**Ansible – Websphere Setup.**

Using Ansible, we would like to accomplish the following role development around the websphere.

1. Websphere Installation
   1. Should cover Installation on Linux 7 & Linux 8
   2. Should cover Installation on Websphere 8 & Websphere 9
2. Websphere DMGR and associated Nodes setup
3. Websphere Configuration
4. Websphere Deployment
5. Websphere Backup & Restore
6. Websphere Reporting -> Report on current config setup in Websphere
7. Websphere Upgrade
   1. Minor Upgrade -> Fix Pack Upgrade
   2. Major Upgrade
8. Websphere Maintenance scripts.
   1. Start & Stop of DMGR and Nodes
   2. Start & Stop of Cluster
   3. Rolling Start and Stop of Cluster

**Pre-requisites on user machine is**

1. Proper Disk Space on servers
2. FID and GID, so that configs can be applied through their FID’s.
3. Python 2 in user machines – Incase if it is RHEL7, Python 3 in user machines on RHEL8.
   1. This can be offered on user machines using Lightspeed Catalogs.

**Challenges Known:**

1. WAS Configuration scripts are in Python 2, in RHEL8 Machine only Python 3 is available – So scripts that are being called should be platform agnostic.
2. On every phase and role that is being developed, a sign off from Websphere admin team – would be better, to identify gaps. - At least in Installation and deployment related roles.
3. Independent Tester outside of the team to review, sign off on the changes applied – Any member from onboarding team would be ideal.

**Ansible Playbooks Structure:**

Proposing the following structure, as this would help other teams within lightspeed to use uniform conventions and variables while developing variable roles.

Sticking to the basics of Ansible helps in problem solving and avoids unnecessary complexity around many areas.

Let's refresh basics once and stick on to the basic setup.

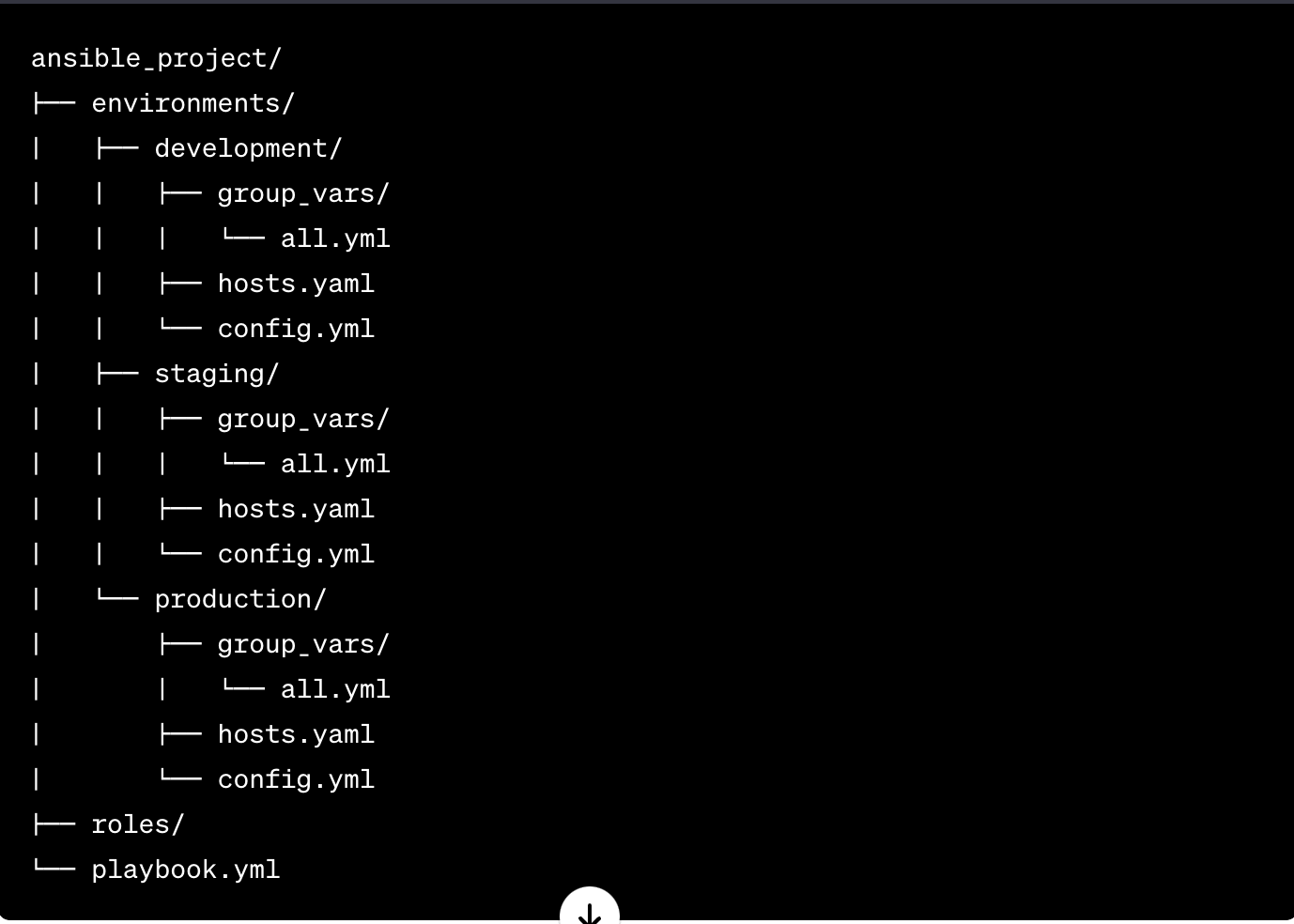
In Ansible, **group\_vars** is a directory that allows you to define variables specific to different groups in your inventory. This directory is part of the standard Ansible directory structure, and it helps you organize and manage variables in a more modular way.

Host Variables Directory **(host\_vars/**): This directory contains YAML files named after individual hosts, and these files include variables that are specific to each host. Variables defined in these files will override any group variables for that specific host.

Okay, now you have variables in group\_vars and host\_vars ? Which one has higher precedence?

***Host\_vars has higher precedence over Group\_vars, it is good to dedicate the use of host\_vars, if some variables are host\_specific only.***

Okay let's look at the proposed user inventory model.



During development, we would need to precisely have a technique to import these variables for better use – since we have group\_vars variables and config.yaml do have variables.

As a standard way, let's import these configs through the role.

Sample role for Debugging pupose:

You can also write it a way

# main.yml

---

- name: Include and Print Environment-Specific Configs

hosts: localhost

gather\_facts: false

tasks:

- name: Include group\_vars

include\_vars:

file: "environments/{{ ansible\_env\_name }}/group\_vars/all.yml"

- name: Include host\_vars

include\_vars:

file: "environments/{{ ansible\_env\_name }}/host\_vars/{{ inventory\_hostname }}.yml"

- name: Include environment config

include\_vars:

file: "environments/{{ ansible\_env\_name }}/config.yml"

- name: Print variables

debug:

var: item

loop:

- "{{ variable\_from\_group\_vars }}"

- "{{ variable\_from\_host\_vars }}"

- "{{ variable\_from\_config }}"

handlers:

- name: Write to log file

copy:

content: |

Variable from group\_vars: {{ variable\_from\_group\_vars }}

Variable from host\_vars: {{ variable\_from\_host\_vars }}

Variable from config: {{ variable\_from\_config }}

dest: "/path/to/your/log/file.log"