DevOpsFinal

Capstone project for dev ops course

Hangout Point

An entertainment company like BookMyShow where users book their tickets have multiple users accessing their web app. Due to less infrastructure availability, they use less machines and provide the required structure. This method includes many weaknesses such as:

Developers must wait till the complete software development for the test results. There is a huge possibility of bugs in the test results. Delivery process of the software is slow. The quality of software is a concern due to continuous feedback referring to things like coding or architectural issues, build failures, test conditions, and file release uploads. The objective is to implement the automation of the build and release process for their product.

Implementation requirements:

1. Set up the Jenkins server in master or slave architecture
2. Use the Jenkins plugins to perform the computation part on the Docker containers
3. Create Jenkins pipeline script
4. Use the GIT web hook to schedule the job on check-in or poll SCM
5. Build an image using the artifacts and deploy them on containers
6. Remove the container stack after completing the job

The following tools must be used:

EC2 Jenkins Docker Ansible, Chef, or Puppet

Solution

Create AWS EC2 Instance

https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/ 1. Create AWS Account 2. Create key pair - Download and place in working directory 3. Create security group - ensure port 22 is open to allow ssh 4. Launch instance - use security group and key pair created above 5. Connect to EC2 instance - ssh -i ec2-user@ 6. Download and install git - sudo yum update -y - sudo yum install git -y 7. Download and install Maven - wget - tar -xvcf /opt/maven 8. Download and install Docker - sudo amazon-linux-extras install docker - sudo service docker start - sudo usermod -a -G docker ec2-user

Set Up Jenkins Server on AWS EC2 Instance

1. Download and install Jenkins

   - sudo yum update -y
   - sudo wget -O /etc/yum.repos.d/jenkins.repo \ https://pkg.jenkins.io/redhat-stable/jenkins.repo
   - sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
   - sudo yum install jenkins java-1.8.0-openjdk-devel -y
   - sudo systemctl daemon-reload
   - sudo systemctl start jenkins
   - sudo systemctl status jenkins

2. Add jenkins user to Docker

   - sudo usermod -a -G docker jenkins

3. Navigate to the jenkins interface running on EC2

   - http://:8080 (note, only http is used)
   - sign in using initial admin password -- sudo cat /var/lib/jenkins/secrets/initialAdminPassword

4. Install suggested plugins

5. Install the "Amazon EC2" plugin
6. Install the "Build Pipeline" plugin
7. Install the "Copy Artifact" plugin

Provide AWS Credentials to Jenkins

1. Sign in to the AWS Management Console and navigate to IAM.
2. Select Users from side navigation and choose Add user.
3. Enter the User name as Jenkins and grant Programmatic access to this user.
4. On the next page, select Attach existing policies directly and choose Create Policy.
5. Select the JSON tab and add the policy at AWSJenkins.json. Replace with your AWS account ID:
6. Choose Review Policy and add a policy name on the next page.
7. Choose Create Policy button.
8. Return to the previous tab to continue creating the IAM user. Choose Refresh and search for the policy name you created. Select the policy.
9. Choose Next Tags and then Review.
10. Choose Create user and save the Access key ID and Secret access key.
11. Create a key pair for Jenkins
12. Navigate back to the Jenkins dashboard, then select Configure a Cloud.
    - Add a new cloud > Amazon EC2
    - Add a new AWS Credential in the Amazon EC Credentials field, use the access key ID and secret access key from 10
    - Add a new SSH Username with private key credential in the EC2 Key Pair's Private Key, use the key from 11

13. Save

Create Git Repository with Application

1. From GitHub, create a new repository
2. Clone empty repository to local working directory
3. Download a sample java app from start.spring.io
    - Maven Project
    - Java
    - 2.5.4
    - War
    - Java 8
    - Add Depenedencies > Spring Web, Thymeleaf
4. Place project files in the git repository directory
5. Push changes to remote repository

Create Ansible Playbook to Provision Webserver Containers

1. In the working directory, create a new file called webServerDocker.yaml and insert the following:

```
---
- hosts: webservers
  become: true

  tasks:
    - name: Install aptitude using apt
      apt: name=aptitude state=latest update_cache=yes force_apt_get=yes

    - name: Install required system packages
      apt: name={{ item }} state=latest update_cache=yes
      loop: [ 'apt-transport-https', 'ca-certificates', 'curl', 'software-properties-common', 'python3-pip', 'virtualenv', 'python3-

    - name: Add Docker GPG apt Key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Add Docker Repository
      apt_repository:
        repo: deb https://download.docker.com/linux/ubuntu xenial stable
        state: present

    - name: Update apt and install docker-ce
      apt: update_cache=yes name=docker-ce state=latest

    - name: Install Docker Module for Python
      pip:
        name: docker

    - name: Pull Docker image
      docker_image:
        name: "spring-app"
        source: pull

    - name: Create container
      docker_container:
        name: "spring-app container"
        image: "spring-app"
        state: present
```

Create Jenkins Build Job

1. From the Jenkins dashboard, select New Item > Freestyle Project. Name the project "Build"
2. Under Source Code Management, select Git. Paste the link to your github repo in the Repository URL field.
3. Under Build Triggers, select Poll SCM. Enter "0 0 * * *" to check every night for new commits to trigger a build.
4. Under Build Environment, select Delete workspace before build starts
5. Under Build, select Add build step > Execute Shell. Enter `export PATH=$PATH:/opt/maven/apache-maven-x.x.x:/opt/maven/apache-maven-x.x.x/bin; cd demo; mvn --version mvn clean package` in the Command field.

6. Under Post-build Actions select Add post-build action > Archive the artifacts.
7. Enter "demo/target/*.jar, Dockerfile, webServerDocker.yaml" in the Files to archive field.
8. Under Post-build Actions select Add post-build action > Build other projects.
9. Enter "Test" in the Projects to build field, and select Trigger only if build is stable.

Create Jenkins Test Job

1. From the Jenkins dashboard, select New Item > Freestyle Project. Name the project "Test"
2. Under Source Code Management, select Git. Paste the link to your github repo in the Repository URL field.
3. Under Build, select Add build step > Execute shell. Enter `export PATH=$PATH:/opt/maven/apache-maven-x.x.x:/opt/maven/apache-maven-x.x.x/bin; cd demo; mvn --version mvn test` in the Command field.
4. Under Post-build Actions select Add post-build action > Build other projects.
5. Enter Deploy in the Projects to build field, and select Trigger only if build is stable.
6. Under Post-build Actions select Add post-build action > Publish Junit test result report.
7. Enter "target/surefire-reports/*.xml" in the Test report XMLs field.
8. Under Post-build Actions select Add post-build action > Build other projects.
9. Enter "Deploy" in the Projects to build field, and select Trigger only if build is stable.

Create Jenkins Deploy Job

1. From the Jenkins dashboard, select New Item > Freestyle Project. Name the project "Deploy"
2. Under Build Environment, select "Delete workspace before build starts"
3. Under Build, select Add build step > Copy artifacts from another project. Set the following
   - Project name: Build
   - Which build: Latest successful build
   - Artifacts to copy: **/*.jar
4. Under Build, select Add build step > Execute Shell. Enter `docker build -t spring-app . ansible-playbook webServerDocker.yaml docker ps` in the Command field.

Create Build Pipeline View

1. From the Jenkins Dashboard, click "New View"
2. Select "Build Pipeline View" and name it "Pipeline View," click OK.
3. Under Pipeline Flow > Upstream/downstream config, set "Build" as the Initial Job.
4. Click OK. Pipeline View should be displayed.

Remove Containers - Create Jenkins Clean Up Job

1. From the Jenkins dashboard, select New Item > Freestyle Project. Name the project "Clean Up"
2. Under Build, select Add build step > Execute Shell. Enter `docker ps docker stop $(docker ps -a -q) docker ps` in the Command field.