Ansible Tasks

1. **Creating/editing & deleting Files (Normal, config, Link, Hidden, zip, tar) in hundreds of Ansible nodes**:
   Using Ansible modules like file and copy, you can create, edit, and delete various types of files across multiple nodes. This saves time by automating file management tasks on hundreds of servers.

2. **Creating, editing & deleting Directories in Ansible Nodes**:
   You can use the file module with the state: directory parameter to manage directories in Ansible nodes. This ensures consistent directory structures across environments.

3. **Managing Files & Folder permissions (Read-write & execute access) in Ansible Nodes**:
   Ansible allows you to manage file permissions using the mode parameter in the file module. This is useful for setting secure permissions on files and directories.

4. **User Administration**:
   Ansible's user module helps in creating and managing user accounts, groups, and permissions across multiple nodes, making administration tasks simpler.

5. **Creating users, groups, permissions, adding, and removing users from groups & changing ownerships**:
   You can automate user and group management using the user and group modules in Ansible, ensuring consistent user permissions and ownership across all nodes.

6. **Installing, Uninstalling, Upgrading, and downgrading packages in Ansible Nodes**:
   With the apt (for Ubuntu/Debian) or yum (for RedHat/CentOS) modules, you can automate software package management, ensuring that the right versions are installed on all servers.

7. **Managing services (Starting/Stopping/Reloading) in Ansible Nodes**:
   Use the service module to start, stop, or reload services, ensuring that they are running or stopped as needed.

8. **Taking backups (copy, hard link files) in Ansible Nodes**:
   Ansible can automate the process of taking backups using the copy or hard link options, allowing you to save important files safely.

9. **Installing big applications (Web, Tomcat, Java, Python.....) in Ansible Nodes**:
   Using playbooks, you can install complex applications like web servers, Tomcat, Java, and Python across many nodes, ensuring all necessary configurations are applied.

10. **Executing shell scripts in Ansible Nodes**:
    Ansible's shell or command module can execute custom shell scripts across nodes, making it easy to automate script-based tasks.

11. **Collecting Info (Like network) from nodes in Ansible Nodes**:
    Use the setup module to gather information about nodes, such as network details, disk usage, and system configurations.
12. **Pushing HTML files to Ansible Nodes**:
    The copy module allows you to upload website-related files like HTML to the nodes, ensuring the latest version of the website is deployed.
13. **Cloning GitHub/Bitbucket repos in Ansible Nodes**:
    The git module can be used to clone repositories from GitHub or Bitbucket into nodes, making it easier to deploy code.
14. **Downloading packages from the internet in Ansible Nodes**:
    You can automate package downloading using the get_url module, which pulls files or software packages from the internet.
15. **Linking to Jenkins in CI-CD process for deployment of .war, .jar files**:
    Ansible can be integrated with Jenkins for CI/CD, automating the deployment of .war or .jar files to various environments.
16. **CI-CD deployment in all environments (Dev, Test, Prod...)**:
    Ansible is often used to deploy applications across development, testing, and production environments, ensuring consistency and reliability.
17. **New admin tasks which emerge every day**:
    As new tasks arise, Ansible allows you to quickly create playbooks to automate repetitive administrative work.
18. **Providing training sessions to Freshers & documenting new inventions**:
    Training sessions on Ansible help freshers learn automation concepts. Documenting new findings ensures future team members can easily understand new methods.
19. **Resolving Jira tickets raised by staff**:
    Ansible can be used to resolve issues tracked in Jira, whether they are about file permissions, service restarts, or user management.
20. **Taking remote sessions (RDP & SSH) of Team member's machines to assess issues that arise daily**:
    Remote troubleshooting can be facilitated using SSH, and Ansible can help automate common fixes across multiple machines.

Ansible Errors and Solutions

1. **Forget to give hosts file (Inventory file) path while running playbooks**:
   Solution: Always specify the inventory file path using the -i option when running playbooks.
2. **Not mentioning the IP of Ansible nodes in the inventory file**:
   Solution: Ensure that the IP addresses of all Ansible nodes are correctly listed in the inventory file.
3. **Not creating common users and common passwords in all Ansible nodes**:
   Solution: Create a common user and password across all nodes using the user module, and ensure consistency for easier management.
4. **Not giving sudo privileges to Ansible users**:
   Solution: Use become: yes in playbooks to ensure sudo privileges for running tasks that require elevated permissions.
5. **Not editing SSHD config file to enable SSH connection between Ansible Server and Nodes**:
   Solution: Ensure the SSHD config file is edited and reloaded using service: name=sshd state=reloaded to allow SSH access.
6. **Not copying public key properly to nodes**:
   Solution: Use the authorized_key module to properly copy SSH keys to all nodes for passwordless access.
7. **Using the same modules for different OS family servers**:
   Solution: Use when conditions or ansible_facts to run OS-specific modules, ensuring compatibility across different operating systems.
8. **Syntax errors in code while writing YAML script**:
   Solution: Validate YAML syntax using tools like yamllint before running playbooks to avoid errors.
9. **Not mentioning the "Become: yes" parameter in the playbook**:
   Solution: Ensure become: yes is included in tasks that require elevated privileges to avoid permission errors.
10. **Get errors if forget to change variable names**:
    Solution: Double-check variable names and ensure they are unique and correctly defined in playbooks.
11. **Get errors if the Handler section name mismatches**:
    Solution: Ensure that the handler names are correctly referenced in the playbook tasks to avoid mismatches.

12. **Not checking playbooks by using a dry run**:
    Solution: Use --check for a dry run to test playbooks before executing them, ensuring tasks behave as expected without making changes.
13. **Not giving the "Gather facts" parameter in Playbooks "Yes"**:
    Solution: Enable gather_facts: yes at the beginning of playbooks when working with OS-specific installations or environment-specific tasks.
14. **Not encrypting playbooks if any sensitive data is there**:
    Solution: Use ansible-vault to encrypt playbooks containing sensitive data like passwords or API keys.
15. **Not understanding Ansible Roles directory structure properly**:
    Solution: Learn the Ansible roles structure and use the ansible-galaxy init command to create roles with the correct folder structure.
16. **Not installing Ansible properly**:
    Solution: Follow the correct installation steps based on the operating system, and verify installation with ansible --version.
17. **Not giving proper permissions to users and groups**:
    Solution: Ensure correct file and folder permissions using the mode, owner, and group parameters in the file module to avoid access issues.

Ansible Questions

1. **Exactly what is Configuration Management?**
   Configuration Management is the process of managing and maintaining systems' configurations, ensuring that software and hardware are in a consistent, known state.
2. **Why do we need a Configuration Management tool?**
   Configuration Management tools like Ansible help automate tasks, reduce human errors, ensure consistency, and speed up the deployment and management of systems.
3. **Configuration Management tool offers what benefits?**
   It offers automation, consistency, version control, and easy rollback of configurations, helping in managing infrastructure at scale.
4. **Currently, what are the available tools for Configuration Management?**
   Popular Configuration Management tools include Ansible, Puppet, Chef, and SaltStack.
5. **What is the process for installing Ansible? What are the steps to follow?**
   You can install Ansible using package managers like apt for Ubuntu or yum for CentOS. After installation, verify it with ansible --version.

6. **Can any user be granted sudo privileges?**
   Yes, users can be granted sudo privileges by adding them to the /etc/sudoers file, allowing them to execute commands with administrative rights.

7. **How do I establish an SSH connection between two Linux servers?**
   Use the ssh command to connect to a remote server, and ensure that both servers have SSH installed and running.

8. **How do I establish a passwordless connection (trust relationship) between Linux servers?**
   Use ssh-keygen to generate a key pair and then copy the public key to the remote server's authorized_keys file for passwordless login.

9. **In Ansible, what is the purpose of the Inventory file?**
   The inventory file lists the servers (nodes) that Ansible will manage. It contains their IP addresses or hostnames.

10. **Can you tell me what Ansible's main configuration file is?**
    Ansible's main configuration file is ansible.cfg, which defines default settings like inventory path and SSH settings.

11. **What is the Host pattern in Ansible?**
    Host patterns define which nodes a playbook will run on. You can specify all nodes, specific groups, or individual hosts.

12. **What are Ansible Ad-Hoc commands? What makes them different from Playbooks?**
    Ad-Hoc commands are one-time, quick commands executed directly in the terminal, while Playbooks are saved scripts that automate tasks and can be reused.

13. **What are modules in Ansible? What are some of the module names you used?**
    Modules are reusable, standalone scripts that perform specific tasks in Ansible. Commonly used modules include apt, yum, file, and service.

14. **What is the set-up module?**
    The setup module collects facts (information) about a system, such as its IP address, operating system, and memory.

15. **Can you tell me what Ansible playbooks are and how they work?**
    Playbooks are YAML scripts that define tasks to be performed on remote nodes. They consist of tasks, handlers, variables, and other sections.

16. **In the Ansible playbook, what sections are available?**
    Playbook sections include tasks, handlers, variables, roles, and conditional logic (like when).

17. **Could you please write down all the playbooks you have used at your company?**
    Playbooks vary, but examples include web server setup, package installation, and user management. They are customized based on project requirements.

18. **What are the playbooks you used in your project?**
    I have used playbooks for deploying web servers, managing databases, handling user permissions, and automating software installation.

19. **How do you write a web server playbook?**
    A web server playbook includes installing the necessary packages (like Apache), configuring the server, and starting the service using modules like apt and service.

20. **What is the difference between a task section and a handler section?**
    Tasks are run immediately during playbook execution, while handlers are triggered only when notified by tasks (e.g., restarting a service after configuration changes).

21. **Variables in playbooks: what are they and how do they work?**
    Variables allow you to store data and reuse it across tasks in playbooks, making the code more dynamic and reusable.

22. **Using a playbook, how do I set up Tomcat?**
    You can use a playbook to install Tomcat by downloading the necessary packages, configuring the service, and starting it with tasks.

23. **How does a playbook handle errors?**
    Playbooks can handle errors using the ignore_errors: yes directive, which allows playbook execution to continue even if a task fails.

24. **How does Ansible's dry run work?**
    A dry run (--check) simulates the playbook execution without making actual changes, allowing you to test the playbook first.

25. **In a playbook, how do you use loops?**
    Loops allow tasks to be repeated multiple times, typically used for installing multiple packages or creating multiple users.

26. **What is the purpose of the when condition in playbooks?**
    The when condition allows tasks to run only if certain conditions are met, making playbooks more flexible.

27. **Can you tell me what Ansible-Vault is? How significant is it?**
    Ansible-Vault is used to encrypt sensitive data like passwords, making playbooks secure when dealing with confidential information.

28. **Can you explain Ansible's roles in detail?**
    Ansible roles organize playbooks into reusable components, each handling a specific part

of the configuration (e.g., web server, database). They make managing complex playbooks easier.

29. **Which version of Ansible that you are using in your company?**
    The version depends on the project's needs, but we typically use the latest stable version to ensure compatibility with modern features and modules. You can check with ansible --version.