**Serverless Content Management System with AWS App Runner**

*A Course Project Report Submitted in partial fulfillment of the course requirements for the award of grades in the subject of*

**CLOUD BASED AIML SPECIALITY**

**(22SDCS07A)**

by

**Sreeja Boddula**

**2210030248**

*Under the esteemed guidance of*

**Ms. P. Sree Lakshmi**

Assistant Professor,

Department of Computer Science and Engineering



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**K L Deemed to be UNIVERSITY**

*Aziznagar, Moinabad, Hyderabad,*

*Telangana, Pincode: 500075*

April 2025

**K L Deemed to be UNIVERSITY**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

*Certificate*

This is Certified that the project entitled **“Serverless Content Management System with AWS AppRunner”** which is a experimental &/ theoretical &/ Simulation&/ hardware work carried out by Sreeja Boddula (2210030248), in partial fulfillment of the course requirements for the award of grades in the subject of  **CLOUD BASED AIML SPECIALITY**, during the year **2024-2025**. The project has been approved as it satisfies the academic requirements.

**Ms.P.Sree Lakshmi Dr. Arpita Gupta**

**Course Coordinator Head of the Department**

**Ms. P. Sree Lakshmi**

**Course Instructor**

**CONTENTS**

Page No.

1. Introduction

2. AWS Services Used as part of the project

3. Steps involved in solving project problem statement

4. Stepwise Screenshots with brief description

5. Learning Outcomes

6. Conclusion

7. References

1. **INTRODUCTION**

*Mini Project Title*

The "Serverless Content Management System with AWS App Runner" project addresses the need for an efficient, scalable, and easily maintainable web application deployment process. Traditional hosting and deployment methods require manual configuration and server management, which can be both time-consuming and prone to error. With the increasing demand for dynamic content delivery and responsive applications, there arises a need for a streamlined, automated approach.

This project aims to harness the capabilities of AWS App Runner, a fully managed service that enables developers to deploy web applications directly from source code repositories like GitHub without the need to manage infrastructure. By leveraging serverless architecture, the project reduces operational overhead, improves scalability, and ensures continuous deployment with minimal effort.

The core application in this project is a personal portfolio website that showcases content such as projects, skills, and achievements. This content can be updated through source code changes, which are automatically reflected on the deployed application thanks to App Runner's integration with GitHub.

The serverless approach is especially beneficial for individual developers and small teams, offering a cost-effective and maintenance-free deployment pipeline. The solution is also designed to scale automatically based on traffic, providing high availability without manual intervention.

Overall, the project demonstrates how cloud-native tools like AWS App Runner can transform the deployment experience by removing the complexities of infrastructure management and enabling rapid, reliable web application delivery.

**2. AWS Services Used as part of the Project**

To build and deploy the serverless content management system, the following AWS services were integrated:

1. **AWS App Runner**  
   AWS App Runner is a fully managed service that makes it easy to deploy containerized web applications and APIs directly from source code repositories like GitHub. It handles building, deploying, load balancing, and scaling of applications automatically without any infrastructure management.
2. **AWS Identity and Access Management (IAM)**  
   IAM was used to securely control access to AWS services. Roles and policies were configured to allow App Runner to pull code from the GitHub repository and deploy it. IAM ensures that only authorized users and services can interact with the deployment resources.
3. **AWS CloudWatch**  
   CloudWatch was used for monitoring the deployed application. It provided logs that helped in debugging and ensuring the health of the application. Through CloudWatch, uptime, performance metrics, and error tracking were effectively managed.

These services combined to deliver a streamlined, scalable, and efficient serverless deployment pipeline for the portfolio project, enabling real-time updates and automated operations.

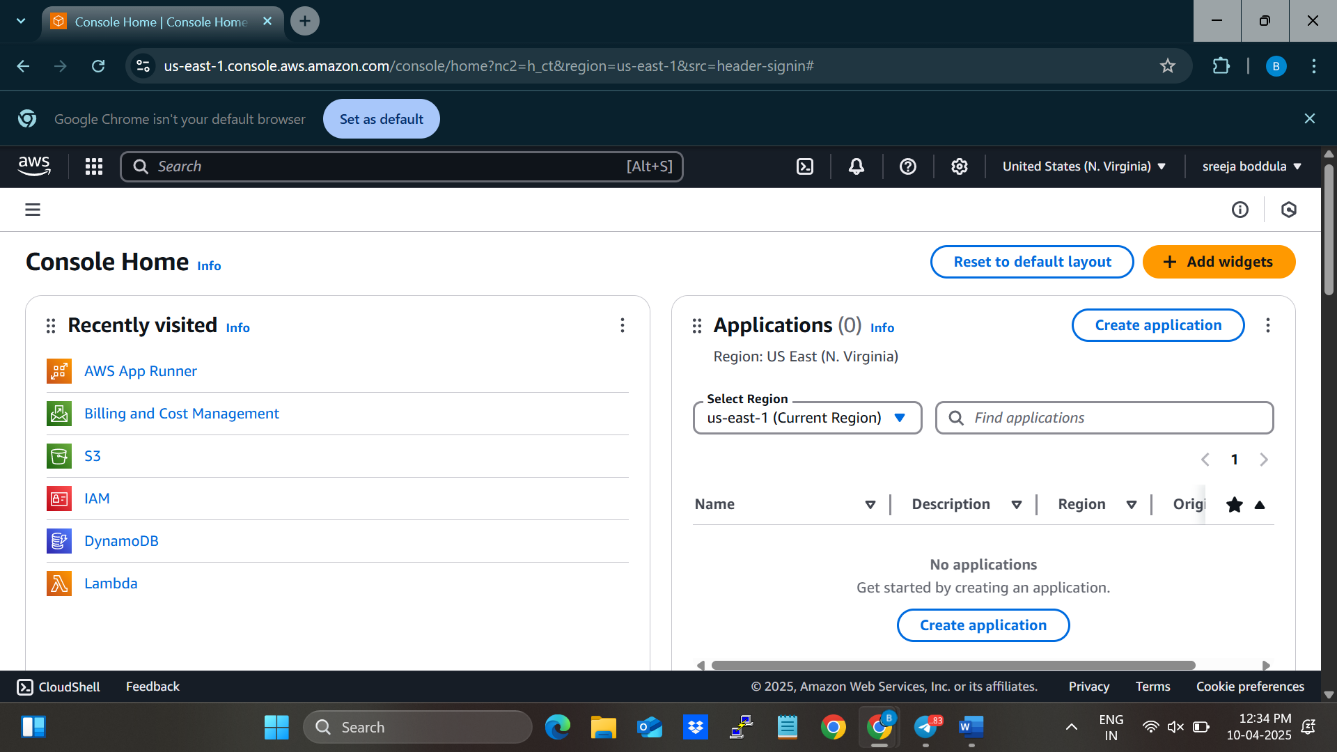
**3. Steps Involved in Solving the Project Problem Statement**

To address the challenge of deploying and managing a content management system without maintaining backend infrastructure, the following steps were taken:

1. **Website Development**: Designed and developed a portfolio website with dynamic/static content suitable for deployment.
2. **GitHub Integration**: Created a public repository on GitHub to host the source code, enabling version control and continuous updates.
3. **AWS App Runner Setup**: Logged into the AWS Management Console and selected App Runner as the service for deploying the web application.
4. **Repository Connection**: Connected the GitHub repository to App Runner using the integrated OAuth authorization, allowing AWS to fetch the source code.
5. **Configuration**: Set up build and start commands in App Runner settings. Selected runtime environment and defined port configurations.
6. **Deployment**: Initiated deployment directly from GitHub via App Runner. The app was built, deployed, and a public URL was generated automatically.
7. **Monitoring**: Utilized AWS CloudWatch to monitor logs and check application health. Confirmed application uptime and successful auto-scaling features.

**4. Stepwise Screenshots with brief description**

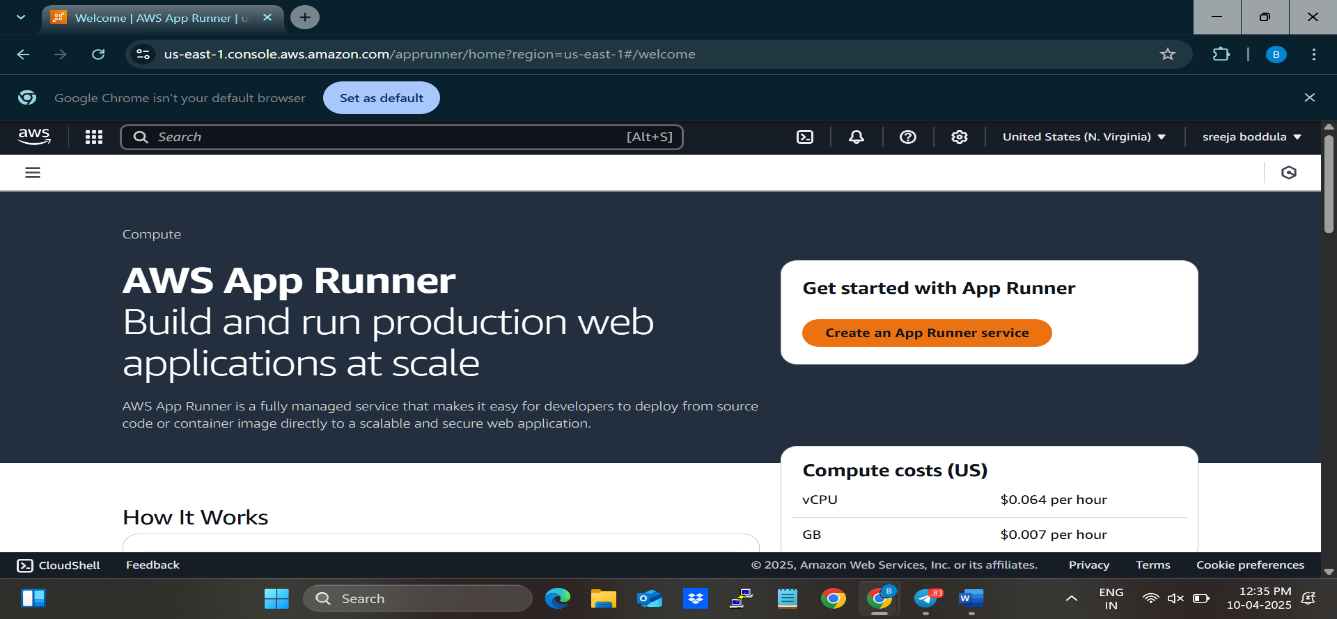
1.Open **AWS console**.



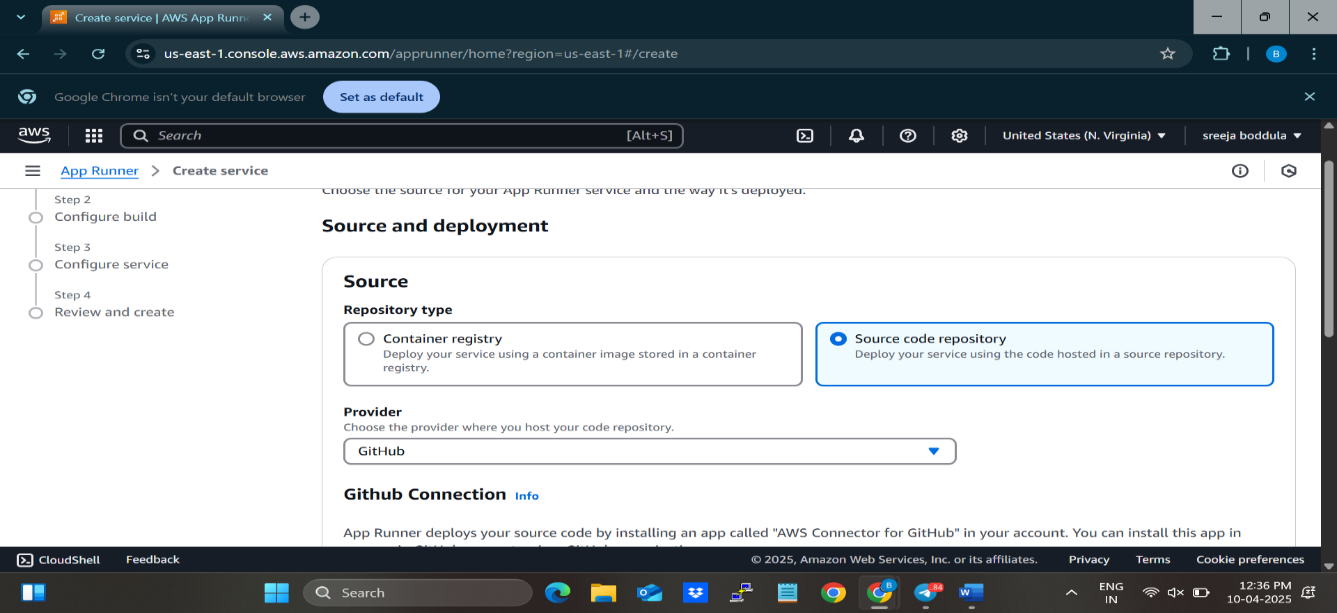
1. Search for **AWS App Runner**.



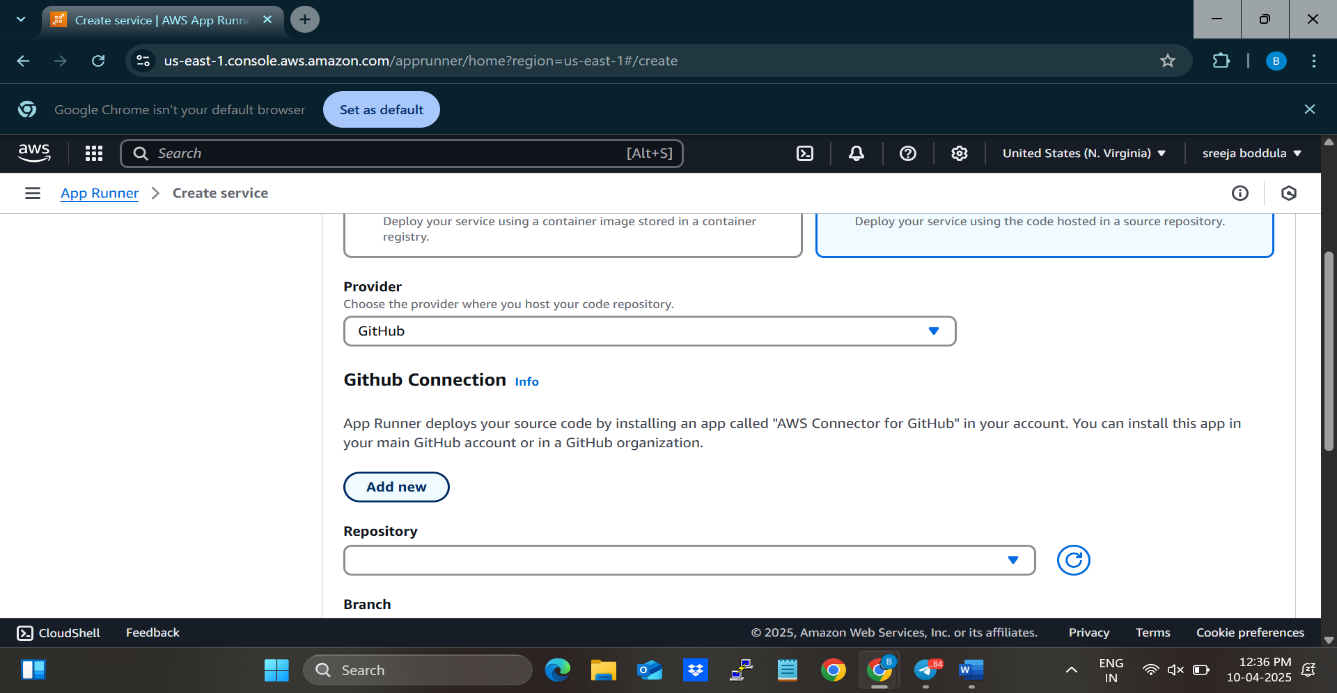
3.Click on **Create an App Runner service** button.



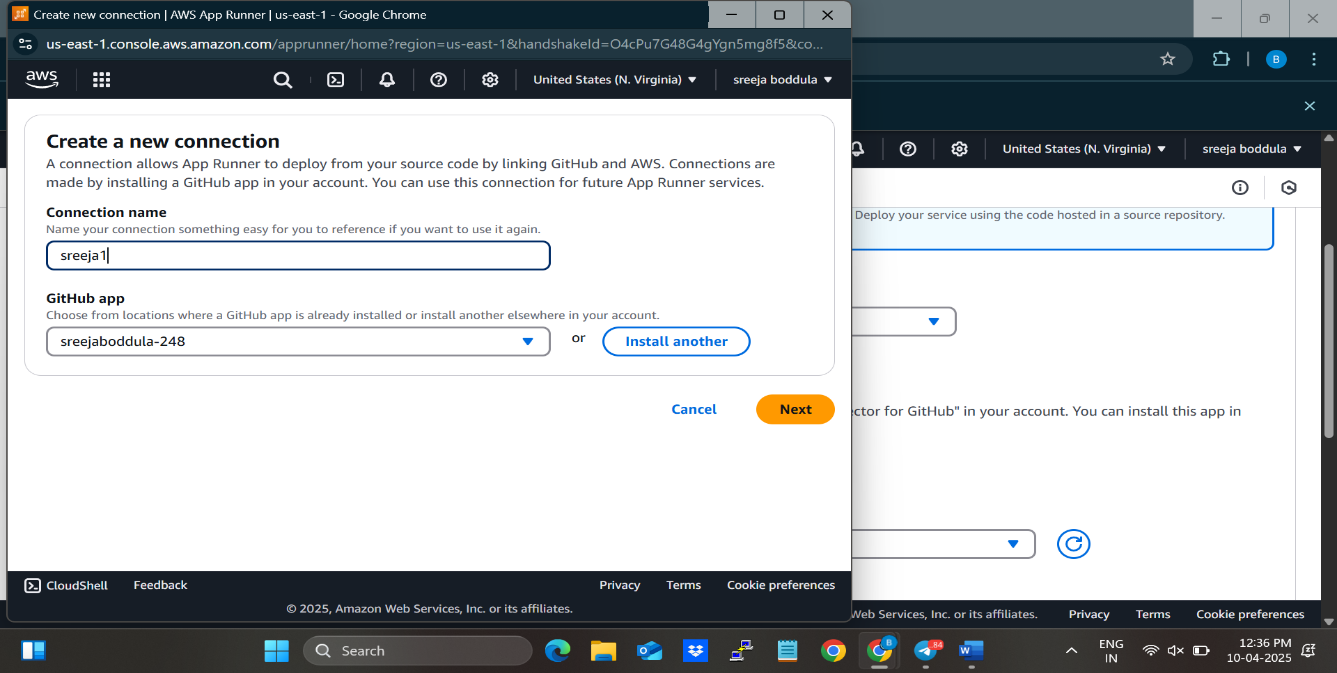
4.Select **Source code repository.**



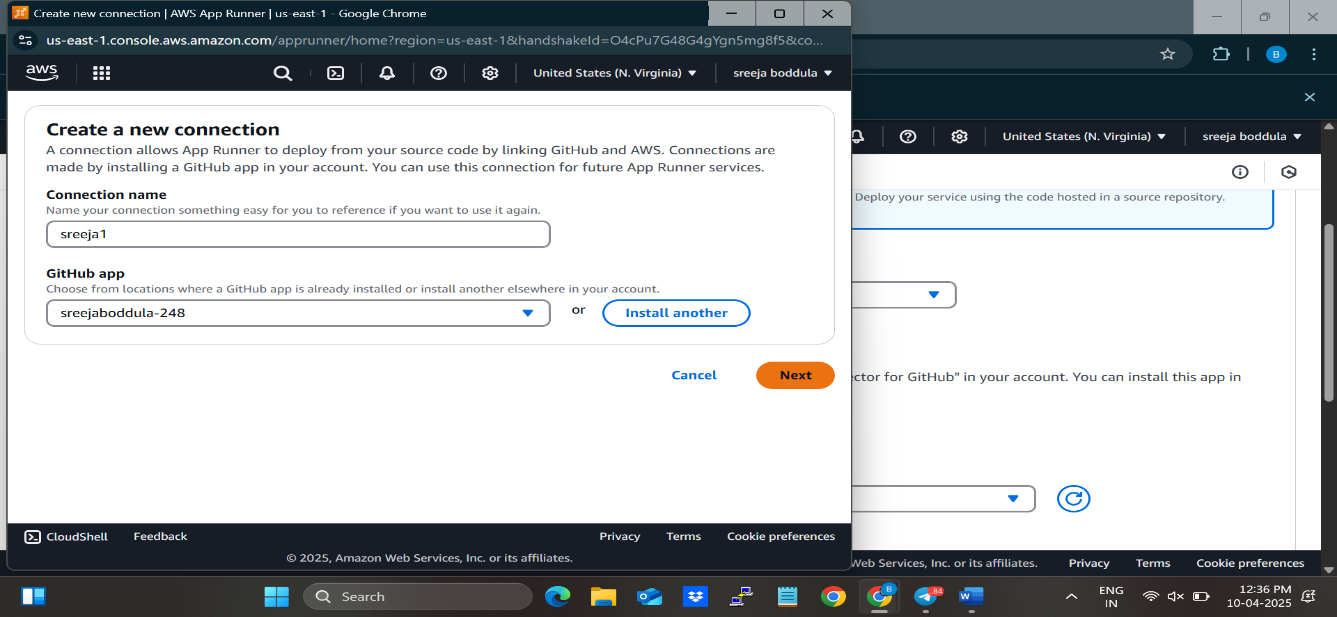
5.Click on **Add new** button.



6. Give a connection name and connect it to your GitHub app.

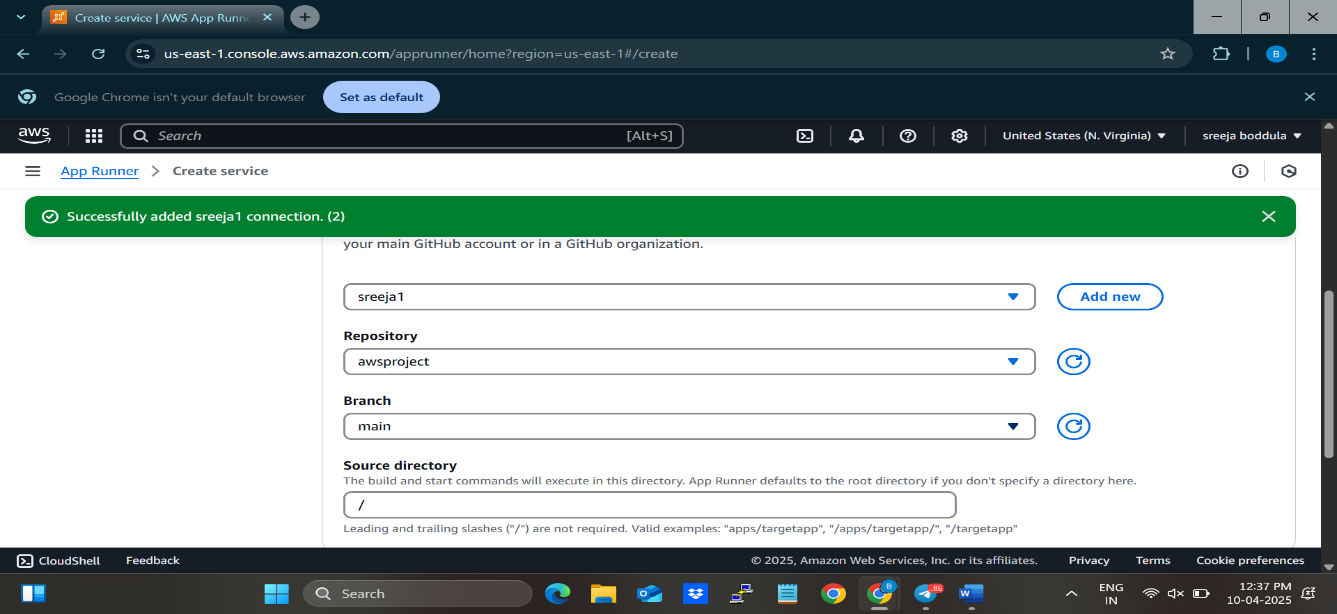


7.Click on **Next** button.

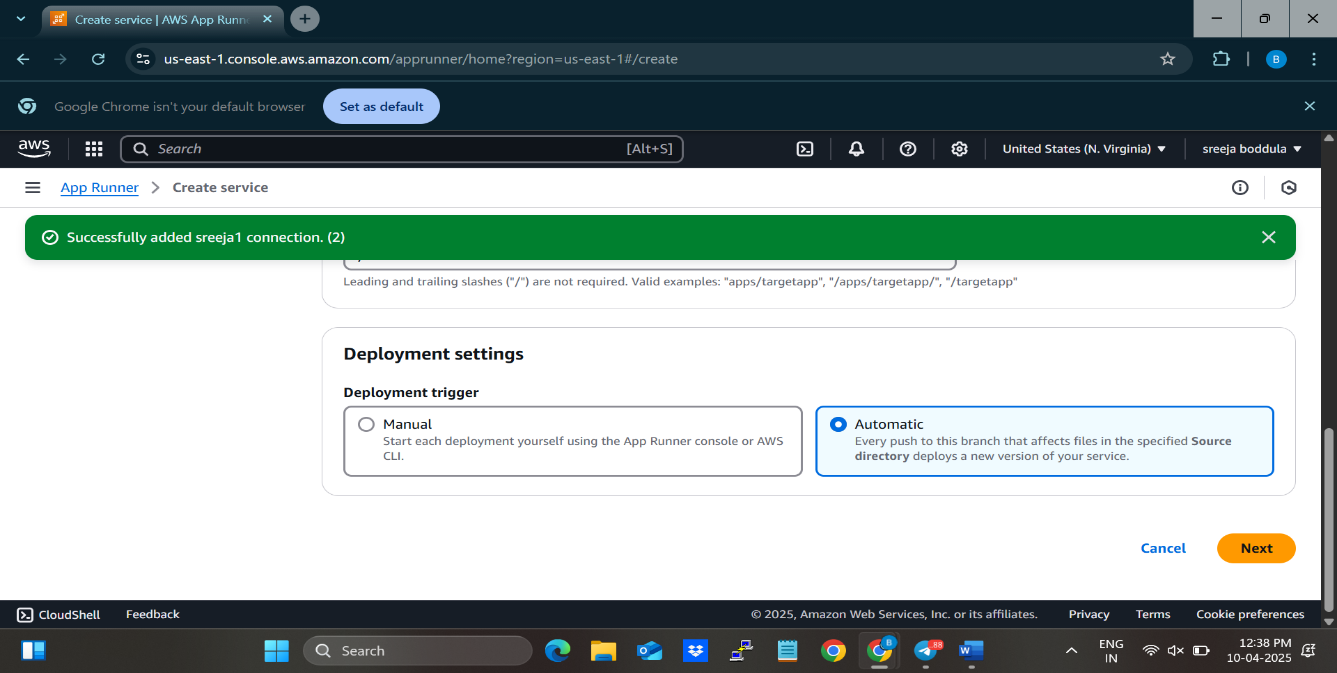


8.You can see the message that Connection is made successfully.

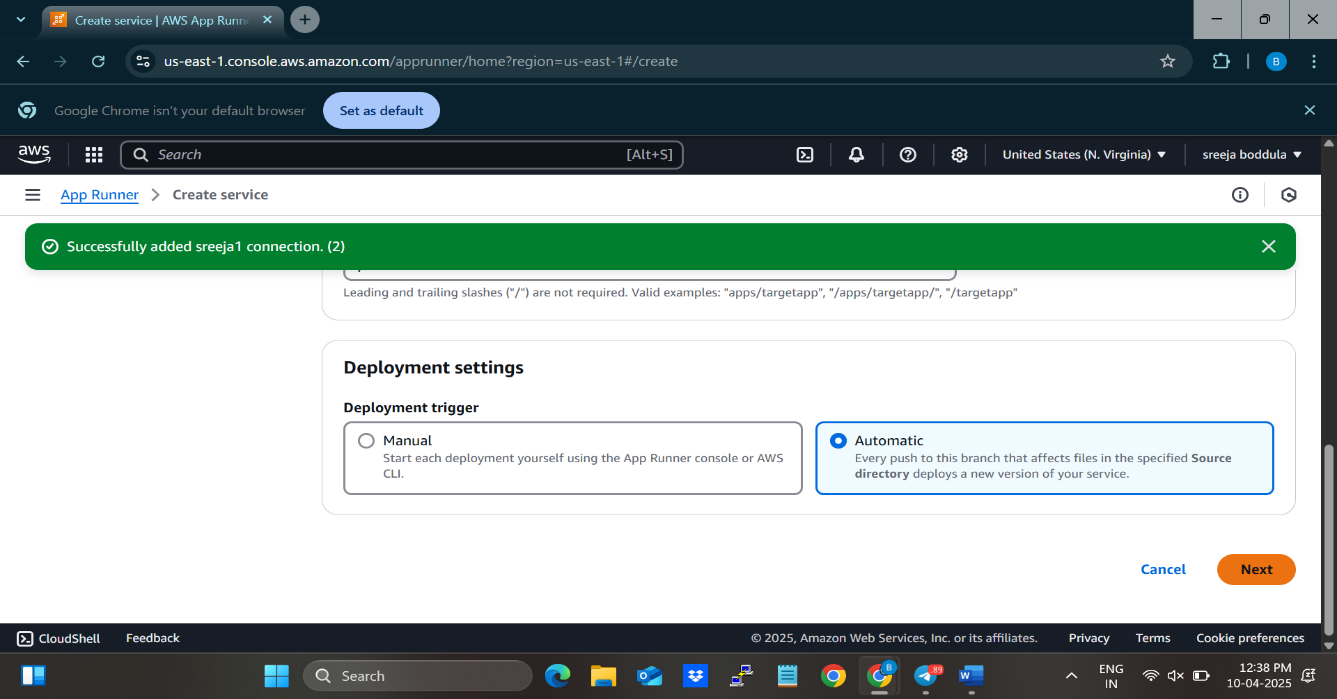
9. Select the repository where your project is available.



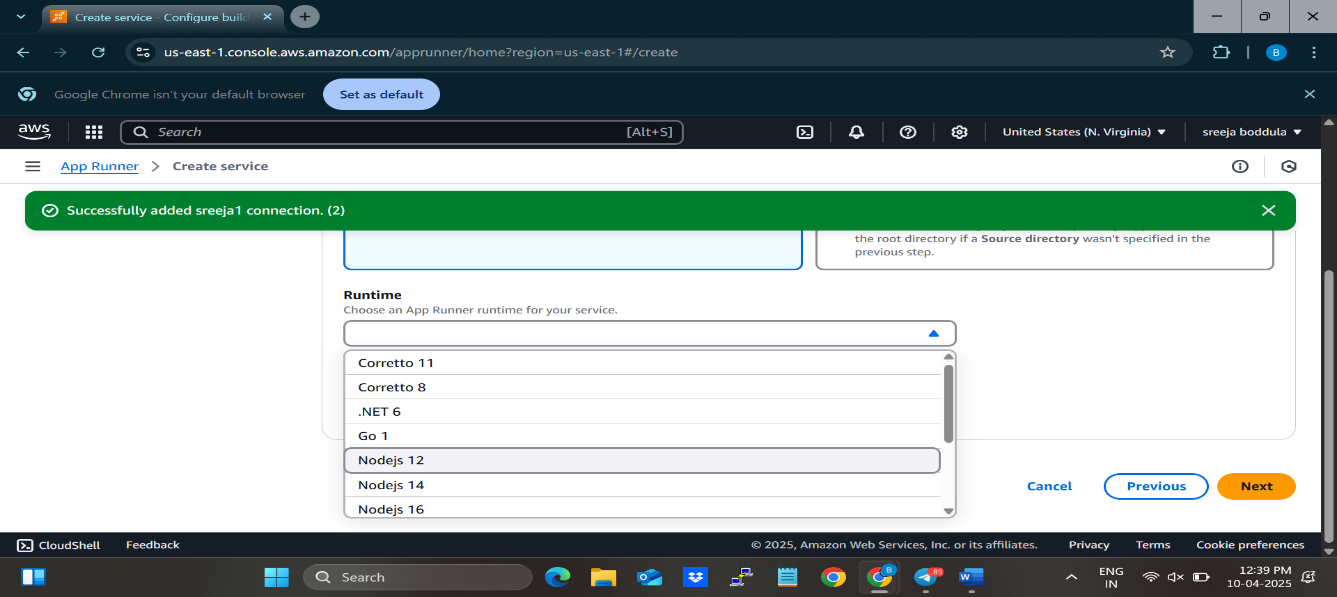
10.Select **Automatic** option.



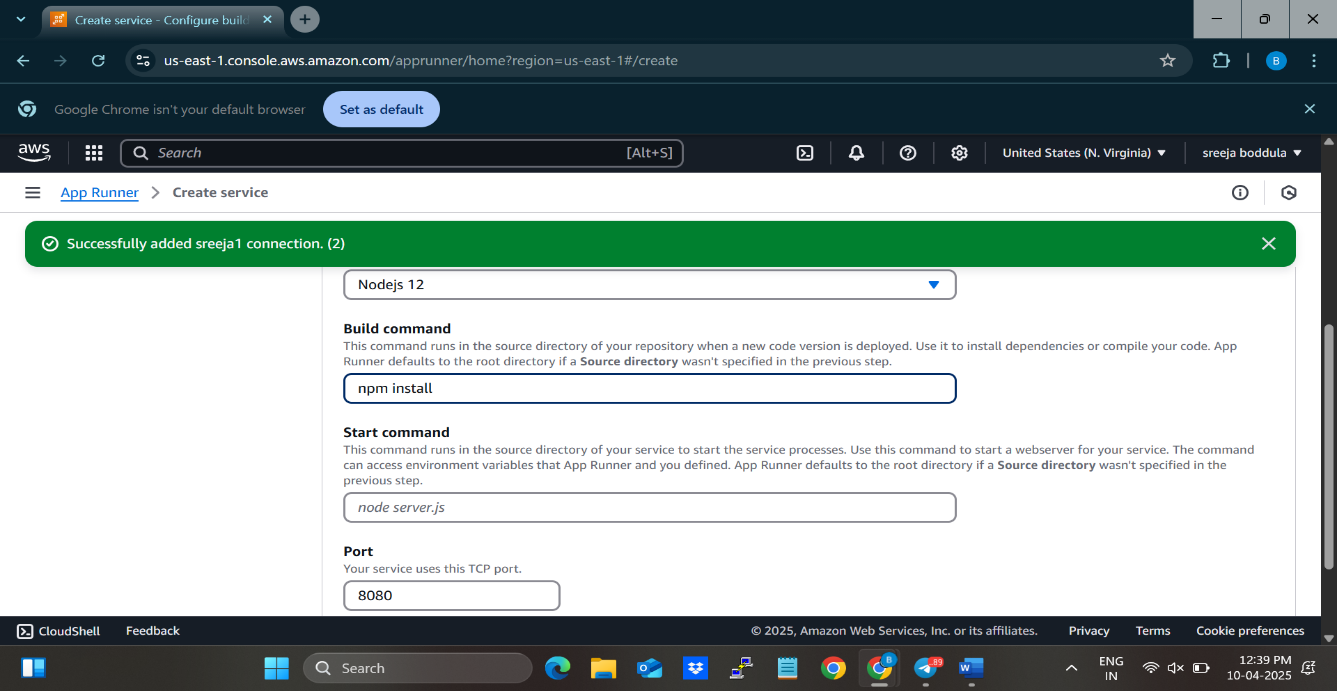
11. Click on **Next** button at the bottom-right.



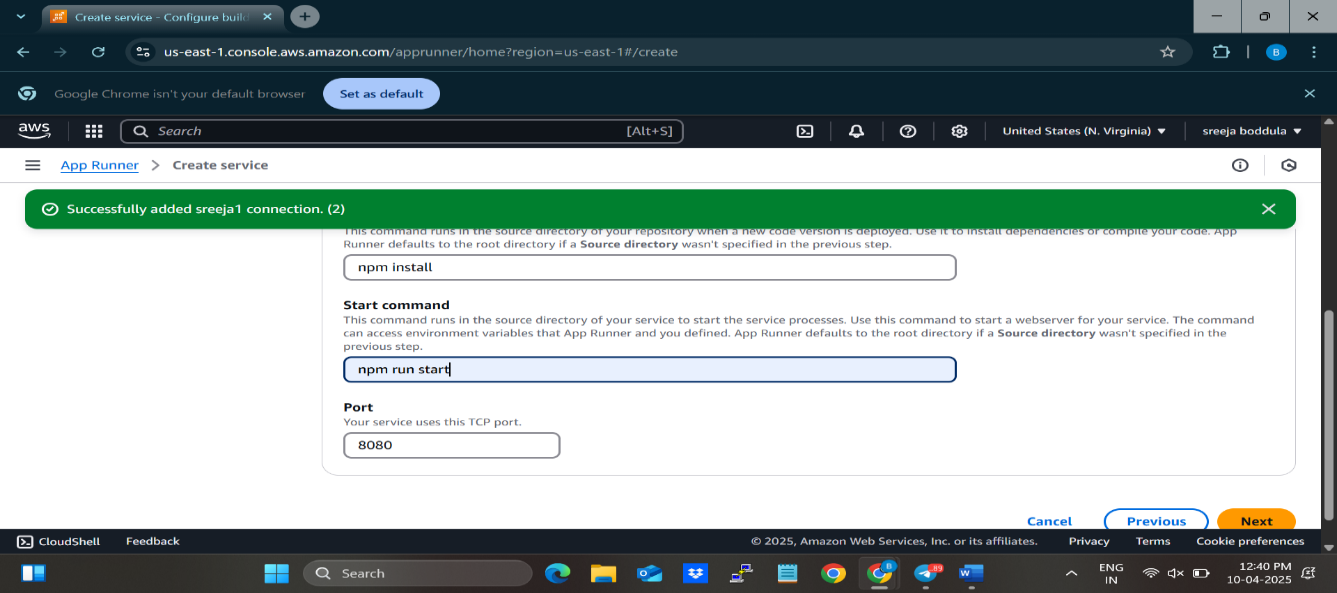
12.In the **Runtime** choose **Nodejs 12.**



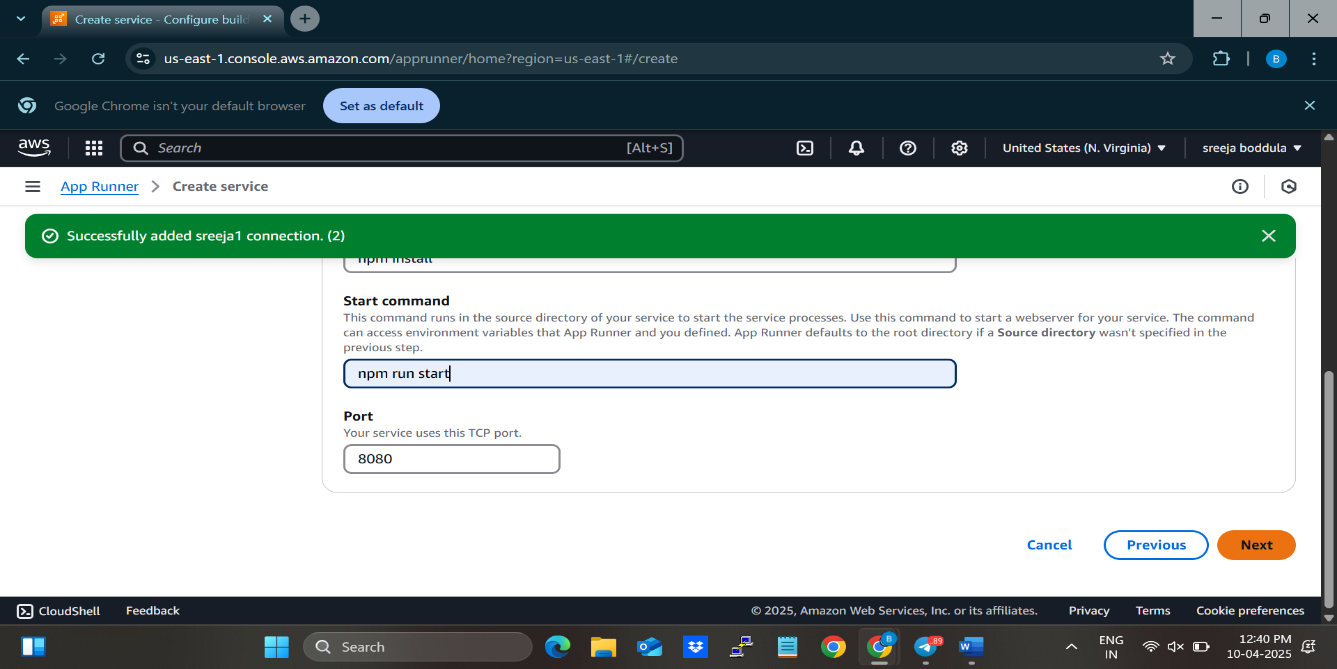
13. In the Build Command type **npm install.**



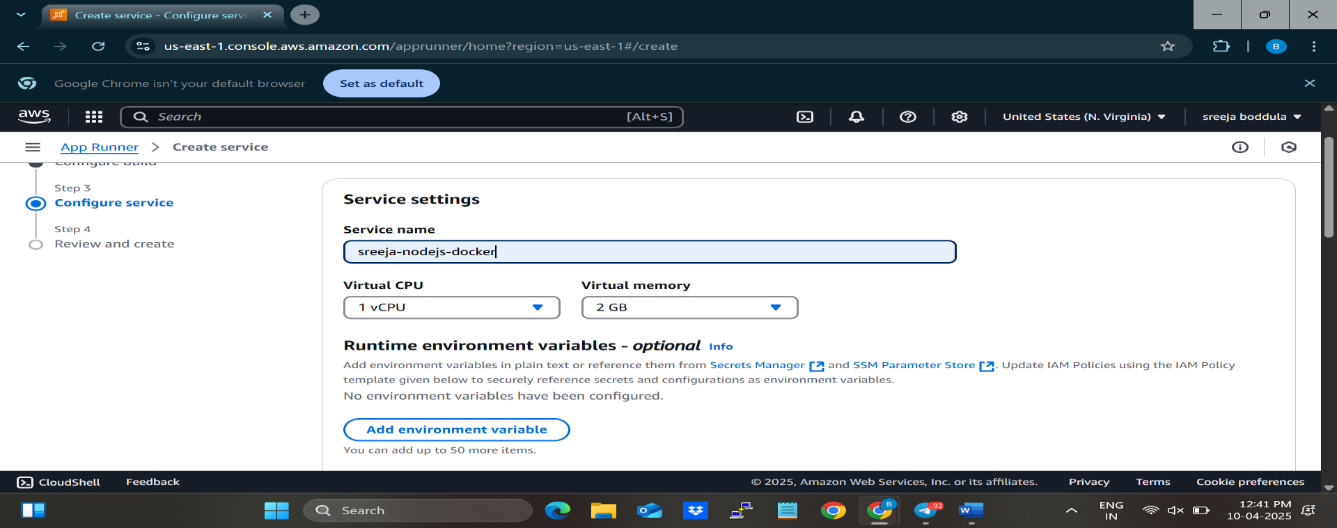
14. In the Start command type **npm run start.**



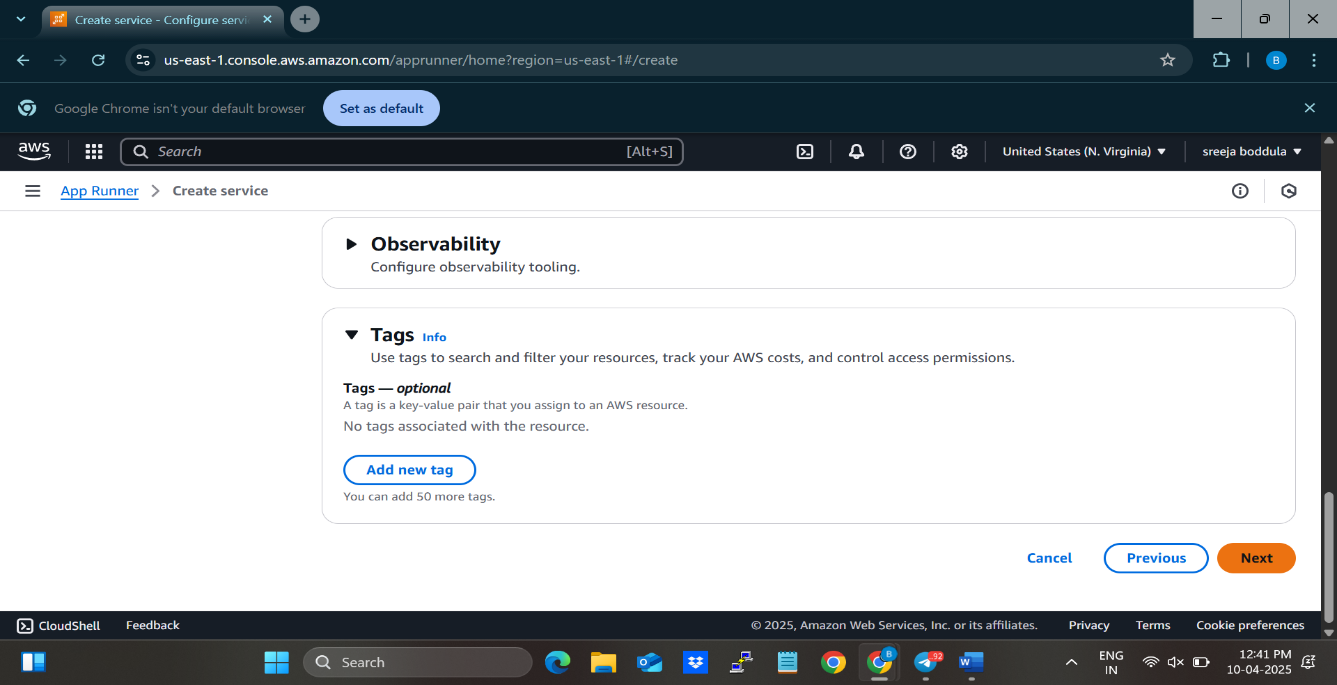
15.Click on **Next** button at the bottom-right of the page.



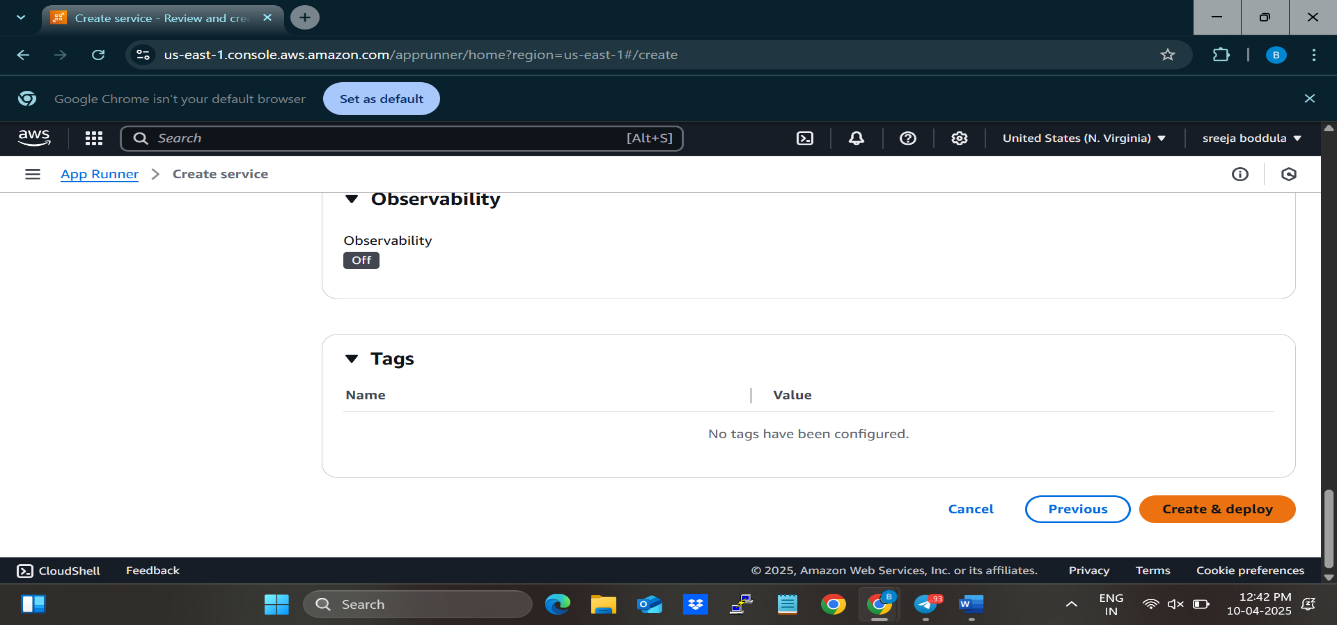
16. Give a name to the **Service name.**



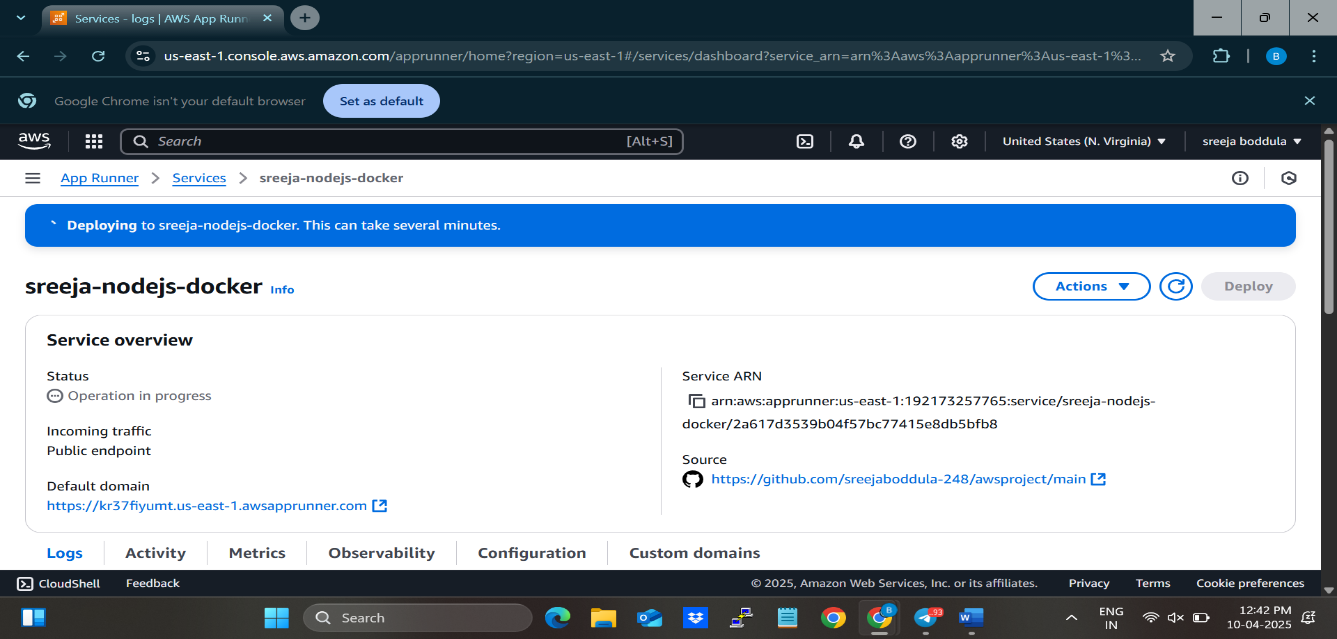
17. Scroll down and click on **Next** button.



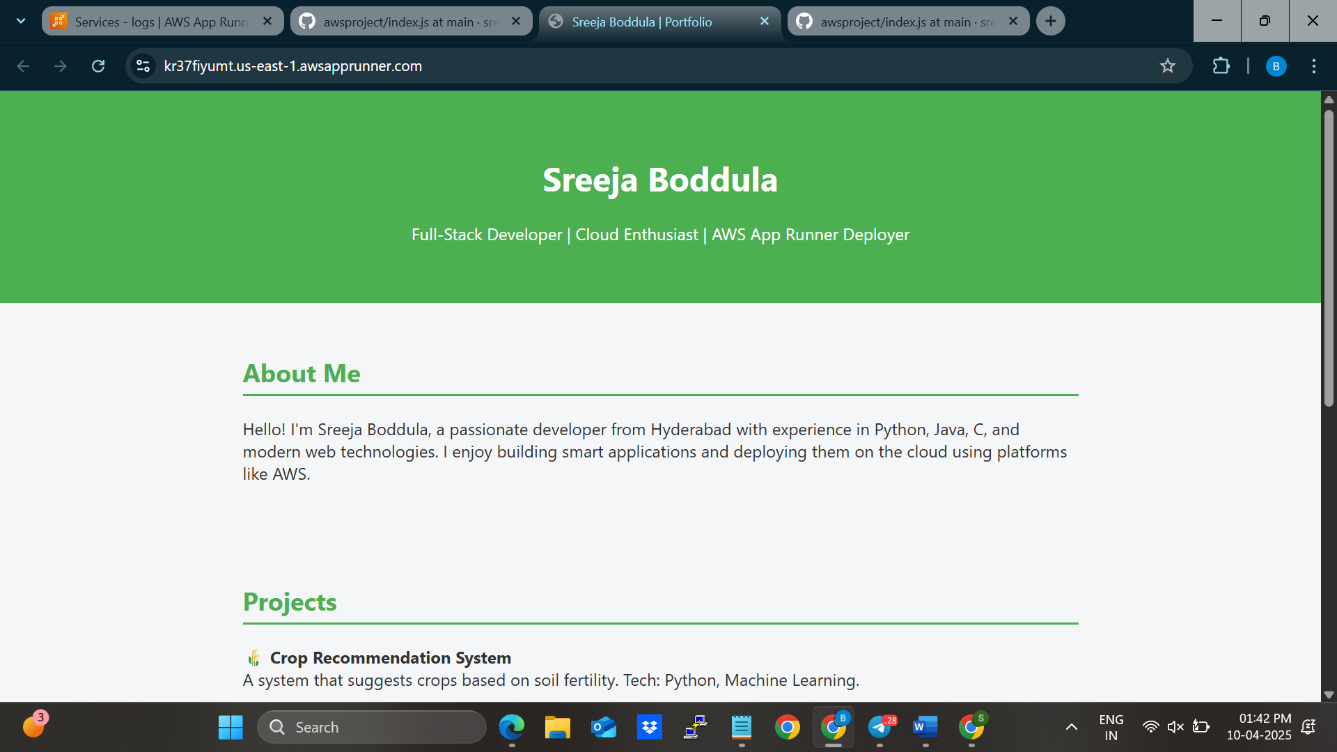
18. Click on **create & deploy** button.

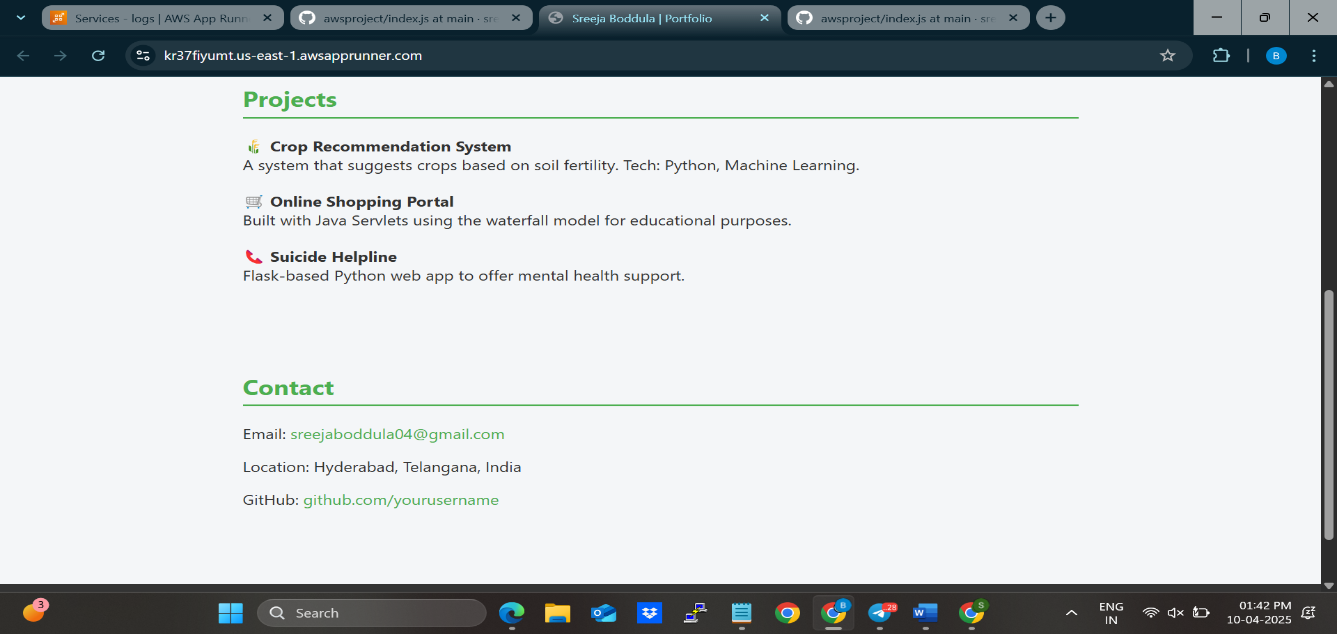


19. Copy the url under the **Default domain** and paste it in the browser.



20. After pasting the url in the browser we can see the following outputs.





**5. Learning Outcomes**

Working on this project provided valuable hands-on experience and deepened understanding in several key areas of modern cloud-based application development. The following are the major learning outcomes:

**✅ Understanding of Serverless Architecture**

* Gained practical knowledge of designing applications without managing servers.
* Learned how serverless components scale automatically and reduce infrastructure maintenance.

**✅ Hands-on Experience with AWS Services**

* Developed proficiency in using a variety of AWS services like App Runner, Lambda, DynamoDB, S3, and API Gateway.
* Understood how to integrate these services to build a fully functional, production-ready system.

**✅ Containerization and Deployment**

* Learned how to containerize applications using Docker.
* Experienced seamless deployment of containerized applications using AWS App Runner.

**✅ Backend API Development**

* Built and deployed RESTful APIs using AWS Lambda and API Gateway.
* Learned how to handle backend logic in a stateless and event-driven environment.

**✅ Cloud Security and Access Control**

* Gained insight into configuring AWS IAM for secure resource access.
* Learned how to manage access permissions and secure APIs effectively.

**✅ Monitoring and Logging**

* Understood how to use Amazon CloudWatch for log analysis and performance monitoring.
* Learned the importance of observability in maintaining cloud applications.

**✅ Real-world Application Design**

* Improved skills in designing a practical CMS used in real-world scenarios.
* Gained confidence in building scalable, maintainable, and cost-efficient web applications using modern technologies.

**6. Conclusion**

The development of the Serverless Content Management System using AWS App Runner successfully demonstrates the power and flexibility of modern serverless architectures. By leveraging AWS services such as App Runner, Lambda, DynamoDB, S3, and API Gateway, the project eliminates the need for manual infrastructure management while ensuring high scalability, availability, and performance.

This serverless approach not only simplified deployment and reduced operational overhead but also allowed for faster development cycles and easier maintenance. The CMS is capable of handling dynamic content creation, secure media management, and seamless user interactions, making it suitable for a wide range of real-world applications such as blogs, news portals, and internal knowledge bases.

Overall, the project highlights the effectiveness of cloud-native solutions and underlines how serverless platforms like AWS App Runner can be used to build robust, cost-efficient, and future-ready applications with minimal effort. It also provided an excellent opportunity to gain practical experience with cloud technologies and adopt industry-standard best practices.

**7. References**

**[1] Amazon Web Services, "AWS App Runner – Documentation," [Online]. Available:** [**https://docs.aws.amazon.com/apprunner/**](https://docs.aws.amazon.com/apprunner/)**. [Accessed: Apr. 10, 2025].**

**[2] Amazon Web Services, "AWS Lambda – Developer Guide," [Online]. Available:** [**https://docs.aws.amazon.com/lambda/**](https://docs.aws.amazon.com/lambda/)**. [Accessed: Apr. 10, 2025].**

**[3] Amazon Web Services, "Amazon DynamoDB – Developer Guide," [Online]. Available:** [**https://docs.aws.amazon.com/amazondynamodb/**](https://docs.aws.amazon.com/amazondynamodb/)**. [Accessed: Apr. 10, 2025].**

**[4] Amazon Web Services, "Amazon API Gateway – Documentation," [Online]. Available:** [**https://docs.aws.amazon.com/apigateway/**](https://docs.aws.amazon.com/apigateway/)**. [Accessed: Apr. 10, 2025].**

**[5] Amazon Web Services, "Amazon Simple Storage Service (S3) – Documentation," [Online]. Available:** [**https://docs.aws.amazon.com/s3/**](https://docs.aws.amazon.com/s3/)**. [Accessed: Apr. 10, 2025].**

**[6] Amazon Web Services, "AWS Identity and Access Management (IAM) – Documentation," [Online]. Available:** [**https://docs.aws.amazon.com/iam/**](https://docs.aws.amazon.com/iam/)**. [Accessed: Apr. 10, 2025].**

**[7] Amazon Web Services, "Amazon CloudWatch – Documentation," [Online]. Available:** [**https://docs.aws.amazon.com/cloudwatch/**](https://docs.aws.amazon.com/cloudwatch/)**. [Accessed: Apr. 10, 2025].**

**[8] Docker Inc., "Docker Documentation," [Online]. Available: https://docs.docker.com/. [Accessed: Apr. 10, 2025].**

**[9] Amazon Web Services, "AWS Serverless Application Model (SAM) – Documentation," [Online]. Available:** [**https://docs.aws.amazon.com/serverless-application-model/**](https://docs.aws.amazon.com/serverless-application-model/)**. [Accessed: Apr. 10, 2025].**

**[10] Amazon Web Services, "AWS Compute Blog," [Online]. Available:** [**https://aws.amazon.com/blogs/**](https://aws.amazon.com/blogs/)**. [Accessed: Apr. 10, 2025].**