**Ansible Assignment 4**

1. Describe the Ansible register.

* register statement is used to capture the output of a command. The output is saved into a temporary variable that can be used later in the playbook for either debugging purposes or to achieve something else, such as a particular configuration based on a command's output.
* This is useful when we have different outputs from each remote host. The register value is valid throughout the playbook execution so we can make use of set\_fact to manipulate the data and provide input to other tasks accordingly.
* Below is the eg. to capture the output of a command for debugging purposes:

|  |
| --- |
| ---  - name: Installs a package and prints the result  hosts: all  tasks:  - name: Install the package  yum:  name: httpd  state: installed  register: install\_result  - debug:  var: install\_result |

* When you run the playbook, the debug module is used to dump the value of the install\_result registered variable to the terminal
* Below is the eg. to search all .txt files in the remote host home folder and then capturing it in find\_txt\_files and displaying that variable.

|  |
| --- |
| - hosts: all tasks:  name: find all txt files in /home shell: "find /home -name \*.txt" register: find\_txt\_files  - debug:  var: find\_txt\_files |

1. In Ansible, how can we delegate tasks?

* Task delegation is basically used when we would want to perform a task on one host with reference to other hosts. We can do this using the delegate\_to keyword.
* In a play, you can delegate a task to run on a different host instead of the current managed host.
* A task delegates the action to a host using the delegate\_to directive.
* This directive points Ansible to the host that will execute the task in place of the corresponding target.
* One of the most common places you might delegate a task is on localhost, the Ansible control node.
* For eg you might do this if you need to talk to an API for a service that can not be reached from the managed host for some reason, but can from the control node.
* The following simple example runs the **uname -a** command on each host in the play, and then runs the **uname -a** command on localhost on behalf of each host in the play

|  |
| --- |
| ---  - hosts: servera.lab.example.com  tasks:  - name: Get system information  command: uname -a  register: server    - name: Display servera system information  debug:  msg: {{ server }}  - name: Get system information  command: uname -a  delegate\_to: localhost  register: local  - name: Display localhost system information  debug:  msg: {{ local }} |

1. What is the best way to install Ansible on a CentOS system?

* Ansible can be installed very easily on centos in just below steps we need to perform.
* Step 1: We need to set EPEL(Extra Packages for Enterprise Linux )Repository
* It is open source and free community based repository project from Fedora team which provides high quality add on software packages for Linux distribution including RHEL,CentOS etc..
* we need to enable EPEL repository by using the below command:

sudo rpm -ivh http://dl.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm

* above command will download all the necessary packages which will be required to install Ansible.
* Step 2: Now we just need to install ansible after adding EPEL repo by using below command

yum install ansible -y

* Step 3 : Check the version by using below command

ansible –version

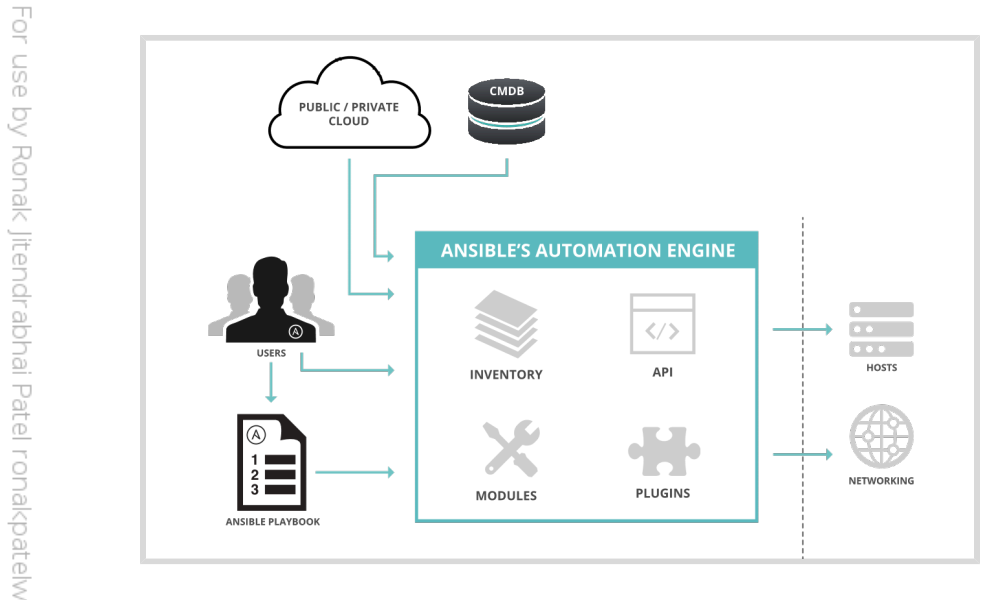
* Now Ansible is installed and we can start using it.

4. What is Ansible and how does it differ from other Configuration Management software?

* Ansible is an open source automation platform. It is used for configuration management, application deployment task automation It is a simple automation language that can perfectly describe an IT application infrastructure in Ansible Playbooks. It is also an automation engine that runs Ansible Playbooks.
* Because of below features and characteristics offered by Ansible it differs from other Configuration management software
* Ansible can manage powerful automation tasks and can adapt to many different workflows and environments. At the same time, new users of Ansible can very quickly use it to become productive
* Ansible Is Simple
* Ansible Is Powerful
* In ansible there two types of servers – Controlling machines and Nodes. Controlling machine is where Ansible is installed and nodes are the ones that are managed by the controlling machines through SSH. There is an inventory file in the controlling machine that holds the location of the node systems. Ansible deploys modules on the node systems by running the playbook on the controlling machine. Ansible is agentless, that means there is no need to have a third party tool to make a connection between one node and the other.
* Ansible is a combination of multiple pieces working together to become an automation tool. Mainly these are modules, playbooks, and plugins.
* Modules are small codes that will get executed. There are multiple inbuilt modules that serve as a starting point for building tasks
* **Agentless**: No agents/software or additional firewall ports that you need to install on the client systems or hosts which you want to automate.
* **Powerful and Flexible**: Ansible’s capabilities allow you to orchestrate the entire application environment regardless of where it is deployed.
* **Efficient**: Ansible introduces modules as basic building blocks for your software. So, you can even customize it as per your needs.

1. What are the various parts of ansible? Describe the architecture of Ansible.

* Below is the basic diagram for architecture of Ansible let us see in depth .



* There are two types of machines in the Ansible architecture
* **control** **nodes** and **managed** **hosts**.
* Ansible is installed and run from a control node and this machine also has copies of your Ansible project files.
* A control node could be an administrator's laptop, a system shared by a number of administrators, or a server running Red Hat Ansible Tower.
* Managed hosts are listed in an inventory, which also organizes those systems into groups for easier collective management. The inventory can be defined in a static text file, or dynamically determined by scripts that get information from external sources.
* The main component of Ansible is the Ansible automation engine.
* This engine interacts with various cloud services, CMDB (Configuration Management Database) and different users who write various playbooks to execute the Ansible Automation engine.
* Below are the few of components
* Ansible APIs work just like any other API and can be used to connect to various public and private Cloud services.
* Modules manage the resources of the system, packages, libraries, files, etc. Ansible modules can automate a wide variety of tasks.
* Plugins: Plugins can be used to execute Ansible tasks as a job. They simplify the execution of a task by building a job like an environment that basically contains pieces of code corresponding to some specific functionality. eg Action plugin, which acts as front ends to modules and can execute tasks on the controller before calling the modules themselves.
* Networking: Ansible can also be used to automate different networks and services. It can do this by creating a playbook or an Ansible role that easily spans different network hardware.
* Hosts: The Ansible Hosts/ Node systems are machines (Linux, Windows, etc) that are getting automated.
* Playbooks: Playbooks are simple code files which describe the tasks that need to be executed. The Playbooks are written in YAML format. They can be used to automate tasks, declare configurations, etc.
* CMDB: It is a database that acts as a storehouse for various IT installations. It holds data about various IT assets (also known as configuration items (CI)) and describes the relationships between such assets.
* Cloud: It is a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server.

1. What are the requirements for Ansible Server?

* For your control node (the machine that runs Ansible), you can use any machine with Python 3.8 or newer installed.
* This includes Red Hat, Debian, CentOS, macOS, any of the BSDs, and so on.
* Windows is not supported for the control node, even though you can install Virtual machine if you are using windows and on top of the that VM you can install any linux and run ansible from thereDescribe a handful of Ansible's basic terms and concepts.
* let us also see for Ansible Tower requirements:
* Supported Operating Systems:
* RHEL 6 and 7, Centos7, Ubuntiu14 LTS 64 bit etc
* The latest stable release of Ansible
* 2 GB RAM minimum (4+ GB RAM recommended)
* 2 GB RAM (minimum and recommended for Vagrant trial installations)
* 4 GB RAM is recommended per 100 forks
* 20 GB hard disk
* 64-bit support required (kernel and runtime)

1. Describe the Infrastructure as Code idea (IaC).

* Infrastructure as Code is a process to manage the operating data servers, storage, system configurations and network infrastructure.
* it is a process that DevOps teams should follow to have a more organized way of managing the infra.
* In olden configuration maanagement system to make any changes even a minute change required manual action by system administrators and the IT support team.
* As technology advance and now it IaC all the configuration details are managed and stored in a standardized file system, wherein the system automatically manages infrastructure changes and deals with system configurations.
* So now with IaC we do not require most of the manual effort since everything is managed and automated by following the IaC approach.
* Tools such as Ansible can be used to implement IaC approach.
* This improves speed, consistency, and accountability.
* A good automation system allows you to implement Infrastructure as Code practices.
* **Infrastructure as Code means that you can use a machine-readable automation language to define and describe the state you want your IT infrastructure to be in.**
* Ideally, this automation language should also be very easy for humans to read, because then you can easily understand what the state is and make changes to it.
* This code is then applied to your infrastructure to ensure that it is actually in that state

1. What do you mean by ad-hoc commands? Give a specific example.

* Basically Ad-hoc commands are just like one-line playbooks to perform a specific task only.
* it is an alternative to writing playbooks.
* Even we can say Ansible ad hoc command uses the /usr/bin/ansible command-line tool to automate a single task on one or more managed nodes. ad hoc commands are quick and easy, but they are not reusable.
* The syntax is ansible [pattern] -m [module] -a "[module options]"
* For example, we need to reboot all servers in the staging group
* You can use an ad hoc task to call the command module and reboot all web servers in lets say GUJARAT, 10 at a time then use below command

ansible GUJARAT -a "/sbin/reboot" -f 10 -u username --become [--ask-become-pass]

* in most of the OS rebooting probably requires privilege access so that can be achieved by **keyword : become**

1. In Ansible, what are the variables?

* Variables in Ansible are just like we assign a value which we can use in playbooks or yaml files.
* It's like variable what we have in other programming languages in same way we have variables to which we can assign some value and we can even manipulate with those variables or apply some conditions.
* Ansible uses variables to manage differences between systems.
* With Ansible you can execute tasks and playbooks on multiple different systems with a single command. To represent the variations among those different systems, you can create variables with standard YAML syntax, including lists and dictionaries
* Let see an below eg. we have defined a variable with name http\_port 80

|  |
| --- |
| - hosts: webservers  vars:  http\_port: 80 |

* After variables are declared administrators can use the variables in tasks.
* Variables arereferenced by placing the variable name in double curly braces ({{ }}).
* Ansible substitutes the variable with its value when the task is executed.
* Below is the sample eg.

|  |
| --- |
| vars:  user: ronak\_poc  tasks:  # This line will read: Creates the user ronak\_poc  - name: Creates the user {{ user }}  user:  # This line will create the user named Joe  name: "{{ user }} |

11. What is the difference between a variable name and a variable that is part of the environment?

* These are two different things let us see below differences

|  |  |
| --- | --- |
| Variable Name | Env Variable |
| User can define variable name in playbook | Env variable are something which is setup at env. level outside program you can say like globally. |
| Variable names are created by adding strings | To access environment variables, you need existing variables. |
| Variables assigned are limited to within that code or play/playbook | Env variable are not limited to program they are set by user/SA admin globally for that system |
| You can easily create multiple variable names by adding strings | To create environment variables we must refer advanced Ansible playbook |
| We use the ipv4 address for variable names | We use {{ ansible\_env.SOME\_VARIABLE }} for remote environment variables. |
|  |  |

12. What are the Ansible Modules, and what do they do? Describe its various types.

* The Ansible modules function as a small set of programs that achieve a specific task.
* Modules can automate a wide range of tasks.
* Ansible ships with hundreds of useful modules that can perform a wide variety of automation tasks.
* They can act on system files, install software, or make API calls.
* When used in a task, a module generally ensures that some particular aspect of the machine is in a particular state. For example, a task using one module might ensure that a file exists and has particular permissions and contents, while a task using a different module might make certain that a particular file system is mounted. If the system is not in that state, the task should put it in that state. If the system is already in that state, it does nothing. If a task fails, the default Ansible

behaviour is to abort the rest of the playbook for the hosts that had a failure.

* **Tasks are idempotent**. This means that you can safely run a playbook on the same hosts multiple times. When your systems are in the correct state, the

playbook makes no changes when you run it. This means that you should be able to run a playbook on the same hosts multiple times safely

* The return value of these are JSON string in stdout and input depends on the type of module.
* These are used by Ansible playbooks. Ansible has hundreds of built-in modules but you can also create custom ones. There are 2 types of modules in Ansible:
* **Core Modules**
* The core Ansible team is responsible for maintaining these modules thus these come with Ansible itself.
* In core module the issues reported are fixed on priority than those in the extras repo.
* The source of these modules is hosted by Ansible on GitHub in Ansible-modules-core.
* **Extras Modules**
* The Ansible community maintains these modules.
* these are being shipped with Ansible but chances are there they might get discontinued in the future. These can be used but if there are any feature requests or issues they will be updated on low priority.
* The source for these modules is hosted by Ansible on GitHub in Ansible-modules-extras.
* Now popular extra modules might enter into the core modules anytime.
* You may find these separate repos for these modules as ansible-modules-core and ansible-modules-extra respectively.

13. What is an Ansible Task, exactly?

* In ansible task is the smallest unit of action you can automate using an Ansible playbook.
* Playbooks in ansible is in yml or yaml format and it will have a single or multiple task in series which user can run to achieve something like say to set up a web server, or to deploy an application to remote environments.
* Ansible will run those tasks in the same order as defined inside a playbook.
* Ansible Tasks allow you to break up bits of configuration policy into smaller files.
* These are blocks of code that can be used to automate any process.
* we can execute a single task once with an ad hoc command.
* you can have multiple small task and create a playbook. For example, to install a package or update a software
* Below is eg. with multiple task to ensure that several network services are enabled to start at boot:

|  |
| --- |
| tasks:  - name: web server is enabled  service:  name: httpd  enabled: true  - name: NTP server is enabled  service:  name: chronyd  enabled: true  - name: Postfix is enabled  service:  name: postfix  enabled: true |