Week 13 - Guide to Deploying your Web Server

For codeworks alumni, but could otherwise be adopted to other cases

(WIP)

Technology Overview

	[Type]	Purpose	Product	[Website]
	Code Version Control:		**GIT**	[https://git-scm.com]
[SaaS]	Code Repository:		**GITHUB**	[https://github.com]
[PaaS]	Container:		**DOCKER**	[https://docker.com]
	Container Repository:	**DOCKERHUB**	[https://hub.docker.com]	
[IaaS]	Server “HW”:		**AWS EC2**	[https://console.aws.amazon.com]
	DB Language:		**MySQL**	[https://www.mysql.com]
	[Web Server/Reverse Proxy]:	**NGINX**	[https://www.nginx.com]	
	Domain Registrar:		**CLOUDFLARE**	[https://cloudflare.com]

Stage 0 – Assumptions:

- Using AWS’s Free Tier/Trial opportunities for new & existing accounts
- Utilizing the latest version of `bcw` (`3.4.1` as of this writing)
  - Run `npm i -g bcw` to globally install the latest version
- Utilizing a vue template created after `12/20/2023` (~~`vue-starter`~~, `express-vue`, `dotnet-vue`)
  - (*this guide is only for full-stack applications at this moment*)
- Utilizing the latest versions of the BCW templates

NOTES

- *Commands with surrounding* **`< >`** *indicate a variable where you supply the value.*

*Do not include the **< >** characters.*

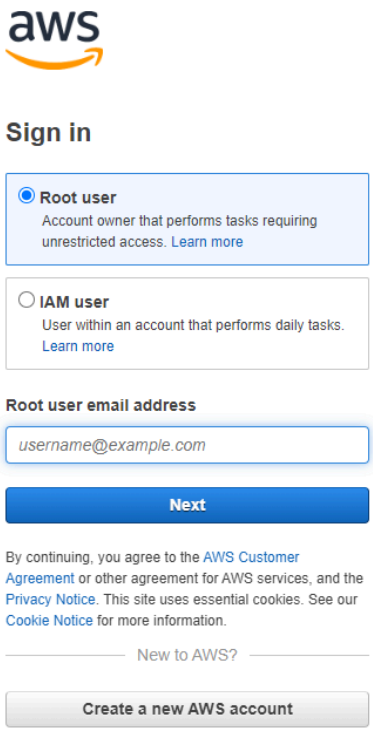
- *Entries **highlighted** with this color have additional notes*
  ***Click** on the **highlighted** area to draw in and focus the note.*

- *Other entries that are highlighted or colored differently are to help identify where something was mentioned and that might be reused later.*

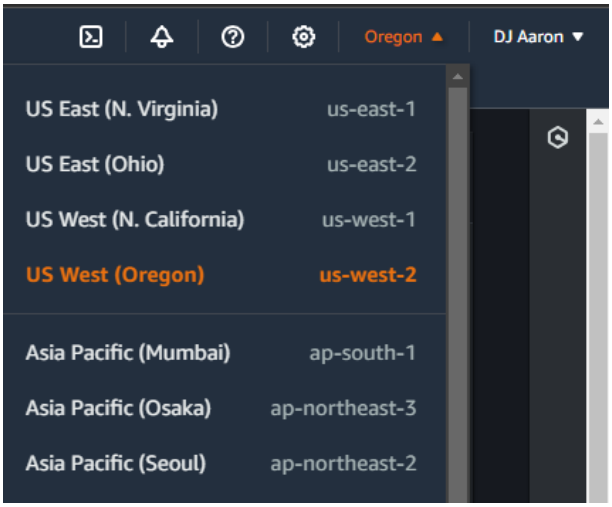
<u>Stage 1</u> – AWS EC2 – CREATE A SERVER INSTANCE

(From a AWS's website - https://console.aws.amazon.com)

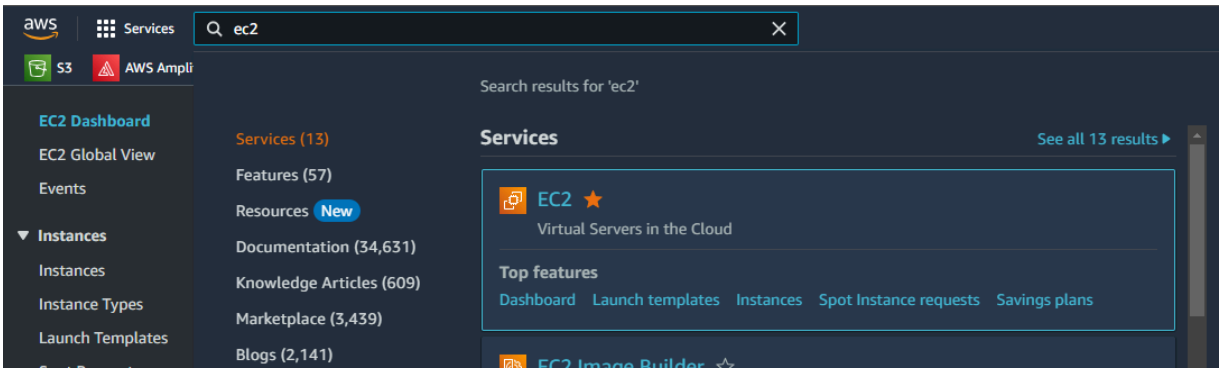
1. **Create a new AWS account** (*below – New to AWS? –*) and add a Credit Card to the account (*required for incidentals, but free tier things are available for the first year*)



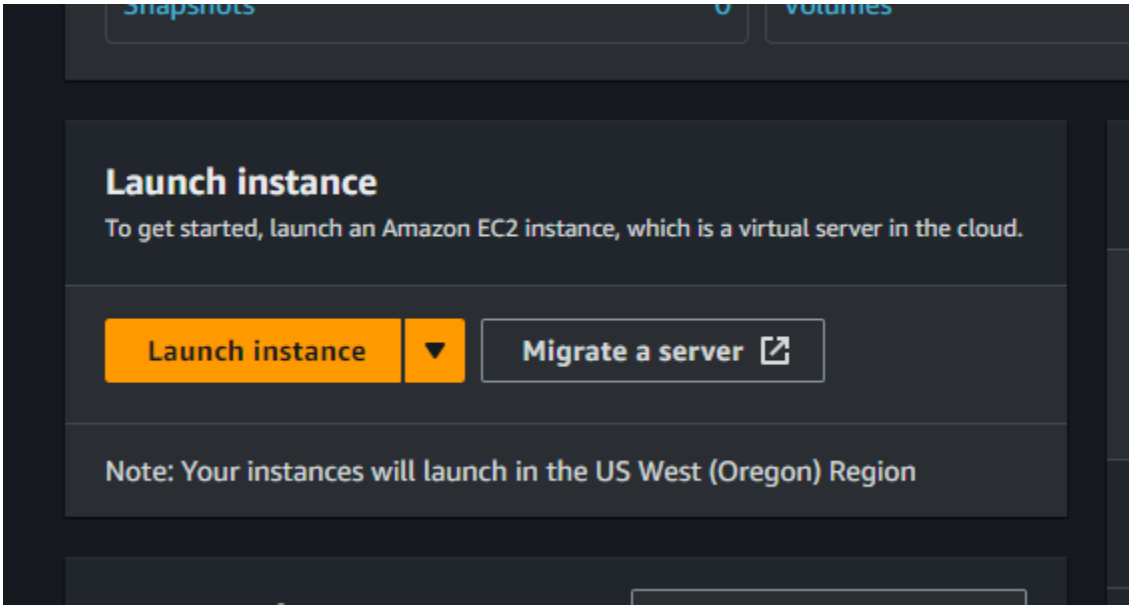
2. Select a region in the top-right – eg. **US West (Oregon)**



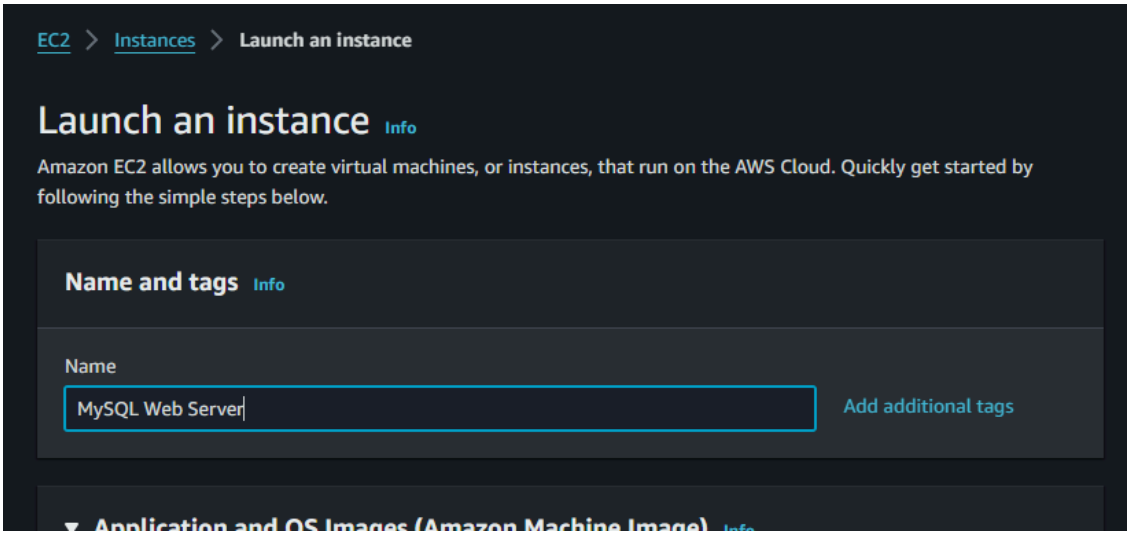
3. Navigate to AWS's EC2 Service panel to get started



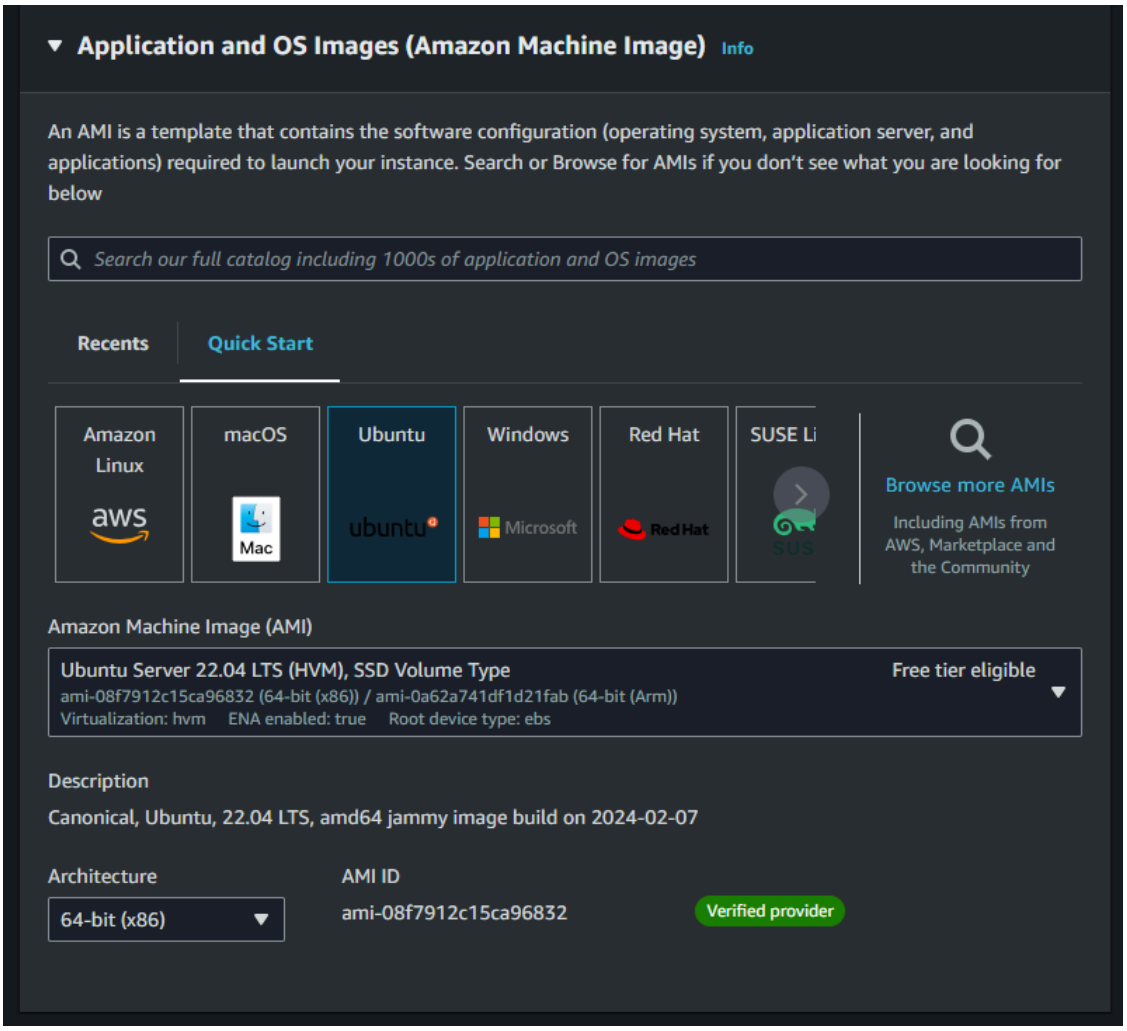
a. Click on `**Launch Instance**`



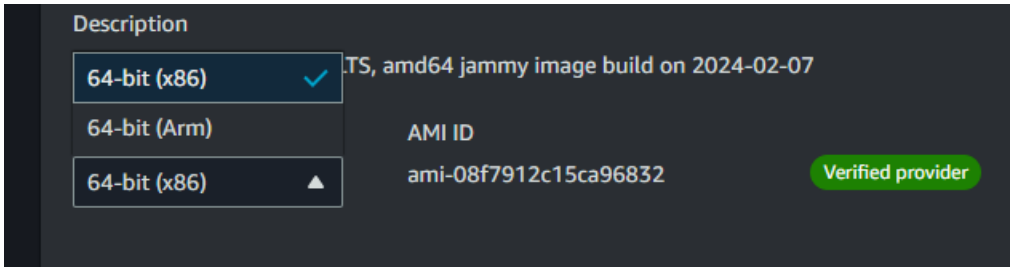
b. (***Give it a name***)



c. Select `**Ubuntu**` for the 'Application and OS Images' option.

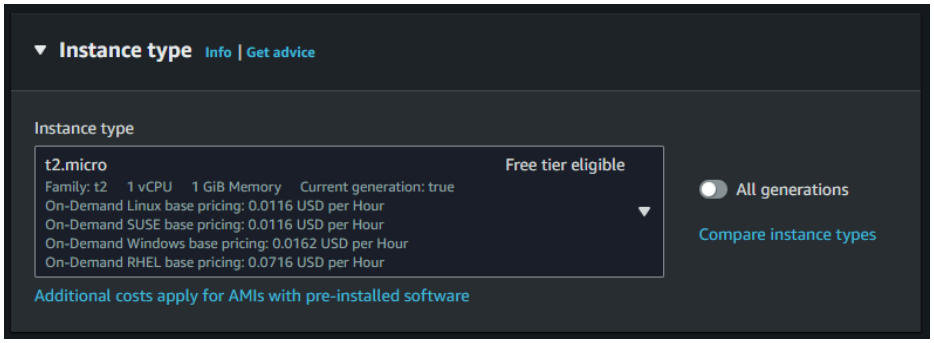


d. [***conditional***] Consider the preferred architecture to use (64-bit : **x86** vs **ARM**) ****NEW**** *****Special*****
Either of the options are fine and neither adversely affect the setup process

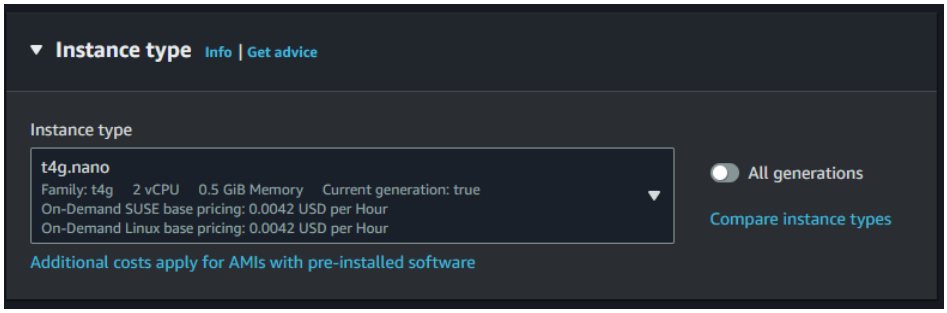


- 64-bit(**x86**) w/ Free Tier = `**t2.micro**` @ 1 vCPU 1GB RAM (1 Yr @ new Account)
  OR
- 64-bit(**Arm**) w/ Free Trial = `**t4g.small**` @ 2 vCPU 2GB RAM (EoY 2024)

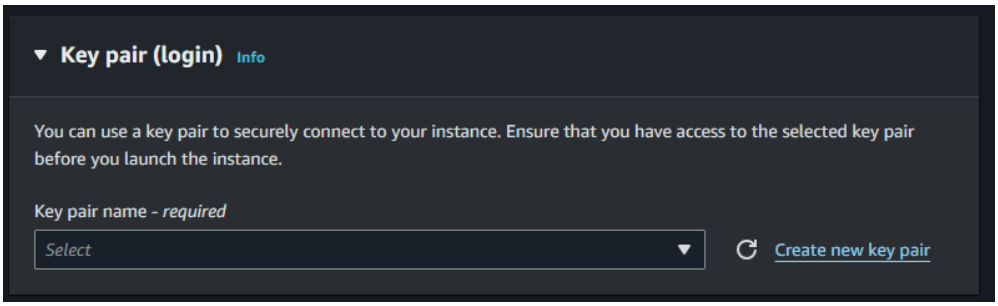
Select the `**t2.micro**` | ***OR*** | `**t4g.small**` under '**Architecture**' depending on your choice above



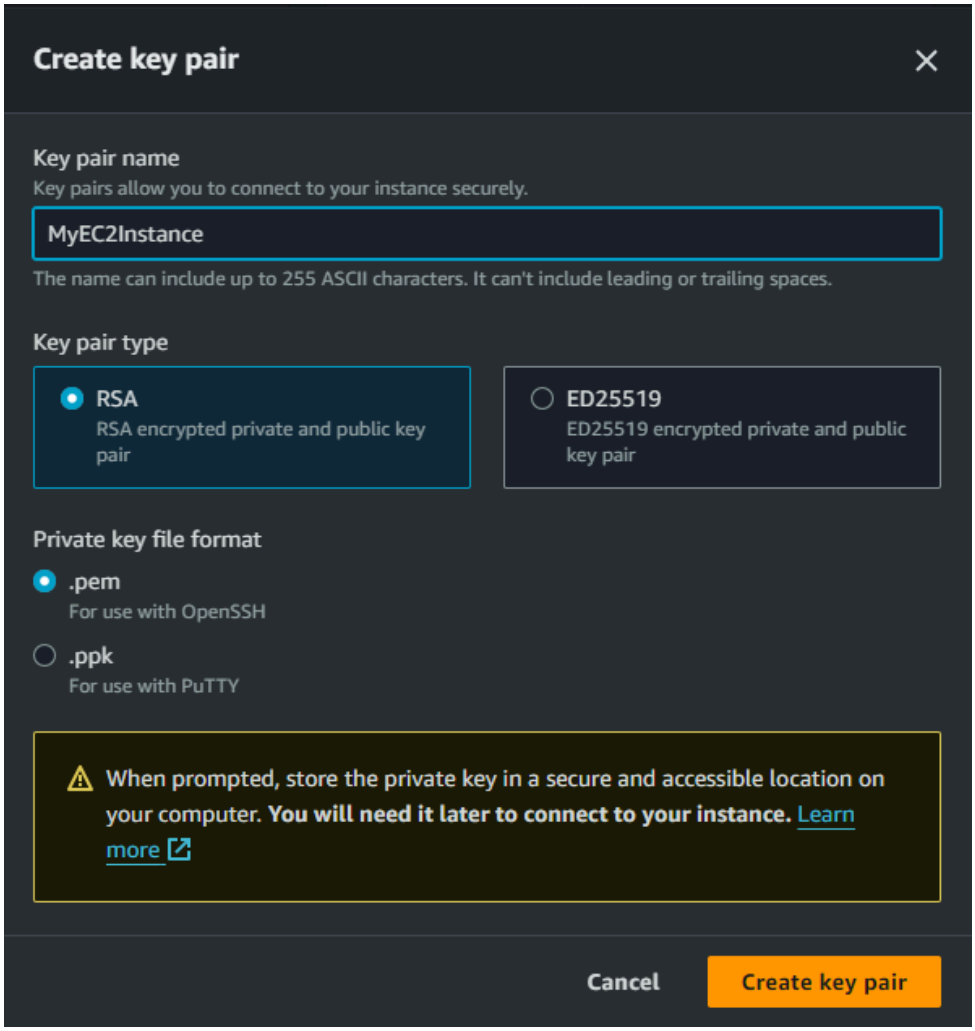
OR



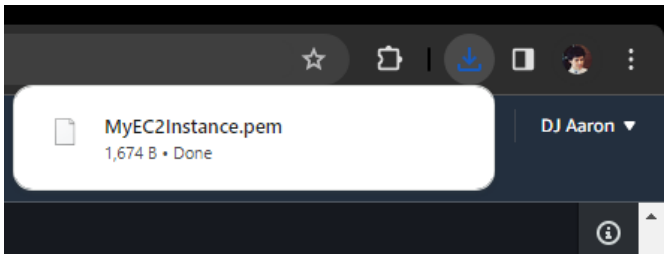
e. '**Create new key pair**' to use as an authentication key *(at your terminal + GitHub secrets)*



- Give it a name, keep the defaults (RSA & .pem), then hit Create
  The file will auto download - move this to a better location outside of your Downloads folder
  ** This will be your **EC2_PEM_KEY** and how you remotely log in to manage your server **

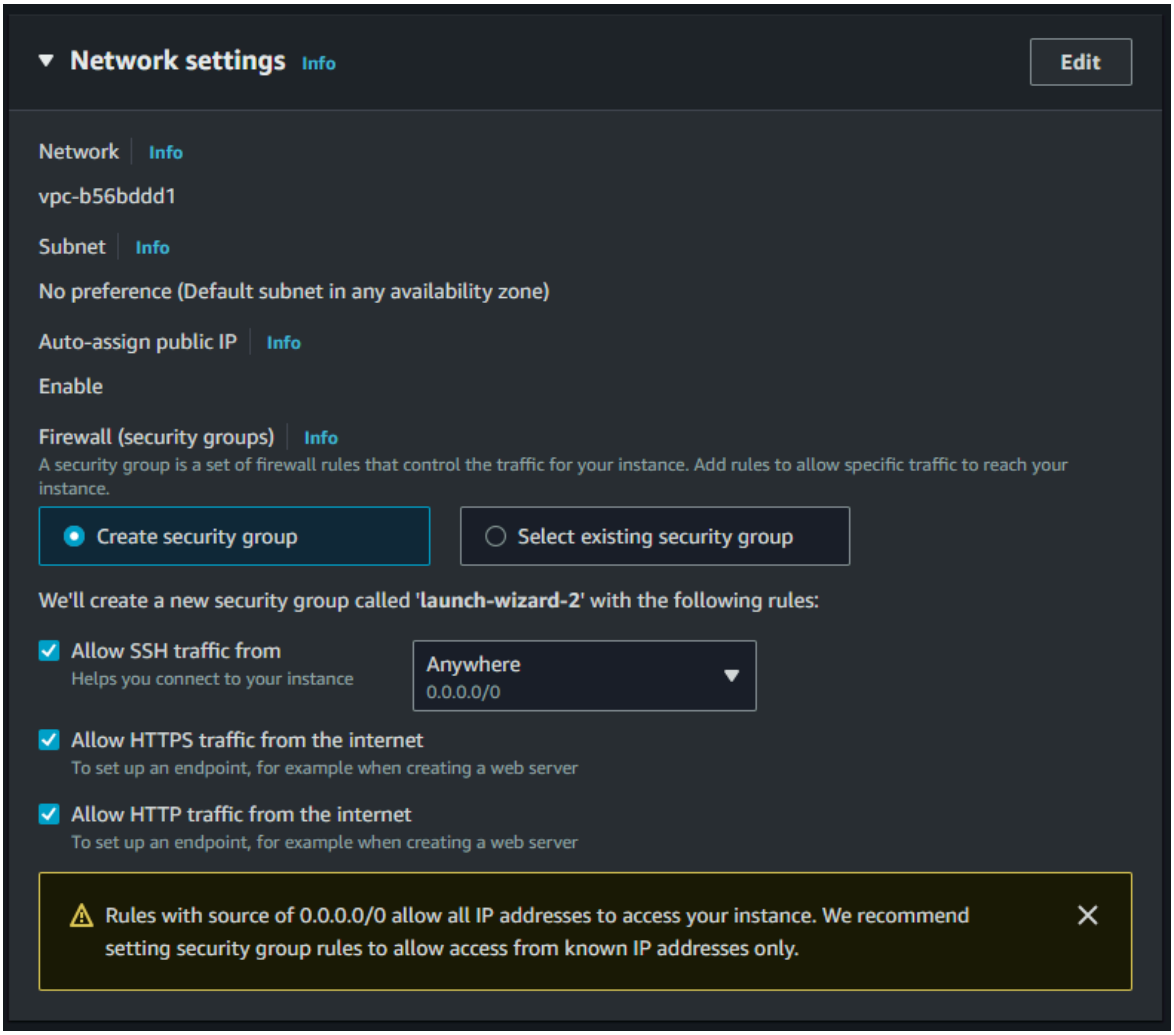


(*KeyPair file downloads to your system)*

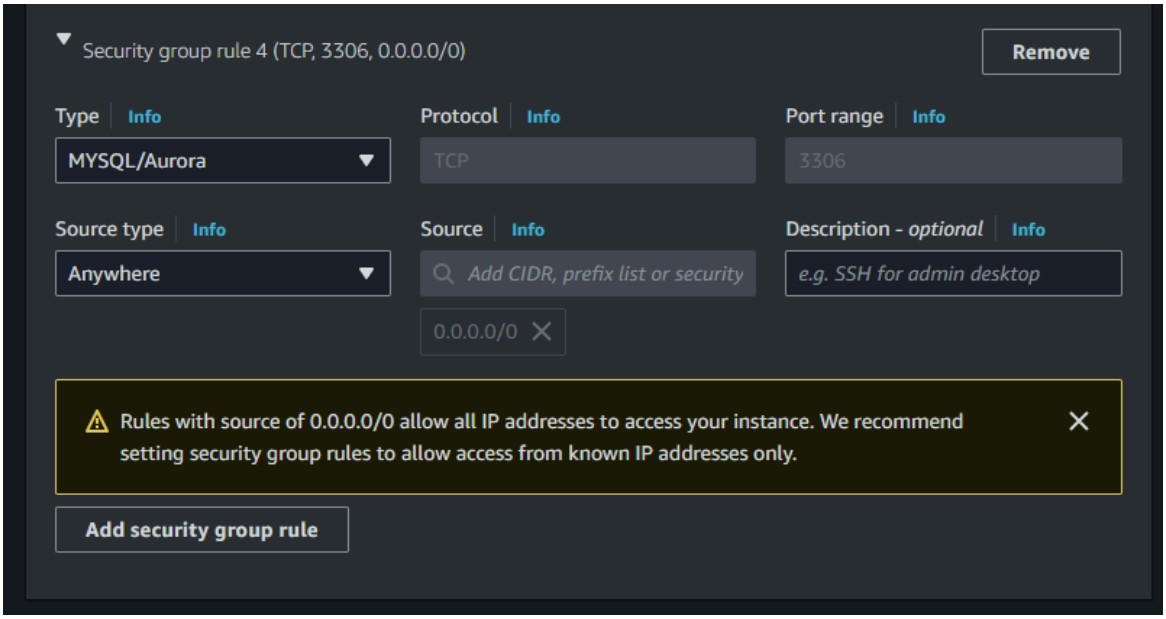


f. Under '**Network Settings**', you'll be creating a security group.

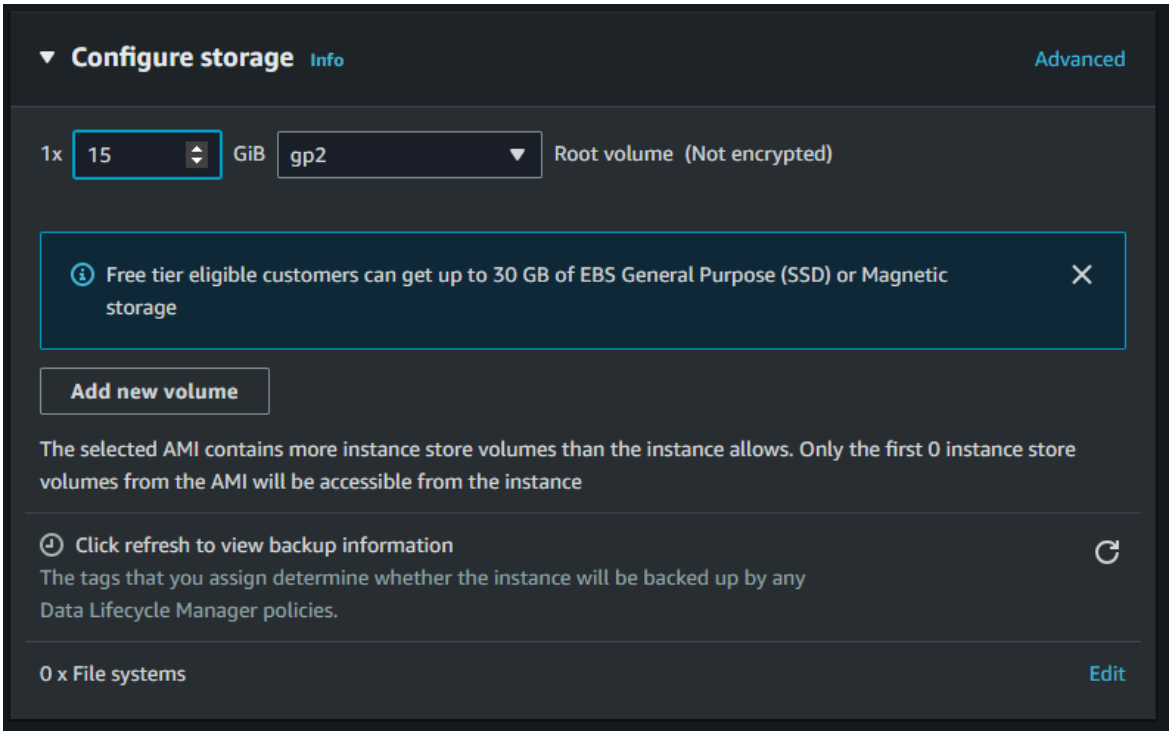
- Check all boxes:
  - Allow SSH traffic from Anywhere
  - Allow HTTPS traffic from the internet
  - Allow HTTP traffic from the internet



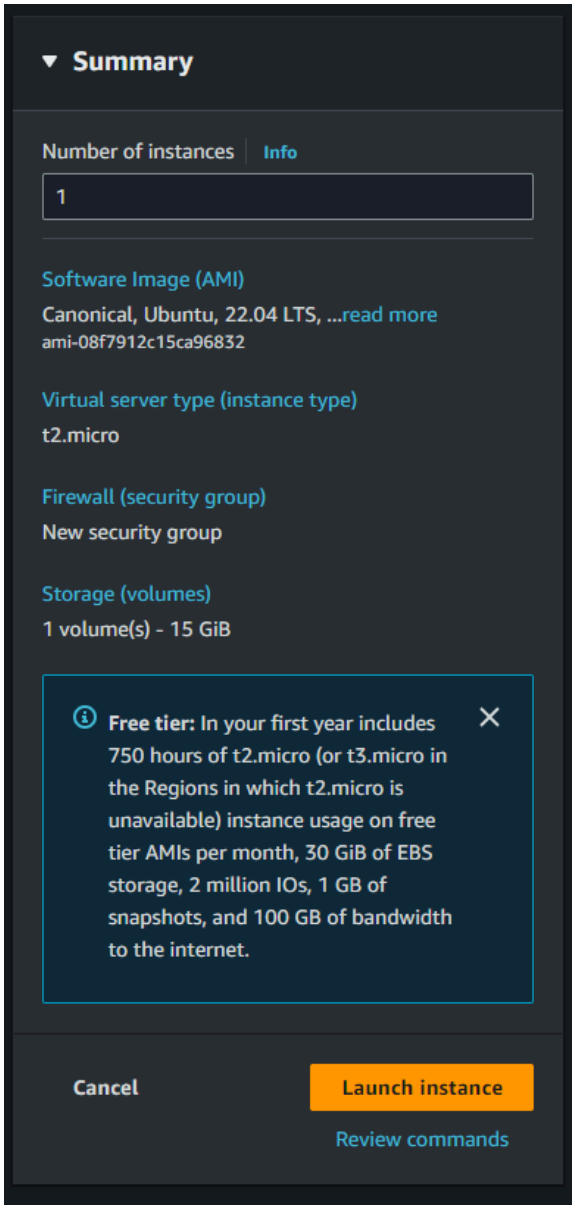
- In the top right section of '**Network Settings**', click on **'edit**'
  - At the bottom of the section, click **'Add security group rule**'
  - Set '**Type'** to '**MYSQL/Aurora**'
  - Set '**Source type**' to '**Anywhere**'



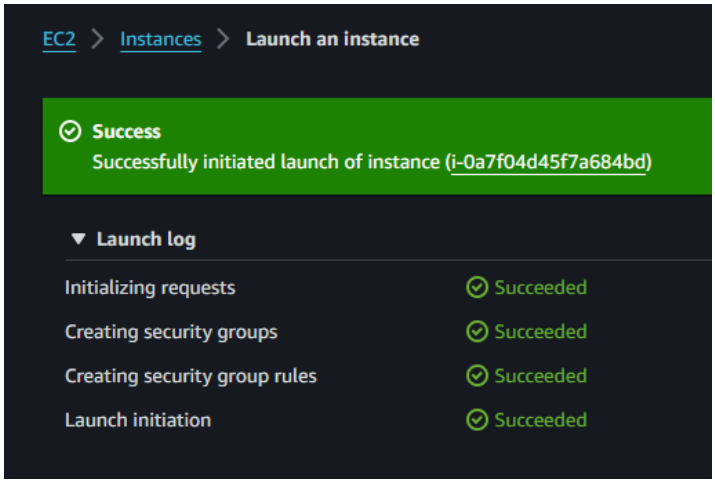
g. **'Storage**' Leave the default 8 GB storage size **or** increase it up to 15 GB



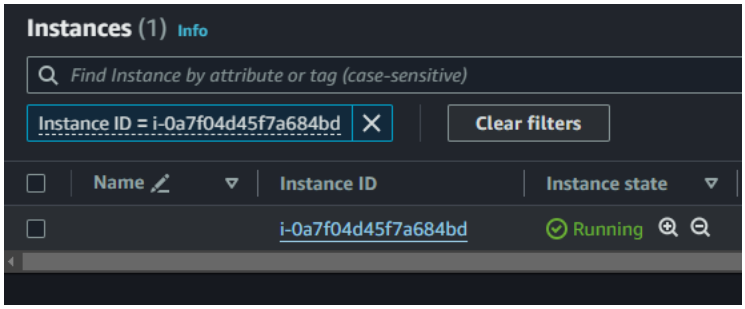
4. Click '**Launch Instance**' at the bottom of the Summary section on the right to begin the EC2 instance build



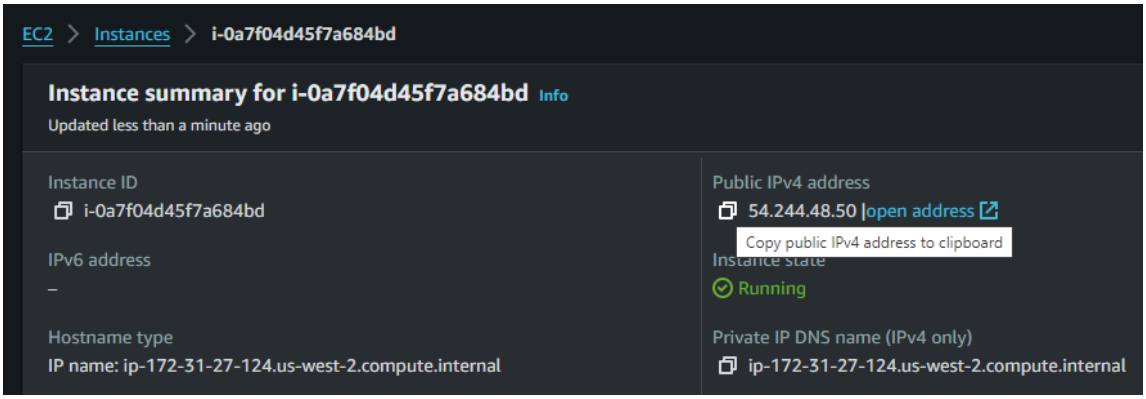
Click on the instance ID (eg. (i-0a7f04d45f7a684bd))



Click on the instance ID again but from the list



5. Find the '**Public IPv4 address**' under the instance details and keep this handy for future tasks
   - Note: This will be used in multiple places
     - `EC2_IP_ADDRESS` GitHub repository secret
     - Connection string in your dev environment (VSCode file: **`appsettings.Development.json`**)



1. Bonus [***optional***] Set up a static IP through ****EC2's Elastic IP**** ****NEW**** **($)**
   a. On the side menu bar, find **Network & Security > Elastic IPs**

   b. Click on "**Allocate Elastic IP address**"
      i. Verify region match under "Network Border Group" - eg. "**us-west-2**" for Oregon
      ii. Keep default of "**Amazon's pool of IPv4 addresses**"
      iii. Click "**Allocate**"

   c. With the new ElasticIP entry selected, Click on Actions > '**Associate Elastic IP address**'
      i. Click in to "**Choose an instance**" and select your EC2 instance
      ii. Click on **'Associate**' to finish
      iii. Verify your EC2 instance has the new Elastic IP allocation associated on the '**Instances**' panel

Stage 2 – AWS EC2 - CUSTOMIZE INSTANCE

(Setup for your EC2 Ubuntu server environment - MySQL, DOCKER, NGINX)

Stage 2.1 – Login, update, then upgrade

(From the EC2 terminal)

1. Log in to your EC2 server from your computer's terminal
   (*navigate to where your key was saved **OR** include the full path to it*)

   `` ssh -i <ec2-server-key-file> ubuntu@<EC2_IP_ADDRESS> ``

   eg. on a Windows command prompt:

   ```
   c:\source\keys>ssh -i MyEC2Instance.pem ubuntu@54.244.48.50
   ```

2. Both Update && Upgrade your system's package manager (apt)
   a. `~$ sudo apt update`

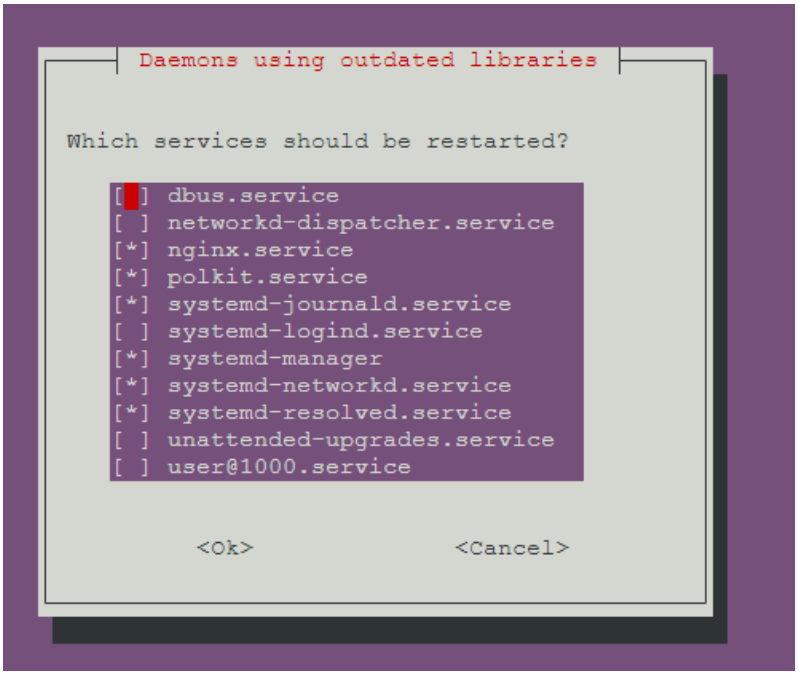
   ```
   Last login: Thu Mar 14 16:40:33 2024 from 18.237.140.164
   ubuntu@ip-172-31-21-230:~$ sudo apt update
   Hit:1 http://us-west-2.ec2.ports.ubuntu.com/ubuntu-ports jammy InRelease
   Get:2 http://us-west-2.ec2.ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [119 kB]
   Get:3 http://us-west-2.ec2.ports.ubuntu.com/ubuntu-ports jammy-backports InRelease [109 kB]
   Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [110 kB]
   Get:5 http://us-west-2.ec2.ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 Packages [1267 kB]
   Get:6 http://us-west-2.ec2.ports.ubuntu.com/ubuntu-ports jammy-updates/universe arm64 Packages [996 kB]
   Get:7 http://us-west-2.ec2.ports.ubuntu.com/ubuntu-ports jammy-updates/universe Translation-en [238 kB]
   Get:8 http://ports.ubuntu.com/ubuntu-ports jammy-security/main arm64 Packages [1058 kB]
   Get:9 http://ports.ubuntu.com/ubuntu-ports jammy-security/universe arm64 Packages [793 kB]
   Fetched 4691 kB in 2s (2323 kB/s)
   Reading package lists... Done
   Building dependency tree... Done
   Reading state information... Done
   20 packages can be upgraded. Run 'apt list --upgradable' to see them.
   ```

   b. `~$ sudo apt upgrade`

   *Type '**y**' and press **enter** to continue when prompted (this may take a minute to complete)*

```
ubuntu@ip-172-31-21-230:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  ubuntu-pro-client
The following packages will be upgraded:
  base-files cloud-init coreutils dpkg flash-kernel iptables libexpat1 libgpgme11 libip4tc2 libip6tc2
  ubuntu-advantage-tools ubuntu-pro-client-l10n ubuntu-release-upgrader-core
20 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
1 standard LTS security update
Need to get 5770 kB of archives.
After this operation, 128 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

   c. *Accept defaults by pressing **enter** to restart the affected daemon services*



Stage 2.2 – DOCKER & NGINX Setup

(From the EC2 terminal)

1. Install **Docker** via Snap
    a. `` `sudo snap install docker` ``

    ***+ NOTE**: Additional Docker related tasks will be handled externally later in Stages 5 & 6*

2. Install **NGINX** via apt
    a. `` `sudo apt install nginx` ``
    b. `` `sudo systemctl enable nginx` ``

    ***+ NOTE**: Additional NGINX related tasks will be handled later in Stage 4*

Stage 2.3 – MySQL Setup **(note)

(From the EC2 terminal, then into the MySQL environment, and finally testing connectivity in VSC)

1. Install MySQL via APT and enable the MySQL service on system boot

    a. `~$ sudo apt install mysql-server`
    b. `~$ sudo systemctl enable mysql`

2. Secure the MySQL installation - Removes default accounts & data

    a. `~$ sudo mysql_secure_installation`
        i. Choose **'yes'** and **confirm** everything by following the prompts
        ii. Select **#2** for strong password requirements

b. `~$ sudo mysql `

c. Assign a password for your MySQL root user account
```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '<password>';
mysql> FLUSH PRIVILEGES;
```

d. Return to the EC2 [ubuntu] server terminal
```
mysql> exit
```

[***recommended***] Set up a password manager or locked file to track all dev environment credentials
eg. via Bitwarden - https://vault.bitwarden.com/#/vault

3. Restrict size potential of MySQL && Reduce RAM usage by disabling performance trackers ****NEW****

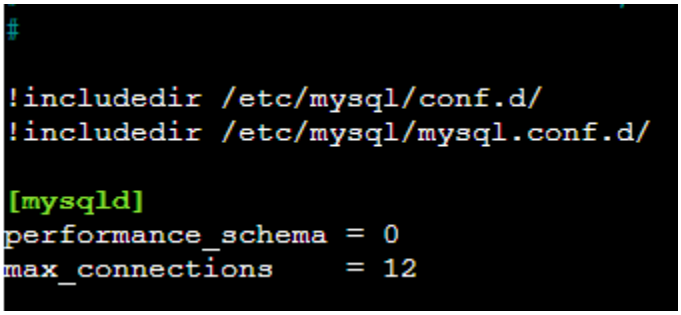
a. Edit the **`my.cnf`** file in your server's **`/etc/mysql`** directory to further customize MySQL

```
~$ sudo nano /etc/mysql/my.cnf
```

b. Copy/Paste in the following highlighted lines at the end of that file:

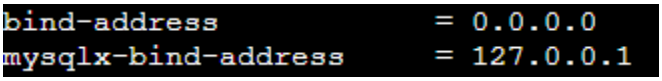
```
[mysqld]
performance_schema = 0
max_connections    = 12
```

c. **`CTRL+S`**, then **`CTRL+X`** (*to Save then Exit from the nano text-editor*)



4. Permit anyone to talk to the MySQL server

a. `~$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf`
b. Locate the [**`mysqld`**] section and modify as: ` **`bind-address = 0.0.0.0`** `
c. **`CTRL+S`**, then **`CTRL+X`** (*to Save then Exit from the nano text-editor*)



5. Restart the mysql service to load in the config changes (*note: no output will be returned on restarts*)
a. `~$ sudo systemctl restart mysql`

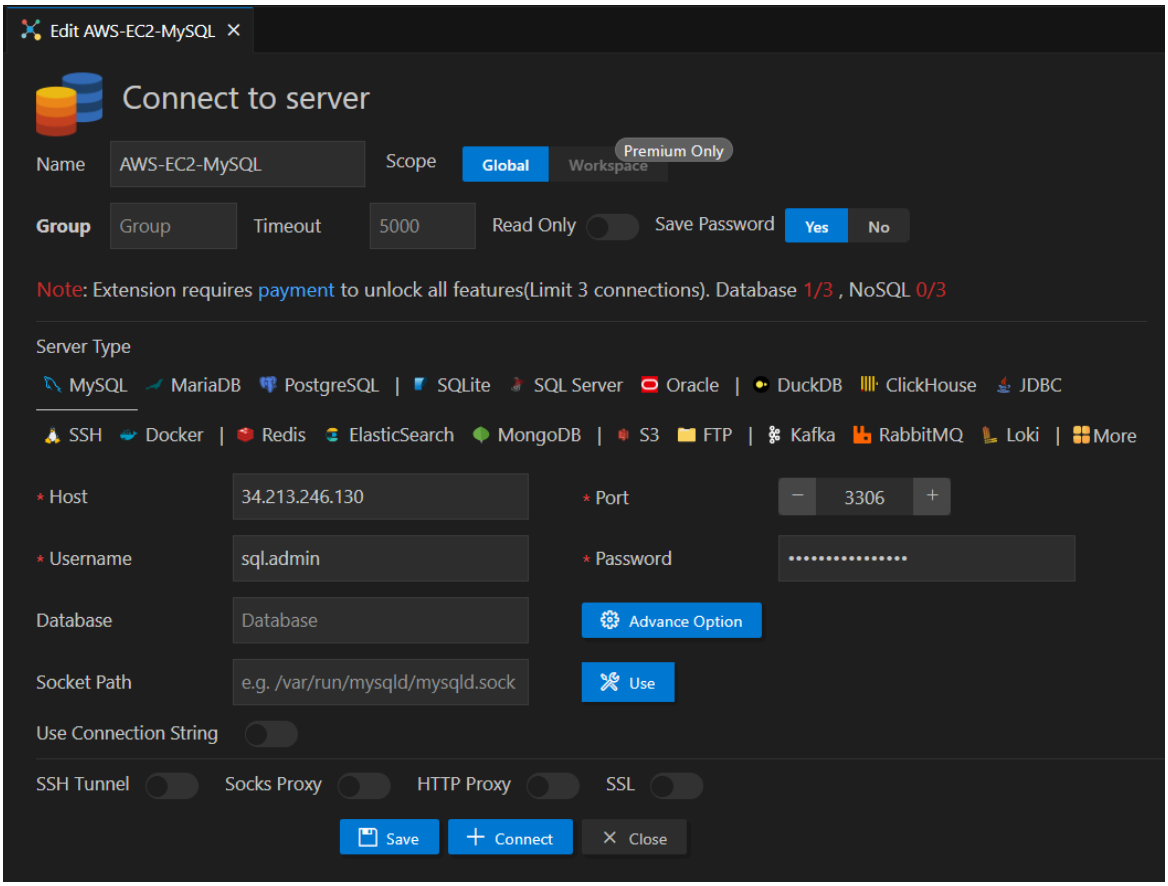
6. Log in to mysql again
`~$ sudo mysql -p` [Enter], then type in your root user's password

7. Add your remote mysql db user
(*only one account required for the sake of this lab - but each user requires sending all the commands listed below*)

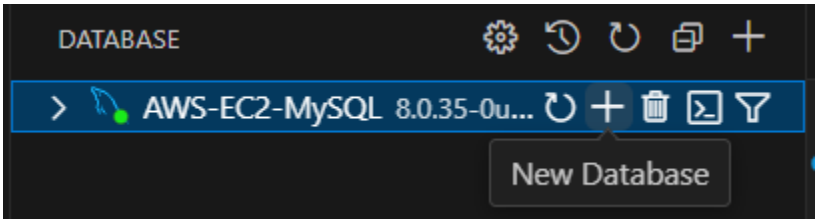
For one remote user with access to everything:

a. `mysql> CREATE USER '<remote.sql.usr>'@'localhost' IDENTIFIED BY '<new-password>';`
b. `mysql> CREATE USER '<remote.sql.usr>'@'%' IDENTIFIED BY '<new-password>';`
c. `mysql> GRANT ALL PRIVILEGES ON *.* to '<remote.sql.usr>'@'localhost';`
d. `mysql> GRANT ALL PRIVILEGES ON *.* to '<remote.sql.usr>'@'%';`
e. `mysql> FLUSH PRIVILEGES;`
f. `mysql> EXIT`

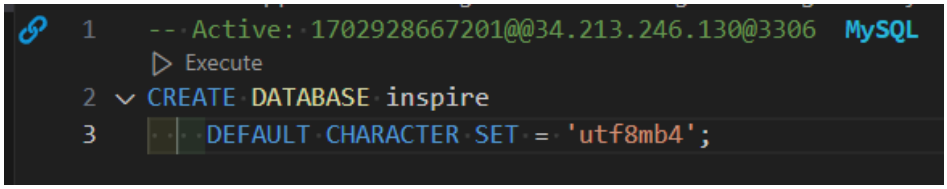
8. Test your MySQL connectivity with the `EC2_IP_ADDRESS` and the remote mysql user just created
   a. Use the VSC MySQL extension (*pancakes*) and create a new connection, save, and connect to test
      i. Select **MySQL** for Server Type
      ii. Input your EC2 server instance's public **IP Address** for Host
      iii. Input the **remote sql user** created earlier for the Username field
      iv. Input the corresponding user's **password**
      v. Note: Setting a Database name here is not required if the specified remote sql user was not limited to a specific database



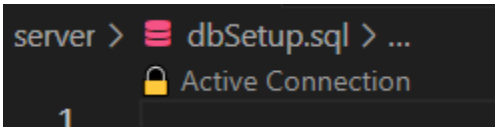
   b. Create the databases that your apps can use (eg. inspire, tower, etc)
      i. Click the + button on the new connection to create a DB



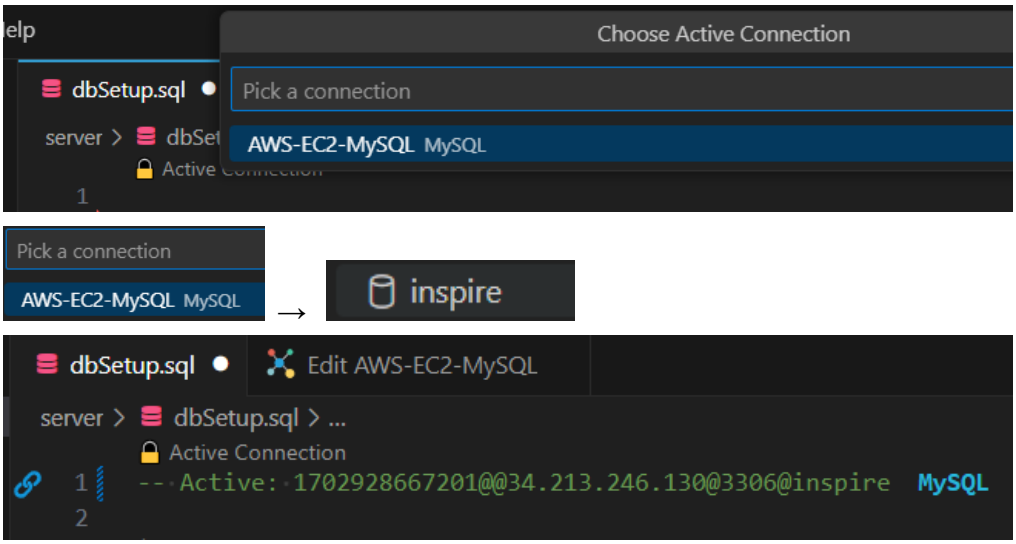
Add a name and click on the "**Execute**" button to send the command
`CREATE DATABASE <db_name> DEFAULT CHARACTER SET = 'utf8mb4';`



c. Assign the desired database association in your VSCode
   i. Open the **`dbSetup.sql`** file
   ii. Click on "🔒 **Active Connection**" at the top of the file



   iii. Navigate to your desired database by clicking through the new Connection and finding the DB name that was just created



1. Optionally try configuring some mysql parameters (*just for these lab conditions*)

```
SET GLOBAL innodb_undo_log_truncate=ON;
SET GLOBAL innodb_max_undo_log_size=104857600;
SET GLOBAL binlog_expire_logs_seconds=604800;
```

Stage 3 – DOMAIN NAME, DNS RECORDS & SSL CERT

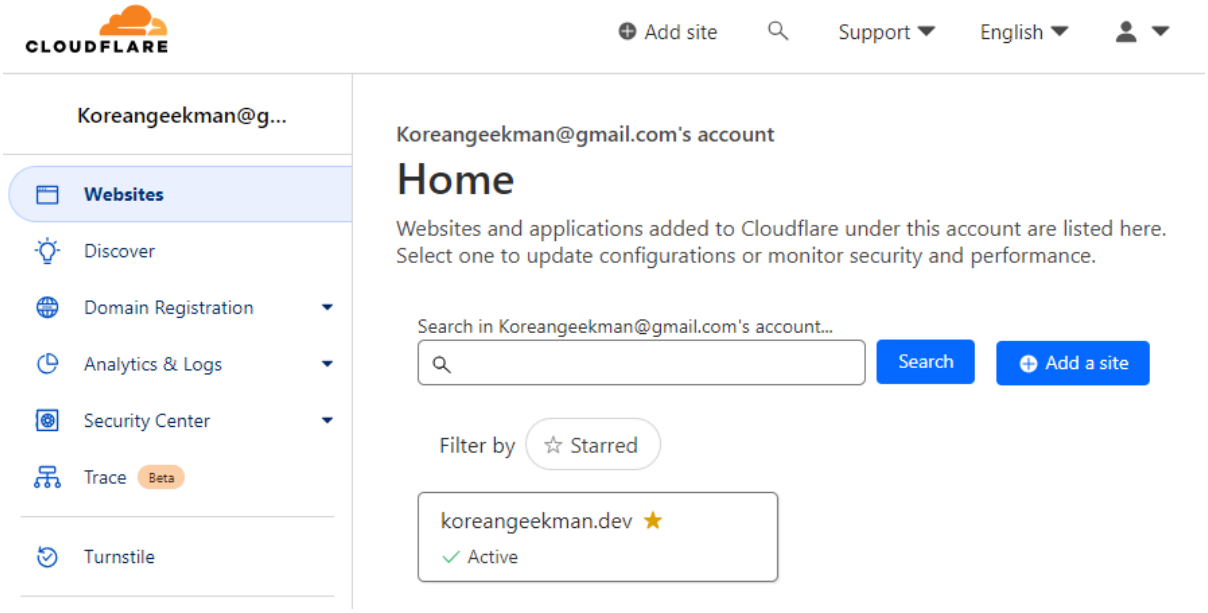
(Obtain your own internet domain ($), point the address, create SSL cert, permit auth0 access)

Stage 3.1 - DNS & SSL setup

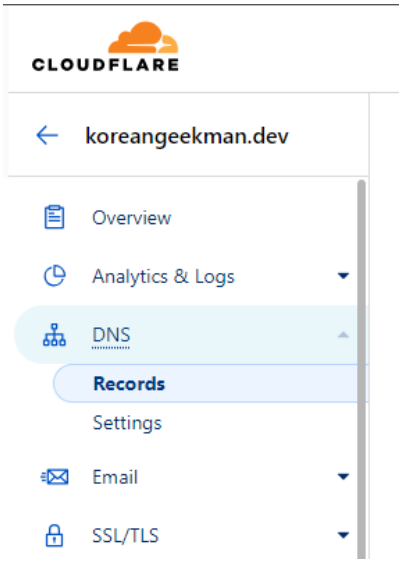
(From a web browser, then back in to EC2)

1. Sign up to a domain registrar and buy a domain ($) of your choice
   Cloudflare is known as a decent registrar to work through **and** provides free SSL certs

2. If not already opened, click in to your purchased domain



3. Navigate to your **DNS > Records** page of the “Website” / Domain on Cloudflare

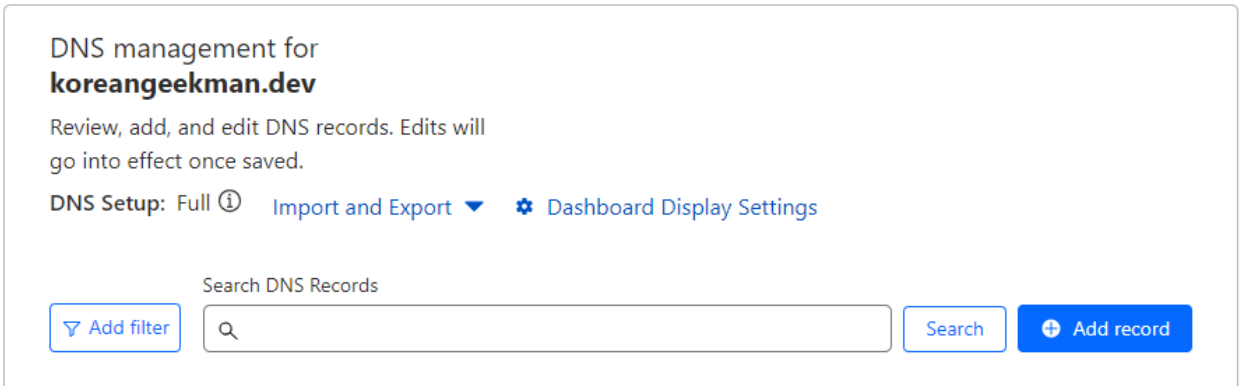


4. Add multiple **A records** for your sites using the `Public IP` of your `EC2` server

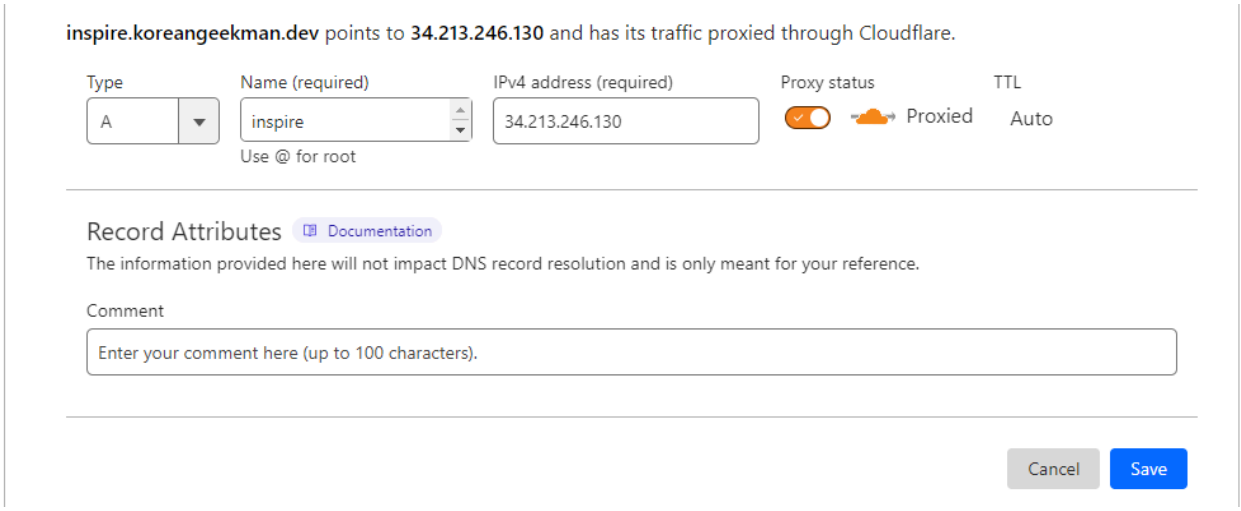
Type	Name	IPv4 address
A	inspire	34.123.248.32
A	tower	34.123.248.32
A	capstone	34.123.248.32

etc..

a. Click on “**Add record**” to expand the form



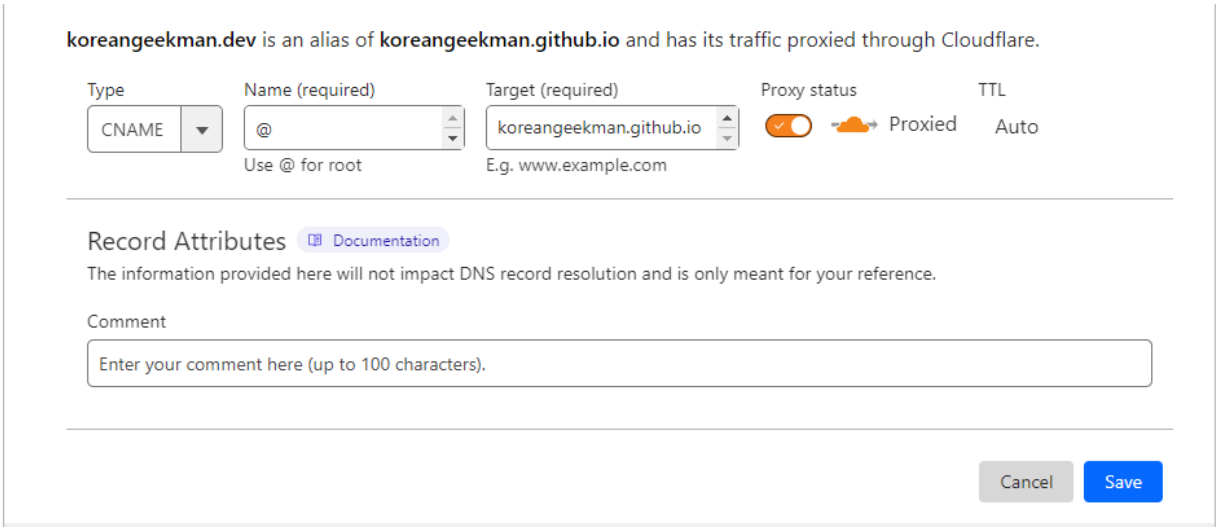
b. Enter the **Name** & Public **IP Address** of your EC2 server (remaining settings on defaults) And hit “**Save**”



5. Add a couple **CNAME records** for your landing page
(Using GitHub as the host - ie. ***<github_username>*.github.io**)

Type	Name	Target address example
CNAME	@	koreangeekman.github.io
CNAME	www	koreangeekman.github.io

a. Click to “**Add Record**”,
b. Select “**CNAME**” under Type,
c. “**@**” to designate the root under Name,
d. Input your GitHub landing page repository name under Target

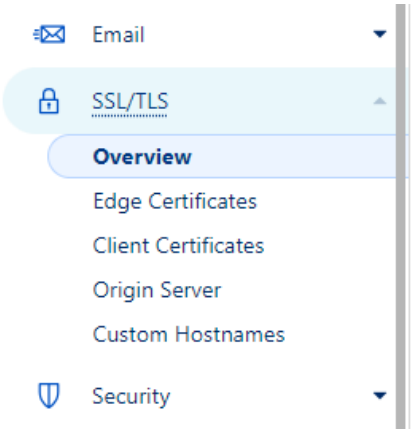


Final resulting list of A & CNAME records will look something like the below image
(The number of A records will vary depending on the number of apps to be deployed)

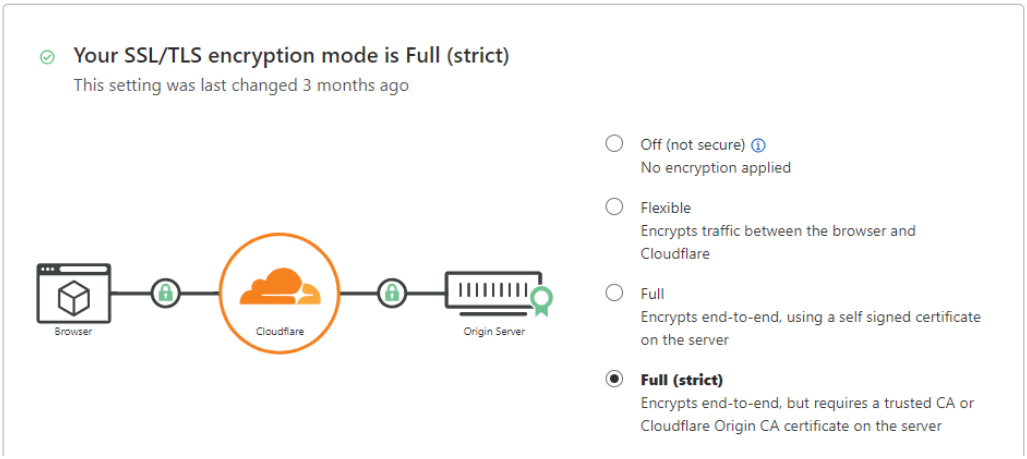
A	topomodoro	34.213.246.130	Proxied	Auto	Edit
A	tower	34.213.246.130	Proxied	Auto	Edit
CNAME	koreangeekman.dev	koreangeekman.github.io	Proxied	Auto	Edit
CNAME	www	koreangeekman.github.io	Proxied	Auto	Edit

6. Enable end-to-end encryption for your domain (between the DNS servers and EC2 Server)

   a. Navigate to **SSL/TLS > Overview** page

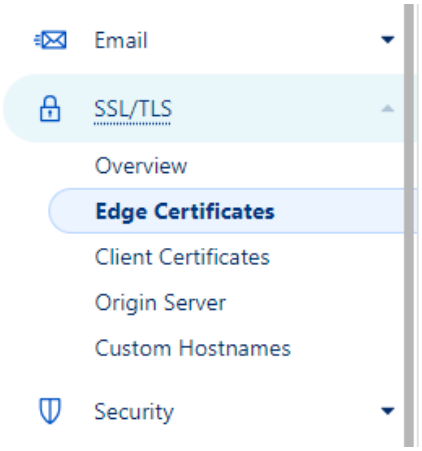


   b. Change the SSL/TLS encryption mode to **Full (strict)**

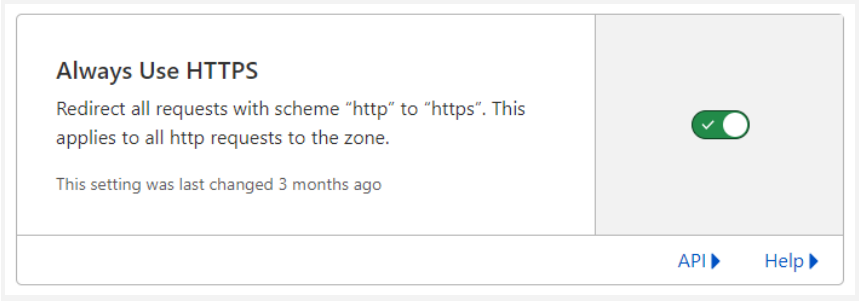


7. Let Cloudflare auto-redirect all HTTP requests to HTTPS at the DNS servers

    a. Navigate to **SSL/TLS > Edge Certificates** page

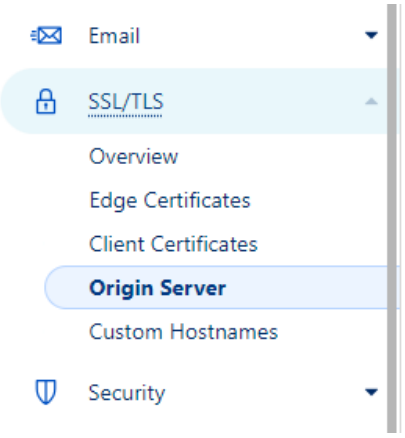


    b. Enable **Always Use HTTPS**

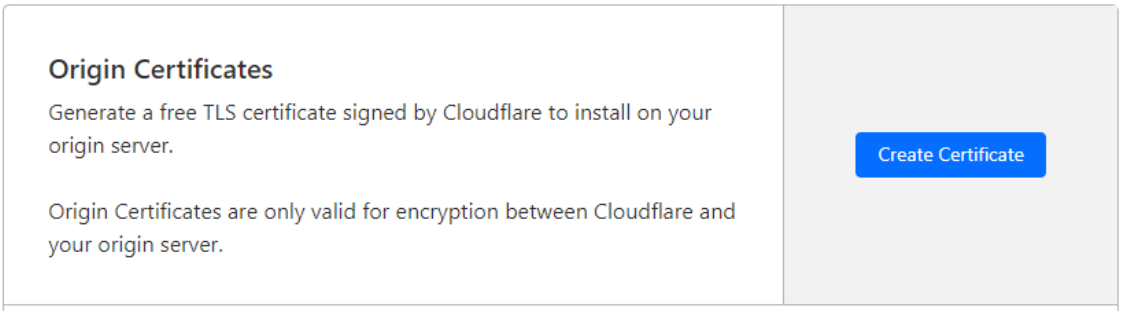


8. Create an Origin SSL certificate for HTTPS functionality

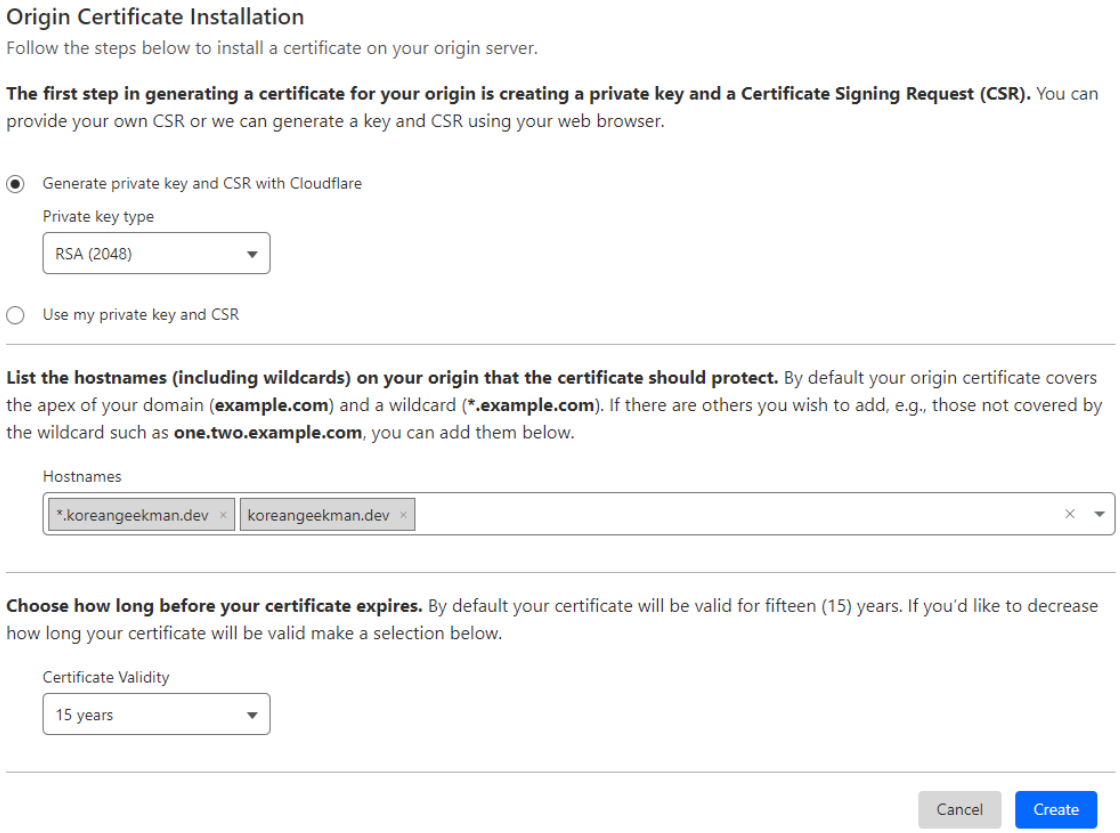
    a. Navigate to **SSL/TLS > Origin Server** page



    b. Click on '**Create Certificate**'

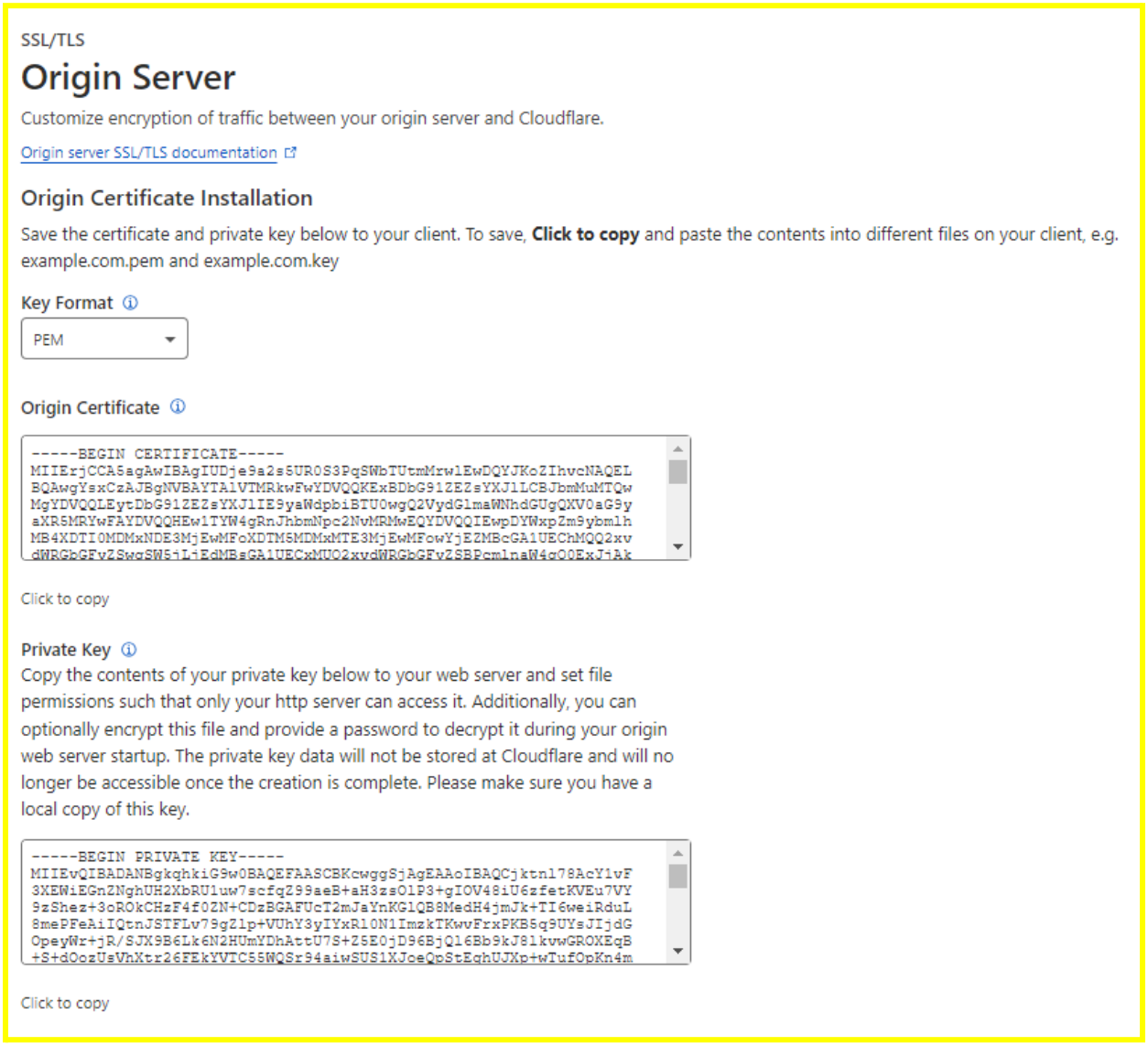


i. Default values are acceptable, click **‘Create**’



c. Note the **Origin Certificate** and **Private Key** fields - we will be copying these two fields in the next step

(*Keep this page open until everything in this stage is complete*)



9. Jump back into your EC2 server instance and add your SSL cert as 2 files

    a. Navigate to the **`/etc/ssl`** folder of your EC2 instance
    `~$` **`cd /etc/ssl`**

    b. Create two new files, **`cert.pem`** & **`key.pem`**, using the **nano** text editor

    `~$` **`sudo nano cert.pem`**
    Copy/paste in ALL contents of the ***Origin Certificate*** *textarea, (no leading/trailing spaces or lines)*
    `CTRL+S` *(to save),* then **`CTRL+X`** *(to exit)*

    `~$` **`sudo nano key.pem`**
    Copy/paste in ALL contents of the ***Private Key*** *textarea, (no leading/trailing spaces or lines)*
    `CTRL+S` *(to save),* then **`CTRL+X`** *(to exit)*

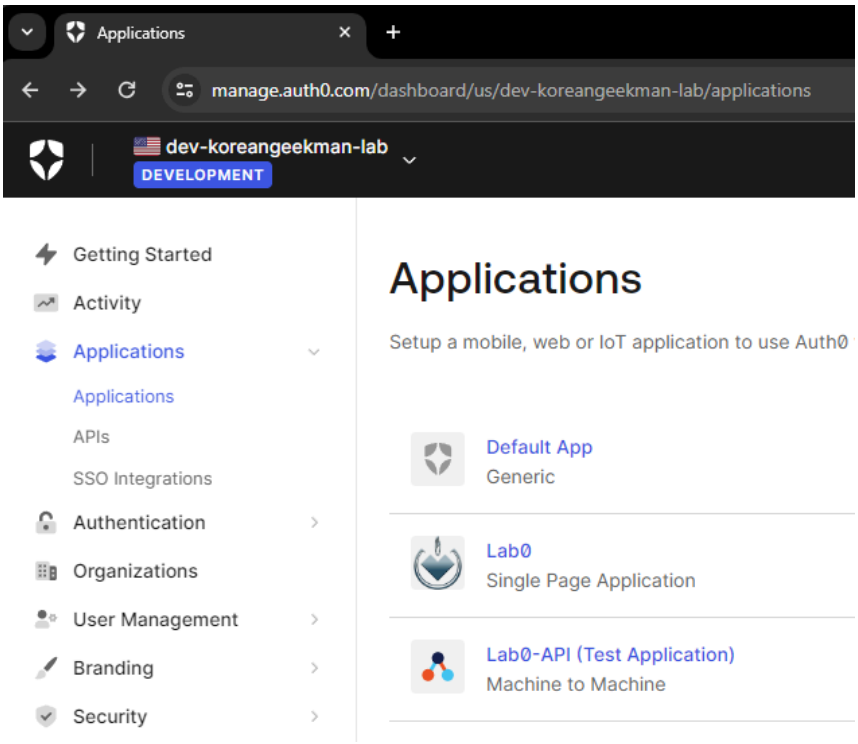
```
ubuntu@ip-172-31-21-230:~$ cd /etc/ssl
ubuntu@ip-172-31-21-230:/etc/ssl$
ubuntu@ip-172-31-21-230:/etc/ssl$ sudo nano cert.pem
ubuntu@ip-172-31-21-230:/etc/ssl$ sudo nano key.pem
ubuntu@ip-172-31-21-230:/etc/ssl$
ubuntu@ip-172-31-21-230:/etc/ssl$ ls -lh
total 48K
-rw-r--r-- 1 root root 1.7K Mar 14 17:34 cert.pem
drwxr-xr-x 2 root root  20K Dec  7 02:13 certs
-rw-r--r-- 1 root root 1.7K Mar 14 17:35 key.pem
-rw-r--r-- 1 root root  13K Oct 13 12:02 openssl.cnf
drwx------ 2 root root 4.0K Mar 16  2022 private
ubuntu@ip-172-31-21-230:/etc/ssl$
```

Stage 3.2 - Permit your domain in Auth0

(From a web browser)

1. Open up the Auth0 dashboard https://manage.auth0.com/dashboard

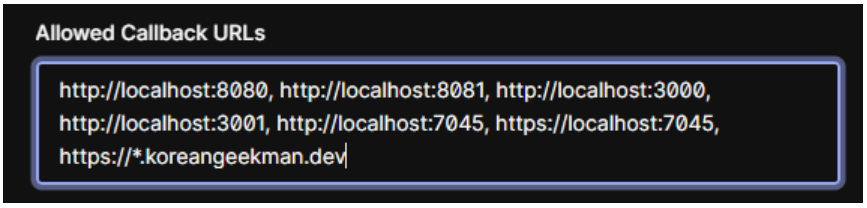
2. Navigate to **Applications/Applications** > open your "**Single Page Application**" entry



3. Under **Settings**, add your domain to each of the four *Allowed* text-blocks as a new entry
    a. Using the wildcard form with https - *eg*. **`https://*.koreangeekman.dev`**
    Add to:
        - **Allowed Callback URLs**
        - **Allowed Logout URLs**
        - **Allowed Web Origins**
        - **Allowed Origins (CORS)**

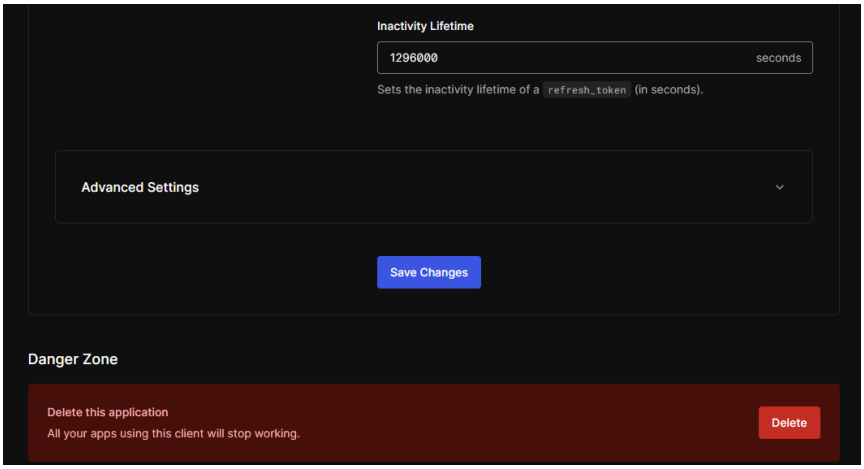
eg. textblock contents as:

`http://localhost:8080, http://localhost:8081, http://localhost:3000, http://localhost:3001, http://localhost:7045, https://localhost:7045, https://*.koreangeekman.dev`



x4 locations

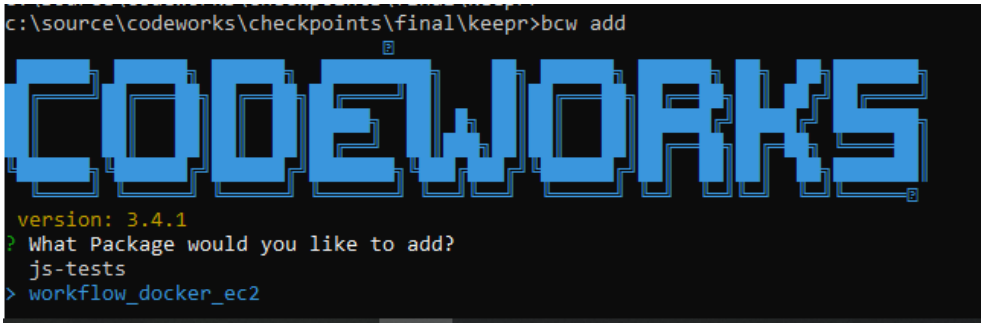
4. Be sure to **[Save Changes]** at the bottom



Stage 4 – ADD FILES & CUSTOMIZE CONFIG

(Back to your local machine terminal & VSC)

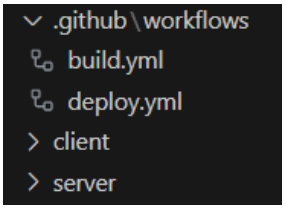
1. Navigate to your app's root folder via terminal
2. Run `bcw add` and select `workflow_docker_ec2`



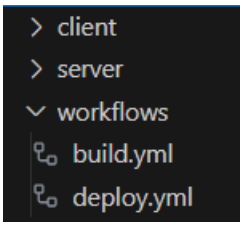
   a. Expected path/file creation: (*2 script files for* `Github Actions` *automation*)

   `/.github/workflows/build.yml`
   `/.github/workflows/deploy.yml`

In VSCode (outside workspace):



In VSCode (inside workspace):



3. [*conditional - <u>only</u> **IF** you have more than one Node or .NET project*]
   Configure a custom port for routing

   <u>DOCKER-COMPOSE:</u> `` `/server/docker-compose.yml` ``

   i. [*conditional*] Customize the port in ***docker-compose.yml*** when necessary:

   1. Node.js projects <u>default</u> to port 3000
      a. Update the docker port translation for additional Node.js projects
      `` `docker-compose.yml : line 8 `= ` - “3001:3000” ` ``

   2. .NET projects <u>default</u> to port 7045
      a. Update the docker port translation for additional .NET projects
      `` `docker-compose.yml : line 8 `= ` - “7046:80” ` ``

   ii. [*optional*] Keep a record of all ports to apps being created for tracking - for example:

```
Port              App                                    DNS entry
DNS [ Type (A | CNAME): Target ]
3000              Tower                                  tower.koreangeekman.dev
A: 34.123.248.32
3001              Capstone     capstone.koreangeekman.devA: 34.123.248.32
7045              Keepr                                  keepr.koreangeekman.dev
A: 34.123.248.32
7046              All Spice    allspice.koreangeekman.devA: 34.123.248.32
----              Landing Page @ [aka] koreangeekman.dev CNAME: koreangeekman.github.io
```

4. [*optional but recommended - for older templates*] Customize the Dockerfile images

   <u>DOCKERFILE:</u> `` `/server/Dockerfile` ``

   i. (***DOTNET** APP*) On **line 36**: ****NEW****

   [***recommended***] For a smaller docker image build, change

   Line 2 : `` `node:20` `` into `` `--platform=linux/amd64 node:20-slim` ``
   Line 19: `` `/sdk:8.0` `` into `` `/sdk:8.0-alpine-amd64` ``
   Line 36: `` `/sdk:8.0` `` into `` `/aspnet:8.0-alpine` ``

ii. (***NODE/EXPRESS** APP)* On **line 2** & **20**: ****NEW****

[***recommended***] For a smaller docker image build, change

Line 2 : `` `node:20` `` into `` `**--platform=linux/amd64 node:20-slim**` ``
Line 20: `` `node:20` `` into `` `**node:20-slim**` ``

<u>Stage 5</u> – FORWARD DOMAIN ROUTES VIA NGINX

(From the EC2 server terminal)

1. Edit your default nginx load file to enable the custom routes

    a. `` `**sudo nano /etc/nginx/sites-available/default**` ``

    b. Either comment out or delete everything currently active and paste in the desired code

SUB-DOMAIN CONFIG

(*Use the entire block-pair for every subdomain/app you want to route*)

```
server {
    listen 80;
    listen [::]:80;
    server_name <App.TargetDomainAddress.tld>;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    ssl_certificate /etc/ssl/cert.pem;
    ssl_certificate_key /etc/ssl/key.pem;

    server_name <App.TargetDomainAddress.tld>;

    location / {
        proxy_pass http://127.0.0.1:<port>;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
        proxy_read_timeout 86400;
    }
}
```

---

c. **CTRL+S**, **CTRL+X** (to save & exit)

d. Test your config changes for obvious errors (*fix any issues Before restarting the nginx service*)
` **sudo nginx -t** `

e. Restart the NGINX service to load in the new default config
` **sudo systemctl restart nginx** `

NOTE: [*optional*] Add a default redirector for all unspecified cases

See **APPENDIX A - NGINX Routing logic & Default Redirection Config** for more details

---

Stage 6 – DOCKER & DOCKERHUB REPOS

(From the Docker website, then into VSC)

1. Sign up for a Docker account - https://hub.docker.com/signup
   *** *A docker user/password pair will be required for automation* ***
   [*Option **1***] - Manually enter an email/password
   [*Option **2***] - Sign up with Github/Gmail, then add a password to the account

2. Create a Dockerhub repo for your app and remember this *** **repo-name** ***
   a. [*optional*] match **exactly** the **repo-name** to the **app-name**.. **IF** possible on these conditions:

      Your repository name must contain a combination of alphanumeric characters
      and may contain the special characters . , _ , or - . **Letters must be lowercase**.

3. Open your ` **docker-compose.yml** ` file from Stage 4, Step 3.a.
   a. Input your Dockerhub username in place of `**your-dockerhub-username**` on line 4

   b. [**IF** unable to match the dockerhub **repo-name** to the **app-name**]
      Adjust the ` image ` and ` env ` entries below to match the Dockerhub repository

      eg. if Dockerhub repo is `**tower_app**`, then

      ```
      image: koreangeekman/tower_app:latest
      env_file:
        - tower_app-env
      ```

Stage 7 – ADD REPO SECRETS TO GITHUB REPO

(From the Github website)

1. Navigate to your app's secrets store
   Github > AppRepo > Settings > Security > Secrets and variables > Actions

   eg. https://github.com/koreangeekman/Tower/settings/secrets/actions

2. Referencing ` /.github/workflows/**deploy.yml** ` for variable names, add your repository secrets

   NOTE 1:
   Each secret will be a single line value, with exception to the **EC2_PEM_KEY** & **ENV_FILE**
   *** Be sure to remove any leading and trailing spaces/blank lines ***

   NOTE 2:
   Secrets will not be visible after saving - editing a secret will always show blank contents

   EC2_IP_ADDRESS: *<EC2 Public IP>*
   EC2_USERNAME: **ubuntu**
   EC2_PEM_KEY: *<full contents from when generated>*
   REPO_NAME: *<docker-repo-name>* (*eg. **tower_app***)
   DOCKER_USERNAME: *<self-explanatory>*
   DOCKER_PASSWORD: *<self-explanatory>*
   ENV_FILE: *<depends on project type - see below *>*

   ***** For (**NODE/EXPRESS** APPs): copy all contents of ` **/server/.env** ` file into **ENV_FILE**
   - Change your ` **NODE_ENV=dev** ` from *'**dev**'* to *'**production**'* when submitting it as a secret

   ***** For (**.NET/C#** APPs): copy, and reformat, env variable contents of the ` **/server/appsettings.Development.json** ` file into **ENV_FILE** – template below:

CONNECTION_STRING=server=<***EC2_IP_ADDRESS***>;database=<***DBName***>;port=3306;user id=<***username***>;password=<***password***>;
AUTH0_DOMAIN=<*auth0-domain*>
AUTH0_AUDIENCE=<*auth0-audience*>

Stage 8 – THE FINAL PUSH

SAVE FILES & PUSH/SYNC, CREATE 'production' BRANCH ON GITHUB, WAIT & HOPE

1. Save & Commit all new files and changes, push/sync up to github

2. Create branch from '**main**' called '**production**' on your github repo

3. 🙏 Pray 😂

4. Watch the process in Github Actions for errors
   For potential solutions: Refer to the troubleshooting guide in **APPENDIX T**
   or reach out to me (DJ)

Stage 9001 – ONGOING MAINTENANCE **NEW**

Every server requires some kind of monitoring or maintenance

Whether automated or manual, something needs to keep watch over a new system and setup in case of bugs or unexpected conditions.

Here are a few things to keep in mind and check on occasion:

- **Disk space utilization**
  - Run the command ` `df -h` ` to check on total drive space usage
    - If running low.. See APPENDIX T - EC2 disk space full (if really close to full, cleanup or expand BEFORE it fills or risk corrupting MySQL)
  - Run ` `sudo du -cha --max-depth=2 /var/ | grep -E "^[0-9\.]*[MG]"` ` for a summary of usage by file & folder in the **/var** directory
  - Run ` `sudo docker images` ` to check on old or orphaned images on your EC2 instance
    - Run ` `sudo docker image prune` ` to clean up any unused old versions of images
    - Run ` `sudo docker image remove <imageID>` ` to remove an unused & orphaned image from local storage
      - Images used in any containers listed from ` `sudo docker ps -a` ` (active or not) CANNOT be removed (docker prevents it)
      - You must remove the container it is locked in association with in order to remove an image from your system
      - This can be set up to run automatically with crontab on a daily basis at one minute past midnight (12:01am) **NEW**
        - Edit your crontab file: ` `crontab -e` ` and select a text editor (default #1 of nano is sufficient)
        - Input this line at the end: ` `1 7 * * * sudo docker image prune -f` `
  - Run ` `sudo snap list –all` ` to check on old or orphaned images on your EC2 instance
    - Run ` `sudo snap remove <snap-name> --revision=<revision#>` ` to remove a disabled image from local storage
    - A default of 3 revisions are kept for rollbacks - you can limit this down to 2 at a minimum with the command: **NEW**
      ` `sudo snap set system refresh.retain=2` `

- **Memory (RAM) utilization**

- Run the command ` **`top`** ` and use the keyboard shortcut: ` **`[SHIFT] + M`** ` to sort by memory utilization
  - ` **`[SPACE]`** or **`[ENTER]`** ` to updates the table on demand (default updates every 3 seconds)
    - To change the auto-update speed, press ` **`[s]`** ` and enter an integer or float value ie. ` **`1`** ` || ` **`1.0`** ` for 1 second, then hit ` **`[ENTER]`** `
      - SIDE NOTE: Update speeds faster than once per second and your free trial instance may be too slow to keep up
  - ` **`[CTRL] + C`** ` to exit the process table

- Run the command ` **`systemctl status <process-name>`** ` to see the memory usage on a single process
  - ie. ` **`systemctl status mysql`** `
    - If your MySQL is using more than 300MB of RAM when idle, be sure to disable the performance monitoring schema
      (Process detailed in Stage 2.2 - Step.3)
    - ~~You may also want to flush your MySQL memory from any past connections~~ ~~mysql>~~ ~~` **`FLUSH HOSTS;`** `~~ *Deprecated as of MySQL 8.0.23 - soon to be removed..*
      Replacement command is ` **`TRUNCATE TABLE performance_schema.host_cache;`** `
      *But if you have disabled the **`performance_schema`** as instructed, it's not a necessary flush*
    - ~~This can be set up to run automatically with crontab on a daily basis at midnight (server time = UTC) (5pm MST)~~ ~~**NEW**~~
      - ~~Edit your crontab file: ` **`sudo crontab -e`** ` and select a text editor (*default option1 of nano is sufficient*)~~
      - ~~Input this line at the end: ` **`0 0 * * * mysql --login-path=dev -e 'flush hosts;'`** `~~

- **CPU utilization**
  in our case, we're extremely limited on the first year Free Tier but our CPU resource requirements are fortunately quite minimal



NOTES & TROUBLESHOOTING GUIDE

APPENDIX A - ADDITIONAL NOTES

CODEWORKS TEMPLATES & BCW

How does the ` **bcw create** ` or ` **bcw add** ` work?

- When running the **create** or **add** features, the program pulls the latest revision of the templates from their respective github repositories.
    - https://github.com/codeworks-templates

- Each of the templates - front end, full-stack, etc - are built from individual component templates and then assembled/organized into their respective folders, where their packages are populated after ` **npm i** ` is run in each directory with package.json dependency lists (server & client folders)

Project Template	Front-End Repo	Back-End Repo
mvc	mvc	n/a
mvc-auth	mvc-auth	n/a
node-server-auth0	n/a	node-server-auth0
express-mvc	mvc-auth	node-server-auth0
express-vue	vue-starter	node-server-auth0
express-react	react-starter	node-server-auth0
vue-starter	vue-starter	n/a
react-starter	react-starter	n/a
dotnet-vue	vue-starter	dotnet-webapi
dotnet-react	react-starter	dotnet-webapi
` **bcw add** `	workflow_docker_ec2	

Alternate BUILD Config for Github Actions

- For MULTI-ARCHITECTURE docker image builds, use the below config as is
- For a specific one, specify just the one desired in the "`platforms`" field below

```
name: Docker Build

on:
  push:
```

```yaml
    branches:
      - production

jobs:
  build-push-deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout Repository 👀
        uses: actions/checkout@v4

      - name: Set up QEMU
        uses: docker/setup-qemu-action@v3

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Login to Docker Hub 🔐
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Build and Push Docker Image 🐋
        uses: docker/build-push-action@v5
        with:
          platforms: linux/amd64,linux/arm64
          tags: ${{ secrets.DOCKER_USERNAME }}/${{ secrets.REPO_NAME }}:latest
          file: server/Dockerfile
          context: .
          push: true
```

GITHUB PRs (Pull Requests) Disabled?

- [**WHEN** creating a pull request **&& WHEN** only **one** commit since the last 'production' branch pull] The pull request gets auto-populated with the commit's information so it's available to click through.

- [**WHEN** creating a pull request **&& WHEN** **multiple** commits since the last 'production' branch pull] The button to create the pull request will be 'disabled' because it needs you to provide a summary of all commits. Add something to the title and the button will re-enable.

ENV_FILE & docker-compose.yml - Verification

(*Available after a 'Deploy to EC2' attempt that successfully logged in to your EC2 server*)

- [**IF** uncertain of the **ENV_FILE || docker-compose.yml** contents used in a deployment]

(*using the Dockerhub repo name*)

- ‘**cat**’ out OR open the file via **nano** - they are created in your EC2 user’s (**ubuntu**) home path

  `` `ls -l ~` ``
  `` `cat ~/<repoName>-env` ``
  `` `cat ~/<repoName>-compose` ``

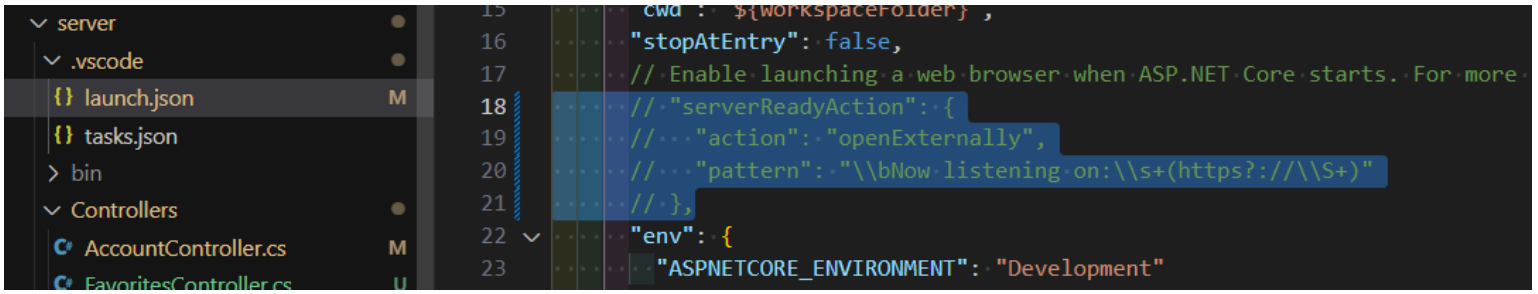
- You can edit the *reponame*-**compose** and *reponame*-**env** files from the home directory then re-run the docker instance build to test immediate changes without re-deploying from github actions

  `` `sudo docker-compose -p <repoName> -f <repoName>-compose up -d` ``

Disable auto-loading Swagger on .NET projects

Tired of getting flash-banged by the bright white Swagger page?

- To disable the auto-browser launch for swagger, you can comment out this section of code in the folder>file: `` `.vscode/launch.json` `` (lines 18-21)



Useful MySQL Commands

- MySQL run-state is managed as a service on the server - use systemctl to control

  `` `systemctl status mysql` ``
  `` `sudo systemctl restart mysql` ``
  `` `sudo systemctl stop mysql` ``

- You can show the **databases created** on your MySQL environment with the following query:

  `mysql> SHOW DATABASES;`

- You can show the **users configured** on your MySQL environment with the following query:

  `mysql> SELECT user,host FROM mysql.user;`

- You can show the **privileges granted** on your MySQL environment with the following query:

  `mysql> SHOW GRANTS for ’<username>’;`
  `mysql> SHOW GRANTS for ’<username>’@’localhost’;`

- You can show the **database sizes** (file-size) on your MySQL environment with the following query:

`mysql> SELECT table_schema AS "Database",`

```
  ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
  FROM information_schema.TABLES
  GROUP BY table_schema;
```

- You can convert from a TABLE view to a vertical column/‘stacked’ view of the data by adding ` **\G** ` at the end in place of the ending semi-colon ` **;** `

MySQL defaults

```
mysql> select user,host from mysql.user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| debian-sys-maint | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
5 rows in set (0.01 sec)


mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)


mysql> SELECT table_schema AS "Database",
       ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS "Size (MB)"
       FROM information_schema.TABLES
       GROUP BY table_schema;
+--------------------+-----------+
| Database           | Size (MB) |
+--------------------+-----------+
| information_schema |      0.00 |
| mysql              |      2.63 |
| performance_schema |      0.00 |
| sys                |      0.02 |
+--------------------+-----------+
4 rows in set (0.16 sec)
```

Useful NGINX Commands

- NGINX run-state is managed as a service on the server - use systemctl to control

  `` `systemctl status nginx` ``
  `` `sudo systemctl restart nginx` ``
  `` `sudo systemctl stop nginx` ``

- **Note**: Useful commands for troubleshooting or additional information (*not required for startup*)

  `` `sudo nginx -t` ``
  `` `sudo nginx -T` ``
  `` `sudo systemctl restart nginx` ``
  `` `systemctl status nginx` ``

Useful DOCKER Commands

- **Note**: Useful commands for troubleshooting or additional information (*not required for startup*)

  `` `sudo docker images` ``
  `` `sudo docker ps` == `sudo docker container ls` ``
  `` `sudo docker ps -a` ``
  `` `sudo docker-compose -p <repoName> -f ~/<repoName>-compose up -d` ``
  `` `sudo docker container prune` ``
  `` `sudo docker image prune` ``
  `` `sudo docker image remove <imageID>` ``
  `` `sudo docker image remove $(sudo docker images -q)` ``
  `` `sudo docker container stop $(sudo docker ps -q)` ``

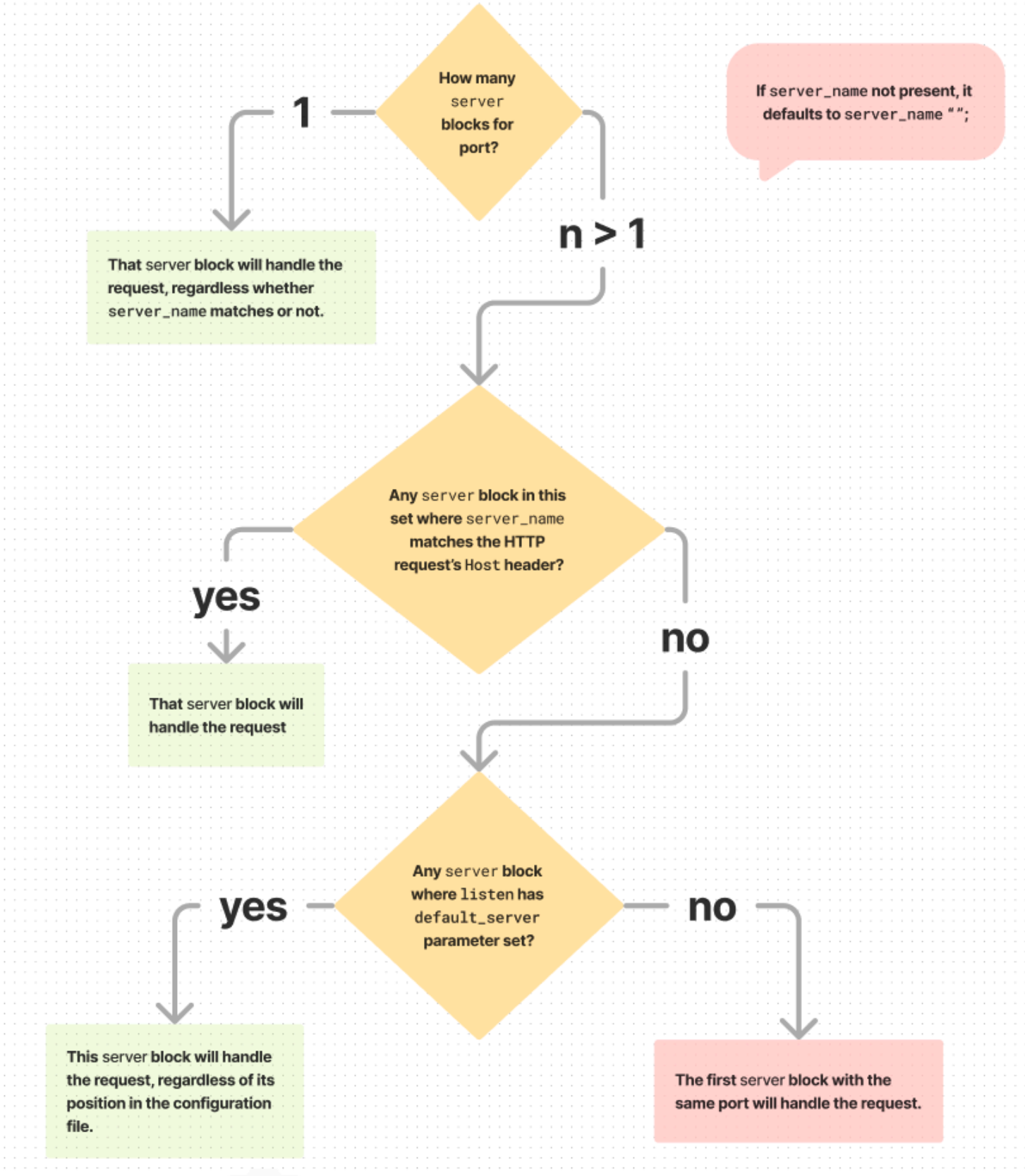
Alternative Port Reassignment Option for Node projects

Change the hard-coded port to run additional node projects at the same time locally

- Search for all instances of port 3000 and update to a new port (*eg. to port 3001*)

  `` `**docker-compose.yml** : line 8` = `- "3001:3001"` ``
  `` `**env.js** : line 2` = `localhost:**3001**` ``
  (*server side*) `` `**main.js** : line 10` = `|| **3001**` ``

NGINX Routing logic & Default Redirection Config



- Add the below highlighted config, customized to your domain, to the END of your NGINX config file to redirect all queries to undefined paths

  NOTE: May break your locally run copy's access to SQL.. but still works on deployment
  Need to do some testing

```
server {
```

```
    listen 443 default_server;
    listen [::]:443 default_server;
    server_name koreangeekman.dev koreangeekman.net;
    ssl_certificate /etc/ssl/cert.pem;
    ssl_certificate_key /etc/ssl/key.pem;
    return 301 https://koreangeekman.dev;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    return 301 https://koreangeekman.info;
}
```

- **Troubleshooting Note**: Docker hosted images are all running plain HTTP and are not handling anything HTTPS, so the translation needs to be to a plain text port & protocol (HTTP) on the internal side of the nginx reverse proxy

---

APPENDIX T - TROUBLESHOOTING

MySQL DB Unresponsive / No data from DB

1. Determine WHERE it is unresponsive / inaccessible
    a. From the local dev environment (via VSCode) to your EC2 server
        i. [**IF** working on the live deployment but not locally on dev]
            1. Check your NGINX and remove the **`default_server`** tag
            2. Save and restart the NGINX service, then give it a few minutes to clear
        ii. Jump to **[Step 2]**

    b. From a deployed live site
        i. [**IF** from a fresh deployment]
            1. Double-check the connection-string in the env file used in deployment
                `` `cat ~/<app-name>-env` ``
                [**IF** incorrect values] Update your Github Secrets and redeploy

        ii. [**IF** from an already open tab (potentially still running off cached page)]
        Assuming the website itself is still fully responsive.. (verify with a full page refresh)
            1. Check another app that also uses the SQL DB

        iii. [**IF** the page refresh fails to load the website itself]
        Your EC2 server itself probably crashed (since it hosts both the SQL DB & Website)
            1. Jump to **[Step 3]**

    c. [**IF** you recently had to rebuild your server]
    Double-check that you've added all your SQL Users **&&** DBs **&&** Tables back in

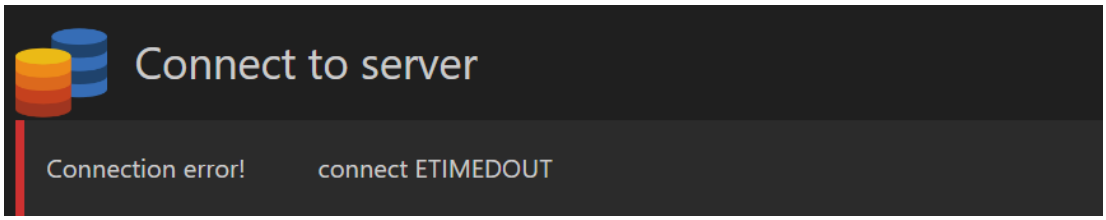
(*Also, the network section of your browser's Dev Tools can often reveal missing tables,etc*)

2. Determine HOW it is unresponsive / inaccessible

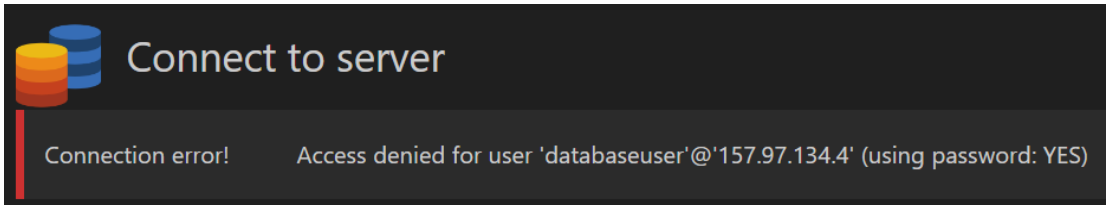
    a. Setup your connection via the VSCode MySQL extension [**SQL Server Client(mssql)**]

    b. Check from the reaction from your connection attempt

        i. Timeout



        ii. Access denied



- [**IF** Timeout]
    - Assuming the service was accessible before: Jump to **[Step 3]**

- [**IF** Access denied]
    - Verify your MySQL credentials
        - Ensure you can access from the extension as a 3rd party test
            - Once that is functional, update related connection strings
        - If additional DB access troubleshooting is required:
            - Refer to **MySQL Remote User Access Troubleshooting** section

    - Where to update:
        - [**IF** Live / Deployed project]
            - **`ENV_FILE`** contents in Github Secrets
        - [**IF** Offline/Dev/Local project]
            - **`[ node ] .env`** - connection string
            - **`[dotnet] appsettings.Development.json -`** connection string

3. Check your EC2 instance status of your MySQL server on your AWS Console (https://console.aws.amazon.com)
    - **EC2** > **Instances** > [*open into the instance details*]

    - Find the ` **Status and alarms** ` tab

    - [**IF** both status checks are reported as '**Running**']
        - Double-check your security
        - Skip to the next step: **[Step 4]**

    - [**IF** either status check is not in a green state]
        - Attempt a reboot on your EC2 instance

- Click on ` **Instance state** ` >> ` **Reboot instance** `
  NOTE: This may take a few minutes to succeed, if it will

4. Attempt to log in to your EC2 server
   ` **ssh -i <*ec2server.pem*> ubuntu@EC2_IP_ADDRESS** `

   - [**IF** the login attempt timed out]
     - Attempt a reboot on your EC2 instance
     - Click on ` **Instance state** ` >> ` **Reboot instance** `
       NOTE: This may take a few minutes to succeed, if it will
       IF after ~5-10 min the server is still unresponsive, jump to Stopping the instance

5. [**IF** able to login to the EC2 server]
   a. Check on the MySQL service status
      ` **systemctl status mysql** ` (*should also have logs at the end of the reply*)

```
ubuntu@ip-172-31-26-93:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2024-01-03 07:25:19 UTC; 2 days ago
   Main PID: 511 (mysqld)
     Status: "Server is operational"
      Tasks: 43 (limit: 1121)
     Memory: 141.4M
        CPU: 13min 32.402s
     CGroup: /system.slice/mysql.service
             └─511 /usr/sbin/mysqld

Jan 03 07:25:14 ip-172-31-26-93 systemd[1]: Starting MySQL Community Server...
Jan 03 07:25:19 ip-172-31-26-93 systemd[1]: Started MySQL Community Server.
ubuntu@ip-172-31-26-93:~$
```

      - [**IF** flopping between active and inactive with uptime < 30sec]
        - Stop the mysql service
          ` **sudo systemctl stop mysql** `
        - Something is preventing it from successfully starting/staying live so look to the logs and troubleshoot **insert magic fix - see an instructor if unable to determine cause**
      - [**IF NOT** active (running)] Continue to 5.b
      - [**IF** active (running)] Jump to 5.c

   b. Re-enable the mysql service
      - From the EC2 terminal
        ` **sudo systemctl start mysql** `
      - Check for any logs that can point to errors/issues preventing startup
      - Re-check the mysql service status

   c. Check on DB accessibility locally from the server - login to mysql
      - Log in to MySQL
        ` **sudo mysql -p** `
      - Issue a command to see if mysql is responsive beyond just the shell
        ` **SELECT user,host FROM mysql.user;** `
      - Issue some kind of query to the targeted DB
        ` **SELECT id,name,email FROM keepr.accounts;** ` (*as the local root account*)

6. [**IF** (unfortunately) all else fails] Rebuild your EC2 instance

---

MySQL Remote User Access Troubleshooting

- Edit your connection via the VSCode MySQL extension [**SQL Server Client(mssql)**]
    - Note which user [and database] is configured
    - Double-check connection ability from the extension
        - [**IF** Timeout]
            - Double-check your server IP matches
            - Make sure the MySQL service is still active
            - Double-check MySQL daemon's bind-address set to allow all (*dev env only*)
            - Double-check your EC2 instance Security Group allows MySQL @ 3306

        - [**IF** Access Denied]
            - Re-verify your remote SQL User/Password combo
            - Reset the passwords on the SQL user account entries (x2)
            - Double-check granted privileges

        - [**ELSE**]
            - Set up a new user in MySQL from the server terminal and test with that
                - (5x commands)

- See **APPENDIX A - Useful MySQL Commands** for supporting informational queries

---

EC2 Instance Crashed & Unresponsive

- Possible causes:
    - OOM (Out Of Memory)
        - SQL defaults use around 400 MB, disable performance trackers to reduce
        - SQL can still use more RAM when multiple users access at the same time
    - Out of Storage/Drive space
        - Need to monitor space usage and set a minimum - 10GB isn't enough b/c of SQL
        - Enable undo file truncations & reduce log period
        - Update Dockerfiles to use smaller deployment images

- [**IF** you had to **stop** / **force stop** your instance **&& IF** you don't have an Elastic IP **($)**]
    - Your server **will** have a new public IP

- There are up to 3 places we need to update with the new IP:

    - Local Machine's SSH connection string to access the EC2 server
      `` `ssh -i <key.pem> ubuntu@<newIPaddress>` ``

    - Github Secrets:
        - `EC2_IP_ADDRESS`

- `ENV_FILE` (*in the connection string of your C#/MySQL projects*)

- To avoid re-entering a new public IP every time, you can use the "*Elastic IP*" service **($)**
  *** this will no longer be free starting Feb.1, 2024*
  - $0.005 /hr charges will be applied [~$44/year or ~$3.65/month per IP]

EC2 Reboot || Stop/Start - Web apps unreachable

- Unless the ` **`restart: unless-stopped`** ` condition was enabled on build, you will need to either manually start up your app instances or re-run the "Deploy to EC2" action on Github.

- To enable auto-instance restarts on server boots:

  - [**IF** your docker instances are already up and running]
    Use this command to update the run state and include an auto-restart of the instance on every active instance as a pre-emptive config change

    ` **`sudo docker update --restart unless-stopped $(sudo docker ps -q)`** `

  - [**IF** you had to reboot or stop/start the instance]
    Add the condition to the compose file then start it back up.
    (*Also add it to your docker-compose.yml in your project files so it's there for future builds*)

    ` **`sudo nano ~/`**<repoName>**`-compose`** `
    Add ` **`restart: unless-stopped`** ` to the end of the file, save and exit
    ` **`sudo docker-compose -p`** <repoName> **`-f`** <repoName>**`-compose up -d`** `

EC2 disk space full

MySQL or your Docker images may fill up your drive faster than thought!
*This is the process to extend your storage space on a linux EC2 instance (a **two-part** process)*

[**IF** completely at 0 space or not enough to process the below steps for expansion]

- Start by checking on your docker images list with **`sudo docker images`**
  [**IF** more than one copy per app] Run **`sudo docker image prune`**
  This is the quickest way to free up space if this is one of the culprits
  *** Remember to set up recurring scheduled task to auto-prune at least weekly if not daily
  (depending on frequency of changes)
  If this is the only issue, you do not necessarily need to expand your storage array
  (in other words: you can stop here)

- Second option is stopping one of your Docker container instances and removing the container & associated image
  (We can easily rebuild the docker instance and is the safest method to quickly free space)

Open your AWS console in a web browser and navigate to the EC2 services

- Open the EC2 instance details
- Click on the **Storage** tab
- Click into the **VolumeID**
- Click into the VolumeID details
    - **Modify** (in the top right)
    - Enter the new **Size** (in GB)
    - Click **Modify** then **Modify** again (pop-up modal has links to [this guide] for linux)
    - Log in to your EC2 instance to finish the steps like below
      (if anything doesn't match exactly, refer to the above linked guide for your case)

HOW you proceed depends on the TYPE that matches your system..
(*from the EC2 terminal*)

```
ubuntu@ip-172-31-21-230:~$ lsblk
NAME         MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
[...]
nvme0n1      259:0    0     8G  0 disk
├─nvme0n1p1  259:1    0   7.9G  0 part /
└─nvme0n1p15 259:2    0    99M  0 part /boot/efi

---=---

ubuntu@ip-172-31-26-93:~$ lsblk
NAME     MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
[...]
xvda     202:0    0    10G  0 disk
├─xvda1  202:1    0   7.9G  0 part /
├─xvda14 202:14   0     4M  0 part
└─xvda15 202:15   0   106M  0 part /boot/efi
```

THEN:

```
ubuntu@ip-172-31-21-230:~$ sudo growpart /dev/nvme0n1 1
[...]

---OR---

ubuntu@ip-172-31-26-93:~$ sudo growpart /dev/xvda 1
CHANGED: partition=1 start=227328 old: size=16549855 end=16777183 new: size=20744159 end=20971487
```

THEN: (*to verify*)

```
ubuntu@ip-172-31-21-230:~$ lsblk
NAME         MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
[...]
nvme0n1      259:0    0    20G  0 disk
├─nvme0n1p1  259:1    0  19.9G  0 part /
[...]
---=---
ubuntu@ip-172-31-26-93:~$ lsblk
NAME     MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
[...]
```

```
xvda     202:0    0    10G  0 disk
├─xvda1  202:1    0   9.9G  0 part /
[...]
```

THEN: (*check on filesystem type*)

```
ubuntu@ip-172-31-21-230:~$ df -hT
Filesystem      Type   Size  Used Avail Use% Mounted on
/dev/root       ext4     8G  4.2G  3.4G  53% /
[...]
---=---
ubuntu@ip-172-31-26-93:~$ df -hT
Filesystem     Type   Size  Used Avail Use% Mounted on
/dev/root      ext4   7.6G  7.5G   66M 100% /

ubuntu@ip-172-31-21-230:~$ sudo resize2fs /dev/nvme0n1p1
[...]
---OR---
ubuntu@ip-172-31-26-93:~$ sudo resize2fs /dev/xvda1
[...]


ubuntu@ip-172-31-21-230:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       20G  4.2G   16G  22% /
[...]
---=---
ubuntu@ip-172-31-26-93:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       9.6G  7.6G  2.0G  80% /
```

.NET app build issue - Server/API side (solution files)

- [**IF** your .NET project's server side isn't starting up **&&** logs show ' **server.sln** ' file, either missing or multiple]
    - Keep just one **`.sln`** file, if any, only in the root directory
      *(your **`/server`** folder does not need a **`.sln`** file - delete it)*
    - Navigate to your app's **server folder** via cmd line, run ` **code .** ` to open just your server side in VSC
    - **CTRL+P**, then type ` **`>.net`** ` and click on ` **`.NET: Generate Assets for Build and Debug`** `
        - Note: If you look into the file changes under [**Source Control**], it will have updated your ` **`/server/.vscode/tasks.json`** ` file to remove the ` **`server.sln`** ` entries (x3) and insert the **`<appname>.csproj`** in its place

---

Client side build issue on Docker image build

Local `npm run build` may seem to 'work' and local debug runs, but Github actions run on more case sensitive environments

ie:

> [linux/amd64 client-builder 6/6] RUN npm run build:
> 1.009 vite v5.0.2 building for production...
> 1.142 transforming...
> 1.709 ✓ 28 modules transformed.
> 1.710 **Could not resolve "./components/NewVaultModal.vue" from "src/App.vue"**
> 1.710 file: /app/client/src/App.vue
> 1.714 error during build:
> 1.714 RollupError: Could not resolve "./components/NewVaultModal.vue" from "src/App.vue"
> 1.714 at error (file:///app/client/node_modules/rollup/dist/es/shared/parseAst.js:337:30)
> 1.714 at ModuleLoader.handleInvalidResolvedId
> (file:///app/client/node_modules/rollup/dist/es/shared/node-entry.js:17935:24)
> 1.714 at file:///app/client/node_modules/rollup/dist/es/shared/node-entry.js:17895:26
> ------
> Dockerfile:17
> --------------------
> 15 |
> 16 | # Build the client-side code
> 17 | >>> RUN npm run build
> 18 |

Check the file-name on the imports match the file name exactly (including case sensitivity)
This MUST be compared to what your **filesystem** sees, not what VSCode tells you

CAUSE: 'Renaming' files from within VSCode by only changing the case on the file names does not update the files themselves with the new case given.
ie. when you rename `**newVaultModal.vue**` to `**NewVaultModal.vue**`

FIX: Match to what your file system sees (Windows Explorer, Finder, Console)
ie. rename all your imported code entries to match the casing of `**newVaultModal.vue**`

-OR- Rename the files from within VSCode by adding/removing characters while changing case or renaming it completely to something new and replacing all used entries to the new name
ie. rename `**newVaultModal.vue**` **to** `**NewVault.vue**` and update all code matches

-OR- Rename the file to something else completely like `**boop.vue**` then back to the name and case you want it to be `**NewVaultModal**`
ie. rename `**newVaultModal.vue**` to `**boop.vue**` then to `**NewVaultModal.vue**`

Full-Stack app just deployed but not loading - fresh setup

- [**IF** You just deployed your app] **&&**
    - the Cloudflare page not found reply **||** the page says redirected too many times
        - Check that **FULL (STRICT)** mode enabled for your domain's SSL/TLS settings
          *See **STAGE 3, STEP 5***

Landing Page via domain name to *.github.io not loading

- [**IF** First time setup **&&** going to **`<username>.github.io`** directly works]
    - Verify CNAME record in Cloudflare w/ root/apex record pointing to **`<username>.github.io`**
    - Go to your github repository > Settings > Pages > add your domain to **Custom Domain**

APPENDIX U - UPDATED FILES

In some instances, it may be simpler to rebuild a project entirely with the updated template by copying over all of your custom code into the new template instead of troubleshooting the old to fit the new

---

BCW Alumni Pre- Fall-23 cohort

- For older versions of the **full-stack templates**, rename the folder structure to **client** & **server**

- For older copies of your **`build`** & **`deploy`** file contents, see/copy/replace with what has changed from the templates repos
    - https://github.com/codeworks-templates/workflow_docker_ec2/tree/main/.github/workflows

- For updated **`Dockerfile`** & **`docker-compose.yml`** file contents, check out the related template repos and pull them from the respective project types
    - https://github.com/codeworks-templates
    - For more info, see **APPENDIX A - CODEWORKS TEMPLATES & BCW**

IF You have apps from older templates, check on these files:

1. **CODE-WORKSPACE**: `` `/code-workspace` ``
    a. [*conditional*] Add this code to the folders list in the code-workspace document if not there

    `` `, { "path": ".github/workflows", "name": "workflows" }` ``

2. **DOCKER-COMPOSE**: `` `/server/docker-compose.yml` ``

a. [*conditional*] Add this as the last line `` `restart: unless-stopped` ``
to auto-start container instances in case of recoverable failures/reboots

5. **DOCKERFILE**: `` `/server/Dockerfile` ``

   a. (*DOTNET APP*) Verify contents of `**Dockerfile**` on line 39

      The name of the `**.dll**` needs to match exactly the **app-name** given upon creation

      You can verify it against a namespace declaration in a controller/service/repo file
      eg. `` `CMD ASPNETCORE_URLS=http://*:$PORT dotnet Tower.dll` ``
      Matches spelling/case exactly from `` `namespace Tower.Controllers;` ``

APPENDIX V - VISUAL DIAGRAMS (TBD/WIP)

To aide in a better understanding of what is going on with the code or behind the scenes

---

Full Stack Web App (File-Structure) **

File organization (dev) vs (prod)

Node.js vs .NET project (Content differences + similarities)

npm i ~= dotnet restore (technically not required if using dotnet publish? Needs verification)
npm run build ~= dotnet publish

Docker (Containers; build & deploy from EC2/Dockerhub high-level)

The layering/processing/functionality

Dockerfile (Build File)

How the file processes https://www.figma.com/file/fqLmy20tGxnUXBOW3v7dWV/Understanding-the-Dockerfile
Stages, Multi-Arch, Parallel processing
Deciding on which packages to use and when (.NET SDK vs ASP.NET vs .NET Runtime)
Shared data constants - external folder inclusion

Github Actions (Build Files)

How the files work and get called

MySQL Connectivity (Network)

VSC MySQL extension
Node API
.NET API

ubuntu@ec2 terminal via root

EC2 Server (from hardware to software to network edges)

Physical server unit > VMs :: Network :: EBS
[Potential service interconnections] like load-balancing, S3, etc
Core > Distribution > Access layers > racks of servers
https://www.figma.com/file/KyqrPDcQkioxjGyDMpY4rF/Deploy-to-Cloud-Overview-for-BCW

EC2 Connectivity (Network - logical)

You > ssh > terminal
User > [80] : 443 > https web frontend
Docker container internals (front-end > back-end > mysql | mongoDB)
    Config file examples surrounding container?, highlighted and line-drawn to show associations
Github Actions
Dockerhub < (pull from)
Docker containers (generalized)
NGINX routing
Elastic IP
EBS Storage (network bound/attached storage block)

APPENDIX Z - zINFORMATIONAL (extra data from testing)

Documenting details from testing

DOCKER IMAGE RESIZING:

Changing the node env to `node:20-slim` trimmed almost 1GB off

For .NET (https://mcr.microsoft.com/en-us/product/dotnet/sdk/tags / https://hub.docker.com/_/microsoft-dotnet-sdk)
https://github.com/dotnet/dotnet-docker/blob/main/documentation/image-variants.md

Odd thing: `dotnet/sdk:8.0-bookworm-slim-amd64` runs the same size as just plain `dotnet/sdk:8.0` ... Where's the slim??? 😅

Size	Image Name
848 MB	dotnet/sdk:8.0
848 MB	dotnet/sdk:8.0-bookworm-slim-amd64 ……???
827 MB	dotnet/sdk:8.0-jammy-amd64
700 MB	dotnet/sdk:8.0-alpine-amd64 (docker image)
700 MB	dotnet/sdk:8.0-alpine (manifest list ^ includes the above)
260 MB	dotnet/aspnet:8.0
118 MB	dotnet/aspnet:8.0-alpine

Final image sizes: (compared to original sizes: 1.1GB+ from **node** and ~850MB+ from **dotnet**)

```
REPOSITORY                  TAG       IMAGE ID       CREATED         SIZE
koreangeekman/keepr         latest    825e561b2096   5 hours ago     118MB
koreangeekman/topomodoro    latest    0bf8aea0b113   10 hours ago    277MB
```

EC2 SERVER RAM MANAGEMENT:

(We can't run more than two apps at a time due to the < 1GB RAM of the server and MySQL taking up ~400MB by itself.. And even then it can crash)

MySQL using about 38%+ of the RAM; both node + .net instances only using around 7-9% RAM, potential variant up to 12% (*just in their immediate environment, does not account for the sub-layers and other container processes*)
Expect each app to typically use around 100 MB of RAM at minimum

BEFORE

```
top - 20:41:51 up 5 min,  1 user,  load average: 0.38, 0.25, 0.12
Tasks: 107 total,   1 running, 106 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    949.7 total,     69.6 free,    628.9 used,    251.2 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    151.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    507 mysql     20   0 1325036 362924   6016 S   0.3  37.3   0:02.84 mysqld
   1213 root      20   0 1279812  84636  25984 S   0.0   8.7   0:03.23 node
   1219 root      20   0  260.4g  55936  44928 S   0.3   5.8   0:00.73 dotnet
    754 root      20   0 1416044  31680   9984 S   0.0   3.3   0:02.79 dockerd
    152 root      rt   0  289312  27392   8960 S   0.0   2.8   0:00.03 multipathd
    113 root      19  -1   97300  21676  20652 S   0.0   2.2   0:00.44 systemd-journal
    829 root      20   0 1356264  19672   6784 S   0.0   2.0   0:01.75 containerd
    392 root      20   0 1245376  18560   9344 S   0.0   1.9   0:01.72 snapd
    380 root      20   0   33080  13304   4480 S   0.0   1.4   0:00.10 networkd-dispat
    498 root      20   0  110096  13184   4992 S   0.0   1.4   0:00.06 unattended-upgr
    327 systemd+  20   0   25532  10852   6528 S   0.0   1.1   0:00.11 systemd-resolve
   1367 root      20   0   16920  10624   8576 S   0.0   1.1   0:00.02 sshd
      1 root      20   0  166216  10600   7400 S   0.0   1.1   0:04.27 systemd
    384 root      20   0 1167568  10308   2688 S   0.0   1.1   0:00.85 amazon-ssm-agen
   1174 root      20   0  719820   9568   7040 S   0.0   1.0   0:00.13 containerd-shim
   1173 root      20   0  719820   9472   6912 S   0.0   1.0   0:00.13 containerd-shim
```

AFTER

```
top - 05:17:05 up  1:27,  1 user,  load average: 0.09, 0.04, 0.01
Tasks: 107 total,   1 running, 106 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    949.7 total,     70.1 free,    442.4 used,    437.2 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    320.4 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6567	mysql	20	0	1015016	151128	29696	S	0.0	15.5	0:14.11	mysqld
1209	root	20	0	260.4g	117708	84096	S	0.0	12.1	0:04.55	dotnet
1215	root	20	0	1282224	79072	16384	S	0.0	8.1	0:06.04	node
112	root	19	-1	103400	35500	34476	S	0.0	3.7	0:00.66	systemd-journal
760	root	20	0	1416044	30592	8832	S	0.0	3.1	0:04.48	dockerd
151	root	rt	0	289312	27392	8960	S	0.0	2.8	0:00.30	multipathd
380	root	20	0	1245376	19880	9984	S	0.0	2.0	0:02.20	snapd
828	root	20	0	1356264	15584	2816	S	0.0	1.6	0:02.94	containerd
373	root	20	0	33080	14840	6016	S	0.0	1.5	0:00.10	networkd-dispat
376	root	20	0	1167568	14016	6272	S	0.0	1.4	0:01.16	amazon-ssm-agen
495	root	20	0	110096	12928	4736	S	0.0	1.3	0:00.07	unattended-upgr
1374	root	20	0	16920	10240	8192	S	0.0	1.1	0:00.03	sshd
1	root	20	0	166336	9712	6384	S	0.0	1.0	0:04.41	systemd
325	systemd+	20	0	25532	9572	5248	S	0.0	1.0	0:00.15	systemd-resolve
475	www-data	20	0	57444	9084	5632	S	0.0	0.9	0:00.11	nginx

MySQL variable allocations after changes

key_buffer_size	8.000 MB
query_cache_size	0.000 MB
innodb_buffer_pool_size	64.000 MB
innodb_additional_mem_pool_size	0.000 MB
innodb_log_buffer_size	16.000 MB
BASE MEMORY	88.000 MB
sort_buffer_size	0.250 MB
read_buffer_size	0.125 MB
read_rnd_buffer_size	0.250 MB
join_buffer_size	0.250 MB
thread_stack	0.125 MB
binlog_cache_size	0.031 MB
tmp_table_size	16.000 MB
MEMORY PER CONNECTION	17.031 MB
Max_used_connections	1
max_connections	8
TOTAL (MIN)	105.031 MB
TOTAL (MAX)	224.250 MB

FYI on Multi-Arch Docker image deployment:

koreangeekman/tower ☆

By koreangeekman • Updated 2 hours ago

Image

Manage Repository

Pulls 18

Overview **Tags**

Sort by Newest ▾ Filter Tags

TAG

latest

Last pushed **2 hours ago** by koreangeekman

`docker pull koreangeekman/tower:…`

DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
d60c7a334dd5	linux/amd64	2 hours ago	93.96 MB
4bdde80d435c	linux/arm64	an hour ago	93.99 MB

TAG

latest

Last pushed **an hour ago** by koreangeekman

`docker pull koreangeekman/keepr:…`

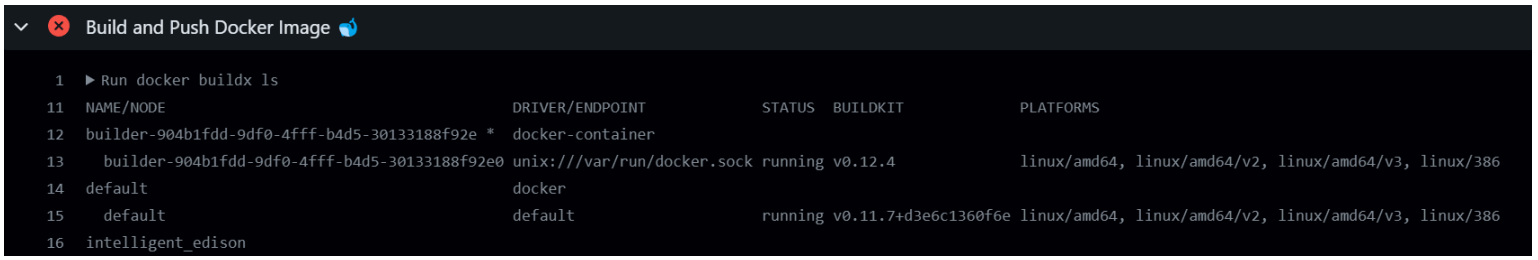
DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
3688658f9f5a	linux/amd64	42 minutes ago	50.08 MB
eedfa1b814cf	linux/arm64	an hour ago	48.18 MB

Open your Dockerhub repo, click on Tags, and you can see which cpu architecture is supported.
Our original Dockerfile/Build config only supported the **amd64** image
AMD64 builds in < 1min with the **node:20-slim** image
Adding the ARM64 image and building both takes ~3.5min to build.. need to improve that
Force amd64 with "`FROM --platform=linux/amd64 node:20-slim AS client-builder`" to reduce processing time back closer to familiar levels but still about 1-2min

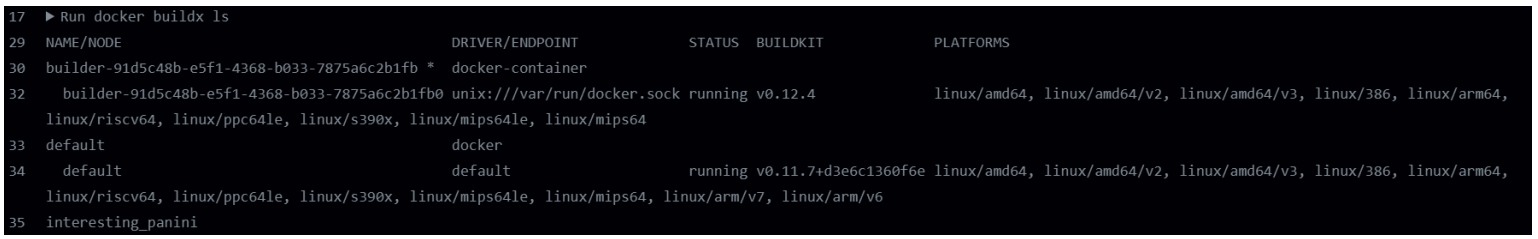
Building ARM64 by itself on .NET ~3min..

When the client-build attempts to run on arm64 versions, it takes an additional 2.5+ minutes to process
Forced the client-build to be only amd64 on both architecture image revisions and entire build process back down under 1-2min.

(*during trial & error stages*)
Testing arm64 builds - quemu emulators required?



With quemu enabled pre-build via github build file
`` `docker run -it --rm --privileged tonistiigi/binfmt --install all` ``



FYI on pruning docker images:

```
ubuntu@ip-172-31-21-230:~$ sudo docker pull koreangeekman/topomodoro:latest
latest: Pulling from koreangeekman/topomodoro
24e221e92a36: Already exists
dbc596c5cca5: Already exists
5ca22430de13: Already exists
328a7946450d: Already exists
9e60e595b46e: Already exists
277901863a8e: Pull complete
fcefe57a161e: Pull complete
88e9a4f5f1c4: Pull complete
546078c1a37f: Pull complete
bfb3de08df8f: Pull complete
Digest: sha256:ed16c3a996a5a8a60595f0947b827d53d868f71415b087f4edae786908b6eaaf
Status: Downloaded newer image for koreangeekman/topomodoro:latest
docker.io/koreangeekman/topomodoro:latest
ubuntu@ip-172-31-21-230:~$
```

```
ubuntu@ip-172-31-21-230:~$ sudo docker images
REPOSITORY                TAG       IMAGE ID       CREATED              SIZE
koreangeekman/topomodoro   latest    6bab450fa3f3   About a minute ago   299MB
koreangeekman/topomodoro   <none>    2b74467756f6   12 minutes ago       299MB
koreangeekman/tower        latest    ed2c4f3738c5   15 hours ago         298MB
ubuntu@ip-172-31-21-230:~$ sudo docker-compose -p topomodoro -f ~/topomodoro-compose up -d
[+] Running 1/1
 ✓ Container topomodoro-app-1  Started
ubuntu@ip-172-31-21-230:~$
ubuntu@ip-172-31-21-230:~$ sudo docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: koreangeekman/topomodoro@sha256:6606397304def20ae45ee706a9c7ef64ba9eb19d37d84587d30b3a2ca0c48e83
deleted: sha256:2b74467756f691832440d98e39fc4f9b7bbe6a6b84c9a1223531dc88ec7abe87
deleted: sha256:e376291239c3acb548969da1af0a76366b57890f51ac0c62050e3f524e7b1f00
deleted: sha256:e9d440ca6b6dab6bff318af3af0f899b32cdf6384f5d8e907d4ade8bb590962a
deleted: sha256:8e112f1bcecb78b7e719a74608626b5c4011011cb81cbeae782974cc9f80443b
deleted: sha256:da93b4edf7e1eacd571183c91f83608c44fa4279f2b86a6e60c38bc4e12dfe57
deleted: sha256:05a343d5a52858e0d3b06f28573aca0ca367ac254f0310632bdbb50435df2821

Total reclaimed space: 76.09MB
ubuntu@ip-172-31-21-230:~$ sudo docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
koreangeekman/topomodoro   latest    6bab450fa3f3   2 minutes ago    299MB
koreangeekman/tower        latest    ed2c4f3738c5   15 hours ago     298MB
ubuntu@ip-172-31-21-230:~$
```

One minor change and rebuild, pulled latest (each “299MB” but only downloaded half the stack), rebuilt with latest, then pruned image stack: reclaimed 76MB from flattening/merging the second image [layer set]. Each image is not the full image ‘size’ listed.

Logging space failure (man-db & logrotate)

```
ubuntu@ip-172-31-21-230:~$ systemctl status
● ip-172-31-21-230
    State: degraded
     Jobs: 1 queued
```

```
   Failed: 2 units
    Since: Mon 2024-01-15 20:09:20 UTC; 2 weeks 6 days ago
[...]

ubuntu@ip-172-31-21-230:~$ systemctl --failed
  UNIT              LOAD   ACTIVE SUB    DESCRIPTION
● logrotate.service loaded failed failed Rotate log files
● man-db.service    loaded failed failed Daily man-db regeneration


ubuntu@ip-172-31-21-230:~$ systemctl status logrotate
× logrotate.service - Rotate log files
     Loaded: loaded (/lib/systemd/system/logrotate.service; static)
     Active: failed (Result: exit-code) since Mon 2024-02-05 18:25:31 UTC; 10 hr ago
TriggeredBy: ● logrotate.timer
       Docs: man:logrotate(8)
             man:logrotate.conf(5)
    Process: 478837 ExecStart=/usr/sbin/logrotate /etc/logrotate.conf (code=exited,
status=226/NAMESPACE)
   Main PID: 478837 (code=exited, status=226/NAMESPACE)
        CPU: 1ms

Feb 05 18:25:31 ip-172-31-21-230 systemd[1]: Starting Rotate log files...
Feb 05 18:25:31 ip-172-31-21-230 systemd[478837]: logrotate.service: Failed to set up mount
namespacing: /run/systemd/unit-root/dev: No space left on device
Feb 05 18:25:31 ip-172-31-21-230 systemd[478837]: logrotate.service: Failed at step NAMESPACE
spawning /usr/sbin/logrotate: No space left on device
Feb 05 18:25:31 ip-172-31-21-230 systemd[1]: logrotate.service: Main process exited,
code=exited, status=226/NAMESPACE
Feb 05 18:25:31 ip-172-31-21-230 systemd[1]: logrotate.service: Failed with result 'exit-code'.
Feb 05 18:25:31 ip-172-31-21-230 systemd[1]: Failed to start Rotate log files.

ubuntu@ip-172-31-21-230:~$ systemctl status man-db
× man-db.service - Daily man-db regeneration
     Loaded: loaded (/lib/systemd/system/man-db.service; static)
     Active: failed (Result: exit-code) since Mon 2024-02-05 18:26:09 UTC; 10hr ago
TriggeredBy: ● man-db.timer
       Docs: man:mandb(8)
    Process: 478862 ExecStart=/usr/bin/install -d -o man -g man -m 0755 /var/cache/man
(code=exited, status=0/SUCCESS)
    Process: 478863 ExecStart=/usr/bin/find /var/cache/man -type f -name *.gz -atime +6 -delete
(code=exited, status=226/NAMESPACE)
   Main PID: 478863 (code=exited, status=226/NAMESPACE)
        CPU: 8ms

Feb 05 18:26:09 ip-172-31-21-230 systemd[1]: Starting Daily man-db regeneration...
Feb 05 18:26:09 ip-172-31-21-230 systemd[478863]: man-db.service: Failed to set up mount
namespacing: /run/systemd/unit-root/dev: No space left on device
Feb 05 18:26:09 ip-172-31-21-230 systemd[478863]: man-db.service: Failed at step NAMESPACE
spawning /usr/bin/find: No space left on device
```

```
Feb 05 18:26:09 ip-172-31-21-230 systemd[1]: man-db.service: Main process exited, code=exited,
status=226/NAMESPACE
Feb 05 18:26:09 ip-172-31-21-230 systemd[1]: man-db.service: Failed with result 'exit-code'.
Feb 05 18:26:09 ip-172-31-21-230 systemd[1]: Failed to start Daily man-db regeneration.
```

After ‘sudo reboot’

cannot create temporary directory for the root file system: No space left on device

```
ubuntu@ip-172-31-21-230:~$ systemctl status
● ip-172-31-21-230
    State: starting
     Jobs: 7 queued
   Failed: 9 units
    Since: Mon 2024-02-05 18:39:23 UTC; 20min ago
```

```
ubuntu@ip-172-31-21-230:~$ systemctl --failed
  UNIT                                      LOAD   ACTIVE SUB    DESCRIPTION
● cloud-config.service                      loaded failed failed Apply the settings specified in cloud-config
● cloud-init-local.service                  loaded failed failed Initial cloud-init job (pre-networking)
● cloud-init.service                        loaded failed failed Initial cloud-init job (metadata service crawler)
● snap.docker.dockerd.service               loaded failed failed Service for snap application docker.dockerd
● snap.docker.nvidia-container-toolkit.service loaded failed failed Service for snap application docker.nvidia-container-toolkit
● snap.lxd.activate.service                 loaded failed failed Service for snap application lxd.activate
● snapd.seeded.service                      loaded failed failed Wait until snapd is fully seeded
● systemd-hostnamed.service                 loaded failed failed Hostname Service
● systemd-resolved.service                  loaded failed failed Network Name Resolution
```

```
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1018-aws aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Jan 25 22:27:15 UTC 2024

  System load:                      0.0
  Usage of /:                       36.6% of 9.52GB   (was not properly reporting)
  Memory usage:                     43%
  Swap usage:                       0%
  Processes:                        158
  Users logged in:                  0
  IPv4 address for br-36c8ec28ea99: 172.22.0.1
  IPv4 address for br-66961326ed07: 172.20.0.1
  IPv4 address for br-998210b1e8b4: 172.18.0.1
  IPv4 address for br-a569365596eb: 172.19.0.1
  IPv4 address for br-c17b47b733f1: 172.21.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for ens5:            172.31.21.230


Expanded Security Maintenance for Applications is not enabled.
```

```
15 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Mon Feb  5 18:56:10 2024 from 18.237.140.164
```

```
ubuntu@ip-172-31-21-230:~$ df -h
Filesystem       Size  Used Avail Use% Mounted on
/dev/root        9.6G  9.6G     0 100% /
tmpfs            924M     0  924M   0% /dev/shm
tmpfs            370M   29M  341M   8% /run
tmpfs            5.0M     0  5.0M   0% /run/lock
/dev/nvme0n1p15   98M  6.3M   92M   7% /boot/efi
tmpfs            185M  4.0K  185M   1% /run/user/1000

ubuntu@ip-172-31-21-230:~$ sudo du -cha / | grep -E "^[0-9\.]*[G]"
1.5G    /usr
2.4G    /var/lib/mysql/undo_001
2.6G    /var/lib/mysql/mysql.ibd
5.7G    /var/lib/mysql
6.7G    /var/lib
1.1G    /var/snap/docker/common/var-lib-docker
1.1G    /var/snap/docker/common
1.1G    /var/snap/docker
1.1G    /var/snap
8.0G    /var
2.8G    /snap
13G     /
13G     total
```

After rebuilding mysql from scratch:

(baseline of file sizes)

```
ubuntu:~$ sudo du -cha --max-depth=3 /var | grep -E "^[0-9\.]*[MG]"
2.5M    /var/log/amazon
4.3M    /var/log/btmp.1
1.6M    /var/log/syslog.1
161M    /var/log/journal/ec29acff36ddf885e4a78978e648851b
161M    /var/log/journal
173M    /var/log
3.2M    /var/lib/command-not-found
4.0M    /var/lib/ubuntu-advantage
1.7M    /var/lib/mysql/performance_schema
16M     /var/lib/mysql/undo_001
```

```
8.2M    /var/lib/mysql/#ib_16384_1.dblwr
101M    /var/lib/mysql/#innodb_redo
12M     /var/lib/mysql/ibdata1
16M     /var/lib/mysql/undo_002
27M     /var/lib/mysql/mysql.ibd
12M     /var/lib/mysql/ibtmp1
195M    /var/lib/mysql
31M     /var/lib/dpkg
154M    /var/lib/apt
91M     /var/lib/mecab
276M    /var/lib/snapd/seed
485M    /var/lib/snapd/snaps
762M    /var/lib/snapd
1.3G    /var/lib
4.1M    /var/cache/debconf
31M     /var/cache/apt/archives
44M     /var/cache/apt/srcpkgcache.bin
44M     /var/cache/apt/pkgcache.bin
117M    /var/cache/apt
1.3M    /var/cache/snapd
3.0M    /var/cache/apparmor/30d07b40.0
1.9M    /var/cache/apparmor/a4dd844e.0
4.8M    /var/cache/apparmor
1.3M    /var/cache/man
129M    /var/cache
975M    /var/snap/docker
975M    /var/snap
1.8M    /var/backups
2.5G    /var
2.5G    total
```

Seeing same issue on my AMD64 system:

```
ubuntu@ip-172-31-26-93:~$ sudo du -cha --max-depth=3 /var | grep -E "^[0-9\.]*[MG]"
4.0M    /var/lib/ubuntu-advantage/apt-esm
4.0M    /var/lib/ubuntu-advantage
902M    /var/lib/snapd/snaps
41M     /var/lib/snapd/seed
945M    /var/lib/snapd
1.3G    /var/lib/mysql/undo_001
12M     /var/lib/mysql/ibdata1
9.1M    /var/lib/mysql/sys
1.7M    /var/lib/mysql/performance_schema
97M     /var/lib/mysql/#innodb_redo
353M    /var/lib/mysql/undo_002
1.4G    /var/lib/mysql/mysql.ibd
8.2M    /var/lib/mysql/#ib_16384_1.dblwr
3.1G    /var/lib/mysql
166M    /var/lib/apt/lists
166M    /var/lib/apt
```

```
91M     /var/lib/mecab/dic
91M     /var/lib/mecab
3.4M    /var/lib/command-not-found/commands.db
3.4M    /var/lib/command-not-found
34M     /var/lib/dpkg/info
35M     /var/lib/dpkg
4.3G    /var/lib
1.8M    /var/backups
1.9M    /var/log/cloud-init.log
1.3M    /var/log/syslog
129M    /var/log/journal/1f7b9499923b4c32a9c0fcc9f8ae6b06
129M    /var/log/journal
1.7M    /var/log/amazon/ssm
1.7M    /var/log/amazon
138M    /var/log
3.3M    /var/cache/snapd/commands.db
3.4M    /var/cache/snapd
3.0M    /var/cache/apparmor/30d07b40.0
1.8M    /var/cache/apparmor/a4dd844e.0
4.7M    /var/cache/apparmor
11M     /var/cache/apt/pkgcache.bin.L8ckQu
12M     /var/cache/apt/pkgcache.bin.CiRto9
91M     /var/cache/apt/archives
47M     /var/cache/apt/srcpkgcache.bin
160M    /var/cache/apt
1.5M    /var/cache/man
2.4M    /var/cache/debconf/templates.dat
2.4M    /var/cache/debconf/templates.dat-old
4.8M    /var/cache/debconf
175M    /var/cache
924M    /var/snap/docker/common
924M    /var/snap/docker
924M    /var/snap
5.6G    /var
5.6G    total

ubuntu@ip-172-31-26-93:~$ systemctl status mysql
× mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Thu 2024-02-08 17:52:10 UTC; 1min 7s ago
    Process: 170386 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited,
status=0/SUCCESS)
    Process: 170394 ExecStart=/usr/sbin/mysqld (code=exited, status=1/FAILURE)
   Main PID: 170394 (code=exited, status=1/FAILURE)
     Status: "Server shutdown complete"
      Error: 2 (No such file or directory)
        CPU: 19.072s

ubuntu@ip-172-31-26-93:~$ systemctl status mysql
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
```

```
     Active: activating (start) since Thu 2024-02-08 17:55:49 UTC; 1min 5s ago
    Process: 171268 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited,
status=0/SUCCESS)
   Main PID: 171276 (mysqld)
     Status: "Server upgrade in progress"
      Tasks: 28 (limit: 1121)
     Memory: 313.9M
        CPU: 17.744s
     CGroup: /system.slice/mysql.service
             └─171276 /usr/sbin/mysqld
```

NOTE:
UNABLE to successfully restart MySQL service even after freeing up space and expanding the volume

Reboot pending.. Reboot also does not allow MySQL to start - likely process/data corruption present
Running a re-install of mysql-server to check
$ **sudo apt reinstall mysql-server**
(*hangs at ~43%.. Waiting…still failed*)

```
ubuntu@ip-172-31-26-93:~$ sudo ls -lh /var/lib/mysql
total 3.3G
-rw-r----- 1 mysql mysql 192K Feb  8 18:41 '#ib_16384_0.dblwr'
-rw-r----- 1 mysql mysql 8.2M Feb  8 18:41 '#ib_16384_1.dblwr'
drwxr-x--- 2 mysql mysql 4.0K Feb  8 18:41 '#innodb_redo' (default-database-name?)
drwxr-x--- 2 mysql mysql 4.0K Feb  8 18:41 '#innodb_temp' (default-database-name?)
drwxr-x--- 2 mysql mysql 4.0K Jan  8 02:20  allspice (database-name)
-rw-r----- 1 mysql mysql   56 Dec 18 19:07  auto.cnf
drwxr-x--- 2 mysql mysql 4.0K Dec 24 18:04  bcw_replay (database-name)
-rw-r----- 1 mysql mysql  201 Jan  7 00:00  binlog.000244
-rw-r----- 1 mysql mysql  201 Jan  8 00:00  binlog.000245
-rw-r----- 1 mysql mysql  665 Jan  8 21:54  binlog.000246
-rw-r----- 1 mysql mysql  201 Feb  6 00:00  binlog.000247
-rw-r----- 1 mysql mysql  180 Feb  6 08:05  binlog.000248
-rw-r----- 1 mysql mysql   80 Feb  6 00:00  binlog.index
-rw------- 1 mysql mysql 1.7K Dec 18 19:08  ca-key.pem
-rw-r--r-- 1 mysql mysql 1.1K Dec 18 19:08  ca.pem
-rw-r--r-- 1 mysql mysql 1.1K Dec 18 19:08  client-cert.pem
-rw------- 1 mysql mysql 1.7K Dec 18 19:08  client-key.pem
-rw-r--r-- 1 mysql mysql    0 Feb  8 18:28  debian-5.7.flag
-rw-r----- 1 mysql mysql 4.1K Feb  6 08:05  ib_buffer_pool
-rw-r----- 1 mysql mysql  12M Feb  8 18:41  ibdata1
drwxr-x--- 2 mysql mysql 4.0K Dec 18 19:48  keepr (database-name)
drwxr-x--- 2 mysql mysql 4.0K Feb  8 18:40  mysql (default-database-name)
-rw-r----- 1 mysql mysql 1.5G Feb  8 18:41  mysql.ibd
-rw-r----- 1 mysql mysql    6 Feb  8 18:41  mysql_upgrade_info
drwxr-x--- 2 mysql mysql 4.0K Dec 18 19:08  performance_schema(default-database-name)
-rw------- 1 mysql mysql 1.7K Dec 18 19:08  private_key.pem
-rw-r--r-- 1 mysql mysql  452 Dec 18 19:08  public_key.pem
-rw-r--r-- 1 mysql mysql 1.1K Dec 18 19:08  server-cert.pem
-rw------- 1 mysql mysql 1.7K Dec 18 19:08  server-key.pem
drwxr-x--- 2 mysql mysql 4.0K Dec 18 19:08  sys
```

```
drwxr-x--- 2 mysql mysql 4.0K Dec 20 15:29  topomodoro (database-name)
-rw-r----- 1 mysql mysql 1.4G Feb  8 18:41  undo_001
-rw-r----- 1 mysql mysql 400M Feb  8 18:41  undo_002
```

https://www.tecmint.com/mysql-backup-and-restore-commands-for-database-administration/
(only useful if mysql is responsive..)

Testing recovery by backing up all files in /var/lib/mysql (*via adding to a compressed file*)
(*tar will save even the directory it's pulled from as specified, so do not need to specify full directory on extraction*)

```
$ tar zcf ~/mysql-backup-1.tgz /var/lib/mysql
$ tar xvf ~/mysql-backup-1.tgz -C /
```

```
ubuntu@ip-172-31-26-93:~$ sudo du -cha --max-depth=1 /var/lib/mysql | grep -E "^[0-9\.]*[MG]"
1.4G    /var/lib/mysql/undo_001
12M     /var/lib/mysql/ibdata1
9.1M    /var/lib/mysql/sys
1.7M    /var/lib/mysql/performance_schema
101M    /var/lib/mysql/#innodb_redo
401M    /var/lib/mysql/undo_002
1.5G    /var/lib/mysql/mysql.ibd
12M     /var/lib/mysql/ibtmp1
8.2M    /var/lib/mysql/#ib_16384_1.dblwr
3.4G    /var/lib/mysql
3.4G    total
```

```
-rw-rw-r-- 1 ubuntu ubuntu 698M Feb  8 19:36 mysql-backup-1.tgz
```

----- fresh install -----

```
ubuntu@ip-172-31-26-93:~$ sudo ls -lh /var/lib/mysql
total 90M
-rw-r----- 1 mysql mysql 192K Feb  8 20:11 '#ib_16384_0.dblwr'
-rw-r----- 1 mysql mysql 8.2M Feb  8 20:11 '#ib_16384_1.dblwr'
drwxr-x--- 2 mysql mysql 4.0K Feb  8 20:11 '#innodb_redo'
drwxr-x--- 2 mysql mysql 4.0K Feb  8 20:11 '#innodb_temp'
-rw-r----- 1 mysql mysql   56 Feb  8 20:11  auto.cnf
-rw-r----- 1 mysql mysql  180 Feb  8 20:11  binlog.000001
-rw-r----- 1 mysql mysql  404 Feb  8 20:11  binlog.000002
-rw-r----- 1 mysql mysql  157 Feb  8 20:11  binlog.000003
-rw-r----- 1 mysql mysql   48 Feb  8 20:11  binlog.index
-rw------- 1 mysql mysql 1.7K Feb  8 20:11  ca-key.pem
-rw-r--r-- 1 mysql mysql 1.1K Feb  8 20:11  ca.pem
-rw-r--r-- 1 mysql mysql 1.1K Feb  8 20:11  client-cert.pem
-rw------- 1 mysql mysql 1.7K Feb  8 20:11  client-key.pem
-rw-r--r-- 1 root  root     0 Feb  8 20:11  debian-5.7.flag
-rw-r----- 1 mysql mysql 3.4K Feb  8 20:11  ib_buffer_pool
-rw-r----- 1 mysql mysql  12M Feb  8 20:11  ibdata1
-rw-r----- 1 mysql mysql  12M Feb  8 20:11  ibtmp1
-rw-r----- 1 mysql mysql    5 Feb  8 20:11  ip-172-31-26-93.pid
drwxr-x--- 2 mysql mysql 4.0K Feb  8 20:11  mysql
-rw-r----- 1 mysql mysql  25M Feb  8 20:11  mysql.ibd
```

```
drwxr-x--- 2 mysql mysql 4.0K Feb  8 20:11  performance_schema
-rw------- 1 mysql mysql 1.7K Feb  8 20:11  private_key.pem
-rw-r--r-- 1 mysql mysql  452 Feb  8 20:11  public_key.pem
-rw-r--r-- 1 mysql mysql 1.1K Feb  8 20:11  server-cert.pem
-rw------- 1 mysql mysql 1.7K Feb  8 20:11  server-key.pem
drwxr-x--- 2 mysql mysql 4.0K Feb  8 20:11  sys
-rw-r----- 1 mysql mysql  16M Feb  8 20:11  undo_001
-rw-r----- 1 mysql mysql  16M Feb  8 20:11  undo_002

ubuntu@ip-172-31-26-93:~$ sudo du -cha --max-depth=1 /var/lib/mysql | grep -E "^[0-9\.]*[MG]"
16M     /var/lib/mysql/undo_001
12M     /var/lib/mysql/ibdata1
1.7M    /var/lib/mysql/performance_schema
101M    /var/lib/mysql/#innodb_redo
16M     /var/lib/mysql/undo_002
26M     /var/lib/mysql/mysql.ibd
12M     /var/lib/mysql/ibtmp1
8.2M    /var/lib/mysql/#ib_16384_1.dblwr
192M    /var/lib/mysql
192M    total

-rw-rw-r-- 1 ubuntu ubuntu 698M Feb  8 19:36 mysql-backup-1.tgz
-rw-rw-r-- 1 ubuntu ubuntu 2.0M Feb  8 20:12 mysql-defaults.tgz
```

MySQL logs

```
2024-02-08T17:30:15.595602Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.36-0ubuntu0.22.04.1) starting as process
165667
2024-02-08T17:30:15.655450Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2024-02-08T17:30:16.694537Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 61440 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.694592Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:30:16.694605Z 0 [ERROR] [MY-012639] [InnoDB] Write to file ./#innodb_redo/#ib_redo1501_tmp failed at offset 3145728,
131072 bytes should have been written, only 61440 were written. Operating system error number 28. Check that your OS and file
system support files of this size. Check also that the disk is not full or a disk quota exceeded.
2024-02-08T17:30:16.694618Z 0 [ERROR] [MY-012640] [InnoDB] Error number 28 means 'No space left on device'
2024-02-08T17:30:16.694640Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.713563Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 57344 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.713599Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:30:16.713612Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.728328Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 61440 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.728361Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:30:16.728374Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.747147Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 57344 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.747182Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
```

```
2024-02-08T17:30:16.747261Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.762264Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 61440 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.762296Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:30:16.762309Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.780563Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 57344 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.780601Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:30:16.780618Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.795656Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes should have been written. Only 61440 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:30:16.795689Z 0 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:30:16.795702Z 0 [ERROR] [MY-012888] [InnoDB] Cannot resize redo log file ./#innodb_redo/#ib_redo1501_tmp to 3 MB
(Failed to set size)
2024-02-08T17:30:16.813746Z 0 [Warning] [MY-012637] [InnoDB] 131072 bytes shoul2024-02-08T17:30:47.377511Z 1 [ERROR] [MY-010334]
[Server] Failed to initialize DD Storage Engine
2024-02-08T17:30:47.380329Z 0 [ERROR] [MY-010020] [Server] Data Dictionary initialization failed.
2024-02-08T17:30:47.381247Z 0 [ERROR] [MY-010119] [Server] Aborting
2024-02-08T17:36:04.477640Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.36-0ubuntu0.22.04.1)
(Ubuntu).

2024-02-08T17:36:05.154385Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.36-0ubuntu0.22.04.1) starting as process
166569
2024-02-08T17:36:05.169256Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2024-02-08T17:36:19.836752Z 1 [Warning] [MY-012637] [InnoDB] 1048576 bytes should have been written. Only 24576 bytes written.
Retrying for the remaining bytes.
2024-02-08T17:36:19.843118Z 1 [Warning] [MY-012638] [InnoDB] Retry attempts for writing partial data failed.
2024-02-08T17:36:19.843137Z 1 [ERROR] [MY-012639] [InnoDB] Write to file ./ibtmp1 failed at offset 0, 1048576 bytes should have
been written, only 24576 were written. Operating system error number 28. Check that your OS and file system support files of this
size. Check also that the disk is not full or a disk quota exceeded.
2024-02-08T17:36:19.844038Z 1 [ERROR] [MY-012640] [InnoDB] Error number 28 means 'No space left on device'
2024-02-08T17:36:19.844974Z 1 [ERROR] [MY-012267] [InnoDB] Could not set the file size of './ibtmp1'. Probably out of disk space
2024-02-08T17:36:19.844990Z 1 [ERROR] [MY-012926] [InnoDB] Unable to create the shared innodb_temporary.
2024-02-08T17:36:19.845000Z 1 [ERROR] [MY-012930] [InnoDB] Plugin initialization aborted with error Generic error.
2024-02-08T17:36:20.282168Z 1 [ERROR] [MY-010334] [Server] Failed to initialize DD Storage Engine
2024-02-08T17:36:20.283929Z 0 [ERROR] [MY-010020] [Server] Data Dictionary initialization failed.
2024-02-08T17:36:20.284365Z 0 [ERROR] [MY-010119] [Server] Aborting
2024-02-08T17:36:20.305912Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.36-0ubuntu0.22.04.1)
(Ubuntu).

2024-02-08T18:07:40.490680Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.36-0ubuntu0.22.04.1) starting as process
4030
2024-02-08T18:07:40.527823Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2024-02-08T18:08:15.187138Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2024-02-08T18:08:19.205750Z 4 [System] [MY-013381] [Server] Server upgrade from '80035' to '80036' started.
2024-02-08T18:08:48.816111Z 4 [ERROR] [MY-013178] [Server] Execution of server-side SQL statement '-- Copyright (c) 2015, 2023,
Oracle and/or its affiliates. -- -- This program is free software; you can redistribute it and/or modify -- it under the terms of
the GNU General Public License as published by -- the Free Software Foundation; version 2 of the License. -- -- This program is
distributed in the hope that it will be useful, -- but WITHOUT ANY WARRANTY; without even the implied warranty of --
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the -- GNU General Public License for more details. -- -- You should have
received a copy of the GNU General Public License -- along with this program; if not, write to the Free Software -- Foundation,
Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA -- NOTE: This needs to be replicated within the sys_config_clean.inc
file INSERT IGNORE INTO sys.sys_config (variable, value) VALUES     ('statement_truncate_len', 64),
('statement_performance_analyzer.limit', 100),     ('statement_performance_analyzer.view', NULL),
('diagnostics.allow_i_s_tables', 'OFF'),     ('diagnostics.include_raw', 'OFF'),     ('ps_thread_trx_info.max_length', 65535); '
failed with error code = 1436, error message = 'Thread stack overrun:  12464 bytes used of a 131072 byte stack, and 160000 bytes
needed.  Use 'mysqld --thread_stack=#' to specify a bigger stack.'.
2024-02-08T18:08:48.850136Z 0 [ERROR] [MY-013380] [Server] Failed to upgrade server.
2024-02-08T18:08:48.853797Z 0 [ERROR] [MY-010119] [Server] Aborting
2024-02-08T18:08:50.659475Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.36-0ubuntu0.22.04.1)
(Ubuntu).

ubuntu@ip-172-31-26-93:~$
```

MySQL Vars executable code

```
#!/bin/sh
sudo mysql -p -e "show variables; show status" | awk '
{
VAR[$1]=$2
}
END {
MAX_CONN = VAR["max_connections"]
MAX_USED_CONN = VAR["Max_used_connections"]
BASE_MEM=VAR["key_buffer_size"] + VAR["query_cache_size"] + VAR["innodb_buffer_pool_size"] +
VAR["innodb_additional_mem_pool_size"] + VAR["innodb_log_buffer_size"]
MEM_PER_CONN=VAR["read_buffer_size"] + VAR["read_rnd_buffer_size"] + VAR["sort_buffer_size"] +
VAR["join_buffer_size"] + VAR["binlog_cache_size"] + VAR["thread_stack"] +
VAR["tmp_table_size"]
MEM_TOTAL_MIN=BASE_MEM + MEM_PER_CONN*MAX_USED_CONN
MEM_TOTAL_MAX=BASE_MEM + MEM_PER_CONN*MAX_CONN
printf "+-----------------------------------------+-------------------+\n"
printf "| %40s | %15.3f MB |\n", "key_buffer_size", VAR["key_buffer_size"]/1048576
printf "| %40s | %15.3f MB |\n", "query_cache_size", VAR["query_cache_size"]/1048576
printf "| %40s | %15.3f MB |\n", "innodb_buffer_pool_size",
VAR["innodb_buffer_pool_size"]/1048576
printf "| %40s | %15.3f MB |\n", "innodb_additional_mem_pool_size",
VAR["innodb_additional_mem_pool_size"]/1048576
printf "| %40s | %15.3f MB |\n", "innodb_log_buffer_size",
VAR["innodb_log_buffer_size"]/1048576
printf "+-----------------------------------------+-------------------+\n"
printf "| %40s | %15.3f MB |\n", "BASE MEMORY", BASE_MEM/1048576
printf "+-----------------------------------------+-------------------+\n"
printf "| %40s | %15.3f MB |\n", "sort_buffer_size", VAR["sort_buffer_size"]/1048576
printf "| %40s | %15.3f MB |\n", "read_buffer_size", VAR["read_buffer_size"]/1048576
printf "| %40s | %15.3f MB |\n", "read_rnd_buffer_size", VAR["read_rnd_buffer_size"]/1048576
printf "| %40s | %15.3f MB |\n", "join_buffer_size", VAR["join_buffer_size"]/1048576
printf "| %40s | %15.3f MB |\n", "thread_stack", VAR["thread_stack"]/1048576
printf "| %40s | %15.3f MB |\n", "binlog_cache_size", VAR["binlog_cache_size"]/1048576
printf "| %40s | %15.3f MB |\n", "tmp_table_size", VAR["tmp_table_size"]/1048576
printf "+-----------------------------------------+-------------------+\n"
printf "| %40s | %15.3f MB |\n", "MEMORY PER CONNECTION", MEM_PER_CONN/1048576
printf "+-----------------------------------------+-------------------+\n"
printf "| %40s | %18d |\n", "Max_used_connections", MAX_USED_CONN
printf "| %40s | %18d |\n", "max_connections", MAX_CONN
printf "+-----------------------------------------+-------------------+\n"
printf "| %40s | %15.3f MB |\n", "TOTAL (MIN)", MEM_TOTAL_MIN/1048576
printf "| %40s | %15.3f MB |\n", "TOTAL (MAX)", MEM_TOTAL_MAX/1048576
printf "+-----------------------------------------+-------------------+\n"
}'
```