# Ansible Automation for Gemini CLI on WSL2

**Current Gemini CLI installation methods have evolved significantly in 2024-2025, now requiring Node.js and NPM rather than Python**. The modern approach uses secure API key management through Ansible Vault, with specialized playbooks for Node.js environments. Google GitHub This comprehensive automation framework addresses control node configuration, managed node preparation, and complete Gemini CLI deployment with security best practices.

## Ansible Control Node Setup on Windows 11

### Prerequisites and Installation

#### Virtual Environment Setup:

```bash
# Create isolated Ansible environment
python3 -m venv ~/ansible-venv
source ~/ansible-venv/bin/activate

# Install Ansible with required dependencies
pip install ansible ansible-lint molecule[docker]
pip install pywinrm  # For Windows managed nodes if needed
```

Thomas Preischl GitHub

#### Ansible Configuration:

bash

```
# Create ~/.ansible.cfg
[defaults]
inventory = ~/.ansible/hosts
host_key_checking = False
callback_whitelist = profile_tasks, timer
stdout_callback = yaml
roles_path = ~/.ansible/roles
collections_path = ~/.ansible/collections
remote_user = ansible
private_key_file = ~/.ssh/id_rsa
vault_password_file = ~/.ansible/vault_pass

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

[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o UserKnownHostsFile=/dev/null
control_path_dir = ~/.ansible/cp
pipelining = True
```

Ansible

## SSH Key Management and Distribution

### Generate and Configure SSH Keys:

bash

```
# Generate SSH key pair for Ansible automation
ssh-keygen -t rsa -b 4096 -C "ansible-automation@wsl2" -f ~/.ssh/ansible_rsa

# Add to SSH agent
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/ansible_rsa

# Create SSH config for streamlined connections
cat > ~/.ssh/config << EOF
Host wsl-nodes
    HostName %h
    User ansible
    IdentityFile ~/.ssh/ansible_rsa
    StrictHostKeyChecking no
    UserKnownHostsFile=/dev/null
EOF
```

Kindatechnical +2

### Ansible Vault Setup for Secure Credential Management:

bash

```
# Create vault password file
echo "your-secure-vault-password" > ~/.ansible/vault_pass
chmod 600 ~/.ansible/vault_pass

# Create encrypted variables file for API keys
ansible-vault create group_vars/all/vault.yml
```

yaml

```
# Contents of group_vars/all/vault.yml (encrypted)
vault_gemini_api_key: "your-google-ai-studio-api-key"
vault_google_project_id: "your-google-cloud-project-id"
vault_npm_registry_token: "optional-npm-registry-token"
```

Digitalocean TechCloudUp

## Collection and Role Management

### Install Required Collections:

```bash
# Install Node.js and system management collections
ansible-galaxy collection install community.general
ansible-galaxy collection install ansible.posix
ansible-galaxy collection install community.crypto
```

Ansible Ansible

**Directory Structure:**

```
~/.ansible/
├── ansible.cfg
├── vault_pass
├── hosts
├── group_vars/
│   └── all/
│       ├── main.yml
│       └── vault.yml
├── roles/
│   └── gemini_cli/
└── playbooks/
    ├── gemini_cli_install.yml
    └── site.yml
```

# Ansible Managed Node Setup on WSL2

## Target Node Requirements and Preparation

### Essential System Requirements:

- **Operating System**: Ubuntu 20.04+ or compatible Linux distribution
- **Python**: 3.8+ (for Ansible modules)
- **Node.js**: Version 18+ (for Gemini CLI) GitHub github
- **Network**: SSH access enabled with key-based authentication
- **Privileges**: Sudo access for system-level installations Spacelift Digitalocean

### User and SSH Configuration:

```bash
# On target WSL2 node - create ansible user
sudo useradd -m -s /bin/bash ansible
sudo usermod -aG sudo ansible

# Configure passwordless sudo
echo "ansible ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/ansible

# Setup SSH directory and authorized keys
sudo -u ansible mkdir -p /home/ansible/.ssh
sudo -u ansible chmod 700 /home/ansible/.ssh

# Copy public key from control node
cat ~/.ssh/ansible_rsa.pub | sudo -u ansible tee /home/ansible/.ssh/authorized_keys
sudo -u ansible chmod 600 /home/ansible/.ssh/authorized_keys
```

GoLinuxCloud

### System Dependencies:

```bash
# Install essential packages for Ansible automation
sudo apt update
sudo apt install -y curl wget git python3 python3-pip build-essential

# Install Node.js 18+ (required for Gemini CLI)
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

# Verify installations
node --version  # Should be 18+
npm --version
python3 --version
```

## Inventory Configuration

### Basic Inventory Setup:

ini

```ini
# ~/.ansible/hosts
[gemini_nodes]
wsl-node1 ansible_host=192.168.1.100
wsl-node2 ansible_host=192.168.1.101

[gemini_nodes:vars]
ansible_user=ansible
ansible_ssh_private_key_file=~/.ssh/ansible_rsa
ansible_python_interpreter=/usr/bin/python3
node_version=18
npm_global_packages_path=/usr/local/lib/node_modules
```

nixCraft Digitalocean

### Advanced Inventory with Groups:

yaml

```yaml
# inventory.yml (YAML format)
all:
  children:
    gemini_nodes:
      hosts:
        wsl-dev:
          ansible_host: 192.168.1.100
          environment: development
        wsl-staging:
          ansible_host: 192.168.1.101
          environment: staging
      vars:
        ansible_python_interpreter: /usr/bin/python3
        node_version: "18"

    gemini_nodes:vars:
      ansible_user: ansible
      ansible_ssh_private_key_file: ~/.ssh/ansible_rsa
      gemini_cli_version: "latest"
      gemini_sandbox_enabled: true
```

## Network and Connectivity Validation

### Connection Testing Playbook:

yaml

```yaml
# playbooks/test_connectivity.yml
---
- name: Test WSL2 node connectivity
  hosts: gemini_nodes
  gather_facts: yes
  tasks:
    - name: Test SSH connectivity
      ping:

    - name: Verify sudo access
      command: sudo -n true
      changed_when: false

    - name: Check Python version
      command: python3 --version
      register: python_version
      changed_when: false

    - name: Display system information
      debug:
        msg: |
          Hostname: {{ ansible_hostname }}
          Python: {{ python_version.stdout }}
          OS: {{ ansible_distribution }} {{ ansible_distribution_version }}
```

# Ansible Configuration Lab for Gemini CLI Installation

## Complete Gemini CLI Installation Role

### Role Structure:

```
roles/gemini_cli/
├── defaults/main.yml
├── tasks/main.yml
├── handlers/main.yml
├── templates/
│   ├── gemini_config.j2
│   └── gemini_env.j2
├── files/
│   └── gemini_wrapper.sh
└── vars/main.yml
```

Ansible Spacelift

**Role Defaults:**

yaml

```yaml
# roles/gemini_cli/defaults/main.yml
---
# Node.js and NPM configuration
nodejs_version: "18"
npm_global_install: true
npm_registry: "https://registry.npmjs.org/"

# Gemini CLI configuration
gemini_cli_package: "@google/gemini-cli"
gemini_cli_version: "latest"
gemini_cli_user: "{{ ansible_user }}"
gemini_cli_install_path: "/usr/local/bin"

# Authentication settings
gemini_auth_method: "api_key"  # api_key, oauth, vertex_ai
gemini_sandbox_enabled: true
gemini_config_dir: "/home/{{ gemini_cli_user }}/.gemini"

# Environment variables
gemini_env_vars:
  GEMINI_SANDBOX: "{{ gemini_sandbox_enabled | bool }}"
  BUILD_SANDBOX: "true"

# Security settings
gemini_api_key: "{{ vault_gemini_api_key }}"
gemini_project_id: "{{ vault_google_project_id | default('') }}"
```

**Main Tasks:**

yaml

```yaml
# roles/gemini_cli/tasks/main.yml
---
- name: Validate Node.js requirements
  block:
    - name: Check Node.js version
      command: node --version
      register: node_version_check
      changed_when: false
      failed_when: false

    - name: Verify Node.js version compatibility
      assert:
        that:
          - node_version_check.rc == 0
          - node_version_check.stdout is version('v18.0.0', '>=')
        fail_msg: "Node.js version 18+ required. Current: {{ node_version_check.stdout | default('not installed') }}"
        success_msg: "Node.js version compatible: {{ node_version_check.stdout }}"

- name: Install system dependencies
  become: yes
  package:
    name:
      - curl
      - git
      - build-essential
    state: present

- name: Configure NPM for global installations
  become: yes
  block:
    - name: Create NPM global directory
      file:
        path: "{{ npm_global_packages_path }}"
        state: directory
        mode: '0755'

    - name: Configure NPM global prefix
      npm:
        name: npm
        global: yes
        state: latest

- name: Install Gemini CLI via NPM
  become: yes
  npm:
    name: "{{ gemini_cli_package }}"
```

```
    version: "{{ gemini_cli_version }}"
    global: "{{ npm_global_install }}"
    registry: "{{ npm_registry }}"
    state: present
  register: gemini_install_result

- name: Verify Gemini CLI installation
  command: gemini --version
  register: gemini_version_check
  changed_when: false

- name: Create Gemini configuration directory
  file:
    path: "{{ gemini_config_dir }}"
    state: directory
    owner: "{{ gemini_cli_user }}"
    group: "{{ gemini_cli_user }}"
    mode: '0700'

- name: Configure Gemini CLI environment
  template:
    src: gemini_env.j2
    dest: "/home/{{ gemini_cli_user }}/.gemini_env"
    owner: "{{ gemini_cli_user }}"
    group: "{{ gemini_cli_user }}"
    mode: '0600'
  no_log: true
  notify: reload shell environment

- name: Create Gemini CLI wrapper script
  template:
    src: gemini_wrapper.sh.j2
    dest: "{{ gemini_cli_install_path }}/gemini-secure"
    mode: '0755'
    owner: root
    group: root
  become: yes

- name: Configure API key authentication
  block:
    - name: Set Gemini API key environment variable
      lineinfile:
        path: "/home/{{ gemini_cli_user }}/.bashrc"
        line: "source ~/.gemini_env"
        state: present
        create: yes
```

```
    - name: Validate API key format
      assert:
        that:
          - gemini_api_key is defined
          - gemini_api_key | length > 10
          - gemini_api_key.startswith('AI')
        fail_msg: "Invalid Gemini API key format"
        success_msg: "API key validation passed"
      no_log: true

    when: gemini_auth_method == "api_key"

  - name: Test Gemini CLI functionality
    shell: |
      source ~/.gemini_env
      echo "Test prompt: What is 2+2?" | gemini --stdin
    become_user: "{{ gemini_cli_user }}"
    register: gemini_test_result
    changed_when: false
    no_log: true

  - name: Display installation summary
    debug:
      msg: |
        Gemini CLI Installation Summary:
        ================================
        Version: {{ gemini_version_check.stdout }}
        Install Path: {{ gemini_cli_install_path }}
        Config Directory: {{ gemini_config_dir }}
        Authentication: {{ gemini_auth_method }}
        Sandbox Enabled: {{ gemini_sandbox_enabled }}
        Test Result: {{ 'PASSED' if gemini_test_result.rc == 0 else 'FAILED' }}
```

Spacelift

**Environment Template:**

bash

```
# roles/gemini_cli/templates/gemini_env.j2
#!/bin/bash
# Gemini CLI Environment Configuration
# Generated by Ansible - Do not edit manually

# Core Gemini CLI settings
export GEMINI_API_KEY="{{ gemini_api_key }}"
{% if gemini_project_id %}
export GOOGLE_CLOUD_PROJECT="{{ gemini_project_id }}"
{% endif %}

# Sandbox configuration
{% for key, value in gemini_env_vars.items() %}
export {{ key }}="{{ value }}"
{% endfor %}

# NPM and Node.js paths
export PATH="/usr/local/bin:$PATH"
export NODE_PATH="{{ npm_global_packages_path }}"

# Security settings
umask 077
```

**Secure Wrapper Script:**

bash

```bash
# roles/gemini_cli/templates/gemini_wrapper.sh.j2
#!/bin/bash
# Secure Gemini CLI Wrapper
# Ensures proper environment and security settings

set -euo pipefail

# Source environment variables
if [[ -f "/home/{{ gemini_cli_user }}/.gemini_env" ]]; then
    source "/home/{{ gemini_cli_user }}/.gemini_env"
fi

# Validate API key is set
if [[ -z "${GEMINI_API_KEY:-}" ]]; then
    echo "Error: GEMINI_API_KEY not set" >&2
    exit 1
fi

# Execute Gemini CLI with secure settings
exec gemini "$@"
```

### Handlers:

yaml

```yaml
# roles/gemini_cli/handlers/main.yml
---
- name: reload shell environment
  shell: source ~/.bashrc
  become_user: "{{ gemini_cli_user }}"

- name: restart gemini service
  systemd:
    name: gemini
    state: restarted
  become: yes
  when: gemini_service_enabled | default(false)
```

## Complete Installation Playbook

### Main Playbook:

yaml

```yaml
# playbooks/gemini_cli_install.yml
---
- name: Install and configure Gemini CLI on WSL2 nodes
  hosts: gemini_nodes
  gather_facts: yes
  become: no
  vars:
    # Override defaults if needed
    gemini_cli_version: "latest"
    gemini_sandbox_enabled: true

  pre_tasks:
    - name: Validate target environment
      block:
        - name: Check if running on WSL2
          shell: grep -qi wsl /proc/version
          register: wsl_check
          changed_when: false
          failed_when: false

        - name: Confirm WSL2 environment
          debug:
            msg: "Running on WSL2: {{ wsl_check.rc == 0 }}"

        - name: Verify internet connectivity
          uri:
            url: "https://api.github.com"
            method: GET
            timeout: 10
          register: connectivity_check

        - name: Validate Ansible Vault variables
          assert:
            that:
              - vault_gemini_api_key is defined
              - vault_gemini_api_key | length > 0
            fail_msg: "Gemini API key must be defined in vault variables"
            success_msg: "Vault variables validated successfully"

  roles:
    - role: gemini_cli
      tags: ['gemini', 'cli', 'install']

  post_tasks:
    - name: Perform post-installation validation
      block:
```

```
    - name: Test Gemini CLI version
      command: gemini --version
      register: final_version_check
      changed_when: false

    - name: Test secure wrapper
      command: gemini-secure --help
      register: wrapper_check
      changed_when: false
      become: yes

    - name: Verify configuration files
      stat:
        path: "{{ item }}"
      loop:
        - "/home/{{ ansible_user }}/.gemini_env"
        - "/home/{{ ansible_user }}/.gemini"
        - "/usr/local/bin/gemini-secure"
      register: config_files

    - name: Display final status
      debug:
        msg: |
          Gemini CLI Installation Complete!
          ==================================
          Version: {{ final_version_check.stdout }}
          Secure Wrapper: {{ 'Available' if wrapper_check.rc == 0 else 'Not Available' }}
          Config Files: {{ config_files.results | selectattr('stat.exists') | list | length }}/{{ config_files.results | length }} present

  handlers:
    - name: Update system package cache
      apt:
        update_cache: yes
      become: yes
```

Spacelift

## Security Validation and Testing Playbook

### Comprehensive Testing:

yaml

```yaml
# playbooks/gemini_cli_test.yml
---
- name: Comprehensive Gemini CLI testing and validation
  hosts: gemini_nodes
  gather_facts: no
  tasks:
    - name: Security validation tests
      block:
        - name: Test file permissions
          stat:
            path: "/home/{{ ansible_user }}/.gemini_env"
          register: env_file_perms

        - name: Validate secure file permissions
          assert:
            that:
              - env_file_perms.stat.mode == '0600'
            fail_msg: "Environment file has insecure permissions"

        - name: Test API key is not exposed in process list
          shell: ps aux | grep -v grep | grep gemini || true
          register: process_check
          changed_when: false

        - name: Verify API key is not visible in processes
          assert:
            that:
              - vault_gemini_api_key not in process_check.stdout
            fail_msg: "API key detected in process list - security risk"

    - name: Functional tests
      block:
        - name: Test basic Gemini CLI functionality
          shell: |
            source ~/.gemini_env
            echo "What is the capital of France?" | timeout 30 gemini --stdin
          register: basic_test
          changed_when: false
          no_log: true

        - name: Test sandbox functionality
          shell: |
            source ~/.gemini_env
            echo "List files in current directory" | timeout 30 gemini --sandbox --stdin
          register: sandbox_test
          changed_when: false
```

```
      no_log: true
      when: gemini_sandbox_enabled

    - name: Validate test results
      assert:
        that:
          - basic_test.rc == 0
          - "'Paris' in basic_test.stdout or 'error' not in basic_test.stdout.lower()"
        fail_msg: "Basic functionality test failed"
        success_msg: "All functionality tests passed"

  - name: Performance and resource tests
    block:
    - name: Check memory usage during operation
      shell: |
        source ~/.gemini_env
        (echo "Simple test" | gemini --stdin &)
        sleep 2
        ps aux | grep gemini | grep -v grep | awk '{print $4}' | head -1
      register: memory_usage
      changed_when: false

    - name: Validate reasonable memory usage
      assert:
        that:
          - memory_usage.stdout | float < 10.0
        fail_msg: "Memory usage too high: {{ memory_usage.stdout }}%"
        success_msg: "Memory usage acceptable: {{ memory_usage.stdout }}%"

  - name: Integration tests
    block:
    - name: Test configuration file loading
      shell: |
        source ~/.gemini_env
        env | grep GEMINI
      register: env_vars
      changed_when: false
      no_log: true

    - name: Verify environment variables
      assert:
        that:
          - "'GEMINI_API_KEY' in env_vars.stdout"
          - "'GEMINI_SANDBOX' in env_vars.stdout"
        fail_msg: "Required environment variables not loaded"

    - name: Test wrapper script execution
```

```
        command: gemini-secure --help
        register: wrapper_test
        changed_when: false
        become: yes

    - name: Generate test report
      debug:
        msg: |
          Gemini CLI Test Report
          ======================
          Basic Functionality: {{ 'PASS' if basic_test.rc == 0 else 'FAIL' }}
          Sandbox Functionality: {{ 'PASS' if sandbox_test.rc == 0 else 'FAIL' }}
          Security Validation: PASS
          Memory Usage: {{ memory_usage.stdout }}%
          Wrapper Script: {{ 'PASS' if wrapper_test.rc == 0 else 'FAIL' }}
          Environment Variables: {{ 'PASS' if 'GEMINI_API_KEY' in env_vars.stdout else 'FAIL' }}
```

Red Hat Hetzner

## Execution and Deployment Commands

### Step-by-Step Deployment:

bash

```
# 1. Test connectivity
ansible-playbook playbooks/test_connectivity.yml

# 2. Install Gemini CLI
ansible-playbook playbooks/gemini_cli_install.yml

# 3. Run comprehensive tests
ansible-playbook playbooks/gemini_cli_test.yml

# 4. Deploy to specific environment
ansible-playbook playbooks/gemini_cli_install.yml --limit staging

# 5. Update configuration only
ansible-playbook playbooks/gemini_cli_install.yml --tags configuration

# 6. Dry run for validation
ansible-playbook playbooks/gemini_cli_install.yml --check --diff
```

Forrestcli

This comprehensive Ansible automation framework provides secure, repeatable, and scalable deployment of Gemini CLI across WSL2 environments. The **modular design enables easy customization** for different

environments while maintaining security best practices through Ansible Vault integration and comprehensive testing protocols. Ansible +2 The solution addresses the modern Node.js-based installation requirements while providing robust error handling and validation mechanisms. Google