Laporan Akhir

Implementasi CI/CD Menggunakan Jenkins untuk Aplikasi Streamlit di AWS EC2

Mata Kuliah Analitik Big Data

Kelas T1A

Kelompok 6

Nafakhatul Fadliyah	225150201111012
Sekar Parameswara Meilia Soegiharto	225150207111068
Theresa Yolanda Sinaga	225150207111074
Callysta Salsabila Srigiardita	225150207111115
Hanidura Ayatulloh	225150207111005

Program Studi Teknik Informatika

Fakultas Ilmu Komputer

Universitas Brawijaya

2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan perangkat lunak modern, efisiensi dan kecepatan merupakan dua hal yang sangat penting. Perusahaan dan pengembang dituntut untuk terus merilis pembaruan sistem secara berkala, memperbaiki bug dengan cepat, serta memastikan bahwa setiap perubahan yang dilakukan pada kode tidak menimbulkan masalah baru di lingkungan produksi. Untuk menjawab tantangan tersebut, praktik *Continuous Integration* (CI) dan *Continuous Deployment/Delivery* (CD) menjadi solusi utama yang diadopsi secara luas dalam alur kerja DevOps.

CI/CD memungkinkan pengembang untuk secara otomatis membangun, menguji, dan mendistribusikan aplikasi dengan lebih andal dan konsisten. Salah satu tools yang umum digunakan untuk membangun sistem CI/CD adalah Jenkins. Jenkins merupakan *automation server* open-source yang menyediakan berbagai plugin dan fitur untuk mendukung proses otomasi dalam pengembangan perangkat lunak. Dengan pipeline Jenkins, seluruh tahapan mulai dari *source code checkout* hingga *deployment* dapat dikelola secara otomatis dan terstruktur.

Dalam konteks laporan ini, Jenkins digunakan untuk mengelola proses CI/CD pada sebuah aplikasi berbasis Streamlit, yaitu framework Python yang digunakan untuk membangun antarmuka data interaktif. Aplikasi akan dijalankan pada server cloud menggunakan layanan AWS EC2, yang memberikan fleksibilitas dalam pengelolaan sumber daya komputasi.

Implementasi ini tidak hanya bertujuan untuk mempermudah proses deploy, namun juga menjadi contoh nyata bagaimana integrasi DevOps dapat meningkatkan produktivitas, efisiensi, dan reliabilitas dalam pengembangan perangkat lunak.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana cara mengimplementasikan pipeline CI/CD menggunakan Jenkins untuk aplikasi berbasis Streamlit?
2. Bagaimana cara mengintegrasikan Jenkins dengan Streamlit untuk otomatisasi deployment?
3. Bagaimana pipeline dapat mendukung proses testing dan deployment aplikasi Python secara otomatis dan efisien?

1.3 Tujuan

1. Menerapkan praktik Continuous Integration dan Continuous Deployment (CI/CD) dalam proses pengembangan perangkat lunak berbasis Python menggunakan tools yang umum digunakan di industri.
2. Membangun dan mengkonfigurasi Jenkins sebagai automation server untuk mengelola pipeline CI/CD secara otomatis mulai dari tahap clone repositori hingga deployment aplikasi.
3. Men-deploy aplikasi Streamlit ke server AWS EC2 secara otomatis dengan pendekatan pipeline berbasis script untuk memastikan kestabilan dan kecepatan deploy.
4. Menyusun dokumentasi teknis lengkap yang dapat menjadi referensi dalam penerapan CI/CD pada proyek pengembangan lainnya.

1.4 Manfaat

1. Meningkatkan efisiensi dan konsistensi dalam proses deployment aplikasi, sehingga pengembang tidak perlu melakukan deploy secara manual setiap kali ada perubahan pada kode.
2. Mengurangi risiko kesalahan manusia (human error) dalam proses build dan deploy melalui sistem otomasi yang terstruktur.
3. Mempercepat siklus pengembangan dan perbaikan aplikasi, karena pipeline memberikan umpan balik (feedback) secara langsung setelah build dan testing dilakukan.
4. Memberikan pengalaman praktis dalam penggunaan Jenkins dan AWS EC2, yang merupakan skill penting dalam industri teknologi saat ini.
5. Meningkatkan kualitas perangkat lunak yang dikembangkan melalui penerapan pengujian otomatis dan deployment yang dapat diulang dengan mudah.

BAB II
KAJIAN PUSTAKA

2.1 *Continuous Integration dan Continuous Deployment (CI/CD)*

Continuous Integration (CI) adalah praktik dalam pengembangan perangkat lunak di mana kode yang dikembangkan oleh beberapa kontributor digabungkan ke repositori bersama secara berkala. Tujuan utama dari CI adalah untuk mendeteksi kesalahan lebih awal, meningkatkan kolaborasi tim, dan mempercepat proses integrasi kode. Setiap perubahan kode yang dikirimkan akan secara otomatis diuji melalui pipeline untuk memastikan integritas sistem tetap terjaga.

Sementara itu, *Continuous Deployment* atau *Continuous Delivery* (CD) merujuk pada proses otomatisasi distribusi aplikasi ke lingkungan staging atau produksi. Setelah lulus pengujian otomatis, perubahan kode dapat langsung di-deploy tanpa intervensi manual, sehingga proses rilis menjadi lebih cepat, konsisten, dan andal. Dengan menerapkan CI/CD, pengembang dapat meminimalkan risiko kerusakan sistem akibat perubahan kode, mempercepat siklus rilis, dan meningkatkan produktivitas secara keseluruhan.

2.2 Jenkins sebagai Automation Server

Jenkins merupakan salah satu tools open-source yang paling populer dan banyak digunakan dalam implementasi sistem otomasi Continuous Integration dan Continuous Deployment (CI/CD). Jenkins memungkinkan pengguna untuk membuat pipeline otomatis yang mencakup berbagai tahapan penting dalam pengembangan perangkat lunak, seperti proses *checkout* source code, *build*, *testing*, hingga *deployment* ke lingkungan server. Pipeline tersebut dapat ditulis dalam bentuk script yang fleksibel, baik menggunakan deklaratif syntax yang lebih mudah dibaca, maupun scripted syntax yang lebih dinamis. Salah satu keunggulan utama Jenkins adalah ketersediaan ribuan plugin yang memungkinkan integrasi dengan berbagai tools dan layanan lain seperti GitHub, Docker, Slack, dan AWS. Selain itu, Jenkins juga dikenal karena fleksibilitasnya dalam mendefinisikan alur kerja DevOps yang kompleks sesuai kebutuhan tim pengembang. Dukungan dari komunitas pengguna

yang besar serta dokumentasi yang lengkap menjadikan Jenkins sebagai pilihan utama dalam membangun sistem otomasi DevOps yang handal dan skalabel.

2.3 Streamlit untuk Aplikasi Interaktif Python

Streamlit adalah sebuah framework open source berbasis Python yang dirancang untuk memudahkan pengembangan aplikasi web interaktif secara cepat, terutama dalam bidang visualisasi data, analisis, dan penerapan machine learning. Keunggulan utama dari Streamlit terletak pada kemudahannya dalam penggunaan, di mana pengembang cukup menulis skrip Python seperti biasa tanpa perlu memahami teknologi frontend seperti HTML, CSS, atau JavaScript. Framework ini sangat ideal digunakan untuk membangun dashboard analisis data, membuat prototipe model machine learning, serta menyajikan hasil riset atau eksperimen secara visual dan interaktif. Selain itu, Streamlit juga mendukung eksekusi melalui command line interface dan dapat dengan mudah dijalankan di server lokal maupun cloud, sehingga sangat cocok diintegrasikan dalam pipeline CI/CD sebagai bagian dari proses otomatisasi deploy aplikasi.

2.4 Amazon Web Services (AWS) dan EC2

Amazon Web Services atau AWS adalah platform cloud computing yang dimiliki oleh Amazon dan menyediakan berbagai layanan infrastruktur serta platform yang mencakup komputasi, penyimpanan, database, dan jaringan. Salah satu layanan utama yang paling sering digunakan adalah Elastic Compute Cloud atau EC2, yang memungkinkan pengguna untuk menjalankan instansi server virtual secara fleksibel dan scalable sesuai dengan kebutuhan. Layanan EC2 memberikan keleluasaan bagi pengembang untuk mengelola akses dan keamanan melalui penggunaan key pair dan konfigurasi security group. Selain itu, EC2 juga mendukung hosting aplikasi baik di sisi backend maupun frontend secara langsung tanpa perlu perangkat fisik tambahan. Dalam konteks proyek ini, layanan EC2 digunakan sebagai lingkungan deploy penuh untuk Jenkins sebagai automation server serta aplikasi Streamlit yang akan dijalankan secara otomatis melalui pipeline CI/CD.

2.5 GitHub dan Webhook

GitHub merupakan platform layanan hosting repositori Git yang mendukung sistem kontrol versi dan kolaborasi dalam pengembangan perangkat lunak. Platform ini memungkinkan banyak pengembang bekerja secara bersamaan dalam satu proyek dengan pengelolaan versi kode yang efisien. Untuk mendukung proses otomatisasi dalam implementasi CI/CD, GitHub menyediakan fitur Webhook, yaitu sebuah mekanisme yang memungkinkan GitHub mengirim notifikasi berupa permintaan HTTP ke server Jenkins setiap kali terjadi perubahan pada repositori, seperti push, pull request, atau commit. Webhook berperan penting dalam memastikan pipeline Jenkins dapat berjalan secara otomatis tanpa perlu pemicu manual, sehingga proses integrasi dan deployment dapat dilakukan secara real-time. Hal ini menjadikan alur kerja pengembangan lebih cepat, responsif, dan minim intervensi manual.

BAB III
IMPLEMENTASI

3.1 Data yang Digunakan

Penelitian ini menggunakan dataset "Pembelian Tahun 2024" dari Toko Air Mancur. Dataset ini berisi pembelian barang-barang dari Toko Air Mancur khususnya pada tahun 2024.

3.2 Pustaka yang Digunakan

Berikut beberapa pustaka yang digunakan:

1. *Amazon Web Services (AWS)*

   AWS merupakan platform komputasi awan yang menyediakan layanan infrastruktur seperti Elastic Compute Cloud (EC2), yang digunakan dalam proyek ini untuk menjalankan Jenkins sebagai server otomasi dan tempat deploy aplikasi Streamlit. EC2 memungkinkan pengelolaan sumber daya komputasi secara fleksibel, termasuk konfigurasi keamanan melalui key pair dan security group.

2. *Jenkins*

   Jenkins adalah automation server open source yang digunakan untuk membangun sistem Continuous Integration dan Continuous Deployment (CI/CD). Jenkins mendukung integrasi pipeline berbasis script yang mampu menangani proses otomatisasi mulai dari checkout kode, pembuatan lingkungan kerja, pengujian, hingga deployment ke server.

3. GitHub

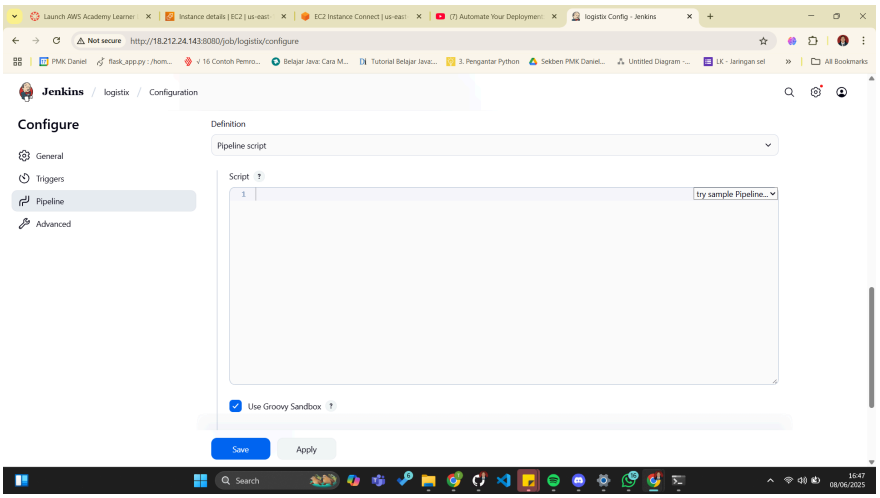
   GitHub digunakan sebagai platform repositori versi yang terintegrasi dengan Jenkins melalui webhook. Dengan webhook, setiap perubahan pada repositori dapat langsung memicu pipeline di Jenkins secara otomatis, mendukung praktik CI/CD secara real-time.

4. Streamlit

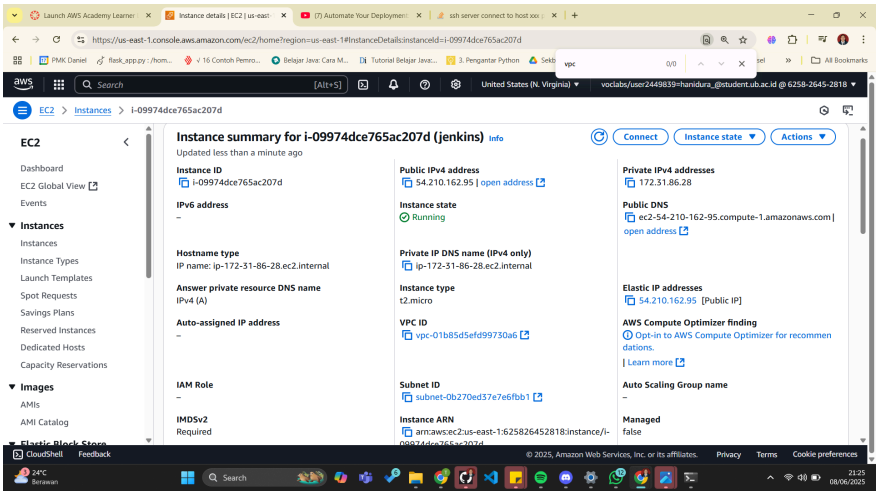
   Streamlit adalah framework Python yang digunakan untuk membuat aplikasi web interaktif dengan cepat, khususnya dalam bidang visualisasi data. Aplikasi berbasis Streamlit dalam proyek ini di deploy secara otomatis menggunakan Jenkins pipeline ke server EC2.

3.3 Tahapan Implementasi

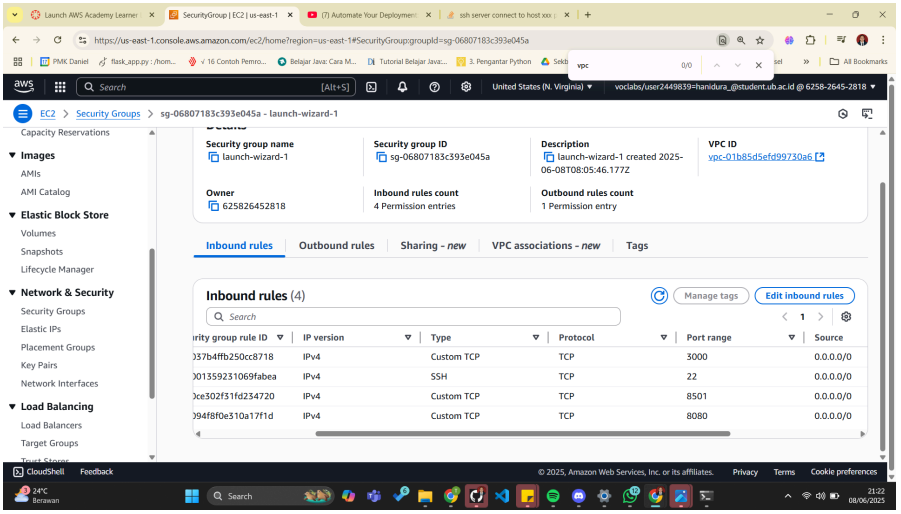
1. Buatkan EC2 untuk Jenkins, pada konfigurasi membuat key pairs dengan nama "`jenkins-key-pair.pem`".



2. Detail Konfigurasi Instance EC2 untuk Jenkins Server.



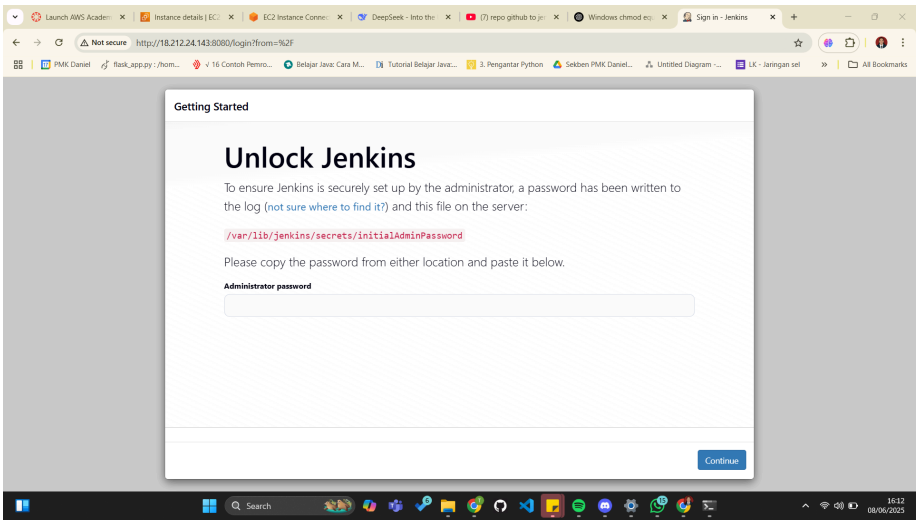
3. Konfigurasi Inbound Rules Security Group AWS EC2.



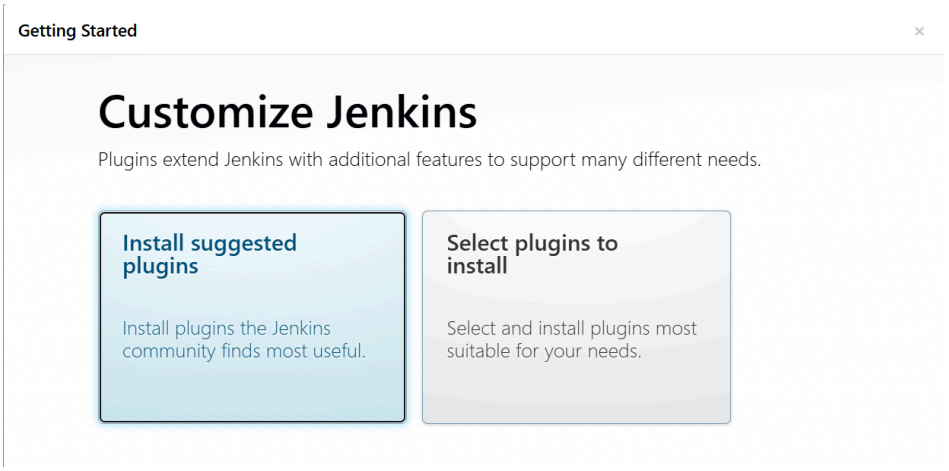
4. Lakukan SSH ke server pada terminal dengan perintah “SSH -i path ke “`jenkins-key-pairs.pem`” ubuntu@54.210.162.95.
5. Lakukan instalasi dengan perintah-perintah berikut :

```
sudo apt update
sudo apt install openjdk-11-jdk -y
wget -q -O -
https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins -y
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

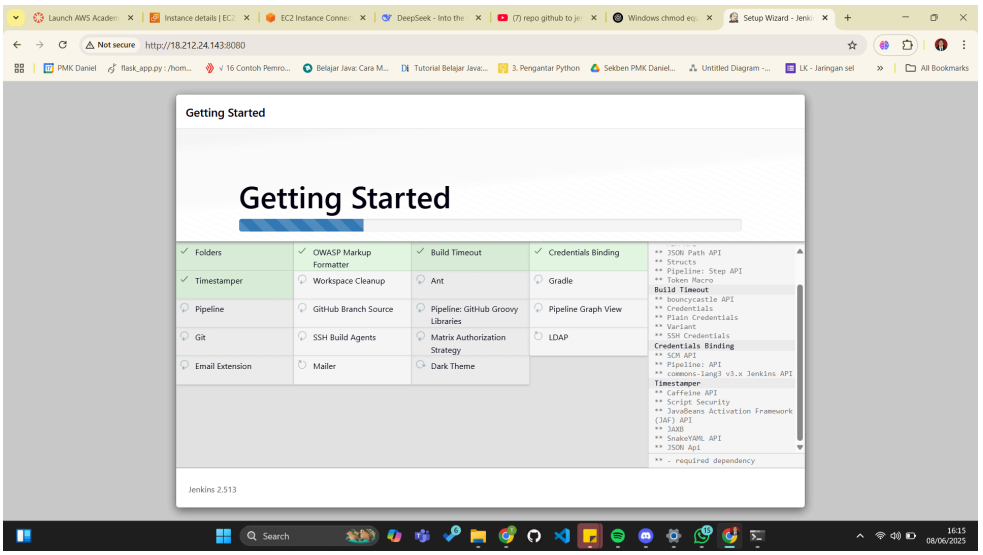
6. Buka http://54.210.162.95:8080 akan terbuka tampilan seperti di bawah



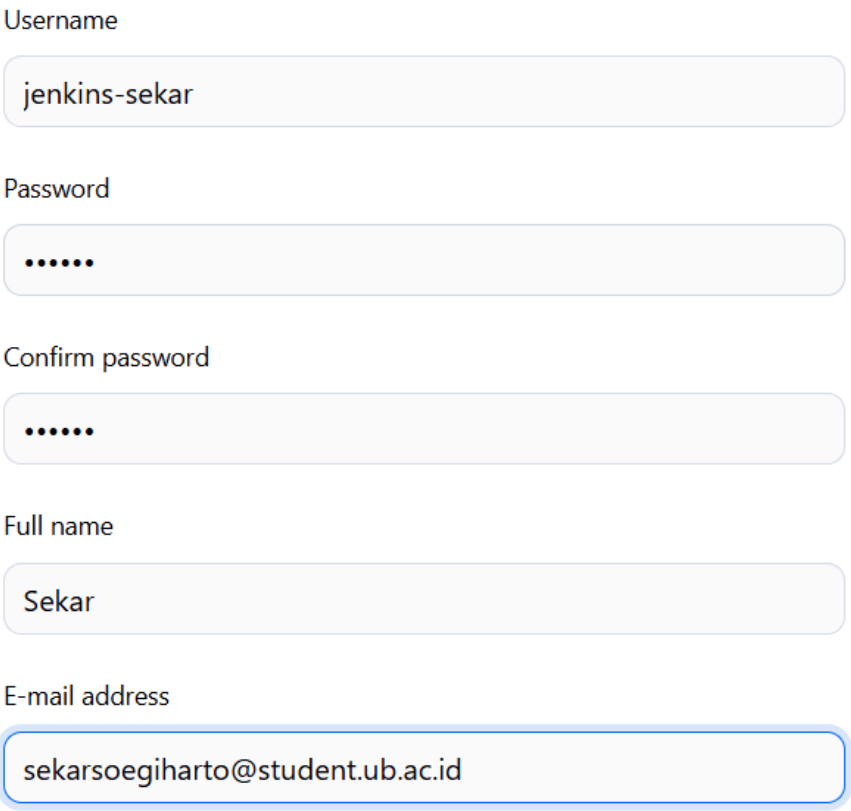
7. Kemudian buka lagi terminal dan jalankan perintah “sudo cat /var/lib/jenkins/secrets/initialAdminPassword” maka akan diberikan string acak yang merupakan Administrator password, masukkan string yang diberikan, klik continue.
8. Kemudian memilih pilihan “install suggested plugin”.



9. Instalasi Plugin Default pada Setup Awal Jenkins.

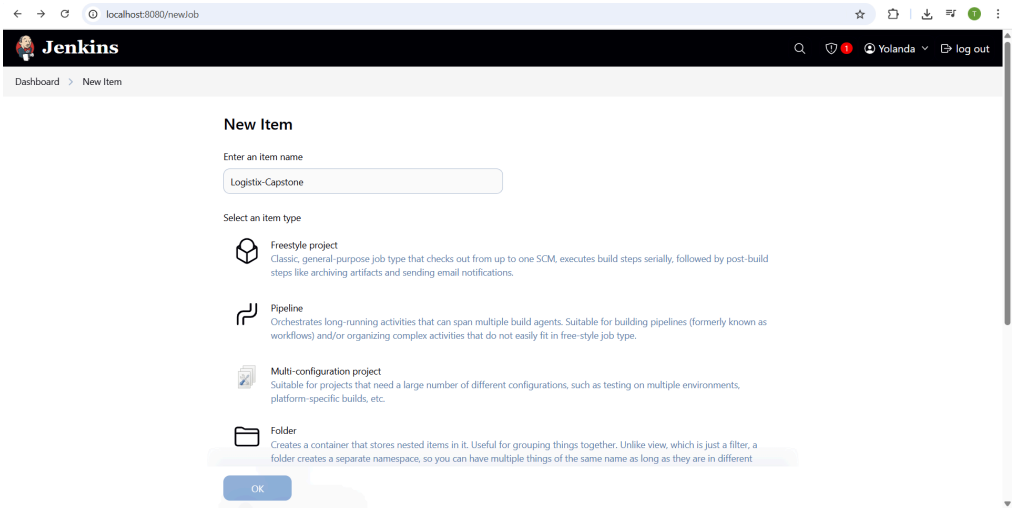


10. Kemudian diarahkan untuk membuat akun jenkins, akan diminta membuat akun admin baru, isi:

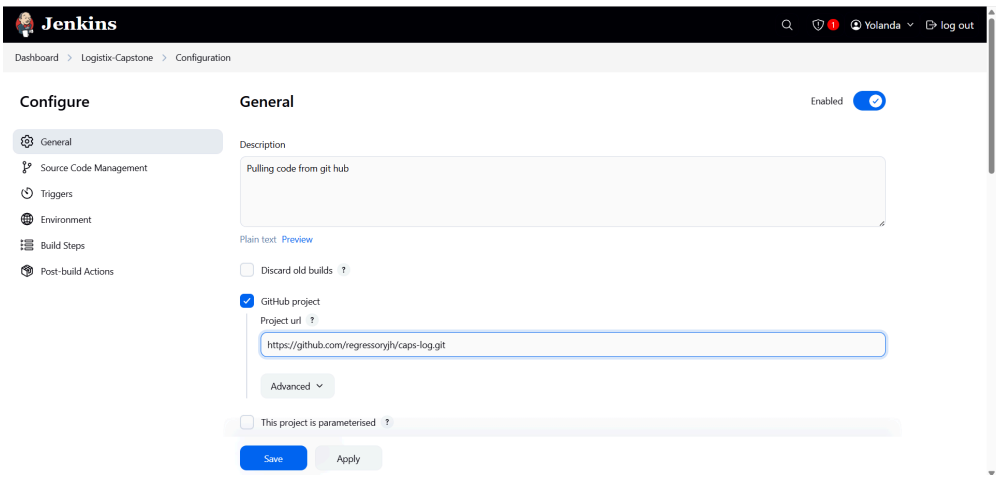


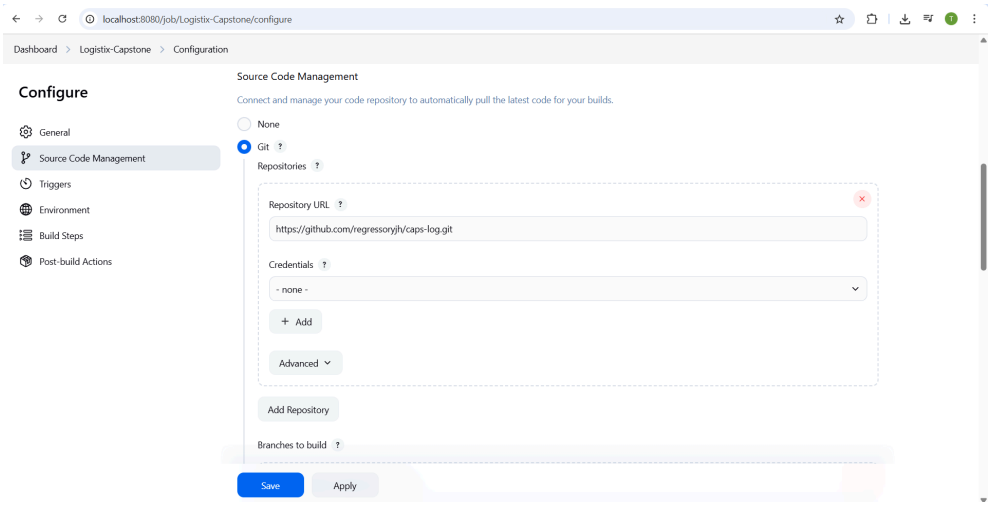
11. Masuk dengan akun jenkins yang dibuat.

12. Pada tab sebelah kiri pilih “new item”.

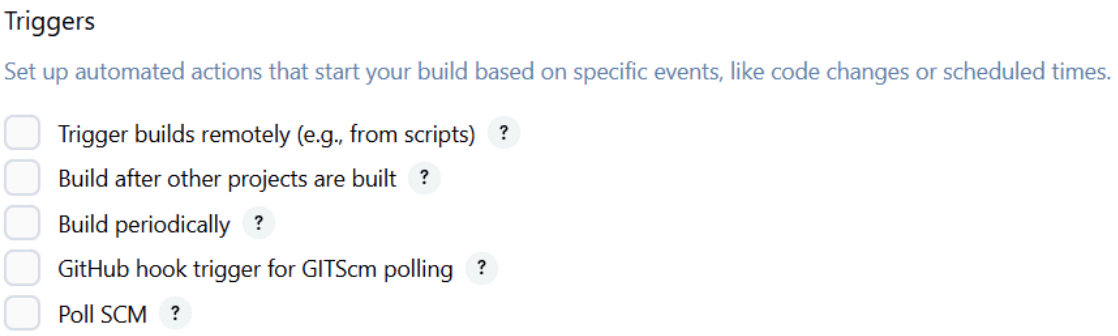


13. Lakukan konfigurasi untuk jenkins.

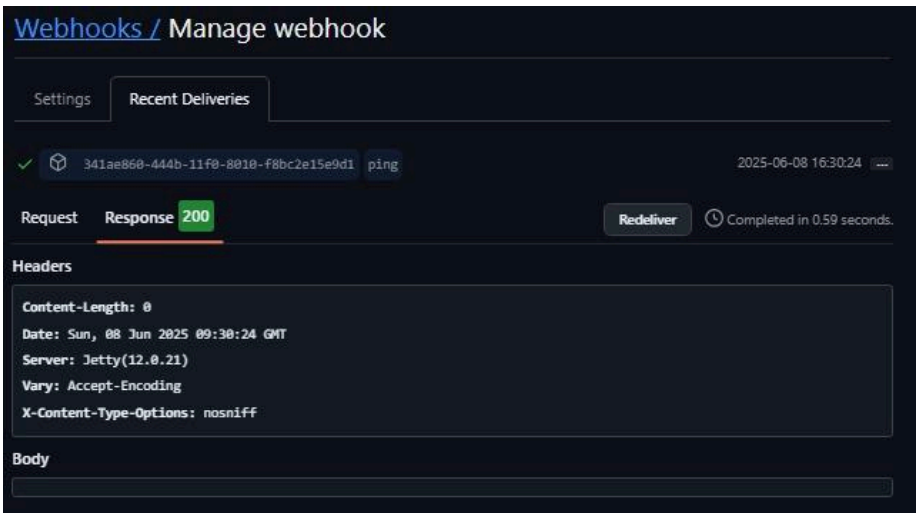




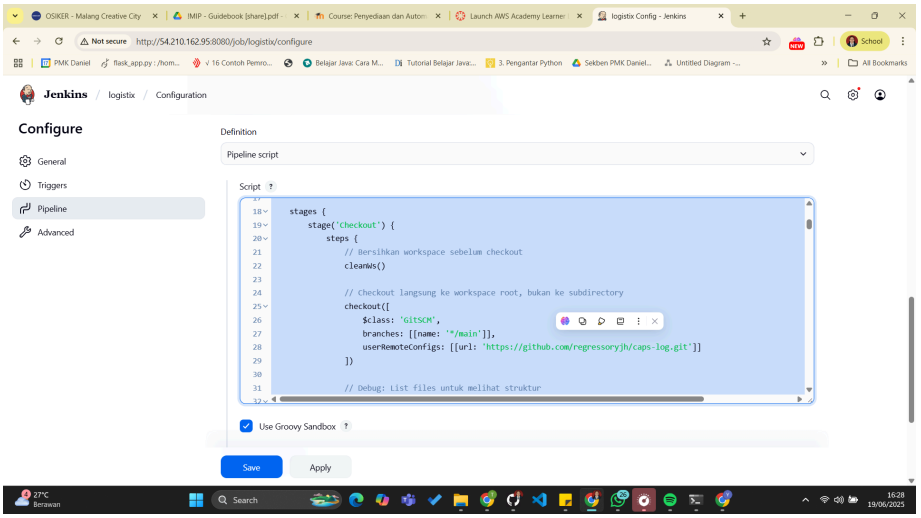
14. Pilih “GitHub hook trigger for GITScm polling”.



15. Pada repositori projek yang akan digunakan, lakukan verifikasi Webhook GitHub dengan memasukkan url: http://18.212.24.143:8080/github-webhook/ jika berhasil (Status 200 OK).

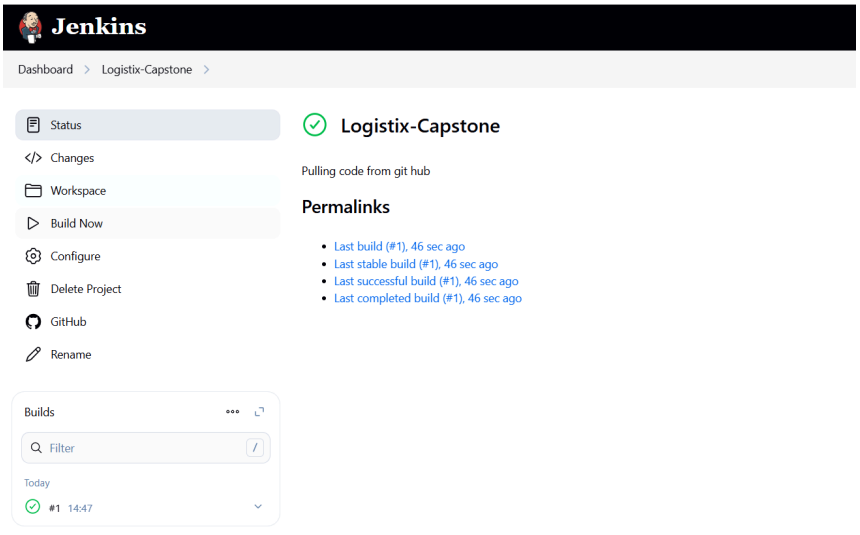


16. Membuat Pipeline CI/CD Menggunakan Jenkins Script.



17. Pilih save.

18. Kemudian tampilan akan seperti berikut.



Notes:

- Jika ingin mengubah kode pipeline, pilih configure pada tab di kiri, masukan kode di “Pipeline syntax” klik save kemudian pada tab yang sama pilih “Build Now”
- Jika terdapat error kita dapat melihat console output untuk mengetahui detail kesalahan yang terjadi selama proses eksekusi pipeline, sehingga kita bisa melakukan debugging dan perbaikan dengan lebih tepat.

19. Membuat kode pipeline.

```
pipeline {
    agent any
    environment {
        VENV_DIR = "venv"
        MAIN_FILE = "app.py"
        PORT = "8501"
        WORK_DIR = "."  // Asumsikan semua file ada di root
        PATH = "/usr/local/bin:/usr/bin:/bin:$PATH"
    }
```

Bagian awal pipeline Jenkins ini menetapkan bahwa job dapat dijalankan di agent mana saja dan mendefinisikan variabel lingkungan penting seperti direktori virtual environment, file utama aplikasi, port Streamlit, direktori kerja, dan path sistem untuk digunakan di seluruh tahap pipeline.

```
    stages {
        stage('Checkout') {
            steps {
                cleanWs()
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: '*/main']],
                    userRemoteConfigs: [[url:
  'https://github.com/regressoryjh/caps-log.git']]
                ])
                sh 'ls -la'
            }
        }
```

Tahap Checkout dalam pipeline Jenkins ini bertugas membersihkan workspace sebelumnya, menarik source code dari branch main repositori GitHub, lalu

menampilkan daftar file untuk memastikan bahwa file berhasil diclone ke dalam workspace.

```
        stage('Build Environment') {
            steps {
                dir("${env.WORK_DIR}") {
                    sh '''
                        rm -rf ${VENV_DIR} || true
                        python3 -m venv ${VENV_DIR}
                        . ${VENV_DIR}/bin/activate
                        pip install --upgrade pip
                        pip install -r requirements.txt
                    '''
                }
            }
        }
```

Tahap Build Environment berfungsi untuk membuat ulang virtual environment Python dari awal, mengaktifkannya, lalu menginstall semua dependensi yang dibutuhkan proyek dari file requirements.txt agar lingkungan siap digunakan untuk proses testing dan deployment.

```
        stage('Run Tests') {
            steps {
                dir("${env.WORK_DIR}") {
                    script {
                        def hasTests = sh(
                            script: 'test -d tests',
                            returnStatus: true
                        ) == 0
                        if (hasTests) {
                            sh '''
                                . ${VENV_DIR}/bin/activate
                                pip install pytest
                                pytest tests/
                            '''
                        } else {
                            echo "⚠ Skipping tests: folder
    'tests/' not found"
                        }
                    }
                }
            }
        }
```

Tahap Run Tests bertujuan untuk memeriksa apakah folder tests/ tersedia; jika ada, pipeline akan mengaktifkan virtual environment, menginstall pytest, dan menjalankan seluruh pengujian otomatis, namun jika folder tidak ditemukan, tahap ini akan dilewati dengan memberikan notifikasi peringatan.

```
        stage('Deploy Streamlit') {
            steps {
                dir("${env.WORK_DIR}") {
                    sh '''
                        pkill -f "streamlit.*${MAIN_FILE}"
|| true
                        . ${VENV_DIR}/bin/activate
                        nohup streamlit run ${MAIN_FILE}
--server.port=${PORT} --server.headless=true
--server.address=0.0.0.0 > streamlit.log 2>&1 &
                        echo $! > streamlit.pid
                        echo "✅ Streamlit running on port
${PORT}"
                    '''
                }
            }
        }
    }
```

Tahap Deploy Streamlit digunakan untuk menghentikan proses Streamlit yang sedang berjalan (jika ada), kemudian menjalankan ulang aplikasi Streamlit secara headless di background pada port yang ditentukan, menyimpan log ke file streamlit.log, dan mencatat PID ke file streamlit.pid agar aplikasi dapat dikelola dengan mudah.

```
    post {
        always {
            dir("${env.WORK_DIR}") {
                archiveArtifacts artifacts: 'streamlit.log',
allowEmptyArchive: true
            }
        }
        success {
            echo "✅ Build & Deploy Succeeded"
        }
        failure {
            echo "❌ Build or Deploy Failed"
        }
    }
```

}

Blok post ini memastikan bahwa file log streamlit.log akan diarsipkan setiap kali pipeline selesai dijalankan (terlepas dari berhasil atau gagal), dan akan menampilkan pesan sukses jika build dan deploy berhasil, atau pesan kegagalan jika terjadi error pada salah satu tahap pipeline.

Kesimpulannya, pipeline Jenkins ini dirancang untuk mengotomatisasi seluruh alur kerja pengembangan aplikasi berbasis Python dengan Streamlit, dimulai dari proses cloning repositori dari GitHub, pembersihan workspace, pembuatan dan konfigurasi virtual environment, instalasi dependensi dari file requirements.txt, hingga pengecekan keberadaan folder tests/ untuk kemudian menjalankan pengujian otomatis menggunakan pytest jika tersedia. Setelah proses build dan test selesai, pipeline akan melanjutkan ke tahap deployment dengan menghentikan proses Streamlit yang sedang berjalan (jika ada), lalu menjalankan ulang aplikasi menggunakan perintah nohup agar tetap berjalan di background pada port 8501. Seluruh proses juga dilengkapi dengan logging ke dalam file streamlit.log, penyimpanan PID untuk pengelolaan proses, serta sistem notifikasi keberhasilan atau kegagalan build, menjadikan pipeline ini modular, terstruktur, dan siap digunakan untuk CI/CD berbasis Jenkins.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil

Setelah seluruh tahapan implementasi CI/CD dilakukan, hasil yang diperoleh menunjukkan bahwa sistem berhasil melakukan otomasi penuh terhadap proses *build*, *test*, dan *deployment* aplikasi Streamlit ke server AWS EC2. Pipeline yang dibangun dengan Jenkins berjalan sesuai alur yang telah dirancang, dimulai dari menarik kode sumber dari GitHub, membangun virtual environment Python, menjalankan pengujian otomatis (jika tersedia), hingga menjalankan aplikasi secara otomatis di server.

Akses ke aplikasi dapat dilakukan melalui alamat publik EC2 di port 8501, menandakan bahwa deployment berjalan dengan sukses. Selain itu, proses logging yang tersimpan dalam file streamlit.log dan pengelolaan PID (streamlit.pid) juga memastikan bahwa proses Streamlit dapat dikontrol dan dipantau dengan baik.

Integrasi webhook dari GitHub ke Jenkins juga berhasil diuji dengan status *200 OK*, yang menandakan bahwa setiap perubahan atau push ke repositori secara otomatis memicu pipeline untuk berjalan tanpa intervensi manual. Hal ini membuktikan bahwa sistem CI/CD yang dibangun telah berhasil menghilangkan kebutuhan akan proses deploy manual dan meningkatkan efisiensi kerja secara signifikan.

4.2 Pembahasan

Implementasi pipeline ini menunjukkan bahwa CI/CD dengan Jenkins dapat digunakan secara efektif dalam proyek pengembangan aplikasi Python yang sederhana maupun kompleks. Jenkins memberikan fleksibilitas penuh dalam mengatur setiap tahapan pipeline, mulai dari *checkout*, *build*, *test*, hingga *deploy*, dan menyediakan umpan balik real-time melalui console output.

Dalam pengujian, pipeline mampu mendeteksi keberadaan folder tests/ dan menjalankan pytest hanya jika diperlukan, yang merupakan salah satu keunggulan dari pendekatan berbasis script karena mampu menangani kondisi dinamis. Selain

itu, penggunaan pkill dan nohup pada tahap deployment juga menunjukkan pendekatan yang praktis dalam mengatur ulang proses aplikasi yang berjalan, tanpa perlu merestart seluruh instance EC2.

Dari sisi DevOps, hasil implementasi ini memberikan banyak manfaat, di antaranya mempercepat waktu pengembangan dan distribusi, meningkatkan konsistensi hasil deployment, serta memungkinkan deteksi dan perbaikan error dengan lebih cepat melalui notifikasi langsung dari Jenkins dan GitHub.

Secara keseluruhan, sistem yang dibangun sudah memenuhi prinsip CI/CD dengan baik dan dapat digunakan sebagai model dasar untuk proyek-proyek serupa di masa depan, baik untuk skala kecil seperti Streamlit dashboard maupun untuk aplikasi produksi yang lebih kompleks.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa penerapan sistem Continuous Integration dan Continuous Deployment (CI/CD) menggunakan Jenkins pada aplikasi Streamlit berhasil diimplementasikan dengan baik dan berjalan sesuai dengan alur yang direncanakan. Proses CI/CD yang dibangun mampu mengotomatisasi seluruh tahapan mulai dari pengambilan kode dari GitHub, pembuatan dan konfigurasi virtual environment, instalasi dependensi, pengujian otomatis, hingga deployment aplikasi ke server AWS EC2 secara headless.

Integrasi antara GitHub dan Jenkins melalui webhook juga berfungsi secara optimal, memungkinkan pipeline dijalankan secara otomatis setiap kali terjadi perubahan pada repositori. Hal ini menunjukkan bahwa sistem yang dibangun tidak hanya mampu meningkatkan efisiensi dalam proses deployment, tetapi juga mendukung praktik pengembangan perangkat lunak yang lebih modern, cepat, dan minim kesalahan manual.

Dengan keberhasilan implementasi ini, Jenkins terbukti dapat menjadi solusi yang handal dan fleksibel dalam membangun sistem otomasi CI/CD untuk berbagai jenis proyek, termasuk aplikasi berbasis Python seperti Streamlit. Infrastruktur yang dibangun dapat dijadikan dasar untuk pengembangan sistem DevOps yang lebih luas dan kompleks di masa mendatang.

5.2 Saran

Berdasarkan hasil implementasi dan evaluasi yang telah dilakukan, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan dan penyempurnaan sistem CI/CD di masa mendatang:

1. Keamanan Server dan Aplikasi

   Disarankan untuk melakukan pengaturan firewall dan security group yang

lebih ketat pada AWS EC2, seperti hanya membuka port yang diperlukan (misalnya 8080 untuk Jenkins dan 8501 untuk Streamlit), serta mengaktifkan autentikasi tambahan untuk menghindari akses tidak sah ke server.

2. Pemisahan Lingkungan Deployment

Untuk proyek berskala besar, disarankan agar lingkungan pengujian (*staging*) dan produksi (*production*) dipisahkan agar proses deploy dan testing tidak saling mengganggu. Jenkins dapat dikonfigurasi untuk mengelola pipeline di kedua lingkungan secara paralel.

3. Monitoring dan Logging Lebih Lanjut

Sebaiknya menambahkan tools pemantauan seperti Grafana, Prometheus, atau integrasi Slack/Email untuk mendapatkan notifikasi real-time jika pipeline gagal atau terjadi error saat deployment.

4. Penyempurnaan Testing

Penggunaan *unit test* dapat dilengkapi dengan *integration test* dan *end-to-end test* untuk memastikan bahwa seluruh sistem berjalan sesuai fungsinya, terutama ketika aplikasi tumbuh menjadi lebih kompleks.

5. Containerization dengan Docker

Implementasi pipeline dapat ditingkatkan dengan menggunakan Docker agar aplikasi dapat dijalankan dalam container yang terisolasi, memudahkan proses deploy lintas server dan meningkatkan konsistensi lingkungan.

6. Backup dan Recovery

Disarankan untuk menambahkan mekanisme backup otomatis terhadap konfigurasi Jenkins, credential, dan log deployment agar dapat dipulihkan jika terjadi kegagalan sistem.

Dengan mengikuti saran-saran ini, sistem CI/CD yang dibangun akan menjadi lebih aman, stabil, dan scalable, serta siap untuk diterapkan dalam pengembangan perangkat lunak di lingkungan produksi yang sesungguhnya.

DAFTAR PUSTAKA

Amazon Web Services. (n.d.). *Amazon EC2 Documentation*. Retrieved from https://docs.aws.amazon.com/ec2/

GitHub. (n.d.). *Creating Webhooks*. Retrieved from https://docs.github.com/en/developers/webhooks-and-events/webhooks/creating-webhooks

Jenkins. (n.d.). *User Documentation*. Retrieved from https://www.jenkins.io/doc/

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations*. IT Revolution Press.

Matplotlib Developers. (n.d.). *Matplotlib: Visualization with Python*. Retrieved from https://matplotlib.org/

McKinney, W. (2018). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.

Seaborn Developers. (n.d.). *Seaborn Statistical Data Visualization*. Retrieved from https://seaborn.pydata.org/

Streamlit Inc. (n.d.). *Streamlit Documentation*. Retrieved from https://docs.streamlit.io/