Absolutely! Here's a comprehensive and well-structured LinkedIn post for your DevOps CI/CD project:

---

🚀 \*\*Project Showcase: Automated CI/CD Pipeline with Terraform, GitHub, Jenkins, and Docker on AWS\*\* 🚀

I’m excited to share a recent project where I integrated various DevOps tools to create a robust CI/CD pipeline for hosting an application. Here's a step-by-step breakdown of the process:

---

### \*\*Introduction\*\*

In this project, I utilized Terraform, AWS, GitHub, Jenkins, and Docker to establish a seamless CI/CD pipeline. This integration streamlined the deployment process, ensuring efficient and reliable application hosting.

---

### \*\*Step 1: Creating AWS Infrastructure with Terraform\*\*

\*\*Tools Used:\*\*

- \*\*Terraform:\*\* Infrastructure as Code software tool.

- \*\*Visual Studio Code:\*\* Source-code editor for managing Terraform scripts.

\*\*Objective:\*\*

Provision and manage AWS infrastructure including EC2 instances, VPCs, and security groups.

\*\*Process:\*\*

1. \*\*Set Up Your Terraform Environment\*\*

- Ensure Terraform is installed on your machine.

- Install Visual Studio Code.

2. \*\*Write the Terraform Code\*\*

- Open Visual Studio Code and create a new directory for your Terraform scripts (e.g., 'LAUNCHING EC2').

3. \*\*Initialize and Apply Terraform\*\*

- Open a terminal in CMD.

- Run `terraform plan` to preview the changes.

- Run `terraform apply` to create the infrastructure.

---

### \*\*Step 2: Forking the Application Code from GitHub\*\*

\*\*GitHub Repository:\*\*

- URL: [DevOps-Project](https://github.com/Ashugore-github/DevOps-Project)

\*\*Process:\*\*

- Fork the application repository to your GitHub account.

- Create a `Dockerfile` in the repository for building the application image.

---

### \*\*Step 3: Installing Jenkins and Docker on RHEL 9 EC2 Instance\*\*

\*\*Process:\*\*

- \*\*Install Jenkins:\*\* Follow the installation guide and create a Jenkins account.

- \*\*Install Docker:\*\* Ensure Docker is installed and running on the EC2 instance.

---

### \*\*Step 4: Integrating GitHub and Jenkins\*\*

\*\*Process:\*\*

1. \*\*Create a Webhook:\*\*

- Set up a webhook on the GitHub repository to trigger Jenkins builds.

2. \*\*Configure Jenkins:\*\*

- Enable the checkbox for Pipeline → Build Triggers → GitHub hook trigger for GITScm polling.

- Generate a URL snippet from Pipeline Syntax for Jenkins to access the GitHub repository.

- Verify the GitHub files in the Jenkins workspace.

---

### \*\*Step 5: Building an Image and Pushing it to Docker Hub\*\*

\*\*Process:\*\*

1. \*\*Build Docker Image:\*\*

- Use commands to build the Docker image with both the latest tag and a build-specific tag (`$BUILD\_ID`).

2. \*\*Push Image to Docker Hub:\*\*

- Create Docker Hub credentials in Jenkins.

- Generate a URL snippet for authentication.

- Push the images to Docker Hub using the appropriate commands.

3. \*\*Cleanup:\*\*

- Delete pre-existing images and Docker containers to ensure a clean environment.

---

### \*\*Step 6: Creating a Container and Accessing the Application\*\*

\*\*Process:\*\*

1. \*\*Create a Docker Container:\*\*

- Run a container named `demoapp` using the latest image, exposing it on port 3000.

2. \*\*Access the Application:\*\*

- Copy the instance IP and append `:3000` to it.

- Open the URL in a browser to access the application (`<instanceIP:3000>`).

---

This project was an incredible learning experience, allowing me to deepen my understanding of DevOps practices and tool integrations. I’m looking forward to applying these skills to future projects and continuing to explore the vast possibilities within the DevOps landscape.

#DevOps #CI/CD #Terraform #AWS #Jenkins #Docker #Automation #CloudComputing #InfrastructureAsCode #GitHub

---

Feel free to customize any part of this post to better fit your style or to add more details.