**DevOps & Git Interview Q&A**

### 1. What does git add do and why is it important before committing?

* git add moves changes from the working directory to the staging area (index).
* Importance:
  + Lets you decide which changes to include in a commit.
  + Prevents accidental commits of unwanted files.

### 2. How do you stage only specific changes in Git?

* Stage a specific file:
* git add file1.yaml
* Stage multiple specific files:
* git add file1.yaml file2.yaml
* Stage certain lines interactively:
* git add -p file.yaml

### 3. If you have to update 2 YAML files in a remote repo as a new joiner, how will you do it?

Steps:

1. Clone the repository:

* git clone <repo-url>  
  cd <repo-name>

1. Create a new branch:

* git checkout -b update-yaml-files

1. Edit the YAML files locally.
2. Stage only the updated files:

* git add file1.yaml file2.yaml

1. Commit with a meaningful message:

* git commit -m "Updated configuration in file1.yaml and file2.yaml"

1. Push your branch to remote:

* git push origin update-yaml-files

1. Raise a Pull Request (PR) for review.

### 4. What is Git stash & when do you use it?

* git stash temporarily saves uncommitted changes and cleans your working directory.
* Use cases:
  + Switch branches quickly without committing.
  + Pull latest changes without losing local progress.
* Commands:
* git stash # save changes  
  git stash list # view stashes  
  git stash apply # reapply last stash  
  git stash pop # reapply & remove from stash

### 5. Difference between git fetch & git pull

* git fetch: Downloads new commits from remote **without merging**.
* git pull: Fetches + merges changes into your local branch.
* Formula: git pull = git fetch + git merge

### 6. What is git cherry-pick & when would you use it?

* git cherry-pick <commit-hash> applies a specific commit from one branch to another.
* Use cases:
  + Apply a bug fix from a feature branch to main.
  + Copy urgent hotfixes to another branch.

Example:

git checkout main  
git cherry-pick a1b2c3d

### 7. Explain your experience in CI/CD tools and how you used it.

* Worked with **Jenkins, GitHub Actions, GitLab CI/CD**.
* Built pipelines for automated build, test, and deployment.
* Integrated **Docker, Kubernetes, SonarQube, and security scans** into pipelines.
* Automated deployment to AWS EKS using Helm charts.

### 8. Give me some commands in Linux

* List files: ls -l
* Change directory: cd /path
* Disk usage: df -h
* Check memory: free -m
* Find a process: ps -ef | grep <process>

### 9. Command to generate SSH Key

ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"

### 10. What if user lost SSH key?

* Generate a new SSH key.
* Add the new public key to GitHub/GitLab/Server authorized\_keys.
* Remove old invalid keys from the system.

### 11. Command to show memory usage & CPU processing

top # real-time view  
htop # interactive (if installed)  
free -h # memory usage

### 12. Command to kill a process

kill -9 <PID>

### 13. How will you change user access or privileges?

* Modify user in Linux:
* usermod -aG <group> <username>  
  chmod 755 <file>  
  chown user:group <file>

### 14. What are GitHub Actions?

* A CI/CD tool integrated with GitHub.
* Automates workflows like building, testing, and deploying applications.
* Uses **YAML workflows** stored in .github/workflows/.

### 15. Difference between GitHub Actions & Jenkins

* **GitHub Actions:** Cloud-native, tightly integrated with GitHub, easy YAML workflows.
* **Jenkins:** Open-source server, plugin-rich, can integrate with any VCS (not limited to GitHub).

### 16. Any Branching strategies you followed in your organization?

* **GitFlow**: Feature, Develop, Release, Hotfix branches.
* **Trunk-Based Development**: Small frequent commits directly to main with feature toggles.

### 17. How much are you confident in Kubernetes & Docker?

* Confident in **building Docker images, managing containers**, deploying apps in Kubernetes, creating services, ingresses, and handling scaling.

### 18. How will you stop a POD in Kubernetes?

kubectl delete pod <pod-name>

### 19. How will you replicate a POD?

* By scaling deployment:

kubectl scale deployment <deployment-name> --replicas=3

### 20. Command to get logs in Kubernetes

kubectl logs <pod-name>

### 21. What will you do if a POD is not responding?

* Check pod status: kubectl describe pod <pod>
* Check events & logs.
* Restart pod if needed: kubectl delete pod <pod> (deployment will recreate it).

### 22. What will you do if a POD is getting more load?

* Scale horizontally: kubectl scale deployment <dep> --replicas=n
* Enable **Horizontal Pod Autoscaler (HPA)**.
* Optimize pod resources (CPU/memory requests & limits).

### 23. What is Docker?

* A platform to build, package, and run applications in lightweight containers.
* Ensures consistency across environments.

### 24. How is Docker useful & how will you implement in your pipeline?

* **Useful:** Provides reproducible environments, faster deployments, and scalability.
* **Implementation:** Build Docker images in CI pipeline → push to registry → deploy on Kubernetes using Helm/Manifests.

### 25. How will you find out merge conflicts?

* While merging: Git shows conflict markers <<<<<<< HEAD.
* Command:
* git status
* → Lists conflicted files.

### 26. What kind of tools will you prefer for SAST & DAST security?

* **SAST (Static):** SonarQube, Checkmarx, Bandit.
* **DAST (Dynamic):** OWASP ZAP, Burp Suite, Nessus.

### 27. How will you manage your ServiceNow task?

* Prioritize tasks based on severity & SLA.
* Pick tasks relevant to sprint/assigned team.
* Document updates, collaborate with stakeholders, and close with resolution notes.