**🔧 Basic to Intermediate Git Scenario Questions**

**1. Scenario: You committed sensitive information (like a password or API key) to a public repo. What do you do?**

**Answer:**

**Remove the sensitive data from the file.**

**Use  git rebase  or  git filter-branch to remove the commit from history:**

**git filter-branch --force --index-filter \ "git rm --cached --ignore-unmatch path/to/file" \ --prune-empty --tag-name-filter cat -- --all**

**Force push the changes:**

**git push origin --force --all**

**Invalidate the exposed key/password immediately.**

**2. Scenario: How do you recover a deleted branch?**

**Answer:  
Find the commit hash:**

**git reflog**

**Then create a new branch from that hash:**

**git checkout -b branch\_name <commit\_hash>**

**3. Scenario: You accidentally committed to the main branch instead of a feature branch. How do you fix it?**

**Answer:**

**Create a new branch from the current commit:**

**git branch feature-branch**

**Reset main to the previous commit (soft or hard depending on your need):**

**git reset --soft HEAD~1 *# Keeps changes in staging***

**Switch to the new branch and continue work:**

**git checkout feature-branch**

**4. Scenario: You're working on a team and someone pushed a broken commit to main. How do you fix it?**

**Answer:  
Use revert if the commit is already pushed and shared:**

**git revert <commit\_hash> git push origin main**

**Use reset only if you're sure no one else has pulled the changes yet:**

**git reset --hard <good\_commit> git push --force**

**5. Scenario: You have a merge conflict during a pull. How do you resolve it?**

**Answer:**

**Git will mark the conflicting files.**

**Open each file and manually resolve the conflict.**

**Mark as resolved:**

**git add <file>**

**Continue the merge:**

**git commit**

**6. Scenario: How do you apply a specific commit from one branch to another?**

**Answer:**

**git cherry-pick <commit\_hash>**

**EX🡪**

**git checkout div**

**git log -3**

**a1b2c3d , d4e5f6g , h7i8j9k ---------- commits in div**

**git checkout add**

**git cherry-pick d4e5f6g**

**git log -3**

**x1b2c3n, r7i89ud , d4e5f6g ------- commits in add**

**7. Scenario: You pushed the wrong commit to a remote branch. How do you undo it?**

**Answer:**

**Undo the last commit locally:**

**git reset --hard HEAD~1**

**Force push to remote:**

**git push origin branch-name --force**

**⚠️ Use this only if you're sure others haven’t pulled the commit.**

**8. Scenario: What's the difference between git merge and git rebase in a real project?**

**Answer:**

| **git merge** | **git rebase** |
| --- | --- |
| **Preserves history** | **Rewrites history** |
| **Creates a merge commit** | **Creates a linear history** |
| **Better for public branches** | **Better for local, private branches** |

**In a team setting, you may use rebase before merging to keep history clean.**

**9. Scenario: You want to see all changes between two branches. How?**

**Answer:**

**git diff branch1..branch2**

**or to compare commits:**

**git log branch1..branch2**

**10. Scenario: A file is showing up in git status, but you want Git to ignore it without modifying .gitignore.**

**Answer:**

**Use assume-unchanged:**

**git update-index --assume-unchanged path/to/file**

**11. You want Jenkins to trigger a build automatically when a pull request is created. How do you do that?**

**Use GitHub Webhooks and a Pipeline in Jenkins.**

**In GitHub: 🡪 Go to Repo → Settings → Webhooks → Add webhook 🡪Set the Payload URL ( <http://<jenkins-server>/github-webhook/> ) 🡪 test this**

**Allow Only Few Members to Push to main in GitHub**

**Go to your GitHub repository. 🡪 Settings 🡪 click Branches. 🡪 Under Branch protection rules, click “Add rule” 🡪 In the Branch name pattern, enter: (main) the branch name 🡪 Restrict who can push to matching branches 🡪 select specific users or teams**

**Ensure code quality with SonarQube in a Jenkins pipeline before merging**

**pipeline {**

**agent linux01**

**stage('SonarQube Analysis') {**

**steps {**

**withSonarQubeEnv("${SONARQUBE\_ENV}") {**

**sh 'mvn sonar:sonar' // or 'sonar-scanner' if not using Maven**

**}**

**}**

**}**

**stage("Quality Gate") {**

**steps {**

**timeout(time: 2, unit: 'MINUTES') {**

**waitForQualityGate abortPipeline: true**

**}**

**}**

**Your Jenkins build uses git checkout and is failing with a “Detached HEAD” issue. Why and how to fix?**

**checkout([$class: 'GitSCM',**

**branches: [[name: '\*/main']],**

**userRemoteConfigs: [[**

**url: 'https://github.com/your-org/your-repo.git',**

**credentialsId: 'your-credentials-id'**

**]]**

**How do you manage secrets (tokens, credentials) in Jenkins?**

**In Jenkins 🡪 Credentials → Add Credentials**

**Use withCredentials block in Jenkinsfile:**

**withCredentials([string(credentialsId: 'my-token’) ])**

**What is a POM file? What are some important elements inside it?**

**POM (Project Object Model) is an XML file (pom.xml) that contains information about the project and configuration details.**

**<groupId> – identifies the project’s group or organization**

**<artifactId> – name of the project**

**<version> – project version**

**<dependencies> – external libraries required**

**<build> – build plugins and configurations**

**Explain Maven lifecycle phases.**

**validate, compile, test, package, verify, install, deploy**

**How to skip tests in Maven build?**

**mvn install –DskipTests**

**Explain the difference between mvn clean install and mvn clean package.**

** mvn clean package: cleans the target directory and builds the package (jar/war).**

** mvn clean install: cleans the target directory and builds the package and also installs the package into the local Maven repository for use by other projects.**

**Your project needs different configurations for dev, QA, and prod. How do you handle this using Maven?**

**Configure env under property in pom.xml**

**mvn clean install –Pdev**

**How would you integrate Maven with Jenkins?**

**under Global Tool Configuration in Jenkins**

**Also in pipeline**

**tools {**

**maven 'Maven 3.8.6' // Name from Jenkins Global Tools**

**}**

**Your Maven build is failing due to memory issues during the Jenkins pipeline execution. How will you fix it?**

**environment {**

**MAVEN\_OPTS = "-Xms512m -Xmx2048m"**

**}**

**How is a Docker container different from a virtual machine (VM)?**

**Containers virtualize the operating system so the application can run independently on any platform.**

**Virtual machines virtualize physical machines, so you can use your hardware resources efficiently.**

**How do you handle secrets in Docker?**

**Avoid hardcoding secrets in Dockerfiles or images.**

**Use environment variables or by using .env file or Kubernetes Secrets for better security.**

**Use environment variable**

**docker run -e DB\_USER=admin -e DB\_PASS=secret123 my-secure-app**

***Use .env file***

**DB\_USER=admin**

**DB\_PASS=secret123**

**Then run: docker run --env-file .env --name myContainer my-secure-app**

**Use Kubernetes Secrets**

**kubectl create secret generic my-secret --from-file=./username.txt --from-file=./password.txt**

**apiVersion: v1**

**kind: Pod**

**metadata:**

**name: my-app**

**spec:**

**containers:**

**- name: app**

**image: my-docker-image**

**volumeMounts:**

**- name: secret-volume**

**mountPath: "/etc/secrets"**

**readOnly: true**

**volumes:**

**- name: secret-volume**

**secret:**

**secretName: my-secret**

**A container crashes immediately after starting. How do you troubleshoot it?**

**Run: docker logs –f <container\_id> to see logs.**

**Check entrypoint and CMD in Dockerfile.**

**Any Errors**

**Check app may be hanging during startup (e.g., waiting on a network resource)**

**Environment variables or configs may be missing.**

**Verify restart policy: docker inspect <container\_id> | grep RestartPolicy**

**Run container in interactive mode using: docker run -it <image\_name> bash**

**Use health checks in Dockerfile to manage service behavior.**

**HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \**

**CMD curl -f http://localhost:3000/health || exit 1**

**How do you optimize Docker image?**

**Use small base image (alpine-based)**

**Clean up unnecessary files**

**RUN apt-get update && apt-get install -y curl git && rm -rf /var/lib/apt/lists/\***

**Avoid copying unnecessary files using .dockerignore**

**Use a multi-stage build.**

**Vi docerfile**

**# -------- Stage 1: Build --------**

**FROM node:18-alpine AS builder**

**# Install build dependencies**

**RUN apk add --no-cache python3 make g++**

**WORKDIR /app**

**# Copy only the files needed for dependencies**

**COPY package\*.json ./**

**# Install dependencies (including devDependencies)**

**RUN npm install**

**# Copy rest of the source code**

**COPY . .**

**# Build the application (optional, depending on app type)**

**RUN npm run build**

**# -------- Stage 2: Production --------**

**FROM node:18-alpine AS runner**

**WORKDIR /app**

**# Copy only the necessary files from the build stage**

**COPY --from=builder /app/node\_modules ./node\_modules**

**COPY --from=builder /app/dist ./dist # if you have a build output**

**COPY --from=builder /app/package\*.json ./**

**# Set the command to run the app**

**CMD ["node", "dist/index.js"]**

**You need to run multiple containers for an app (web, db, redis). What do you use?**

**Use Docker Compose to define multi-container apps in a docker-compose.yml file**

**version: '3'**

**services:**

**web:**

**build: .**

**ports:**

**- "8080:80"**

**redis:**

**image: redis**

**db:**

**image: postgres**

**Docker image is vulnerable due to an old dependency. What do you do?**

**Update vulnerable packages in the Dockerfile.**

**Rebuild and retest the image before deploying.**

**Automate vulnerability scanning in CI/CD.**

**anchore-cli image vuln ${FULL\_IMAGE} all > anchore-report.txt**

**anchore-cli evaluate check ${FULL\_IMAGE}**

**You need zero downtime deployment. How do you achieve this using Docker?**

**Use blue-green deployment: deploy a new version alongside the old, switch traffic after verification.**

**Or canary deployment: slowly route a percentage of traffic to the new version. Combine with Kubernetes rolling updates and health checks.**

**CI/CD Pipeline Uses Docker?**

**Build image: docker build -t app:v1 .**

**Push to registry: docker push registry/app:v1**

**Automate vulnerability scanning using jfrog xray / anchore**

**Use Docker Compose for integration testing**

**A container is running, but you can't access the application from the browser. What do you check?**

**Use docker ps to confirm port mapping**

**o/p 🡪**

**CONTAINER ID IMAGE COMMAND PORTS NAMES**

**b3c0e0e16f3a nginx "nginx -g 'daemon of…" 0.0.0.0:8080->80/tcp my-nginx**

**Ensure port is published: docker run -p 8080:8080**

**Check the app is binding to 0.0.0.0 not localhost**

**In Dockerfile or docker-compose.yml 🡪 command: uvicorn app.main:app --host 0.0.0.0 --port 8000**

**docker exec -it <container\_name> netstat –tuln**

**o/p 🡪 tcp 0 0 0.0.0.0:5000 0.0.0.0:\* LISTEN**

**container is generating too many logs. How do you manage this?**

**Use log rotation tools**

**docker run --log-driver json-file --log-opt max-size=100m --log-opt max-file=3 your-image**

**The log come to /var/lib/docker/containers/<container-id>/<container-id>-json.log**

**Or**

**docker run -v /var/log/myapp:/logs your-image > /logs/app.log 2>&1**

**Rolling Updates with Zero Downtime using ansible and k8**

**K8s Deployment YAML**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: myapp**

**spec:**

**replicas: 3**

**strategy:**

**type: RollingUpdate**

**rollingUpdate:**

**maxUnavailable: 0 # Ensure zero downtime**

**maxSurge: 1 # Add one extra pod during update**

**selector:**

**matchLabels:**

**app: myapp**

**template:**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**containers:**

**- name: myapp**

**image: myapp:1.0**

**ports:**

**- containerPort: 80**

**Anisble to perform rolling update for new app version**

**- name: Rolling update with image override**

**hosts: localhost**

**vars:**

**app\_name: myapp**

**container\_name: myapp**

**tasks:**

**- name: Set new image on deployment**

**ansible.builtin.command:**

**cmd: kubectl set image deployment/myapp focusContainer=myapp:{{ new\_version }}**

**rollout status via Ansible:**

**- name: Wait for rollout to complete**

**ansible.builtin.command:**

**cmd: kubectl rollout status deployment/myapp**

**register: rollout\_status**

**retries: 10**

**delay: 6**

**until: "'successfully rolled out' in rollout\_status.stdout"**

**Ansible project with environment-specific inventory files and group-specific variables?**

**ansible-project/**

**├── inventory/**

**│ ├── hosts.yaml**

**│ ├── dev/**

**│ │ └── group\_vars/**

**│ │ └── dev.yml**

**│ ├── qa/**

**│ │ └── group\_vars/**

**│ │ └── qa.yml**

**├── thePlaybook.yml**

**all:**

**children:**

**dev:**

**hosts:**

**dev-web1:**

**ansible\_host: 192.168.1.10**

**qa:**

**hosts:**

**qa-web1:**

**ansible\_host: 192.168.2.10**

**inventory/dev/group\_vars/dev.yml**

**# Variables specific to devServers group in the dev environment**

**env\_name: dev**

**inventory/qa/group\_vars/qa.yml**

**# Variables specific to qaServers group in the qa environment**

**env\_name: qa**

**thePlaybook.yaml**

**- name: Deploy to environment-specific hosts**

**hosts: all**

**gather\_facts: no**

**tasks:**

**- name: Show environment-specific config**

**debug:**

**msg: "Environment: {{ env\_name }}”**

**ansible-playbook -i inventory/hosts.yaml thePlaybook.yaml -l dev**

**Conditional Tasks in ansible?**

**- name: Only install apache on Debian/Ubuntu**

**apt:**

**name: apache2**

**state: latest**

**when: ansible\_distribution == 'Ubuntu'**

**Copy a file only if a specific file exists in ansible?**

**- name: Check if the source file exists**

**stat:**

**path: /path/to/source/file.txt**

**register: source\_file\_check**

**- name: Copy file only if the source file exists**

**copy:**

**src: /path/to/source/file.txt**

**dest: /path/to/destination/file.txt**

**when: source\_file\_check.stat.exists**

**Looping using Ansible?**

**# Incase of with\_nested loop for each outer element all the inner element will be executed**

**# with\_subelements is a loop construct used to iterate over a list of dictionaries and then over a sublist within each dictionary.**

**# with\_items iterate over a list of items.**

**---**

**- hosts: yamlnodes**

**become: true**

**tasks:**

**- name: create user using with\_subelements by expanding the subelements**

**user:**

**name: "{{ item.1 }}"**

**comment: "{{ item.1 | title }} {{ item.0.comments }}" 🡪 # item.0 is the parent dictionary**

**with\_subelements:**

**-**

**- comments: this is normal user**

**username:**

**- sumantakumar**

**- akhilesh**

**- comments: this is group user**

**username:**

**- smita**

**- username 🡪 🡪 # tells Ansible the key holds subelements.**

**tags: expand\_subelement**

**##### O/P 🡪**

**##### item.0 = { comments: "this is normal user", username: ["sumantakumar", "akhilesh"] }**

**##### item.1 = "sumantakumar"**

**- name: create the user directory**

**file:**

**dest: "/home/{{ item.0 }}/{{ item.1 }}"**

**owner: "{{ item.0 }}"**

**group: "{{ item.0 }}"**

**state: directory**

**with\_nested:**

**- [ sumantakumar , akhilesh , smita ]**

**- [ devops , movie , photo ]**

**tags: create\_dir**

**- name: delete user using with\_items**

**user:**

**name: "{{ item }}"**

**state: absent**

**remove: yes**

**with\_items: # list argument in yaml form**

**- sumantakumar**

**- smita**

**- akhilesh**

**tags: delete\_user**

**$ df –h command output**

**Filesystem Size Used Avail Use% Mounted on**

**/dev/sda1 50G 25G 23G 53% / dev/shm**

**ls –lrt – List files by time in reverse order**

**-rw-r--r-- 1 user user 123 Jul 25 12:00 file1.txt**

**ps -f – Process Status (full-format listing)**

**UID PID PPID C STIME TTY TIME CMD**

**user 11300 11234 0 08:46 pts/0 00:00:01 python3 script.py**

**What is the syntax of a crontab entry?**

**Answer:**

**pgsql**

**\* \* \* \* \* command-to-be-executed**

**- - - - -**

**| | | | |**

**| | | | +----- Day of the week (0 - 7) (Sunday is 0 or 7)**

**| | | +------- Month (1 - 12)**

**| | +--------- Day of the month (1 - 31)**

**| +----------- Hour (0 - 23)**

**+------------- Minute (0 - 59)**

**How can you run a job every 10 minutes between 8 AM and 10 AM?**

**Answer:**

**\*/10 8-10 \* \* \* /path/to/command.sh**

**Backup a Directory Every Day at Midnight**

**0 0 \* \* \* tar -czf /backups/home\_backup\_$(date +\%F).tar.gz /home/user/**

**Clear Temp Files Every Sunday at 2 AM**

**0 2 \* \* 0 rm -rf /tmp/\***

**Restart a Web Server if Down**

**\*/5 \* \* \* \* pgrep nginx > /dev/null || systemctl restart nginx**

**Send Disk Usage Alert If Over 90%**

**df -h / | awk '$5+0 > 90 {system("echo Disk Full | mail -s \"Disk Alert\" admin@example.co**

**Clean Log Files Older Than 7 Days**

**find /var/log/myapp/ -type f -mtime +7 -delete**

**Bottom of Form**