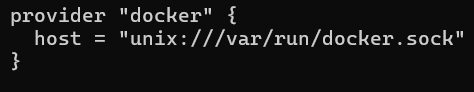
**Terraform Infrastructure Documentation on VPS Platform**

Source Code Workflow Explanation (Line by Line)

1. Root main.tf



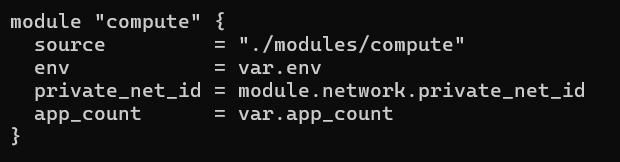
* This tells Terraform: **talk to Docker** on your VPS using the Docker socket (/var/run/docker.sock).
* Without this, Terraform cannot manage containers.

**Example:** Like saying *“Hey Terraform, use this phone line to call Docker.”*

A black background with white text

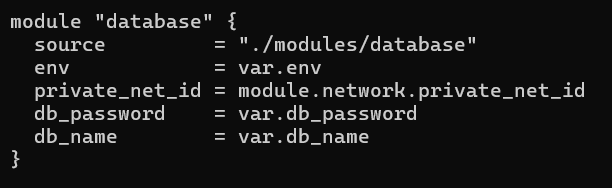
AI-generated content may be incorrect.

* Calls the **network module**.
* source → where module code lives (modules/network).
* env → passes environment name (dev).

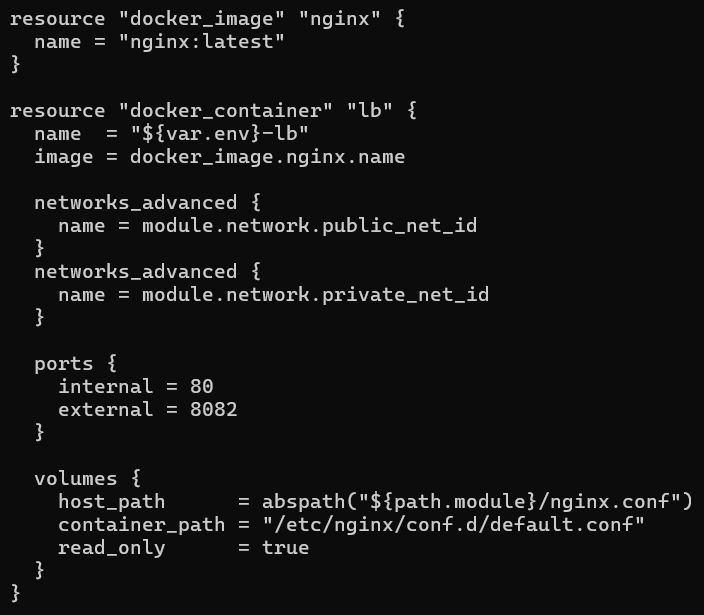


* Deploys apps.
* private\_net\_id = module.network.private\_net\_id
* Takes private network ID created by network module and attaches apps there.
* app\_count = var.app\_count
* Number of app containers to run.

**How it connects:** Apps join private network, so DB + LB can talk to them.

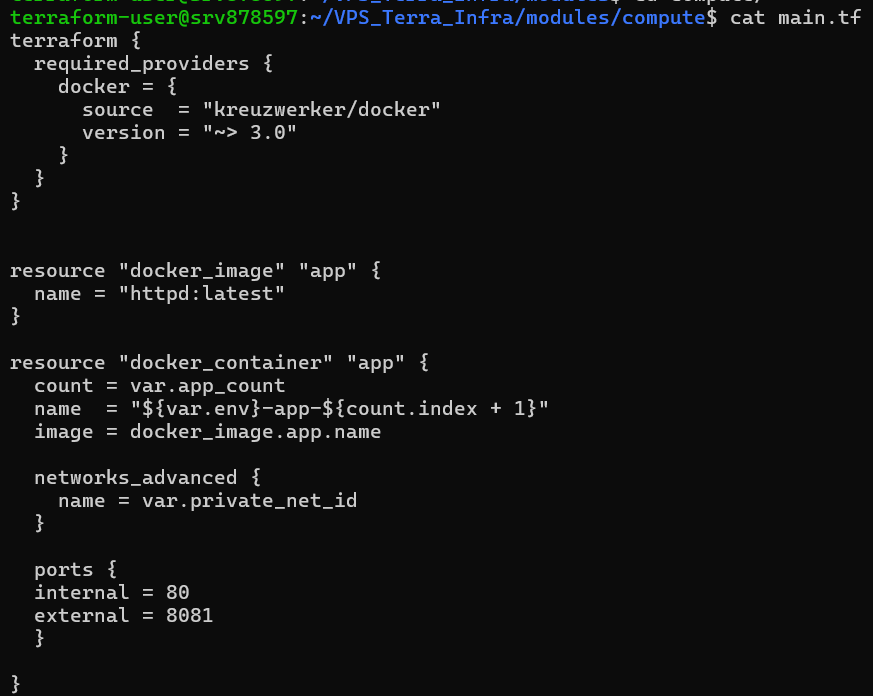


* Deploys DB.
* Joins private network (same as apps).
* Gets password & db name from variables.
* **How it connects:** Apps can connect to DB because they share same private network.



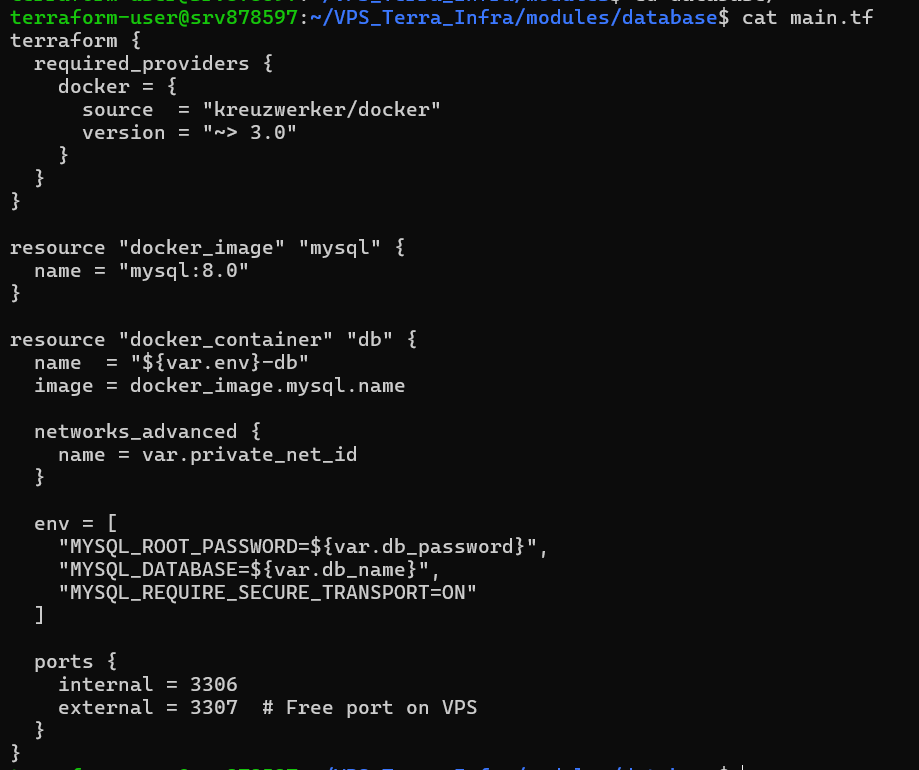
* Deploys Nginx load balancer container.
* networks\_advanced → Connect LB to **both networks** (public & private).
* Public → receive traffic from outside.
* Private → send traffic to apps.
* ports → map VPS 8082 → container port 80.
* So visitors access LB on http://<VPS>:8082.
* volumes → mount your custom nginx.conf into container.
* Replaces default config.
* **How it connects:** LB becomes a **bridge** → public users → private apps.

1. Compute Module



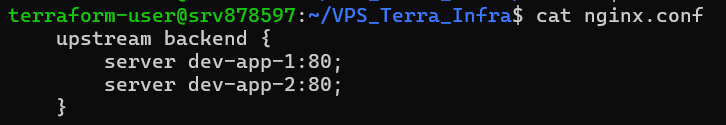
* Deploys multiple app containers.
* count = var.app\_count → if 2, creates 2 containers (dev-app-1, dev-app-2).
* name = "${var.env}-app-${count.index + 1}" → names each container automatically.
* For env=dev: dev-app-1, dev-app-2.
* networks\_advanced { name = var.private\_net\_id } → attach to private network only.
* ports → exposes app on VPS 8081 (but LB mainly uses private IP).
* How it connects:
* LB uses nginx.conf → server dev-app-1:80; server dev-app-2:80;.
* Since both LB and apps are on same private net, LB can resolve dev-app-1, dev-app-2 as DNS names.

1. Database Module



* Sets environment variables for MySQL container.
* MYSQL\_ROOT\_PASSWORD → secures DB with a root password.
* MYSQL\_DATABASE → auto-creates a database.
* MYSQL\_REQUIRE\_SECURE\_TRANSPORT → forces SSL (good for production).
* How it connects:
* DB gets IP from private network.
* Apps can connect using this private IP (172.x.x.x:3306).
* Terraform output shows db\_endpoint.

1. nginx.conf



* upstream backend {}
* Creates a **pool of servers** called backend.
* Inside, you list the app containers (dev-app-1, dev-app-2).
* server dev-app-1:80; / server dev-app-2:80;
* These are **DNS names** automatically created by Docker when containers join the same private network.
* Nginx will round-robin load balance between them.
* Example:
* First request → goes to dev-app-1:80.
* Second request → goes to dev-app-2:80.

A screen shot of a computer

AI-generated content may be incorrect.

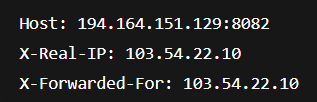
* server {} block
* Defines how Nginx should respond to requests.
* listen 80;
* Inside container → Nginx listens on port 80.
* In Terraform → you mapped VPS port 8082 → container port 80.
* So, users hit http://<VPS>:8082, which routes here.
* location / {}
* Defines rules for all requests (/).
* proxy\_pass <http://backend>;
* Forward all requests to the backend pool (apps).
* proxy\_set\_header Host $host;
* Passes the original Host header from client (e.g., example.com).
* Useful when apps need to know which domain was requested.
* proxy\_set\_header X-Real-IP $remote\_addr;
* Sends the **client’s real IP address** to the app.
* Without this, apps would only see LB’s internal IP.
* proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;
* Standard header chain that tracks the original user IP when traffic passes through proxies.
* Apps (or logs) can check this to know the real client IP.
* How it Works Together

1. User → <http://194.164.151.129:8082/>.
2. Nginx (LB) listens on port 80 inside container.
3. It forwards request to **upstream backend** (dev-app-1:80 or dev-app-2:80).
4. Headers are rewritten so app sees:

* **Correct domain** (Host).
* **Real user IP** (X-Real-IP, X-Forwarded-For).

1. Response comes back → LB → User.

* Example in Action
* User **Alice** at IP 103.54.22.10 visits <http://194.164.151.129:8082/>.
* LB receives request.
* LB chooses dev-app-1:80.
* Forwards request, while sending headers:



* dev-app-1 handles request, maybe queries DB, sends response back.
* Next request might go to dev-app-2.