CONTAINER NETWORKING AND STORAGE**Introduction & Learning Objectives**

**Chapter Overview**

CONTAINER NETWORKING AND STORAGE Introduction & Learning Objectives Chapter Overview CONTAINER NETWORKING AND STORAGE Knowledge Check Congratulations on completing Chapter 7 - Container Networking and Storage. Take this quiz to check your understanding of the concepts you've learned about so far. Question 7.1 Which of the following is not the default network setup for Docker? A. Host B. None C. Overlay D. Bridge

ChatGPT

C. Overlay

The statement is correct.

Among the options listed, "Overlay" is not the default network setup for Docker. The default network setup for Docker is "Bridge."

When you install Docker, it automatically creates a default bridge network named "bridge." Containers launched without specifying a network explicitly will be attached to this default bridge network. This default bridge allows containers to communicate with each other on the same host using internal IP addresses.

The other network options mentioned are:

A. Host: When you run a container with the **--network=host** option, it shares the host's network namespace, and the container's network stack is directly connected to the host's network. This allows the container to access services on the host network without any network isolation.

B. None: When you run a container with the **--network=none** option, the container is not attached to any network. This means the container cannot communicate with other containers or the outside world unless you explicitly connect it to other networks or use other network settings.

D. Overlay: Overlay networks are used in cluster environments or when deploying applications across multiple hosts. They provide network communication between containers running on different Docker hosts. Overlay networks are typically used with container orchestration systems like Docker Swarm or Kubernetes to enable communication between containers across multiple nodes in the cluster.

In summary, the default network setup for Docker is "Bridge," and you can use other network options like "Host," "None," or "Overlay" depending on your specific networking requirements.



Question 7.2 You can segment networks to isolate containers running on the same host. True or False? A. True B. False

ChatGPT

A. True

The statement is True.

You can segment networks to isolate containers running on the same host. Docker provides various network drivers and configurations that allow you to create separate network segments to isolate containers from each other. This isolation provides security and prevents unintended communication between containers.

Some of the network drivers available in Docker that support network segmentation and isolation are:

1. **Bridge Network Driver:** The default network driver in Docker is "bridge," which creates a virtual network bridge on the host. Each container connected to the bridge gets its IP address and is isolated from other containers connected to the same bridge. This provides network segmentation and prevents direct communication between containers on the same bridge.
2. **Overlay Network Driver:** The "overlay" network driver is used in Docker Swarm or Kubernetes clusters to provide network communication between containers running on different Docker hosts. Overlay networks segment containers across multiple hosts and allow them to communicate securely through encrypted connections.
3. **Macvlan Network Driver:** The "macvlan" network driver allows containers to have their own MAC addresses, making them appear as separate physical devices on the network. This enables containers to be on the same network segment as other physical devices, providing complete isolation.
4. **IPvlan Network Driver:** The "ipvlan" network driver is similar to macvlan but operates at the IP layer, allowing each container to have its own IP address, making them appear as separate IP devices on the network.
5. **Custom Bridge Networks:** You can create custom bridge networks with the **docker network create** command and configure them with specific IP ranges and other network settings. Containers connected to different custom bridge networks are isolated from each other.

By using these network drivers and configuring them appropriately, you can achieve network segmentation and isolation for containers running on the same host, enhancing security and providing distinct network environments for different applications or services.



Question 7.3 Which of the following is a storage managed by Docker? A. Volume B. Bind mount C. tmpfs D. All of the above

ChatGPT

D. All of the above

All the options mentioned (Volume, Bind mount, and tmpfs) are storage options managed by Docker.

1. **Volume:** A Docker volume is a directory that is managed by Docker and is mounted into the container. Volumes are designed to persist data even after a container is stopped or removed. They provide a way to store data separately from the container, making it easy to share data between containers or with the host system. Docker volumes are the recommended way to manage persistent data in containers.
2. **Bind mount:** A bind mount is a file or directory that is mounted into the container from the host system. With bind mounts, you can directly access files and directories from the host's file system inside the container. Unlike volumes, bind mounts do not create a separate storage area managed by Docker; they directly reference files and directories from the host.
3. **tmpfs:** Tmpfs is a temporary file system stored in the host's memory. When you mount a tmpfs volume into a container, the data is stored in memory, and it is not persisted after the container stops or is removed. Tmpfs is useful when you need to store temporary data that does not need to be persisted.

Using these storage options, you have flexibility in managing data in Docker containers. Volumes are the preferred choice for managing persistent data, bind mounts provide direct access to the host's file system, and tmpfs is useful for temporary in-memory storage.

In summary, Docker provides multiple storage options to manage data in containers, and each option serves different use cases depending on the need for persistence and accessibility.



Question 7.4 Which of the following storages is most suitable for fast, in-memory applications that require no persistence? A. fastfs B. tmpfs C. btrfs D. overlayfs

ChatGPT

B. tmpfs

The most suitable storage option for fast, in-memory applications that require no persistence is **tmpfs**.

Tmpfs is a temporary file system that stores data in the host's memory (RAM). When you mount a tmpfs volume into a container, the data is stored in memory, and it does not persist after the container is stopped or removed. Since the data resides in RAM, it offers very fast access times and is ideal for applications that need quick read and write operations without the need for data persistence.

Tmpfs is particularly useful for temporary data that is generated and used during the lifetime of a container but doesn't need to be saved for future use. Examples of such applications are caching systems, temporary file storage, or any application that requires a fast, in-memory storage for temporary data.

On the other hand:

* **fastfs** is not a standard storage option in Docker, and it is not a recognized storage driver.
* **btrfs** and **overlayfs** are file systems and storage drivers in Linux, but they are not specifically designed for fast, in-memory applications without persistence. Both **btrfs** and **overlayfs** provide persistent storage features and are not as suitable as **tmpfs** for fast in-memory data access.

In summary, if you have an application that requires fast access to data in memory and doesn't need persistence, **tmpfs** is the most appropriate choice among the options provided.



Question 7.5 It is possible to launch two containers with isolated processes that share network interface and IP address. True or False? A. True B. False

ChatGPT

B. False

The statement is False.

It is not possible to launch two containers with isolated processes that share the same network interface and IP address. Containers are designed to provide process-level isolation, which means each container runs its own isolated processes with its own network namespace.

Each container has its own unique network stack, including its own network interface, IP address, routing tables, and firewall rules. This isolation ensures that containers do not interfere with each other's network connectivity and that they cannot directly communicate using the same IP address or network interface.

If you need to allow communication between containers, you can use Docker networking features to create custom networks, allowing containers to communicate with each other using their IP addresses, DNS names, or container names. Docker provides different network drivers, such as the default bridge network, overlay network, and others, that facilitate communication between containers and with the host system.

However, even when containers are connected to the same network, they still have their individual network namespaces, and each container's processes operate in isolation from one another. This prevents one container's processes from accessing or interfering with another container's network configuration or traffic.

In summary, containers are isolated from each other at the process level and cannot share the same network interface and IP address directly. Docker's networking features are used to facilitate communication between containers while maintaining their individual network namespaces and isolation.



Question 7.6 How do you create a network in Docker? A. docker network create <net\_name> -d <driver\_choice> B. docker -d <driver\_choice> network create <net\_name> C. docker <net\_name> create network D. docker create network <net\_name> -d <driver\_choice>

ChatGPT

A. docker network create <net\_name> -d <driver\_choice>

Option A is the correct way to create a network in Docker.

To create a custom network in Docker, you use the **docker network create** command followed by the network name and the **-d** flag to specify the driver choice. The correct syntax is:

phpCopy code

docker network create <net\_name> -d <driver\_choice>

* **<net\_name>**: The name you want to give to the custom network you are creating.
* **-d <driver\_choice>**: The driver choice specifies the network driver to be used for the custom network. The choice of driver determines the behavior and capabilities of the network. For example, you can use the default bridge driver, overlay driver, macvlan driver, or others depending on your requirements.

For example, to create a custom bridge network named "my\_network," you would use the following command:

luaCopy code

docker network create my\_network -d bridge

This will create a new custom bridge network named "my\_network" that containers can be connected to.

Option B, C, and D have incorrect syntax or wrong placement of options and are not valid commands for creating a Docker network. The correct command for creating a network in Docker always starts with **docker network create**, followed by the network name and any additional options such as the network driver choice using the **-d** flag.



Question 7.7 In bridge networking, the container receives an IP from the host's subset. True or False? A. True B. False

ChatGPT

A. True

The statement is True.

In bridge networking, when a container is connected to a bridge network (default bridge or custom bridge), it receives an IP address from the host's subset of the network. The bridge acts as a virtual network switch that allows containers to communicate with each other and with the host system.

When you create a bridge network, Docker automatically assigns an IP address to the bridge interface on the host. By default, the bridge interface is configured with an IP address range of **172.17.0.0/16**. When a container is connected to this bridge network, it is assigned an IP address from this subnet, allowing it to communicate with other containers on the same bridge and with the host.

For example, if a container is connected to the default bridge network, it might be assigned an IP address like **172.17.0.2**. Containers connected to the same bridge network can communicate with each other using these IP addresses.

However, it's important to note that this behavior is specific to bridge networking. When using other networking options like host network mode or overlay networks, the IP assignment and networking behavior may differ. But in the case of bridge networking, containers do receive IP addresses from the host's subset, and this enables them to communicate with each other and the host.

CONTAINER NETWORKING AND STORAGE

**Introduction & Learning Objectives**

**Chapter Overview**

### CONTAINER NETWORKING AND STORAGE

### Knowledge Check

### Congratulations on completing Chapter 7 - Container Networking and Storage. Take this quiz to check your understanding of the concepts you've learned about so far.

CONTAINER NETWORKING AND STORAGE Introduction & Learning Objectives Chapter Overview CONTAINER NETWORKING AND STORAGE Knowledge Check Congratulations on completing Chapter 7 - Container Networking and Storage. Take this quiz to check your understanding of the concepts you've learned about so far. Question 7.1 Which of the following is not the default network setup for Docker? A. Host B. None C. Overlay D. Bridge

ChatGPT

C. Overlay

The statement is correct.

Among the options listed, "Overlay" is not the default network setup for Docker. The default network setup for Docker is "Bridge."

When you install Docker, it automatically creates a default bridge network named "bridge." Containers launched without specifying a network explicitly will be attached to this default bridge network. This default bridge allows containers to communicate with each other on the same host using internal IP addresses.

The other network options mentioned are:

A. Host: When you run a container with the **--network=host** option, it shares the host's network namespace, and the container's network stack is directly connected to the host's network. This allows the container to access services on the host network without any network isolation.

B. None: When you run a container with the **--network=none** option, the container is not attached to any network. This means the container cannot communicate with other containers or the outside world unless you explicitly connect it to other networks or use other network settings.

D. Overlay: Overlay networks are used in cluster environments or when deploying applications across multiple hosts. They provide network communication between containers running on different Docker hosts. Overlay networks are typically used with container orchestration systems like Docker Swarm or Kubernetes to enable communication between containers across multiple nodes in the cluster.

In summary, the default network setup for Docker is "Bridge," and you can use other network options like "Host," "None," or "Overlay" depending on your specific networking requirements.



Question 7.2 You can segment networks to isolate containers running on the same host. True or False? A. True B. False

ChatGPT

A. True

The statement is True.

You can segment networks to isolate containers running on the same host. Docker provides various network drivers and configurations that allow you to create separate network segments to isolate containers from each other. This isolation provides security and prevents unintended communication between containers.

Some of the network drivers available in Docker that support network segmentation and isolation are:

1. **Bridge Network Driver:** The default network driver in Docker is "bridge," which creates a virtual network bridge on the host. Each container connected to the bridge gets its IP address and is isolated from other containers connected to the same bridge. This provides network segmentation and prevents direct communication between containers on the same bridge.
2. **Overlay Network Driver:** The "overlay" network driver is used in Docker Swarm or Kubernetes clusters to provide network communication between containers running on different Docker hosts. Overlay networks segment containers across multiple hosts and allow them to communicate securely through encrypted connections.
3. **Macvlan Network Driver:** The "macvlan" network driver allows containers to have their own MAC addresses, making them appear as separate physical devices on the network. This enables containers to be on the same network segment as other physical devices, providing complete isolation.
4. **IPvlan Network Driver:** The "ipvlan" network driver is similar to macvlan but operates at the IP layer, allowing each container to have its own IP address, making them appear as separate IP devices on the network.
5. **Custom Bridge Networks:** You can create custom bridge networks with the **docker network create** command and configure them with specific IP ranges and other network settings. Containers connected to different custom bridge networks are isolated from each other.

By using these network drivers and configuring them appropriately, you can achieve network segmentation and isolation for containers running on the same host, enhancing security and providing distinct network environments for different applications or services.



Question 7.3 Which of the following is a storage managed by Docker? A. Volume B. Bind mount C. tmpfs D. All of the above

ChatGPT

D. All of the above

All the options mentioned (Volume, Bind mount, and tmpfs) are storage options managed by Docker.

1. **Volume:** A Docker volume is a directory that is managed by Docker and is mounted into the container. Volumes are designed to persist data even after a container is stopped or removed. They provide a way to store data separately from the container, making it easy to share data between containers or with the host system. Docker volumes are the recommended way to manage persistent data in containers.
2. **Bind mount:** A bind mount is a file or directory that is mounted into the container from the host system. With bind mounts, you can directly access files and directories from the host's file system inside the container. Unlike volumes, bind mounts do not create a separate storage area managed by Docker; they directly reference files and directories from the host.
3. **tmpfs:** Tmpfs is a temporary file system stored in the host's memory. When you mount a tmpfs volume into a container, the data is stored in memory, and it is not persisted after the container stops or is removed. Tmpfs is useful when you need to store temporary data that does not need to be persisted.

Using these storage options, you have flexibility in managing data in Docker containers. Volumes are the preferred choice for managing persistent data, bind mounts provide direct access to the host's file system, and tmpfs is useful for temporary in-memory storage.

In summary, Docker provides multiple storage options to manage data in containers, and each option serves different use cases depending on the need for persistence and accessibility.



Question 7.4 Which of the following storages is most suitable for fast, in-memory applications that require no persistence? A. fastfs B. tmpfs C. btrfs D. overlayfs

ChatGPT

B. tmpfs

The most suitable storage option for fast, in-memory applications that require no persistence is **tmpfs**.

Tmpfs is a temporary file system that stores data in the host's memory (RAM). When you mount a tmpfs volume into a container, the data is stored in memory, and it does not persist after the container is stopped or removed. Since the data resides in RAM, it offers very fast access times and is ideal for applications that need quick read and write operations without the need for data persistence.

Tmpfs is particularly useful for temporary data that is generated and used during the lifetime of a container but doesn't need to be saved for future use. Examples of such applications are caching systems, temporary file storage, or any application that requires a fast, in-memory storage for temporary data.

On the other hand:

* **fastfs** is not a standard storage option in Docker, and it is not a recognized storage driver.
* **btrfs** and **overlayfs** are file systems and storage drivers in Linux, but they are not specifically designed for fast, in-memory applications without persistence. Both **btrfs** and **overlayfs** provide persistent storage features and are not as suitable as **tmpfs** for fast in-memory data access.

In summary, if you have an application that requires fast access to data in memory and doesn't need persistence, **tmpfs** is the most appropriate choice among the options provided.



Question 7.5 It is possible to launch two containers with isolated processes that share network interface and IP address. True or False? A. True B. False

ChatGPT

B. False

The statement is False.

It is not possible to launch two containers with isolated processes that share the same network interface and IP address. Containers are designed to provide process-level isolation, which means each container runs its own isolated processes with its own network namespace.

Each container has its own unique network stack, including its own network interface, IP address, routing tables, and firewall rules. This isolation ensures that containers do not interfere with each other's network connectivity and that they cannot directly communicate using the same IP address or network interface.

If you need to allow communication between containers, you can use Docker networking features to create custom networks, allowing containers to communicate with each other using their IP addresses, DNS names, or container names. Docker provides different network drivers, such as the default bridge network, overlay network, and others, that facilitate communication between containers and with the host system.

However, even when containers are connected to the same network, they still have their individual network namespaces, and each container's processes operate in isolation from one another. This prevents one container's processes from accessing or interfering with another container's network configuration or traffic.

In summary, containers are isolated from each other at the process level and cannot share the same network interface and IP address directly. Docker's networking features are used to facilitate communication between containers while maintaining their individual network namespaces and isolation.



Question 7.6 How do you create a network in Docker? A. docker network create <net\_name> -d <driver\_choice> B. docker -d <driver\_choice> network create <net\_name> C. docker <net\_name> create network D. docker create network <net\_name> -d <driver\_choice>

ChatGPT

A. docker network create <net\_name> -d <driver\_choice>

Option A is the correct way to create a network in Docker.

To create a custom network in Docker, you use the **docker network create** command followed by the network name and the **-d** flag to specify the driver choice. The correct syntax is:

phpCopy code

docker network create <net\_name> -d <driver\_choice>

* **<net\_name>**: The name you want to give to the custom network you are creating.
* **-d <driver\_choice>**: The driver choice specifies the network driver to be used for the custom network. The choice of driver determines the behavior and capabilities of the network. For example, you can use the default bridge driver, overlay driver, macvlan driver, or others depending on your requirements.

For example, to create a custom bridge network named "my\_network," you would use the following command:

luaCopy code

docker network create my\_network -d bridge

This will create a new custom bridge network named "my\_network" that containers can be connected to.

Option B, C, and D have incorrect syntax or wrong placement of options and are not valid commands for creating a Docker network. The correct command for creating a network in Docker always starts with **docker network create**, followed by the network name and any additional options such as the network driver choice using the **-d** flag.



Question 7.7 In bridge networking, the container receives an IP from the host's subset. True or False? A. True B. False

ChatGPT

A. True

The statement is True.

In bridge networking, when a container is connected to a bridge network (default bridge or custom bridge), it receives an IP address from the host's subset of the network. The bridge acts as a virtual network switch that allows containers to communicate with each other and with the host system.

When you create a bridge network, Docker automatically assigns an IP address to the bridge interface on the host. By default, the bridge interface is configured with an IP address range of **172.17.0.0/16**. When a container is connected to this bridge network, it is assigned an IP address from this subnet, allowing it to communicate with other containers on the same bridge and with the host.

For example, if a container is connected to the default bridge network, it might be assigned an IP address like **172.17.0.2**. Containers connected to the same bridge network can communicate with each other using these IP addresses.

However, it's important to note that this behavior is specific to bridge networking. When using other networking options like host network mode or overlay networks, the IP assignment and networking behavior may differ. But in the case of bridge networking, containers do receive IP addresses from the host's subset, and this enables them to communicate with each other and the host.

### QUIZ RESULTS

YOU GOT 4 OUT OF 7 CORRECT: 57%

7 7 4 3

* TOTAL (7)
* ANSWERED
* CORRECT
* INCORRECT

Retake Quiz Review All Question Results

[Continue to Next Section](https://trainingportal.linuxfoundation.org/learn/course/containers-for-developers-and-quality-assurance/container-networking-and-storage/knowledge-check/completed)

# Question 7.1

Which of the following is **not** the default network setup for Docker?

Correct Answer

* **A.** Host
* **B.** None
* **C.** Overlay

Your Answer:

Correct

* **D.** Bridge

# Question 7.2

You can segment networks to isolate containers running on the same host. True or False?

Correct Answer

* **A.** True

Your Answer:

Correct

* **B.** False

# Question 7.3

Which of the following is a storage managed by Docker?

Incorrect Answer

* **A.** Volume

Correct Answer

* **B.** Bind mount
* **C.** tmpfs
* **D.** All of the above

# Question 7.4

Which of the following storages is most suitable for fast, in-memory applications that require no persistence?

Correct Answer

* **A.** fastfs
* **B.** tmpfs

Your Answer:

Correct

* **C.** btrfs
* **D.** overlayfs

# Question 7.5

It is possible to launch two containers with isolated processes that share network interface and IP address. True or False?

Incorrect Answer

* **A.** True

Correct Answer

* **B.** False

# Question 7.6

How do you create a network in Docker?

Correct Answer

* **A.** docker network create <net\_name> -d <driver\_choice>

Your Answer:

Correct

* **B.** docker -d <driver\_choice> network create <net\_name>
* **C.** docker <net\_name> create network
* **D.** docker create network <net\_name> -d <driver\_choice>

# Question 7.7

In bridge networking, the container receives an IP from the host's subset. True or False?

Incorrect Answer

* **A.** True

Your Answer:

Incorrect

* **B.** False

Correct Answer