Lab 1 — Install Docker on Ubuntu (from scratch)

System prep & repo

- `sudo apt update && sudo apt upgrade -y` → refresh package lists & apply updates.
- `sudo apt install apt-transport-https ca-certificates curl software-properties-common` → tools to add HTTPS repos & manage keys.
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg` → add Docker's GPG key so apt can trust packages.
- `echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null` → add official Docker repo.
- `sudo apt update` → load repo metadata (now includes Docker).

Install & verify

- `sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin` → install engine, CLI, runtime, Buildx, Compose (v2).
- `sudo systemctl enable --now docker` → enable and start daemon (if not already).
- `sudo docker run hello-world` → pull + run test image; prints success text if everything works.
- (Optional) `sudo usermod -aG docker $USER && newgrp docker` → run Docker without `sudo`.

Lab 2 — Linux Commands & Scripting (inside a safe container)

Start a disposable Ubuntu container (saves work to host)

- `sudo docker run -it --rm -v ~/lab_out:/root ubuntu:24.04 bash` → enter an interactive shell; mount `~/lab_out` for persistence.
- Inside container:
    - `apt update && apt install -y file tar gzip findutils coreutils` → basic tools.

Navigation & inspection

- `pwd` / `ls` / `ls /usr/bin` / `cd /usr/local` / `cd` → path, list, move.
- `which sh` → show command's absolute path in `PATH`.
- `file /bin/ls` → file type (ELF binary? symlink?).
- `test -x /bin/sh && echo "Executable" || echo "Not Executable"` → quick permission check.

Permissions & counts

- `ls -l /bin/ls > /root/ls_perm.txt` → long listing to file.
- `printf "\nExplanation: ...\n" >> /root/ls_perm.txt` → append your note (r/w/x meaning).
- `ls -1 /bin | wc -l > /root/bin_count.txt` → count entries in `/bin`.

Find + sort + head pattern (very exam-able)

- `find /usr -type f -printf '%s %p\n' 2>/dev/null | sort -nr | head -n 3 > /root/top3_usr.txt` → top 3 largest files under `/usr`. (Understand each flag.)

Write a reusable shell script

Create:

```
 cat > /root/top3_archive.sh <<'SH'
#!/usr/bin/env sh
TARGET=${1:-/usr}; OUT_DIR=${2:-/root}; N=${3:-3}; TMP=/tmp/top_files.list
mkdir -p "$OUT_DIR"
find "$TARGET" -type f -printf '%s %p\n' 2>/dev/null | sort -nr | head -n "$N" > "$TMP"
if [ -s "$TMP" ]; then awk '{print $2}' "$TMP" | tar -czvf "$OUT_DIR/top_${N}_files.tar.gz" -T -; fi
rm -f "$TMP"
SH
```

- `chmod +x /root/top3_archive.sh` → make executable; run it: `/root/top3_archive.sh /usr /root 3`.

Ownership & chmod

- `touch my_file.txt && ls -l my_file.txt` → create & view owner/group.
- `useradd -m labuser` → create user (lab).
- `chown labuser my_file.txt` → transfer ownership.
- `chmod 600 my_file.txt` → owner read/write only (600). Explain 6=rw, 0,0.

Processes

- `ps aux` → list all processes with details.
- `apt install -y psmisc && pstree -p` → process tree (needs psmisc).
- `sleep 3600 &` → background process.
- `ps aux | grep sleep` → find PID; `kill <PID>` → terminate it.

- `exit` → leave container; check files preserved in `~/lab_out` on host.

Lab 3 — Docker Images, Containers, Volumes, Compose (and meaning of commands)

Basics & housekeeping

- `docker --version` / `docker login` → check install; auth to Hub.
- Pull images: `docker pull nginx` / `docker pull redis` / `docker pull mongodb/mongodb-community-server:7.0.2-ubi8` → fetch by name:tag.
- `docker images` → list local images; `docker rmi <image>` → delete image by name/ID.
- Prune: `docker image prune` (dangling) / `docker image prune -a` (all unused) / `docker system prune [-a]` (containers + networks + images).
- Tagging: `docker tag nginx:latest nginx:22sep` → second name pointing to same image ID.

Build & run

Minimal nginx site: write `index.html`, then Dockerfile:

```
 FROM nginx:stable-alpine3.17-slim
COPY index.html /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- Build & run: `docker build -t justasample:v1 .` → build from Dockerfile; `docker run -p 8080:80 justasample:v1` → map host 8080→container 80.

- Push (if you want): `docker tag justasample:v1 <hubuser>/sampleapp:v1` then `docker push <hubuser>/sampleapp:v1`.

Containers: lifecycle & logs

- Start Redis: `docker run -d --name myredis -p 6379:6379 redis:latest` (detached).
- `docker ps` / `docker ps -a` → running / all containers. `docker logs <id>` → view container logs.
- Stop/remove: `docker stop <id>` then `docker container prune` or `docker rm -f <id>` to force remove. `docker start <id>` to restart.
- Copy files: `docker cp <cid>:/etc/nginx/nginx.conf ./` and `docker cp ./check.txt <cid>:/etc/nginx/` → move between host & container.
- Exec shell: `docker exec -it <cid> /bin/bash` → open shell inside container.
- `docker info` → daemon & environment details.

Storage types

- Volumes: `docker volume create myvol` / `docker volume ls` / `docker volume inspect myvol` / `docker volume rm myvol` / `docker volume prune`.
- Use a volume:
  `docker run -d --mount source=myvol,target=/app/data --name mount_container busybox sleep 3600` → persists at `/var/lib/docker/volumes/...`.
  Bind mount: `docker run -d -v /home/user/data:/container/data --name bind_container busybox sleep 3600` (host directory into container).

Compose (multi-container)

- Core workflow: `docker-compose up -d` / `docker-compose down` / `docker-compose up --build -d` / `docker-compose logs -f` / `docker-compose ps` / `docker-compose up -d --scale <svc>=N`.

Minimal NGINX:

```
 version: "3.9"
services:
  web:
    image: nginx:latest
    ports: ["8080:80"]
```

- Then `docker-compose up -d` → browse http://localhost:8080.
- Example Hadoop (NN+DN) compose with ports & volumes also included in notes for reference.

Lab 4 — Hadoop (Single-Node / Phase-1 & Phase-2)

Pre-reqs & cloning

- Ensure Docker + Compose v2 installed (see Lab 1). `sudo apt install -y docker-compose-plugin` if missing.

Workspace:

```
 mkdir -p ~/docker/hadoop && cd ~/docker/Hadoop
git clone https://github.com/tomwhite/hadoop-book.git
git clone https://github.com/big-data-europe/docker-hadoop.git
cd docker-hadoop
```

- (Optional) expose UI ports in `docker-compose.yml`: NN:9870, RM:8088, NM:8042, HS:8188 (and DN UI for debug).

Bring up the cluster

- `docker compose up -d` → start NN/DN/RM/NM/HS (single worker in Phase-1).

- `docker compose ps` / `docker compose logs <service> --tail 50` → check status/logs.

HDFS + WordCount (classic flow)

- Make HDFS input dir: `docker compose exec namenode hdfs dfs -mkdir -p /user/root/input`
- (If using tiny sample) copy a file: `docker compose cp ~/docker/hadoop/words.txt namenode:/tmp/words.txt` → `docker compose exec namenode hdfs dfs -copyFromLocal /tmp/words.txt /user/root/input`

Get examples JAR (host):

```
 cd ~/docker/hadoop
curl -L -O
https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-mapreduce-examples/2.7.1/hadoop-mapreduce-examples-2.7.1.jar
docker compose cp hadoop-mapreduce-examples-2.7.1.jar namenode:/tmp/
```

-
- Run WordCount:
  `docker compose exec namenode hadoop jar /tmp/hadoop-mapreduce-examples-2.7.1.jar org.apache.hadoop.examples.WordCount /user/root/input /user/root/output_wc`
  View output: `docker compose exec namenode hdfs dfs -cat /user/root/output_wc/part-r-00000` (redirect to file if huge).

JVM daemons & cleanup

- `docker compose exec <service> jps` → verify NN/DN/RM/NM/HS Java procs.
- `docker compose down` (keep volumes) / `docker compose down -v` (wipe HDFS).

Lab 5 — MapReduce Detailed (multi-node, datasets, monitoring, snapshots)

Full cleanup (fresh start)

Create lab dir & stop anything old:

```
 cd ~ && mkdir -p hadoop-lab && cd hadoop-lab
docker compose down || docker-compose down
docker rm -f namenode datanode1 datanode2 datanode3 resourcemanager nodemanager1
nodemanager2 nodemanager3 historyserver || true
sudo rm -rf ./data
rm -f docker-compose.yml docker-compose.phase2.yml part-r-00000_*
```

- (Use `docker compose` for v2; `docker-compose` for v1.)

Compose cluster (NN+3 DN + YARN + HS) & bring up

- Write the provided `docker-compose.yml` (services, env, ports, healthchecks).
- `mkdir -p data/namenode data/datanode1 data/datanode2 data/datanode3`
- `docker compose up -d` → start; `docker ps` to verify. (RM RPC exposed on 8035; UIs mapped.)

Prepare input data

- Base text: create `words.txt` on host; copy in: `docker cp words.txt namenode:/tmp/words.txt` → verify `docker exec namenode ls -lh /tmp/words.txt`.
- Generate 4 datasets inside NN (set0 small; set1≈1 GB; set2≈1.5 GB; set3≈2 GB): the loop appends until byte target; then `ls -lh words_set*.txt`. (All done via `docker exec namenode bash -c '…'` block in notes.)

Load to HDFS (idempotent)

- Cleanup old HDFS dirs: `hdfs dfs -rm -r -f /user/inputdata /user/output_wc_set{0,1,2,3}`
- Make dirs: `hdfs dfs -mkdir -p /user/inputdata/set{0,1,2,3}`

Upload:

hdfs dfs -put /tmp/bigdata/words_set0.txt /user/inputdata/set0/
hdfs dfs -put /tmp/bigdata/words_set1.txt /user/inputdata/set1/
hdfs dfs -put /tmp/bigdata/words_set2.txt /user/inputdata/set2/
hdfs dfs -put /tmp/bigdata/words_set3.txt /user/inputdata/set3/

-
- Inspect: `hdfs dfs -ls -R /user/inputdata` (take a screenshot).

Find & stage examples JAR inside NN

- `find /opt/hadoop* /usr/local/hadoop* -name "*examples*.jar"` → locate JAR; copy to `/tmp/hadoop-examples.jar`.

Run WordCount on all datasets

- For each dataset:
  - Remove old output: `hdfs dfs -rm -r -f /user/output_wc_setX`
  - Time & run: `hadoop jar /tmp/hadoop-examples.jar wordcount /user/inputdata/setX /user/output_wc_setX`
  - Grab result: `hdfs dfs -get /user/output_wc_setX/part-r-00000 /tmp/part-r-00000_setX`
- After all: `ls -lh /tmp/part-r-00000_*` then copy to host: `docker cp namenode:/tmp/part-r-00000_set{0,1,2,3} ./` → preview: `head -20 part-r-00000_set3`. (Low unique-token count explains short output.)

Cluster health (must-know admin commands)

- HDFS usage: `hdfs dfs -du -h /user/inputdata` → per-dir sizes.
- Counts: `hdfs dfs -count -h /user/inputdata` → files/dirs/bytes.
- Cluster report: `hdfs dfsadmin -report` → NN view of DNs, capacity, used, remaining.
- Integrity: `hdfs fsck / -files -blocks -locations | head -50` → list files, blocks, and replica locations.

YARN job tracking

- `yarn application -list -appStates ALL` → list all apps (IDs, states, times).
- `yarn application -status <application_id>` → details of one job (use any ID from list).

Web UIs (prove everything ran)

- NN: `http://localhost:9870` (overview + Datanodes tab)
- RM: `http://localhost:8088` (Applications → WordCount jobs)
- HS: `http://localhost:8188` (finished jobs history)
- NM: `http://localhost:8042` (per-node)
- DN UIs: `9864/9865/9866` if mapped
- Quick probe (one-liner curl check provided) also in notes.

Snapshots & replication (very important)

- Enable & create: `hdfs dfs -createSnapshot /user/inputdata snap_lab` → point-in-time copy.
- List snapshots: `hdfs dfs -ls /user/inputdata/.snapshot`

- Set replication: `hdfs dfs -setrep -w 3 /user/inputdata/set3/words_set3.txt` → enforce 3 replicas (waits `-w`).
- Verify blocks/replicas: `hdfs fsck /user/inputdata/set3/words_set3.txt -files -blocks`
- Delete snapshot: `hdfs dfs -deleteSnapshot /user/inputdata snap_lab`. (Great screenshot/explain question.)

Stop & (optionally) clean

- `docker compose down` → stop cluster, keep data in `./data`.
- (Reset) `sudo rm -rf ./data` → wipe NN/DN storage to start clean next time.

Tiny index by "what sir can ask"

- **Meaning of Docker commands** (pull/run/ps/logs/stop/rm/exec/cp/prune/tag/build/compose up-down-logs-scale) → Lab 3.
- **Linux file system/default dirs/permissions & scripts** (pwd/ls/cd/which/file/test/ls -l/chmod/chown/find/sort/head/heredoc/chmod +x) → Lab 2.
- **Hadoop admin & data locality** (dfs -mkdir/-put/-ls/-du/-count; dfsadmin -report; fsck; run MR; YARN app list/status; UIs) → Labs 4 & 5.
- **Snapshots & replication** (`-createSnapshot`, `-deleteSnapshot`, `-setrep`, `fsck`) → Lab 5.
- **Installing Docker/Compose correctly** (GPG key, repo, compose plugin) → Lab 1.