ANNEX A · COMPOSING SERVICES · PRACTICAL 6.5

Name: _____ **Date:** _____

LAB A · THE INTRANET AS A COMPOSE FILE

1 · Tool and file. On `ubu-srv` (`ssh srv`): `sudo apt install -y podman-compose`, then `podman rm -f web intranet db`, `cd ~/intranet` and `nano compose.yaml`. Two spaces per level, no tabs:

```
services:
  web:
    image: docker.io/library/nginx:1.30
    ports:
      - "8080:80"
    volumes:
      - ./site:/usr/share/nginx/html:ro
```

☐ **Done when:** `cat -A compose.yaml` shows no `^I` (that would be a tab).

2 · Up.

```
podman compose up -d
podman compose ps
curl -s http://localhost:8080 | grep "<p>"
podman compose logs
```

☐ **Done when:** `curl` shows your page. The container's name is: _____

3 · Down, change, up. `podman compose down`. Change the host port to 8090 in the file. `podman compose up -d`.

☐ **Done when:** `curl http://localhost:8090` answers and port 8080 does not. Then `podman compose down`: port 8080 must be free for the practical.

PRACTICAL 6.5 · TRAMUNTANA GIT

Tramuntana Apps wants its own Git server: **Gitea**, with a **PostgreSQL** database. Report `practical-6.5-surname.md` on Google Classroom: commands, outputs (text), answers of task 9.

4 · The two files. `mkdir ~/app && cd ~/app` (the project's application lives in `~/app`). First the secret, in a file called `.env` (choose your own password), then `chmod 600 .env`:

```
DB_PASSWORD=write-your-own-password-here
```

Then `compose.yaml`. **Fill in the three blanks** before you type it (the application must answer on port 8080 of the server):

```
services:
  server:
    image: docker.gitea.com/gitea:28.0.0
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - GITEA__database__DB_TYPE=postgres
      - GITEA__database__HOST=______:5432
      - GITEA__database__NAME=gitea
      - GITEA__database__USER=gitea
      - GITEA__database__PASSWD=${DB_PASSWORD}
    restart: always
    volumes:
      - gitea-data:/data
    ports:
      - "______:3000"
      - "2222:22"
    depends_on:
      - db
```

```
  db:
    image: docker.io/library/postgres:18
    restart: always
    environment:
      - POSTGRES_USER=gitea
      - POSTGRES_PASSWORD=${______}
      - POSTGRES_DB=gitea
    volumes:
      - db-data:/var/lib/postgresql

volumes:
  gitea-data:
  db-data:
```

The right box continues the same file, below the left one. Keep the blank line before `db:` and before `volumes:`.

☐ **Done when:** `ls -la` shows `.env` with permissions `-rw-------` and `compose.yaml`.

5 · Up.

```
podman compose up -d
podman compose ps
podman compose logs db
podman volume ls
```

☐ **Done when:** two containers are `Up` and the database log ends with `ready to accept connections`.

6 · First-run setup. In the browser of your host: `http://192.168.122.10:8080`. On the page Initial Configuration:

- Database Settings: already filled in. Where did these values come from? _____
- General Settings: Site Title `Tramuntana Git` · SSH Server Port `2222` · Gitea Website URL `http://192.168.122.10:8080/` (leave HTTP Server Port at 3000)
- Optional Settings → Administrator Account Settings: your first name, an e-mail, a password (not the database one).
- Press Install Gitea. Sign in. Create a repository called `intranet`.

☐ **Done when:** the browser tab says "Tramuntana Git" and you see your empty repository `intranet`.

7 · Where is the data?

```
podman exec app_db_1 psql -U gitea -c "SELECT name FROM repository;"
podman compose down
podman ps -a
podman volume ls
podman compose up -d
```

☐ **Done when:** after `down` there are no `app_...` containers but two `app_...` volumes; after `up` you can sign in and `intranet` is still there, with no setup page.

8 · Survive your logout.

```
sudo loginctl enable-linger $USER
exit
```

☐ **Done when:** with your SSH session closed, the browser still opens Tramuntana Git. Add today's entry to `~/server-log.md` and tick line 8 of the project checklist.

9 · Answer in the report.

1. Why can the service `server` reach the database using just the name `db`?
2. Why does the service `db` have no `ports` section?
3. What would `podman compose down -v` do? (Do not run it.)

Marking (10): files and `.env` permissions 2 · both containers up 2 · setup, repository and the `psql` row 2 · `down` / `up` test 2 · answers 2.

EXIT TICKET

☐ **1.** What is a compose file for?
A) Building an image B) Describing all the services of an application in one file
C) Making backups of volumes D) Installing Podman

☐ **2.** How does one service find another one in the same compose project?
A) By `localhost` B) By the server's IP C) It cannot D) By the service name

☐ **3.** After `podman compose down`, what is still there?
A) The named volumes B) The containers C) The network D) Nothing

☐ **4.** In YAML, what decides that `image` belongs to the service `web`?
A) Capital letters B) The order of the lines C) The indentation D) A semicolon

☐ **5.** Why can a PC in the office not connect to the PostgreSQL of our Gitea?
A) It is encrypted B) The `db` service publishes no port C) PostgreSQL is stopped D) Its password is long