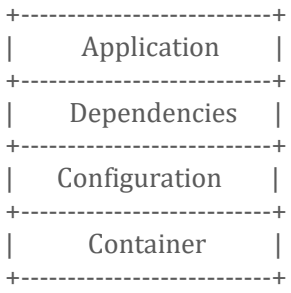
Introduction to Container-based Delivery and Its Benefits

Once upon a time, in the land of software development, applications were deployed on physical servers or virtual machines. Each application had its own unique environment, with specific dependencies, libraries, and configurations. It was like each application lived in its own custom-built house. While this approach worked, it had its challenges. Moving applications between different environments was like relocating to a new house – it required careful planning, packing, and unpacking to ensure everything worked smoothly in the new location.

But then, a revolutionary concept emerged: container-based delivery. It was as if someone invented a magical box that could contain an application and all its dependencies in a compact and portable way. This magical box was called a container.



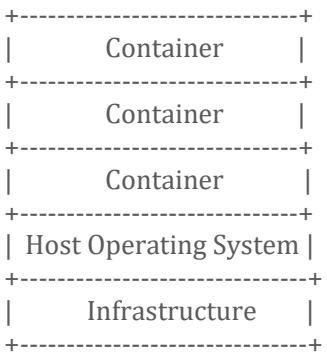
Imagine a container as a shipping container that you see on cargo ships. Just like how a shipping container can hold various goods and be easily transported from one place to another, a container in the software world holds an application and all its necessary components, making it easy to move and deploy across different environments.

Container-based delivery offers several key benefits:

1. **Portability**: Containers are like portable homes for applications. They encapsulate all the necessary dependencies, libraries, and configurations, making it easy to move applications between different environments, such as development, testing, and production. It's like packing your application and its essentials into a compact suitcase, ready to travel wherever it needs to go.
2. **Consistency**: With containers, you can ensure that your application runs consistently across different environments. It's like having a recipe book that precisely describes how to prepare a dish. No matter where you cook it, as long as you follow the recipe, the dish will turn out the same. Similarly, containers guarantee that your application will behave the same way, regardless of the underlying infrastructure.
3. **Isolation**: Containers provide a level of isolation between applications. Each container runs in its own isolated environment, with its own filesystem,

processes, and network resources. It's like giving each application its own dedicated apartment, preventing conflicts and ensuring that applications don't interfere with each other.

4. **Scalability**: Containers make it easier to scale your applications horizontally. Just like stacking multiple shipping containers on a cargo ship, you can run multiple instances of your containerized application across different nodes or servers. This allows you to handle increased traffic and demand by simply adding more containers, without the need to provision new physical servers.
5. **Efficiency**: Containers are lightweight and start up quickly compared to traditional virtual machines. They share the host operating system's kernel, which means they require fewer resources and can be spun up and down rapidly. It's like having a fleet of compact and fuel-efficient cars that can quickly navigate through traffic, compared to heavy and resource-intensive trucks.



Container-based delivery, powered by technologies like Docker and Kubernetes, has revolutionized the way applications are packaged, deployed, and managed. It has made the software development process more efficient, reliable, and scalable.

In the next lesson, we will explore how Jenkins integrates with Docker, one of the most popular container technologies. We will learn how to switch from traditional Jenkins agents to Docker-based agents and unlock the benefits of container-based delivery in our CI/CD pipelines.

Get ready to embark on an exciting journey into the world of containers and discover how Jenkins can streamline your container-based delivery processes!