🔒 Why Secure User Management in Docker Matters?

> 🧠 By default, Docker containers run processes as **root**, which is:

- A **huge security risk** 🧨
- Can lead to **host exploitation**
- Bad for **CI/CD and prod environments**

⚠️ NEVER ship containers that run as root in production!

🔍 Real-World Analogy

> 🏠 Giving root access is like giving a guest 🔓 *the master key to your house*, including bank vaults, server room, and more. 🥷 Instead, give them **only what they need** – just one room!

☑️ How to Add a Secure User in Docker

📦 Example (Linux-based):

```
# Create a group & user with no login shell
RUN addgroup --system --gid 1001 appgroup \
 && adduser --system --uid 1001 --ingroup appgroup --disabled-password appuser

# Switch to non-root user
USER appuser
```

🔑 Command	Purpose
`--system`	Marks as a system-level user/group
`--disabled-password`	Prevents password login
`USER appuser`	Runs all next steps as a **non-root user**

🔁 Typical Secure Dockerfile Flow

```
FROM node:20-alpine

WORKDIR /app

# Copy and build with root privileges
COPY . .
RUN npm install && npm run build
```

```
# 🔒 Create a secure user
RUN addgroup -S appgroup && adduser -S appuser -G appgroup

# ☑️ Drop privileges
USER appuser

CMD ["node", "dist/index.js"]
```

🧠 Best Practices for Secure User Management

☑️ Best Practice	💬 Why It's Important
🥷 **Avoid root** in final image	Reduces attack surface
🔐 Use `USER` instruction	Ensures all commands run as non-root
📂 Set correct permissions (`chown`)	Ensure new user can access copied files
🔍 Audit with `docker scan` or `trivy`	Catch misconfigurations
🎯 Keep image minimal	Less packages = fewer CVEs
📜 Use `.dockerignore`	Prevent leaking `.env`, `keys`, `.git`

🛡️ Preventing Permission Issues with Files

```
COPY --chown=appuser:appgroup . .

# OR fix it manually
RUN chown -R appuser:appgroup /app
```

☑️ Ensures the `appuser` has access to source files ⛔ Otherwise you might get `EACCES` or permission denied errors.

🔐 Dockerfile Security Summary Table

Feature	Good Practice	Why?
`USER`	Use non-root user	🧱 Least privilege
`COPY`	Use `--chown` flag	🍪 File ownership fix
`RUN`	Avoid `sudo`, limit shell access	🔒 Prevent privilege escalation
`ENTRYPOINT`/`CMD`	Should not run as root	☑️ Always run app as secure user

---

🧪 Check Current User in Container

You can debug by checking UID:

```
docker run -it your-image whoami
docker run -it your-image id
```

💼 Bonus Tip: Use Docker Compose Securely

```yaml
services:
  api:
    image: dpvasani56/secure-api
    user: "1001:1001"
```

📌 You can enforce user ID even if Dockerfile doesn't specify it.

☑️ Final Checklist for Secure User Management

☑️ Task	Status
Create system user & group	✅
Assign proper UID:GID	✅
Switch user with `USER`	✅
Set file ownership (`--chown`)	✅
Remove unnecessary packages	✅
Test permissions inside container	✅
