Deploy Onyx

Deploy on GCP

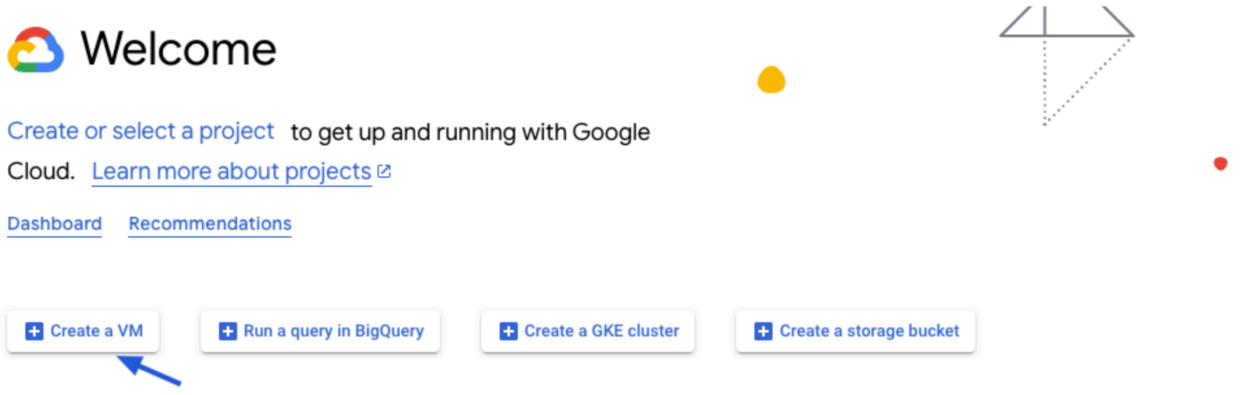
Setup Onyx on GCP

You can use GCP Google Compute Engines (VM Instance) to deploy Onyx. The steps are very similar to deploying on AWS EC2. Before we get started, make sure you have an account with Google Cloud Platform and have the necessary permissions to create a VM instance. Feel free to reach out to us via the links on our **Contact Us** page for more individual help.

Getting an Instance

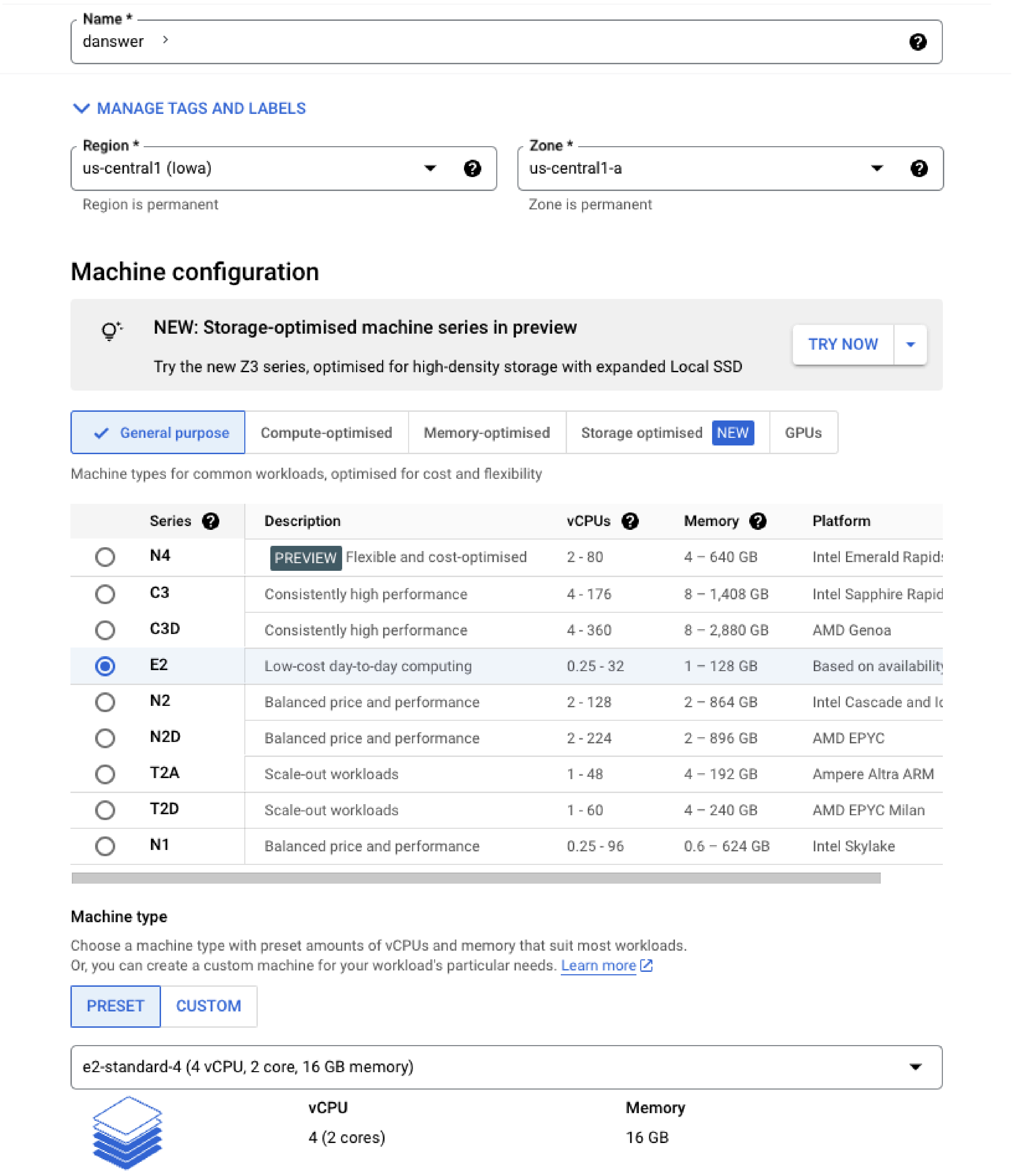
Steps to create a VM instance from Google Cloud Console:

1. Go to the **Google Cloud Console Dashboard** to create a VM Instance



2. Select an existing project or create a new one under your organization.
3. We recommend at least 16GB of RAM, 4-8vCPU cores, and 500GB of disk. The exact instance type to choose depends on your document volume and query load, but a `e2-standard-4` / `e2-standard-8` is a good starting point. For more details on GCP instances, you can refer to the **GCP**

documentation. For more information on sizing, refer to our resourcing guide.



Name * danswer

MANAGE TAGS AND LABELS

Region * us-central1 (Iowa)
Region is permanent

Zone * us-central1-a
Zone is permanent

Machine configuration

NEW: Storage-optimised machine series in preview
Try the new Z3 series, optimised for high-density storage with expanded Local SSD
TRY NOW

General purpose | Compute-optimised | Memory-optimised | Storage optimised NEW | GPUs

Machine types for common workloads, optimised for cost and flexibility

	Series	Description	vCPUs	Memory	Platform
○	N4	PREVIEW Flexible and cost-optimised	2 - 80	4 – 640 GB	Intel Emerald Rapid
○	C3	Consistently high performance	4 - 176	8 – 1,408 GB	Intel Sapphire Rapid
○	C3D	Consistently high performance	4 - 360	8 – 2,880 GB	AMD Genoa
●	E2	Low-cost day-to-day computing	0.25 - 32	1 – 128 GB	Based on availabilit
○	N2	Balanced price and performance	2 - 128	2 – 864 GB	Intel Cascade and I
○	N2D	Balanced price and performance	2 - 224	2 – 896 GB	AMD EPYC
○	T2A	Scale-out workloads	1 - 48	4 – 192 GB	Ampere Altra ARM
○	T2D	Scale-out workloads	1 - 60	4 – 240 GB	AMD EPYC Milan
○	N1	Balanced price and performance	0.25 - 96	0.6 – 624 GB	Intel Skylake

Machine type

Choose a machine type with preset amounts of vCPUs and memory that suit most workloads.
Or, you can create a custom machine for your workload's particular needs. Learn more

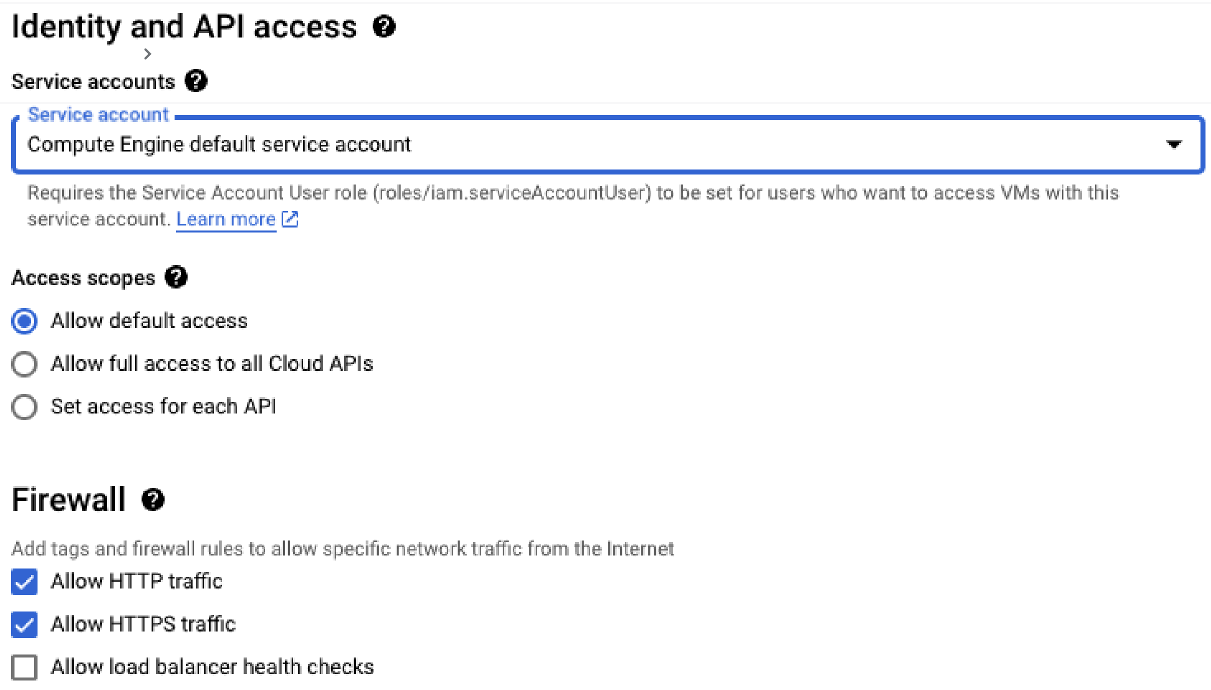
[✓ PRESET] CUSTOM

e2-standard-4 (4 vCPU, 2 core, 16 GB memory)

vCPU
4 (2 cores)

Memory
16 GB

4. Make sure to allow HTTPS traffic in the firewall settings and valid scopes for the instance.


Identity and API access

Service accounts

Service account: Compute Engine default service account

Requires the Service Account User role (roles/iam.serviceAccountUser) to be set for users who want to access VMs with this service account. Learn more

Access scopes

- [✓] Allow default access
- [ ] Allow full access to all Cloud APIs
- [ ] Set access for each API

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

- [✓] Allow HTTP traffic
- [✓] Allow HTTPS traffic
- [ ] Allow load balancer health checks

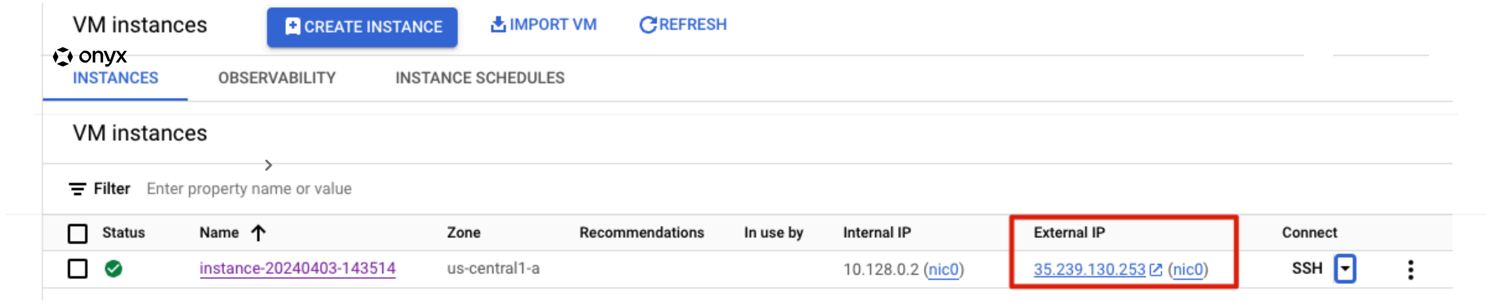
For the below guide, we will assume that you've chosen to use GCP Debian GNU/Linux machine with the recommended `e2-standard-4` instance. For more details, you can follow the steps in the **GCP documentation** to create a **VM instance**.

Pointing your Domain to the Instance

Next, we should point your domain to the VM instance we just created. To do this, we need to go to your DNS and add two records. For this guide, I'll be assuming your DNS provider is GoDaddy, but it should be almost exactly the same for any DNS provider.

If you don't have a domain to use yet, then you can either buy one from a DNS provider like **GoDaddy** or just skip HTTPS for now.

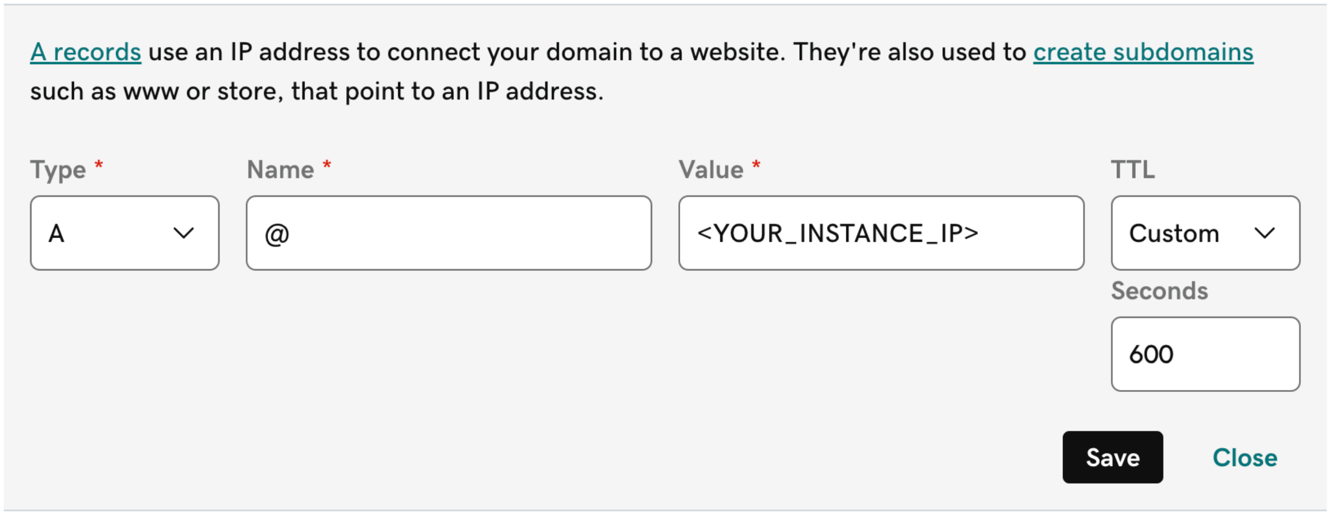
First, we need to grab the External IP address of the instance. You can get that from the VM Instance list page on GCP Console.


VM instances

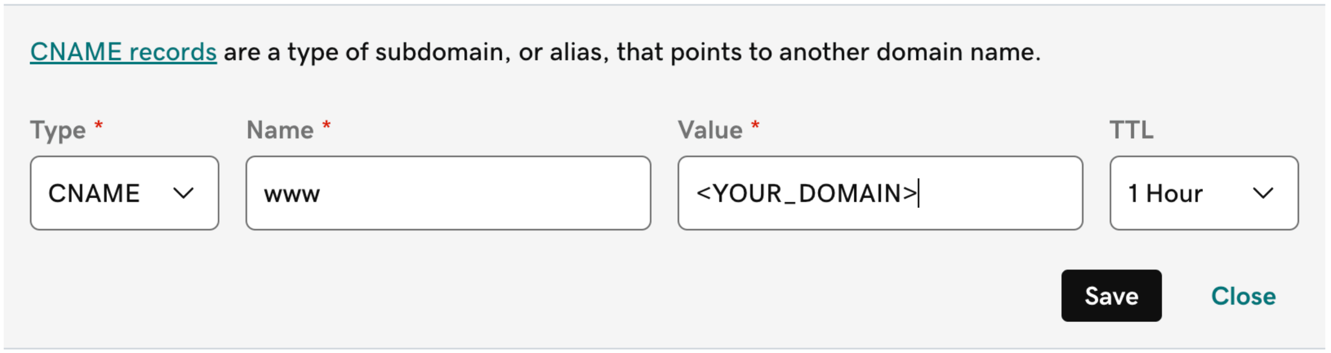
VM instances

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	instance-20240403-143514	us-central1-a			10.128.0.2 (nic0)	35.239.130.253 (nic0)	SSH

Finally, we need to head to the DNS provider and add two entries into the DNS:


A records use an IP address to connect your domain to a website. They're also used to create subdomains such as www or store, that point to an IP address.

Type *	Name *	Value *	TTL
A	@	<YOUR_INSTANCE_IP>	Custom
			Seconds: 600


CNAME records are a type of subdomain, or alias, that points to another domain name.

Type *	Name *	Value *	TTL
CNAME	www	<YOUR_DOMAIN>	1 Hour

The first record directs traffic to that domain to your GCP VM instance. The second record will handle `www.<YOUR_DOMAIN>` and ensure that this also takes the user to your VM instance.

Installing Dependencies

Next, we need to prepare the instance so we can actually get Onyx up and running. To do this, you'll need three things: `git` , `docker` , and `docker compose` . For Debian Linux 12, this can be done with the following:

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/key
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https:/

sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compos
```

If using CentOS, Red Hat Linux, or similar, you can use the following:

```
sudo yum update -y

sudo yum install docker -y
sudo service docker start

sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(una
sudo chmod +x /usr/local/bin/docker-compose

sudo yum install git
```

Starting up Onyx

Now that we have everything we need we can startup Onyx.

First, let’s clone the repo:

```
git clone https://github.com/onyx-dot-app/onyx.git
```

Next, let’s set the necessary env variables:

```
cd onyx/deployment/docker_compose
touch .env
touch .env.nginx
```

In the `.env` file, you can copy past the following (filling in the missing fields as needed):

```
WEB_DOMAIN=<YOUR_DOMAIN>  # something like "onyx.app"

# if your email is something like "chris@onyx.app", then this should be "onyx.app"
# this prevents people outside your company from creating an account
VALID_EMAIL_DOMAINS=<YOUR_COMPANIES_EMAIL_DOMAIN>

AUTH_TYPE=basic
# if you want to enable email verification, uncomment the following
# REQUIRE_EMAIL_VERIFICATION=true
# SMTP_USER=<GMAIL_ACCOUNT_EMAIL_YOU_WANT_TO_SEND_VERIFICATION_EMAILS_WITH>
# SMTP_PASS=<GMAIL_ACCOUNT_PW_YOU_WANT_TO_SEND_VERIFICATION_EMAILS_WITH>

# if you've gone through the Google OAuth setup guide, then comment out
# the above and uncomment the following
# AUTH_TYPE=google_oauth
# GOOGLE_OAUTH_CLIENT_ID=
# GOOGLE_OAUTH_CLIENT_SECRET=
# SECRET=<RANDOMLY_GENERATED_UUID>

# Default values here are what Postgres uses by default, feel free to change.
POSTGRES_USER=postgres
POSTGRES_PASSWORD=password
```

In the `.env.nginx` file, put the following:

```
DOMAIN=<YOUR_DOMAIN>  # something like "onyx.app"
```

Next, let's get our SSL certificate from **letsencrypt**. To do this, we can simply run:

```
sudo ./init-letsencrypt.sh
```

If are skipping the HTTPS setup, you should start things up with: `sudo docker compose -f docker-compose.dev.yml -p onyx-stack up -d --pull always` instead of the above. You can then access Onyx from the IP address from earlier or from the instance `External IP` provided on the instance's page in the GCP VM console.

Voila, you're all done! 🎉

After waiting a few minutes (you can monitor the progress with `sudo docker logs onyx-stack-api_server-1 -f`; once you see a log for `INFO: Application startup complete.` then everything should be good to go).

‹ ECS

Deploy on Azure ›