Jenkins Ansible Docker React Pipeline

This Project will take you through the steps of a general CI/CD Pipeline that deploys a React Application. There are a number of AWS Servers (EC2s) used throughout, each with a specific purpose. Server (1) will be the Jenkins Main where our pipeline will be executed. Server (2) will be the Jenkins build Agent that will actually run the pipeline stages. Server (3) will be a Development Environment where our app can be built and containerized using Ansible and Docker. Server (3) will also push an image of the built app to DockerHub (This is also where testing should be done). Finally, Server (4) is the Production Environment, it will pull a published image from DockerHub and run the containerized app.

1. Make EC2 for Jenkins Main(1), Build Agent EC2 (2), Ansible EC2 (3), Docker EC2 (4)
    a. Install node plugin and install correct global tool config for nodejs version on (1)
    b. Install publish ssh plugin on 1. Add ssh config to ansible server (3)
    c. Make sure Ansible is installed on (3)
        i. https://www.ktexperts.com/how-to-install-ansible-in-amazon-linux-machine/
        ii. https://devopsmyway.com/how-to-install-ansible-on-amazon-linuxec2/#content
    d. Make Sure Docker is installed on (3)
        i. Sudo yum install docker -y
        ii. `sudo amazon-linux-extras install docker -y`

2. Make a Freestyle Project on Jenkins Main and a pipeline job that connects to a Git Repo (Should also use a Multibranch so you can use a Jenkinsfile from a repo)
    a. Source Code Management and paste git repo url, following uses Freestyle
        i. Make sure git is installed on jenkins ec2
        ii. Made poll SCM build trigger, * * * * * in schedule
        iii. Build environment, provide node and bin/ folder path
            1. Pick node installation version you installed on jenkins ec2
        iv. Build Step execute shell, npm install, npm run build

    b. Use a Jenkins Script to have Build Agent(2) pull from git repo, install npm on itself and make a build of the react project

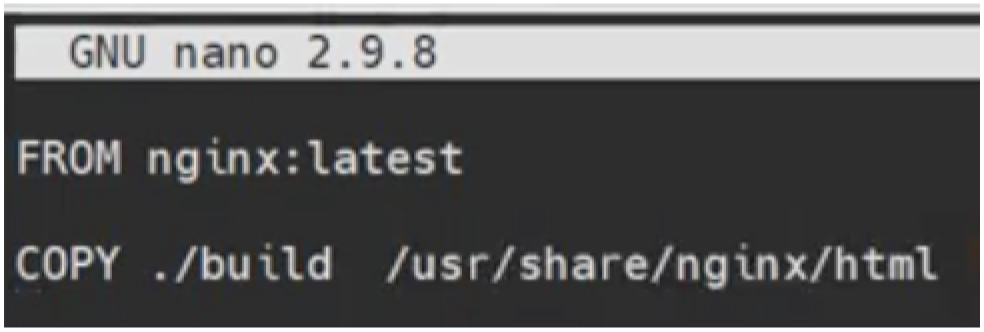
```
pipeline {
    agent {
        label 'Build Agent'
    }

    tools {nodejs 'NodeJS 14.17.3'}

    stages {
        stage('Build') {
            steps {
                git
'https://github.com/Fordonez20/scans-on-set-v2.git'
                sh 'npm install'
                sh 'npm -v'
                sh 'npm run build'
            }
        }
    }
}
.
```

   c. Next added to Script so the Agent(2) copies over build folder contents to the Ansible Server(3)
      i. First make sure build agent can scp to ansible server
         1. https://blog.microideation.com/2016/05/26/secure-copy-files-scp-between-two-ec2-instances-in-aws/
         2. Used a bash script on (2) that runs a scp command

```
scp -i "/home/jenkins/.ssh/KeyForHWDeployment.pem" -r
/home/jenkins/workspace/build-react-pipeline-to-agent/build
ec2-user@172.31.16.6:/home/ec2-user
```

3. Making Sure Server A can command Server B
   a. Server A must have a ssh-keygen made, its public and private key can be found in the ssh folder of you user or root account
   b. Make sure the public key contents of Server A are pasted into the “authorized_keys” file in the .ssh folder of Server B

4. Initiating Docker on the Built Project on Node (3)
   a. Make Sure Docker is running on Ansible Node (3)
      i. Sudo service docker start
      ii. Check with: sudo docker ps
      iii. To add user docker group: sudo su then run, usermod -aG docker ec2-user
   b. Make a Dockerfile in same directory as transferred build folder in(3)
      i. sudo nano Dockerfile



```
GNU nano 2.9.8

FROM nginx:latest

COPY ./build  /usr/share/nginx/html
```

5. Add to Jenkins Script so that Node(2) tells Node(3) to create a docker image
   a. Here I added the following to the bash script I used to transfer the build folder
      i. ssh -i ./.ssh/my-key.pem ubuntu@54.147.254.33 'echo "Hello World"'

Important: Here we are running a bash script on the Build Server (2), the script atm runs docker on Server (3) and before that copied the build folder over. Eventually those last 2 commands in the script will run ansible playbooks

```
  GNU nano 4.8                    runTransfer
#!/bin/bash

scp -i "/home/jenkins/.ssh/KeyForHWDeployment.pem" -r
/home/jenkins/workspace/build ec2-user@172.31.16.6:/home/ec2-user

ssh -i "/home/jenkins/.ssh/KeyForHWDeployment.pem"
ec2-user@172.31.16.6 'sudo docker build -t react-app .'
ssh -i "/home/jenkins/.ssh/KeyForHWDeployment.pem"
ec2-user@172.31.16.6 'sudo docker run --name react-app -d -p 80:80
react-app'
```

- At this point, the react app is running on a docker container hosted on the third node on port 80, so you can check the local ip of Node(3)

6. Using Ansible on Node (3) to Recreate a Docker Image and running container of the app.
   a. Make Sure ansible is installed on (3)
   b. Create your ansible playbook in project directory in (3)

```
- name: build and run container
 hosts: localhost
```

```
become: yes
become_user: ec2-user

tasks:
- name: stop running container
  command: sudo docker stop react-app
  ignore_errors: yes

- name: delete the container
  command: sudo docker rm react-app
  ignore_errors: yes

- name: delete the image
  command: sudo docker rmi react-app
  ignore_errors: yes

- name: build the image
  command: sudo docker build -t react-app .
  args:
    chdir: /home/ec2-user

- name: run the container
  command: sudo docker run --name react-app -d -p 80:80 react-app
```

c. Create your ansible hosts file
   i. In this case ansible this "host" is just the localhost for server (3)
   ii. It will later build the image and upload it to dockerhub
   iii. Our 4th server will pull from dockerhub and run the app container

7. Edit Bash Script used in Jenkins Script to run the ansible playbook
   a. sudo ansible-playbook -i /home/ec2-user/hosts /home/ec2-user/ansible-build-docker-local.yml
   b. ansible-playbook /home/ec2-user/ansible-build-docker-local.yml
   c. In (3), Went into etc/ansible/ansible.cfg and made sure host_key_checking=False

8. Set Up Docker Production Server (4)
    a. Ensure that Server (3) can ssh into Server (4), this is needed because (4) will be a remote host targeted by the ansible hosts file
        i. Helps to reload sshd after changes service with: sudo service sshd reload
    b. In (3) make sure /etc/ansible/hosts marks server (4) as possible host:
    c. [docker]
    d. ubuntu@<privateip>
    e. Here had to actually use private ip of remote host
    f. In folder → /etc/ansible

9. Rewrite 2 playbooks, ansible-playbook-local.yml and ansible-playbook-remote.yml
    a. The local will make an image locally and push it to dockerhub

```
- name: build and push image to dockerhub
 hosts: localhost
 become: yes
 become_user: root

 tasks:
 - name: build the image
   command: docker build -t react-app .
   args:
     chdir: /home/ec2-user

 - name: login to dockerhub
   command: docker login -u "fordonez20" -p "<pw here>" docker.io

 - name: tag the image
   command: docker tag react-app
fordonez20/react-app-published:latest

 - name: push the image
   command: docker push fordonez20/react-app-published:latest

 - name: remove the build image and the tagged one
   command: docker rmi react-app
fordonez20/react-app-published:latest
   ignore_errors: yes
```

b. The remote will pull from dockerhub and run the container on its localhost

```
- name: build and run container
 hosts: localhost
 become: yes
 become_user: root

 tasks:
 - name: stop running container
   command: docker stop react-app
   ignore_errors: yes

 - name: delete the container
   command: docker rm react-app
   ignore_errors: yes

 - name: login to dockerhub
   command: docker login -u "fordonez20" -p "<pw here>" docker.io

 - name: delete the image
   command: docker rmi fordonez20/react-app-published:latest
   ignore_errors: yes

 - name: build the image
   command: docker pull fordonez20/react-app-published:latest

 - name: run the container
   command: docker run --name react-app -d -p 80:80
fordonez20/react-app-published:latest
```

10. Finally change the runAnsiblePlaybook script to run the local playbook and then the remote playbook.

```
#!/bin/bash

ssh -i "/home/jenkins/.ssh/KeyForHWDeployment.pem" ec2-user@172.31.16.6
'ansible-playbook /home/ec2-user/ansible-playbook-local.yml'
```

```
ssh -i "/home/jenkins/.ssh/KeyForHWDeployment.pem" ec2-user@172.31.16.6
'ansible-playbook /home/ec2-user/ansible-playbook-remote.yml'
```