## Objective

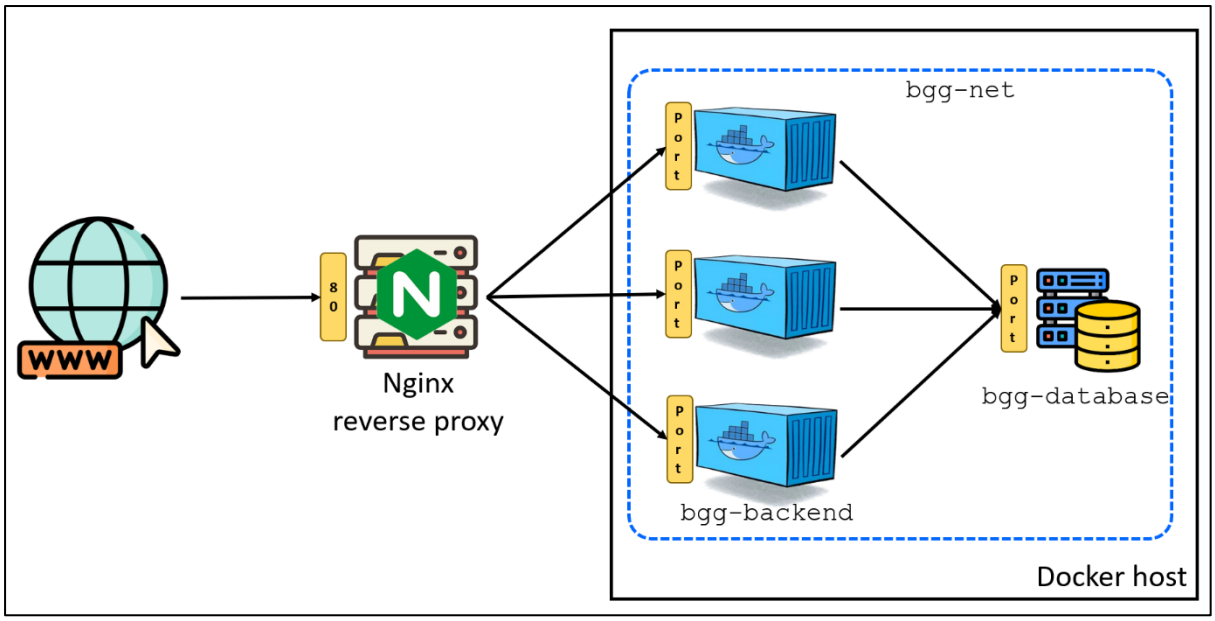
The objective of this workshop is use Terraform's HCL to write scripts to provision Docker containers and a reverse proxy.

## Setup

For this workshop create a directory call `workshop01` in the repository you have create in step a. above. All the files for this workshop should be created in `workshop01` directory.

## Workshop

In this workshop you will automate the provisioning of the following infrastructure shown in the following diagram.




The infrastructure stack consists of

1. Docker network called `bgg-net`
2. Container running MySQL database (`bgg-database`) inside `bgg-net`
3. A specified number of containers running a Nodejs application (`bgg-backend`). These web applications connect to MySQL database. These applications are also provisioned inside `bgg-net`
4. An instance of Nginx running on a separate server which routes traffic to the `bgg-backend` instances.

The following are detail description of provisioning each of the resource in the stack.

### Network (`bgg-net`)

- Create a Docker network called `bgg-net`. This network will be used for all the containers in our application.

### Database (`bgg-database`)

- Provision a Docker volume to be used by the database.
- Use the image `chukmunnlee/bgg-database:v3.1` to create the `bgg-database` container
- Mount the Docker volume that you have created under `/var/lib/mysql`. The database will be created in this volume rather that inside the container
- Expose MySQL port 3306
- The database should be created inside `bgg-net` network

### Application (`bgg-backend`)

- Create 3 instances of the application using the following image: `chukmunnlee/bgg-backend:v3`
- Add the following environment variables
  - `BGG_DB_USER` set to `root`
  - `BGG_DB_PASSWORD` set to `changeit`
  - `BGG_DB_HOST` set to the application database resource name
- The internal port of the application is 3000. Choose a suitable external port to port bind to

### Nginx Reverse Proxy

- Provision a Ubuntu server. Use Ubuntu 20.04 x64
- Add a SSH key to the server so you can SSH into the server
- Install Nginx and enable the service with the following commands
  - `/usr/bin/apt update -y`
  - `/usr/bin/apt upgrade -y`

  - `/usr/bin/apt install nginx -y`
  - `/usr/bin/systemctl start nginx`
  - `/usr/bin/systemctl enable nginx`
- Create a Nginx configuration file called `nginx.conf` with the container endpoints. Use the following template

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;

events {
  worker_connections 768;
}

http: {
  access_log /var/log/nginx/access.log;
  error_log /var/log/nginx/error.log;
  gzip on;
  upstream apps {
    least_conn;
    # the following list the container endpoints
    # one server line for each endpoint
    # eg server <docker_host_ip>:<exposed_port>;
    server docker_host_ip:exposed_port_0;
    server docker_host_ip:exposed_port_1;
    server docker_host_ip:exposed_port_2;
  }
  server {
    listen 80;
    location / {
      proxy_pass http://apps;
    }
  }
}
```

  Hint: this configuration file should be generated from the `bgg-backend` external ports

- Replace the `/etc/nginx/nginx.conf` on the reverse proxy with your `nginx.conf`.
- Signal Nginx to reload the new configuration with the following command
  - `/usr/sbin/nginx -s reload`, or
  - `/usr/bin/systemctl restart nginx`

### Outputs

Your Terraform script should produce the following artefacts and outputs

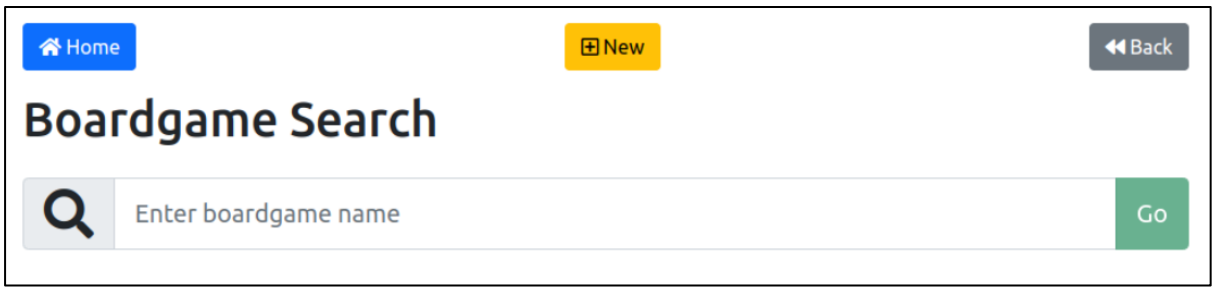
- Reverse proxy IP address
- List of all the container endpoint in the following format

  `<docker_host_ip>:<exposed_port>`

- An empty file call `root@<reverse_proxy_ip >`

## Test

Test your deployment by browsing to `http://<reverse_proxy_ip>`.
You should see the following




## Submission

When you have completed this workshop, commit your work to the repository.

The instructor will clone your repository at the end of the course.