How do containers in a Pod communicate with each other?

Containers in a **Kubernetes Pod** communicate with each other in the following ways:

1. Shared Network Namespace (localhost communication)

- All containers in a Pod share the **same network namespace**.
- This means they can communicate **via** `localhost` **(127.0.0.1)`** and ports.
- Example:
  If one container runs a web server on port **5000**, another container in the same Pod can access it using:

```
sh
------
curl http://localhost:5000
```

2. Shared Storage (Volumes)

- Containers can communicate indirectly by writing and reading from a **shared volume**.
- Example:
  - A logging container writes logs to a shared volume.
  - Another container reads and processes these logs.

3. Inter-Process Communication (IPC)

- Containers within the same Pod can use **inter-process communication (IPC)** like **Unix domain sockets**.

4. Environment Variables and Files

- One container can expose configuration details as **environment variables** or files (e.g., using a ConfigMap), and another container can read them.

To see all containers

kubectl get pod multi-container-pod -o jsonpath="{.spec.containers[*].name}"

- **View logs of a specific container in a pod:**

```
sh

kubectl logs <pod-name> -c <container-name>
```

Example

Container 1 (`web-server`)

- Runs an **NGINX web server** on **port 80**.
- Writes a **message to a shared volume** (`/usr/share/nginx/html/index.html`).
- **Container 2 (`env-reader`)**
- Reads the **environment variable** from `web-server`.
- Prints it to logs.
- **Container 3 (`volume-reader`)**
- Reads the **shared file** from the **volume**.
- Prints the content to logs.
- **Container 4 (`http-client`) (Newly added)**
- Uses `curl` to **make an HTTP request** to `web-server`.
- Fetches `index.html` and prints the response.

🎯 Summary

✅ `web-server` exposes an HTTP web server on **localhost**.
✅ `env-reader` reads **environment variables** and logs them.

✅ `volume-reader` reads **shared volume files** and prints them.
✅ `http-client` **makes an HTTP request** to `web-server` and retrieves the message.

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-container-pod
spec:
  volumes:
    - name: shared-storage
      emptyDir: {}

  containers:
    # Container 1: Web Server (Exposes HTTP, Writes to Shared Volume)
    - name: web-server
      image: nginx
      env:
        - name: MESSAGE
          value: "Hello from Web Server!"
      volumeMounts:
        - name: shared-storage
          mountPath: /usr/share/nginx/html
      command: ["/bin/sh", "-c"]
      args:
        - echo "Shared Data: Hello from Web Server!" > /usr/share/nginx/html/index.html;
          nginx -g "daemon off;"

    # Container 2: Reads from Environment Variable
    - name: env-reader
      image: busybox
      env:
        - name: MESSAGE
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
      command: ["/bin/sh", "-c"]
      args:
        - echo "Pod Name: $MESSAGE";
          echo "Env Var from Web Server: $MESSAGE";
          sleep 3600;

    # Container 3: Reads from Shared Volume
    - name: volume-reader
      image: busybox
      volumeMounts:
        - name: shared-storage
          mountPath: /data
      command: ["/bin/sh", "-c"]
      args:
```

```
      - echo "Reading from Shared Volume:";
        cat /data/index.html || echo "File not found!";
        sleep 3600;

  # Container 4: Makes HTTP Request to web-server
  - name: http-client
    image: curlimages/curl
    command: ["/bin/sh", "-c"]
    args:
      - echo "Making HTTP Request to web-server...";
        curl -s http://localhost/index.html;
        sleep 3600;
```

How do containers within the same Kubernetes Pod communicate with each other?

A) By using separate network namespaces for each container
B) By using the loopback interface (localhost) and shared network namespace
C) By assigning a unique IP address to each container within a Pod
D) By using a Kubernetes Service to expose internal communication

Answer: B) By using the loopback interface (localhost) and shared network namespace