Ansible Lab Setup Guide for Windows 11

Critical Architecture Update

Important Discovery: Ansible **cannot** run natively on Windows 11 as a control node. Ansible This is a fundamental technical limitation due to Ansible's reliance on POSIX systems and the fork() system call. Spacelift +4 However, we can create an excellent lab environment using WSL2 for both control and managed nodes.

Revised Architecture

Instead of the original plan, we'll implement:

- **Ansible Control Node**: WSL2 Ubuntu (primary instance)
- **Ansible Managed Node**: WSL2 Ubuntu (secondary instance or same instance for local testing)
- **Target Use Case**: Automate Claude Code setup and development tools
- **Networking**: Native Windows 11 mirrored mode or traditional NAT with port forwarding Microsoft Microsoft

This approach provides the same automation capabilities with better performance and compatibility.

Prerequisites and System Requirements

System Requirements

- Windows 11 (22H2+ recommended for mirrored networking) Microsoft
- 8GB RAM minimum (16GB recommended)
- 50GB available disk space
- Administrator privileges for initial setup
- Stable internet connection

Windows Features Required

```powershell
# Run as Administrator
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

Phase 1: WSL2 Environment Setup

Step 1: Install and Configure WSL2

Install WSL2 with Ubuntu:

```powershell
# Install WSL2 with default Ubuntu distribution
wsl --install

# Verify installation
wsl --version
wsl --list --verbose
```

Configure WSL2 for optimal performance:

Create `%USERPROFILE%\.wslconfig`: Microsoft CodingEasyPeasy

```ini
[wsl2]
# Allocate 6GB RAM for automation workloads
memory=6GB
processors=4
# Enable localhost forwarding for Ansible connectivity
localhostForwarding=true
# Enable mirrored networking (Windows 11 22H2+)
networkingMode=mirrored
# Enable DNS tunneling for VPN compatibility
dnsTunneling=true
autoProxy=true
swap=2GB

[experimental]
# Enable automatic memory reclaim
autoMemoryReclaim=gradual
sparseVhd=true
```

Restart WSL2 to apply changes:

```powershell
wsl --shutdown
wsl
```

Step 2: Ubuntu Initial Configuration

Configure Ubuntu distribution:

bash

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Install essential packages
sudo apt install -y curl wget git vim nano build-essential python3-pip python3-venv

# Create WSL configuration for systemd support
sudo tee /etc/wsl.conf << EOF
[boot]
systemd=true

[automount]
enabled = true
root = /mnt/
options = "metadata,rw,umask=22,fmask=11"
mountFsTab = false

[network]
generateHosts = true
generateResolvConf = true

[interop]
enabled = true
appendWindowsPath = true
EOF

# Restart WSL to enable systemd
exit
```

From PowerShell:

powershell

```
wsl --shutdown
wsl
```

Verify systemd is running:

bash

```
systemctl --version
```

Microsoft

Phase 2: SSH Server Configuration

Step 3: SSH Server Setup for Ansible Communication

Install and configure OpenSSH server:

bash

```
# Install SSH server
sudo apt install -y openssh-server

# Generate SSH host keys
sudo ssh-keygen -A

# Configure SSH daemon
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.backup
```

Create optimized SSH configuration:

bash

```
sudo tee /etc/ssh/sshd_config.d/ansible.conf << EOF
# Ansible-optimized SSH configuration
Port 2222
ListenAddress 0.0.0.0
PermitRootLogin no
PasswordAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
MaxAuthTries 3
MaxSessions 10
ClientAliveInterval 300
ClientAliveCountMax 2
X11Forwarding no
TCPKeepAlive yes
LogLevel INFO
EOF
```

Start and enable SSH service:

```bash
# Start SSH service
sudo systemctl enable ssh
sudo systemctl start ssh
sudo systemctl status ssh

# Verify SSH is listening
sudo ss -tlnp | grep :2222
```

Ask Ubuntu Microsoft

Step 4: User Account Setup

Create dedicated Ansible user:

```bash
# Create ansible user with sudo privileges
sudo useradd -m -s /bin/bash ansible
sudo usermod -aG sudo ansible
sudo passwd ansible

# Configure passwordless sudo for Ansible
sudo tee /etc/sudoers.d/ansible << EOF
ansible ALL=(ALL) NOPASSWD: ALL
EOF

sudo chmod 440 /etc/sudoers.d/ansible
```

Phase 3: Networking Configuration

Step 5: Network Connectivity Setup

For Windows 11 22H2+ (Mirrored Mode - Recommended):

If using mirrored networking mode, the WSL2 instance is directly accessible via `localhost`: Microsoft Microsoft

```bash
# Test connectivity from WSL2
ping localhost
```

For Traditional NAT Mode:

Configure port forwarding from Windows to WSL2: Hanselman Microsoft

powershell

```powershell
# Get WSL2 IP address
$wsl_ip = (wsl hostname -I).trim()

# Add port forwarding rule
netsh interface portproxy add v4tov4 listenport=2222 listenaddress=0.0.0.0 connectport=2222 connectaddress=$wsl_ip

# Verify port forwarding
netsh interface portproxy show all
```

Configure Windows Firewall:

powershell

```powershell
# For mirrored mode
Set-NetFirewallHyperVVMSetting -Name '{40E0AC32-46A5-438A-A0B2-2B479E8F2E90}' -DefaultInboundAction Allow

# For traditional NAT mode
New-NetFirewallRule -DisplayName "WSL SSH" -Direction Inbound -Action Allow -Protocol TCP -LocalPort 2222
```

Phase 4: Ansible Installation and Configuration

Step 6: Install Ansible in WSL2

Install Ansible via pip (recommended):

bash

```bash
# Install Python virtual environment
python3 -m venv ~/ansible-env
source ~/ansible-env/bin/activate

# Install Ansible
pip install --upgrade pip
pip install ansible ansible-lint

# Verify installation
ansible --version
```

Make Ansible environment persistent:

bash

```
# Add to ~/.bashrc
echo 'source ~/ansible-env/bin/activate' >> ~/.bashrc
source ~/.bashrc
```

Step 7: SSH Key Management

Generate SSH keys for Ansible:

bash

```
# Generate ED25519 key pair
ssh-keygen -t ed25519 -C "ansible-lab-automation" -f ~/.ssh/ansible_lab_key

# Set proper permissions
chmod 700 ~/.ssh
chmod 600 ~/.ssh/ansible_lab_key
chmod 644 ~/.ssh/ansible_lab_key.pub
```

Configure SSH key authentication:

bash

```
# Copy public key to ansible user (for local testing)
sudo -u ansible mkdir -p /home/ansible/.ssh
sudo -u ansible chmod 700 /home/ansible/.ssh

# Add public key to authorized_keys
cat ~/.ssh/ansible_lab_key.pub | sudo -u ansible tee /home/ansible/.ssh/authorized_keys
sudo -u ansible chmod 600 /home/ansible/.ssh/authorized_keys
```

Test SSH connectivity:

bash

```
# Test SSH connection
ssh -i ~/.ssh/ansible_lab_key -p 2222 ansible@localhost

# Test with Ansible ping
ansible all -m ping -i inventory --private-key ~/.ssh/ansible_lab_key
```

Phase 5: Ansible Configuration

Step 8: Create Ansible Configuration

Create project directory structure:

bash

```
mkdir -p ~/ansible-lab/{inventory,playbooks,roles,group_vars,host_vars,templates}
cd ~/ansible-lab
```

Ansible

Create ansible.cfg:

ini

```
[defaults]
inventory = ./inventory/hosts
host_key_checking = False
remote_user = ansible
private_key_file = ~/.ssh/ansible_lab_key
log_path = ./logs/ansible.log
forks = 10
gathering = smart
fact_caching = memory
fact_caching_timeout = 86400
retry_files_enabled = False

[inventory]
enable_plugins = host_list, script, auto, yaml, ini

[ssh_connection]
ssh_args = -C -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking=no
control_path_dir = ~/.ansible/cp
pipelining = True
timeout = 30

[colors]
highlight = white
verbose = blue
warn = bright purple
error = red
ok = green
changed = yellow
```

Create inventory file:

bash

```
mkdir -p inventory
cat > inventory/hosts << EOF
[local_lab]
localhost ansible_connection=local ansible_python_interpreter=python3

[wsl_nodes]
wsl-ansible ansible_host=127.0.0.1 ansible_port=2222 ansible_user=ansible

[development:children]
local_lab
wsl_nodes

[all:vars]
ansible_ssh_private_key_file=~/.ssh/ansible_lab_key
ansible_ssh_common_args='-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null'
EOF
```

Ansible Ansible

Step 9: Create Logging Directory

bash

```
mkdir -p logs
```

Phase 6: Testing and Validation

Step 10: Connectivity Testing

Test basic Ansible connectivity:

bash

```
# Test localhost connection
ansible local_lab -m ping

# Test WSL node connection
ansible wsl_nodes -m ping

# Test all nodes
ansible all -m ping

# Gather facts
ansible all -m setup
```

Ansible

Debug connection issues:

bash

```
# Verbose testing
ansible wsl_nodes -m ping -vvv

# Test specific commands
ansible wsl_nodes -a "whoami"
ansible wsl_nodes -a "uname -a"
```

Phase 7: Claude Code Automation Playbook

Step 11: Create Claude Code Setup Playbook

Create playbook for Claude Code automation:

yaml

```yaml
# playbooks/claude-code-setup.yml
---
- name: Claude Code Development Environment Setup
  hosts: wsl_nodes
  become: true
  vars:
    dev_user: ansible
    dev_home: /home/ansible
    node_version: "20"

  tasks:
    - name: Update package cache
      apt:
        update_cache: true
        cache_valid_time: 3600
      tags: [system]

    - name: Install system prerequisites
      package:
        name: "{{ item }}"
        state: present
      loop:
        - curl
        - wget
        - git
        - gnupg2
        - software-properties-common
        - apt-transport-https
        - ca-certificates
        - python3-pip
        - python3-venv
      tags: [system]

    - name: Add NodeSource repository key
      apt_key:
        url: https://deb.nodesource.com/gpgkey/nodesource.gpg.key
        state: present
      tags: [nodejs]

    - name: Add NodeSource repository
      apt_repository:
        repo: "deb https://deb.nodesource.com/node_{{ node_version }}.x {{ ansible_distribution_release }} main"
        state: present
        update_cache: true
      tags: [nodejs]
```

```yaml
    - name: Install Node.js and npm
      package:
        name:
          - nodejs
          - npm
        state: present
      tags: [nodejs]

    - name: Create npm global directory
      file:
        path: "{{ dev_home }}/.npm-global"
        state: directory
        owner: "{{ dev_user }}"
        group: "{{ dev_user }}"
        mode: '0755'
      tags: [npm]

    - name: Configure npm global prefix
      shell: npm config set prefix '{{ dev_home }}/.npm-global'
      become_user: "{{ dev_user }}"
      tags: [npm]

    - name: Add npm global bin to PATH
      lineinfile:
        path: "{{ dev_home }}/.bashrc"
        line: 'export PATH={{ dev_home }}/.npm-global/bin:$PATH'
        create: true
        owner: "{{ dev_user }}"
        group: "{{ dev_user }}"
      tags: [npm]

    - name: Install VS Code
      block:
        - name: Import Microsoft GPG key
          apt_key:
            url: https://packages.microsoft.com/keys/microsoft.asc
            state: present

        - name: Add VS Code repository
          apt_repository:
            repo: "deb [arch=amd64,arm64,armhf] https://packages.microsoft.com/repos/code stable main"
            state: present
            update_cache: true

        - name: Install VS Code
          package:
            name: code
```

```
        state: present
    tags: [vscode]

  - name: Install Claude Code
    shell: |
      export PATH={{ dev_home }}/.npm-global/bin:$PATH
      npm install -g @anthropic-ai/claude-code
    become_user: "{{ dev_user }}"
    environment:
      HOME: "{{ dev_home }}"
      NPM_CONFIG_PREFIX: "{{ dev_home }}/.npm-global"
    register: claude_install
    changed_when: "'added' in claude_install.stdout"
    tags: [claude]

  - name: Install additional development tools
    npm:
      name: "{{ item }}"
      global: true
      path: "{{ dev_home }}"
    become_user: "{{ dev_user }}"
    environment:
      PATH: "{{ dev_home }}/.npm-global/bin:{{ ansible_env.PATH }}"
    loop:
      - typescript
      - eslint
      - prettier
      - nodemon
    tags: [dev-tools]

  - name: Install VS Code extensions
    shell: code --install-extension {{ item }}
    become_user: "{{ dev_user }}"
    loop:
      - redhat.ansible
      - ms-python.python
      - ms-vscode.vscode-json
      - ms-vscode-remote.remote-ssh
      - ms-vscode-remote.remote-wsl
      - streetsidesoftware.code-spell-checker
      - esbenp.prettier-vscode
    register: vscode_extensions
    changed_when: "'is already installed' not in vscode_extensions.stdout"
    failed_when: vscode_extensions.rc != 0 and 'is already installed' not in vscode_extensions.stdout
    tags: [vscode]

  - name: Create development directories
```

```
    file:
      path: "{{ item }}"
      state: directory
      owner: "{{ dev_user }}"
      group: "{{ dev_user }}"
      mode: '0755'
    loop:
      - "{{ dev_home }}/projects"
      - "{{ dev_home }}/.config"
      - "{{ dev_home }}/.config/Code/User"
    tags: [config]

  - name: Verify installations
    block:
      - name: Check Node.js version
        command: node --version
        register: node_version
        changed_when: false

      - name: Check Claude Code installation
        shell: "{{ dev_home }}/.npm-global/bin/claude --version"
        become_user: "{{ dev_user }}"
        register: claude_version
        changed_when: false
        failed_when: false

      - name: Display installation summary
        debug:
          msg:
            - "Development environment setup completed!"
            - "Node.js: {{ node_version.stdout }}"
            - "Claude Code: {{ 'Installed' if claude_version.rc == 0 else 'Installation may have failed' }}"
            - "VS Code: Installed with extensions"
            - "Development directory: {{ dev_home }}/projects"
    tags: [verify]
```

Ansible

Step 12: Execute the Playbook

Run the Claude Code setup playbook:

bash

```
cd ~/ansible-lab

# Test playbook syntax
ansible-playbook playbooks/claude-code-setup.yml --syntax-check

# Run in check mode (dry run)
ansible-playbook playbooks/claude-code-setup.yml --check

# Execute the playbook
ansible-playbook playbooks/claude-code-setup.yml

# Run with specific tags
ansible-playbook playbooks/claude-code-setup.yml --tags "nodejs,claude"
```

Ansible

Phase 8: Advanced Configuration and Troubleshooting

Step 13: Performance Optimization

Optimize WSL2 for automation workloads:

bash

```
# Add performance tuning to sysctl
sudo tee -a /etc/sysctl.d/99-ansible-performance.conf << EOF
# Network performance tuning
net.core.somaxconn=65536
net.ipv4.tcp_max_syn_backlog=65536
net.ipv4.tcp_fin_timeout=30
net.ipv4.tcp_keepalive_time=120

# Memory management
vm.swappiness=10
vm.vfs_cache_pressure=50
EOF

sudo sysctl -p /etc/sysctl.d/99-ansible-performance.conf
```

Ceos3c Toxigon

Step 14: Security Hardening

Implement security best practices:

bash

```
# Install and configure fail2ban
sudo apt install -y fail2ban

# Configure fail2ban for SSH
sudo tee /etc/fail2ban/jail.local << EOF
[sshd]
enabled = true
port = 2222
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600
findtime = 600
EOF

sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

Server Fault +2

Configure firewall rules:

bash

```
# Install and configure UFW
sudo apt install -y ufw

# Allow SSH on custom port
sudo ufw allow 2222/tcp

# Enable firewall
sudo ufw --force enable
```

Step 15: Troubleshooting Common Issues

SSH Connection Issues:

bash

```
# Check SSH service status
sudo systemctl status ssh

# Check SSH configuration
sudo sshd -t

# Monitor SSH logs
sudo tail -f /var/log/auth.log

# Test SSH connectivity manually
ssh -i ~/.ssh/ansible_lab_key -p 2222 ansible@localhost -v
```

Ansible Pilot +2

Ansible Connection Issues:

bash

```
# Test with verbose output
ansible wsl_nodes -m ping -vvv

# Check inventory parsing
ansible-inventory --list

# Validate configuration
ansible-config dump --only-changed
```

Performance Issues:

bash

```
# Monitor resource usage
htop

# Check WSL2 memory usage
free -h

# Monitor network connections
ss -tulpn | grep :2222
```

Phase 9: Additional Automation Examples

Step 16: Create Additional Playbooks

System maintenance playbook:

yaml

```yaml
# playbooks/system-maintenance.yml
---
- name: System Maintenance Tasks
  hosts: wsl_nodes
  become: true
  tasks:
    - name: Update package cache and upgrade packages
      apt:
        update_cache: true
        upgrade: dist
        autoremove: true
        autoclean: true
      tags: [update]

    - name: Clean up temporary files
      file:
        path: "{{ item }}"
        state: absent
      loop:
        - /tmp/*
        - /var/tmp/*
      tags: [cleanup]

    - name: Check disk usage
      shell: df -h
      register: disk_usage
      changed_when: false

    - name: Display disk usage
      debug:
        var: disk_usage.stdout_lines
      tags: [info]
```

Roelof Jan Elsinga

Development project setup playbook:

yaml

```yaml
# playbooks/project-setup.yml
---
- name: Setup Development Project
  hosts: wsl_nodes
  become_user: ansible
  vars:
    project_name: "{{ project_name | default('my-project') }}"
    project_path: "/home/ansible/projects/{{ project_name }}"

  tasks:
    - name: Create project directory
      file:
        path: "{{ project_path }}"
        state: directory
        mode: '0755'

    - name: Initialize Git repository
      git:
        repo: "{{ git_repo | default('') }}"
        dest: "{{ project_path }}"
      when: git_repo is defined

    - name: Create package.json if Node.js project
      template:
        src: templates/package.json.j2
        dest: "{{ project_path }}/package.json"
      when: project_type == "nodejs"

    - name: Setup Python virtual environment
      python:
        virtualenv: "{{ project_path }}/venv"
        virtualenv_command: python3 -m venv
      when: project_type == "python"
```

Phase 10: Monitoring and Maintenance

Step 17: Setup Monitoring

Create monitoring script:

bash

```
# Create monitoring script
cat > ~/ansible-lab/scripts/monitor-lab.sh << 'EOF'
#!/bin/bash
# Ansible Lab Monitoring Script

echo "=== Ansible Lab Status ==="
echo "Date: $(date)"
echo

echo "=== WSL2 Status ==="
wsl --status
echo

echo "=== SSH Service Status ==="
systemctl status ssh --no-pager -l
echo

echo "=== Ansible Connectivity Test ==="
cd ~/ansible-lab
ansible all -m ping | grep -E "(SUCCESS|FAILED|UNREACHABLE)"
echo

echo "=== System Resources ==="
free -h
df -h / | tail -1
echo

echo "=== Network Connectivity ==="
ss -tlnp | grep :2222
echo

echo "=== Recent Ansible Logs ==="
if [ -f logs/ansible.log ]; then
    tail -5 logs/ansible.log
else
    echo "No ansible.log found"
fi
EOF

chmod +x ~/ansible-lab/scripts/monitor-lab.sh
```

Step 18: Backup and Recovery

Create backup script:

```bash
# Create backup script
cat > ~/ansible-lab/scripts/backup-lab.sh << 'EOF'
#!/bin/bash
# Ansible Lab Backup Script

BACKUP_DIR="$HOME/ansible-lab-backups"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_NAME="ansible-lab-backup-$DATE"

mkdir -p "$BACKUP_DIR"

echo "Creating backup: $BACKUP_NAME"

# Backup Ansible configuration
tar -czf "$BACKUP_DIR/$BACKUP_NAME.tar.gz" \
    -C "$HOME" \
    ansible-lab \
    .ssh/ansible_lab_key* \
    .ansible/ \
    --exclude="ansible-lab/logs/*" \
    --exclude="*.pyc" \
    --exclude="__pycache__"

echo "Backup created: $BACKUP_DIR/$BACKUP_NAME.tar.gz"

# Keep only last 5 backups
cd "$BACKUP_DIR"
ls -t ansible-lab-backup-*.tar.gz | tail -n +6 | xargs -r rm

echo "Cleanup completed. Available backups:"
ls -la ansible-lab-backup-*.tar.gz
EOF

chmod +x ~/ansible-lab/scripts/backup-lab.sh
```

Summary and Best Practices

Key Takeaways

1. **Architecture Correction**: Ansible must run in WSL2, not natively on Windows 11 (Spacelift +6)
2. **Dual WSL2 Setup**: Use WSL2 for both control and managed nodes
3. **Networking**: Mirrored mode (Windows 11 22H2+) provides optimal connectivity (Microsoft) (Microsoft)
4. **Security**: Implement proper SSH key management and firewall rules

5. **Performance**: Optimize WSL2 configuration for automation workloads Ceos3c +2

Best Practices Implemented

- **Idempotent playbooks** with proper changed detection Ansible +2
- **Comprehensive error handling** and validation Ansible +2
- **Security hardening** with dedicated users and key management CloudThat Resources Jhooq
- **Performance optimization** for automation workloads
- **Monitoring and maintenance** procedures
- **Backup and recovery** strategies

Next Steps

1. **Expand inventory** to include additional WSL2 instances or remote nodes Ansible
2. **Implement Ansible Vault** for sensitive credential management Jhooq
3. **Create custom roles** for common automation tasks
4. **Set up CI/CD pipelines** for playbook testing
5. **Explore Ansible Galaxy** for community roles and collections Ansible +2

This comprehensive setup provides a robust foundation for Ansible automation on Windows 11, enabling efficient development environment management and process automation while maintaining security and performance best practices.