Indian Institute of Technology Patna

CS-565 Cloud Computing

Assignment #3

Submission Date: 09th May 2024

Baskar Natarajan - 2403res19(IITP001799)

SEMESTER-1

MTECH AI & DSC

INDIAN INSTITUTE OF TECHNOLOGY PATNA.

1. Explain the concept of Network Function Virtualization (NFV) and its relationship with Software Defined Networking (SDN). Provide examples to illustrate how NFV and SDN work together to optimize network architecture.

a. Network Function Virtualization (NFV)

i. Existing System and Problems

- In traditional network setups, important functions like firewalls, routers, and load balancers are performed by specialized hardware appliances.
- This hardware-centric approach can be costly, inflexible, and difficult to scale.
- Vendor lock-in: Limited to functionalities offered by the specific vendor's hardware.

ii. How does NFV overcome existing Problems?

- NFV replaces dedicated hardware with software-based virtual instances running on standard servers.
- This virtualization allows for greater flexibility, scalability, and cost-effectiveness by enabling network functions to be deployed and managed more dynamically.

iii. How NFV Works?

- NFV involves virtualizing network functions, such as firewalls and routers, and running them on standard servers or in the cloud.
- These virtual instances can be deployed, scaled, and managed more efficiently compared to traditional hardware-based solutions.

iv. Advantages of NFV

Cost savings: NFV reduces the need for specialized hardware, leading to lower capital and operational expenses.

Flexibility: Virtualized network functions can be deployed and scaled more quickly and easily to meet changing demands.

Scalability: NFV enables elastic scaling of network resources, allowing organizations to adapt to fluctuating traffic loads. New services can be deployed quickly by spinning up VNFs on-demand.

Vendor Independence: NFV can run on any standard hardware, promoting vendor neutrality.

Simplified management: Centralized management and orchestration of virtualized functions streamline network operations.

Innovation: NFV fosters innovation by enabling rapid deployment and testing of new network services and applications.

v. Disadvantages of NFV

Performance concerns: Virtualized network functions may not always match the performance of dedicated hardware appliances.

Complexity: Implementing NFV requires expertise in virtualization technologies and network architecture.

Security risks: Virtualized environments may introduce new security challenges that need to be addressed

vi. Real Time example or Use-case for NFV

- An example of NFV in action is a telecommunications provider using virtualized network functions to deliver services like virtualized firewalls, virtual private networks (VPNs), and session border controllers (SBCs) to their customers.
- By virtualizing these functions, the provider can offer more flexible and scalable services while reducing costs.

b. Software Defined Networking (SDN)

- SDN separates the control plane (deciding how to route traffic) from the data plane (the actual data forwarding) in network devices.
- This allows for centralized, programmatic control of the network through an SDN controller. The controller acts like a conductor, telling network devices where to send traffic based on pre-defined policies.

i. What SDN does?

- SDN decouples the network control plane from the data plane, enabling centralized control and programmability of network resources.
- It separates the logic that determines how traffic is routed from the physical devices that forward the traffic.

ii. Why is it required?

- SDN must address the limitations of traditional networking, such as manual configuration, limited flexibility, and scalability challenges.
- By centralizing control and introducing programmability, SDN enables more efficient management and optimization of network resources.

iii. Advantages of SDN

Automation: SDN automates network provisioning and configuration tasks, reducing manual intervention and human errors.

Scalability: SDN scales more efficiently than traditional networking approaches, allowing networks to grow and adapt without significant architectural changes.

Centralized control: SDN provides a centralized view and control over the entire network, simplifying management and troubleshooting.

Flexibility: SDN enables dynamic and programmable network configuration, allowing organizations to adapt to changing requirements more easily.

iv. Limitations of SDN

Complexity: Implementing SDN requires expertise in software development and network architecture, which can be challenging for some organizations.

Security concerns: Centralizing control introduces new security risks that need to be carefully addressed.

Interoperability: Ensuring compatibility and interoperability between different SDN solutions and legacy systems can be complex.

c. Relationship between NFV and SDN

NFV and SDN work together beautifully to optimize network architecture:

- **SDN provides the brains**: The SDN controller orchestrates VNFs based on network traffic and service needs.
- **NFV offers the brawn**: VNFs running on standard servers perform the actual network functions.

NFV and SDN are complementary concepts that aim to modernize and optimize network architectures.

- NFV focuses on virtualizing network functions,
- while SDN centralizes control and programmability.

d. How NFV and SDN Work together and optimize network Architecture?

- NFV and SDN work together to create more flexible, scalable, and efficient network architectures.
- NFV virtualizes network functions, while SDN provides centralized control and programmability to dynamically orchestrate and manage these virtualized functions.

e. Real time example of NFV and SDN work together?

Assume an e-commerce website experiencing a traffic spike. Here's how NFV and SDN collaborate:

i. The SDN controller detects the rising traffic.
ii. It instructs the NFV orchestrator to spin up additional VNF firewalls on available servers.
iii. The network scales dynamically to handle the increased traffic without manual intervention.

f. Conclusion

- NFV and SDN represent significant advancements in network architecture, offering greater flexibility, scalability, and efficiency compared to traditional approaches.

- By virtualizing network functions and centralizing control, organizations can optimize their network infrastructures to meet the demands of modern applications and services.
- However, successful implementation of NFV and SDN requires careful planning, expertise, and consideration of potential challenges and limitations.

2. Highlight the differences between a virtual machine (VM) image and a VM instance.

a. Virtual Machine (VM)

A virtual machine (VM) is a software emulation of a physical computer that operates and executes programs like a physical machine.

b. Virtual Machine Image:

Definition: A VM image is a snapshot or template of a virtual machine's operating system, applications, and configurations at a specific point in time.

Usage: VM images are used as a basis for creating multiple VM instances. They serve as blueprints from which virtual machines are deployed and run.

Location: VM images are typically stored in a cloud environment or on-premises data centers. They are maintained and managed by system administrators or cloud service providers.

State: VM images are static. Once created, they remain unchanged unless explicitly updated or modified by the administrator.

i. Where is VM Image used?

- Creating new VM instances with the same configuration.
- Standardizing server environments.
- Backing up and restoring VM configurations.

c. Virtual Machine Instance:

Definition: A VM instance is a running instantiation of a virtual machine based on a VM image. It represents the active execution environment of the virtual machine.

Usage: VM instances are utilized for various computing tasks, such as running applications, hosting websites, or performing data analysis. They provide the computational resources needed to execute specific workloads.

Location: VM instances run on physical servers within a data center or on virtualized infrastructure provided by cloud service providers. They can be deployed, scaled, and managed dynamically according to demand.

State: VM instances are dynamic and ephemeral. They can be started, stopped, or terminated as needed, and their state may change based on user interactions or automated processes.

i. Where is it used?

- Running applications and workloads.
- Deploying servers and services.
- Testing and development environments.

d. Difference between Virtual Machine Image and VM Instance

Nature: A VM image is a static template used for creating multiple VM instances, while a VM instance is a dynamic entity representing the running state of a virtual machine.

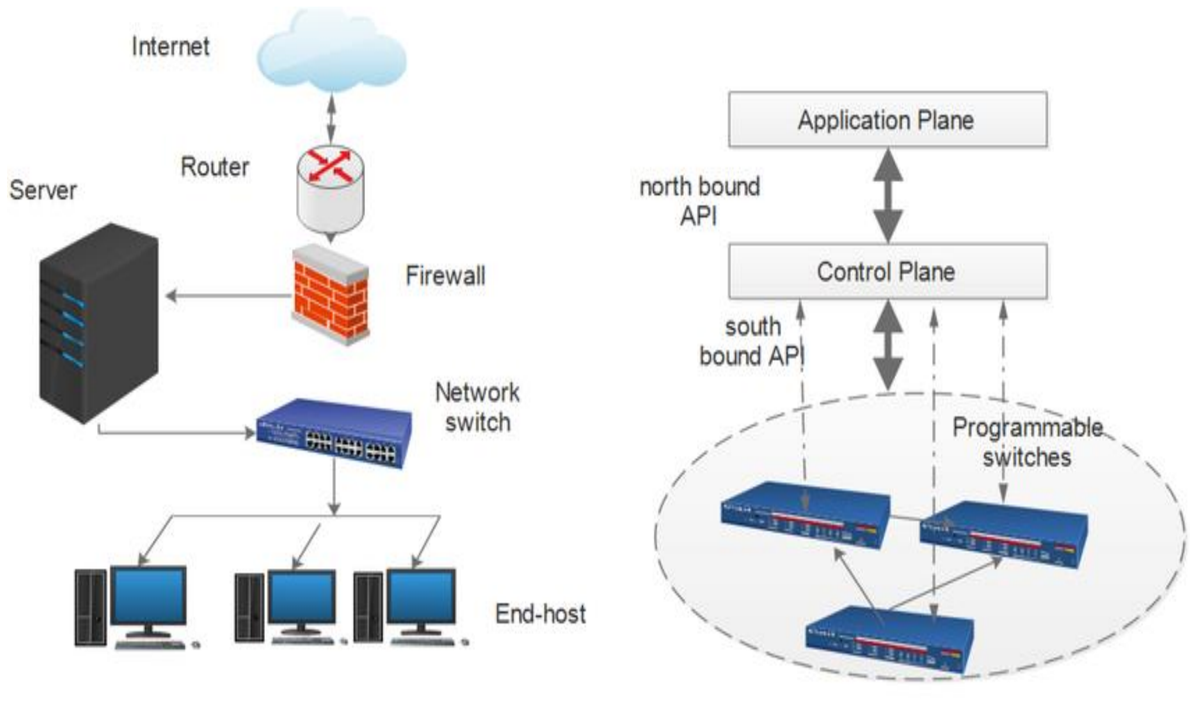
Usage: VM images are used as a blueprint for creating VM instances, while VM instances are utilized for executing specific tasks or workloads.

Persistence: VM images are typically persistent and remain unchanged unless updated, whereas VM instances are dynamic and can be started, stopped, or terminated based on demand.

In **summary**, while VM images serve as templates for creating VM instances, VM instances represent the actual execution and utilization of computing resources within a virtualized environment.

3. Compare and contrast the conventional network architecture with Software Defined Networking (SDN) architecture. Discuss the benefits of decoupling the control plane from the data plane and centralizing the network controller in SDN.

a. conventional network architecture VS SDN



a Traditional Network architecture

b SDN architecture

i. Components

- **Routers and Switches**: Devices responsible for forwarding data packets within the network.
- **Firewalls**: Security devices that control and monitor incoming and outgoing network traffic based on predetermined security rules.
- **Load Balancers**: Devices that distribute incoming network traffic across multiple servers to ensure efficient resource utilization and high availability.

- **Cabling and Physical Infrastructure**: Physical components such as cables, connectors, and networking equipment housed in data centers or office spaces.

ii. *Advantages*

Maturity: Conventional network architectures have been in use for decades, leading to well-established best practices and widespread industry expertise.

Performance: In many cases, traditional network architectures can offer high performance and low latency, especially in environments with optimized hardware and configurations.

Compatibility: Conventional network technologies are widely supported by various vendors and interoperable with different networking devices.

iii. *Disadvantages*

Scalability: Scaling traditional network architectures can be complex and expensive, requiring significant manual configuration and hardware upgrades.

Flexibility: Changes or updates to network configurations often require manual intervention and can be time-consuming.

Limited Automation: Conventional networks typically lack advanced automation capabilities, leading to slower deployment times and increased operational overhead.

b. Software Defined Networking Architecture

- It enables the control and management of network using software applications.
- Through Software Defined Network (SDN) networking behavior of entire network and its devices are programmed in centrally controlled manner through software applications using open APIs.
- Software Defined Network improves performance by network virtualization.
- In SDN software-controlled applications or APIs work as basis of complete network management that may be directing traffic on network or to communicate with underlying hardware infrastructure.
- So, in simple we can say SDN can create virtual networks, or it can control traditional networks with the help of software.

I. Components

SDN Controller: Centralized software responsible for managing the network and controlling the flow of traffic.

SDN Switches: Networking devices that separate the control plane from the data plane and can be programmatically controlled by the SDN controller.

Southbound APIs: Interfaces that connect the SDN controller to the underlying network infrastructure, allowing for communication and control.

Northbound APIs: Interfaces that enable communication between the SDN controller and higher-level network management applications or orchestration systems.

II. Advantages

- **Centralized control:** Easier to manage and configure the entire network from a single point.
- **Programmability:** Networks can be automated and dynamically reconfigured based on needs.
- **Increased flexibility:** Easier to adapt to changing traffic patterns and security requirements.
- **Improved scalability:** Simplifies adding new devices or functionalities.
- **Vendor independence:** OpenFlow allows interoperability with switches from different vendors.

III. Disadvantages

- **Complexity:** Requires additional software and expertise to manage the SDN controller.
- **Single point of failure:** If the SDN controller fails, the entire network can be disrupted.
- **Security concerns:** Security vulnerabilities in the SDN controller could be exploited.

c. What is control Plane

- The control plane in networking is responsible for managing the routing and forwarding of data packets within the network.
- It determines the optimal paths for data traffic based on network topology, routing protocols, and administrative policies.

d. What is Data Plane

- The data plane, also known as the forwarding plane, is responsible for forwarding data packets within the network according to the instructions provided by the control plane.
- It performs tasks such as packet forwarding, switching, and routing.

e. Control Plane vs. Data Plane:

a. **Control Plane:** The "brain" of the network, responsible for making decisions about how to route traffic. It determines the paths packets take and configures network devices accordingly. (Think of it as the GPS in your car, deciding which route to take.)
b. **Data Plane:** The "muscles" of the network, responsible for forwarding traffic according to the control plane instructions**.** Switches and routers belong to the data plane. (Think of it as the car itself, following the directions from the GPS.)

f. Benefits of decoupling the control plane from the data plane

- **Separation of concerns:** Simplifies network management by dividing control logic from data forwarding.
- **Increased agility:** Allows for faster network changes and adaptations.
- **Programmability:** Enables automation and dynamic network configurations.
- **Vendor independence:** OpenFlow allows using switches from different vendors with a common control plane.

g. Centralizing the network controller in SDN

Simplified Management: Administrators can centrally manage and orchestrate network resources, leading to simplified operations and reduced complexity.

Global Visibility: Centralized controllers have a holistic view of the network, allowing for better visibility and control over network traffic and performance.

Policy Enforcement: Centralized controllers can enforce consistent network policies across the entire infrastructure, ensuring compliance and security.

Dynamic Adaptation: Centralized controllers can dynamically adjust network configurations in response to changing traffic patterns or application requirements, improving network agility and responsiveness.

4. Explain the role of OpenFlow protocol in Software Defined Networking (SDN). Discuss how OpenFlow enables programmable network control and facilitates communication between the SDN controller and network devices.

a. OpenFlow Protocol:

i. How it works?

- OpenFlow is a communications protocol that enables the control of the forwarding plane of a network switch or router by a separate controller.
- It works by separating the control plane (decision-making) from the data plane (forwarding of traffic).
- In an OpenFlow-enabled network, the switch or router forwards packets based on instructions received from a centralized controller via the OpenFlow protocol.

Here's a simplified breakdown:

1. **Flow Matching:** The switch examines incoming packets based on header information (like source IP, destination IP, port numbers).
2. **Flow Table Lookup:** The switch consults its flow table, which stores pre-defined rules from the controller.
3. **Action Taken:** Based on matching rules, the switch takes actions like forwarding, dropping, or modifying the packet.
4. **Communication:** The switch can send messages back to the controller with statistics or report errors.

ii. Advantages

Centralized Control: OpenFlow enables centralized control and management of network devices, allowing administrators to define and enforce network-wide policies from a single point.

Programmability: OpenFlow facilitates network programmability, enabling administrators to dynamically adjust network configurations and policies through software-defined rules.

Flexibility: OpenFlow-based networks are highly flexible and adaptable, allowing for rapid deployment of new services and applications.

Traffic Optimization: OpenFlow enables intelligent traffic engineering and optimization by allowing the controller to dynamically adjust traffic paths based on real-time network conditions.

Interoperability: OpenFlow is an open standard supported by a wide range of networking vendors, ensuring interoperability and compatibility across different hardware and software platforms.

iii. Disadvantages

Increased complexity**:** Requires additional software (controller) and expertise to manage.
Security concerns: Vulnerabilities in the controller or OpenFlow implementation could be exploited.
Performance overhead: Processing flow rules on switches might add slight overhead compared to traditional configurations.
Limited functionality: OpenFlow primarily focuses on packet forwarding and might not handle all aspects of network management.

b. Role of OpenFlow in SDN

OpenFlow serves as the southbound API in SDN, providing a standardized way for the SDN controller to communicate with and program OpenFlow switches. It allows the controller to:

- Install flow rules on switches, specifying how to handle different traffic types.

- Modify existing flow rules to adapt to changing network conditions.
- Remove flow rules when they are no longer needed.
- Monitor switch performance and collect statistics.

c. How OpenFlow enables programmable network control

OpenFlow allows the controller to write flow rules that define how packets should be treated. These rules can be based on various packet header fields, enabling the network to be programmed for specific functionalities like:

- **Security policies:** Filtering malicious traffic or implementing access control lists.
- **Traffic engineering:** Optimizing traffic flow by prioritizing specific types of traffic.
- **Load balancing:** Distributing traffic across multiple paths or servers.
- **Quality of Service (QoS):** Guaranteeing bandwidth or low latency for critical applications.

d. How OpenFlow facilitates communication between the SDN controller and network devices.

OpenFlow uses messages to facilitate communication between the SDN controller and OpenFlow switches. These messages can be:

Controller to Switch: The controller sends flow rules, configuration updates, or requests for switch information.

Switch to Controller: The switch sends statistics about traffic flow, reports errors, or sends asynchronous messages about network events.

OpenFlow messages are typically sent over a secure channel using TLS (Transport Layer Security) for encryption and authentication.

5. Define containerization and Docker. Discuss the components of Docker and the deployment lifecycle of a container image.

a. What is Containerization?

- Containerization is a software development technique for packaging an application with all its dependencies (libraries, configuration files, runtime) into a standardized unit called a container.
- This container image ensures the application runs consistently across different environments, regardless of the underlying operating system.

i. How it works?

  - Containerization works by leveraging operating system-level virtualization to create isolated environments, known as containers, on a single host operating system.
  - Containers share the host OS kernel but are isolated from each other, providing lightweight, portable, and consistent runtime environments for applications.

ii. Advantages

Portability: Containers can run consistently across different environments, including development, testing, and production, without changes or modifications.

Efficiency: Containers are lightweight and share the host OS kernel, resulting in faster startup times, reduced resource consumption, and higher resource utilization.

Isolation: Containers provide process-level isolation, ensuring that applications and their dependencies are isolated from each other and do not interfere with each other's runtime environment.

Consistency: Containers encapsulate everything needed to run an application, ensuring consistency in development, testing, and deployment workflows.

Scalability: Containers can be easily scaled up or down based on demand, enabling efficient resource allocation and utilization.

iii. Disadvantages

Security Concerns: Containers share the host OS kernel, which can pose security risks if not properly configured or managed. Vulnerabilities in the host kernel can potentially affect all containers running on the same host.

Complexity: Implementing containerization and managing containerized environments can be complex, requiring specialized skills and expertise in container orchestration and management tools.

Persistence: Containers are typically ephemeral, meaning that any data or changes made within a container may be lost when the container is stopped or removed, requiring additional measures for data persistence and storage.

b. What is Docker?

- Docker is a popular open-source platform that provides tools and technologies for building, deploying, and managing containerized applications.
- Docker offers a user-friendly interface and a vast ecosystem of tools and libraries for containerization.

i. How it works?

- Docker works by using a client-server architecture, where the Docker client interacts with the Docker daemon, which is responsible for building, running, and managing containers.
- Docker utilizes containerization technology, such as Linux namespaces and control groups (cgroups), to create isolated environments for running applications.

ii. Advantages

Easy to Use: Docker provides a simple and intuitive CLI for building and managing containers.

Large Community: Docker has a vast and active community that provides support and a wealth of resources.

Portability: Docker containers run consistently across different platforms.

Security Features: Docker offers features like user namespaces and security profiles to enhance container security.

iii. Disadvantages

Learning Curve: While Docker is user-friendly, there's still a learning curve for beginners.

Security Concerns: Security vulnerabilities in Docker or container images can pose risks.

Resource Management: Managing resource allocation for multiple containers can be complex.

Vendor Lock-in: While Docker images can run on other container runtimes, there might be some vendor lock-in with Docker-specific tools.

c. Components of Docker

Docker Engine: The core component of Docker that runs and manages containers on a host system.

Docker Client: The command-line interface (CLI) tool used to interact with the Docker Engine and manage containers and images.

Docker Images: Immutable templates used to create containers. Images contain application code, runtime, system libraries, and dependencies.

Docker Containers: Runnable instances of Docker images. Containers are isolated environments that encapsulate everything needed to run an application.

Docker Registry: A centralized repository for storing and distributing Docker images. Docker Hub is the official public registry for Docker images, but private registries can also be used for storing proprietary or sensitive images.

d. Deployment life cycle of a container Image

The deployment lifecycle of a container image typically involves the following stages:

Image Creation: The container image is created using a Dockerfile, which defines the application's dependencies, configuration, and runtime environment.

Image Build: The Docker image is built using the docker build command, which executes the instructions in the Dockerfile to create a layered image.

Image Registry: The Docker image is pushed to a Docker registry, such as Docker Hub or a private registry, where it is stored and can be accessed by other users or systems.

Image Pull: The Docker image is pulled from the registry to the target host or environment using the docker pull command.

Container Creation: The Docker image is used to create a container instance using the docker run command. The container runs in isolation with its own filesystem, network, and process space.

Container Execution: The containerized application is executed within the container environment, running as a separate process on the host system.

Container Management: The Docker container can be managed and monitored using various Docker commands, such as docker start, docker stop, docker restart, docker logs, etc.

Scaling and Orchestration: In a production environment, containerized applications may be scaled up or down dynamically based on demand, and orchestrated using container orchestration platforms like Kubernetes, Docker Swarm, or Amazon ECS.

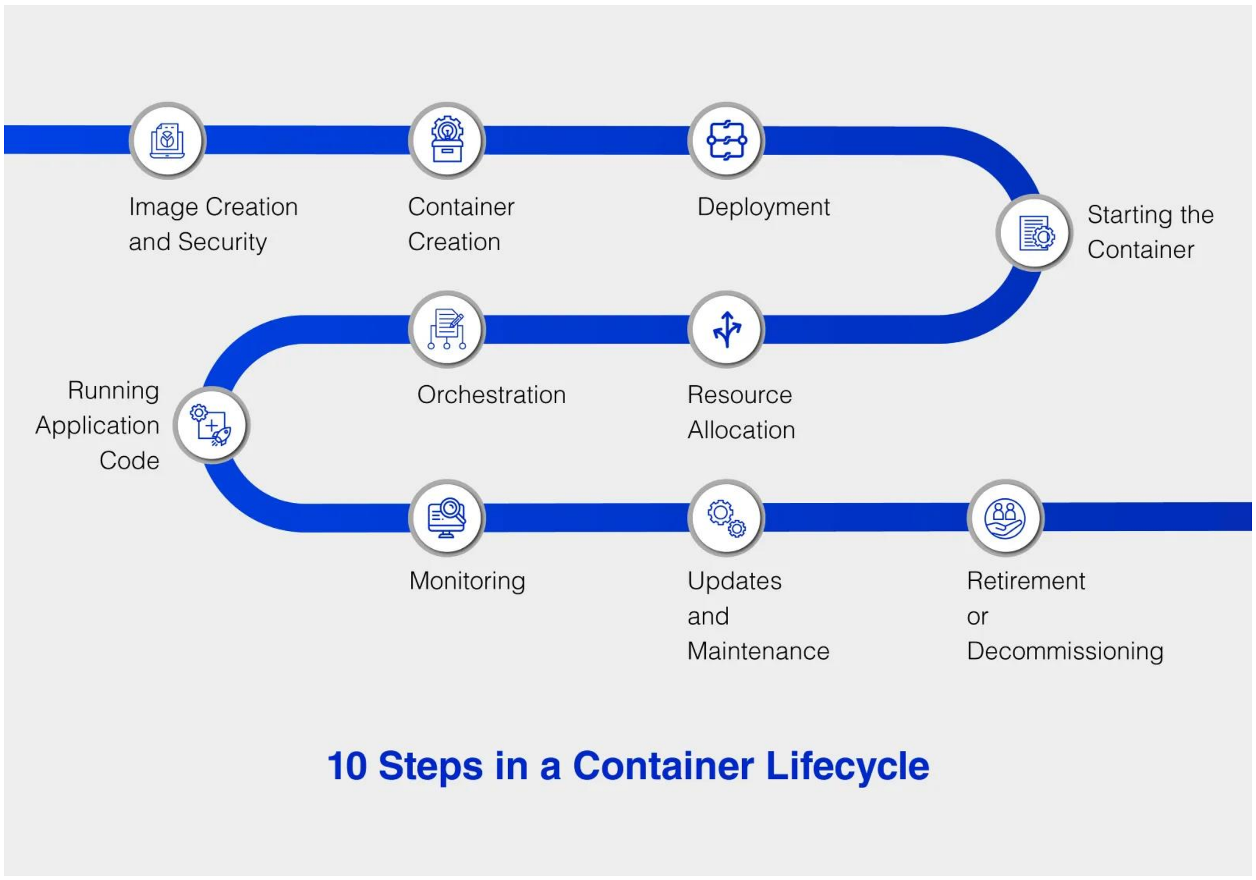


Image Creation and Security
Container Creation
Deployment
Starting the Container
Orchestration
Resource Allocation
Running Application Code
Monitoring
Updates and Maintenance
Retirement or Decommissioning
10 Steps in a Container Lifecycle