

## Lab 06 – Automation with Ansible

**Name:** Anthony Campbell  
**Course/Section:** IS-3033-0N2  
**Date:** 11/19/24

### INTRODUCTION

This lab focuses on the implementation of Ansible, an open-source platform designed to streamline IT automation. Ansible simplifies configuration management and deployment processes through an agentless architecture, using SSH for communication with target nodes. This lab will introduce key elements of Ansible's functionality, such as YAML-based playbooks, the ansible-navigator interface, and execution environments, making how tasks are organized and automated easily understood.

### INTRODUCTION TO THE ENVIRONMENT

The first step, like most when exploring new tools, is to become familiar with the various interface sections. Having done a little coding myself, this code-server is very familiar as it is just Visual Studio Code on a remote server via a web browser. We have our Explorer pane, Code/Project Editor pane, and Terminal Pane. We also have our lab instructions show, though that isn't part of code-server.

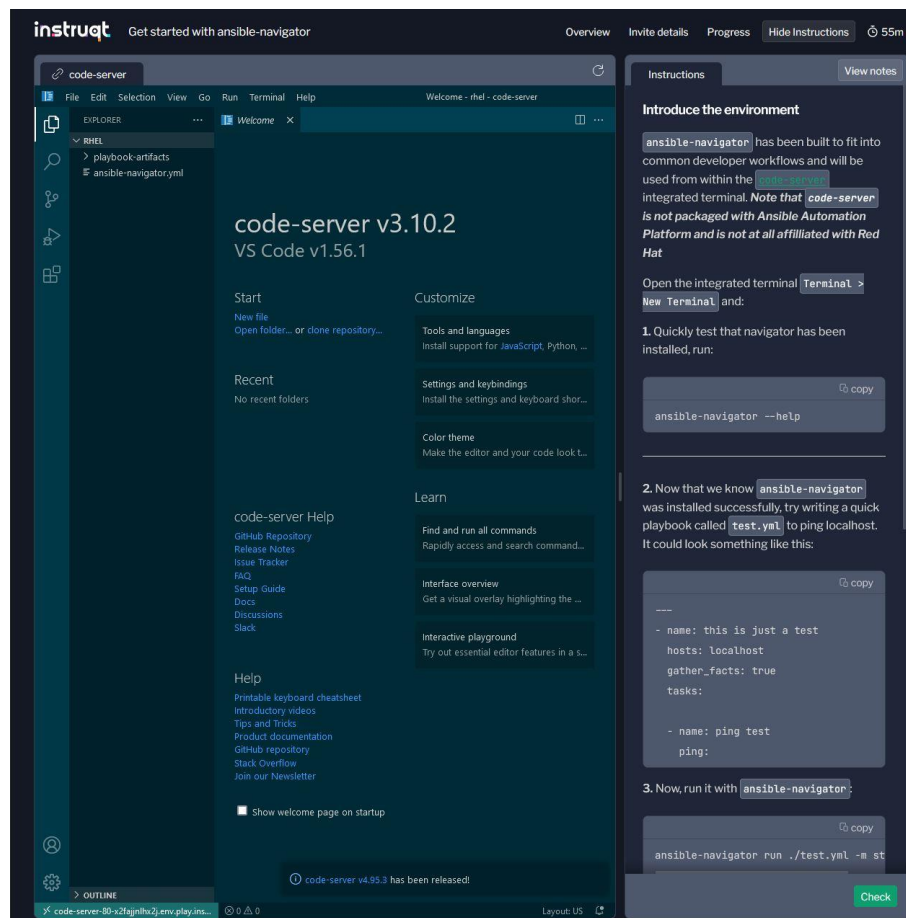


Figure 1: Code-server basics, and our Redhat instruct lab

To become more familiar with ansible, we are instructed to perform a command that is my all time favorite regardless of application, a `-help` command. So, after an `ansible-navigator -help` command and some reading we are well on our way.

Next, we use vim to write out a test playbook, that we name test.yml to ping localhost. A “playbook”, regarding Ansible, is a YAML file used to define several tasks that are executed on a group on managed nodes. It acts like a sort of blueprint for automating various IT processes, such as configuring servers or deploying applications. The playbooks are divided into one or more “plays” and each play targets a specific group of hosts and outlines a set of tasks to be carried out. The tasks use Ansible modules to install software, manage files, or even restart services and the tasks are mostly idempotent, so they only make changes if necessary to keep a consistent result regardless of how many times the playbook is run.

Ansible playbooks also support variables that allow customization and handlers that are triggered when certain tasks notify them. The flexibility makes playbooks more useful for a wide range of purposes, including system configuration, application deployment, and complex orchestration. In using playbooks, organizations can automate infrastructure management and ensure that systems are consistently configured and maintained across various environments. [1]

After writing our test playbook to run a single play, to ping the local host, we run it with Ansible using `ansible-navigator run ./test.yml -m stdout` with the `-mode` standard output keeping our output in the active terminal. We can see that it gives us the name of our play, and our two tasks that are marked complete or “ok” and with that the playbook has been run successfully. The play recap on the last line I like particularly as that would be an invaluable summary.

```
[rhel@code-server ~]$ ansible-navigator run ./test.yml -m stdout
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [this is a test] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [ping test] *****
ok: [localhost]

PLAY RECAP *****
localhost                : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[rhel@code-server ~]$
```

Figure 2: Our first playbook, all ok

## THE ANSIBLE-NAVIGATOR INTERFACE

Next we explore the ansible-navigator interface and learn some of its features. Using *ansible-navigator* command opens the text user interface (TUI) and we can view subcommands. We are once again prompted to use a *help* command and we get so much useful information on some commands and features.

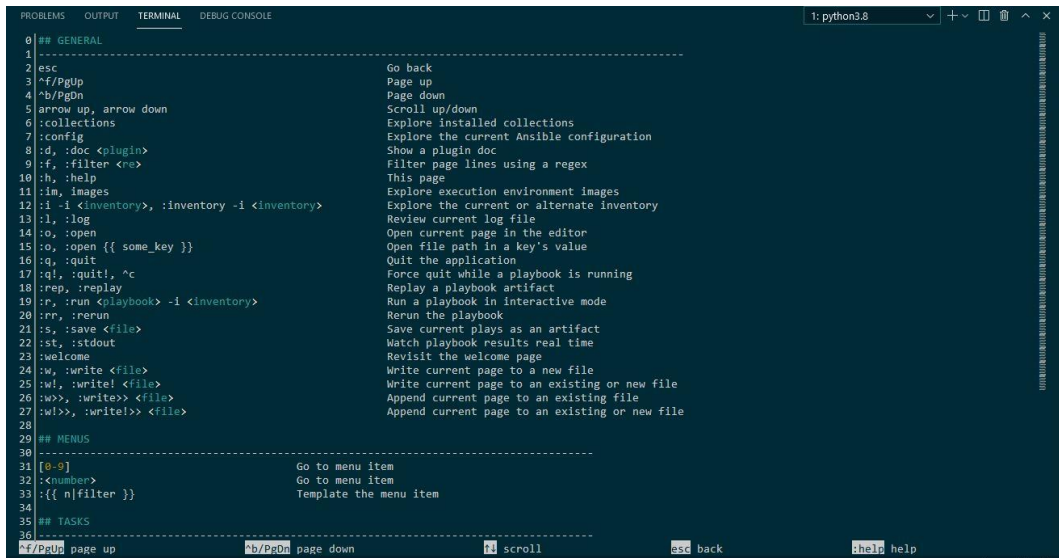
The screenshot shows the Ansible Navigator TUI in a terminal window. The interface has a dark blue background with white text. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'DEBUG CONSOLE'. The 'TERMINAL' tab is active. The main content area displays a list of commands and their descriptions, organized into sections: 'GENERAL', 'MENUS', and 'TASKS'. The 'GENERAL' section lists commands like :esc, :f/PgUp, :b/PgDn, :arrow up, :arrow down, :collections, :config, :d, :doc <plugin>, :f, :filter <re>, :h, :help, :im, :images, :i, :i <inventory>, :l, :log, :o, :open, :o, :open {{ some\_key }}, :q, :quit, :q!, :quit!, ^c, :rep, :replay, :r, :run <playbook>, :rr, :rerun, :s, :save <file>, :st, :stdout, :welcome, :w, :write <file>, :w!, :write! <file>, :w>, :write> <file>, and :w!>, :write!> <file>. The 'MENUS' section lists :[0-9] and :<number>. The 'TASKS' section lists :[n|filter]. At the bottom, there is a status bar with navigation keys: :f/PgUp page up, :b/PgDn page down, :t scroll, :esc back, and :h/help help.

Figure 3: An absolute wealth of information regarding Ansible and what its TUI offers

Next we use the *:doc ping* command to view documentation for the ping module, and are able to view documentation for other modules with the *:doc* command. From there we ran out test playbook again, this time without the standard output mode, to see our tasks and information in regard to them in the ansible-navigator TUI. Using the number pad, I select line 0, to see more details. The TUI is straightforward and not only is it easy to navigate, but easy to understand as well.

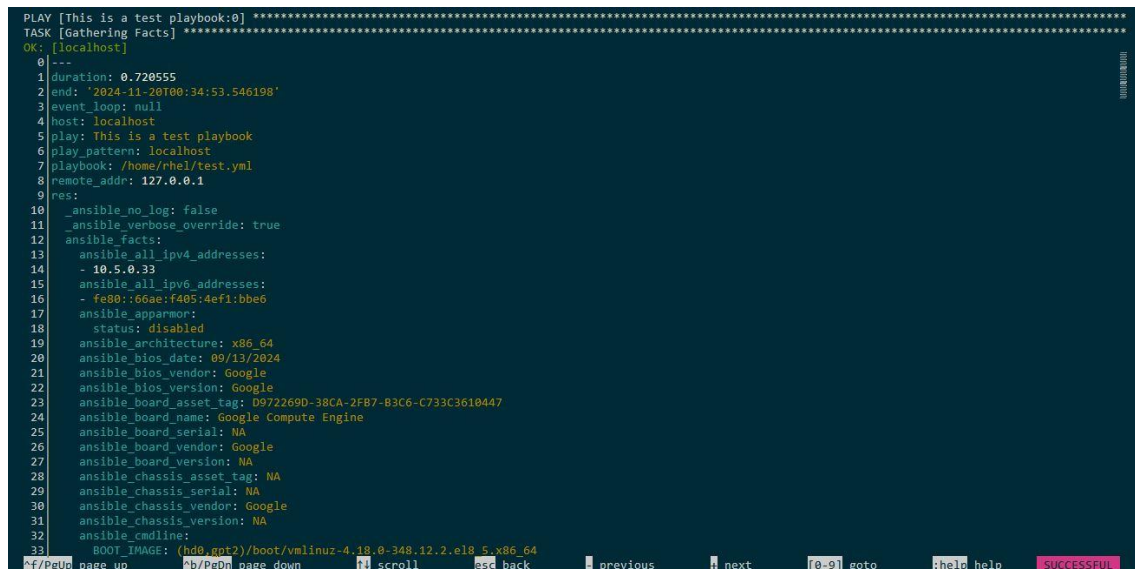
The screenshot shows the Ansible Navigator TUI displaying the details of a task. The top section shows the task name 'TASK [Gathering Facts]' and the host 'OK: [localhost]'. Below this, a list of task details is shown, including duration, end time, event loop, host, play name, play pattern, playbook path, remote address, and various Ansible configuration options like \_ansible\_no\_log, \_ansible\_verbose\_override, ansible\_facts, ansible\_all\_ipv4\_addresses, ansible\_all\_ipv6\_addresses, ansible\_apparmor, ansible\_architecture, ansible\_bios\_date, ansible\_bios\_vendor, ansible\_bios\_version, ansible\_board\_asset\_tag, ansible\_board\_name, ansible\_board\_serial, ansible\_board\_vendor, ansible\_board\_version, ansible\_chassis\_asset\_tag, ansible\_chassis\_serial, ansible\_chassis\_vendor, and ansible\_cmdline. At the bottom, there is a status bar with navigation keys: :f/PgUp page up, :b/PgDn page down, :t scroll, :esc back, :previous, :next, :[0-9] goto, :h/help help, and a 'SUCCESSFUL' status indicator.

Figure 4: Details from the gather\_facts task

Running the `:doc` command, we can view more information about the `gather_facts` module we used for this task, so as long as one is even remotely familiar with ansible they should be able to better understand information being told to them from a task after reading documentation on the module used – extremely helpful. Lastly I ran `ansible-navigator doc setup` from the command line to jump immediately to the documentation for the setup module.

## CONFIGURING COMMON ANSIBLE-NAVIGATOR OPTIONS

Next we use vim to open the `ansible-navigator.yml` file in the `/home/rhel` directory to review its contents, and add some settings of our own to use code-server as the editor.

```
---
ansible-navigator:
  execution-environment:
    container-engine: podman
    image: ee-supported-rhel8
    enabled: false
    pull-policy: never

  playbook-artifact:
    save-as: /home/rhel/playbook-artifacts/{playbook_name}-artifact-{ts_utc}.json

  logging:
    level: debug

  editor:
    command: code-server {filename}
    console: false
```

Figure 5: Preferences and settings adjusted

Now we can run our test playbook again and inspect the play once more, viewing our task output in a new tab within the code editor and making it a little easier to inspect. We are able to configure settings, including custom settings, that we may need specifically for our projects.

## USING EXECUTION ENVIRONMENTS

Next, we configure ansible-navigator to use an execution environment. We open the `ansible-navigator.yml` and enable the execution environment by setting `enabled: true` under the execution-environment settings. Then we rerun our test playbook and can see it pause for a moment as it pulls the execution environment from the container registry. Using `:collections` command we can inspect the execution environment and use our number pad to navigate to `ansible_utils` and locate the author of the `fact_diff` – which is authored by the Ansible Community and a Bradley “cidrblock” Thornton.

```
28 | version: 2.4.2
29 | doc:
30 |   author: Bradley Thornton (@cidrblock)
31 |   description:
32 |     - Parse cli output or text using a variety of parsers
33 |   module: cli_parse
```

Figure 6: Information so easy to find

## CONCLUSION

This lab provided hands-on experience with Ansible and the ansible-navigator, a useful tool in IT automation. We learned how to navigate the interface, create and run playbooks, and explore and read documentation for various modules. We configured the settings in the ansible-navigator.yml file, which customized our environment for easier inspection and task outputs. We also became familiar with execution environments, allowing us to pull from the container registries for improved playbook execution. With these exercises, a better understanding of Ansible infrastructure and management has been found that simplifies the automation process. One can never have too many automation tools under their belt to help manage a network of different devices.

## REFERENCES

R. Mitra, "*Lab 06: Automation with Ansible*," The University of Texas at San Antonio (2024). Last accessed: *November 19, 2024*

- [1] Ansible Community Documentation, "Ansible playbooks," Ansible, 19 November 2019. [Online]. Available: [https://docs.ansible.com/ansible/latest/playbook\\_guide/playbooks\\_intro.html#desired-state-and-idempotency](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html#desired-state-and-idempotency). [Accessed 16 November 2024].

## GENERATIVE AI SEARCHES

No generative AI searches were used during this lab.

## COLLABORATION

I did not collaborate with anyone during this lab.