Tech test

**Task1 – to up and run the Ibsite on port 80 and should handle the Public traffic**

**Task2- clone the solution onto every EC2 instance**

Task 1 Activity

1. To get Enviourment ready - I cleaned all the services before I start working on project by using these command ( systemctl stop httpd / systemctl stop http)
2. I check whether services stopped or not by using status command (systemctl status)
3. I cloned the git repo on the Folder WWW
4. In www folder I created git clone using - git clone https://github.com/whitesunset/wannacrypt\_balance
5. I added missing information on dockerfile - added RUN mkdir -p /run/nginx
6. I tried to build docker image and tried to launch the container image but there was error where I started troubleshoot (nginx: [emerg] open() "/run/nginx/nginx.pid" failed (2: No such file or directory). then I added RUN mkdir -p /run/nginx this command to dockerfile
7. I run the command sudo docker build -t Ibserver ( -t - Image name tagging)
8. Then I run this command where you can see I call Ibserver(tagged name) sudo docker run -dit -p 80:80 -v /home/ec2-user/tech-test/www:/www Ibserver
9. The problem occurred in step 5 is no more occurring and issue has been resolved.
10. I successfully able to browse the Ibsite of the git Repo which I clone in early steps

**Feasible other automation solution - Task2- clone the solution onto every EC2 instance**

* 1. To replicate the clone of this server (Ec2 Instance), I may need to use docker-compose.yml to automate the process
  2. EC2 automation AWS code deployment (CI/CD) vendor specific, Versioning, deploy, Commit, Push.
  3. Inside the Docker- I can use Nginx revers proxy to automatically Index the Github report ( No need to Gitrepo, I can directly use Nginx reverse proxy)
  4. docker- Compose.Yml – I thought about these solution by creating 2 different dockerfile , 1 for Nginx server/rever Proxy and 2 Git repo automatic update and then compose both together in Docker Composer, but was not sure whether this will help us or not.
  5. Create a new task and launched a task and the task, based on the task definition, it gave us one container.  A new task has come up and replaced our previously stopped task. So what a service does or what the service has an advantage over a task is that if, like, if we somehow lose a container, the service will automatically create a new container to take place of the stopped container. So this is more of like a auto healing measure to ensure that whatever app we're running in ECS is maintained and up and running at all times.