APISIX Readonly Proxy

Prerequisites

Before you start following your guide you need prepare this list in advance

- Virtual Machine (VM) with **Ubuntu** Installed
- SSH access and **Public IP** of that machine
- Assigned Domain Name on that Public Ip e.g. - **client-exante-proxy.company.com**
- Ports **22, 80, 443** should be open

Bearer Token to be provided (e.g. in case of Exante)

Generate Auth Token

Open a terminal window.

1. Type this command, replacing `YOUR_API_KEY` and `YOUR_SECRET_KEY` with your real keys:

   ```
   echo -n "YOUR_API_KEY:YOUR_SECRET_KEY" | base64
   ```

2. Press Enter.
3. The terminal will show a long string of letters and numbers. This is your encoded token.
4. Copy that string.
5. When you call your API, add this header (showing your copied string in place of `ENCODED_STRING`):

   ```
   Authorization: Basic ENCODED_STRING
   ```

6. If your API really needs a Bearer header instead, use:

   ```
   Authorization: Bearer ENCODED_STRING
   ```

That’s it. Now your API call will send the correct token.

Installing Docker

1. **Open Terminal**
   - Find the “Terminal” app and click it.
2. **Run the Easy Install Command**
   - In Terminal, type or paste this whole line and press **Enter**:

     ```
     wget -qO- https://get.docker.com/ | sh
     ```

   - This downloads a small script and runs it. It will install Docker for you.
3. **Wait for It to Finish**
   - The script will show text as it downloads and sets up Docker.
   - When the text stops, Docker is installed.
4. **Check Docker Works**
   - Still in Terminal, type:

     ```
     sudo docker run hello-world
     ```

   - Press **Enter**. Docker will download a tiny test program and run it.
   - If you see “Hello from Docker!”, it means Docker is ready.
5. **Let You Use Docker Without “sudo” (Optional)**
   - By default, you need to type `sudo` before every Docker command. If you want to skip typing `sudo`, do this:
     1. Type:

        ```
        sudo usermod -aG docker $USER
        ```

        Press **Enter**.
     2. Close Terminal and log out of your Ubuntu account. Then log back in.
     3. Now you can run Docker commands without `sudo`. For example:

        ```
        docker run hello-world
        ```

That’s it! Docker is now installed on your Ubuntu 22.04 in just one simple step.

Installing APISIX

1. Open a terminal on your VM.
2. Make a folder for APISIX:

```
sudo mkdir -p /opt/apisix
```

3. Create the file `/opt/apisix/apisix.yaml`:

```
sudo nano /opt/apisix/apisix.yaml
```

4. Copy and paste exactly this content into `apisix.yaml`. Replace `YOUR_BASE64_DEMO_TOKEN` and `YOUR_BASE64_LIVE_TOKEN` with your real Base64(API_KEY:SECRET_KEY) strings:

```
apisix:
  node_listen: 9080  # APISIX will listen on port 9080 for HTTP

consumers:
  - username: finmars  # a name for this client
    plugins:
      basic-auth:
        username: foo  # user name you want to use for APISIX basic auth
        password: bar  # password you want to use for APISIX basic auth

upstreams:
  - id: 1
    nodes:
      api-demo.exante.eu:443: 1
    scheme: https
    pass_host: node
    type: roundrobin

  - id: 2
    nodes:
      api-live.exante.eu:443: 1
    scheme: https
    pass_host: node
    type: roundrobin

plugin_configs:
```

```
  - id: 1
    plugins:
      basic-auth: {}
      proxy-rewrite:
        headers:
          set:
            # Replace this with your Base64(API_KEY:SECRET_KEY) for the demo
environment
            Authorization: "Basic %TOKEN%"
        regex_uri:
          - "^/demo/(.*)"
          - "/$1"

  - id: 2
    plugins:
      basic-auth: {}
      proxy-rewrite:
        headers:
          set:
            # Replace this with your Base64(API_KEY:SECRET_KEY) for the live
environment
            Authorization: "Basic %TOKEN%"
        regex_uri:
          - "^/live/(.*)"
          - "/$1"

routes:
  - uris:
      - /demo/md/*/accounts
      - /demo/md/*/symbols/*
      - /demo/md/*/summary/*
      - /demo/md/*/ohlc/*
      - /demo/md/*/transactions
      - /demo/md/*/crossrates/*
      - /demo/trade/*/orders/*
    upstream_id: 1
    plugin_config_id: 1

  - uris:
```

```
      - /live/md/*/accounts
      - /live/md/*/symbols/*
      - /live/md/*/summary/*
      - /live/md/*/ohlc/*
      - /live/md/*/transactions
      - /live/md/*/crossrates/*
      - /live/trade/*/orders/*
    upstream_id: 2
    plugin_config_id: 2

deployment:
  role: data_plane
  role_data_plane:
    config_provider: yaml



#END
```

5. Save and close the file:
   - Press `Ctrl+O`, then `Enter` to save.
   - Press `Ctrl+X` to exit Nano.

---

2. Create a restart script for APISIX

1. In the terminal, make a new script:

```
nano /opt/apisix/restart_apisix.sh
```

2. Copy and paste this into `restart_apisix.sh`:

```
#!/bin/bash

# If a container named "apache-apisix" exists, stop and remove it
if docker ps -a --format '{{.Names}}' | grep -Eq "^apache-apisix\$"; then
  echo "Stopping and removing existing apache-apisix container..."
  docker stop apache-apisix
  docker rm apache-apisix
fi
```

```
echo "Starting a new APISIX container..."
docker run -d \
  --name apache-apisix \
  -p 9080:9080 \
  -e APISIX_STAND_ALONE=true \
  -v /opt/apisix/apisix.yaml:/usr/local/apisix/conf/config.yaml \
  -v /opt/apisix/apisix.yaml:/usr/local/apisix/conf/apisix.yaml \
  apache/apisix

echo "APISIX container is now running."
```

3. Save and close:
   - Press `Ctrl+O`, then `Enter`.
   - Press `Ctrl+X`.
4. Make the script executable:

```
chmod +x /opt/apisix/restart_apisix.sh
```

---

3. Run APISIX for the first time

1. In the terminal, run:

```
./restart_apisix.sh
```

2. Wait a few seconds. APISIX will start in Docker, listening on port 9080, reading your `/opt/apisix/apisix.yaml` as both `config.yaml` and `apisix.yaml`.
3. Check that it is running:

```
sudo docker ps
```

   You should see a line for `apache/apisix` with `0.0.0.0:9080->9080/tcp`.
4. Look at the logs to ensure no errors:

```
sudo docker logs apache-apisix
```

   You should see messages like “config file … reloaded” and no errors.

---

4. Test APISIX locally with curl (no Nginx yet)

1. In the terminal, run:

   ```
   curl -u foo:bar http://127.0.0.1:9080/demo/md/3.0/accounts
   ```

   - `-u foo:bar` sends your Basic Auth.
   - The path `/demo/md/3.0/accounts` matches your route pattern `/demo/md/*/accounts`.
2. If everything is correct, you will see JSON returned from Exante.
   If you see `{"error_msg":"404 Route Not Found"}`, double-check that the path matches exactly and that your tokens are correct.

Install Nginx Proxy

5. Install and configure Nginx and Let’s Encrypt

5.1. Install Nginx

1. Update package lists:

```
sudo apt update
```

2. Install Nginx:

```
sudo apt install nginx -y
```

3. Start and enable Nginx:

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

5.2. Install Certbot for Let’s Encrypt

1. Add repositories and update:

```
sudo apt install software-properties-common -y
sudo add-apt-repository universe
sudo add-apt-repository ppa:certbot/certbot -y
sudo apt update
```

2. Install Certbot with Nginx plugin:

```
sudo apt install certbot python3-certbot-nginx -y
```

5.3. Obtain a certificate for your domain

1. Make sure your DNS A record for `abeta-proxy.finmars.com` points to your VM `PUBLIC_IP`.
2. Run:

```
sudo certbot --nginx -d abeta-proxy.finmars.com
```

3. Follow the prompts:
   1. Enter your email, then press `Enter`.
   2. Agree to terms by typing `A`, then `Enter`.
   3. Choose option `2` to redirect HTTP to HTTPS, then `Enter`.

Certbot will create an Nginx site file and install the certificate under `/etc/letsencrypt/live/abeta-proxy.finmars.com/`.

---

6. Configure Nginx to proxy to APISIX

1. Open the site file Certbot created:

```
sudo nano /etc/nginx/sites-available/default
```

2. Inside the `server { ... }` block for port 443, find these lines:

```
listen 443 ssl;
server_name abeta-proxy.finmars.com;

ssl_certificate /etc/letsencrypt/live/abeta-proxy.finmars.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/abeta-proxy.finmars.com/privkey.pem;
```

3. Right below them, add:

```
location / {
    proxy_pass http://127.0.0.1:9080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
```

   After editing, that `server { }` block looks like:

```
server {
    listen 443 ssl;
    server_name abeta-proxy.finmars.com;

    ssl_certificate /etc/letsencrypt/live/abeta-proxy.finmars.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/abeta-proxy.finmars.com/privkey.pem;
```

```
    location / {
        proxy_pass http://127.0.0.1:9080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

4. Make sure there is also a block that redirects HTTP to HTTPS. It looks like:

```
server {
    listen 80;
    server_name abeta-proxy.finmars.com;
    return 301 https://$host$request_uri;
}
```

5. Save and close:
   - Press `Ctrl+O`, then `Enter`.
   - Press `Ctrl+X`.
6. Test Nginx configuration:

```
sudo nginx -t
```

   You should see “syntax is ok” and “test is successful”.
7. Reload Nginx so it uses the new config:

```
sudo systemctl reload nginx
```

---

7. Open firewall ports (if you use UFW)

1. Allow HTTP (port 80) and HTTPS (port 443), and APISIX port (9080) in UFW:

```
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw allow 9080/tcp
```

2. Check UFW status:

```
sudo ufw status
```

Final Tests

8. Final tests

1. **Test HTTPS through Nginx:**
   In a browser or terminal, go to:

   ```
   https://abeta-proxy.finmars.com/demo/md/3.0/accounts
   ```

   - Enter Basic Auth user `foo` and password `bar`.
   - If your Base64 tokens are correct, you see JSON from Exante.
2. **Test other routes:**

   ```
   https://abeta-proxy.finmars.com/live/md/3.0/accounts
   ```

   - Use the same Basic Auth.
   - Should return JSON if live token is correct.
3. **Test local APISIX again (no Nginx):**

   ```
   curl -u foo:bar http://127.0.0.1:9080/demo/md/3.0/accounts
   ```

   - This hits APISIX directly, without Nginx.
   - Should return JSON if config is correct.

---

9. Automatic certificate renewal

1. Certbot already set up automatic renewal.
2. To test renewal, run:

   ```
   sudo certbot renew --dry-run
   ```

3. If it says “Congratulations, all renewals succeeded,” your auto-renew is working.

---

10. How to update your APISIX config later

1. Edit `/opt/apisix/apisix.yaml` any time:

   ```
   sudo nano /opt/apisix/apisix.yaml
   ```

2. Save changes and exit.
3. Run the restart script:

   ```
   ./restart_apisix.sh
   ```

4. Check logs:

   ```
   sudo docker logs apache-apisix
   ```

5. Test again with `curl` or in a browser.

Complete Recap

Complete Recap

1. **Create folder** `/opt/apisix`.
2. **Create and fill** `/opt/apisix/apisix.yaml` (with `role: data_plane`, consumers, upstreams, plugin_configs, routes, and `#END`).
3. **Make** `restart_apisix.sh` script that stops any old container and starts a new one, mounting `/opt/apisix/apisix.yaml` as both `config.yaml` and `apisix.yaml`.
4. **Run** `./restart_apisix.sh` to start APISIX.
5. **Test** APISIX locally: `curl -u foo:bar http://127.0.0.1:9080/demo/md/3.0/accounts`.
6. **Install Nginx** (`sudo apt install nginx`).
7. **Install Certbot** (`sudo apt install certbot python3-certbot-nginx`).
8. **Get SSL**: `sudo certbot --nginx -d abeta-proxy.finmars.com`.
9. **Edit Nginx site** at `/etc/nginx/sites-available/default` to add:

```
location / {
    proxy_pass http://127.0.0.1:9080;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
```

10. **Reload Nginx** (`sudo nginx -t` then `sudo systemctl reload nginx`).
11. **Open firewall** ports 80, 443, 9080 (`sudo ufw allow ...`).
12. **Test** `https://abeta-proxy.finmars.com/demo/md/3.0/accounts` in a browser.
13. **Auto-renew** is handled by Certbot.
14. **To update**, edit `/opt/apisix/apisix.yaml` and run `./restart_apisix.sh`.

That is the full, clear set of instructions. Now your APISIX runs behind Nginx with a Let’s Encrypt SSL certificate, and you can update the config anytime by editing the file and restarting with the script.