**Troubleshooting Docker Image Failures 🚀**

If your Docker image fails to build or run, follow these **systematic troubleshooting steps** to identify and fix the issue.

# 1. Check Build Logs (If Build Fails)

docker build -t myapp .

docker build -t myapp . | tee build.log

tail -n 20 build.log

docker build --no-cache -t myapp .

# 2. Check Running Container Logs

docker logs <container\_id>

docker-compose logs

docker run -it myapp /bin/sh

# 3. Run Container in Debug Mode

docker run -it --entrypoint /bin/sh myapp

node server.js

# 4. Check if Ports Are Already in Use

sudo netstat -tulnp | grep 3000

sudo systemctl stop apache2

# 5. Check Container Exit Code

docker ps -a

docker inspect <container\_id>

docker logs <container\_id>

# 6. Check File Paths & Permissions

docker run -it myapp ls -lah /app

chmod +x entrypoint.sh

# 7. Check Environment Variables

docker inspect <container\_id> | grep -i env

docker run -e ENV\_VAR=value myapp

# 8. Clean Up Docker System

docker system prune -a

docker builder prune

# 9. Increase Build Timeout (If Build Hangs)

docker build --network=host -t myapp .

docker build --progress=plain -t myapp .

# 10. Verify Docker Daemon & Logs

sudo systemctl restart docker

journalctl -u docker --no-pager | tail -n 50

**How Does SSH Work? (Secure Shell)**

SSH (Secure Shell) is a cryptographic network protocol used to securely connect to remote machines over an unsecured network. It provides **authentication, encryption, and integrity** to ensure secure communication.

**🚀 How SSH Works Step by Step**

**1️⃣ SSH Client and Server Communication**

* **SSH Client**: The machine from which you are initiating the SSH connection.
* **SSH Server**: The machine you are connecting to remotely.
* **Port 22**: SSH operates on **port 22** by default.

🔹 When you run:

ssh username@remote\_host

* The SSH **client** contacts the SSH **server**.
* The server sends its **public key** for authentication.
* If the key is trusted, the client encrypts a session key using this public key.
* The server decrypts the session key using its **private key**.
* Both sides establish an **encrypted** connection.

**2️⃣ Authentication Methods**

SSH uses different authentication methods:

✅ **1. Password-based authentication** (less secure)

ssh user@server

🔹 The server prompts for a password.

✅ **2. Key-based authentication** (more secure)

ssh -i ~/.ssh/id\_rsa user@server

🔹 Uses **public-key authentication**:

* The client has a **private key** (~/.ssh/id\_rsa).
* The server has the **public key** (~/.ssh/authorized\_keys).
* If they match, access is granted **without a password**.

✅ **3. Multi-factor authentication (MFA)**

* Uses both a password and an SSH key for extra security.

**3️⃣ Establishing an Encrypted Tunnel**

Once authenticated, SSH:  
✅ **Encrypts** the session using symmetric encryption (AES, ChaCha20, etc.).  
✅ **Verifies data integrity** using hashing algorithms (SHA-2, MD5).  
✅ **Prevents MITM (Man-in-the-Middle) attacks** using key verification.

To see the fingerprint of a remote server:

ssh-keygen -lf /etc/ssh/ssh\_host\_rsa\_key.pub

**4️⃣ SSH Features**

SSH is not just for logging into servers! It can also:

🔹 **Copy files securely**

scp file.txt user@server:/path/to/destination

🔹 **Mount remote directories**

sshfs user@server:/remote/path /local/mount

🔹 **Forward ports securely**

ssh -L 8080:localhost:80 user@server

* **Local Port Forwarding**: Access remote services as if they were local.

**✅ Summary**

| **Step** | **Description** |
| --- | --- |
| **1. Connection** | Client connects to SSH server on port 22 |
| **2. Authentication** | Password, key-based, or MFA |
| **3. Encryption** | Secure tunnel established |
| **4. Secure Access** | Remote shell, file transfer, port forwarding |

**SSH ensures secure communication between remote systems, protecting data from interception and attacks.**