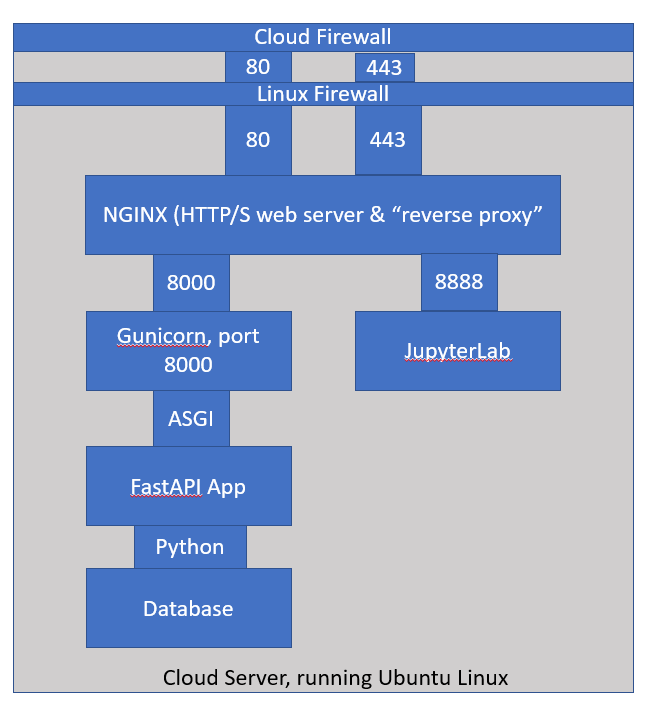
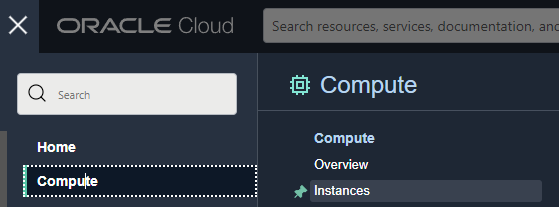
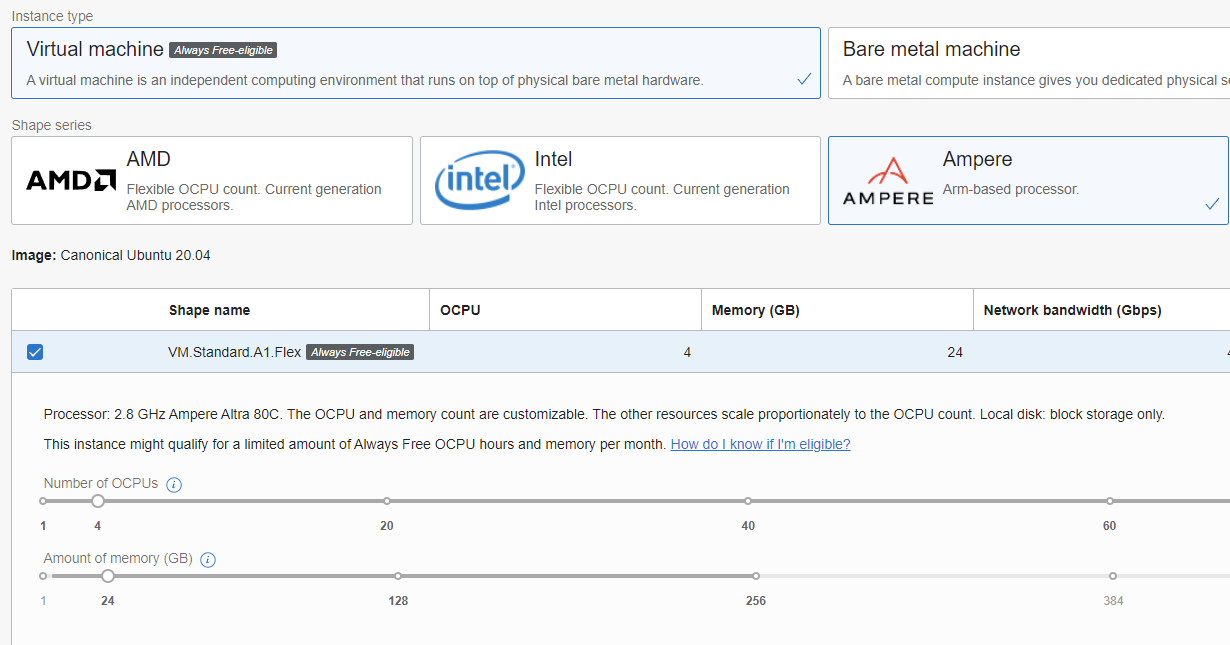
****

1. **Create your Oracle Cloud account.** 
   1. You should have received an email asking you to setup your Oracle Academy Cloud Program account. Do this now. It does not require a credit card, but does require you to enter a phone number.
   2. Set your home to us-ashburn (east)
2. **Create a new cloud compute instance**
   1. Login to cloud.oracle.com
   2. Go to your compute instances

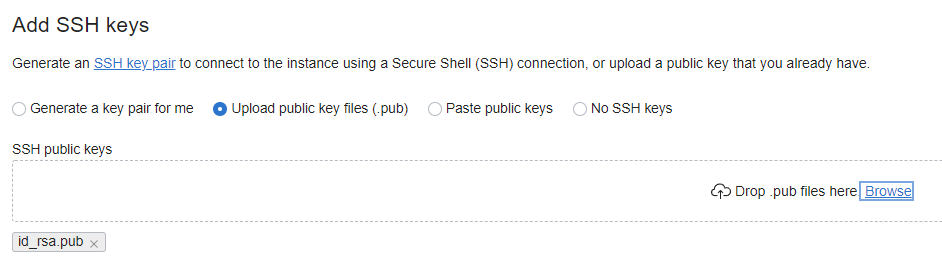




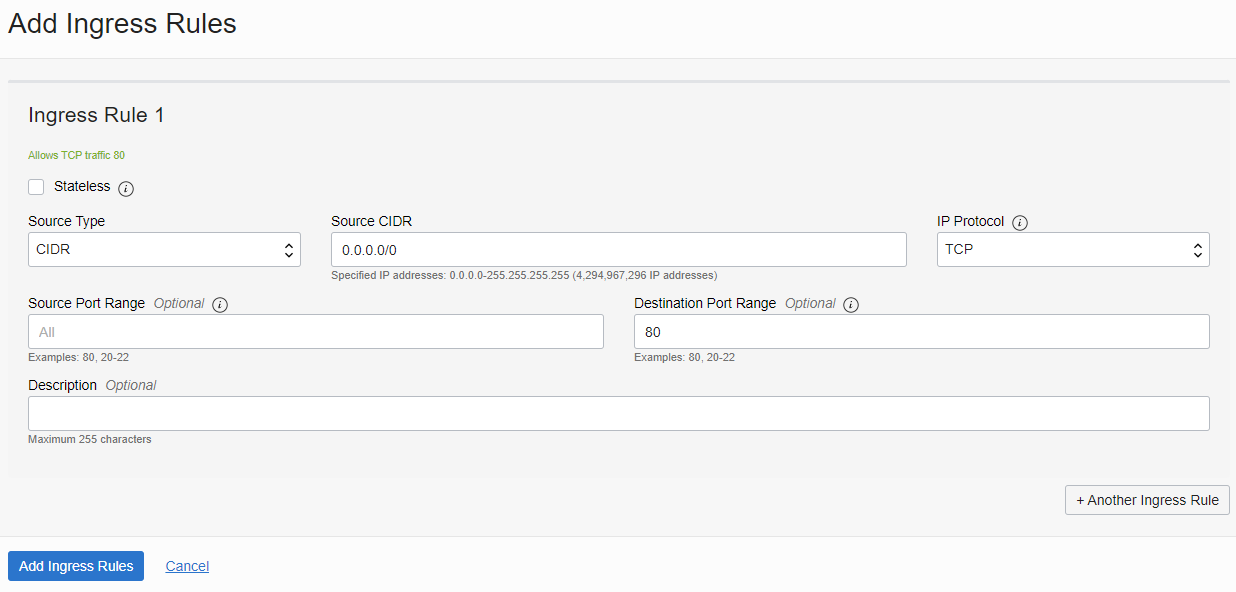
* 1. Click 
  2. Set the Image and Shape:
     1. Image - Canonical Ubuntu 20.04
     2. Shape - change to ampere 4 core, 24gb (or lower)

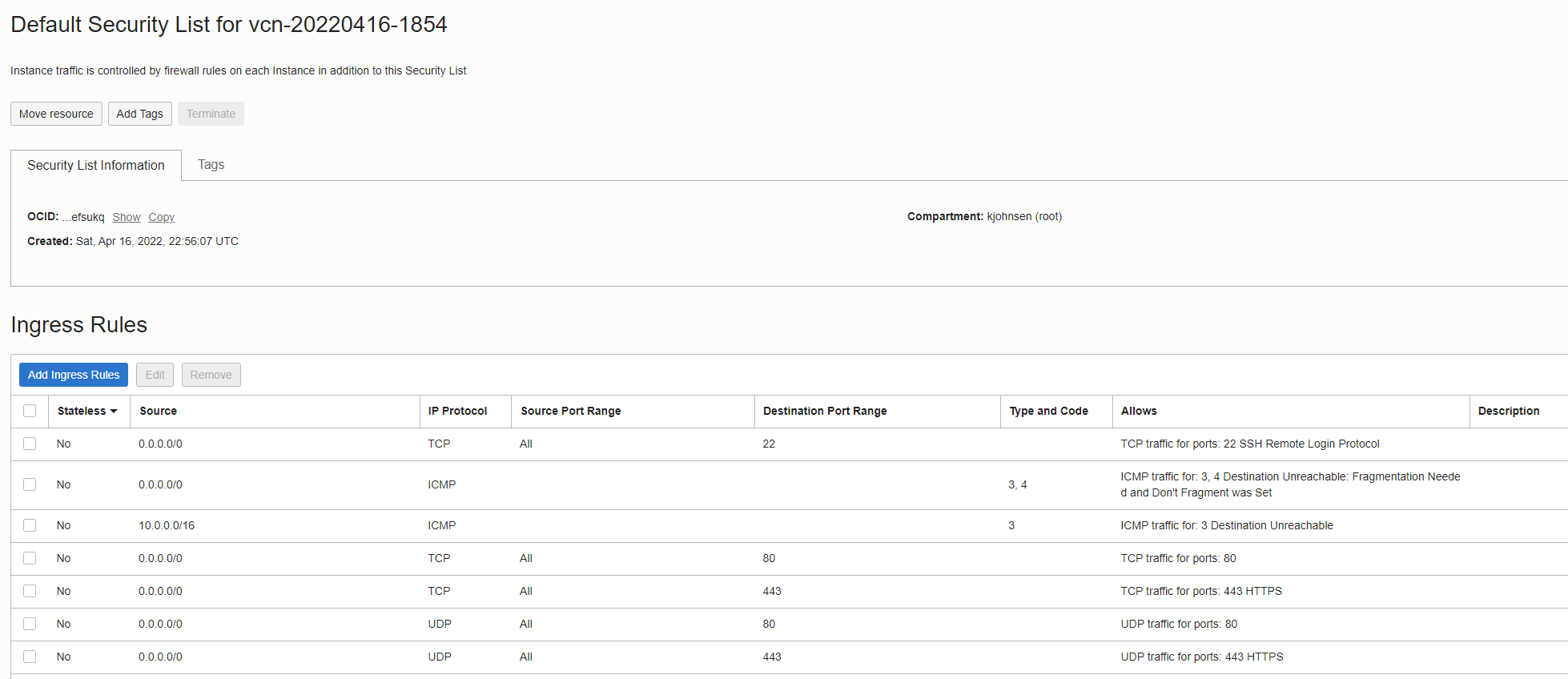


* 1. add your public SSH key (same as github is fine). You can also create a pair here.



* 1. Once it is created
     1. click on subnet/Default Security List and add tcp/udp 80/443 for 0.0.0.0/0 (all HTTP/HTTPS traffic)





1. **Setup NGINX Webserver and configuring the "uncomplicated" firewall**
   1. ssh into ubuntu@ipaddress
2. sudo apt update
3. sudo apt install nginx
4. sudo ufw allow OpenSSH
5. sudo ufw allow "Nginx Full"
6. sudo ufw enable
7. sudo reboot
   1. check <http://ipaddress> from your browser to verify that nginx is working
8. **Setup HTTPS on a domain**
   1. Register subdomain with https://freedns.afraid.org/ (e.g. kjjohnsen.crabdance.com), or https://freenom.com for a full domain (just not a .com) e.g kjjohnsen.ga
      1. Add a dns A record to point at the ip address of your server
   2. **Wait for that to work (wait until http://yourdomain works)! It may take several minutes.**
   3. Secure it with letsencrypt (certbot)
      1. sudo snap install core
      2. sudo snap refresh core
      3. sudo snap install --classic certbot
      4. sudo ln -s /snap/bin/certbot /usr/bin/certbot
      5. sudo certbot --nginx (handle prompts)
   4. Check that <https://yourdomain> works!
9. **Install miniconda**
   1. ssh into ubuntu@domain
   2. wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-aarch64.sh
   3. chmod +x Miniconda3-latest-Linux-aarch64.sh
   4. ./Miniconda3-latest-Linux-aarch64.sh
   5. **exit** and ssh in again (You should now see (base) in front of your shell prompt)
10. **Install jupyterlab and configure** 
    1. conda install -c conda-forge jupyterlab
    2. jupyter lab --generate-config
    3. edit the config file at ~/.jupyter/jupyter\_lab\_config.py

c.ServerApp.allow\_remote\_access = True

c.ServerApp.base\_url = "/jupyter/"

c.KernelSpecManager.ensure\_native\_kernel = False

* 1. Setup jupyter to use your conda environment
     1. python -m ipykernel install --user --name base
     2. jupyter kernelspec list
     3. jupyter kernelspec uninstall python3

1. **Setup reverse proxy for jupyterlab**
   1. edit the /etc/nginx/sites-enabled/default file

outside of a server block, include

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

inside of the https server block

location /jupyter/{

#the main thing

proxy\_pass http://127.0.0.1:8888/jupyter/;

#makes things work properly for the service

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

# websocket stuff

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection $connection\_upgrade;

proxy\_set\_header X-Scheme $scheme;

proxy\_buffering off;

}

* 1. save/exit
  2. sudo systemctl restart nginx
  3. test
     1. jupyter lab
     2. Go to <https://yoursite/jupyter> from your browser

1. Setting up Jupyter Lab to run on startup w/ systemd
   1. create a new systemd service configuration file jupyter.service

[Unit]

Description=Jupyter Lab

Requires=network.target

After=network.target

[Service]

User=ubuntu

Group=ubuntu

Environment="PATH=/home/ubuntu/miniconda3/bin"

WorkingDirectory=/home/ubuntu/

ExecStart=/home/ubuntu/miniconda3/bin/jupyter lab

[Install]

WantedBy=multi-user.target

* 1. copy jupyter.service to /etc/systemd/system/
  2. sudo systemctl enable jupyter
  3. sudo systemctl start jupyter
  4. test at https://yoursite/jupyter

1. **Getting your own app working**
   1. conda install -c conda-forge fastapi uvicorn gunicorn
   2. Create a main.py file

from fastapi import FastAPI

app = FastAPI(root\_path="/myapp")

@app.get("/")

async def root():

return {"message": "Hello World"}

* 1. Add a reverse proxy for your app in /etc/nginx/sites-enabled/default

location /myapp/{

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_buffering off;

proxy\_pass http://127.0.0.1:8000/;

}

* 1. sudo systemctl restart nginx
  2. run your app with uvicorn main:app
  3. test at https://yoursite/myapp
  4. [optional] Setup to run at startup, if desired, by creating another service file (similar to above

[Unit]

Description=My app

Requires=network.target

After=network.target

[Service]

User=ubuntu

Group=ubuntu

Environment="PATH=/home/ubuntu/miniconda3/bin"

WorkingDirectory=/home/ubuntu/

ExecStart=/home/ubuntu/miniconda3/bin/gunicorn -w 4 -k uvicorn.workers.UvicornWorker main:app

[Install]

WantedBy=multi-user.target

Aside: full nginx configuration file:

#this part is the port HTTP static files server that nginx ships with by default

server {

listen 80 default\_server;

listen [::]:80 default\_server;

root /var/www/html;

index index.html index.htm index.nginx-debian.html;

server\_name \_;

location / {

try\_files $uri $uri/ =404;

}

}

#this part is necessary for websockets (jupyter) to work

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

#this is your core HTTPS server block

server {

root /var/www/html;

index index.html index.htm index.nginx-debian.html;

server\_name info8000sp22.ga; # managed by Certbot

location / {

try\_files $uri $uri/ =404;

}

location /jupyter/{

#the main thing

proxy\_pass http://127.0.0.1:8888/jupyter/;

#makes things work properly for the service

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

# websocket stuff

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection $connection\_upgrade;

proxy\_set\_header X-Scheme $scheme;

proxy\_buffering off;

}

listen [::]:443 ssl ipv6only=on; # managed by Certbot

listen 443 ssl; # managed by Certbot

ssl\_certificate /etc/letsencrypt/live/info8000sp22.ga/fullchain.pem; # managed by Certbot

ssl\_certificate\_key /etc/letsencrypt/live/info8000sp22.ga/privkey.pem; # managed by Certbot

include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

#this part redirects some domains to HTTPS if they try to request HTTP

server {

if ($host = info8000sp22.ga) {

return 301 https://$host$request\_uri;

} # managed by Certbot

listen 80 ;

listen [::]:80 ;

server\_name info8000sp22.ga;

return 404; # managed by Certbot

}