Notes on Today's Learnings & Solutions (Date: Dec 1, 2024)

Problem Definition:

I have created an EC2 instance on AWS using the Ubuntu operating system. On this instance, I installed the latest version of Java and Jenkins. Jenkins is running successfully and accessible via my EC2's public IP at port 8080.

I am working on hosting a static website built using HTML, CSS, and JavaScript. To achieve this, I created a freestyle project in Jenkins and configured it by adding the GitHub repository URL and setting up the GitHub SSH/GPG key for secure integration.

Objective:

I want to follow these steps to deploy and automate the static website hosting process:

1. **Serve the website using Nginx**: First, I want to configure Nginx to serve the static website.
2. **Dockerize the website**: Once the website is being served correctly with Nginx, I plan to create a `Dockerfile` to containerize the website.
3. **Automate the process**: Finally, I want to automate the entire deployment process, including building the Docker image and deploying the static website, so that the workflow can be executed automatically from Jenkins.

Next Steps:

- Configure Nginx to serve the static website.
- Write a `Dockerfile` to containerize the website.
- Create a Jenkins pipeline to automate the Docker image build and deployment.

This revised version gives a clearer picture of what you've done so far and what you aim to achieve, making it easier for others (or yourself) to follow your progress and understand the objective at hand.

---

1. Install and Configure Nginx

Update and Install Nginx:

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

1. **Configure Nginx to Serve Your Static Website**:

Navigate to `/etc/nginx/sites-available`:

```
cd /etc/nginx/sites-available
```

Create a configuration file for your site:

```
sudo nano your-site.conf
```

Add the following configuration:

```
server {
        listen 80;
        server_name your-ec2-public-ip;

        root /var/www/your-site;
        index index.html;

        location / {
            try_files $uri $uri/ =404;
        }
    }
```

Replace `your-ec2-public-ip` with your actual EC2 instance public IP and `/var/www/your-site` with the directory where your website files will reside.

Yes, in the current configuration of Nginx, you need to place your website files in the directory specified by the root directive, which is /var/www/your-site in the example. Here's how you can move your files:

1. Create the Target Directory:
2.
```
sudo mkdir -p /var/www/your-site
```

3. Copy Your Website Files: Assuming your website files are in a directory on your local machine (e.g., `~/my-website`):
4.
```
sudo cp -r ~/my-website/* /var/www/your-site/
```

Alternative: Change the Nginx Root

If you don't want to move your files to /var/www/your-site, you can modify the root directive in the Nginx configuration to point to the directory where your website files are currently located. For example:

1. Edit the configuration file:
2. `sudo nano /etc/nginx/sites-available/your-site.conf`
3. Update the `root` directive:
4. `root /path/to/your/current/website-files;`
5. Test and reload Nginx:
6. `sudo nginx -t sudo systemctl reload nginx`

Enable the configuration:

```
sudo ln -s /etc/nginx/sites-available/your-site.conf /etc/nginx/sites-enabled/
```

   ○

2. **Copy Your Static Website Files**: Copy your website files into the directory you specified in the `root` directive (e.g., `/var/www/your-site`).

---

2. Create a Dockerfile for Your Static Website

Create a `Dockerfile` in your website project directory:

Add your user to the Docker group: Run the following command to add your user to the `docker` group:

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

```
FROM nginx:latest
COPY . /usr/share/nginx/html
EXPOSE 80
```

1. This uses the Nginx base image and copies your static website files into the default Nginx web root.

Build the Docker Image: Navigate to your project directory and run:

```
docker build -t your-site-image .
```

Run the Docker Container: Test the Docker container by running:

```
docker run -d -p 8080:80 your-site-image
```

2. Access your site at `http://<your-ec2-public-ip>:80`.

---

3. Automate with Jenkins

1. **Set Up SSH Key for GitHub**:
   - Add your Jenkins public SSH key to GitHub as a deploy key (ensure it has read access).
2. **Create a Freestyle Project**:
   - **Source Code Management**:
     - Select "Git" and provide the repository URL.
     - Add your Git credentials if needed.
   - **Build Triggers**:
     - Enable "Poll SCM" or configure a webhook for automation.

Build Steps: Add a build step to:

```
 docker build -t your-site-image .
docker stop your-site-container || true
docker rm your-site-container || true
docker run -d -p 8080:80 --name your-site-container your-site-image
```

   -

3. **Test the Jenkins Build**: Trigger a build manually to ensure your pipeline correctly fetches the repository, builds the Docker image, and runs the container.

---

4. Extend with a CI/CD Pipeline

Later, you can:

- **Add Tests**: Write simple scripts to verify your site loads correctly.
- **Enhance Dockerfile**: Optimize it by using multi-stage builds for scalability.

- **Deploy to Docker Hub**: Push the Docker image to Docker Hub or an ECR registry for production deployment.

Let me know if you encounter any issues! 🚀