ITI8801 · Cloud Architectures and DevOps

Practice 5 · Assignment 5: your own service, in containers, on your VM

Siim Rebane · TalTech · Autumn 2026

What you build

A small web service that you write yourself, implementing an API document you are given. It runs as a container from an image you build, next to a reverse proxy and a PostgreSQL database, all three from one Compose file, on a VM you create with Terraform in your cloud.

From the internet, one port answers: 80, the reverse proxy. It forwards to your service; your service stores its data in PostgreSQL.

Part	Written by you
the service	any language and framework
its image	a Dockerfile
the three containers together	a `compose.yaml`
the VM and its firewall	Terraform

Your API

Every student implements the same API for a different resource, chosen by the last digit of the student code (the digit before the letters):

Digit	Resource	Number field		Digit	Resource	Number field
0	`/api/bookmarks`	`visits`		5	`/api/songs`	`seconds`
1	`/api/recipes`	`minutes`		6	`/api/trips`	`kilometres`
2	`/api/exercises`	`repetitions`		7	`/api/boardgames`	`players`
3	`/api/plants`	`heightCm`		8	`/api/lectures`	`attendees`
4	`/api/tickets`	`priority`		9	`/api/devices`	`batteryPercent`

The OpenAPI document for your code: on the validator's page, `https://iti8801.digivader.com/lab5-validator/`, enter your student code and download it. The document is the specification; where this slide and the document differ, the document is right.

The API in short

For `/api/boardgames` with `players`; yours has your resource and field. The data is stored in PostgreSQL.

Request	Answer
`GET /health`	200 `{"status":"ok"}` when the database answers; 503 when it does not
`GET /api/boardgames`	200, a JSON array of all items
`GET /api/boardgames?name=Catan`	200, only the items with exactly that name
`POST /api/boardgames` `{"name":"Catan","players":4}`	201, the created item with the `id` the service assigned
`POST` with no name, an empty name, a negative number, a number as a string, or not JSON	400 `{"error": "..."}`
`GET /api/boardgames/{id}`	200 the item, or 404
`DELETE /api/boardgames/{id}`	204 with no body, or 404

An item is `{"id": 1, "name": "Catan", "players": 4}`: `id` and the number field are JSON numbers, not strings; `name` is 1 to 100 characters, without zero bytes (PostgreSQL cannot store them).

The image

Requirement	Checked how
built from your Dockerfile, pushed to a **public** repository	the validator pulls it without a login
has a `linux/amd64` variant	the VMs are amd64; an image built on an Apple-silicon Mac is arm64 unless told otherwise
at most **150 MB** compressed	the sum of its layers in the registry
runs as a user that is not root	the image's config: `USER` set, not `0`, not `root`
no **fixable** HIGH or CRITICAL findings	Trivy, with `--ignore-unfixed`: findings without a fixed version do not count

Registries the validator can read without a login: GitHub Container Registry (`ghcr.io`, the package made public) and Docker Hub (`docker.io`, a public repository). An Azure or AWS registry works if its repository allows anonymous pulls.

The Compose file

One `compose.yaml` runs three services:

Service	Image	Rules
a reverse proxy	nginx, Caddy or Traefik	the only service with `ports:`; publishes 80 and forwards to your service
your service	your image	reads the database address and password from its environment
the database	`postgres` (any version from 16)	its data in a named volume; a `healthcheck`

- At least two networks. The reverse proxy and the database share none, so the proxy cannot reach the database
- Your service starts after the database is healthy
- The database password comes from a `.env` file next to `compose.yaml`; the `.env` file is not in git and not in the upload

The VM

Terraform creates the VM in your cloud and everything it needs: network, public IP, firewall.

	Azure	AWS
the VM	`azurerm_linux_virtual_machine`	`aws_instance`
its firewall	`azurerm_network_security_group`	`aws_security_group`
inbound allowed	80 from the internet; 22 only from your own address, or not at all	the same

Docker and the Compose plugin are installed on the VM; how is up to you (a boot script, or by hand after it starts).

The validator

`https://iti8801.digivader.com/lab5-validator/` : your student code, your VM's public address, and your image's full name, for example `ghcr.io/you/a5-service:1` .

Box	Green when
Address is in a cloud	the address belongs to Azure's or AWS's published ranges
API	every step of the API test answers as your document says; the test stops at the first step that does not, and says what it got
Only port 80	22, 5432, 8080, 8000, 3000, 5000 and 8081 do not answer from the validator
Image	amd64, at most 150 MB, not root, no fixable HIGH or CRITICAL findings; each finding listed with its package and fixed version
Configuration	your upload is accepted

The first four are checked live. The fifth is your files, uploaded on the page once the four are green. Every check has its own code.

What the upload must contain

- your **Dockerfile**
- your `compose.yaml`
- every `.tf` file

Refused: `.env`, state files, `.tfvars`, keys. A refused upload lists every reason and stores nothing. Accepted when:

- the Dockerfile has a `FROM` line
- the Compose file has at least three services, exactly one with `ports:`, a `postgres` service without ports, with a named volume and a healthcheck, at least two networks, and no network shared by the published service and the database
- the `.tf` files contain the VM and firewall resource types of your cloud

What to submit, and when

- The validator's code of a check that is **PASSED**: the four live boxes green and the upload accepted. Paste it into Moodle.
- Deadline: Sunday next week, 23:59, as for every assignment.
- Then `terraform destroy`.

Reference: test your API before the validator does

Schemathesis tests a running API against an OpenAPI document, as in lecture 5:

```
docker run --rm --network host -v $PWD:/spec schemathesis/schemathesis:stable \
    run /spec/boardgames-api.yaml --url http://localhost:8080
```

- `--network host` reaches the service on your laptop's `localhost`
- `/spec/boardgames-api.yaml` is the document you downloaded, in the current directory
- `--url` is where your service listens

Schemathesis checks every answer against the document. The validator also checks behaviour the document states in words, such as `?name=` returning only the items with exactly that name.

Reference: scan your image before pushing

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v trivycache:/root/.cache \
    aquasec/trivy:latest image --ignore-unfixed --severity HIGH,CRITICAL <your image>
```

The same options the validator uses. Base images carry findings of their own, measured on 2026-09-27:

Base image	Fixable HIGH / CRITICAL	In
`python:3.13-slim`, `-alpine`	2 HIGH	`setuptools`, `msgpack`: packages pip brings along
`node:22-slim`, `-alpine`	8 HIGH	packages inside npm itself
`eclipse-temurin:21-jre`, `-jre-alpine`	0	
`mcr.microsoft.com/dotnet/aspnet:8.0-alpine`	0	

Trivy's `Target` column says where a finding is: an OS package, a language package, or a tool such as pip or npm.

Reference: building for the VM on a Mac

A Mac with Apple silicon builds `linux/arm64` images; the VM runs `linux/amd64`. On the VM such an image fails with `exec format error`, and the validator's image box says the image has no `linux/amd64` variant.

```
docker build --platform linux/amd64 -t ghcr.io/<you>/a5-service:1 .
```

`docker image inspect <image> --format '{{.Os}}/{{.Architecture}}'` prints what an image was built for.

Reference: pushing

GitHub Container Registry:

```
echo <token> | docker login ghcr.io -u <github user> --password-stdin
docker push ghcr.io/<github user>/a5-service:1
```

- the token: GitHub → Settings → Developer settings → Personal access tokens, with the scope `write:packages`
- a new package is **private**. On GitHub: your profile → Packages → the package → Package settings → Change visibility → Public

Docker Hub:

```
echo <token> | docker login -u <docker hub user> --password-stdin
docker push <docker hub user>/a5-service:1
```

- the token: Docker Hub → Personal access tokens, Read & Write
- a repository created by a push is public unless your account's default says otherwise

Reference: Azure for Students regions and sizes

An Azure for Students subscription may deploy only to some regions. The list is a policy on the subscription:

```
az policy assignment list --query "[].{name:displayName, params:parameters}" -o json
```

On one course subscription, on 2026-09-27: `denmarkeast`, `francecentral`, `italynorth`, `swedencentral`, `norwayeast`. Anywhere else, every resource is refused with `RequestDisallowedByAzure`.

A size can be out of stock in a region: `SkuNotAvailable … currently not available in location`. Another size or another allowed region works. The regional quota is 6 vCPUs; `az vm list-usage -l <region>` shows what is used.

Reference: running it on the VM

Copy the files the VM needs, and start:

```
scp compose.yaml nginx.conf azureuser@<ip>:~/
ssh azureuser@<ip>
nano .env                          # DB_PASSWORD=… and the image name, if compose.yaml reads it
docker compose up -d
docker compose ps
curl -s localhost/health
```

- the image is pulled from your registry; the VM needs no login for a public repository
- a new image: push it, then `docker compose pull && docker compose up -d` on the VM
- `docker compose logs -f <service>` shows why a container stops



Questions