DevOps Interview Preparation Guide

Jenkins, Ansible & Terraform Q&A

Table of Contents

Jenkins Questions

🟢 Beginner Level

Q1: What is Jenkins?

- **Answer:** Jenkins is an open-source automation server used for Continuous Integration (CI) and Continuous Deployment (CD). It helps automate building, testing, and deploying applications, enabling faster and more reliable software delivery.

Q2: What are the main features of Jenkins?

- **Answer:**
  - Easy installation and configuration
  - Hundreds of plugins available
  - Distributed builds support
  - Easy upgrades
  - Platform independent (Java-based)
  - Web-based GUI

Q3: What is a Jenkins job/project?

- **Answer:** A Jenkins job is a runnable task that Jenkins can execute. It contains configuration details like source code location, build triggers, build steps, and post-build actions.

Q4: What are Jenkins plugins? Name some popular ones.

- **Answer:** Plugins extend Jenkins functionality. Popular plugins include:

- Git Plugin (version control)
- Maven Integration Plugin
- Pipeline Plugin
- Docker Plugin
- Blue Ocean (modern UI)
- Email Extension Plugin

Q5: What is Jenkins Workspace?

- **Answer:** A workspace is a directory on the Jenkins agent where the build job runs. It contains source code, build artifacts, and temporary files for that specific job.

Q6: What is the difference between Jenkins and Hudson?

- **Answer:** Hudson was the original project. Jenkins is a fork of Hudson created in 2011 due to disputes over governance. Jenkins became more popular and is actively maintained.

🟡 Intermediate Level

Q7: What is Jenkins Pipeline? Explain Declarative vs Scripted Pipeline.

- **Answer:**
  - **Pipeline:** A suite of plugins supporting continuous delivery pipelines in Jenkins
  - **Declarative Pipeline:** Uses a more structured syntax with predefined sections (pipeline, agent, stages, steps). Easier to read and validate.
  - **Scripted Pipeline:** Uses Groovy syntax, more flexible but complex. Starts with `node` block.

Q8: What are Jenkins Build Triggers? Name different types.

- **Answer:** Build triggers determine when a Jenkins job should run:
  - **Poll SCM:** Checks version control periodically
  - **Webhook Triggers:** External systems trigger builds
  - **Build after other projects:** Downstream builds
  - **Build periodically:** Cron-like scheduling
  - **GitHub/GitLab hooks:** Automatic triggers on code changes

Q9: What is Jenkins Master-Slave architecture?

- **Answer:**
  - **Master:** Controls the pipeline, schedules builds, dispatches jobs
  - **Slave/Agent:** Executes build jobs assigned by master
  - Benefits: Distributed builds, parallel execution, platform diversity

Q10: How do you secure Jenkins?

- **Answer:**
  - Enable security and authentication
  - Use authorization strategies (Matrix-based, Role-based)
  - Implement HTTPS/SSL
  - Regular updates and security patches
  - Use credentials plugin for sensitive data
  - Network security (firewall, VPN)

Q11: What is Blue Ocean in Jenkins?

- **Answer:** Blue Ocean is a modern, intuitive UI for Jenkins pipelines. It provides:
  - Visual pipeline editor
  - Better pipeline visualization
  - Faster navigation
  - Mobile-friendly interface

🔴 Advanced Level

Q12: How do you implement Jenkins Pipeline as Code?

- **Answer:** Using Jenkinsfile stored in source control:

```groovy
```

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'make build'
            }
        }
        stage('Test') {
            steps {
                sh 'make test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'make deploy'
            }
        }
    }
}
```

Q13: How do you handle failures and error handling in Jenkins Pipeline?

- **Answer:**
  - Use `try-catch` blocks in scripted pipeline
  - `post` sections in declarative pipeline
  - `catchError` step to continue pipeline on failure
  - Conditional execution with `when` directive
  - Retry mechanisms and timeout settings

Q14: What is Jenkins Shared Library?

- **Answer:** Shared Libraries allow sharing common pipeline code across multiple projects. They:
  - Promote code reuse
  - Maintain consistency
  - Enable centralized updates
  - Support custom steps and variables

Q15: How do you optimize Jenkins performance?

- **Answer:**
  - Increase JVM heap size
  - Use distributed builds

- Clean up old builds regularly
- Optimize plugin usage
- Use SSD storage
- Monitor system resources
- Implement build caching

Ansible Questions

🟢 Beginner Level

Q16: What is Ansible?

- **Answer:** Ansible is an open-source automation tool for configuration management, application deployment, and task automation. It uses SSH for communication and doesn't require agents on target machines.

Q17: What are the main components of Ansible?

- **Answer:**
  - **Control Node:** Machine where Ansible is installed
  - **Managed Nodes:** Target machines managed by Ansible
  - **Inventory:** List of managed nodes
  - **Modules:** Units of code executed by Ansible
  - **Playbooks:** YAML files containing automation instructions

Q18: What is an Ansible Playbook?

- **Answer:** A playbook is a YAML file containing a series of plays. Each play maps a group of hosts to well-defined tasks using modules.

Q19: What is Ansible Inventory?

- **Answer:** Inventory is a file containing information about target hosts/servers. It can be static (INI/YAML files) or dynamic (scripts/plugins that query external systems).

Q20: What are Ansible Modules? Give examples.

- **Answer:** Modules are discrete units of code that Ansible executes. Examples:
  - **copy:** Copy files to remote hosts
  - **service:** Manage services
  - **yum/apt:** Package management

- **file:** File/directory management
- **command/shell:** Execute commands

🟡 Intermediate Level

Q21: What are Ansible Roles? Why use them?

- **Answer:** Roles are reusable units of organization for playbooks. Benefits:
  - Code reusability
  - Better organization
  - Sharing across teams
  - Simplified maintenance
  - Standard directory structure

Q22: What is Ansible Vault?

- **Answer:** Ansible Vault encrypts sensitive data like passwords, keys, and certificates. Commands:
  - `ansible-vault create` - Create encrypted file
  - `ansible-vault encrypt` - Encrypt existing file
  - `ansible-vault edit` - Edit encrypted file
  - `ansible-vault decrypt` - Decrypt file

Q23: Difference between Ansible and other configuration management tools?

- **Answer:**
  - **Agentless:** No agents needed (vs Puppet, Chef)
  - **Push-based:** Control node pushes configs
  - **YAML syntax:** Human-readable
  - **SSH-based:** Uses existing SSH infrastructure
  - **Simpler learning curve**

Q24: What are Ansible Facts?

- **Answer:** Facts are system information automatically gathered by Ansible about managed nodes. Examples: OS version, IP addresses, memory, disk space. Access via `ansible_facts` or `setup` module.

Q25: How do you handle different environments in Ansible?

- **Answer:**
  - Separate inventory files for each environment
  - Group variables (group_vars directory)
  - Host variables (host_vars directory)

- Environment-specific playbooks
- Conditional execution based on groups

🔴 Advanced Level

Q26: What are Ansible Custom Modules? How do you create one?

- **Answer:** Custom modules extend Ansible functionality. Created using Python:

```python
#!/usr/bin/python
from ansible.module_utils.basic import AnsibleModule

def main():
    module = AnsibleModule(
        argument_spec=dict(
            name=dict(required=True, type='str'),
        )
    )
    # Module logic here
    module.exit_json(changed=True, msg="Success")

if __name__ == '__main__':
    main()
```

Q27: How do you optimize Ansible playbook performance?

- **Answer:**
  - Use `strategy: free` for parallel execution
  - Enable SSH pipelining
  - Use `async` and `poll` for long-running tasks
  - Minimize fact gathering (`gather_facts: no`)
  - Use connection multiplexing
  - Implement proper error handling

Q28: What is Ansible Tower/AWX?

- **Answer:** Enterprise web-based interface for Ansible:
  - Web UI and REST API
  - RBAC (Role-Based Access Control)
  - Job scheduling
  - Workflow management

- Credential management
- Audit logging

Terraform Questions

🟢 Beginner Level

Q29: What is Terraform?

- **Answer:** Terraform is an Infrastructure as Code (IaC) tool that allows you to define, provision, and manage infrastructure using declarative configuration files (HCL - HashiCorp Configuration Language).

Q30: What are the main Terraform commands?

- **Answer:**
  - `terraform init` - Initialize working directory
  - `terraform plan` - Show execution plan
  - `terraform apply` - Apply changes
  - `terraform destroy` - Destroy infrastructure
  - `terraform fmt` - Format configuration files
  - `terraform validate` - Validate configuration

Q31: What is Terraform State?

- **Answer:** Terraform state is a JSON file that keeps track of resources Terraform manages. It maps real-world resources to your configuration and tracks metadata.

Q32: What are Terraform Providers?

- **Answer:** Providers are plugins that enable Terraform to interact with APIs of cloud providers, SaaS providers, and other services. Examples: AWS, Azure, Google Cloud, GitHub.

Q33: What is HCL (HashiCorp Configuration Language)?

- **Answer:** HCL is Terraform's configuration language. It's human-readable and machine-friendly, using blocks, arguments, and expressions to define infrastructure.

🟡 Intermediate Level

Q34: What are Terraform Modules?

- **Answer:** Modules are reusable Terraform configurations. They help:
  - Organize configuration

- Encapsulate groups of resources
- Enable reusability across projects
- Provide abstraction

Q35: What is Remote State in Terraform? Why use it?

- **Answer:** Remote state stores Terraform state file in a remote location (S3, Azure Blob, etc.). Benefits:
  - Team collaboration
  - State locking
  - Better security
  - Backup and versioning

Q36: What are Terraform Variables and Outputs?

- **Answer:**
  - **Variables:** Input parameters for modules/configurations
  - **Outputs:** Return values from modules/configurations
  - Enable parameterization and data sharing between modules

Q37: What is Terraform Workspace?

- **Answer:** Workspaces allow managing multiple environments (dev, staging, prod) with the same configuration but separate state files.

Q38: Difference between Terraform and CloudFormation?

- **Answer:**
  - **Terraform:** Multi-cloud, HCL syntax, larger community
  - **CloudFormation:** AWS-specific, JSON/YAML, native AWS integration
  - **Terraform:** State management, plan before apply
  - **CloudFormation:** Stack-based, rollback capabilities

🔴 Advanced Level

Q39: How do you manage Terraform State locks?

- **Answer:**
  - Use remote backends with locking (DynamoDB for S3)
  - State locks prevent concurrent modifications
  - Force unlock only when necessary (`terraform force-unlock`)
  - Monitor lock timeouts and deadlocks

Q40: What are Terraform Data Sources?

- **Answer:** Data sources fetch information from existing resources not managed by current Terraform configuration:

```hcl
data "aws_ami" "example" {
  most_recent = true
  owners      = ["self"]

  filter {
    name   = "name"
    values = ["myami-*"]
  }
}
```

Q41: How do you implement Terraform CI/CD pipeline?

- **Answer:**
  - Version control Terraform configurations
  - Automated `terraform plan` in PR/MR
  - Automated `terraform apply` after approval
  - State file security and backup
  - Environment-specific pipelines
  - Policy validation (Sentinel, OPA)

Q42: What is Terraform Import?

- **Answer:** `terraform import` brings existing infrastructure under Terraform management:

```bash
terraform import aws_instance.example i-1234567890abcdef0
```

Requires writing configuration that matches the existing resource.

General DevOps Concepts

Key Questions for DevOps Interviews

Q43: What is DevOps?

- **Answer:** DevOps is a cultural and professional movement that emphasizes collaboration between development and operations teams, automation, continuous integration/delivery, and rapid, frequent deployment of software.

Q44: What is CI/CD?

- **Answer:**
  - **CI (Continuous Integration):** Frequently integrating code changes into shared repository
  - **CD (Continuous Delivery/Deployment):** Automated deployment to production or staging environments

Q45: What is Infrastructure as Code (IaC)?

- **Answer:** IaC manages and provisions computing infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Q46: Difference between Containerization and Virtualization?

- **Answer:**
  - **Virtualization:** Multiple OS on single hardware
  - **Containerization:** Multiple applications sharing single OS kernel
  - Containers are lighter, faster, and more efficient

Interview Tips for Beginners

Preparation Strategy

1. **Understand Fundamentals:** Focus on core concepts before diving into advanced topics
2. **Hands-on Practice:** Set up local labs and practice with real scenarios
3. **Documentation:** Read official documentation for each tool
4. **Community Resources:** Join DevOps communities and forums

During the Interview

1. **Be Honest:** Don't pretend to know something you don't
2. **Think Aloud:** Explain your thought process
3. **Ask Questions:** Show curiosity and engagement
4. **Practical Examples:** Use real-world scenarios when possible

Common Beginner Mistakes to Avoid

1. Memorizing without understanding

2. Focusing only on theory without practice
3. Not understanding the "why" behind tools and practices
4. Overlooking security and best practices

Resources for Continued Learning

- Official documentation websites
- GitHub repositories with example code
- Online labs and sandbox environments
- DevOps certification programs
- Community forums and Slack channels

Practice Labs Suggestions

Jenkins

- Set up Jenkins on local machine/VM
- Create simple CI pipeline
- Integrate with Git repository
- Practice with different build tools

Ansible

- Set up control and managed nodes
- Write basic playbooks
- Create and use roles
- Practice with different modules

Terraform

- Create AWS/Azure free tier account
- Write basic infrastructure code
- Practice with modules
- Implement remote state

Good luck with your DevOps interview! Remember to practice hands-on scenarios and understand the concepts deeply rather than just memorizing answers.