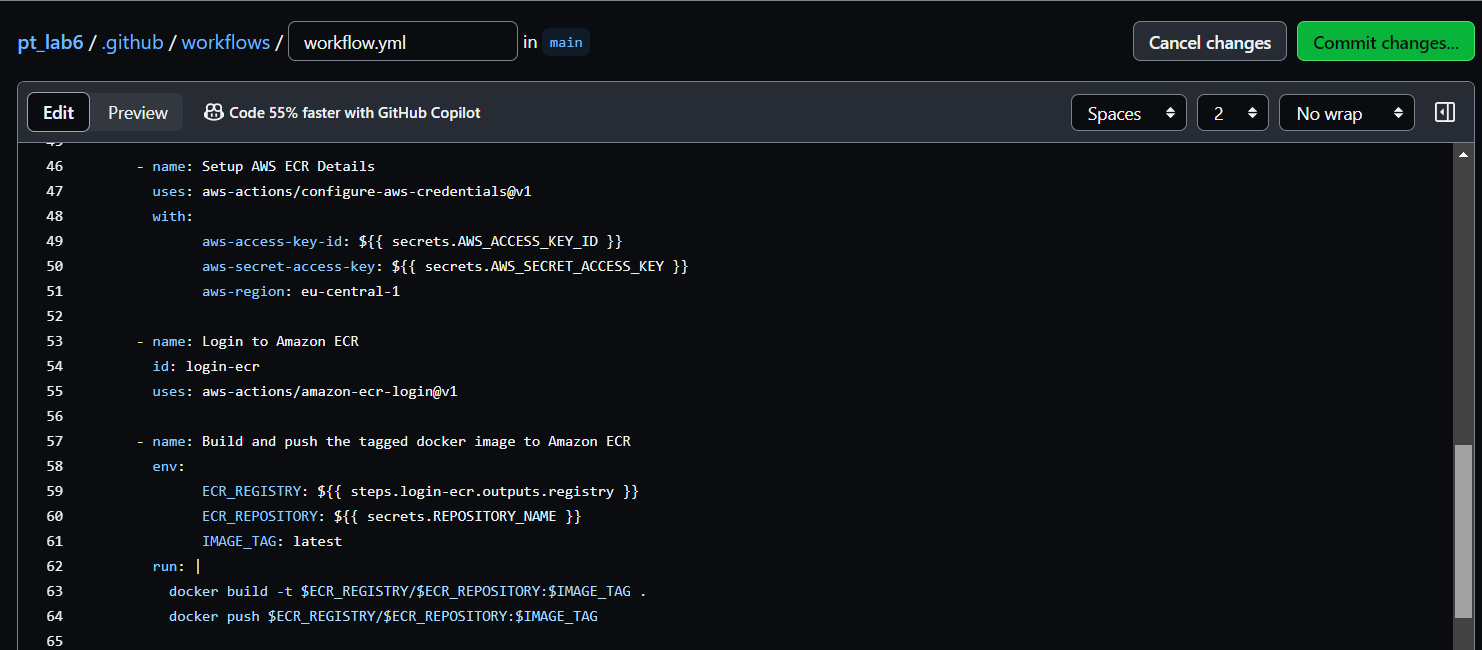


Додам ще два секрети у GitHub репозиторії: REPOSITORY\_NAME (назва репозиторію) та REPOSITORY\_URI (посилання на репозиторій):

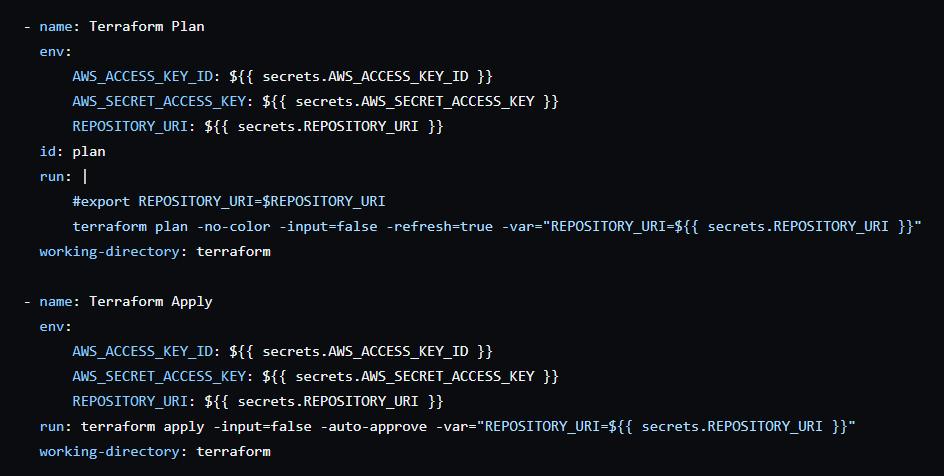




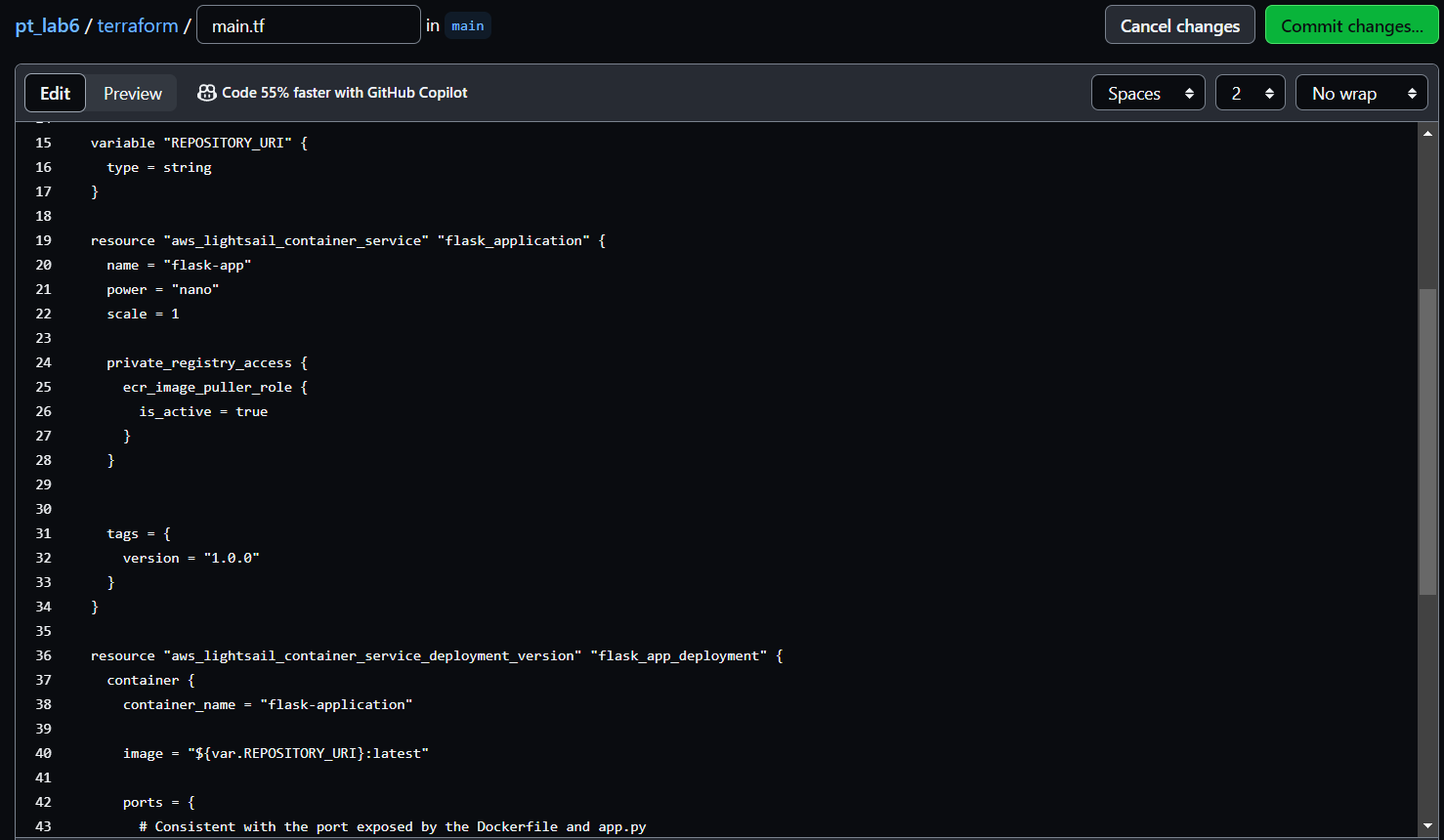
Доповнюю workflow.yml кодом для підключення до AWS, формування Docker-образу та його вивантаження в репозиторій Amazon ECR:



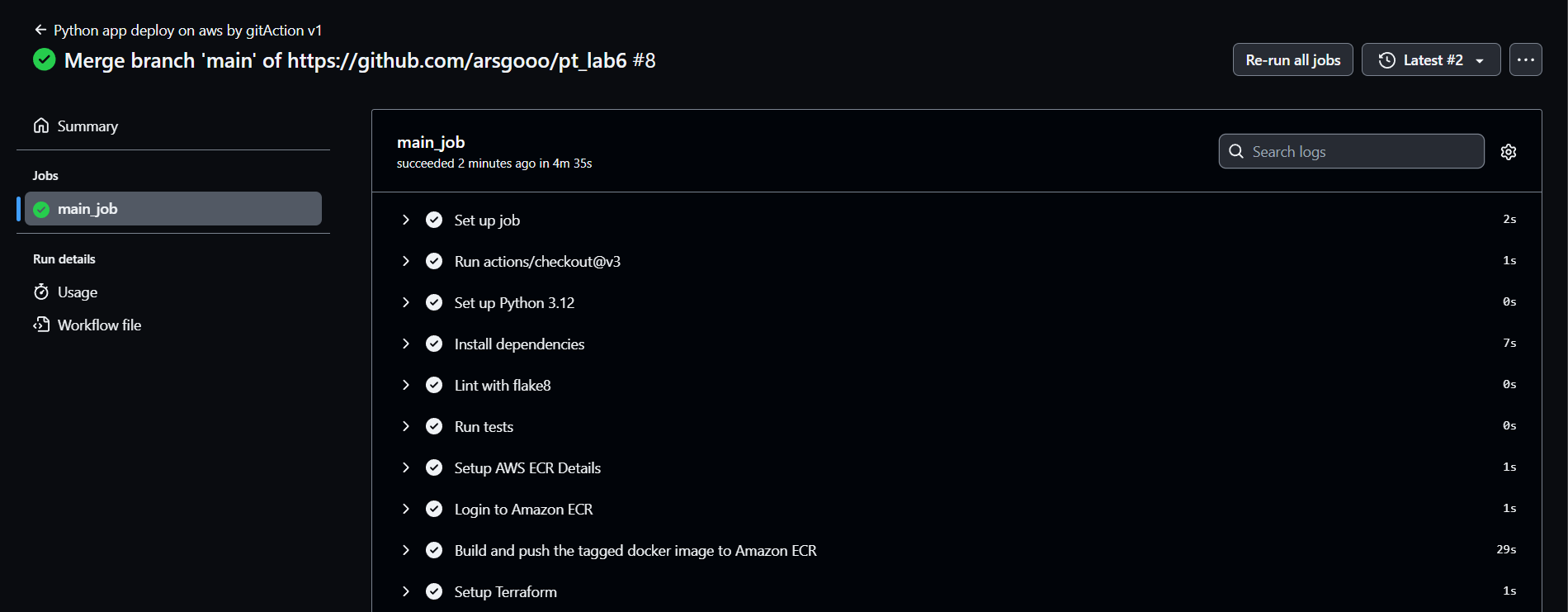
Задаю адресу образу через змінну середовища REPOSITORY\_URI для того, щоб Terraform міг в подальшому розгорнути Lightsail:



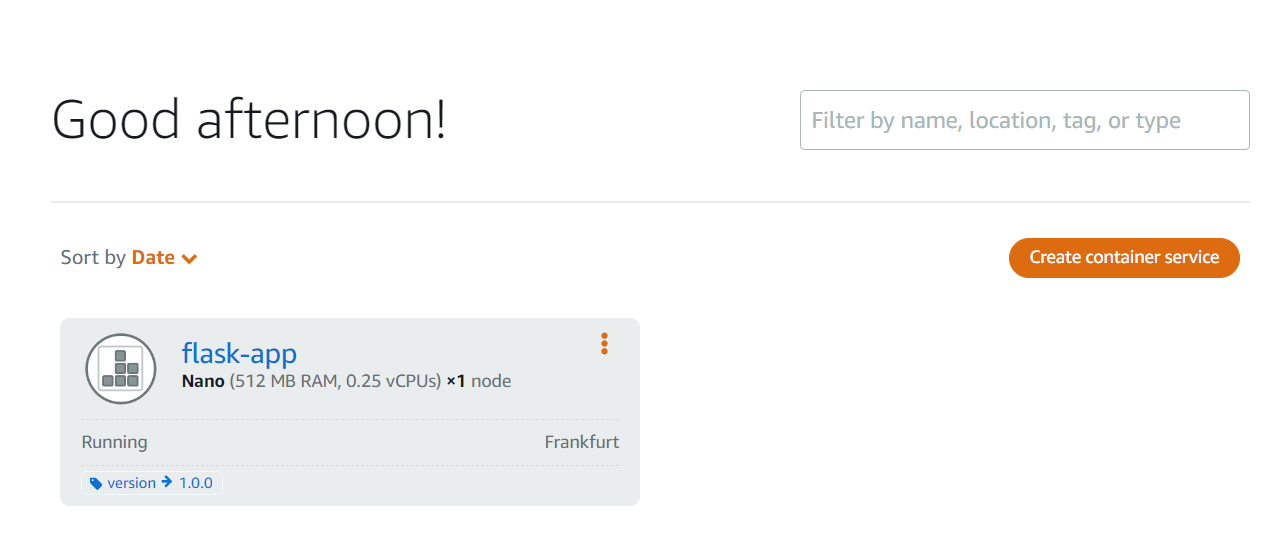
Також редагую вміст файлу main.tf:



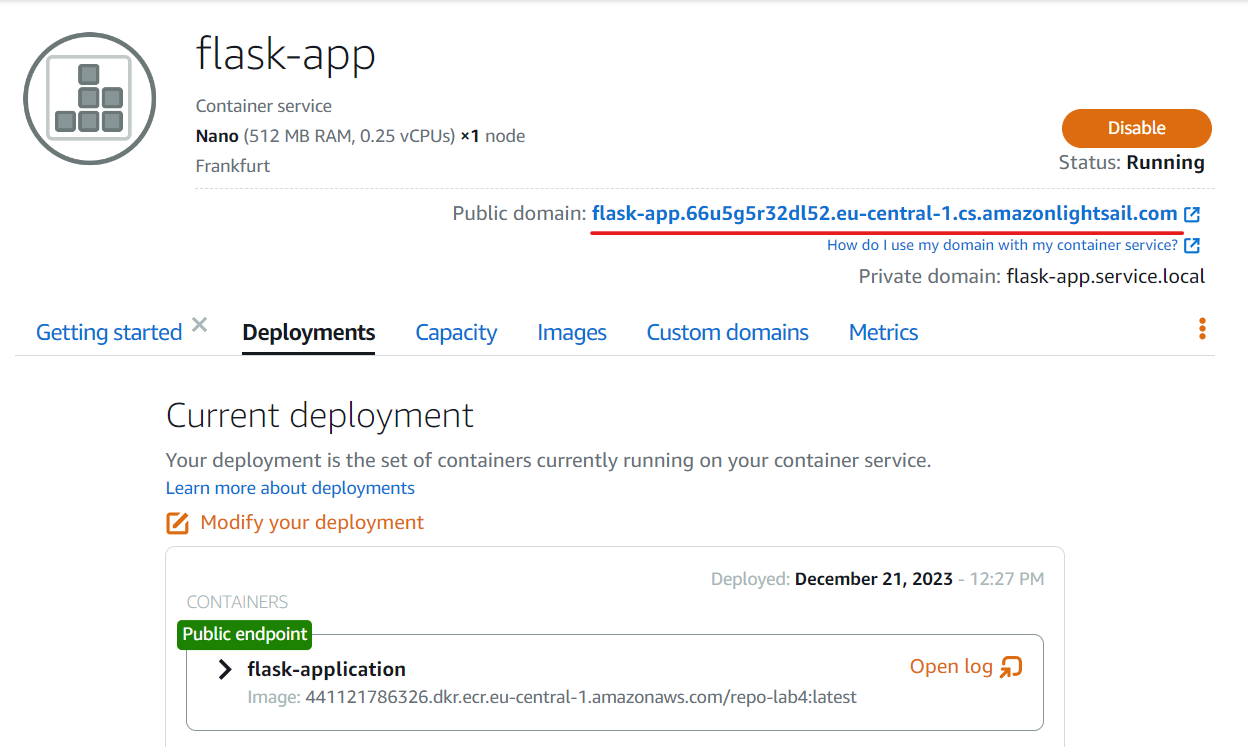
Конвеєр вже традиційно успішно відпрацьовує:

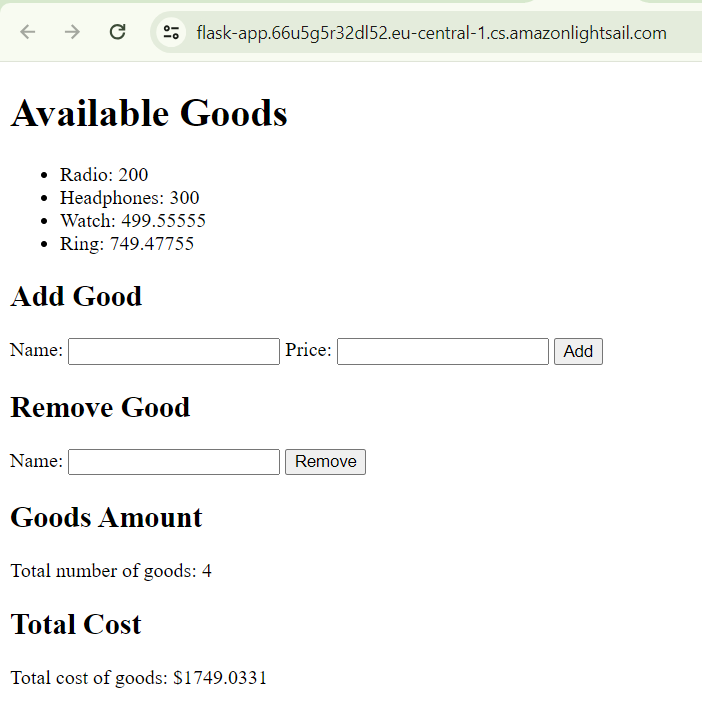


Перейшовши у сервіс Lightsail, можна побачити новостворений контейнер:



Для перевірки працездатності застосунку заходжу в цей контейнер та переходжу за посиланням public domain:





**Завдання 2 (дослідження довільного сервісу AWS та його розгортання у хмарі з використанням Terraform та GitHub Actions)**

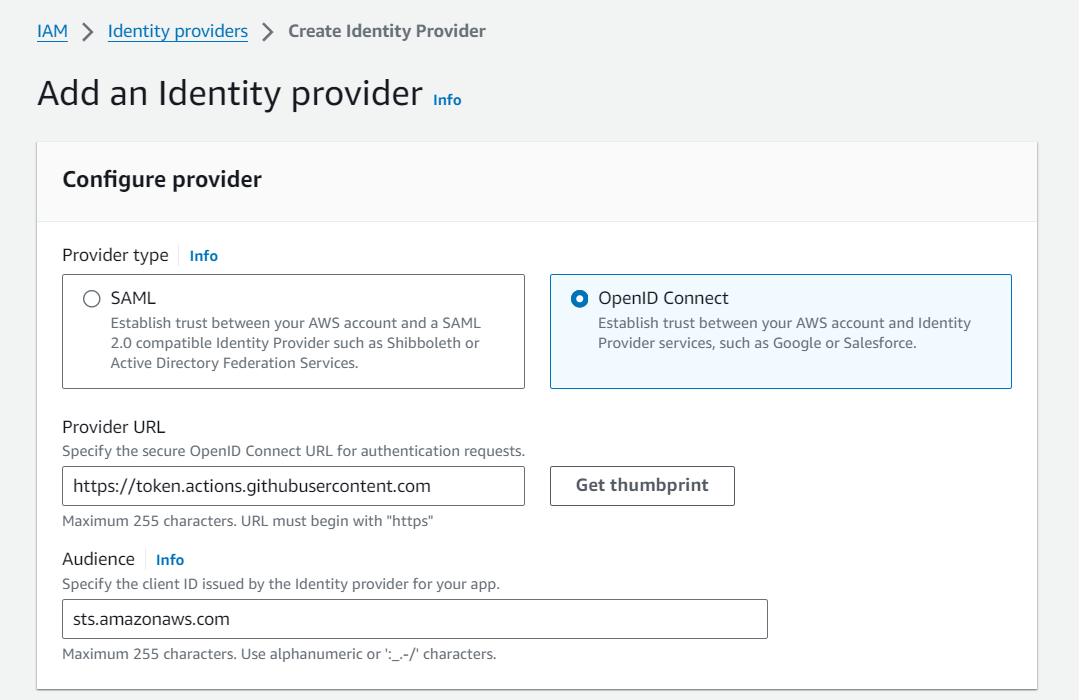
Для реалізації цього завдання скористаюся технологією OpenID.

OpenID – це відкритий стандарт аутентифікації, який дозволяє користувачам автентифікуватися на різних веб-сайтах та службах, використовуючи одні й ті самі облікові дані. Це стандарт для одноразової аутентифікації (Single Sign-On, SSO), який спрощує процес реєстрації та входу на веб-ресурси.

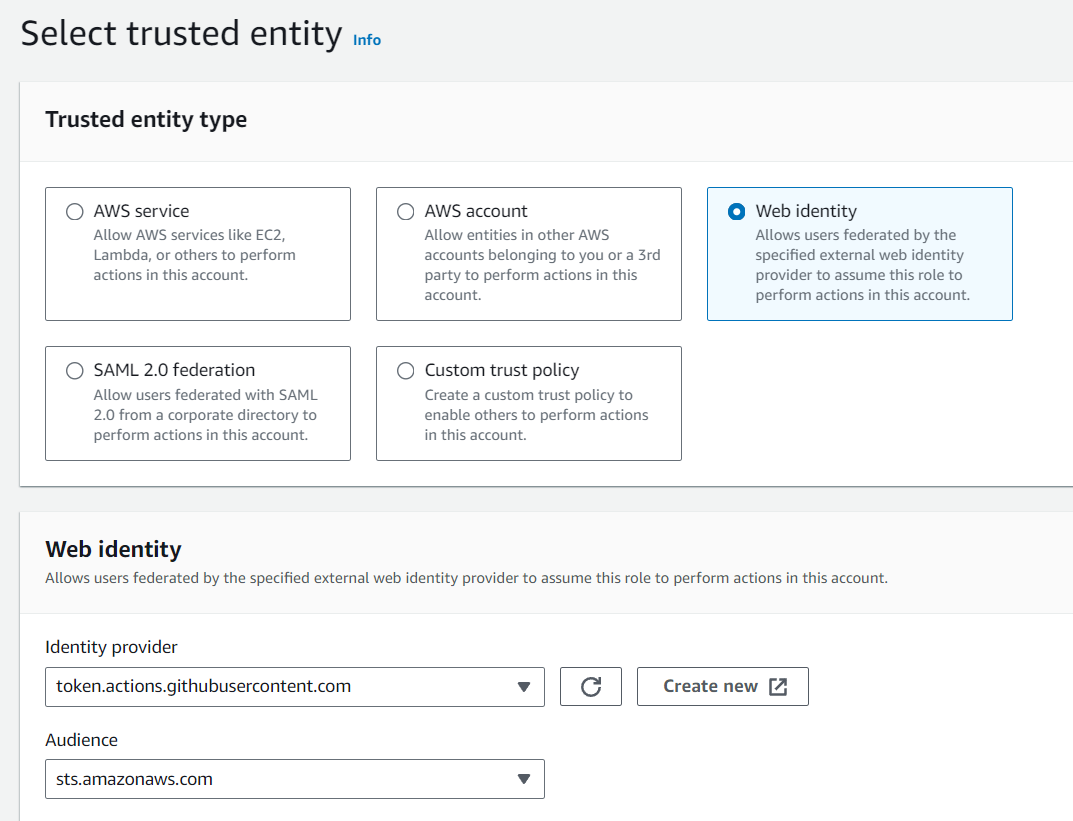
Основні складові OpenID включають:

1. **OpenID Provider (OP)**: Це система, яка виконує аутентифікацію користувачів і надає їм унікальні ідентифікатори (OpenID Identifier). Сервер OpenID Provider може бути розгорнутий на окремому веб-сайті або вбудований в інші послуги.
2. **OpenID Relying Party (RP)**: Це веб-сайти або служби, які використовують OpenID для аутентифікації своїх користувачів. Relying Party отримує унікальний ідентифікатор від користувача через процес OpenID.
3. **OpenID Identifier**: Унікальний ідентифікатор, який видається користувачеві OpenID Provider. Він може бути представлений URL або ідентифікатором.

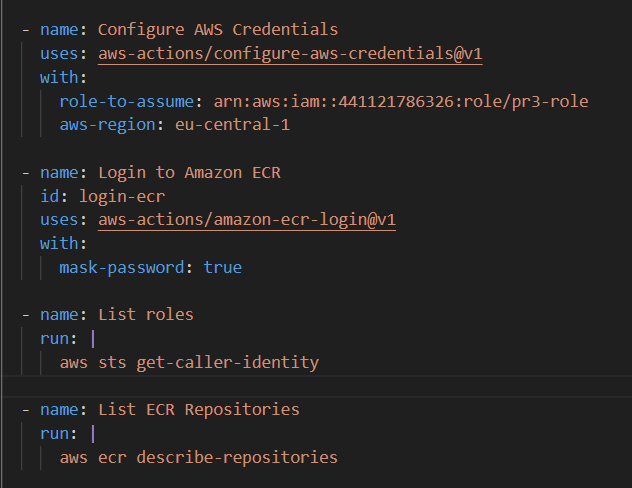
Спершу створюю провайдера ідентифікаційних даних для GitHub. Для цього переходжу в IAM -> Access management -> Identity providers -> Add provider. У вікні, що розгорнулося, вказую такі дані:



Також створюю нову роль (вкладка Roles):

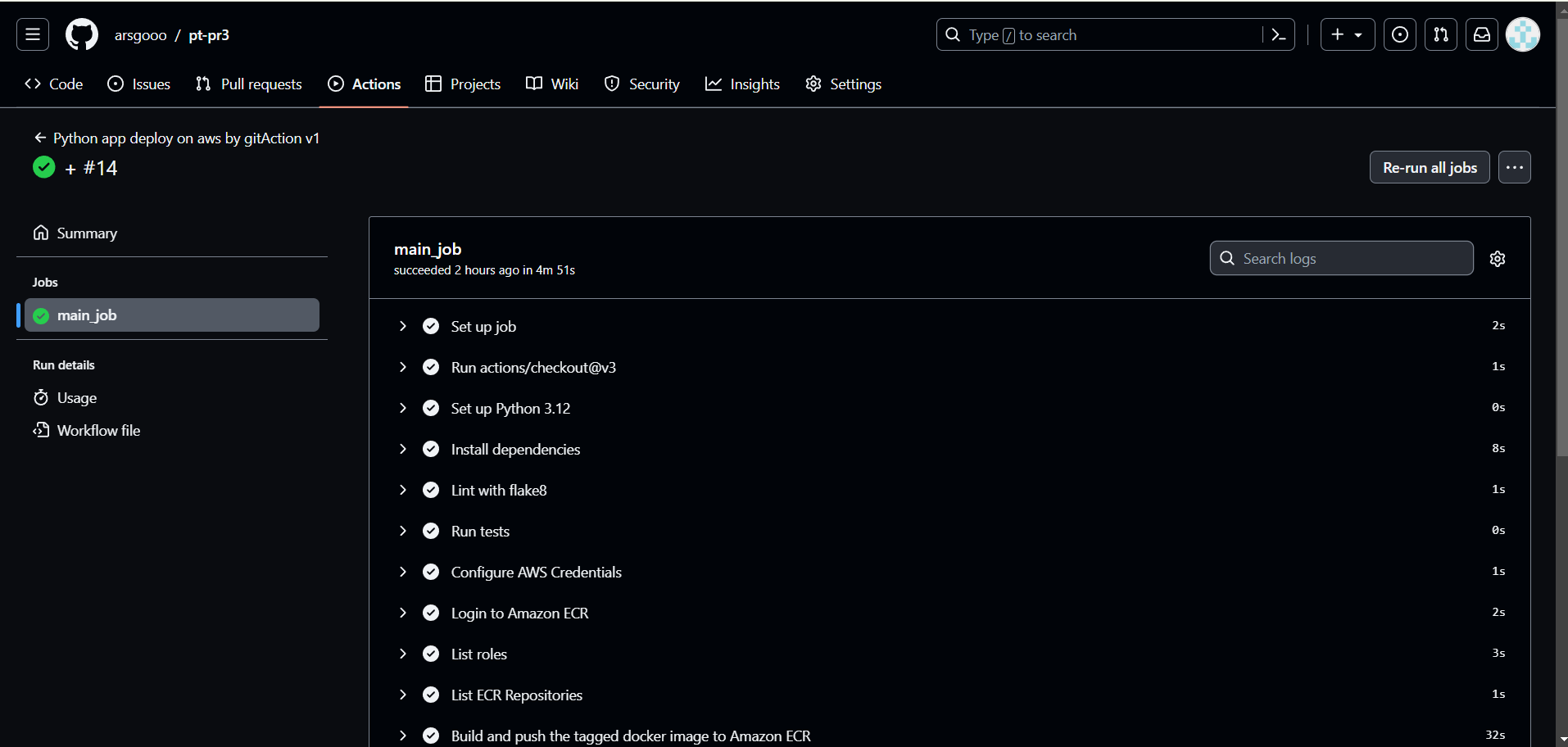


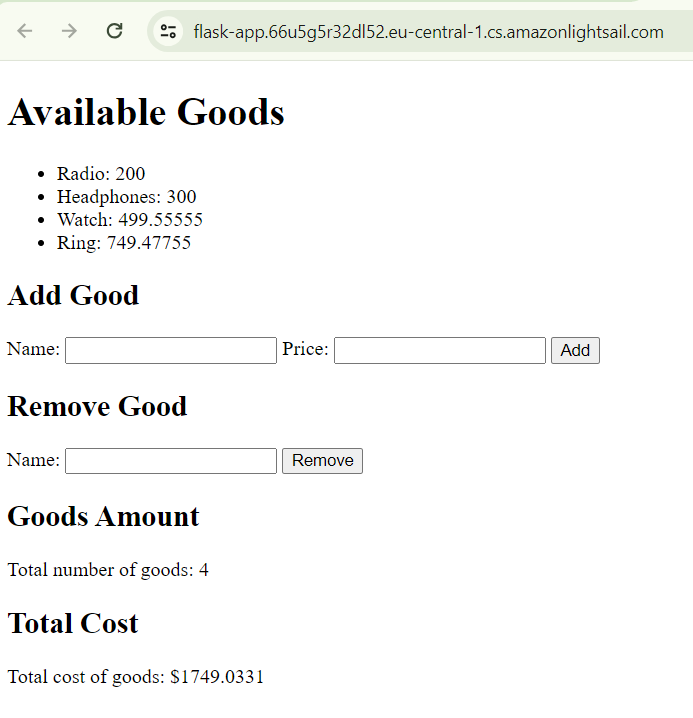
Подальші кроки аналогічні до тих, що були виконані у першому завданні (створення репозиторію, надання дозволів, створення файлу terraform тощо). Проте суттєвою відмінністю є вміст файлу workflow.yml, який цього разу виглядає так (наведено лише відмінні фрагменти):



Параметр **role-to-assume** вказує ARN (Amazon Resource Name) роль, яку GitHub Actions має «приміряти на себе». В цьому випадку це роль **pr3-role**. Параметр **mask-password: true** захищає конфіденційні дані пароля, приховуючи їх у виведеній інформації (інакше при запуску конвеєра буде виведено попередження про те, що пароль неприхований). **List roles** виводить інформацію про поточну роль, а **List ECR Repositories** виводить інформацію про репозиторії контейнерів, доступні в Amazon ECR.

Роблю останній комміт і перевіряю працездатність конвеєра:





**Висновок:** за час виконання лабораторної роботи було ретельно опрацьовано поняття інфраструктури як коду та розгорнуто її за допомогою Terraform та GitHub Actions. В першому завданні це було реалізовано шляхом залучення секретів. В другому ж використовувався стандарт OpenID, який дозволяє створити єдиний обліковий запис для автентифікації на багатьох непов’язаних між собою інтернет-ресурсах.