Module 2 Summary: Jenkins Fundamentals – Building CI Pipeline Manually

Congratulations on completing Module 2 of the Essentials of Jenkins for DevOps Practitioners course! In this module, we explored the fundamentals of Jenkins and learned how to manually build a CI pipeline for a Node.js application.

We started by familiarizing ourselves with the Jenkins Dashboard and UI, navigating through the key components and understanding the role of each section. We also explored the System Configuration options, which allow us to tailor our Jenkins instance to suit our specific requirements.

Next, we dove into a real-world use case, the Craftista frontend application, and learned how to fork the Git repository to create our own copy of the codebase. We discussed the significance of Express.js as a web application framework for Node.js and its benefits in building scalable and efficient server-side applications.

Throughout the module, we focused on setting up a modern CI pipeline for the Craftista frontend application using Jenkins and its ecosystem of plugins and tools. We walked through the process step by step, leveraging cutting-edge technologies to create an efficient and automated CI workflow.

Key topics covered in the video lessons and lab guide:

1. **Creating and configuring Jenkins Jobs**: We learned how to create and configure Jenkins jobs for building, testing, and packaging the Craftista frontend application. We explored the various options and settings available in Jenkins job configuration.
2. **Connecting Jobs with Upstream and Downstream Configs**: We discovered how to establish relationships between Jenkins jobs using upstream and downstream configurations. This allows us to create a sequence of jobs that depend on each other, forming a pipeline.
3. **Visualizing Connected Jobs with Build Pipeline Plugin**: We utilized the Build Pipeline Plugin to create a visual representation of our connected jobs. This plugin provides a graphical view of the pipeline, making it easier to understand the flow and dependencies between jobs.

Throughout the module, we emphasized the importance of containerization using Docker, defining pipelines as code with Jenkinsfiles, leveraging multi-branch pipelines, and utilizing the Blue Ocean UI for a modern and intuitive Jenkins experience.

By completing this module, you have gained hands-on experience in manually building a CI pipeline for a Node.js application using Jenkins. You now have a solid

foundation in Jenkins fundamentals and are ready to explore more advanced concepts and techniques in the upcoming modules.

Key takeaways from Module 2:

- Understanding the Jenkins Dashboard, UI, and System Configuration options
- Forking a Git repository to create a copy of the codebase
- Creating and configuring Jenkins jobs for building, testing, and packaging a Node.js application
- Establishing relationships between jobs using upstream and downstream configurations
- Visualizing the pipeline using the Build Pipeline Plugin
- Containerizing the application with Docker
- Defining pipelines as code using Jenkinsfiles
- Leveraging multi-branch pipelines and the Blue Ocean UI

In the next module, we'll dive into the world of Declarative Pipelines and learn how to define our entire CI/CD pipeline as code using Jenkinsfiles.

Keep up the excellent work, and let's continue our journey towards mastering Jenkins for DevOps!