Ab basically yaha hum yeh dekhaingay kay application ko as a **root user** run krnay may and **non-root user** run krnay may kya difference hai ? and it is one of important security factor in Containers.

# create a user with permissions to run the app

# -S -> create a system user

# -G -> add the user to a group

# This is done to avoid running the app as root

# If the app is run as root, any vulnerability in the app can be exploited to gain access to the host system

# It's a good practice to run the app as a non-root user

RUN addgroup app && adduser -S -G app app

Now above is the command for running app as a **non-root user**

Ab pehlay yeh smjhlein kay app ko as a **non-root user** and **root-user** run krnay may kya farak hai:

**\*\*1. Root User in Docker Container\*\*:**

- \*\*Powerful Access\*\*: Jab aap apni application ko container ke andar root user ke taur par run karte hain, uska matlab yeh hai ke us container ke andar wo application sab kuch kar sakti hai, jese system files ko modify karna, nayi files create karna, aur system settings change karna.

- \*\*Potential Risk\*\*: Agar application mein koi vulnerability hai (jaise koi security loophole), aur wo application root user ke taur par run ho rahi hai, toh koi attacker is vulnerability ka faida utha kar container ke andar full control hasil kar sakta hai. \*\*Aur agar attacker ne container ke andar root access hasil kar liya, toh uske liye container se host system tak access milna asaan ho sakta hai\*\*.

**\*\*2. Non-Root User in Docker Container\*\*:**

- \*\*Limited Access\*\*: Agar aap application ko non-root user ke taur par run karte hain, toh wo user sirf limited tasks kar sakta hai. Iska matlab yeh hai ke application container ke andar sirf wahi kaam kar sakti hai jo us non-root user ke pass permissions hain.

- \*\*Reduced Risk\*\*: Agar application mein vulnerability ho, aur attacker isko exploit karne ki koshish kare, toh uska access sirf usi non-root user ke level par limited hoga. Wo host system tak ya critical system files tak nahi pahunch sakta.

**\*\*Example\*\*:**

- \*\*Root User Scenario\*\*: Agar app ko root user ke taur par run kiya ja raha hai aur usme koi vulnerability hai, toh attacker container ke andar root access le sakta hai aur container ke boundaries tod kar host system tak pahunch sakta hai.

- \*\*Non-Root User Scenario\*\*: Agar app ko non-root user ke taur par run kiya ja raha hai aur usme koi vulnerability hai, toh attacker sirf us non-root user ke privileges tak hi limited rahega. Uske liye host system tak pahunchna ya system files ko modify karna mushkil ho jayega.

Isliye, Docker container mein application ko non-root user ke taur par run karna ek best practice hai, jo system security ko maintain karne mein madad karta hai.

Now understanding the syntax of running app as **non-root user:**

**\*\*Purpose\*\*:**

The command you're using in your Dockerfile is aimed at creating a new user and group within the Docker container to run your application securely. This is important because running applications as the `root` user inside a container can pose security risks. Running the app as a non-root user limits the potential damage that could be caused if the app is compromised.

\*\*Command Breakdown\*\*:

```Dockerfile

**RUN addgroup app && adduser -S -G app app**

```

1. \*\*`RUN`\*\*:

- This is a Dockerfile instruction that executes a command in the container during the image build process.

2. \*\*`addgroup app`\*\*:

- This command creates a new group called `app` inside the container.

3. \*\*`adduser -S -G app app`\*\*:

- \*\*`adduser`\*\*: This creates a new user.

- \*\*`-S` (System User)\*\*: This flag specifies that the user being created is a system user. System users typically have lower privileges and are used for running background processes or applications rather than logging into the system.

- \*\*`-G app`\*\*: This adds the new user to the `app` group that was created earlier.

- \*\*`app` (final argument)\*\*: This is the name of the new user being created. So, you're creating a system user named `app` and adding it to the `app` group.

**\*\*Why This is Important\*\*:**

- \*\*Security\*\*:

- \*\*Non-Root User\*\*: Running your app as a non-root user (instead of `root`) is a best practice for security. If an attacker finds a vulnerability in your app, they would only have limited access to the system through the non-root user, rather than full control.

- \*\*Preventing Escalation\*\*: Running as a non-root user prevents attackers from easily gaining access to the host system if they exploit a vulnerability in your app.

- \*\*Good Practice\*\*:

- \*\*Isolation\*\*: By creating a dedicated user and group specifically for running the app, you isolate the app’s permissions, reducing the risk of security breaches.

- \*\*Limiting Privileges\*\*: The app only has the privileges of the non-root `app` user, which are much more limited than those of the root user.

\*\*Summary\*\*:

The command `RUN addgroup app && adduser -S -G app app` is creating a non-root system user named `app` and a group named `app` within the Docker container. This user will run your application, which is a security measure to prevent potential vulnerabilities in the app from compromising the host system. Running the app as a non-root user is a best practice in Docker and containerized environments.

Q- Now its mean kay ab jo user srf maa create krunga wohi app ko access krskta hai or koi user nhi krsktay like yaha par : 'RUN addgroup app && adduser -S -G app app' maa ek system user app group ma add kr rha hun toh srf wohi app ko fully access krskta hai

* Haan, ab jo specific user (app) aap create kar rahe hain, wahi user us application ko fully access kar sakta hai. Doosre users ke pass wohi access nahi hoga, aur yeh practice aapki application aur system ki security ko badhane ke liye ek important step hai.

Your understanding is close, but there’s a key clarification needed. Let me explain:

**- \*\*System User Definition\*\*:**

- A \*\*system user\*\* is indeed a specific user account that you create within the container. It’s not a general category that applies to all users; rather, it’s a specific user you create with limited privileges, intended for running specific processes or applications.

- In your case, when you run the command `RUN addgroup app && adduser -S -G app app`, you’re creating a specific user named `app` that belongs to a group also named `app`.

**- \*\*Access and Permissions\*\*:**

- This `app` user is the only user that will run the application inside the container if you set it up that way. It’s not that every user accessing the container becomes a system user; rather, the container's application is explicitly run by this `app` user.

- The `app` user will have access to files and directories that it owns or that are accessible to the `app` group. It will not have access to files owned by other users or groups unless those files are world-readable.

**\*\*Key Points\*\*:**

1. \*\*Specific User (`app`)\*\*:

- The `app` user is a specific, non-root user that you have created within the container. It’s not a general system-wide user that applies to everyone.

2. \*\*Not All Users Become `app`\*\*:

- Other users who might access the container (if allowed) do not automatically become the `app` user. They retain their own identities and permissions.

3. \*\*Access Control\*\*:

- The `app` user’s access is limited to the files and directories that it owns or that are accessible to the `app` group. This limits what the `app` user can do within the container, enhancing security.

**\*\*Summary\*\*:**

- \*\*Correct Understanding\*\*: The `app` user you created is a specific, limited-privilege user within the container. It’s not a general user for everyone; rather, it’s a dedicated user meant to run your application securely. Only this `app` user (or processes running as `app`) will have the permissions and access you've defined.

- \*\*Incorrect Assumption\*\*: Not every user accessing the container becomes a system user or part of the `app` group. Other users retain their own identities and permissions.

--------------------------------------------------------------------------------------------------------------

# set the user to run the app

**USER app**

* Ab in the most first explanation humnay yeh toh smjh lia kay User kasay create krna hai or uska maqsad kya hai , but as we know by default docker run app as **root system,** so usko non-root system pa convert krnay kay liya humay user set krwana hoga using **USER** command , now in above command yehi kia hai kay **USER** command ka through Docker ko user set krdia and **app** is the name of user jo humnay create kia hai.
* Ab jab humna user set krdia toh its mean kay docker nay ab root system say non-root system pa switch krlia and ab jitni be command container ma run hongi wo as a **app** user.

Q- ok But muja yaha yeh smjh nhi aya kay chlo thk hai humna uper 'app' user create krliya and usko set be krdia using USER command but ab jasay let suppose humara container run ho rha hai different user will come on it, Ab Docker ko kesay pta chlega kay yeh 'app' user hi hai ?

* So in short iska answer yeh haka, basically jo hum yeh user create krtay hain iska mtlb yeh nhi hota kay jab be koi user Docker container ko access krega toh wo pehlay usko authenticate krega then access dega, Nhi ! basic yeh saray kam from the creation of User to the setting of User using USER command iss lia kia ja rha hai taka Docker jo by default app ko as **root system** run krta hai , wo ussay switch krka as a **non-root system** app ko run kray.
* Or iss tarah jab **non-root system** pa app run hogi toh jo be user ayega wo jab docker container ko access krega toh uskay pss limited access hogi bcuz humari jo app hai wo as **non-root system** run ho rhi hai.

**More explanation**

Exactly! You've got it right. Let's summarize and clarify:

**\*\*Key Points\*\*:**

1. \*\*Switching from Root to Non-Root\*\*:

- By default, Docker containers run as the `root` user, which has full access to everything inside the container. This is why we use the `USER` command in the Dockerfile to switch from running as `root` to running as a specific non-root user (e.g., `app`).

- When you specify `USER app`, Docker switches the execution context from the powerful `root` user to the more restricted `app` user. This means any commands, processes, or applications inside the container will now run with the limited permissions of the `app` user.

2. \*\*Limited Access\*\*:

- The purpose of switching to a non-root user is to limit the permissions and capabilities of the processes running inside the container. This helps enhance security by reducing the risk that an attacker could exploit vulnerabilities to gain full control over the container or the host system.

- Since the container runs as a non-root user (`app`), it won’t have the ability to perform actions that require root privileges, like modifying system-wide files, installing new software, or changing critical configurations.

3. \*\*No Authentication Mechanism\*\*:

- Docker does not implement any authentication mechanism for switching users within the container. When you use the `USER` command, Docker simply runs all processes under the specified user (`app`), but it doesn’t require any login or authentication.

- This means that once the container starts, everything inside runs as `app` automatically, and there’s no user switching or authentication involved. Docker ensures that the environment operates under the permissions and restrictions of the `app` user, but it doesn’t authenticate users in the traditional sense.