



# Lab 5 — Test Connectivity Using Ad-Hoc Commands

**Author:** Sandeep Kumar Sharma

---



## Learning Objectives

In this lab, you will learn: - How to test connectivity between the Ansible master and managed nodes - How to use the most important ad-hoc modules for connectivity validation - How to run basic ping, shell/command, and file/copy operations - How connectivity testing differs on Ubuntu and Amazon Linux

---



## Learning Outcomes

After completing this lab, you will: - Confirm that your Ansible master can reach all target nodes - Execute basic ad-hoc commands using core Ansible modules - Validate SSH, Python interpreter, and network readiness - Be ready to run more advanced commands and playbooks

---



## Why Do We Test Connectivity?

Before running any serious automation, you must ensure: - SSH connection is working - Python interpreter is available on nodes - Inventory is correctly configured - Ansible can execute remote commands

This lab confirms that your cluster is ready for automation.

---



## SECTION A — Ping Module

The `ping` module verifies basic communication.

**Run:**

```
ansible dev -m ping
```

Expected output:

```
SUCCESS => {"changed": false, "ping": "pong"}
```

If you get this output, the node is reachable and Ansible is working.

---



## SECTION B — Command & Shell Module

These modules allow you to execute operating system commands.

### 1. Check Hostname

```
ansible dev -m command -a "hostname"
```

### 2. Check Uptime

```
ansible dev -m command -a "uptime"
```

### 3. Run Shell Command (example using echo)

```
ansible dev -m shell -a "echo Connectivity OK > /tmp/connectivity.txt"
```

### 4. Display File Content

```
ansible dev -m shell -a "cat /tmp/connectivity.txt"
```



## SECTION C — File Module

The file module is used to create, modify, or delete files and directories.

### 1. Create a Directory

```
ansible dev -m file -a "path=/tmp/connect_test state=directory mode=0755"
```

## 2. Create an Empty File

```
ansible dev -m file -a "path=/tmp/connect_test/info.txt state=touch"
```

## 3. Delete a File

```
ansible dev -m file -a "path=/tmp/connect_test/info.txt state=absent"
```



## SECTION D — Copy Module

The copy module transfers files from the master to remote nodes.

### 1. Create a File on Master

```
echo "This is a connectivity test file" > /tmp/connectivity_local.txt
```

### 2. Copy File to All Nodes

```
ansible dev -m copy -a "src=/tmp/connectivity_local.txt dest=/tmp/connectivity_remote.txt"
```

### 3. Validate Copy

```
ansible dev -m shell -a "cat /tmp/connectivity_remote.txt"
```



## Ubuntu vs Amazon Linux Differences

For this lab, both OS types behave the same because: - ping module works identically - command/shell modules behave the same - file/copy modules are universal

No OS-specific changes are required.



## Hands-On Checklist

Run all of the following to complete the lab:

- [] ansible dev -m ping
  - [] ansible dev -m command -a "hostname"
  - [] ansible dev -m command -a "uptime"
  - [] ansible dev -m shell -a "echo OK > /tmp/msg.txt"
  - [] ansible dev -m file -a "path=/tmp/check state=directory"
  - [] ansible dev -m file -a "path=/tmp/check/test.txt state=touch"
  - [] ansible dev -m copy -a "src=/tmp/local dest=/tmp/remote"
- 



## Lab Summary

In this lab, you learned how to: - Test connectivity using ping - Execute OS commands remotely - Use file and copy modules - Validate master-node communication

Your environment is now fully validated and ready for deeper automation.

Next Lab:  [Lab 6 — Using Variables in Playbooks](#)