**Refer URL Link:**

**YouTube URL:** <https://www.youtube.com/watch?v=O4i5Wh5JvyM&list=PLdpzxOOAlwvLxd5nmtmORCmhD5jkrNbuE&index=1>

**GitHub URL:**

<https://github.com/iam-veeramalla/ansible-zero-to-hero>

https://docs.ansible.com/

Idea URL :

https://medium.com/@msmounica1192/getting-started-with-ansible-98009aa75409

**DAY:1**

Introduction to Ansible and Getting Started

* Overview of Ansible: What is Ansible, its advantages, and why use it?
* Comparison with Shell and Python scripting for automation.
* Installing Ansible on different platforms.
* IDE(VS Code) and Plugin configuration.

**What is ansible?**

🡺 Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code.

🡺 Control Node as your node where **ansible install**.

🡺 Master Node as your **target Node.**

**What is the difference between Ansible and Puppet/ Chef?**

**Ansible:**

1. Ansible is agentless

2. Ansible uses to Communicates with remote systems via SSH or APIs.

3. Uses YAML-based playbooks for configuration.

**Puppet &Chef:**

1. while Puppet and Chef require agents installed on

managed nodes

2. whereas Puppet and Chef use their own agents.3.Both Utilizes a domain-specific language (DSL) for configuration.

**Advantages:**

➢ Ansible is free to use by everyone.

➢ Ansible is very consistent and light weight and no constrains regarding the OS or

underlying hardware are present.

➢ It is very secure due to its agentless capabilities and open SSH security features.

➢ Ansible doesn’t need any special system administrator skills to install and use it.

➢ It is push mechanism.

**Disadvantages**:

➢ Insufficient user interface, though ansible tower is GUI, but it is still in development stage.

➢ Cannot achieve full automation by ansible.

➢ New to the market, therefore limited support and document is available.

**Terms used in Ansible:**

* **Ansible Server:** the machine where ansible is installed and from which all tasks and Playbooks will be run.
* **Module:** basically, a module is a command or set of similar commands meant to be executed on the client side.
* **Task:** a task is section that consist of a single procedure to be completed.
* **Role:** a way of organizing tasks and related files to be later called playbook.
* **Fact:** information fetched from the client form the global variables with the gather facts operation.
* **Inventory:** file containing data about the ansible client servers.
* **Play:** execution of playbook.
* **Handler:** task which is called only if notifier is present.
* **Notifier**: section attributed to a task which calls a handler if the output is changed.
* **Playbooks:** it consists code in YAML format which describes tasks to be executed.
* **Host:** nodes which are automated by ansible.

**How do you install Ansible?**

sudo apt-get install ansible # For Ubuntu/Debian

sudo yum install ansible # For CentOS/RHEL

**Install:**

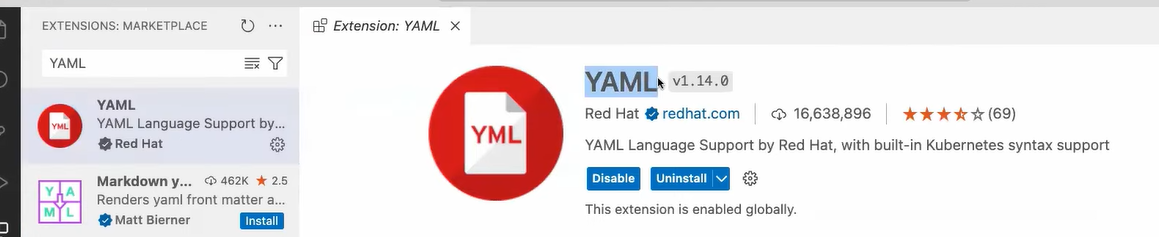
A screenshot of a computer

Description automatically generated

🡺 **Pip install ansible**

**🡺 ansible --version**

**VS-code external Install or setup:**



A red circle with a white letter in it

Description automatically generated

**Different b/w Shell Scripting and ansible**

**Shell Scripting:**

1. **Language:** Written in scripting languages like Bash, sh (Bourne Shell), or other Unix shells.
2. **Imperative:** Defines a sequence of commands to be executed one after another.
3. **Direct Commands:** Executes commands directly on the target system, often using system commands and utilities.
4. **Limited Abstraction:** Offers limited built-in abstractions for managing configurations and infrastructure.
5. **Portability:** May be less portable between different operating systems due to variations in shell syntax and commands.

**Ansible:**

1. **Language:** Uses YAML (YAML Ain't Markup Language) syntax for defining playbooks.
2. **Declarative:** Describes the desired state of the system, focusing on what should be done rather than how it should be done.
3. **Agentless:** Communicates with remote systems over SSH or APIs without installing agents, reducing overhead and simplifying management.
4. **Rich Abstraction:** Provides rich abstractions for configuration management and automation, including modules, roles, and playbooks.
5. **Portability:** Highly portable, as playbooks are platform-independent and can manage various operating systems, making it suitable for managing heterogeneous environments.

=======

**Day 2:** **Ansible Adhoc Commands**

* Password less Authentication
* Ansible Inventory
* Understanding Adhoc commands and their usage.
* Examples of common Adhoc commands for system management tasks.
* Exploring the power of Adhoc commands for quick tasks.

**How to setup Passwordless Authentication:**

**Using Public Key:**

🡺 ssh-copy-id -f "-o IdentityFile <PATH TO PEM FILE>" ubuntu@<INSTANCE-PUBLIC-IP>

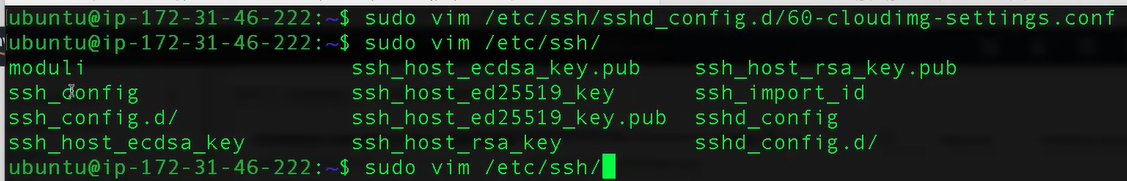
* **ssh-copy-id:** This is the command used to copy your public key to a remote machine.
* **-f:** This flag forces the copying of keys, which can be useful if you have keys already set up and want to overwrite them.
* **"-o IdentityFile** ": This option specifies the identity file (private key) to use for the connection. The -o flag passes this option to the underlying ssh command.
* **ubuntu@:** This is the username (ubuntu) and the IP address of the remote server you want to access.

**Using Password:** with out using SSH-Key

🡺Go to the file **/etc/ssh/sshd\_config.d/60-cloudimg-settings.conf**

🡺Update **PasswordAuthentication** yes

🡺Restart SSH -> **sudo systemctl restart ssh**



🡺To connect using password:

- $ **sudo password ubuntu**

- New password : xxxxxxxx

- Retype New password: xxxxxx

A computer diagram with blue rectangles and blue rectangles

Description automatically generated

**What is Ansible Inventory?**

🡺 Ansible inventory is a crucial component in Ansible, **helping you manage the hosts you want to automate tasks on**.

🡺 It defines a collection of hosts, which can be grouped, organized, and assigned variables.

🡺 The default location for inventory is a file called **/etc/ansible/hosts**. You can also specify a different inventory file at the command line using the -i <path> option. You can pull the inventory file from dynamic or cloud sources or different formats (YAML, ini). Ansible has inventory plugins to make it flexible and customize.

**🡺 ansible-playbook -i inventory <Adhoc command or Playbook.yml>**

**There are two main types of Ansible inventory:**

**1. Static Host Inventory:** Manually created text files with a list of hosts, groups, and variables. Suitable for a set of servers that don’t change frequently.

🡺 A static inventory file is typically a plain text file (usually named hosts or inventory) and is structured in **INI or YAML format.** Here are examples of both formats:



2. **Dynamic Host Inventory**: A dynamic inventory is generated by a script or plugin and can be used for environments where hosts are constantly changing (e.g., cloud environments). The script or plugin fetches the list of hosts from a source like AWS, GCP, or any other dynamic source.

A screen shot of a computer code

Description automatically generated

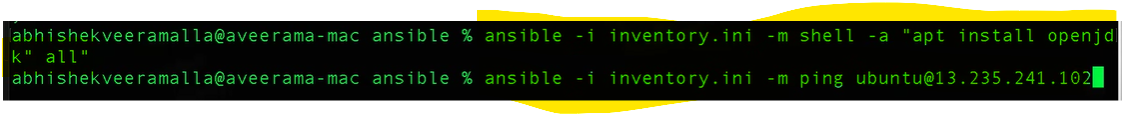
**What is Adhoc command?**

🡺 The command in Ansible is a one-time command that you run directly from the command line without the need to create a playbook.

🡺 It allows you to perform quick tasks or execute modules on remote servers. Ad hoc commands are particularly useful for tasks that don't require complex orchestration or automation.

🡺 **ansible <Inventory\_file> -m <Module> -a <arguments>**



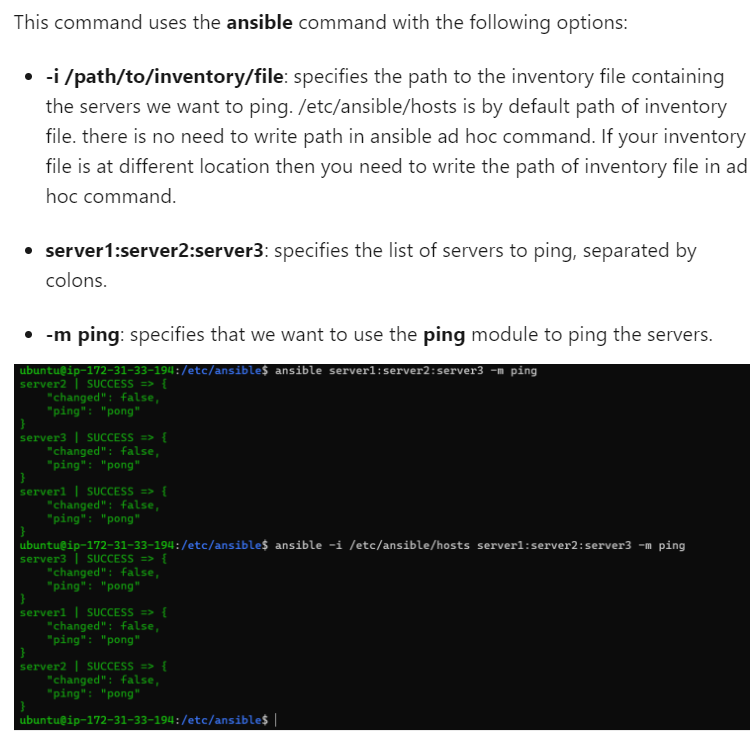


* **ansible:** The command to run Ansible.
* <inventory>: Specifies the hosts or groups of hosts on which to execute the ad hoc command.
* -m <module>: Indicates the module to be executed on the remote hosts. Modules provide specific functionalities such as managing files, installing packages, or running commands.
* -a "<arguments>": Provides arguments or options to the module being executed. The arguments vary depending on the module used.

**Tasks**

1.Write an ansible ad hoc ping command to ping 3 servers from inventory file

🡺 **ansible -i /path/to/inventory/file server1:server2:server3 -m ping**



2. Write an ansible ad hoc command to check uptime

**🡺 ansible -i /path/to/inventory/file all -m command -a uptime**

**A screen shot of a computer

Description automatically generated**

3.ansible ad hoc command to check the free memory or memory usage of hosts

**🡺 ansible -i /path/to/inventory/file all -a "free -m"**

**4**.ansible ad hoc command to get physical memory allocated to the host

**🡺 ansible all -m shell -a "cat /proc/meminfo|head -2"**

**5.**To check the disk space on all hosts in an inventory file

**🡺 ansible -i inventory\_file all -m shell -a 'df -h'**

6.To check the status of a specific service on all hosts in an inventory file

🡺**ansible -i inventory\_file all -m service -a 'name=apache2 state=started'**

**7**.Create a Directory with 755 permission using ansible ad hoc command

🡺 **ansible all -m file -a "path=/home/ubuntu/ansible state=directory mode=0755" -b**

**8.** Create a file with 755 permissions using ansible ad hoc commands

**🡺 ansible all -m file -a "path=/path/to/file state=touch mode=0755"**

**what is difference between ad hoc command and ansible playbook in ansible?**

**Ad hoc Commands:**

* One-liners executed directly from the command line.
* Ideal for quick tasks or one-off operations.
* Simple and don't require separate files.
* Suited for tasks like checking server status, running simple commands, or performing quick configuration changes.
* Not easily repeatable or maintainable.
* Executed using the ansible command with specific options and modules.

**Ansible Playbooks:**

* YAML files containing organized tasks.
* Designed for automating complex tasks and configurations.
* Support conditionals, loops, and variable usage.
* Used for tasks that need to be repeated, scaled, or maintained over time.
* Provide flexibility and control over the automation process.
* Easily repeatable, scalable, and maintainable.
* Can be version-controlled, shared, and reused across different environments.
* Executed using the ansible-playbook command with the playbook file as an argument.

**Day 3:** Writing Your First Ansible Playbook

* Understanding YAML basics and Ansible playbook structure.
* Introduction to Ansible structure: Playbook, Play, Modules, Tasks and Collections.
* Hands-on: Writing a playbook to install apache2 and deploy a static app on aws.

**What is YAML?**

🡺 YAML (YAML Ain't Markup Language) is a human-readable data serialization format that is commonly used for configuration files and data exchange between languages with different data structures.

A screenshot of a phone

Description automatically generated

🡺A dictionary is represented in a simple key: value form (the colon must be followed by a space):

A screenshot of a computer

Description automatically generated

EX:

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Become: True** - When we put this in play that play will run as a **Root user**

**What is gather Facts?**

🡪 It basically it trying to connect to **that ec2 instance are not and get all the required information from it**.

**What is Ansible Playbook?**

🡺 A Playbook is a YAML file that defines a series of actions to be executed on managed nodes. It contains one or more "plays" that map groups of hosts to roles.

🡺 Playbooks are divided into many sectors like.

a. **Target section:** defines the host against which playbooks task has to be executed.

b. **Variable**: define variables.

c. **Task section:** list of modules that we need to run in an order.

**Syntex:**

---

- **name:** <servers\_name>

**hosts:** <servers>

**remote**: <root>

**tasks:**

- **name:** <Module>



A screenshot of a computer program

Description automatically generated

**What is play?**

🡺 A Play is a single, complete execution unit within a playbook. It specifies which hosts to target and what tasks to execute on those hosts.

🡺 Plays are used to group related tasks and execute them in a specific order.

A screenshot of a computer

Description automatically generated

**What is Module?**

**🡺** Modules are the building blocks of Ansible tasks.

🡺They are small programs that perform a **specific action on a managed node, such as installing a package, copying a file, or managing services.**

**A screenshot of a computer

Description automatically generated**

**What is Tasks?**

**🡺** Tasks are individual actions within a play that use modules to perform operations on managed nodes. Each task is executed in order and can include conditionals, loops, and handlers.

**A screenshot of a computer program

Description automatically generated**

**What is Collections?**

**🡺** Collections are a distribution format for Ansible content. 🡺They bundle together multiple roles, modules, plugins, and other Ansible artifacts. Collections make it easier to share and reuse Ansible content.

**A computer code with text

Description automatically generated with medium confidence**

**A white text with black text

Description automatically generated**

What are exactly built-in Modules & Custom modules in ansible?

1. **Built-in Modules**:

* Pre-packaged modules that come with Ansible.
* Cover a wide range of tasks such as system administration, file manipulation, package management, cloud provisioning, and network management.
* Examples include copy, file, yum, apt, shell, command, and ping.
* Ready to use without any additional setup.
* Widely used and documented.

2. **Custom Modules**:

* Created or obtained from external sources to extend Ansible's functionality.
* Written in various programming languages like Python, Perl, Ruby, or shell scripts.
* Useful for tasks not covered by built-in modules or for integrating with specific applications or systems.
* Can be stored in a directory structure and referenced in playbooks.
* Installed in the appropriate module path for easy access.
* Follow a similar structure to built-in modules and are invoked in playbooks using the module directive.

**Day 4: Understanding Ansible Roles**

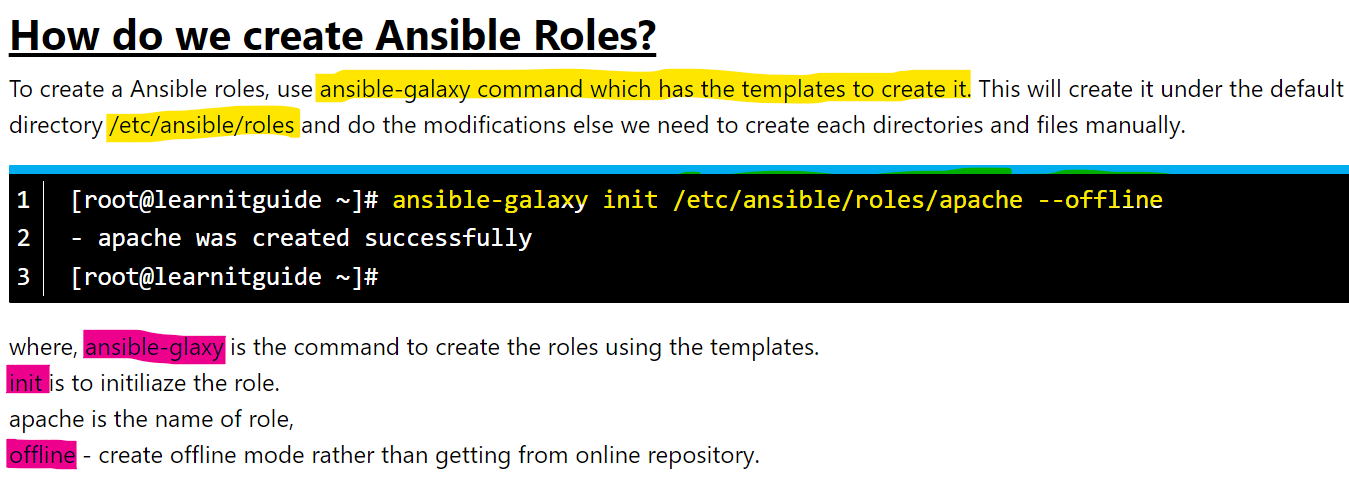
* + - What are Ansible roles and why use them?
    - Exploring the folder structure of Ansible roles.
    - Comparing roles with playbooks and understanding their advantages.
    - Hands-on: Creating a simple role and using it in a playbook.

**What is ansible role?**

🡺 An Ansible role is a reusable, self-contained unit of automation that is used to organize and manage tasks, variables, files, templates, and handlers in a structured way.

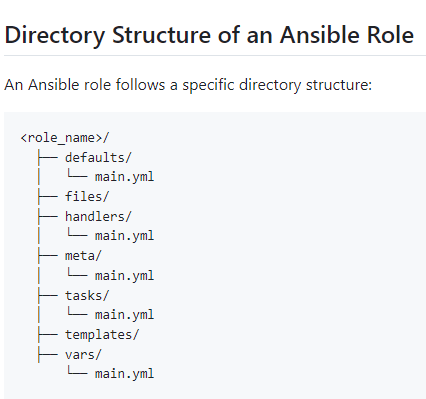
🡺Roles help to encapsulate and modularize the logic and configuration needed to manage a particular system or application component.

🡺 This modular approach promotes reusability, maintainability, and consistency across different playbooks and environments.



🡺 ansible-galaxy role init <Name \_of\_Role\_Creating>

Ex: ansible -galaxy role init test



A list of tasks with text

Description automatically generated with medium confidence

**tasks** - It contains the main list of tasks that the role must complete.  
**handlers** - Tasks that are triggered by changes in other tasks, typically used for actions like restarting services.  
**defaults** - Default variables for the role, which can be overridden.  
**vars** - Variables that are used within the role.  
**files** - Static files that need to be transferred to managed hosts.

**templates** - a collection of templates that can be deployed using this role.  
**meta** — Metadata about the role, including dependencies on other roles. (author, dependency, versions, examples, and so on).

**Library** - Custom modules or plugins used within the role.

**Module\_defaults -** Default module parameters for the role.

**Lookup\_plugins** - Custom lookup plugins for the role.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

🡺 In Vi code terminal is there any extra space we need to remove that

1**. Ese Ctrl +V** you can remove all of them (Click – D or Delete button).

**Why Use Ansible Roles?**

1. **Modularity**: Roles allow you to break down complex playbooks into smaller, reusable components. Each role handles a specific part of the configuration or setup.

2. **Reusability**: Once created, roles can be reused across different playbooks and projects. This saves time and effort in writing redundant code.

3.**Maintainability**: By organizing related tasks into roles, it becomes easier to manage and maintain the code. Changes can be made in one place and applied consistently wherever the role is used.

4.**Readability**: Roles make playbooks cleaner and easier to read by abstracting away the details into logically named roles.

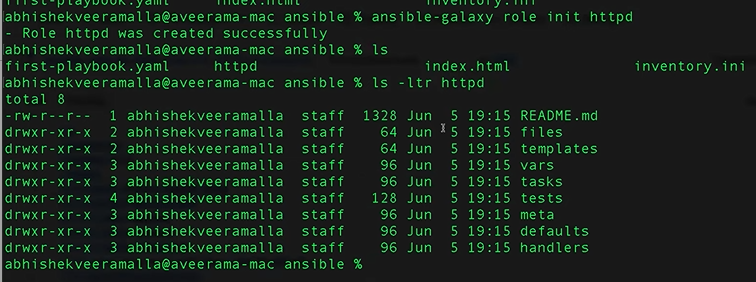
5.**Collaboration:** Roles facilitate collaboration among team members by allowing them to work on different parts of the infrastructure independently.

6. **Consistency:** Using roles ensures that the same setup and configuration procedures are applied uniformly across multiple environments, reducing the risk of configuration drift.

**Convert ansible playbook into Role what will do..?**

**🡺First: we need to do called ansible galaxy command and assigned**

**🡺 ansible-galaxy role init httpd**

****

**🡺 first-palybook.yml task text we need to remove there and add inside the https/task/main.yml file.**

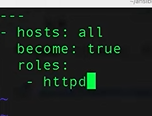


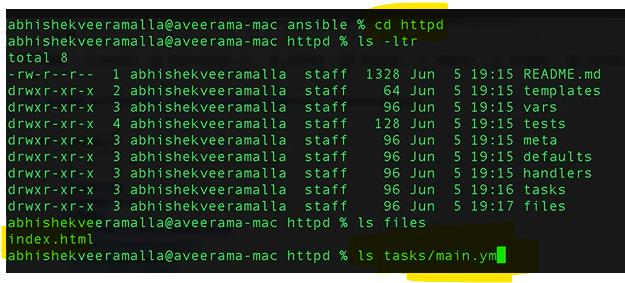
**A computer screen shot of a computer program

Description automatically generated**



🡺 Inside first-playbook.yml we need add some text are content .





**Day 5:** Deep Dive into Ansible Roles with Demo

* Ansible Galaxy - Exploring pre-built Ansible roles.
* Ansible Galaxy - Importing and Installing roles.
* DEMO: Advanced usage of Ansible roles with a practical example project.
* Best practices for organizing roles and playbook structure.

**What is ansible Galaxy?**

🡺Ansible Galaxy is a **repository for Ansible roles**, offering a centralized platform to find, share, and manage automation components. It promotes role reusability, community collaboration, version control, and ensures role quality through ratings and feedback, making it an essential tool for efficient Ansible automation deployment.

**Different Between Ansible and Ansible Galaxy?**

**Ansible**: It is a powerful automation engine that simplifies tasks like configuration management, application deployment, and infrastructure orchestration through declarative playbooks written in YAML.

**Ansible Galaxy**: This platform serves as a centralized hub where Ansible roles are shared and accessed. Roles encapsulate reusable automation logic, making it easier to leverage community-contributed solutions for common automation tasks. Galaxy provides versioning, quality ratings, and facilitates collaboration among Ansible users worldwide.

A screenshot of a computer

Description automatically generated

**Import the Role to Ansible Galaxy**

🡺 **ansible-galaxy role import <your\_github\_username> <role-name>**

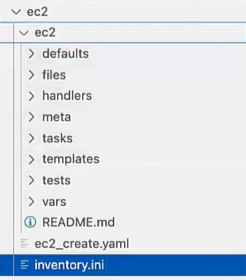
**Ex:**

**-** **<role-name>:** Replace this with the name of the role repository you want to import.

Day 6: **Ansible Variables and Precedence**

* Create AWS Resources using Ansible (Collections)
* Understanding Ansible variables and their scope with an example
* Jinja2 Templating - Utilizing advanced templating features.
* Variable precedence: How Ansible resolves conflicts between different variable sources.
* Hands-on: Using variables in playbooks and roles.

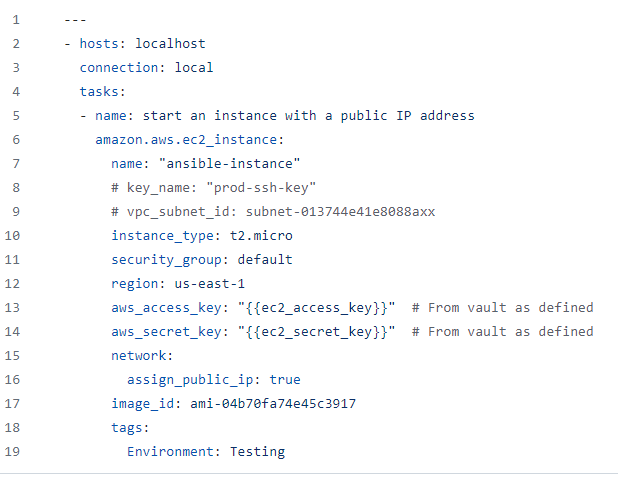
🡺 **ansible-galaxy role init ec2**



**What is collections?**

🡺 Collections are a distribution format for Ansible content. You can package and distribute playbooks, roles, modules, and plugins using collections.

🡺 They allow you to organize your Ansible code more effectively and share it with others. Collections provide a structured and standardized format for bundling related automation assets.



**Setup EC2 Collection and Authentication**

**🡺 Install boto3**

**- pip install boto3**

**🡺 Install AWS Collection**

**-** **ansible-galaxy collection install amazon.aws**

**🡺 Set vault:**

1. Create a password for vault.
2. Add your AWS credentials using the below vault command.

****

**What is ansible\_vault ?**

* **Ansible Vault** is a tool that encrypts sensitive data in Ansible playbooks and roles.
* It uses AES256 encryption to secure passwords, API keys, and other confidential information.
* Encrypted files can be edited, viewed, and used in playbooks with the ansible-vault command-line tool.
* Integration with version control systems allows secure sharing and collaboration among team members.
* Usage: Encrypt files with **ansible-vault encrypt** **filename.yml** and run playbooks with **ansible-playbook --ask-vault-pass playbook.yml** to prompt for the vault password.

**Commands:**

****

**Idea:**

**aws\_access\_key: "{{ec2\_access\_key}}" # From vault as defined**

**aws\_secret\_key: "{{ec2\_secret\_key}}" # From vault as defined**

**Day 7: Ansible Conditionals and Loops**

1. Using conditionals in Ansible to control task execution.
2. Implementing loops for repetitive tasks.
3. Practical examples of conditionals and loops in playbooks.

**What are Ansible loops?**

🡺 Loops in Ansible are sets of instructions that automate repeated tasks, making it easier to perform the same action multiple times without manual repetition.

🡺They work similarly to other basic programming looping concepts such as for\_each or while. Ansible loops can be used, for example, for installing multiple packages, creating numerous users, or modifying a set of files.

A screenshot of a computer

Description automatically generated

**Task-1:**



# **Task -2:** **Passwordless Authentication**

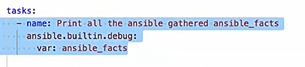
**🡺Using Public Key**

**ssh-copy-id -f "-o IdentityFile <PATH TO PEM FILE>" ubuntu@<INSTANCE-PUBLIC-IP>**

Ex: **ssh-copy-id -f “-o IdentityFile ~/download/abcd-keypair.pem” ‘ec2-user@12.0.1.198’**

**Task-3: shutdown only Ubuntu Instances**

**Hint:** Use **when** condition on ansible **gather\_facts**



**A screenshot of a computer code

Description automatically generated**

**Day 8: Error Handling in Ansible**

* Dealing with errors and failures in Ansible playbooks.
* Error handling techniques and best practices.
* Demonstrating error handling in practical scenarios.