# Setting Up and Testing a Flask Application with NGINX Proxy Configuration

In this guide, we'll walk through how to set up and test an NGINX configuration for two Flask applications: `aidu` and `interaktiv`. The goal is to route requests to these services properly and serve static assets for each.

## Prerequisites

1. **NGINX** installed on the server.
2. Two Flask applications running on different ports locally:
   - `aidu` on port **5000**
   - `interaktiv` on port **5050**
3. Basic knowledge of command-line tools like `curl`.

## NGINX Configuration

The configuration file should look like the one attached. Save it to

```
/etc/nginx/conf.d/default.conf #on AWS cloud or
/etc/nginx/conf.d/reverse-proxy.conf #on fedora
```

(or equivalent path) on your server, then reload or restart NGINX to apply the configuration (e.g., `sudo systemctl reload nginx` on fedora or `sudo nginx -s reload` AWS minimal instances where you miss systemd).

### Configuration Breakdown

### 1. API Routing

- **Aidu Backend**: Accessible at `/aidu/api/`.
- **Interaktiv Backend**: Accessible at `/interaktiv/`.

The configuration routes each backend to the correct Flask application, using NGINX's `proxy_pass` directive.

```
# Proxy API requests to aidu backend
location /aidu/api/ {
    proxy_pass http://localhost:5000/;
    ...
}
```

```
# Proxy requests to interaktiv backend
location /interaktiv/ {
    proxy_pass http://localhost:5050/;
    ...
}
```

### 2. Serving Static Files

- **Aidu Frontend**: Static files served from `/usr/share/nginx/html/aidu/`, accessed via `/aidu/assets/`.
- **Interaktiv Static Files**: Static files for Interaktiv at `/usr/share/nginx/html/interaktiv/static/`, accessed via `/interaktiv/static/`.

For example, images in the `aidu` application can be served from `/aidu/assets/`.

```
# Static files for aidu
location /aidu/assets/ {
    alias /usr/share/nginx/html/aidu/assets/;
    try_files $uri =404;
}
```

**3. Health Check Endpoint** This is a simple health check endpoint used by load balancers like AWS ELB to check if the server is healthy.

```
location /health {
    return 200 "OK\n";
    add_header Content-Type text/plain;
}
```

### Testing with `curl`

Now that we have our configuration set up, let's test each endpoint with `curl`.

**1. Testing Aidu API** To test the Aidu API endpoint, send a request to `http://yourserver/aidu/api/config`.

```
curl -X GET http://yourserver/aidu/api/config
```

You should receive a response from the Aidu Flask application if everything is configured correctly.

**2. Testing Interaktiv API** To test the Interaktiv API, use the following `curl` command:

```
curl -X GET http://yourserver/interaktiv/health
```

This should return a response from the Interaktiv Flask application.

**3. Testing Aidu Static Assets** To check if static assets are being served correctly for `aidu`, try accessing the favicon located in `/usr/share/nginx/html/aidu/assets/`:

```
curl -I http://yourserver/aidu/assets/favicon.ico
```

If the configuration is correct, you should receive a `200 OK` response with the asset's content type.

**4. Testing Interaktiv Static Files** Similarly, test a static file for `interaktiv` by fetching any file you have under `/usr/share/nginx/html/interaktiv/static/`.

```
curl -I http://yourserver/interaktiv/static/favicon.ico
```

This request should return `200 OK` if the asset is available and properly configured.

**5. Testing Health Check** Confirm that the health check endpoint is working:

```
curl -I http://yourserver/health
```

This should return a `200 OK` status with "OK" as the response body.

## Final Notes

- Ensure the Flask applications are running on the specified ports (5000 for `aidu` and 5050 for `interaktiv`).
- You can use the `netstat` command to see which services are listening on which ports. This command lists network connections, and you can filter it to check if ports 5000 and 5050 are active.

  ```
  sudo netstat -tuln | grep ':5000\|:5050'
  ```

- Use `nginx -t` to test your NGINX configuration before reloading, ensuring there are no syntax errors.
- Reload NGINX to apply changes:

  ```
  sudo systemctl reload nginx
  ```

This setup and testing procedure should help even inexperienced developers confirm that the server is properly configured and routing requests as expected.