Deploy Docker Image to Ubuntu Server - Step by Step

Your Docker Image: `socket-io-app:1.0` (1.48GB)
Server OS: Ubuntu (fresh install)
Goal: Run your app on the server

📋 Prerequisites

Before starting, you need:

1. **Server IP Address** - Example: `192.168.1.100` or `12.34.56.78`
2. **SSH Access** - Username and password (or SSH key)
3. **Domain Name** (optional) - Example: `apg-socket.com`
4. **Your laptop/computer** with your Docker image already built

---

🚀 Step 1: Install Docker on Ubuntu Server

1.1 SSH Into Your Server

Open Terminal/Command Prompt and connect:

```
ssh root@your-server-ip
# Example:
ssh root@192.168.1.100
```

Type: `yes` when asked to accept fingerprint

Enter your password

1.2 Update System

```
apt update
apt upgrade -y
```

Wait for it to finish (may take 2-3 minutes)

1.3 Install Docker

```
# Install Docker
apt install docker.io -y

# Install Docker Compose
apt install docker-compose -y

# Verify installations
docker --version
docker-compose --version
```

You should see:

```
Docker version 24.0.0
Docker Compose version 2.x.x
```

1.4 Verify Docker is Running

```
docker ps
```

If it shows `Cannot connect to Docker daemon`, run:

```
systemctl start docker
```

📤 Step 2: Upload Docker Image to Server

You have **3 options**. Pick ONE:

Option A: Using Docker Hub (Easiest - Recommended)

A1: Create Docker Hub Account

1. Go to https://hub.docker.com
2. Click "Sign Up"
3. Create account (example: `yourusername`)
4. Verify email

A2: Tag Your Local Image

On your computer, open Terminal:

```
# Login to Docker Hub
docker login

# Enter your Docker Hub credentials
# Username: yourusername
# Password: your-password
```

Tag your image:

```
docker tag socket-io-app:1.0 yourusername/socket-io-app:1.0
```

A3: Push to Docker Hub

```
docker push yourusername/socket-io-app:1.0
```

⏱️ This may take 5-15 minutes (uploading 1.48GB)

You'll see:

```
Pushing 1.48GB...
yourusername/socket-io-app:1.0: digest: sha256:abc123...
```

A4: Pull on Server

SSH back into your server:

```
ssh root@your-server-ip
```

Pull the image:

```
docker pull yourusername/socket-io-app:1.0
```

⏱️ Takes 3-10 minutes (downloading 1.48GB)

When done, verify:

```
docker images
```

You should see:

```
REPOSITORY                TAG     IMAGE ID
yourusername/socket-io-app  1.0     c524efcc2771
```

Option B: Using Git + Build on Server (Recommended for Most)

This builds the image directly on the server from your code.

B1: Install Git on Server

```
ssh root@your-server-ip

apt install git -y
```

B2: Clone Your Repository

```
# Create directory
mkdir -p /opt
cd /opt

# Clone your repo (replace with your repo URL)
git clone https://github.com/yourusername/socket-io-realtime-app.git
cd socket-io-realtime-app
```

B3: Build Image on Server

```
docker build -t socket-io-app:1.0 .
```

⏱️ Takes 5-10 minutes

You'll see:

```
Step 1/10 : FROM node:20-alpine
...
Successfully built c524efcc2771
Successfully tagged socket-io-app:1.0
```

B4: Verify

```
docker images
```

Option C: Save & Transfer Image File (If Internet Slow)

This is slower but doesn't need Docker Hub.

C1: Save Image on Your Computer

```
# On your computer
docker save socket-io-app:1.0 -o socket-io-app.tar
```

You'll have a `socket-io-app.tar` file (~1.5GB)

C2: Upload via SCP

```
# On your computer
scp socket-io-app.tar root@your-server-ip:/tmp/
```

Enter password, wait for upload (~5-15 minutes)

C3: Load on Server

```
# SSH into server
ssh root@your-server-ip

# Load the image
docker load -i /tmp/socket-io-app.tar

# Verify
docker images
```

🎯 Step 3: Create docker-compose.yml on Server

SSH into server (if not already connected):

```
ssh root@your-server-ip
```

If you used **Option B (Git Clone)**, skip this - file already exists!

If you used **Option A or C**, create the file:

```
cd /opt/socket-io-realtime-app

# Create docker-compose.yml
nano docker-compose.yml
```

Paste this:

```
version: '3.8'

services:
  app:
    image: socket-io-app:1.0
    ports:
      - "3000:3000"
      - "3001:3001"
    environment:
      - NODE_ENV=production
      - SOCKET_PORT=3001
      - DATABASE_PATH=/app/socket_events.db
    volumes:
      - ./socket_events.db:/app/socket_events.db
    restart: unless-stopped
    container_name: socket-io-app
```

Save: Press `Ctrl+X`, then `Y`, then `Enter`

🚀 Step 4: Run Your App

4.1 Start Container

```
cd /opt/socket-io-realtime-app

docker-compose up -d
```

Wait a few seconds...

4.2 Verify It's Running

```
docker ps
```

You should see:

```
CONTAINER ID   IMAGE                 PORTS                      STATUS
abc123def456   socket-io-app:1.0     0.0.0.0:3000->3000/tcp     Up 5 seconds
```

4.3 Check Logs

```
docker logs -f socket-io-app
```

Look for:

```
[2024-11-08T12:00:00Z] [INFO] Socket.IO server running {"port":3001}
```

If you see errors, press `Ctrl+C` and check troubleshooting section.

🌐 Step 5: Setup Domain & SSL (Optional but Recommended)

5.1 Point Domain to Server

1. Go to your domain registrar (GoDaddy, Namecheap, etc.)
2. Find **DNS Settings**
3. Add/Edit **A Record**:

```
Type: A
Name: @
```

```
Value: your-server-ip
```

4. Wait 5-30 minutes for DNS to update

Test:

```
ping apg-socket.com
# Should show your server IP
```

5.2 Install Nginx (Reverse Proxy)

Nginx forwards traffic from domain to your Docker app.

```
apt install nginx -y
```

5.3 Create Nginx Config

```
nano /etc/nginx/sites-available/socket-io.conf
```

Paste:

```
upstream socket_server {
    server 127.0.0.1:3001;
}

upstream next_app {
    server 127.0.0.1:3000;
}

server {
    listen 80;
    server_name apg-socket.com www.apg-socket.com;

    location / {
        proxy_pass http://socket_server;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Replace `apg-socket.com` with your domain.

Save: `Ctrl+X`, `Y`, `Enter`

5.4 Enable Nginx Config

```
ln -s /etc/nginx/sites-available/socket-io.conf \
  /etc/nginx/sites-enabled/socket-io.conf

# Test config
nginx -t

# Restart Nginx
systemctl restart nginx
```

5.5 Setup HTTPS (SSL Certificate - Free)

```
# Install Certbot
apt install certbot python3-certbot-nginx -y

# Get SSL certificate
certbot certonly --nginx \
  -d apg-socket.com \
  -d www.apg-socket.com
```

Follow prompts (enter your email, agree to terms)

5.6 Update Nginx for HTTPS

```
nano /etc/nginx/sites-available/socket-io.conf
```

Replace entire content with:

```
upstream socket_server {
    server 127.0.0.1:3001;
```

```
}

upstream next_app {
    server 127.0.0.1:3000;
}

# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name apg-socket.com www.apg-socket.com;
    return 301 https://$server_name$request_uri;
}

# HTTPS
server {
    listen 443 ssl http2;
    server_name apg-socket.com www.apg-socket.com;

    ssl_certificate /etc/letsencrypt/live/apg-socket.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/apg-socket.com/privkey.pem;

    location / {
        proxy_pass http://socket_server;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Save and restart:

```
nginx -t
systemctl restart nginx
```

5.7 Test HTTPS

```
curl https://apg-socket.com
```

You should see HTML response (no SSL errors).

✅ Verification Checklist

Run these commands to verify everything works:

Check Container Running

```
docker ps
```

Shows your container? ✅

Check Ports Open

```
netstat -tlnp | grep -E '3000|3001'
```

Shows both ports? ✅

Check Nginx

```
systemctl status nginx
```

Shows `active (running)` ? ✅

Test Socket Connection

On your computer:

```
# Test HTTP
curl http://your-server-ip:3001

# Test HTTPS (if domain setup)
curl https://apg-socket.com
```

Check Logs

```
docker logs socket-io-app
```

Any errors? 🔴 Check troubleshooting
All good? 🟢 You're done!

🔄 Useful Commands for Server Management

View Logs

```
# Last 50 lines
docker logs --tail 50 socket-io-app

# Follow live (like tail -f)
docker logs -f socket-io-app

# Quit: Ctrl+C
```

Restart App

```
docker-compose restart
```

Stop App

```
docker-compose down
```

Restart Server Services

```
systemctl restart docker
systemctl restart nginx
```

Check Server Resources

```
# Memory
free -h
```

```
# Disk space
df -h

# Running processes
top
# Quit: Ctrl+C
```

Update App (After Code Changes)

```
cd /opt/socket-io-realtime-app

# Get latest code
git pull origin main

# Rebuild if changed Dockerfile
docker build -t socket-io-app:1.0 .

# Restart
docker-compose down
docker-compose up -d
```

🐛 Troubleshooting

Problem: "Cannot connect to Docker daemon"

```
# Start Docker
systemctl start docker

# Check status
systemctl status docker
```

Problem: Port 3000 or 3001 already in use

```
# Find what's using the port
lsof -i :3000

# Kill the process
```

```
kill -9 <PID>

# Or change port in docker-compose.yml:
ports:
  - "3002:3000"  # Changed from 3000
```

Problem: "Container exits immediately"

```
# Check logs for error
docker logs socket-io-app

# Common causes:
# 1. Database locked
# 2. Out of memory
# 3. Missing environment variable
```

Problem: "Nginx connection refused"

```
# Check nginx status
systemctl status nginx

# Check nginx config
nginx -t

# Check if ports open
netstat -tlnp | grep nginx
```

Problem: "Cannot access domain/SSL errors"

```
# Check DNS resolution
nslookup apg-socket.com

# If wrong IP, update DNS at registrar (wait 15-30 min)

# Check certificate
certbot certificates

# If expired, renew:
certbot renew
```

Problem: "Out of disk space"

```
# Check usage
df -h

# See what's taking space
du -sh /*

# Clean Docker
docker system prune -a

# Or increase disk (if using cloud provider)
```

Problem: "App works but slow"

```
# Check memory
docker stats

# If using too much, increase in docker-compose.yml:
# (Add under services > app)
deploy:
  resources:
    limits:
      memory: 2G
```

📊 Recommended Server Size

Minimum

```
CPU:    2 cores
RAM:    2-4 GB
Disk:   50 GB
```

Recommended

```
CPU:    4-8 cores
RAM:    8 GB
```

```
Disk:   100 GB
```

Popular Providers

- **DigitalOcean** - $5-24/month
- **Linode** - $5-30/month
- **Vultr** - $2.50-24/month
- **AWS** - Variable pricing
- **Google Cloud** - Variable pricing

🔐 Basic Security Setup

1. Change Root Password

```
passwd
```

2. Create New User (Don't use root)

```
useradd -m -s /bin/bash deploy
passwd deploy

# Add to sudoers
usermod -aG sudo deploy

# Add to docker group
usermod -aG docker deploy
```

3. Setup SSH Keys (Disable Password Login)

On your computer:

```
ssh-keygen -t ed25519

# Follow prompts, accept defaults
```

Copy key to server:

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub deploy@your-server-ip
```

Test:

```
ssh deploy@your-server-ip
# Should login without password
```

4. Disable Root SSH Login

```
sudo nano /etc/ssh/sshd_config

# Find: #PermitRootLogin yes
# Change to: PermitRootLogin no

# Find: #PasswordAuthentication yes
# Change to: PasswordAuthentication no

# Save and restart
sudo systemctl restart sshd
```

📈 Setup Auto-Backup (Recommended)

Create backup script:

```
sudo nano /opt/backup-socket-db.sh
```

Paste:

```
#!/bin/bash
BACKUP_DIR="/backups/socket-db"
mkdir -p $BACKUP_DIR

# Backup database
cp /opt/socket-io-realtime-app/socket_events.db \
  $BACKUP_DIR/socket_events_$(date +%Y%m%d_%H%M%S).db

# Keep only last 30 days
find $BACKUP_DIR -name "*.db" -mtime +30 -delete
```

```
echo "Backup completed at $(date)"
```

Make executable:

```
sudo chmod +x /opt/backup-socket-db.sh
```

Schedule daily at 2 AM:

```
sudo crontab -e

# Add this line:
0 2 * * * /opt/backup-socket-db.sh
```

✨ Summary - Quick Reference

```
# 1. SSH to server
ssh root@your-server-ip

# 2. Install Docker
apt update && apt install docker.io docker-compose -y

# 3. Clone your project
git clone https://github.com/yourusername/socket-io-app.git
cd socket-io-app

# 4. Build image (on server) OR pull from Docker Hub
docker build -t socket-io-app:1.0 .

# 5. Run app
docker-compose up -d

# 6. Check it's running
docker ps

# 7. Setup domain + SSL (optional)
# See "Step 5" above

# Done! 🎉
```

🎯 What You Should Have Now

- ✅ Docker image running on Ubuntu server
- ✅ App accessible on port 3000 (web) and 3001 (socket)
- ✅ Domain pointing to server (optional)
- ✅ HTTPS/SSL working (optional)
- ✅ Database persisted
- ✅ Auto-restart enabled
- ✅ Logs accessible
- ✅ Backup setup

📞 Need Help?

If something doesn't work:

1. **Check logs:** `docker logs -f socket-io-app`
2. **Check troubleshooting** section above
3. **Google the error** - Most problems have solutions online
4. **Search:** "[your error] Ubuntu Docker"

Congratulations! Your app is now live on a server! 🚀