Student Study Plan: Docker and Containerization

1. Introduction and Learning Goals

This study plan aims to help students understand the concepts, evolution, and practices of Docker and containerization. By the end of the study period, students will:
- Know the history and development of container technology.
- Understand the differences between virtual machines and containers.
- Learn how Docker works and how to create, manage, and secure containers.
- Apply containerization principles in modern software development and deployment.

2. Week-by-Week Study Outline

Week 1: The Origins of Containerization

Topics:
- The need for process isolation in operating systems.
- Early systems like chroot, FreeBSD Jails, and Solaris Zones.
- The evolution from virtual machines to lightweight containers.

Learning Outcome:
Students will be able to explain how containerization evolved and why it became an important technology.

Activity:
Create a short timeline or infographic showing the major milestones in container technology.

Week 2: Virtual Machines vs. Containers

Topics:
- Understanding virtual machine architecture.
- Comparing VMs and containers: performance, scalability, and resource usage.
- How containers achieve isolation through shared kernels.

Learning Outcome:
Students can differentiate VMs from containers and discuss the advantages and trade-offs.

Activity:
Draw a diagram comparing VM and container architecture.

Week 3: The Docker Ecosystem

Topics:
- Overview of Docker architecture: Client, Daemon, and Registry.
- Docker images and containers explained.

- How Docker simplifies application deployment.

Learning Outcome:
Students will understand Docker components and their roles.

Activity:
Install Docker and run a 'Hello World' container.

Week 4: Building and Running Containers

Topics:
- Writing and understanding a Dockerfile.
- Common Docker commands: build, run, stop, start, and remove.
- Managing container lifecycle.

Learning Outcome:
Students can build and execute containerized applications using Docker commands.

Activity:
Create a simple Node.js or Python app, write a Dockerfile, and run it inside a container.

Week 5: Networking and Storage in Docker

Topics:
- How containers communicate using networks.
- The role of bridge networks and exposed ports.
- Managing persistent data using volumes and bind mounts.

Learning Outcome:
Students will know how to connect containers and store data persistently.

Activity:
Create a container that connects to a database container and stores persistent data using volumes.

Week 6: Multi-Container Applications

Topics:
- Introduction to Docker Compose.
- Using YAML files to define multi-service applications.
- Managing environments for web apps and databases.

Learning Outcome:
Students can create and run multi-container setups efficiently.

Activity:
Build a simple WordPress or Flask + PostgreSQL project using Docker Compose.

Week 7: Orchestration and Scaling

Topics:
- Why orchestration is needed in production.
- Introduction to Kubernetes and Docker Swarm.
- Concepts of clusters, pods, and self-healing containers.

Learning Outcome:
Students can explain how orchestration tools automate deployment and scaling.

Activity:
Explore Kubernetes using Minikube or an online simulator.

Week 8: Security and Best Practices

Topics:
- Importance of image scanning for vulnerabilities.
- Managing secrets and credentials securely.
- Using non-root users and multi-stage builds for optimization.
- Maintaining small and efficient images.

Learning Outcome:
Students understand how to secure Docker containers.

Activity:
Scan a Docker image for vulnerabilities using Trivy or Docker Scout and document results.

Week 9: Docker in CI/CD Pipelines

Topics:
- Integrating Docker in CI/CD workflows.
- Building automated pipelines with Jenkins or GitHub Actions.
- The process of building, testing, pushing, and deploying container images.

Learning Outcome:
Students understand how Docker fits into DevOps and automation.

Activity:
Create a CI/CD pipeline diagram showing container deployment stages.

3. Recommended Learning Resources

- Books: 'Docker Deep Dive' by Nigel Poulton; 'The Docker Book' by James Turnbull.
- Online Tutorials: Docker Docs (https://docs.docker.com/), Play with Docker, Katacoda Labs.
- Tools: Docker Desktop, Docker Hub, Minikube, Kubernetes Dashboard.

4. Evaluation

- Quizzes: Weekly short-answer questions.
- Practical Tasks: Build and deploy containerized applications.
- Final Project: Develop a mini project using Docker Compose showing container management and security.

5. Conclusion

Docker and containerization have transformed how software is built, tested, and deployed. Through this study plan, students gain knowledge from historical foundations to modern best practices, preparing them for real-world software engineering and DevOps environments.