Can you write a simple Jenkins file to build a Docker image and push it to AWS ECR ?

|  |
| --- |
| node {  def AWS\_ACCOUNT\_ID = '123456789012'  def AWS\_REGION = 'us-east-1'  def ECR\_REPO = 'my-app-repo'  def IMAGE\_TAG = 'latest'  def ECR\_URL = "${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_REGION}.amazonaws.com/${ECR\_REPO}"  stage('Checkout Code') {  checkout scm  }  stage('Login to AWS ECR') {  withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials']]) {  sh """  aws ecr get-login-password --region ${AWS\_REGION} | \  docker login --username AWS --password-stdin ${ECR\_URL}  """  }  }  stage('Build Docker Image') {  sh "docker build -t ${ECR\_REPO}:${IMAGE\_TAG} ."  }  stage('Tag & Push to ECR') {  sh """  docker tag ${ECR\_REPO}:${IMAGE\_TAG} ${ECR\_URL}:${IMAGE\_TAG}  docker push ${ECR\_URL}:${IMAGE\_TAG}  """  }  stage('Cleanup') {  sh "docker rmi ${ECR\_REPO}:${IMAGE\_TAG} || true"  }  } |

|  |
| --- |
| node {  stage('Parallel Execution') {  parallel(  Build: {  stage('Build') {  echo 'Building the application...'  }  },  Test: {  stage('Test') {  echo 'Running tests...'  }  }  )  }  stage('Deploy') {  echo 'Deploying the application...'  }  } |

*Parallel execution:*

A deployment to production via Jenkins pipeline introduced a critical bug. How would you roll back safely?

*It depends on where the application is deployed*

*Containerize then Image needs to roll back.*

*If its in tomcat the WAR file needs to revert.*

How would you configure a Jenkins pipeline to deploy the same application to Dev, Staging, and Prod with different configurations?

Add a **String Parameter** named ENV with values: dev, staging, prod

|  |
| --- |
| properties([  parameters([    string(name: 'ENV', defaultValue: 'dev', description: 'Target environment: dev, staging, prod')  ])  ]) |

How would you securely manage AWS credentials and API keys in a Jenkins pipeline?

How do you set up Jenkins to trigger a deployment automatically when a new Docker image is pushed to ECR?

*Use Jenkins Credentials Store for API keys*

*Use IAM roles for EC2-based Jenkins*

*Use AWS Secrets Manager for centralized security*

*Never hardcode credentials in scripts*

What key metrics do you use to measure the quality of a CI/CD pipeline?

One Line below commands -

1. **Static Application Security Testing (SAST)**
   1. **Purpose: Detect vulnerabilities in source code before build.**
2. **Software Composition Analysis (SCA)**
   1. **Software Composition Analysis (SCA)**

**3. Dynamic Application Security Testing (DAST)**

**a. Purpose: Simulates attacks on running applications.**

**4. Container Security Scanning**

1. **Purpose: Detect vulnerabilities in Docker images.**

**5. Infrastructure as Code (IaC) Security**

1. **Purpose: Scan Terraform, CloudFormation for misconfigurations.**

How would you implement a Blue-Green Deployment strategy in Jenkins to minimize downtime?

Rolling Deployment

- Updates instances gradually by replacing old versions with new ones.

- Ensures zero downtime as some instances remain active during deployment.

- Example: Kubernetes rolling updates.

Blue-Green Deployment

- Two environments: \*\*Blue\*\* (current) and \*\*Green\*\* (new).

- Traffic is switched to \*\*Green\*\* after successful testing.

- Provides easy rollback by switching back to \*\*Blue\*\* if issues arise.

Canary Deployment

- Releases updates to a small percentage of users before a full rollout.

- Helps detect issues in real-time with minimal impact.

- Example: Deploying to 5% of users first, then scaling up if stable.

Recreate Deployment

- Stops all running instances before deploying the new version.

- Causes downtime but ensures a clean deployment.

- Suitable for environments where rolling updates aren’t feasible.

Shadow Deployment

- Routes real-time traffic to the new version \*\*without exposing it to users\*\*.

- Used for performance testing and validating new features.

- Example: Testing AI models before fully integrating them into production.

A/B Testing Deployment

- Deploys multiple versions to different user groups for comparison.

- Helps measure user engagement and select the best-performing version.

- Example: Showing Version A to 50% of users and Version B to the other 50%.

How do you zero-downtime

Zero-Downtime Deployment Strategies -

Rolling Deployment – Gradually updates instances without downtime.

Blue-Green Deployment – Switches traffic between two environments.

Canary Deployment – Releases updates to a small group before full rollout.

Load Balancer Traffic Shifting – Redirects traffic dynamically between versions.

Feature Flags – Deploys features but enables them selectively.

Database Migrations – Ensures schema changes don’t break existing services.

Shadow Deployment – Tests new versions with real traffic without user impact.

Ansible CD/CD -

|  |
| --- |
| - name: Deploy Application  hosts: web\_servers  become: yes  tasks:  - name: Pull latest code  git:  repo: 'https://github.com/user/repo.git'  dest: '/var/www/app'  version: main  - name: Install dependencies  command: pip install -r /var/www/app/requirements.txt  - name: Restart application  systemd:  name: myapp  state: restarted |