Demo Project: Deploy Web Application on EC2 Instance (manually)

This guide demonstrates how to:

1. Create and configure an EC2 instance on AWS.
2. Connect to the EC2 instance via SSH.
3. Install Docker on the remote EC2 instance.
4. Deploy a Docker container from a private DockerHub repository.
5. Make the application accessible publicly.

1. Create and Configure an EC2 Instance on AWS
2. Install Docker on the EC2 Instance
3. Deploy the Docker Image from a Private Docker Repository
4. Make the Application accessible from the Browser
Troubleshooting

1. Create and Configure an EC2 Instance on AWS

1. **Log in to the AWS Management Console**:
    - Visit AWS Console.
2. **Launch an EC2 Instance**:
    - Navigate to **EC2 Dashboard** → **Launch Instances**.
    - Configure the following:
        - **Name**: `my-instance`.
        - **Tags**: Add tags for better resource management:

    - **Key:** `Type`
    - **Value:** `web-server-with -docker`
  - **AMI (Amazon Machine Image)**: Use **Amazon Linux 2023 AMI**.
  - **Instance Type**: Select **t2.micro** (Free Tier eligible).
  - **Key Pair**: Create or select an existing **SSH key pair** for secure access. For this project create a new Key Pair as `docker-server.pem`.
  - VPC: Choose VPC if you have a specific VPC that EC2 needs to be deployed, else leave in default VPC.
  - Subnet: Choose the subnet if you have a specific VPC that EC2 needs to be deployed, else leave as “No preference”
  - **Network Settings**:
    - **VPC**: Select your specific VPC or leave it as **Default VPC**.
    - **Subnet**: Choose a specific subnet if applicable or leave it as **No Preference**.
    - Auto-assign a **public IP**.
    - Add a **Security Group** with:
      - Add **Inbound Rules** for:
        - **SSH (Port 22)**: Your IP only (e.g., `203.0.113.0/32`).
  - **Storage**: Allocate at least **8GB** SSD (GP3).
- Click **Launch Instance**.

3. Move the Key Pair to .ssh: `mv Downloads/docker-server.pem ~/.ssh/`

   Confirm it was moved successfully: `ls -l .ssh`

4. Update Key Permissions: `chmod 400 .ssh/docker-server.pem`

5. **Access the EC2 Instance via SSH**:
   - Copy the **public IP address** of the instance.
   - Use the following command to connect:

     `ssh -i <path-to-your-private-key.pem> ec2-user@<public-ip-address>`

Example: `ssh -i .ssh/docker-server.pem ec2-user@52.71.72.116`

---

2. Install Docker on the EC2 Instance

1. **Update the Package List:** `sudo yum update -y`
2. **Install Docker:** `sudo yum install docker -y`
3. **Verify Docker Installation:** `docker --version`
4. **Start Docker:** `sudo service docker start`
5. Check docker is running by checking it's process: `ps aux | grep docker`
6. **Add the EC2 User to the Docker Group**:

   `sudo usermod -aG docker ec2-user`

   - **Why This Step is Necessary**:
     - The Docker daemon runs as the **root user**.
     - Regular users (like `ec2-user`) cannot execute Docker commands unless they are part of the **Docker group**.
     - Adding the EC2 user (`ec2-user`) to the Docker group with the `usermod` command grants it the necessary permissions to run Docker commands **without using sudo**.
   - **Apply Changes**: Log out and log back in for the changes to take effect:

     ```
     exit
     ssh -i <path-to-your-private-key.pem> ec2-user@<public-ip-address>
     ```

---

3. Deploy the Docker Image from a Private Docker Repository

1. **Log in to Your Private Docker Repository**:
   - Replace `<username>` and `<password>` with your DockerHub credentials or token:

     `docker login -u <username> -p <password>`

2. **Pull the Docker Image**:
    - Replace `<image-name>` with your private repository's image name

      `docker pull <username>/<image-name>:<tag>`

      Example: `docker pull eduardobautistamaciel/demo-app:1.0`
3. **Run the Docker Container**:
    - Replace `<port>` with the port your application uses:

      `docker run -d -p <port>:<container-port> <username>/<image-name>:<tag>`

      Example: `docker run -d -p 3000:3080 eduardobautistamaciel/demo-app:1.0`
4. Verify the container is running: `docker ps`

---

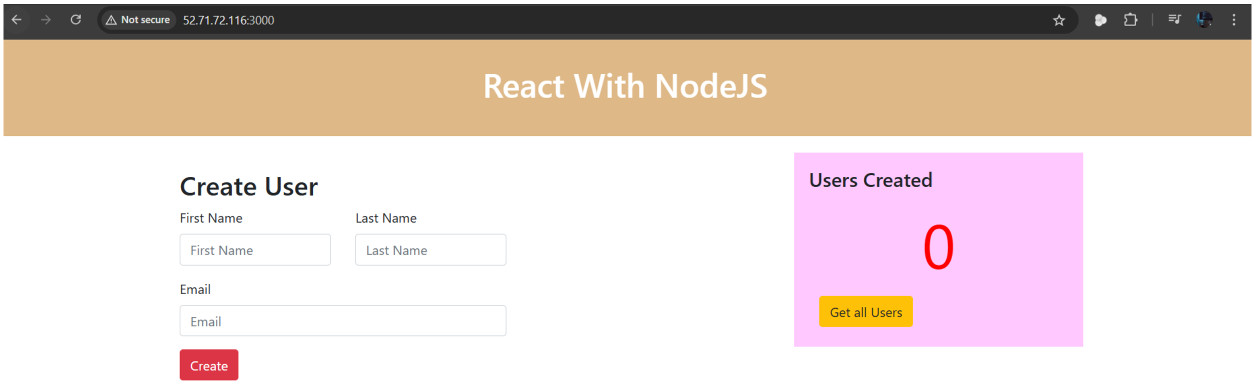
4. Make the Application accessible from the Browser

1. **Update the Security Group**:

- Go to **AWS EC2 → Security Groups → Edit Inbound Rules**.
- Add a rule:
    - **Type**: Custom TCP Rule.
    - **Port**: 3000.
    - **Source**: Anywhere (`0.0.0.0/0`).

2. **Verify the Application**:
    - Open your browser and navigate to:

      `http://<public-ip-address>:<port>`
    - Example: `http://52.71.72.116:3000`



Troubleshooting

1. **Cannot Connect to EC2 Instance**:
   - Verify the **Security Group rules** allow SSH access from your IP.
   - Ensure your key pair matches the instance's key pair.
2. **Docker Permission Denied**:
   - **Cause**: Docker commands require root privileges.
   - **Solution**: Add the EC2 user to the Docker group (`usermod -aG docker ec2-user`), log out, and log back in to apply the group change.
3. **Application Not Accessible in Browser**:
   - Ensure the **Security Group** allows traffic on port `3000`.
   - Verify the container is running (`docker ps`).
   - Check the container logs: `docker logs <container-id>`
4. **Cannot Pull Docker Image**:
   - Verify your credentials or token for the private Docker repository.
   - Ensure the image name and tag are correct.