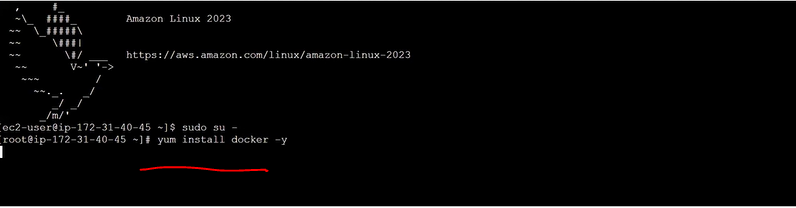
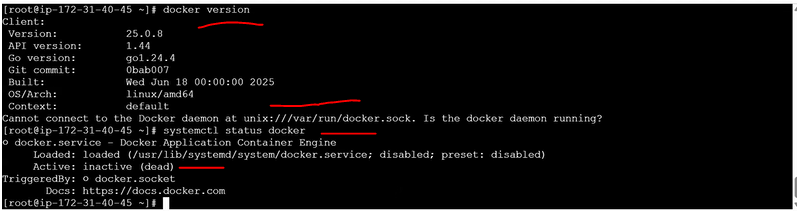
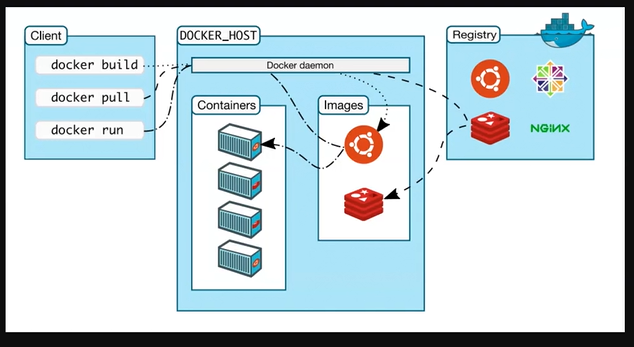
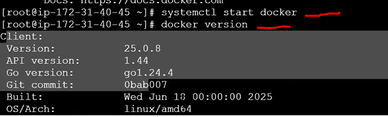
1. Step 1: Launch an EC2 Instance  
   - Create an EC2 instance.  
   - Configure the Security Group (SG) to allow required inbound rules (SSH 22, HTTP 80, HTTPS 443 as needed).  
   - Connect to the EC2 instance and switch to the root user.  
     
   Step 2: Install Docker  
   Run the following commands to install and enable Docker:  
   sudo yum update -y  
   sudo yum install docker -y  
   sudo systemctl start docker  
   sudo systemctl enable docker



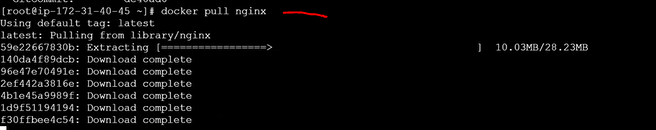
1. Step 3: Verify Docker Installation  
   Check Docker version:  
   docker --version  
     
   Show client and server (daemon) details:  
   docker version  
     
   If Docker daemon shows as inactive, start it:  
   sudo systemctl start docker
2. When we run 'docker version', it displays both the client and the server (daemon) details.  
   - If the Docker daemon is not running, the server section appears inactive.  
   - Start the daemon with: sudo systemctl start docker  
     
   When you run 'docker pull' or 'docker build', Docker first checks if the client can connect to the daemon.  
   If not connected, the command fails. If connected, Docker checks for the image locally; if absent, it pulls from Docker Hub and stores it locally.







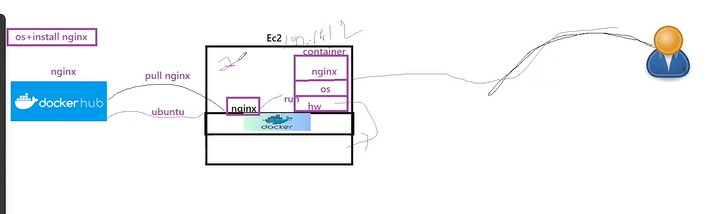
1. Step 4: Pull an Image from Docker Hub  
   Pull any image from Docker Hub. Example:  
   docker pull nginx  
     
   If you don't specify a tag, Docker pulls the 'latest' tag by default.



1. Note: If you don't specify a version/tag, Docker pulls 'latest'.  
   You can verify tags on Docker Hub.  
   Example explicit tag:  
   docker pull nginx:1.25
2. Step 5: List Local Images  
   List images available locally:  
   docker images

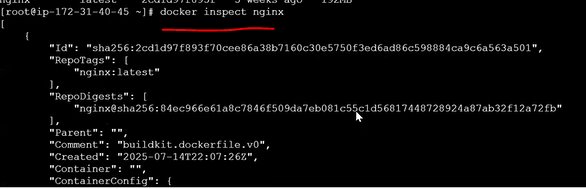


1. Step 6: Run a Container from an Image  
   Create and run a container:  
   docker run -it ubuntu /bin/bash  
   This starts Ubuntu in interactive mode with a bash shell.

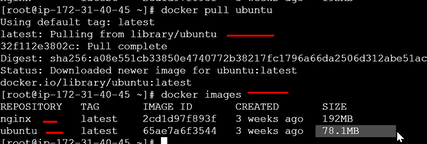


Note: When you run an image, it always runs inside a container.

1. Step 7: Inspect an Image  
   Get detailed information about an image:  
   docker inspect <image\_id\_or\_name>



1. Example: Pull another image and re-check images list:  
   docker pull amazonlinux  
   docker images



Tip: 'nginx' is a service image you can run as a process (web server). 'ubuntu' is a base OS image; you typically start it with a shell like /bin/bash to interact inside.

1. Step 8: Run Modes - Interactive vs Detached  
   - -it (Interactive Terminal): Foreground, attaches your terminal to the container.  
   - -d (Detached): Background, container runs without an attached terminal.
2. - \*\*-it (Interactive Terminal)\*\*: Connects directly to the container in the foreground. Exiting the shell ends that session.  
   - \*\*-d (Detached)\*\*: Runs in the background without attaching. The container continues until the main process exits.
3. Detached mode runs in the background; not connected to the terminal directly. Useful for services that should keep running.
4. About /bin/bash:  
   We pass '/bin/bash' to keep the container alive interactively. If no process runs in a container, it exits. Using bash provides a continuous process so the container remains active.



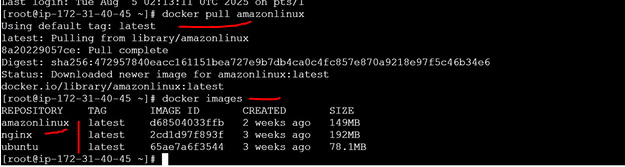
1. Exiting the interactive bash shell stops that process. If bash was the main process, the container will also stop.



1. Step 9: Check Containers  
   Running containers:  
   docker ps  
     
   All containers (including stopped):  
   docker ps -a



1. Step 10: Amazon Linux Image  
   Pull and run the Amazon Linux image when needed for testing minimal OS behavior.



1. In a full EC2 Amazon Linux instance, many background processes run by default.  
   In a minimal Docker image, fewer dependencies are included, giving tighter control.



1. Now see In EC2 amazon linux 119 process are running in background but if we will login into docker linux image and see how many process are running.
2. Login into an Amazon Linux container:  
   docker run -it amazonlinux /bin/bash



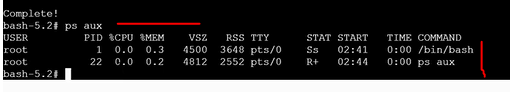
Inside the Amazon Linux container, you'll see far fewer processes. The 'ps' command may be missing until you install it:  
yum update -y  
yum install -y procps  
ps -ef



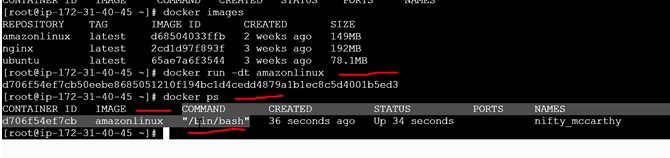
1. Install 'ps' (procps) to check running processes inside the container:  
   yum install -y procps



1. After installing, 'ps aux' will display running processes inside the container.

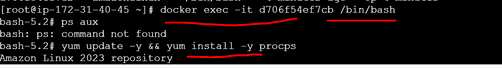


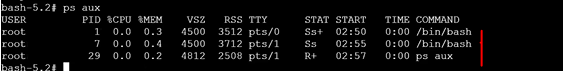
1. Important: A container stays active only while its main process runs. If the main process exits, the container stops.
2. If you run a container in detached mode without a long-running command, it may exit immediately. Docker uses the image's default CMD (from the Dockerfile) if no command is provided.



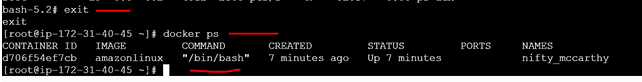
1. Step 11: Exec Into a Running Container(Login into container)  
   Open a shell into a running container:  
   docker exec -it <container\_id\_or\_name> /bin/bash

If you exec into a container while another process is already running, you may see multiple bash processes. Exiting the exec shell will not stop the main process.

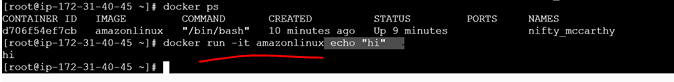




Exiting an 'exec' shell does not stop a detached container; the main process continues.



1. Step 12: Overriding CMD and Entrypoint  
   Specifying a command after the image name overrides the image's default CMD.  
   Example:  
   docker run -it ubuntu sleep 60  
   This runs 'sleep 60' instead of '/bin/bash'. The container exits after 60 seconds.
2. If you provide a different command in '-it' mode, it replaces the default CMD for that run. The container lives only for the duration of that command.

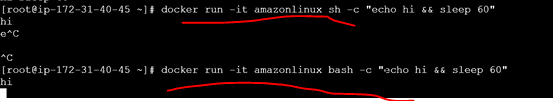




'/bin/bash' keeps the session active. You remain inside until you exit or kill the process.



1. Example of a short-lived process inside a container:  
   sleep 60  
   After 60 seconds, the process ends and the container stops (if that was the main process).



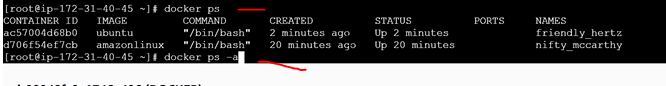
After 60 sec its automatically killed.





Key takeaway: A container remains active only while its main process is running. Ensure a long-running process (e.g., bash, nginx, python) for persistent containers.

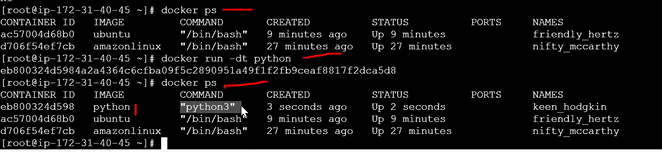
1. Step 13: List Containers by State  
   - Running: docker ps  
   - All: docker ps -a  
   - Exited only: docker ps -f status=exited



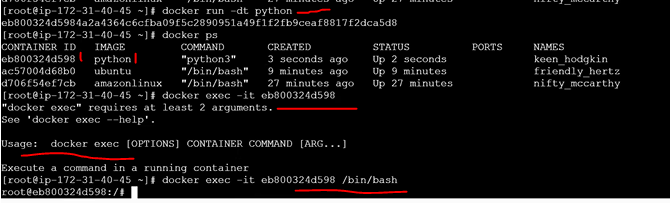
1. Show only exited containers:  
   docker ps -f status=exited



1. Whether '-d' or '-it', container lifetime depends on the command (main process). If it ends, the container stops.



1. Tip: If a production container is started with a non-bash CMD, use 'docker exec -it <container> /bin/bash' (or 'sh') to get a shell without altering how the service started.



1. Step 14: Stop or Kill Containers  
   Stop gracefully:  
   docker stop <container\_id\_or\_name>  
   Force kill:  
   docker kill <container\_id\_or\_name>
2. Step 15: Python Image Example  
   The official Python image often uses 'python3' as the default CMD, which keeps the container running when appropriate.  
   Override CMD to open a bash shell:  
   docker run -it python /bin/bash

