Generating a Production-Ready Ansible Project Using Python CLI and Claude Prompt

1. Objective

The goal is to create a **Python CLI tool** that can bootstrap a **new Ansible project** with production-level best practices. The tool will be interactive, allowing users to:

- Name the project
- Choose optional features (Git init, install dependencies, Hello World playbook/role)
- Check and install dependencies (Python, Ansible, ansible-lint)
- Dynamically generate project files and folders
- Display a summary after creation

This approach allows flexibility in designing the Ansible project structure while maintaining best practices.

---

2. Claude Prompt Overview

The **Claude prompt** for generating this Python CLI tool is as follows:

```
You are an expert DevOps and Ansible automation engineer.

Create a Python CLI script that initializes a new Ansible project with
production-level best practices. Do not enforce a predefined folder structure —
let the user design it.

Requirements:
1. Prompt user for project name
2. Ask if Git should be initialized
3. Ask if Ansible dependencies should be installed via pip
4. Ask if a sample Hello World playbook/role should be created
5. Check for dependencies:
   - Python3 or Python
   - Ansible
   - ansible-lint
   - If missing, offer installation
6. Dynamically create project files/folders based on user choices
7. Implement modular Python CLI using argparse/inquirer, pathlib, subprocess,
colorama
8. Provide clear summary after creation
9. Include example interactive console session
```

3. Step-by-Step Guide to Generate the Project

Step 1: Prepare Your Environment

- Make sure Python 3 is installed.
- Optional: Install `pip` if not available.

```
python3 --version
python3 -m pip --version
```

- Install Python packages needed for the CLI:

```
pip install inquirer colorama
```

Step 2: Use Claude Prompt to Generate Python CLI Script

1. Open Claude or any compatible AI platform.
2. Paste the **Claude prompt** (from Section 2) into the input field.
3. Ask Claude to **generate the Python CLI script** (`bootstrap_ansible.py`).

Step 3: Save and Prepare the Script

1. Save the output from Claude as `bootstrap_ansible.py` .
2. Make it executable (optional):

```
chmod +x bootstrap_ansible.py
```

Step 4: Run the Python CLI Tool

1. Execute the CLI:

```
python3 bootstrap_ansible.py
```

1. Follow the interactive prompts:
2. Enter project name
3. Choose whether to initialize Git
4. Choose whether to install Ansible dependencies

5. Choose whether to create a Hello World playbook/role

6. The CLI will check for dependencies:

7. Python 3 (`python3 --version` fallback `python --version`)
8. Ansible (`ansible --version`)
9. ansible-lint (`ansible-lint --version`)

If missing, the CLI will prompt to install them automatically.

Step 5: Review Generated Project

After the CLI finishes, the project will contain:

- Root folder named as your project
- README.md
- Optional Git repository initialized
- Optional Hello World playbook and role
- Optional `ansible.cfg` and configuration files
- Clear success summary with next steps

Step 6: Test the Project

- Navigate to the project folder:

```
cd <project-name>
```

- Run the sample playbook (if created):

```
ansible-playbook playbooks/hello.yml -i inventories/dev/hosts.yml
```

- Run linter checks (if included):

```
ansible-lint playbooks/
```

---

7. Example Interactive Session

```
$ python3 bootstrap_ansible.py
Enter project name: my_ansible_project
Initialize Git repository? (Y/n) Y
Install Ansible dependencies? (Y/n) Y
Create sample Hello World playbook and role? (Y/n) Y
```

```
Checking dependencies...
Python 3 found: 3.11.2
Ansible not found. Install now? (Y/n) Y
Installing Ansible via pip...

Creating project structure...
Initializing Git repository...
Creating Hello World playbook and role...

✅ Ansible project "my_ansible_project" created successfully!
Next steps:
  cd my_ansible_project
  Run or extend your playbooks and roles as needed
```

---

8. Notes and Best Practices

- Keep `group_vars/` and `host_vars/` for environment-specific configuration
- Use `roles/` to encapsulate reusable logic
- Use `Makefile` or scripts for automation convenience
- Document the project with a `README.md` detailing usage and playbooks