md-to-pdf

Quick Guide: Generate Ansible Project with Claude Code in VS Code

1. Setup VS Code & Claude Code

```
# Install VS Code (if needed)
# Download from: https://code.visualstudio.com/

# Open VS Code
code .
```

Install Claude Code Plugin: - Press `Ctrl+Shift+X` (Extensions) - Search for “Continue” or “Claude Code” - Click Install - Restart VS Code

2. Open Claude Code & Use the Prompt

Open Claude Code: - Press `Ctrl+L` (Windows/Linux) or `Cmd+L` (Mac) - Or use Command Palette (`Ctrl+Shift+P`) → Type “Continue”

Paste this prompt:

```
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. When executed, prompt the user:
   - `Enter the project name:`
     (This will be used as the root folder name and in README)

2. Prompt the user if they want to include optional features:
   - Initialize Git repository (Yes/No)
   - Install Ansible dependencies via pip (Yes/No)
```

   - Create sample Hello World playbook and role (Yes/No)

3. Check for required dependencies:
   - Python 3 (`python3 --version` fallback to `python --version`)
   - Ansible (`ansible --version`)
   - ansible-lint (`ansible-lint --version`)
   - If any are missing, interactively ask the user if they want to install:
     - Python missing → open the official installer page
     - Ansible missing → install via `pip install --user ansible ansible-lint`

4. Dynamically create project files and folders based on user input:
   - Root folder named as project name
   - README.md with project name
   - Optional Git initialization
   - Optional Hello World playbook and role
   - Config files (`ansible.cfg`) if requested

5. Implement a **clean, modular Python CLI script**:
   - Use `argparse` or `inquirer`-style CLI prompts
   - Use `os` and `pathlib` to create folders/files
   - Use `subprocess` for dependency checks and installation
   - Use `colorama` for colored console messages (optional)
   - Gracefully handle errors and missing tools
   - Print a clear summary at the end:
     ```
       Ansible project "<project-name>" created successfully!
     Next steps:
       cd <project-name>
       Run or extend your playbooks and roles as needed
     ```

6. Include example **interactive console session** showing project creation, dependency checks, and Hello World playbook/role execution if created.

### Deliverable

- Output a **complete Python CLI script** implementing all of the

~~~
above
- The script should be **ready to save and run** in any terminal
to bootstrap a new Ansible project
- Do not enforce a specific folder structure; allow user
flexibility


## 3. Save & Run the Generated Script

```bash
# Save the generated code as
create_ansible_project.py

# Make executable and run
chmod +x create_ansible_project.py
python3 create_ansible_project.py
~~~

4. Interactive Session Example

```
Enter project name: my-infrastructure
Initialize Git repository? (Y/n): y
Create sample Hello World playbook and role? (Y/n): y

  Project 'my-infrastructure' created successfully!
```

5. Generated Structure

```
my-infrastructure/
├── ansible.cfg
├── inventory/hosts
├── playbooks/
│   └── hello_world.yml
├── roles/
│   └── hello_world/
│       ├── tasks/main.yml
│       ├── defaults/main.yml
│       └── meta/main.yml
└── README.md
```

6. Test Hello World

```bash
# Enter the project
cd my-infrastructure
```

```
# Run Hello World playbook
ansible-playbook -i inventory/hosts playbooks/hello_world.yml

# Output:
PLAY [Hello World Playbook] ***
TASK [Display greeting message] ***
ok: [localhost] => {
    "msg": "Hello from localhost!"
}
TASK [Include hello_world role] ***
ok: [localhost]
  Playbook run successful!
```

That’s it! You now have a working Ansible project with a testable Hello World example.