Complete Guide: Creating Ansible Project Generator with Claude Code in VS Code

Prerequisites Setup

Step 1: Install VS Code and Claude Code Plugin

1. **Install VS Code** (if not already installed):

   ```
    # macOS
   brew install --cask visual-studio-code

   # Ubuntu/Debian
   sudo apt update
   sudo apt install code

   # Or download from: https://code.visualstudio.com/
   ```

2. **Install Claude Code Plugin**:
   - Open VS Code
   - Go to Extensions (Ctrl+Shift+X / Cmd+Shift+X)
   - Search for "Claude Code" or "Continue" (the AI coding assistant)
   - Click Install
   - Configure with your API key if required

Step 2: Create Project Directory

```
 # Create a new directory for our Ansible generator project
mkdir ansible-project-generator
cd ansible-project-generator

# Open in VS Code
code .
```

Using Claude Code to Generate the Script

Step 3: Open Claude Code Interface

1. In VS Code, open the Command Palette (Ctrl+Shift+P / Cmd+Shift+P)
2. Type "Claude" or "Continue" and select the option to open the AI assistant
3. Or use the keyboard shortcut (usually Ctrl+L / Cmd+L)

Step 4: Use the Main Prompt

Copy and paste this exact prompt into Claude Code:

```
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 pre
---
### 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 execut
---
### 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
```

Step 5: Review and Save Generated Code

1. Claude Code will generate a complete Python script
2. Review the generated code
3. Save it as `create_ansible_project.py` in your project directory

The Generated Script

Here's what Claude Code should generate (example output):

```
#!/usr/bin/env python3
"""
Ansible Project Generator
A Python CLI tool to initialize new Ansible projects with best practices.
"""

import os
import sys
import subprocess
import shutil
```

```python
import webbrowser
from pathlib import Path
from typing import Optional, Dict, Any

# Try to import colorama for colored output
try:
    from colorama import init, Fore, Style
    init(autoreset=True)
    HAS_COLOR = True
except ImportError:
    HAS_COLOR = False
    # Fallback color definitions
    class Fore:
        GREEN = RED = YELLOW = BLUE = CYAN = ""
    class Style:
        BRIGHT = RESET_ALL = ""

class AnsibleProjectGenerator:
    """Main class for generating Ansible projects."""

    def __init__(self):
        self.project_name = ""
        self.project_path = None
        self.config = {}

    def print_banner(self):
        """Print a welcome banner."""
        print(f"{Fore.CYAN}{Style.BRIGHT}")
        print("=" * 60)
        print("       ANSIBLE PROJECT GENERATOR")
        print("       Production-Ready Project Scaffolding")
        print("=" * 60)
        print(f"{Style.RESET_ALL}\n")

    def colored_print(self, message: str, color: str = ""):
        """Print colored message if colorama is available."""
        if HAS_COLOR and color:
            print(f"{color}{message}{Style.RESET_ALL}")
        else:
            print(message)

    def prompt_yes_no(self, question: str, default: bool = False) -> bool:
        """Prompt user for yes/no question."""
        default_str = "Y/n" if default else "y/N"
        while True:
            response = input(f"{question} ({default_str}): ").strip().lower()
            if not response:
                return default
            if response in ['y', 'yes']:
                return True
            if response in ['n', 'no']:
                return False
            print("Please answer 'yes' or 'no'")

    def check_command_exists(self, command: str, version_flag: str = "--version") -> bool:
        """Check if a command exists in the system."""
        try:
            subprocess.run([command, version_flag],
                         stdout=subprocess.PIPE,
                         stderr=subprocess.PIPE,
                         check=True)
            return True
        except (subprocess.CalledProcessError, FileNotFoundError):
            return False

    def get_python_command(self) -> Optional[str]:
```

```
    def get_python_command(self) -> Optional[str]:
        """Get the available Python command."""
        for cmd in ['python3', 'python']:
            if self.check_command_exists(cmd):
                return cmd
        return None

    def check_dependencies(self) -> Dict[str, bool]:
        """Check for required dependencies."""
        self.colored_print("\n□ Checking system dependencies...", Fore.BLUE)

        deps = {
            'python': False,
            'ansible': False,
            'ansible-lint': False
        }

        # Check Python
        python_cmd = self.get_python_command()
        if python_cmd:
            deps['python'] = True
            self.colored_print(f"  ✓ Python found ({python_cmd})", Fore.GREEN)
        else:
            self.colored_print("  ✗ Python not found", Fore.RED)

        # Check Ansible
        if self.check_command_exists('ansible'):
            deps['ansible'] = True
            self.colored_print("  ✓ Ansible found", Fore.GREEN)
        else:
            self.colored_print("  ✗ Ansible not found", Fore.YELLOW)

        # Check ansible-lint
        if self.check_command_exists('ansible-lint'):
            deps['ansible-lint'] = True
            self.colored_print("  ✓ ansible-lint found", Fore.GREEN)
        else:
            self.colored_print("  ✗ ansible-lint not found", Fore.YELLOW)

        return deps

    def install_missing_dependencies(self, deps: Dict[str, bool]):
        """Offer to install missing dependencies."""
        if not deps['python']:
            self.colored_print("\n⚠  Python is required but not found!", Fore.RED)
            if self.prompt_yes_no("Open Python download page in browser?", True):
                webbrowser.open("https://www.python.org/downloads/")
                print("Please install Python and run this script again.")
            sys.exit(1)

        missing = []
        if not deps['ansible']:
            missing.append('ansible')
        if not deps['ansible-lint']:
            missing.append('ansible-lint')

        if missing:
            self.colored_print(f"\n⚠  Missing tools: {', '.join(missing)}", Fore.YELLOW)
            if self.prompt_yes_no(f"Install {', '.join(missing)} via pip?", True):
                python_cmd = self.get_python_command()
                cmd = [python_cmd, '-m', 'pip', 'install', '--user'] + missing
                try:
                    self.colored_print(f"Running: {' '.join(cmd)}", Fore.CYAN)
                    subprocess.run(cmd, check=True)
                    self.colored_print("✓ Dependencies installed successfully!", Fore.GREEN)
                except subprocess.CalledProcessError:
```

~~~
                    self.colored_print("✗ Installation failed. Please install manually.", Fore.RED)

    def gather_user_input(self):
        """Gather project configuration from user."""
        # Get project name
        while not self.project_name:
            self.project_name = input("\n⍰ Enter the project name: ").strip()
            if not self.project_name:
                print("Project name cannot be empty!")
            elif not self.project_name.replace('-', '').replace('_', '').isalnum():
                print("Project name should contain only letters, numbers, hyphens, and underscores!")
                self.project_name = ""

        self.project_path = Path.cwd() / self.project_name

        # Check if directory exists
        if self.project_path.exists():
            if not self.prompt_yes_no(f"Directory '{self.project_name}' already exists. Overwrite?"):
                print("Exiting...")
                sys.exit(0)
            shutil.rmtree(self.project_path)

        # Gather optional features
        print("\n⍰ Configure optional features:")
        self.config['git'] = self.prompt_yes_no("  Initialize Git repository?", True)
        self.config['install_deps'] = self.prompt_yes_no("  Install Ansible dependencies via pip?", True)
        self.config['hello_world'] = self.prompt_yes_no("  Create sample Hello World playbook and role?", True)
        self.config['ansible_cfg'] = self.prompt_yes_no("  Create ansible.cfg configuration file?", True)
        self.config['requirements'] = self.prompt_yes_no("  Create requirements.yml for Galaxy dependencies?", True)

    def create_readme(self):
        """Create README.md file."""
        readme_content = f"""# {self.project_name}

An Ansible automation project.

## Overview

This project contains Ansible playbooks, roles, and configurations for infrastructure automation.

## Prerequisites

- Python 3.8+
- Ansible 2.9+
- ansible-lint (optional, for linting)

## Installation

1. Clone this repository:
   ```bash
   git clone <repository-url>
   cd {self.project_name}
~~~

2. Install Ansible (if not already installed):

   ```
   pip install ansible ansible-lint
   ```

3. Install Galaxy requirements (if any):

   ```
   ansible-galaxy install -r requirements.yml
   ```

Usage

Run playbooks using:

```
ansible-playbook -i inventory/hosts playbook.yml
```

Project Structure

```
{self.project_name}/
├── README.md
├── ansible.cfg
├── requirements.yml
├── inventory/
├── group_vars/
├── host_vars/
├── roles/
└── playbooks/
```

Contributing

1. Fork the repository
2. Create a feature branch
3. Commit your changes
4. Push to the branch
5. Create a Pull Request

License

This project is licensed under the MIT License. """

```
     readme_path = self.project_path / "README.md"
    readme_path.write_text(readme_content)
    self.colored_print(f"  ✓ Created README.md", Fore.GREEN)

def create_ansible_cfg(self):
    """Create ansible.cfg configuration file."""
    if not self.config.get('ansible_cfg'):
        return

    ansible_cfg_content = """[defaults]
```

Inventory file location

inventory = inventory/hosts

Roles path

roles_path = roles

Host key checking

host_key_checking = False

Retry files

retry_files_enabled = False

Output formatting

stdout_callback = yaml stderr_callback = yaml

Facts gathering

gathering = smart fact_caching = jsonfile fact_caching_connection = /tmp/ansible_facts fact_caching_timeout = 3600

SSH settings

remote_user = ansible private_key_file = ~/.ssh/id_rsa

Performance tuning

forks = 10 pipelining = True

[privilege_escalation] become = True become_method = sudo become_user = root become_ask_pass = False

[ssh_connection] ssh_args = -o ControlMaster=auto -o ControlPersist=60s control_path = /tmp/ansible-%%h-%%p-%%r """

```
     cfg_path = self.project_path / "ansible.cfg"
    cfg_path.write_text(ansible_cfg_content)
    self.colored_print(f"  ✓ Created ansible.cfg", Fore.GREEN)

def create_requirements(self):
    """Create requirements.yml for Galaxy dependencies."""
    if not self.config.get('requirements'):
        return

    requirements_content = """---
```

Ansible Galaxy requirements

Install with: ansible-galaxy install -r requirements.yml

collections:

- name: community.general version: ">=3.3.0"
- name: ansible.posix version: ">=1.3.0"

roles:

- name: geerlingguy.docker

version: "4.1.0"

- name: geerlingguy.nginx

version: "3.1.0"

"""

```
     req_path = self.project_path / "requirements.yml"
    req_path.write_text(requirements_content)
    self.colored_print(f"  ✓ Created requirements.yml", Fore.GREEN)

def create_hello_world(self):
    """Create sample Hello World playbook and role."""
    if not self.config.get('hello_world'):
        return

    # Create directories
    playbooks_dir = self.project_path / "playbooks"
    playbooks_dir.mkdir(parents=True, exist_ok=True)

    roles_dir = self.project_path / "roles"
    roles_dir.mkdir(parents=True, exist_ok=True)

    # Create Hello World playbook
    playbook_content = """---
```

Hello World Playbook

Run with: ansible-playbook -i inventory/hosts playbooks/hello_world.yml

- name: Hello World Playbook hosts: all gather_facts: yes

  tasks:

  - name: Display greeting message debug: msg: "Hello from {{ ansible_hostname }}!"
  - name: Show system information debug: msg: | OS: {{ ansible_distribution }} {{ ansible_distribution_version }} Kernel: {{ ansible_kernel }} Python: {{ ansible_python_version }}
  - name: Include hello_world role include_role: name: hello_world

"""

```
     playbook_path = playbooks_dir / "hello_world.yml"
    playbook_path.write_text(playbook_content)
    self.colored_print(f"  ✓ Created playbooks/hello_world.yml", Fore.GREEN)

    # Create Hello World role structure
    role_dir = roles_dir / "hello_world"
    role_dirs = [
        "tasks",
        "handlers",
        "templates",
        "files",
        "vars",
        "defaults",
        "meta"
    ]

    for subdir in role_dirs:
        (role_dir / subdir).mkdir(parents=True, exist_ok=True)

    # Role main task
    role_task_content = """---
```

Hello World role main tasks

- name: Create hello world file copy: content: | Hello from Ansible! Generated on: {{ ansible_date_time.iso8601 }} Hostname: {{ ansible_hostname }} dest: /tmp/hello_ansible.txt mode: '0644'
- name: Display role message debug: msg: "Hello World role executed successfully!"

- name: Check if file was created stat: path: /tmp/hello_ansible.txt register: hello_file

- name: Show file status debug: msg: "File created: {{ hello_file.stat.exists }}" """

  ```
   task_path = role_dir / "tasks" / "main.yml"
  task_path.write_text(role_task_content)

  # Role defaults
  defaults_content = """---
  ```

Default variables for hello_world role

hello_message: "Hello from Ansible role!" hello_file_path: "/tmp/hello_ansible.txt" """

```
     defaults_path = role_dir / "defaults" / "main.yml"
    defaults_path.write_text(defaults_content)

    # Role meta
    meta_content = """---
```

galaxy_info: author: Your Name description: Hello World demonstration role license: MIT min_ansible_version: 2.9 platforms: - name: Ubuntu versions: - all - name: Debian versions: - all - name: EL versions: - all dependencies: [] """

```
     meta_path = role_dir / "meta" / "main.yml"
    meta_path.write_text(meta_content)

    self.colored_print(f"  ✓ Created roles/hello_world", Fore.GREEN)

def create_inventory(self):
    """Create basic inventory structure."""
    inventory_dir = self.project_path / "inventory"
    inventory_dir.mkdir(parents=True, exist_ok=True)

    # Create hosts file
    hosts_content = """# Ansible Inventory File
```

Define your hosts and groups here

[local] localhost ansible_connection=local

[development]

dev-server-01 ansible_host=192.168.1.10

dev-server-02 ansible_host=192.168.1.11

[staging]

stage-server-01 ansible_host=192.168.2.10

stage-server-02 ansible_host=192.168.2.11

[production]

prod-server-01 ansible_host=192.168.3.10

prod-server-02 ansible_host=192.168.3.11

[all:vars]

ansible_user=ansible

ansible_ssh_private_key_file=~/.ssh/id_rsa

"""

```
     hosts_path = inventory_dir / "hosts"
    hosts_path.write_text(hosts_content)
    self.colored_print(f"  ✓ Created inventory/hosts", Fore.GREEN)

    # Create group_vars and host_vars directories
    (self.project_path / "group_vars").mkdir(parents=True, exist_ok=True)
    (self.project_path / "host_vars").mkdir(parents=True, exist_ok=True)

    # Create sample group_vars/all.yml
    all_vars_content = """---
```

Global variables for all hosts

Common settings

timezone: UTC ntp_servers:

- 0.pool.ntp.org
- 1.pool.ntp.org

Package management

update_cache: yes """

```
     all_vars_path = self.project_path / "group_vars" / "all.yml"
    all_vars_path.write_text(all_vars_content)
    self.colored_print(f"  ✓ Created group_vars/all.yml", Fore.GREEN)

def init_git_repo(self):
    """Initialize Git repository."""
    if not self.config.get('git'):
        return

    try:
        # Create .gitignore
        gitignore_content = """# Ansible
```

*.retry *.log ansible.log .ansible/

Python

*.pyc **pycache**/ .env venv/ env/ .venv/

IDE

.vscode/ .idea/ *.swp *.swo *~ .DS_Store

Credentials (never commit these!)

*.pem *.key *.crt vault_pass.txt .vault_pass **/vault.yml **/secrets.yml

Terraform (if used alongside)

*.tfstate *.tfstate.* .terraform/ """

```
        gitignore_path = self.project_path / ".gitignore"
        gitignore_path.write_text(gitignore_content)

        # Initialize git
        subprocess.run(['git', 'init'], cwd=self.project_path,
                     stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        subprocess.run(['git', 'add', '.'], cwd=self.project_path,
                     stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        subprocess.run(['git', 'commit', '-m', 'Initial commit'],
                     cwd=self.project_path,
                     stdout=subprocess.PIPE, stderr=subprocess.PIPE)

        self.colored_print(f"  ✓ Initialized Git repository", Fore.GREEN)
    except Exception as e:
        self.colored_print(f"  ⚠ Git initialization failed: {e}", Fore.YELLOW)

def install_dependencies(self):
    """Install Ansible dependencies if requested."""
    if not self.config.get('install_deps'):
        return

    python_cmd = self.get_python_command()
    if not python_cmd:
        return

    try:
        # Create requirements.txt for Python dependencies
        requirements_txt = """ansible>=2.9
```

ansible-lint>=5.0 molecule>=3.0 yamllint>=1.26 jinja2>=2.11 """

```
        req_txt_path = self.project_path / "requirements.txt"
        req_txt_path.write_text(requirements_txt)

        # Install Python dependencies
        cmd = [python_cmd, '-m', 'pip', 'install', '-r', 'requirements.txt']
        self.colored_print(f"  Installing Python dependencies...", Fore.CYAN)
        subprocess.run(cmd, cwd=self.project_path, check=True,
                     stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        self.colored_print(f"  ✓ Python dependencies installed", Fore.GREEN)

        # Install Ansible Galaxy requirements if they exist
        if (self.project_path / "requirements.yml").exists():
            self.colored_print(f"  Installing Ansible Galaxy requirements...", Fore.CYAN)
            subprocess.run(['ansible-galaxy', 'install', '-r', 'requirements.yml'],
                         cwd=self.project_path, check=True,
                         stdout=subprocess.PIPE, stderr=subprocess.PIPE)
            self.colored_print(f"  ✓ Galaxy requirements installed", Fore.GREEN)
    except Exception as e:
        self.colored_print(f"  ⚠ Dependency installation failed: {e}", Fore.YELLOW)

def create_project(self):
    """Create the Ansible project structure."""
    self.colored_print(f"\n� Creating project '{self.project_name}'...", Fore.CYAN)

    # Create project directory
    self.project_path.mkdir(parents=True, exist_ok=True)

    # Create project files and structure
    self.create_readme()
    self.create_ansible_cfg()
    self.create_requirements()
    self.create_inventory()
```

```
        self.create_hello_world()

        # Create additional directories
        additional_dirs = ['filter_plugins', 'library', 'module_utils', 'playbooks']
        for dir_name in additional_dirs:
            (self.project_path / dir_name).mkdir(exist_ok=True)

        # Initialize Git
        self.init_git_repo()

        # Install dependencies
        self.install_dependencies()

    def print_summary(self):
        """Print project creation summary."""
        print(f"\n{Fore.GREEN}{'=' * 60}")
        print(f"{Style.BRIGHT}□ Ansible project '{self.project_name}' created successfully!")
        print(f"{'=' * 60}{Style.RESET_ALL}")

        print(f"\n□ Project location: {self.project_path}")

        print(f"\n□ Project features:")
        if self.config.get('git'):
            print(f"  ✓ Git repository initialized")
        if self.config.get('ansible_cfg'):
            print(f"  ✓ ansible.cfg configuration")
        if self.config.get('requirements'):
            print(f"  ✓ requirements.yml for Galaxy")
        if self.config.get('hello_world'):
            print(f"  ✓ Hello World playbook and role")
        if self.config.get('install_deps'):
            print(f"  ✓ Python dependencies installed")

        print(f"\n□ Next steps:")
        print(f"  1. cd {self.project_name}")
        print(f"  2. Review and customize ansible.cfg")
        print(f"  3. Add your hosts to inventory/hosts")
        if self.config.get('hello_world'):
            print(f"  4. Test with: ansible-playbook -i inventory/hosts playbooks/hello_world.yml")
        else:
            print(f"  4. Create your playbooks in playbooks/")
            print(f"  5. Create your roles with: ansible-galaxy init roles/<role_name>")

        print(f"\n□ Useful commands:")
        print(f"  • Check syntax: ansible-playbook --syntax-check playbooks/*.yml")
        print(f"  • Lint playbooks: ansible-lint playbooks/*.yml")
        print(f"  • List hosts: ansible all -i inventory/hosts --list-hosts")
        print(f"  • Test connection: ansible all -i inventory/hosts -m ping")

        print(f"\n{Fore.CYAN}Happy automating! □{Style.RESET_ALL}\n")

    def run(self):
        """Main execution method."""
        self.print_banner()

        # Check dependencies
        deps = self.check_dependencies()
        self.install_missing_dependencies(deps)

        # Gather user input
        self.gather_user_input()

        # Create project
        self.create_project()

        # Print summary
    self.print_summary()
```

def main(): """Main entry point.""" try: generator = AnsibleProjectGenerator() generator.run() except KeyboardInterrupt: print(f"\n{Fore.YELLOW}⚠ Operation cancelled by user.{Style.RESET_ALL}") sys.exit(1) except Exception as e: print(f"\n{Fore.RED}✗ Error: {e}{Style.RESET_ALL}") sys.exit(1)

if **name** == "**main**": main()

~~~

Step 6: Test the Generated Script

Install Required Python Package (Optional)


```bash
# Install colorama for colored output (optional)
pip install colorama
~~~

### Make the Script Executable

```
 chmod +x create_ansible_project.py
```

Run the Script

```
 ./create_ansible_project.py
# Or
python3 create_ansible_project.py
```

Step 7: Interactive Session Example

Here's what the interactive session looks like:

```
 ============================================================
       ANSIBLE PROJECT GENERATOR
       Production-Ready Project Scaffolding
============================================================

□ Checking system dependencies...
  ✓ Python found (python3)
  ✓ Ansible found
  ✗ ansible-lint not found

⚠  Missing tools: ansible-lint
Install ansible-lint via pip? (Y/n): y
Running: python3 -m pip install --user ansible-lint
✓ Dependencies installed successfully!

□ Enter the project name: my-infrastructure

□ Configure optional features:
  Initialize Git repository? (Y/n): y
  Install Ansible dependencies via pip? (Y/n): y
  Create sample Hello World playbook and role? (Y/n): y
  Create ansible.cfg configuration file? (Y/n): y
  Create requirements.yml for Galaxy dependencies? (Y/n): y

□ Creating project 'my-infrastructure'...
  ✓ Created README.md
  ✓ Created ansible.cfg
  ✓ Created requirements.yml
  ✓ Created inventory/hosts
  ✓ Created group_vars/all.yml
  ✓ Created playbooks/hello_world.yml
  ✓ Created roles/hello_world
  ✓ Initialized Git repository
  Installing Python dependencies...
  ✓ Python dependencies installed
  Installing Ansible Galaxy requirements...
  ✓ Galaxy requirements installed

============================================================
□ Ansible project 'my-infrastructure' created successfully!
============================================================

□ Project location: /home/user/my-infrastructure

□ Project features:
  ✓ Git repository initialized
  ✓ ansible.cfg configuration
  ✓ requirements.yml for Galaxy
  ✓ Hello World playbook and role
  ✓ Python dependencies installed

□ Next steps:
  1. cd my-infrastructure
  2. Review and customize ansible.cfg
  3. Add your hosts to inventory/hosts
  4. Test with: ansible-playbook -i inventory/hosts playbooks/hello_world.yml

□ Useful commands:
  • Check syntax: ansible-playbook --syntax-check playbooks/*.yml
  • Lint playbooks: ansible-lint playbooks/*.yml
  • List hosts: ansible all -i inventory/hosts --list-hosts
  • Test connection: ansible all -i inventory/hosts -m ping

Happy automating! □
```

Step 8: Test the Generated Ansible Project

```
# Navigate to the created project
cd my-infrastructure

# Test the Hello World playbook
ansible-playbook -i inventory/hosts playbooks/hello_world.yml

# Check project structure
tree -L 2
```

Expected output:

```
my-infrastructure/
├── README.md
├── ansible.cfg
├── filter_plugins/
├── group_vars/
│   └── all.yml
├── host_vars/
├── inventory/
│   └── hosts
├── library/
├── module_utils/
├── playbooks/
│   └── hello_world.yml
├── requirements.txt
├── requirements.yml
└── roles/
    └── hello_world/
```

Advanced Tips for Using Claude Code

1. Iterative Refinement

If you want to modify the generated script, you can ask Claude Code:

```
"Add a feature to create a Makefile with common Ansible commands"
```

2. Add More Templates

```
"Add an option to create a Kubernetes deployment playbook template"
```

3. Enhanced Error Handling

```
"Improve error handling and add verbose logging option"
```

4. Docker Support

```
"Add an option to create a Dockerfile for running Ansible in a container"
```

Troubleshooting

Common Issues and Solutions

1. **Claude Code not responding:**
    - Check your API key configuration
    - Restart VS Code
    - Check internet connection

2. **Script permission denied:**

   ```
   chmod +x create_ansible_project.py
   ```

3. **Python not found:**
   - Install Python 3.8+
   - Use virtual environment: `python3 -m venv venv && source venv/bin/activate`

4. **Ansible installation fails:**

   ```
    # Try with sudo
   sudo pip3 install ansible ansible-lint

   # Or use --user flag
   pip3 install --user ansible ansible-lint
   ```

Project Structure Best Practices

The generated project follows Ansible best practices:

```
ansible-project/
├── ansible.cfg             # Ansible configuration
├── requirements.yml        # Galaxy dependencies
├── requirements.txt        # Python dependencies
├── inventory/
│   ├── hosts              # Inventory file
│   ├── group_vars/        # Group variables
│   └── host_vars/         # Host variables
├── roles/                 # Custom roles
│   └── role_name/
│       ├── tasks/
│       ├── handlers/
│       ├── templates/
│       ├── files/
│       ├── vars/
│       ├── defaults/
│       └── meta/
├── playbooks/             # Playbooks
├── filter_plugins/        # Custom filters
├── library/               # Custom modules
└── module_utils/          # Module utilities
```

Conclusion

You now have a complete Ansible project generator created using Claude Code in VS Code. This tool:

1. ▯ Checks for system dependencies
2. ▯ Offers to install missing tools
3. ▯ Creates production-ready project structure
4. ▯ Includes sample playbooks and roles
5. ▯ Sets up Git repository
6. ▯ Provides comprehensive documentation

The script is modular, extensible, and follows Python best practices, making it easy to customize for your specific needs.