Ansible Troubleshooting Runbook Pack

A senior engineer's incident runbook for playbooks that fail on connection, escalation, templating, and everything in between. DevOps AI ToolKit · devopsaitoolkit.com

An Ansible run fails for one of a small number of reasons: it can't reach the host (SSH/UNREACHABLE), it can't become root (privilege escalation), it can't parse your files (YAML/Jinja2), it can't find your hosts (inventory), or the module can't run on the far side (Python interpreter). The error text usually names the layer — the trick is reading it top-to-bottom and confirming with a read-only check before you change anything. Work the failure the way the play executes: parse, connect, escalate, template, run.

> Examples assume a control node running a recent `ansible-core` (2.12+) against Linux hosts named `web-1`, in a group `webservers`. Adjust for your inventory. Run `--syntax-check`, `--check`, and `--diff` before any real change; only touch production with a change record.

1. First-Response Triage

```
# 1. Does the playbook even parse? (offline, no hosts touched)
ansible-playbook site.yml --syntax-check

# 2. Static best-practice / structural problems
ansible-lint site.yml

# 3. Can the control node reach the hosts at all?
ansible webservers -i inventory.ini -m ping

# 4. Reproduce the real failure with full connection + module detail
ansible-playbook site.yml -i inventory.ini -l web-1 --check -vvv
```

- `--syntax-check` catches YAML and play-structure errors without connecting — always run it first.
- `-m ping` is the Ansible `ping` module (Python round-trip), **not** ICMP; success proves SSH *and* the remote interpreter work.
- Escalate verbosity deliberately: `-v` (task results) → `-vvv` (SSH command + args) → `-vvvv` (connection plugin internals). `-vvv` shows the exact `ssh` line Ansible ran.

2. SSH / UNREACHABLE

```
# The message: "Failed to connect to the host via ssh: ... UNREACHABLE"
ansible web-1 -i inventory.ini -m ping -vvv 2>&1 | grep -E "ESTABLISH|ssh:|Permission|timed out"

# Reproduce the exact SSH Ansible attempts (copy the command from -vvv)
ssh -o StrictHostKeyChecking=accept-new deploy@web-1 'echo ok'
```

```
# group_vars/webservers.yml — pin connection identity explicitly
ansible_user: deploy
ansible_ssh_private_key_file: ~/.ssh/deploy_ed25519
ansible_host: 10.0.3.11            # when the inventory name isn't resolvable
ansible_ssh_common_args: '-o ProxyJump=bastion.example.com'
```

- "Host key verification failed" → the key changed or is unknown. Fix the `known_hosts` entry properly; only disable `host_key_checking = False` in `ansible.cfg` for throwaway/ephemeral hosts.
- "Permission denied (publickey)" → wrong `ansible_user` or key. Confirm with the raw `ssh` line from `-vvv`.
- "Connection timed out" → routing/firewall/security group, not Ansible. Reach through a bastion with `ProxyJump` rather than copying keys onto it.

3. Privilege Escalation (become)

```
# The message: "Missing sudo password" or "Incorrect sudo password"
# Prompt for the become password at runtime instead of storing it
ansible-playbook site.yml -i inventory.ini --become --ask-become-pass -l web-1 --check

# Confirm escalation works in isolation
ansible web-1 -i inventory.ini -b -m command -a 'id' -vvv
```

```
# Per-play or per-task escalation — least privilege
- name: Restart the app service
  become: true
  become_user: root            # default; set app-user for non-root escalation
  become_method: sudo          # sudo | su | pbrun | doas ...
  ansible.builtin.systemd:
    name: web
    state: restarted
```

- "Missing sudo password" means passwordless sudo isn't configured for `ansible_user` — either fix the sudoers rule or pass `--ask-become-pass`.
- Never bake the become password into a plain var; use `--ask-become-pass` or an Ansible Vault–encrypted `ansible_become_password`.
- Escalating to a service account? Set `become_user:` — `become: true` alone targets root.

---

4. Undefined Variables & Jinja2

```
# The message: "AnsibleUndefinedVariable: 'foo' is undefined"
#           or "'dict object' has no attribute 'bar'"
# Dump what Ansible actually sees for a host
ansible web-1 -i inventory.ini -m debug -a "var=hostvars[inventory_hostname]"
ansible web-1 -i inventory.ini -m debug -a "var=app_config"
```

```
# Guard optional values with the default filter instead of assuming they exist
- name: Render config
  ansible.builtin.template:
    src: app.conf.j2
    dest: /etc/app/app.conf
  vars:
    listen_port: "{{ app_port | default(8080) }}"
    log_level:   "{{ app.get('log_level', 'info') }}"   # safe dict access
```

- `'dict object' has no attribute 'x'` → you used `mydict.x` but `x` isn't set. Use `mydict.x | default('...')` or `mydict['x']`.
- Wrap every `{{ }}` that starts a value in quotes — a bare `{{ ... }}` at the start of a YAML value is a parse error (see §5).
- `| default(omit)` is the correct way to leave a module parameter unset rather than passing an empty string.

---

5. YAML & Syntax Errors

```
# The message: "Syntax Error while loading YAML ... could not find expected ':'"
ansible-playbook site.yml --syntax-check
yamllint site.yml            # line/column of the offending token
```

```
# Common breakages and their fixes
- name: Value that starts with a Jinja brace MUST be quoted
  ansible.builtin.debug:
    msg: "{{ app_version }}"          # not: msg: {{ app_version }}

- name: A value containing a colon MUST be quoted
  ansible.builtin.debug:
    msg: "Deploying app: web-1"       # not: msg: Deploying app: web-1
```

- The reported line is where the parser *gave up*, often one or two lines **after** the real mistake (a missing quote or bad indent above it).
- Indentation is spaces only — never tabs. `yamllint` flags mixed indentation instantly.
- "mapping values are not allowed here" almost always means an unquoted colon inside a value.

---

6. Inventory Problems

```
# The message: "Could not match supplied host pattern, ignoring: webservers"
# See exactly what Ansible parsed and which groups exist
ansible-inventory -i inventory.ini --list
ansible-inventory -i inventory.ini --graph
ansible webservers -i inventory.ini --list-hosts
```

```
# inventory/aws_ec2.yml — a dynamic inventory plugin (not a shebang script)
plugin: amazon.aws.aws_ec2
regions:
  - us-east-1
keyed_groups:
  - key: tags.Role
    prefix: role
```

- "Could not match supplied host pattern" → the group/host doesn't exist in the parsed inventory. Confirm the name with `--graph`; check you passed the right `-i`.
- An empty inventory usually means a wrong path or a dynamic plugin that returned nothing — run `ansible-inventory --list` to see the raw output.
- Dynamic inventory failing silently? Add `ANSIBLE_DEBUG=1` or run the plugin directly; verify the plugin is enabled in `ansible.cfg` `[inventory] enable_plugins`.

---

7. Module & Interpreter Errors

```
# The message: "Failed to import the required Python library (e.g. docker) on web-1"
# What interpreter did Ansible auto-discover, and what does it have?
ansible web-1 -i inventory.ini -m setup -a 'filter=ansible_python*'
ansible web-1 -i inventory.ini -m command -a '/usr/bin/python3 -c "import docker"'
```

```
# host_vars/web-1.yml — pin the interpreter when discovery guesses wrong
ansible_python_interpreter: /usr/bin/python3
```

- "Failed to import the required Python library" → the module's dependency isn't installed **on the target**, under the interpreter Ansible chose. Install it there (`pip`/package), or point `ansible_python_interpreter` at the venv that has it.
- Interpreter-discovery warnings mean Ansible guessed the path — pin `ansible_python_interpreter` per host/group to silence and stabilize it.
- "No module named ..." on the *control* node instead means a missing collection: install it with `ansible-galaxy collection install <namespace.name>`.

---

8. Ansible Vault

```
# The message: "Decryption failed" or "ERROR! Attempting to decrypt but no vault secrets found"
# Point at the password source (never type secrets on the CLI)
export ANSIBLE_VAULT_PASSWORD_FILE=~/.vault/prod.pass
ansible-playbook site.yml -i inventory.ini --vault-id prod@"$ANSIBLE_VAULT_PASSWORD_FILE" --check

# Inspect / rotate an encrypted file safely
ansible-vault view   group_vars/prod/secrets.yml
ansible-vault rekey  group_vars/prod/secrets.yml
```

- "no vault secrets found" → you didn't pass a password source. Supply `--vault-id`, `--vault-password-file`, or `ANSIBLE_VAULT_PASSWORD_FILE`.
- "Decryption failed" with a password present → wrong password for that file, or multiple vault-ids and the labels don't match. Use `--vault-id label@source` so each file matches its own secret.
- Encrypt a single value inline with `ansible-vault encrypt_string` rather than encrypting the whole file when only one secret needs protecting.

---

9. Idempotency & changed_when

```
# Symptom: a task reports "changed" on every run when nothing changed
# Preview what WOULD change, and show the diff, without applying
ansible-playbook site.yml -i inventory.ini --check --diff -l web-1
```

```
# command/shell are always "changed" — tell Ansible how to judge them
- name: Regenerate config only when inputs changed
  ansible.builtin.command: /usr/local/bin/gen-config
  register: gen
  changed_when: "'wrote' in gen.stdout"
  check_mode: false            # let this read-only probe run even under --check

- name: Idempotent by design — prefer a module over shell
  ansible.builtin.copy:
    src: app.conf
    dest: /etc/app/app.conf
```

- `command`/`shell` have no idempotency of their own — always add `changed_when:` (and often `creates:`/`removes:`) or switch to a purpose-built module.
- `--check --diff` is your safe rehearsal: `--check` predicts, `--diff` shows the exact before/after text. Some modules skip real work under `--check`; set `check_mode: false` on read-only probes.
- A task that's "always changed" often hides a real drift bug — investigate before suppressing it with `changed_when: false`.

---

10. Connection Plugins & Performance

```
# ansible.cfg — the highest-leverage performance knobs
[defaults]
forks            = 25              # parallel hosts per task (default 5)
gathering        = smart           # skip re-gathering already-known facts
fact_caching     = jsonfile
fact_caching_connection = /tmp/ansible_facts
fact_caching_timeout    = 7200

[ssh_connection]
pipelining       = True            # fewer SSH ops per task (needs no requiretty in sudoers)
ssh_args         = -o ControlMaster=auto -o ControlPersist=60s
```

```
# Skip fact gathering where you don't need facts; run a task from one host
- hosts: webservers
  gather_facts: false            # cut a full SSH round-trip per host
  tasks:
    - name: Talk to the load balancer from the control node
      ansible.builtin.uri:
        url: https://lb.internal/health
      delegate_to: localhost
      retries: 3
      delay: 5
      register: r
      until: r.status == 200
```

- Slow runs are usually fact gathering + low `forks`. Enable `fact_caching` and raise `forks` before blaming the network.
- `pipelining = True` is the single biggest per-task speedup; it needs `requiretty` disabled in the target's sudoers.
- Use `delegate_to: localhost` for API/LB calls and `retries`/`until` for eventually-consistent checks instead of `sleep`.

---

11. Decision Tree

```
Playbook run failed — where?
 ├─ Won't parse (before any host)? ── yes → --syntax-check / yamllint (§1, §5)
 └─ Parses, then fails
     ├─ "UNREACHABLE / Failed to connect via ssh"? → §2  (ansible_user, key, ProxyJump, host key)
     ├─ "Missing sudo password" / become error?     → §3  (--ask-become-pass, become_user/method)
     ├─ "AnsibleUndefinedVariable" / dict attribute? → §4  (default filter, debug var)
     ├─ "Could not match supplied host pattern"?    → §6  (ansible-inventory --graph, -i path)
     ├─ "Failed to import required Python library"? → §7  (ansible_python_interpreter, deps on target)
     ├─ "Decryption failed / no vault secrets"?     → §8  (--vault-id, VAULT_PASSWORD_FILE)
     ├─ Task "changed" every run?                   → §9  (changed_when, --check --diff)
     └─ Correct but painfully slow?                 → §10 (forks, gather_facts, pipelining, caching)
```

---

12. Copy/Paste One-Shot Triage

```
PB=site.yml; INV=inventory.ini; HOST=web-1
echo "== syntax ==";      ansible-playbook "$PB" --syntax-check
echo "== lint ==";        ansible-lint "$PB" 2>/dev/null | head -20
echo "== reachability =="; ansible "$HOST" -i "$INV" -m ping
echo "== identity ==";    ansible "$HOST" -i "$INV" -m debug -a "var=ansible_user" 2>/dev/null
echo "== interpreter =="; ansible "$HOST" -i "$INV" -m setup -a 'filter=ansible_python_interpreter' 2>/dev/null | grep interpreter
echo "== inventory ==";   ansible-inventory -i "$INV" --graph
echo "== become ==";      ansible "$HOST" -i "$INV" -b -m command -a 'id' 2>&1 | tail -2
echo "== dry run ==";     ansible-playbook "$PB" -i "$INV" -l "$HOST" --check --diff -vvv 2>&1 | tail -30
```

---

13. Incident Notes Template

```
INCIDENT: Ansible playbook failure
Started (UTC):          ____    Detected by: ____   Sev: ____
Playbook / role:        ____    Target host/group: ____
Layer confirmed:        [ ] parse/YAML  [ ] SSH/UNREACHABLE  [ ] become  [ ] vars/Jinja2
                        [ ] inventory   [ ] interpreter/module  [ ] vault  [ ] idempotency  [ ] perf
Exact error line:       ____
Reproduced with:        ansible-playbook ... --check -vvv  [ ] yes  [ ] no
Key evidence:           (raw ssh line / undefined var / consumers / interpreter path) ____
Action taken:           (var pinned / key fixed / dep installed / config changed + who + when)
Result:                 [ ] recovered @ ____    [ ] escalated to ____
Root cause:             ____
Follow-ups:             (fact caching, sudoers, vault-id, lint in CI, interpreter pin) ____
```
