# 

# Install and setup the ansible in 3 instances one has master and other two as nodes, configure the communication between the master and nodes, also give me an example of ansible playbook to install idk 11 in them using hosts.yaml ?

### **Installing and Setting up Ansible with a Master and Two Nodes**

In this setup, we will install and configure **Ansible** on a **Master node**, and two **Node instances** (e.g., node1 and node2). The nodes will be managed remotely from the master instance using SSH.

### **Assumptions:**

1. You have 3 Linux instances:
   * **Master Node (Ansible control machine)**: This instance will manage the other nodes.
   * **Node 1 and Node 2 (Managed nodes)**: These are the target systems where you'll install software, in this case, **Oracle JDK 11**.
2. All instances are reachable via SSH, and you have **root or sudo** access to all of them.
3. You will use a simple **inventory file (hosts.yaml)** to define the master and nodes and an **Ansible Playbook** to automate the installation of **Oracle JDK 11**.

### **Step 1: Install Ansible on the Master Node**

#### **On the Master Node (Control Machine):**

**Update your package list** (for Debian/Ubuntu-based systems):  
bash  
Copy code  
sudo apt update

1. **Install Ansible**:

On Ubuntu/Debian:  
bash  
Copy code  
sudo apt install ansible -y

On CentOS/RHEL:  
bash  
Copy code  
sudo yum install epel-release -y

sudo yum install ansible -y

Verify the installation by checking the version:  
bash  
Copy code  
ansible --version

1. This should show the version of Ansible installed.

### **Step 2: Set Up Communication Between Master and Nodes**

#### **1. Set up SSH Key-Based Authentication for Passwordless SSH:**

To manage the nodes via SSH, you need to configure **SSH key-based authentication** from the master to the nodes.

**Generate SSH key pair** (if not already done):  
On the **Master node** (control machine):  
bash  
Copy code  
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id\_rsa

* + Press Enter to leave the passphrase empty (optional but recommended for automation).

**Copy the public key to the nodes** (node1 and node2):  
Use the ssh-copy-id command to copy the SSH public key to each node. Replace node1\_ip and node2\_ip with the IP addresses of the nodes:  
bash  
Copy code  
ssh-copy-id user@node1\_ip

ssh-copy-id user@node2\_ip

* + This will allow you to log in to the nodes without a password, which is essential for Ansible to communicate with them.

**Test SSH connectivity**:  
After copying the key, test that you can SSH into the nodes without entering a password:  
bash  
Copy code  
ssh user@node1\_ip

ssh user@node2\_ip

1. If this works without prompting for a password, SSH is set up correctly.

### **Step 3: Configure the Inventory File (hosts.yaml)**

1. Create a new file called hosts.yaml (or hosts.ini if you prefer the INI format) to define your **inventory** of nodes.

**Create hosts.yaml** (in YAML format):  
yaml  
Copy code  
all:

children:

master:

hosts:

master-node-ip:

nodes:

hosts:

node1-ip:

node2-ip:

1. Replace master-node-ip, node1-ip, and node2-ip with the actual IP addresses of your instances. This YAML file organizes the hosts into groups (master and nodes).  
   You can also use Ansible’s built-in groups like webservers, dbservers, or any other grouping mechanism.

### **Step 4: Create the Ansible Playbook to Install Oracle JDK 11**

Now that communication between the master and nodes is set up, let's create an **Ansible Playbook** to install **Oracle JDK 11** on the nodes.

**Create the playbook file**: Create a file named install-jdk.yml.  
yaml  
Copy code  
---

- name: Install Oracle JDK 11 on all nodes

hosts: nodes

become: yes # Use sudo to install the package

tasks:

- name: Add Oracle JDK 11 repository (Ubuntu/Debian)

apt\_repository:

repo: 'ppa:openjdk-r/ppa'

state: present

when: ansible\_os\_family == "Debian"

- name: Add Oracle JDK 11 repository (CentOS/RHEL)

yum\_repository:

name: 'oraclejdk11'

description: 'Oracle JDK 11'

baseurl: 'https://yum.oracle.com/repo/OracleLinux/OL7/latest/x86\_64/ol7/latest/RPMS/'

enabled: yes

gpgcheck: no

when: ansible\_os\_family == "RedHat"

- name: Install Oracle JDK 11

package:

name: openjdk-11-jdk

state: present

- name: Verify JDK installation

command: java -version

register: java\_version

- name: Print JDK version

debug:

msg: "Installed JDK version: {{ java\_version.stdout }}"

### **Explanation of the Playbook:**

* **hosts**: This tells Ansible to apply the playbook to the nodes group defined in your hosts.yaml.
* **become**: This makes sure Ansible uses sudo to perform the installation tasks.
* **Tasks**:
  + The playbook first adds the appropriate JDK repository based on the OS type (Ubuntu/Debian or CentOS/RHEL).
  + Then it installs openjdk-11-jdk on the nodes.
  + Finally, it runs a command to check and verify the installed Java version.

### **Step 5: Run the Playbook**

**Run the playbook** using the following command:  
bash  
Copy code  
ansible-playbook -i hosts.yaml install-jdk.yml

1. This command tells Ansible to use the **hosts.yaml** file as the inventory and execute the playbook **install-jdk.yml**.
2. Ansible will connect to node1 and node2 and perform the tasks in the playbook:
   * Add the JDK repository (if required).
   * Install OpenJDK 11.
   * Print the installed JDK version.

### **Step 6: Verify the Installation**

Once the playbook completes, verify that **Oracle JDK 11** has been installed on the nodes:

SSH into node1 and node2 and run:  
bash  
Copy code  
java -version

1. This should display the Oracle JDK 11 version, confirming the installation.

### **Conclusion**

You have now set up Ansible in a master-slave configuration, where:

* The **Master node** (control machine) communicates with **Node 1** and **Node 2**.
* You have created an **inventory file** to manage your nodes.
* You’ve written and run an **Ansible Playbook** to install **Oracle JDK 11** on the nodes.

This setup can be extended to automate many more tasks across multiple machines with minimal manual intervention.

# What is the ansible arch and how does it work ?

### **Ansible Architecture and How It Works**

Ansible is a powerful **open-source automation tool** used for configuration management, application deployment, and task automation. It is agentless, meaning it doesn’t require any software to be installed on the managed nodes, and it communicates over standard protocols like **SSH** (or **WinRM** for Windows) to execute tasks remotely.

Here is a detailed overview of **Ansible's architecture** and how it works:

### **Key Components of Ansible Architecture**

1. **Ansible Control Node (Master Node)**:
   * **Role**: The control node (often called the "master" node) is where Ansible is installed and run from. It contains the playbooks, inventory files, and configuration files. It is responsible for executing the Ansible automation tasks.
   * **Requirements**: The control node does not need to have agents or daemons running on it. It only needs **Python** (on Linux systems) and **Ansible** installed.
   * **Example**: A laptop, VM, or server where you manage and control your infrastructure.
2. **Managed Nodes (Target Nodes)**:
   * **Role**: Managed nodes (also called "target nodes") are the systems you want to configure, deploy software to, or perform tasks on. These nodes are typically remote machines that Ansible communicates with.
   * **Communication**: Ansible communicates with these nodes over **SSH** (on Linux/Unix systems) or **WinRM**(on Windows systems). Ansible does not require any agent or daemon running on the managed nodes.
   * **Example**: Web servers, database servers, application servers, etc.
3. **Ansible Modules**:
   * **Role**: Ansible modules are the building blocks of automation. They are scripts that execute specific tasks like managing files, packages, services, users, and more.
   * **Examples**:
     + apt (for installing packages on Debian-based systems).
     + yum (for installing packages on RedHat-based systems).
     + service (for managing services like starting or stopping services).
     + copy (for copying files to the target nodes).
     + user (for managing user accounts).
   * Modules are executed remotely on the target nodes and do not need to be installed on the target nodes beforehand (Ansible pulls the required modules automatically).
4. **Inventory**:
   * **Role**: The **inventory** file contains a list of managed nodes (or groups of nodes) that Ansible will target. It defines the machines and how to connect to them.
   * **Format**: The inventory can be in several formats, including:
     + **INI format**: A simple text file format.
     + **YAML format**: A more readable and flexible format.

**Example**:  
ini  
Copy code  
[webservers]

web1.example.com

web2.example.com

[dbservers]

db1.example.com

db2.example.com

1. **Playbooks**:
   * **Role**: **Playbooks** are the main way to define a series of tasks to be executed on managed nodes. They are written in **YAML** format and describe the tasks Ansible should perform.
   * **Playbooks** are structured as a list of **plays**. Each play is associated with a group of managed nodes and specifies the tasks to be executed on those nodes.

**Example**:  
yaml  
Copy code  
- name: Install Apache web server

hosts: webservers

become: yes

tasks:

- name: Install httpd package

yum:

name: httpd

state: present

- name: Start the httpd service

service:

name: httpd

state: started

1. **Ansible Configuration File**:
   * **Role**: The **Ansible configuration file** (ansible.cfg) allows you to set global options and parameters for how Ansible runs.
   * You can configure settings like the **inventory file**, **SSH connection settings**, **default module options**, and more.

Example ansible.cfg:  
ini  
Copy code  
[defaults]

inventory = ./hosts

remote\_user = ubuntu

### **How Ansible Works:**

Ansible works through a series of steps that execute tasks defined in playbooks, which are executed on the **Control Node**. Here's a high-level flow of how Ansible executes a task:

1. **Define the Inventory**:
   * Ansible first needs to know which systems it will manage. You define this in an **inventory** file.
   * You can specify groups of hosts (e.g., web servers, database servers), IP addresses, or even use dynamic inventories from cloud providers like AWS or GCP.
2. **Write a Playbook**:
   * A **playbook** is written, which includes a series of tasks you want Ansible to perform on the managed nodes.
   * Each task usually uses a module (e.g., yum, apt, copy) to interact with the target system.
3. **Run the Playbook**:
   * Once you have your **inventory file** and **playbook**, you can run the playbook from the **Control Node** with the ansible-playbook command.

Example:  
bash  
Copy code  
ansible-playbook -i inventory.ini playbook.yml

1. **Ansible Connects to Managed Nodes**:
   * Ansible connects to the target nodes **over SSH** (for Linux/Unix systems) or **WinRM** (for Windows systems).
   * The **Ansible Control Node** uses the **SSH key** (or password if required) to authenticate to the nodes.
   * **No agent is needed on the managed nodes**. Ansible uses **Python** (on Linux/Unix systems) on the target node to run the tasks, so it is important that Python is available on the managed nodes.
2. **Execute Tasks Remotely**:
   * Ansible runs the tasks defined in the playbook on the target nodes using the appropriate **Ansible modules**.
   * These modules are executed **remotely** on the managed nodes and do not need to be pre-installed on them. Ansible will automatically download and execute the necessary modules for each task.
   * Example task (e.g., installing Apache web server):
     + The **apt** or **yum** module is used to install software.
     + The **service** module is used to start/stop a service (e.g., Apache).
3. **Ansible Reports the Results**:
   * After executing the tasks on the managed nodes, Ansible provides a summary of the execution, including which tasks succeeded or failed, the number of changes made, etc.
   * Ansible provides this information in a human-readable format and can also generate JSON or other formats if needed.

### **Key Features of Ansible:**

1. **Agentless**:
   * No agent needs to be installed on the managed nodes. Ansible uses **SSH** (or **WinRM** for Windows) to manage nodes, which simplifies the setup process and reduces maintenance overhead.
2. **Declarative**:
   * Ansible playbooks are declarative, meaning you specify **what** you want the final state to be, and Ansible figures out **how** to get there. For example, in a playbook, you might define that you want a package to be installed, but Ansible handles the actual installation process.
3. **Idempotent**:
   * Ansible is **idempotent**, meaning running the same playbook multiple times will not cause unintended changes or side effects. If the desired state is already achieved, Ansible will make no changes.
4. **Extensible**:
   * Ansible supports the creation of custom modules, plugins, and roles to extend its functionality. This allows users to tailor automation processes to their specific needs.
5. **Human-readable Syntax**:
   * Ansible uses **YAML** for writing playbooks, which is simple and easy to understand. YAML is structured and readable, making it easier for teams to collaborate on automation tasks.
6. **Cross-Platform**:
   * Ansible can manage both **Linux/Unix** and **Windows** nodes, making it a versatile tool for managing a mixed infrastructure.

### **Ansible Workflow Example:**

Let’s look at a simplified example of how Ansible works:

**Inventory File** (hosts.ini):  
ini  
Copy code  
[webservers]

web1.example.com

web2.example.com

**Playbook** (install\_apache.yml):  
yaml  
Copy code  
---

- name: Install Apache on webservers

hosts: webservers

become: yes

tasks:

- name: Install Apache package

yum:

name: httpd

state: present

- name: Start Apache service

service:

name: httpd

state: started

**Run Playbook**:  
bash  
Copy code  
ansible-playbook -i hosts.ini install\_apache.yml

1. **Execution**:
   * Ansible connects to each web1.example.com and web2.example.com via SSH.
   * It installs Apache (httpd) and starts the service on both web servers.
2. **Result**:
   * Ansible outputs a summary, showing success or failure for each task.

### **Conclusion:**

Ansible’s architecture is designed to be **simple**, **flexible**, and **powerful**. It operates without requiring agents on managed nodes and uses SSH for communication. By defining tasks in playbooks and organizing systems in inventory files, Ansible automates IT processes such as configuration management, application deployment, and system maintenance, while remaining easy to use and scalable.